2010-03-10

A Bayesian Decision Theoretical Approach to Supervised Learning, Selective Sampling, and Empirical Function Optimization

James Lamond Carroll
Brigham Young University - Provo

A Bayesian Decision Theoretical Approach to Supervised Learning, Selective Sampling, and Empirical Function Optimization

James L. Carroll

A dissertation submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

Brigham Young University

April 2010

ABSTRACT

A Bayesian Decision Theoretical Approach to Supervised Learning,

Selective Sampling, and Empirical Function Optimization

James L. Carroll

Department of Computer Science

Doctor of Philosophy

Many have used the principles of statistics and Bayesian decision theory to model specific learning problems. It is less common to see models of the processes of learning in general. One exception is the model of the supervised learning process known as the "Extended Bayesian Formalism" or EBF. This model is descriptive, in that it can describe and compare learning algorithms. Thus the EBF is capable of modeling both effective and ineffective learning algorithms.

We extend the EBF to model un-supervised learning, semi-supervised learning, supervised learning, and empirical function optimization. We also generalize the utility model of the EBF to deal with non-deterministic outcomes, and with utility functions other than 0-1 loss. Finally, we modify the EBF to create a "prescriptive" learning model, meaning that, instead of describing existing algorithms, our model defines how learning should optimally take place. We call the resulting model the Unified Bayesian Decision Theoretical Model, or the UBDTM. WE show that this model can serve as a cohesive theory and framework in which a broad range of questions can be analyzed and studied. Such a broadly applicable unified theoretical framework is one of the major missing ingredients of machine learning theory.

Using the UBDTM, we concentrate on supervised learning and empirical function optimization. We then use the UBDTM to reanalyze many important theoretical issues in Machine Learning, including No-Free-Lunch, utility implications, and active learning. We also point forward to future directions for using the UBDTM to model learnability, sample complexity, and ensembles. We also provide practical applications of the UBDTM by using the model to train a Bayesian variation to the CMAC supervised learner in closed form, to perform a practical empirical function optimization task, and as part of the guiding principles behind an ongoing project to create an electronic and print corpus of tagged ancient Syriac texts using active learning.

Keywords: Machine Learning, Supervised Learning, Function Optimization, Empirical Function Optimization, Statistics, Bayes, Bayes Law, Bayesian, Bayesian Learning, Decision Theory, Utility Theory, Unified Bayesian Model, UBM, Unified Bayesian Decision Theoretical

Model, UBDTM, Learning Framework, No-Free-Lunch, NFL, *a priori* Learning, Extended Bayesian Formalism, EBF, Bias, Inductive Bias, Hypothesis Space, Function Class, Active Learning, Uncertainty Sampling, Query by Uncertainty, Query by Committee, Expected Value of Sample Information, EVSI.

ACKNOWLEDGMENTS

First and foremost, I must thank my wonderful wife, Heidi. She worked for many years to help put me through school, and she certainly deserves the majority of the credit for this accomplishment. She spent many long hours without me while I sat in front of my computer, trying to write.

When I first began to struggle in elementary school, it was my mother who spent hours and hours helping me learn to read and going over my spelling lists with me while I quite literally stood on my head. She passed away just last year, and it is unfortunate that she did not get to see me complete the work she started. Much of my academic success I owe to my wonderful mother. I love you, and you will be missed.

My adviser, Kevin Seppi, put in many long hours assisting me, making suggestions, and reviewing the writing. I also especially need to thank my co-authors, Kevin Seppi, Chris Monson, Neil Toronto, and Robbie Haertel, who each contributed greatly to the publications that this final dissertation is based upon.

I would also like to thank David Wingate, Andrew McNabb and the rest of the AML lab for their numerous suggestions. NFD and the fallacy of classifying decisions grew from a conversation with David Wingate about making decisions in the UBDTM, and Andrew contributed greatly to my understanding of statistical theory, and had many great suggestions to make my proofs more readable, understandable, and on occasion, more correct. He also deserves some credit for putting up with my philosophy on "compromise." At one point or another, everyone in the lab took a turn reading and commenting on a chapter, and they all deserve thanks for their efforts.

I would especially like to thank Jimmy Lee Carroll and Mary E. Carroll for putting up with all 300 plus pages, for enduring my egregious spelling, and for improving my erratic comma usage.

BRIGHAM YOUNG UNIVERSITY

SIGNATURE PAGE

of a dissertation submitted by

James L. Carroll

The dissertation of James L. Carroll is acceptable in its final form including (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory and ready for submission.

Date	Kevin Seppi, Chair
Date	Eric K. Ringger
Date	Dan Ventura
Date	David W. Embley
Date	Charles Knutson

Date	Kent E. Seamons, Graduate Coordinator
Date	Thomas W. Sederberg, Associate Dean, College of Physical and Mathematical Sciences

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Today's computers are capable of performing many tasks that previously only people could perform. This is partly due to recent techniques that have allowed computer systems to "learn" from experience. Machine Learning (ML) is the field of computer science that attempts to understand what it means to learn, and how such learning can be performed.

Many types of "learning" fall under the umbrella of Machine Learning. For example, un-supervised learning is the process of learning from unlabeled examples while supervised learning is the process of learning functions from labeled examples. Semi-supervised learning is the process of learning from a mix of labeled and unlabeled examples. When the function to be learned is discrete, supervised learning is often called classification. When the function is continuous, supervised learning is often called function approximation or regression. As we shall see, empirical function optimization (EFO) can also be seen as part of machine learning. Function optimization is related to search and is the process of discovering a function's "extremum" (or multiple extrema). Usually the extremum is the minimum or maximum of the function, but function optimization is related to search in that it is possible to *search* for any important point in the function that a user might want to discover. Function optimization can take many forms. Empirical function optimization is the case where the function is repeatedly sampled in an effort to discover the optimum. Since the function must be repeatedly sampled, and the location of the extremum learned from these samples, EFO is a type of machine learning and is related to supervised learning since both must discover information about an unknown function from examples.

These four domains of machine learning: un-supervised learning, semi-supervised learning, supervised learning, and empirical function optimization all share many common characteristics. We will call these four domains of machine learning "function learning," and the purpose of this dissertation will be to create a common underlying unified theory and framework for dealing with these four domains of machine learning.

1.1 The Need for a Unified Theoretical Framework for Machine Learning

Given the importance of function learning to machine learning, it should not be surprising that a large body of theory has been generated to explain the fundamental principles behind them. Examples include: PAC learnability framework [114], Vapnik's "uniform coverage" (or VC) framework [115], the Extended Bayesian Formalism (EBF), and the associated No-Free-Lunch (NFL) theorems [122]. Even models based upon statistical physics have been applied to the task of understanding the fundamental principles of supervised learning [119].

There are many types of questions that a good theory or framework for machine learning should be able to answer [120, 13]. Some of these questions are:

1. What is the formal nature of the relationship between empirical function optimization, un-supervised, semi-supervised, and supervised Learning?
2. What problems can be learned from data (learnability)?
3. How much data will be required to learn a given problem (sample complexity)?
4. Which algorithms perform better over the space of all possible functions that we might want to learn (No-Free-Lunch)?
5. Is an inductive bias necessary for learning (the futility of bias free learning)?
6. Does the NFL result lead to the futility of bias free learning (NFL and Bias)?
7. How can a learning algorithm's inductive bias be quantified, expressed, compared, or made explicit (the theory of bias)?

8. What is the relationship between the concepts of inductive bias, hypothesis space, and function class (bias and sample complexity)
9. What is the underlying cause of overfit, and how does this relate to an algorithm's inductive bias (the theory of bias)?
10. Why do ensembles work, what are their theoretical foundations and motivations (ensembles and the theory of inductive bias)?
11. How should utility and decision theory impact the theory and use of machine learning?
12. How can the learner determine the information which would be most useful for its learning and actively request the most useful information (active learning and selective sampling)?
13. If an algorithm performs well on one function or function class, how can we use that information to predict its performance on another function or function class (meta learning)?
14. How can information learned from one problem or function be leveraged to improve the learning of another problem or function (meta learning, transfer learning, and learning to learn)?

To date, none of the proposed frameworks have been able to answer all of these questions in a unified manner. For example, the EBF can be used to prove the NFL theorems, but has less to say about computational complexity and learnability. It is possible to use VC dimensionality or PAC learnability to determine if a function is "learnable" by placing theoretical upper bounds on the number of examples required to learn a given problem, but these frameworks have less to say about NFL, or active sampling. Furthermore, these bounds often vastly over estimate the number of samples required [13] [40]. Clearly an improved approach is required.

We believe that a unified theory for these four domains of machine learning is needed that can answer the above questions in a unified way, within a common notation and framework. Therefore, our goal is to create such a common framework in which all of the above questions can be answered. For example, such a framework would help to explain why No-Free-Lunch theorems exist for both empirical function optimization, search, and supervised learning, and would illuminate through the common framework a common cause for these results in each domain (see Chapter 3 and Chapter 7). It would facilitate discussion of NFL, active learning, meta learning, transfer learning, learnability, and sample complexity for all four of the domains of function learning in a single unified framework.

1.2 The Proposed Solution

Perhaps the best definition of machine learning was given by Mitchell. He defined machine learning as:

Definition 1.2.0.1. Machine Learning is improving performance in the presence of data as measured by some performance measure [74]

This definition requires both a technique for improving performance given data, and for measuring performance. We propose that Bayes law and decision theory are fundamental parts to the mathematics of learning under this definition. This is because Bayes law dictates how one should learn from data, while decision theory dictates how to use what we have learned to improve performance while providing a formal way to express this performance measure. In Section 1.2.1 we will discuss Bayes law as the fundamental mathematical principle of learning, then in Section 1.2.2 we will discuss how decision theory relates to the mathematics of learning, and then in Section 1.2.3 we will explain how we will combine these two foundations into a single framework for understanding the theory of function learning.

1.2.1 Bayes Law as the Foundational Theorem of Learning

Bayes law [6] is a fundamental theorem that can be trivially derived [95, p. 479] from the axioms of probability [51]. This law can be seen as the fundamental, universal law of learning.

It dictates the mathematically correct way that belief B should be updated in the presence of data D:

$$P(B|D) = \frac{P(D|B)P(B)}{P(D)}.$$

$P(B|D)$ is known as the posterior, $P(D|B)$ is sometimes called the likelihood, while $P(B)$ is known as the prior. The prior is what was believed before seeing data, while the posterior $P(B|D)$ is what should be believed after seeing data. Thus, if learning involves updating probabilistic belief in the presence of data in any way (which we might assume given Mitchell's definition) then Bayes Law is the only mathematically correct way to learn. There is an equals sign in this equation, which means that any other approach to updating statistical belief in the presence of data must either fail, follow Bayes Law, or approximate it in some way. In essence, Bayes law is "right," and every other learning technique is either "wrong" or is an approximation to Bayes law [55].

One common complaint against the Bayesian interpretation of learning is that it is not objective since it is conditioned on a prior. However, since Bayes Law is a fundamental mathematical principle, a more reasonable view would be to say that this Law shows that fully objective inference is impossible, and since all techniques that work do so because they are either equivalent to Bayes law for some prior, or they approximate Bayes law for some prior. Many algorithms have an implicit bias or prior, and the Bayesian approach simply makes their bias explicit. This idea will become more significant when we discuss No-Free-Lunch and the futility of bias free learning (see Section 5.2).

There are some approaches to learning that attempt to avoid the need for a bias by avoiding the prior. For example, maximum likelihood inference is an inference technique that is only conditioned on the data, thereby ignoring the prior completely. Instead of computing $P(B|D)$, maximum likelihood inference uses the likelihood, $P(D|B)$, and computes $\hat{B} = \text{argmax}_B\, P(D|B)$ where $\hat{B}$ is used as a point estimator for $E[B|D]$ the expected posterior belief given the data. For example, if we are trying to infer the mean weight of mushrooms from a sample of mushrooms, one approach would be to assume that the mean weight of a type of mushrooms $\hat{\mu}$ is approximately equal to the mean weight m of the sample of mushrooms in D. This approach is the maximum likelihood solution because $\hat{\mu} = \operatorname{argmax}_{\mu} P(D|\mu)$. This technique works quite well in practice when there are large amounts of data. However, $E[B|D] = \int_B P(B|D)BdB$ where $P(B|D)$ is equal to the result of Bayes law. Notice that $\operatorname{argmax}_B P(D|B) \neq E[B|D]$, and that Bayes Law is needed to actually compute $E[B|D]$. Thus, from a Bayesian perspective, the maximum likelihood technique is flawed on a fundamental mathematical level. Luckily, if the prior is sufficiently wide, then, in the presence of large amounts of data, it can be shown that the maximum likelihood technique approximates the correct answer ($\operatorname{argmax}_B P(D|B) \approx E[B|D]$ for large D). Because the maximum likelihood technique approximate the correct one, its theoretical failings do not mean that maximum likelihood techniques are not useful in practice, rather they mean that in theory they should only be viewed as useful approximations to the correct Bayesian approach. In fact, since the Bayesian approach *is* the mathematical solution to learning, a Bayesian would argue that every learning solution that gives good results in practice, must do so because it approximates the true solution in some way (see Section 5.2 and Section 3.4). Therefore, a theory of the mathematics of learning should be built upon the Bayesian approach, since it is the only mathematically correct way to solve these problems, and since other successful approaches are simply approximations to the correct approach.

The only way to disagree with Bayes law as a model of learning is to take issue with the axioms of probability which demand Bayes law as a solution. Yet Kolmogorov showed that the axioms flow smoothly from set theory, and that they accurately model frequencies of occurrences [51]. Jaynes showed that these same axioms are the only consistent way of extending traditional logic to model degrees of belief [45]. Finally, DeFinetti showed that if an agent makes decisions according to any other principle than utility theory built upon these same axioms of probability, then that agent can be exploited [23]. In the past these authors have represented competing groups who have intensely debated the true meaning of probability theory, with the followers of Kolmogorov being called "frequintists" and the followers of DeFinetti being called “Bayesians.” However, a better view presented by Jaynes is that each of these authors have actually strengthened the case for what he calls simply “probability theory” neither Bayesian nor frequintist. Taken together these authors show that probability theory is the only right way to model uncertain logical inference, frequencies of occurrence, or the beliefs necessary for decision making in uncertain environments. Each of these approaches strengthens the others, and shows that these sorts of problems can only be solved consistently by a model that follows the axioms of probability [45]. Because Bayesian inference flows from these axioms of probability, it can be seen as the only correct solution to inference, while Frequentist inference can be seen as an extremely useful tool for approximating the true answer, and both approaches are now useful in their proper place.

1.2.2 Decision Theory as a Fundamental Model of Learning

Perhaps the most important practical advantage of the Bayesian approach to learning is “the principle of Bayesian optimality” which states that when used in conjunction with decision theory, decisions based on inference using Bayes law will always produce an expected utility equal to or greater than decisions based upon any other technique [25], and if you make decisions according to any other principle, you will either behave identically to the Bayesian approach or you can be exploited [23]. Thus, the Bayesian approach to learning can be said to be “optimal” with respect to expected utility. Throughout this work we will make repeated reference to this essential idea. This brings us to the next element that should be an essential part of any theory of learning, namely decision theory.

Machine learning is sometimes analyzed in isolation from control theory or decision theory. However, in practice we do not cluster things, classify things, approximate functions, or search for function optima for the intrinsic value of clustering, classifying, approximating, or optimizing. Rather, we do these things to aid in decision making. This implies that the theory of learning should be intimately tied to the theory of making decisions.

Decision theory uses the principles of statistics and wagers to construct a formalism for expressing preferences in terms of "utility" functions or "loss" functions. Utility theory also provides a technique for making choices in a way to maximize expected utility given risk averse or risk seeking assumptions. This approach formalizes the expression of the performance measure required by Mitchell's definition of learning. Decision Theory also provides a mechanism for making decisions based on that belief in order to maximize the performance measure. Many of the questions that a theory of learning should be able to answer depend directly on decision theory. For example, computational complexity, active learning, and traditional No-Free-Lunch.

There are two ways in which utility affects function learning. The first is through end use. End use utility involves all utility that comes as a result of using the outputs of the learner. The other way in which utility can affect function learning is through sample costs. Sample costs play an important role in fields such as sample complexity and active learning.

1.2.3 A Bayesian Decision Theoretical Model of Machine Learning

In the past, many approaches to creating a theoretical framework for machine learning have attempted to create a theory that can describe all possible learners. Instead we prescriptively produce the correct, optimal solution to learning. This will mean that there are some learning approaches that can not be directly modeled in our theory, however, as we have argued above, if such learning techniques produce good results in practice, they must do so because they approximate the correct Bayesian solution, thus they work because they are approximations to a learner that can be modeled by our approach. A theoretical framework that only addresses the correct solution can be much cleaner and simpler than a more inclusive theoretical framework, while still having important things to say about the mathematics of learning in general.

We therefore propose a Bayesian Decision Theoretical approach to the theory of function learning. This approach is based on a graphical model which analyzes the relationships between the various parts of an un-supervised learning, semi-supervised learning, supervised learning, and empirical function optimization problem. We will see that these problems all share the same basic graphical model. This approach, the Unified Bayesian Decision Theoretic Model or (UBDTM), can have three main uses:

First: the model can be used to produce important theoretical results by allowing us to write down the mathematical definition of the problems that we are trying to solve. Perhaps more importantly, by thinking about the function learning problems in this way, we can create a shared vocabulary/framework for discussing learning techniques, we can clarify our thoughts, and we can more easily understand what is really happening, and why our algorithms are or are not working.

Second: the model can be used to guide the generation of new learning techniques. The model dictates how a given inductive bias should be used to generate a learning algorithm. The model will often dictate a solution that is computationally complex. However, when developing a heuristic approximation the model can help us to understand exactly what simplifying assumptions we are making. Knowing what our simplifying assumptions are can help us to predict when our heuristic will perform well, and when we would expect it to fail.

Third: The model can be used to better understand existing techniques. All existing techniques that work do so precisely because they are approximating the correct solution under some simplifying assumptions. By comparing an existing technique to the true solution built upon the UBDTM, we can analyze and better formalize the simplifying assumptions that the existing technique is making, and thereby better predict when the existing technique will succeed and when it will fail.

1.3 Related Work

This dissertation consists of a statistical decision theoretical approach, theory, and model of function learning. We will focus on supervised learning but will also show that the model unifies supervised learning with un-supervised learning, semi-supervised learning, and empirical function optimization, showing that these four problems essentially share the same underlying statistical model. As such, it covers a broad set of divergent areas. Especially relevant are approaches to statistically modeling the supervised learning and function optimization problems. Also relevant are theoretical frameworks for learning. Since our framework is based on Bayesian statistics and decision theory, Bayesian approaches to supervised learning and function optimization are especially relevant. Applications to active learning and selective sampling are also relevant.

In Section 1.3.1 we will cover the supervised learning and empirical function optimization problems and briefly review the traditional non-Bayesian approaches to these problems. In Section 1.3.2 we will overview the most common Bayesian approaches to these problems. After discussing techniques for solving the supervised learning and function approximation problems we will focus on various theoretical issues including: No-Free-Lunch (Section 1.3.3), meta learning and transfer (Section 1.3.4), active learning (Section 1.3.5), followed by learning frameworks and models (Section 1.3.6).

1.3.1 Traditional Learning Techniques

Supervised Learning

Supervised learning is the problem of learning a function from labeled examples. When the function outputs are discrete this problem is sometimes called classification. When the function outputs are continuous, supervised learning is sometimes called regression. The problem of learning functions from examples is quite common and is encountered in many domains including natural language processing, control, artificial intelligence, and expert systems.

There are literally thousands of techniques for solving supervised learning problems. Some of the most common include: artificial neural networks (ANNs), decision trees, support vector machines, and nearest neighbor techniques. Several good surveys of this broad class of algorithms include [74] and [29]. We will give more specific attention to Bayesian techniques in Section 1.3.2.

One traditional supervised learning technique that is central to our research involves the Cerebellar Model Articulation Controller (CMAC) first proposed by J. S. Albus [1]. The CMAC is essentially an ANN with a very specific network topology. Although it was initially proposed for control, it actually has much broader applications for supervised learning.

Empirical Function Optimization

The function optimization problem involves finding the extremum (or multiple extremum) of a function. Usually this is a minimum or a maximum. Finding the extremum of a known function is known as computational function optimization. Finding the extremum of an unknown function by repeatedly sampling the function is known as empirical function optimization. Because our work focuses on empirical function optimization, we will focus our related work on sampling approaches to function optimization. The samples from the function are similar to the examples used in supervised learning. Thus, sampling approaches to function optimization can be seen as a special case of supervised learning where we only care about learning the location of the extremum.

There are many sample based approaches to function optimization, the most common are simulated annealing [95], gradient descent [95], particle swarm optimization, and genetic algorithms.

Particle swarm optimization (PSO) is an evolutionary optimization algorithm that was discovered during experiments with simulated bird flocking [49, 50]. It has proven to be an effective approach to the optimization of a useful class of functions. Additionally, PSO's are attractive because of their simple implementation with relatively few parameters to tune.

Genetic Algorithms (GA) represent an older and more popular set of evolutionary algorithms for function optimization [42]. In its most basic form, a GA creates an initial population, evaluates each member, then combines information from selected members to form a new population, and the process starts again. Each generation in the population is formed using some combination of crossover and mutation.

Good brief surveys of these techniques have been created by Russell and Norvig [95] and Mitchell [74].

1.3.2 Bayesian Learning Techniques

Bayesian statistics involves using Bayes law to turn the probability of data given some parameters into the probability of the parameters given some data. Degroot provides a good survey of Bayesian statistics [26] and of the use of decision theory with Bayesian statistics [25].

Several of the most common un-supervised and semi-supervised clustering approaches have a decidedly statistical interpretation. Such techniques include k means [59], and Expectation Maximization (EM) [27].

Bayesian approaches to the supervised learning problem are gaining in popularity. Typically the probability of a function output given the model parameters and some features (the function input) are given. Bayesian approaches to supervised learning typically use Bayes law to compute the reverse probability of the model parameters given function input-output examples (the training data).

Bayesian networks, graphical models, and decision networks (decision graphs) [46] are essential to understand in order to follow the majority of the literature surrounding Bayesian learning. It is useful to depict the variables and relationships as a graph where nodes represent random variables and a lack of an edge represents a conditional independence assumption. In a directed graphical model (or Bayesian network), the edges typically represent causal relationships, but can also be used to model correlation between non causally related nodes. Every node in the model has an associated distribution: conditional if the node has parents, unconditional if not. Thus, a Bayesian Network is essentially a graphical representation of the independence assumptions made between random variables. By graphically representing which variables influence which other variables, the relationships of any proposed model become immediately apparent. Further, in this representation a problem's natural causality can be easily represented and visualized. These conditional independence assumptions are also algorithmically important since they can minimize the amount of computational and representational complexity required to ask and answer questions about random variables in the network.

Bayesian Networks (together with approximate sampling techniques) allow researchers to specify a set of random variables, the conditional independence assumptions made among these random variables, and then ask and answer important questions about the probabilities of any random variable in the graph given the values of any of the other variables. These models are typically acyclic when used for Bayesian inference. In this case, Bayes Law and simple rules of probability can be combined to compute a distribution over any query variable given information about any number of evidence variables in the model [95]: $\rho(\text{query}|\text{evidence})$. Any variables that do not represent queries or evidence are integrated away in the inference process. This is a powerful tool for determining information about hidden state from available observations. Most Bayesian approaches to supervised learning can be expressed as Bayesian Networks.

One of the most common Bayesian techniques for supervised learning is known as Naive Bayes [69]. This technique produces simple solutions where function input features are considered to be conditionally independent given the function output. This independence assumption is the primary weakness of these techniques, but it also allows them to be solved with reduced computational complexity. Naive Bayes has proven especially effective for problems such as document classification and spam filters.

Other supervised learning techniques not normally considered to be statistical can often be recast in a Bayesian way. Supervised learning often involves dynamically setting various parameters of a given learning model. For example, Artificial Neural Networks consist of a set of weights, and the goal of training is to set those weights in a way that will effectively represent the function that the network is attempting to learn. When a Bayesian encounters an unknown parameter (such as the weights of a neural network) the obvious thing to do is to place a prior distribution over the possible values that we might expect the weights to have. Then, if they can define what the output of the system would be given a specific weight setting, they can use Bayes law to determine the posterior probability of the weights given some training examples. This sort of approach to parameter setting in supervised learning has been widely explored, and has yielded valuable results. Bayesian linear regression, for example, places probability on linear functions which corresponds to the probability density of the parameters of the linear function. Similar approaches have also been successfully applied to decision trees [13], neural networks [24, 62, 66, 60, 87], and support vector machines [9]. Bishop and Tippling have produced a good survey of such techniques [9].

Statistical approaches have also been influential in the domain of un-supervised and semi-supervised learning. Some of the most common approaches include EM [27], and k means [59].

Perhaps the most similar application of Graphical Models to our own use of Graphical Models for EFO involves an application of graphical models to modeling the adaptive filtering task [126].

One reason why Bayesian statistics is so appealing is its connection to Utility Theory. The principles of Utility Theory were first used by Blaise Pascal in his famous "Pascal's Wager" which argues for wagering on the existence of God [89]. Utility Theory proposes a technique for selecting actions that will optimize expected utility given distributions over the random variables involved in the decision process. These techniques are guaranteed to be optimal with respect to expected utility only if the distributions were computed according to the rules of Bayesian Statistics [25]. Most supervised learning approaches assume a 0-1 loss function. However some work has been done on learning with more general cost functions [30] [125].

We will also be applying the Bayesian approach to function optimization. Optimization through Bayesian inference is not unique to this work: Stuckman [105] and Törn [110] survey a number of Bayesian optimization approaches. Mockus [75, 76] and Törn [110] also contribute important ideas to these methods. Many existing techniques rely on Gaussian Processes [61], including Kriging [52]. Particle Swarm Optimization has also been adapted to create and use distributions inferred through a Bayesian model [78, 79].

Other optimization algorithms exist that specify a function class. In particular, meta models are used to tune an algorithm for a specified function class by reducing the need for expensive function evaluations [47, 96, 31].

Estimation of Distribution Algorithms (EDAs) generate distributions over likely minima [53]. While similar in spirit, EDAs and our approach have a critical difference: EDAs rely on a *fixed* and usually *implicit* function class, encoded in the distribution representation and the algorithm used to obtain it; whereas we will rely on an *exchangeable* and *explicit* function class, encoded in the model distributions.

1.3.3 No-Free-Lunch

The theory behind function learning is just as important as the various techniques used. The No-Free-Lunch theorems represent one of the most important theoretical results in supervised learning, function optimization, and search.

The concepts of the NFL theorem for supervised learning were first introduced by Mitchell who attempted to show the futility of what he called “bias free learning” [72]. This idea was more formally discussed by Wolpert who coined the phrase “No-Free-Lunch” (NFL) to describe the phenomenon [122]. These theorems deal with the generalization potential of supervised learning to inputs that are “off training set,” that is, to function inputs that have not been previously observed. These papers essentially show that if we average over the space of all possible discrete functions, then all classes are equally likely in locations that have not yet been sampled (that are off training set).

It should be noted that when the machine learning community refers to the term "bias" they usually mean the "inductive bias," alternatively represented as the learning model's preference for one function *over* another, or as a prior in Bayesian approaches. This meaning of the term bias differs substantially from the statistical use of the term, which usually refers to the "bias" of a statistical estimator. When we use the term bias, we will be referring to the inductive bias of a machine learning algorithm unless stated otherwise.

Since empirical function optimization is so closely related to supervised learning, we should not be surprised that in the context of search or of function optimization all search policies are equally effective off training set [124, 123] when averaged over the same space of all possible finite discrete functions. If it is impossible to predict values off training set given the uniform function prior (i.e. if there is no supervised free lunch) then we should not be surprised that any sampling or search technique for selecting a new sample location (an off training set sample location) would perform equally well. This means that all empirical function optimization techniques including gradient ascent, gradient descent, and even random all perform equally well when averaged over the space of all possible finite discrete functions. While this idea casts a shadow over the possibility of developing an efficient black-box optimization algorithm, it is in many ways unsurprising. Whatever the function on which an algorithm performs efficiently, another function may be crafted which causes the algorithm to fail.

There are some cases where NFL can hold, even for distributions over expected functions other than uniform. Fortunately, this simply states that uniformity of function distribution is a sufficient but not a necessary condition for NFL to hold; it says nothing about whether some classes of functions may be chosen for which an algorithm may be expected to do better than random search. In fact, many useful function classes exist over which algorithms may be meaningfully ranked by performance [19, 44]. Computational complexity theory lends intuition to this last idea. The sorting of integers, for example, is a problem that has a proven worst-case complexity; various sorting algorithms may be ranked even when considering all possible instances of the sorting problem, since some obtain the proven complexity result and others do not. In fact, the concepts of computational complexity and No-Free-Lunch are closely linked; rather than rank algorithms over a class of functions, it is also possible to rank the complexity of function optimization without respect to the algorithm which is applied [117] [67]. Though a great deal of work has been focused on No-Free-Lunch and related theorems, it is still not entirely clear how to characterize the complexity of a given optimization or supervised learning problem.

1.3.4 Meta Learning and Transfer

One technique often employed to improve supervised learning is to use information from one learned task to improve the learning of another related task. Humans can often improve their performance on one task by learning related tasks. Learning research has naturally attempted to duplicate this ability in machine learning systems.

Learning to learn is the process of learning a set of related tasks at the same time in a way that will hopefully improve the learning of all the tasks. Transfer learning is the related process of using information from one task to improve the learning of another task. Thrun has edited a good collection of papers on this topic and this text is a good overview of techniques in this area [107].

The NFL theorems indicate that learning requires a bias. Meta learning is the process of learning a meta bias. Learning a meta bias is one way of learning to learn and of implementing transfer learning. Baxter has shown that meta learning can be couched in a Bayesian perspective using hierarchical Bayesian models. He has shown that this approach can significantly reduce the theoretical complexity of learning [4, 5].

1.3.5 Active Learning and Selective Sampling

One of the major bottlenecks of machine learning is the availability of good training data. The examples used in supervised learning are often very expensive to gather. One method of combating the scarcity of training data is to allow the computer to actively request training data that it believes will most improve its performance. This will hopefully allow for better performance with less training data. Some initial work has also been done to combine active learning with semi-supervised learning, which has also been used to deal with the sparsity of labeled training data [127].

Active learning is actually just one technique in the larger domain of selective sampling approaches. Selective sampling can take several forms, including: pedagogical learning [104] where a teacher selects the most useful examples for the machine learning student; pool based active learning where the machine learner selects the most useful examples from a pool of unlabeled examples; query filtering where the machine learner is presented with a stream of examples and the machine learner must decide whether to request help on each example; and on site learning [70] which is like query filtering except that training and testing are no longer considered to be distinct phases. Rather the learner must learn the function, label some examples, and request help with others while on line during actual use.

Several techniques for active learning have been proposed, including: sampling where we do not have data [116]; sampling where we perform poorly [58]; sampling where we expect sampling to change our model [3]; sampling near where we previously found data that improved the model [100]; sampling near the decision boundary [97] [109] [101]; sampling where the learner is most uncertain in its outputs (Query by Uncertainty, QBU, or uncertainty sampling) [106]; sampling where a committee of learners most disagree with each other (Query by Committee or QBC) [103]; and statistically motivated techniques [20]. The most important of these has been QBU, QBC, and the statistical techniques.

Uncertainty sampling originates with Thrun [106]. We refer to uncertainty sampling as Query by Uncertainty (QBU). Early experiments involving QBU were conducted by Lewis and Gale [57] on text classification, where they demonstrated significant benefits of the approach. Lewis and Catlett [56] examined its application for non-probabilistic learners in conjunction with other probabilistic learners. Anderson and Moore [2] explored QBU using HMMs.

Query by Committee (QBC) was introduced by Seung [103]. Freund [32] provided a careful analysis of the approach and proved that in some cases QBC can exponentially reduce the number of queries needed. Within QBC there are many different ways in which the committee members are created. The committee can be generated using bootstrap sampling on the training data, by using redundant views of the function's features [86] [35], or by sampling from the hypothesis space either uniformly, or according to some prior or posterior distribution over functions in the hypothesis space [21]. One advantage of QBC over QBU is that QBC can be performed with purely discriminatory learning techniques because class probabilities are not used directly [99].

The most common statistically motivated techniques revolve around computing the expected improvement generated by each possible example [20]. Such techniques are often considered to be statistically optimal [94] [127], despite the fact that they are actually greedy approximations, and therefore sub-optimal. They are usually computationally complex, and computing the true optimal solution (without the greedy assumption) is most often prohibitively expensive. These techniques are also only optimal with respect to very limited cost metrics, and experimentation with more general cost measures is needed. Some initial strides in cost sensitive active learning have been made [68]. However, the true statistically optimal solution to active sampling problems is known as the Expected Value of Sample Information or EVSI [90]. EVSI is the statistically "optimal" technique based on the laws of Bayesian inference and decision theory. The primary difference between this technique and the so called statistically "optimal" technique sometimes presented in supervised Active Learning papers [20] is that EVSI is a more general class of techniques and does not make the assumption that the cost/utility function is misclassification error rate. EVSI requires that the cost/utility function be specified explicitly and becomes equivalent to these simpler techniques when the misclassification error rate is used.

One application of active learning related to our work involves corpus creation [34] [108]. Most results using active learning have been positive. However in some situations the examples chosen by active learning are also often harder for humans to label correctly [35]. In this situation active learning should be employed with caution.

Function optimization techniques (such as particle swarms [49] [50]) and most evolutionary algorithms (such as Genetic Algorithms [42]) primarily consist of policies for determining sample locations. As such, they can be though of as performing active learning, but with a different goal, namely, finding the function's extremum.

1.3.6 Theories and Models

Less work has been done on modeling the process of supervised learning and function optimization directly. Good models and theories of the general processes involved are useful because they can help us ask questions about what functions are learnable, how difficult they are to learn, and whether one learner can outperform another on average. We will here survey some of the most relevant theoretical results in supervised learning and function optimization.

Supervised Learning

Perhaps the two most commonly used theories of supervised learning include PAC learnability and VC dimensionality. PAC learnability attempts to expand the general concepts of computability to learnability, allowing us to quantify the set of functions that are "learnable." It does this by determining classes of problems are probably approximately correct (PAC) after a polynomial amount of training data [114]. The VC dimension of a hypothesis space is a measure of the expressability of a hypothesis space, and thus of the sample complexity of learning in that hypothesis space [115]. Both of these techniques allow loose upper bounds on the amount of training examples needed to learn with given techniques (the sample complexity) but the bounds are worst case instead of average case and are often too loose to be useful in practice [13] [40]. VC dimensionality differs substantially from a Bayesian approach in that it considers only the hypothesis space (the representational bias of a learning algorithm) without placing any probability measure on the contents of the hypothesis space (thereby considering all hypotheses in the hypothesis space to be equally likely, and thus ignoring the preferential bias of the algorithm).

Buntine explored some of the implications of a Bayesian approach to the theory of machine learning [13]. He computed bounds on sample complexity that took preferential bias (expressed as a distribution over the elements in the hypothesis space) into account and found that his bounds were much tighter than those obtained using PAC or VC dimensionality. Unfortunately his work was done before the insights of NFL, and although he discusses some of the ideas, he does not investigate the implications of a Bayesian model on active learning.

In order to prove his NFL theorems Wolpert used a formalism known as the Extended Bayesian Framework (EBF) [122]. This formalism consists of function inputs $\mathbf{x}$, function outputs y, the actual function that maps them f, and the hypothesis h. However, Wolpert only uses this formalization to prove NFL, and fails to put any distribution other than uniform over f. A more general framework could use these ideas to discuss learning with a given bias by placing distributions other than uniform on f.

Baxter's Meta Learning paper [4] could also be thought of as a theory paper inasmuch as it connects the concepts of Hierarchical Bayes and meta learning.

Function Optimization

Monson [77] models the empirical function optimization problem as a Bayesian network. This model for optimization forms the basis of our model for supervised learning. This dissertation also includes a section published elsewhere and co-authored by myself which introduces the idea that a Bayesian Decision Theoretical approach can be applied to active learning in the function optimization case [80], but which does not extend these ideas to supervised learning as we will do here.

1.4 Outline rest of dissertation

The first part of the dissertation will provide a set of theoretical results based upon our Bayesian model of the function learning problem. In Chapter 2, we will introduce our model of un-supervised learning, supervised learning, semi-supervised learning, and empirical function optimization. We will also briefly describe some of the implications of such a model. In Chapter 3, we will compare our model with Wolpert's EBF, and we will deal with the implications of our model for No-Free-Lunch. We will provide our own version of the NFL theorems based upon our model, and examine the differences between these results and the NFL results obtained using Wolpert's EBF. In Chapter 4, we will deal with the implications of end use on the model, and introduce the "No-Free-Dinner" theorem. In Chapter 5, we will show the existence of *a priori* distinctions between learning algorithms, and we will discuss the implications of our model for the futility of bias free learning. We will show that the traditional NFL results do not actually show the futility of bias free learning, and we will propose our own reasons for believing in the futility of bias free learning. In Chapter 6, we will extend our model to model sample costs and to deal with selective sampling and active learning for the supervised learning case. We will there present the optimal solutions to these sampling problems, and produce a set of theoretical observations about the selective sampling problems derived from these formulas. We will also present an active learning No-Free-Lunch theorem. In Chapter 7, we will compare our model with Wolpert's model for EFO, and we will then deal with the active learning problem for EFO, and provide a set of theoretical results about EFO derived from our model, including an EFO No-Free-Lunch theorem based upon our model which we will then compare and contrast to the traditional EFO No-Free-Lunch theorems based upon Wolpert's model.

The second part of the dissertation will provide some more concrete applications of the preceding theory. In Chapter 8, we will use the principles of our model to produce an improved Bayesian training rule for the CMAC ANN topology that can be solved in closed form, and we will compare the performance of this algorithm to the standard CMAC training.

In Chapter 9, we will apply the principles of the active learning extensions of our model to a real world corpus annotation project. In Chapter 10, we will demonstrate how the principles of the model can be used to generate an actual EFO algorithm with an example.

In Chapter 11, we will conclude and propose some future work. Chapter 12 is an appendix, where we will present some statistical distributions and derivations.

These chapters were for the most part derived from independent published papers, but some effort has been made to unify the material, and reduce duplication wherever possible. However, each chapter still begins with its own abstract and introduction, and still ends with its own conclusion. This will allow the interested reader to quickly overview the material and contribution of each chapter independently. Because of our attempt to remove duplicated material, however, each chapter no longer stands completely on its own, as must chapters continually refer back to the model, which material has been removed from each chapter and unified in Chapter 2.

Chapter 2

The Unified Bayesian Decision Theoretical Model

Abstract

In this Chapter, we will present our graphical model for function learning (the Unified Bayesian Model of function learning or UBM), and then extend that model to deal with end use utility using decision theory (the Unified Bayesian Decision Theoretical Model of function learning or UBDTM). We will then show how un-supervised learning, supervised learning, semi-supervised learning, and empirical function optimization can be seen as special cases of the more general model. We will also briefly overview some of the implications of such an approach to function learning.

Publication: Some of the material in this chapter has been previously published elsewhere, including the use of the UBM for empirical function optimization (although in those earlier works the model was known by a different name because it had not yet been unified with the model for the other function learning situations) [80, 77], and the use of the UBM and UBDTM for supervised learning [16, 15, 14, 17]. The use of the UBDTM for un-supervised learning, and for semi-supervised learning is presented here for the first time. Some of the general implications of the UBM and UBDTM have also been published elsewhere [16, 15, 17] while others are presented here for the first time.

2.1 Introduction

We will now introduce our Bayesian model of function learning, which we define to be: unsupervised learning, supervised learning, semi-supervised learning, and empirical function optimization. It is common to use graphical models for specific learning situations, however, in this case, we are creating a more general model that represents the actual problem of learning from data for any of the function learning problems. By modeling these learning scenarios, it is possible to use the model to make general theoretical statements that will hold for all the specific instances of these learning situations. This approach is similar to that taken by Wolpert when defining the Extended Bayesian Formalism (The EBF), a general model of supervised learning which he then used to prove No-Free-Lunch for all instances of supervised learning which follow the EBF assumptions[121], (for a comparison of our model and the EBF see Section 3.4).

We will first define the Unified Bayesian Model (UBM), a general utility-free model of the function learning problems. We will then define the Unified Bayesian Decision Theoretic Model (UBDTM), which adds the concepts of utility and decision theory to the UBM. By separating these portions of the learning process, it is possible to carefully analyze the interactions between utility and learning, and to formally differentiate between the parts of the learning process that are utility dependent and the parts of the learning process that are utility independent. For example, in Chapter 3 we will see that parts of the No-Free-Lunch concepts depend on utility, and other parts are utility free. We will show that each of the function learning situations are actually just special cases of the more general model, that is, by making additional independence assumptions, or by simply changing the goal of inference, the same graphical model can represent all of these learning situations. This unification will allow us to formalize the exact differences between the learning situations (the exact additional assumptions necessary to simplify the more general model to the specific cases) and to see exactly how these learning situations are related, and thus to make theoretical observations that will hold for all the learning situations simultaneously.

Using these models it is possible to determine the mathematical formulas that represent the optimal solutions (with respect to expected utility) to the function learning scenarios. In practice these problems are not normally solved optimally, given the computational complexity of the optimal solution. However, the optimal solutions provide a theoretical glimpse of how these problems *should* be solved. From our perspective, the equations for the optimal solutions *are* the mathematical definition of the the function learning problems. Even when these problems can not be directly solved optimally, understanding the mathematics of the problem's definition is still useful. Such an understanding can be important theoretically, can improve our understanding of why existing heuristics work, can help to explain when and why they sometimes fail, and can aid in the creation of better heuristics in the future.

2.2 The Unified Bayesian Model (UBM)

The Unified Bayesian Model (UBM) is a utility free, general model of the function learning problems where un-supervised learning, supervised learning, semi-supervised learning, and empirical function optimization can all be seen as special cases of this more general model. This model is based on two very simple fundamental assumptions:

UBM Assumption 1: function learning involves some function f that maps a vector of function inputs $\mathbf{x}$ (sometimes known as features), to function outputs y or to a distribution over function outputs, formally: $p(output = y|input = \mathbf{x}, function = f)$.

UBM Assumption 2: The sampling distribution (the distribution over the feature vectors) is dependent only on f, formally: $\xi(input = \mathbf{x}|function = f)$.

To simplify notation we will abbreviate $p(output = y|input = \mathbf{x}, function = f)$ as $p(y|\mathbf{x}, f)$, and $\xi(input = \mathbf{x}|function = f)$ as $\xi(\mathbf{x}|f)$ etc.

Notice that there is a link between f and $\mathbf{x}$ represented by $\xi(\mathbf{x}|f)$. Yet f is just a mapping between function inputs and outputs. There is no causal relationship between this mapping and the sampling distribution, and in this case, these statistical relationships only measure correlations and not causation. It is quite common for there to be a correlation between the mapping and the sampling distribution. For example, imagine a function that maps mushroom features (size, color, etc) to mushroom toxicity. Although this mapping does not cause the sampling distribution, we would expect some correlation between the mapping and the sampling distribution, since in this case both are caused by the different species of mushrooms encountered. As we shall see, in some situations, it will be useful to add an additional independence assumption and view the sampling distribution as conditionally independent of f as well: $\xi(\mathbf{x}|f) = \xi(\mathbf{x})$. This additional independence assumption can be seen as a special case of the more general model because *lack* of edges represent additional conditional independence assumptions. Therefore as more edges are removed, more assumptions have been made. As we shall see, whether to include a more restrictive independence assumption on the sampling distribution will determine whether the model matches the assumptions of supervised learning or the assumptions of un-supervised or semi-supervised learning.

In order to deal with function optimization, it will be important to also model the distribution over possible extremum values (y^*) and extremum feature vectors (those $\mathbf{x}$ values that map to the extremum). Since multiple feature vectors can lead to the extremum, we will represent the set of such locations as $\mathscr{S}_{\mathbf{x}^*}$. Then let $\xi(\mathbf{x}^*|f)$ represent the probability that feature location $\mathbf{x}^*$ is in the set $\mathscr{S}_{\mathbf{x}^*}$ given a function f. Notice that without making any further assumptions, the function f will uniquely determine the distribution over possible extremum values $p(y^*|f)$, and the distribution over possible extremum feature vectors $\xi(\mathbf{x}^*|f)$. This is because each f defines the mapping between $\mathbf{x}$ and y, and the extremum value and the set of extremum locations are uniquely defined by that mapping.

There are two ways to think about the y^* and $\mathbf{x}^*$ nodes. They could be considered separate random variables as we have done above, or they could be considered as attributes of the F variable and left out (as we will often do in the future). The nodes for the extremum are usually only considered when dealing with function optimization. However, the extremum vectors and value still exists even when performing supervised, semi-supervised, or un-supervised learning. As we shall show, when these nodes are unobserved they will have no impact on inference in the rest of the model, therefore they can be safely ignored when they are unobserved, or when discovering their location is not the goal of learning. It is this observation that allows us to unify the model for learning with the model for function optimization.

The core idea of the UBM is that the function learning problems share a set of common elements that can each be viewed as random variables: F, $\mathbf{X}$, Y, $\mathbf{X}^*$, and Y^*, and a set of common conditional independence assumptions (UBM Assumptions 1 and 2) between these random variables. It is then possible to infer information about some of these random variables by observing others [13]. Especially note that we are treating the unknown function as a random variable. As we shall see, this will allow us to make the bias of the learning model explicit using $\rho(f)$ (See Sections Section 2.4.1 through Section 2.4.2).

As an aid to intuition, we will represent the relationships between these random variables as a graphical model. Graphical models provide a technique for simply representing dependence and independence relationships among random variables [46]. Nodes in a graphical model represent random variables and the lack of an edge represents conditional independence. Shaded nodes represent observed quantities while non-shaded nodes represent unobserved quantities. Plates represent elements of the model that are repeated. To this standard notation, we add nodes on a plate that are partly shaded (striped), which we use when some of the repeated nodes are observed, while others are not.

Given the two assumptions for the UBM, these random variables have a specific relationship, represented by the graphical model in Figure 2.1. This set of relationships is very intuitive, and implies that function outputs y are only dependent on the feature vectors $\mathbf{x}$ and the function f that maps them (UBM Assumption 1) while the sampling distribution will directly depend on nothing but the function (UBM Assumption 2). UBM Assumption 1 also requires that $\mathbf{x}^*$ and y^* will depend on nothing but the function.

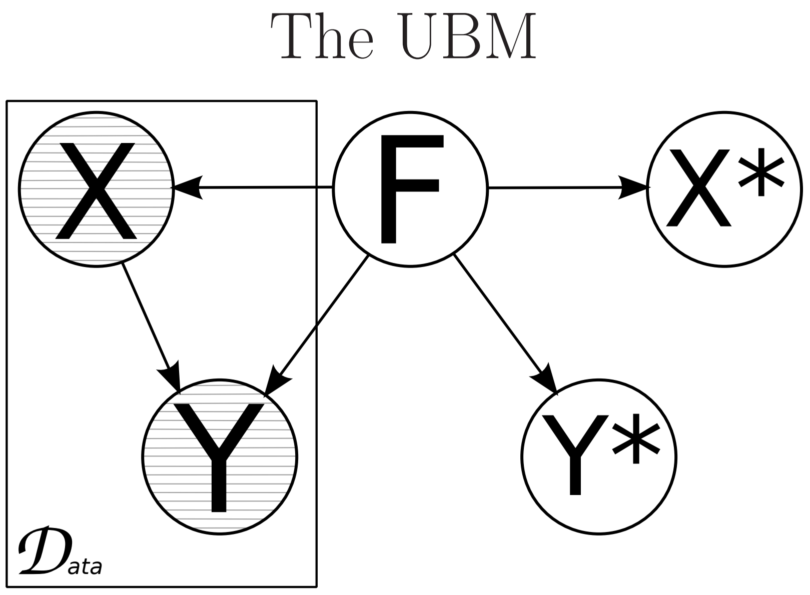


Figure 2.1: The Unified Bayesian Model (UBM) of Supervised Learning, Semi-Supervised Learning, Un-Supervised Learning, and Empirical Function Optimization.

The UBM groups all the "examples" ($\mathscr{D}_{ata}$) into one plate where the $\mathbf{x}$ and y nodes will either be observed or unobserved depending on the kind of inference being performed. As we use the UBM for specific cases, it will be useful to split the examples into sub-groups depending on whether they are observed and on how they are used. In machine learning, it is traditional to call these groups of data "sets."[1] We will define the following sub-sets of examples: all the data $\mathscr{D}_{ata}$; all the observed data $\mathscr{D}^{o}_{ata}$; all of the un-observed data $\mathscr{D}^{-o}_{ata}$; the training set $\mathscr{D}_{train}$ (a set of observed $\dot{\mathbf{x}}$ and $\dot{y}$ pairs); the test set $\mathscr{D}_{test}$, (a set of either observed or un-observed $\ddot{\mathbf{x}}$ and un-observed $\ddot{y}$ pairs); and the pool set $\mathscr{D}_{pool}$ (a set of observed $\breve{\mathbf{x}}$ and unobserved $\breve{y}$ pairs which can then be observed for some sample cost) used in active learning. For more on active learning and pool sets see Chapter 6. It will sometimes be useful to refer just to the $\mathbf{X}$ or Y part of a set. We will represent this idea by placing an $\mathbf{X}$ or a Y as a superscript, for example $\mathscr{D}^{X}_{train}$ and $\mathscr{D}^{Y}_{train}$ represent the $\mathbf{X}$ and the Y parts of the training set. Similar notation can be used with the other sets. By creating special

[1]This convention is unfortunate, since these data sets can have repeated elements, and so are actually not "sets" but are "multisets." However, we will often use the term "set" to refer to these multisets because of the weight of traditional convention.

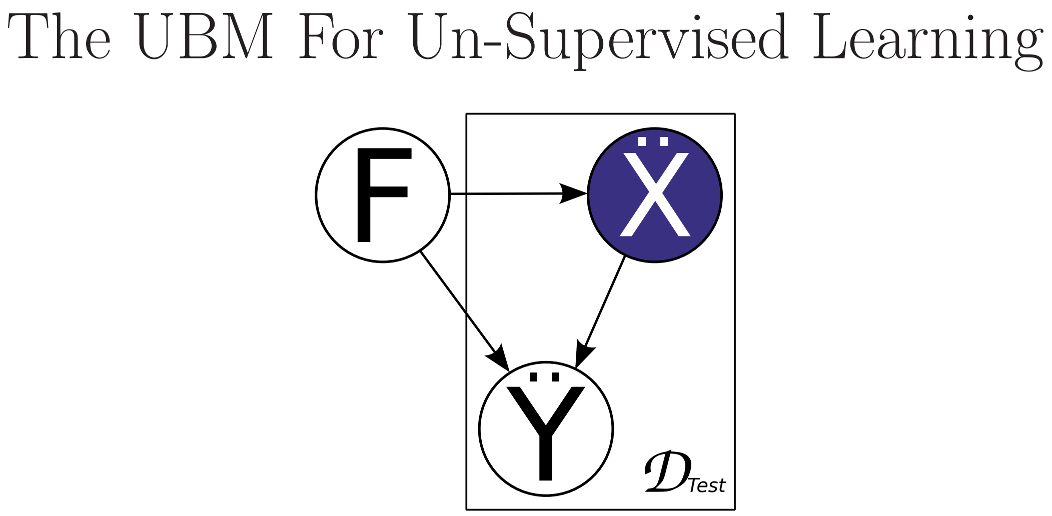


Figure 2.2: The Unified Bayesian Model (UBM) applied to Un-Supervised Learning.

cases of the UBM where different nodes are observed, and where the examples are split into different groups or sets, many varied learning problems can be modeled as special cases of the UBM.

2.2.1 The UBM for Un-Supervised Learning

Given the assumptions of the UBM, it is possible to formally define the mathematical solution to un-supervised learning. To use the UMB for un-supervised learning, a special case of the UBM is developed where $\ddot{\mathbf{x}}$ are observed while $\ddot{y}$ are un-observed (what we called a test set above). Then information about the function f can be inferred from $\ddot{\mathbf{x}}$ (see Figure 2.2). Any situation where $\mathbf{x}$ alone is used to learn about f can be considered to be un-supervised learning. Usually, this information is used to place each $\ddot{\mathbf{x}}$ into a cluster. The goal is usually to get the clusters to match the class outputs $\ddot{y}$ as closely as possible. However, many other un-supervised learning scenarios are possible.

Using Bayes law it can be shown that:

$$\rho(f|\ddot{\mathbf{x}}) = \frac{\xi(\ddot{\mathbf{x}}|f)\rho(f)}{\int \xi(\ddot{\mathbf{x}}|f)\rho(f)df} \,. \tag{2.1}$$

Equation 2.1 can be used iteratively in order to compute the posterior $\rho(f|\mathscr{D}^X_{test})$. Notice that although the $\mathbf{X}^*$ and Y^* nodes are technically present during un-supervised learning, since they are unobserved they will play no role in inference about F, $\mathbf{X}$, or Y, and so they can be ignored in this case.

Then, the posterior probability that an example belongs to a class (the posterior predictive) can be computed by marginalization as follows:

$$p(\ddot{y}|\ddot{\mathbf{x}}, \mathscr{D}^X_{test}) = \int p(\ddot{y}, f|\ddot{\mathbf{x}}, \mathscr{D}^X_{test})df \ , \tag{2.2}$$

using the chain rule we get:

$$p(\ddot{y}|\ddot{\mathbf{x}}, \mathscr{D}^X_{test}) = \int p(\ddot{y}|\ddot{\mathbf{x}}, \mathscr{D}^X_{test}, f)\rho(f|\ddot{\mathbf{x}}, \mathscr{D}^X_{test})df \ , \tag{2.3}$$

which can be simplified using the independence assumptions of the UBM to:

$$p(\ddot{y}|\ddot{\mathbf{x}}, \mathscr{D}^X_{test}) = \int p(\ddot{y}|\ddot{\mathbf{x}}, f)\rho(f|\mathscr{D}^X_{test})df \ , \tag{2.4}$$

For the remainder of this chapter we will omit these sorts of simple algebraic steps.

In un-supervised learning the class or function output is usually not directly observed; therefore, depending on the prior for F and how $\ddot{\mathbf{X}}$ influences F, exchangeability can be a problem. This is not always the case, but when this happens it is not possible to predict the class labels. However, it is still possible to determine the probability that any two $\ddot{\mathbf{x}}$'s belong to the same class:

$$p(\ddot{y}_1 = \ddot{y}_2|\ddot{\mathbf{x}}_1, \ddot{\mathbf{x}}_2, \mathscr{D}^X_{test}) = \int p(\ddot{y}_1 = \ddot{y}_2|\ddot{\mathbf{x}}_1, \ddot{\mathbf{x}}_2, f)\rho(f|\mathscr{D}^X_{test})df \ . \tag{2.5}$$

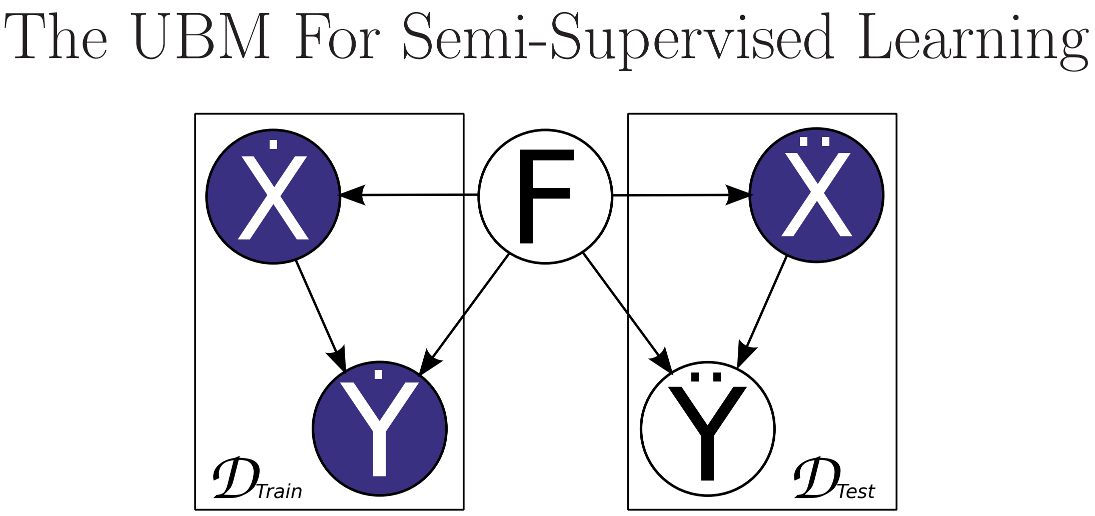


Figure 2.3: The Unified Bayesian Model (UBM) applied to Semi-Supervised Learning.

Notice that $p(\ddot{y}_1 = \ddot{y}_2|\ddot{\mathbf{x}}_1, \ddot{\mathbf{x}}_2, f)$ is again defined uniquely by the function f. This exchangeability problem is why the goal of un-supervised learning is often not to predict the classes or function outputs, but to put the examples into clusters. Thus, un-supervised learning is often referred to as "clustering." It is important to note that this need not always be the case, and un-supervised learning can be thought of as learning any function by observing only function inputs, regardless of whether exchangeability is a problem. There are many variations of un-supervised learning beyond simple clustering. For example, if exchangeability is not a problem, then un-supervised learning can actually classify, or classes need not be discrete, and continuous variations of the un-supervised learning problem could be imagined, allowing unsupervised learning to perform tasks similar to those of regression. Which variation will depend upon the sampling distribution, the prior, and the goal (end use utility, which we will deal with in section 2.3.1).

2.2.2 The UBM for Semi-Supervised Learning

Given the assumptions of the UBM, it is possible to formally define the optimal solution to semi-supervised learning. To use the UBM for semi-supervised learning, the examples are divided into a training set composed of observed $\dot{\mathbf{x}}$ and $\dot{y}$ pairs and a test set where $\ddot{\mathbf{x}}$ are observed and $\ddot{y}$ are un-observed. Information about the function f is inferred from both the training set and the test set. This information is then used to compute the posterior predictive for each $\ddot{y}$ (see Figure 2.3).

Equation 2.1 can be used iteratively in order to compute the posterior $\rho(f|\mathscr{D}^X_{test})$. Then $\rho(f|(\dot{\mathbf{x}}, \dot{y}))$ can be computed using the definition of conditional probability and the chain rule as follows:

$$\rho(f|(\dot{\mathbf{x}}, \dot{y})) = \frac{\rho(f)\xi(\dot{\mathbf{x}}|f)p(\dot{y}|\dot{\mathbf{x}}, f)}{\int \rho(f)\xi(\dot{\mathbf{x}}|f)p(\dot{y}|\dot{\mathbf{x}}, f)\, df} . \tag{2.6}$$

Equation 2.6 can be used iteratively in order to compute the posterior $\rho(f|\mathscr{D}_{train})$. Then, the results of Equation 2.1 can be used as the prior for Equation 2.6 to produce the posterior $\rho(f|\mathscr{D}_{train}, \mathscr{D}^X_{test})$. Notice that the semi-supervised assumption that the sampling distribution depends on the function not only allows information from the test set to update belief about F, but it also should influence the way in which the training set affects F. As we shall see in the next section, when the sample distribution is independent of the function, $\rho(f|\mathscr{D}_{train})$ will be computed differently.

The goal of semi-supervised learning is to compute the posterior predictive $p(\ddot{y}|\ddot{\mathbf{x}}, \mathscr{D}_{train}, \mathscr{D}^X_{test})$. Formally:

$$p(\ddot{y}|\ddot{\mathbf{x}}, \mathscr{D}_{train}, \mathscr{D}^X_{test}) = \int p(\ddot{y}|\ddot{\mathbf{x}}, f)\rho(f|\mathscr{D}_{train}, \mathscr{D}^X_{test})df . \tag{2.7}$$

2.2.3 The UBM for Supervised Learning

The UBM can be used to model the inference portion of the supervised learning problem. In the UBM, supervised learning is identical to semi-supervised learning, except for the additional independence assumption that $\mathbf{x}$ is conditionally independent of f (see Figure 2.4).

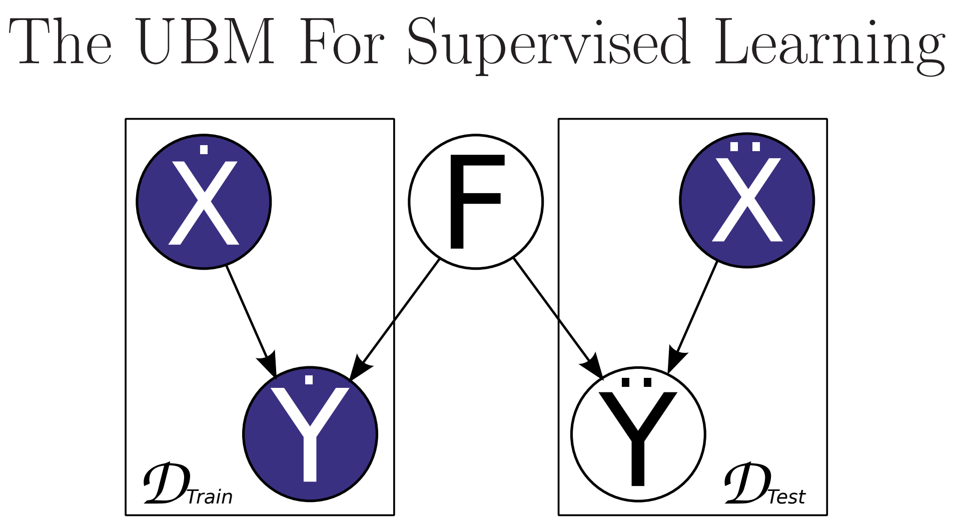


Figure 2.4: The Unified Bayesian Model (UBM) applied to Supervised Learning.

Thus, for supervised learning $\xi(\mathbf{x}|f) = \xi(\mathbf{x})$. This means that supervised learning is a special case of semi-supervised learning, and a special case of the UBM.

With this assumption $\rho(f|\mathscr{D}_{test}) = \rho(f)$ and $\rho(f|\mathscr{D}_{train})$ can be computed by iteratively applying the following:

$$\rho(f|(\dot{\mathbf{x}}, \dot{y})) = \frac{p(\dot{y}|\dot{\mathbf{x}}, f)\rho(f)}{\int p(\dot{y}|\dot{\mathbf{x}}, f)\rho(f)df} \,. \tag{2.8}$$

The goal of supervised learning is to compute the posterior predictive $p(\ddot{y}|\ddot{\mathbf{x}}, \mathscr{D}_{train})$. Formally:

$$p(\ddot{y}|\ddot{\mathbf{x}}, \mathscr{D}_{train}) = \int p(\ddot{y}|\ddot{\mathbf{x}}, f)\rho(f|\mathscr{D}_{train})df \,. \tag{2.9}$$

2.2.4 The UBM for Empirical Function Optimization

Empirical function optimization has a lot in common with the other function learning scenarios we saw above. Empirical function optimization must also update belief about the function F from the observable data. The real difference between EFO and the other function learning situations is that EFO usually implies an active sampling approach, with the specific purpose of learning the location of the function's extremum $\mathbf{x}^*$, while supervised learning may or may not use active sampling, and is focused on learning more about the function than just the location of the extremum. Nevertheless, EFO's focus on active sampling has not changed the fact that semi-supervised learning, supervised learning, and EFO must all update their belief about the function in the presence of data, and given the assumptions of the UBM, that updating must happen in the same way for EFO as it does for the other function learning situations (See Figures 2.5-2.6).

Depending on the data available and the independence assumptions that are relevant to the specific EFO problem at hand, F could be updated either using the semi-supervised approach taken by equations 2.1 and 2.6 (see Figure 2.5), or using the supervised approach of equation 2.8 (see Figure 2.6). In this respect, Empirical Function Optimization *is* supervised or semi-supervised learning with a specific goal (to learn the location of the extremum). This means that although terminology such as a "training set" is not usually used for EFO, the concept of a training set is very much present, and the beauty of our approach is that we can now borrow ideas and concepts from the other learning situations and apply those ideas to EFO.

It is possible to use either a supervised or semi-supervised approach for EFO. Usually, when performing EFO a test set with observed inputs and unobserved outputs is unavailable, so we will usually be most interested in $\rho(f|\mathscr{D}_{train})$, although it is theoretically possible to use $\rho(f|\mathscr{D}_{train}, \mathscr{D}^X_{test})$. In the remainder of this work when performing EFO we will assume that we are only using a training set unless explicitly stated otherwise, furthermore, we will assume the supervised learning independence assumption between the sample distribution and the training set. Although this is not always the case, it is the most common case, and, when it does not hold, one need only substitute the unsupervised learning equations for the semi supervised learning equations.

The extremum does not need to be a maximum or a minimum, but can be any point of interest that we are attempting to find. The term "function optimization" is usually

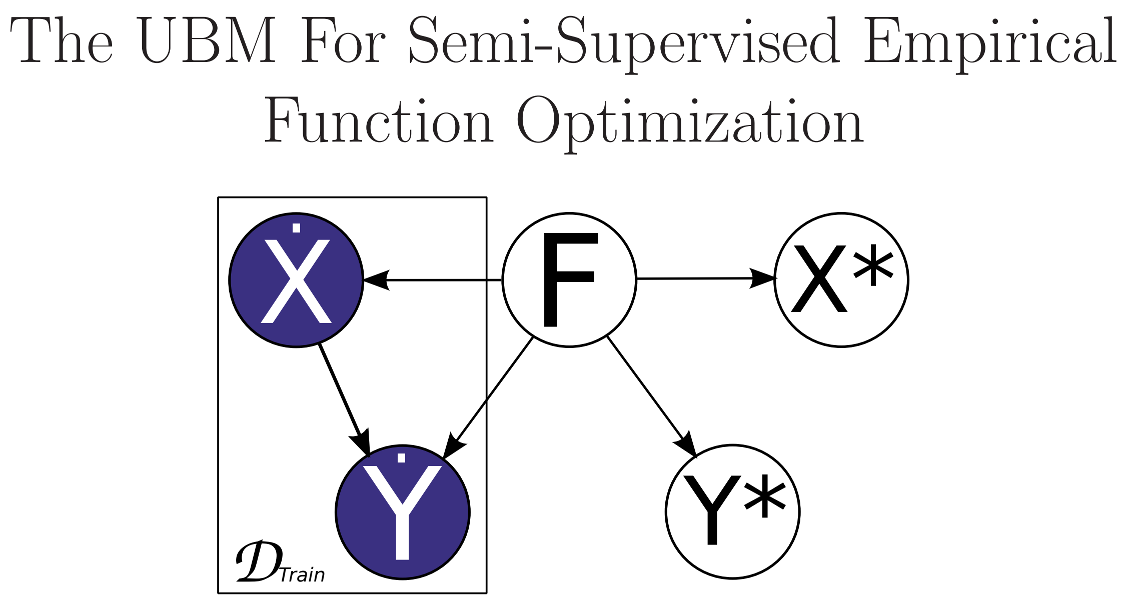


Figure 2.5: The Unified Bayesian Model (UBM) applied to Empirical Function Optimization, with a semi-supervised learning approach.

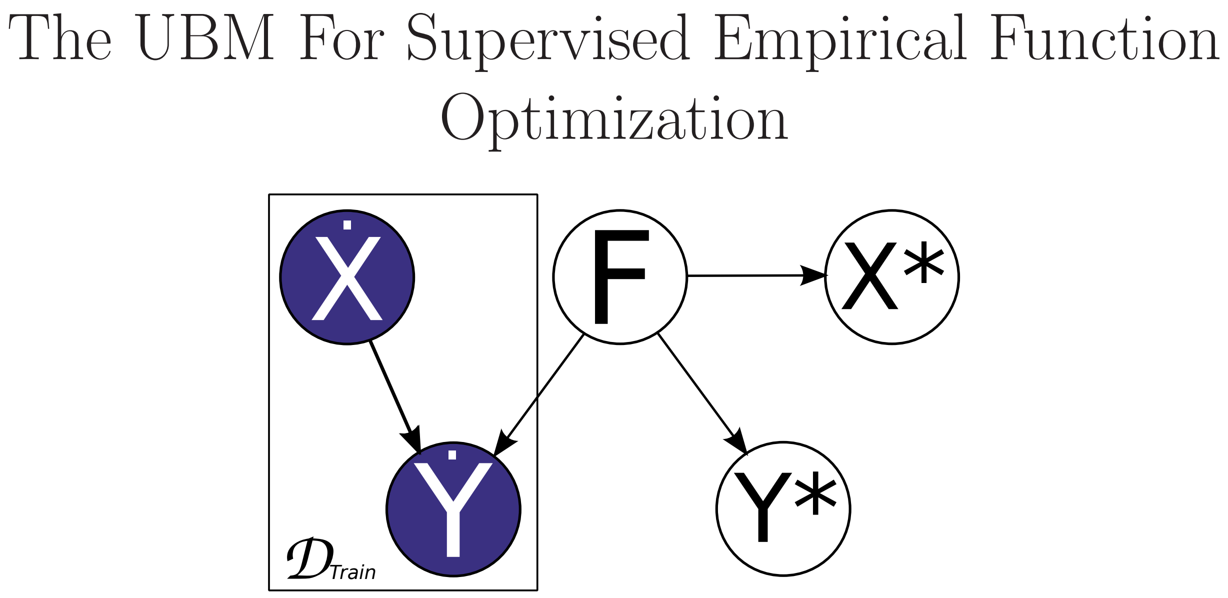


Figure 2.6: The Unified Bayesian Model (UBM) applied to Empirical Function Optimization with a supervised learning approach.

used to refer to a search for a maximum or a minimum, but this is just a special case of the broader class of search algorithms. This formalizes the connection between search and function optimization, which is just search for a specific point (the extremum). Thus both general search and EFO have the same graphical model, and much of our results for EFO also hold for search by simply substituting the idea of the extremum with a more general point of interest.

We will discuss more about active learning, and the application to active learning to EFO in Chapters 6 and 10. For now, we are only interested in the inference needed to perform EFO. Notice that the existence of the $\mathbf{X}^*$ and Y^* nodes does not change the way in which inference about f is performed so long as they remain unobserved.

Once the posterior $\rho(f|\mathscr{D}_{train})$ is found, the goal of EFO requires that we compute the following posterior predictive:

$$\xi(\mathbf{x}^*|\mathscr{D}_{train}) = \int \xi(\mathbf{x}^*|f)\rho(f|\mathscr{D}_{train})df\,. \tag{2.10}$$

Earlier we differentiated between the concepts of *computational* function optimization and *empirical* function optimization. Now, with the notation of the UBM, we can formalize these concepts. Computational function optimization is the problem of determining the extremum location for a known function, in other words, of computing $\xi(\mathbf{x}^*|f)$. Empirical function optimization is the problem of finding an extremum in an unknown function given samples $\xi(\mathbf{x}^*|\mathscr{D}_{train})$. From Equation 2.10 it can be seen that the empirical function optimization problem involves an embedded, repeated computational function optimization problem (we will discuss the implications of this in greater detail in Section 10.3.2). Although it may be computationally intense to solve, each f uniquely defines its set of extremum locations, and, therefore, the computational function optimization problem has a unique determined solution.

The connection between the UBM and empirical function optimization is a connection between machine learning and a specific type of search. Connections between machine learning and search are not new, for example, Mitchell pointed out that supervised learning can be understood as a search through the hypothesis space in an attempt to maximize or minimize some criterion [73]. In that case, learning requires search. In the *empirical* function optimization case, optimization requires first learning about the function from samples, and only then attempting to determine the extremum from the learned $\rho(f)$. This therefore represents a fundamentally different connection between learning and search than the one presented by Mitchell.

In our earlier publications, before this model was unified with the other function learning problems, this special case of the UBM was known as the Graphical Optimization Model (GOM) [80, 77].

2.3 The Unified Bayesian Decision Theoretic Model

Utility most commonly impacts the function learning problems in two ways, through end use and through sampling costs. End use utility involves the utility that arises from actually putting the machine learner's results to use. The UBDTM is an expansion of the UBM to include end use utility assumptions, the expansion of the UBDTM to deal with sample cost assumptions and active learning will be made in Chapter 6 and Chapter 10.

End use in function learning involves some decision d from the space $\mathbb{D}_D$ and some outcome o from the space $\mathbb{D}_O$. The UBDTM makes all of the UBM assumptions, adds the axioms of utility, and makes the following additional assumptions governing D and O:

UBDTM Assumption 1: The outcome depends only on the decision, examples, and the function, formally: $p(o|d, \mathscr{D}_{ata}, f)$.

UBDTM Assumption 2: The utility function depends only upon the outcome, formally: $U(o)$.

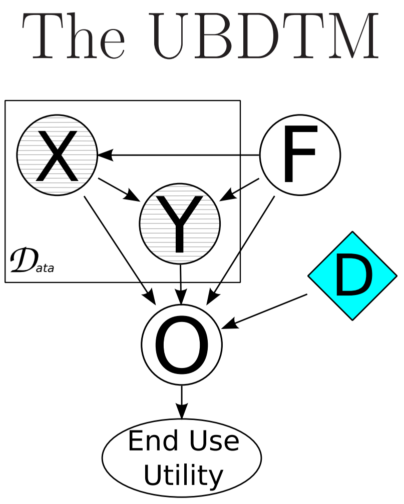


Figure 2.7: The Unified Bayesian Decision Theoretical Model (UBDTM).

These additional assumptions lead to the graphical model of Figure 2.7. At first it may appear strange that the outcome depends upon the data, however, although it is rare for the outcome to depend upon data in the training set, it is quite common for the outcome to depend upon values from the test set. The decision d will most commonly be a vector of decisions (either one for each example, one for each example in a test set, or any other list of decisions). However, in some situations there will be a single decision. Notice also that it is not necessary to explicitly represent the influence $\mathbf{X}^*$ and Y^* have on the outcome, although such influence is still present in the model. This is because these random variables can be viewed as part of F, and are determined by each f. Thus, their influence on the outcomes is modeled by the link from f to o. As we shall see, this influence is especially important in EFO.

As was the case with the UBM, the UBDTM represents the most general case of the function learning problems and each of the function learning scenarios can be seen as special cases of the UBDTM made with specific simplifying assumptions.

Let the observed part of the data (whether it be an observed function input or output, part of a training, test, or pool set) be represented as $\mathscr{D}^o_{ata}$, and let the unobserved part (again,

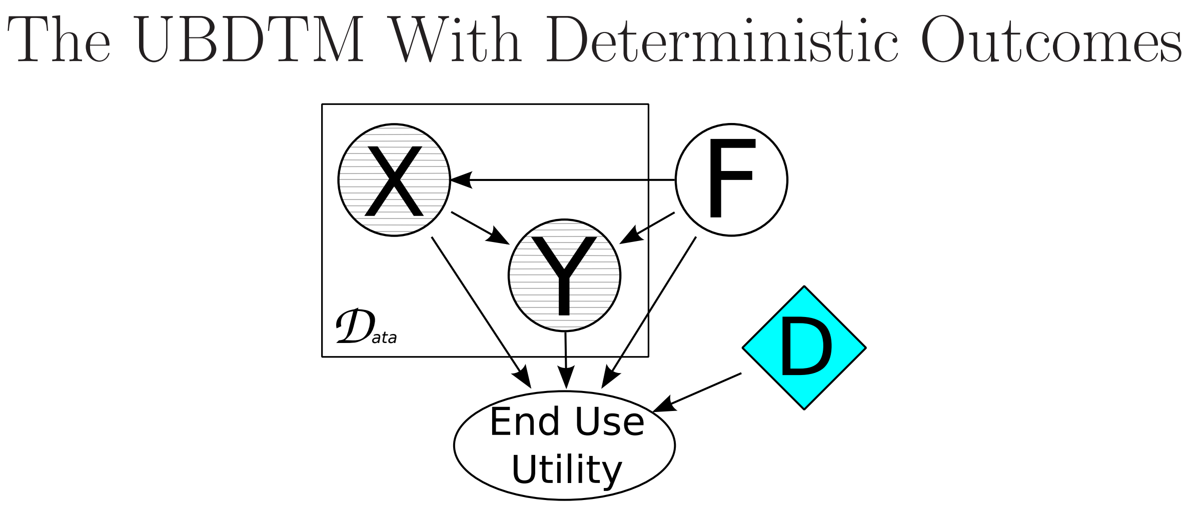


Figure 2.8: The Unified Bayesian Decision Theoretical Model (UBDTM), showing potential simplifications that can be made in the deterministic case.

whether it be a function input or output, part of a test, training, or pool set) as $\mathscr{D}_{ata}^{-o}$, and let all the data be represented as $\mathscr{D}_{ata}$. Then the most general solution to the expected utility in the UBDTM is:

$$E[U|d, \mathscr{D}_{ata}^{o}] = \int_{f \in \mathbb{D}_F} \int_{o \in \mathbb{D}_O} \int_{\mathscr{D}_{ata}^{-o} \in \mathbb{D}_{\mathscr{D}}} U(o) \rho(f, \mathscr{D}_{ata}^{-o}, o | d, \mathscr{D}_{ata}^{o}) df\, do\, d\mathscr{D}_{ata}^{-o}, \tag{2.11}$$

where $\rho(f, \mathscr{D}_{ata}^{-o}, o|d, \mathscr{D}_{ata}^{o})$ can be expanded using the chain rule for Bayesian networks and the independence assumptions of the model as follows:

$$\rho(f, \mathscr{D}_{ata}^{-o}, o|d, \mathscr{D}_{ata}^{o}) = \rho(f|\mathscr{D}_{ata}^{o}) \nu(\mathscr{D}_{ata}^{-o}|f, \mathscr{D}_{ata}^{o}) p(o|d, \mathscr{D}_{ata}^{o}, \mathscr{D}_{ata}^{-o}, f).$$

This is the most general case, and is computationally complex. Luckily, for most specific applications, the distinctions between $\mathscr{D}_{ata}^{o}$ and $\mathscr{D}_{ata}^{-o}$ will be much simpler and, as we shall see, specified by whether we are doing un-supervised, supervised, or semi-supervised type learning. Furthermore, extra independence assumptions can drastically simplify this computation. One common simplification is to assume that the outcome given its inputs is deterministic. This causes the model to simplify as seen in Figure 2.8. Then the expected utility of a decision given the training data is:

$$E[U|d, \mathscr{D}_{ata}^{o}] = \int_{f \in \mathbb{D}_F} \int_{\mathscr{D}_{ata}^{-o} \in \mathbb{D}_{\mathscr{D}}} U(d, \mathscr{D}_{ata}^{o}, \mathscr{D}_{ata}^{-o}, f) \rho(f, \mathscr{D}_{ata}^{-o}|d, \mathscr{D}_{ata}^{o}) df d\mathscr{D}_{ata}^{-o}, \tag{2.12}$$

where $\rho(f, \mathscr{D}_{ata}^{-o}|d, \mathscr{D}_{ata}^{o})$ can be expanded using the chain rule for Bayesian networks and the independence assumptions of the model as follows:

$$\rho(f, \mathscr{D}_{ata}^{-o}|d, \mathscr{D}_{ata}^{o}) = \rho(f|\mathscr{D}_{ata}^{o}) \nu(\mathscr{D}_{ata}^{-o}|f, \mathscr{D}_{ata}^{o}).$$

Notice that this simpler case does not actually break UBDTM assumptions 1 or 2, since the utility still depends only on the now deterministic outcome which, in turn, depends only on the function, decision, and examples, which makes it possible to deterministically determine the utility from the function and the data without explicitly referring to the outcome node. Although the non-deterministic case is the simplest, it is also less general, so for the remainder of this work, unless stated otherwise, we will be using the more general non-deterministic version of the model, and the simplifying deterministic assumptions can easily be applied when appropriate.

Given a "pure" strategy, the optimal decision $d^* \in \mathbb{D}_D$ that will maximize the expected utility can be found as follows:

$$d^* = \underset{d \in \mathbb{D}_D}{\operatorname{argmax}} \, E[U|d, \mathscr{D}_{ata}] \tag{2.13}$$

The above forms the agent's optimal "pure" strategy. In some cases it can be beneficial for an agent to select a "mixed" strategy. Mixed strategies do not use a deterministic decision strategy, instead they return a probabilistic distribution over possible decisions, and then select a decision stochastically according to that distribution. Mixed strategies can be especially helpful in multiagent situations where acting predictably can lead to being exploited. In such situations, optimal performance is usually a function of the policy of *both* agents, and must be defined in terms of fixed point solutions such as the Nash Equilibrium, or Pareto Optimality. In these cases the optimal strategy can be a mixed strategy.

Another typical use for mixed strategies is for encouraging exploration by returning some sub-optimal decision some percent of the time. Softmax or Boltzman exploration involves using mixed strategies to encourage exploration. As we shall see, this is usually the wrong approach, since a better technique can be found by selecting the *sample* with the highest expected utility (see Chapter 6). One mixed strategy that has historically been important (especially for active learning) is to select the decision in a manner proportional to the expected utility as follows:

$$p(d) = \frac{E[U|d]}{\sum_{q \in \mathbb{D}_D} E[U|d_q]}, \tag{2.14}$$

where the utilities have been shifted so that all utilities are scaled to be between 0 and 1. If we assume that the task is to correctly select a class, and that the outcomes are deterministic with the 0,1 utility function (the optimistic inverse of the pessimistic 0,1 loss function), then $p(d) = p(y)$. As we shall see, it is the assumption that the agent is acting according to such a mixed strategy that would justify active learning algorithms that attempt to minimize Bayes Risk (see Chapter 6).

As we shall see in the sections that follow, in each of the function learning scenarios, the equations for the expected utility, and thus for finding the optimal decision, will be simplifications of the above equations made by adding additional simplifying assumptions.

2.3.1 The UBDTM for Un-Supervised Learning

To use the UBDTM to model un-supervised learning, we add the assumptions of the un-supervised UBM, and add the simplifying assumption that the outcome only depends upon a vector of decisions $\mathbf{d}$ (generally one for each $\ddot{y}_i$) and a vector of individual function outputs

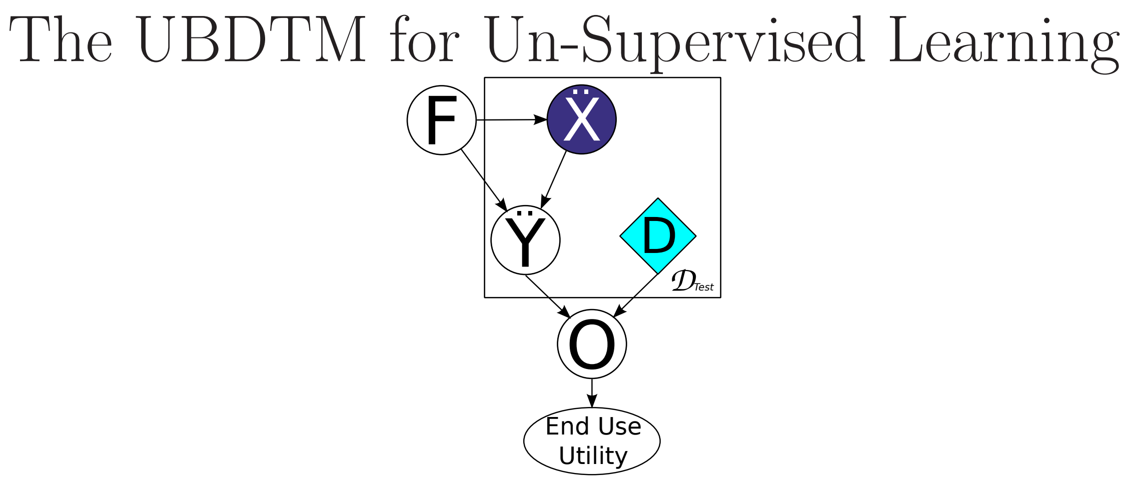


Figure 2.9: The Unified Bayesian Decision Theoretical Model (UBDTM), applied to un-supervised learning.

$\ddot{y}_i$ (see Figure 2.9). Thus, the model of end use in semi-supervised learning can be seen as a special case of the UBDTM with additional independence assumptions.

As we saw before, in un-supervised learning the class is never directly observed, therefore, exchangeability is often (but not always) a problem. Thus, the learner often can not predict the class labels, but it can often determine whether two $\ddot{\mathbf{x}}$ are more likely to belong to the same class, or different classes. Thus, the decision is often to place each $\ddot{\mathbf{x}}$ into a cluster, and the utility is often some measure of inner cluster cohesiveness (how many items in the same cluster are in the same class) and extra cluster differentiation (how often items in different clusters belong to different classes). Notice that O has been placed outside of the plate so that the outcome can depend on the assignments of all the features to classes/clusters. This allows for such issues as inner cluster cohesiveness and extra cluster differentiation to be taken into account.

Un-supervised learning is usually considered to be synonymous with clustering. However, this need not be the case. Any situation where f is a mapping between $\mathbf{x}$ and y, and where information about f is learned by observing $\mathbf{x}$ without observing y would be an un-supervised learning situation. Any number of possible decisions could then be made, including the traditional decision to put examples into clusters, or to predict $\ddot{y}$ (if exchangeability is not a problem), or to select the location of a cluster mean, etc. There are clearly many possibilities that would fall into this category. Each of these differences can be modeled by the choice of the outcome function $p(o|\mathscr{D}^Y_{ata}, \mathbf{d})$ and the utility function $U(o)$.

Whatever the choice for these two distributions, the expected utility of a vector of clustering decisions with n test examples, can be computed as follows:

$$E[U|\mathbf{d}]=\int_{\ddot{y}_1\in\mathbb{D}_Y}...\int_{\ddot{y}_n\in\mathbb{D}_Y}\int_{o\in\mathbb{D}_O}U(o)p(o|(\ddot{y}_1...\ddot{y}_n, d_1...d_n))p(\ddot{y}_1...\ddot{y}_n|\ddot{\mathbf{x}}_1...\ddot{\mathbf{x}}_n)d\ddot{y}_1...d\ddot{y}_n, do. \quad (2.15)$$

2.3.2 The UBDTM for Semi-Supervised Learning

In order to use the UBDTM for semi-supervised learning, we first make all of the simplifying assumptions we used for semi-supervised learning in the UBM, while adding the assumption the decision will consist of a vector of decisions, $\mathbf{d}$, one for each item in the test set. Further we assume that the outcome is a vector $\mathbf{o}$ made up of individual outcomes o_i (again one for each element in the test set), where each individual outcome depends only on the corresponding decision and the corresponding function output $\ddot{y}_i$ from the test set that we are trying to predict $p(o|d_i, \ddot{y}_i)$. We assume the utility is a sum of sub utility functions which are conditioned only upon their corresponding outcome $U_i(o_i)$. Thus, each classification task is assumed to be independent of the other decisions and the other items in the test set (see Figure 2.10) given the posterior predictive for the corresponding $\ddot{y}$ value. This last simplifying assumption allows us to deal with making classification decisions for each item in the test set independently of the others given the posterior predictive. It can readily be seen that this is a special case of the general UBDTM with some simplifying independence assumptions since both UBDTM Assumption 1 and 2 still hold, because the final summed utility depends only upon the outcome vector, which depends only upon the decision vector and the vector of $\mathscr{D}^Y_{test}$.

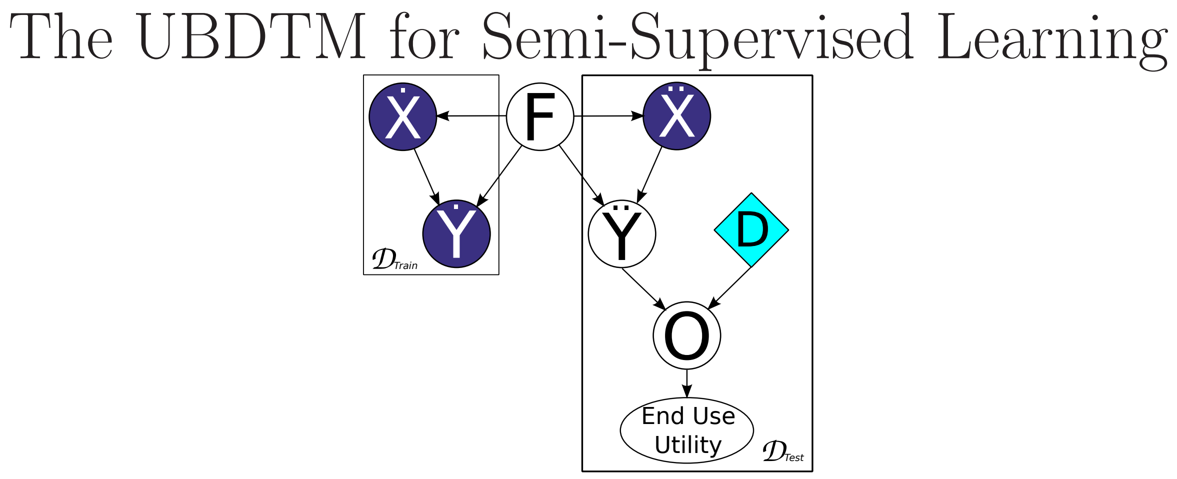


Figure 2.10: The Unified Bayesian Decision Theoretical Model (UBDTM), applied to semi-supervised learning.

The decision for semi-supervised learning is usually to predict the unknown function output for each item in the test set $\mathscr{D}_{test}$. However, this need not be the case. For example, if we are performing semi-supervised learning on mushroom toxicity, the decision might be to either eat or not eat the mushroom. However, in both cases, the outcome depends only on the decision and the actual function output $\ddot{y}$ from the test set that we are trying to predict.

The expected utility of an individual decision can therefore be computed as follows:

$$E[U|d_i, \mathscr{D}_{train}, \mathscr{D}_{test}] = \int_{\ddot{y}_i \in \mathbb{D}_Y} \int_{o \in \mathbb{D}_O} U(o) p(o|\ddot{y}_i, d) p(\ddot{y}_i|\ddot{\mathbf{x}}_i, \mathscr{D}_{train}, \mathscr{D}_{test}) d\ddot{y}_i, do, \tag{2.16}$$

and the total expected utility of classifying the entire test set can be found as follows:

$$E[U|\mathbf{d}, \mathscr{D}_{train}, \mathscr{D}_{test}] = \sum_i E[U|d_i, \mathscr{D}_{train}, \mathscr{D}_{test}]. \tag{2.17}$$

2.3.3 The UBDTM for Supervised Learning

To use the UBDTM for supervised learning, we make all the assumptions made above for semi-supervised learning, while adding the supervised learning assumption from the UBM, namely that the samples are conditionally independent of f. It can readily be seen that this is a special case of the general UBDTM with some simplifying independence assumptions (see Figure 2.11). As before, the total end use utility is the sum of the individual end use utilities from each classification.

The decision for supervised learning is usually to predict the unknown function output for each item in the test set $\mathscr{D}_{test}$. However, as with semi-supervised learning this need not be the case. For example, if we are performing supervised learning on mushroom toxicity, the decision might be to either eat or not eat the mushroom. However, either way, the outcome depends only on the decision and the actual function output $\ddot{y}$ from the test set that we are trying to predict.

The expected utility of an individual decision can therefore be computed as follows:

$$E[U|d_i, \mathscr{D}_{train}] = \int_{\ddot{y}_i \in \mathbb{D}_Y} \int_{o \in \mathbb{D}_O} U(o) p(o|\ddot{y}_i, d) p(\ddot{y}_i|\ddot{\mathbf{x}}_i, \mathscr{D}_{train}) d\ddot{y}_i, do, \tag{2.18}$$

and the total expected utility of classifying the entire test set can be found as follows:

$$E[U|\mathbf{d}, \mathscr{D}_{train}] = \sum_i E[U|d_i, \mathscr{D}_{train}]. \tag{2.19}$$

In the supervised learning literature, there is some confusion about the definition of a supervised learner. In some of the literature, supervised learning involves computing the posterior predictive, which is modeled by the UBM. But in other places, supervised learning involves end use decisions, and so is defined in a way that requires the UBDTM to model. For example, ANNs are considered to be supervised learners, but they do not return the posterior predictive, instead, they return a point estimate that can perhaps best be seen as a guess as to the most likely class, a decision that we shall show in Chapter 4 requires a model of end use. Thus it is unclear whether the term "supervised learning" refers to a problem that would be modeled by the UBM, or by the UBDTM.

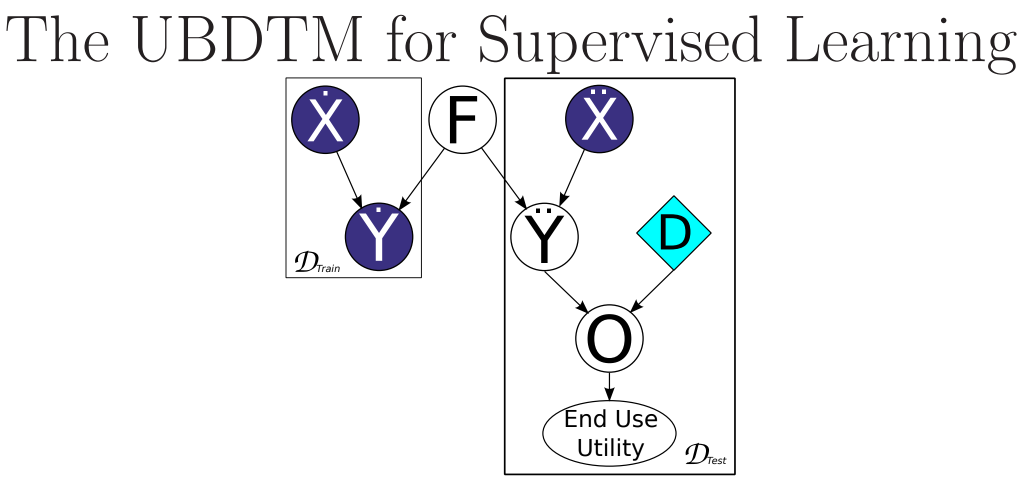


Figure 2.11: The Unified Bayesian Decision Theoretical Model (UBDTM), applied to supervised learning.

One advantage of our approach is that it allows us to be more formal about what we mean by the term "supervised learning." In our model of supervised learning, there are two parts, the inference portion of supervised learning, and the decision portion. We will use the term "supervised learning" or a "supervised learner" to refer to the combination of both problems, as modeled by the UBDTM. Formally:

Definition 2.3.3.1. Supervised Learning Algorithm: A supervised learning algorithm a takes data $\mathscr{D}_{train}$ and a model of end use (a model of the outcome o, how it depends on the classes and the decisions, and a utility function u) and then returns a mapping between function inputs and decisions.

Under the above definition, the UBDTM can be used to represent many different supervised learning algorithms, one for each prior, $\rho(f)$.

2.3.4 The UBDTM for Empirical Function Optimization

In *empirical* function optimization one must learn about the function from examples before information about the extremum can be inferred. Normally EFO examples are selected actively, but for now, we will analyze what happens given a set of examples however they

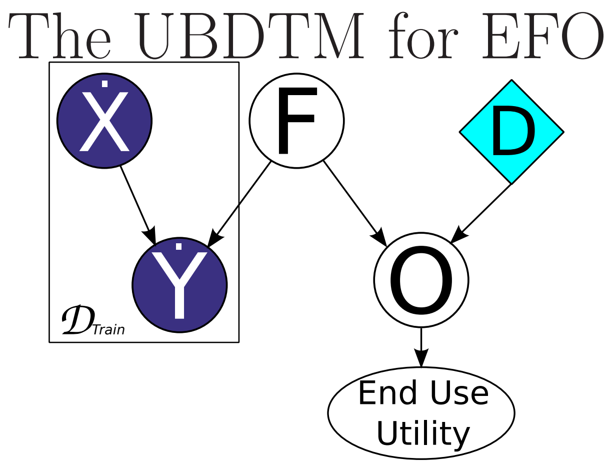


Figure 2.12: The Unified Bayesian Decision Theoretical Model (UBDTM), applied to empirical function optimization.

were attained, and deal with active learning issues in Chapters 6 and 10. The function can be learned from these examples using un-supervised, semi-supervised, or supervised learning approaches. For now, we will assume the most common supervised learning approach just as we did when applying the UBM to EFO. Thus, to use the UBDTM to model end use in EFO, we assume that elements of the training set impact the function in the same way that they did in the supervised learning version of the UBM (we make all of the UBM-EFO assumptions). We then assume that a single decision d is made which results in an outcome o which depends on the decision d and the function f (see Figure 2.12). Notice that we are not explicitly referring to the extremum, however, both the set of feature locations leading to an extremum $\mathbf{x}^*$, and the value of the extremum y^* are inherently present in the function f. Typically, the decision d will be to predict the location of the extremum $d = \mathbf{x}^*$, and the outcome will be the value at the extremum $p(o|d, f) = p(\ddot{y}|d = \ddot{\mathbf{x}}, f)$. This need not always be the case, and other variations could be imagined, for example, the decision might be to buy or sell stock in some company based upon the extremum of the price point of some commodity that the company manufactures. However, even in this case, the outcome depends upon only the function f (and by extension, the location and value of its extremum which is part of the function), and the decision d. Thus, EFO is a special case of the UBDTM with some additional simplifying assumptions. The expected utility of an EFO decision can then be found as follows:

$$E[U|d] = \int_{f \in \mathbb{D}_F} \int_{o \in \mathbb{D}_O} U(o) p(o|d, f) \rho(f|\mathscr{D}_{train}) df, do. \tag{2.20}$$

The optimal decision d should normally be selected by decision theory using the results of inference, and not selected directly by inference as has sometimes been done in the past. The optimal decision is just the decision that maximizes the expected utility from Equation 2.20, $d^* = \text{argmax}_d\, E[U|d]$. In a graphical model using decision theory, one usually does not infer decisions, rather one uses the principle of maximum expected utility given inference on the other nodes of the network in order to make the optimal decision. However, when the optimal decision is to simply select the location of the most likely extremum, then the decision theory equations simplify to the MAP estimate for $\mathbf{x}^*|\mathscr{D}_{ata}$. One common situation that can cause this result is when $U(o) = o$, and the domain of the outcome is the same as the domain of y, and $p(o|d, f) = p(y = o|\mathbf{x} = d, f)$, in other words, when likelihood of the outcome given a decision is equal to the likelihood of the function's output given the decision is to sample at a given $\mathbf{x}$ location. In this special case inference can directly yield the result we would normally use decision theory to determine. A similar result is produced when the extremum sought is a minimum by setting $U(o) = -o$.

2.4 Inductive Bias, Hypothesis Space, Function Class, and Priors in the UBM-UBDTM

There are several important conceptual and theoretical implications to our model of function learning. Some of the most important of these revolve around the concepts of inductive bias, hypothesis space, function class, and priors.

2.4.1 The Concepts of "Hypothesis Space" and "Function Class" in the UBM-UBDTM

The UBM and UBDTM have important implications for the concepts of hypothesis space and function class. The hypothesis space of a learning algorithm has traditionally been loosely defined as the *set* of functions that can be represented by a learning approach. Similarly, a function class has traditionally been loosely defined as the *set* of functions that we expect a learning algorithm to be used to solve (see Section 1.3.6). Thus, both of these concepts are very similar, in that it is usually a good idea if the set of functions that an algorithm is applied to be loosely similar to the set of functions that can be represented by that learning algorithm. Usually this concept is defined with crisp sets, meaning that a function is either in the hypothesis space or not, and is either in the function class or not. This is the way in which formalisms like VC dimensionality use the concept of a hypothesis space [115], and this is one reason why they produce worst case bounds on sample complexity [13]. On the other hand, Buntine showed that if the idea of a hypothesis space is expanded to a set of functions *and* a measure of the likelihood of the occurrence of each function in the set, (something that the UBM models as a probability distribution over functions $\rho(f)$), then it is possible to compute average case sample complexity [13] (for a further discussion of the relationship between sample complexity and function class see Section 11.3.6). For the remainder of this work, we will use the distributional definition of "function class."

In the UBM, the concepts of hypothesis space and function class are both handled by the prior $\rho(f)$. Because this hypothesis space and function class are represented by a probability distribution, they follow Buntine's approach, and if applied to sample complexity, should yield better results than the traditional approach to a hypothesis space. Further, one important advantage of our approach is that in the UBM and UBDTM the concepts of hypothesis space, function class, and Bayesian prior are all unified in a single distribution, and thus, by specifying a distribution over functions that we want an algorithm to perform well on (a function class), we have *automatically* generated an algorithm with a matching hypothesis space simply by using that function class as the prior in the UBM.

In the following sections we will now explore the connection between the function class and the concepts of inductive bias, the futility of bias free learning, and overview various ways in which $\rho(f)$ can be specified.

2.4.2 The UBDTM and the Theory of Inductive Bias

Inductive bias has been a difficult theoretical issue in machine learning. Perhaps the two most difficult questions regarding inductive bias have revolved around precisely defining the concept of an inductive bias, and determining whether bias free learning is possible. This section will formally define inductive bias, and discuss how this definitions of inductive bias impact the UBM and UBDTM.

There are two prevalent definitions of inductive Bias that are widely used in the Machine Learning community. The first defines inductive bias as follows:

Definition 2.4.2.1. An Inductive Bias is "a preference for one hypothesis *over* another."

This definition was used by Mitchell in his paper "The Futility of Bias Free Learning" [72], and was also used implicitly by Wolpert in his interpretation of the No-Free-Lunch Theorems [121]. This UBM specifies a preference for one function over another using the distribution $\rho(f)$. Thus, under this definition, the concept of an inductive bias is similar to, but not identical to, the concept of the prior, hypothesis space, and function class in the UBM. This is because any prior other than the uniform prior would adequately represent the inductive bias under Definition 2.4.2.1 while a uniform prior would be considered to be bias free under this definition because it provides no "a priori" mechanism for preferring one hypothesis *over* another. It is under this definition that both Wolpert and Mitchell were able to call algorithms that consider all functions to be equally likely "un-biased."

The second definition of inductive bias was also given by Mitchell [74, p. 43]. This definition of bias initially applied only to non probabilistic learners. His definition put into our notation is:

Definition 2.4.2.2. Consider a learning algorithm L over the instances $\mathbf{x} \in \mathbb{D}_X$ and the labels $y \in \mathbb{D}_Y$. Let $L(\mathbf{x}_i, \mathscr{D}_{train})$ denote the classification assigned to the instance $\mathbf{x}_i$ by algorithm L after training on the training set $\mathscr{D}_{train}$. The *inductive bias* of L is any minimal set of assertions B such that for any target y and corresponding training examples $\mathscr{D}_{train}$

$$(\forall \mathbf{x}_i \in \mathbb{D}_X)[(B \wedge \mathscr{D}_{train} \wedge input = \mathbf{x}_i) \vdash L(\mathbf{x}_i, \mathscr{D}_{train})]$$

where $a \vdash b$ means that b follows deductively from a.

Thus, in this second definition, Mitchell is proposing that the inductive bias is represented by all the additional information necessary to deductively produce the inductive algorithms outputs. Mitchell's second definition can be simply extended to encompass probabilistic learning algorithms by modifying his use of L such that it encompasses any output of a learning algorithm rather than just a discrete class. This would allow a learning algorithm to return either a discrete class, a distribution over class labels, a continuous function output, a decision, or even a function extremum. This simple extension of Definition 2.4.2.2 leads to our definition of inductive bias:

Definition 2.4.2.3. Consider a learning algorithm L over the instances $\mathbf{x} \in \mathbb{D}_X$ and the labels $y \in \mathbb{D}_Y$. Let $L(\mathbf{x}_i, \mathscr{D}_{train})$ be the *output* assigned to the instance $\mathbf{x}_i$ by algorithm L after training on the training set $\mathscr{D}_{train}$. The *inductive bias* of L is any minimal set of assertions B such that for any target y and corresponding training examples $\mathscr{D}_{train}$

$$(\forall \mathbf{x}_i \in \mathbb{D}_X)[(B \wedge \mathscr{D}_{train} \wedge input = \mathbf{x}_i) \vdash L(\mathbf{x}_i, \mathscr{D}_{train})]$$

where $a \vdash b$ means that b follows deductively from a.

In the remainder of this work, unless expressly stated otherwise, when we refer to inductive bias, we will be using this definition. There are many important ways in which the concepts of the UBM and UBDTM interact with this definition of inductive bias.

We have defined inductive bias as the additional assumptions that must be made in order to deduce the learner's results. Deduction involves many axioms, and by the term "additional" we mean, those assumptions that must be made beyond the classical axiomatic ones. Thus, the inductive bias would be those assumptions that a learner makes beyond those that are considered to be axiomatic to the deductive process. If we assume the standard axioms of mathematics such as algebra and measure theory, and if we additionally assume Kolmogorov's axioms of probability, then all of the laws of statistics (including Bayes Law) can be inferred by the principles of deduction [51]. Thus, statistical inductive inference involves some axioms, model and prior assumptions, and deduction. If the principles of mathematics, measure theory, and Kolmogorov's axioms are considered to be axiomatic to the universe and to our deductive process (and thus not directly part of the bias), then the bias in the UBM can be formally defined in terms of only its model and prior assumptions.

If the model assumptions of the UBM are also taken to be the axiomatic, and as simply the definitions of the function learning problems, and therefore also not directly part of the inductive bias, then the definition of inductive bias becomes even more specific. In order to perform inference in the UBM only a few distributions must be chosen. In the un-supervised and semi-supervised case, the user must specify $\xi(\mathbf{x}|f)$ and $\rho(f)$. In the supervised learning case they must only specify $\rho(f)$. Once these values are determined $\dot{\mathbf{x}}$ and $\ddot{\mathbf{x}}$ are both observed while $p(y|\mathbf{x}, f)$ is uniquely determined by f since f is a mapping from features to a distribution over function outputs. Furthermore, $\xi(\mathbf{x}^*|f)$ and $p(y^*|f)$ are uniquely determined by f and the choice of the extremum that we are looking for. Thus, in the supervised learning case, the only real free parameter that can be used to modify the behavior of the UBM, is the prior, $\rho(f)$. Therefore, given a prior, it is possible to use the principles of deductions and axiomatic assumptions to deduce the output of the UBM. Thus, for supervised learning under Definition 2.4.2.3, the inductive bias of a UBM based learning algorithm is identical to $\rho(f)$, which also already represents the hypothesis space, function class, and the prior. For semi-supervised or un-supervised learning, it is only necessary to add $\xi(\mathbf{x}|f)$ to the prior to specify the inductive bias. If we define learning as more than just producing the posterior predictive, but as actually making the decision found in the UBDTM model, then the inductive bias involves the UBM inductive bias plus the model of end use.

It can be difficult to formally define the inductive bias of traditional algorithms such as Decision Trees or Backpropagation Neural Networks. However, the UBM has the advantage of having a formal, explicit, and easily interpretable Bias when Bias is defined as in Definition 2.4.2.3. Thus, one of the principle advantages of the UBM is that the inductive bias of a UBM based supervised learning algorithm is made synonymous with the hypothesis space and function class over which the learning algorithm should be applied. If we are given a function class, then the necessary hypothesis space, bias, and corresponding learning algorithm are immediately provided by presenting that function class as the prior to the UBM, and conversely, if we are given a supervised learning algorithm based upon the UBM, then the function class over which that algorithm will perform well is explicitly defined by the the algorithm's prior in such a way that it can be easily interpreted, evaluated, and understood. This can make targeting the algorithm for the problem at hand easier.

There are several ways in which this definition differs from Definition 2.4.2.1. In Definition 2.4.2.1, the uniform distribution for $\rho(f)$ did not prefer one hypothesis *over* another, and there for the uniform prior was bias free. However, in Definition 2.4.2.3, the uniform prior would *not* be considered bias free because some prior distribution is still part of the necessary extra information that would allow deductive reasoning to then lead to a conclusion. Without this information that all functions are equally likely, it would not be possible to deduce the conclusion. For more information on this idea of bias free learning, and how these formalizations of bias affect these ideas, see Section 5.2.

An inductive bias can be representational (sometimes called restriction or language bias) or preferential [72]. A representational bias is caused by the representational power of the learning technique chosen and determines the representational limits of the algorithm (a traditional crisp hypothesis space). A preferential bias is caused by the technique's preference (or lack of preference) for one representable function over another (the measure of likelihood for each hypothesis in the crisp hypothesis space). The representational bias of an algorithm based upon the UBM will be determined by the support of the prior, while the relative likelihood of representable functions will represent the algorithm's preferential bias.

2.4.3 The Flexibility of the UBDTM, Setting the Prior

In the UBM, it is possible to select many different representations for $\rho(f)$. This allows the UBM to mimic the representational bias of a wide variety of classical learning algorithms.

Thus, the UBM is surprisingly versatile. Usually there will be two parts to any specification of $\rho(f)$: the representation chosen and the parameters for that representation. Let θ represent the parameters of f. For example, $\rho(f)$ could be distributed so as to have zero mass on all functions that are not composed of the sum of a set of weights, organized into an artificial neural network. The representation of f is now a neural network, and the parameters for this representation, θ, are the values for the network weights. To complete the specification of $\rho(f)$, a set of priors must be chosen for the weights. Then Equation 2.8 will determine the true posterior probability distribution over the set of network weights given some training data. Thus, by selecting $\rho(f)$ according to the structure of an ANN the graphical model provides an optimal technique for learning those weights. A similar approach to learning weights in an ANN was taken by Freitas et al. [24], and his work can be seen as a special case of the UBM, with the representational bias of an ANN.

In fact, the representational bias of almost any machine learning algorithm can be recast in terms of the UBM so long as it is possible to compute the likelihood of the data given the parameters of the model. The UBDTM has led to improvements for CMAC function approximation [15], Empirical Function Optimization [80], and natural language processing [14] [37]. The UBM has even been an inspiration for a novel technique for image super resolution [111]. A similar approach to the UBM has been taken with several learning models including neural networks [24] [87], and support vector machines [9]. In all of these cases this approach has led to improved performance over many test cases.

2.5 Utilities and Optimality in the UBM-UBDTM

Some measure of quality is necessary in order to assess the performance of a function learning algorithm. The UBDTM uses utility theory to represent end use. One advantage of this approach is that when correct Bayesian inference is used, the results will be "optimal" with respect to expected utility given the prior, meaning that the expected utility of the Bayesian approach will be as good as or better than any other approach to the extent that the prior is accurate. This is known as the Principle of Bayesian Optimality [25] (see Section 1.2.2), and it will have many significant implications for the remainder of this work. Thus, if the function learning task itself can be modeled as a Bayesian process (the UBM), then the optimal function learning algorithm would be to simply perform inference in the UBM, and any decision based upon those inference results using the UBDTM would be guaranteed to have an "optimal" expected utility. This would imply that there is a "best" function learning algorithm (see Chapter 5). Therefore, when developing a learning system we are no longer interested in selecting the best algorithm (as that has already been done), rather, the algorithm designer's task is to select the correct prior (and therefore the correct bias and function class). We will further explore these ideas and their implications for NFL theory in Chapter 3.

Notice that the computation of the posterior predictive for $\dot{y}$ and for $\mathbf{x}^*$ are conditionally independent of remainder of the model so long as O and U remain unobserved. This means that even after adding end use utility assumptions, inference on the posterior predictive remains independent of any end use utility assumptions, (see Theorem 4.2.0.1 in Chapter 4).

The degree to which the system is risk averse, risk seeking, or risk neutral is determined by the utility function chosen. Once a utility function is chosen, these equations allow us to quantify the risk of using the network's outputs for decision making. This is especially important for high assurance systems [15]. Furthermore, this value will change as more data is acquired and the learner becomes more confident in its outputs. We can use this to determine how many samples are needed for the given problem at hand (see Section 11.3.6).

2.5.1 Full Probabilities for Posteriors

Another major advantage of the UBM, is that it returns full distributions for posterior predictives, and it returns the "true" posterior predictives rather than an approximation or point estimate. As we shall show in Chapter 4, point estimates are not sufficient to produce decisions that maximize expected utility over all utility functions. Only when we return the full posterior can the algorithm's user make optimal decisions using this posterior for an arbitrary utility function. Since it is often the case that the eventual end use to which our algorithms will be put is not known when we are creating the algorithm, it makes sense to either use the UBM to create a distributional posterior, or to use an approximation that returns more than a point estimate. These issues will be explored more fully in Chapter 4.

2.6 Conclusions

The UBM and the UBDTM represent a model of the function learning problems. By modeling these learning scenarios generally, it is possible to make important, general, theoretical assertions that will hold in each of the function learning scenarios regardless of the details of what is being learned, or of other application specifics. This modeling approach to learning theory is similar to that taken by Wolpert with his EBF model of supervised learning, function optimization, and search. However, unlike the Wolpert's EBF [121], the UBM makes no assumptions about the utility function, or about the function being discrete or finite, or about the distribution of $p(f)$. Further, the UBM models both the discrete and continuous cases except that in the discrete case the above integrals become sums, and pmfs become pdfs. For a more detailed comparison between the EBF and the UBM see Section 3.4.

If you accept the axioms of statistics and the relationships between the random variables represented in Figure 2.1, then these equations are the only mathematically correct technique for un-supervised learning, semi-supervised learning, supervised learning and empirical function optimization. Given that we believe the above axioms, then all other techniques that work are either identical to these equations, approximate them in some way, or assume some different relationship between the random variables. The fact that these equations represent the mathematical underpinnings of these problems make them a powerful tool for analyzing the fundamental theory behind these learning problems.

In the sections that follow, we take these model assumptions to be axiomatic. For the remainder of this work we will focus primarily on supervised learning and Empirical Function Optimization with the supervised learning assumptions. Therefore, in what follows, we will assume that $\mathbf{x}$ is conditionally independent of f for both supervised learning and for EFO unless expressly stated otherwise. However, it is important to note that the concepts of the UBM and UBDTM are more broad, and we leave a more detailed exploration of the situations where $\mathbf{x}$ and f are not conditionally independent (un-supervised learning, and semi-supervised learning) for future work.

Chapter 3

The Supervised Bayesian No-Free-Lunch Theorems

Abstract

In this Chapter, we will explore the supervised No-Free-Lunch theorems from a Bayesian perspective. We present Wolpert's Extended Bayesian Formalism (EBF) as a *descriptive* graphical model, and then show how the UBM and the UBDTM can be interpreted as a *prescriptive* graphical model to answer some of the same questions. We will compare and contrast the EBF and the UBM, showing that although the UBM cannot model as large a space of learning algorithms, every algorithm representable by EBF and not representable by the UBM is sub-optimal for all function classes. Using the UBM and UBDTM we re-evaluate the No-Free-Lunch theorems, showing how learning *should* be done *both* on and off training set, and we present a new utility independent Supervised Bayesian No-Free-Lunch theorem, demonstrating how learning *should* be done off training set with a uniform prior.

Publication: The majority of the material in this chapter is published here for the first time, while some of the material was drawn from the following earlier publications: [16] [17].

3.1 Introduction

The traditional No-Free-Lunch (NFL) theorems are a set of mathematical theorems that asks the question: *how do learning algorithms compare off training set when all possible functions are equally likely.* An "off training set" example, $\ddot{\mathbf{x}}$, is one that does not appear in the training set, that is $\ddot{\mathbf{x}} \notin \mathscr{D}^{X}_{train}$. The traditional NFL theorems show that all algorithms perform the same off training set when all functions are equally likely given the 0,1 gain utility function as a measure of performance. This result has been highly influential in the machine learning community.

Wolpert proved the NFL theorems using a statistical formalism of the supervised learning problem known as the Extended Bayesian Formalism or EBF. The EBF is similar to the UBDTM in many ways, but has some significant differences. Most notably, the EBF attempts to model all possible learning approaches, even the sub-optimal ones. Thus, the EBF is primarily descriptive instead of prescriptive, in that it describes how learning does work in existing algorithms instead of prescribing how learning *should* ideally work.

This chapter will primarily deal with NFL issues related to supervised learning. We will expand these NFL concepts to empirical function optimization in Chapter 7, and to active learning for both supervised learning and empirical function optimization in Chapter 6. We will present our Bayesian approach to supervised learning NFL in the following manner: In Section 3.2, we will present the traditional EBF formalism, but we represent it as graphical model and modify the notation to coincide with that of the UBM. In Section 3.3, we will present the traditional NFL theorems. In Section 3.4, we will compare and contrast the EBF with the UBDTM, showing that the EBF can be seen as a descriptive model of all supervised learning approaches, while the UBDTM can be seen as a prescriptive model that attempts to show how supervised learning *should* be done. In Section 3.5, we will discuss how the UBDTM says that learning should be performed under the NFL assumptions. In Section 3.6, we will give the Bayesian NFL theorem for supervised learning, which will provide a utility independent interpretation of the NFL concept. In Section 3.7 we will conclude. Section 12.1 in the Appendix is also related to this chapter in that it provides some additional mathematical information about the Product of Dirichlet Distribution, which will be important to our Bayesian NFL approach.

3.2 The Extended Bayesian Formalism (EBF) As a Graphical Model

Wolpert used a statistical model of the supervised learning problem in order to prove his No-Free-Lunch theorems. He called this model the Extended Bayesian Formalism (EBF). The EBF has many things in common with the UBDTM, in that it is not an attempt to model a specific supervised learning scenario, but to model the supervised learning problem in general. The idea being that, inasmuch as the EBF models the general supervised learning problem, properties (such as NFL) that could be proved on the EBF would hold for supervised learning in general. For Wolpert's presentation of the EBF see [121], and for Wolpert's comparison between the EBF, PAC, VC dimensionality, and the Bayesian approach see [119].

Although Wolpert never represented the EBF graphically, Figure 3.1 is a graphical representation of this formalism. By representing the EBF graphically, the EBF can be more easily compared to the UBM and the UBDTM. To further facilitate this comparison, we have made minor notational changes to the EBF, renaming a few variables to cause its notation to align with that of the UBDTM. The ideas presented in this graphical model can be summarized in a few assumptions which we will call the EBF assumptions.

EBF Assumption 1: In the EBF, and unknown function f maps a vector of function inputs $\mathbf{x}$, to function outputs y or to a distribution over function outputs $p(y|\mathbf{x}, f)$.

EBF Assumption 2: In the EBF, a learner's hypothesis h is also a function that maps a vector of test set inputs $\ddot{\mathbf{x}}$, to function output predictions d or to a distribution over function output predictions $p(d|\ddot{\mathbf{x}}, h)$. In the EBF, these predictions are considered to be the outcome of the learning algorithm.

EBF Assumption 3: The sampling distribution of the test set, $\xi(\ddot{\mathbf{x}})$, is conditionally independent of f, h, and $\mathscr{D}_{train}$.

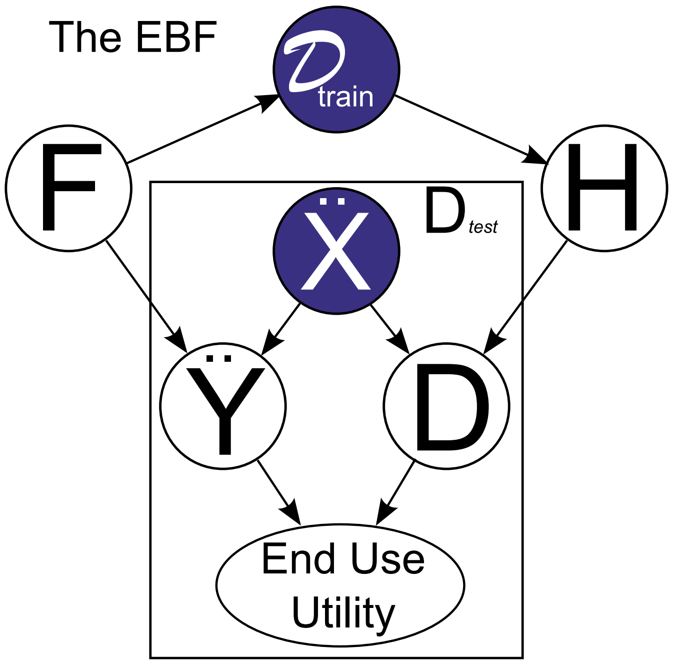


Figure 3.1: The Extended Bayesian Formalism (EBF) represented as a graphical model.

EBF Assumption 4: The probability of the training set depends upon the function, formally: $\nu(\mathscr{D}_{train}|f)$, while the probability of a hypothesis depends only on the training data, formally: $\rho(h|\mathscr{D}_{train})$.

EBF Assumption 5: For each feature vector in the test set, the learning algorithm will select a decision d. The EBF limits the space of the decision $\mathbb{D}_D$ to predictions of the output class Y, thus $d \in \mathbb{D}_Y$. This decision will be made by some decision strategy, which will depend only upon the features and the learner's hypothesis $p(d|\ddot{\mathbf{x}}, h)$.

EBF Assumption 6: The EBF models end use utility as a 0,1 loss function (sometimes called the misclassification error rate loss function). We have generally used positive utility functions rather than the equivalent, but more pessimistic, loss functions. We therefore represent the EBF's 0,1 loss function as the equivalent 0,1 gain utility function: $U(\ddot{y}, d) = \delta(\ddot{y}, d)$, where $\delta(\ddot{y}, d) = 1$ if $d = \ddot{y}$ and 0 if $d \neq \ddot{y}$. The expected utility $E[U|p(\ddot{y}), p(d)] = \sum_{\ddot{y} \in \mathbb{D}_Y} \sum_{d \in \mathbb{D}_Y} \delta(\ddot{y}, d)$.

As in the UBM, since f is a mapping between $\mathbf{x}$ and y, therefore $p(y|\mathbf{x}, f)$ is uniquely determined by each f. We will differentiate features in the training set as $\dot{\mathbf{x}}$, and features in the test set as $\ddot{\mathbf{x}}$. A feature in the test set can be considered "on training set" (meaning that the same feature vector also appears in the training set) and "off training set" (meaning that the same feature vector does not appear in the training set). The EBF has traditionally been used to evaluate only off training set learning scenarios, where all features in the test set are off training set; however, the EBF itself could be used to model on training set learning as well. Although the EBF is generally used to model off training set error, EBF still assume that both on and off training set examples are drawn from the same function F.

Note that because of EBF Assumption 3, the EBF does not model potential direct interactions between F and $\mathbf{X}$ (for both $\dot{\mathbf{X}}$ and $\ddot{\mathbf{X}}$). This means that in its current form the EBF only models supervised learning, and does not accurately portray interactions found in semi-supervised or un-supervised learning. Notationally: $\xi(\dot{\mathbf{x}})$ is the probability that $\dot{\mathbf{x}}$ is in the training set, while $\xi(\ddot{\mathbf{x}})$ is the probability that $\ddot{\mathbf{x}}$ will appear in the test set.

Under the six EBF assumptions, in the case of deterministic functions, the expected off training set utility given a function, hypothesis, and training set is:

$$E[U|f, h, \mathscr{D}_{train}] = \sum_{\ddot{\mathbf{x}} \notin \mathscr{D}^X_{train}} \xi(\ddot{\mathbf{x}}) U(f(\ddot{\mathbf{x}}), h(\ddot{\mathbf{x}})),$$

where $\mathscr{D}^X_{train}$ represents the set of feature values in the training set, $\dot{\mathbf{x}}$. In the case of non-deterministic functions, the expected off training set utility given a function, hypothesis, and training set is:

$$E[U|f, h, \mathscr{D}_{train}] = \sum_{\ddot{\mathbf{x}} \notin \mathscr{D}^X_{train}} \xi(\ddot{\mathbf{x}}) E[U|f(\ddot{\mathbf{x}}), h(\ddot{\mathbf{x}})],$$

which expands to:

$$E[U|f, h, \mathscr{D}_{train}] = \sum_{\ddot{\mathbf{x}} \notin \mathscr{D}^X_{train}} \sum_{\ddot{y} \in \mathbb{D}_Y} \sum_{d \in \mathbb{D}_Y} \xi(\ddot{\mathbf{x}}) p(\ddot{y}|\ddot{\mathbf{x}}, f) p(d|\ddot{\mathbf{x}}, h) U(\ddot{y}, d).$$

As we will see in the next section, the above equations will lead to the inner product rule, which will indicate that the expected off training set utility depends primarily upon how "aligned" the hypothesis is with the function.

3.3 Traditional No-Free-Lunch Theorems for Supervised Learning

We will now review several No-Free-Lunch related theorems that have been given by Wolpert and Mitchell, while placing them in our notation. The traditional NFL proofs all assume the above EBF assumptions together with three additional NFL assumptions. In Wolpert's publication, these assumptions were all considered to be part of the EBF, however, we have separated the EBF assumptions that model supervised learning in general (above) from the three EBF assumptions that specifically relate to the NFL learning situation (below). We have done this because we will eventually apply these three NFL assumptions to the UBDTM to create our Bayesian approach to NFL.

The traditional NFL theorems assume all the above EBF model assumptions together with the following three additional NFL assumptions:

NFL Assumption 1: The domain and range of F (and H if it is modeled separately from F) are finite and discrete, meaning that each f or h is either a mapping between a discrete $\mathbb{D}_{\mathbf{X}}$ and a discrete $\mathbb{D}_Y$, or between a discrete $\mathbb{D}_{\mathbf{X}}$ and a distribution $p(y|\mathbf{x}, f)$ over discrete $\mathbb{D}_Y$.

NFL Assumption 2: All functions are equally likely: $\forall f_1, f_2 \;\; \rho(f_1) = \rho(f_2)$.

NFL Assumption 3: The test set is off training set: $\ddot{\mathbf{x}} \notin \mathscr{D}^X_{train} \forall \ddot{\mathbf{x}} \in \mathscr{D}^X_{test}$.

Note that just because the domain and range of f are discrete, this does not mean that the parameters of the distribution $p(y|\mathbf{x}, f)$ need be discrete, just as a Multinomial distribution models a discrete set of classes with a continuous set of parameters.

The uniform prior over f is sometimes called the "*a priori*" case, or the "unbiased learning" case. However, this depends on the definition of inductive bias chosen. We will deal with this issue in greater detail in Section 5.2.

Since these theorems have been widely discussed elsewhere, we will give them here without proof or considerable comment.

Wolpert's formulation of the No-Free-Lunch theorems for supervised learning show that all supervised learning algorithms perform equally well off-training-set for the EBF's 0,1 gain utility function when uniformly averaged over all f. Wolpert's original publication of the NFL Theorem involved several theorems and conclusions, but we will focus on the following:

Theorem 3.3.0.1 (Inner Product Rule)**:** *Given the EBF model assumptions and the NFL assumptions, $E(U|\mathscr{D}_{train})$ can be written as a (non-Euclidean) inner product between the distributions $\rho(h|\mathscr{D}_{train})$ and $\rho(f|\mathscr{D}_{train})$:*

$$E[U|\mathscr{D}_{train}] = \sum_{h\in\mathbb{D}_H}\sum_{f\in\mathbb{D}_F}\rho(h|\mathscr{D}_{train})\rho(f|\mathscr{D}_{train})E[U|f,h,\mathscr{D}_{train}].$$

Proof. For the proof of this theorem, see Wolpert's publications: [121, 122]. □

The Inner Product Rule basically indicates that the performance of an algorithm depends primarily upon how "aligned" its hypothesis is to the true posterior function class. This is one reason not to model f and h differently, since h should be as much like f as possible.

Theorem 3.3.0.2 (The Traditional Supervised Learning No-Free-Lunch Theorem)**:** *Consider the off-training-set utility function. Let $E_i(.)$ indicate an expected value evaluated using learning algorithm i. Given the EBF and NFL assumptions, then for any two learning algorithms $P_1(h|\mathscr{D}_{train})$ and $P_2(h|\mathscr{D}_{train})$, independent of the sampling distribution, and uniformly averaged over all f, for any training set $\mathscr{D}_{train}$, $E_1(U|f,\mathscr{D}_{train}) - E_2(U|f,\mathscr{D}_{train}) = 0$.*

Proof. For the proof of this theorem, see Wolpert's publications: [121, 122]. □

This theorem means that under the EBF and NFL assumptions, all algorithms perform equally well with respect to the 0,1 gain utility function.

Although Wolpert never focused on this fact, an interesting corollary of Theorem 3.3.0.2, was given by Mitchell in 1980 [72], several years before Wolpert's publication on the supervised NFL theorems:

Corollary 3.3.0.3 (Uniform Hypotheses)**:** *If the hypothesis space contains every possible deterministic function mapping between discrete inputs and outputs, then off training set, for all training sets and $\forall\, \ddot{y}_1$ and $\ddot{y}_2 \in \mathbb{D}_Y$ there will be exactly as many hypotheses which are consistent with the training set that predict $\ddot{y}_1$ as there are that predict $\ddot{y}_2$.*

Proof. For the proof of this theorem, see Mitchell's publication [72]. □

Corollary 3.3.0.3 implies that if the hypothesis space is fully expressive, and if there is no reason to prefer one hypothesis over another (what Mitchell called the "un-biased" learning case) then all classes are equally likely [72][74, p. 42-45]. In terms of the EBF, this means that when the decision d is simply to select a class $\ddot{y}$, when all classes are equally likely with a 0,1 gain utility function, then the expected utility is the same for all possible decisions.

This corollary is closely related to Theorem 3.3.0.2 since the reason that all supervised learning algorithms perform equally well on average given the NFL assumptions is because there are as many consistent hypotheses that predict one class as there are that predict another off training set, and when all functions are equally likely, there is no reason to prefer one consistent hypothesis over another, therefore, all classes are equally likely. Because of the specific utility function assumed by the EBF, if all classes are equally likely then all decisions available to the supervised learning techniques will all have the same expected utility.

Example 3.3.1: Assume the EBF and NFL assumptions, and assume the task of correctly labeling mushrooms as either poisonous or edible. Assume that the decision, d, is to select a class of edible or poisonous given the distribution over $p(d|\ddot{\mathbf{x}}, h)$.

By Theorem 3.3.0.2, we know that the expected utility of all decisions d is the same off training set. This means that the same expected utility will be achieved for all decision strategies, including random and even the optimal one presented in Equation 2.13.

Since all decision strategies are the same, it does not matter whether the decision strategy $d = \phi(\ddot{\mathbf{x}}, h)$ is given a hypothesis that accurately maximizes $p(d|\ddot{\mathbf{x}}, \mathscr{D}_{train})$ or not. We could use the UBM as our technique for producing h, and we could use the UBDTM as our technique for producing d. In this case, the correct $p(\ddot{y}|\ddot{\mathbf{x}}, \mathscr{D}_{train})$ for this situation could be computed using Equations 2.9 and 2.8 and a uniform prior for $\rho(f)$ in the UBM, and then a decision could be made using (2.13). Although this technique is guaranteed to perform as well as or better than all other techniques by the principle of Bayesian optimality, in this case it will only perform *as well as*, and *no better than* any other approach because all decisions have the same expected utility.

In summary, for all off training set mushrooms, given the EBF and NFL assumptions, Theorem 3.3.0.2 proves that the classification of poisonous has the same expected utility as the classification of edible. Therefore, all decision techniques produce decisions with the same expected utility, and all supervised learning techniques produce decisions with the same expected utility. This will remain true, so long as we are dealing with off training set error, and with the 0,1 gain utility function.

This means that under the EBF and NFL assumptions, all supervised learning strategies will produce the same expected utility, including random, backpropagation, decision trees, perceptrons trained with gradient descent on the error function, and even perceptrons trained with gradient *ascent* on the error function.

The NFL property that all decisions have the same expected utility also holds for *some* utility functions other than 0,1 gain so long as they do not impose a "geometrical structure" over Y. "If the error function induces a geometrical structure over Y then we can have *a priori* distinctions between learning algorithms" [122]. However, in such cases, this existence of *a priori* distinctions between learning algorithms is imposed primarily by the end use utility function, since all classes are still equally likely off training set.

3.4 EBF and UBDTM, A Comparison

EBF Assumption 1 is the same as UBM Assumption 1, but the EBF adds the concept of a separate hypothesis h (EBF Assumption 2) that can differ from the function f. In the UBM the algorithm's hypothesis is fixed to be equal to f and therefore no real distinction is necessary, since any deviation between h and f results in sub-optimality. EBF Assumption 3 is analogous to the supervised learning assumption in the UBM. EBF Assumption 3 allows the posterior for h to be formulated in any conceivable way, while the UBM is more restrictive in that it forces the learner's hypothesis to follow Bayes law for some prior. EBF Assumption 5 and 6 allow the EBF to deal with end use, however, EBF's model of end use is far more restrictive than was the UBDTM's, since it restricts the decision to predicting the class and restricts the utility function to the 0,1 gain utility function.

The primary difference between the EBF and UBM is the manner in which they each represent a learning algorithm. Learning algorithms are represented in the UBM as choices for the function class $\rho(f)$, with the posteriors and posterior predictives then dictated by the definition of f and the laws of statistics. In the EBF, a machine learning algorithm is represented by a set of distributional choices for $\rho(h|\mathscr{D}_{train})$ for all training sets. This distribution represents the probability of generating a hypothesis after observing some training data. Thus, an EBF learner is free to select any posterior and to behave in completely different ways for each potential $\mathscr{D}_{train}$. This is how the EBF can represent all possible learning approaches, even the sub-optimal ones. More formally, we can define the concept of a learning algorithm being EBF representable as follows:

Definition 3.4.0.1. A supervised learning algorithm is EBF representable if, for all possible hypothesis h, and for all potential training sets $\mathscr{D}_{train}$, the probability that the algorithm will produce a hypothesis h given some training data can be written as $\rho(h|\mathscr{D}_{train})$.

Then, we can define the concept of a learning algorithm a being UBM representable as follows:

Definition 3.4.0.2. A supervised learning algorithm $\rho(h|\mathscr{D}_{train})$, is UBM representable if, there exists a prior $\rho(f)$ such that for all possible hypothesis h, and for all potential training sets $\mathscr{D}_{train}$, $\rho(f|\mathscr{D}_{train}) = \frac{\nu(\mathscr{D}_{train}|f)\rho(f)}{\nu(\mathscr{D}_{train})} = \rho(h|\mathscr{D}_{train})$.

There are some parts of the EBF that resemble the UBM. In the EBF, the posterior $\rho(f|\mathscr{D}_{train})$ could be computed according to Bayes Law with some prior $\rho(f)$, and with the likelihood forced by the nature of the mapping inherent in f. If this approach is taken, then the EBF will compute f in the same way as the UBM. However, although it is possible to use the EBF to compute the posterior for f, this is not the EBF's representation of a learning algorithm, and classification decisions are not based optimally on f, but potentially sub-optimally upon h.

Since a hypothesis is a mapping between function inputs and function outputs, $\nu(\mathscr{D}_{train}|h)$ should depend only upon $\mathscr{D}^X_{train}$ and the mapping inherent in h just as f does. Thus, the likelihood of the data given the mapping should also be the same for f and h when their mappings overlap. This likelihood of the Data, together with a prior $p(h)$, *should* dictate $\rho(h|\mathscr{D}_{train})$ according to Bayes Law, and if it did, then the posterior of $h|\mathscr{D}_{train}$ would be the same as the posterior for $f|\mathscr{D}_{train}$ when the priors for f and h are the same. However, in the EBF, no such restrictions are placed upon h, and $\rho(h|\mathscr{D}_{train})$ can represent *any* technique for generating a hypothesis from data. This concept that h should be the same as f in order to generate the optimal posterior is one of the observations that initially lead to the development of the UBM.

The EBF is primarily descriptive, in that it attempts to describe how learning techniques, which EBF define as $\rho(h|\mathscr{D}_{train})$, behave given the function class where all functions are equally likely (a uniform $\rho(f)$) and with a specific utility function. The UBM is primarily prescriptive, in that it attempts to determine how learning *should* behave given a function class $\rho(f)$ regardless of the utility function. This means that the UBM has $\rho(h|\mathscr{D}_{train})$ fixed to the optimal $\rho(f|\mathscr{D}_{train})$ in other words, the UBM assumes that the unknown function is part of a function class $\rho(f)$, and then uses statistical inference and the fact that $p(y|\mathbf{x}, f)$ is fixed by the fact that f is a mapping between function inputs and outputs in order to infer the posterior for the unknown function using Bayes Law.

It has been suggested that it is rarely possible to compute $\rho(f|\mathbf{x}, y)$ in practice, since $\rho(f)$ is rarely known, and that this is one reason to prefer the EBF over approaches like the UBM [122]. However, from a Bayesian perspective this need for a prior is at the very heart of the need for a bias in supervised learning and function optimization. Each choice for $\rho(h|\mathscr{D}_{train})$ that generalizes well does so only for a certain sub-class of functions, and $\rho(f)$ is the mathematical representation of this function class. As we shall see in Section 5.2, the presence of the prior in Bayes Law is the mathematical realization of the need for an inductive bias in supervised learning. Every effective learning algorithm $\rho(h|\mathscr{D}_{train})$ that could be chosen has some implicit assumptions about $\rho(f)$, the UBM simply makes these assumptions explicit instead of implicit. The fact that there are many so-called "black box" learning algorithms that tend to perform well in a wide variety of practical situations simply means that there is a function class $\rho(f)$ that will perform well over a wide variety of practical situations, and it is just as easy (if not easier) to specify one of these "black box" function classes as it is to specify a "black box" algorithm with some implicit, unknown function class.

It is worth noting that the number of possible supervised learning algorithms that are EBF representable is larger than the number of supervised learning algorithms that are UBM representable. Both H ad F represent the possible mappings between discrete function inputs and function outputs, meaning that distributions over H and F have the same cardinality. However, set of UBM representable functions is no larger that the space of all possible priors $\rho(f)$ because the UBM posterior is fixed by Bayes law. The space of UBM priors has the same cardinality as one EBF posterior $\rho(h|\mathscr{D}_{train})$. Yet the EBF can generate one posterior for *each* data set independently, not being fixed by Bayes law. All this means is that there are many ways of generating a hypothesis given data other than the one that maps to the Bayes Optimal solution.

A simple example may help to illustrate this idea. Assume two priors $\rho_1(f)$ and $\rho_2(f)$ and assume that they are not the same. Now, it is possible to define an EBF representable algorithm that for $\mathscr{D}_{train\,1}$ computes the posterior $\rho(h|\mathscr{D}_{train\,1}) = \frac{\nu(\mathscr{D}_{train\,1}|f)\rho(f)_1}{\nu(\mathscr{D}_{train\,1})}$ and for all other data sets computes $\rho(h|\mathscr{D}_{train} \neq \mathscr{D}_{train\,1}) = \frac{\nu(\mathscr{D}_{train}|f)\rho(f)_2}{\nu(\mathscr{D}_{train})}$. This is clearly EBF representable, but there is no single prior that would create this behavior in the UBM. This means that there are some learning inference algorithms that can be represented in the EBF, that can not be represented in the UBM. However, such functions are sub-optimal for all function classes:

Theorem 3.4.0.4 (The Sub-optimality of non UBM Representable Supervised Learning Inference Algorithms)**:** *Any EBF representable learning algorithm $\rho(h|\mathscr{D}_{train})$ that is not UBM representable is an algorithm that is sub-optimal for all possible function classes $\rho(f)$ with respect to expected utility for all possible utility functions.*

Proof. The proof is by contradiction. First, assume that the theorem is false. Then there must exists an EBF representable supervised learning algorithm $\rho(h|\mathscr{D}_{train})$ which is optimal for some function class $\rho(f)$, but which is also not UBM representable. Yet, if the function class $\rho(f)$ is used as the prior for the UBM, then the UBM will produce the optimal posterior for that function class $\rho(f|\mathscr{D}_{train})$ for all training sets. Yet, we know that the optimal posterior for all training sets is $\rho(h|\mathscr{D}_{train})$. There is only one posterior that will lead to optimal decisions over the space of all possible utility functions, and this means that $\rho(f|\mathscr{D}_{train}) = \rho(h|\mathscr{D}_{train})$ for all training sets when $f = h$ (the mapping in f is the same as the mapping in h). Thus the algorithm is UBM representable by using the prior $\rho(f)$ which is a contradiction. □

Since the space of learning algorithms is larger in the EBF than in the UBM, it follows that there are some algorithms that the EBF can represent that the UBM can not represent. However, by Theorem 3.4.0.4, those extra functions representable by the EBF must be those functions that are not optimal for any function class with respect to expected utility over all possible utility functions. If they were optimal for some function class, that function class could be used as the prior for the UBM, making the algorithm UBM representable.

The fact that the UBM can not directly represent those algorithms that are sub-optimal for all function classes does not mean that the UBM has nothing useful to say concerning those algorithms. Such sub-optimal approaches can often serve as useful heuristics which approximate the optimal solution when the optimal solution is too difficult to represent or when it is too computationally intense to compute. We do not imply that such heuristics do not have a place in machine learning, rather we are implying that they should be thought of as approximations to the mathematically correct solutions implied by the UBM-UBDTM. If such approximations are useful, it will be because they approximate the results of the UBM-UBDTM on some function class. This means that they UBM-UBDTM can be used to both define the correct solution and to explain why some heuristics to the correct solution work. For example, we will show in Chapter 8 that the traditional update rule for the CMAC ANN is actually an iterative maximum likelihood approximation for the mean of the weights dictated by the UBM.

3.5 A Bayesian Perspective on Learning Under NFL Assumptions 1 and 2

The traditional NFL results came by making the all the EBF and the three NFL assumptions. Instead, in this section, we will make the supervised UBM, and supervised UBDTM assumptions from Chapter 2, and the first two NFL assumptions from Section 3.3 (the discrete domain and range for f, and the uniform $\rho(f)$ assumptions). We will then evaluate the theoretical results that follow. This will allow us to make some general statements about how learning *should* be performed both on and off training set under these assumptions. These observations will then lead to the Bayesian NFL Theorems which we will give in the next section by adding the third NFL assumption (the off training set assumption).

There are two cases we must consider, the simplest is the deterministic function case, and the more complex non-deterministic function case.

Theorem 3.5.0.5 (The Uniform Prior Deterministic Bayesian Supervised Learning Theorem)**:** *Given the UBM supervised learning assumptions, and the NFL assumptions 1 and 2, if the function is deterministic, then:*

Case 1, the example $\mathbf{x}$ *is on training set: then the posterior predictive* $p(y_j|\mathbf{x}_i, \mathscr{D}_{train}) = 1$ *if* $y_j|\mathbf{x}_i$ *is consistent with the training set, and* $p(y_j|\mathbf{x}_i, \mathscr{D}_{train}) = 0$ *if* $y_j|\mathbf{x}_i$ *is inconsistent with the training set.*

Case 2, the example $\mathbf{x}$ *is off training set: then* $p(y_j|\mathbf{x}_i, \mathscr{D}_{train}) = \frac{1}{|Y|}$.

Proof. The proof for the deterministic case is already in place and can be found in the definition of a deterministic function for on training set locations, and in Corollary 3.3.0.3 for the off training set locations which shows that there are as many consistent functions that classify one way as that classify another. Because all these functions were equally likely, the posterior must be uniform off training set. □

The non-deterministic case is substantially more complex:

Theorem 3.5.0.6 (The Uniform Prior Non-Deterministic Bayesian Supervised Learning Theorem)**:** *Given the UBM supervised learning assumptions, and the NFL assumptions 1 and 2, if the function is non-deterministic, then the posterior is distributed as a Product of Dirichlet distribution, and the posterior predictive is distributed as a set of independent Multinomial distributions, one for each sample location with parameters* $\alpha_{i,j} = 1 + c_{i,j}$ *where* $c_{i,j}$ *is the number of times the* $\mathbf{x}_i, y_j$ *pair appear in the data set. Thus:* $p(y_j|\mathbf{x}_i, \mathscr{D}_{train}) = \frac{\alpha_{i,j}}{\sum_j \alpha_{i,j}}$.

Proof. $\mathbf{X}$ and Y are finite and discrete according to NFL Assumption 1. When F was a deterministic mapping from $\mathbf{X}$ to Y, it was possible to compute the probability of each f as $P(f) = 1/|F|$. However, in the non-deterministic case, F is a mapping from $\mathbf{x}$ to $p(y|\mathbf{x}, f)$, and there are an infinite number of possible $f \in \mathbb{D}_F$. Given the infinite domain for F, the likelihood of a given f must be some constant, $\mathscr{C}$, then $\forall f, f' \;\; \rho(f) = \rho(f') = \mathscr{C}$.

Non-deterministic supervised learning functions can be seen as a matrix, mapping a total of m discrete input feature vectors, $\mathbf{x}$, to a distribution over a total of n discrete function outputs, y.

$$f = \begin{bmatrix} p(y_1|\mathbf{x}_1) & p(y_2|\mathbf{x}_1) & \cdots & p(y_n|\mathbf{x}_1) \\ p(y_1|\mathbf{x}_2) & p(y_2|\mathbf{x}_2) & \cdots & p(y_n|\mathbf{x}_2) \\ \vdots & \vdots & & \vdots \\ p(y_1|\mathbf{x}_m) & p(y_2|\mathbf{x}_m) & \cdots & p(y_n|\mathbf{x}_m) \end{bmatrix}$$

For a given $\mathbf{x}_i$:

$$\mathscr{F}_i = \begin{bmatrix} p(y_1|\mathbf{x}_i) & p(y_2|\mathbf{x}_i) & \cdots & p(y_n|\mathbf{x}_i) \end{bmatrix}$$

Where $\mathscr{F}_i$ is constrained to the unit simplex. In other words, $\sum_j p(y_j|\mathbf{x}_i) = 1$ for all i. This would mean that the likelihood of a set of function outputs, $\mathscr{D}^Y_{train}$, all sampled from a single sample location x_i is a Multinomial distribution with the parameters $\mathscr{F}_{i,1}...\mathscr{F}_{i,n}$ where $p(y_j|\mathbf{x}_i) = \mathscr{F}_{i,j}$ and

$$\nu(\mathscr{D}^Y_{train}|x_i, f) = \frac{|\mathscr{D}^Y_{train}|!}{\prod_{j=1}^n c_{i,j}!} \prod_{j=1}^n \mathscr{F}_{i,j}^{c_{i,j}},$$

where $c_{i,j}$ is the number of times the $\mathbf{x}_i, y_j$ pair appear in the data set. And the likelihood of a set S of samples is:

$$\nu(S|f) = \prod_{i=1}^m \frac{|S_{x=i}|!}{\prod_{j=1}^n c_{i,j}!} \prod_{i=1}^m \prod_{j=1}^n \mathscr{F}_{i,j}^{c_{i,j}}$$

Then $\rho(f|\mathscr{D}_{train})$ can be computed as:

$$\rho(f|\mathscr{D}_{train}) \propto \nu(\mathscr{D}_{train}|f)\rho(f),$$

and since $\rho(f) = \mathscr{C}$,

$$\rho(f|\mathscr{D}_{train}) \propto \nu(\mathscr{D}_{train}|f)\mathscr{C}$$

Regardless of how this uniform prior $\mathscr{C}$ was represented, the posterior can be shown to be distributed as a product of Dirichlets with posterior parameter matrix α such that $\alpha_{i,j} = c_{i,j} + 1$.

$$\alpha = \begin{bmatrix} c_{1,1}+1 & c_{1,2}+1 & c_{1,n}+1 \\ c_{2,1}+1 & c_{2,2}+1 & c_{2,n}+1 \\ \vdots & \vdots & & \vdots \\ c_{m,1}+1 & c_{m,2}+1 & c_{m,n}+1 \end{bmatrix}$$

Notice that it was possible to show that the likelihood $\rho(f|\mathscr{D}_{train})$ was distributed as a product of Dirichlets using only the UBM assumptions for supervised learning, the discrete $\mathbf{X}$ and Y assumption, and the uniform $\rho(f) = \mathscr{C}$ assumption. For more information about this distribution and its properties, see Appendix 12.1. Because $\nu(S|f)$ is distributed as a product of multinomials, and because the product of multinomials is the conjugate prior for the product of Dirichlets, the posterior $\rho(f|\mathscr{D}_{train} + \mathscr{D}_{ata})$ will also be distributed as a product of Dirichlets for any data set $\mathscr{D}_{ata}$. It is also possible to work backwards, and discover that one way to represent the prior that caused $\rho(f) = \mathscr{C}$ was with a product of Dirichlets with $\alpha_{i,j} = 1$, and that therefore $\mathscr{C} = \prod_{i=1}^{m} \frac{\Gamma(\sum_{j=1}^{n} \alpha_{i,j})}{\prod_{j=1}^{n} \Gamma(\alpha_{i,j})} = (\Gamma(n))^m$.

Given the above observations, the posterior predictive distribution is then:

$$p(y_j|\mathbf{x}_i \mathscr{D}_{train}) = \int p(y_j|\mathbf{x}_i, \mathscr{F}_{i,j})\rho(\mathscr{F}_{i,j}|\alpha_{i,j})d\mathscr{F}_{i,j}$$

and since $p(y_j|\mathbf{x}_i, \mathscr{F}_{i,j}) = \mathscr{F}_{i,j}$,

$$p(y_j|\mathbf{x}_i \mathscr{D}_{train}) = \int \mathscr{F}_{i,j}\ \rho(\mathscr{F}_{i,j}|\alpha_{i,j})d\mathscr{F}_{i,j} = E[\mathscr{F}_{i,j}].$$

However $\mathscr{F}_{i,j}$ is distributed as the product of Dirichlets, and one of the properties of the product of Dirichlets is that the expected value can be found as follows:

$$p(y_j|\mathbf{x}_i, \mathscr{D}_{train}) = E[\mathscr{F}_{i,j}] = \frac{\alpha_{i,j}}{\sum_j \alpha_{i,j}}$$

$\square$

The above equations will dictate how learning *should* take place both on and off training set for non-deterministic functions under the first two NFL assumptions. Using these facts, we are now prepared to present the Supervised Bayesian NFL theorems by evaluating how this learner behaves off training set.

3.6 The Supervised Bayesian No-Free-Lunch Theorem

In this section we will use the UBM to determine how a supervised learner *should* behave given the three NFL assumptions. Of special note is that we are no longer making the specific 0,1 utility function assumption made by the EBF, rather, we are interested in developing a set of results that will hold for all utility functions. The result is what we call the Bayesian Supervised Learning NFL Theorem:

Theorem 3.6.0.7 (The Bayesian Supervised Learning NFL Theorem)**:** *If the supervised learning UBM assumptions and the three NFL assumptions hold then for an off-training set example,* $\ddot{\mathrm{x}}$*, the posterior predictive,* $\ddot{y}|\ddot{\mathrm{x}}, \mathscr{D}_{train}$*, is uniform. Formally,* $p(\ddot{y}|\ddot{\mathrm{x}}, \mathscr{D}_{train}) = \frac{1}{|Y|}$ *where* $|Y|$ *is the number of possible classes.*

Proof. If f is deterministic, then the result has already been shown in Theorem 3.5.0.5 case 2.

If f is non-deterministic, then as we showed in Theorem 3.5.0.6, when $\forall f, \rho(f) = \mathscr{C}$, then $\rho(f|\mathscr{D}_{train})$ must be distributed as a product of Dirichlets. Since $\ddot{\mathbf{x}}$ is off training set $\alpha_{\ddot{\mathbf{x}}} = \begin{bmatrix} 1, & 1, & \cdots, & 1 \end{bmatrix}$. The posterior predictive can be found as follows: $p(\ddot{y}_j|\ddot{\mathbf{x}}_i, \mathscr{D}_{train}) = E[\mathscr{F}_{i,j}] = \frac{\alpha_{i,j}}{\sum_j \alpha_{i,j}}$. Given that $\alpha_{i,j} = 1$ off training set, and $\sum_{j=1}^{n} \alpha_{i,j} = |Y|$ therefore $p(\ddot{y}|\ddot{\mathbf{x}}, \mathscr{D}_{train}) = \frac{1}{|Y|}$. $\square$

There are several complementary routes to prove this theorem, and each one is useful for developing intuition as to the implications of this theorem. We will therefore also present another route to the same conclusion, which is to use the results of the traditional NFL theorem and work backwards:

Under the EBF assumptions, with the 0,1 gain utility function all decisions $d|x', \mathscr{D}_{train}$ produce the same expected utility by Theorem 3.3.0.2. Given this deterministic utility function of the EBF, $E[U|d] = p(\ddot{Y} = d|\ddot{\mathbf{x}}, \mathscr{D}_{train})$. All decisions can only have the same expected utility off training set for this utility function iff $p(\ddot{y}|\ddot{\mathbf{x}}, \mathscr{D}_{train}) = 1/|Y|$ off training set. In both the EBF and the UBM, $p(\ddot{y}|\ddot{\mathbf{x}}, \mathscr{D}_{train})$ can be computed independently of the end use utility function because in both networks the $\ddot{y}$ node is conditionally independent of the utility or outcome nodes when they are unobserved (compare Figures 2.7 and 3.1). This means that in both the EBF and UBM, if the $\ddot{y}$ node is uniform for one utility function, it must be uniform for all utility functions. Thus, it follows that in the EBF, $\ddot{y}$ is uniform off training set for all utility functions, not just the 0,1 gain utility functions. There is enough similarity between the EBF and the UBM, that if $\ddot{y}$ is uniform off training set in the EBF, it must also be uniform off training set in the UBM since in the EBF, when the training set and $\ddot{\mathbf{x}}$ are observed and the utility is unobserved, then $\ddot{y}$ is independent of all other nodes in the network but f, and the influence between $\ddot{y}$, $\ddot{\mathbf{x}}$, f, and $\mathscr{D}_{train}$ are the same in both networks. This means that the relevant portions of the EBF network are identical to the relevant portions of the supervised UBM network. The nodes that differ between the EBF and the UBM all do not play a part in inference in this particular case. Therefore, $p(\ddot{y}_j|\ddot{\mathbf{x}}_i, \mathscr{D}_{train})$ is also uniform off training set in the UBM.

One common way to reach the NFL results used in the past has been by an appeal to symmetry. This approach notes that, off training set, there are as many functions consistent with the training data that predict one class as there are that predict another class (see Corollary 3.3.0.3). When all functions are equally likely, then there is no reason to prefer one of these consistent hypotheses over another. This approach is easy for deterministic functions, but can be rather tricky for stochastic functions. It is a measure of the success of the Bayesian approach that it allows a much simpler path to the proof.

NFL has traditionally been interpreted in terms of what Theorem 3.3.0.2 says about uniform expected utilities for decisions given a specific set of utility functions. However, we believe that it is the new, utility independent, property expressed in Theorem 3.6.0.7 that is most significant. Intuitively this theorem says that uniform priors lead to uniform posterior predictives off training set regardless of the utility function involved.

3.6.1 The Connection Between, No-Free-Lunch, Bias, Overfit, and Uncertainty

As we will see in Section 11.3.2, one possible approach to reducing overfit is to enrich the hypothesis space in proportion to the amount of training data available. This prevents the learner from selecting overly complex hypotheses with insufficient data to justify such a conclusion. Such an approach approximates the more abstract concept embodied by Occam's razor, but it has been difficult to determine how to expand the hypothesis space in a principled way. The Bayesian approach to creating this bias is to produce a prior where complex functions have less likelihood than simple functions. Then the principles of Bayes law will determine when enough data has been seen to justify believing the more complex hypotheses. This would provide a principled technique for determining how much data is needed before more complex functions can be believed. Equation 2.9 integrates over each function in the function space, taking the uncertainty about the correct hypothesis into account when computing the posterior predictive. The more possible functions that are integrated over, the more uncertainty is added to the probability of $\ddot{y}$. In one sense, this can be seen as trading overfit for uncertainty (see Section 11.3.2). This concept of adding uncertainty to the posterior predictive as more functions are integrated over provides one way of understanding the Bayesian NFL result.

When trading overfit for uncertainty, the obvious question is, "how many hypotheses should I average over?" The theoretical answer (computational considerations aside) would be to average over all of them. This should return the best result because it would account for *all* of the uncertainty in the posterior predictive that comes from uncertainty about the correct hypothesis. Some early approaches to hypothesis expansion only expanded the representational hypothesis space, assuming that all representable hypotheses were equally likely [119][38] [10][39]. However, we know that the theory demands that we should integrate over all hypotheses, and we are now in a position to ask questions such as, "How uncertain should the posterior predictive be off training set when we average over all possible functions when they are weighted uniformly?" The answer comes from Theorem 3.6.0.7.

Thus, one interpretation of Theorem 3.6.0.7 would be to see it in terms of what Bayesian No-Free-Lunch has to say about trading overfit for uncertainty by answering the questions "how uncertain should an algorithm be once it takes all hypotheses into account equally?" The Bayesian No-Free-Lunch theorems answer this question as follows: once all hypotheses are taken into account equally, then maximum uncertainty is produced for the posterior predictive off training set (the uniform distribution is the maximum entropy distribution, thus the distribution of "maximum uncertainty"). This can only be avoided by taking a preferential bias into account as well as a representational bias. This would allow us to integrate over all possible functions as demanded by the math, without producing a uniform posterior predictive because of the non-uniform preferential bias in the prior.

The concepts of bias, overfit, NFL, ensembles, sample complexity, and meta learning are all connected with this idea of integrating over functions to produce the posterior predictive as demanded by Equation 2.9. We will discuss these ideas in greater detail in Sections 11.3.2-11.3.6.

3.7 Conclusions

In this Chapter, we have represented the EBF as a graphical model, and compared the EBF to the UBM and UBDTM, showing that it is possible to see the EBF as a descriptive model of supervised learning, and the UBDTM as a prescriptive model of supervised learning. We have reformulated the tradition NFL result in our notation, and re-presented the traditional Supervised NFL theorems. We then used the principles of the UBM to produce a Bayesian perspective for learning under the NFL assumptions, and showed that if the function is non-deterministic and if $p(f)$ is uniform, then the posterior is distributed according to a Product of Dirichlet Distribution. This provides the “optimal” supervised learning algorithm under the uniform prior assumption. This allowed us to introduce a Bayesian NFL result, namely that uniform priors lead to uniform posterior predictives off training set, and that this is the optimal behavior of a supervised learner in the uniform prior *a priori* case.

This idea of an “optimal” supervised learner under the NFL assumptions at first seems to be at variance with the NFL concept that all algorithms perform equally well under the NFL assumptions. In actuality, these concepts are not at variance, since the “optimal” approach does perform no better than other techniques off training set given the specific 0,1 utility function. However, as we will show in Chapter 5, it can perform better than other learning approaches for other utility functions, or over on training set error rates, and it will never perform worse than any other approach for any utility function.

Chapter 4

End Use Utility Theory in the UBDTM and the No-Free-Dinner Theorems

Abstract

There are two parts to the supervised learning problem: inference (modeled by the UBM) and decision making (modeled by the UBDTM). In this chapter, we will explore the theoretical implications of end use utility on decision making using the UBDTM for supervised learning. We explore which parts of supervised learning depends on utility, and which parts are utility free. We show that making decisions always involves some sort of end use utility assumptions.

Publication: Some of the material in this chapter has been previously published in [17].

4.1 Introduction

One effect of NFL has been to force practitioners to focus on the function classes for which their algorithms will be applied. The reasoning goes that since for the 0,1 gain utility function, if one algorithm generalizes better than another over some set of functions, it does so by generalizing worse over another set of functions. Thus, when creating or selecting a supervised learning algorithm, it is important to think carefully about the function class for which it will be used.[1]

The purpose of this chapter will be to encourage similar reflections with regard to utility. There are two ways in which utility can impact supervised learning, through sample costs and through end use utility assumptions. If we assume that a training set is already given, then supervised learning is independent of sample costs, and the only way in which utility effects supervised learning is through end use utility assumptions. In this chapter, we will assume that the training set is given, and thus, that utility only impacts supervised learning through end use utility assumptions.[2]

There are two parts to the supervised learning problem, inference and decision making. Inference is the portion of supervised learning that determines some information about the posterior predictive, either by returning the full posterior predictive, a statistic of the posterior predictive, or some approximation of the above (the process modeled by the UBM). Decision making is the process of using end use utility assumptions to make a decision (as modeled by the UBDTM). Decision making is just as important a part of supervised learning as inference. There is no intrinsic value in knowing the posterior predictive. Producing the posterior predictive is only useful inasmuch as this information serve to aid decision making.

This means that when a supervised learning practitioner selects a supervised learner, they must determine not only the function class for which that learner will be used but also the end use utility function over which it will be applied.

[1]We will carefully analyze and formalize these concepts in Chapter 5, since they are often misunderstood.

[2]We will deal with the situation where the training set is not given and therefore where supervised learning is not independent of sampling costs in Chapter 6.

In Section 4.2 we will show that the inference portion of supervised learning given a training set is independent of utility assumptions. In Section 4.3, we will show that if non-sufficient statistics on the posterior predictive are reported, then the algorithm must perform sub-optimally for some utility assumptions. In Section 4.3.1, we will discuss the shortcomings of one common approach to making decisions using supervised learning, namely classifying the appropriate decision given the situation. We will show that this simply pushes the utility assumptions down into the training data, where they are harder to deal with. In Section 4.3.2, we will show that if non-sufficient statistics are returned, it is still possible to make optimal decisions for some but not all end use assumptions. In Section 4.4, we will create a companion theorem to NFL that analyzes what happens when all possible utility functions are equally likely. We will show that it is still possible to create an optimal learning algorithm before end use utilities are known, so long as those utilities will be known at the time that the algorithm is used. We will show that if the decision must be made before the utility function is known, and if all utility functions are equally likely, then all algorithms have the same expected utility.

Some of the theorems/observations in this chapter are novel, however, we will also be including some known theorems/observations since our goal is to provide a coherent set of theory that explains the interaction between supervised learning and utility. In these cases the known theorem will be re-cast in terms of the Bayesian decision theoretical model. This will lead to new ways of looking at existing ideas. We believe that there is value in restating even some of these known ideas since many machine learning researchers have ignored or misinterpreted this information. Thus, presenting these theorems within a unified framework will be a valuable contribution to the theory and general understanding of how utility affects supervised learning.

4.2 Utility Free Supervised Inference

If the training set is given, then supervised learning will be independent of the utility that results from sample costs, which will only play a role in acquiring the training data. Thus, if utility affects supervised with a known training set, then the utility must come from end use.

The problem of supervised learning has alternatively been defined as the problem of producing the posterior predictive given training data and some features (the inference portion of supervised learning), or alternatively as the problem of producing the "best" function output, or classification, or decision given training data and some features (the decision portion of supervised learning). In this section we will show that the inference portion of supervised learning given a training set is independent of end use utility.

Theorem 4.2.0.1 (Utility Free Supervised Learning Inference)**:** *If the UBDTM assumptions hold and a training set is given, then $p(\ddot{y}|\ddot{\mathbf{x}}, \mathscr{D}_{train})$ can be computed without regard to utility from end use or sample cost so long as the decision, outcome, and end use utility remain unobserved.*

Proof. Recall Equation 2.9 in the UBM:

$$p(\ddot{y}|\ddot{\mathbf{x}}, \mathscr{D}_{train}) = \int p(\ddot{y}|\ddot{\mathbf{x}}, f)\rho(f|\mathscr{D}_{train})df \; . \tag{4.1}$$

Notice that the posterior predictive depends only on the likelihood, $p(\ddot{y}|\ddot{\mathbf{x}}, f)$ and on the prior $\rho(f|\mathscr{D}_{train})$. The utility function, and the rest of the model of end use play no role. This is the case because Y is independent of the model for D, O, and U so long as D, O, and U are unobserved. $\square$

If either D, O, or U are observed, then the end use utility function could potentially have an impact on the posterior predictive because information can flow backwards from these nodes back to Y using the principles of Bayes law to reverse the arrows in the graphical model of Figure 2.7.

Example 4.2.1: Assume that Google is creating a new spam filter for gmail. Computing the probability that a message is spam given the features of the message and some training data is independent of end use utility assumptions, while deciding whether to place the message in the inbox or spam filter is not independent of end use.

Example 4.2.2: A hiker is lost in the wilderness, and is beginning to get hungry. Luckily, our hiker brought his trusty PDA with him, and his PDA has a classifier trained on the UCI mushroom data set to determine if mushrooms are poisonous or edible. The hiker finds a wild mushroom and enters the height, weight and color into his PDA. Notice that the space of the classifier's outputs does not necessarily match the space of the hiker's decisions. In this case, the classifier's outputs have to do with the probability that the mushroom is poisonous or edible, while the hiker's decisions have to do with using that information to decide whether or not to eat the mushroom.

The inference part of supervised learning would now simply predict the probability that the mushroom was poisonous using Equation 2.9 and return the full posterior predictive without regard to how hungry the hiker was, the likelihood that he will starve if he does not eat the mushroom, or to the possibility that the hiker may have spent the last five years building up an immunity to poisonous mushrooms. All of these considerations would affect the end use utility function associated with this task. Therefore this part of the supervised learning problem is independent of end use utility assumptions. However, for the hiker to make use of this information, he must eventually approximate the probability of an outcome given the end use decision of eating or not eating the mushroom, and he must supply his utility function for these outcomes. Only then is it possible to use Equation 2.13 to determine if he should eat the mushroom. The posterior predictive was useless without these additional end use assumptions. So, although end use utility assumptions were delayed by returning the full posterior predictive, they were not removed, and it was useless to discuss "learning" the mushroom data set without eventual reference to end use.

The decision of whether or not to eat a mushroom would certainly not be independent of the end use utility function, but the posterior predictive probability that the mushroom is poisonous is independent of end use. Traditionally, this has not always been the case. Learners would often "hedge" their bets and cautiously report that a mushroom was poisonous when they were unsure just to play it safe, in effect balancing precision with recall. However, the proper manner in which learners should hedge their bets is naturally not independent of end use. By reporting the full posterior predictive instead of hedging in any way, it is possible to delay end use decisions until the utility function is known, since reporting the full posterior predictive is independent of end use by Theorem 4.2.0.1.

This theorem indicates that, when training a supervised learner, which reports full posterior predictives, the sample cost and end use are irrelevant. What is necessary is a training set and a prior over f. Naturally, generating the training set could involve sample cost considerations, and we will deal with that in greater detail in the Sections 6.7 and 11.3.6. Unfortunately, issues such as precision and recall are often inappropriately merged with inference in supervised learning. Such approaches create supervised learners that are specific to the utility considerations inherent in their precision and recall behavior, whereas supervised learners that report full posterior predictives are independent of end use. However, end use assumptions must eventually be made, so this independence of end use is simply a matter of delaying the eventual utility decisions.

4.3 End Use and Non-Sufficient Statistics on the Posterior Predictive

Supervised learning has often been defined as the problem of selecting the best class or function output given some features and training data. In this section we will show that if this is the definition of supervised learning, then supervised learning depends on end use utility.

It is not always practical to compute the full posterior predictive. Often this value must be approximated for computational complexity reasons. Sometimes this approximation involves a point estimate which is often a non-sufficient statistic of the posterior predictive. In this section we show that when this happens, an end use utility assumption has been made. We will do this by showing that whenever a non-sufficient statistic is reported, that there exists a utility function for which the reported statistic is not sufficient for making an optimal decision. This means that by choosing to report the non-sufficient statistic, the user has made a utility assumption, if only that the utility is not the one that leads to the sub-optimal result. In the next section we will show that some non-sufficient statistics can still result in optimality, but not for all utility functions, and only for some specific utility assumptions. Together these two observations mean that every time a non-sufficient statistic is returned by the inference portion of a supervised learner, some end use utility assumptions have been made.

First, we will define what is meant by the terms "sufficient statistic," and "non-sufficient statistic."

Definition 4.3.0.1. A statistic $T(Y)$ is a **sufficient statistic** for parameter θ of the distribution g if $\forall y, g(Y = y|T(Y) = t, \theta) = g(Y = y|T(Y) = t)$, or in shorthand, $\forall y, g(y|t, \theta) = g(y|t)$.

Definition 4.3.0.2. A statistic $T(Y)$ is a **non-sufficient statistic** for parameter θ of distribution g if $\exists y, g(Y = y|T(Y) = t, \theta) \neq g(Y = y|T(Y) = t)$, or in shorthand, $\exists y, g(y|t, \theta) \neq g(y|t)$.

Typically, sufficient statistics are usually given of data, but in the most general sense, these ideas can also be applied to a pdf or pmf. When applied to the posterior predictive, we first define the parameters of the posterior predictive to be θ such that $p(y|\theta) = p(y|\mathbf{x}, \mathscr{D}_{test})$. Then we can define a statistic t for parameter θ in the posterior predictive to be sufficient for the parameter θ if $\forall y, p(y|t) = p(y|\theta)$. Similarly, can define a statistic for the parameter in the posterior predictive to be non-sufficient if $\exists y, p(y|t) \neq p(y|x, \mathscr{D}_{test})$. A simple example can clarify. If we know that the posterior predictive is distributed normally, then the mean and variance together would for a sufficient statistic, and an inference algorithm that returned both would be considered an inference algorithm that reported sufficient statistics, while an inference algorithm that just reported the mean would be considered an inference algorithm that reported a non-sufficient statistic of the posterior predictive.

With the above definitions in mind, we are now prepared to discuss the effect of having a supervised learning inference algorithm return non-sufficient statistics of the posterior predictive.

Theorem 4.3.0.2 (Non-sufficient Statistics of the Posterior Predictive Assume End Use)**:** *If the UBDTM assumptions hold, and if the inference portion of a supervised learner reports a non-sufficient statistic, t, for the posterior predictive given a specific $\mathbf{x}$ and $\mathscr{D}_{test}$ then there exists a utility function U^s for which the expected utility of making decisions based on only the information in t will be sub-optimal.*

Proof. Let t, be a non-sufficient statistic for $p(\ddot{y}|\ddot{\mathbf{x}}, \mathscr{D}_{train})$. By the definition of a non-sufficient statistic there must exist a non empty set of function outputs, w_1, where $\ddot{y} \in w_1$ if $p(\ddot{y}|\ddot{\mathbf{x}}, \mathscr{D}_{train}) \leq p(\ddot{y}|t)$ and the set of all other function outputs w_2 where $p(\ddot{y}|\ddot{\mathbf{x}}, \mathscr{D}_{train}) > p(\ddot{y}|t)$. Thus, w_1 is the set of all function outputs for which the non-sufficient statistic predicts a larger or equal probability or likelihood, while w_2 is the set of function outputs for which the non-sufficient statistic predicts a lesser probability or likelihood. It is now possible to find the ϵ such that $p(\ddot{y} \in w_1|\ddot{\mathbf{x}}, \mathscr{D}_{train}) = p(\ddot{y} \in w_1|t) - \epsilon$, and since distributions sum to 1, and it is guaranteed that $\ddot{y}$ is either in w_1 or w_2, $p(\ddot{y} \in w_2|\ddot{\mathbf{x}}, \mathscr{D}_{train}) = p(\ddot{y} \in w_2|t) + \epsilon$.

Let U^s be a deterministic utility function with two decisions d_1 and d_2. Let E represent the expectation using the full posterior predictive, and $E^{\hat{s}}$ represent the expectation taken using the non-sufficient statistic. We choose to to construct this utility function such that $E(U^s|d_1) \neq E(U^s|d_2)$, $E(U^s|d_1) = E^{\hat{s}}(U^s|d_2)$ and $E(U^s|d_2) = E^{\hat{s}}(U^s|d_1)$. If this is the case, then the decision based upon the non-sufficient statistic would make the opposite choice from a decision based on the posterior predictive, and since the expected utilities of these choices are not equal, it would receive a different true expected utility. Since we know by the principles of Bayesian optimality that the decision based on the full posterior predictive is optimal with respect to expected utility, the decision based upon the non-sufficient statistic must be sub-optimal. There are many possible utility functions that match the above criteria, but we only have to show that one such utility function exists. One solution to this system of equations is to construct the utility such that for decision d, the utility function is $U^s(w_1, d) = p(\ddot{y} \in w_1|\ddot{\mathbf{x}}, \mathscr{D}_{test})$ if $d = d_1$ and $3p(\ddot{y} \in w_1|\ddot{\mathbf{x}}, \mathscr{D}_{test}) - \epsilon - 2$ if $d = d_2$; and $U^s(w_2, d) = 1 - p(\ddot{y} \in w_2|\ddot{\mathbf{x}}, \mathscr{D}_{test})$ if $d = d_1$ and $-3p(\ddot{y} \in w_2|\ddot{\mathbf{x}}, \mathscr{D}_{test}) - \epsilon + 2$ if $d = d_2$.

For the above utility function, the expected utility of the first decision $E(U^s|d_1) = 2p(y' \in w_1|\ddot{\mathbf{x}}, \mathscr{D}_{test}) - 1$, and the expected utility for the second decision $E(U^s|d_2) = 2p(\ddot{y} \in w_1|\ddot{\mathbf{x}}, \mathscr{D}_{test}) - 1 - \epsilon$. This means that decision 1 has a higher expected utility than decision 2, and is the best decision. However, when the non-sufficient statistic is used to make the decision, it incorrectly appears that the expected utilities are exactly backwards, $E^{\hat{s}}(U^s|d_1) = 2p(\ddot{y} \in w_1|\ddot{\mathbf{x}}, \mathscr{D}_{test}) - 1 - \epsilon$ $E^{\hat{s}}(U^s|d_2) = 2p(\ddot{y} \in w_1|\ddot{\mathbf{x}}, \mathscr{D}_{test}) - 1$. This means that an agent basing its decisions on the non-sufficient statistic would select decision 2 instead of decision 1, and would perform sub-optimally with respect to expected utility. □

This theorem means that whenever a non-sufficient statistic is returned of the posterior predictive, that some assumption must have been made about the end use utility function. At least, this assumption would be that the utility function is not the one constructed by the above proof. However, there are many such utility functions that could be constructed, and the utility function of the proof is just one such possible function.

Notice that in order to effectively use a classification technique to make general decisions, that technique must provide a full distribution over $p(\ddot{y}|\ddot{\mathbf{x}}, \mathscr{D}_{train})$ (see Equation 2.18). Techniques that simply report the most probable class are not useful for making optimal decisions for all potential utility functions, since they give no indication of how certain the learner is in its result. Even in the deterministic case where f maps a feature vector $\mathbf{x}$ to a single deterministic y, uncertainty about f should still lead to important uncertainty about the output $\ddot{y}$.

Table 4.1: Example utility for a classification task. All utilities scaled between -1 and 1.

D	Y'	O	U
Eat	Poisonous	Sick & Full	-.95
Eat	Edible	Not Sick & Full	.5
Not Eat	Poisonous	Not Sick & Hungry	-.1
Not Eat	Edible	Not Sick & Hungry	-.1

Example 4.3.1: Google is constructing a spam filter, and would like to return something more intuitive to their end users than simply the full posterior predictive (in this case, the simple probability that an email is spam). Although the full posterior predictive is the most generally applicable information, most users do not want to sort through email with probabilistic spam tags. Instead, Google wants to simply label the emails as "spam" or "not-spam." However, some of their users are more concerned about potentially missing an important message than they are about reading a few spam emails, while other of their users are less concerned missing a few emails so long as they are not bothered by spam, and there may even be a few users who like getting spam (surprisingly the author actually knows of one such person, although I assume that they are relatively rare). This is the classic precision vs. recall tradeoff.

One potential solution is to return the most likely label. Yet this is a non-sufficient statistic of the posterior predictive, so Theorem 4.3.0.2 tells us that such an approach will be sub-optimal for the utility of at least some of their users.

Example 4.3.2: Let us return to our example of a hiker lost in the wilderness and his potentially poisonous mushrooms. Assume that the classifier is a traditional ANN, and reports a non-sufficient statistic, namely a point estimate on whether the mushroom is poisonous or edible.

In this case, assume that the classifier reports that the mushroom is edible. Unfortunately, the hiker has no indication of how certain the learner was in its predictions because this was part of the information that was lost in the choice of the non-sufficient statistic. The classifier's point estimate can be viewed as a non-sufficient statistic of the full posterior predictive, and therefore, the classifier must have made some utility assumptions by Theorem 4.3.0.2.

One possible utility function is illustrated in Table 4.1. The decision $\hat{d}$ that maximizes the expected utility (the Maximum Expected Utility decision or MEU) is found by Equation 2.18.

If the algorithm returned the full distribution and indicated that there is a 51% chance that the mushroom is edible, and a 49% chance that it is poisonous, then given the above utilities, the expected utility for eating the mushroom is approximately -.21 while the expected utility for not eating the mushroom is -.1. Thus the optimal decision is to not eat the mushroom despite the fact that the Maximum Likelihood estimate (and thus any classifier based on it) determined that the mushroom is most likely edible. Traditionally, issues such as this are solved by tuning the algorithm itself to generate the desired precision and recall for its intended use. If the learner "hedged" its bets to match some desired ROC behavior, (for example, by only reporting "edible" if the the classifier is 95% certain that the mushroom is edible) then the right amount of "hedging" depends on some hidden implicit utility assumptions, with the 0-1 gain utility assumption resulting in simply reporting the most likely class.

Unfortunately, our hiker has no idea if the classifier's implicit hidden utilities are the same as his. In fact, since he might potentially starve to death if he does not eat something, they are almost certainly not the same as his. What the hiker really needs to know in order to optimally make this decision is: how hungry is he, how likely is he to be found, or to find another source of food, how sick will the mushroom make him if it is poisonous (these things together form his utility function), and how sure the classifier is in its classification of edible (the full posterior predictive or a sufficient statistic thereof). If the algorithm actually returned the entire probability distribution over possible outcomes, (which it can do independent of utility assumptions by Theorem 4.2.0.1 then this distribution could be used to select an optimal decision for any utility function desired by Equation 2.18.

4.3.1 The Fallacy of Classifying Decisions

One possible solution the the problem in the above examples would be if the classifier was trained to predict the decision d directly rather than to predict the class y. Classifying decisions is one way in which supervised learning practitioners have attempted to avoid end use utility assumptions. In fact, one way to view all of artificial intelligence is as a type of classification problem, where we simply need to predict the right behavior (a classification output) given the state (the features). Such an approach would provide one alternative to the UBDTM, since it would produce a different model of how decisions should be made. In this alternate model, decisions would be directly inferred from data rather than determined using decision theory given inferences about the world.

However, as we will see in this section, in general this is not a good approach since usually users are not actually interested in discovering the probability that a decision is the best decision, but are interested in making the decision that will maximize their expected utility. Furthermore, such an approach obfuscates the role of utility in making decisions by hiding the utility in the training examples.

The following example is an illustration of why predicting decisions using supervised learning instead of predicting classes and using decision theory to produce decisions is not a good solution in general:

Example 4.3.3: In the case of the lost hiker attempting to decide whether to eat a mushroom, suppose that a classifier existed that had been trained to classify whether a reasonable person would eat the mushroom rather than to classify whether the mushroom was edible or poisonous. Now the utility function still exists, but it was pushed into the training examples themselves, since the training examples are now based on the decisions of "reasonable" people with *their* utility functions. At first glance it now appears that the output of the classifier: "eat," or "don't eat," is more likely to be useful for the hiker. However, this result is a non-sufficient statistic, even on the posterior predictive for the decision d, so by Theorem 4.3.0.2, it is also making some implicit end use utility assumptions by not returning the full posterior predictive for d. Even if the full posterior predictive for d was returned, this posterior was learned given training examples of decisions made by "reasonable people" each with a specific set of potentially differen end use utility assumptions, which are now implicitly hidden in the training examples.

Unfortunately, as before, the hiker's utility function is likely not the same as those "reasonable" people's hidden utility assumptions, since he may be facing starvation if he does not eat something soon, while they most likely were not. In order to adapt the classifier to his situation, it would now be necessary to completely retrain the classifier with samples drawn from people with a utility function identical to our hiker's (meaning with individuals who were in his exact situation, lost for the same amount of time, equally hungry, with an equal chance of being rescued). The chance of this training set existing is very low. On the other hand, if the classifier reported a full posterior predictive, then our hiker could adapt the results of the classifier to his particular situation.

We can see from the above example that training classifiers to classify decisions does not actually remove the need for utility, it simply pushes the utility into the training data where it is implicit instead of explicit, and where it can be much more difficult to deal with and reason about, and where it can no longer be adapted to other situations.

4.3.2 Optimal Decisions With Non-Sufficient Statistical Summaries Given End Use

Theorem 4.3.0.2 showed that reporting a non-sufficient statistic for the posterior predictive resulted in poor performance for some end use utility functions, thus reporting a non-sufficient statistic is equivalent to some end use utility assumptions (at least the assumption that the end use is *not* one of those end use utility functions constructed in the proof). Now, we will analyze the inverse concept, namely that given a specific end use utility function, it is possible to compute a non-sufficient statistic that will maintain optimality. In fact, the maximum expected utility decision itself *is* a non-sufficient statistic of the posterior predictive, but if it is computed correctly, it will be optimal for the utility function that was used to construct the maximum expected utility decision. Again, this is important since many approaches must report non-sufficient statistics or other approximations of the posterior predictive due to computational complexity issues, but when they do so, they are making end use utility assumptions, and it can be important to know exactly what they are.

Theorem 4.3.2.1 (Optimal Decisions With Non-Sufficient Statistics Given End Use)**:** *If the supervised learning UBDTM assumptions hold, then non-sufficient statistics of the posterior predictive exist for which it is possible to make optimal decisions using that non-sufficient statistic instead of the full posterior predictive for* some *but not all end uses.*

Further, for all non-sufficient statistics for the posterior predictive that could be reported, there exists a utility function for which it is still possible to make optimal decisions using that non-sufficient statistic instead of the full posterior predictive.

Proof. In the degenerate case, it is possible to construct a utility function for which all decisions are optimal with respect to expected utility. Thus whatever statistic is reported, optimality is achieved for this utility function, thereby making the proof trivial.

However, in a more useful case, the decision itself when made by Equation 2.13 is a function of the posterior predictive and is therefore a statistic for the posterior predictive. And this statistic is not a sufficient statistic for arbitrary posterior predictives, since it does not contain enough information to reconstruct an arbitrary posterior predictive. However, given the utility function used in Equation 2.13, it will be a statistic that gives enough information to produce optimal behavior with respect to expected utility for any posterior predictives (even the ones for which it was not a sufficient statistic) for at least the utility function used in Equation 2.13. ☐

Equation 2.11, Equation 2.13, Theorem 4.3.0.2, and Theorem 4.3.2.1 together say that a sufficient statistic is a sufficient and a necessary condition for the decision to be made optimally when end use is not known, and that a sufficient statistic is sufficient but not a necessary condition for the decision to be made optimally when the utility is known. Whenever a non-sufficient statistic is returned by a supervised learning inference algorithm, optimality may still be possible for some but not all utility functions.

Corollary 4.3.2.2 (Optimal Decisions With Non-Sufficient Statistical Summaries Given a Distribution Over End Uses)**:** *If the supervised learning UBDTM assumptions hold, then non-sufficient statistics of the posterior predictive exist for which it is possible to make optimal decisions using that non-sufficient statistic instead of the full posterior predictive for* some *distributions over potential end uses.*

Further, for all non-sufficient statistics for the posterior predictive that could be reported, there exists a distribution over end uses for which it is still possible to make optimal decisions using that non-sufficient statistic instead of the full posterior predictive.

Proof. Again, it is possible to construct a utility function for which all decisions are optimal with respect to expected utility (the degenerate case). This makes the proof trivial.

However, in a more useful case, if the end use utility function is unknown, but a distribution over possible utility functions exists $p(f_u)$, then the optimal decision can be computed by expecting over the unknown utility function:

$$E[U|d] = \int_{f_u \in \mathbb{D}_U} \int_{\ddot{y} \in \mathbb{D}_Y} \int_{o \in \mathbb{D}_O} f_u(o) p(o|\ddot{y}, d) p(\ddot{y}|\ddot{\mathbf{x}}, \mathscr{D}_{train}) p(f_u) do\, d\ddot{y} df_u. \tag{4.2}$$

Similar approaches can be taken when other elements of end use (decision space, outcome space, outcome distribution etc) are unknown, so long as a distribution over possible end uses exists. These decisions are statistics for the posterior predictive, yet they are often non-sufficient. Such statistics may not be optimal for every utility function in the distribution over end uses, but will still produce the optimal expected utility over the distribution of end uses. □

The important observation here is that the decision itself can be seen as a summary statistics of the posterior predictive [7]. For most distributions a decision will be a non-sufficient statistic for the posterior predictive. Yet these decisions will still be optimal for a specific utility function if they were made according to Equation 2.13 using that specific utility function. This means that optimality can be maintained when reporting some non-sufficient statistics, however, we know from Theorem 4.3.0.2 that this statistic will not produce enough information to produce optimal behavior for *all* utility functions because there will be at least one (and likely many more than one) utility function for which it will be sub-optimal. Therefore, it is possible to maintain optimality when reporting a non-sufficient statistic for specific utility functions, but not for arbitrary utility functions. Thus, the move from a sufficient to a non-sufficient statistic implies some utility assumptions. For example, it is well known that reporting the MAP estimate can be justified by the 0,1 gain utility function, and the mean can sometimes be justified by the mean squared error utility function, while the mode can sometimes be justified by the absolute error utility function. Unfortunately, in supervised learning practice where sufficient statistics for the posterior predictive are rarely reported, these utility assumptions are often implicit and hidden, and this can make it unclear when a given learning technique would be applicable.

Example 4.3.4: It is clear that Google's spam filter should not just report the posterior predictive for each email to its users, it must make a decision (a non-sufficient statistic for the posterior predictive). Yet, we have seen that it is impossible to make that decision independent of some utility assumptions. Yet, we have also seen that each of their users do not necessarily have the same utility function for missing emails vs. reading more spam.

One solution would be to give each user a "permissiveness" setting on the filter. Without knowing exactly what they were doing, the users would in effect be selecting their utility function. Given the permissiveness setting, it would be possible to determine a utility function that matches that level of permissiveness in the filter, and then to make optimal decisions given this utility function.

Another solution to this problem comes from Corollary 4.3.2.2. This Corollary indicates that it would be possible to make the optimal decision over a distribution of possible end uses. Thus, if it were possible to create a distribution over user's potential end use preferences, then it would be possible to make a decision that would be optimal on average over these users. This seems to be the implicit design choice of most spam filter designers, who simply try to produce behavior that is correct for their average user, without explicitly going through the exercise of explicitly determining the actual distribution over potential end user utilities.

Example 4.3.5: If the utility function for classifying mushrooms is the 0,1 gain function, then reporting the MAP estimate on the posterior predictive (in this case, the most likely classification), is sufficient to produce optimal decisions because the decision to choose the most likely class will produce the optimal expected utility under this utility function. Other non-sufficient statistics besides the decision itself can maintain optimality for some utility functions. For example, if the problem had a continuous domain, and the goal was to approximate the mushroom's toxicity level on some continuous scale, then for unimodel posterior predictives reporting the mode would be optimal for some utility functions based on absolute error, while reporting the mean would be optimal for some utility functions based on the squared error.

The implication of the theorems in this section is that any non-sufficient statistic involves some utility assumptions. At first glance it may appear possible to construct a decision rule that is "irrational" in the sense that it does not correspond to any utility assumption. However, all decision strategies satisfy Equation 2.13 for the degenerate set of utility functions, that is, the set of utility functions for which all actions have the same expected utility. For these degenerate utility functions, all decision strategies perform equally well and are "optimal." Simply by applying different techniques for breaking the ties in the argmax, in Equation 2.13, it is possible to use that equation to produce any desired decision strategy all of which will be optimal.

Thus, for every decision strategy there will be some non-empty set of utility functions for which that decision strategy is "optimal," (where optimal means having an expected utility greater than or equal to all other strategies) even if that set only consists of the degenerate utility functions. Every choice of a non-sufficient statistic on the posterior predictive restricts the set of potential decision strategies that can be produced using that non-sufficient statistic in some way which is guaranteed to be suboptimal for at least one utility function (see Theorem 4.3.0.2), but which will maintain optimality for some non empty subset of possible utility functions. Therefore, some utility assumptions have been made every time a non-sufficient statistic of the posterior predictive is reported, restricting some utility functions while still allowing others.

These results may be unsurprising from a mathematical perspective, but their implications for machine learning have been under-appreciated. It is well known in statistics that statistical summaries can depend on the loss function, however, in machine learning, we have been slow to recognize that not reporting full posterior predictives constitutes implicit end use assumptions which are often not met in practice.

4.4 No-Free-Dinner

It is natural to want to design a black box supervised learning algorithm that will work independent of its possible end uses. As we have seen, eventually some end use utility assumptions must be made, but reporting full posterior predictives (or sufficient statistics thereof) is the only way way to delay these decisions and create such a black box learning system. Any other non-sufficient statistic that could be reported makes some end use utility assumptions.

An alternate view is provided by Corollary 4.3.2.2 which indicates that it is possible to create non-sufficient statistics that are optimal for a distribution over possible end uses. In this case, the exact end use utility function might not be known for sure, even at the time that the decision is made, but it is possible to still be optimal over some distribution over end use utility assumptions.

If we take a similar approach to that of No-Free-Lunch, we could assume that a uniform distribution over end uses (the maximum entropy distribution over end uses) represents a sort of maximum level of uncertainty about the possible uses to which an algorithm could be put. If No-Free-Lunch is about the behavior of algorithms when all possible functions are equally likely, we are now exploring the behavior of algorithms where all possible end uses are equally likely, a situation which we call "No-Free-Dinner."

Theorem 4.4.0.3 (No-Free-Dinner)**:** *If the distribution over possible end use utility functions is uniform, then all decisions have the same expected utility.*

Proof. Equation 4.2 determines how to compute the expected utility when the utility function is unknown, but when a distribution over possible utility function exists.

$$E[U|d] = \int_{f_u \in \mathbb{D}_U} \int_{\ddot{y} \in \mathbb{D}_Y} \int_{o \in \mathbb{D}_O} f_u(o) p(o|\ddot{y}, d) p(\ddot{y}|\ddot{\mathbf{x}}, \mathscr{D}_{train}) p(f_u) do\, d\ddot{y} df_u. \tag{4.3}$$

Because all utility functions are equally likely this becomes:

$$E[U|d] = \frac{1}{|\mathbb{D}_U|} \int_{f_u \in \mathbb{D}_U} \int_{\ddot{y} \in \mathbb{D}_Y} \int_{o \in \mathbb{D}_O} f_u(o) p(o|\ddot{y}, d) p(\ddot{y}|\ddot{\mathbf{x}}, \mathscr{D}_{train}) do\, d\ddot{y} df_u. \tag{4.4}$$

$$E[U|d] = \frac{1}{|\mathbb{D}_U|} \int_{\ddot{y} \in \mathbb{D}_Y} \int_{o \in \mathbb{D}_O} p(o|\ddot{y}, d) p(\ddot{y}|\ddot{\mathbf{x}}, \mathscr{D}_{train}) \left[\int_{f_u \in \mathbb{D}_U} f_u(o) df_u \right] do\, d\ddot{y}. \tag{4.5}$$

We assume that the output of all utility functions are bound by a finite minimum and maximum range. It is then possible to normalize the set of all such functions so that the maximum is 1 and the minimum is -1.

If there is one utility function that gives $f_{u_1}(o) = a$, then it is possible to construct another utility function which gives $f_{u_2}(o) = -a$ for any outcome o simply by negating the result of u_1. However, since all utility functions are equally likely, $\rho(u_1) = \rho(u_2)$. This means that the integral $\int_{f_u \in \mathbb{D}_U} f_u(o) df_u = 0$ regardless of the outcome o. From this, it follows that

$$E[U|d] = \frac{1}{|\mathbb{D}_U|} \int_{\ddot{y} \in \mathbb{D}_Y} \int_{o \in \mathbb{D}_O} p(o|\ddot{y}, d) p(\ddot{y}|\ddot{\mathbf{x}}, \mathscr{D}_{train}) \left[0 \right] do\, d\ddot{y}, \tag{4.6}$$

and thus

$$E[U|d] = 0. \tag{4.7}$$

This means that the expected utility will be independent of the outcome o, and therefore of the decision d, and of the posterior predictive $p(\ddot{y}|\ddot{\mathbf{x}}, \mathscr{D}_{train})$. It also means that, all decisions d have the same expected utility. $\square$

It is also interesting to note that since all decisions are equally good, it does not matter if the posterior predictive was summarized with a non-sufficient statistic or not, since the expected utility is completely independent of the decision and the posterior predictive.

Example 4.4.1: For the email spam filter, if all potential utility functions are equally likely, then no spam filter policy is better than any other. In effect, all spam filters are "optimal" for this uniform distribution over utility functions. Thus, any filter policy that they decide on must involve some utility assumptions more restrictive than uniform, even if they are not explicit.

Example 4.4.2: A researcher is designing a mushroom classification algorithm, and does not know how their algorithm will eventually be used, if they assume maximum uncertainty over possible end use utility functions, namely that all possible utility functions are equally likely, then all classification techniques have the same expected utility, including random, and all summary statistics, including returning the most likely class, or returning a random class, or even returning the *least* likely class, all have the same expected utility. Thus there is no reason to prefer one supervised learning technique over another supervised learning technique independent of some eventual assumptions of end use more specific than a uniform distribution over possible end use utility functions.

This means that even returning the full posterior predictive for the probability of the mushroom being poisonous has no expected benefit over any other statistic so long as end use utility functions are all equally likely at the time that the decision must be made. However, if a single end use, or a more informative distribution over end use utility functions will become known at the time that the decision is made, then according to Theorem 4.3.2.1 returning the full posterior predictive has the advantage of providing enough information to construct an optimal decision strategy *after* this additional information becomes available.

The traditional supervised No-Free-Lunch Theorem (Theorem 3.3.0.2) says that for a uniform distribution $p(f)$, and a specific end use (the 0-1 gain utility function), all decisions d have the same expected utility off training set. The Bayesian No-Free-Lunch Theorem (Theorem 3.6.0.7) says that for a uniform distribution $p(f)$, regardless of end use, the posterior predictive is uniform off training set. The No-Free-Dinner theorem is much stronger than the previous two. It says that for a uniform distribution over possible end use utility functions, all decisions d have the same expected utility regardless of the prior $p(f)$ (machine learning bias), regardless of the posterior predictive $p(\ddot{y}|\ddot{\mathbf{x}}, \mathscr{D}_{train})$, and regardless of whether $\ddot{\mathbf{x}}$ is on or off training set. This means that if all utility functions are equally likely, then the expected utility of all supervised learning techniques is the same. This means that it is impossible to rank supervised learning techniques, even in the presence of a strong prior $p(f)$, when all end uses are equally likely. In this specific case there really is no "best" supervised learner.

This means that if the decision itself must be made while end use is still unknown and if all end uses are equally likely, then the expected performance of all algorithms remains equal, algorithms that return random decisions perform just as well as algorithms that perform the maximum expected utility calculation based on the posterior predictive, since the expected utility is uniform for all decisions. Even algorithms that return the minimum expected utility decision will perform just as well as any other.

However, if more information about the utility function will become available at the time of use, then a best decision technique will also become available at that point so long as the full posterior predictive has been preserved. Thus, the supervised learning inference technique that returns the full posterior predictive may be no better than any other while end use remains unknown. However, this supervised learning technique has maintained its flexibility, and can become superior to other techniques once the end use becomes known, and is guaranteed to be optimal with respect to expected utility at that point. Thus, reporting the full posterior predictive is an effective technique for delaying end use assumptions.

This has broad implications for machine learning. It means that some assumptions about end use (beyond the uniform assumption) must be made for one algorithm to outperform another, either at the time the algorithm was designed, or at the time the algorithm is used to make the final decision. Furthermore, Theorems 4.3.0.2 and 4.4.0.3 together indicate that summaries of the posterior predictive should not be made before some end use assumptions stronger than the uniform assumption can be made. Approaches that report MAP estimates, ML estimates, or any other non-sufficient statistic of the posterior predictive are either making some implicit end use utility assumptions, or they are producing sub optimal approximations to the supervised learning problem. Although heuristic approximations are often necessary, heuristics that approximate full posterior predictives are more likely to be closer to optimal and provide flexibility over possible end use scenarios. Such approximations are therefore more desirable than heuristics that report point estimate summaries. Reporting full posterior predictives is also important in sample complexity and active learning as we will show in Sections 11.3.6 and 6.7.

4.5 Image Super Resolution, A Practical Example

By thinking of supervised learning in terms of the UBDTM, it is possible to more easily determine when utility assumptions are necessary and when they can safely be ignored. One example of this approach involves the Recapture model of image super resolution. Although we were not directly involved in this publication, Toronto et. al. initially proposed this model of super resolution based upon our work with the UBM [111].

In this formulation, $\mathbf{x}$ represents a camera, y represents the image it outputs, and f represents the scene and a process of image capture (in this case a rich feature model of gradients and edges). The inference process involves taking a known low resolution camera model $\dot{\mathbf{x}}$, and an observed low resolution image $\dot{y}$, and inferring the scene model f. This process is analogous to computer vision, and would be synonymous with computer vision if the scene model f was more complex.

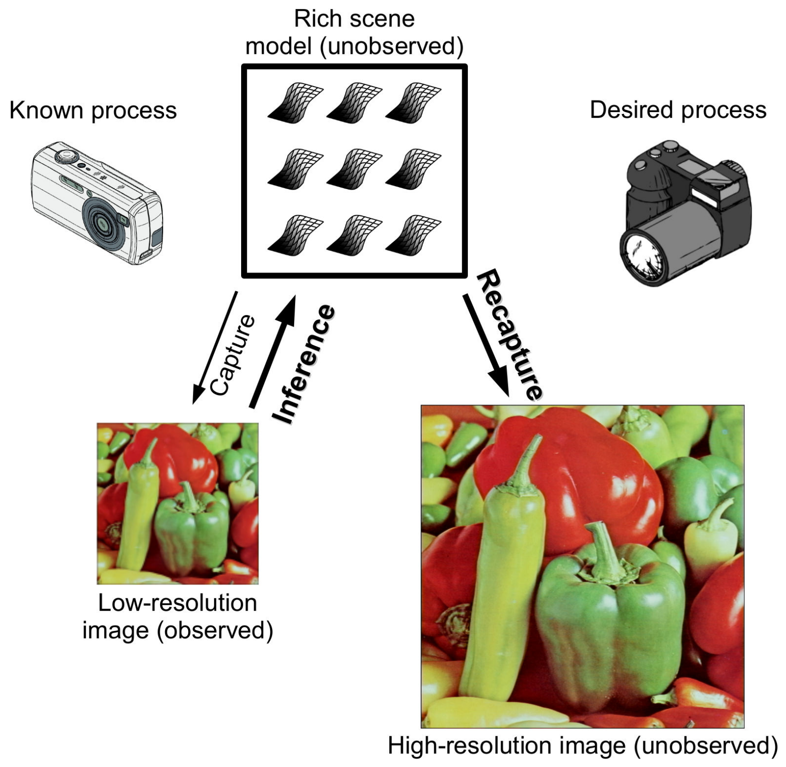


Figure 4.1: An example of how to use the UBM for image super resolution, with capture processes, and with the initial low-resolution image and the inferred high-resolution image.

Then, in order to perform image super resolution, an observed high resolution camera model $\ddot{\mathbf{x}}$ is proposed. Then the posterior predictive of the un-observed high resolution image $p(\ddot{y}|\ddot{\mathbf{x}}, \dot{\mathbf{x}}, \dot{y})$ is inferred given information gained from the lower resolution image (see Figure 4.1). The unanswered question was, how should this posterior predictive over potential high resolution pixel values actually be turned into a high resolution image to return. In the publication, samples were simply taken from the posterior predictive, and the results were composed into the final high resolution image, and this high resolution image was then returned. This technique has yielded good results [111]. Furthermore, simply reporting the mode or the mean of the posterior predictive was observed to return images that were not subjectively as "appealing" to human observers as images drawn randomly from the posterior predictive. However, given the space constraints of the publication, no justification for this sampling approach was presented nor were details of the observed poor performance of other approaches discussed. This sampling approach was eventually justified using the principles of end use in the UBDTM presented in this chapter in a publication by Toronto and myself [17]. These ideas are briefly recounted here as an example of how the results of this chapter can be used to make sense of the behavior of existing algorithms.

Example 4.5.1: The posterior predictive for the high resolution image can be computed independent of end use by Theorem 4.2.0.1. This means that no utility assumptions are necessary in order to produce this posterior predictive. Uncertainty in the posterior predictive consists of uncertainty from two sources: uncertainty about the scene f, and uncertainty inherent in the noisy capture process $\ddot{\mathbf{x}}$.

If uncertainty from the scene is ignored, then all remaining uncertainty comes from the noisy capture process. Thus, images sampled from the posterior predictive can be regarded as representing an approximation to a simulated noisy high-resolution capture process. Since (it is argued) the goal of image super-resolution is to produce such samples, and since an approximation to the full posterior predictive was reported from the inference stage, no end-use utility considerations were necessary. This is in contrast to earlier attempts to define some utility function involving subjective human image appreciation.

Thus, by thinking of the desired results as samples, it is possible to justify this sampling approach in a manner that was completely independent of utility since the posterior predictive can be computed independent of end use assumptions, and since the goal itself was defined as taking noisy samples from this posterior predictive.

However, the above approach does not explain *why* human observers preferred images created according to this noisy sampling approach, nor did it provide a way to quantify their preferences. An alternative way of looking at the image super-resolution problem is to assume that the proper output (the larger image) is a decision d rather than the result of simulating a higher-resolution capture process. The decision would be made based on the posterior predictive and some utility function, and then this utility function can be used as a method of modeling human preferences:

Example 4.5.2: If producing the final image is considered to be a decision, based upon the posterior predictive, then the end use is always the same: to make the decision (produce an image) that is the most plausible to human observers. There is likely no formal utility function that would perfectly match individual human perceptions of image plausibility. Therefore, one goal of the researcher might be to discover a utility function (and therefore an accompanying decision strategy) that approximated human preferences by producing final images which humans felt were subjectively plausible. Indeed, understanding the human conception of plausibility is an active research topic [81]. What the UBDTM provides is a framework in which the human conception of plausibility can be studied and formalized.

Because inference has been factored from decision making in the UBDTM, it is possible to interpret the final image as decision made based on the posterior predictive. Then it is possible to represent the human observer's concepts of plausibility as the end use utility function that governs the decision strategy that turns the posterior predictive into the final image. According to Equation 2.13, a utility function implies a specific decision strategy. In some cases it is also possible to work backwards and infer the utility function that would lead to the sorts of final images that users prefer. This view provides a mechanism for researchers to turn qualitative results such as "image 1 looks better than image 2 to a human observer" into formal utility functions that can approximate human concepts of plausibility.

In Toronto et al. [111], final images included per-pixel Gaussian white noise with the authors giving as a reason that the final image is a simulation from a fictional capture process—but also noting that sampling without this noise yields overly smooth results and that a little noise tends to increase "plausibility" [111]. Using noise to approximate detail is often called "livening" in the super resolution literature. In the past livening has been seen as an unprincipled hack, but in the UBDTM, livening has a reasonable theoretical basis. In the UBM, livening can be considered the result of simulating draws from the posterior predictive, which will naturally produce noise. Perhaps more convincingly, from a decision-theoretic perspective, livening is an approximate non-sufficient statistic for the posterior predictive distribution and a utility function that rewards appropriate levels of balance between detail and noise. Thinking about the problem in this way may be the first step towards formalizing some concepts of human image plausibility, and future work should involve a user study where different detail to noise ratios are evaluated by human observers for image "plausibility" in order to better quantify human image preferences.

4.6 Conclusions

Looking at end use utility for supervised learning in terms of the UBM and UBDTM leads to several interesting conclusions. We separate the supervised learning concepts of inference (producing a probability distribution over the class output), and decision making (determining the best decision given the probabilistic classification). Different parts of the supervised learning process are utility dependent and others are utility independent. If full posterior predictives are reported, then supervised learning inference can be utility independent. If non-sufficient statistics are reported, then supervised learning inference is dependent on utility. Often this dependence has been implicit instead of explicit, but we have argued that there are many benefits to making the utility explicit as in UBDTM.

Much of the machine learning literature has attempted to analyze the supervised learning problems independent of explicit end use, and have instead made implicit 0-1 gain utility assumptions which they see as part of the definition of supervised learning. However, people do not usually classify things for the intrinsic joy of putting things into classes. Rather, they classify things in order to help them make other decisions. Often, there is not even a one to one correspondence between the decisions and the classes. Therefore, the 0-1 gain utility function often does not match the actual practice of supervised learning and is certainly not definitionally mandated by the supervised learning process. Most decision processes return a non-sufficient statistic of the posterior predictive. All such results are sub-optimal for some utility functions, while maintaining optimality for some utility function (if only for the degenerate utility function). Thus, these decision processes all make some utility assumptions, either implicitly or implicitly.

Some have argued that it is possible to classify decisions directly, and that this is one approach to avoid the difficulties of utilities. If this approach is taken, then the UBDTM would be un-necessary, and it would be sufficient to model supervised learning with a modified version of the UBM. However, we have shown that this does not avoid the need for utility, but simply pushes the utility function down into the training examples, making the utility function implicit instead of explicit. This can be undesirable since it could require one training set for each possible utility function.

Even if the decision is no more complex than to report the best label, utilities are involved. For example, one utility function for classification which will result in the learner's reporting the most likely class would be the 0,1 gain utility function. On the other hand, if precision is more important than recall, or if recall was more important than precision, (as was the situation with the mushroom example of predicting poisonous mushrooms), then there is a utility function that would cause the learner to "hedge" its bets in the right way.

The utilities of Table 4.1 is one example of such a utility function. This is a variant of the common precision recall problem. Recall from our mushroom example that it would be easy to mark all poisonous mushrooms as poisonous (perfect recall) by simply marking all mushrooms as poisonous, but at the cost of precision.

In our model, the balance between precision and recall is embodied in the utility function. The balance between precision and recall in traditional algorithms usually involves some implicit utility assumptions. In traditional techniques, if a different balance between precision and recall was needed it was often necessary to redesign the algorithm itself. In our technique a new utility function automatically adjusts the outputs of the system, and no internal modifications are necessary. Furthermore, it is often easier to reason about utilities directly than it is to reason about Receiver Operator Characteristic (ROC) curves, Precision-Recall (PR) curves, or other traditional techniques for balancing precision and recall [22].

Finally, we have shown that when all end use utility functions are equally likely all decisions will have the same expected utility and all classification techniques perform equally well. We call this the "No-Free-Dinner" Theorem. However, if the end use utility assumptions will become known at the time of their use, it is possible to produce the "optimal" supervised learner by delaying end use considerations by reporting the full posterior predictive, and then using decision theory to make the maximum utility decision only once the end use utility function becomes known.

These results should impact the practice of supervised learning in several ways. Thinking about the supervised learning problem in this light forces us to think about utility functions other than 0-1 gain, causes us to prefer algorithms that have some explicit representation of their end use utility assumptions, and forces us to prefer algorithms that have some technique for reporting confidence in their outputs (thereby more closely approximating the full posterior predictive).

We have seen that thinking about image generation in this way may be the first step towards formalizing some concepts of human image plausibility, and future work should involve a user study where different detail to noise ratios are evaluated by human observers for image "plausibility" in order to better quantify the concept of human image plausibility.

Chapter 5

A Priori Distinctions Between Learning Algorithms, No-Free-Lunch and the "Futility of Bias Free Learning"

Abstract

We analyze four common claims made about NFL in the context of the UBM, UB-DTM, and the results of the previous chapters. This will allow us to show that all supervised learning algorithms do not perform equally well in general, that *a priori* distinctions between learning algorithms do indeed exist, that there is a best supervised learning algorithm, and that although bias free learning is futile the previous attempts to show the futility of bias free learning using the NFL results do not actually demonstrate the futility of bias free learning. Instead, we will demonstrate the futility of bias free learning using the need for a prior in the supervised learning UBM. We will also attempt to harmonize the concepts of NFL with those of Bayesian Optimality which at first glance appear to be in conflict.

Publication: Some of the material in this chapter has been previously published in [17] and [16].

5.1 Introduction

The No-Free-Lunch theorems have been highly influential in the machine learning community. There have been many beneficial results that have come from the realizations that have come from NFL. For example, we are now less likely to claim that our supervised learning algorithm is better than another because of improved performance on a few simple test problems. However, NFL has also lead to a large number of incorrect assumptions. Although the formal statement of the traditional supervised learning NFL theorem (Theorem 3.3.0.2) is correct, its implications have been widely misunderstood and misapplied. Many mistaken ideas about NFL still commonly circulate. The following statements are representative of these common misconceptions about NFL:

1. "Bias free learning has been shown to be futile because of the NFL results."
2. "All supervised learning algorithms perform equally well on average."
3. "There is no *a priori* distinction between learning algorithms."
4. "There is no best supervised learning algorithm."

In this chapter, we will show that under reasonable definitions and from a Bayesian perspective, all of the above statements are false. These misunderstandings most commonly arise from a lack of formality in what is meant by terms such as "on average" or "best" or "bias" or "futile" or from a lack of specificity, such as by leaving out restrictions like "off training set" or "for some specific specific utility functions." Although most learning theorists understand that these caveats apply to NFL, sweeping generalizations of the implications of NFL which ignore these caveats are nevertheless common, and often lead to significant misconceptions. As we shall see, it is because of these misconceptions that NFL often appears to contradict the principles of Bayesian optimality.

In what follows, we will give formal definitions for what we mean by terms such as "on average" or "best" and then proceed to show that under those definitions:

1. Bias free learning can be shown to be futile, but not because of the NFL results.
2. All supervised learning algorithms do not perform equally well on average.
3. *A priori* distinction between learning algorithms do exist.
4. There is a best supervised learning algorithm.

Most of the differences between the Bayesian and traditional views of NFL have to do with their differing philosophical perspectives on the learning problem, for example, the traditional focus on the 0,1 utility function, or on off training set error is foreign to the Bayesian perspective. Although some work has been done to harmonize traditional NFL with the Bayesian approach [119] [16] [17], more remains to be done. Here we will attempt to harmonize the competing views of NFL and Bayesian optimality, and show how the Bayesian NFL results help to provide this middle ground. Our purpose will be to address the philosophical differences between the two approaches and to present the NFL concepts from a new Bayesian perspective.

In Section 5.2 we will discuss the connection between NFL and the so called "futility of bias free learning." We will do this in three parts. In Section 5.2.1, we will formalize the concept of "futile" learning, and show that NFL does not lead to futile learning. In Section 5.2.2, we will formalize the concept of "bias free" learning, and show that the NFL case does not actually deal with a "bias free" learning scenario. Finally, in Section 5.2.3, we will provide an alternative view of the futility of bias free learning based upon the need for a prior in the supervised learning UBM..

Next, in Section 5.3, we will attempt to harmonize the concepts of NFL and Bayesian optimality, and we will show that there does indeed exist a "best" supervised learner despite the NFL results. Again, this will be done in parts. First, we will present some definitions of concepts like a "supervised learner," "average performance," and "best." Then, in Section 5.3.1, we show that under these definitions all supervised learning algorithms *do not* perform equally well on average, and in Section 5.3.2, that *a priori* distinctions between learning algorithms do indeed exist, and in Section 5.3.3, we will show that there *is* a best supervised learning algorithm, and a best *a priori* supervised learning algorithm off training set. In Section 5.4 we will conclude by summarizing the implications of these results.

5.2 No-Free-Lunch and the "Futility of Bias Free Learning"

With increasing levels of formality Hume [43], Mitchell [72], Schaffer [98], and Wolpert [121] [122] [124] [67] [123] have attempted to show what is often called "the futility of bias free learning" [74, p. 42]. This concept of the futility of bias free learning has been very influential in both the philosophy and machine learning communities.

In this Section, we will analyze whether the Traditional NFL theorems (Theorem 3.3.0.2) and Mitchell's related Uniform Hypotheses Corollary (Corollary 3.3.0.3) actually demonstrate the "futility of bias free learning." This analysis depends particularly on the formal definitions of the concepts of "bias" and "futility." Traditionally, the uniform distribution over functions has been considered to be "bias free," while the uniform posterior predictive returned off training set in this situation has been considered to cause learning to be "futile." Taken together, this perspective causes the NFL results to appear to demonstrate the futility of bias free learning. However, as we shall see, there are good reasons why both this definition of "bias" and "futile" are problematic. We will first deal with the definition of "futile," then deal with the definition of "bias" and "bias free," and then we will propose an alternative approach to proving the futility of bias free learning.

5.2.1 NFL and "Futile" Learning

First we will deal with the definition of "futile learning." If we define learning as we did in Definition 1.2.0.1, that Machine Learning is improving performance in the presence of data as measured by some performance measure [74], then a futile learning result would be one that for some reason does not aid in decision making in a way that improves performance. Notice that Mitchell's the definition of "learning" does not restrict itself to "generalization" (off training set) improvement. On the contrary, it seems to explicitly require improvement on the part of the test set that will be encountered in practice. Further, it does not restrict itself to a specific utility function. Learning can be futile for one utility function while not being futile in general. These ideas leads to the following definition of futile supervised learning:

Definition 5.2.1.1. Futile Supervised Learning: Supervised learning is futile when inference cannot be mathematically performed, or when all supervised learning decision algorithms (including random) have the same expected utility for all possible test sets and for all possible utility functions.

Even if we temporarily accept the false assertion that the uniform $\rho(f)$ is somehow "bias free," learning under this uniform assumption is not futile under Definition 5.2.1.1. The Bayesian NFL theorem (Theorem 3.6.0.7) indicates that under this uniform assumption the posterior predictive is uniform off training set. There are two reasons that a uniform posterior predictive does not result in futile learning under this definition of futile learning. First, for learning to be "futile" the uniform posterior predictive would have to not provide information that was useful for making decisions, which is false for most utility functions. Second, for learning to be "futile" a uniform posterior predictive would have to be returned both on *and* off training set. But Theorems 3.5.0.5 and 3.5.0.6 showed that this is not the case. We will deal with each of these in turn.

In order to assume that the uniform posterior predictive results in "futile learning," it would be necessary to show that the uniform posterior predictive causes the expected utility of all decisions to be the same (so that no decision algorithm can perform any better than any other). But this is not the case for most utility functions. In many situations knowing that all possible classes have the same probability is hardly useless information. The uniform posterior predictive can provide valuable information that can aid in decision making. For example, knowing that a mushroom is just as likely to be poisonous as it is to be edible is invaluable information when deciding not to eat the mushroom. Therefore, it seems that in general it would be a mistake to assume that the uniform posterior predictive demanded by the NFL theorems is a "futile" leaning situation. It is only in a sub set of the possible utility functions, that the uniform posterior predictive makes all decisions have equal utility. This sub set includes the popular 0,1 gain utility function, and the popularity of this one utility function is one reason for this misunderstanding, even though this function is actually quite rare in practice. A more reasonable definition of futile specifically says that for learning itself to be futile, then it must be futile for *all* utility functions, not just the 0,1 gain utility function, which is simply not the case.

Even if we mistakenly considered a uniform posterior predictive to be a futile learning situation, this uniform posterior predictive is only returned off training set. But Theorems 3.5.0.5 and 3.5.0.6 showed that this is not the case. Thus, in order to assume that NFL demonstrates the "futility of Bias free learning" we must assume that we are only interested in off training set error, which is not the case under our definition of futile, which specifically says that all machine learning algorithms have the same expected utility for *all* possible test sets, including those test sets that include on training set examples. This part of the definition of futile is more reasonable under Mitchell's definition of learning. Clearly there are many practical supervised learning situations where the ability to predict the value of a noisy function is improved with repeated samples all taken in the same location. Therefore, Mitchell's definition of learning demands that we deal with on training set error as well as off training set error. In Section 3.5 we showed the right way to update belief on training set when all functions are equally likely. Clearly in that case it is possible to improve performance on training set as more data is acquired, and therefore learning is not futile.

Although analyzing the off training set error is interesting from the perspective of the "Mathematics of Generalization" [118], it is almost never what actual users of supervised learning are interested in. From the perspective of decision theory, we are most likely to be interested in the expected utility of the learning algorithm over the possible test sets weighted by the likelihood of a test set based on the posterior for the sampling distribution.

In fact, this is what is meant by “inductive” learning. “Transductive learning” involves generalizing to a known test set, while “inductive learning” involves generalizing over an unknown test set, and therefore learning so as to generalize over the entire domain of the function (for more information about inductive and transductive learning see Section 6.3.3). Equation 2.11 computes the expected utility of a vector of decisions given a specific set of data (a transductive expected utility). For inductive learning, we are interested in the average performance for an average but unknown test set, which means that, in order to evaluate the inductive performance of a supervised learning algorithm, we would need to analyze its expected performance over all possible test sets, weighted by the sampling distribution. Therefore, strictly speaking, inductive learning by definition must include some possibility of on training set examples in the test set.

The very assumptions that lead to the proof of a uniform posterior predictives off training set, namely that the function f is discrete and finite and that $\rho(f)$ is uniform (the only cases for which NFL has been proven to hold) means that on training set error *will* play some role in computing the expected utility for inductive learning because the sampling distribution is over a discrete space, and therefore is sure to include some mass for those test sets that include on training set examples. This means that the approach of using expected utility and decision theory to evaluate the inductive performance of an inductive learning algorithm implies that all examples that could be in the test set should be have some contribution to the expected inductive utility performance of that algorithm, and that off training set error is an inappropriate error measure to use.

It is these concepts that lead to *a priori* distinctions between learning algorithms that we will discuss in greater detail in Section 5.3.2.

5.2.2 NFL and "Bias Free" Learning

We will now deal with the definitions of "bias" and "bias free" learning. In order for the NFL proofs to have something to say about bias free learning, the uniform prior for $\rho(f)$ in the NFL learning situation must be considered to be bias free.

Recall from Section 2.4.2 that the concept of an inductive bias has been formalized in several ways. According to Definition 2.4.2.1, an inductive bias is a preference for one function *over* another. In that case, the uniform prior of the NFL theorems does indeed represent bias free learning. However, there are many good reasons to prefer the deductive assumptions definition of inductive bias given by Definition 2.4.2.3, which defines bias as the extra information necessary to deduce the results of the learning algorithm. One major advantage of this approach is that the concept of the hypothesis space, a function class, the prior, and the inductive bias, are all unified into the distribution for $\rho(f)$. This makes the bias interpretable, explicit and clear. Under this definition, it is the prior that provides the bias of an inference approach. These two definitions of bias are mutually exclusive, since in the first, a uniform $\rho(f)$ is bias free, while in the second a uniform prior $\rho(f)$ is still a prior, and it still represents extra information needed before the answer can be deduced, and is therefore an inductive bias.

Nor is the uniform prior bias free in the sense of being influence free. It actually has a very strong influence, namely the bias that all functions are equally likely. An attempt to approach priors that have *no* influence on inference uses what are called "non informative priors" which are very different from uniform priors, nor are such priors actually mathematically correct (they are sometimes known as "improper priors") but can sometimes be interpreted as an approximation to what happens when priors have less influence in the limit. The use of such priors are highly controversial and, "their most reasonable interpretation is as approximations in situations where the likelihood dominates the prior density" [33, p. 62]. Under this interpretation, as with frequentist approaches, these improper priors work because they approximate the behavior of wide proper priors in the presence of a large amount of data. The uniform prior is not one of these non informative priors, and is not bias free. The uniform prior does represent the maximum entropy prior, and the prior which follows Laplace's principle of insufficient reason. Uniform priors at best could perhaps be described as the "least" biased proper prior, but can not be considered to be bias free.

In his book "Machine Learning," Thomas Mitchell gives a definition of inductive bias similar to the deduction definition of inductive bias (Definition 2.4.2.3), but then he endeavors to show what he calls "the futility of bias free learning" by showing what happens when all hypotheses are *equally likely*. Thus, he has changed definitions of bias in the middle. If we were to stick with his initial definition of bias, then all he and the traditional NFL theorems have actually shown is what happens with *one* specific bias, and they have actually said nothing about *bias free* learning. All the NFL proofs really show is that uniform priors lead to uniform posteriors off training set, which can lead to all decisions being equally good for *some*, but *not all*, end use utility functions, and then only for off training set error. As we shall see, it is in Bayes Law itself that the true futility of bias free learning is found.

5.2.3 The Bayesian Futility of Bias Free Learning

If we stick with Definition 2.4.2.3 for inductive bias, and if we assume that "futility" actually means that induction can not be mathematically performed (see Definition 5.2.1.1), then it is possible to construct a proof of the futility of bias free learning as follows:

Theorem 5.2.3.1 (The Futility of Supervised Bias Free Learning)**:** *If the supervised learning UBDTM axioms hold, if inductive bias is defined according to Definition 2.4.2.3, then supervised learning induction (as defined by deducing the posterior predictive) can not be performed without an inductive bias, and supervised learning is therefore futile under Definition 5.2.1.1.*

Proof. Given the standard axioms of mathematics, Kolmogorov's axioms of probability, the axioms of utility, and the axioms of the UBDTM, the only "extra" information needed to deduce the posterior predictive is the prior, $\rho(f)$. Equation 2.9 and Equation 2.8 flow deductively from the axioms, and indicate that the posterior predictive can not be deduced without a prior. Therefore supervised learning inference is impossible without the prior, and therefore without a "bias." Thus, bias free learning is futile. Even the so called "improper" priors provide extra information necessary to deduce the posterior, and therefore would be part of an inductive bias. $\square$

This perspective on Bayesian learning is perhaps best explained by Joseph Kadane who wrote "I believe that assumptions are useful to state [explicitly] in statistical practice, because they impose a discipline on the user. Once a full set of assumptions is stated, the conclusion should follow. (Actually, only a Bayesian analysis can meet this standard...)" [48]. Since the answer requires a prior, and since there is an equals sign in Bayes Law, deduction can only produce the answer using the prior. Techniques that attempt to perform induction without a prior (including frequentist approaches such as maximum likelihood, or non informative priors) can only approximate the answer, and can best be seen as approximations to the correct Bayesian solution which function when there is a large amount of data. They often make this approximation under hidden biases and assumptions [8], and therefore do not represent truly unbiased learning.

Notice that, unlike the NFL rout to the futility of bias free learning, this result actually says something about truly unbiased learning, holds for all utility functions instead of just for a small subset of utility functions, and that it holds both on and off test set, and requires no assumptions about the discrete or finite nature of the domain and range of f.

5.3 Bayesian Optimality and the Existence of a Best Supervised Learning Algorithm

We will now show that under reasonable definitions: "all supervised learning algorithms do not perform equally well on average," "*a priori* distinction between learning algorithms do exist," and "there is a best supervised learning algorithm."

In order to do this, it is necessary to formally define the following terms: "supervised learning algorithm," "on average," "best supervised learning algorithm," and "a priori."

First, we must define what is meant by a “supervised learning algorithm. Recall from Definition 2.3.3.1 that a supervised learning algorithm is defined as a learning algorithm a that takes data $\mathscr{D}_{train}$ and a model of end use (a model of the outcome o, how it depends on the classes and the decisions, and a utility function u) and then returns a mapping between function inputs and decisions. Notice that, as we have modeled the supervised learning process in Chapter 2, there are two parts to this supervised learner, the inductive part (the UBM) and the decision making part (the UBDTM). It is tempting to define supervised learning as simply the inference portion of the problem, and indeed this definition might make more sense, however, as we saw in Chapter 4, supervised learning techniques such as ANNs which return an insufficient statistic on the posterior predictive are actually making a decision, and have therefore made some utility assumptions. Yet ANNs are seen as solving the supervised learning problem, and therefore any definition of supervised learning which attempts to be consistent with prior practice must include the decision portion of the supervised learning problem. The learning algorithm must take the utility function as an input to allow for approaches such as the UBDTM that will return a different decision for different utility functions. However, this definition is still general enough to include learners like traditional ANNs that ignore the utility function and return the same decision for all utility functions.

Next we will define what we mean by performing better “on average.” Every average is some sort of a weighted average. However, usually the term “average” means a uniformly weighted average. In terms of the NFL, the average performance of a supervised learning algorithm usually refers to a uniform average over possible functions f. However, it is possible to represent all possible averages using a function class $\rho(f)$, and then this concept of average performance is related to the concept of expected performance. Thus, the most rational definition of how a supervised learning algorithm perform on average is:

Definition 5.3.0.1. The average performance of a supervised learning algorithm: The average performance of a supervised learning algorithm is defined as the expected utility of that supervised learning algorithm, for a specific utility function, and expected over a function class $\rho(f)$.

Since we are measuring performance with respect to utility, the average performance of a supervised learning algorithm is identical to the expected utility of that supervised learning algorithm with respect to a given $\rho(f)$. Notice that nothing in this definition requires this "average" over functions to be a uniform average, and a more rational average would be over the expected function class for which the algorithm will be applied in practice.

Next it also is necessary to formally define the often used term *a priori*. The current use of this term is unfortunate, because it implies a sort of bias free beginning assumption. As we saw in Section 5.2.2, the uniform prior is neither bias free, nor is there any good reason to consider it to be *a priori*. Unfortunately, the term *a priori* is usually used to refer to the uniform prior for $\rho(f)$, and so we will reluctantly use it in this way, since that is what most people mean when they discuss *a priori* distinctions between learning algorithms.

Definition 5.3.0.2. A Priori Learning: The *a priori* learning scenario means the situation where all functions are equally likely, in other words, a uniform distribution for $\rho(f)$.

Next, we will formally define what we mean by a "best supervised learning algorithm." According to our definition of "on average," the "best" approach "on average" will vary with the function class $\rho(f)$. Therefore we will define the "best supervised learning algorithm" only with respect to a specific function class as follows:

Definition 5.3.0.3. A Best Supervised Learning Algorithm: There are two parts to our definition of a "Best" supervised learning algorithm.

First, a "best" supervised learning algorithm b^* for a given function class $\rho(f)$ is an algorithm with an expected utility that dominates all other algorithms for all possible utility functions over the given function class $\rho(f)$.

Second, for each rival algorithm a', the best algorithm b^* is either identical to the rival (meaning that they return the same decisions for all data sets and for all utility functions, ignoring difference in how they might brake ties between decisions that both maximize the expected utility) or else there exists at least one utility function for which the best algorithm b^* will strictly dominate the expected utility of the other algorithm a'.

Note that the definition of a supervised learning algorithm takes a utility function as an input and that the definition of a best supervised learning algorithm requires that an algorithm dominate all its rivals for all utility functions. This choice is motivated by the observations we were able to make in Chapter 4. A large part of the misunderstandings about NFL grow from the lack of adequate distinctions between the inference and decision portions of the supervised learning problem. For example, someone might believe that there is no best supervised learning algorithm because there is no best supervised learning algorithm for his or her favorite end use utility function, while a Bayesian would see the production of the posterior predictive (the inference portion) as the heart of the supervised learning algorithm, and so any definition of a best supervised learning algorithm would have to measure that posterior predictive generation technique's performance over *all* possible utility functions, not just one. Our definition of best allows for a best inference algorithm, which would then lead to a decision procedure with an expected utility that dominates but does not strictly dominate all rivals for all utility functions, so long as there is at least one end use utility function for each rival where the best algorithm strictly dominates that rival.

At first, it would seem to be sufficient to say that a “best” supervised learning algorithm is one that has an expected utility greater than or equal to all other algorithms for a given function class, however, this would allow there to be a best supervised learning algorithm given a function class for which all algorithms perform equally well for all utility functions, and, therefore, where no algorithm strictly dominates any rival. In that situation, all algorithms would be “best.” To avoid that possibility, our definition of a best supervised learning algorithm requires that the best algorithm strictly dominates each of its rivals in at least one utility function for each rival.

Given these definitions, we are now prepared to tackle each of the above misconceptions in turn:

5.3.1 All Supervised Learning Algorithms *do Not* Perform Equally Well on Average

We will now show that all supervised learning algorithms *do not* perform equally well on average. Under NFL in the EBF all algorithms performed equally well on average, but there are several reasons that this does not apply in general. The first is because this average performance was only for the uniform average, the second is because this same performance was only over off training set error, and the third is because it only applies for a single utility function. In this section we will be especially interested in the fallacy of dealing only with the uniform distribution over $\rho(f)$, and we will deal with the other issues in the following sections.

NFL assumes a uniform average, and therefore a uniform $\rho(f)$. Yet one reason why learning is so useful in practice is because function classes exist for which some algorithms can drastically out perform random. When the prior accurately represents the belief of the learning practitioner, it is unlikely to actually be uniform. Therefore, the uniform distribution over functions can be quite rare in practice. Since the term "average" most commonly refers to the uniform average, the misconception sounds more reasonable. However, since we *should* be averaging over functions in a way that accurately represent the practitioners belief about the likelihood of a given function, it becomes clear that the idea that all algorithms perform equally well on average is actually misleading, because it seems to imply that all techniques for learning perform equally well for all function class, which is not the case.

Example 5.3.1: When classifying mushroom toxicity it is unlikely that the practitioner actually believes that all functions f are actually equally likely. If a subjective prior $\rho(f)$ can be elicited which is more informative than uniform, then the optimal posterior predictive given the better prior can also be computed according to Equations 2.8-2.9, and this will also be optimal, meaning

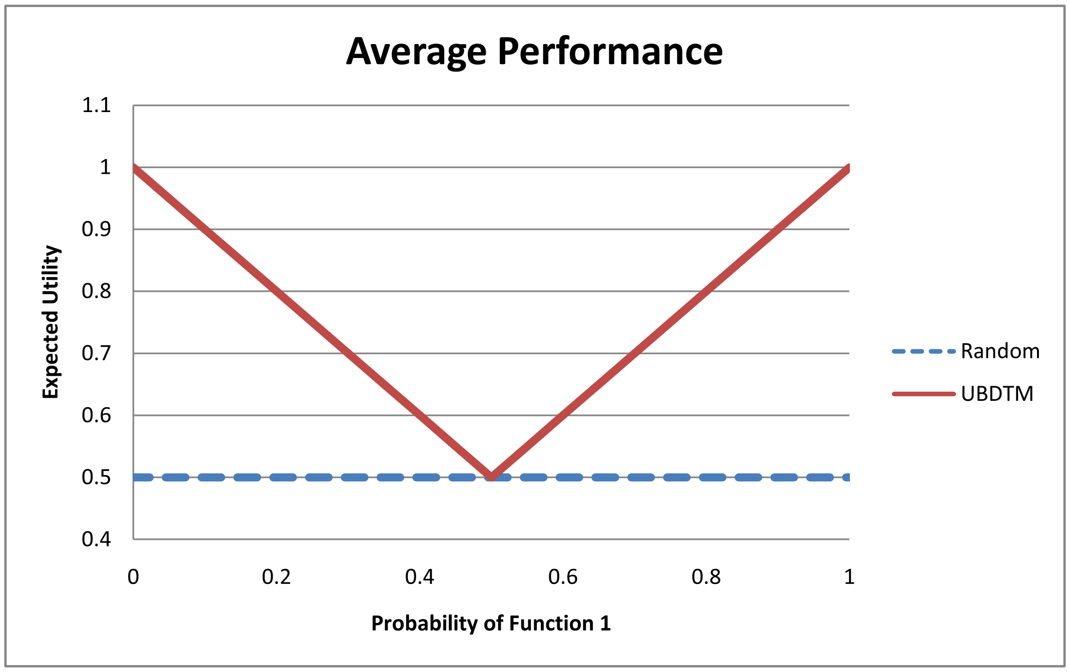


Figure 5.1: Average off training set performance of random vs. the UBDTM with the 0,1 gain utility function. Assume deterministic functions with a single sample location. In this case there are two functions, function 1, where all mushrooms in this sample location are poisonous, and function 2, where all mushrooms in this sample location are edible. The probability of function 1 is presented in the horizontal axis, with the probability of function 2 being one minus the probability of function 1, and with the off training set expected 0,1 gain utility in the vertical axis. When both of these two functions are equally likely (when the probability of function 1 is .5) then all algorithms perform equally well on average. Thus, the average performance of the UBDTM dominates, but does not strictly dominate, the off training set expected utility of all other algorithms, including random over the space of potential function classes. In all cases besides the uniform prior the UBDTM strictly dominates the performance of random.

that the Bayesian approach is optimal for all utility functions and for all priors. This means that for all function classes the UBDTM will dominate all other techniques including random. Further, for some some function classes the UBDTM will strictly dominate random (see Figure 5.1).

It is a mistake to be only interested in the performance of algorithms over one specific rare function class, but when comparing learning techniques we should analyze the average performance of each algorithm for all function class, and when we do, we find that all algorithms do not perform equally well on average. Unfortunately, it is true that most people who say “on average” usually mean uniformly averaged, and we will deal with the misconception that all algorithms perform the same over this uniform average in the next section, where we will show that even under this uniform average all algorithms do not perform equally well in general.

5.3.2 The Existence of *a priori* Distinction Between Learning Algorithms

We will now show the existence of *a priori* distinctions between learning algorithms. The term *a priori* means that we are now dealing with the situation where we want to know if all algorithms have the same average performance for the uniform function class. However, even when we assume a uniform $\rho(f)$, all algorithms do not perform equally well because we are interested in utility functions other than 0,1 gain, and because we are not just interested in off training set performance.

Expected utility is the best measure of performance since it is difficult to determine a situation where the axioms of utility do not characterize the desired behavior for preferences. Once those axioms are accepted, then all performance measures and decision frameworks are either identical to using maximum expected utility, or produce sub-optimal performance, again, by the principle of Bayesian optimality (see Section 1.2.2). We showed in Section 5.2.1 that once expected utility is used as the performance measure of choice, it is difficult to justify the use of strictly off training set error, since for inductive learning an algorithm’s expected performance must also expect over all possible test sets, including those with on training set examples. It is well known that all inference algorithms do not have the same expected utility performance on training set, and that the Bayesian inference algorithm (given in Theorem 3.5.0.5 and 3.5.0.6) together with the principle of maximum expected utility for the decision portion, will produce the maximum expected utility on training set, therefore, this algorithm will dominate, even in the *a priori* case.

Even if we accept the bad idea of using only off training set error, distinctions between learning algorithms still exist because all algorithms only have the same *a priori* expected

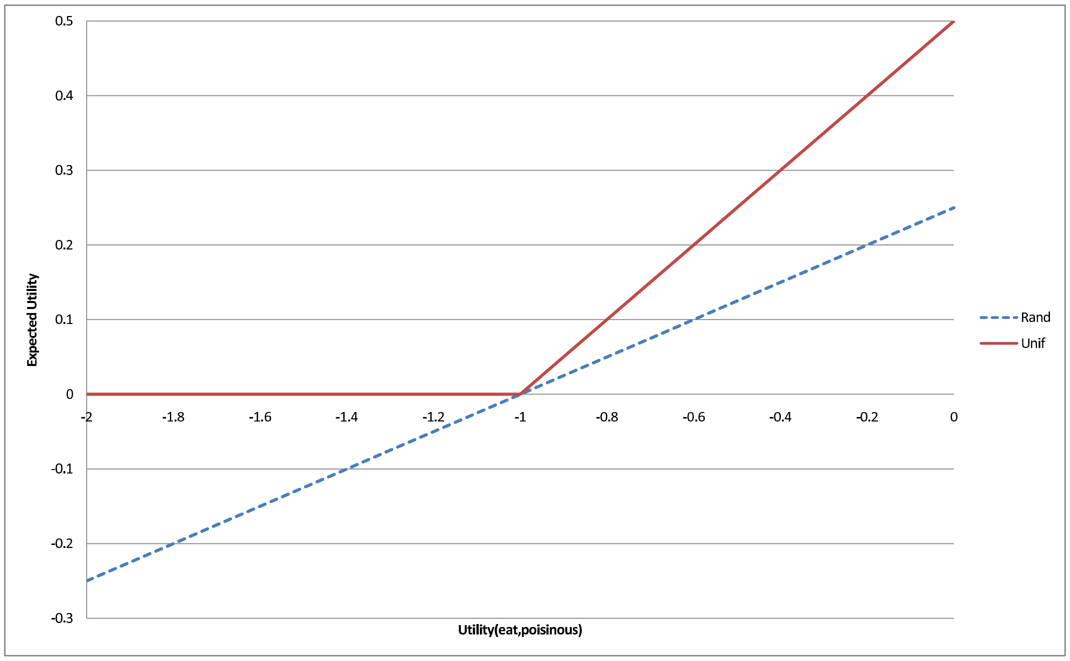


Figure 5.2: Expected utilities of making so called *a priori* off training set decisions either randomly or according to the laws of Bayesian statistics and decision theory. The utility function varies in the horizontal axis, and it is equivalent to a shifted variant of the 0,1 gain utility function when $U(eat, poisinous) = -1$. Only in that case do all algorithms perform equally well off training set. In the other utility functions the UBDTM strictly dominates the random approach.

off training set utility for some (but not all) utility functions. Wolpert has argued that cost functions other than the 0,1 gain utility function produce trivial solutions off training set. This may be true, but just because a solution is trivial does not mean that it is useful. We have argued in Chapter 4 that such cost functions are more common in practice than those utility functions for which *a priori* distinctions between learning algorithms do not exist since it is rare for precision to exactly balance recall. It is only when such utility functions are ignored that no algorithm can perform better than any other *a priori* off training set.

Example 5.3.2: It is simple to construct an example where the expected utility of all supervised learning algorithms is not the same, even in the *a priori* off training set case, by simply varying the utility function (see Figure 5.2).

This simple example illustrates the existence of off training set *a-priori* distinctions between learning algorithms for some utility functions. Here we grant the unlikely assumption that $\rho(f)$ is uniform. Then Figure 5.2 represents expected utility results for the mushroom problem from the Machine Learning Database for different utility functions. In this case, the utility function is 0 if you do not eat the mushroom, and 1 if you eat an edible mushroom, and g if you eat a poisonous mushroom. When $g > -1$ precision in labeling a mushroom as poisonous is more important, when $g < -1$ the recall of catching all poisonous mushrooms is more important. When $g = -1$ the utility function is equivalent to a shifted variant of 0,1 gain (the traditional NFL utility assumption). The case where recall is more important than precision is by far the most likely end use scenario for this particular problem. Note that the expected utility of using the Bayesian approach strictly dominates except in the case where $g = -1$. For this single unlikely case traditional NFL holds. However, Bayesian NFL holds for all settings of g since the posterior predictive that lead to these optimal decisions was uniform off training set regardless of end use. By reporting the true posterior predictive, it is possible to adapt the decision in an optimal way for each utility function. Reporting a posterior predictive other than that dictated by the UBM would have led to sub-optimal decisions even in the *a priori* case.

5.3.3 There *is* a Best Supervised Learning Algorithm, and a Best *a priori* Supervised Learning Algorithm Off Training Set

The final misconception that we will address is the idea that "there is no best supervised learning algorithm." Now that we have formally defined what we mean by a "best" supervised learning algorithm with Definition 5.3.0.3, it is possible to show that there is indeed a best supervised learning algorithm for each function class. We will show that under this strict definition, there is a "best" machine learning algorithm for every possible function class $p(f)$, even for the uniform *a priori* function class. Furthermore, we will show that the UBDTM given a prior *is* that best supervised learner for that function class, even for the *a priori* function class, and even using off-training set error.

Theorem 5.3.3.1 (The best supervised learning algorithm)**:** *If the random variables involved in supervised learning depend upon each other as modeled in the supervised learning UBM and the supervised learning UBDTM, then if the UBDTM is given a function class as a prior, the resulting algorithm is the "Best" supervised learning algorithm for that function class.*

Proof. Under Definition 5.3.0.3, there are two parts to this proof. Part 1 must show that the UBDTM has an expected utility greater than or equal to all other algorithms, while Part 2 must show that there exists a utility function for which the UBDTM will strictly dominate the expected utility of all non-identical rivals.

Part 1 is simply a restatement of the principle of Bayesian optimality from Section 1.2.2, while Part 2 has already been proved in Theorem 4.3.0.2. □

This happens because when finding the best learning algorithm we are interested in utility functions other than the 0,1 gain utility function, and in on as well as off training set error. People have long known that NFL doesn't hold under these conditions and definitions, however, we have argued that these are the very conditions and definitions that we *should* be interested in, and therefore that the generalizations that people often make based on NFL that there is no best learning algorithm does not actually hold.

The above holds for each function class, including the uniform *a priori* function class. Although we have argued that the *a priori* prior is rare, and that off training set error is not the best measure of the performance of a supervised learning algorithm, nevertheless, it is interesting to note that there is also a best *a priori* supervised learning algorithm, even when evaluated using only off training set error:

Corollary 5.3.3.2 (The Best *a priori* Supervised Learning Algorithm Off Training Set)**:** *If the random variables involved in supervised learning depend upon each other as modeled in the supervised learning UBM and the supervised learning UBDTM, then if the UBDTM is given the* a priori *function class as a prior, the resulting algorithm will return the uniform posterior predictive* *off training set, and this posterior predictive will result in the "Best" supervised learning algorithm performance off training set for the uniform function class.*

Proof. There are three parts to this proof. In Part 1, we must show that the UBDTM returns the uniform posterior predictive off training set; in Part 2, we must show that this uniform posterior produces an expected off training set utility greater than or equal to all other approaches; and in Part 3: we must show that for each rival algorithm, either the rival is identical in its behavior, or there exists a utility function for which the UBDTM will strictly dominate that rival.

Part 1, the UBDTM returns the uniform posterior predictive: This was already proved in Section 3.5 and SEC:TheSupBayNFLTheorems, and formalized in the Supervised Bayesian NFL Theorem, Theorem 3.6.0.7.

Part 2, the uniform posterior predictive dominates off training set: Equation 2.11 is the technique for using the posterior predictive to compute the expected utility, regardless of whether the posterior predictive was on or off training set. Then Equation 2.13 selects the decision that will maximize this expected utility for the posterior predictive that corresponds to the supplied $\mathbf{x}$ value, again whether it is on or off training set. Thus, the UBDTM will select decisions that will maximize the expected utility for any test set, regardless of whether it is on or off training set. This means that the decisions chosen will have an expected utility greater than or equal to all others, or it would have chosen the other decision. Simply put, we know that the UBDTM will select the maximum expected utility decision because there is a maximum over the expected utility in the decision strategy of Equation 2.13.

Part 3, for each rival algorithm, either the rival is identical in its behavior, or there exists a utility function for which the UBDTM will strictly dominate that rival: If a rival algorithm is not identical to the UBDTM, then there must exist a utility function u' for which the rival would choose a different decision d_r than would the UBDTM d^*, and it must do so in a situation where the expected utilities differ, otherwise they are defined to have identical performance by Definition 5.3.0.3 (which defined identical algorithm behavior as making the same decisions for all utility functions except when breaking ties between expected utility). But since the decision d^* made by the UBDTM is the maximum expected utility decision given the uniform prior, then an algorithm that makes any other non-tie breaking decision must have sub-optimal expected utility performance. $\square$

For the 0,1 utility function, all algorithms (including random) perform equally well off training set for the *a priori* function class by the traditional NFL theorem (Theorem 3.3.0.2). However, under our definition of "best" it is possible to have equal performance for one utility function, so long as the best algorithm dominates all others and so long as for each of the algorithm's rivals there exists a utility function for which the algorithm strictly dominates that rival. Given that the posterior predictive is uniform off training set, any utility function that does not balance precision and recall precisely will result in the UBDTM selecting decisions in a way that will have a higher expected utility than random (see Example 5.3.2). And for all the other rivals besides random, if those rivals are not identical to the best approach, there will be some utility function for which the UBDTM will have a higher expected utility off training set (see Theorem 4.3.0.2).

The traditional NFL theorem (Theorem 3.3.0.2) specifically only apply to certain utility functions, therefore it should not be surprising that it does not hold for most utility functions. Even though this fact is well known, incorrect statements such as "there is no best supervised learning algorithm" continue to be common. This misunderstanding is partly due to an over emphasis on the 0,1 gain utility function and other utility functions like it, and to a lack of separation between the inference part of the learning problem and the end use part of the supervised learning problem. In practice, utility functions that allow *a priori* distinctions between learning algorithms are quite common, so it is important to accurately report the posterior predictive, even if that posterior predictive is uniform [16] since differences in expected utility between decision can often exist despite the uniform posterior predictive.

5.4 Conclusions

At first glance, the above discussion of a "best" machine learning algorithm appears to contradict the implications of the traditional NFL theorem. However, once the conditions under which the NFL results hold, and the definition of the term "best" are carefully formalized, the seeming contradictions disappear. However, important philosophical differences remain. Although Wolpert used a statistical formalism (the Extended Bayesian Formalism or EBF) in the proofs for his NFL theorems, his presentation of the NFL concepts did not come from a Bayesian perspective or philosophy. Recent efforts in the theory of machine learning have focused on Bayesian approaches [13], and this new focus is forcing a shift in perspective. This shift in perspective forces a new set of definitions, and thus a new set of conclusions.

While traditional theory describes the NFL situation as *a-priori* learning, Bayesians tend to think in terms of learning functions with a uniform prior $\rho(f)$. From the traditional NFL perspective, this uniform prior is somehow *a priori* in the sense that it is the starting point before data is acquired. However, from the Bayesian perspective, the prior should accurately represent the (often subjective) prior belief of the practitioner, while the only real uniqueness of the uniform prior is that it represents the maximum entropy prior. Traditional NFL theory tends to believe that the uniform prior is bias free, while the Bayesian approach connects the idea of a prior and a bias, and therefore sees this uniform prior as part of a bias, and therefore not as bias free. The traditional NFL philosophy tends to focus on the 0,1 gain utility function, while the Bayesian decision theoretical philosophy tends to focus on expected utility for many different end use utility functions. The traditional NFL approach tends to focus on the mathematics of generalization, and therefore on off training set error only, while the inductive Bayesian approach forces one to focus on the expected utility over many possible test sets, and therefore is less interested in focusing on off training set error. In traditional supervised learning theory, the supervised learning problem is to produce a prediction for the class. In Bayesian supervised learning theory the supervised learning problem has two parts, first to produce a distribution over possible classes (the posterior predictive), and then in a separate process to take that distribution and a utility function to produce a decision. Any non-sufficient summary of the posterior predictive is a decision and involves some utility assumptions.

It is because of these philosophical differences that the different approaches lead to different conclusions about the implications of the NFL results. Traditional theory thinks of NFL as showing that all supervised learning algorithms performing equally well. However, Bayesians tend to think in terms of the principle of Bayesian optimality, which implies that there is one algorithm that out-performs all others on average, or that is "best." Traditional NFL theory uses a specific utility function to show that all supervised learning algorithms perform the same off training set, while admitting that this result does not hold for other utility functions. However, from a Bayesian perspective, the production of the posterior predictive is at the heart of supervised learning and is independent of the utility function, and if the full posterior predictive is reported can lead to optimal decisions given any utility function. Therefore, from a Bayesian perspective there is a best technique for producing this posterior predictive independent of the utility function, and a best technique for making decisions given a utility function. Therefore there is a best supervised learning technique for each function class and utility function. Traditional NFL theory tends to claim that there is no best learning approach off training set. However, for inductive learning, Bayesians care about the expected utility over all test sets, not just over the test sets that are off training set. Traditional NFL theory believes that the NFL proofs show the futility of bias free learning because they see the lack of a preference for one hypothesis over another as bias free. However, from a Bayesian perspective, the uniform prior is not bias free, and therefore traditional NFL fails to show the futility of "bias free" learning, because NFL does not say anything about a bias free learning situation.

In this chapter, we have employed the Bayesian perspective to re-evaluated the proofs of the futility of bias free learning. We have shown that the traditional NFL proofs do not necessitate the futility of bias free learning because they only show that one specific prior (which is not bias free) leads to one specific result (which is not futile for most utility functions). We have proposed an alternative proof for the futility of bias free learning based on a formal definition of bias, and based on the presence of a prior in Bayes law and in the UBM.

In this Chapter, we have employed the Bayesian perspective to show the existence of *a priori* distinctions between supervised learning algorithms for some utility functions, and have argued that there is a "best" algorithm for supervised learning given a function class. We have further argued that this best algorithm for each function class is the UBDTM with that function class as the prior. The idea that there is a best algorithm for supervised learning does not mean that there is a best prior for all situations, since there is only a best algorithm for each function class. The UBDTM represents a different supervised learning algorithm for each prior and utility function that is used. For each function class, the optimal prior to use is the distribution over functions that is identical to the function class. Therefore there is a different optimal algorithm for each function class. It is easy to show that different priors work better in different situations. There is therefore no reason to believe that the so called *a priori* uniform prior is a better prior than any other, since it is only optimal for the uniform function class, just as any other prior is only optimal for the function class that it matches.

These results and this philosophical shift have many important philosophical consequences for the Supervised Learning researcher which have not been fully appreciated. For example, in some of our earlier publications, we have justified the use of a distribution over functions $\rho(f)$ in the UBM using the NFL results. The reasoning was that since no algorithm performed better than another off training set on average, therefore any algorithm that actually performs well does so only for a specific function class $\rho(f)$, and that it therefore makes sense for any model of EFO or supervised learning to model this distribution over $\rho(f)$ [16, 15, 80]. However, it was a mistake to believe that no algorithm performs better than any other by Theorem 5.3.3.1, and Corollary 5.3.3.2. Furthermore, Theorem 3.6.0.7, Theorem 7.6.0.3, and Theorem 5.2.3.1 indicate that the need for a prior (and therefore the need for an inductive bias) is a more fundamental fact, required by the mathematical certainty of Bayes law itself, and that this need for a prior over functions drives the NFL results (when that prior is uniform) and *not* the other way around.

Perhaps most significantly, this philosophical shift indicates that the challenge of the supervised learning researcher is no longer to design the best supervised learning algorithm, as that has already been done. The best supervised learning algorithm is simply inference in the UBM given a function class or prior followed by decision theory in the UBDTM. The challenge of supervised learning research is now to select the best prior to use for each application, and then to provide effective heuristic approximations for that prior when inference with that prior leads to intractable learning.

Chapter 6

Supervised Selective Sampling Theory

Abstract

In this chapter we extend the UBDTM to deal with supervised selective sampling, pedagogical sampling, and supervised active learning, by adding the ability to model pool sets, dynamic functions, and sample costs to the UBDTM. There is no single scenario for selective sampling, we therefore survey some of the most common selective sampling scenarios, and create a taxonomy of these learning scenarios. We provide the mathematical definitions of the optimal solutions for these learning scenarios, and evaluate the theoretical implications that this extended AL-UBDTM has for such issues as what variables impact selective sampling, and active learning NFL.

Publication: Some of the material in this chapter has been previously published in [17, 14].

6.1 Introduction

The purpose of this chapter will be to extend the UBDTM to model selective sampling. Selective Sampling attempts to reduce sample complexity by selecting the most useful samples. It is in the context of selective sampling that the utility issues arising from sample costs are most apparent. When performed correctly, selective sampling can balance exploration with exploitation, taking into account end use improvements and sample cost requirements.

If we phrase selective sampling for supervised learning in terms of the UBDTM, then the goal of selective sampling is to select the example $\mathbf{x}$ that, if we knew its corresponding y, would be most beneficial over the rest of the test set. There are at least two types of selective sampling, "active learning" and "pedagogical learning." In "active learning," the learner "actively" determines what information would be most helpful for it to collect in order to perform well on the rest of the problem. In "pedagogical learning," a teacher selects samples for the student that the teacher believes will be most beneficial to the student.

In this chapter, we will explore the optimal selective sampling solutions, based upon the principles of statistics and decision theory. This solution is related to the expected value of sample information, or EVSI. Computing this optimal solution can be computationally intensive, but it is the actual expected utility of acquiring a specific sample. Therefore, other techniques that perform well *must* do so because they approximate this sampling strategy. In many cases, simplifying assumptions can be made. Even when simplifying assumptions are made, it can be helpful to know what the optimal solution is. Therefore, it is important to understand EVSI, even when EVSI is not the implementation of choice. By knowing the actual value that we want to approximate, it is possible to explain why many of the previous techniques work, and to predict when they will fail. For example, Monson has shown that some evolutionary approaches (such as particle swarms) actually function because they sample in locations with the highest EVSI under some simplifying assumptions [77].

Unfortunately, there is no single optimal selective sampling solution. There are many sampling situations, each with a different EVSI solution. Any so called optimal active learning approach will only be optimal for a given active learning situation. Issues such as whether the function is discrete or continuous, whether learning will be on-line or off-line, how the learner will be evaluated (inductively or transductively), how the agent will make decisions (with a mixed or a pure strategy), and the type of active sampling required (pedagogical or active, generative, pool based, or stream based etc.) all impact the optimal solution. We will, therefore, present a taxonomy of selective sampling situations in which we will expand and explain in detail the issues raised above, and then we will provide the expected utility equations for each sampling situation.

In Section 6.2 we will give a short tutorial on the principles of value of information, and the techniques of using decision theory for selective sampling. We will use a simple oil drilling example to illustrate this idea. Then, in Section 6.3 we will explore the many different active learning situations that often arise. Then in Section 6.4 we will describe passive sampling, the standard approaches to sampling either uniformly or according to an approximation to $\xi(\ddot{\mathbf{x}})$. In Section 6.5, we will describe the optimal active learning approach for some of the most common possible active sampling situations. In Section 6.6 we will explore how to use the same ideas to extend active sampling to the pedagogical learning situation. In Section 6.7, we will explore some of the broad theoretical implications of this approach to selective sampling, including an NFL result for active learning. In Section 6.8 we will conclude and propose future work.

6.2 EVSI Tutorial and Example: Drilling For Oil

Consider an oil company that has an opportunity to purchase a plot of land for \$1 million [90]. If the site contains oil, it is worth \$10 million, that is, \$9 million net. Given this information, a utility function may be defined in terms of the boolean decision variable D =

purchase or -purchase, and the boolean query variable Y = oil or -oil:

$$U(purchase, oil) = \$9\text{M} \qquad U(purchase, -oil) = -\$1\text{M}$$
$$U(-purchase, oil) = \$0 \qquad U(-purchase, -oil) = \$0 \ .$$

We use the shorthand $\rho(oil) \equiv \rho(Y = oil)$ etc. If the site's potential for containing oil is known in terms of the prior probabilities, e.g., $\rho(oil) = 0.6$, then the most rational action can be calculated using the expected utility of purchasing, $E(U|purchase)$, and not purchasing, $E(U| - purchase)$:

$$E(U|d) = \sum_{y \in \{oil, -oil\}} U(d, y)p(y)\,.$$

Thus,

$$E(U|d) = \begin{cases} \$5\text{M} & \text{if } d = purchase \\ \$0 & \text{if } d = -purchase \end{cases} .$$

This means that, on average, drilling a plot like this one will produce a \$5 million profit, so the choice to purchase is rational. This result, however, says nothing about whether this *specific* plot should be purchased, only that the prior information suggests good return *on average*. Given such an isolated opportunity, a company would be unlikely to make a purchase based on such a result. Instead, it would likely gather more information first, perhaps in the form of a geological survey. We are interested in discovering the expected value (added utility) of performing this survey. This is the expected value of this additional sample information (or EVSI).

The presence of additional information can improve the company's chances of making a profitable decision, so the availability of a geological survey ("test") transforms the problem from a purchase decision into a decision as to whether the expected value of the survey is greater than the expected cost of the survey. EVSI computes the expected value of the test. Let us suppose, then, that the survey test costs \$0.1 million and can return two possible values S=positive (pos) or S=negative (neg), and has the following characteristics:

$$p(pos|oil) = 0.95 \qquad\qquad p(pos|-oil) = 0.20 \ .$$

This survey test is not very accurate, but may still provide useful information. The outcome of the test may affect the purchase decision, so the expected utility is now conditional upon the survey result as well as the purchase decision:

$$E(U|s,d) = \sum_{y\in\{oil,-oil\}} U(d,y)p(y|s),$$

where $p(y|s)$ is the Bayesian posterior

$$p(y|s) = \frac{p(s|y)p(y)}{p(s)},$$

and the marginal on s is

$$p(s) = \sum_{y\in oil,-oil} p(s|y)p(y),$$

and the prior for the test results is

$$p(pos) = 0.65$$
$$p(neg) = 0.35 \ .$$

Giving that

$$p(oil|pos) \approx 0.88 \qquad\qquad p(-oil|pos) \approx 0.12$$

$$p(oil|neg) \approx 0.09 \qquad\qquad p(-oil|neg) \approx 0.91 \ ,$$

therefore

$$E(U|s,d) = \begin{cases} \$7.8\text{M} & \text{if } s = pos, d = purchase \\ -\$0.1\text{M} & \text{if } s = neg, d = purchase \\ \$0 & \text{if } d = -purchase \end{cases} .$$

The presence of information changes the expected utility of drilling and, therefore, affects the purchase decision. Specifically, if the survey is positive, then the maximum expected utility decision would be to purchase the land, but if the survey is negative, then the maximum expected utility decision would be to not purchase the land. Therefore, knowledge of the survey's outcome is desirable because it guides the decision process such that maximum expected utility (MEU) can be increased.

However, knowing that the survey has some value is not the same as knowing that the survey should be performed since the survey itself costs \$0.1 million. Thus, it would be helpful to know the expected value of performing the test. If this expected value of sample information (EVSI) is larger than the cost of the survey, then performing the survey will improve utility on average, while if the EVSI is less than the cost of the survey, then performing the survey will reduce expected utility on average, and we need to approximate this value *before* the survey is performed. EVSI is the right way to calculate this expected utility while taking the accuracy of the test into account:

$$\begin{aligned}
EVSI_{Survey} &= E(\max_d E(U|s,d)) - \max_d E(U|d), \\
&= \mathrm{E}\left[\begin{cases} \$7.8\text{M} & \text{if } S = pos \\ \$0 & \text{if } S = neg \end{cases}\right] - \$5\text{M} \\
&= ((\$7.8\text{M})\, p(pos) + (\$0)\, p(neg)) - \$5\text{M} \\
&= \$5.07\text{M} - \$5\text{M} \\
&= \$0.07\text{M}
\end{aligned}$$

Survey results are on average worth about \$0.07 million, but the survey is not cost-effective because the survey costs \$0.1 million. Were the survey tests more accurate, the expected value improvement might be higher; as it is, a lower price for the survey should be negotiated or a more accurate test utilized if one is available.

In this example, EVSI has answered an important question: whether or not to pay for the survey before making a purchase decision. If two surveys with distinct costs and accuracies are available, EVSI can also be used to choose between them based on their net expected utilities. The test with the highest EVSI should be taken. It is in this latter context that EVSI can be used in selective sampling and active learning, since the teacher or student should select the sample with the highest EVSI. This approach will produce the optimal sampling procedure with respect to expected utility. Since EVSI gives the value of exploration in terms of utility, the same measure used for exploitation, it is also possible to use the principles of EVSI to explicitly and optimally balance exploration vs. exploitation.

6.2.1 Maximum Expected Utility Sampling Decisions, An Alternative Perspective to the Same Solution

EVSI is the expected utility with the added sample information (the New Maximum Expected Utility or NMEU) minus the expected utility without the added sample information (the Previous Maximum Expected Utility or PMEU), thus: $EVSI = NMEU - PMEU$. As we have seen, the principle of EVSI can be used to answer two questions, which sample should be taken, and whether any potential sample is expected to return more benefit than the cost of taking that sample. If the EVSI of all possible samples locations are less than the expected cost, then the best action is to stop sampling.

An alternative approach that yields the same results is to simply take the sampling decision with the highest net maximum expected utility. This approach is equivalent to the above procedure, but is sometimes mathematically easier, especially in situations where some sample *must* be taken, causing $PMEU$ to have no meaning.

We will use the same oil situation as an example to illustrate this alternative approach. The goal is to decide whether to take the survey or not. The random variable X can be either $X = survey$, meaning to perform the survey, or $X = -survey$, meaning to not perform the survey. Then the optimal decision as follows:

$$Result = \underset{x \in \{survey, -survey\}}{\text{argmax}} \left[NME(U|survey), NME(U|-survey \right], \tag{6.1}$$

where NME stands for the net maximum expectation (maximized over all decisions, and expected over all outcomes, and subtracting all sample costs). Remember that the survey costs \$0.1 million, and we assume that there is no cost associated with not taking the survey.

Then,

$$
\begin{aligned}
NME(U|survey) &= E(\max_d E(U|s,d)) - C(survey) \\
&= ((\$7.8\text{M})\, p(pos) + (\$0)\, p(neg)) - \$0.1\text{M} \qquad (6.2)\\
&= \$5.07\text{M} - \$.1\text{M} = \$4.97\text{M}, \qquad (6.3)
\end{aligned}
$$

and,

$$
\begin{aligned}
NME(U|-survey) = \max_d E(U|d) - C(-survey) = \\
\$5\text{M} - \$0\text{M} = \$5\text{M} \qquad (6.4)
\end{aligned}
$$

As before, the best thing to do is to not take the survey, but notice that the second choice could have just as easily been to take a different survey instead. Thus, unlike the first approach, this approach does not require there to be a well defined $PMEU$ term, but will be functionally equivalent if there is. This second approach will be especially useful in some online learning situations, where the term $PMEU$ is not always well defined, when *some* sample must be taken at each time step. It is easy to see why these two approaches are synonymous. In the first, we subtracted \$5.07M−\$5M and determined that the resulting VOI was less than the cost of .1, and so the best action was to not take the survey. In the second approach, we subtracted the costs from the expected utility for each sampling possibility, .1 from taking the survey, and 0 from not taking the survey, and then we noted that the expected net utility of not taking the survey was the best choice. These two formulations will always return the same sampling strategy and we will use them interchangeably in this Chapter, depending on which leads to the simplest exposition for each situation. We will refer to them both as involving the “expected value of sample information” although that term could perhaps best be applied only to the first approach.

6.3 A Taxonomy of Sampling Situations

In order to apply the principles of the Value of Information to the UBDTM, it will be necessary to extend the UBDTM into a dynamic Bayesian network. Dynamic Bayesian networks involve distributions that progress over time.

When applying the principles of expected utility to sampling in the UBDTM, there is no single dynamic model that accurately matches all of the many different selective sampling situations. Although the optimal selective sampling *approach* is known (EVSI on a dynamic variation of the UBDTM), there are often fundamental differences between how the UBDTM should be extended for each of the many different selective sampling situations. Thus, in order to give the optimal selective sampling utility equations, it is first necessary to produce a taxonomy of some of the most common selective sampling situations, and to evaluate how the UBDTM should be extended in each of these situation.

Here we will outline the sorts of differences that are found between selective sampling situations and provide a taxonomy of some of the sampling situations. In the sections that follow we will go into detail about some of the most important of these sampling situations, and apply the principles of EVSI to produce the optimal sampling approach for each situation.

6.3.1 Functions

Selective sampling equations depend on the type of functions that are to be learned. Specifically, optimal selective sampling often requires tracking belief about the function over time and integrating over the domains and the ranges of the function.

Functions can be stationary or dynamic. Active learning for the traditional supervised learning situation usually assumes that the function is static and does not change over time, while selective sampling for dynamic pricing usually assumes a dynamic supply and demand which changes from time to time. This difference can have important implications in the optimal active learning solution and for the value of information. For example, the value of exploration may be lower for a dynamic function since information about the function now may not be as useful later, while that same information about a static function could have exploitable uses for a long time into the future.

Functions can also be either discrete or continuous, and they can be finite or infinite in their domains and ranges. For the most part, such differences will only change the active learning equations by turning sums into integrals and by changing the ranges integrated over. We will assume that all the functions we are working with in this Chapter have a discrete domain and range, with the understanding that by simple exchanging the sums for integrals, the continuous versions of the optimal sampling equations can be produced.

Functions can also be either deterministic or stochastic. An active learner that queries a deterministic function will have sure knowledge of the function at that sample location, and there will never be reason to sample in the same location again, while it may be useful to sample in the same location twice when performing active sampling on a stochastic function. Usually these differences are taken care of in the probability distributions involved and, although they affect the active learner through those distributions, they do not affect the optimal model or the optimal equations. We can therefore temporarily ignore such function differences when producing the expected sampling utility equations.

6.3.2 Query Scenarios

There can be many ways in which selective sampling can generate its queries.

For example, a pool based active learner has a pool set $\mathscr{D}_{pool}$, with observed $\breve{\mathbf{x}}$ features and un-observed $\breve{y}$ labels that can be revealed for some cost. This sort of query generation limits the selective sampler to examples from the pool set.

Another common selective sampling scenario is "stream based" selective sampling. In stream based selective sampling, a stream of feature locations are sent to the active learner. The task of the learner is to either label the example itself, or to pass the example to an oracle that would label the sample for some cost.

Another common selective sampling scenario is “generative” selective sampling. If the selective sampler has access to an oracle that can reveal the label for an arbitrary $\mathbf{x}$ feature location for some cost, then that selective sampler could *generate* any $\mathbf{x}$ sample location and present that query location to its oracle. We call such active samplers “generative” active samplers as opposed to a “pool based” or “stream based” active samplers. Algorithms that perform generative sampling are sometimes called “Constructive Query Algorithms” [70], and the process of generating these queries is sometimes called “Membership Query Synthesis” [102].

6.3.3 Methods of Evaluation

Inductive and Transductive Learning

Function learning can be evaluated in many ways. For example, the goal may be to create a learner that will have good performance uniformly over its domain (weak inductive learning), or that will have good performance over a portion of its domain determined by some non-uniform sampling distribution (strong inductive learning), or that will have good performance over a previously determined test set (transductive learning). For example, learning to classify mushrooms could involve weak or strong induction depending on how much is known *a-priori* about the sampling distribution, while learning to correctly label parts of speech (POS tagging) for a known corpus (as we will do in Chapter 9) involves transductive learning.

Transductive learning can either have a locked test set or a queryable test set. A locked test set is a test set with labels that can not be revealed, although it may be possible to query in a location with the same features. Such a query will only be identical to the example in the transductive test set if the function is deterministic. On the other hand, a queryable transductive test set has labels that can be revealed for some cost, after which the learner is guaranteed to know the actual label in the test set, even for non deterministic functions.

Each of these situations will affect the active learning approach. For example, a weak inductive assumption may encourage an active sampler to take relatively uniform samples across the function's domain, while a transductive learning assumption may encourage the learner to take samples from the test set if the test set is queryable, or from sample locations identical or similar to those of a transductive test set that is locked.

On-line Verses Off-line

Another important distinction in the way in which functions will be evaluated is whether the performance of the learner is evaluated on-line vs off-line.

An off-line selective sampler actively selects samples (explores) off-line, then then uses the information gathered during the exploration phase in order to perform well (exploit) at some later date. Thus, an off-line learner focuses on optimal exploration and exploitation in distinct and separate stages.

On the other hand, an on-line learner is evaluated at the same time as it actively selects samples. Since there is no distinct exploration and exploitation stages in on-line selective sampling, on-line learning is more concerned with balancing exploration with exploitation.

6.3.4 Agent Strategies

Agents can produce their decisions according to a mixed or a pure strategy:

In most decision theory situations, there will be a single optimal decision. Agents that return this single optimal decision are said to have a "pure" strategy. Recall from Section 2.3 that the expected utility given the observed data $E(U|d, \mathscr{D}^o_{ata})$ can be computed according to Equation 2.11. Then, for a pure strategy, the optimal decision $d^* \in \mathbb{D}_D$ that will maximize the expected utility can be found by Equation 2.13 as follows:

$$d^* = \operatorname*{argmax}_{d \in \mathbb{D}_D} E(U|d, \mathscr{D}_{ata})$$

This will usually be manifest in the math by a max in the EVSI equations as EVSI assumes that the decision will be this optimal decision.

On the other hand, when it is important for an agent to act in a non-deterministic manner, an agent might return a “mixed strategy.” A mixed strategy agent would determine some distribution over possible decisions, and then select its decision randomly according to this distribution. A mixed strategy will often be preferable in adversarial multi-agent situations when it is important for an agent to not behave in a predictable manner. In such cases it is possible to define optimality in terms of Nash equilibrium or pareto optimal solutions. These solutions often involve mixed strategies. One mixed strategy that has historically been important (especially for active learning) is to select the decision in a manner proportional to the expected utility as follows:

$$p(d) = \frac{E(U|d)}{\sum_{q \in \mathbb{D}_D} E(U|d_q)},$$

where the utilities have been shifted so that all utilities are scaled to be between 0 and 1. If we assume that the task is to correctly select a class, and that the outcomes are deterministic with the 0,1 utility function, then $p(d) = p(y|.)$. Such mixed strategies often result in replacing the maxes in the EVSI equations with average case sums, leading to measures of performance such as the reduction in “Bayes risk.” Unfortunately, Bayes risk reduction is sometimes used inappropriately for pure strategy active learning, which is incorrect (for an example of this common error see [2]).

6.3.5 Types of Sampling

Active learning may be the most common form of selective sampling, but it is by no means the only type of selective sampling. Sampling can be broken down into passive sampling, active sampling, and pedagogical sampling.

Weak passive sampling involves sampling uniformly across the domain of the function. Strong passive sampling on the other hand samples according to the sampling distribution $\xi(\mathbf{x})$. Although strong passive sampling is still passive, it can greatly improve the sample complexity of a learning algorithm by sampling rarely in feature locations that are rare in practice.

Selective sampling involves selecting samples according to some informed distribution. Machine learning has been long aware of active learning (the active sampling situation), where the learner itself attempts to determine the next sample location. However, there is another, often neglected selective sampling situation, namely pedagogical sampling. In pedagogical sampling a teacher who knows the true function selects samples so as to help the student learn the function as effectively as possible [104]. The mathematics of pedagogical sampling is essentially the mathematics of optimal teaching. It is often the case that pedagogical sampling can reduce the sample complexity of a function much more effectively than can active learning.

6.3.6 The Taxonomy

There are at least 2304 different ways in which these various possibilities can be combined, just with the variants that we provided above, and there are likely many other special cases and variations that could be enumerated. Clearly we can not discuss them all. Therefore, we will select a few of the most important and common examples and give both a graphical model and an optimal sampling utility equation for each. Although our selection will not be exhaustive, it is hoped that we will provide enough examples that the reader can produce the graphical model and optimal solution for their specific problem.

Some of the above differences can be organized into a type of useful taxonomy or hierarchy of potential sampling techniques (see Figure 6.1). We are not listing all possibilities in this figure, for example, each leaf node can be continuous or discrete, inductive or transductive, etc. However, this figure does illustrate some of the major divisions and

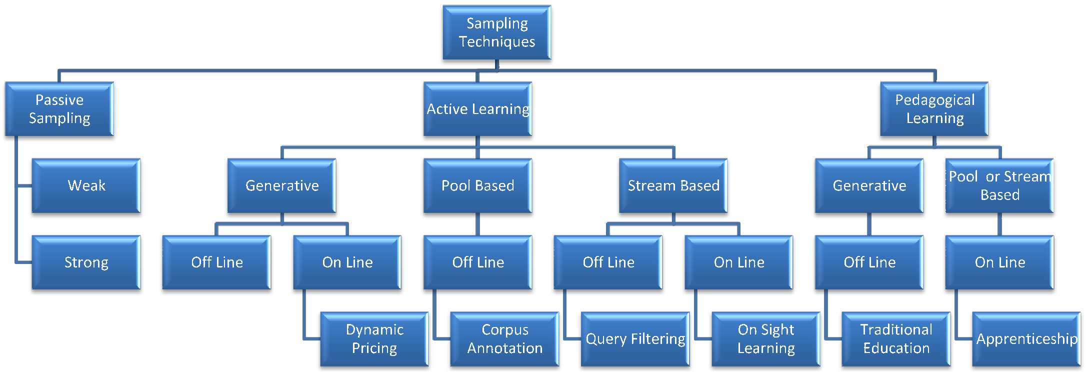


Figure 6.1: Taxonomy of Selective Sampling Scenarios.

common situations. Some specific sampling scenarios have been named. For example, selective sampling in the active, generative, on line, simplified, approximately continuous case is known as "dynamic pricing," while selective sampling in the active, stream based, off line case is sometimes known as "query filtering." Selective sampling in the active, stream based, on line case is sometimes known as "on sight learning."

We will provide a graphical model and an optimal equation for each of these selective sampling approaches, beginning with the examples on the left of the figure, and moving to the right. In some cases we will further break the solutions down into inductive or transductive situations.

6.4 Passive Sampling

As we have seen, there are two main forms of passive sampling, weak and strong. Weak passive sampling involves drawing samples uniformly. Strong passive sampling involves drawing samples according to the same sampling distribution of the test set. This can be helpful because samples will not be taken in locations that will not be encountered in the test set, and can therefore improve sampling efficiency. However, there is no teacher or learner that is active trying to select the samples that will be most helpful to the learner. Nevertheless, strong passive sampling can be seen as a first step towards active or pedagogical sampling.

6.5 Optimal Active Learning

For pedagogical sampling we can assume that the teacher knows the true function f^*, but what if the student, who does not know the true function, is selecting the samples that he or she thinks are best? This is the *active* learning scenario. Because the student does not know f^*, they do not know the outcomes of the experiments that he or she might try in the same way that a teacher would. However, the student does have a prior, or a posterior predictive given the data seen so far, and it is therefore possible for the student to use his or her beliefs to expect over the possible outcomes of the experiments he or she might perform, much as we did in the EVSI tutorial of Section 6.2.

In the active learning scenario there is no teacher, and we are approaching the concept of optimality from the perspective of the student, so we will assume that given data, the student updates beliefs optimally according to Equations 2.9, and 2.8, and makes decisions optimally according to Equations 2.13 or 2.14.

In order to extend the UBDTM to deal with active learning, we must make some additional model assumptions in order to model sample costs and to deal with time steps and the potential for dynamic functions. Thus, the AL-UBDTM extends the UBDTM by adding sample costs and the potential for a dynamic function to the UBDTM, just as the UBDTM added end use utility to the UBM.

AL-UBDTM Assumption 1: Sample costs at a given time step, i, directly depend only upon the sample location and sample value at the same time step, $C(\mathbf{x}_i, y_i)$.

AL-UBDTM Assumption 2: The function at time step i, denoted f_i, depends only upon the pervious function f_{i-1} through a linear transformation represented by the transition matrix T.

It is possible to think of discrete functions as a vector, and thus, to model the way a function might change over time with a matrix. This was the approach taken by Wolpert to deal with dynamic functions [124], and future work is necessary to deal with continuous dynamic functions, however, so long as some variable T determines how the function changes over time, then the math that follows holds, even if f is continuous, and if T is not a matrix.

This more complex model for dynamic functions can still model the static function case simply by setting T to the identity matrix. This more complex model also requires some slight changes to the math we had in Chapter 2. For example, each element in any data set must now also contain time stamps, so that the data, $\mathscr{D}_{ata}$, now consists of a set of triplets with time t, $\mathbf{x}$, and y. Notationally, we can define $\mathscr{D}_i$ to be all the data observed up to time i. Then, the posterior predictive becomes:

$$p(y_i|\mathbf{x}_i, \mathscr{D}_{i-1}) = \sum_{f_i \in \mathbb{D}_F} p(y_i|\mathbf{x}_i, f_i) P(f_i|\mathscr{D}_{i-1}), \tag{6.5}$$

and $P(f_i|\mathscr{D}_{i-1})$ can be computed as follows:

$$P(f_i|\mathscr{D}_{i-1}) = \sum_{f_{i-1} \in \mathbb{D}_F} P(f_i|T, f_{i-1}) P(f_{i-1}|\mathscr{D}_{i-1}), \tag{6.6}$$

and $P(f_j|\mathscr{D}_j)$ can be computed recursively as follows:

$$P(f_j|\mathscr{D}_j) = \sum_{f_{j-1} \in \mathbb{D}_F} P(f_j|T, f_{j-1}) P(f_{j-1}|\mathscr{D}_{j-1}), \tag{6.7}$$

if $j > 1$ and otherwise with:

$$P(f_1|\mathscr{D}_1) = \sum_{f_0 \in \mathbb{D}_F} P(f_1|T, f_0, \{\dot{\mathbf{x}}_1, \dot{y}_1\})P(f_0), \tag{6.8}$$

where the prior is a distribution over initial function states f_0. One common notion in active learning is that of a seed set, $\mathscr{D}_{seed}$. This is the set of training data acquired before active learning begins, often with data acquired according to the sampling distribution $\xi(\dot{\mathbf{x}}|f_j)$. This can allow the use of semi-supervised or un-supervised learning. Notice that although un-supervised and semi-supervised learning can be performed with the seed set, the standard un-supervised and semi-supervised learning approaches can not be used with actively acquired data, since that data was not been acquired according to the sampling distribution. From this point on, we will generally ignore the presence of a seed set. This can be done without loss of generality, since the posterior for one learning round can be used as the prior for the next round of learning. Thus, $P(f_0)$ could have been acquired either with a seed set or not, without changing how active learning would proceed from that point on.

With these additional assumptions, we are now ready to use the principles of the AL-UBDTM to produce the optimal solution to several active learning problems.

6.5.1 Generative Active Learning

Recall from Section 6.3.2, that "generative" selective sampling, as opposed to a "pool based" or "stream based" selective sampling, involves the situation where the sampler can select an arbitrary $\mathbf{x}$ feature location to be labeled for some sample cost. Such a selective sampler could be said to *generate* any $\mathbf{x}$ sample location and present that query location to its oracle. Algorithms that perform generative sampling are sometimes called "Constructive Query Algorithms" [70], and the process of generating these queries is sometimes called "Membership Query Synthesis" [102].

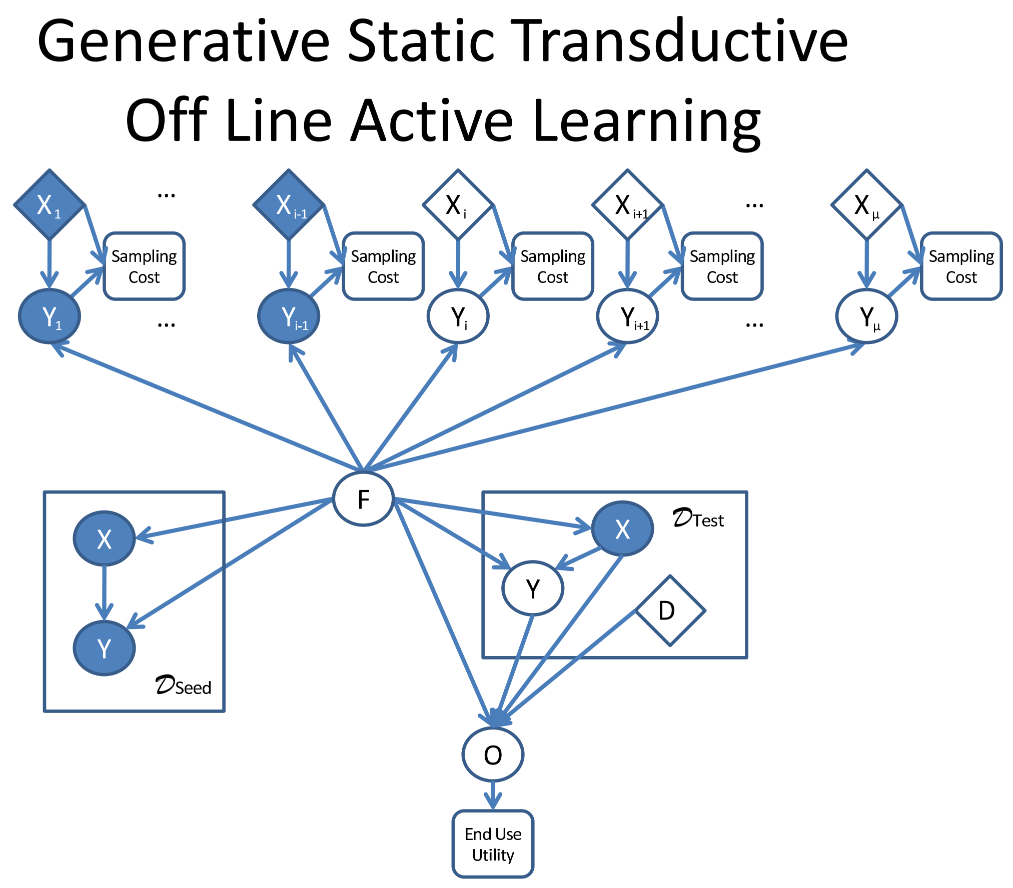


Figure 6.2: The generative dynamic transductive off line active learning case at the ith time step.

Generative Off Line Active Learning

We will begin with the generative, static, transductive, off line, active learning case. There is no single graphical model for this case, because which nodes are observed, and which decisions have yet to be made change over time, creating a new graphical model for each time step. The graphical model for this case at time step i is shown in Figure 6.2.

The expected utility of sampling the $\mathbf{x}_i$ location at time step i, given the data $\mathscr{D}_{i-1}$ and the fact that we expect to eventually take μ samples can be found as follows:

$$\begin{aligned} E(U|\mathbf{x}_i, \mathscr{D}_{i-1}, \mu) = & \sum_{y_i \in \mathbb{D}_Y} p(y_i|\mathbf{x}_i, \mathscr{D}_{i-1}) \\ & \max_{\mathbf{x}_{i+1} \in \mathbb{D}_X} \sum_{y_{i+1} \in \mathbb{D}_Y} p(y_{i+1}|\mathbf{x}_{i+1}, \mathscr{D}_i) \\ & \ldots \\ & \max_{\mathbf{x}_\mu \in \mathbb{D}_X} \sum_{y_\mu \in \mathbb{D}_Y} p(y_\mu|\mathbf{x}_\mu, \mathscr{D}_{\mu-1}) \\ & \sum_{x_k \in \mathscr{D}^X_{test}} \max_{d_k \in \mathbb{D}_D} \sum_{f \in \mathbb{D}_F} P(f|\mathscr{D}_\mu) \sum_{\ddot{y}_k \in \mathbb{D}_Y} p(\ddot{y}_k|\ddot{\mathbf{x}}_k, f) \\ & \sum_{o_k \in \mathbb{D}_O} p(o_k|\ddot{\mathbf{x}}_k, \ddot{y}_k, f, d_k)\big[U(o_k) - C(\mathscr{D}_\mu \setminus \mathscr{D}_{i-1})\big] \end{aligned} \tag{6.9}$$

The first four lines of the above equation involve exploration, while the last two lines involve the off line exploitation. It is this separation that makes this an off line version of the problem. Once we can compute $E(U|\mathbf{x}_i, \mathscr{D}_{i-1}, \mu)$, then the optimal sampling approach at each time step would be to select the $\mathbf{x}_i$ with the highest utility:

$$\operatorname*{argmax}_{\mathbf{x}_i} E(U|\mathbf{x}_i, \mathscr{D}_{i-1}, \mu)$$

The above situation is transductive because the exploitation part of the equation involves a sum over all the values in a known test set. This equation will optimally select samples to help learn a specific test set, but may not perform well on examples outside of that test set.

If the function were dynamic, then very little actually must change. The generative, dynamic, transductive, off line, active learning graphical model, at time step i is shown in Figure 6.3. Everything is the same except that now there is a new f node at each time step. The maximum expected utility equation is also almost identical:

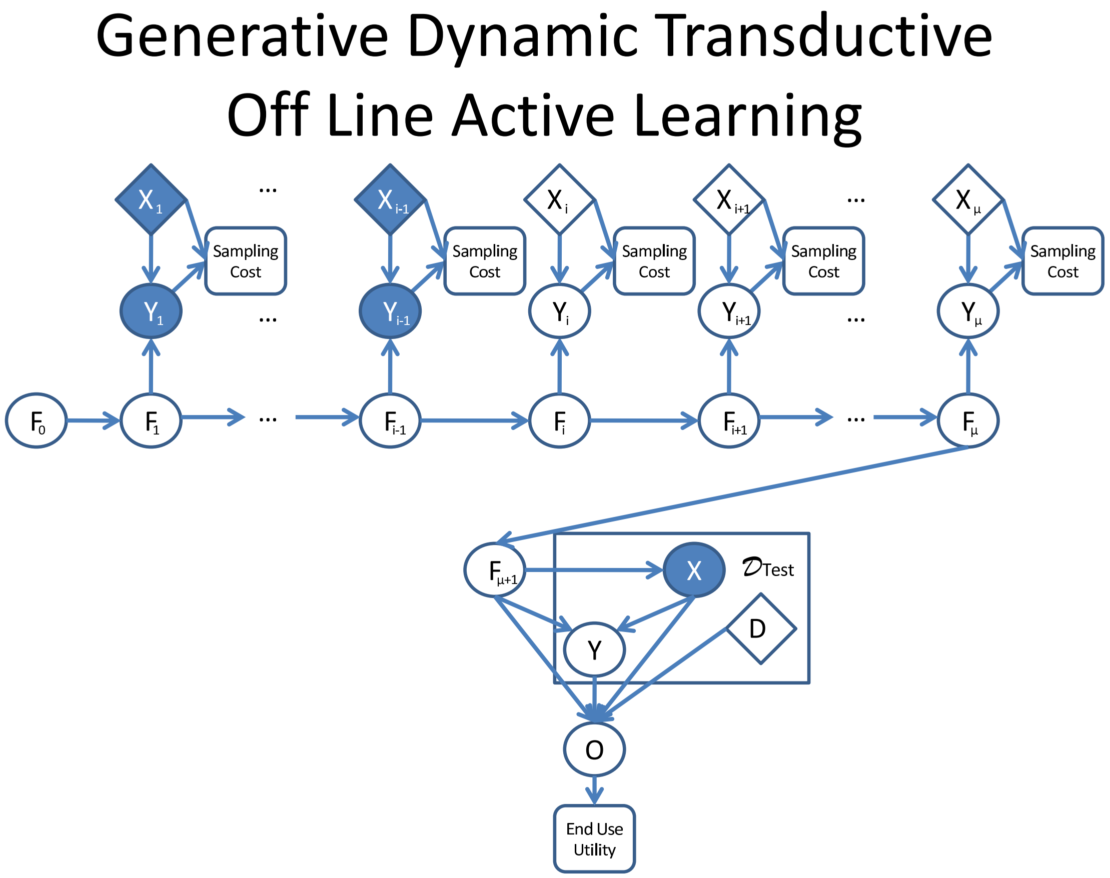


Figure 6.3: The generative dynamic transductive off line active learning case at the ith time step. Compare to the static case shown in Figure 6.2.

$$\begin{aligned}E(U|\mathbf{x}_i, \mathscr{D}_{i-1}, \mu) = &\sum_{y_i \in \mathbb{D}_Y} p(y_i|\mathbf{x}_i, \mathscr{D}_{i-1}) \\ &\max_{\mathbf{x}_{i+1} \in \mathbb{D}_X} \sum_{y_{i+1} \in \mathbb{D}_Y} p(y_{i+1}|\mathbf{x}_{i+1}, \mathscr{D}_i) \\ &\ldots \qquad (6.10) \\ &\max_{\mathbf{x}_\mu \in \mathbb{D}_X} \sum_{y_\mu \in \mathbb{D}_Y} p(y_\mu|\mathbf{x}_\mu, \mathscr{D}_{\mu-1}) \\ &\sum_{x_k \in \mathscr{D}^X_{test}} \max_{d_k \in \mathbb{D}_D} \sum_{f_\mu + 1 \in \mathbb{D}_F} P(f_{\mu+1}|\mathscr{D}_\mu, T) \sum_{\ddot{y}_k \in \mathbb{D}_Y} p(\ddot{y}_k|\ddot{\mathbf{x}}_k, f_{\mu+1}) \\ &\sum_{o_k \in \mathbb{D}_O} p(o_k|\ddot{\mathbf{x}}_k, \ddot{y}_k, f_{\mu+1}, d_k)\big[U(o_k) - C(\mathscr{D}_\mu \setminus \mathscr{D}_{i-1})\big]\end{aligned}$$

This equation assumes that all testing happens at time step $mu+1$, but if the function remains dynamic during testing, then this may not be the case. Perhaps each test will take place at a new f. If this is the case, then the equations only require us to use $f_{\mu+k}$ for each exploitation time step. This can be done so long as the examples in the test set are sorted, and we can predict which time step they will be used on. Otherwise, a much more complex expectation would have to be done over which examples will be seen at which time step.

Notice from the above equations that the value of information will increase as the size of the test set increases because each sample will have the chance to help the user over more examples from the test set. Samples that would not be worth the cost with small test set sizes can be worth the cost with large test set sizes (see Observation 6.7.0.1).

Inductive Vs. Transductive Active Learning

The above examples were both transductive. We will now demonstrate the difference between transductive and inductive learning. The best way to do this is by demonstrating with an example. The generative, dynamic, *inductive*, off line, active learning case can be seen in

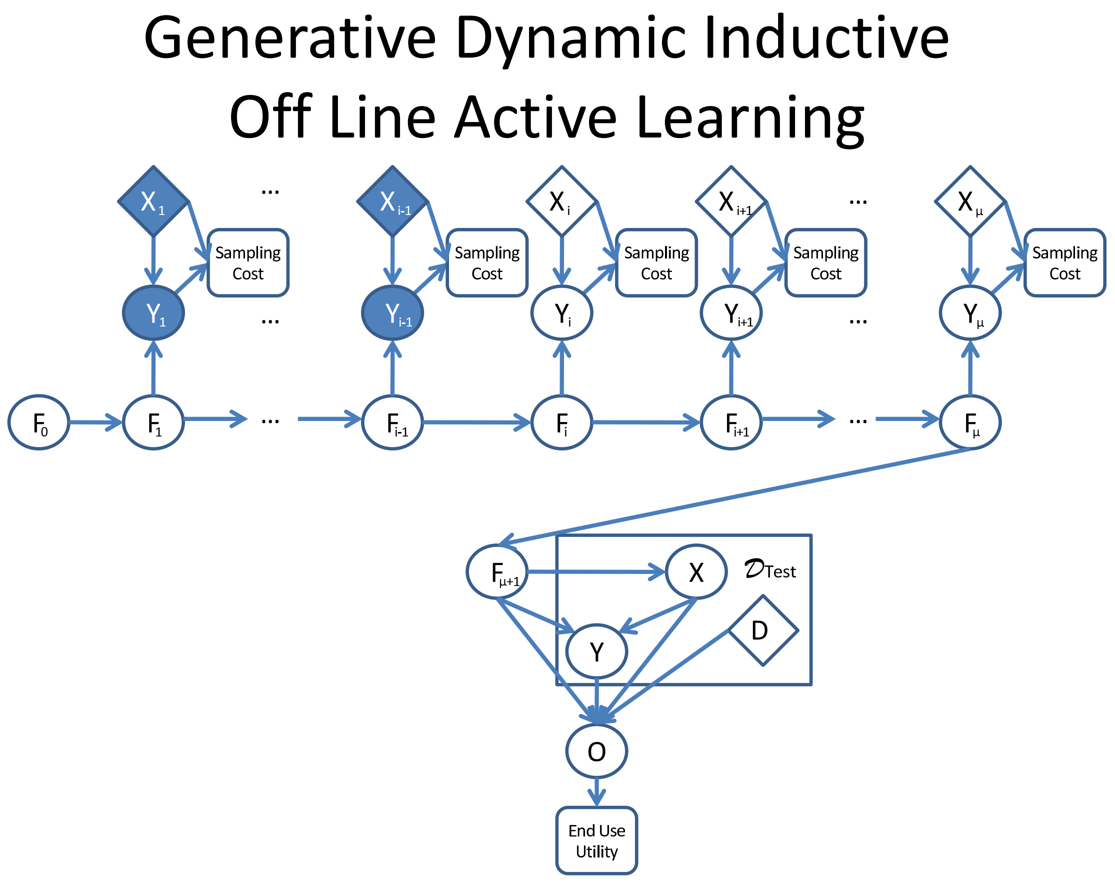


Figure 6.4: The generative, dynamic, inductive, off line active learning case at the ith time step. This figure illustrates the difference between inductive and transductive active learning when compared to Figure 6.3. Notice that in that figure, that $\mathbf{X}$ in the test set is observed, while in this figure the $\mathbf{X}$ in the test set is unobserved.

Figure 6.4. This is identical to the case in Figure 6.3, except that it is inductive instead of transductive, and the difference can readily be seen by comparing these two figures.

The expected utility is computed exactly the same as in the transductive case, except that $\sum_{x_k \in \mathscr{D}^X_{test}}$ changes to $\nu \sum_{x_k \in \mathbb{D}_X} \xi(x_k|f_\mu + 1)$ where ν is the number of times the information will be exploited (the planed size of the test set). This changes equation (6.10) to equation (6.11).

$$
\begin{aligned}
E(U|\mathbf{x}_i, \mathscr{D}_{i-1}, \mu) = & \sum_{y_i \in \mathbb{D}_Y} p(y_i|\mathbf{x}_i, \mathscr{D}_{i-1}) \\
& \max_{\mathbf{x}_{i+1} \in \mathbb{D}_X} \sum_{y_{i+1} \in \mathbb{D}_Y} p(y_{i+1}|\mathbf{x}_{i+1}, \mathscr{D}_i) \\
& \ldots \\
& \max_{\mathbf{x}_\mu \in \mathbb{D}_X} \sum_{y_\mu \in \mathbb{D}_Y} p(y_\mu|\mathbf{x}_\mu, \mathscr{D}_{\mu-1}) \\
& \nu \sum_{x_k \in \mathbb{D}_X} \xi(x_k|f_\mu + 1) \max_{d_k \in \mathbb{D}_D} \sum_{f_\mu + 1 \in \mathbb{D}_F} P(f_{\mu+1}|\mathscr{D}_\mu, T) \sum_{\ddot{y}_k \in \mathbb{D}_Y} p(\ddot{y}_k|\ddot{\mathbf{x}}_k, f_{\mu+1}) \\
& \sum_{o_k \in \mathbb{D}_O} p(o_k|\ddot{\mathbf{x}}_k, \ddot{y}_k, f_{\mu+1}, d_k) \big[U(o_k) - C(\mathscr{D}_\mu \setminus \mathscr{D}_{i-1})\big]
\end{aligned}
\tag{6.11}
$$

In the transductive case, the value of sampling went up with a larger test set, but in inductive learning there is no known test set, so the value of information goes up with the number of times that we plan to be able to exploit the data, ν, (see Observation 6.7.0.1). This means that samples that are not worth their cost for low values of ν, could be worth the cost for high values of ν. This also means that when performing inductive active sampling, some idea of ν is needed. If ν is unknown, it is possible to expect over this value so long as we have some distribution over possible values of ν as follows:

$$
\begin{aligned}
E(U|\mathbf{x}_i, \mathscr{D}_{i-1}, \mu) = &\sum_{y_i \in \mathbb{D}_Y} p(y_i|\mathbf{x}_i, \mathscr{D}_{i-1}) \\
&\max_{\mathbf{x}_{i+1} \in \mathbb{D}_X} \sum_{y_{i+1} \in \mathbb{D}_Y} p(y_{i+1}|\mathbf{x}_{i+1}, \mathscr{D}_i) \\
&\ldots \\
&\max_{\mathbf{x}_\mu \in \mathbb{D}_X} \sum_{y_\mu \in \mathbb{D}_Y} p(y_\mu|\mathbf{x}_\mu, \mathscr{D}_{\mu-1}) \\
&\sum_{\nu_j \in \mathbb{D}_\nu} \nu_j \, p(\nu_j) \sum_{x_k \in \mathbb{D}_X} \xi(x_k|f_\mu + 1) \max_{d_k \in \mathbb{D}_D} \sum_{f_\mu + 1 \in \mathbb{D}_F} P(f_{\mu+1}|\mathscr{D}_\mu, T) \\
&\sum_{\ddot{y}_k \in \mathbb{D}_Y} p(\ddot{y}_k|\ddot{\mathbf{x}}_k, f_{\mu+1}) \sum_{o_k \in \mathbb{D}_O} p(o_k|\ddot{\mathbf{x}}_k, \ddot{y}_k, f_{\mu+1}, d_k) \big[U(o_k) - C(\mathscr{D}_\mu \setminus \mathscr{D}_{i-1})\big]
\end{aligned}
\tag{6.12}
$$

Notice that $\sum_{\nu_j \in \mathbb{D}_\nu} \nu_j \, p(\nu_j)$ just takes the expected value of ν. Thus, active learning requires at least the expected value for the potential number of times the information can be exploited. Many traditional active learning techniques do not explicitly handle this information, and thus make some implicit, and often incorrect, assumptions about ν.

Generative On Line Active Learning

We will now deal with the on line variations of generative active learning. On line means that the agent is tested as it goes. This forces the agent to balance exploration and exploitation at each step since there are no separate exploration vs. exploitation stages. But in the generative case, the agent also selects $\mathbf{x}$ at each time step. There are actually many potential on line situations, but one of the more common cases is where this $\mathbf{x}_i$ selected by the agent is also be the $\mathbf{x}$ where exploitation will happen. In this case, the agent selects the $\mathbf{x}_i$, then selects the max decision d_i for that same $\mathbf{x}_i$ location without knowing y_i, only after the utility and sample costs are counted does the agent get to observe y_i, and this information is now available for the $i + 1$ time step. This case is shown in Figure 6.5. Notice that transductive learning does not really make sense in this situation, since we are selecting all future $\mathbf{x}$, and

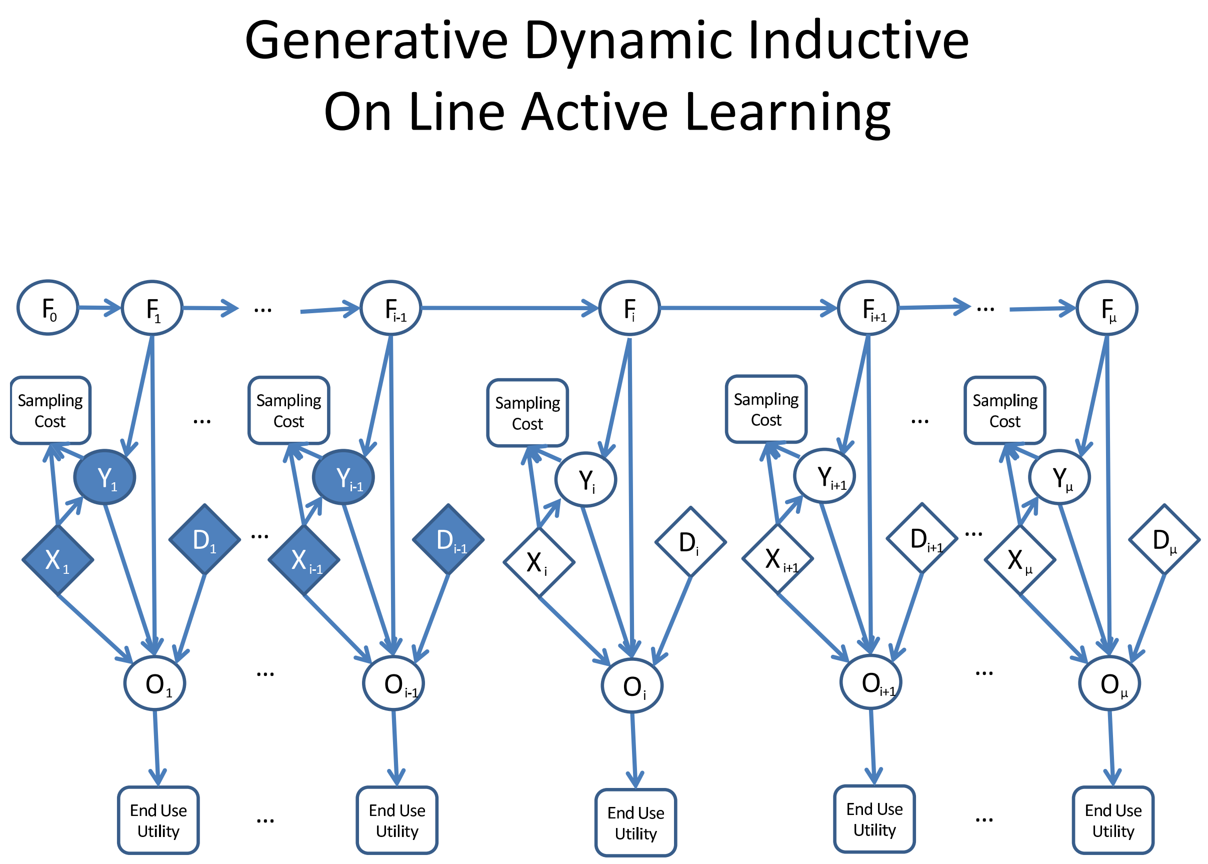


Figure 6.5: The generative dynamic inductive on line active learning case at the ith time step.

it is unclear exactly what would be meant by making them observed. However, notice that since we get to select these future **x** cases, it is possible in some situations for the optimal solution to focus exploration *and* exploitation on specific parts of the function, while leaving the remainder of the function unexplored. Whether this is a viable strategy will depend on the outcome and utility functions. Notice also that in this on line learning situation the number of times we expect to be able to explore, μ, and the number of times we expect to be able to exploit, ν, are the same thing.

For this situation, the maximum expected utility equations are given as follows:

$$
\begin{aligned}
EU(\mathbf{x}_i|\mathscr{D}_{i-1},\mu) = & \max_{d_i} \sum_{f_i} P(f_i|\mathscr{D}_{i-1}) \sum_{y_i} p(y_i|\mathbf{x}_i, f_i) \\
& \Bigg[-C(\mathbf{x}_i, y_i) + \sum_{o_i \in \mathbb{D}_O} p(o_i|\ddot{\mathbf{x}}_i, \ddot{y}_i, d_i, f_i) U(o_i) + \\
& \max_{\mathbf{x}_{i+1}} \max_{d_{i+1}} \sum_{f_{i+1}} P(f_{i+1}|f_i) \sum_{y_{i+1}} p(y_{i+1}|\mathbf{x}_{i+1}, f_{i+1}) \\
& \Bigg[-C(\mathbf{x}_{i+1}, y_{i+1}) + \sum_{o_{i+1} \in \mathbb{D}_O} p(o_{i+1}|\ddot{\mathbf{x}}_{i+1}, \ddot{y}_{i+1}, d_{i+1}, f_{i+1}) U(o_{i+1}) + \\
& \ldots \\
& \max_{\mathbf{x}_\mu} \max_{d_\mu} \sum_{f_\mu} P(f_\mu|f_{\mu-1}) \sum_{y_\mu} p(y_\mu|\mathbf{x}_\mu, f_\mu) \\
& \Bigg[-C(\mathbf{x}_\mu, y_\mu) + \sum_{o_\mu \in \mathbb{D}_O} p(o_\mu|\ddot{\mathbf{x}}_\mu, \ddot{y}_\mu, d_\mu, f_\mu) U(o_\mu) \Bigg] \ldots \Bigg] \Bigg]
\end{aligned}
\tag{6.13}
$$

Another potential "on line" learning situation might take turns exploring with a chosen **x**, and then exploiting on an unobserved (inductive), or observed (transductive) test location. It is worth noting that in this variation "on line transductive" learning makes sense. There is insufficient space to deal with every possible variation, and we will not deal with this variation of "on line" learning here, since it could be constructed by a repeated variation to the off line solution.

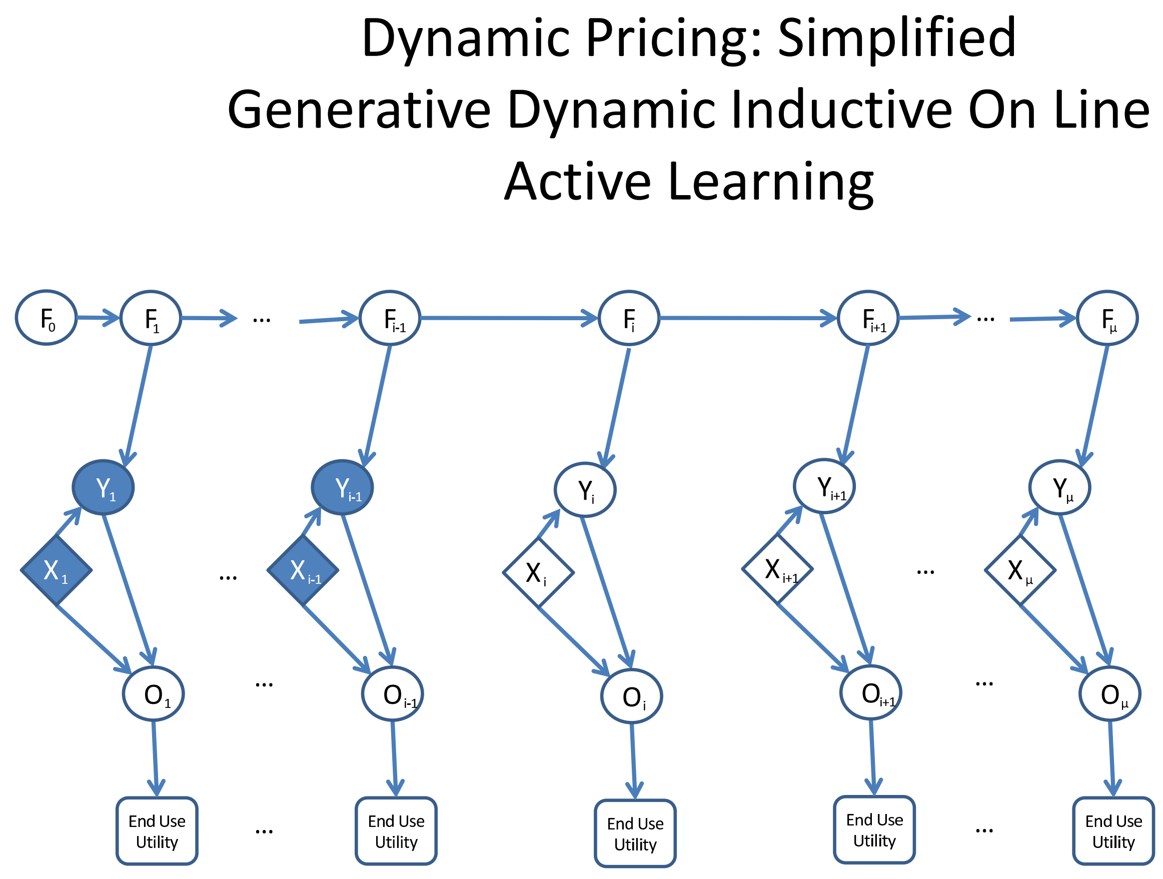

Dynamic Pricing: Simplified Generative Dynamic Inductive On Line Active Learning
F0
F1
Fi-1
Fi
Fi+1
Fμ
Y1
Yi-1
Yi
Yi+1
Yμ
X1
Xi-1
Xi
Xi+1
Xμ
O1
Oi-1
Oi
Oi+1
Oμ
End Use Utility

Figure 6.6: Dynamic Pricing, a special, simplified case of the generative, dynamic, inductive, on line, active learning case, shown at the ith time step. Compare to the standard generative, dynamic inductive, on line, active learning case shown in Figure 6.5.

Dynamic Pricing

One extremely common variation of generative on line active learning is known as "dynamic pricing." The dynamic pricing situation is illustrated by Figure 6.6. Dynamic pricing is identical to the situation of Figure 6.5, except for a few simplifying assumptions. In dynamic pricing, it is assumed that the sample costs are 0. This does not mean that there is no cost for exploration. In dynamic pricing, exploration costs are found in lost opportunity cost which comes from not making the optimal end use utility decision. Another simplifying assumption is that in dynamic pricing there is no separate decision, selecting the sample location *is* the decision that will affect the outcome at that time step. Also, dynamic pricing assumes that the outcome only depends on the sample location, and on the sample outcome at that time step. In dynamic pricing the function is a mapping between the prices for a given day, $\mathbf{x}_i$, and the number of items sold that day, y_i, and is influenced by the current supply and demand, and may change over time (f is dynamic). The outcome o_i is the profit that day, and is influenced by the number of items sold, and the price chosen. The utility then represents some concept of the utility of money. The goal is to set the price each day so as to maximize total utility.

The expected utility for this simplified situation follows:

$$\begin{aligned}
E(U|\mathbf{x}_i, \mathscr{D}_{i-1}, \mu) = \sum_{f_i} P(f_i|\mathscr{D}_{i-1}) \sum_{y_i} p(y_i|\mathbf{x}_i, f_i) \Bigg[\sum_{o_i \in \mathbb{D}_O} p(o_i|\ddot{\mathbf{x}}_i, \ddot{y}_i) U(o_i) + \\
\max_{\mathbf{x}_{i+1}} \sum_{f_{i+1}} P(f_{i+1}|f_i) \sum_{y_{i+1}} p(y_{i+1}|\mathbf{x}_{i+1}, f_{i+1}) \Bigg[\sum_{o_{i+1} \in \mathbb{D}_O} p(o_{i+1}|\ddot{\mathbf{x}}_{i+1}, \ddot{y}_{i+1}) U(o_{i+1}) + \\
\ldots \\
\max_{\mathbf{x}_\mu} \sum_{f_\mu} P(f_\mu|f_{\mu-1}) \sum_{y_\mu} p(y_\mu|\mathbf{x}_\mu, f_\mu) \Bigg[\sum_{o_\mu \in \mathbb{D}_O} p(o_\mu|\ddot{\mathbf{x}}_\mu, \ddot{y}_\mu) U(o_\mu) \Bigg] \ldots \Bigg] \Bigg]
\end{aligned} \tag{6.14}$$

Samples further from the expected optimum tend to have more information, and so sampling away from the expected optimum leads to exploration, but at an expected opportunity cost. Samples at the expected optimum have the highest immediate expected payoff, but often have less exploration value.

The difficulty of dynamic pricing comes from the fact that greedy approximations (approximations that assume $\nu = \mu = 1$) almost always vastly under explore [18]. This happens because such approximations do not take into account the total number of future steps where information could be exploited. Yet, non-greedy approximations are extremely computationally intense. Also, determining how long information gained will be useful depends on correctly modeling how quickly f can change over time. If f changes rapidly, then exploration now will have limited usefulness any time into the future; however, if f changes relatively slowly, then exploration now can provide huge payoffs far into the future. One potential solution to this problem designed by Mullen (who's work is related to ours) involved simulated runs into the future, (which can now be seen as an approximate sampling approach to the model presented here). This approach has proved highly effective [82, 83, 84, 85].

6.5.2 Pool Based Active Learning

Recall from Section 6.3.2, that pool based sampling involves a pool of sample locations with labels that can be revealed for some sample cost. As examples are taken from this "pool set," they are removed so that the pool set for the future time steps is a smaller subset of the initial pool set. In pool based sampling, the potential sample locations are restricted to the pool set, while in generative sampling the potential sample locations are unrestricted.

It is difficult to accurately represent pool based active learning as a graphical model because the pool set is changing over time. At each time step, the pool set is the initial pool set with the previous samples removed. Thus, pool based active learning involves a much more complex graphical model that did generative sampling (see Figure 6.7), however, the difference between the utility equations for generative sampling vs. pool based sampling is rather small. The maximum expected utility of taking $\mathbf{x}_i$ from the pool set at time i in the Pool Based, Dynamic, Transductive, Off-Line, Active Learning case is:

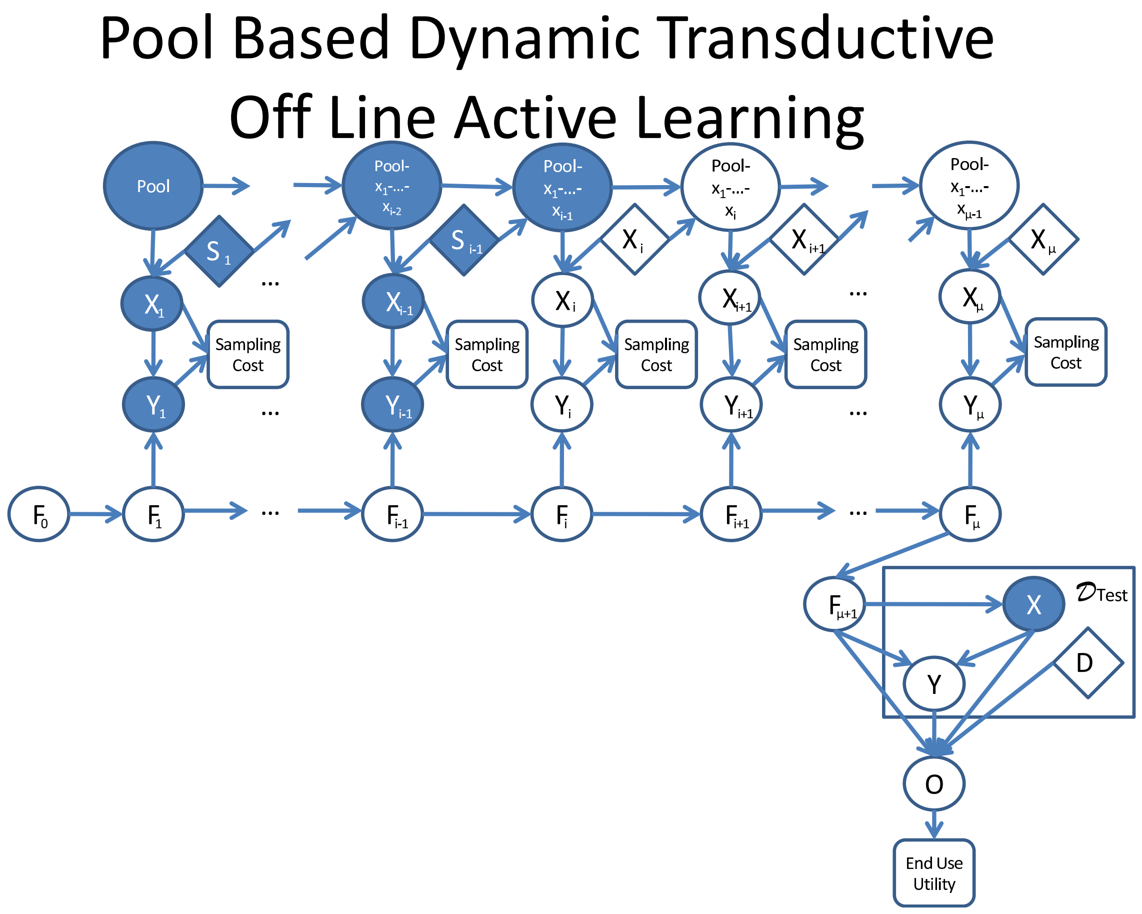


Figure 6.7: The pool based dynamic transductive off line active learning case, shown at the ith time step.

$$
\begin{aligned}
E(U|\mathbf{x}_i, \mathscr{D}_{i-1}, \mu) = & \sum_{y_i \in \mathbb{D}_Y} p(y_i|\mathbf{x}_i, \mathscr{D}_{i-1}) \\
& \max_{\mathbf{x}_{i+1} \in \mathscr{D}^X_{pool_{i+1}}} \sum_{y_{i+1} \in \mathbb{D}_Y} p(y_{i+1}|\mathbf{x}_{i+1}, \mathscr{D}_i) \\
& \dots \\
& \max_{\mathbf{x}_\mu \in \mathscr{D}^X_{pool_\mu}} \sum_{y_\mu \in \mathbb{D}_Y} p(y_\mu|\mathbf{x}_\mu, \mathscr{D}_{\mu-1}) \\
& \sum_{x_k \in \mathscr{D}^X_{test}} \max_{d_k \in \mathbb{D}_D} \sum_{f_\mu+1 \in \mathbb{D}_F} P(f_{\mu+1}|\mathscr{D}_\mu, T) \sum_{\ddot{y}_k \in \mathbb{D}_Y} p(\ddot{y}_k|\ddot{\mathbf{x}}_k, f_{\mu+1}) \\
& \sum_{o_k \in \mathbb{D}_O} p(o_k|\ddot{\mathbf{x}}_k, \ddot{y}_k, f_{\mu+1}, d_k)\big[U(o_k) - C(\mathscr{D}_\mu \setminus \mathscr{D}_{i-1})\big]
\end{aligned}
\tag{6.15}
$$

Notice that this is identical to the generative case (see Figure 6.10), except that **x** is summed over the pool set for that time period instead of over the entire domain for **X**. Thus, it is trivially possible to modify any of the above generative situations to their pool based form in a similar manner.

There are several examples of when the pool based active learning scenario might be applicable. First, it could be used as an approximation to generative sampling. True generative sampling is often very difficult because it involves searching over an infinite or nearly infinite set of potential sample locations. In this sense, pool based sampling is significantly simpler, since it only has to search over the examples in the pool set. One common approximation to generative sampling is to randomly select a new reasonably sized pool set at each time step, and then to perform pool based sampling instead. Thus, pool based learning is often used to approximate generative learning, but with a new pool set at each time step. This differs from true pool based sampling which uses the same pool set at each time step, with the previously sampled locations removed.

Another example of pool based learning (in this case, a transductive on-line pool based active learner) would be a situation similar to our oil well example. Potential oil fields would serve as a pool. The goal is to select from the pool, and then to query an imperfect oracle for a soil sample test, then the decision is whether to drill on that field or not.

Another example might involve a food taster. Again this example is transductive, but off line. In this case, the pool consists in the food prepared for some sovereign, and the food taster selects from the pool some food items to sample for the possible cost of the death or sickness of the taster. Then, if the taster does not get sick, the food can be given to the sovereign with added confidence that he will not get sick.

Another pool based learning situation might involve training our mushroom classification agent. The pool might consist of a bucket of mushrooms. The agent can select a mushroom from this pool, and for some cost have its chemical composition tested for toxins. The goal is to actively select the mushrooms from the bucket in such a way as to optimally learn the mushroom toxicity function while limiting the costs from chemical analysis.

A final example might involve providing part of speech tags for a corpus. The given corpus is the pool set, and the agent selects examples from the pool set to give to a human to tag for some cost (this is the case we will more fully explore in Chapter 9).

Notice from the above examples, that in some cases the pool set and the test set are the same; for example, the food taster's pool set is also the test set, and the corpus pool set is also the final corpus that must be tagged. On the other hand, the mushroom pool set is not the same as the final pool set that the learner will eventually be tested on. This difference is important. We call this having a "queryable test set" if the pool set and the test set are the same, and an "un-queryable test set" or a "locked test set" if the pool set and the test set are not the same. This difference can substantially change the behavior of the algorithm, especially if the oracle provides perfect information, since each query could then guarantee a correct classification on that corresponding test set example if the test set is queryable. Whereas, if the test set is not queryable, then samples from the pool set only have a probabilistic chance of helping to classify the test set.

6.5.3 Stream Based Active Learning

Recall from Section 6.3.2, that stream based sampling involves the situation where potential samples are presented to the agent one at a time, and the agent must decide whether to classify the example itself, or to request help from an oracle. The agent can not go back in time to earlier examples in the stream, nor can the agent skip ahead to future samples. Stream based sampling is sometimes called "Query Filtering" [70].

One potential advantage of stream based active learning is that the stream is often distributed according to the sampling distribution, thus bestowing some of the advantages of strong passive sampling. This can be advantageous, especially in inductive situations, since it prevents the agent from sampling in areas that will not show up during exploitation [70].

For stream based learning we will write: $E(U|\mathscr{T}_i, \mathscr{D}_{i-1}, \mathscr{D}^X_\mu, \mu)$ as the expected net utility of querying the oracle at time i, written $\mathscr{T}_i$, given the data seen so far $\mathscr{D}_{i-1}$ (which includes the current sample location), and optionally, if transductive, given $\mathscr{D}^X_\mu$ (all the future values in the stream), and given the number of elements in the stream μ. Then $E(U|\mathscr{F}_i, \mathscr{D}_{i-1}, \mathscr{D}^X_\mu, \mu)$ represents the same net expected utility if the algorithm does not query the oracle at time i, written $\mathscr{F}_i$. The other effect of querying is that the results of the oracle query are then made available at the next time steps (changing $\mathscr{D}_i$ used at the next time step).

Off Line Stream Based Active Learning

The stream based, dynamic, transductive, off line, active learning case can be seen in Figure 6.8. The net expected utility equations are as follows:

$$E(U|\mathscr{T}_i, \mathscr{D}_{i-1}, \mathscr{D}^X_\mu, \mu) = \sum_{y_i \in \mathbb{D}_Y} p(y_i|\mathbf{x}_i, \mathscr{D}_{i-1}) max \Big[E(U|\mathscr{T}_{i+1}, \mathscr{D}_{i-1} \cup \{\mathbf{x}_i, y_i\}, \mathscr{D}^X_\mu, \mu), \quad (6.16)$$
$$E(U|\mathscr{F}_{i+1}, \mathscr{D}_{i-1} \cup \{\mathbf{x}_i, y_i\}, \mathscr{D}^X_\mu, \mu) \Big] - C(\mathscr{T}_i),$$

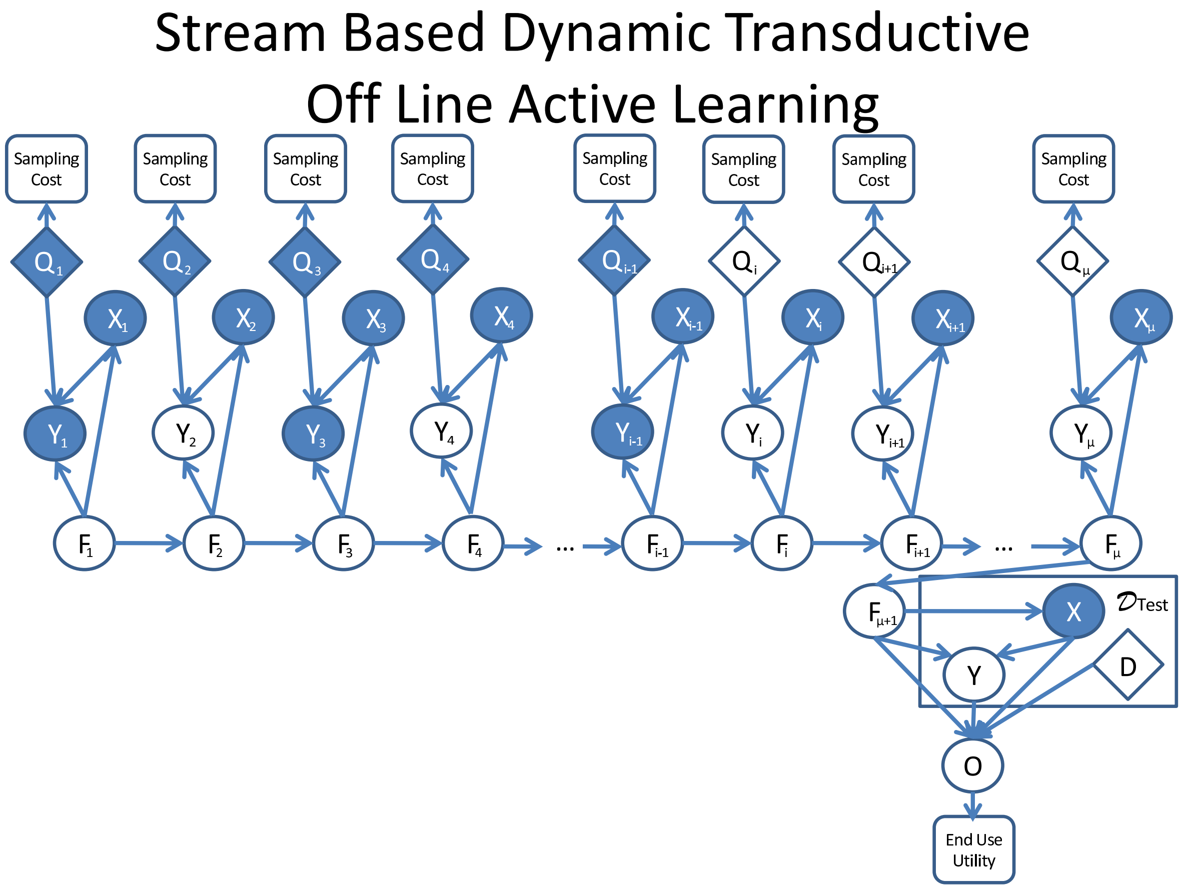


Figure 6.8: The stream based, dynamic, transductive, off line, active learning case at time i. The Q nodes represent a query decision. If the agent chooses for Q to be true, then for some cost the Y node becomes observed, and if Q is false, then the Y node is unobserved. In this figure all odd queries were true, while the even queries were false.

and

$$E(U|\mathscr{F}_i, \mathscr{D}_{i-1}, \mathscr{D}_{\mu}^X, \mu) = \sum_{y_i \in \mathbb{D}_Y} p(y_i|\mathbf{x}_i, \mathscr{D}_{i-1}) max \Big[E(U|\mathscr{T}_{i+1}, \mathscr{D}_{i-1} \cup \{\mathbf{x}_i\}, \mathscr{D}_{\mu}^X, \mu), \quad E(U|\mathscr{F}_{i+1}, \mathscr{D}_{i-1} \cup \{\mathbf{x}_i\}, \mathscr{D}_{\mu}^X, \mu) \Big], \tag{6.17}$$

and when $i = \mu + 1$ then

$$\begin{aligned} E(U|\mathscr{T}_{\mu+1}, \mathscr{D}_{\mu}, \mathscr{D}_{\mu}^X, \mu) =& E(U|\mathscr{F}_{\mu+1}, \mathscr{D}_{\mu}, \mathscr{D}_{\mu}^X, \mu) = \\ & \sum_{x_k \in \mathscr{D}_{test}^X} \max_{d_k \in \mathbb{D}_X} \sum_{f_{\mu+1} \in \mathbb{D}_F} P(f_{\mu+1}|f_{\mu}, T) \sum_{\ddot{y}_k \in \mathbb{D}_Y} p(\ddot{y}_k|\ddot{\mathbf{x}}_k|f_{\mu+1}) \\ & \sum_{o_k \in \mathbb{D}_O} p(o_k|\ddot{\mathbf{x}}_k, \ddot{y}_k, f_{\mu+1}, d_k) U(o_k). \end{aligned} \tag{6.18}$$

To make the above inductive instead of transductive, the future $\mathbf{x}$ values in the stream become unobserved, and we simply compute $E(U|\mathscr{T}_i \,\text{or}\, \mathscr{F}_i, \mathscr{D}_{i-1}, \mu)$ instead. Then it would be necessary to add an extra expectation over all possible future values in the stream. The details of this case are left up to the reader.

On Line Stream Based Active Learning

In order to make stream based learning "on line," end use decisions must be made at each time step. Then the agent either classifies the example on its own, and then makes an end use decision, receiving utility, or else it requests help from the oracle for some cost, then making the end use decision given the information from the oracle. When this sort of on line stream based sampling is inductive, it is sometimes called "on-site" learning [70].

There are many examples of this sort of on line stream based learning. For example, as parts move through an assembly line, a computer vision system might classify the parts as "passing" or "defective," or, for some cost, the system could ask a human to look at the part in question. The information gained from the human inspection actually provides useful

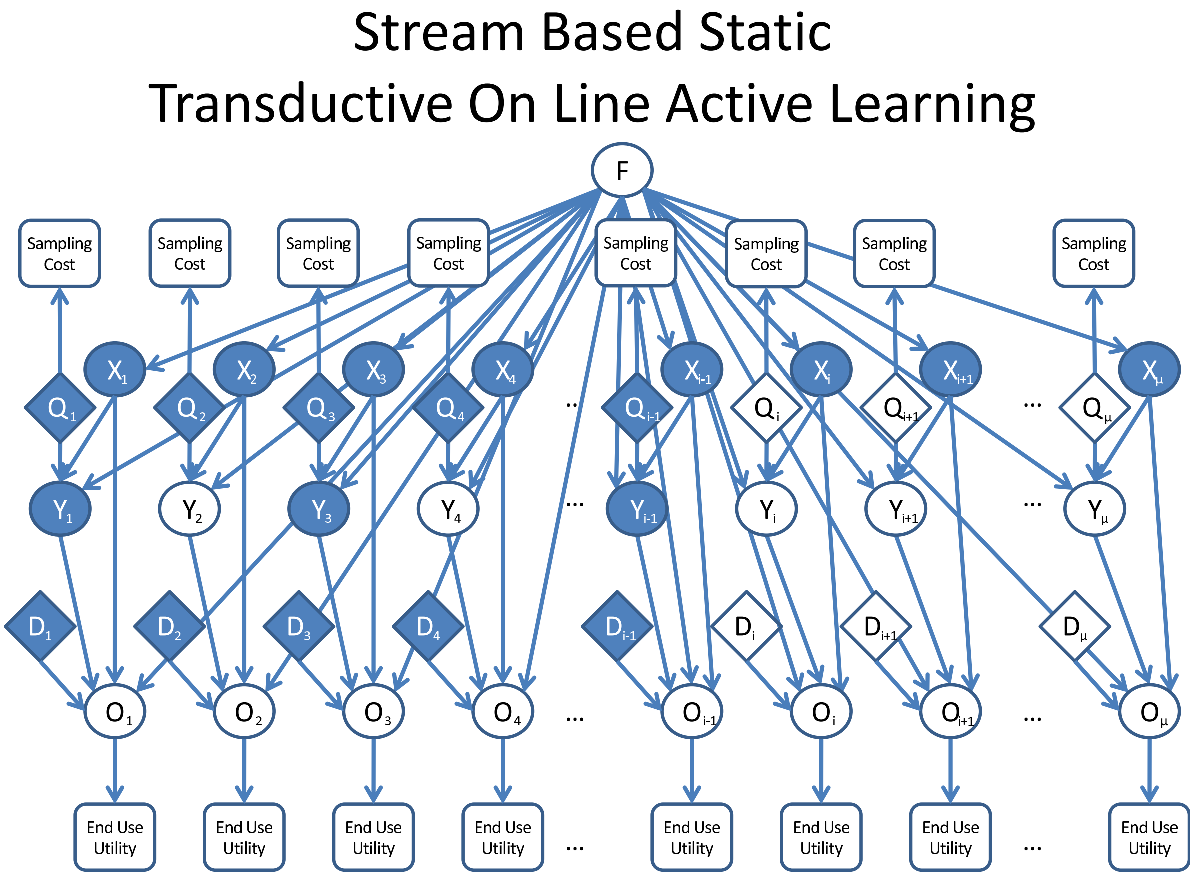


Figure 6.9: The stream based, static, transductive, on line, active learning case. The Q nodes represent a query decision. If the agent chooses for Q to be true, then for some cost the Y node becomes observed, and if Q is false, then the Y node is unobserved. In this figure all odd queries were true, while the even queries were false.

information to help train the agent, making it more accurate in future evaluations. At each time step, the actions of the agent would be to either reject the part, sending it back to be scrapped and recycled, or the agent could pass the part, in which case it would be sent through the assembly line and either shipped, or used as a sub-part for a larger work.

The stream based, static, transductive, on line, active learning situation is shown in Figure 6.9. For this case, the net expected utility equations are:

$$
\begin{aligned}
&E(U|\mathscr{T}_i, \mathscr{D}_{i-1}, \mathscr{D}_\mu^X, \mu) = \\
&\sum_{y_i \in \mathbb{D}_Y} p(y_i|\mathbf{x}_i, \mathscr{D}_{i-1}) \max_{d_i \in \mathbb{D}_D} \sum_{f \in \mathbb{D}_F} P(f|\mathscr{D}_{i-1} \cup \{\mathbf{x}_i, y_i\}) \sum_{o_i} p(o_i|d_i, f, \mathbf{x}_i, y_i) U(o_i) \\
&max\Big[E(U|\mathscr{T}_{i+1}, \mathscr{D}_{i-1} \cup \{\mathbf{x}_i, y_i\}, \mu), E(U|\mathscr{F}_{i+1}, \mathscr{D}_{i-1} \cup \{\mathbf{x}_i, y_i\}, \mu)\Big] - C(\mathscr{T}_i),
\end{aligned} \tag{6.19}
$$

and

$$
\begin{aligned}
&E(U|\mathscr{F}_i, \mathscr{D}_{i-1}, \mathscr{D}_\mu^X, \mu) = \\
&\max_{d_i \in \mathbb{D}_D} \sum_{f \in \mathbb{D}_F} P(f|\mathscr{D}_{i-1} \cup \{\mathbf{x}_i\}) \sum_{y_i \in \mathbb{D}_Y} p(y_i|\mathbf{x}_i, f) \sum_{o_i} p(o_i|d_i, f, \mathbf{x}_i, y_i) U(o_i) \\
&max\Big[E(U|\mathscr{T}_{i+1}, \mathscr{D}_{i-1} \cup \{\mathbf{x}_i\}, \mu), E(U|\mathscr{F}_{i+1}, \mathscr{D}_{i-1} \cup \{\mathbf{x}_i\}, \mu)\Big],
\end{aligned} \tag{6.20}
$$

and when $i = \mu + 1$ then

$$
E(U|\mathscr{T}_{\mu+1}, \mathscr{D}_\mu, \mathscr{D}_\mu^X, \mu) = E(U|\mathscr{F}_{\mu+1}, \mathscr{D}_\mu, \mathscr{D}_\mu^X, \mu) = 0. \tag{6.21}
$$

Notice that these equations maximize d at a different time depending on whether the oracle query was made or not. If the oracle query is to be made, then there is an expectation over the result of the oracle query (the class), with a maximization step after the results are known. If the oracle is not queried, then there is a decision that must be made, and then the result of that decision depends on an expectation over the class. In both cases, there is an expectation over the class, and a maximum over the decision, but in the case where the oracle is queried, the maximization is made after the results of the expectation are known, while if the oracle is not queried, then the maximization must take place before the expectation. Naturally, the other result of querying is that the result of the query is added to the information that will be available at the next time step.

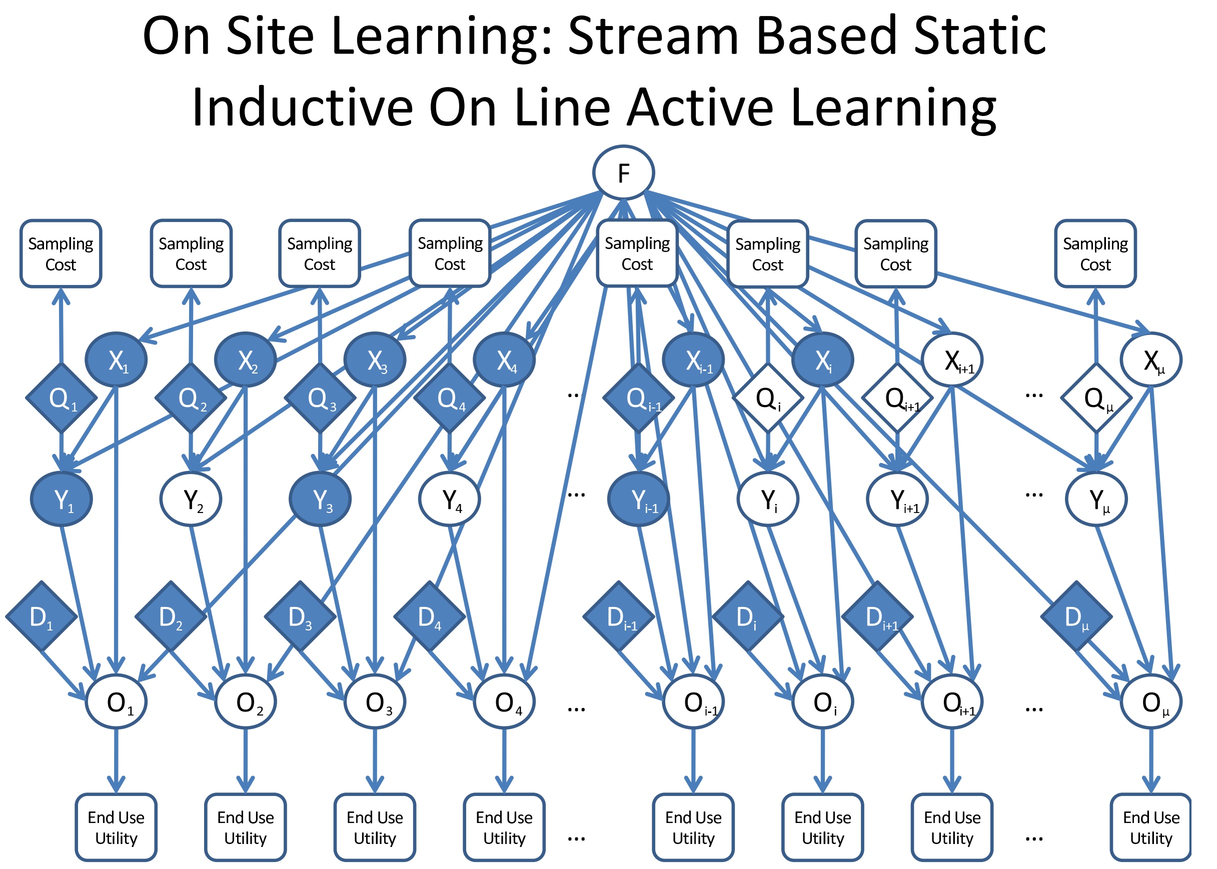

On Site Learning: Stream Based Static Inductive On Line Active Learning
F
Sampling Cost
X_1 X_2 X_3 X_4 … X_{i-1} X_i X_{i+1} … X_μ
Q_1 Q_2 Q_3 Q_4 … Q_{i-1} Q_i Q_{i+1} … Q_μ
Y_1 Y_2 Y_3 Y_4 … Y_{i-1} Y_i Y_{i+1} … Y_μ
D_1 D_2 D_3 D_4 D_{i-1} D_i D_{i+1} D_μ
O_1 O_2 O_3 O_4 … O_{i-1} O_i O_{i+1} … O_μ
End Use Utility

Figure 6.10: On Site Learning, the stream based, static, inductive, on line, active learning case (for the difference between inductive and transductive in the on line stream based case, compare Figure 6.9). The Q nodes represent a query decision. If the agent chooses for Q to be true, then for some cost the Y node becomes observed, and if Q is false, then the Y node is unobserved. In this figure all odd queries were true, while the even queries were false.

In the stream based on line learning situation, the differences between transductive and inductive learning can be seen by comparing Figure 6.9 to Figure 6.10. In transductive learning the future stream's $\mathbf{x}$ are observed, while they are not observed in the inductive case. The inductive case is sometimes called "On Site Learning," [70]. As with the off line case, the inductive expected utility equations involve dropping $\mathscr{D}_\mu^X$, and adding an expectation over the future $\mathbf{x}$ values based on the sampling distribution.

It is also possible to use a dynamic function in on line stream based learning. The stream based, dynamic, transductive, on line, active learning situation is shown in Figure 6.11. The net expected utility equations for this case are:

$$\begin{aligned}
&E(U|\mathscr{T}_i, \mathscr{D}_{i-1}, \mathscr{D}_\mu^X, \mu) = \\
&\sum_{y_i \in \mathbb{D}_Y} p(y_i|\mathbf{x}_i, \mathscr{D}_{i-1}) \max_{d_i \in \mathbb{D}_D} \sum_{f_i \in \mathbb{D}_F} P(f_i|\mathscr{D}_{i-1} \cup \{\mathbf{x}_i, y_i\}) \sum_{o_i \in \mathbb{D}_O} p(o_i|d_i, f_i, \mathbf{x}_i, y_i) U(o_i) \\
&max\Big[E(U|\mathscr{T}_{i+1}, \mathscr{D}_{i-1} \cup \{\mathbf{x}_i, y_i\}, \mu), E(U|\mathscr{F}_{i+1}, \mathscr{D}_{i-1} \cup \{\mathbf{x}_i, y_i\}, \mu)\Big] - C(\mathscr{T}_i),
\end{aligned} \tag{6.22}$$

and

$$\begin{aligned}
&E(U|\mathscr{F}_i, \mathscr{D}_{i-1}, \mathscr{D}_\mu^X, \mu) = \\
&\max_{d_i \in \mathbb{D}_D} \sum_{f_i \in \mathbb{D}_F} P(f_i|\mathscr{D}_{i-1} \cup \mathbf{x}_i) \sum_{y_i \in \mathbb{D}_Y} p(y_i|\mathbf{x}_i, f_i) \sum_{o_i} p(o_i|d_i, f_i, \mathbf{x}_i, y_i) U(o_i) \\
&max\Big[E(U|\mathscr{T}_{i+1}, \mathscr{D}_{i-1} \cup \{\mathbf{x}_i\}, \mu), E(U|\mathscr{F}_{i+1}, \mathscr{D}_{i-1} \cup \{\mathbf{x}_i\}, \mu)\Big],
\end{aligned} \tag{6.23}$$

and when $i = \mu + 1$ then

$$E(U|\mathscr{T}_{\mu+1}, \mathscr{D}_\mu, \mathscr{D}_\mu^X, \mu) = E(U|\mathscr{F}_{\mu+1}, \mathscr{D}_\mu^X, \mathscr{D}_\mu, \mu) = 0. \tag{6.24}$$

These equations look very similar to the static case, but now $P(f_i|\mathscr{D}_{i-1})$ is a marginal found by marginalizing f_{i-1} out of the equation as shown in Equations 6.5 to 6.8. In order to make

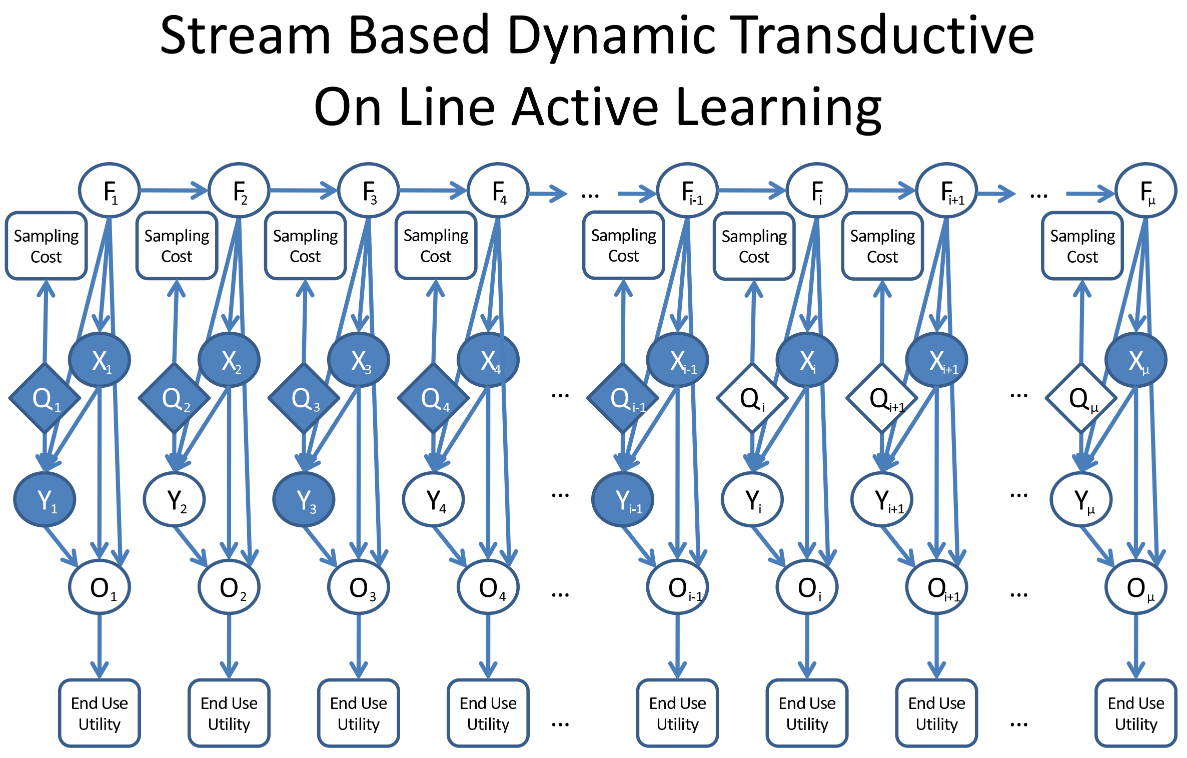

Stream Based Dynamic Transductive
On Line Active Learning
F1 F2 F3 F4 ... Fi-1 Fi Fi+1 ... Fμ
Sampling Cost
X1 X2 X3 X4 ... Xi-1 Xi Xi+1 ... Xμ
Q1 Q2 Q3 Q4 ... Qi-1 Qi Qi+1 ... Qμ
Y1 Y2 Y3 Y4 ... Yi-1 Yi Yi+1 ... Yμ
O1 O2 O3 O4 ... Oi-1 Oi Oi+1 ... Oμ
End Use Utility

Figure 6.11: Stream based, dynamic, transductive, on line, active learning case (for the difference between static and dynamic functions in the on line stream based case, compare Figure 6.9). The Q nodes represent a query decision. If the agent chooses for Q to be true, then for some cost the Y node becomes observed, and if Q is false, then the Y node is unobserved. In this figure all odd queries were true, while the even queries were false.

the dynamic version inductive, we would make the same changes that we did in the static version.

6.6 Optimal Pedagogical Sampling

Recall from Section 6.3.5 that pedagogical sampling involves the situation where a teacher who knows the true function better than the student, actively selects the samples that the teacher believes will be of the most use to aid the student's learning of the function.

Pedagogical sampling works because the teacher knows the function better than the student. When modeling pedagogical sampling, we will make the simplifying assumption that the teacher perfectly knows the true function, f^{tch}, while the student does not. Thus F is an observed random variable form the teacher's perspective and an unobserved random variable from the learner's perspective (in situations where the teacher does not know the true function perfectly, but does know it better than the student, then it is possible to put a distribution over the teacher's belief about f^{tch}, which would change the equations that follow by placing an expecting over possible f^{tch}). We will represent the student's belief about the function as f^{stu}, which is influenced by the student's prior $P(f^{stu})$, and the data presented to the student, creating the student's posterior $P(f^{stu}|\mathscr{D}_\mu)$.

Just as with active learning, there are many variations of pedagogical sampling. Pedagogical learning can operate on discrete or continuous functions, with mixed or pure strategies, with generative, pool based, or stream based selection techniques, with dynamic or static functions, it can be transductive or inductive, and it can be on line or off line. Thus, we could repeat all the cases we did for active learning again for pedagogical learning. However, in the interest of space, we will just give a single example of pedagogical learning. We will present the generative, static, transductive, off line, pedagogical learning case. We believe that with this example, it should be possible for the reader to modify the other cases from the active learning section to make them pedagogical.

There have been very few attempts to mathematically model the pedagogical sampling procedure. One exception is Patrick Shafto and Noah Goodman [104], who showed only that there is no single optimal solution, because the optimal way to teach depends on the way the student will learn, and the best way for the student to learn depends on the way in which the teacher is teaching. Thus, the solution to the pedagogical learning situation must involve a multiagent fixed point solution. Our approach is a large step forward, since their approach only modeled the most basic elements of pedagogical sampling, whereas teaching situations such as traditional educational lectures and apprenticeship teaching can both be modeled in our approach. Some of the differences between the two teaching paradigms can be mathematically formalized in this way. In our framework, the traditional educational lectures approach involve a generative, static or dynamic, inductive or transductive, off line, pedagogical learning situation. On the other hand, the older apprenticeship teaching approach involves a stream based, static or dynamic, inductive or transductive, on line pedagogical learning situation.

Some intangibles such as student motivation are not captured by our mathematical model. However, there is good reason to suppose that the on line nature of an apprenticeship provides “legitimate peripheral participation” which has been shown to be beneficial to student motivation and which fosters a better learning environment [54].

The differences between pedagogical sampling and active sampling do not show up readily in a graphical model, since the differences involve who is making the decisions, and what they know at the time they make them. In active learning the agent is making both the sampling and the end use decisions. In pedagogical learning the teacher who knows the function is making the sampling decisions while the student (who does not know the function) is still making the end use decisions. Thus, we will not repeat the graphical models for pedagogical learning.

As an example, we are using the generative, static, transductive, off line, pedagogical learning situation, one of the models that can represent a traditional lecturing approach to teaching. In this case the expected utility equations are:

$$
\begin{aligned}
E(U|\mathbf{x}_i, \mathscr{D}_{i-1}, \mu) = &\sum_{y_i \in \mathbb{D}_Y} p(y_i|\mathbf{x}_i, f^{tch}) \\
&\max_{\mathbf{x}_{i+1} \in \mathbb{D}_X} \sum_{y_{i+1} \in \mathbb{D}_Y} p(y_{i+1}|\mathbf{x}_{i+1}, f^{tch}) \\
&... \\
&\max_{\mathbf{x}_\mu \in \mathbb{D}_X} \sum_{y_\mu \in \mathbb{D}_Y} p(y_\mu|\mathbf{x}_\mu, f^{tch}) \\
&\sum_{x_k \in \mathscr{D}^X_{test}} \phi(d^{stu}_k|\mathbf{x}_k, \mathscr{D}_\mu) \sum_{\ddot{y}_k \in \mathbb{D}_Y} p(\ddot{y}_k|\ddot{\mathbf{x}}_k, f^{tch}) \\
&\sum_{o_k \in \mathbb{D}_O} p(o_k|\ddot{\mathbf{x}}_k, \ddot{y}_k, f^{tch}, d^{stu}_k) U(o_k)
\end{aligned} \tag{6.25}
$$

To compare this equation to its active learning equivalent, compare to Equation 6.9. In the above equation, $\phi(d^{stu}_k|\mathbf{x}_k, \mathscr{D}_\mu)$ represents the teacher's belief that the student will make decision d^{stu}_k in situation $\mathbf{x}_k$, given the data set $\mathscr{D}_\mu$. Notice that our model reveals that the best way to teach depends (unsurprisingly) on the teacher modeling how he or she thinks that the student will behave given the instruction. If the teacher assumes that the student will update their prior rationally (where rationally means according to Bayes Law), and if the teacher knows (or can estimate) the student's prior, then the teacher can often approximate the $\phi(d^{stu}_k|\mathbf{x}_k, \mathscr{D}_\mu)$ that would result from an optimal rational student. It is also possible for the teacher to assume that the student is irrational when determining $\phi(d^{stu}_k|\mathbf{x}_k, \mathscr{D}_\mu)$.

However, recall that Patrick Shafto and Noah Goodman showed that the best way for the student to learn depends on the way the student believes that the teacher is teaching [104]. Specifically, the distribution of samples given by the teacher can tell the student something about the function through the correlation between f^{tch} and the samples $\dot{\mathbf{x}}$ given to the student by the teacher (and seen in the UBM's model of un-supervised or semi-supervised learning). Thus, by analyzing the teacher's sampling distribution, it is possible for the student to infer something more about the function. Yet if the student assumes that the teacher is rational, then the teacher's sampling distribution is related to the true function according to Equation 6.25. And thus the student's learning behavior should depend on the teacher's sampling behavior, which should depend in turn upon the student's learning behavior. It is through this relationship that the results of Patrick Shafto and Noah Goodman [104] are manifest in the PL-UBDTM. This creates a typical multiagent dilemma, since the best way to teach depends on how the student will respond to the teaching, and the best way to respond to the teaching depends on the way in which the teacher is teaching. In fact, it has been shown that there can be multiple "best" solutions to these sorts of multiagent learning situations [104].

6.7 Active Learning Theory

Now that we have formalized the selective sampling process, we will focus on active learning, and discuss some theoretical implications to our approach. The optimal solutions developed above will be key to this theoretical analysis. For example, it is now clear that there are several pieces of information that are necessary in order to solve the active learning equations. These include: the number of times that the information will be exploited (ν or the size of the pool set); the number of samples that will be taken μ; either a known test set or a known sampling distribution $\xi(\ddot{\mathbf{x}}|f)$; a prior $P(f)$; the full posterior predictive $p(y|\mathbf{x}, \mathscr{D}_{train})$; knowledge about the sample costs for gathering a set of data $C(\mathscr{D}_{ata})$; and finally, a model of end use including the domains of U and O, the distribution $p(o|y, d)$, the utility function $U(o)$. This list indicates exactly what issues can theoretically affect active learning, while the model tells us *how* they each affect active learning.

Observation 6.7.0.1: *From equations 6.9 to 6.24 it can be seen that active learning depends on the number of times the information can be exploited.*

For inductive active learning this is represented by ν, and for transductive active learning this is represented by the size of the test set. For stream based active learning $\nu = \mu$. In some online dynamic situations this can effectively depend on how quickly the function is changing, embodied by T.

One way to understand Observation 6.7.0.1 is by noting that doubling ν effectively doubles the value of information. Information that might have appeared too expensive with low values for ν can be worth sampling with higher values for ν. Thus the optimal active learning approach will depend directly on ν.

Multiplication by a constant does not change the relative ordering, and thus the ν term does not change the relative ordering of the value of information (VOI) of the experiments. However, when combining the value of information with sample cost (computing the *net* value of information or the expected utility of sampling), this multiplicative term can change the optimal test to perform. The active learner will be more willing to select an expensive test with more information if it expects to be able to exploit that information more times.

Example 6.7.1: Assume that one expert e_1 can label mushrooms as poisonous with a higher accuracy than the other expert e_2. Also assume that e_1 costs more than e_2, yet both do provide some value of information. For sufficiently low values of ν it can be more cost effective to use expert e_2, while for sufficiently high values of ν, it can be more cost effective to pay more for e_1.

Example 6.7.2: Perhaps the best example of the effect of T on the number of times data can be exploited can be seen in dynamic pricing. There is some opportunity cost associated with exploration, and some benefit associated with exploitation, and one of the largest challenges in dynamic pricing is to balance exploration with exploitation. Because the function is dynamic and constantly changing, the number of times the information gained in one time step can be effectively exploited in subsequent time steps is unknown and depends upon the rate at which the function is changing over time. Techniques that assume a single time step of exploitation consistently under explore. Yet it is also possible to set ν too high, in which case the algorithm

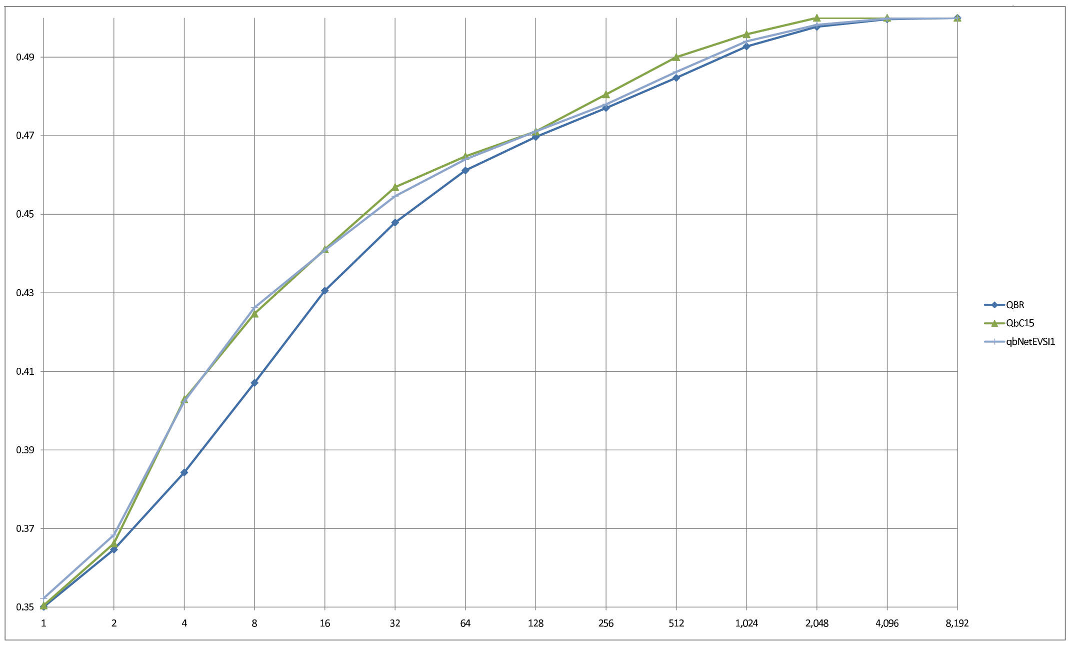


Figure 6.12: QBC verses Greedy Evsi on the gfqbc function.

will over explore [18]. The correct approach is to model T, and then allow the active learning equations to determine the correct balance between exploration and exploitation [82][85][84].

Observation 6.7.0.2: *From equations 6.9 to 6.24 it can be seen that the expected value of getting the first sample can change depending on the number of subsequent samples that will be taken,* μ.

Example 6.7.3: One example of this effect can be seen with a simple discrete benchmark function known as gfqbc, short for "good for query by committee." This function has five discrete values in the domain and the range has three discrete classes, and a uniform sample cost. This simple function was chosen to be simple enough to explore the behavior of the full EVSI equations for reasonable values of μ. When μ was set to 1 (the greedy assumption) sampling based on EVSI did well at first. However, after about 256 samples, QBC (with a committee size of 15), began to outperform Greedy EVSI (see Figure 6.12).

After exploring the reasons for this, it was found that after a certain number of samples, $EVSI(\mathbf{x}_T, \mu = 1) = 0 - C(\mathbf{x}_T)$ for all values of $\mathbf{x}_T$. If no single sample is sufficient to change

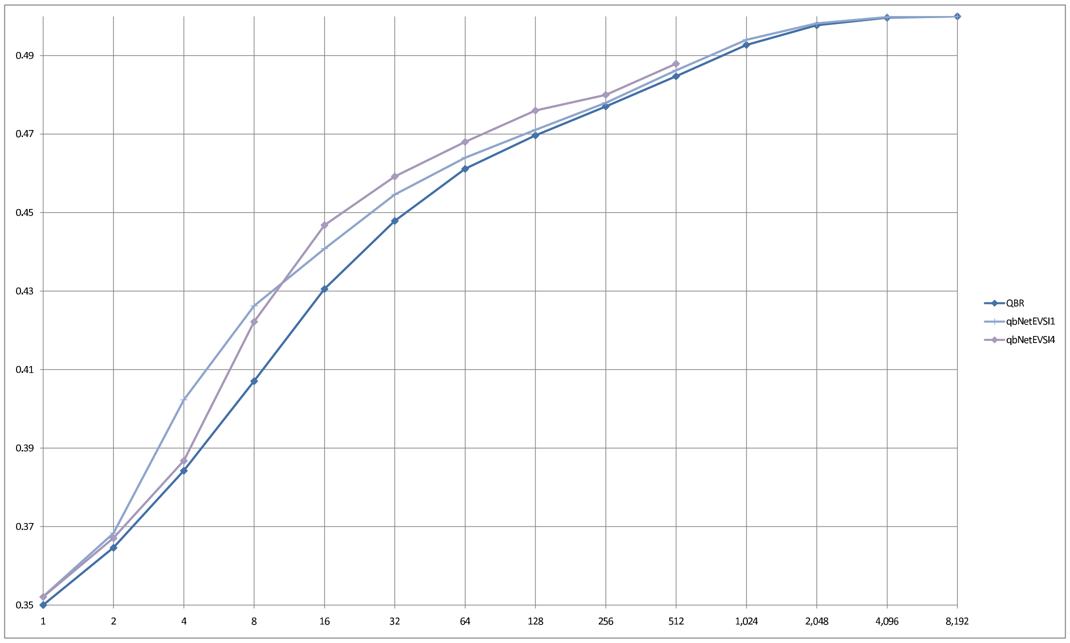


Figure 6.13: Greedy EVSI vs. four step look ahead EVSI on the gfqbc function.

the maximum expected utility decision, then it can be shown that $\forall x_T, EVSI(\mathbf{x}_T, \mu = 1) = 0 - C(\mathbf{x}_T)$. This happens because the value of information flows from its ability to change the agent's behavior. If a single sample can have no effect on the agent's behavior, then it has no value of information. Once greedy EVSI was the same for all potential sample locations, the algorithm began exploring randomly. However, if two samples together have the potential to change the maximum expected utility decision, then it is possible to have a non zero $EVSI(\mathbf{x}, \mu > 1)$. This effect can be seen in Figure 6.13, where EVSI with higher values for μ outperformed EVSI for lower values of μ (and eventually out performed QBC). Thus, the expected value of taking the first sample changed depending on the number of samples that the equations expected to be able to take later.

The computational complexity of the active learning equations depends greatly on μ. Because of this, it has been common to make the greedy assumption that $\mu = 1$. Unfortunately, this assumption is rarely met in practice. Very common simple situations can cause the greedy assumption to fail. Example 6.7.3 illustrates one such simple case. This is one reason greedy active learning approaches can sometimes perform poorly.

Observation 6.7.0.3: *From equations 6.9 to 6.24 it can be seen that it is possible to use decision theory to determine how many samples to take, but this computation requires an upper bound for μ.*

EVSI can be used to decide whether to take a sample. When the expected value of taking a sample is less than the costs for taking the sample, then sampling should end. Alternatively, it is possible to determine when to stop sampling by creating a potential sample with no information about f and with 0 sample cost, and then determine if this "null" sample is the maximum expected utility sample; if it is, then sampling should end. However, in order to compute either of these values, an upper bound for μ is needed. It is possible to use the above principles to determine if fewer than μ samples should be taken, but this upper bound is needed for the calculation. With no effective bound on μ, it can be impossible to decide whether or not to take the first sample.

One way to use EVSI as a stopping criteria is to assume that $\mu = 1$, the greedy assumption. For complexity reasons this is how EVSI is most often used as a stopping criteria. However, given 6.7.0.2, this can cause problems. To optimally use EVSI as a stopping criteria, it is necessary to compute the value of taking a sample and then making sampling decisions optimally from that point on. We define $VO(S)$ as the expected value of starting with the training set $\mathscr{D}_{train}$, and sampling an additional set S, and then behaving optimally from then on. This leads to the recursive definition:

$$VO(S) = max[VOI(S) - C(S), \max_{\mathbf{x}_t}[\sum_{c \in \mathbb{D}_Y} p(y_c|\mathbf{x}_t, \mathscr{D}_{train}) VO(S \cup (\mathbf{x}_t, y_c))]],$$

and we have seen many variants of the above in Section 6.7.0.2. This recursive formulation either returns the value of not getting any more samples, or of taking another sample, and then recursively determining whether to take any more samples from that point on [37]. This recursion will go on forever if there is not some upper bound to the number of samples to be taken.

There are several ways in which a bound on the number of samples can be created. For pool based active learning the size of the pool naturally bounds the number of samples. In non-pool based active learning, when there is an upper bound on the end use utility and a finite positive sample cost, then an upper bound can be placed on the value of sample information. If the sample costs are finite and positive then it is possible to stop once those costs are larger then the upper bound on the value of sample information.

Observation 6.7.0.4: *From equations 6.9 to 6.24 it can be seen that active learning depends on the sampling distribution of the test set.*

When performing simple supervised learning, $\ddot{\mathbf{x}}$ was observed at the time the end use decision was made, and so this issue was less important. However, for active learning, at the time the sampling decision is made, the test set sample locations may or may not be observed, yet this information is necessary to optimally make the optimal sampling decision.

In transductive learning this information comes from the observed future $\ddot{\mathbf{x}}$ values in the observed test set. For inductive learning the sum over the test set is replaced by a weighted sum over the domain of $\ddot{\mathbf{X}}$. This requires an explicit model of the sampling distribution of the test set. In some on-line learning situations the future $\ddot{\mathbf{x}}$ values may be selectable by the agent, and there is no simple distribution over future samples. However, even in this case, the knowledge that the future sample locations can be chosen by the agent will impact the optimal active learning approach.

Example 6.7.4: If the mushroom classifier will be used in a mushroom packing facility that only deals with African mushrooms, then samples drawn from American mushrooms will have less value of information over the expected $\xi(\ddot{\mathbf{x}}|f)$ that the algorithm will be tested on. On the other hand, if the classifier will be used in a mushroom packing facility that only deals with American mushrooms, then samples drawn from African mushrooms will have less value of information.

Thus, changes in the distribution of the features of the test set can dramatically affect active learning.

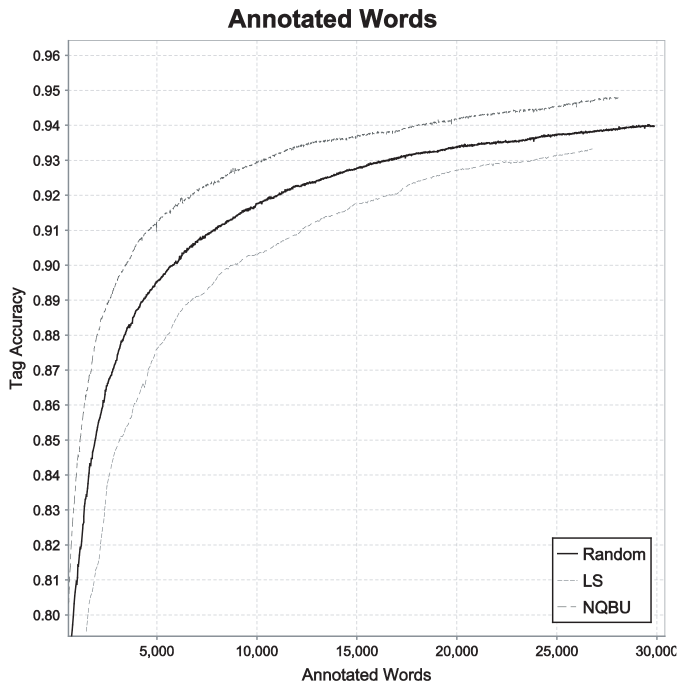
Observation 6.7.0.5: *From equations 6.9 to 6.24 it can be seen that active learning depends upon the sample cost, and different sample costs should lead to different active learning solutions.*

Example 6.7.5: One active learning project involved annotating parts of speech. Active learning was applied to maximize the value of the expensive human annotation. However, it was unclear how the human annotators would be paid. Annotators could be paid either by the sentence, by the word, or by the hour, and the decision on which payment technique to use had not yet been made. Experiments were run to determine the best active learning approach and the annotator payment technique was varied to determine how it affected the active learning approaches performance. Not only did the best active learning technique change with the way annotators were paid, but the technique that was best in one situation actually performed worse than random in the other situation (See Figure 6.14). The complete results are not presented here for space considerations, but have been previously published elsewhere [14]. It was thus impossible to select the best active learning technique until the mechanism for paying annotators was determined. Despite the importance of these issues, several published studies on active learning for part of speech tagging have all ignored how the annotators were paid.

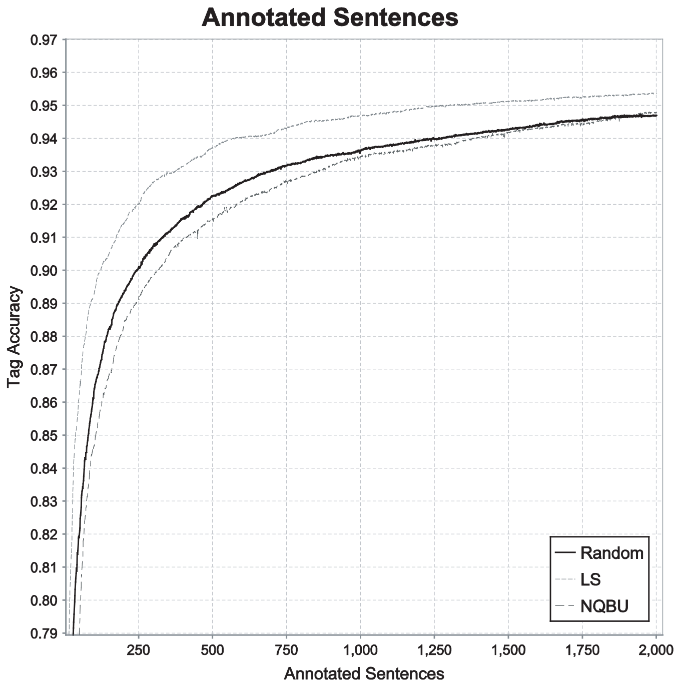
Observation 6.7.0.6: *From equations 6.9 to 6.24 it can be seen that active learning depends intimately upon end use.*

Since active learning requires the computation of the expected utility of sampling in a given location, this maximum expected utility computation requires a thorough model of how utility is received. Thus, active learning can not be performed without some indication of end use.

Example 6.7.6: The toxicity level of mushrooms can have a greater effect on pregnant women than on other members of the community. If the regression task is to approximate the toxicity level $\ddot{y}$, and the decision task is to determine if the mushroom should be eaten, then the decision



(a)



(b)

Figure 6.14: a) Learning curves for Active Learning when paying by the word. Notice that a variant of QBU, normalized by the number of words in the sentence performs best in this case, and that "longest sentence" actually performs worse than random. b) Learning curves for Active Learning when paying by the sentence. Notice that the "longest sentence" Active Learning procedure performs best in this case and that normalized QBU actually performs worse than random.

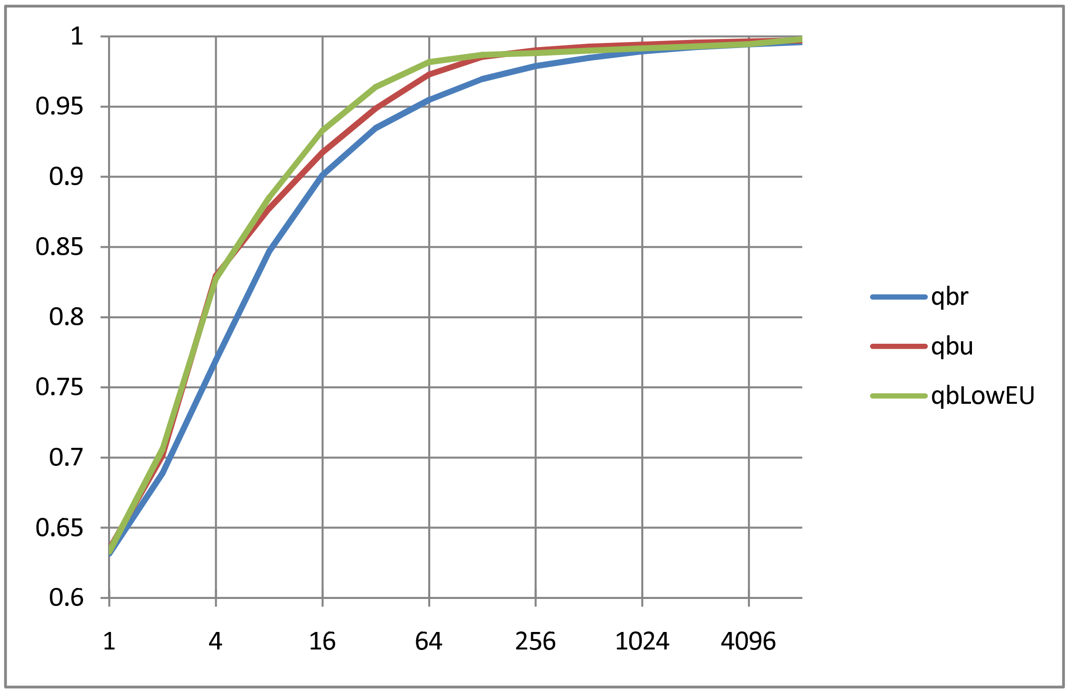
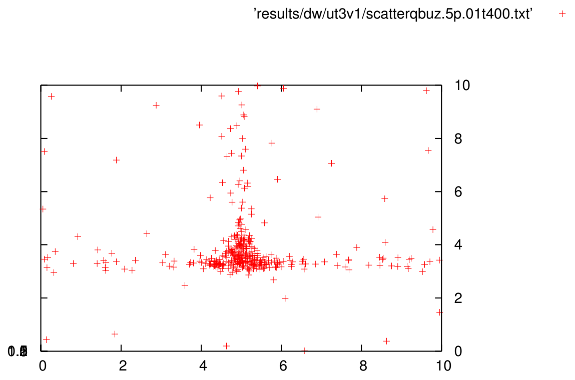


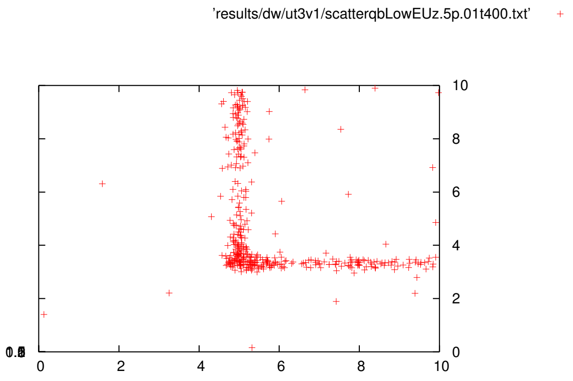
Figure 6.15: QBU verses QBLowEU on ut3v1 function.

boundary between eating and not eating the mushroom will be in a different location for pregnant women than for the rest of the population. Samples near the decision boundary tend to have the most information and many active learning techniques attempt to concentrate samples near this boundary. An active learning technique designed for pregnant women should therefore actively select different training data than an active learning technique designed for the rest of the population.

Example 6.7.7: Another example involves a simple two dimensional classification task with three different classes. If the utility function involves misclassification error rate, then all the decision boundaries are important, and an active learner should sample in locations surrounding all decision boundaries (see Figure 6.17(a)). On the other hand, if we change the utility function so that misclassifications between two of the classes are irrelevant, then it would be un-necessary for the active learner to sample along that decision boundary (see Figure 6.17(b)). Uncertainty Sampling, (sometimes called Query by Uncertainty, or QBU) and Query by Lowest Expected Utility (QBLowEU) perform differently because they each make different end use utility assumptions.



(a)



(b)

Figure 6.16: a) QBU scatter plot. b) QBLowEU scatter plot.

QBU assumes misclassification error rate, and thus samples along the un-necessary decision boundary (Figure 6.17(a)) while QBLowEU performs better in this case (See Figure 6.15) by ignoring this boundary (Figure 6.17(b)) because it was able to take information about the utility function into account.

Thus, utility affects active learning in two main ways, first through how much the information aids in decision making (an element of end use) and second through sample costs C. It can be shown that many of the current active learning approaches approximate the more computationally complex optimal solution under several simplifying assumptions. Some of the most common simplifying assumptions involve the greedy assumption ($\mu = 1$), the 0-1 loss end use assumption, and the idea that C is uniform for all sample locations. In a forthcoming paper we will show that the common techniques of Uncertainty Sampling

[106][57][56][2], and Query by Committee [103][32][21] both make all of the above assumptions.

Observation 6.7.0.7: *From equations 6.9 to 6.24 it can be seen that a technique for estimating full posterior predictives is necessary for active learning for arbitrary utility function. Classifiers that do not report full posterior predictives can be used, but will only be correct for certain end uses.*

All of the equations for the active learning situations covered in this chapter use the posterior predictive in some way. This is not surprising since they all must at least expect over the potential results of the samples that they could take. We know that simple supervised learning with arbitrary utility functions requires the full posterior predictive (see Chapter 4), and it should therefore be of no surprise that the same is true for active learning. For active learning, just as with supervised learning, it is only when the utility function is known that insufficient statistics can be used instead of the full posterior predictive, and for the same reason (again see Chapter 4).

Many approximate techniques exist for active learning, and some (such as QBC) do not use classifiers that report full posterior predictives, but this observation means that these active learners are either approximating the full posterior predictive in some way, or making some sort of utility assumptions.

Example 6.7.8: Query-by-Committee is one such example. However, according to Observation 6.7.0.7, such algorithms must have some technique for approximating the uncertainty in the posterior predictive, or they are making some end use utility assumptions. In the case of Query-by-Committee, the committee can be seen as producing a monte carlo approximation to the uncertainty about the maximum decision for whatever utility assumptions are made by the classifiers that constitute the committee [21].

6.7.1 The Supervised Active Learning NFL Theorems

Theorem 6.7.1.1 (The Supervised Active Learning No-Free-Lunch Theorem)**:** *For the active learning scenarios listed in Section 6.5 if the prior, $P(f)$ is discrete uniform, if the function is static, if the sample cost is uniform, if all sample locations that lead to end use (usually the test set) are off training set, then the expected utility of sampling in one potential sample location is the same as sampling in another.*

Proof. There are an infinite number of active learning scenarios, and so it is impossible to prove this theorem for all possible active learning scenarios so we are here only dealing with the ones listed in Section 6.5. Furthermore, the theorem only holds for the static variants in that section. The theorem only holds for those cases, and we will only prove the theorem for the generative, static, transductive, off line, active learning case while leaving the proofs for the other cases to the reader. [1]

[1] To prove this theorem for the other cases, note that all utility either comes from end use $U(.)$, or from sample cost $C(.)$. From equations 6.9 to 6.24 it can be seen that in all static active learning situations which apply, all references to U and C are found in a portion of the equation much like the following:

$$\sum_{f \in \mathbb{D}_F} P(f|\mathscr{D}_\mu) \sum_{\ddot{y}_k \in \mathbb{D}_Y} p(\ddot{y}_k|\ddot{\mathbf{x}}_k, f) \sum_{o_k \in \mathbb{D}_O} p(o_k|\ddot{\mathbf{x}}_k, \ddot{y}_k, f, d_k)\big[U(o_k) - C(\mathscr{D}_\mu \setminus \mathscr{D}_{i-1})\big], \tag{6.26}$$

with only minor variations in what is being summed over etc. By following very similar steps to the main proof line, it is possible to make these equation fragments independent of the samples taken for each of the active learning scenarios. The basic idea is simply that the posterior predictive on the test set was uniform off training set, and it remains uniform no matter what off test set samples are taken, thus those samples have no impact, and thus no information on test set locations.

The expected sampling utility for this case is:

$$\begin{aligned} E(U|\mathbf{x}_i, \mathscr{D}_{i-1}, \mu) = &\sum_{y_i \in \mathbb{D}_Y} p(y_i|\mathbf{x}_i, \mathscr{D}_{i-1}) \\ &\max_{\mathbf{x}_{i+1} \in \mathbb{D}_X} \sum_{y_{i+1} \in \mathbb{D}_Y} p(y_{i+1}|\mathbf{x}_{i+1}, \mathscr{D}_i) \\ &\ldots \\ &\max_{\mathbf{x}_\mu \in \mathbb{D}_X} \sum_{y_\mu \in \mathbb{D}_Y} p(y_\mu|\mathbf{x}_\mu, \mathscr{D}_{\mu-1}) \\ &\sum_{x_k \in \mathscr{D}^X_{test}} \max_{d_k \in \mathbb{D}_D} \sum_{f \in \mathbb{D}_F} P(f|\mathscr{D}_\mu) \sum_{\ddot{y}_k \in \mathbb{D}_Y} p(\ddot{y}_k|\ddot{\mathbf{x}}_k, f) \\ &\sum_{o_k \in \mathbb{D}_O} p(o_k|\ddot{\mathbf{x}}_k, \ddot{y}_k, f, d_k)\big[U(o_k) - C(\mathscr{D}_\mu \setminus \mathscr{D}_{i-1})\big] \end{aligned} \tag{6.27}$$

Because the sample costs are uniform, the sampling costs will be the same for all sample locations, and we can therefore safely ignore sample costs, simplifying the above to:

$$\begin{aligned} E(U|\mathbf{x}_i, \mathscr{D}_{i-1}, \mu) = &\sum_{y_i \in \mathbb{D}_Y} p(y_i|\mathbf{x}_i, \mathscr{D}_{i-1}) \\ &\max_{\mathbf{x}_{i+1} \in \mathbb{D}_X} \sum_{y_{i+1} \in \mathbb{D}_Y} p(y_{i+1}|\mathbf{x}_{i+1}, \mathscr{D}_i) \\ &\ldots \\ &\max_{\mathbf{x}_\mu \in \mathbb{D}_X} \sum_{y_\mu \in \mathbb{D}_Y} p(y_\mu|\mathbf{x}_\mu, \mathscr{D}_{\mu-1}) \\ &\sum_{x_k \in \mathscr{D}^X_{test}} \max_{d_k \in \mathbb{D}_D} \sum_{f \in \mathbb{D}_F} P(f|\mathscr{D}_\mu) \sum_{\ddot{y}_k \in \mathbb{D}_Y} p(\ddot{y}_k|\ddot{\mathbf{x}}_k, f) \\ &\sum_{o_k \in \mathbb{D}_O} p(o_k|\ddot{\mathbf{x}}_k, \ddot{y}_k, f, d_k) U(o_k) \end{aligned} \tag{6.28}$$

Because we are only interested in supervised learning, the supervised learning assumption that $p(o_k|\ddot{\mathbf{x}}_k, \ddot{y}_k, f, d_k) = p(o_k|\ddot{y}_k, d_k)$ applies, allowing us to simplify to:

$$\begin{aligned}
E(U|\mathbf{x}_i, \mathscr{D}_{i-1}, \mu) = \sum_{y_i \in \mathbb{D}_Y} & p(y_i|\mathbf{x}_i, \mathscr{D}_{i-1}) \\
& \max_{\mathbf{x}_{i+1} \in \mathbb{D}_X} \sum_{y_{i+1} \in \mathbb{D}_Y} p(y_{i+1}|\mathbf{x}_{i+1}, \mathscr{D}_i) \\
& \ldots \\
& \max_{\mathbf{x}_\mu \in \mathbb{D}_X} \sum_{y_\mu \in \mathbb{D}_Y} p(y_\mu|\mathbf{x}_\mu, \mathscr{D}_{\mu-1}) \\
& \sum_{x_k \in \mathscr{D}^X_{test}} \max_{d_k \in \mathbb{D}_D} \sum_{f \in \mathbb{D}_F} P(f|\mathscr{D}_\mu) \sum_{\ddot{y}_k \in \mathbb{D}_Y} p(\ddot{y}_k|\ddot{\mathbf{x}}_k, f) \\
& \sum_{o_k \in \mathbb{D}_O} p(o_k|\ddot{y}_k, d_k) U(o_k)
\end{aligned} \tag{6.29}$$

Now that o and f are conditionally independent of each other, we can simplify to:

$$\begin{aligned}
E(U|\mathbf{x}_i, \mathscr{D}_{i-1}, \mu) = \sum_{y_i \in \mathbb{D}_Y} & p(y_i|\mathbf{x}_i, \mathscr{D}_{i-1}) \\
& \max_{\mathbf{x}_{i+1} \in \mathbb{D}_X} \sum_{y_{i+1} \in \mathbb{D}_Y} p(y_{i+1}|\mathbf{x}_{i+1}, \mathscr{D}_i) \\
& \ldots \\
& \max_{\mathbf{x}_\mu \in \mathbb{D}_X} \sum_{y_\mu \in \mathbb{D}_Y} p(y_\mu|\mathbf{x}_\mu, \mathscr{D}_{\mu-1}) \\
& \sum_{x_k \in \mathscr{D}^X_{test}} \max_{d_k \in \mathbb{D}_D} \sum_{\ddot{y}_k \in \mathbb{D}_Y} p(\ddot{y}_k|\ddot{\mathbf{x}}_k, \mathscr{D}_\mu) \\
& \sum_{o_k \in \mathbb{D}_O} p(o_k|\ddot{y}_k, d_k) U(o_k).
\end{aligned} \tag{6.30}$$

Because we are off training set, ($\ddot{\mathbf{x}}_k \notin \mathscr{D}_\mu$), and because Y is discrete, therefore $p(\ddot{y}_k|\ddot{\mathbf{x}}_k, \mathscr{D}_\mu) = \frac{1}{|\mathbb{D}_Y|}$ by the supervised learning NFL theorem, (see Theorem 3.6.0.7). Therefore we can simplify to:

$$
\begin{aligned}
E(U|\mathbf{x}_i, \mathscr{D}_{i-1}, \mu) = &\sum_{y_i \in \mathbb{D}_Y} p(y_i|\mathbf{x}_i, \mathscr{D}_{i-1}) \\
&\max_{\mathbf{x}_{i+1} \in \mathbb{D}_X} \sum_{y_{i+1} \in \mathbb{D}_Y} p(y_{i+1}|\mathbf{x}_{i+1}, \mathscr{D}_i) \\
&\ldots \\
&\max_{\mathbf{x}_\mu \in \mathbb{D}_X} \sum_{y_\mu \in \mathbb{D}_Y} p(y_\mu|\mathbf{x}_\mu, \mathscr{D}_{\mu-1}) \\
&\sum_{x_k \in \mathscr{D}^X_{test}} \max_{d_k \in \mathbb{D}_D} \sum_{\ddot{y}_k \in \mathbb{D}_Y} \frac{1}{|\mathbb{D}_Y|} \sum_{o_k \in \mathbb{D}_O} p(o_k|\ddot{y}_k, d_k) U(o_k).
\end{aligned}
\tag{6.31}
$$

Notice that at this point all utility comes from the last line of the equation. Notice also that this portion of the equation is independent of the previous samples taken. Let $\sum_{x_k \in \mathscr{D}^X_{test}} \max_{d_k \in \mathbb{D}_D} \sum_{\ddot{y}_k \in \mathbb{D}_Y} \frac{1}{|\mathbb{D}_Y|} \sum_{o_k \in \mathbb{D}_O} p(o_k|\ddot{y}_k, d_k) U(o_k) = c$, and thus:

$$
\begin{aligned}
E(U|\mathbf{x}_i, \mathscr{D}_{i-1}, \mu) = &\sum_{y_i \in \mathbb{D}_Y} p(y_i|\mathbf{x}_i, \mathscr{D}_{i-1}) \\
&\max_{\mathbf{x}_{i+1} \in \mathbb{D}_X} \sum_{y_{i+1} \in \mathbb{D}_Y} p(y_{i+1}|\mathbf{x}_{i+1}, \mathscr{D}_i) \\
&\ldots \\
&\max_{\mathbf{x}_\mu \in \mathbb{D}_X} \sum_{y_\mu \in \mathbb{D}_Y} p(y_\mu|\mathbf{x}_\mu, \mathscr{D}_{\mu-1}) c
\end{aligned}
\tag{6.32}
$$

$$
\tag{6.33}
$$

Since $\mathbf{x}_\mu$ and y_μ do not show up again in the remainder of the equation, we can factor the remainder of the equation (just c) out of the sum and max, and then $\sum_{y_\mu \in \mathbb{D}_Y} p(y_\mu|\mathbf{x}_\mu, \mathscr{D}_{\mu-1}) =$

1 this simplifies the expected utility equation further to:

$$\begin{aligned} E(U|\mathbf{x}_i, \mathscr{D}_{i-1}, \mu) = &\sum_{y_i \in \mathbb{D}_Y} p(y_i|\mathbf{x}_i, \mathscr{D}_{i-1}) \\ &\max_{\mathbf{x}_{i+1} \in \mathbb{D}_X} \sum_{y_{i+1} \in \mathbb{D}_Y} p(y_{i+1}|\mathbf{x}_{i+1}, \mathscr{D}_i) \\ &\ldots \qquad (6.34) \\ &\max_{\mathbf{x}_\mu \in \mathbb{D}_X} 1\, c \end{aligned}$$

$$(6.35)$$

Repeating the same technique leaves us with

$$E(U|\mathbf{x}_i, \mathscr{D}_{i-1}, \mu) = c, \qquad (6.36)$$

and now, because c is independent of the sample location, it follows that:

$$\forall \mathbf{x}_1\, \mathbf{x}_2,\, E(U|\mathbf{x}_1, \mathscr{D}_{i-1}, \mu) = E(U|\mathbf{x}_2, \mathscr{D}_{i-1}, \mu), \qquad (6.37)$$

$\square$

The conditions necessary to bring about an AL No-Free-Lunch situation are not sufficient to cause there to be a standard No-Free-Lunch situation. Specifically there can be no active learning free lunch even when there is still an optimal decision, since AL No-Free-Lunch says nothing about the end use utility function. It is possible for there to be an optimal decision (a supervised learning free lunch), while still having both a Bayesian NFL and an AL NFL. The latter do not require end use utility assumptions other than the supervised learning assumptions about end use.

Utility functions other than misclassification error rate can create a free lunch for supervised learning, but they do not help with active learning. They create a "best" d, but that best d will be the same for all off training set data points and, therefore, for all active learning approaches to gathering that data will perform equally well.

Example 6.7.9: If we are classifying mushrooms, and our prior is that all mushroom classification functions are equally likely, then there is no value in sampling examples that are off test set. Since no two mushrooms are exactly identical, all potential mushroom samples will be off test set, and active learning will be useless. If the sample costs are uniform, then all active learning approaches will perform the same off training set, because there is no information gained about the test set by sampling in off test set locations. Thus, if the sample costs are finite, then the sampling approach that takes no samples will have the best expected utility.

So long as all the sample costs are uniform, then no technique for actively selecting samples will out perform any other with respect to expected utility. From our perspective, this property that the expected utility of sampling in one off test set location is the same as sampling in another is really at the heart of the NFL theorems for search and empirical function optimization (see Chapter 7).

This means that uniform priors can be more disruptive for active learning than they were for supervised learning. In traditional supervised learning, uniform priors only demanded a uniform posterior, which could lead to informed decisions for some utility function. However, with uniform priors there is no value of sample information in any off training set location.

Active Learning Free Lunches

As with the supervised learning NFL theorem, there are many cases where the AL-NFL theorem does not hold. Under these situations, it is possible to have an active learning free lunch.

The case where $\mathbf{x}$ impacts o directly is allowed by the UBDTM, but not by the supervised learning special case of the UBDTM. One common situation where $\mathbf{x}$ impacts o is when errors in one part of the function are more important than errors in another.

Although this does not match our formal definition of simple supervised learning, it is a case that is common, and could perhaps be classified as a more complex variant of supervised learning. In this case, there can be an active learning free lunch, because sampling in the part of the function where errors are more important will be more useful than sampling in the part of the function where errors are less important. Another way in which the influence from $\mathbf{x}$ to o can create a free lunch is in the on-line case, where the agent's sampling decision also impacts the samples which will be involved in testing. If there is some influence between $\mathbf{x}$ and o in the testing phase, then it is possible for some $\mathbf{x}$ sampling decisions to produce a higher expected utility than another when the sampling decisions also select the testing samples.

Another way in which there can be an active learning free lunch is if the sampler is allowed to repeat samples and if the test set is not guaranteed to be off training set. Sampling strategies can have different expected utility based on how often they repeat samples, *and* based on which samples they repeat.

Another way to have an active learning free lunch is if F is dynamic. This can cause off training set samples to be effectively on training set in future time periods depending on T. This effect was first noted by Wolpert in his traditional EFO NFL results [124]. There he noted that although dynamic F can cause an active learning free lunch, uniformly averaging over all possible T created another NFL like situation [124].

Another way to have an active learning free lunch is if the costs are not uniform. In that case sampling in areas with a lower expected cost can have a higher expected net utility than strategies that sample in areas with higher expected cost.

6.8 Conclusions

To the best of our knowledge, this is the first time the full AL equations have been given. Most do not do this because it is computationally intensive to actually solve these equations. However, knowing what they are has an impact on theory. Using this approach we were able to make several theoretical observations about AL, and use the equations to prove an Active Supervised Learning NFL theorem.

There are nearly an infinite number of potential AL scenarios, and we have therefore only selected some of the more common situations and created a graphical model to illustrate the situation, and given the optimal AL equations for that situation. It is hoped that we have provided enough examples that the correct equations can be created for any active learning situation.

Researchers do not always know how their algorithms will be used. It makes sense to want to be able to create a supervised learning sampler that will perform reasonably well over a wide range of sample cost functions and over a wide range of end uses. However, as we showed in Chapter 4, it has been unclear exactly what parts of the supervised learning problem depend on sample cost and end use and which are independent of sample cost or end use. In this section, we have explored how the behavior of many active learning strategies are often dramatically dependent on certain end use and sample cost assumptions, as well as on assumptions about the number of samples that will be taken, and the number of times the information can be exploited.

Just as past supervised learning research has been overly focused on the 0,1 gain utility function, so past active learning research has overly focused on uniform sample costs. Sample cost (when it has been addressed at all) has traditionally been assumed to be uniform for all samples. This is often not the case in practice, just as some examples can have a higher value of information, some samples can also come at a higher cost. For example, in part of speech tagging, the cost of labeling a single example often varies by the length of the sentence [37] [92].

6.8.1 Future Work

Query by Uncertainty, and Query by Committee are some of the most common active learning techniques. In this chapter, we have given the actual equations which form the solutions to the problems that QBU and QBC were designed to solve. If QBU and QBC are useful tools for solving these problems, it must be because they approximate the true answers given here. Future work will involve taking the optimal sampling equations and, by making a series of simplifying assumptions, turning those equations into QBU or QBC. This would allow us to explain why techniques such as QBU and QBC work at all, and to predict when they should succeed or fail. We would expect these techniques to function well when their simplifying assumptions are met, and to perform poorly when these assumptions are not met.

Chapter 7

The Empirical Function Optimization No-Free-Lunch Theorems

Abstract

In this chapter we will return to Empirical Function Optimization and take an approach similar to that taken for Supervised Learning. We will review the traditional model used to produce the EFO-NFL theorems (the EFO-EBF), and we will compare and contrast this model with the active learning extensions to the UBDTM. We will then propose new Bayesian EFO-NFL Theorems from the perspective of the expected value of sample information.

Publication: Some of the ideas in this chapter have been previously published in [17], but the majority are presented here for the first time.

7.1 Introduction

Empirical function optimization (EFO) involves repeatedly sampling from an unknown function in order to find some extremum (see Sections 2.2.4 and 2.3.4). Given the similarities between supervised learning and empirical function optimization revealed by the UBM and UBDTM, it should not be a surprise that No-Free-Lunch theorems also exist for empirical function optimization [124] [67]. The traditional No-Free-Lunch theorems for optimization state that all search policies perform equally well when uniformly averaged over all possible discrete finite functions for optimization sampling policies that do not repeat sample locations and for end use utility functions that depend only upon a histogram of sample outputs obtained. This means that when searching for a maximum, gradient decent performs as well as gradient ascent, which both perform as well as random search.

Although a statistical model was used to prove these traditional NFL theorems, they were not evaluated from a Bayesian perspective. In this Chapter, we will re-evaluate these traditional NFL concepts for EFO from a Bayesian perspective. While the traditional NFL perspective tends to talk about all search algorithms that do not repeat values performing equally well on average, a Bayesian perspective would take a different view, and be more interested in the value of information. Because the UBDTM models both supervised learning and EFO, some results from supervised learning also hold for EFO, including the uniform posterior predictive $p(\ddot{y}|\ddot{\mathbf{x}}, \mathscr{D}_{train})$ off training set given a uniform prior $\rho(f)$. Since we are here concerned with EFO algorithms that do not repeat sample values, all sampling is by definition off training set at the time the samples are taken. In this Chapters, we will show that this uniform posterior predictive will lead to all potential non repeating sample locations having the same expected value of sample information.

In Section 7.2, we will present the model that was used as a framework to prove the traditional EFO-NFL Theorems. We call this model the EFO-EBF. In Section 7.3, we will compare the EFO-EBF with the simple supervised learning EBF. In Section 7.4, we will compare the EFO-EBF to the UBDTM's extensions to selective sampling. In Section 7.5, we

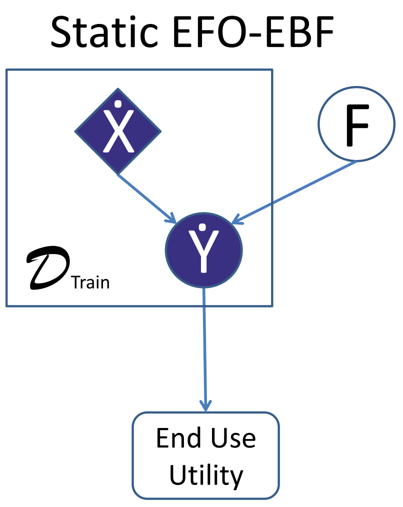


Figure 7.1: The Extended Bayesian Formalism for Empirical Function Optimization (EFO-EBF) with a static function represented as a graphical model to illustrate the relationships between features, function outputs, and the function.

will present the traditional NFL theorem for empirical function optimization. In Section 7.6, we will use the UBM to give the Bayesian EFO-NFL theorem, a utility free interpretation of NFL for empirical function optimization. In Section 7.7, we will present an active learning NFL theorem for EFO. Finally, in Section 7.8, we will conclude and suggest future work.

7.2 EFO-EBF

In order to prove the traditional NFL results for EFO, Wolpert and Macready use a statistical model similar to, but not identical to, the supervised learning EBF which we will refer to as the EFO-EBF. In this section we will present the model assumptions of the EFO-EBF. As when we presented the EBF, we will be changing their notation slightly to keep notation consistent with our notation. We will also present the EFO-EBF as a graphical model. Analogies between EFO and supervised learning will allow us to use concepts such as a training set for the observed samples, and an active learner for the algorithm that selects the samples to take, and EVSI for the optimal sampling approach.

EFO-EBF Assumption 1: In the EFO-EBF, and unknown deterministic function, f, maps a vector of function inputs $\mathbf{x}$, to function outputs y, formally $f : \mathbf{x} \rightarrow y$. The function may be static or dynamic, and if it is dynamic, the transitions are modeled by the transition matrix T (compare Figure 7.1 with Figure 7.2).

EFO-EBF Assumption 2: Samples that have been taken so far can be seen as analogous to a training set, $\dot{\mathbf{x}}$ and are selected by an active learning algorithm a.

EFO-EBF Assumption 3: All samples in the training set $\mathscr{D}^X_{train}$ are unique.

EFO-EBF Assumption 4: The active learner, a, is represented as either a deterministic or a non-deterministic mapping from the previously visited set of points to a single, new, previously unvisited point in $\mathbb{D}_{\mathbf{X}}$. Formally, $a : \mathscr{D}_{train} \rightarrow \ddot{\mathbf{x}}$ such that $\ddot{\mathbf{x}} \in \mathbb{D}_{\mathbf{X}}$, or $a : \mathscr{D}_{train} \rightarrow \xi(\ddot{\mathbf{x}})$ such that $\ddot{\mathbf{x}} \in \mathbb{D}_{\mathbf{X}}$.

EFO-EBF Assumption 5: End use utility is modeled as a deterministic function of the histogram of function outputs obtained by the search strategy, formally: $U(\vec{c})$, where $\vec{c}$ is a histogram of function outputs in the training set obtained, $\mathscr{D}^Y_{train}$.

EFO-EBF Assumption 6: Sample costs are implicitly assumed to be uniform. Although they never make this assumption explicitly, by leaving all concept of sample costs out of their formulation, they were implicitly either assuming that they were 0, and so disappeared from the computation entirely, or they were implicitly assuming that they were uniform, and had a constant, ignorable effect.

Notice that, as before, since f is a mapping between $\mathbf{x}$ and y, such that $p(y|\mathbf{x}, f)$ is uniquely determined by each f. However, the EFO-EBF assumption 1 differs from the EBF assumption 1, in that the EFO-EBF does not deal with nondeterministic functions, forcing $p(y|\mathbf{x}, f)$ to be one in a single location and zero elsewhere.

Because we are interested in comparing EFO with supervised learning, we have represented the samples that have been taken by the function optimizer with the same notation we used for a training set, $\mathscr{D}_{train}$. As we shall see, these samples (sometimes called a population in evolutionary empirical function optimization) actually *are* a training set, and serve

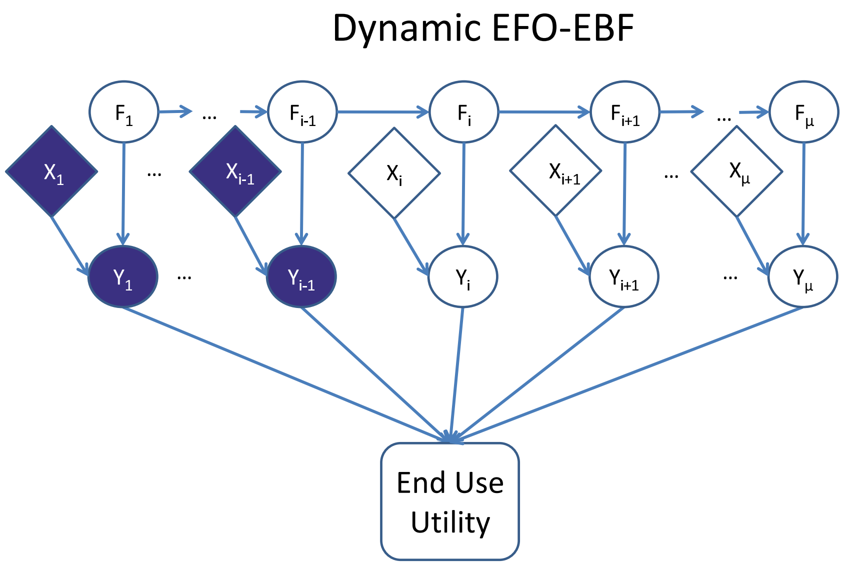


Figure 7.2: The Extended Bayesian Formalism for Empirical Function Optimization (EFO-EBF) for a dynamic function represented as a graphical model at time step i, with choice nodes.

the same purpose, namely, they provide information that will be used to infer information about f. Notice also that EFO-EBF Assumption 3 demands that all of these samples taken by the algorithm are unique. As we shall see, this assumption is essential for NFL in EFO, since it insures that each new sample is "off training set" at the time it is taken.

EFO-EBF Assumption 5 specifies the way in which end use utility is determined. This assumption is less restricted than was the utility assumptions made by the EFO for supervised learning. There the utility was restricted to the 0,1 gain utility function, however, here the utility function can be any desired function of $\mathscr{D}^{Y}_{train}$. Nevertheless, we shall later take issue with this definition of end use utility for function optimization.

EFO-EBF Assumption 6 was implicit in the formulation used by Wolpert and Macready. However, if they had not made this assumption, then traditional EFO-NFL would not have held because one algorithm could have outperformed another by simply sampling in less expensive sample locations.

In EBF Assumption 3, it was assumed that the sampling distribution was conditionally independent of f. EFO-EBF Assumption 2 has a similar but slightly different effect. Here the samples are no longer random variables, but are decision nodes, chosen by the EFO algorithm, a, which by EFO-EBF Assumption 4 can only use the previous samples in making its decision. This means that the samples are again conditionally independent of f given a and the previous samples, $\mathscr{D}_{train}$.

7.3 The EFO-EBF compared with the Supervised Learning EFO

The EFO-EBF differs from the Supervised learning EBF in several ways. The supervised learning EBF was only used to prove an NFL result for a specific utility function. However, the EFO-EBF proves a NFL result for EFO that does not depend on the utility function chosen, so long as this function only depends on the sample vector. Thus the EFO-EBF does not make any specific assumptions about *how* the utility might be computed from this vector of samples.

The EFO-EBF was designed to illustrate the point that all EFO algorithms perform equally well regardless of how or even whether they model the function. Thus, the EFO-EBF does not have any specific way of modeling the learner's hypothesis h, and although it does model the function f, the learner is specifically forbidden from directly using this information (although this restriction does not prevent a from having its own model of the function as part of a). This is most likely why the h node appears in the Wolpert's supervised learning EBF, but not in his EFO-EBF.

Of special significance is that the differences between the EBF and the EFO-EBF prevent results from one being used for the other, a situation that is remedied in the UBM and UBDTM.

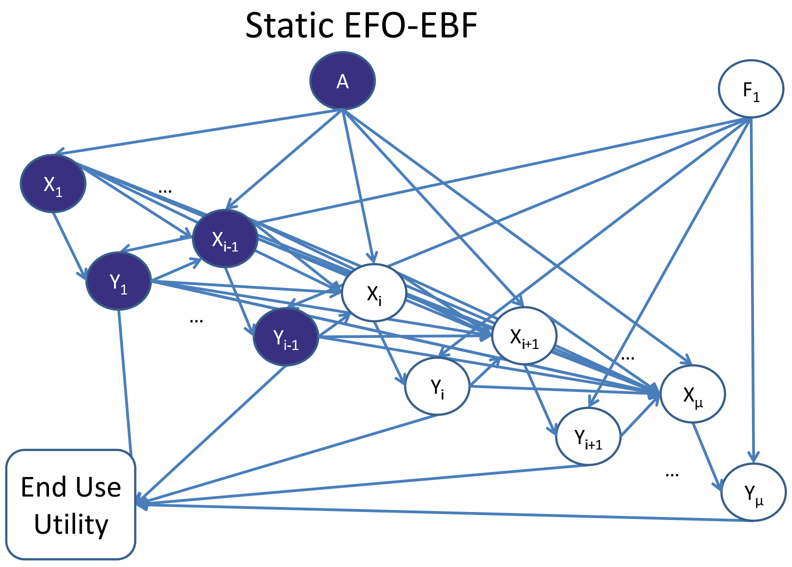


Figure 7.3: The Extended Bayesian Formalism for Empirical Function Optimization (EFO-EBF) for a static function represented as a graphical model at time step i, with the active learning strategy a used to infer the new sample locations instead of considering them to be choice nodes.

7.4 The EFO-EBF Compared with the Active Learning Extended UBDTM

Empirical Function Optimization is essentially an active learning problem. Therefore, we would expect the active learning extensions to the UBDTM made in Chapter 6 to have some similarity to the EFO-EBF. As before, the EFO-EBF is primarily a descriptive model designed to prove the EFO-NFL results, while the active learning extensions to the UBDTM form a prescriptive model designed to show how active learning *should* be performed. This distinction will drive the primary differences between the two approaches. There are several reasons to prefer the UBDTM model of EFO over that of the EFO-EBF.

Since the purpose of the UBDTM is to show how learning *should* be performed, it must model its belief about the function f. This node is present in the EFO-EBF, but it was not used to determine the sampling strategy, while it is intimately used to determine the sampling strategy in the extended UBDTM.

Normally, decision nodes in a graphical decision network affect the other nodes in the network (represented by arrows going out of the decision node), but are not affected by the other nodes in the network (meaning that they usually do not have arrows going into them). Decision theory then uses the effect that making a decision has on the rest of the network to select the decision that will maximize the expected utility of the entire network. Thus, the maximum expected utility of a decision can depend on many of the other nodes in the network, but we would never expect a line from one of these other nodes *to* a decision node in a graphical decision network. We have represented the EFO-EBF in this way in Figure 7.1 and Figure 7.2. However, the assumptions of the EFO-EBF are difficult to completely capture in a traditional graphical decision network because the EFO-EBF limits the decision in highly unusual ways. Specifically, in the EFO-EBF the decisions are made by a stochastic process that *only* depends on the previous samples taken up to this point and on the sampling algorithm, with each potential sampling algorithm representing only a different mapping between previous samples and the next sample. A given EFO-EBF sampling algorithm can still be identical to the decisions that would be made by EVSI with a specific prior, but since the prior is not an input to the technique, it is impossible to design a technique that would produce optimal behavior over all priors. Thus the general optimal solution to the problem is expressly forbidden by EFO-EBF Assumption 5.

This also means that we can compute the probability that the sampling technique will choose a specific $\ddot{\mathbf{x}}$ as follows, $\xi(\ddot{\mathbf{x}}_t|\mathscr{D}_{t-1}, a)$. Just as with $p(y|\mathbf{x}, f)$, this distribution is uniquely determined by the mapping inherent in a. This allows us to actually use the principles of statistical inference to *infer* the next decision rather that using the principles of maximum expected utility to *choose* the best next decision, which would cause arrows to point *into* a node that would normally be considered a decision node, but which now must be considered a random variable just like any other. If we take this approach, then the assumptions of the EFO-EBF can be represented as in Figure 7.3.

In the EFO-EBF, $a : \mathscr{D}_{train} \rightarrow \ddot{\mathbf{x}} \in \mathbb{D}_{\mathbf{X}}$ if a is deterministic (returns a pure strategy). In the extended UBDTM, it is also possible to talk about an algorithm, a, that is some function that models the system's sampling strategy. However, in the UBDTM, a uses *all* the available observed variables in the UBDTM, together with information about the end use and sample costs, in order to determine the next sample to take. By taking all this extra information into account, it is possible to make a single strategy that can adapt to different utility functions, or to different sample costs, or to different priors. One way to look at this is to say that each a in the EFO-EBF is encoding a lot of implied information about the function class, end use, and sample cost that is nowhere expressly stated as an input to a, and that every sampling strategy must making some assumptions about these things much as we say in Chapter 6.

Another important difference between the EFO approach and the UBDTM, is that in the EFO, in order to model supervised learning one model was created (the supervised learning EFO model), while another very different model had to be created in order to model EFO (the EFO-EBF). This stands in contrast to the UBDTM. In the UBDTM we saw that it was possible to create special cases of the more general UBDTM by adding additional independence assumptions. This is also true of the new Extended UBDTM that has been extended to include active learning. With one set of additional independence assumptions we can produce a model of active supervised learning, while with another set of additional independence assumptions we can produce a model of active learning in empirical function optimization. This will allow us to use some of the results from supervised learning to analyze EFO, which could not be done with the EBF and EFO-EBF.

Perhaps one of the most significant differences between the EFO-EBF and the UB-DTM applied to EFO is in the way in which utility is modeled. In the extended UBDTM, the sample cost is explicitly taken into account and modeled, while this information was ignored by the EFO-EBF, which implied some sort of uniform cost model (EFO-EBF Assumption 6). This is important, since the UBDTM is designed to determine how empirical function optimization *should* be performed, which, in the real world, should depend to a large degree on sample costs. Also, in the Extended UBDTM, end use utility depends on the function f. Since both $\mathbf{x}^*$ and y^* are part of f, this allows the utility to depend directly on the unobserved location and value of the extremum, even if that value has not been sampled by the learner up to that point. The EFO-EBF on the other hand specifically defines the end use utility in terms of a histogram of the samples actually taken by the active learner, thus only upon the observed $\mathscr{D}^Y_{train}$ values.

At first this may appear to be a reasonable assumption. For example, if the goal was to maximize, then it might make sense to assess the performance of the empirical function optimizer by the maximum value it was able to actually observe. However, as we shall see in Chapter 10, it is often the case that the optimal sampling strategy will purposefully sample in locations other than at the optimum in order to determine the location of the optimum. It turns out that this case is quite common. But this situation can cause the optimal algorithm to never actually take samples in the location of the extremum, and thus the extremum's value may not show up in the histogram of the data. This can cause serious problems when comparing the performance of two techniques that may have the same histogram of samples taken so far, but where one technique is better able to use that information to infer information about the extremum's location than can the other. This also would force any active sampling technique based on the EFO-EBF to actually try and sample the extremum, in order to get high utility, instead of sampling in the locations that have the most value of sample information (for an example where this is the case, see Section 10.3).

7.5 Traditional No-Free-Lunch For Empirical Function Optimization

Traditional NLF for empirical function optimization is interested in the *a priori* performance of sampling algorithms designed to locate a function's extremum, as measured exclusively by the histogram of the samples taken by some active learning approach. Thus, NFL for EFO involves many of the same NFL assumptions that were made for supervised NFL. Thus, in addition to the EFO-EBF assumptions, the traditional EFO-NFL Theorems make the following NFL assumptions:

EFO-NFL Assumption 1: f is finite and discrete, meaning that each f is a mapping between a discrete and finite $\mathbb{D}_{\mathbf{X}}$ and a discrete and finite $\mathbb{D}_Y$.

EFO-NFL Assumption 2: All functions are equally likely: $\forall f_1, f_2\ \rho(f_1) = \rho(f_2)$.

For supervised learning, NFL Assumption 3 was that all samples in the test set are off training set. However, in this case, there is no test set. Yet, EFO-EBF Assumption 3 plays this role, since it assures that each new sample is unique, and thus not in the set of samples taken up until that point, essentially insuring that each new sample is off training set.

Under the EFO-EBF and NFL assumptions, the traditional static EFO-NFL Theorem shows that:

Theorem 7.5.0.1 (The Traditional Empirical Function Optimization No-Free-Lunch Theorem)**:** *For any pair of algorithms a_1 and a_2,*

$$\sum_f P(\vec{c}|f, m, a_1) = \sum_f P(\vec{c}|f, m, a_2),$$

where $\vec{c}$ is a histogram of function output ($\mathscr{D}^Y_{train}$) values that a_i, has obtained after taking m samples from f.

Proof. For the proof of this theorem see the publications of David H. Wolpert and William G. Macready [123][124]. □

Wolpert and Macready argue that, because of Theorem 7.5.0.1, there is no *a priori* distinctions between EFO algorithms. As before, they are using the term *a priori* to refer to the situation where all functions are equally likely. Thus, Theorem 7.5.0.1 shows the *a priori* lack of distinctions between EFO algorithms because it leads to the following more significant result:

Theorem 7.5.0.2 (The Traditional Empirical Function Optimization No-Free-Lunch Theorem Restated)**:** *For any pair of algorithms a_1 and a_2,*

$$\sum_f P(\vec{c}|f, m, a_1)\rho(f) = \sum_f P(\vec{c}|f, m, a_2)\rho(f).$$

Proof. In the *a priori* case, where EFO-NFL Assumption 2 holds, $\rho(f) = 1/|F|$. Thus:

$$\sum_f P(\vec{c}|f, m, a_1)\rho(f) = \sum_f P(\vec{c}|f, m, a_2)\rho(f).$$

Leads to:

$$\sum_f P(\vec{c}|f, m, a_1)\frac{1}{|F|} = \sum_f P(\vec{c}|f, m, a_2)\frac{1}{|F|},$$

and,

$$\sum_f P(\vec{c}|f, m, a_1) = \sum_f P(\vec{c}|f, m, a_2),$$

which is true by Theorem 7.5.0.1. $\square$

Note that $\vec{c}$ is a histogram with $|Y|$ unique bins each representing a unique possible function output value. This implies that the performance of any two algorithms (measured for example as the depth of the extremum found) is, on average, identical when uniformly averaged over all possible functions f.

While this idea casts a shadow over the possibility of developing an efficient black-box optimization algorithm, it is in many ways unsurprising; for each function function on which an EFO algorithm performs efficiently, another function may be crafted which causes the algorithm to fail.

Although NFL does not hold in general when $\rho(f)$ is not uniform, an important corollary to traditional EFO-NFL shows that it is possible to construct some priors other than uniform for which NFL holds [19, 44]. Fortunately, this simply states that uniformity of function distribution is a sufficient but not a necessary condition for NFL to hold; it says nothing about whether some classes of functions may be chosen for which an algorithm may be expected to do better than random search. In fact, many useful function classes exist over which algorithms may be meaningfully ranked by performance [19, 44]. Computational complexity theory lends intuition to this last idea. The sorting of integers, for example, is a problem that has a proven worst-case complexity; various sorting algorithms may be ranked even when considering all possible instances of the sorting problem, since some obtain the proven complexity result and others do not. In fact, the concepts of computational complexity and No-Free-Lunch are closely linked; rather than ranking algorithms over a class of functions, it is also possible to rank the complexity of function optimization problems without respect for the algorithm which is applied [117, 67].

Notice again that these results hold when success is measured in terms of the histogram of values obtained under the identical number of oracle queries. Therefore, any use of this result to discuss the *a priori* behavior of EFO algorithms assumes that all oracle queries cost the same amount, at least on average (the expected cost of sample information, ECSI, is the same for all samples).

7.6 Bayesian No-Free-Lunch For Empirical Function Optimization

The traditional EFO-NFL theorem is essentially a statement about *a priori* distinctions between *active* learning algorithms. However, it is possible to make some concrete statements about the posterior for f under the NFL assumptions independent of the active technique used to select the samples. We will first produce a Bayesian EFO-NFL result that is independent of the active sampling technique used, and then use that result to produce an active learning Bayesian EFO-NFL result that will compare active learning techniques under the NFL assumptions.

From a Bayesian perspective, the reasons for the empirical function optimization No-Free-Lunch theorem becomes clear from our earlier discussion of supervised learning. Under the UBDTM, f, y, and $\mathbf{x}$ are related in the same way for EFO and for supervised learning. This means that some results for supervised learning actually directly apply to EFO. Specifically, if a uniform prior leads to an off training set uniform posterior predictive for $\ddot{y}$ in supervised learning, then it must also lead to an off training set uniform posterior predictive for $\ddot{y}$ in EFO. Intuitively, given that uniform posterior predictive for $\ddot{y}$, there is no reason to believe that the extremum is more likely to be in one off training set location than in another off training set location. We may know that the extremum is *not* in some *on* training set locations, but we have no basis for preference *off* training set. If a sampling approach never repeats samples, then all samples were *off* training set when they were taken. This line of reasoning leads to the Bayesian EFO-NFL Theorem:

Theorem 7.6.0.3 (The Bayesian Empirical Function Optimization No-Free-Lunch Theorem)**:** *if the UBDTM EFO assumptions hold, if the function is deterministic, and the EFO-NFL assumptions hold, then for all* $\ddot{\mathbf{x}}_1$ *and* $\ddot{\mathbf{x}}_2 \notin \mathscr{D}^X_{train}$*, it can be shown that* $\rho(\ddot{\mathbf{x}}_1 \in \mathscr{S}_{\mathbf{x}^*}|\mathscr{D}_{train}) = \rho(\ddot{\mathbf{x}}_2 \in \mathscr{S}_{\mathbf{x}^*}|\mathscr{D}_{train})$.

Proof. In the deterministic case, the distribution $\rho(\ddot{\mathbf{x}}_i \in \mathscr{S}_{\mathbf{x}^*}|\mathscr{D}_{train})$ is the probability that the sample location $\ddot{\mathbf{x}}_i$ is in the set of sample locations $\mathscr{S}_{\mathbf{x}^*}$ that all lead to the extremum value, y^* given the training set. What we want to show is that:

$$\rho(\ddot{\mathbf{x}}_1 \in \mathscr{S}_{\mathbf{x}^*}|\mathscr{D}_{train}) = \rho(\ddot{\mathbf{x}}_2 \in \mathscr{S}_{\mathbf{x}^*}|\mathscr{D}_{train}). \tag{7.1}$$

These probabilities over the location of the extremum, $\rho(\ddot{\mathbf{x}}_i \in \mathscr{S}_{\mathbf{x}^*}|\mathscr{D}_{train})$, can be written in terms of a sum of $p(\ddot{y}_i, y_i = y^*|\ddot{\mathbf{x}}_i, \mathscr{D}_{train})$, the probability that the value $\ddot{y}_i$ is observed when sampling $\ddot{\mathbf{x}}_i$, and that that value is equal to the unknown extremum value y^*:

$$\rho(\ddot{\mathbf{x}}_i \in \mathscr{S}_{\mathbf{x}^*}|\mathscr{D}_{train}) = \sum_{\ddot{y}_i in \mathbb{D}_Y} p(\ddot{y}_i, y_i = y^*|\ddot{\mathbf{x}}_i, \mathscr{D}_{train}). \tag{7.2}$$

However, some of these potential $\ddot{y}_i \in \mathbb{D}_Y$ values can not be the extremum given the data. Therefore, we now define the new set of $\ddot{y}$ values that can possibly be the extremum given the data: $S = \{y \in \mathbb{D}_Y | p(y = y^* | \mathscr{D}_{train}) \neq 0\}$. Since $p(\ddot{y}_i, y_i = y^* | \ddot{\mathbf{x}}_i, \mathscr{D}_{train}) = 0$ when $y_i \notin S$, we can rewrite this sum as follows:

$$\rho(\ddot{\mathbf{x}}_i \in \mathscr{S}_{\mathbf{x}^*} | \mathscr{D}_{train}) = \sum_{\ddot{y}_i \in S} p(\ddot{y}_i, y_i = y^* | \ddot{\mathbf{x}}_i, \mathscr{D}_{train}). \tag{7.3}$$

Using the chain rule and Equation 7.3 we can write:

$$\rho(\ddot{\mathbf{x}}_i \in \mathscr{S}_{\mathbf{x}^*} | \mathscr{D}_{train}) = \sum_{\ddot{y}_i \in S} p(y_i = y^* | \ddot{\mathbf{x}}_i, \mathscr{D}_{train}) p(\ddot{y}_i | \ddot{\mathbf{x}}_i, \mathscr{D}_{train}, y_i = y^*). \tag{7.4}$$

Using the independence assumptions of the model, $p(y_i = y^* | \ddot{\mathbf{x}}_i, \mathscr{D}_{train}) = p(y_i = y^* | \mathscr{D}_{train})$, since the value of the extremum does not depend on a specific sample location, allowing the simplification:

$$\rho(\ddot{\mathbf{x}}_i \in \mathscr{S}_{\mathbf{x}^*} | \mathscr{D}_{train}) = \sum_{\ddot{y}_i \in S} p(y_i = y^* | \mathscr{D}_{train}) p(\ddot{y}_i | \ddot{\mathbf{x}}_i, \mathscr{D}_{train}, y_i = y^*). \tag{7.5}$$

Now, we can use the fact that:

$$p(\ddot{y}_i | \ddot{\mathbf{x}}_i, \mathscr{D}_{train}, y_i = y^*) = \sum_{f \in \mathbb{D}_F} p(\ddot{y}_i | \ddot{\mathbf{x}}_i, \mathscr{D}_{train}, y_i = y^*, f) P(f | \ddot{\mathbf{x}}_i, \mathscr{D}_{train}, y_i = y^*),$$

in order to expand Equation 7.5 to:

$$\rho(\ddot{\mathbf{x}}_i \in \mathscr{S}_{\mathbf{x}^*}|\mathscr{D}_{train}) = \sum_{\ddot{y}_i \in S} p(y_i = y^*|\mathscr{D}_{train}) \sum_{f \in \mathbb{D}_F} p(\ddot{y}_i|\ddot{\mathbf{x}}_i, \mathscr{D}_{train}, y_i = y^*, f)$$
$$P(f|\ddot{\mathbf{x}}_i, \mathscr{D}_{train}, y_i = y^*). \tag{7.6}$$

Using the independence assumptions of the UBDTM, we can simplify to:

$$\rho(\ddot{\mathbf{x}}_i \in \mathscr{S}_{\mathbf{x}^*}|\mathscr{D}_{train}) = \sum_{\ddot{y}_i \in S} p(y_i = y^*|\mathscr{D}_{train}) \sum_{f \in \mathbb{D}_F} p(\ddot{y}_i|\ddot{\mathbf{x}}_i, f) P(f|\mathscr{D}_{train}, y_i = y^*). \tag{7.7}$$

In order for $\rho(\ddot{\mathbf{x}}_1 \in \mathscr{S}_{\mathbf{x}^*}|\mathscr{D}_{train}) = \rho(\ddot{\mathbf{x}}_2 \in \mathscr{S}_{\mathbf{x}^*}|\mathscr{D}_{train})$ to hold, it must be true that:

$$0 = \sum_{\ddot{y}_i \in S} p(y_i = y^*|\mathscr{D}_{train}) \sum_{f \in \mathbb{D}_F} p(\ddot{y}_i|\ddot{\mathbf{x}}_1, f) P(f|\mathscr{D}_{train}, y_i = y^*) -$$
$$\sum_{\ddot{y}_i \in S} p(y_i = y^*|\mathscr{D}_{train}) \sum_{f \in \mathbb{D}_F} p(\ddot{y}_i|\ddot{\mathbf{x}}_2, f) P(f|\mathscr{D}_{train}, y_i = y^*). \tag{7.8}$$

This leads to:

$$= \sum_{\ddot{y}_i \in S} p(y_i = y^*|\mathscr{D}_{train}) \Bigg[\sum_{f \in \mathbb{D}_F} p(\ddot{y}_i|\ddot{\mathbf{x}}_1, f) P(f|\mathscr{D}_{train}, y_i = y^*) -$$
$$\sum_{f \in \mathbb{D}_F} p(\ddot{y}_i|\ddot{\mathbf{x}}_2, f) P(f|\mathscr{D}_{train}, y_i = y^*) \Bigg], \tag{7.9}$$

and

$$= \sum_{\ddot{y}_i \in S} p(y_i = y^*|\mathscr{D}_{train}) \Bigg[\sum_{f \in \mathbb{D}_F} P(f|\mathscr{D}_{train}, y_i = y^*) \Big[p(\ddot{y}_i|\ddot{\mathbf{x}}_1, f) - p(\ddot{y}_i|\ddot{\mathbf{x}}_2, f) \Big] \Bigg]. \tag{7.10}$$

We can now define the set $F' = \{f \in \mathbb{D}_F | P(f|\mathscr{D}_{train}, y_i = y^*) \neq 0\}$. Because f is discrete and deterministic, and because the prior for f was $P(f) = 1/|F|$, the posterior $P(f|\mathscr{D}_{train}, y_i =$

Table 7.1: The possibilities for f in Equation 7.11

f	$p(\ddot{y}_i\|\ddot{\mathbf{x}}_1, f)$	$p(\ddot{y}_i\|\ddot{\mathbf{x}}_2, f)$	$p(\ddot{y}_i\|\ddot{\mathbf{x}}_1, f) - p(\ddot{y}_i\|\ddot{\mathbf{x}}_2, f)$
ϕ_0	0	0	0
ϕ_1	0	1	-1
ϕ_2	1	0	1
ϕ_3	1	1	0

$y^*) = 1/|F'|$. Leaving:

$$= \sum_{\ddot{y}_i \in S} p(y_i = y^* | \mathscr{D}_{train}) \left[\sum_{f \in \mathbb{D}_F} \frac{1}{|F'|} \left[p(\ddot{y}_i | \ddot{\mathbf{x}}_1, f) - p(\ddot{y}_i | \ddot{\mathbf{x}}_2, f) \right] \right]. \tag{7.11}$$

Because $1/|F'|$ is a constant:

$$= \sum_{\ddot{y}_i \in S} p(y_i = y^* | \mathscr{D}_{train}) \left[\sum_{f \in \mathbb{D}_F} \left[p(\ddot{y}_i | \ddot{\mathbf{x}}_1, f) - p(\ddot{y}_i | \ddot{\mathbf{x}}_2, f) \right] \right]. \tag{7.12}$$

There are several possibilities that could cause the above to equal 0, but it is sufficient to show that:

$$0 = \sum_{f \in \mathbb{D}_F} \left[p(\ddot{y}_i | \ddot{\mathbf{x}}_1, f) - p(\ddot{y}_i | \ddot{\mathbf{x}}_2, f) \right] \tag{7.13}$$

Since the functions are deterministic, there are only four possible ways that each function can respond to the probability for $\ddot{y}_i$ in $\ddot{\mathbf{x}}_1$ and $\ddot{\mathbf{x}}_2$, shown in Table 7.1.

Thus, F' is the union of four disjoint sets (cases) that we are interested in:

$$\phi_0 = \{f \in F' | p(\ddot{y}_i | \ddot{\mathbf{x}}_1, f) = 0, p(\ddot{y}_i | \ddot{\mathbf{x}}_2, f) = 0\}$$

$$\phi_1 = \{f \in F' | p(\ddot{y}_i | \ddot{\mathbf{x}}_1, f) = 0, p(\ddot{y}_i | \ddot{\mathbf{x}}_2, f) = 1\}$$

$$\phi_2 = \{f \in F' | p(\ddot{y}_i | \ddot{\mathbf{x}}_1, f) = 1, p(\ddot{y}_i | \ddot{\mathbf{x}}_2, f) = 0\}$$

$$\phi_3 = \{f \in F' | p(\ddot{y}_i | \ddot{\mathbf{x}}_1, f) = 1, p(\ddot{y}_i | \ddot{\mathbf{x}}_2, f) = 1\}$$

Because these sets are all disjoint, and we can rewrite Equation 7.13 as:

$$
\begin{aligned}
=&\sum_{f\in\phi_0}\Big[p(\ddot{y}_i|\ddot{\mathbf{x}}_1,f)-p(\ddot{y}_i|\ddot{\mathbf{x}}_2,f)\Big]+\sum_{f\in\phi_1}\Big[p(\ddot{y}_i|\ddot{\mathbf{x}}_1,f)-p(\ddot{y}_i|\ddot{\mathbf{x}}_2,f)\Big]+\\
&\sum_{f\in\phi_2}\Big[p(\ddot{y}_i|\ddot{\mathbf{x}}_1,f)-p(\ddot{y}_i|\ddot{\mathbf{x}}_2,f)\Big]+\sum_{f\in\phi_3}\Big[p(\ddot{y}_i|\ddot{\mathbf{x}}_1,f)-p(\ddot{y}_i|\ddot{\mathbf{x}}_2,f)\Big],
\end{aligned}
$$

leading to,

$$=0+|\phi_1|-|\phi_2|+0. \tag{7.14}$$

This will be true if $|\phi_1|=|\phi_2|$. We know several things that will help. First, we know that if there exists an f_1 in ϕ_1, then the proposition $\ddot{y}_i=y^*$ is consistent with the training data, and that it is consistent with all we know to see $\ddot{y}_i$ in the off training set location $\ddot{\mathbf{x}}_2$ or else f_1 would not have been in F', and therefore not in ϕ_1. However, if it is consistent to see $\ddot{y}_i$ in the off training set location $\ddot{\mathbf{x}}_2$, it must also be consistent with all we know to see $\ddot{y}_i$ in the off training set location $\ddot{\mathbf{x}}_1$. Similarly, if it is consistent for $\ddot{y}_i$ to *not* be in $\ddot{\mathbf{x}}_1$, then it must also be consistent for $\ddot{y}_i$ to *not* be in $\ddot{\mathbf{x}}_2$. Thus, if $p(\ddot{y}_i|\ddot{\mathbf{x}}_1,f)=0, p(\ddot{y}_i|\ddot{\mathbf{x}}_2,f)=1$, is consistent with all we know then $p(\ddot{y}_i|\ddot{\mathbf{x}}_1,f)=1, p(\ddot{y}_i|\ddot{\mathbf{x}}_2,f)=0$, must also be consistent. Therefore, for every function in ϕ_1, it is possible to construct exactly one other function which will be in ϕ_2 as follows:

$$
f_2(\ddot{\mathbf{x}})=\begin{cases}
f_1(\ddot{\mathbf{x}}_2) & \text{if } \ddot{\mathbf{x}}=\ddot{\mathbf{x}}_1\\
f_1(\ddot{\mathbf{x}}_1) & \text{if } \ddot{\mathbf{x}}=\ddot{\mathbf{x}}_2\\
f_1(\ddot{\mathbf{x}}) & \text{otherwise}
\end{cases}
$$

This same mapping will also turn every function in ϕ_2 into exactly one function in ϕ_1. This is a bijection, and, therefore, $|\phi_1| = |\phi_2|$, which means that $\rho(\ddot{\mathbf{x}}_1 \in \mathscr{S}_{\mathbf{x}^*}|\mathscr{D}_{train}) = \rho(\ddot{\mathbf{x}}_2 \in \mathscr{S}_{\mathbf{x}^*}|\mathscr{D}_{train})$. $\square$

The above proof is long and complex, but the intuition behind it is rather simple. If we limit ourselves to sampling techniques that never repeat samples, then each new sample is guaranteed to be off training set at the time it is taken. By the Supervised Bayesian NFL theorem, we know that all off training set locations have a uniform posterior predictive and, therefore, the probability that one off training set location is an extremum is the same as the probability that another off training set location is an extremum.

This result says nothing about the sampling algorithms, but is a statement about the state of belief at any given point in time for all sampling algorithms. The next step will be to determine the relative performance of different sampling approaches.

7.7 Active Bayesian Empirical Function Optimization NFL Theorems

Discussions of EFO traditionally involve not just how information from the samples are used to learn about the function, but also involve the sampling method for acquiring those samples. Thus, EFO is intimately related to supervised active learning and active sampling in general. Thus we are now in a situation similar to that of Chapter 6.

As we saw in that chapter, there are nearly an infinite number of potential active sampling scenarios. Although it is possible to imagine pool or stream based EFO scenarios, the most common EFO situation involves off line generative sampling, and for space considerations, we will only deal with that case. Since NFL only holds for EFO in the case of static functions, we will therefore restrict ourselves to the static, generative, situation. Furthermore, in EFO there is no test set, so the distinctions between inductive and transductive sampling no longer makes sense.

The active learning equation for this situation is produced by modifying Equation 6.9 It is possible to simplify this equation, by applying the additional EFO UBDTM inde-

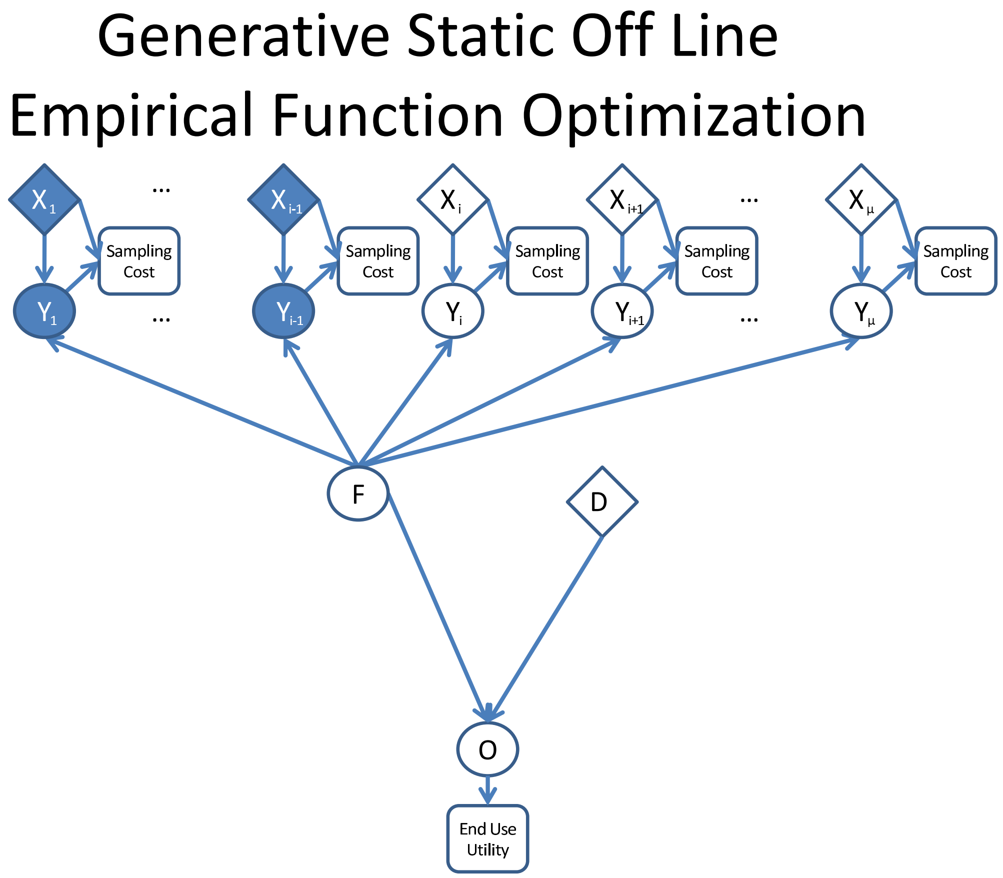


Figure 7.4: The generative static off line empirical function optimization case at the ith time step, compare Figure 6.2.

pendence assumptions from Section 2.3.4. This yields the EFO active sampling situation represented by Figure 7.4. The following equation then represents the expected utility of sampling in the generative, static, off line, empirical function optimization case:

$$
\begin{aligned}
E(U|\mathbf{x}_i, \mathscr{D}_{i-1}, \mu) = &\sum_{y_i \in \mathbb{D}_Y} p(y_i|\mathbf{x}_i, \mathscr{D}_{i-1}) \\
&\max_{\mathbf{x}_{i+1} \in \mathbb{D}_X \cap \bar{\mathscr{D}}_i^X} \sum_{y_{i+1} \in \mathbb{D}_Y} p(y_{i+1}|\mathbf{x}_{i+1}, \mathscr{D}_i) \\
&\ldots \\
&\max_{\mathbf{x}_\mu \in \mathbb{D}_X \cap \bar{\mathscr{D}}_{\mu-1}^X} \sum_{y_\mu \in \mathbb{D}_Y} p(y_\mu|\mathbf{x}_\mu, \mathscr{D}_{\mu-1}) \\
&\max_{d_k \in \mathbb{D}_D} \sum_{f \in \mathbb{D}_F} P(f|\mathscr{D}_\mu) \sum_{o_k \in \mathbb{D}_O} p(o_k|d_k, f) \big[U(o_k) - C(\mathscr{D}_\mu \setminus \mathscr{D}_{i-1})\big]
\end{aligned}
\tag{7.15}
$$

We are now prepared to give an active Bayesian NFL theorem for this case.

Theorem 7.7.0.4 (The Active Bayesian Empirical Function Optimization No-Free-Lunch Theorem)**:** *For generative static off line empirical function optimization, if the prior, $P(f)$ is discrete and uniform, if the sample costs are uniform, if the sampling strategy never repeats samples, and if $p(o_k|d_k, \mathscr{D}_\mu) = p(y = o_k|\mathbf{x} = d_k, \mathscr{D}_\mu)$, then the expected utility of sampling in one potential new sample location is the same as sampling in another, but not the same as not sampling at all.*

Proof. Because the cost of sampling is uniform, we can simplify Equation 7.15 to:

$$E(U|\mathbf{x}_i, \mathscr{D}_{i-1}, \mu) = \sum_{y_i \in \mathbb{D}_Y} p(y_i|\mathbf{x}_i, \mathscr{D}_{i-1})$$
$$\max_{\mathbf{x}_{i+1} \in \mathbb{D}_X \cap \mathscr{D}_i^{\bar{X}}} \sum_{y_{i+1} \in \mathbb{D}_Y} p(y_{i+1}|\mathbf{x}_{i+1}, \mathscr{D}_i)$$
$$... \tag{7.16}$$
$$\max_{\mathbf{x}_\mu \in \mathbb{D}_X \cap \mathscr{D}_{\mu-1}^{\bar{X}}} \sum_{y_\mu \in \mathbb{D}_Y} p(y_\mu|\mathbf{x}_\mu, \mathscr{D}_{\mu-1})$$
$$\max_{d_k \in \mathbb{D}_D} \sum_{f \in \mathbb{D}_F} P(f|\mathscr{D}_\mu) \sum_{o_k \in \mathbb{D}_O} p(o_k|d_k, f)\big[U(o_k)\big]$$

It is given that the sampling technique never repeats a sample location, this means that at each time step, the sampler must select samples from off training set locations. By the Supervised Learning Bayesian NFL theorem (Theorem 3.6.0.7), we can simplify further to:

$$E(U|\mathbf{x}_i, \mathscr{D}_{i-1}, \mu) = \sum_{y_i \in \mathbb{D}_Y} \frac{1}{|\mathbb{D}_Y|} \max_{\mathbf{x}_{i+1} \in \mathbb{D}_X \cap \mathscr{D}_i^{\bar{X}}} \sum_{y_{i+1} \in \mathbb{D}_Y} \frac{1}{|\mathbb{D}_Y|} ... \max_{\mathbf{x}_\mu \in \mathbb{D}_X \cap \mathscr{D}_{\mu-1}^{\bar{X}}} \sum_{y_\mu \in \mathbb{D}_Y} \frac{1}{|\mathbb{D}_Y|}$$
$$\max_{d_k \in \mathbb{D}_D} \sum_{f \in \mathbb{D}_F} P(f|\mathscr{D}_\mu) \sum_{o_k \in \mathbb{D}_O} p(o_k|d_k, f)\big[U(o_k)\big]. \tag{7.17}$$

Algebra yields:

$$E(U|\mathbf{x}_i, \mathscr{D}_{i-1}, \mu) = \sum_{y_i \in \mathbb{D}_Y} \frac{1}{|\mathbb{D}_Y|} \max_{\mathbf{x}_{i+1} \in \mathbb{D}_X \cap \mathscr{D}_i^{\bar{X}}} \sum_{y_{i+1} \in \mathbb{D}_Y} \frac{1}{|\mathbb{D}_Y|} ... \max_{\mathbf{x}_\mu \in \mathbb{D}_X \cap \mathscr{D}_{\mu-1}^{\bar{X}}} \sum_{y_\mu \in \mathbb{D}_Y} \frac{1}{|\mathbb{D}_Y|}$$
$$\max_{d_k \in \mathbb{D}_D} \sum_{f \in \mathbb{D}_F} \sum_{o_k \in \mathbb{D}_O} p(o_k|d_k, f) P(f|\mathscr{D}_\mu)\big[U(o_k)\big]. \tag{7.18}$$

Now, $\sum_{f \in \mathbb{D}_F} \sum_{o_k \in \mathbb{D}_O} p(o_k|d_k, f)P(f|\mathscr{D}_\mu)$ is simply marginalization, and therefore: $\sum_{f \in \mathbb{D}_F} \sum_{o_k \in \mathbb{D}_O} p(o_k|d_k, f)P(f|\mathscr{D}_\mu) = \sum_{o_k \in \mathbb{D}_O} p(o_k|d_k, \mathscr{D}_\mu)$, leading to:

$$E(U|\mathbf{x}_i, \mathscr{D}_{i-1}, \mu) = \sum_{y_i \in \mathbb{D}_Y} \frac{1}{|\mathbb{D}_Y|} \max_{\mathbf{x}_{i+1} \in \mathbb{D}_X \cap \mathscr{D}_i^{\bar{X}}} \sum_{y_{i+1} \in \mathbb{D}_Y} \frac{1}{|\mathbb{D}_Y|} \cdots \max_{\mathbf{x}_\mu \in \mathbb{D}_X \cap \mathscr{D}_{\mu-1}^{\bar{X}}} \sum_{y_\mu \in \mathbb{D}_Y} \frac{1}{|\mathbb{D}_Y|}$$
$$\max_{d_k \in \mathbb{D}_D} \sum_{o_k \in \mathbb{D}_O} p(o_k|d_k, \mathscr{D}_\mu) \big[U(o_k)\big]. \tag{7.19}$$

We have assumed that: $p(o_k|d_k, \mathscr{D}_\mu) = p(y = o_k|\mathbf{x} = d_k, \mathscr{D}_\mu)$. Notice that if this is not true, there can be a free lunch, since without some restriction, the outcome can depend upon the data in any way, including by selecting better outcomes simply if certain $\mathbf{x}$ were sampled. With this assumption there will be two distinct cases: the case where the best decision is to select an $\mathbf{x}$ that is on training set, and the case where the best decision is to select an $\mathbf{x}$ that is off training set. Now, by the Bayesian NFL theorem:

$$E(U|\mathbf{x}_i, \mathscr{D}_{i-1}, \mu) = \sum_{y_i \in \mathbb{D}_Y} \frac{1}{|\mathbb{D}_Y|} \max_{\mathbf{x}_{i+1} \in \mathbb{D}_X \cap \mathscr{D}_i^{\bar{X}}} \sum_{y_{i+1} \in \mathbb{D}_Y} \frac{1}{|\mathbb{D}_Y|} \cdots \max_{\mathbf{x}_\mu \in \mathbb{D}_X \cap \mathscr{D}_{\mu-1}^{\bar{X}}} \sum_{y_\mu \in \mathbb{D}_Y} \frac{1}{|\mathbb{D}_Y|} \tag{7.20}$$
$$\max\left[\max_{d_k \in \mathscr{D}_\mu^{Xo}} \sum_{o_k \in \mathbb{D}_O} p(o_k|d_k, \mathscr{D}_\mu)\big[U(o_k)\big], \max_{d_k \in \mathscr{D}_\mu^{X-o}} \sum_{o_k \in \mathbb{D}_O} \frac{1}{|\mathbb{D}_Y|}\big[U(o_k)\big]\right],$$

and algebra yields:

$$E(U|\mathbf{x}_i, \mathscr{D}_{i-1}, \mu) = \sum_{y_i \in \mathbb{D}_Y} \frac{1}{|\mathbb{D}_Y|} \max_{\mathbf{x}_{i+1} \in \mathbb{D}_X \cap \mathscr{D}_i^{\bar{X}}} \sum_{y_{i+1} \in \mathbb{D}_Y} \frac{1}{|\mathbb{D}_Y|} \cdots \max_{\mathbf{x}_\mu \in \mathbb{D}_X \cap \mathscr{D}_{\mu-1}^{\bar{X}}} \sum_{y_\mu \in \mathbb{D}_Y} \frac{1}{|\mathbb{D}_Y|} \tag{7.21}$$
$$\max\left[\max_{d_k \in \mathscr{D}_\mu^{Xo}} \sum_{o_k \in \mathbb{D}_O} p(o_k|d_k, \mathscr{D}_\mu^X)\big[U(o_k)\big], \sum_{o_k \in \mathbb{D}_O} \frac{1}{|\mathbb{D}_Y|}\big[U(o_k)\big]\right].$$

Now, the off training set case is independent of the samples and, therefore, of the sampling technique. It only remains to show that the on training set case is also independent of the samples taken. Theorems 3.5.0.5 and 3.5.0.6 dictate how learning should progress on training set. The posterior predictive is is distributed as a Multinomial, with probability 1 for the y observed, and 0 for all y not observed if the function is deterministic, is distributed as a Multinomial, with probability $\frac{1}{|\mathbb{D}_Y|+1}$ for all possible y other than the one observed, and $\frac{2}{|\mathbb{D}_Y|+1}$ for the y that was observed, if the function is not deterministic.

In the deterministic case:

$$E(U|\mathbf{x}_i, \mathscr{D}_{i-1}, \mu) = \sum_{y_i \in \mathbb{D}_Y} \frac{1}{|\mathbb{D}_Y|} \max_{\mathbf{x}_{i+1} \in \mathbb{D}_X \cap \mathscr{D}_i^{\bar{X}}} \sum_{y_{i+1} \in \mathbb{D}_Y} \frac{1}{|\mathbb{D}_Y|} \cdots \max_{\mathbf{x}_\mu \in \mathbb{D}_X \cap \mathscr{D}_{\mu-1}^{\bar{X}}} \sum_{y_\mu \in \mathbb{D}_Y} \frac{1}{|\mathbb{D}_Y|} \tag{7.22}$$
$$\max\left[\max_{y_z \in \mathscr{D}_\mu^Y} \left(\sum_{o_k \neq y_z} 0\big[U(o_k)\big] + 1\big[U(y_z)\big] \right), \sum_{o_k \in \mathbb{D}_O} \frac{1}{|\mathbb{D}_Y|} \big[U(o_k)\big] \right].$$

Now, the first half of this equation has to do with sampling, while the second half has to do with the utility acquired given those samples. Notice that this second half of the equation is completely independent of the $\mathbf{x}$ sampling locations, and only depends on the histogram of y samples achieved through sampling, represented by $\mathscr{D}_\mu^Y$. By the traditional EFO NFL theorem (Theorem 7.5.0.2), we know that this histogram is not independent of the number of samples chosen, but is independent of the sampling technique chosen, thus:

$$E(U|\mathbf{x}_i, \mathscr{D}_{i-1}, \mu) = \sum_{\mathscr{D}_\mu^Y} p(\mathscr{D}_\mu^Y|\mu) \max\left[\max_{y_z \in \mathscr{D}_\mu^Y} \left(\sum_{o_k \neq y_z} 0\big[U(o_k)\big] + 1\big[U(y_z)\big] \right), \right.$$
$$\left. \sum_{o_k \in \mathbb{D}_O} \frac{1}{|\mathbb{D}_Y|} \big[U(o_k)\big] \right]. \tag{7.23}$$

Notice that this final equation is independent of x_i, but is not independent of μ, and thus the expected utility of sampling in one potential new sample location is the same as sampling in another, but not the same as not sampling at all for the deterministic case.

We must now repeat this conclusion for the non-deterministic case:

$$E(U|\mathbf{x}_i,\mathscr{D}_{i-1},\mu) = \sum_{y_i \in \mathbb{D}_Y} \frac{1}{|\mathbb{D}_Y|} \max_{\mathbf{x}_{i+1} \in \mathbb{D}_X \cap \bar{\mathscr{D}}_i^X} \sum_{y_{i+1} \in \mathbb{D}_Y} \frac{1}{|\mathbb{D}_Y|} \cdots \max_{\mathbf{x}_\mu \in \mathbb{D}_X \cap \mathscr{D}_{\mu-1}^{\bar{X}}} \sum_{y_\mu \in \mathbb{D}_Y} \frac{1}{|\mathbb{D}_Y|} \tag{7.24}$$
$$\max\left[\max_{y_z \in \mathscr{D}_\mu^Y} \left(\sum_{o_k \neq y_z} \frac{1}{|\mathbb{D}_Y|+1}\big[U(o_k)\big] + \frac{2}{\mathbb{D}_Y+1}\big[U(y_z)\big]\right), \sum_{o_k \in \mathbb{D}_O} \frac{1}{|\mathbb{D}_Y|}\big[U(o_k)\big]\right].$$

Now, the first half of this equation has to do with sampling, while the second half has to do with the utility acquired given those samples. Notice that this second half of the equation is completely independent of the $\mathbf{x}$ sampling locations, and only depends on the histogram of y samples achieved through sampling, represented by $\mathscr{D}_\mu^Y$. By the traditional EFO NFL theorem (Theorem 7.5.0.2), we know that this histogram is not independent of the number of samples chosen, but is independent of the sampling technique chosen, thus:

$$E(U|\mathbf{x}_i,\mathscr{D}_{i-1},\mu) = \sum_{\mathscr{D}_\mu^Y} p(\mathscr{D}_\mu^Y|\mu) \max\left[\max_{y_z \in \mathscr{D}_\mu^Y} \left(\sum_{o_k \neq y_z} \frac{1}{|\mathbb{D}_Y|+1}\big[U(o_k)\big] + \frac{2}{\mathbb{D}_Y+1}\big[U(y_z)\big]\right),\right.$$
$$\left.\sum_{o_k \in \mathbb{D}_O} \frac{1}{|\mathbb{D}_Y|}\big[U(o_k)\big]\right]. \tag{7.25}$$

Notice that this final equation is independent of x_i, but is not independent of μ, and thus the expected utility of sampling in one potential new sample location is the same as sampling in another, but not the same as not sampling at all for the non-deterministic case. □

Again, the same sorts of caveats apply that we saw with supervised learning NFL apply. If the cost of acquiring training examples is not uniform, or if the outcome does not follow $p(o_k|d_k,\mathscr{D}_\mu) = p(y = o_k|\mathbf{x} = d_k,\mathscr{D}_\mu)$ then there can be an *a priori* distinction between empirical function optimization algorithms. This is because for some sample acquisition cost functions it can be more cost effective to sample in one location than in another and for some utility functions, and knowing that the extremum is equally likely to reside in all off training set locations is actually useful information that can lead to some decisions being better than others. If the EFO algorithm is more than just the active sampling technique, but also involves how this final decision is made, then clearly some techniques can perform better than others for some utility functions.

It is also possible to have a free lunch if the algorithm can repeat sample locations, since then samples are no longer effectively "off training set." It is also possible to have a free lunch if the function is dynamic, because information from one sample location can move to other sample locations over time, thus effectively causing the sampling technique to be on training set.

Notice that, as with supervised learning, function optimization algorithms based on our graphical model and the rules of active sampling will be optimal with respect to maximum expected utility for all priors not just for the uniform prior.

7.8 Conclusions and Future Work

We have presented the EFO-EBF as a graphical model, and have presented the traditional NFL theorems which flow from this model. We have also compared and contrasted this model with the active learning extensions to the UBDTM.

We have shown that the UBDTM with active learning extensions can also be applied to EFO, and that it represents the optimal method for EFO given a function class. Once defined, this approach allows optimization to be conceptualized as a rational decision process: rather than assuming that sample locations are provided by some external source, an agent is created that selects them. In contrast to most other evolutionary algorithms based on probabilistic models, this rational agent will create populations based on both probabilistic models and well-defined utilities that reflect the goal of optimization: exploring the domain so as to acquire information about the global optimum. An example of using the active learning extensions to the UBDTM to actually produce an EFO algorithm for a specific problem will be given in Chapter 10.

We have also seen that this approach leads to a different approach to NFL than does the EBF, and that this approach unifies the NFL results for supervised learning with the NFL results for EFO. Specifically, we have seen that it is the uniform posterior predictive from the Bayesian supervised learning NFL theorem that produces the EFO NFL result.

For future work we plan on exploring the NFL situation that can be achieved by uniformly averaging over all possible T. This result was shown in the EBF, but we did not deal with this case in the UBDTM, and so future work could involve creating a Bayesian variation of this NFL situation as well.

Chapter 8

Application to Supervised Learning (The BCMAC for Learning in High Assurance Systems)

Abstract

*U*sing the UBM, we develop an improved training procedure for a neural network with the CMAC topology. Parameters in this model can be inferred in closed form and will be theoretically optimal with respect to utility given the assumptions and priors. This technique simplifies training, allows the training of arbitrary kernel functions, and allows us to easily associate actual distributions with the function outputs of the CMAC, thus quantifying our confidence in its results. Such confidences are necessary when using the outputs of a learning technique for decision making and are especially useful in high assurance systems. We provide empirical evidence that the Bayesian learning procedure is superior for three synthetic data sets and for the Abalone data set.

Publication: Most of the material in this chapter has been previously published in [15], and a few elements, such as the kernel version and the empirical Bayes results are published here for the first time.

8.1 Introduction

We have introduced the UBDTM, and shown some of its uses for learning theory. Now we will demonstrate the practical uses of the UBM, by using it to create a new learning approach that has some applications to high assurance systems.

Artificial Neural Networks (ANNs) are a common technique for learning classification and control from examples. However, there are several drawbacks to using ANNs in high assurance systems. These drawbacks involve issues with unspecified output uncertainty, weight interpretability, output generalization, expert knowledge inclusion, and the sufficiency of training data. We will discuss each of these problems in greater detail, and then propose solutions based on using the UBM to optimally learn the weights of the CMAC [1] network topology.

8.1.1 Output Uncertainty

The most common complaint about ANNs in high assurance systems is *output uncertainty*. It is tempting to try and remove all uncertainty in situations where the wrong response can have disastrous consequences. Unfortunately, this is not always possible. There are many sources of uncertainty including intrinsic non-determinism in the function to be learned and insufficient or inappropriate training data. Since it is usually impossible to remove all uncertainty (whether or not an ANN is being used), we believe that the best solution to these problems is to reduce risk where possible through expert knowledge inclusion, and to quantify uncertainty when it is not. Then, through the principles of Bayesian statistics and decision theory, the risk involved can be quantified and each decision can be made in the context of that uncertainty.

Traditional ANNs return only the answer that they consider most probable, but do not provide measures of their confidence in that answer. This makes traditional ANNs especially inappropriate for high assurance systems. Returning to the mushroom example we have used before, if an ANN were trained on the mushroom dataset [88] to predict whether a mushroom will be edible or poisonous given its appearance, how could that network be used to make decisions about eating a given mushroom? This is an example of a high assurance system. If you ate a poisonous mushroom you could get very sick, or even die. If the network predicts that the mushroom is edible should you eat it? Of course, that depends on how hungry you are and on how certain the network is in its prediction. Thus, in order to be effective at aiding decision making, ANNs should provide more than just the most likely output, they should also return a distribution over possible outputs.

8.1.2 Weight Interpretability

Another major drawback to using ANNs for high assurance control involves *weight interpretability*. ANNs form function outputs by passing input values through a network of nodes, each with a set of interconnections and weights. The output is formed by summing the weights of interconnected nodes. Such a structure can be extremely difficult to interpret, and although we can query the network with a specific input to see its output, it is difficult to understand exactly why the network is generating the specific output. Thus, it can be difficult to look at a given network and determine that it has found a "correct" or "safe" policy.

8.1.3 Output Generalization

The next major drawback to using ANNs for high assurance control involves *output generalization*. The weights of an ANN are learned through training examples rather than by specifying what the outputs should be. This can actually result in improved performance, since hand coded rules can be very brittle to unexpected inputs. If the input to the network is not in its training set, the weights will hopefully return a reasonable output based on the similar examples that it has seen. The process of determining appropriate outputs in areas of the function for which there is no training data is known as generalization. Yet, this seeming advantage can also be problematic in high assurance control, since the network's behavior is often unpredictable when given unexpected inputs. People who deal with high assurance systems often prefer techniques where the outputs are directly specified rather than learned for this reason. Further, because an ANN's policy is not easily interpretable by looking at the weights, this can make this problem of network predictability in novel situations even more problematic.

8.1.4 Expert Knowledge Inclusion

Some of these problems could be solved by *expert knowledge inclusion*. The user could provide some simple safety suggestions such as "a is less likely to be correct when b is observed," or, in some high assurance situations, stronger rules could be provided such as "never pull the trigger when looking down the barrel" even in the absence of training data telling you what might happen if you did. This could greatly reduce the uncertainty of the system and thereby improve the applicability of ANNs to high assurance control. Unfortunately, the very problem that made the network weights un-interpretable also makes it difficult to provide the network with such a bias. Since it is difficult to determine the behavior of the network given its weights, it is equally difficult to determine a set of weights that will produce a certain behavior. Thus, traditional ANN learning is limited by information in the data, and it can be difficult to provide external information to the system.

8.1.5 The Sufficiency of Training Data

The next difficulty involves the *sufficiency of training data*. ANN learning improves with training data, however, it is difficult to determine how much data is necessary to be certain that the network is likely to perform within safety limits. This is also due to the lack of confidence reporting in the network outputs. Attempts to answer these questions with PAC learnability and VC dimensions have provided some loose bounds, but have failed to answer these questions in an average case [40].

8.1.6 The Bayesian Solution

Our proposed solution to the above problems is to recast the supervised learning paradigm in terms of a graphical model and decision theory (The UBM and UBDTM).

In previous chapters, we have shown that Bayesian inference as a supervised learning training algorithm is appealing since it is generic, well reasoned, produces the distributional outputs necessary for decision making, and when mixed with utility theory it is optimal with respect to expected utility given the prior assumptions made. Perhaps one reason why this approach is not more popular is because most machine learning technique's representations for F were not originally designed to be trained by Bayesian inference. For example, the weights for a general neural network could be learned using a Bayesian approach nearly equivalent to our graphical model [24]. Unfortunately, since the neural network model was never designed to be learned in this manner, Bayesian inference in this case did not provide a closed form solution, and required an approximate inference method. Although from a Bayesian perspective this is the correct way to train a neural network, techniques such as backpropogation have remained more popular in the community because the Bayesian inference techniques necessary are approximate and computationally intensive.

To truly make the graphical model for function approximation and classification practical, a representation of the function class is needed that is flexible, which covers many of the standard function classes encountered in real world applications, and which can be solved in closed form so that the inference technique is quick and simple. Furthermore, in order to be able to deal with large training set sizes, it seems reasonable to require that the function representation be "eager" like decision trees or neural networks rather than "lazy" like k-Nearest Neighbor, locally weighted regression, or like case-based reasoning [74, p 244]. One simple function class representation that meets all of the above requirements is based on the CMAC network typology.

We will show that when the CMAC ANN is trained using the UBM as a guide: the network will produce a true posterior distribution for the weights and for the outputs, thereby providing a measure of confidence in the network's outputs; prior distributions can be used as a mechanism for providing input from a human expert; the CMAC network topology is easily interpretable, making this technique for the inclusion of expert knowledge practical; the weight interpretability also allows the learned policy of the network to be directly analyzed for correctness; and finally, this particular network topology causes the Bayesian approach to have a closed form solution.

In Section 8.2 we will introduce the standard CMAC. We will discuss the traditional CMAC training procedure and show that it can be interpreted as an iterative technique for approximating the maximum likelihood estimation of the network weights given the training data. In Section 8.3 we propose a new Bayesian learning procedure for the CMAC (which we call the BCMAC), and discuss its theoretical advantages to the traditional CMAC. We will show how this technique can be trivially expanded to include arbitrary kernel functions (KCMAC). We will also discuss various techniques for setting the priors. In Section 8.4, we will empirically show that the BCMAC outperforms the simple CMAC on several illustrative functions, and on a variant of the Abalone data set. We will test a simple kernel variant, and methods for setting the priors. We will demonstrate the system's ability to report confidences in its outputs, an essential ability for the system's application to decision making. Finally, in Sections 8.5 and 8.6, we will discuss the implications of this approach, summarize, and propose future work.

8.2 The Traditional CMAC

The CMAC is a well known ANN topology which has been shown to be useful for many applications [1]. It is modeled on the human cerebellum, and functions by mapping weights $w[i]$ to tiles which are interpreted spatially (see Figure 8.1). Inputs are mapped to the correct bins by means of an association function $b[i](x)$, where $b[i](x) = 0$ when x does not fall within the spatial region assigned to bin i and where $b[i](x) = 1$ when it does.

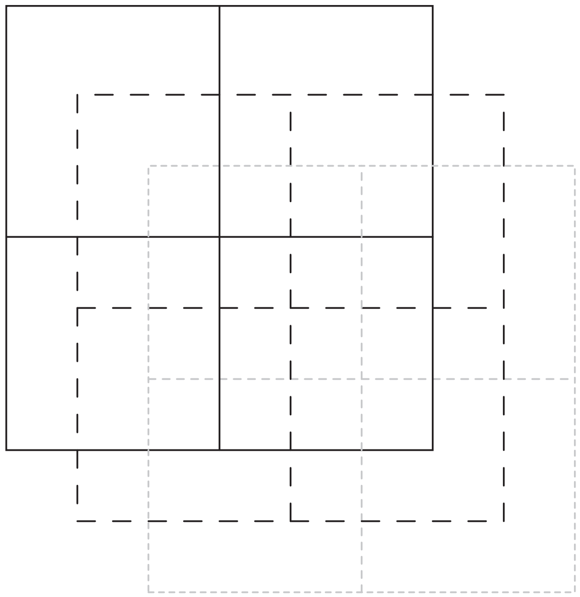


Figure 8.1: Tile structure for a CMAC with three layers, and four tiles per layer.

The output y of the network at any position x is the sum of the weights for the tiles that overlap that position,

$$f_{CMAC}(x) = \sum_i w[i]b[i](x).$$

The error at location x is:

$$e(x) = f_{CMAC}(x) - f_{observed}(x).$$

Traditionally the weights are then updated as follows:

$$w[i] \leftarrow \alpha \frac{e(\dot{\mathbf{x}})}{\sum_i b[i](\dot{\mathbf{x}})},$$

where α is a learning rate.

Because of its structure, a CMAC can have more unique regions than it has weights, which has been shown to provide some minimal relief from dimensional explosion [1]. Furthermore, the structure of the tiles causes smoothing across regions of the state space such that the CMAC's representational bias assumes that regions that are spatially near each

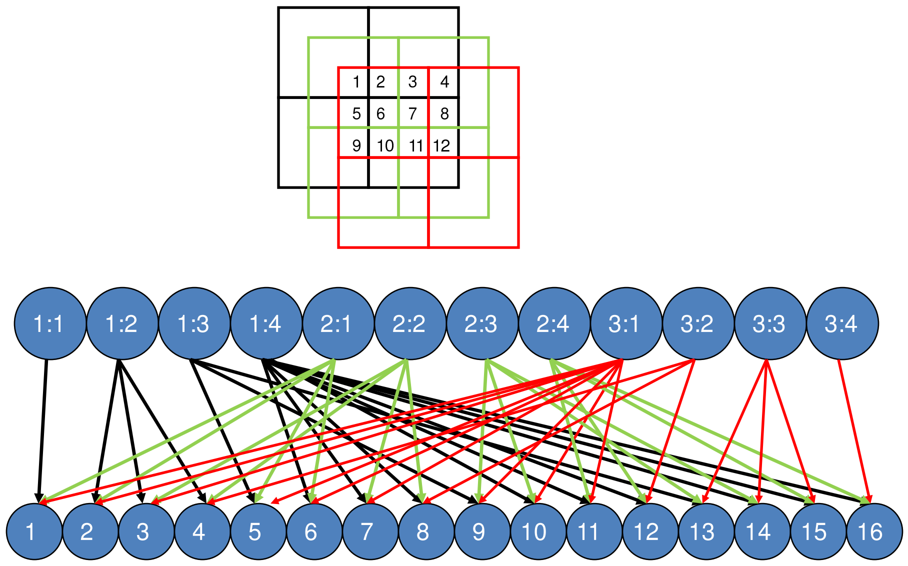


Figure 8.2: The CMAC viewed as an ANN with a specific spatially interpreted topology.

other tend to have similar values, a common and useful bias. Most importantly, the values at each position are computed with a sum which will greatly simplify statistical learning of the CMAC.

However, several questions remain: Although the system is motivated by the human cerebellum, is there a statistically grounded theoretical motivation behind the traditional CMAC update procedure? How many times should the training data be sent through the system? How should the learning rate be set? After training, how certain is the CMAC in its results? How can arbitrary kernels be used with the CMAC? We will attempt to answer these questions.

8.2.1 A Statistical Interpretation of the Simple CMAC

The traditional CMAC can be interpreted within a statistical paradigm. In order to create a statistical foundation for CMAC learning, it can be helpful to first think of a simple example.

Consider the case of a CMAC with a single layer with non overlapping regions. This would effectively be equivalent to multidimensional statistical "bins." In order to reduce mean squared error, the weight should be the mean of all the values that will fall within that bin in the test set. One approximation for this value would be to take the maximum likelihood estimate using the training set as follows:

$$w[i] = \frac{\sum_j \dot{y}_j b[i](\dot{\mathbf{x}}_j)}{\sum_k b[i](\dot{\mathbf{x}}_k)},$$

essentially the average of all the training set values within the weight's spatial influence.

In fact, if we kept a unique α for each bin, and if $\alpha(i) = 1/n(i)$ where $n(i)$ is the number of training examples seen so far in bin i, then a single layer CMAC would return the maximum likelihood result by keeping a running average of all the data within each relevant bin. It turns out that this also holds for the multi layer case where the update rule can be interpreted as an approximation to an iterative technique for finding the maximum likelihood estimation of a multivariate normal. These observations motivate our multivariate Bayesian procedure which follows.

8.3 A Bayesian Learning Procedure for the CMAC

We have seen that the traditional CMAC learning algorithm is an approximation to a maximum likelihood estimator for the weights. We will now create a Bayesian technique for computing the parameters of a CMAC representation for F.

First, we assume that the function that we are trying to approximate is stationary, and that our observations $\mathbf{y}$ have linear Gaussian noise with covariance $\mathbf{\Sigma}_y$. For this work we will assume that all observations are of equal quality. We initialize $\mathbf{\Sigma}_y$ as $\mathbf{\Sigma}_y = \sigma_y^2\mathbf{I}$ where σ_y is a constant that models how much we should expect the output of each y to vary from the true value of the function (for example, this can account for sensor noise). We also assume that the training and test sets are identically distributed, and thus $p(\dot{y}|\dot{\mathbf{x}}, f) = p(\ddot{y}|\ddot{\mathbf{x}}, f)$, a common assumption in Machine Learning.

Since a CMAC models functions using a sum of weights, it is convenient to model the prior for F as follows:

$$p(f) = \begin{cases} p(\mathbf{w}) & \text{if } f \in \{\lambda \mathbf{x}.g(\mathbf{x}, \mathbf{w}) | \forall \mathbf{x} \in \mathbb{R}^D.g(\mathbf{x}, \mathbf{w}) = \\ & \sum_{i \in \mathcal{T}} w[i]b[i](x)\} \\ 0 & \text{otherwise} \end{cases}$$

$$p(\mathbf{w}) = N(\mathbf{w}|\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$$

where $\mathcal{T}$ is a list of all tiles in the CMAC and where $\mathbf{w}$ is a vector-valued random variable with a multivariate normal distribution with prior means $\boldsymbol{\mu_0}$ (corresponding to the weights of each tile) and prior covariance $\boldsymbol{\Sigma}_0$.

The distribution of $\mathbf{X}$ does not need to be modeled since its values are given. Then, the relationship between $\mathbf{X}$, Y, and $\mathbf{F}$ can be modeled as follows:

$$p(y|\mathbf{x}, f) = N(y|f(\mathbf{x}), \boldsymbol{\Sigma}_y).$$

At this point it is helpful to note that $p(y|\mathbf{x}, f)$ can be rewritten as follows:

$$p(y|\mathbf{x}, f) = N(y|\mathbf{H}\mathbf{w}, \boldsymbol{\Sigma}_y),$$

where $\mathbf{H}$ can be thought of as an association matrix. $\mathbf{H}_{i,j} = 1$ if tile j influences the training example i. That is, $\mathbf{H}_{i,j} = b[j](x_i)$. Notice that $\mathbf{x}$ is used to construct $\mathbf{H}$, and then drops out of the equation. Arbitrarily complex kernels can be represented by simply modifying the $\mathbf{H}$ matrix such that $\mathbf{H_{i,j}} = k[j](x_i)$, where $k[j](x_i)$ is not binary but rather is a function of the distance from x_i to the center of tile j. Thus $\mathbf{H}$ encodes the amount of influence each tile has on a given x_i. This creates a simple and intuitive method for implementing arbitrary kernel functions for CMAC ANNs.

The weights are related to our observations according to a multivariate normal model [25] with prior parameters $\boldsymbol{\mu}_0$ and $\boldsymbol{\Sigma}_0$. The parameters of the posterior distribution are then

$$\boldsymbol{\mu}_1 = \boldsymbol{\mu}_0 + \mathbf{K}_1(\mathbf{y} - \mathbf{H}\boldsymbol{\mu}_0),$$

and

$$\boldsymbol{\Sigma}_1 = (\mathbf{I} - \mathbf{K}_1\mathbf{H})(\boldsymbol{\Sigma}_0),$$

where

$$\mathbf{K}_1 = (\boldsymbol{\Sigma}_0)\mathbf{H}^T(\mathbf{H}(\boldsymbol{\Sigma}_0)\mathbf{H}^T + \boldsymbol{\Sigma}_y)^{-1}.$$

Now the probability of the weights given some training data is distributed as follows:

$$\rho(\mathbf{w}|\mathscr{D}_{train}) = N(\mathbf{w}|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1).$$

In the more complex case where the function can change over time in a linear manner with Gaussian noise these equations become identical to the Kalman Filter Equations where the parameters at each time step are:

$$\boldsymbol{\mu}_{t+1} = \mathbf{G}\boldsymbol{\mu}_t + \mathbf{K}_{t+1}(\mathbf{y}_{t+1} - \mathbf{H}\mathbf{G}\boldsymbol{\mu}_t),$$

and

$$\boldsymbol{\Sigma}_{t+1} = (\mathbf{I} - \mathbf{K}_{t+1}\mathbf{H})(\mathbf{G}\boldsymbol{\Sigma}_t\mathbf{G}^T + \boldsymbol{\Sigma}_x),$$

where

$$\mathbf{K}_{t+1} = (\mathbf{G}\boldsymbol{\Sigma}_t\mathbf{G}^T + \boldsymbol{\Sigma}_x)\mathbf{H}^T(\mathbf{H}(\mathbf{G}\boldsymbol{\Sigma}_t\mathbf{G}^T + \boldsymbol{\Sigma}_x)\mathbf{H}^T + \boldsymbol{\Sigma}_y)^{-1},$$

where $\mathbf{G}$ is the linear transition matrix and the Gaussian noise has covariance $\boldsymbol{\Sigma}_t$.

This observation means that, given a prior over CMAC weights and some training data, a well known and widely studied filtering technique can be applied to solve in closed form for both the posterior distribution over the CMAC weights, and for the posterior predictive distribution of the CMAC outputs. We call this technique the Bayesian-CMAC or BCMAC.

One major contribution of this technique is that it can be used to train arbitrary kernel functions. Traditionally, training arbitrary kernel functions was difficult since it was unclear how to update the weights given the kernel for some kernel functions. To incorporate kernels into the BCMAC, the kernel simply dictates **H** and inference proceeds normally. Thus, kernel functions that may have been unusable with traditional CMAC techniques can now be implemented with ease. We call the BCMAC with an **H** representing a kernel function other than the simple overlapping tiles the Kernel CMAC or KCMAC.

Learning in the BCMAC and KCMAC is simply Bayesian inference in the Kalman Filter. This allows us to solve common training issues such as the problem of finding the best number of passes through the training data (data now passes through the system exactly once). This technique also allows us to place a prior on our BCMAC weights, which can be beneficial if we empirically or subjectively know something about the function before we begin observing data, making the inclusion of expert knowledge simple.

Perhaps most importantly this technique can simply produce an actual posterior distribution over possible outputs given the training data, $p(\ddot{y}|\ddot{\mathbf{x}}, \mathscr{D}_{train})$. Equation 2.9 tells us that this is a large and often difficult integral. However, because in this case we are dealing with the sum of a set of normal random variables this can be solved simply in closed form:

$$p(\ddot{y}|\ddot{\mathbf{x}}, \mathscr{D}_{train}) = N(\ddot{y}|\mathbf{H}\mathbf{w}_1, \mathbf{H}^T\mathbf{\Sigma}_1\mathbf{H}). \tag{8.1}$$

This makes it possible to know how much to trust the BCMAC's outputs in each part of the function. This can reveal when more data is needed and where such data would be most useful. Such a confidence is also essential in order to use the outputs of the algorithm to make optimal decisions from a decision theoretic standpoint.

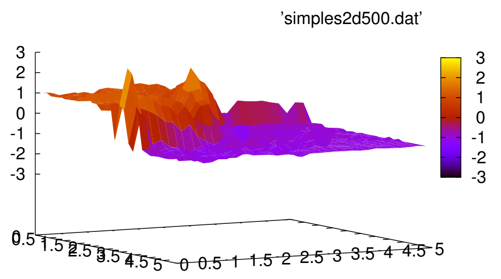
Priors can come from many places. If detailed information is known *a-priori* concerning the function, then a detailed prior can be provided, allowing the system to perform well with low amounts of data. This is possible because of the interpretability of weights in the CMAC topology. If less is known, then a wide subjective prior can be chosen. If the prior is sufficiently wide, it will have little effect on the posterior. This will cause the system to require more sample points before it becomes confident in its outputs. Although the data in a specific tile may be sparse, the cumulative data over the entire function will often be more rich. By analyzing the cumulative data, it is possible to use that information to produce an "empirical" prior for each tile which can often improve performance over a wide subjective prior. Theoretically, a hierarchical model would be more statistically correct, but at the expense of a more complex model, which is unwarranted in this case.

8.4 Results

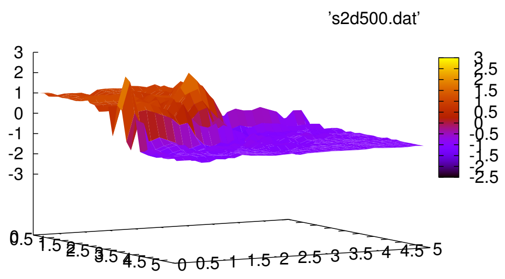
To test the power of the BCMAC and KCMAC, we performed several experiments. We used several synthetic functions, and tested on a variant of the UCI Abalone data set. We analyzed the feasibility of setting the priors on the weights for individual tiles using Empirical Bayes over the set of all tiles in the problem. We also explored the ability of the algorithm to express confidences in its outputs.

8.4.1 CMAC, BCMAC, and KCMAC Compared

First, we chose to illustrate the algorithm's behavior with a simple two dimensional function called step2d because the results could easily be visualized. The function was designed to be difficult for a CMAC representation. If $x_1^2 + x_2^2 < 10$ the function returns 1 otherwise it



(a)



(b)

Figure 8.3: CMAC output, viewed from the side to show the jagged overshoot artifacts. (a) is the simple CMAC while (b) is the BCMAC. Notice that the BCMAC has smaller jagged overshoot artifacts along the boundary between high and low outputs.

returns -1. This causes a steep transition, and a curved boundary to that transition, both of which are difficult features for a CMAC to capture. Points in the function fall in the domain between 0 and 5 for both x_1 and x_2 We created 100 different training sets by drawing 100 different sets of 500 examples each from the function, drawn uniformly. We then trained both a traditional CMAC and the BCMAC (with a wide subjective prior mean $\mu_o = 1$ and covariance $\mathbf{\Sigma}_o = 20I$) on these training sets, and compared the mean of their outputs to a single test set of size 20,000.

Figure 8.3 gives example outputs of the CMAC and BCMAC after training on the step2d function. The jagged overshoot artifacts are common in CMAC function approximation when attempting to match a steep discontinuity. Note that the artifacts are larger for the CMAC than they are for the BCMAC. The results are summarized numerically in Table 8.1. Although the error can vary substantially from one run to another depending on the quality of the training set chosen, the ratio of the standard CMAC error over the Bayesian CMAC error is consistent. This means that although the quality of the training set is an important factor in obtaining a good result on this function, in our experiments the Bayesian CMAC always outperformed the standard CMAC on whatever training set it was given.

Table 8.1: Sum^2 error between the CMAC outputs and the actual function outputs on the test set, for the traditional CMAC and for the Bayesian CMAC, and the ratio between these errors.

Run	Simple CMAC Sum^2 Error	BCMAC Sum^2 Error	CMAC/ BCMAC
1	1,857.654	1,459.136	1.27
2	1,917.688	1,412.675	1.36
3	1,323.653	1,141.215	1.16
4	1,593.757	1,326.135	1.20
5	1,728.339	1,401.232	1.23
6	1,660.846	1,327.345	1.25
7	1,621.135	1,292.013	1.25
8	1,472.336	1,273.971	1.16
9	1,271.723	1,121.361	1.13
...	...	...	...
100	1,950.859	1,534.583	1.27
Average	1,581.205	1,305.917	1.21

We also ran the same comparison between the BCMAC and the KCMAC (with a Gaussian based kernel function). Again the values varied widely, but the ratio was consistently larger than 1 (see Table 8.2). This indicates the potential benefits of kernel functions and the advantage of being able to train them simply.

This process was then repeated for a variant of the two-dimensional egg carton function $y = sin(x1 * 2.5) + sin(x2 * 2.5)$, 2dEgg, and its three dimensional variant $y = sin(x1 * 2.5) + sin(x2 * 2.5) + sin(x3 * 2.5)$, 3dEgg. Again, wide subjective priors were used. These results are summarized in Table 8.3. In the interests of space, only the averages will be reported here, however, the full tables were much like those shown for step2d, where the individual values varied greatly depending on the training set chosen, but where the ratio between the results of the various methods was consistent.

We also evaluated the CMAC and KCMAC on the UCI Abalone data set [88]. This problem was chosen because of its continuous features and ordinal output which can be treated as continuous. All nominal features were dropped, as well as the many redundant features, leaving the most informative weight measurements (shell weight) and the most

Table 8.2: Sum^2 error between the CMAC outputs and the actual function outputs on the test set, for the Bayesian CMAC and for the KCMAC with a gaussian kernel function, and the ratio between these errors.

Run	BCMAC Sum^2 Error	KCMAC Sum^2 Error	BCMAC/ KCMAC
1	1,459.136	989.768	1.47
2	1,412.675	949.683	1.49
3	1,141.215	847.299	1.35
4	1,326.135	934.228	1.42
5	1,401.232	881.560	1.59
6	1,327.345	931.826	1.42
7	1,292.013	950.709	1.36
8	1,273.971	1,026.587	1.24
9	1,121.361	944.116	1.19
...	...	...	...
100	1,534.583	1,126.498	1.36
Average	1,305.917	944.614	1.38

Table 8.3: Average Sum^2 errors for the Simple CMAC, the Bayesian CMAC and the Kernel Based CMAC with a gaussian kernel, averaged over 100 different training sets on several different benchmark functions.

Function	CMAC	BCMAC	KCMAC
step2d	1581.205	1305.917	944.614
2dEgg	510.409	499.885	408.001
3dEgg	483.814	481.560	162.050
abalone	0.9063	0.7543	0.7349

informative size measurements (diameter). The first 1500 examples were used as test data, while the remaining examples were used as training data (see Table 8.3). Since we had less intuition for selecting subjective priors on this problem, priors were set "empirically" as discussed in Section 8.3.

No principle component analysis was used, which would have improved the performance of all three techniques. The point was the relative performance of the three techniques not to achieve the best performance on this particular data set.

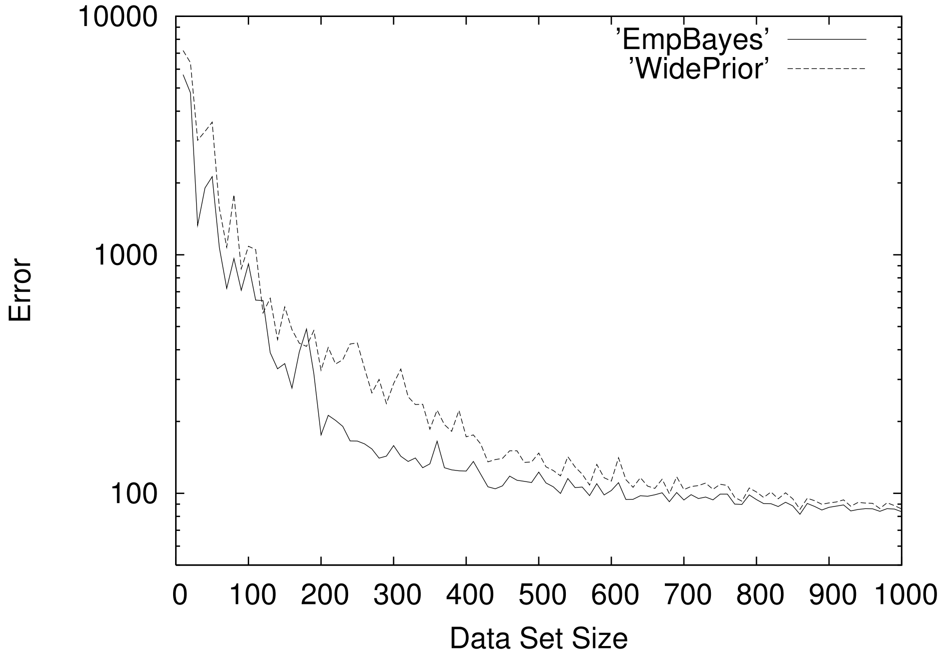


Figure 8.4: Using Empirical Bayes to produce the priors for the CMAC. For low sample sizes the Empirical technique outperformed the wide subjective prior.

8.4.2 Empirical Bayes

To analyze the potential of an "empirical" prior we compared an empirical prior to a wide subjective prior (with a tile mean of 1 and a variance of 20) on the step2d function and found that, as expected, the empirical prior performed better for low sample sizes (see Figure 8.4). Theoretically, this result could be improved upon by using a hierarchical technique at the expense of a more complex Bayesian model which could no longer be solved in closed form. Empirical Bayes can be seen as an approximation to the more correct hierarchical version which allows us to maintain the closed form nature of our solution. In this case the approximation is sufficient.

8.4.3 Expressing Confidence

We have seen that when the BCMAC reports its most likely output values it consistently has a lower mean squared error than the traditional CMAC on all the functions that we

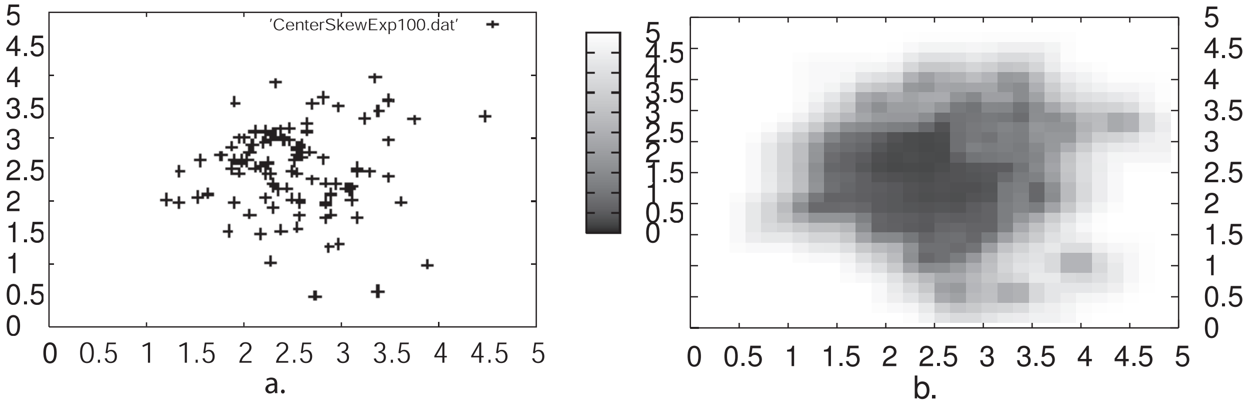


Figure 8.5: a) Training points for the BCMAC, selected from a function exponentially biased towards the center. b) Tile variance after training with data points in part a, darker areas represent regions of less variance. Note that the algorithm is more confident in areas where more data has been observed.

tested. However, the real power of this technique is not in its lower error alone. Because the training technique for the BCMAC is Bayesian in nature, a full distribution is available for the predictive posterior. If more data has been encountered in one region, then the algorithms will be more confident in their answers in that region. If less data has been encountered in another region they will be less confident in their answers in that region.

To illustrate, we again used the step2d function to demonstrate BCMAC's ability to return confidences in its results. We generated data that is exponentially skewed towards the center of the domain. After training, we graphed the algorithm's confidence (see Figure 8.5). This ability to report confidence in a statistically meaningful way is an important advantage of our algorithms. Such a confidence is essential in decision theory in order to make optimal decisions with respect to maximum expected utility [25]. Because this value quantifies risk, it is an essential part of any high assurance system. Furthermore, the distributional output can be extremely useful in situations such as active learning.

8.5 Discussion

Notice that the posterior predictive of the BCMAC (Equation 8.1) is a normal distribution. Returning to our mushroom example, assume that mushrooms with toxicity levels below 10 are harmless, but with levels over 10 their negative effects begin to grow rapidly. Assume that we have been lost in the forest for 10 hours and are beginning to be hungry. Now we come across a patch of mushrooms. Whether or not we should eat the mushrooms depends on our utility function. Because we are hungry there is a slight negative utility (-1) if we don't eat the mushroom. If we eat the mushroom, the utility will depend on the toxicity level of the mushroom as follows:

$$\begin{aligned} U(notEat, tox) &= -1 \\ U(eat, tox) &= \begin{cases} 1 & \text{if } tox < 10 \\ -5 * tox + 51 & \text{if } tox > 10 \end{cases} \end{aligned}$$

Now, if our mushroom toxicity predicting algorithm only returns the maximum likelihood result, as most neural networks do, and if it returns 10.1 as the most likely toxicity. If this were the true toxicity, the utility of eating these slightly toxic mushrooms would be .5. The utility of not eating the mushrooms is -1. At first glance it would seem that the best thing to do would be to eat the mushrooms. However, this all depends on how sure the algorithm is in its prediction. If the most likely toxicity is 10.1, then there is some chance that the actual toxicity is 15, or perhaps even 20, in which case eating the mushrooms would be very dangerous. The problem here is not that we are uncertain, but that we are unable to quantify our uncertainty and risk. It is impossible to effectively make this decision without more information.

If we used the BCMAC, the posterior predictive value would be a normal distribution. Suppose that our algorithm returned a mean of 10.1 with a variance of 3 for the toxicity of the mushroom. Now the expected utility of eating the mushroom can be computed by integrating as given in Equation 2.11. The expected utility of eating the mushroom can now be computed and is -5.2, and the best action is to not eat the mushroom even though the utility at the maximum likelihood value would seem to indicate that we should eat the mushroom. If we had been lost longer, the utility of not eating the mushroom could drop considerably as the likelihood of starving increases. Once this value drops below -5.2, the optimal decision will be to eat the mushrooms despite the risk.

If we were designing the ANN for mushroom toxicity detection, we might have noted that for many common mushrooms the variance returned was simply too high in too many common cases. This could be caused by a naturally noisy function, but if the amount of intrinsic noise in the function is such that the designer believes that the ANN should be able to do better, then this would indicate that either more data is needed, or that the resolution of the BCMAC needs to be increased. Furthermore, if there is more variance in one part of the output than in another, then this could indicate not only that more data is needed, but also where data would be most useful. This can be helpful in active learning situations. Thus, because the network reports its confidence, the risk of using the system can be evaluated beforehand and improvements made when appropriate.

Another way of lowering the variance of the predicted outputs would be to provide the algorithm with expert human knowledge. If a human expert knows that dark mushrooms of a certain size are always very poisonous, while light mushrooms of the same size are never poisonous, then that information can easily be introduced into the priors for the weights. Because the weights are spatially interpretable, it is clear which weight priors need to be adjusted to achieve the desired effect. Furthermore, the strength of the prior is determined by the variance on the weights, allowing our expert to express how confident he is in his information.

This ability is illustrated by the above step2d experiments. In those experiments we set the mean to 1 and the variance to 20 because we did not want to give the BCMAC an unfair advantage. However, since we knew beforehand that the values would vary between -1 and 1 we could have selected a more informed prior with a mean of 0 and a standard deviation of 1. It would have been possible to do even better by setting the prior mean of all tiles with centers $x1^2 + x2^2 < 10$ to 1 with all others set to -1, and then lowering the variance accordingly, in which case little or no learning would have been necessary because of the accuracy of the prior.

8.6 Conclusions and Future Work

By modeling the regression and classification process as a graphical model we have shown that the traditional CMAC update rule is actually approximating the maximum likelihood estimation of the weights and produced a Bayesian variant of the CMAC that can be solved in closed form. We call this technique the BCMAC. This approach also allows for the training of arbitrary kernel functions. We call the kernel variant of the BCMAC the KCMAC. We have shown that these techniques outperform the traditional training techniques for learning the weights of a CMAC for several example functions including step2d, 2dEgg and 3dEgg, as well as for the Abalone data-set. Furthermore, the BCMAC and the KCMAC not only determine the most likely weights, but they also give a statistically correct estimate of how certain they are of their weights and outputs at every position, which is essential for making good decisions from a decision theoretic perspective.

We have shown that the BCMAC and KCMAC have several advantages in high assurance systems. Their weights are interpretable, their priors can be used to introduce expert knowledge, and their distributional outputs can be used to quantify risk, to determine if they have enough data, and to determine if the algorithm is sufficiently accurate to be used in a given context. This can greatly improve the applicability of ANNs to high assurance situations.

8.6.1 Future work

The posterior covariance term $\mathbf{\Sigma}_1$ can become very large as it is the covariance between each tile in the system. However, $\mathbf{\Sigma}_1$ is usually very sparse. Experimentation is required to determine how sparse it is in practice on various functions. Sparse matrix techniques should be evaluated to determine whether the BCMAC can scale up to larger sized problems, and to determine how sensitive the system is to even sparser approximations to this covariance. Other future work involves implementing a Multinomial version of the system for discrete classification, and making a more detailed comparison to other Bayesian regression models of F. Once this is done we can explore the real advantages of such Bayesian techniques. In the graphical model, meta learning could be performed by creating a hierarchical model of F. Furthermore, transfer could be performed by empirical or hierarchical learning of the meta parameters. Because we know the confidence at each position in feature space, active learning in this model could be reduced to the Expected Value of Sample Information (EVSI) (see Chapter 6), which would avoid the issues normally encountered with query by committee and query by uncertainty.

Chapter 9

Application to Active Learning, Modeling the Annotation Process for Ancient Language Corpus Creation

Abstract

This Chapter will provide an example of the use of active learning in a variant of the supervised learning case. In corpus creation human annotation is expensive. Annotation costs can be minimized through machine learning and active learning, however, there are many complex interactions among the machine learner, the active learning technique, the annotation cost, human annotation accuracy, the annotator user interface, and several other elements of the process. To date these interactions have been poorly understood. We use a Bayesian decision-theoretic model of the annotation process (an adaptation of the UBDTM) suitable for ancient corpus annotation that clarifies these interactions and can guide the development of a corpus creation project.

Publication: The majority of this Chapter was previously published in [14], and that publication appears here largely unchanged. Minor errors have been fixed, and some attempt to standardize notation has been made, but the chapter still largely stands on its own. This approach has lead to a small amount of redundancy, and some material presented in this chapter repeats information presented elsewhere in the dissertation. The initial publication was primarily theoretical and motivational, and was written before the later results were obtained. It presented the initial theoretical underpinnings that were to guide the Syriac project. This theoretical work was largely justified and verified in later publications of which

I was only a co-author [41, 93, 92, 36, 37], and these later results will only be summarized here, however, it is important to understand that many of these later results grew out of the theoretical ideas presented here.

9.1 Introduction

The ideas in this Chapter arose from a project to develop an electronic corpus and concordance of ancient Syriac literature. We will use this project to illustrate many of the ideas in this chapter. The Syriac project at Brigham Young University involves many individuals from several departments including Linguistics, Computer Science, and the Center for the Preservation of Ancient Religious Texts. The project team also includes scholars from Oxford and Princeton Universities. Syriac texts have been transcribed manually by teams of Maronite, West Syrian and East Syrian Christians and Monks located in Lebanon, Rome, Iraq, Chicago and Oxford. The proximate goal of this project is to produce a corpus annotated with part of speech morphological data for the writings of the fourth century Syriac poet-theologian Ephrem the Syrian (d. 373). This initial corpus is approximately half a million words in size. A further four million words have been added to the corpus in draft format. These texts originate from the third to the thirteenth century. However, the majority of the texts are from the fourth to the seventh centuries, the so-called Classical period of Syriac literature. It is the long-term aim of the project to build a comprehensive corpus of Syriac literature, working diachronically through the available texts. Much of Syriac literature has already been published, and these published texts are used in the corpus. However, a great deal of Syriac literature is available only in manuscripts. It is impossible to precisely estimate the size of the final corpus; however, it is not improbable that the corpus extends to over 30,000,000 words.

We do not have the resources to fully annotate a corpus of this size with morphological tags. We are taking a pragmatic approach to annotating texts for the corpus. The first stage is to prepare a draft transcription with machine annotation. Texts will then be proofread and annotated by hand as scholarly interest is raised to a sufficiently high level to complete the work. Many texts in the corpus may never be fully proofread or annotated. Some text collections, beginning with Ephrem, will, however, be thoroughly proofed and tagged, sufficiently to produce a full print concordance, and then published in print and internet form. The remainder of the work will be published only on the internet. We will refer to these two portions of the corpus as the print corpus and the internet corpus, respectively. A higher level of accuracy will be required for the print portion of the corpus than for the internet portion of the corpus.

The production of electronic corpora for ancient languages involves several “annotation” tasks. Transcription, morphological part of speech tagging, grammatical parsing, and semantic tagging can all be seen as annotation tasks. For example, in transcription the user takes an image and labels (or annotates) the image with transcribed text. In part-of-speech tagging the user takes a transcribed text and annotates the text with parts of speech. Thus, annotation is central to each step in the creation of a useful electronic corpus. The goal of our part of the Syriac literature project is to reduce human morphological part of speech annotation cost as much as possible through the appropriate use of machine learning and active learning techniques. We also seek to achieve lower error rates than could be achieved through human annotation alone and to appropriately balance the value of annotator time on the print corpus with the value of annotator time on the internet corpus. The goal of this Chapter will be to produce a sound theoretical framework for solving these practical problems.

9.1.1 Issues in Corpus Creation

Human annotation can be very expensive, and this expense is often the limiting factor in the creation of electronic corpora. Since ancient languages are generally less well known, their annotation requires more specialized language knowledge, which can make human annotation even more expensive.

One solution to this problem is to use machine learning to automatically annotate the data. Machine learning approaches are available for transcription (OCR or Optical Character Recognition), part of speech tagging, parsing and semantic role labeling. Unfortunately, machine learning approaches to annotation often have higher error rates than human annotation and they often require a large set of previously labeled data in order to "train" the machine learning model. Typically, the larger the initial training set the better the algorithm will perform. Often this initial training set is either nonexistent or extremely small, especially when dealing with ancient languages.

These problems with machine learning can be overcome by combining machine learning with human annotation. The goal of such a combination is to use the expensive but more accurate human annotation in the most beneficial way to lower the error rate of the entire annotated corpus as inexpensively as possible. Typically, the computer selects the examples to be annotated by the human that it believes will be the most beneficial. This process is called "active learning." Active learning is invaluable when there are insufficient resources to use a human annotator over the entire corpus. Even when there are sufficient resources to annotate the entire corpus by hand, many errors likely remain. Active learning can focus human effort for error discovery on the most problematic sections.

Several complex and poorly understood interactions arise when attempting to integrate human annotation with machine learning and active learning in the creation of a large annotated corpus. These interactions arise among the following components of the system:

1. The expense of human annotation
2. The machine learner
3. The active learning technique
4. How annotators are paid
5. The user interface
6. Human annotation error rates
7. Variability in error significance various portions of the corpus (in our case for the print vs. internet portions of the corpus)

In order to effectively integrate all these elements it is important to understand which elements affect which other elements of the system, and to know how these interactions occur.

For example, how we pay our annotators affects the cost of annotating a sentence and can affect the sentence selection of the active learner. User interface design can also affect the cost of annotating a sentence as well as the accuracy of the annotations gained. Since human annotations are not 100% accurate, their accuracy must be modeled in order to determine what we should believe and how sure we should be about what we believe given a set of human annotations. Human annotations compose the machine learner's training set, therefore, the model of human annotation accuracy should affect the outputs of the machine learner given the training set. There are many other such interactions, many of which have not been to date fully explored, understood, or implemented.

Understanding these interactions is important in order to answer several questions which are important for our project. How often should we employ a second annotator and where would that annotator's work be most effective? Is it better for the second annotator to annotate a completely unseen example, or to verify the work of an annotator whose answer disagrees with the machine learner's annotations? Should we attempt to model the differences in ability between various annotators? Should we attempt to learn the abilities of each annotator separately? Should we give the annotators sentences where we already know the answer in order to determine their abilities? If so, how many should we give them since such sentences do not contribute to annotating new material? How should we effectively deal with the fact that errors in one portion of the corpus are more important than errors in another portion?

9.1.2 Our Proposed Solution

In this Chapter we propose a Bayesian, decision-theoretic, model of the corpus creation process based upon the ideas of the AL-UBDTM. The model helps to answer the above questions and clarifies the above interactions. Given the model of the process we can construct the theoretically optimal techniques for answering many of the above questions. The optimal solutions to these questions can be computationally infeasible; however, the model provides a clear way of thinking about the problem. With the optimal solution in mind better heuristics can be developed which approximate the optimal solution. This approach (developing an optimal model and then approximating it to achieve a computationally tractable solution) has guided several advances in the annotation process in the Syriac language project as well as in many other machine learning projects.

In Section 9.2, we present a Bayesian, decision-theoretic, model of the corpus annotation process itself based upon the principles of the AL-UBDTM by extending the AL-UBDTM to deal with the sequential data of natural language, and propose an MEMM as an approximation to the computationally intense Bayesian solution. In Section 9.3 we further extend this model to incorporate active learning for sequential data, and propose several approximations for active learning that will be employed. In Section 9.4 we describe how utility in the model can be used to deal with situations where errors in one part of the corpus are more important than errors in another part (as is the case with the print and internet portions of our Syriac project). In Section 9.5 we use the model to illuminate how annotation costs affect the active learning technique. In Section 9.6 we discuss the interaction between the user interface, and the annotation cost, and illustrate this with a cost model obtained from a user study on English. In Section 9.7 we extend the model to incorporate human annotation error rates. In Section 9.8 we conclude by providing recommendations for the Syriac corpus creation project, and by briefly summarizing the further results that have followed after the initial publication of this work which have further validated and verified the concepts found here.

9.2 Modeling the Corpus Creation Process

We believe that the best way to think about the supervised machine learning (ML) problem is as a graphical model (or Bayesian Network) [16, 17, 13]. By thinking about the problem in terms of a Bayesian Decision Network, utility is handled explicitly and we can see the theoretically optimal solution to computing the probability of a tag $p(y)$ that will maximize

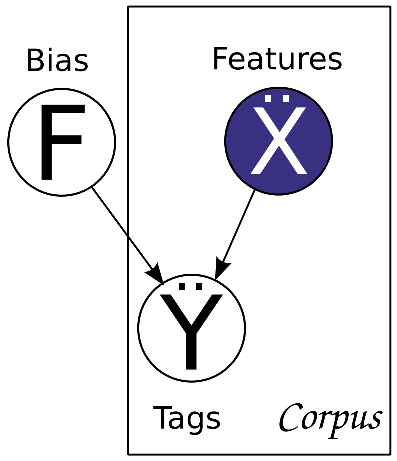


Figure 9.1: Using the UBM for POS-Tagging in NLP)

the expected utility [16]. Then, if this optimal solution is too computationally intense to compute, we can at least design heuristics in a more principled way.

We have shown that supervised learning inference is a special case of the UBM (see Section 2.2.3). Since POS tagging is an instance of making a decision (selecting a tag) based on supervised learning inference, we would expect the UBDTM to also model the NLP task of tagging, however, POS tagging is not just a case of simple supervised learning, and several additions to the UBM must be made to make it accurately model POS tagging inference. We will first discuss how the UBM and tagging relate, and then discuss the additions that are necessary to model POS tagging.

In simple supervised learning there are features $\mathbf{x}$, classes y, a training set, and a test set. In the case of POS tagging these are features, POS tags, and a corpus respectively (see Figure 9.1). The classes are simply the POS tags. The unlabeled corpus examples form a test set, while the labeled examples form the training set.

There are many ways to model end use utilities in the POS tagging task (see Figure 9.2), and they will involve knowing exactly how the final corpus will be used by the linguistic and religious scholars who will be the corpus' clientele. However, since the final corpus will have multiple possible uses, the rational behavior is to report a full distribution over POS tags instead of picking a single tag, thereby, delaying the end-use utility decision

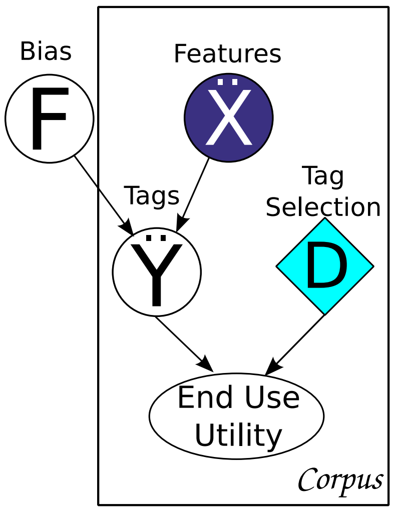


Figure 9.2: Using the UBDTM for POS-Tagging in NLP)

until the end use is determined as we discussed in Chapter 4. Unfortunately, linguistic corpus users are unaccustomed to probabilistically labeled data. This will, unfortunately, force us to report a single label for each word. The most reasonable decision seems to be to report the most likely label. We will, therefore, likely release the corpus in two versions, one tagged with probabilistic labels and the other tagged with the most likely labels.

It is hoped that this choice of the most likely labels will be helpful for many of the multiple end uses to which the corpus will be put. One way to look at this decision, motivated by the results of Chapter 4, is to realize that when there are multiple possible end uses (as in this situation), it can be reasonable to return a point estimate (such as the most likely class) so long as the average uses to which the corpus will be put justifies such a decision (see Corollary 4.3.2.2). This will be the case if on average over possible end uses, the expected utility of having a correct label and the expected cost of having an incorrect label are equally balanced for all POS tags. Thus, by returning the most likely tag, we are essentially making the above simplifying assumption about the potential end uses to which the corpus will be put. There are other subtle issues that are unique to the corpus creation process, which will impact active learning. For example, in our case we care more about errors in one part of the corpus than we do in another.

Thus, we will be using a variant of the UBDTM for modeling the corpus creation process, with some specific extensions to deal with some of the unique situations that arise in corpus creation and with a utility assumption that allows us to return the most likely class as our decision. We will now discuss the the extensions that are necessary to the UBDTM to make it model the sequential nature of most NLP problems.

9.2.1 NLP Extensions to the UBDTM

In many natural language processing (NLP) problems the network is much more complex than in the above examples. Primarily this is because data in NLP is inherently sequential. In machine learning there are many $\mathbf{x}$ and y pairs that have been observed (the training set) and a set of $\mathbf{x}$ and y pairs where the y node is unobserved (the test set). In the sequential tagging problem there are sequences of words (sentences) in the training and test sets. This means that the label at one time step can actually be part of the feature set for the next time step (as in Figure 9.3). Worse, the label at one time step can be part of the feature set of all the labels that follow. Obviously this model can be computationally intense, however, some common approaches such as a Maximum Entropy Markov Model (MEMM) can be viewed as providing approximations to this model. A Maximum Entropy (MaxEnt) model is a log-linear model whose parameters are those that create the distribution of maximum entropy that still satisfies the constraints imposed by the evidence found in the training data [91, 113, 112]. A gradient descent optimization procedure, such as LBFGS, can be used to find the parameters during training.

An MEMM is a Conditional Markov Model (CMM) in which a Maximum Entropy (MaxEnt) classifier is employed to estimate the distribution

$$p(y_i|\mathbf{w}, \mathbf{y_{1 \ldots i-1}}) \approx p_{ME}(y_i|w_i, \mathbf{v_i}, y_{i-1}, y_{i-2})$$

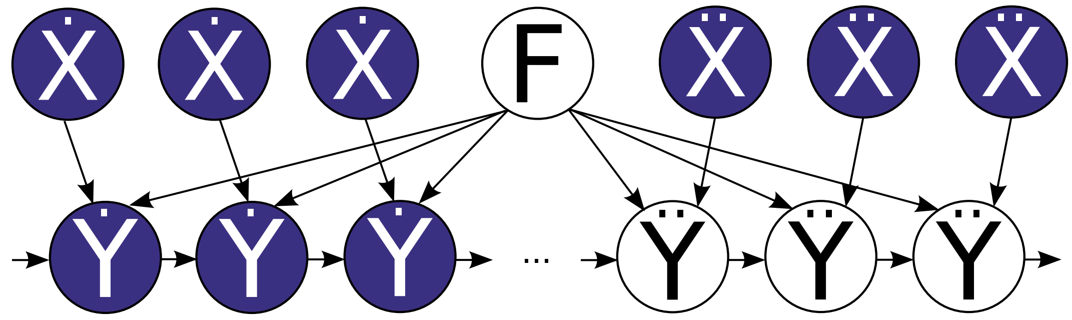


Figure 9.3: Using the UBM for Sequential POS-Tagging in NLP)

over possible labels y_i for each word $\mathbf{w}$ in the sequence. Here we have split $\mathbf{x}$ into the words $\mathbf{w}$ and the rest of the features $\mathbf{v}$. We use w_i to refer to individual words in a sentence $\mathbf{w}$. The model has access to any predefined attribute features (represented here by the collection $\mathbf{v_i}$) for each word in the sequence and to the labels of previous words $\mathbf{y_{1 \ldots i-1}}$ trained from labeled data. Our implementation employs an order-two Markov assumption so the classifier has access only to the two previous tags y_{i-}, y_{i-2}. We refer to the features $(w_i, t_i, t_{i-1}, t_{i-2})$ from which the classifier predicts the distribution over tags as "the local trigram context" [93]. State-of-the-art Part-of-Speech tagging results have been achieved with MEMMs [91, 113, 112]. Part of the success of MEMMs can be attributed to the absence of independence assumptions among predictive features and the resulting ease of feature engineering.

9.3 Active Learning

The sequential classification network can also be used to model active learning. The active learning task involves selecting an unobserved node (known as the test) and observing it. In the sequential classification network, this is done by selecting an un-annotated sentence which is then "observed" by having a human annotate it. The goal is to select a test that will provide the greatest improvement to the computer's estimates over the rest of the unlabeled classes. To select the test that will achieve the greatest improvement the sequential classification network is expanded into a decision network by adding decision nodes and utility measures

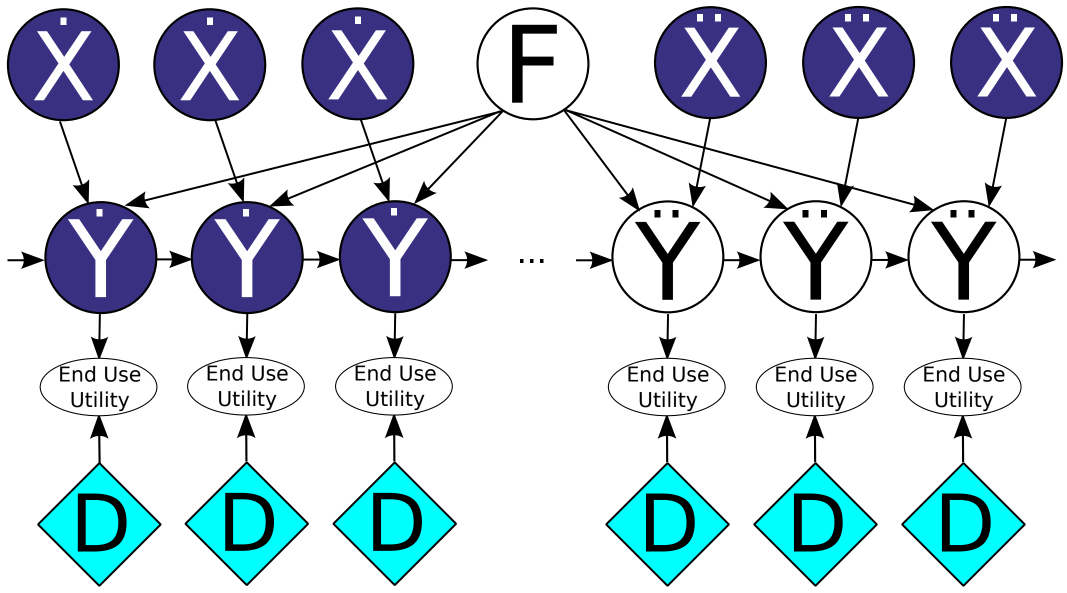


Figure 9.4: Using the UBDTM for Sequential POS-Tagging in NLP by adding utility to the sequential network to prepare for active learning. Notice the assumption of a deterministic function.

to the sequential version (see Figure 9.4) as was done previously to the non sequential version (see Figure 9.2). The decision to be made is what part of speech tag should be applied to each word. Since the objective of the annotation process is to correctly annotate the corpus, for now, we use accuracy as our measure of the utility of such a choice (the 0,1 gain utility function).

In this decision theoretic context, the active learning task is to select from the un-annotated data a sentence or word to be annotated from the corpus to maximize expected utility. The samples are drawn from a corpus which constitutes a pool, causing this to be a pool based active learning situation. The test set consists of the same corpus of words as did the pool set, so this is a pool based learning situation with a queryable test set. The goal is to select the sample which maximizes the expected gain in utility over the entire corpus of annotated and un-annotated data. This value is known as EVSI, or the Expected Value of Sample Information [90]. $EVSI(\breve{\mathbf{x}}_i)$ involves the computation of the expected improvement in utility that would result from revealing the value in a particular hidden node (in our case a specific $\breve{y}_i$) in a Bayesian network. For computational reasons, we will restrict ourselves to the greedy approximation. For POS tagging, and for other related problems, greedy approximations can do reasonably well. This greedy approximation to EVSI, which we will refer to as GEVSI can be computed as follows:

$$
\begin{aligned}
GEVSI(\breve{\mathbf{x}}_i) = & \sum_{\breve{y}_i \in \mathbb{D}_Y} p(\breve{y}_i | \breve{\mathbf{x}}_i, \mathscr{D}_{train}) \sum_{\ddot{\mathbf{x}}_k \in \mathscr{D}_{test}} \xi(\ddot{\mathbf{x}}_k) \max_{d \in \mathbb{D}_D} \sum_{\ddot{y}_k \in \mathbb{D}_Y} p(\ddot{y}_k | \ddot{\mathbf{x}}_k, \mathscr{D}_{train} \cup \{\breve{\mathbf{x}}_i, \breve{y}_i\}) U(d, \ddot{y}_k) \\
& - \sum_{\ddot{\mathbf{x}}_k \in \mathscr{D}_{test}} \xi(\ddot{\mathbf{x}}_k) \max_{d \in \mathbb{D}_D} \sum_{\ddot{y}_k \in \mathbb{D}_Y} p(\ddot{y}_k | \ddot{\mathbf{x}}_k, \mathscr{D}_{train}) U(d, \ddot{y}_k) \qquad (9.1)
\end{aligned}
$$

Intuitively we are taking the expectation over every possible result of the test $\breve{y}_i$ and computing the expected utility of making the decision if we had that information minus the expected utility of making the decision without any extra information. The net expected value of a node in the network is: $ENET(\breve{\mathbf{x}}_i) = GEVSI(\breve{\mathbf{x}}_i) - ECSI(\breve{\mathbf{x}}_i)$ where $ECSI$ (The Expected Cost of Sample Information) is the expected cost of gathering the information about node $\breve{\mathbf{x}}_i$.

Unfortunately, the computation of ENET is computationally difficult and awkward. If, for example, the utility for EVSI is computed in terms of expected improvement in classification accuracy, while ECSI is measured in hours spent by the human annotator, then the difference $ENET(\breve{\mathbf{x}}_i) = EVSI(\breve{\mathbf{x}}_i) - ECSI(\breve{\mathbf{x}}_i)$ only makes sense when both EVSI and ECSI are converted into the same scale (units). Since it is awkward to convert the number of hours of annotation to an increase in model accuracy, and it is likewise awkward to convert accuracy to hours, we could convert both to some other measure. In other decision theory problems it is sometimes possible to convert all measures to some common measure such as money. In annotation, such a conversion would be very much a function of the particular project.

Note also that the computation of a single step of ENET alone is insufficient in annotation. If the active learner could only perform a single test then the optimal policy would be to sample the test with the highest ENET value. However, if the learner can perform multiple tests it is possible for two tests taken together to have a higher net value than any one single test alone (in other words, the greedy assumption does not always hold). To get the optimal answer, EVSI must be performed over every possible combination of tests, but this would make an already intractable computation worse [18] (see Chapter 6). Such an approach is useful for theory, but in practice the greedy assumption is often forced by computational considerations.

An intuitive heuristic can be used which can help both of the above problems. Tests can be selected greedily based upon the quotient EVSI/ECSI. This can be thought of as selecting the test with the most "Expected Bang per Buck" (EBPB) rather than the test with the highest ENET value. In economics theory, this quotient is sometimes called the "Return on Investment" or (ROI) [37]. Geometrically this can be thought of as greedily selecting line segments with the highest slope rather than the combination of line segments with the highest endpoint. Although this approach is still greedy, it can outperform greedily selecting the sample with the highest endpoint since several shorter line segments with higher slope can eventually lead to a higher final endpoint with less cost.

The relative ordering of $EVSI/ECSI$ will be the same as $\alpha EVSI/ECSI$ for all positive α. If EVSI and ECSI can be placed in the same units by a linear transformation then the "expected bang per buck" technique does not force us to define EVSI and ECSI in the same units. For our Syriac problem, cost/hour is a reasonable measure that can be used together with an estimated time to annotate a sentence in order to approximate ECSI. Further, it is reasonable to assume that the usefulness of the corpus is also some linear function of its accuracy, thus allowing us to use the ratio of these two values without trying to put them into the same units.

Unfortunately, computing even the GEVSI (and thus EBPB) of a node even in this simplified situation is far too computationally intense. Things only get worse when we begin to add the complexity imposed by the sequential nature of NLP problems. Therefore, we need an approximation to GEVSI.

Several common techniques for performing active learning include Query by Uncertainty (QBU) and Query by Committee (QBC). QBU is a technique that selects the next sample as the node with the maximum uncertainty concerning its value. QBC trains multiple learners and selects the node with maximum disagreement between the learners. It can be shown that these techniques actually approximate EVSI given some simplifying assumptions (see Section 11.3.5). Under those simplifying assumptions the GEVSI of a node will be proportional to the uncertainty in that node, or to the disagreement between learners for that node. The existence of these heuristics can make the approximation of EVSI tractable inasmuch as the simplifying assumptions are met. Traditionally, these techniques are used directly to perform the active learning selection. For our purposes, however, they will be used as a proportional approximation to GEVSI, which is an approximation to EVSI, which will allow the EBPB calculation. If ECSI is uniform then this will be equivalent to the standard active learning techniques, but when ECSI is not uniform, EBPB will allow cost aware active learning. For example, if we are paying annotators by the word, we could compute $EBPB = EVSI/ECSI \approx QBU/N$ where N is the number of words in the sentence to be annotated. We call this approximation of EBPB "NQBU" (or Normalized QBU). Given more complex approximations to the cost of annotating a sentence than the sentence length, this measure can be further refined. This is an example of the model leading to improved heuristics even when the model itself is too computationally intense to be solved directly.

9.4 Variability in Error Importance

For our Syriac corpus we have part of the corpus that will be published to print and on the internet and part that will be published only on the internet. Errors in the print portion of the corpus are more significant than errors in the internet portion. This means that we need to modify our active learner to account for this discrepancy.

A simple typical solution to this problem is to have two annotators annotate the print corpus, with a third annotating only when they fist two annotators disagree, and then to spend any remaining money on the internet portion. However, given our utility model, this approach is sub-optimal. Do we really want to spend the cost of a human annotator (especially of the second human annotator) on portions of the print corpus with an extremely high degree of certainty? The implicit utility implication of demanding the second annotator, regardless of how certain we are, is to assume that there is infinite utility in the print corpus with finite or zero utility in the internet corpus. However, the utility function that produces the behavior of always using the second annotator is inconsistent with the behavior of only using the third annotator when the first two annotators disagree. Furthermore, do we really want to be spending money on the internet corpus when there are portions of the print corpus that have low certainty even when the two human annotators agree? This situation can be detected when the computer disagrees with both human annotators. In this case would it not be worthwhile to use a third annotator in such locations despite the agreement of the first two human annotators? Thus, the typical policy assumes either that we are certain of the correct annotation after two annotators (which we are not) or that the cost of errors goes down after two annotators agree (which it does not). This common combination of behaviors is, therefore, irrational for all utility models and a better approach is needed.

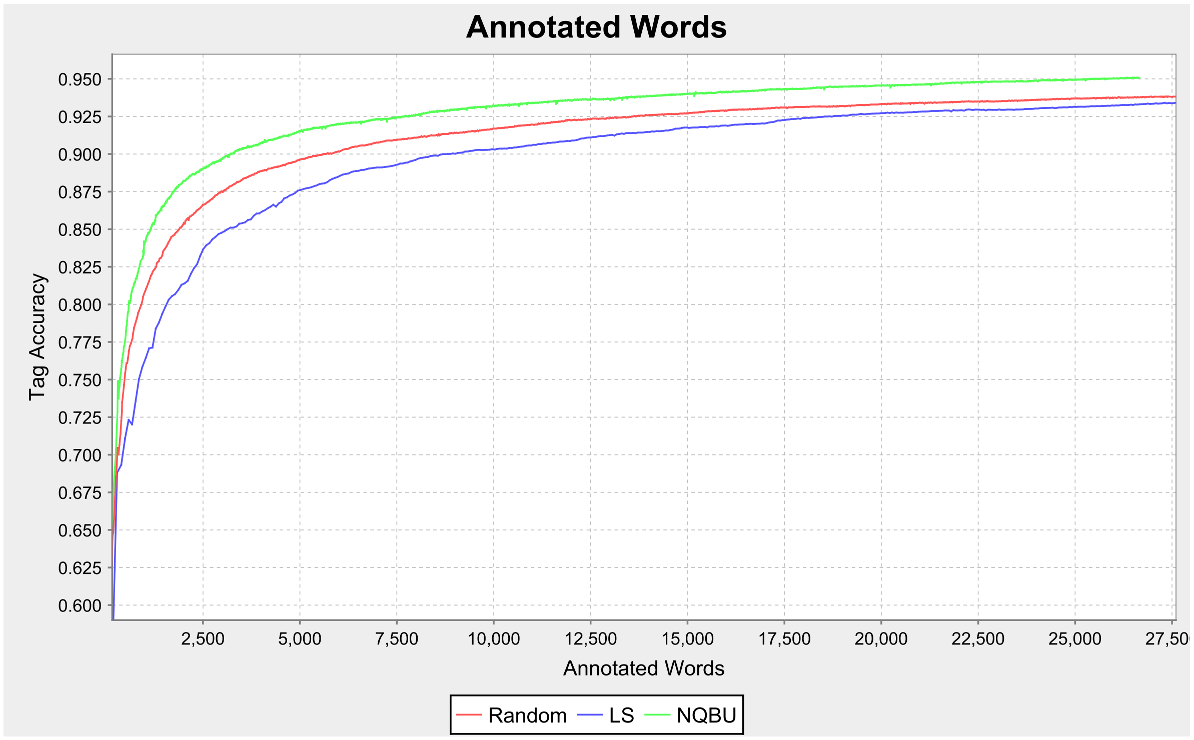
In order to correctly balance annotation costs on the print and internet portions of the corpus, it is necessary to be explicit in the cost of an error in the print and internet corpora. A reasonable approximation might be to assume that $EVSI \propto (U \quad UNC)/N$ where UNC is the uncertainty and U is inversely proportional to the cost of a mistake in that part of the corpus where the example is found. Thus, if we decide that errors in the print corpus are twice as important as errors in the internet corpus, then we could use a variation of QBU or normalized QBU where we would simply double the value of UNC if the example lies in the print corpus, and this could approximate the EVSI under this assumption.

Of course this approximation will not be exact, since sample information in one part of the corpus could aid in the other part of the corpus. For example, even if errors are twice as bad in the print corpus, the EVSI of annotating a sentence in the print corpus may be slightly less than twice what its value would be if it lay in the internet corpus, since annotating a sentence in the internet corpus can improve the computer's ability to label other sentences in the print corpus. However, this could still be a reasonable approximation. Other more accurate (but more computationally intense) approximations could be imagined; however, it is clear that any approximation will require some measure of utility. Specifically we must be explicit about the relative importance of errors in each portion of the corpus in order to make rational decisions about the allocation of annotator effort.

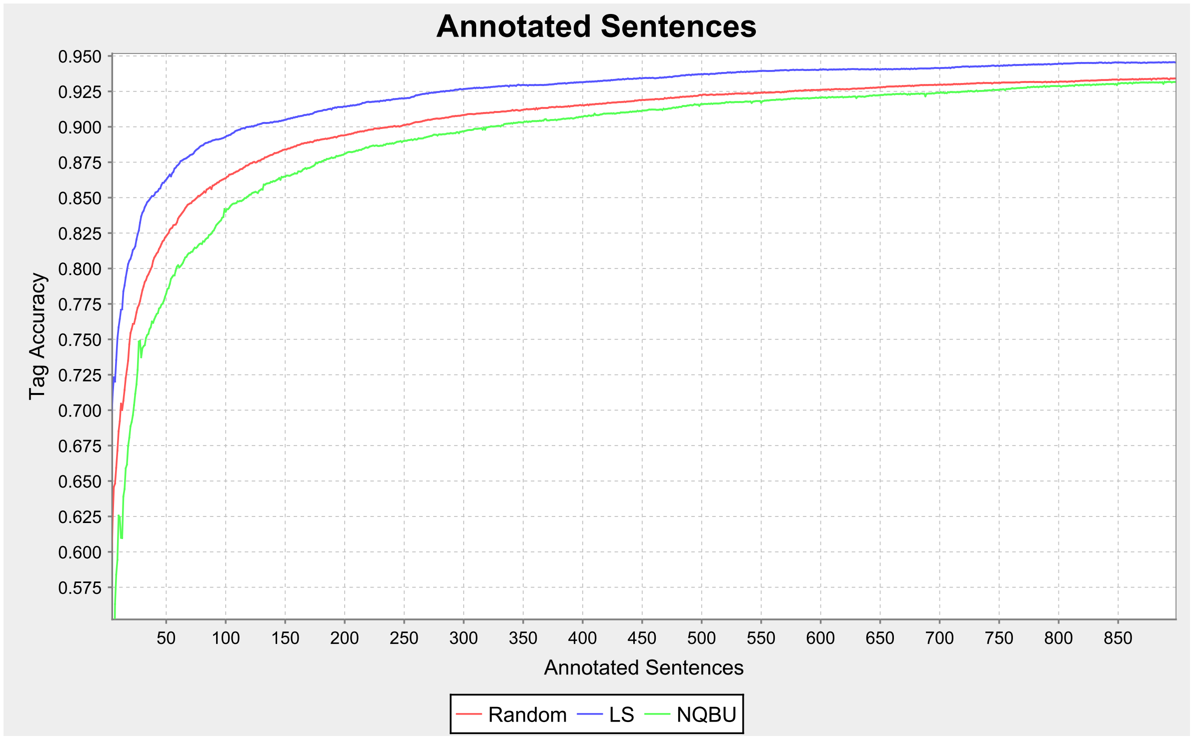
9.5 Annotation Cost

Many active learning techniques (including QBU and QBC) ignore ECSI, yet it plays an important part in the mathematics of active learning: showing up in both $ENET = EVSI - ECSI$, and "expected bang per buck" $EBPB = EVSI/ECSI$. For either technique ECSI is central to the calculation. QBU and QBC both approximate EVSI but either completely ignore ECSI, or they assume that for all samples it is either 0 or 1 depending on whether you assume that they are approximating the NET or BPB. Unfortunately, in corpus annotation it is often the case that some samples will cost more than other samples depending on the user interface involved. This can make a huge difference for active learning. For example, three reasonable ways that annotators could be paid are by the word, by the sentence, or by the hour. Different active learning techniques perform better depending on which cost metric is applied. Figure 9.5 involves an English POS labeling experiment. When annotators are paid by the sentence a rather simple active learning technique (select the longest sentence) performs well, while NQBU performs worse than random. On the other hand, when annotators were paid by the word, then longest sentence performs worse than random, while NQBU performs well. Not only did the method of payment affect the results, its influence was dramatic.

It is reasonable to assume that for sentence $\breve{\mathbf{x}}_i$, $EVSI(\breve{\mathbf{x}}_i) \overset{\propto}{\sim} LengthSent(\breve{\mathbf{x}}_i)$, because longer sentences tend to have more information in them. If $ECSI$ is constant for sentences



(a)



(b)

Figure 9.5: A comparison of three possible active learning techniques, random, longest sentence and NQBU (QBU normalized by the sentence length as suggested in Section 9.3). Notice that if the annotators are paid by the word then NQBU is the best approach (a) but if the annotators you are paid by the sentence then longest sentence is the best active learning approach (b).

of any length, then $EBPB(\breve{\mathbf{x}}_i) \overset{\propto}{\sim} LengthSent(\breve{\mathbf{x}}_i)$ and selecting the longest sentence is a reasonable policy if you pay by the sentence. On the other hand, if your annotator charges by the word, then $ECSI(\breve{\mathbf{x}}_i) \overset{\propto}{\sim} LengthSent(\breve{\mathbf{x}}_i)$. Thus EVSI will tend to be larger with larger sentences, but so will ECSI, and it is no surprise that a normalized measure of the uncertainty per word (NQBU) is now the preferred measure. Notice that it is the model which helps to explain the behavior of these approximations.

It is possible to do better when paying by the hour if it is possible to predict how long the annotators will take to annotate a given sentence. This time will then be proportional to the ECSI of paying by the hour. Then using an approximation to EVSI you can compute $EBPB \propto EVSI(\breve{\mathbf{x}}_i)/Time(\breve{\mathbf{x}}_i)$. This will appropriately penalize longer sentences because it will take your annotator longer to annotate them. Thus, without some model of how long it will take an annotator to annotate a sentence, it is impossible to correctly determine whether that sentence should be selected by active learning when paying by the hour. Yet, paying by the hour is the most likely case. This observation motivates the user study which we will present in the next section.

9.6 A User Interface and User Studies to approximate ECSI

We have seen that ECSI is an essential component of active learning. We have already shown how to approximate the time that it will take a user to annotate a sentence with the length of the sentence, however, there are many factors that determine the time it will take a user to tag a sentence beyond the simple length of the sentence. We performed a user study motivated by the above ideas to determine the expected time for annotating the part of speech of a sentence. Although our goal is to repeat this experiment on the Syriac interface, because English annotators are easier to come by, initial experiments were performed in English with the Penn Treebank tag-set [92], and a similar experiment is planned for Syriac.

The user interface made suggestions based on the machine learner's current model and the user only had to change those words that were annotated incorrectly. Using the data from this study we developed a linear model for part of speech annotation cost suitable for use as the expected annotation cost in the context of active learning algorithms. The details can be found in [92]. Here we will simply note that the final rational cost model discovered by the model was: $ECSI(\check{\mathbf{x}}_i) \approx 3.795l_i + 5.387c + 12.57$, where l is the length of the sentence and c is the number of words the user had to change. The resulting model has an appealing intuitive interpretation: the annotator reads each word and decides whether or not it needs to be corrected (3.795 seconds per word); then corrects each word needing correcting (5.387 seconds per correction); finally, there are 12.57 seconds of overhead per sentence.

The model uses only a small subset of the raw statistics we collected. There are two reasons for this: first, some of the statistics which we collected (for example, "Self Evaluation of Tagging Proficiency") were not included in the model because we explicitly wish to assume that tagging will be conducted by a mix of people with tagging skills similar to the mix of skills tested in the user study; second, some variables fail to have a statistically meaningful effect on the resultant model. We employed linear regression and the Bayesian Information Criterion (as implemented in the LEAPS package in R) to assess which variables should be included in the model.

We would expect the results to be different depending on the language, the tag set, and the user interface. The results of the user study provide an approximation to the expected cost of sample information and will be an essential element to any effective active learning technique for language annotation. Notice that the cost model implies that there is some overhead in reading each sentence. Furthermore, the amount of overhead (for English approximately 12.57 seconds) can have significant impact on the best way to present data to the annotators. It may be better to give the annotators a single most uncertain word in a sentence and ask them to only correct that one word so that they do not waste time on other words in the sentence which the computer may already have a good model for; or it may be better to let them annotate the rest of the sentence since the time overhead for reading the entire sentence has already been taken. Which approach is better will depend on the language and the interface. A user study is imperative for making such decisions.

Notice also that, for our user interface, the cost model is directly related to the accuracy of the classification machine learning model. The better the machine learning model, the fewer corrections the user has to make. For our user interface, these corrections accounted for a large proportion of the cost. This means that early improvements in the machine learning model allow us to collect more human annotated data because that human annotated data is now cheaper.

Any other technique for speeding up human annotation will clearly result in being able to afford more human annotation. Thus, any active learning project for corpus annotation should involve the development of the best user interface possible. Time spent designing and evaluating user interfaces can have significant benefits later on. The best interface will likely differ from language to language. Changes in the user interface can affect more than just the speed with which an annotator annotates data. Different user interfaces could lead the same annotator to different levels of accuracy. Multiple user interfaces should be proposed and then evaluated for both speed and accuracy. Future experimentation on user interfaces for the Syriac project is on-going.

9.7 Dealing with Human Annotation Error

Until now, we have assumed that human annotation reveals the true tag $\breve{y}$. Unfortunately, human annotators are not one hundred percent accurate. Therefore, a class/tag/annotation is never actually directly observed. Rather an annotator's opinion concerning the correct class is observed. An example graphical model that takes this into account for three different annotators is shown in Figure 9.6.

These changes to the model will have implications for active learning. We are no longer interested in selecting the most informative $\breve{y}$ node, but in selecting the most important annotation node $A_{(a,i)}$ (where a is the annotator involved and i is the instance to be

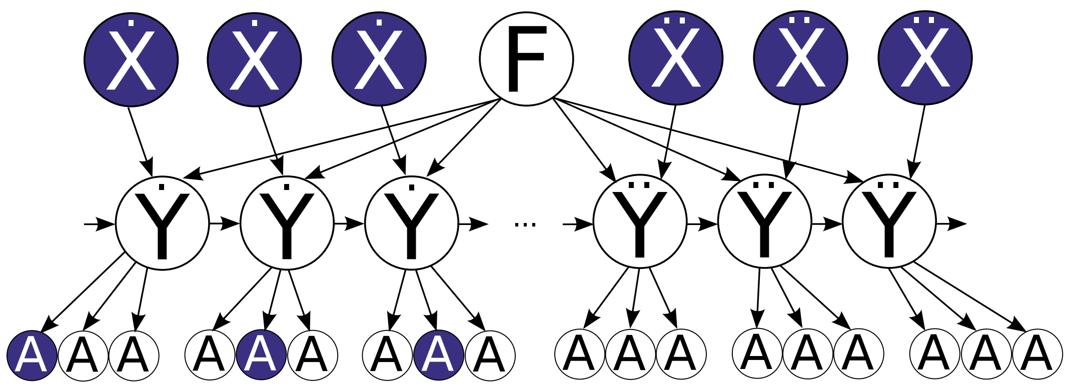


Figure 9.6: Adding multiple annotators, and annotator uncertainty. In this example there are three annotators. Notice that the annotators annotations depend on the true value of the class, but the true value of the class is never actually observed.

annotated). For example, if annotator number 1 is willing to annotate another example then the goal of active learning would be to select from the $A_{(1,i)}$ nodes the node that provides the maximum improvement in expected utility. Notice that the distinction between the training set, pool set, and the test set begins to break down. The training set used to be the set of examples that had been labeled by this point, with the pool set being the remaining set that were not yet labeled, and the test set was the total set of examples. But now, no examples are directly observed, and it is possible to annotate the same example more than once, with different results each time. It may be that the optimal $\breve{y}$ node for annotator 1 to annotate could have already been annotated by annotator 2. In that case, the pool set is now the set of annotated and un-annotated examples, and the training set is the set of annotated examples, and the test set is still the total corpus.

The optimal solution will come from directly solving the EVSI equations on the annotation nodes of the model, but this is again too computationally intense and a heuristic is needed. Luckily, if we are using uncertainty to model the EVSI of sampling a given annotation, and if we assume that our uncertainty about an annotator's annotation is proportional to our model's uncertainty about the class y, then we can fall back to QBU on the y nodes. This means that the active learning technique can remain relatively unchanged and still accurately model this situation.

This indicates that the effect of an annotator's annotation on the machine learner's belief about the class y should be directly modeled $P(A_a|y_i)$. This leads to a stochastic belief about the label rather than to a sure knowledge of the label. Typically, human annotations are used as training examples for most machine learning. This approach ignores the fact that there could be errors in the human annotations and relies on the machine learning algorithm's robustness to noise to compensate for this shortcoming. This can be problematic, especially when combined with active learning. This is because both the actual EVSI calculation and the QBU approximation are both dependent on the uncertainty of a given word after it has been annotated by a human annotator. Although the new model did not change the active learning technique, it will change the values that the active learner will use to make decisions through changes in the machine learner. The uncertainty of an annotation after a human has already annotated it is an important piece of information, especially for determining if a second annotator should be used to validate the results of the first. Unless we can compute the probability of an error after an annotator has annotated a word, then the active learner cannot appropriately make this decision. This means that machine learning techniques that incorporate probabilistically annotated training data are necessary in order to take full advantage of active learning with multiple annotators, especially if the annotations are poor.

Recent projects like Wikipedia and YouTube have illustrated the promise of user generated content on the web, an approach commonly known as "crowdsourcing" [11]. Opening the corpus creation process to user involvement could be beneficial since it could increase the number of annotators available. Unfortunately, the quality of annotations obtained in this way can vary widely. In such situations average annotator accuracy may be insufficient, and it is important to model each annotator's abilities separately. If we could spot a bad annotator, and appropriately adapt both our machine learner and active learning technique to his level of ability, then we could allow anyone to provide annotations with no fear that they would lower our overall accuracy.

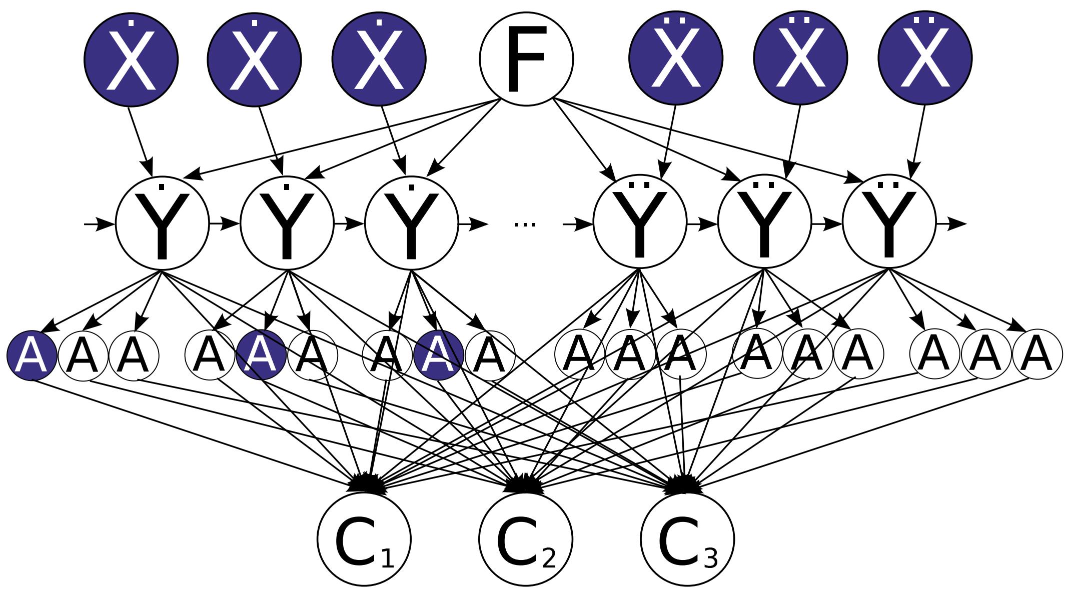


Figure 9.7: Learning the annotation abilities of multiple annotators.

The model in Figure 9.6 is only correct if we assume that all annotators are equally accurate in their annotation abilities, a simplifying assumption that may be reasonable for some applications. However, in order to model each annotator's abilities separately, we need to model $P(A_{(a,i)}|y_i, c_a)$, as shown in Figure 9.7, where c_a represents the annotation abilities of annotator a. Then, in order to learn the abilities of an annotator c_a from examples, we need to model $P(c_a|A_{(a,1)}...A_{(a,n)}, y_1...y_n,)$, and compute $P(c_a|Annotations, \mathscr{D}^o_{ata})$, by integrating out f, y, and the ability of the other annotators. This more complex model also has implications for active learning. Now there are two possible reasons why a sample location could have value. Each annotation provides information about the true value of the class (and thus about the model f). Each annotation also provides information about c_a the annotation abilities of annotator a. Again the optimal solution could be found through the EVSI equations. Interestingly enough, this optimal solution could involve giving an annotator a problem with a well known solution because it teaches us about the annotator's annotation abilities. An active learning algorithm must balance the need for information about the annotator's abilities with the need for information about the class. These desires can sometimes be mutually exclusive since querying in locations where y is known with some high degree of certainty often gives more information about the quality of the annotator, while sampling in locations where y is unknown often gives more information about y and f. The principles of decision theory and EVSI will automatically balance these issues.

If EVSI in the simple network was intractable then this is far worse. Therefore, a heuristic is again required to balance learning about f and c. Luckily, it is possible to learn about both f and c simultaneously, and they are not always mutually exclusive, which could indicate that finding the exact optimal balance is unnecessary and a rough heuristic will do. Even when we sample a location with high uncertainty we will still learn something about c. There are at least two reasons for this. First, the quality annotators will produce annotations that lower the entropy over the system's belief about the model f, while bad annotations will be less likely to reduce the entropy over $\rho(f)$. This happens because good annotations lead to a coherent pattern, while usually bad annotations often lead to chaos. Secondly, sample locations that are unknown now will become more certain after more data is collected from the other good annotators, at which time they can provide more information about c.

This means that we can often fulfill both goals at once. Several possible approximate policies could be tried. For example, a subjective prior could be chosen to estimate the initial probability of annotator error. Then we could begin collecting annotations. Assuming that our prior is correct we could build a machine learning model (compute $\rho(f)$). Then assuming that the model is correct we could recompute the annotator's accuracy c. This process could be repeated in an EM like fashion. Once sufficient data is available for each annotator, then we could begin to model the abilities of each annotator separately. It is hoped that so long as the number of good annotators is greater than the number of bad annotators, this system will eventually converge to reasonable approximations for both f and c. Infinitely more complex models could also be imagined. Some annotators could be better at certain types of tags and worse at others. Whether or not to take such factors into consideration will depend on the complexity of the model created and on the amount of available training data.

9.8 Consequences, Results, Conclusions, and Future Work

The purpose of this chapter has been to analyze the machine learning sequence annotation problem from the perspective of Bayesian utility and decision theory and to make suggestions based upon these observations that can be used to improve accuracy and decrease cost in the upcoming creation of an annotated Syriac corpus. It is hoped that many of these suggestions will be widely relevant in other annotated corpus creation projects. We will now present several suggestions based upon the observations made above concerning: how to deal with different parts of the corpus having different requirements of quality; building a user interface to minimize ECSI; performing a user study to assess/approximate ECSI; estimating the quality of an annotator; necessary modifications and enhancements of the machine learning algorithm itself; and selecting examples for active learning.

We propose the following technique for performing selection for active learning. ECSI should be first minimized using various user studies on several possible user interfaces. QBU or QBC can be used to approximate EVSI, and then ECSI can be approximated appropriately depending on the method used for paying annotators. If annotators are paid by the sentence or by the word, then ECSI can be computed directly. If they are paid by the hour, then this value can be approximated through user studies. There is some utility in giving an annotator a problem with a known solution; however, with enough data we will eventually learn the abilities of any annotator so specifically sampling for this purpose is less important. We propose that a good prior for the abilities of annotators can be selected subjectively. A few set questions with a known solution could be initially asked each annotator to refine this prior, but the number of such questions need not be large. Samples should be taken in locations with the highest EBPB. The abilities of each annotator can then be refined in an EM like fashion as detailed above.

In order to take advantage of these suggestions and observations the machine learning algorithms used will need to have several properties. Obviously simply solving the Bayesian network gives the theoretically optimal solution but will be computationally intense. Algorithms used as an approximation to this network will need to be able to report its uncertainty, preferably over the possible annotator's responses, but at least over the possible output tags y. This measure of uncertainty is essential to QBU. Next, an approximation will need to be able to deal with probabilistic training data. In other words, it will need to be able to deal with training data that comes from an annotator with a known error rate and to be able to deal with different annotators each with different error rates. Our current MEMM machine learners report their uncertainty but currently do not deal with uncertain training data. Creating a learner that can handle probabilistic training data is an important part of our future work.

This work has been primarily theoretical and motivational. Its purpose has been to motivate the research that has followed. Many of the unsubstantiated claims of this paper have been validated in subsequent publications, which we can only briefly overview here. The interested reader is referred to the actual publications for further details.

The goals of the Syriac corpus creation project were initially published in "A Computational Perspective on Syriac Corpus Development and Annotation" [41]. The initial bottleneck for Syriac has involved the transcription of the texts and Syriac dictionaries that will be involved in the morphological tagging process. While these transcription tasks have gone forward, we have evaluated various approaches in English with the goal of eventually applying the best techniques to the Syriac project once the transcription has been finished. English has the further advantage of having a large amount of previously tagged data that can be used to evaluate different approaches. In "Active Learning for Par-of-Speech Tagging: Accelerating Corpus Annotation" we applied active learning to English prose and poetry part of speech tagging tasks [93]. In this work we noted that the current state of the art POS tagging results are obtained using MEMMs, thus, it seems reasonable to want to apply POS tagging to MEMM learning. To the best of our knowledge, this work is the first to present results using MEMMs in an active learning framework. We used previously tagged data to simulate the presence of a human annotator. In this work we used the sentence as the level of granularity, and the active learning selected an entire sentence to be annotated. Since active learning requires repeatedly retraining the MaxEnt model, we proposed a simple adaptation called "fast MaxEnt" that speeds this retraining process. We explored the relative performance of QBU and QBC. Standard QBU does not directly apply to the situation with sentence level uncertainty. To compute the uncertainty of a sentence, we tried various approaches, including summing the per word entropy (QBUSumEnt). We also tried several adaptations inspired by the modeling process such as "Weighted Query by Uncertainty (WQBU)" which weights QBU by the word's corpus frequency, and "QBUV" a variation in how the uncertainty is computed that uses $1 - P(\hat{t})$, where $P(\hat{t})$ is the probability of the best tag; in essence, it selects a sample consisting of the sentences having the highest probability that the Viterbi sequence (the sequence of tags that would eventually be chosen) is wrong. In this work we answered several important questions. We showed that smaller batch sizes lead to slight improvements for low amounts of data, but that these improvements quickly disappear for larger amounts of data. We also showed that there was little difference between QBC with three committee members and with seven committee members. QBUSumEnt was slightly superior to QBUV for small amounts of data, but QBUV quickly overtook QBUSumEnt, and both were superior to QBC. However, all these approaches were comparable, and differences were small. On the other hand, WQBU's performance was surprisingly poor because it gave too much weight to very common words with low information to be gained from further sampling. We also showed that it was advantageous to start active learning as early as possible instead of having a large "seed" training set.

In "Assessing the Costs of Sampling Methods in Active Learning for Annotation" [36], we showed that the optimal active learner depends greatly on the cost model. We demonstrated this with three cost models, paying by the word, by the sentence, or by the number of words changed, and we showed that the active learning approaches behaved differently under each of these assumptions. Some techniques that performed very well under one cost model performed worse than random under a different cost model. We also proposed new normalized variants which are explicitly normalized by the expected costs. We showed that these normalized active learning variants outperformed their non-normalized counterparts.

In "Assessing the Costs of Machine-Assisted Corpus Annotation Through a User Study" [92], we developed and tested a user study for annotating PEN Treebank style English data. We evaluated user performance both annotating a word at a time or a sentence at a time, and determined that for this interface on this data $ECSI_i \approx 3.795l_i + 5.387c + 12.57$ (as we briefly discussed in Section 9.6).

In "Return on Investment for Active Learning" [37], we validated the ROI scheme for approximating the optimal NET performance in a greedy fashion. Evaluating the performance in this case is difficult because, although our the data set can be used to simulate human annotations, it can not simulate human annotation times. In this publication we showed that if annotators are paid by the hour, and if the results of our user study are used to predict the time of the human annotators, and to normalize the active learning approaches, then these normalized approaches outperform other techniques. This still leads much to be explored since these results depend entirely on the accuracy of the learned model from the user study. If actual annotation times are not accurately captured by the normalizing cost model, problems could result. Future work should involve learning the cost model on the fly, and normalizing by this learned cost model in a more realistic system with real human annotators. In any event, these results do indicate that it would be beneficial to normalize by cost or by an approximation to cost if such information is available.

These publications have validated many of the theoretical claims made in this Chapter, however, some work remains to be done to validate the theoretical results with regard to using probabilistically labeled data with differing annotator abilities, and with regard to dealing with the different importance of the print vs. internet portions of the corpus. Future work will evaluate and validate these ideas.

Further work is also necessary to extend these results to Syriac. User interfaces for Syriac have been developed and some tagged data is now available. Some initial morphological tagging experiments have been done on the Syriac data, and we are now prepared to repeat many of the above experiments on Syriac. This will allow us to compare several competing Syriac user interface designs, to develop a cost model for Syriac, and to compare it to the English cost model. We can then evaluate whether the other English active learning results also hold for Syriac.

Chapter 10

Application to Empirical Function Optimization

Abstract

In this Chapter we apply the principles from Chapter 7 to create a greedy optimal technique for EFO, and we apply that algorithm to a simple precipitate maximization problem. This will validate some of the ideas presented in the earlier chapters. We will show that this approach allows us to address several pervasive issues in optimization, including the traditionally difficult problem of selecting an algorithm that is most appropriate for a given task. By using the principles of decision theory, we are able to formulate the optimization approach that is optimal for each expressly stated function class. The resulting solution methodology can optimally navigate exploration-exploitation tradeoffs using well-motivated decision theory, while providing the process with a natural stopping criterion. Finally, the model naturally accommodates the expression of dynamic and noisy functions, setting it apart from most existing algorithms that address these issues as an afterthought.

Because this optimal algorithm is intractable, we will approximate this optimal solution with a greedy approximation and with a particle filter to approximate the inference process.

Publication: the majority of the material in this chapter has been previously published in [77]. This publication pre-dated the development of the UBDTM, and the active sampling algorithm was then known as the utile function optimizer (UFO) and was based on the graphical optimization model (GFO). These algorithms and model can be seen as the pre-curser of the unified graphical model and active sampling framework that came later as the principles in this publication were unified with un-supervised, semi-supervised, and supervised learning.

10.1 Introduction

As we have shown in Chapter 2, continuous empirical optimization can be conceptualized as an inference process: given observed function values, conclusions are drawn about the hidden location of the global optimum. These conclusions are frequently drawn in the presence of uncertainty, where critical characteristics of the function are unknown and sampling produces expensive and often sparse information Then, this information about potential extremum locations are used together with a model of end use to produce an optimal sampling strategy, followed by an optimal final exploitation decision strategy.

In this chapter, we will validate these ideas with a concrete example. We will explore a simple definition of a common "cone like" function class, and provide a simple definition of utility and explore agents that use it to make rational sampling decisions, producing sophisticated behavior when provided with intuitive and straightforward declarations of practitioner intent.

This chapter is organized as follows: In Section 10.2 we develop a graphical model of the evolutionary optimization problem, introduce optimal sampling in this model using utility theory, and we discuss how to implement this sampling technique and thus optimally select the membership of the next population using a specific utility model. In this section we also introduce the complete Utile Function Optimizer algorithm (UFO). In Section 10.3 we summarize and clarify these concepts with a simple case study, and demonstrate how the various graphical model specifications can be produced. In Section 10.4 we discuss the problem of selecting an appropriate function class and compare it to the more traditional problem of selecting an appropriate algorithm. Finally, in Section 10.5 we give conclusions and suggest future work.

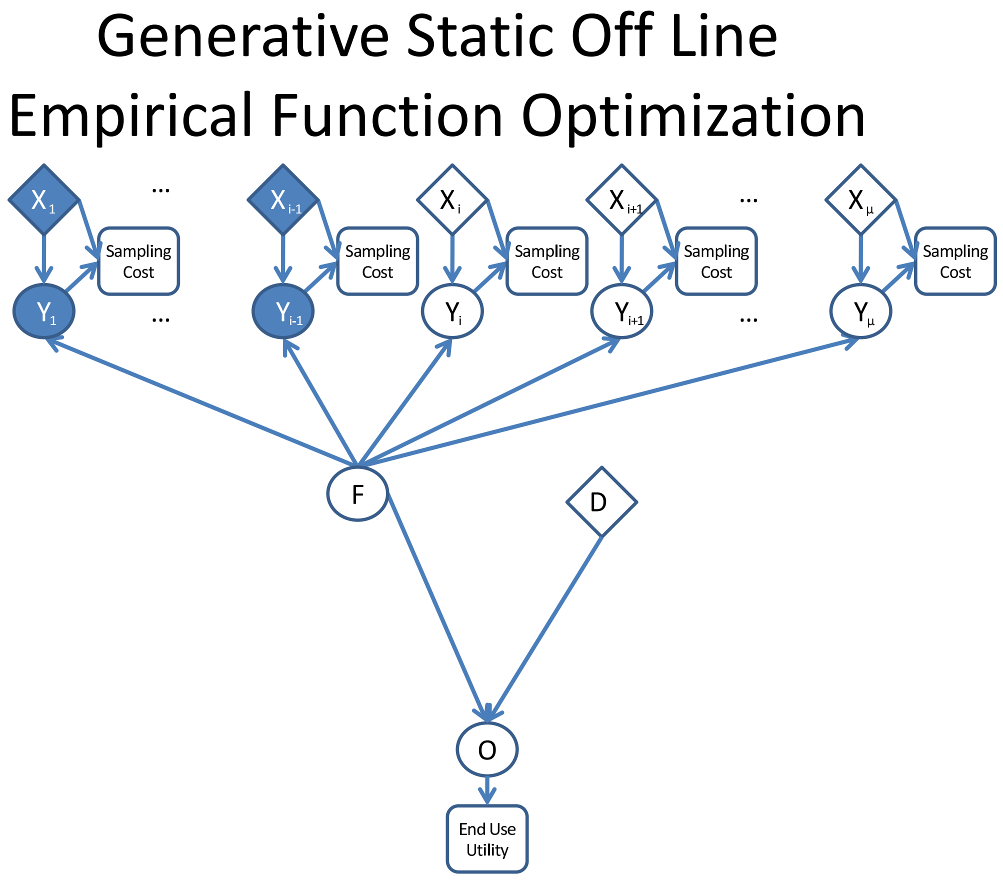


Figure 10.1: The generative static off line empirical function optimization case at the ith time step.

10.2 A Utile Function Optimizer: Approximating EVSI in the EFO-UBDTM

Recall from Chapter 7 that the equations for optimal selective sampling for off line EFO are:

$$
\begin{aligned}
E(U|\mathbf{x}_i, \mathscr{D}_{i-1}, \mu) = \sum_{y_i \in \mathbb{D}_Y} & p(y_i|\mathbf{x}_i, \mathscr{D}_{i-1}) \\
& \max_{\mathbf{x}_{i+1} \in \mathbb{D}_X \cap \mathscr{D}_i^{\bar{X}}} \sum_{y_{i+1} \in \mathbb{D}_Y} p(y_{i+1}|\mathbf{x}_{i+1}, \mathscr{D}_i) \\
& \dots \qquad\qquad (10.1) \\
& \max_{\mathbf{x}_\mu \in \mathbb{D}_X \cap \mathscr{D}_{\mu-1}^{\bar{X}}} \sum_{y_\mu \in \mathbb{D}_Y} p(y_\mu|\mathbf{x}_\mu, \mathscr{D}_{\mu-1}) \\
& \max_{d_k \in \mathbb{D}_D} \sum_{f \in \mathbb{D}_F} P(f|\mathscr{D}_\mu) \sum_{o_k \in \mathbb{D}_O} p(o_k|d_k, f)\big[U(o_k) - C(\mathscr{D}_\mu \setminus \mathscr{D}_{i-1})\big]
\end{aligned}
$$

Then, the Expected Net Value of Sample Information (NVSI) can be computed as follows:

$$E(NVSI|\mathbf{x}_i, \mathscr{D}_{i-1}, \mu) = E(U|\mathbf{x}_i, \mathscr{D}_{i-1}, \mu) - E(U|Null, \mathscr{D}_{i-1}, i-1).$$

Solving this equation is extremely computationally intense. In order to develop this theory into a practical algorithm, some simplifying assumptions must be made. We will begin by using a greedy approximation to the expected NVSI, $E(NVSI|\mathbf{x}_i, \mathscr{D}_{i-1}, \mu) \approx E(NVSI|\mathbf{x}_i, \mathscr{D}_{i-1}, i)$. This allows us to use a vastly simplified approximation to the expected utility:

$$
\begin{aligned}
E(U|\mathbf{x}_i, \mathscr{D}_{i-1}, \mu) \approx \sum_{y_i \in \mathbb{D}_Y} & p(y_i|\mathbf{x}_i, \mathscr{D}_{i-1}) \\
& \max_{d_k \in \mathbb{D}_D} \sum_{f \in \mathbb{D}_F} P(f|\mathscr{D}_i) \sum_{o_k \in \mathbb{D}_O} p(o_k|d_k, f)\big[U(o_k) - C(\mathscr{D}_\mu \setminus \mathscr{D}_{i-1})\big]. \qquad (10.2)
\end{aligned}
$$

All the necessary distributions may be approximated from the network using approximate empirical methods such as particle filters [95], the method of choice for this work. The Utile Function Optimization algorithm then, is to repeatedly sample in the location with the maximum $E(NVSI|\mathbf{x}_i, \mathscr{D}_{i-1}, i)$, until this value becomes negative (at which time the greedy approximation to EVSI is less than the expected cost of the sample information).

Application of this algorithm provides the practitioner with approximations to important information: where next to sample in order to obtain maximal information about $\mathbf{x}^*$, and when further sampling is no longer expected to be cost efficient.

The specification of function class and the specification of a reasonable representation of end use utility are best described with a simple example.

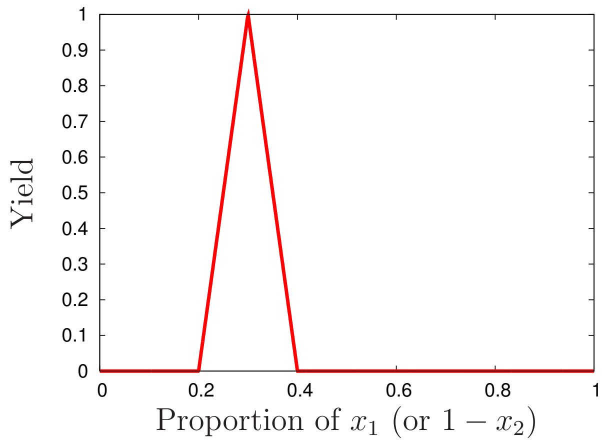
10.3 Example: Optimal Chemistry

Consider a laboratory technician tasked with finding an optimal mixture of chemicals in a solution, where optimality is achieved by maximizing the percentage yield of a precipitate. The technician receives a commission based on achieved production yield, but must pay for all experimental ingredients. This is, therefore, an off line active learning situation. Every experiment has a concrete cost (the cost in dollars of the ingredients) in the same units as the utility of its output (the dollar value of the commission).

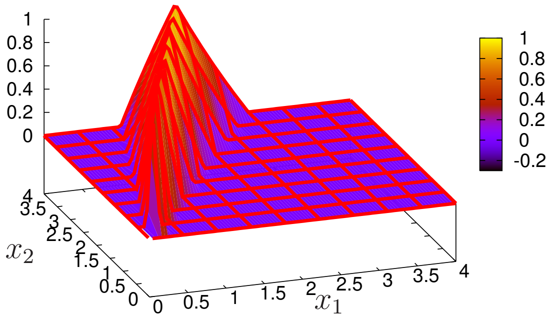
This example will be used to describe each step of the setup procedure for the parameters of the UBDTM, and the creation of a corresponding Utile Function Optimizer (UFO), that is, representation of the function class including a representation of the sample noise, representation of the goal of optimization, construction of a utility model for the output of the optimizer, and the construction of a cost model for sampling.

10.3.1 Function Class

The technician's first task is to declare the function class. Remember that the function is a mapping between $\mathbf{X}$ and a distribution over Y such that $f(\mathbf{x}) = p(y|\mathbf{x}, f)$. We will assume that $p(y|\mathbf{x}, f)$ is normally distributed such that each f is uniquely representable by some parameters $\mathbf{a}$ such that $f(x|\mathbf{a}) = p(y|\mathbf{x}, f) = N(y; y - f_m(\mathbf{x}, \mathbf{a}), f_\sigma(\mathbf{x}))$



(a) Proportions



(b) Absolute quantities

Figure 10.2: The truncated cone function

For this example, we will assume that f_m consists of *truncated cones* in the proportion space[1]:

$$f_m(\mathbf{x}, \mathbf{a}) = \max \left\{ 0, 1 - \left\| 10 \left(\frac{\mathbf{x}}{\sum_{i=1}^{D} x_i} - \mathbf{a} \right) \right\|_2 \right\} \tag{10.3}$$

where $x_i, a_i > 0$ and $\sum_{i=1}^{D} a_i = 1$. The true yield $f^\star$ is a member of this class with $\mathbf{a} = (0.3, 0.7)^\top$ as in Figure 10.2. The goal of our learning algorithm will be to optimally mix ingredients in order to determine this ratio.

An important aspect of the function class is sample noise, in this case due to measurement uncertainty. As ingredient quantities approach zero, accurate measurements become increasingly difficult to achieve. Uncertainty also increases as quantities become very large, since the error of measuring using available equipment compounds when multiple measurements must be taken to achieve larger quantities. As the measurements become very large, the experiments take on absurd characteristics: it is increasingly difficult to accurately measure vats, lakes, or oceans of constituent ingredient quantities.

[1]This model is purposely simplistic for pedagogical purposes; highly flexible models exist that, surprisingly, are only slightly more complex.

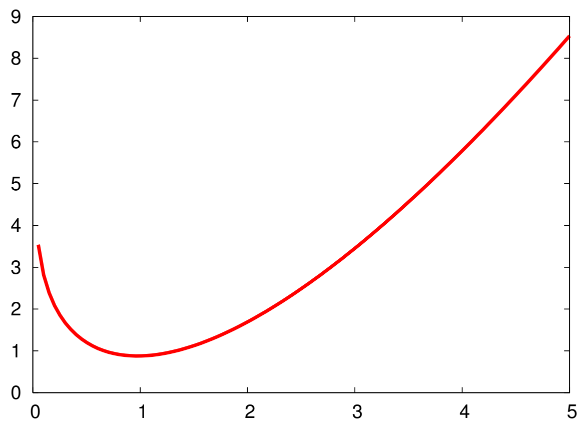


Figure 10.3: $1 + \ln \Gamma(3x/2)$: Suitable for scaling $\sigma_{\vec{y}}$

Assuming that the sample noise is additive Gaussian, this implies that the standard deviation of noise changes with the absolute ingredient quantities. The size of that standard deviation might be conveniently described using a function like the shifted and scaled log-gamma shown in Figure 10.3, defined precisely within this specification of the likelihood:

$$f_\sigma(\mathbf{x}) = \sigma_{\vec{y}} \left(1 + \Gamma \left(\frac{3}{2} \frac{1}{D} \sum_{i=1}^{D} x_i \right) \right) \quad , \tag{10.4}$$

where we assume that the scale parameter $\sigma_{\vec{y}} = 0.1$. This definition assumes that the most accurate measurement is scaled to 1 unit. It is also assumed that overall accuracy is a function of average quantities.

The distribution $\rho(f)$ is our prior, and our function class, and it is entirely represented by $p(\mathbf{a})$ as follows:

$$p(f) = \begin{cases} p(\mathbf{a}) & \text{if } f \in \{\lambda \mathbf{x}.g(\mathbf{x},\mathbf{a}) | \forall \mathbf{x} \in \mathbb{D}_X . g(\mathbf{x},\mathbf{a}) = \\ & p(y|\mathbf{x}, f) = N(y; y - f_m(\mathbf{x},\mathbf{a}), f_\sigma(\mathbf{x})) \\ 0 & \text{otherwise} \end{cases}$$

$p(a_1) = N(a_1; 0.5, 0.2)$ and $a_2 = 1 - a_1$

We have now completing the definition of this "cone-like" function class: $y - f_m$ dictates that the class contains cones, and f_σ defines how far outside of the true cone a sample value may be due to measurement error and still be considered consistent with the function.

10.3.2 The Goal of Optimization

In this example, the goal of our optimization process is to find the maximum. Recall from Section 2.2.4, that

$$\xi(\mathbf{x}^*|\mathscr{D}_{train}) = \int \xi(\mathbf{x}^*|f)\rho(f|\mathscr{D}_{train})df\,. \tag{10.5}$$

$\xi(\mathbf{x}^*|\mathscr{D}_{train})$ is the *empirical* function optimization problem, while buried within this empirical problem lies a repeated *computational* function optimization problem, $\xi(\mathbf{x}^*|f)$. The location of the maximum of any function in the truncated cone function class is $\mathbf{a}$, therefore, $\xi(\mathbf{x}^*|f)$ should place maximal probability mass on the value $\mathbf{x}^* = \mathbf{a}$. Thus, the computational function optimization problem is simply:

$$\xi(\mathbf{x}^*|f) = \delta(\mathbf{x}^* - \mathbf{a}) \tag{10.6}$$

where δ represents Dirac's delta distribution, which places all of its density at the origin. That the maximum for any function in the class can be trivially extracted is not coincidental, but the product of a careful choice of function class representation. If the representation of f does not admit trivial extraction of the optimum from its parameters, the evaluation of $\xi(\mathbf{x}^*|f)$ may hide a complex internal optimization problem. It is common to solve computational function optimization problems using empirical optimization techniques. If $\xi(\mathbf{x}^*|f)$ is not simpler than $\xi(\mathbf{x}^*|\mathscr{D}_{ata})$, and if empirical approaches are then used to solve $\xi(\mathbf{x}^*|f)$, this can lead to an impossible infinite recursion of function optimization problems. Clearly this must be avoided and function classes should be chosen such that their embedded computational function optimization problems are as easy to solve as possible.

In order to perform active learning on this model, it only remains to model end use utility and sample costs.

10.3.3 End Use Utility

For our problem, the end use decision will be to select s set of ingredients, and the outcome will then represent the yield, and so the outcome is a function of f, and d. As we mentioned in Chapter 2, when the outcome is deterministic, it is possible to skip the outcome and write the utility as a function of f and d directly as follows:

$$u(f, d) = \$100 f(d) \ . \tag{10.7}$$

Notice that even though the goal of the technician is to *maximize information* about regions of higher payoff, that goal is not directly reflected in the utility function. Instead, utility is defined as before, in terms of exploitation. In contrast to many existing techniques, it is unnecessary (and even inappropriate) to explicitly define a utility of exploration. Instead, the principles of utility and expected value of net sample information will be used to compute it.

10.3.4 Sample Cost

An obvious and trivial way to define sampling cost in this example is as the sum of the prices of the constituent ingredients that go into an experiment:

$$c = \sum_{i=1}^{D} c_i x_i \tag{10.8}$$

where D is the number of ingredients to mix, and c_i is the cost of a single unit of ingredient i. This is a straightforward definition of cost from the technician's perspective, who is required to buy his own ingredients.

When mixing a solution in order to achieve a precipitate, however, it is often possible to add ingredients *incrementally* to adjust the percentage yield, even after some amount of precipitate has been previously removed and measured. In other words, it is possible to perform a new experiment by *continuing an old one*, giving rise to a more interesting cost function:

$$c_t = \begin{cases} \sum_{i=1}^{D} c_i x_{t,i} & \text{if } \textit{restart}_t \\ \sum_{i=1}^{D} c_i \left(x_{t,i} - x_{t-1,i}\right) & \text{otherwise} \end{cases} \tag{10.9}$$

where

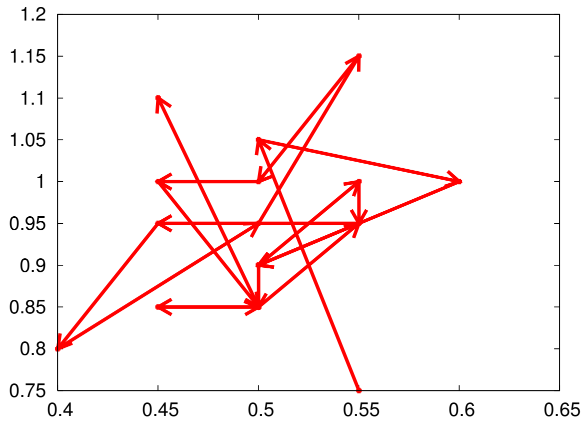
$$\textit{restart}_t = \left(\exists i x_{t,i} < x_{t-1,i}\right) \vee \left(\vec{x}_t = \vec{x}_{t-1}\right) \quad . \tag{10.10}$$

If the technician desires to reduce an ingredient or to duplicate an experiment, the cost is calculated by totaling the cost of the constituents. The only way to repeat or reduce constituent quantities is to begin again. If, however, he wishes to adjust the balance by adding a small amount of an ingredient, the cost is the price of the additional material, not of the full solution.

10.3.5 Algorithm Behavior

The results of using EVSI for experiment selection in this section assume the following:

- Sample noise has the (scale) parameter $\sigma_{\vec{y}} = 0.1$,
- All ingredients cost \$10.00 per unit, and
- The best proportion is $\mathbf{a} = (0.3, 0.7)^\top$.


(a) No cost specified

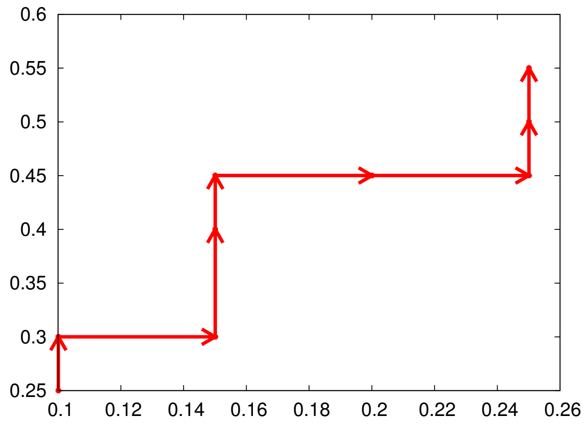

(b) Cost specified

Figure 10.4: Chemistry experiments with EVSI

A number of interesting behaviors are observed when applying the UFO (the algorithm based on choosing the sample with the highest greedy approximation to Expected Net Value of Sample Information in the AL-EFO-UBDTM) to this problem.

Figure 10.4, for example, illustrates the path taken by EVSI through the space of absolute quantities: Figure 10.5(a) shows the path of experiments when EVSI is not aware of the sampling cost, and Figure 10.5(b) illustrates the path when costs are supplied. In the first case, it continues experimenting until a pre-specified maximum number of iterations has been reached (20 for this example) and does not appear to follow any particular pattern. In the second, it always adds ingredients incrementally and stops after 9 iterations. In both cases the resulting posteriors allocated more than 99% of the probability on solutions within 1% of the true optimal.

It is significant that the technician that does not supply a declaration of costs spends $222.00 and uses all of the 20 allowed experiments, and the technician that does supply cost spends $8.00 on ingredients before determining that 9 experiments are sufficient. Even if the first technician had stopped after 9 iterations, he would have spent $117.00 on ingredients.

The fact that the algorithm stopped the sampling process early is significant because the determination of a suitable stopping criterion is a common problem in empirical optimization. Generally it is assumed that one must put a cap on the number of samples taken for practical reasons, implying that it has an associated cost. The use of greedy EVSI and the admission of an explicit cost specification makes the stopping criterion obvious: stop sampling (exploring) when cost exceeds the expected improvement in utility, then exploit the learned values.

It is also noteworthy that both sets of experiments choose absolute ingredient amounts that do not deviate too far from 1 unit of measurement. This is expected because ingredient quantities close to 0 and much higher than 1 have higher sample noise. Therefore, even though cost minimization is an important issue for this problem, taking measurements that are too small to admit much certainty does little to help the technician; the experimental process dictated by EVSI and the supplied declarations match this reasoning well.

One benefit of this approach is the ability to define various natural costs, including *opportunity cost*, defined as the expected loss due to lack of exploitation. In the absence of other cost information, opportunity cost is a natural and easily-applied definition that at least takes into consideration the fact that exploration often precludes exploitation. For example, if the technician were adjusting controls on a production mixer, opportunity cost would be appropriate; exploration that does not immediately produce better outputs costs money because it keeps a known better output from occurring. This idea is automatically applied in the context of the UBDTM.

The sophisticated behaviors of the UFO, i.e., its tendency to minimize cost, stay away from noisy regions, and stop when sampling is no longer valuable, are a direct result of the application of standard principles of decision theory to a generalized model of optimization. This connection obviates the need for carefully crafted heuristics and makes powerful decision-theoretic tools available for solving optimization problems.

10.4 Function Class Specification

Setting up an optimization problem so that the UFO can be applied requires prior knowledge of the function class of interest and the ability to codify this knowledge. Of all the required specifications in the UBDTM, this is likely to be the most difficult. Sample noise and utility are often easily obtained, but the function class definition generally contains some of the information that is desired but not directly available in optimization.

The function class declaration can take many forms, some more general and interesting than others, but the model, Bayes law, and NFL all dictate that *some* function class must be chosen by the practitioner, even when presented with a completely unknown function. This fact is hidden by common practice in optimization. Practitioners often have a toolbox of existing algorithms at their disposal, from which they select one to apply to a given problem. Such a toolbox may contain PSOs, GAs, EDAs, etc. Without knowledge of the function class, each must be applied with various parameter settings before one may be selected that is at least good enough for the given function.

The function class specification requirement imposed by the UBDTM initially appears to be even more onerous: if a practitioner has no prior knowledge about the function, how can he appropriately declare the function's class? The problem of function class specification is, in fact, the same as algorithm selection. Lacking any prior knowledge of the function, a practitioner will try various function classes in UFO to find one that works well. Once it is discovered that a particular problem performs well with a given function class, the practitioner gains specific information about the problem, namely, that it belongs to a specific function class. No such information is gained when selecting opaque algorithms which do not explicitly indicate their intended function class. Furthermore, because a Bayesian framework is used to perform optimization, the various classes at a practitioner's disposal may be compared, contrasted, and even combined using principled statistical methods [63, 65].

10.5 Conclusions and Future Work

The UFO is purely declarative, requiring the following:

- The function representation, including the nature of sampling noise (f_m, and f_σ),
- A distribution over the potential functions ($\rho(f)$),
- End use utility ($p(o|d, f)$ and $u(o)$), and
- The cost of samples ($c(\mathbf{x})$).

With the possible exception of the function class, each of these is typically available and can be specified in the UBDTM in its natural form. The class specification, while requiring a different and non-traditional approach to optimization, has many benefits and is not as difficult as it may seem. Simple yet highly flexible function class definitions exist, and several other examples of such function class specifications and examples have been published by Monson [77].

One benefit not directly explored in this work is the fact that the UBDTM admits the natural expression of uncertainty in function output due to real nondeterminism (noise), subjective uncertainty, or a combination of the two. Uncertainty is expressed in its native language: as the distribution $p(y|\mathbf{x}, f)$. The UBDTM also admits a natural expression of functions that change over time, similarly expressed as the distribution $\rho(f_t|f_{t-1}, T)$. This expressive flexibility inherent in the EFO-UBDTM is fairly unique among function optimization techniques, since existing methods are frequently designed only for the static, deterministic case; noise and dynamics are left to future research. That the addition of measurement noise in the chemistry example was so natural as to not merit specific mention is significant: it was simply folded naturally into the function and then forgotten.

The iterative nature of the AL-UBDTM places it in company with other Evolutionary algorithms based on probabilistic models. It bears strong resemblance to EDAs but uses expected utility rather than a random variable to induce exploration. It also may be related to Particle Swarm Optimization in that PSO has been shown to have surprisingly deep ties to Bayesian reasoning [79].

The EFO-UBDTM has been labeled as a model of optimization, but in reality it is a more general model of targeted search: the distribution $\xi(\mathbf{x}^*|f)$ may be defined in arbitrary ways, allowing for the expression of multiple simultaneous target locations, none of which is required to be an optimum of the function. In fact, it can direct search to *any region of interest* and is therefore simply a specification of practitioner intent. This flexibility lends credibility to the model because optimization is fundamentally a search problem with a very simple and specific definition of success.

The importance of the EFO-UBDTM and its associated UFO algorithm is best understood from the perspective of No Free Lunch: any optimizer that is more successful than random search must so be on a well-delimited function class [67, 124]. Discovery of that class is an implicit goal of most optimization research, generally approached using empirical approaches involving benchmarks [117]. The UBDTM changes the process of discovery into one of specification; because the function class is known to exist, it makes more sense to incorporate its specification directly and transparently into the algorithm design process. As a result, it becomes possible to specify more of the information that is available to a practitioner but that is often difficult to incorporate such as utilities and costs.

This work has introduced an idea that is both interesting and rich enough to admit only a brief and rather dense introduction to evolutionary optimization in terms of Bayesian inference. The idea has been explored in two steps. First, the EFO-UBDTM was introduced in Chapter 2. This model describes the optimization process as inference, using samples to obtain useful information about the location of the optimum. Then this model was expanded to incorporate active learning, and theoretical NFL results were reached in Chapter 7. Finally, the UFO algorithm was developed to put all these ideas together and determine reasonable sample locations based on a rational decision process using available information and clear and explicit declaration of practitioner intent. The model and corresponding algorithm together make a powerful way of thinking about, and working with, optimization problems, obtaining sophisticated behavior from simple distributions and rules.

Chapter 11

Conclusions and Future Work

Abstract

In this chapter, we will present a summary of the questions we have been able to address using the UBM or UBDTM, and we will present a series of questions for future work that we believe can eventually be answered by the UBM or UBDTM. We will then conclude by describing and summarizing the contributions of the dissertation.

11.1 Introduction

One of the purposes of this dissertation has been to create a framework based upon a set of statistical models that will be able to answer an extremely broad range of questions about function learning in a unified, principled manner. We believe that the UBM and the UBDTM, their special cases for each of the function learning learning problems, together with the active learning extensions, represents such a theory and framework. There is no way a single dissertation could adequately cover the broad range of implications that this framework presents. Instead, we have selected a few of the many implications, and covered those in some detail. However, it is important to understand that we believe that this approach will be useful in answering many more questions than we have been able to cover here.

This will lead to a large list of future work potential research directions which we will present in this chapter. For evidence that the UBM or UBDTM will eventually be able to answer these future work questions, in some cases we will be able to point the reader to relevant work by others using an approach that can be seen as related to the UBM or UBDTM, and for the rest we will briefly explaining why we believe that the UBM or UBDTM is the correct tool for eventually addressing these questions.

We will first review the implications and results of our approach, then we will propose future work, and then we will conclude by summarizing the contributions of this work.

11.2 Review of Implications and Results

Many of the theorems presented in this dissertation were almost trivial to prove. We do not see this as a drawback of the current work, rather, we believe that this represents added evidence that we are on the right track. In the past, issues such as NFL have required long journal-length articles to prove and to justify, however, once the UBDTM was introduced, such proofs became relatively simple and the implications behind the results became even more clear an intuitive. Once the model is correct, many things that were difficult in the past have become remarkably simple in this new framework, and this new simplicity is one of the major contributions to our approach.

11.2.1 Flexibility of the UBM-UBDTM

We have seen that the UBM is a highly flexible model. It is possible to select many different representations for $\rho(f)$. This allows the UBM to mimic the representational bias of a wide variety of classical learning algorithms. Thus the UBM is surprisingly versatile. As an example, an approach that can be reformulated as a variant of the UBM has been used by others to train the weights of an arbitrary artificial neural network using approximate inference approaches [24] [66]. In fact, the representational bias of almost any machine learning algorithm can be recast in terms of the UBM so long as it is possible to compute the likelihood of the data given the parameters of the model. A similar approach has been taken with several learning models including neural networks [24], support vector machines [9], image super resolution [111] and evolutionary computation [77]. In all of these cases this approach has lead to improved performance over a wide range of test cases, and in all of these publications, the authors used a technique that can be re-formulated into a variation of the UBDTM.

We have shown that the UBM can also be used to model the weights of a CMAC (see Chapter 8), and that in this special case, the posterior for the weights can be solved in closed form. We also discussed how the standard training rule for a CMAC can be viewed as an iterative approximate approach to the maximum likelihood value for the weights (see Section 8.2.1), and that the UBM can be used to generate the true posterior predictives for the CMAC outputs, which can also be solved in closed form (see Section 8.3).

We have also expanded the UBM to sequential learning tasks for NLP POS-tagging where we used MEMM's as an approximation to the results of this ideal network (see Section 9.2.1).

We have seen that there are some learning algorithms that can be represented by the Extended Bayesian Formalism (EBF), that can not be represented by the UBDTM (see Section 3.4). This is because the UBM and UBDTM are primarily proscriptive, in the sense that they attempt to model how learning *should* be optimally done. On the other hand, the EBF is primarily descriptive, describing how many learning algorithms actually behave, including those that are sub-optimal for every function class (see Theorem 3.4.0.4). In this sense, the UBM and UBDTM are less broad, and thus less flexible than the EBF. However, since every algorithm that can not be represented by the UBDTM is sub-optimal for all function classes (see Theorem 3.4.0.4), it seems that if such other sub-optimal algorithms function in practice at all, it must be because they approximate the correct answer in some way. Further, such approximations can often be better designed if we start with the correct answer and then simplify. This was the case with the CMAC, where the traditional learning algorithm worked because it was an iterative approximation to the maximum likelihood approximation to the true answer. It is in this domain of learning theory, and representing the actual mathematical answer to the learning problems that we are trying to solve where the UBM and UBDTM are most useful.

Even though the UBM and UBDTM only represent how learning *should* be performed, instead of modeling how learning is sometimes incorrectly done, the UBM and UBDTM are still useful when reaching theoretical conclusions. In some instances such models can actually be more useful. For example, the UBM and UBDTM can be used to reach many similar theoretical NFL results as were found with the more broad EBF (see Chapters 3 and 7), but by using our models, we were often able to make the proofs simpler and the implications of these theorems more intuitive and exact (see Chapters 5 and 4).

11.2.2 True Probabilities for the Posterior Predictives from the UBM

One major advantage of the UBM, is that it returns full distributions for posterior predictives (see Chapter 4). Also, the model helps to illuminate the need for these sufficient statistics (see Theorems 4.2.0.1-4.4.0.3). Thus, it was the UBDTM that most clearly illustrated the fact that point estimates are not helpful tools for producing decisions that maximize expected utility over general utility functions. Only when we return the full posterior can the algorithm's user make optimal decisions using this posterior for an arbitrary utility function. Since it is often the case that the eventual end use to which our algorithms will be put is not known when we are creating the algorithm, it makes sense to either use the UBM to create a distributional posterior predictive, or to use an approximation to this full posterior predictive other than point estimate approximations, such as maximum likelihood estimates which are, unfortunately, so common in the community, (see Chapter 4).

11.2.3 Optimality of the UBDTM

Another major advantage of using the UBDTM model with the rules of Bayesian inference when performing function learning, is that subsequent decisions based upon the results of inference in the UBDTM will lead to the optimal machine learning algorithm given a prior with respect to maximum expected utility (see Chapter 4). As we have said before, function learning is not usually performed for its own sake, but we learn or optimize functions to aid in decision making, and there is some decision that will be made based upon the output of the learning algorithm. The UBDTM adds this decision process to the principles of the UBM, and provides a method of selecting the decision that will maximize expected utility (see Section 2.3).

The well known Bayesian optimality proofs (see Section 1.2.2) show that if the problems you encounter are actually distributed according to your prior, if you compute the posteriors for your parameters following the rules of Bayesian statistics, and if you make expected utility decisions using your posteriors, then you are guaranteed to have an expected utility greater than or equal to decisions made using parameters chosen by any other technique [26, 25].

In theory, there is therefore a "best" function learning algorithm, which will perform as well as, or better than all other function learning algorithms (see Theorem 5.3.3.1 and Corollary 5.3.3.2). This result, and its connections to NFL have been widely misunderstood in the literature to date. This result has many important implications for function learning algorithm design. When developing a supervised learning system the algorithm designer's task is no longer to select the best algorithm (as that has already been done). Rather, their task is to select the correct prior (and therefore the correct bias) for the situation or set of situations which they expect the algorithm to be used for, and to design the best approximation possible to this value when it is too computationally intense to compute directly.

It is hoped that these results will inspire more work into algorithms that either produce or that directly approximate this optimal result, and that return full posterior distributions instead of point estimates.

11.2.4 Function Class and Bias in the UBM-UBDTM

Inductive bias has been a difficult theoretical issue in machine learning. Perhaps the two most difficult questions regarding bias have revolved around precisely defining the concept of inductive bias, and determining whether bias free learning is possible. We will deal with the first question here, and with the second in the next section.

There are two prevalent definitions of Bias that are widely used in the Machine Learning community. The first defines bias as "a preference for one hypothesis over another." This definition was used by Mitchell in his paper "The Futility of Bias Free Learning" [72], and was also used implicitly by Wolpert in his interpretation of the No-Free-Lunch [121] Theorems. Under this definition of bias, an algorithm that considers all hypotheses to be equally likely before seeing data (a uniform $\rho(f)$) would be considered to be "bias free."

The second definition of bias (and the definition which best matches with the concepts of the UBM, was also given by Mitchell [74, p. 43]. This definition of bias initially applied only to non probabilistic learners, but we showed that it can be simply extended to encompass probabilistic learning algorithms. In this definition, a bias is the "extra" information needed to deductively deduce the inductive algorithm's outputs (see Definition 2.4.2.3). These two definitions are fundamentally divergent because the uniform $\rho(f)$ which was bias free under the first definition is *not* bias free under the second. We have shown that the UBDTM makes it clear that there are many reasons to prefer this second definition of inductive bias.

We have shown that under this second definition of bias, if we assume the axioms of probability [51], then the output of the learning algorithm can be deduced with only the extra information present in the prior. Therefore, under this definition of an inductive bias, the inductive bias and the prior are the same thing. It can be difficult to formally define the inductive bias of traditional algorithms such as Decision Trees, or Backpropagation Neural Networks. However, for algorithms based upon the UBM, the inductive bias is easily quantified, and explicitly represented, interpretable, and at the same time, defines the probabilistic set of functions that we would expect the algorithm to perform well on. This set of functions that we would expect our algorithm to perform well on is sometimes known as a "function class." Traditionally, this function class has been nothing more than a set of functions over which we expect the algorithm to perform well [115], and this definition of a function class is the basis for such ideas as VC Dimensionality. However, there can be many advantages of instead using a probabilistic definition of function class (a set together with a measure of the likelihood of occurrence for each member of the set). For example, Buntine showed that it allows the computation of the average case sample complexity [13] (see Section 11.3.6). If this more complex (and we have argued, better) definition for function class has been adopted, then our approach unifies the three concepts of bias, prior, and function class.

If we are given a function class, then the necessary bias and corresponding learning algorithm are immediately provided by presenting that function class as the prior to the UBM and, conversely, if we are given an algorithm based upon the UBM then the function class over which that algorithm will perform well is explicitly defined by the the algorithm's prior in such a way that it can be easily interpreted, evaluated, and understood. This can make targeting the algorithm for the problem at hand easier (see Sections 2.4.1- 2.4.2).

11.2.5 NFL and the Futility of Bias Free Learning in the UBM-UBDTM

With increasing levels of formality Hume [43], Mitchell [72], Schaffer [98], and Wolpert [121] [122][124][67] [123] have attempted to show the futility of bias free learning. This result is related to the No-Free-Lunch Theorems which showed the lack of *a priori* distinctions between learning algorithms with a uniform $\rho(f)$ and with the specific 0,1 gain utility function off training set. These results have proved highly influential in the machine learning community.

We have presented Wolpert's EBF as a graphical model in Section 3.2 and compared the EBF with the UBDTM in Section 3.4. We have also used the UBDTM to extend the NFL results to a utility free variant which we call the Bayesian No-Free-Lunch Theorems. In this context, we were able to use the concept of Bayesian optimality and the above definition of bias to show the existence of *a priori* distinctions between learning algorithms. At first glance these principles of Bayesian optimality seem to contradict the traditional No-Free-Lunch theorem. This is because the No-Free-Lunch theorems have been interpreted as saying that all machine learning algorithms are equal for a uniform $p(f)$, and that no one algorithm can outperform another. This is not the case. Bayesian techniques are "optimal" given a specific prior, and the uniform distribution of functions is a specific prior for which the Bayesian approach will be optimal. Thus, the Bayesian approach is optimal, even in this so called *a priori* case. Yet this is the same case for which No-Free-Lunch has been falsely interpreted to show that all algorithms perform equally well. This misunderstanding happens because people often forget the restrictions of "off training set" and "for the 0-1 gain utility function" that apply to NFL. In Chapter 3 we have shown that, from a decision theoretic perspective, there is a "best" algorithm even in the situation where all functions are equally likely when "best" is defined in terms of utility and decision theory. We have also shown that this assertion is not at variance with the formal results of the No-Free-Lunch theorems despite how these results have often been misinterpreted.

In Section 5.2, we have shown that the traditional NFL proofs fail to actually demonstrate the futility of bias free learning. There are two reasons for this. First, given our formal definition of bias, the function class where all functions are equally likely is a function class, and thus a bias, so the case where NFL holds is not actually an "unbiased" case (see Definition 2.4.2.3). Second, inference is possible on this uniform function class, and we would expect the results of that inference process to be optimal (see Example 5.3.2). Corollary 3.6.0.7 dictates that the posterior predictive will be uniform for such cases, but there will be some utility functions for which this uniform posterior predictive will provide important information, useful or even essential to accurate decision making. Together, these observations lead us to conclude that there is indeed a "best" supervised learning algorithm (see Theorem 5.3.3.1), and even a best supervised learning algorithm off training set (see Corollary 5.3.3.2). Indeed, since the UBDTM can model more than just supervised learning, but all cases of function learning learning, it seems reasonable that there is a "best" function learning learning algorithm, although we will leave these proofs for future work.

Although we have shown that the NFL results do not actually show the futility of bias free learning, we have also shown that the need for a bias in machine learning (and therefore the futility of bias free learning) can still be proved, but not using the NFL results. Instead, we appeal to the mathematics of the UBM. Since the UBM represents the mathematics of function learning learning, since the UBM requires a prior for inference, and since we have unified the concepts of prior and inductive bias, the need for a prior in the UBM can be seen as a proof of the need for a bias in function learning learning (see Section 5.2).

11.2.6 Explicit Utility and NFD in the UBM-UBDTM

According to Mitchell's definition of machine learning (see Definition 1.2.0.1), some measure of quality is necessary in order to assess the performance of a function learning In Chapter 4 we have argued that the best measure of the quality of a learning algorithm is how well it performs on the problem it was designed for in terms of the end use utility function for that problem. One of the advantages of the UBDTM is that it makes all such utility assumptions explicit, and determines the optimal technique for making decisions given these explicit utility assumptions.

We have analyzed the impact of utility on function learning in greater detail in Chapter 4. We have determine exactly what parts of the function learning problems can be performed independently of any utility assumptions, and which parts will require utility assumptions. We have demonstrated that posterior predictives can be computed independent of end use utility (see Theorem 4.2.0.1). We have pointed out that reporting insufficient summaries of the posterior predictive makes some end use assumptions (see Theorem 4.3.0.2 and 4.3.2.1. As part of our analysis of what can be done independent of some end use utility assumptions, we have also analyzed what happens when we assume that all possible end use utility functions are equally likely, producing what we refer to as the No-Free-Dinner theorem (see Theorem 4.4.0.3).

The purpose behind learning is to use the outputs of the inference portion of the learner (the UBM) to make decisions (end use in the UBDTM). Traditional machine learning research has often assumed that the decision is to simply pick the most likely class, or the most likely location for the extremum. In practice, decisions are often far more complex; for example, they might involve balancing precision and recall. The balance between precision and recall in traditional algorithms usually involves some implicit utility assumptions. In traditional techniques, if a different balance between precision and recall was needed, it was often necessary to redesign the algorithm itself. In our model, the balance between precision and recall is embodied in the utility function. If a new balance between precision and recall is desired, it can be achieved by changing the utility function in the UBDTM without redesigning the fundamental inference algorithm itself (the UBM). Furthermore, it is often easier to reason about utilities directly than it is to reason about Receiver Operator

Characteristic (ROC) curves, Precision-Recall (PR) curves, or other traditional techniques for balancing precision and recall [22].

Thus, we have shown that thinking about function learning in terms of the UBM and the UBDTM makes utility explicit, explains what parts of learning depend on end use and which parts are utility independent, leads to many interesting theoretical conclusions such as NFD, and automatically deals with issues such as appropriately balancing precision and recall.

11.2.7 Active Learning in the UBM-UBDTM

Another important implication of the ideas present in the UBM and UBDTM involves their application to active learning. We have extended the principles of the UBDTM in order to model the active learning process by adding multiple time steps and nodes for sample costs (see Chapter 6). Active sampling can take several forms, and so the active learning extensions to the UBDTM do not represent a single graphical model, but a series of principles that can be used to turn each active sampling situation into an expected utility for taking a specific sample. Then, the optimal sampling approach is trivially found by taking the sample with the highest expected value. In the literature this principle is closely related to the Expected Value of Sample Information or EVSI [90]. These ideas can also be used to determine if any sample should be taken depending on whether the expected utility of taking no sample is higher than the expected utility of taking a sample (or equivalently, by sampling if the EVSI is greater than 0).

The primary difference between our technique and the so called statistically "optimal" techniques sometimes presented in supervised Active Learning papers [20] is that we have proposed a more general class of techniques and do not make the assumption that the cost is uniform or that the end use utility function is the misclassification error rate, nor does it necessarily make the greedy assumption. Our approach requires that the cost and utility functions be specified explicitly and becomes equivalent to these simpler techniques under some simplifying assumptions. To the best of our knowledge, this is the first time that the full optimal sampling approach for supervised learning has ever been presented. This is likely because such an approach can be prohibitively complex in practice.

However, by starting with the full solution before us, it is possible to come to some important theoretical conclusions. We have shown that there are several elements that all impact the active learning problem: utility from end use and sample costs, the number of times that the information will be exploited ν; the number of samples that will be taken μ; a distribution over $p(\mathbf{x})$; a prior $p(f)$; the full posterior predictive; end use; and the sample cost for gathering a set of data $C(S)$ (see Section 6.7). We also show that uniform priors for $p(f)$, the expected utility of sampling in one potential off training set sample location is the same as sampling in another. We call this the Active Learning No-Free-Lunch Theorem (see Theorem 6.7.1.1).

As we have shown in Chapter 6, the equations for optimal active learning are dependent upon the utility function. Therefore, techniques that perform active learning without specifying a utility function have implicit utility assumptions that will affect how these techniques perform. Again, by understanding what these implicit assumptions are, we can predict when these techniques will perform well. However, understanding these equations does not just help us to evaluate existing techniques, they can help create new simplifications that will perform well for different utility functions or under different simplifying assumptions.

As we have show in Chapter 10, this approach is especially useful in empirical function optimization techniques where the UBDTM leads to a technique for finding the value of information which can be used to produce optimal sample locations which is such an essential part of empirical function optimization.

11.3 Future Work

There are many implications to this approach to learning theory that we have not been able to include in this dissertation, and which we hope to investigate in the future.

11.3.1 Un-Supervised and Semi-Supervised Learning

One of the advantages of the UBM and the UBDTM, is that they represent unified models for unsupervised learning, semi-supervised learning, supervised learning, and empirical function optimization. In this dissertation, we have mathematically presented all four of these cases (see Chapter 2), but other than this brief mathematical definition, we have primarily focused our results on supervised learning and empirical function optimization. Further work is needed to give similar coverage to the other two cases, un-supervised and semi-supervised learning.

11.3.2 A Theory of Overfit, bias, and Uncertainty Using the UBM-UBDTM

The model has several implications for the theory of overfit, bias, and the ties between the concepts of overfit and uncertainty that should be more thoroughly explored. We will first discuss some of the model's potential implications for overfit and bias, followed by a discussion of overfit and uncertainty.

Overfit and Bias

In Machine Learning, traditionally the cause of overfit has been assumed to be selecting a complex function with insufficient data to justify such a choice. This interpretation would imply that the issue of overfit is intimately connected with sample complexity which we will discuss in more detail in Section 11.3.6. Note that by saying that more data is required in order to justify a more complex hypothesis, this interpretation is proposing a bias similar to that of Occam's razor, namely, a preference for simple hypothesis over complex hypothesis. However, over the space of all functions there is no reason to prefer simpler functions over complex ones. Yet, in practice, it has been found that simpler functions often generalize off training set better than more complex functions which, in an attempt to match every example, can end up matching (or "overfitting") noise in the training data. Therefore, one common explanation for why a given algorithm might overfit is that its hypothesis space was too complex for the amount of training data available [119].

One possible solution to overfit would be to restrict the size of the hypothesis space H to simpler hypotheses. Unfortunately, this means that more complex hypotheses are not ever considered, even when there is enough data to justify them. One way to balance these issues which has been proposed in the past, has been to dynamically increasing the size of H to accommodate increasingly more complex hypotheses as the sample size increases [119][38] [10][39]. Unfortunately, it can be unclear which hypothesis should be removed to simplify the space, and it can be unclear how much data should be collected before reintroducing these hypotheses. It would seem that there is no uniformly best rate at which to increase the size of H. For example, if the system is deterministic, then H should be expanded sufficiently to explain every training example, if H is noisy then H should be expanded more slowly to avoid overfitting the noise, thus the rate of expansion should be inversely related to how noisy we believe the function to be, but positively related to how complex we believe the function to be. Worse, after seeing data it would appear that it is not possible to determine how much noise we believe a given function has, without some assumptions about function complexity since it is always possible that an extremely complex function accounts for all the seemingly noisy observations. Thus, we must balance what we believe about noise with what we believe about function complexity.

The UBM could be used to solve the above problems by using its concept of a probabilistic function class instead of the traditional crisp hypothesis space. Then the prior together with Bayes law would provide a mechanism for determining how much weight to assign each hypothesis in the hypothesis space after seeing data by computing $\rho(f|\mathscr{D}_{test})$. If simple functions really are more common than complex ones, then you could place a prior over the space of functions that will cause the system to give more weight to simple functions at first, but then give more weight to more complex functions only when enough data becomes available to justify their consideration. Because these decisions are being made with statistical inference, the rate at which the system will consider more complex functions will grow at exactly the correct rate where the "correct" rate will be determined by the amount of weight given to each function in the prior. Inasmuch as your prior is accurate, the rate at which more complex functions are considered will be optimal with respect to expected utility. If the prior is set up correctly, then the Bayesian approach can embody biases such as Occam's razor, and will only consider complex functions when there is sufficient data to justify their consideration.

Thus, one way to view the Bayesian approach is as a principled mechanism for modifying the hypothesis space (function class) as data becomes available (using data to move from $\rho(f)$ to $\rho(f|\mathscr{D}_{train})$). The important observation here is to connect the concepts of prior, bias, and function class to the concept of a hypothesis space, and note that all four of these things are really the same thing. However, as we shall see in the next section, the problem of overfit is not just caused by the complexity or simplicity of the hypothesis space searched, but by the fact that the learner was forced to choose a single hypothesis from that space while ignoring its uncertainty that the hypothesis was correct.

Overfit, Uncertainty, and NFL

What should happen when there is insufficient evidence to justify a complex hypothesis? Traditional thinking would recommend that we should instead select a simpler hypothesis. However, unless we are sure that a simple hypothesis holds, the UBM indicates that this lack of evidence would instead increase our uncertainty in the posterior $\rho(f|\mathscr{D}_{train})$, and, therefore, in the posterior predictive $p(\ddot{y}|\ddot{\mathbf{x}}, \mathscr{D}_{train})$. Thus, the real problem with overfit is not really in selecting a complex hypothesis with insufficient evidence, rather it is in selecting *a single* hypothesis with insufficient evidence. By ignoring the learner's uncertainty concerning the hypothesis, the learner is forced to make a decision prematurely. This situation is illustrated in figure 11.1. The dividing line represents a simple hypothesis that maximized the likelihood of the data. If this hypothesis is accepted, the unknown location ? must be classified as

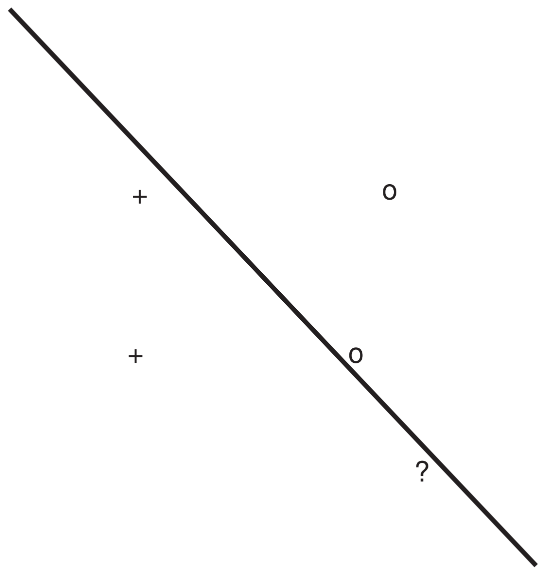


Figure 11.1: A single hypothesis that maximizes the likelihood of the data.

a +. However, there are many other hypothesis that also maximize the likelihood of the data but which classify the unknown location ? differently. If we are forced to select a single hypothesis we can overfit. Instead, we should recognize our uncertainty about the classification given the amount of data currently seen.

We have already discussed the merits of having a function learning algorithm return full posterior predictives rather than point estimates for their posterior predictives. Traditional techniques search through the hypothesis space H in an attempt to find the *best* hypothesis. A more theoretically correct (though often a more computationally intense) approach is to sum the influence of every possible hypothesis in the hypothesis space weighted by the posterior probability of that hypothesis [64]. In the UBM this is accomplished as follows:

$$p(\ddot{y}|\ddot{\mathbf{x}}, \mathscr{D}_{train}) = \int p(\ddot{y}|\ddot{\mathbf{x}}, f)\rho(f|\mathscr{D}_{train})df.$$

An algorithm that behaves in this way will trade overfit for uncertainty. Where it is uncertain of the correct answer, rather than guessing either a simple or a complex hypothesis, it will integrate over all possible hypotheses, and create a posterior predictive that truly expresses the learner's uncertainty. Although overfit is usually discussed in terms of supervised learning, from the above discussion we can see that this same idea of integrating over functions is involved in all four of the domains of function learning in the UBM.

The more possible functions that are uniformly integrated over, the more uncertainty is added to the probability of $\ddot{y}$. This concept of adding uncertainty is connected to the Bayesian interpretation of No-Free-Lunch since we are now in a position to ask questions such as, "How uncertain do our outputs off training set become as we average over all possible functions and what happens as we weight these functions uniformly?" As we showed in Chapter 3, if we uniformly average over every possible function we become completely uncertain off training set (the posterior predictive is uniform off training set, see Theorem 3.6.0.7). This view of overfit, bias, and uncertainty represents another way of looking at the Bayesian interpretation of No-Free-Lunch.

Inasmuch as overfit is caused by choosing a single hypothesis, equation (2.9) should allow us to reduce overfit, but at the cost of creating additional uncertainty in the output class. Further experimental evidence is necessary to validate this interpretation of overfit and to use these theoretical ideas to create practical and useful techniques for avoiding overfit.

Interestingly, this approach of considering multiple hypothesis functions is related to common ensemble techniques as we will discuss in the next section.

11.3.3 Ensembles in the UBM-UBDTM

The UBM seems to suggest two reasons that ensembles improve performance. First as a technique for accounting for uncertainty about the model, and second, as a way of constructing a more expressive hypothesis space.

A UBM based interpretation of ensembles would see each learner in the ensemble as proposing a different approximation for f. Diversification is obtained either by varying the samples via the bootstrap (as in bagging [12]) or by using several different learning algorithms with the same set of samples. The ensembles then vote for the output class.

Traditional techniques stop here, however, the proportions of votes for each class could be interpreted as an estimate of the posterior probability distribution. If this approach is taken, such techniques can be thought of as approximating the integral in Equation 2.9 for a few sample hypotheses. Such an approach would trade overfit for uncertainty by accounting for uncertainty in the model.

As an example, imagine an ensemble of perceptrons were each perceptron is trained using maximum likelihood. Using a single perceptron would overfit (see 11.1). However, by integrating over the possible perceptrons according to Equation 2.9, we can substitute overfit for uncertainty. If this is the purpose of the ensemble, then using the Bayesian approach is guaranteed to be the best way of combining these perceptrons, and this is the approach taken by Bayesian Model Averaging (BMA) based ensembles. However, often ensembles are created to do more than account for uncertainty about the correct model, they are also often used as a technique for constructing more expressive hypothesis spaces.

Ensembles can be used to construct more complex hypothesis spaces than those of the individual members of the ensemble. The perceptron ensemble described above can be a good example. A single perceptron can only learn linearly separable functions, but an ensemble of perceptrons can learn non-linearly separable functions. The most common "Bayesian" approach to ensemble building is known as Bayesian Model Averaging (or BMA). It computes the posterior predictive by first computing the posterior for each element in the ensemble $\rho(f|\mathscr{D}_{train})$, and then computing:

$$p(\ddot{y}|\ddot{\mathbf{x}}, \mathscr{D}_{train}) = \int p(\ddot{y}|\ddot{\mathbf{x}}, f)\rho(f|\mathscr{D}_{train})df. \tag{11.1}$$

This approach looks Bayesian, but it is actually incorrect. There is no reason to believe that the so called "Bayesian" technique for combining the individual perceptrons would out perform other ensemble techniques in this case. Peter Domingos showed that Bayesian Model Averaging of Classifiers can actually overfit worse than more *ad hoc* techniques because they tend to focus on a single model to the exclusion of the others [28]. In his abstract, Domingos claimed that the Bayesian Model Averaging approach was the optimal approach to this problem, and so he expressed surprise at this result. This indicates a significant misunderstanding about the Bayesian approach that is prevalent in the machine learning community. Bayesian Model Averaging assumes that *one* of its components is correct, and it is simply trying to integrate out uncertainty about *which* of the components is correct. It is, indeed, the optimal approach to determining which component is correct, and to dealing with uncertainty about this question. However, BMA does not consider the possibility that the correct hypothesis is actually some combination of the Ensemble components. Thus, when the goal of the ensemble is to combine the elements of the ensemble to create a richer hypothesis space, then this approach *should* fail, and certainly is not the optimal solution given this new goal of model enrichment. This is not a mistake in Bayes law or in Equation 11.1, remember there is an equals sign in these equations, rather it is a misapplication of the Bayesian approach. The Bayesian approach is the optimal way of learning parameters and combining uncertainty from competing hypotheses, it was never intended as a way to combine models to create more expressive models and simple Bayesian Model Averaging is therefore not the optimal solution to this problem. Thus, Bayesian Model Averaging is not Bayesian Model Combination [71], which is what is typically desired in an ensemble.

Future work would involve building Bayesian ensembles in a more principled manner. The correct way to build an ensemble of perceptrons with the purpose of creating a more expressive hypothesis space would be to create a set of parameters that explain the positions of each perceptron in the ensemble (the weights), and a set of parameters that explain how the perceptrons are combined to create the more expressive hypothesis space, and then to use Bayes law to learn this new more expressive set of parameters. In this case, the principle of Bayesian Optimality should guarantee that this Bayesian approach would outperform all other ensemble approaches for learning these parameters. Our BCMAC can be seen as an example of this and as a proof of concept. The BCMAC can be viewed as an ensemble, with the elements consisting of the layers. Thus, our BCMAC uses the Bayesian approach to combine these models effectively. If we had used BMA on the layers of the CMAC, then we would expect worse results. Future work should validate this supposition.

11.3.4 Meta Learning and Transfer Learning in the UBM-UBDTM

One of the goals of meta learning is to automatically determine the best algorithm in a given situation. Thus, the field of meta learning is intimately concerned with the ideas of bias, meta bias, and transfer learning. Such concerns can be conveniently addressed in a Bayesian framework. The Bayesian approach is uniquely suited to represent biases and meta biases through its priors and hierarchical structures [4].

We have shown that there is a best algorithm for producing $p(y|\mathbf{x}, \mathscr{D}_{train})$ for a specific prior $\rho(f)$. However, if the prior $\rho(f)$ is no more expressive than the uniform prior, then the posterior will also be uniform off training set. Although this posterior can provide some useful information for decision making, a more informative prior would be preferred.

How should a more informed prior $\rho(f)$ be chosen and what justification could be used for a distribution other than uniform? It is often assumed that the universe provides problems that we might want to solve according to some distribution other than uniform, $\rho(m)$ (sometimes known as the set of "interesting" functions). If we could characterize this distribution of functions then we could create much more informed priors than the uniform prior.

If meta learning is recast in terms of a Bayesian graphical model, then the task of meta learning is to attempt to learn the distribution over interesting functions, $p(m)$. The available data for learning this function would be a set of learned functions encountered in real life situations $p(f)_i$. Thus we would want to compute $p(m|f_1...f_n)$. It is hoped that this posterior would make an effective prior for learning a new function $p(f)_{n+1} = p(m|f_1...f_n)$. Thus, it may be possible to express both meta learning and transfer learning as a hierarchical version of UBDTM.

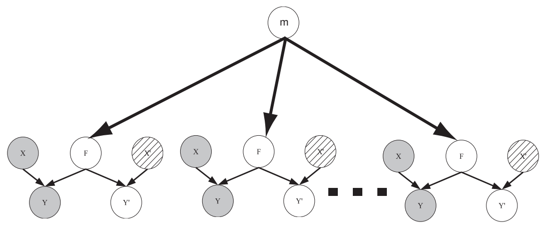


Figure 11.2: Hierarchical Bayesian model for meta learning.

This more complex model requires the specification of a prior over the meta distribution $p(m)$ and the specification of the likelihood $p(f|m)$. The distribution $p(m|f_1...f_n)$ would be determined using Bayes law. Now the probability of class $\ddot{y}_{n+1}$ for an off training set feature $\ddot{\mathbf{x}}_{n+1}$ in a new function f_{n+1} is given by:

$$\begin{aligned} p(\ddot{y}_{n+1}|\ddot{\mathbf{x}}_{n+1}, \mathscr{D}_{test\ n+1}, f_1...f_n) = \quad & \int\int p(\ddot{y}_{n+1}|\ddot{\mathbf{x}}_{n+1}, f_{n+1}) \\ & p(f_{n+1}|\mathscr{D}_{test\ n+1}, m) \\ & p(m|f_1...f_n) df_{n+1} dm\,. \end{aligned}$$

Of course this does not completely absolve us from the need for an informed prior since, if we begin with a uniform prior over $p(m)$, then we will end up with a $p(f)_{n+1}$ which is uniform for all locations not in the training sets of the other tasks. Future work will involve showing that this will, in turn, will lead to a uniform distribution over off training set function outputs $\ddot{y}_{n+1}$. This would be a meta learning NFL theorem, and would mean that our meta learner needs a bias just as much as a standard learner did. Again this would not be surprising since it is related to Wolpert discussion about unknown distributions other than uniform in the EBF [122, see footnote 4].

This approach to meta learning using a hierarchical version of the UBM is similar to that of Baxter [5], and could help to explain why existing meta learning techniques are successful, and could inspire new meta learning and transfer learning techniques. Thus, future work would involve experimental verification of the theoretical results presented here, as well as a formalization of a meta-learning NFL theorem, and a more thorough examination of the theoretical implications of the UBM and UBDTM to metal learning.

11.3.5 Additional Work on Active Learning in the UBM-UBDTM

Needless to say, computing EVSI can be computationally intensive, but it is the actual expected utility of acquiring a specific sample. Therefore, other techniques that perform well must do so because they approximate this sampling approach. In many cases, simplifying assumptions can be made. Even when simplifying assumptions are made, it can be helpful to know what the optimal solution is. Therefore, it is important to understand EVSI, even when EVSI is not the implementation of choice. By knowing the actual value that we want to approximate, it may be possible to explain why many of the previous approximate techniques work, and to predict when they will fail. For example, Monson has shown that some evolutionary approaches (such as particle swarms) actually function because they are approximating this approach [77].

We hope to show in forthcoming publications that QBU works because $EVSI_{x_T} \approx UNC_{x_T}$, where $UNC_{\mathbf{x}_T}$ is the uncertainty at location $\mathbf{x}_T$, under several simplifying assumptions. Some of these assumptions include: the misclassification error rate assumption, the assumption that $p(\mathbf{x})$ is uniform, the assumption that each sample removes uncertainty in future locations in proportion to the amount of uncertainty at that input location, and several others. When these assumptions hold, we would expect QBU to perform well, when they do not hold we would expect QBU to perform poorly. We hope to also take a similar approach to QBC, showing that it too functions because it approximates the sampling approach that maximizes the expected utility under several simplifying assumptions, and to show experimentally that when these assumptions hold, QBC does well, and when they do not hold, QBC does poorly.

11.3.6 Sample Complexity and Learnability in the UBM-UBDTM

Traditionally, the concept of sample complexity has been related to the concepts of hypothesis space and function class (see Section 11.2.4). We believe that it should also be related to active learning (see Section 11.2.7) and to the concept of "learnability," and should be more carefully connected to the concepts of utility and sample costs. The UBDTM is a mechanism that makes this possible.

Bounds on sample complexity that have been developed so far all depend on some concept of a function class. For example, VC dimensionality assumes that all functions in a set are equally likely [115], and this is one reason why they produce worst case bounds on sample complexity; PAC learnability makes a similar assumption. On the other hand, Buntine used a probabilistic definition of function class similar to our own, and showed that it is then possible to compute average case sample complexity [13] using this definition of a function class, and assuming a simple random sampling strategy. He did not explore how this sample complexity changes under the influence of active learning.

Bounds on the sample complexity should depend upon more than just the function class, they should also depend on the sampling approach. We have already shown that by sampling intelligently it is possible to reduce the sample complexity of a given function. The very point of active sampling is to reduce the sample complexity of learning a function. Therefore, it seems that the optimal sampling approach should provide the lower bound to the expected sample complexity. Optimal sampling does not guarantee a specific sample complexity, but it does guarantee that the *expected* sample complexity will be lower than if one samples according to some other strategy with respect to expected utility. Thus, it seems that sample complexity should be more tightly bound to the concepts of utility, and that sample complexity *should* be more intimately connected to the equations of EVSI than they have currently been. We believe that it is this connection to EVSI that will also allow the connection to end use and sample cost to sample complexity.

Sample complexity is usually defined as the number of samples needed to learn a function to within some epsilon of accuracy [114] [115]. Although it is possible to define sample complexity in this way, this restricts current sample complexity approaches to a single end use (misclassification error rate) and sample cost (uniform) scenario. More flexible approaches are needed. We propose that sample complexity should be defined in terms of end use and sample cost as follows:

Definition 11.3.6.1. "what are the expected cost of the samples needed to drive the expected utility to either convergence or to drive the expected utility above some satisfycing threshold if samples are taken according to some strategy?" where convergence means less than some ϵ.

If the strategy is optimal, then this definition would involve use of the EVSI equations to compute the sample complexity.

There are several observations that lead to the above definition. We will give examples to validate each of these observations.

Observation 11.3.6.1: *Sample complexity should be defined in terms of sample strategy.*

Example 11.3.1: The expected sample complexity of learning to classify many different types of toxic mushrooms will be lower if one samples according to the dictates of EVSI, then if one repeatedly samples the same mushroom over and over, while ignoring other mushroom types. This could produce an infinite sample complexity, because no matter how many mushrooms are sampled, the learner may never learn to classify other mushroom types.

Observation 11.3.6.2: *Sample complexity should be defined in terms of end use.*

Example 11.3.2: Mushroom toxicity can be much more dangerous for a pregnant woman than for someone who is not pregnant. The sample complexity of learning the mushroom function should be higher for end use involving pregnant women than for learning the same function in less critical cases.

Sample complexity is not just a product of the function to be learned but of the use to which the function will be put. When end use changes, the sample complexity should change to match. If the end use requires high assurance then the sample complexity should be high, but if the end use requires less assurance, then the sample complexity of the same function should go down [15]. Traditional sample complexity approaches all assume a specific end use (usually 0,1 loss).

Researchers have attempted to define the sample complexity independent of the utility. At first this seems reasonable. For example, if you do not know how someone is going to use the mushroom classification function, you do not know what utility function they will be using, you still might want to know the sample complexity of learning the mushroom toxicity function itself. However, most attempts to define the sample complexity of a given function independently of its intended use have ended up employing an implicit misclassification error rate utility function.

Observation 11.3.6.3: *Sample complexity should be defined in terms of sample cost.*

Example 11.3.3: Assume that mushrooms of a certain color are easily discernable as poisonous. Training examples of this type of mushroom would then be readily available and cheep to obtain. However, other examples might require extensive chemical testing to determine if they are toxic. In this case certain training examples would be cheep, while others would be expensive. The number of samples required to learn the function is thus a non-scenical measure of sample complexity. The total sample cost of acquiring sufficient samples of each kind makes more sense.

The number of samples needed is only an accurate measure when all samples have uniform cost. If some samples cost more than others, then analyzing the cost required to gather the right samples becomes important. Selecting the "best" samples balancing end use and sample cost is the job of active learning.

Given the intractability of the EVSI equations, it seems likely that these new "Bayesian sample complexity" equations will be even more complex. However, it is hoped that simplifying assumptions or approximate techniques will be able to create reasonable approximations to the sample complexity equations, and it is hoped that by understanding exactly what it is we are trying to approximate, important observations can be made about existing techniques, and improved new techniques can be developed.

Future work will involve a sort of NFL or NFD result for sample complexity showing that all sample complexity approaches must involve some end and sample cost assumptions.

11.3.7 Compression and Information Theory

Because of the intimate connections between sample complexity and the related fields of compression and information theory, it also seems reasonable that the UBDTM will be related to these fields as well. Future work will attempt to formalize and quantify this connection.

11.4 Conclusions

In summary, we have used a Bayesian Decision Theoretical approach to create a body of unified theory that we have shown can be used to address questions such as:

1. What is the formal nature of the relationship between empirical function optimization, unsupervised learning, semi-supervised learning, and supervised learning?
2. How can a learning algorithm's inductive bias be quantified, expressed, compared, or made explicit (the theory of inductive bias)?
3. What is the relationship between hypothesis space, function class, inductive bias, and prior?
4. Which algorithms perform better over the space of all possible functions that we might want to learn (No-Free-Lunch)?
5. Is there a "best" supervised learning algorithm? (No-Free-Lunch)?
6. Is a bias necessary for learning (No-Free-Lunch and the theory of bias)?

7. How should utility and decision theory impact the theory and use of machine learning (utility, decision theory, and no-free-dinner)?
8. How can the learner determine the information which would be most useful for its learning and actively request the most useful information (active learning)?

and which we hope will be able to eventually address questions such as:

1. What problems can be learned from data (learnability)?
2. How much data will be required to learn a given problem (sample complexity)?
3. What is the underlying cause of overfit, and how does this relate to an algorithms inductive bias (the theory of overfit and inductive bias)?
4. Why do ensembles work, what is their theoretical foundations and motivations (ensembles and the theory of inductive bias)?
5. If an algorithm performs well on one function or function class, how can we use that information to predict its performance on another function or function class (meta learning)?
6. How can information learned from one problem or function be leveraged to improve the learning of another problem or function (meta learning and transfer learning, learning to learn)?
7. What is the connection between the UBDTM and the related fields of information theory and compression?

We have taken a few of the above questions, and attempted to demonstrate in detail how this approach answers these questions, and we have presented in future work an outline of how the same approach should be able to answer the other questions in this list.

We believe that the Bayesian Decision Theoretical approach actually provides the mathematical, theoretical definition of the above questions. Unfortunately, Bayesian solutions to these questions can be computationally intense, however, an increased understanding of the theoretically optimal solution can often lead to better heuristics, and an increase in understanding of why such heuristics work.

Chapter 12

Appendix

12.1 Properties of the Product of Dirichlet Distribution

The Product of Dirichlet distribution (ProdDirichlet) is the product of a set of independent Dirichlet distributions, and is defined as follows:

$$\theta = \begin{bmatrix} \theta_{1,1} & \theta_{1,2} & \cdots & \theta_{1,n} \\ \theta_{2,1} & \theta_{2,2} & \cdots & \theta_{2,n} \\ \vdots & \vdots & & \vdots \\ \theta_{m,1} & \theta_{m,2} & \cdots & \theta_{m,n} \end{bmatrix}$$

$$\theta_i = \begin{bmatrix} \theta_{i,1} & \theta_{i,2} & \cdots & \theta_{i,n} \end{bmatrix}$$

Where θ_i is constrained to the unit simplex. In other words, $\sum_j \theta_{i,j} = 1$ for all i.

$$\rho(\theta_i) = \rho(\theta_{i,1}...\theta_{i,n}; \alpha_{i,1}...\alpha_{i,n}) = \frac{\Gamma(\sum_{j=1}^{n} \alpha_{i,j})}{\prod_{j=1}^{n} \Gamma(\alpha_{i,j})} \prod_{j=1}^{n} \theta_{i,j}^{\alpha_{i,j}-1}$$

$$\rho(\theta) = \prod_{i=1}^{m} \rho(\theta_i)$$

$$\rho(\theta) = \prod_{i=1}^{m} \frac{\Gamma(\sum_{j=1}^{n} \alpha_{i,j})}{\prod_{j=1}^{n} \Gamma(\alpha_{i,j})} \prod_{j=1}^{n} \theta_{i,j}^{\alpha_{i,j}-1}$$

Property 1:

When $\alpha_{i,j} = 1$ for all i and j, then $\rho(\theta)$ is a constant for all θ, meaning that $\forall \theta, \theta' \; \rho(\theta) = \rho(\theta')$:

$$\rho(\theta) = \rho(\theta') = \prod_{i=1}^{m} \frac{\Gamma(\sum_{j=1}^{n} \alpha_{i,j})}{\prod_{j=1}^{n} \Gamma(\alpha_{i,j})} = (\Gamma(n))^m$$

Property 2:

$$\rho(\theta|S) \propto \phi(S|\theta)\rho(\theta)$$

$$\phi(S|\theta) = \prod_{i=1}^{m} \frac{|S_{x=i}|!}{\prod_{j=1}^{n} c_{i,j}!} \prod_{i=1}^{m}\prod_{j=1}^{n} \theta_{i,j}^{c_{i,j}}$$

$$\rho(\theta|S) \propto \prod_{i=1}^{m}\prod_{j=1}^{n} \theta_{i,j}^{c_{i,j}} \prod_{i=1}^{m}\prod_{j=1}^{n} \theta_{i,j}^{\alpha_{i,j}-1}$$

$$\rho(\theta|S) \propto \prod_{i=1}^{m}\prod_{j=1}^{n} \theta_{i,j}^{c_{i,j}+\alpha_{i,j}-1}$$

Therefore $\rho(\theta|S) \sim ProdDirichlet(c_{1,1} + \alpha_{1,1}...c_{m,n} + \alpha_{m,n})$.

$$\rho(\theta|S) \propto \prod_{i=1}^{m}\prod_{j=1}^{n} \theta_{i,j}^{c_{i,j}+\alpha_{i,j}-1}$$

Property 3:

The expected value is computed as follows:

$$E[\theta] = \frac{\alpha_{i,j}}{\sum_{y \in \mathbb{D}_Y} \alpha_{i,y}}$$

References

[1] J. S. Albus. A new approach to manipulator control: The cerebellar model articulation controller (CMAC). *Journal of Dynamic Systems, Measurement, and Control*, 97(3):220–227, 1975.

[2] Brigham Anderson and Andrew Moore. AL for hidden markov models: objective functions and algorithms. *Proceedings of the 22nd International Conference on Machine Learning*, pages 9–16, 2005.

[3] Les E. Atlas, David A. Cohn, and Richard E. Ladner. Training connectionist networks with queries and selective sampling. In *Neural Information Processing Systems (NIPS)*, 1990.

[4] Jonathan Baxter. A Bayesian/information theoretic model of bias learning. *Proceedings of the Ninth Annual Conference on Computational Learning Theory*, pages 77 – 88, 1996.

[5] Jonathan Baxter. Theoretical models of learning to learn. In Sebastian Thrun and Lorien Pratt, editors, *Learning to Learn*, chapter 4, pages 71–94. Morgan Kaufmann, San Francisco, California, 1998.

[6] Reverand Thomas Bayes. Essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London*, 1764.

[7] James O. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer-Verlag New York, Inc., New York and Berlin and Heidelberg, 1980.

[8] James O. Berger and Donald A. Berry. Statistical analysis and the illusion of objectivity. *American Scientist*, 76, 1988.

[9] Christopher M. Bishop and Michael E. Tippling. Bayesian regression and classification. *Advances in Learning Theory: Methods, Models and Applications*, 190:267–285, 2003.

[10] A Blumer, A Ehrenfeucht, D Haussler, and MK Warmuth. Learnability and the vapnik-chervonenkis dimension. *Journal of the Association for Computing Machinery*, 1989:929–965, 1989.

[11] Daren C. Brabham. Crowdsourcing as a model for problem solving: An introduction and cases. In *Convergence: The International Journal of Research into New Media Technologies*, volume 14(1), pages 75–90, 2008.

[12] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[13] Wray Lindsay Buntine. *A Theory of Learning Classification Rules.* PhD thesis, University of Technology, Sydney, November 1992.

[14] James L. Carroll, Robbie Haertel, Peter McClanahan, Eric Ringger, and Kevin Seppi. Modeling the annotation process for ancient corpus creation. *Proceedings of the International Conference of Electronic Corpora of Ancient Languages (ECAL), or Chatressar 2007*, 2007.

[15] James L. Carroll, Christopher K. Monson, and Kevin D. Seppi. A bayesian cmac for high assurance supervised learning. *Applications of Neural Networks in High-Assurance Systems, NASA-IJCNN Workshop*, 2007.

[16] James L. Carroll and Kevin D. Seppi. No-free-lunch and bayesian optimality. *Meta-Learning IJCNN Workshop*, 2007.

[17] James L. Carroll, Neil Toronto, Robbie Haertel, and Kevin Seppi. Explicit utility in supervised learning. *NIPS Workshop on Cost-Sensitive Machine Learning*, 2008.

[18] Alexandre X. Carvalho and Martin L. Puterman. Dynamic pricing and learning over short time horizons. Working Paper, University of British Columbia, Vancouver, BC, Canada, 2003.

[19] Steffen Christensen and Franz Oppacher. What can we learn from no free lunch? a first attempt to characterize the concept of a searchable function. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2001)*, pages 1219–1226. Morgan Kauffman, 2001.

[20] David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. Active learning with statistical models. In *Journal of Artificial Intelligence Research*, volume 4, pages 129–145. AI Access Foundation and Morgan Kaufmann Publishers, 1996.

[21] I. Dagan and S. Argamon-Engelson. Committee-based sample selection for probabilistic classifiers. *Journal of Artificial Intelligence Research*, 11:335–360, 1999.

[22] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. *23rd International Conference on Machine Learning (ICML)*, 2006.

[23] Bruno de Finetti. Le prvision: ses lois logiques, ses sources subjectives. *Ann. Inst. Poincar*, 7:1–68, 1937.

[24] J. de Freitas, M. Niranjan, A. Gee, and A. Doucet. Sequential monte carlo methods for optimisation of neural network models. Technical Report TR-328, Cambridge University Engineering Department, Cambridge, England, 1998.

[25] Morris H. DeGroot. *Optimal Statistical Decisions.* McGraw-Hill Book Company, New York, New York, 1970.

[26] Morris H. DeGroot. *Probability and Statistics, Seceond Edition.* Addison-Wesley Publishing Company, Inc., Reading Massachusetts, 1986.

[27] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Jounal of the Royal Statistical Society*, B, 39 (1):1–38, 1977.

[28] Pedro Domingos. Bayesian averaging of classifiers and the overfitting problem. *Machine Learning*, pages 223–230, 2000.

[29] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification Second Edition.* John Wiley & Sons, Inc., New York, 2001.

[30] Charles Elkan. The foundations of cost-sensitive learning. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 973–978, 2001.

[31] M. Emmerich, A. Giotis, M. Ozdemir, T. Back, and K. Giannakoglou. Metamodel-assisted evolution strategies. In *Parallel Problem Solving from Nature (PPSN)*, volume VII, pages 361–370. Springer-Verlag, 2002.

[32] Yoav Freund, H. Sebastian Seung, Eli Shamir, and Naftali Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28(2-3):133–168, 1997.

[33] Andrew Gelman, John B. Carlin, Hal S. Stern, and Donald B. Rubin. *Bayesian Data Analysis, Second Edition.* Chapman Ad Hall, Texts in statistical Science Series, Boca Raton and London and New York and Washington, D.C., 2004.

[34] Philippe H. Gosselin and Matthieu Cord. A comparison of active classification methods for content-based image retrieval. In *CVDB '04: Proceedings of the 1st International Workshop on Computer Vision Meets Databases*, pages 51–58, New York, New York, USA, 2004. ACM.

[35] B. Hachey, B. Alex, and M. Becker. Investigating the effects of selective sampling on the annotation task. In *Proceedings of the 9th Conference on Computational Natural Language Learning*, 2005.

[36] Robbie Haertel, Eric Ringger, Kevin Seppi, James Carroll, and Peter McClanahan. Assessing the costs of sampling methods in active learning for annotation. In *Proceedings of the Conference of the Association of Computational Linguistics (ACL-NAACL: HLT 2008)*, 2008.

[37] Robbie A. Haertel, Kevin D. Seppi, Eric K. Ringger, and James L. Carroll. Return on investment for active learning. In *NIPS Workshop on Cost-Sensitive Machine Learning*, Whistler, British Columbia, Canada, 2008.

[38] Jr. Halbert L. White. Learning in artificial neural networks: A statistical perspective. *Neural Computation*, 1:425–464, 1989.

[39] David Haussler. Decision theoretic generalizations of the pac model for neural net and other learning applications. In *Information and Computation*, volume 100(1), pages 78–150, 1992.

[40] David Haussler and Manfred Warmuth. The probably approximately correct (PAC) and other learning models. In Tom Dietterich and Bill Swartout, editors, *8th National Conference on Artificial Intelligence*, pages 1101–1108, Cambridge, Massachusetts: MIT, 1990.

[41] Kristian Heal, Carl Griffin, Eric Ringger, Peter McClanahan, James Carroll, Joshua Heaton, and et al. A computational perspective on syriac corpus development and annotation. In *a Presentation Given at the XIXth Congress of the International Organization for the Study of the Old Testament (IOSOT), and the International Syriac Language Project (ISLP)*, July 2007.

[42] John H. Holland. *Adaptation in Natural and Artificial Systems.* The University of Michigan Press, 1975.

[43] Hume. *Hume's Treatise of Human Nature.* Stanford University Press, New York, 1739-1740.

[44] Christian Igel and Marc Toussaint. A no-free-lunch theorem for non-uniform distributions of target functions. *Journal of Mathematical Modelling and Algorithms*, 3:313–322, 2004.

[45] Edwin Thompson Jaynes. *Probability Theory: The Logic of Science.* University of Cambridge, Unighted Kingdom, 2003.

[46] Finn V. Jensen. *Bayesian Networks and Decision Graphs.* Springer-Verlag, New York, 2001.

[47] D. Jones, M. Schonlau, and W. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, 1998.

[48] Joseph Kadane. Comment. *Statistical Science*, 1(1), 1986.

[49] James Kennedy and Russell C. Eberhart. Particle swarm optimization. In *International Conference on Neural Networks IV*, pages 1942–1948, Piscataway, New Jersey, 1995. IEEE Service Center.

[50] James Kennedy and Russell C. Eberhart. *Swarm Intelligence.* Morgan Kaufmann Publishers, 2001.

[51] Andrey Nikolaevich Kolmogorov. *Foundations of the Theory of Probability.* Chelsea Pub. Co., Chelsea, New York, 1950.

[52] D. G. Krige. A statistical approach to some basic mine valuation problems on the witwatersrand. *Journal of the Chemical, Metal and Mining Society of South Africa*, 52(6):119–139, 1951.

[53] Pedro Larrañaga and José A. Lozano, editors. *Estimation of Distribution Algorithms : A New Tool for Evolutionary Computation*, volume 2. Kluwer Academic Publishers Group, 2001.

[54] J. Lave and E. Wenger. *Situated Learning: Legitimate Peripheral Participation.* Cambridge University Press, New York, 1991.

[55] Michael Lavine. What is Bayesian statistics and why everything else is wrong. Technical report, Duke University, North Carolina, 2000.

[56] D. D. Lewis and J Catlett. Heterogeneous uncertainty sampling for supervised learning. *W. W. Cohen and H. Hirsh (Eds.), Proceedings of ICML-94, 11th International Conference on Machine Learning*, pages 148–156, 1994.

[57] D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. *The 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3–12, 1994.

[58] Alexander Linden and Frank Weber. Implementing inner drive through competence reflection. In *Proceedings of the Second International Conference on From Animals to Animats 2 : Simulation of Adaptive Behavior*, pages 321–326, Cambridge, Massachusetts, USA, 1993. MIT Press.

[59] Stuart P. Lloyd. Least squares quantization. *PCM IEEE Transactions on Information Theory*, IT-2:129–137, 1982.

[60] D. MacKay. *Bayesian Methods for Adaptive Models.* PhD thesis, California Institute of Technology, Pasadend, CA, 1992.

[61] D. J. MacKay. Introduction to gaussian processes. In C. M. Bishop, editor, *Neural Networks and Machine Learning, vol. 168 of NATO Asi Series. Series F, Computer and Systems Sciences*. Springer Verlag, 1998.

[62] D. J. C. MacKay. A practical Bayesian framework for backprop networks. In J. E. Moody, S. J. Hanson, and R. P. Lippmann, editors, *Advances in Neural Information Processing Systems 4*, pages 839–846, 1991.

[63] David J. C. MacKay. Bayesian interpolation. *Neural Computation*, 4(3):415–447, 1992.

[64] David J. C. MacKay. The evidence framework applied to classification networks. *Neural Computation*, 4(5):720–736, 1992.

[65] David J. C. MacKay. Information-based objective functions for active data selection. *Neural Computation*, 4(4):590–604, 1992.

[66] David J. C. MacKay. A practical Bayesian framework for backprop networks. *Neural Computation*, 4(3):448–472, 1992.

[67] William G. Macready and David H. Wolpert. What makes an optimization problem hard? *Complexity*, 5, 1996.

[68] Dragos D. Margineantu. Active cost-sensitive learning. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence. Edinburgh, Scotland*, 2005.

[69] M. E. Maron. Automatic indexing: An experimental inquiry. *J. ACM*, 8(3):404–417, 1961.

[70] Ofer Matan. On-site learning. Technical report, Stanford, 1995.

[71] Thomas P. Minka. Bayesian model averaging is not model combination. Technical report, Carnegie Mellon University, 2000.

[72] Tom M. Mitchell. The need for biases in learning generalizations. Technical Report CBM-TR-117, Rutgers Computer Science, New Brunswick, New Jersey, 1980.

[73] Tom M. Mitchell. Generalization as search. *Artificial Intelligence*, 18(2), 1982.

[74] Tom M. Mitchell. *Machine Learning*. WCB/McGraw-Hill, USA, 1997.

[75] J. Mockus. *Bayesian Approach to Global Optimization*. Kluwer Academic Publishers, New York, 1989.

[76] J. Mockus, W. Eddy, A. Mockus, L. Mockus, and G. Reklaitis. *Bayesian Heuristic Approach to Discrete and Global Optimization*. Kluwer Academic Publishers, Dordrecht, Netherlands, 1996.

[77] Christopher K. Monson. *No Free Lunch, Bayesian Inference, and Utility: A Decision-Theoretic Approach to Optimization*. PhD thesis, Brigham Young University, Department of Computer Science, 2006.

[78] Christopher K. Monson and Kevin D. Seppi. The Kalman swarm. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2004)*, volume 1, pages 140–150, Seattle, Washington, 2004.

[79] Christopher K. Monson and Kevin D. Seppi. Bayesian optimization models for particle swarms. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2005)*, volume 1, pages 193–200, Washington, D.C., 2005.

[80] Christopher K. Monson, Kevin D. Seppi, and James L. Carroll. A utile function optimizer. In *The Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*. IEEE Press, Sept 2007.

[81] Bryan S. Morse and Duane Schwartzwald. Image magnification using level-set reconstruction. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages 333–340, 2001.

[82] Patrick Mullen. Learning in short-time horizons with measurable cost. Master's thesis, Brigham Young University, Department of Computer Science, Provo, Utah, USA, 2006.

[83] Patrick B. Mullen, Christopher K. Monson, and Kevin D. Seppi. Particle swarm optimization in dynamic pricing. In *Proceedings of the IEEE Congress on Evolutionary Computation (CECC)*, pages 4375–4382, Piscataway, New Jersey, July 2006. IEEE Press.

[84] Patrick B. Mullen and Kevin D. Seppi. Dynamic pricing with artificial neural netowrks. Tech Report, Brigham Young University, Department of Computer Science, Provo, UT, USA, 2006.

[85] Patrick B. Mullen, Kevin D. Seppi, and Sean C. Warnick. Dynamic pricing on commercial websites: A computationally intensive approach. In *Proceedings of the 8th Joint Conference on Information Sciences (JCIS)*, pages 1001–1004, Salt Lake City, Utah, July 2005.

[86] Ion Muslea, Steven Minton, and Craig A. Knoblock. Selective sampling with redundant views. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 621–626. AAAI Press / The MIT Press, 2000.

[87] Radford M. Neal. *Bayesian learning for neural networks.* Springer-Verlag New York, Inc., New York, 1996.

[88] D. J. Newman, S. Hettich, C.L. Blake, and C.J. Merz. UCI repository of machine learning databases, 1998.

[89] Blaise Pascal. *Pascal's Penseés with Introduction by T. S. Eliot.* E. P.l Dutton & Co., Inc., New York, 1670, translated 1958.

[90] H. Raiffa and R. Schlaifer. *Applied Statistical Decision Theory.* MIT Press, Cambridge, Mass., 1961.

[91] Adwait Ratnaparkhi. A maximum entropy model for part-of-speech tagging. In *Empirical Methods on Natural Language Processing (EMNLP)*, 1996.

[92] Eric Ringger, Marc Carmen, Robbie Haertel, Kevin Seppi, Deryle Lonsdale, Peter McClanahan, James Carroll, and Noel Ellison. Assessing the costs of machine-assisted corpus annotation through a user study. In *The Proceedings of the Language Resources and Evaluation Conference (LREC)*, Morocco, 2008.

[93] Eric Ringger, Peter McClanahan, Robbie Haertel, George Busby, Marc Carmen, James Carroll, Kevin Seppi, and Deryle Lonsdale. Active learning for part-of-speech tagging: Accelerating corpus annotation. In *Proceedings of the ACL Linguistic Annotation Workshop (LAW)*, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[94] Nicholas Roy and Andrew Mccallum. Toward optimal active learning through sampling estimation of error reduction. In *Procedings of the 18th International Conference on Machine Learning*, pages 441–448. Morgan Kaufmann, 2001.

[95] Stuart Russel and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, Englewood Cliffs, New Jersey, second edition, 2003.

[96] Michael James Sasena. *Flexibility and Efficiency Enhancements for Constrained Global Design Optimization with Kriging Approximations*. PhD thesis, University of Michigan, Department of Mchanical Engineering, 2002.

[97] Manabu Sassano. An empirical study of active learning with support vector machines for japanese word segmentation. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 505–512, Morristown, New Jersey, USA, 2001. Association for Computational Linguistics.

[98] C. Schaffer. Overfit avoidance as bias. In *Machine Learning*, volume 10, 1994.

[99] Andrew Ian Schein. *Active learning for logistic regression*. PhD thesis, University of Pennsylvania, Philadelphia, PA, USA, 2005. Supervisor-Lyle H. Ungar.

[100] J. Schmidhuber and J. Storck. Reinforcement driven information acquisition in nondeterministic environments. In *International Conference on Artificial Neural Networks*, 1995.

[101] Greg Schohn and David Cohn. Less is more: Active learning with support vector machines. In *Proc. 17th International Conf. on Machine Learning*, pages 839–846. Morgan Kaufmann, San Francisco, CA, 2000.

[102] Burr Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison, 2009.

[103] H. S. Seung, M. Opper, and H. Sompolinsky. Query by committee. *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, pages 287–294, 1992.

[104] Patrick Shafto and Noah Goodman. Teaching games: Statistical sampling assumptions for learning in pedagogical situations. In *Proceedings of the Thirtieth Annual Conference of the Cognitive Science Society*, 2008.

[105] B. E. Stuckman and E. E. Easom. A comparison of bayesian/sampling global optimization techniques. *IEEE Transactions on Systems, Man and Cybernetics*, 22(5):1024–1032, 1992.

[106] S. Thrun and K. Moeller. Active exploration in dynamic environments. *Advances in Neural Information Processing Systems*, 4:531–538, 1992.

[107] Sebastian Thrun and Lorien Pratt, editors. *Learning to Learn.* Morgan Kaufmann, San Francisco, California, 1998.

[108] K. Tomanek, J. Wermter, and U. Hahn. An approach to text corpus construction which cuts annotation costs and maintains reusability of annotated data. *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 486–495, 2007.

[109] Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. In *Journal of Machine Learning Research*, pages 999–1006. Morgan Kaufmann, 2000.

[110] Aimo Törn and Antanas Žilinskas. *Global Optimization.* Springer, Berlin, 1989.

[111] Neil Toronto, Bryan S. Morse, Kevin Seppi, and Dan Ventura. Super-resolution via recapture and Bayesian effect modeling. In *Computer Vision and Pattern Recognition (CVPR)*, 2009.

[112] Kristina Toutanova, Dan Klein, and Christopher D. Manning. Feature-rich part-of-speech tagging with a cyclic dependency network. *HLT-NAACL*, pages 252–259, 2003.

[113] Kristina Toutanova and Christopher D. Manning. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000)*, pages 63–70, 2000.

[114] L. G. Valiant. A theory of the learnable. *ACM*, 27(11):1134–1142, 1984.

[115] V. N. Vapnik and A. Ya. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971.

[116] Steven D. Whitehead. A study of cooperative mechanisms for faster reinforcement learning. Technical report, CS-365. University of Rochester, Rochester, New York, 1991.

[117] Darrel Whitley and Jean Paul Watson. Complexity theory and the no free lunch theorem. In Edumnd K. Burke and Graham Kendall, editors, *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*. Springer, 2006.

[118] David H. Wolpert, editor. *The Mathematics of Generalization*. The Addison-Wesley Publishing Company, 1995.

[119] David H. Wolpert. The relationship between PAC, the statistical physics framework, the bayesian framework, and the VC framework. *The Mathematica of Generalization*, pages 117–214, 1995.

[120] David H. Wolpert. The status of supervised learning science circa 1994: The search for a concensus. *The Mathematica of Generalization*, pages 1–10, 1995.

[121] David H. Wolpert. The lack of a priori distinctions between learning algorithms. *Neural Computation*, 8:1341–1390, 1996.

[122] David H. Wolpert. The supervised learning no-free-lunch theorems. Technical Report 269-1, NASA Ames Research Center, 2001.

[123] David H Wolpert and William G Macready. No free lunch theorems for search. Technical Report SFI-TR-05-010, Santa Fe Institute, 1995.

[124] David H. Wolpert and William G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, April 1997.

[125] Bianca Zadrozny and Charles Elkan. Learning and making decisions when costs and probabilities are both unknown. In *Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining*, pages 204–213. ACM Press, 2001.

[126] Yi Zhang. *Bayesian graphical models for adaptive filtering*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 2005. Chair-Jamie Callan.

[127] Xiaojin Zhu, John Lafferty, and Zoubin Ghahramani. Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *ICML 2003 workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, pages 58–65, 2003.