August 2017

Classification with Large Sparse Datasets: Convergence Analysis and Scalable Algorithms

Xiang Li
The University of Western Ontario

Supervisor
Charles X. Ling
The University of Western Ontario

Graduate Program in Computer Science

A thesis submitted in partial fulfillment of the requirements for the degree in Doctor of Philosophy

Abstract

Large and sparse datasets, such as user ratings over a large collection of items, are common in the big data era. Many applications need to classify the users or items based on the high-dimensional and sparse data vectors, e.g., to predict the profitability of a product or the age group of a user, etc. Linear classifiers are popular choices for classifying such datasets because of their efficiency. In order to classify the large sparse data more effectively, the following important questions need to be answered.

1. Sparse data and convergence behavior. *How different properties of a dataset, such as the sparsity rate and the mechanism of missing data systematically affect convergence behavior of classification?*

2. Handling sparse data with non-linear model. *How to efficiently learn non-linear data structures when classifying large sparse data?*

This thesis attempts to address these questions with empirical and theoretical analysis on large and sparse datasets. We begin by studying the convergence behavior of popular classifiers on large and sparse data. It is known that a classifier gains better generalization ability after learning more and more training examples. Eventually, it will converge to the best generalization performance with respect to a given data distribution. In this thesis, we focus on how the sparsity rate and the missing data mechanism systematically affect such convergence behavior. Our study covers different types of classification models, including generative classifier and discriminative linear classifiers. To systematically explore the convergence behaviors, we use synthetic data sampled from statistical models of real-world large sparse datasets. We consider different types of missing data mechanisms that are common in practice. From the experiments, we have several useful observations about the convergence behavior of classifying large sparse data. Based on these observations, we further investigate the theoretical reasons and come to a series of useful conclusions. For better applicability, we provide practical guidelines for applying our results in practice. Our study helps to answer whether obtaining more data or missing values in the data is worthwhile in different situations, which is useful for efficient data collection and preparation.

Despite being efficient, linear classifiers cannot learn the non-linear structures such as the low-rankness in a dataset. As a result, its accuracy may suffer. Meanwhile, most non-linear methods such as the kernel machines cannot scale to very large and high-dimensional datasets. The third part of this thesis studies how to efficiently learn non-linear structures in large sparse data. Towards this goal, we develop novel scalable feature mappings that can achieve better accuracy than linear classification. We demonstrate that the proposed methods not only out-

perform linear classification but is also scalable to large and sparse datasets with moderate memory and computation requirement.

The main contribution of this thesis is to answer important questions on classifying large and sparse datasets. On the one hand, we study the convergence behavior of widely used classifiers under different missing data mechanisms; on the other hand, we develop efficient methods to learn the non-linear structures in large sparse data and improve classification accuracy. Overall, the thesis not only provides practical guidance for the convergence behavior of classifying large sparse datasets, but also develops highly efficient algorithms for classifying large sparse datasets in practice.

Keywords: Machine learning, large-scale classification, data sparsity, classifier behavior

Acknowlegements

First of all, I would like to thank my supervisor, Dr. Charles Ling, who have guided me through the path of becoming a better researcher and have taught me important lessons in doing solid research. I am really grateful that Dr. Ling has always given me encouragement and actionable suggestions when facing difficulties in research. Without his supervision, this thesis would not have been possible.

My gratitute also goes to my thesis committee members, Dr. Xianbin Wang, Dr. John Barron, Dr. Michael Bauer and Dr. Xiaodan Zhu, who graciously agreed to serve on my committee.

I would like to express my thanks to my lab members, Yan Luo, Shuang Ao, Jun Wang, Chang Liu, Renfeng Liu, Xiao Li, Tanner Bohn for all the inspirations, encouragements and help, in research and also in real life. Many thanks to Dr. Shuang Ao, we have many cooporations in research and jointly published several papers. I would also like to thank Dr. Bin Gu, who had been a visiting scholar to our lab in 2014. Since then, Dr. Gu has given me much guidance in doing machine learning research, and we have coorporated in publication.

I would like to thank Dr. Huaimin Wang, the advisor of my Bachelor and Master study at National University of Defense Technology. He recommended me to conduct doctoral study at Western University and has always been supportive to me throughout my Ph.D. study.

Finally, my gratitude goes to my parents, for their love, sacrifice and tremendous support.

My research is supported by NSERC Grants, China Scholarship Council (CSC) and National Natural Science Foundation of China (No. 61432020, 61472430). This thesis would not have been possible without the generous resources provided by the Department of Computer Science, Western University.

Contents

List of Figures

List of Tables

List of Appendices

Acronyms

AdaGrad Adaptive Gradient algorithm. 13, 14

AUC Area Under the receiver operating characteristic Curve. ix, x, xii, 4, 34–37, 39, 43–48, 50, 52–61, 64, 81, 86, 102

BNB Bernoulli Naïve Bayes. 16

BTE Bernoulli-trial Expansion. vi, xi, 42–46, 48, 49, 52, 54, 57–59, 61, 68, 72, 73, 76–79, 81, 83–85

ERM Empirical Risk Minimization. 11, 13, 85

GNB Gaussian Naïve Bayes. 16

JE Just-1 Expansion. vi, xi, 42–44, 47–49, 52, 54, 57–59, 61, 68, 75–78, 80, 81, 83–85

KARMA Kernelized Algorithm for Risk-minimization with Missing Attributes. xii, 5–8, 87–89, 91, 93, 94, 97–100, 102

MAR Missing At Random. 40, 41, 43, 49, 66, 90

MCAR Missing Completely At Random. 23, 40, 90

MNAR Missing Not At Random. 40, 61, 90

MNB Multinomial Naïve Bayes. 16

PAC Probably Approximately Correct. 7, 18–20, 27, 32, 33, 58, 83, 101

pdf prodability density function. 21

PMF Probabilistic Matrix Factorization. vi, 19–21, 23, 49, 50, 52, 56, 61, 68, 69, 71, 72

SDCA Stochastic Dual Coordinate Ascent. xii, 13–15, 93, 94, 97, 99

SGD Stochastic Gradient Descent. 13, 14

SIBM Stochastic Inference method for Binary Matrices. 21–23

SRM Structural Risk Minimization. 13

SVM Support Vector Machine. v, xi, 4, 9, 10, 12, 13, 15, 17–20, 22, 24, 26–28, 30, 32–34, 63, 64, 67, 69, 72–76, 85, 87, 88, 95, 101

UD Uniform Dilution. x, 40–43, 64, 68–70, 72, 76

UM Uniform Missing. x, xi, 23, 64, 66–70, 74, 76–78, 81–85

VC-dimension Vapnik-Chervonenkis dimension. 11, 19, 32, 65, 83

Chapter 1

Introduction

Large and sparse datasets are prevalent in the big data era. Specifically, with the rapid emergence of online services and digital devices, tons of data that track our selections, usages and feedback are generated and stored. Meanwhile, as more items are being digitized, these user behavior datasets become sparser and sparser. For example, the famous Netflix dataset [3] contains $100,480,507$ ratings given by $480,189$ users to $17,770$ movies, where 98.822% of the rating values are missing. Apart from rating datasets, there are many more examples of large sparse data such as our *like*s on posts, photos and videos; our purchases on online shopping website; our clicks on webpages or advertisements, and so forth.

Throughout this thesis, we use *data sparsity* to denote the percentage of missing or unobserved values in a dataset. For a given dataset, its *missing pattern* can be represented by a binary matrix where each entry indicates missing (0) or not (1). Figure 1.1 uses a toy dataset to illustrate these different concepts. For the purpose of data modeling, we often assume that there exists a certain *missing data mechanism* that causes the missing pattern. For example, a simple missing data mechanism is that all data points are missing completely at random. We will further discuss different missing data mechanisms in later chapters.

It is no surprise that many datasets have such a high sparsity. With the huge number of items in a website, each user can only consume a small portion of them. Table 1.1 gives the number of users, items and data sparsity of several publicly-available datasets. Specifically, the movielens datasets[1] (*ml-1m, ml-100k*) and the yahoo-movies dataset[2] (*ymovie*) contain large sparse user ratings on a large number of movies. Meanwhile, the *tafeng* dataset[3] contains shopping records of a large number of users on the grocery products; the *book* dataset [93] is about user ratings

[1] http://grouplens.org/datasets/movielens/

[2] http://webscope.sandbox.yahoo.com/catalog.php?datatype=r

[3] http://recsyswiki.com/wiki/Grocery_shopping_datasets

	a_1	a_2	a_3	a_4
$\mathbf{x}_1$	3	1	2	4
$\mathbf{x}_2$	2	5	1	3
$\mathbf{x}_3$	5	5	3	1
$\mathbf{x}_4$	3	2	1	1

	a_1	a_2	a_3	a_4
$\mathbf{x}_1$	3	?	?	?
$\mathbf{x}_2$	?	5	?	?
$\mathbf{x}_3$	?	?	?	1
$\mathbf{x}_4$	3	?	?	?

	a_1	a_2	a_3	a_4
$\mathbf{x}_1$	1	0	0	0
$\mathbf{x}_2$	0	1	0	0
$\mathbf{x}_3$	0	0	0	1
$\mathbf{x}_4$	1	0	0	0

Figure 1.1: A toy dataset of four instances $\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\}$ and 4 features $\{a_1, a_2, a_3, a_4\}$. From left to right: the full dataset in hindsight; the actually observed dataset with 12 missing values (sparsity $s = 75\%$); and its missing pattern.

on books collected from the *bookcrossing.com* website; the epinions dataset [65] is about user ratings on various types of products collected from *epinions.com*. Finally, the flickr dataset [8] contains information about whether a user has *liked* a certain photo in *flickr.com*.

Classification is one of the most fundamental data analysis technique and is of great importance in mining and making use of the large sparse data. For example, a company may need to classify the profitability of a product/service based on sparse user feedback vectors. For the public datasets shown in Table 1.1, various types of data labels (shown in the last column) are also provided for building interesting and useful classifiers. Besides, in the literature, much previous work has demonstrated that sparse data are highly useful for predictive modeling. For example, Brian et al. [14] have demonstrated that classifying user web browsing data, which is high-dimensional and sparse, is an effective solution for online display advertising. Kosinski et al. [50] have used the *likes* in Facebook, which are sparse atomic behavioral data, to accurately predict the personality trait of each person. The same type of data have also been used in De Cnudde et al. [15] for improving micro-finance credit scoring. Meanwhile, large and sparse fine-grained transactional (invoicing) data have been used in Junqué de Fortuny et al. [41] to build effective linear classification models for corporate residence fraud detection. Mcmahan et al. [63] have demonstrated how extremely sparse data and linear classification can solve the advertisement click prediction tasks in the industry. Martens et al. [61] have used massive, sparse consumer payments data to build linear predictive models for targeted marketing.

All these prior works rely heavily on linear models in classifying large sparse data. The most important reason for doing this is efficiency: the time complexity of linear classifiers scale linearly with the number of non-missing values. This would be especially important because the sample size and feature dimension of these data could be extremely large, while predictions often have to be made in a real-time fashion [63]. Besides, high-dimensional and sparse data

Table 1.1: Several real-world large sparse datasets. *Instances* and *Features* correspond to the number of users and items in a dataset, respectively. *Sparsity* is calculated as the percentage of unobserved values in each dataset.

Datasets	*Instances*	*Features*	*Sparsity*	*Domains*	*Classification Label*
ml-100k	943	1,682	0.937	movie	user gender
ml-1m	6,040	3,952	0.958	movie	user gender
ymovie	7,620	11,914	0.998	movie	user age
tafeng	32,266	23,812	0.999033	grocery	user age
book	61,308	282,700	0.99996	book	user age
epinions	113,629	318,114	0.99997	product	user trustworthiness
flickr	11,195,143	497,470	0.9999987	photo	photo comment length

vectors tend to be linearly separable, which makes linear modeling effective for classifying such data. As a result, linear classification is still an essential tool for mining large sparse data in practice.

However, despite its scalability, a linear classifier is not always optimal in terms of classification accuracy. Notice that linear classifiers have assumed that the input vector $\mathbf{x}$ and the predicted value $f(\mathbf{x})$ is of a linear relationship, i.e., $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}$. As a result, they cannot learn complex data structures. In recent years, the success of matrix factorization algorithms in many collaborative filtering applications have verified the assumption that large sparse data tend to be *low-rank*, which cannot be fitted by a linear model. The low rank assumption also has a nice intuition as items/users can often be well-described by a relatively small number of latent factors. Obviously, the major challenge of applying non-linear models on large datasets is scalability. In this thesis, we will also try to address how to learn the non-linear data structures such as low-rankness, in a more scalable manner.

1.1 Research Questions

In order to classify large sparse data more effectively, several important questions need to be answered. The first question is about how data sparsity would affect classification performance:

(1). *How different properties of a dataset, such as the sparsity rate, the mechanism of missing data systematically affect convergence behavior of classification?*

Theoretically, a classifier's generalization ability improves as it learns from more and more training data. If the training size keeps increasing, it will converge to the optimal classifier for

a given hypothesis class. To visualize this theoretical result, in Figure 1.2, we conduct several classification experiments on large sparse datasets as an example. We train linear Support Vector Machine (SVM) using 5-fold cross validation with larger and larger sample size. As can be seen in the figures, both AUC (Area Under the receiver operating characteristic Curve) [22] and accuracy (0/1 loss) keep increasing until the entire datasets are used. Meanwhile, from the curves of the largest dataset we have (*flickr*), we could see the trend that AUC/accuracy starts to converge.

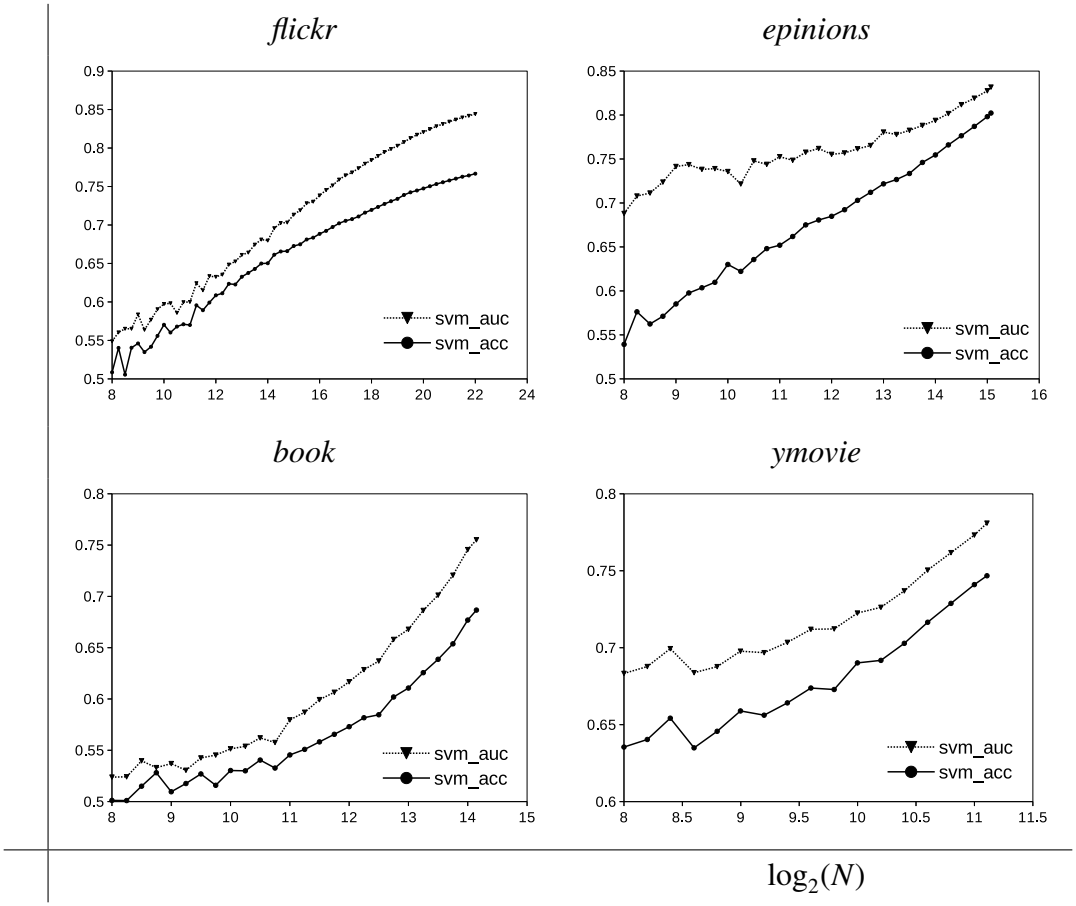


Figure 1.2: Learning curves of several real-world datasets measured by 5-fold cross-validation accuracy and AUC. All data samplings are repeated for 5 times, the regularization parameter of the SVM is tuned using a held-out dataset on the grid $[2^{-10}, 2^{-9}, \ldots, 2^{10}]$.

If we want to further improve classification performance, Figure 1.2 implies that collecting more data instances is almost always beneficial for these large sparse datasets. The same result has also been observed in the experiments performed by Junqué de Fortuny et al. [40], which suggest that *bigger data is indeed better*.

However, in order to have a deeper understanding of this observation or to estimate the

amount of data needed for getting a certain accuracy/AUC, we need to know how fast a learning curve converges and its converged value (asymptote). In this paper, we denote these two properties as the *convergence rate* and the *asymptotic performance* of learning, respectively. They are also basic problems in the Statistical Learning Theory [88], which have been extensively studied. Unfortunately, previous work has not studied how data sparsity under different missing data mechanisms would affect these two learning curve behaviors. Chapters 3-5 of this thesis will provide an in-depth study of this research question for (discriminative) linear classifiers and naïve Bayes classifier, which is a generative classifier.

Knowing the relationship between data sparsity and the learning convergence behavior of classification is necessary to estimate the gain (in terms of accuracy) and cost (in terms of data collection) of a sparse data classification task. Meanwhile, when a (large and sparse) dataset is given, it is important to have a classifier that can get better accuracy in a scalable manner. Though the popular linear classifiers are very efficient, they can hardly learn the non-linear structures in the data, hence accuracy may suffer. This motivates us to study the following research question:

(2). *How to efficiently learn non-linear data structures when classifying large sparse data?*

It is acknowledged that linear models do not have a rich learning capacity. As a result, they could hardly fit the complex structures in a dataset which in turn harms classification accuracy. In recent years, the field of collaborative filtering has witnessed the success of various low-rank methods in learning large sparse data, which implies that large sparse data tend to have latent structures that are non-linear (e.g., the low rank structure).

However, the biggest obstacle for applying non-linear models to large datasets is scalability. In fact, the very high feature dimensionality makes it hard to apply deep neural network methods; meanwhile, the large sample size also makes kernel machines intractable. Apart from scalability, another challenge is to find an appropriate non-linear model that can learn data low-rankness during classification. In this thesis, we will focus on the KARMA (Kernelized Algorithm for Risk-minimization with Missing Attributes) kernel proposed in [29], which has theoretical guarantee in learning low-rank structures in the sparse data. We investigate strategies to scale up the method for handling very large datasets.

1.2 Challenges and Our Approach

Here we summarize the challenges and our approach for each research question.

1.2.1 Large sparse data and learning convergence behavior

There is no universal answer for how data sparsity would affect learning convergence behavior. It depends on the *missing data mechanism* that causes data sparsity. Unfortunately, for a real-world sparse dataset, the actual mechanism that causes sparsity is usually complicated. For example, a missing movie rating does not necessarily mean that the user has not watched the movie. Some users may prefer to rate movies that they like, some may rate those they dislike while some may prefer to rate when (s)he disagrees with the current average rating. The missing pattern of a real-world large sparse dataset is a complex mixture of many different factors. To control this complexity, in this thesis, we consider several basic missing data mechanisms that are common for large sparse data in general.

Another challenge is how to empirically study the learning behaviors, especially the convergence behavior of classifying large sparse data. As can be seen in Figure 1.2, publicly available real-world sparse datasets are hardly large enough to reveal the convergence of classification. In this thesis, we address this challenge by using synthetic data generation. To ensure that the generated data are realistic, our approach is to sample data instances from the probabilistic model of the real-world datasets. After generating synthetic datasets, which can be arbitrarily large, we are able to systematically vary data sparsity with different missing data mechanisms and empirically study the convergence behaviors of different classifiers.

The third challenge is how to understand and verify the observations obtained from synthetic experiments. Specifically, do the observations hold consistently or do they hold only under certain conditions? For this purpose, we study different types of missing data mechanisms for binary data as well as real-valued data. Besides, our study covers different popular classifiers for large sparse data, including naïve Bayes and also discriminative linear classifiers. For these different classifiers, we provide in-depth theoretical analyses for our experiment observations based on previous results such as the Statistical Learning Theory [89] and the convergence analysis of naïve Bayes [69].

1.2.2 Large sparse data and non-linear learning

As demonstrated in previous studies on collaborative filtering tasks, low-rankness is arguably the most significant non-linear structure that exists in many large sparse data. For classification tasks, Hazan et al. [29] have developed the KARMA framework for learning low-rank structures in the sparse data. Under the low-rank assumption, the framework has theoretical guarantee to outperform linear classifiers, which was demonstrated with experiments on sev-

eral small to medium sized data [29]. However, the KARMA framework is hardly scalable to large datasets because of the expensive kernel computation. In Chapter 6 of this thesis, we investigate the inner mechanism of the KARMA kernel with theoretical and empirical analyses. We find that a KARMA kernel is not only possible but also effective (in terms of accuracy) to be approximated using scalable feature mapping strategies. We demonstrate that the proposed feature mappings not only significantly outperform linear classification, but also is scalable for large and sparse datasets even on a normal desktop computer.

1.3 Thesis Structure

In this introductory chapter, we have described the problem, the motivation and the research questions of the thesis. In Chapter 2, we describe the theoretical background of classifying large sparse data. We introduce different types of efficient classifiers on large sparse data, including naïve Bayes which is a generative classifier and discriminative linear classifiers such as linear SVM and Logistic Regression [23]. In particular, we address the reason why these classifiers can efficiently handle data sparsity.

In Chapter 3, we study how data sparsity could affect the convergence behavior of a linear SVM classifier. We propose a novel approach to generate large sparse binary data from real-world datasets, using statistical inference and the data sampling process of the PAC (Probably Approximately Correct) framework [89]. We then study the convergence behavior of linear SVM with synthetic experiments, and make several important observations. We also offer theoretical proofs for our observations by studying the Bayes risk and PAC bound. These experimental and theoretical results are valuable for learning large sparse datasets with linear SVM.

In Chapter 4, we extend our study to naïve Bayes classifier which is a simple generative model widely used on large sparse data because of its efficiency. For naïve Bayes, we study how different mechanisms of missing data, data sparsity and the number of attributes systematically affect its learning curves and convergence. We consider several common missing data mechanisms and propose novel data generation methods based on these mechanisms. We generate large and sparse data systematically, and study the entire AUC learning curve and convergence behavior of naïve Bayes. We not only have several important observations, but also provide detailed theoretical studies. Finally, we summarize the results as a guideline for classifying large sparse data in practice.

In Chapter 5, we further extend our study from linear SVM and naïve Bayes to all discrim-

inative linear classifiers. While the previous chapters only consider missing data mechanisms and synthetic generations of binary data, we now generalize them to real-valued datasets. Using synthetic experiments, we observe several important learning curve behaviors under different missing data mechanisms. We derive several lemmas which prove that our observations consistently hold for different linear classifiers and different loss measures. Practically, our studies help to determine if or when obtaining more data and/or obtaining missing values in the data is worthwhile or not. This can be very valuable in many applications.

In Chapter 6, we study how to efficiently learn non-linear structures when classifying large sparse datasets. Many studies suggest that large sparse data often have a low-rank structure. By finding the polynomial approximation to the low-rank space, Hazan et al. [29] developed a kernel algorithm (KARMA) for classifying such datasets with a higher accuracy. In Chapter 6, we develop novel scalable feature mappings to efficiently approximate the kernels used in KARMA. In experiments, our method is comparable with KARMA on medium-sized data and scales well to larger datasets that KARMA does not. Our method also significantly outperforms linear classifiers on datasets of various sizes.

Finally, in Chapter 7, we conclude the thesis and discuss possible future directions.

Chapter 2

Background

In this chapter, we introduce the basic settings of a standard classification task. We distinguish the definitions of a generative classifier and a discriminative classifier. In the second part of this chapter, we introduce the basic learning convergence behavior of these classifiers following the Statistical Learning Theory [89]. Finally, we give some concrete examples of classification algorithms which will be used in later chapters, including linear SVM, logistic regression and naïve Bayes. We not only describe their problem formulations and solutions but also illustrate their efficiency in dealing with sparse data.

2.1 Classification

The general purpose of a classification algorithm is to map a given input $\mathbf{x}$ to its output y, where y is the target of interest. Formally, let us assume that the data space of input $\mathbf{x}$ is $\mathcal{X} = \mathbb{R}^d$, and the space of the target variable (label) is $\mathcal{Y} = \{1, 2, \ldots, C\}$ (while for regression task the domain of $\mathcal{Y}$ is continuous), where C is the number of classes and d is the number of input features. The data are assumed to be i.i.d. (individually independently distributed) according to distribution $\mathcal{D} =^{def} p(\mathbf{x}, y)$ over space $\{\mathcal{X} \times \mathcal{Y}\}$.

For learning, the classifier is given a set of training data $S_{\text{train}} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1,\ldots,N}$ which are also distributed according to $\mathcal{D}$. The goal of classification is to find a *decision function* f using the training data, such that for an unseen input $\mathbf{x}$, $f(\mathbf{x})$ can provide a good prediction of its unknown label y, where $(\mathbf{x}, y)$ is also from $\mathcal{D}$. Based on how the decision function f is learned, we can divide classifiers into *discriminative* classifiers and *generative* classifiers.

Discriminative Classifiers. A discriminative classifier tries to either directly learn $p(y|\mathbf{x})$ (without learning $p(\mathbf{x}|y)$ and $p(y)$, e.g., Logistic Regression [23]) or to directly find the mapping

f (such as SVM) from a predefined class of functions $\mathcal{H}$.

Following the Statistical Learning Framework [89], a discriminative classifier will target a non-negative loss function $L(f(\mathbf{x}), y) \in \mathbb{R}^+$, which quantifies the penalty if the true target for $\mathbf{x}$ is y, while the predicted target is $f(\mathbf{x})$. The goal of the training process is to minimize the following quantity, which is denoted as the *generalization risk* (true risk) of classification

$$R_{\mathcal{D}}(f) = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\Big[L(f(\mathbf{x}), y)\Big] = \int_{\mathcal{X}\times\mathcal{Y}} L(f(\mathbf{x}), y)dp(\mathbf{x}, y). \tag{2.1}$$

The optimal function will be

$$f^{\text{Discriminative}} = \arg\min_{f\in\mathcal{H}} \{R_{\mathcal{D}}(f)\}, \tag{2.2}$$

which can then be used for predicting unseen data in the future. When the class of functions $\mathcal{H}$ is rich enough to contain all possible mappings, $\mathcal{H} = \{f : \mathcal{X} \to \mathcal{Y}\}$, the optimal $f^{\text{Discriminative}}$ will be equivalent to the *Bayes classifier* which knows everything about the distribution $p(y|\mathbf{x})$:

$$f^{\text{Bayes}} = \arg\min_{f:\mathcal{X}\to\mathcal{Y}} \{R_{\mathcal{D}}(f)\} \tag{2.3}$$

$$f^{\text{Bayes}}(\mathbf{x}) = \arg\min_{y'} p(y'|\mathbf{x})L(y', y), \tag{2.4}$$

and the risk incurred by the Bayes classifier is denoted as the *Bayes risk*:

$$R^{\text{Bayes}}_{\mathcal{D}}(f) = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\Big[L(f^{\text{Bayes}}(\mathbf{x}), y)\Big]. \tag{2.5}$$

It is obvious that the Bayes risk is the smallest risk a classifier can get given $\mathcal{D}$, even in an idealistic setting.

Generative Classifiers. Unlike a discriminative classifier, a generative classifier tries to estimate $p(\mathbf{x}|y)$ and $p(y)$ by assuming a certain probabilistic model M for the data. The model M is a prior belief of what $p(\mathbf{x}|y)$ and $p(y)$ should be like, which in turn allows the classifier to estimate the two distributions, $\hat{p}(\mathbf{x}|y, M)$ and $\hat{p}(y)$, from the given data. Afterwards, the classifier uses the Bayes rule to find the estimation of $p(y|\mathbf{x})$:

$$p(y|\mathbf{x}) \approx \hat{p}(y|\mathbf{x}, M) = \frac{\hat{p}(\mathbf{x}|y, M)\hat{p}(y)}{\hat{p}(\mathbf{x})}. \tag{2.6}$$

Finally, it acts the same as a Bayes classifier (Eq. (2.4)) in learning f, except it uses the estimated distribution rather than the true distribution:

$$f^{\text{Generative}} = \arg\max_{y} \hat{p}(y|\mathbf{x}, M) \tag{2.7}$$

as the learned mapping.

Ng and Jordan [69] have provided detailed discussion of the advantages and disadvantages of discriminative and generative classifiers. However, this is not the focus of this thesis. In the previous chapter, we have demonstrated how linear classifiers behave as we increase the size of large sparse data (Figure 1.2). Here we introduce the theoretical background behind this phenomenon, i.e., the convergence of classification.

2.2 Convergence of Discriminative Classifiers

As described earlier, since only the training data $\mathcal{S}_{\text{train}}$ are given, a discriminative classifier will use the *Empirical Risk*

$$R^{\text{emp}}_{\mathcal{S}_{\text{train}}\sim\mathcal{D}}(f) = \frac{1}{N}\sum_{i=0}^{N} L(f(\mathbf{x}^{(i)}), y^{(i)}) \tag{2.8}$$

to estimate the generalization risk in Equation (2.1). A most simple discriminative classifier will use the ERM (Empirical Risk Minimization) principle [89] for training[1]. In other words, it finds the decision function by

$$f^*_{\mathcal{S}_{\text{train}}} := \arg\min_{f\in\mathcal{H}} \left\{R^{\text{emp}}_{\mathcal{S}_{\text{train}}\sim\mathcal{D}}(f)\right\}. \tag{2.9}$$

Because of the Uniform Convergence of ERM [88], when N becomes sufficiently large, the empirical risk, $f^*_{\mathcal{S}_{\text{train}}}$ will converge to the optimal generalization (true) risk:

$$R^{\text{emp}}_{\mathcal{S}_{\text{train}}\sim\mathcal{D}}(f^*_{\mathcal{S}_{\text{train}}}) \to^{p}_{N\to\infty} \min_{f\in\mathcal{H}}\{R_{\mathcal{D}}(f)\} := \hat{R}_{\mathcal{D}}, \tag{2.10}$$

Here the symbol $\to^{p}_{N\to\infty}$ represents the *convergence in probability* when $N \to \infty$, which follows the definition in Vapnik et al. [88]. We denote the *asymptotic risk* as $\hat{R}_{\mathcal{D}}$. Notice that the above results hold so long as $\mathcal{H}$ has bounded complexity (VC-dimension [88]), which is true for discriminative *linear classifiers*:

$$\mathcal{H}_{\text{linear}} = \{f | f(\mathbf{x}) = \mathbf{w}\cdot\mathbf{x}, \mathbf{w}\in\mathbb{R}^d\}. \tag{2.11}$$

As we increase training size, N, the *convergence rate* of learning describes how fast the empirical risk converges to the asymptotic risk $\hat{R}_{\mathcal{D}}$. For illustration purpose, Figure 2.1 gives the convergence behavior of empirical risk and true risk of the discriminative training process.

[1] here we ignore the regularization problem, which will be discussed later

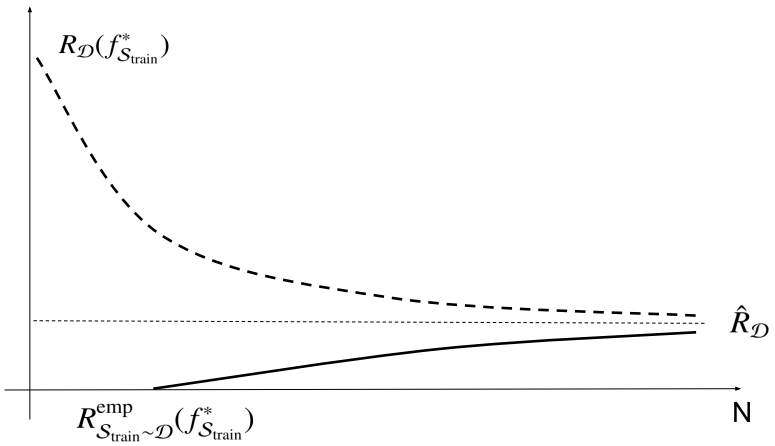


Figure 2.1: As the training size N increases, the empirical risk ($R^{\text{emp}}_{\mathcal{S}_{\text{train}}\sim\mathcal{D}}$) and true risk ($R_{\mathcal{D}}$) of a classifier $f^*_{\mathcal{S}_{\text{train}}}$ will converge.

2.3 Convergence of Generative Classifiers

The learning convergence behavior of a generative classifier is similar to that of a discriminative classifier. Basically, the learning process converges because of the convergence of the estimated probabilities to their true values. In other words, the optimal classifier $f^{\text{Generative}}(\mathbf{x})$ computed by:

$$f^{\text{Generative}}(\mathbf{x}) = \arg\max_y \hat{p}(y|\mathbf{x}, M) = \arg\max_y \hat{p}(\mathbf{x}|y, M)\hat{p}(y) \tag{2.12}$$

converges because of the convergence of $\hat{p}(\mathbf{x}|y, M)$ and $\hat{p}(y)$ to their corresponding asymptotes. Finally, if we quantify the predictive performance using the risk of classification in Equation (2.1), we expect a very similar convergence curve as the discriminative classifiers.

In the next subsection, we describe concrete examples of discriminative and generative classifiers, which will be used in the thesis.

2.4 Discriminative Linear Classifiers

Because of their efficiency in dealing with large datasets, we focus on discriminative linear classifiers, including linear SVM and Logistic Regression. A practical algorithm for discriminative classification often optimizes an objective function of the form

$$\min_{f\in\mathcal{H}} R^{\text{emp}}_{\mathcal{S}_{\text{train}}\sim\mathcal{D}}(f) + \lambda C(f), \tag{2.13}$$

where $C(f)$ quantifies the complexity of the function f and λ is the parameter that controls the strength of regularization. This strategy (adding regularization to ERM principle) is known as the Structural Risk Minimization (SRM) principle [89], which is especially necessary when the sample size is small.

Without loss of generality, consider the binary classification problem where $\mathcal{Y}$ has only two distinct elements $\{+1, -1\}$. Both linear SVM and Logistic Regression consider the following objective function:

$$\min_{\mathbf{w}\in\mathbb{R}^d} \frac{1}{N}\sum_{i=1}^{N} L(f(\mathbf{x}^{(i)}), y^{(i)}) + \frac{\lambda}{2}||\mathbf{w}||^2, \tag{2.14}$$

where $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ is the decision function whose sign can be used for prediction. For linear SVM, we set the loss function as:

$$L_{\text{hinge}}(f, y) = \max\{0, 1 - yf\}, \tag{2.15}$$

which is known as the *hinge loss*, while for logistic regression, we use the *logistic loss*:

$$L_{\text{logistic}}(f, y) = \ln(1 + e^{-yf}). \tag{2.16}$$

There are many other loss functions for linear prediction, such as squared loss $L_{\text{squared}}(f, y) = (y - f)^2$, and squared hinge loss $L_{\text{squared-hinge}}(f, y) = (\max\{0, 1 - yf\})^2$. These commonly used loss functions are actually convex surrogates [2] of the *error rate* (0-1 loss), which is defined as $L_{0\text{-}1}(f, y) = \mathbb{I}(\text{sign}(f) \neq y)$, where $\mathbb{I}(A)$ equals 1 if event A is true and 0 otherwise..

There are many algorithms and their corresponding software packages that can solve Equation (2.14). Traditional methods use the quadratic programming algorithms which cannot scale well to large scale learning problems. Recently, efficient solvers have been proposed using SGD (Stochastic Gradient Descent) [81, 19] or coordinate ascent [34, 82], which work on the primal and dual problem of Eq. (2.14), respectively. To illustrate the efficiency of these modern solvers on learning sparse data, we briefly review the AdaGrad (Adaptive Gradient algorithm) [19] and the SDCA (Stochastic Coordinate Dual Ascent) [82] algorithm as representatives of each family of optimizers.

AdaGrad Let us consider the standard SGD algorithm. At iteration t, SGD randomly chooses a data instance $(\mathbf{x}^{(t)}, y^{(t)})$ to observe, then updates the model parameter of (2.14) by

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t \frac{\partial h_t}{\partial \mathbf{w}_t}, \tag{2.17}$$

where η_t is the learning rate and $h_t = L(f(\mathbf{x}^{(t)}), y^{(t)}) + \frac{\lambda}{2}\|\mathbf{w}\|^2$ is the regularized loss for the current instance. With a good step-size strategy for η_t, f_t can converge to the optimal linear function with $\tilde{O}(1/t)$ convergence rate.

Compared to other SGD variants, AdaGrad [19] is the most suitable solver for learning sparse data. The most important design of AdaGrad is to update each attribute with different learning rates to adapt for their different sparsity rates. Unlike Equation (2.17), AdaGrad updates using:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t G_t^{-1/2} \frac{\partial h_t}{\partial \mathbf{w}_t}, \tag{2.18}$$

where

$$G_t = \sum_{\tau=1}^{t} \frac{\partial h_\tau}{\partial \mathbf{w}_\tau} (\frac{\partial h_\tau}{\partial \mathbf{w}_\tau})^T \tag{2.19}$$

is a $d \times d$ matrix whose diagonal elements are the sums of squared historical gradients on each attribute. This adaptation is especially useful because sparsity rates for each attribute are often quite different.

SDCA As its name suggests, SDCA [82] works on the dual problem of Eq. (2.14) using stochastic coordinate ascent. If we consider Eq. (2.14) as the primal problem:

$$\min_{\mathbf{w}\in\mathbb{R}^d} P(\mathbf{w}), \text{ where } P(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N} L(f(\mathbf{x}^{(i)}), y^{(i)}) + \frac{\lambda}{2}\|\mathbf{w}\|^2 \tag{2.20}$$

its dual can be formulated as:

$$\max_{\alpha\in\mathbb{R}^N} D(\alpha), \text{ where } D(\alpha) = \frac{1}{N}\sum_{i=1}^{N} -\phi_i^*(-\alpha^{(i)}) - \frac{\lambda}{2}\|\frac{1}{N\lambda}\sum_{i=1}^{N} \alpha^{(i)}\mathbf{x}^{(i)}\|^2 \tag{2.21}$$

α is the N-dimensional dual coefficient and $\phi_i^*(\cdot)$ is the convex conjugate of $L(\cdot)$, see [83] for detail.

The duality theory implies that the gap between the primal and dual objectives are always non-negative:

$$\Delta_{\text{gap}}(\alpha) := P(\mathbf{w}(\alpha)) - D(\alpha) \geq P(\mathbf{w}(\alpha)) - P(\mathbf{w}^*) \geq 0, \tag{2.22}$$

where $\mathbf{w}(\alpha) := \frac{1}{N\lambda}\sum_{i=1}^{N} \alpha^{(i)}\mathbf{x}^{(i)}$. This *duality gap* can be used to measure the accuracy of a current solution α, w.r.t. the optimal one.

The implementation of the SDCA solver is simple. Briefly speaking, SDCA initializes α to be $\alpha_0 = \mathbf{0}$. At each iteration t, it randomly picks an element $\alpha_t^{(i)}$ from $\{\alpha_t^{(1)}, \alpha_t^{(2)} \ldots, \alpha_t^{(N)}\}$ (with replacement), and updates $\alpha_t^{(i)}$ to be $\alpha_{t+1}^{(i)}$ by maximizing the dual objective in Eq. (2.21) along

the selected coordinate i. This process is repeated until the duality gap is sufficiently small, e.g., $\Delta_{\text{gap}}(\alpha) < 0.001$.

While this idea is quite simple, we focus more on its efficiency in dealing with highly-sparse data. The analysis in Shalev et al. [82, 83] implies that the time complexity for the SDCA solver to achieve $\Delta_{\text{gap}}(\alpha_t) \leq \epsilon$ on the logistic regression problem is

$$\tilde{O}\Big((1-s)\cdot d\Big(N + \min\Big\{\frac{R^2}{\lambda}, \sqrt{\frac{NR^2}{\lambda}}\Big\}\Big)\Big) \tag{2.23}$$

while on the linear SVM problem is:

$$\tilde{O}\Big((1-s)\cdot d\Big(N + \min\Big\{\frac{R^2}{\lambda\epsilon}, \sqrt{\frac{NR^2}{\lambda\epsilon}}\Big\}\Big)\Big), \tag{2.24}$$

where $R := \max_i \|x_i\|^2$ measures the radius of the input. As before, s represents the sparsity rate (percentage of zeros) in the data. It can be seen that this algorithm has nice scalability since it only scales linearly with the number of non-missing values (i.e., $(1-s)dN$).

We would like to mention that AdaGrad and SDCA have similar optimization speed according to our empirical results on classifying large sparse data. In our later experiments, we extensively use SDCA as the optimizer for discriminative linear classifiers on large sparse data.

2.5 Naïve Bayes Classifier

Naïve Bayes makes assumptions about the generation process and the underlying distribution of the observed data. This is different from the discriminative classifiers described above, which have adopted a *distribution free* approach.

Specifically, the Naïve Bayes model assumes that the data distribution of each feature x_j is independent of one another, when its label y is given. In other words, the conditional probability $p(\mathbf{x}|y)$ can be factorized as:

$$p(\mathbf{x}|y) = \prod_{j=1}^{d} p(x_j|y). \tag{2.25}$$

Finally, a classification decision is made by

$$f_{\text{NB}}(\mathbf{x}) = \arg\max_y p(y)p(\mathbf{x}|y) = \arg\max_y p(y)\prod_{j=1}^{d} p(x_j|y). \tag{2.26}$$

The training process of a naïve Bayes classifier is basically to estimate all the probability densities appearing in the right-hand side of Equation (2.26). Notice that the naïve indepen-

dence assumption greatly simplifies density estimation, especially when the number of dimension d is high: it reduces the density estimation of the joint distribution $p(\mathbf{x}|y)$ to the estimation of a collection of univariate probabilities.

In practice, there are different prior models for estimating the density of $p(x_j|y)$ for different types of data. For example, when $x_j \in \mathbb{R}$ is a continuous variable, $p(x_j|y)$ is often estimated using a one-dimensional Gaussian prior:

$$p(x_j|y) \sim \mathcal{N}(\mu_{j,y}, \sigma^2_{j,y}), \tag{2.27}$$

where parameters $\mu_{j,y}$ and $\sigma^2_{j,y}$ are computed using maximum likelihood estimation. This model is denoted as Gaussian Naïve Bayes (GNB) classifier.

In situations like text classification, a feature value x_j may represent a discrete value. For example, whether each word of the vocabulary appears in a document (binary) or its frequency of appearance (ordinal). In this case, $p(x_j|y)$ can be modeled using either a multinomial distribution or a multivariate Bernoulli distribution, which leads to the Multinomial Naïve Bayes (MNB) and Bernoulli Naïve Bayes (BNB) classifier, respectively. One of the major difference between the two models is that BNB only considers the binary state of whether each event occurs while MNB also models the frequency of appearance. In practice, both BNB and MNB are highly effective for classifying sparse data such as text documents, though each model has its own preferred setting [62].

We would like to mention that MNB is also a linear classifier in the sense that the decision function can be represented as a linear combination of the input feature values; see Rennie et al. [75] for the detailed derivation. Meanwhile, BNB is not a linear classifier. In Chapter 4, we will study BNB and its convergence behavior on large sparse data.

For handling sparse data, a Gaussian Naïve Bayes model can simply ignore the features with missing values in computing the decision function in Eq. (2.26). Meanwhile, both MNB and BNB can naturally model the missing features because both Multinomial and Bernoulli distribution can model the absence of an event. Moreover, the time complexity of a naïve Bayes classifier scales linearly with the number of non-missing values only. Take BNB as an example, estimating the marginal probability $p(x_j|y)$ amounts to counting the number of non-missings in feature x_j and class y.

2.6 Conclusion

In this chapter, we have introduced the general setting of a classification task. Moreover, we introduce the discriminative and also generative classifiers which are trained using different

principles. To illustrate the key issues of large-scale classification, we also reviewed the theories of learning convergence behavior for different types of classifiers. Finally, we use concrete algorithms of discriminative linear classifiers and naïve Bayes classifiers to illustrate how these classifiers handle large sparse data and achieve fast learning.

Starting from the next chapter, we present the major contributions of the thesis, i.e., analyzing how data sparsity would affect the convergence behavior of a classifier. Our study covers both discriminative linear classifiers such as linear SVM and also generative classifier such as naïve Bayes.

Chapter 3

Data Sparsity in Linear SVM

Large sparse datasets are common in many real-world applications. Linear SVM has been shown to be very efficient for classifying such datasets. However, it is still unknown how data sparsity[1] would affect its convergence behavior. To study this problem in a systematic manner, we propose a novel approach to generate large and sparse data from real-world datasets, using statistical inference and the data sampling process of the PAC framework [89]. We first study the convergence behavior of linear SVM experimentally, and make several observations, useful for real-world applications. We then offer theoretical proofs for our observations by studying the Bayes risk and PAC bound. Our experiment and theoretical results are valuable for learning large sparse datasets with linear SVM.

3.1 Introduction

As described earlier, large sparse datasets are common in many real-world applications. Linear SVM solvers such as Pegasos [81] and LibLinear [21] are popular choices to classify large sparse datasets efficiently, because they scale linearly with the number of non-missing values. Nowadays, it is possible to train linear SVM on very large datasets. Theoretically, it is known that larger training data will lead to lower generalization error, and asymptotically it will converge to the lowest error that can be achieved [88]. However, it is still hard to answer the following important questions about linear SVM and data sparsity:

(1). If we put in effort to reduce data sparsity, would it decrease the asymptotic generalization error of linear SVM?

[1] In our published manuscript of this chapter [54], we use the term *sparseness*. However, *sparsity* and *sparseness* refer to the same concept. For consistency, we use data sparsity throughout this thesis

(2). Would data sparsity affect the amount of training data needed to approach the asymptotic generalization error of linear SVM?

These questions essentially concern the convergence behavior of learning, which has been addressed in previous works. In Statistical Learning Theory [88], PAC bound gives a high-probability guarantee on the convergence between training and generalization error. Once the VC-dimension of the problem is known, PAC bound can predict the amount of training instances needed to approach the asymptotic error. Barlett et al. [1] have shown that the VC-dimension of linear SVM is closely related to the hyperplane margin over the data space. As an initial step of understanding the impact of data sparsity on the margin of hyperplanes, Long and Servedio [57] have given bounds for integer weights of separating hyperplanes over the k-sparse Hamming Ball space $\mathbf{x} \in \{0,1\}^m_{\leq k}$ (at most k of the m attributes are non-zero). There also exist several SVM variants that could deal with missing data [70, 9]. However, there is still no work could explicitly predict the convergence behavior of linear SVM when data is highly sparse.

In this chapter, we will answer this question by systematic experiments and then verify our findings through theoretical study. We propose a novel approach to generate large and sparse synthetic data from real-world datasets. First, we infer the statistical distribution of real-world movie rating datasets using a recent Probabilistic Matrix Factorization (PMF) inference algorithm [32]. From the inferred distribution, we then sample a large number of data instances following the PAC framework, so we can study the generalization error with various training sizes.

In order to study the effect of data sparsity, we consider a simple missing data model, which allows us to systematically vary the degree of data sparsity and compare the learning curves of linear SVM. To follow the PAC framework, we study the curves of training and testing error rates as we keep increasing the training size. We have made several important observations about how data sparsity would affect the asymptotic generalization error and the rate of convergence when using linear SVM; see Section 3.3.

We then analyze our observations with a detailed theoretical study. For asymptotic generalization error, we study the change of Bayes risk as we vary data sparsity. With proper assumptions, we have proved that higher sparsity will increase the Bayes risk of the data, see Section 3.4.1. For the asymptotic rate of convergence, we observe that different sparsity would not change the amount of training data needed to approach convergence. We study its theoretical rationale using the PAC bound, see Section 3.4.2.

Our results are very useful for using linear SVM in real-world applications. Firstly, they

indicate that sparser data would generally increase the asymptotic error, which encourages practitioners to put effort in reducing data sparsity. Secondly, our results also imply that sparser data will not lead to slower learning curve convergence when using linear SVM, although the asymptotic generalization error rate would increase.

The rest of the chapter is organized as follows. Section 3.2 gives details of our data generation approach. Section 3.3 describes our experiment and observations. Section 3.4 provides mathematical proofs for our observations. Section 3.5 concludes the chapter.

3.2 A Novel Approach to Generate Sparse Data

To systematically study the impact of data sparsity on linear SVM, in this section, we first propose a novel approach to generate sparse data.

3.2.1 Basic Settings

To simplify the problem setting, we only consider binary attribute values $\mathbf{x} \in \{0, 1\}^m$ and binary label $y \in \{+, -\}$. Our data generation approach follows the standard setting of the PAC framework [88], which assumes that data $(\mathbf{x}, y)$ are independently sampled from a fixed distribution $P(\mathbf{x}, y) = P(y)P(\mathbf{x}|y)$. In addition, our approach uses the distribution inferred from real-world datasets, so the generated data are realistic in a statistical sense. Specifically, we use the datasets of recommendation systems, such as movie ratings to infer $P(\mathbf{x}, y)$.

Recently, datasets of recommendation systems are widely studied. They usually contain ordinal ratings (e.g., 0 to 5) given by each user to each item. We consider each user as a data instance and its rating on items corresponds to each attribute. To make the data binary, we only consider the presence/absence of each rating. We choose the gender of each user as its label $y \in \{+, -\}$.

To infer the distribution $P(\mathbf{x}, y)$, we use Probabilistic Matrix Factorization [67], which is a widely-used probabilistic modeling framework for user-item ratings data. In the next section, we will briefly review the PMF model for binary datasets proposed by Hernandez et al. [32], which will be used in our data generation process.

3.2.2 Review of PMF for Sparse Binary Data

Consider L users and M items, the $L \times M$ binary matrix $\mathbf{X}$ indicates whether each rating exists. In this case, $\mathbf{X}$ could be modeled by the PMF given in Hernandez et al. [32] which has good

predictive performance:

$$p(\mathbf{X}|\mathbf{U}, \mathbf{V}, z) = \prod_{i=1}^{L} \prod_{j=1}^{M} p(x_{i,j}|\mathbf{u}_i, \mathbf{v}_j, z), \tag{3.1}$$

where a Bernoulli likelihood is used along with the Matrix Factorization assumption $\mathbf{X} \approx \mathbf{U} \cdot \mathbf{V}$ to model each binary entry of $\mathbf{X}$ as:

$$p(x_{i,j}|\mathbf{u}_i, \mathbf{v}_j, z) = Ber\left(x_{i,j}|\delta(\mathbf{u}_i \mathbf{v}_j^T + z)\right) \tag{3.2}$$

where $\delta(\cdot)$ is the logistic function

$$\delta(x) = \frac{1}{1 + exp(-x)}$$

used to 'squash' a real number into the range of $(0, 1)$, $Ber(\cdot)$ is the pdf (prodability density function) of a Bernoulli distribution and z is a global bias parameter in order to handle data sparsity.

This PMF model further assumes that all latent variables are independent by using fully factorized Gaussian priors:

$$p(\mathbf{U}) = \prod_{i=1}^{L} \prod_{d=1}^{D} \mathcal{N}(u_{i,d}|m_{i,d}^u, v_{i,d}^u), \tag{3.3}$$

$$p(\mathbf{V}) = \prod_{j=1}^{M} \prod_{d=1}^{D} \mathcal{N}(v_{j,d}|m_{i,d}^v, v_{i,d}^v), \tag{3.4}$$

and

$$p(z) = \mathcal{N}(z|m^z, v^z), \tag{3.5}$$

where $\mathcal{N}(\cdot|m, v)$ denotes the pdf of a Gaussian distribution with mean m and variance v. Given a binary dataset $\mathbf{X}$, the model posterior $p(\mathbf{U}, \mathbf{V}, z|\mathbf{X})$ could be inferred using the SIBM (Stochastic Inference method for Binary Matrices) algorithm [32]. The posterior predictive distributions are used for predicting each binary value in $\mathbf{X}$:

$$p(\widetilde{\mathbf{X}}|\mathbf{X}) = \int p(\widetilde{\mathbf{X}}|\mathbf{U}, \mathbf{V}, z) p(\mathbf{U}, \mathbf{V}, z|\mathbf{X}) d\mathbf{U} d\mathbf{V} dz. \tag{3.6}$$

3.2.3 The Distribution for Data Sampling

Based on the above PMF model, we next describe the distribution for sampling synthetic data. Since the distribution is inferred from a given real-world dataset $\mathbf{X}$, we denote it as

$$p(\widetilde{\mathbf{x}}, y|\mathbf{X}) = p(y)p(\widetilde{\mathbf{x}}|y, \mathbf{X}). \tag{3.7}$$

For $p(y)$, we simply use a balanced class prior, $p(+) = p(-) = 0.5$. This ensures equal chances of getting positive and negative instances when sampling.

Now we describe how we get $p(\widetilde{\mathbf{x}}|y, \mathbf{X})$. We first divide the real-world dataset $\mathbf{X}$ into the positive labeled set $\mathbf{X}_+$ and the set of negative instances, $\mathbf{X}_-$. We infer the model posteriors $p(\mathbf{U}, \mathbf{V}, z|\mathbf{X}_-)$ and $p(\mathbf{U}, \mathbf{V}, z|\mathbf{X}_+)$ separately using the SIBM algorithm.

Notice that these inferred models cannot be directly used to generate infinite samples: suppose the number of users in $\mathbf{X}_+$ (and $\mathbf{X}_-$) is L_+ (and L_-), the posterior predictive (3.6) only gives probabilities for each of these L_+ (L_-) existing users. In other words, the predictive distribution could only be used to reconstruct or predict the original $\mathbf{X}$, as did in the original paper [32].

In order to build a distribution for sampling an infinite number of synthetic users, we employ the following stochastic process: whenever we need to sample a new synthetic instance $\widetilde{\mathbf{x}}$, we first randomly choose a user i from the L_+ (or L_-) existing users, and we then sample the synthetic instance using the posterior predictive of this user. Using this process, the pdf of $p(\widetilde{\mathbf{x}}|y, \mathbf{X})$ is actually

$$p(\widetilde{\mathbf{x}}|y, \mathbf{X}) = \sum_{i=1}^{L_y} p(i|y) \cdot \int p(\widetilde{\mathbf{x}}|\mathbf{U}, \mathbf{V}, z, i)p(\mathbf{U}, \mathbf{V}, z|\mathbf{X}_y)d\mathbf{U}d\mathbf{V}dz, \tag{3.8}$$

where $y \in \{+, -\}$, $p(i|y) = \frac{1}{L_y}$ and

$$p(\widetilde{\mathbf{x}}|\mathbf{U}, \mathbf{V}, z, i) = \prod_{j=1}^{M} p(\widetilde{x}_{i,j}|\mathbf{u}_i, \mathbf{v}_j, z) \tag{3.9}$$

is the likelihood for each instance (existing user) of $\mathbf{X}_y$, which is equivalent to Eq. (3.1). The process of sampling data from $p(\widetilde{\mathbf{x}}, y|\mathbf{X})$ can be implemented by the following pseudo code.

Algorithm 1 Data Sampling from a Binary PMF model

infer $p(\mathbf{U}, \mathbf{V}, z|\mathbf{X}_-)$ and $p(\mathbf{U}, \mathbf{V}, z|\mathbf{X}_+)$ using SIBM;
sample $\mathbf{V}_+, z_+$ from $p(\mathbf{U}, \mathbf{V}, z|\mathbf{X}_+)$;
sample $\mathbf{V}_-, z_-$ from $p(\mathbf{U}, \mathbf{V}, z|\mathbf{X}_-)$;
for each new instance $(\widetilde{\mathbf{x}}, y)$ **do**
  randomly sample y from $\{+, -\}$;
  randomly sample i from $\{1, ..., L_y\}$;
  sample $\mathbf{u}_i$ from $p(\mathbf{U}, \mathbf{V}, z|\mathbf{X}_y)$;
  for j in $1...M$ **do**
    sample $\widetilde{x}_j$ from $Ber(\widetilde{x}_j|\delta(\mathbf{u}_i \cdot \mathbf{v}_{j,y} + z_y))$;
  end for
end for

This algorithm allows us to sample infinite instances from the fixed distribution $p(\widetilde{\mathbf{x}}, y|\mathbf{X})$, to be used for generating data of various sizes. Next, we will discuss how to systematically vary the sparsity of the sampled data.

3.2.4 Missing Data Model

We employ a simple missing data model, to add and vary data sparsity. Our missing data model assumes that each attribute has the same probability to become 0, which follows the Missing Completely At Random (MCAR) assumption [30].

Given the probability of missing s, the missing data model will transform an instance $\widetilde{\mathbf{x}} = (\widetilde{x}_1, \widetilde{x}_2, ..., \widetilde{x}_m)$ to $\mathbf{x} = (x_1, x_2, ..., x_m)$ following:

$$p(\mathbf{x}|\widetilde{\mathbf{x}}, s) = \prod_{j=1}^{m} Ber(\widetilde{x}_j \cdot (1 - s)). \tag{3.10}$$

To ease our illustration, we hereafter call this sparsification process as *Uniform Missing* (UM), and the resultant data as the *sparsified data*. Now if instances are originally sampled from $p(\widetilde{\mathbf{x}}|y)$, the distribution of the sparsified data can be computed by

$$p(\mathbf{x}|y, s) = \int p(\mathbf{x}|\widetilde{\mathbf{x}}, s)p(\widetilde{\mathbf{x}}|y)d\widetilde{\mathbf{x}}.$$

When we apply this sparsification process to data generated from Algorithm 1, we denote the resultant distribution as $p(\mathbf{x}|y, \mathbf{X}, s) = \int p(\mathbf{x}|\widetilde{\mathbf{x}}, s)p(\widetilde{\mathbf{x}}|y, \mathbf{X})d\widetilde{\mathbf{x}}$.

Using the above missing data model, additional sparsity will be introduced uniformly to each attribute and higher s will lead to sparser data. This enables us to vary s and study the impact of data sparsity systematically, as we will describe in the next section.

3.3 Experiment

In this section, we describe how we use the proposed data generation and sparsification process to study our research question with experiments.

Data. We use two movie rating datasets: the movielens 1M dataset[2] (*ml-1m*) and the Yahoo Movies dataset[3] (*ymovie*). As mentioned earlier, we only consider absence/presence of each rating. The preprocessed *ml-1m* dataset is $3,418$ (instances) $\times 3,647$ (attributes) with balanced classes and an original sparsity of 0.9550. The preprocessed *ymovie* dataset is $9,489$ (instances) $\times 4,368$ (attributes) with balanced classes and an original sparsity of 0.9977.

Training. To study the impact of data sparsity, we use various values of s; see the legends of Figure 3.1 and 3.2. For each s, we generate training samples of various sizes l from $p(\mathbf{x}, y|\mathbf{X}, s)$ using the proposed data generation and sparsification process. To solve the linear SVM problem, we use Lib-linear [21] with default parameters to train the classifier and get the corresponding training error $\epsilon_{train}(l, s)$.

Testing. It is empirically impossible to test a classifier on the whole distribution and get the true generalization error. For this reason, we generate a large test dataset from $p(\mathbf{x}, y|\mathbf{X}, s)$ for each setting of s. For the *ymovie* experiment, the size of test sets is 7.59 million instances (800 times of the real data size). For the *ml-1m* experiment, the test size is 0.68 million (200 times of the real data size). We test the classifiers on the corresponding test set and get the test error $\epsilon_{test}(s)$.

For each setting of s and training size l, we repeat the data generation (including sparsification), training and testing process for 10 times, and record the average training and testing error rates. The most time consuming step in our experiment is data generation: the largest training dataset ($l = 2^{20}$ in the *ymovie* experiment) costs us one day on a modern Xeon CPU to generate each time; Meanwhile, training on this particular dataset only costs several minutes, thanks to the efficiency of linear SVM. To speed up the computation, we straight-forwardly parallelize the data generation jobs on a cluster.

As we systematically vary s, the resultant learning curves of averaged training and testing error rates are given in Figure 3.1. Figure 3.2 shows the difference between the two errors in order to show the rate of convergence. We have two important observations which are obvious in both experiments:

Observation 1. *asymptotic generalization error*: Higher sparsity leads to a larger asymptotic generalization error rate;

[2] http://grouplens.org/datasets/movielens/

[3] http://webscope.sandbox.yahoo.com/catalog.php?datatype=r

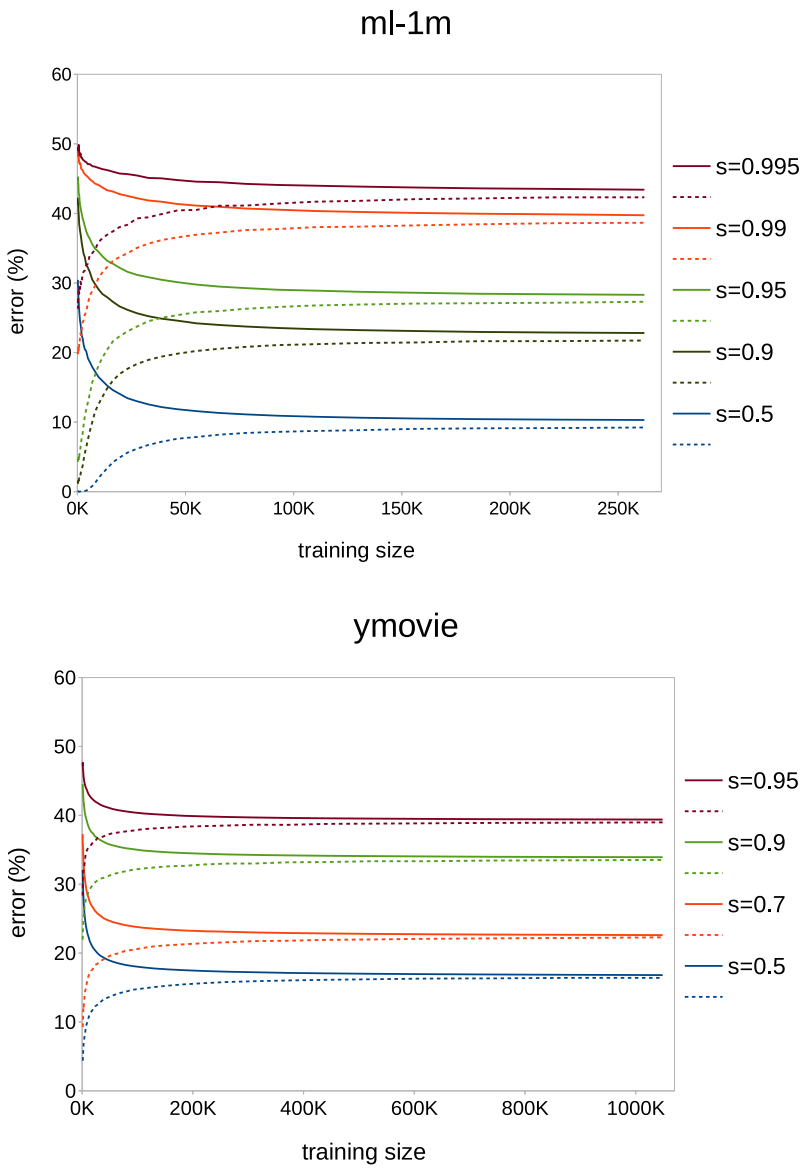


Figure 3.1: Training (dashed) and generalization error rates for different missing data probability s. **Observation:** higher sparsity leads to larger asymptotic generalization error rate.

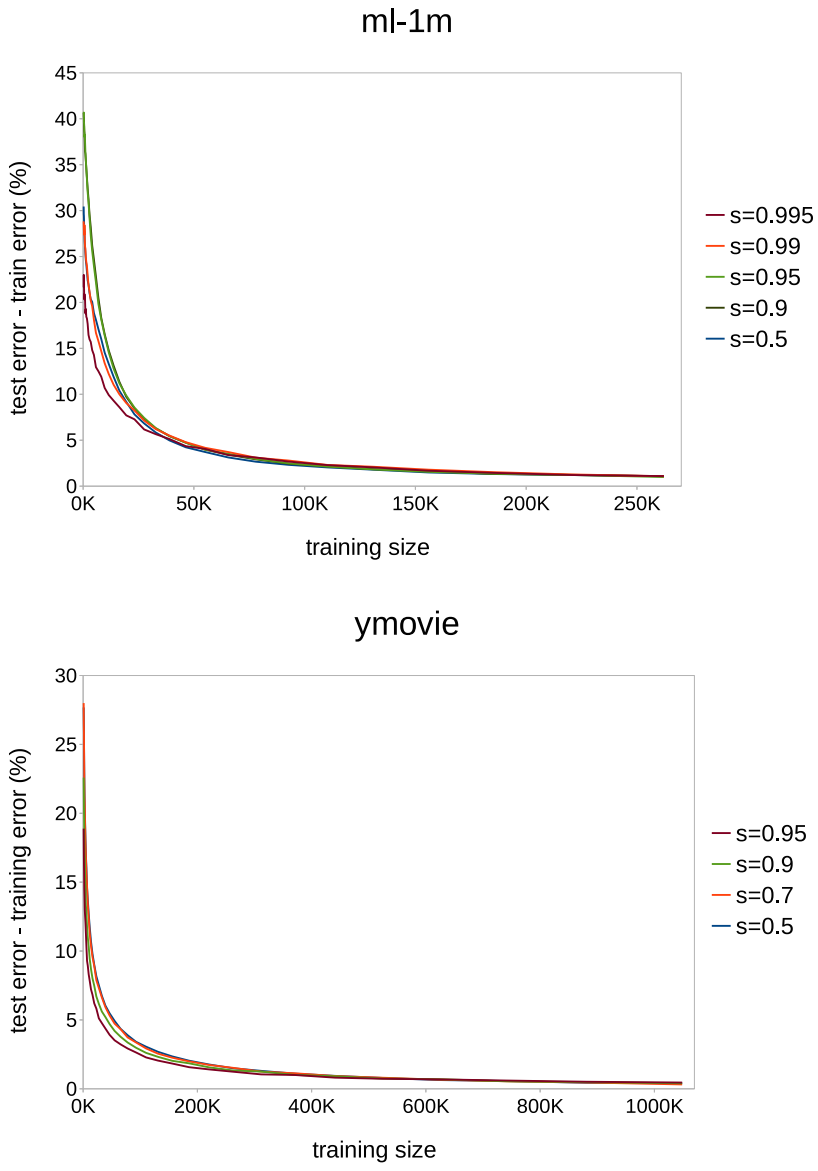


Figure 3.2: The difference between training and generalization error rates for different data missing probability s. **Observation:** asymptotic rate of convergence is almost the same for different sparsity.

For example, in the *ymovie* experiment, the asymptotic (training size $l > 10^6$) generalization error rates are 16.6%, 22.4%, 33.7% and 39.2% for $s = 0.5, 0.7, 0.9$ and 0.95, respectively.

Observation 2. *asymptotic rate of convergence:* the asymptotic rate of convergence is almost the same for different sparsity.

In other words, different sparsity would not change the amount of training data needed to approach convergence. For example, in the *ymovie* experiment, the minimum training size l for the two error rates to be within 1% is almost the same ($l = 370K$), regardless of the value of s.

In the next section, we will study the theoretical reasons behind these observations.

3.4 Theoretical Analysis

In Section 3.4.1, we study the theoretical reason for observation 1, and in Section 3.4.2 we show that the theoretical reason of observation 2 can be easily explained using the PAC bound.

3.4.1 Asymptotic Generalization Error

Suppose the original data distribution is $p(\mathbf{x}|y)$,[4] after applying the missing data model described in Section 3.2.4, we denote the sparsified data distribution as $p(\mathbf{x}|y, s)$. We assume that the class prior is balanced, i.e., $p(+) = p(-) = 0.5$. From the Bayesian Decision Theory [20], we know that the asymptotic generalization error rate of linear SVM can be lower bounded by the Bayes risk, which is the best classification performance that any classifier can achieve over a given data space:

$$R = \int \arg\min_{y'} L_{0\text{-}1}(y', y) p(y'|\mathbf{x}) p(\mathbf{x}) d\mathbf{x} = \int \min_{y'} p(y'|\mathbf{x}) p(\mathbf{x}) d\mathbf{x}, \tag{3.11}$$

where $L_{0\text{-}1}(y', y)$ is the 0/1 loss. We denote the Bayes risk for $p(\mathbf{x}|y, s)$ as $R(s)$, thus the asymptotic generalization error rate is lower bounded by:

$$\lim_{l \to \infty} \epsilon(s) \geq R(s). \tag{3.12}$$

Notice that $\mathbf{x}$ lives in the discrete data space $\{0, 1\}^m$, which allows us to write the Bayes risk as the following form:

$$R(s) = \sum_{\mathbf{x}} p(\mathbf{x}) \cdot \min_{y \in \{+,-\}} p(y|\mathbf{x}, s) = \sum_{\mathbf{x}} \min_{y \in \{+,-\}} p(y, \mathbf{x}|s) = 0.5 \sum_{\mathbf{x}} \min_{y \in \{+,-\}} p(\mathbf{x}|y, s) \tag{3.13}$$

[4] Our proof does not require $p(\mathbf{x}|y)$ to be the specific distribution $p(\widetilde{\mathbf{x}}|y, \mathbf{X})$ introduced earlier.

We will next prove that higher s leads to larger $R(s)$ using three steps. We first consider the case if only one of the attributes is sparsified. In this case, we could prove that the Bayes risk will not decrease by any chance, and with high probability it will increase (Lemma 3.4.1). We next consider the case if we still sparsify only one attribute but with different s; we prove that higher sparsity will lead to larger Bayes risk (Lemma 3.4.2). Based on these results, we finally prove that higher s leads to larger Bayes risk $R(s)$ (Theorem 3.4.3).

When we only sparsify one of the m attributes, x_j, we denote the rest of the attributes as $\mathbf{x}_{-j} = (x_1, ...x_{j-1}, x_{j+1}, ..., x_m)$. Since the order of attributes does not matter, we denote $\mathbf{x}$ as $(\mathbf{x}_{-j}, x_j)$. We denote the corresponding distribution as $p^{(j)}(\mathbf{x}|y, s)$ and Bayes risk as $R^{(j)}(s)$. We now prove:

Lemma 3.4.1 *$R^{(j)}(s) \geq R(0)$ always holds. Specifically, with high probability, $R^{(j)}(s) > R(0)$.*

Proof Since we have only sparsified x_j, we first expand the computation of the Bayes risk in Eq. (3.13) along x_j:

$$R^{(j)}(s) - R(0) = 0.5 \sum_{\mathbf{x}_{-j}} [Z^{(j)}(\mathbf{x}_{-j}, s) - Z(\mathbf{x}_{-j})], \tag{3.14}$$

where $Z(\mathbf{x}_{-j})$ denotes the sum of probability mass of $p(\mathbf{x}|y)$ at $(\mathbf{x}_{-j}, 0)$ and $(\mathbf{x}_{-j}, 1)$, each minimized among classes:

$$Z(\mathbf{x}_{-j}) := \min_{y\in\{+,-\}} p(\mathbf{x}_{-j}, 0|y) + \min_{y\in\{+,-\}} p(\mathbf{x}_{-j}, 1|y). \tag{3.15}$$

$Z^{(j)}(\mathbf{x}_{-j}, s)$ is also defined accordingly for $p^{(j)}(\mathbf{x}|y, s)$. $\sum_{\mathbf{x}_{-j}}$ denotes the summation over all $\mathbf{x}_{-j}$ in the $\{0, 1\}^{m-1}$ space. Now define $\Delta_Z(s) := Z^{(j)}(\mathbf{x}_{-j}, s) - Z(\mathbf{x}_{-j})$, we next prove that $\Delta_Z(s) \geq 0$ holds for all $\mathbf{x}_{-j} \in \{0, 1\}^{m-1}$:

Since x_j is sparsified with s, according to Eq. (3.10), this means $\forall \mathbf{x}_{-j} \in \{0, 1\}^{m-1}$, we have

$$p^{(j)}(\mathbf{x}_{-j}, 0|y, s) = p(\mathbf{x}_{-j}, 0|y) + s \cdot p(\mathbf{x}_{-j}, 1|y) \tag{3.16}$$

and

$$p^{(j)}(\mathbf{x}_{-j}, 1|y, s) = (1 - s) \cdot p(\mathbf{x}_{-j}, 1|y). \tag{3.17}$$

We denote $y_h \in \{-, +\}$ as the label that has a higher probability mass at $(\mathbf{x}_{-j}, 1)$, and we denote the other label as y_l:[5]

[5]Notice that y_h and y_l will change for different $\mathbf{x}_{-j}$. Strictly, we should use the notation $y_h(\mathbf{x}_{-j}, 1)$ and $y_l(\mathbf{x}_{-j}, 1)$ if not for the purpose of brevity.

$$p(\mathbf{x}_{-j}, 1|y_h) \geq p(\mathbf{x}_{-j}, 1|y_l), \tag{3.18}$$

$$\min_{y\in\{+,-\}} p(\mathbf{x}_{-j}, 1|y) = p(\mathbf{x}_{-j}, 1|y_l) \tag{3.19}$$

and then Equations (3.17) and (3.18) lead to:

$$\min_{y\in\{+,-\}} p^{(j)}(\mathbf{x}_{-j}, 1|y, s) = p^{(j)}(\mathbf{x}_{-j}, 1|y_l, s). \tag{3.20}$$

In order to write out $\min_{y\in\{+,-\}} p(\mathbf{x}_{-j}, 0|y)$ and $\min_{y\in\{+,-\}} p^{(j)}(\mathbf{x}_{-j}, 0|y, s)$, we first define

$$g(\mathbf{x}_{-j}) := \frac{p(\mathbf{x}_{-j}, 0|y_l) - p(\mathbf{x}_{-j}, 0|y_h)}{p(\mathbf{x}_{-j}, 1|y_h) - p(\mathbf{x}_{-j}, 1|y_l)}. \tag{3.21}$$

For each $\mathbf{x}_{-j}$ there are in total of three different situations to consider:

$$\textbf{Case 1}. \quad p(\mathbf{x}_{-j}, 0|y_h) \geq p(\mathbf{x}_{-j}, 0|y_l), \tag{3.22}$$

$$\textbf{Case 2}. \quad \begin{cases} p(\mathbf{x}_{-j}, 0|y_h) < p(\mathbf{x}_{-j}, 0|y_l) \\ s \geq g(\mathbf{x}_{-j}) \end{cases}, \tag{3.23}$$

and

$$\textbf{Case 3}. \quad \begin{cases} p(\mathbf{x}_{-j}, 0|y_h) < p(\mathbf{x}_{-j}, 0|y_l) \\ s < g(\mathbf{x}_{-j}) \end{cases}. \tag{3.24}$$

We could straight-forwardly compute $\Delta_Z(s)$ for each case.

Case 1:

$$\Delta_Z(s) = 0, \tag{3.25}$$

Case 2:

$$\Delta_Z(s) = p(\mathbf{x}_{-j}, 0|y_l) - p(\mathbf{x}_{-j}, 0|y_h) > 0 \tag{3.26}$$

and Case 3:

$$\Delta_Z(s) = s \cdot [p(\mathbf{x}_{-j}, 1|y_h) - p(\mathbf{x}_{-j}, 1|y_l)] > 0. \tag{3.27}$$

For brevity, detailed derivation of Equations (3.25), (3.26) and (3.27) is deferred to the appendix.

Now that $\Delta_Z(s) \geq 0$ always holds, $R^{(j)}(s) \geq R(0)$ is true because of Equation (3.14). Moreover, $R^{(j)}(s) = R(0)$ is true only if Case 1 happens for all $\mathbf{x}_{-j} \in \{0, 1\}^{m-1}$, which has low probability. ∎

In the following lemma, we consider the case when we vary the sparsity s on the one sparsified attribute.

Lemma 3.4.2 *$\forall s_1, s_2$ s.t. $1 \geq s_2 > s_1 \geq 0$, we have $R^{(j)}(s_2) \geq R^{(j)}(s_1)$. Specifically:*

(1). only when both s_1 and s_2 are close enough to 1.0, $R^{(j)}(s_2) = R^{(j)}(s_1)$ will be true with high probability.

(2). other wise, $R^{(j)}(s_2) > R^{(j)}(s_1)$.

Its proof could be derived by studying the value of $\Delta_Z(s)$ in different situations. It is straightforward after we have derived Equations (3.25), (3.26) and (3.27) in the proof of Lemma 3.4.1.

Proof We first prove that $\Delta_Z(s_2) \geq \Delta_Z(s_1)$ always holds. For any given $\mathbf{x}_{-j} \in \{0,1\}^{m-1}$:

(1). If Case 1 is true, we will always have $\Delta_Z(s_2) = \Delta_Z(s_1) = 0$, regardless of the value of s_1 and s_2.

(2). If Case 1 is false and $1 \geq s_2 > s_1 \geq g(\mathbf{x}_{-j})$, then Case 2 holds for both s_1 and s_2, and we have:

$$\Delta_Z(s_1) = \Delta_Z(s_2) = p(\mathbf{x}_{-j}, 0|y_l) - p(\mathbf{x}_{-j}, 0|y_h).$$

(3). If Case 1 is false and $s_2 \geq g(\mathbf{x}_{-j}) > s_1$, then Case 2 holds for s_2 and Case 3 holds for s_1. Thus:

$$\Delta_Z(s_2) = p(\mathbf{x}_{-j}, 0|y_l) - p(\mathbf{x}_{-j}, 0|y_h) > s_1 \cdot [p(\mathbf{x}_{-j}, 1|y_h) - p(\mathbf{x}_{-j}, 1|y_l)] = \Delta_Z(s_1).$$

(4). Finally, if Case 1 is false and $g(\mathbf{x}_{-j}) > s_2 > s_1 \geq 0$, then Case 3 holds for both s_1 and s_2. From Eq. (3.27) we know that $\Delta_Z(s_2) > \Delta_Z(s_1)$.

Once we have proved that $\Delta_Z(s_2) \geq \Delta_Z(s_1)$ always hold, substitute into Eq. (3.14) yielding:

$$R^{(j)}(s_2) - R^{(j)}(s_1) = [R^{(j)}(s_2) - R(0)] - [R^{(j)}(s_1) - R(0)] = \sum_{\mathbf{x}_{-j}} [\Delta_Z(s_2) - \Delta_Z(s_1)] \geq 0 \quad (3.28)$$

From the above analysis, we further notice that $R^{(j)}(s_2) = R^{(j)}(s_1)$ only happens if for $\forall \mathbf{x}_{-j} \in \{0,1\}^{m-1}$ either Case 1 happens or Case 2 holds for both s_1 and s_2. Notice for given $p(\mathbf{x}, y)$, it is very unlikely that all $\mathbf{x}_{-j}$ could satisfy Case 1. For those $\mathbf{x}_{-j}$ where Case 1 is not satisfied, Case 2 is required to hold for both s_1 and s_2, which is only likely when both s_1 and s_2 are close enough to 1.0. ∎

The next theorem will generalize to the case when sparsity is introduced to all attributes, which is the goal of our proof:

Theorem 3.4.3 *$\forall s_1, s_2$ s.t. $1 \geq s_2 > s_1 \geq 0$, we have $R(s_2) \geq R(s_1)$. Specifically:*

(1). only when both s_1 and s_2 are close enough to 1.0, $R(s_2) = R(s_1)$ will be true with high probability.

(2). other wise, $R(s_2) > R(s_1)$.

The basic idea of our proof is to find a hypothetical process that varies the sparsification of data from s_2 to s_1, one attribute at a time, so we could leverage the result of lemma 3.4.2.

Proof We use F to denote the total attribute set $\{x_1, ..., x_m\}$. At a certain state, we use $F(s_2) \subset F$ to denote the set of attributes that have been sparsified by s_2; and $F(s_1) \subset F$ the set of attributes that have been sparsified by s_1. We denote the corresponding distribution as $p^{F(s_1),F(s_2)}(\mathbf{x}|y, s_1, s_2)$ and its Bayes risk as $R^{F(s_1),F(s_2)}(s_1, s_2)$.

Now we consider the following process that iteratively changes the elements of $F(s_1)$ and $F(s_2)$: we start from $F(s_1) = \emptyset \wedge F(s_2) = F$, and move each attribute in $F(s_2)$ to $F(s_1)$ one after another. After m such steps we come to $F(s_1) = F \wedge F(s_2) = \emptyset$.

After step i, we denote the current $F(s_1)$ as $F_i(s_1)$, $F(s_2)$ as $F_i(s_2)$. Using this notation, the initial state is $F_0(s_1) = F \wedge F_0(s_2) = \emptyset$, and final state is $F_m(s_1) = \emptyset \wedge F_m(s_1) = F$. It's obvious that

$$R^{F_0(s_1),F_0(s_2)}(s_1, s_2) = R(s_1)$$

and

$$R^{F_m(s_1),F_m(s_1)}(s_1, s_2) = R(s_2).$$

We next prove that for $\forall i \in \{0, ..., m-1\}$:

$$R^{F_{i+1}(s_1),F_{i+1}(s_2)}(s_1, s_2) \geq R^{F_i(s_1),F_i(s_2)}(s_1, s_2). \tag{3.29}$$

Notice that in each step i, there is only one attribute (e.g., x_j) changes from $F_i(s_2)$ to $F_i(s_1)$, i.e., $F - [F_i(s_1) \cup F_{i+1}(s_2)] = \{x_j\}$.

Now consider the distribution $p^{F_i(s_1),F_{i+1}(s_2)}(\mathbf{x}|y, s_1, s_2)$, which corresponds to an intermediate state between step i and $i+1$ with x_j being not sparsified. Now if we sparsify x_j by s_2, we have $p^{F_{i+1}(s_1),F_{i+1}(s_2)}(\mathbf{x}|y, s_1, s_2)$; Instead, if we sparsify x_j by s_1, we have $p^{F_i(s_1),F_i(s_2)}(\mathbf{x}|y, s_1, s_2)$. Using Lemma 3.4.2, it now becomes obvious that Equation (3.29) is true, which further leads to

$$R(s_2) = R^{F_m(s_1),F_m(s_1)}(s_1, s_2) \geq ... \geq R^{F_0(s_1),F_0(s_2)}(s_1, s_2) = R(s_1). \tag{3.30}$$

We further notice that $R(s_2) = R(s_1)$ only holds when all m equal signs hold simultaneously. As illustrated in Lemma 3.4.2, each equal sign holds with high probability only when both s_2 and s_1 are close enough to 1.0. ∎

This theorem tells us that higher sparsity leads to larger Bayes risk, which explains our observation on the asymptotic generalization error. This result is important for understanding how data sparsity affects the asymptotic generalization error of linear SVM.

3.4.2 Asymptotic Rate of Convergence

Our observation on the asymptotic rate of convergence could be explained by the PAC bound [1]. From the PAC bound, we know that the convergence of generalization error rate ϵ and the training error rate ϵ_{tr} is bounded in probability by training size l and the complexity of the function class:

$$Pr\left\{\epsilon > \epsilon_{tr} + \sqrt{\frac{\ln|\mathcal{H}| - \ln\delta}{2l}}\right\} \leq \delta, \tag{3.31}$$

where the VC-dimension $|\mathcal{H}|$ is a measure of the capacity or complexity of the class of functions $\mathcal{H}$ that we consider [88]. For fixed δ, the rate of convergence is approximately:

$$\frac{\partial(\epsilon - \epsilon_{tr})}{\partial l} \approx -\sqrt{\frac{\ln\frac{|\mathcal{H}|}{\delta}}{2l^3}} \tag{3.32}$$

Though different data sparsity could change $|\mathcal{H}|$, for linear SVM, $|\mathcal{H}| < m + 1$ holds true regardless of data sparsity.[6] From Eq. (3.32) we could see that asymptotically (when $2l^3 \gg \ln\frac{m+1}{\delta}$), the rate of convergence is mostly influenced by the increase in training size l rather than by the change of $|\mathcal{H}|$, since $|\mathcal{H}|$ is always upper bounded by $m + 1$. In other words, varying sparsity will have little impact on the asymptotic rate of convergence, which verifies our observation. Notice that VC-dimension can be very large for some non-linear versions of SVM, how their convergence rate will be affected by data sparsity could be an interesting future work.

When using linear SVM in real-world applications, it is important to know whether sparser data would lead to slower convergence rate. If so, practitioners will have to collect more training instances in order for linear SVM to converge on highly sparse datasets. Our observation and analysis shows that the rate of convergence is minimally affected by different data sparsity for linear SVM.

[6] As far as we know, how data sparsity affects the VC-dimension of a problem is still an open problem.

3.5 Conclusion

Linear SVM is efficient for classifying large sparse datasets. In order to understand how data sparsity affects the convergence behavior of linear SVM, we propose a novel approach to generate large and sparse data from real-world datasets using statistical inference and the data sampling process of the PAC framework. From our systematic experiments, we have observed:

1. Higher sparsity will lead to larger asymptotic generalization error rate;

2. The convergence rate of learning is almost unchanged for different sparsity.

We have also mathematically proved these findings. Our experiment and theoretical results are valuable for learning large sparse datasets with linear SVM.

Chapter 4

Convergence Behavior of Naïve Bayes on Large Sparse Data

Classifying large sparse data is an important topic for machine learning and data mining. In the previous chapter, we have studied the convergence behavior of linear SVM on large sparse data. Practically, naïve Bayes is also a popular classification algorithm for large sparse datasets, as its time and space complexity scales linearly with the size of non-missing values. However, several important questions about the behavior of naïve Bayes have not been answered. For example, how different mechanisms of missing data, sparsity rate and the number of attributes systematically affect the learning curves and convergence? In this chapter, we address several common missing data mechanisms and propose novel data generation methods based on these mechanisms. We generate large and sparse data systematically, and study the entire AUC (Area Under ROC Curve) learning curves and the convergence behavior of naïve Bayes. We not only have several important experiment observations, but also provide detailed theoretical studies. Finally, we summarize our empirical and theoretical results as an intuitive decision flowchart and a useful guideline for classifying large sparse datasets in practice.

4.1 Introduction

The naïve Bayes classifier has been reported to have good predictive performance on large and sparse datasets despite its simple model assumption [78, 18, 26, 47, 71]. More importantly, naïve Bayes is popular in practice and its complexity scales linearly with the size of non-missing values only, as analyzed in Junqué de Fortuny et al. [40]. For datasets with a large number of attributes, it is common to assume the existence of an underlying sparse model

structure. However, when evaluated on several large sparse user behavior datasets, naïve Bayes performs even better than discriminative linear classifiers that encourage a sparse model ($l1$-regularization); see Section 4.2. For the above reason, we are motivated to focus our study on naïve Bayes.

In comparing discriminative and generative classifiers, Ng and Jordan [69] have studied the convergence behavior of naïve Bayes generalization error with respect to the number of attributes. They discover that a naïve Bayes classifier needs a training set of size $l = \Omega(\log(m))$ to obtain near-optimal generalization ability. In other words, the convergence rate of learning is a logarithm to m, which is the number of attributes. Despite these important results, previous studies could not address how data sparsity affects the learning behavior of naïve Bayes.

Meanwhile, from an empirical perspective, Junqué de Fortuny et al. [40] have evaluated the learning behavior of naïve Bayes on several large sparse datasets with binary attributes. When continually enlarging the cross-validation datasets, the authors observed that classification AUC keeps growing even at a very large sample size, which in turn supports the argument that *bigger data is better*.

Theoretically, it is quite obvious that more training data usually improve generalization performance, and the learning curve converges as training size approaches infinity. In practice, however, it is still hard to know how sparsity s, the number of attributes m and different missing data mechanisms could affect the learning curves and convergence.

Unfortunately, existing works on classifying *real-world* large sparse datasets using naïve Bayes could not provide us the answer. The main reason is a lack of systematic settings that could address common missing data mechanisms. Besides, m and s of a real-world sparse dataset cannot be easily varied in a systematic manner. Moreover, as demonstrated in Junqué de Fortuny et al. [40], learning curves usually do not converge even at the end for many large-scale user behavior datasets. For asymptotic learning behavior study, synthetic datasets have to be used. We further verify this necessity of using synthetic data in Section 4.2.

In this chapter, we consider two missing data mechanisms that are common for real-world large sparse data. The first mechanism is based on the observation that user behaviors are often diluted across similar items. The other missing data mechanism is based on a Bayesian probabilistic model for the user behavior data. We propose novel data generation approaches for these missing data mechanisms which make it possible to systematically change m and s and study entire naïve Bayes learning curves; see Section 4.3 and 4.5 for details.

In our synthetic data experiments, we use AUC to measure classification performance. Compared to 0/1 loss, AUC is not only proved to be a more discriminating criterion, but also a

common measure for the ranking ability of a classifier [27, 72, 12].

For both missing data mechanisms, we study full AUC learning curves as we systematically vary m and s. The details will be given in Section 4.4 and 4.6. We have several experiment observations that are useful for using naïve Bayes on large sparse datasets. We provide both an intuitive explanation and rigorous proofs for our observations. Specifically, our theoretical study considers the convergence rate and asymptote of AUC risk [76] for naïve Bayes classifiers.

Our empirical and theoretical results can answer practical questions about classifying large sparse data with naïve Bayes, including:

1. How does data sparsity affect *convergence rate* of AUC as data increases?

2. How does data sparsity affect the *asymptote* (upper bound) of AUC?

3. To improve AUC, do we need (a) *more data*, (b) *more attributes* or (c) *to reduce the sparsity of existing data*?

To easily answer these questions in practice, we also provide an intuitive decision flowchart and a guideline in Section 4.8.

The rest of the chapter is organized as follows. In Section 4.2, we describe our results of classifying real-world user behavior data, which demonstrates the effectiveness of naïve Bayes on such data and also explains the necessity of using synthetic data for asymptotic study. In Section 4.3 and 4.5, we describe the missing data mechanisms and the corresponding data generation approaches. In Section 4.4 and 4.6, we describe experiments and several important observations from classifying the generated sparse datasets. Section 4.7 provides detailed theoretical study for our experiment observations. In Section 4.8, we discuss how our results could be used in practice. Section 4.9 reviews other related work and Section 4.10 gives our conclusion.

4.2 Experiments with Real-World Data

To simplify the problem, we consider binary attributes throughout this chapter. For user behavior datasets, a binary value $x_j \in \{0, 1\}$ is a response indicator for whether a certain behavior is observed (or missing). This type of binary data are also denoted as *implicit user feedback* [36, 74], in contrast to explicit feedback signals such as ratings, which are often ordinal or continuous values. It is obvious that implicit feedback data are much easier to collect and thus are more common. In fact, any explicit feedback data can also be treated implicitly. For simplicity of analysis, our study in this chapter only focuses on implicit feedback data (binary attributes).

We will extend our study to real-valued attributes in Chapter 5.

4.2.1 Effectiveness of Naïve Bayes

As demonstrated in Junqué de Fortuny et al. [40], each attribute of a user behavior dataset often contains non-negligible information. Attribute selection is not optimal for classifying such large sparse data. Following this observation, here we further demonstrate that a standard naïve Bayes classifier which takes all attributes into account can have better performance comparing to a linear classifier [1] that encourages a sparse model structure (l1-regularization) using experiments on several large sparse user behavior data (Table 1.1). Besides, we also find naïve Bayes is more time-efficient as it does not need the extra time for tuning the best regularization parameter, see Figure 4.1. This set of experiments indicate that naïve Bayes is effective and efficient on large sparse user behavior data.

4.2.2 Learning Behavior Study – Inadequacy of Using Real-world Data

In order to systematically study how sparsity s and number of attributes m affect the naïve Bayes learning behavior, we first experiment with several real-world large sparse datasets. 5-fold cross-validation AUC are recorded as we gradually increase the data size to plot the learning curves. We use a naïve way to vary m by selecting subsets of attributes at random. To change data sparsity s, we simply impose a uniform probability of missing $\theta \in (0, 1)$ on each attribute, which is also known as *blankout noise* in other literature [58, 11, 90]. Using these simple strategies, we vary m and keep s fixed (Figure 4.2) and then change m but keep s fixed (Figure 4.3).

As can be seen from experiment results, adding sparsity s and reducing the number of attributes m always lead to a lower classification AUC. However, we cannot tell how s and m affect the *when* (convergence rate) and *where* (upper bound) the AUC learning curve converge. Besides, we cannot tell the difference between reducing sparsity and adding attributes. These experiment observations indicate that real-world user behavior datasets are not suitable for convergence (asymptotic) study, because of the limited data size and also because a lack of systematic manner to change m and s.

In order to answer the three research questions proposed in Section 4.1, we have to use synthetic sparse datasets generated with missing data mechanisms that resemble real-world data.

[1] we do not consider kernel classifiers since its time complexity is too high especially when the number of attributes and instances are huge.

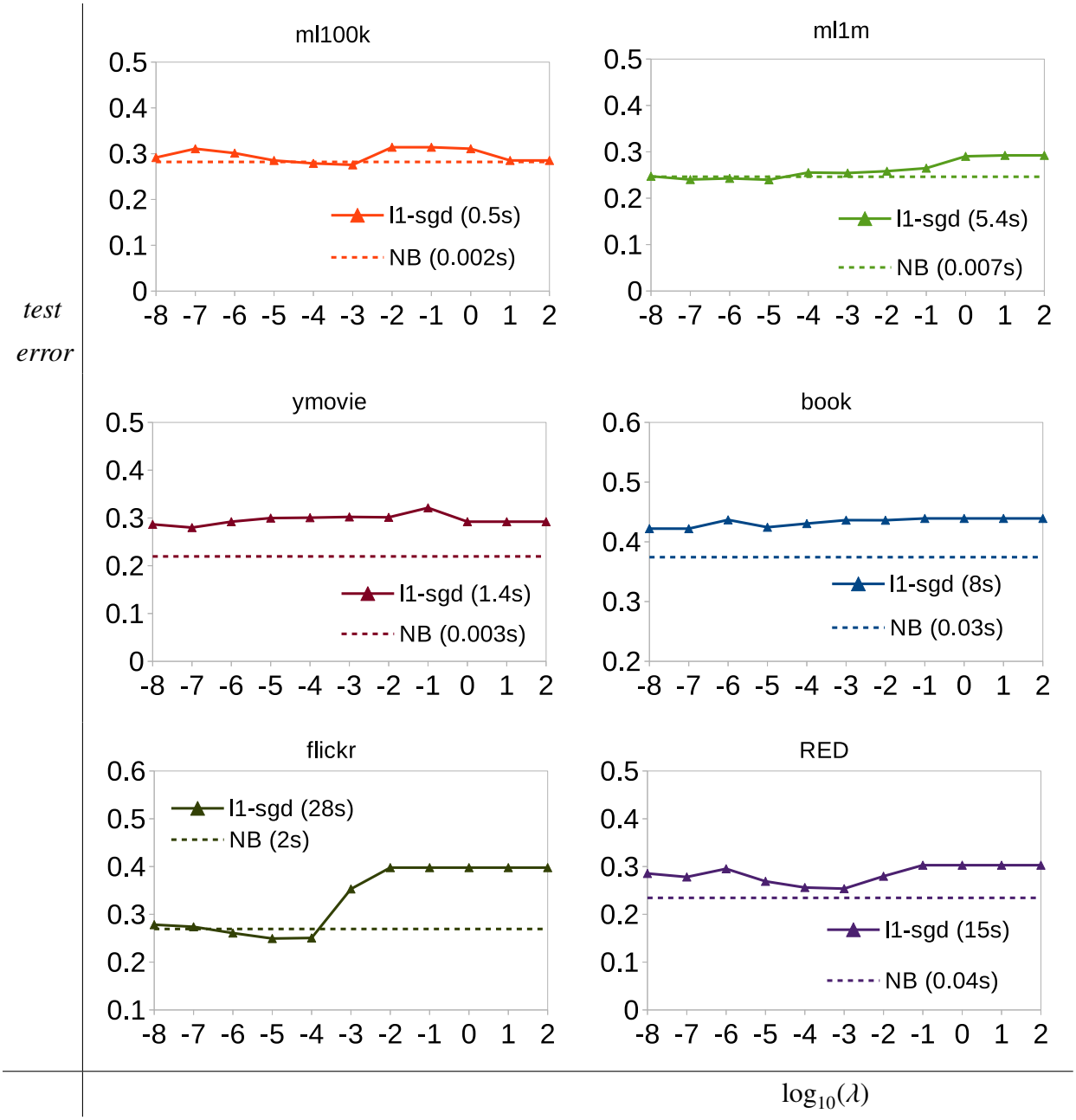


Figure 4.1: Classification error and CPU time comparison between Naïve Bayes (NB) and $l1$-regularized linear classifier ($l1$-sgd) on different user behavior datasets. The l1-regularized linear classifier is optimized with Stochastic Gradient Descent with 10^7 gradient updates. The x-axis is the logarithm scale of the $l1$-sgd regularization parameter λ. *Observations: after careful parameter tuning (significantly more CPU time), l1-sgd outperforms NB on* 50% *of the datasets; However, NB gives lower error in most of the settings.*

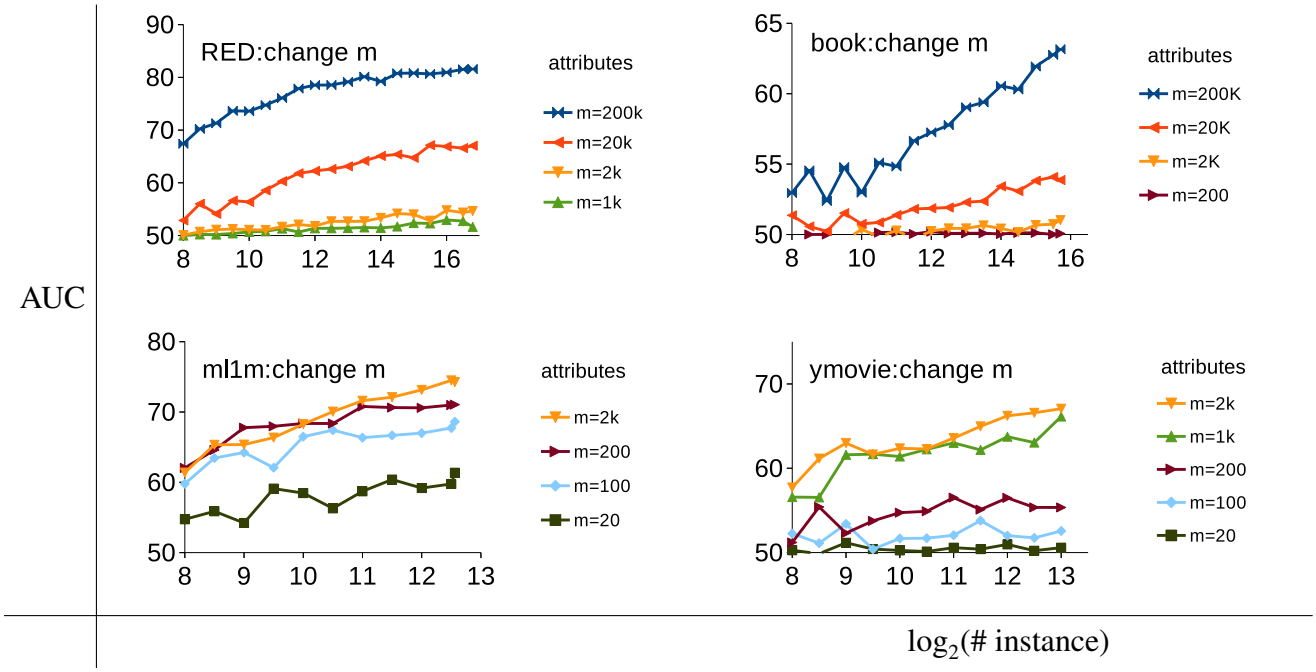


Figure 4.2: Naïve Bayes AUC learning curves on real-world user behavior datasets as we systematically vary *m*. *Observation: More attributes is always better, but we cannot see the convergence behavior.*

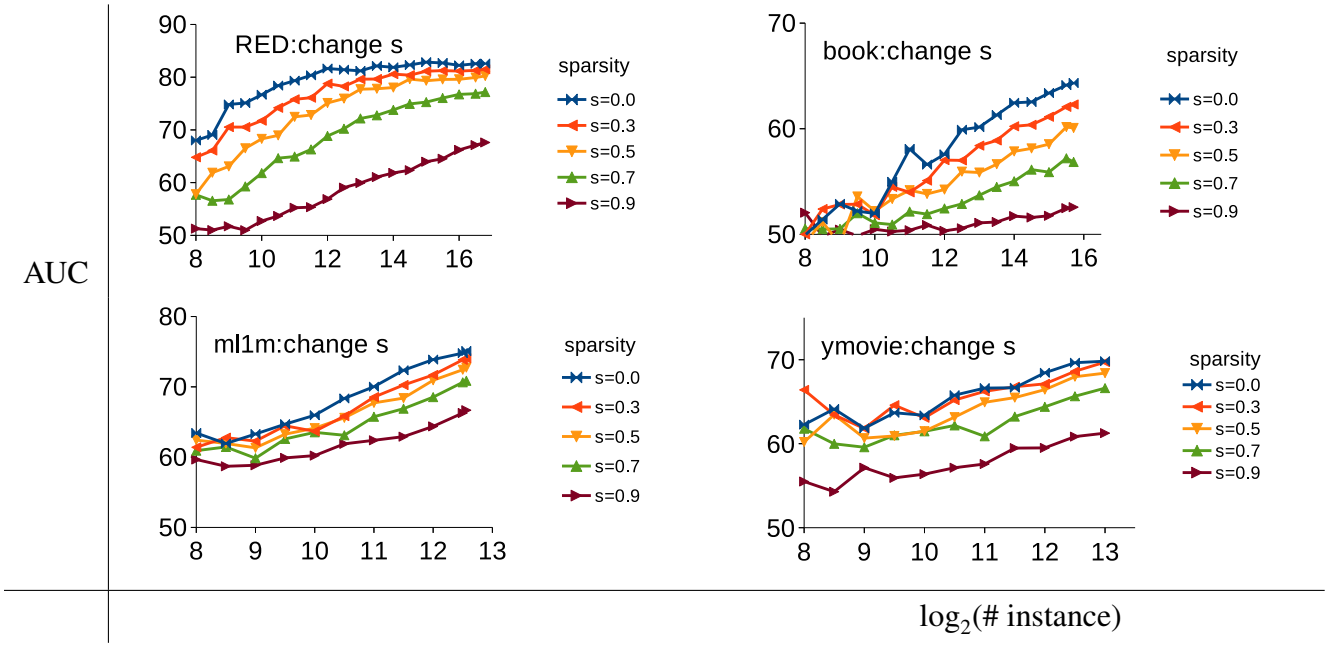


Figure 4.3: Naïve Bayes AUC learning curves on real-world user behavior datasets as we systematically vary *s*. *Observation: Lower sparsity is better, but we cannot see the convergence behavior.*

According to the missing data theory [56, 77], missing data mechanisms could be broadly classified into three types: (1). MCAR (Missing Completely At Random), in which the probability of missing does not depend on the values of observed as well as the unobserved data; (2). MAR (Missing At Random), in which probability of missing does not depend on values of unobserved data, but depends on observed data; and (3). MNAR (Missing Not At Random), where probability of missing does depend on values of unobserved data themselves. Real-world user behavior data with explicit feedback are MNAR, because the missing likelihood of a data entry often depends on its explicit value such as rating [60, 31]. However, since our study considers data of binary attributes (implicit feedback), we should only use Missing At Random mechanisms to model the data.

Starting from the next section, we describe two typical MAR missing data mechanisms for real-world user behavior data. The first mechanism is based on observations that many items (attributes) of a user behavior dataset are similar and information is often uniformly diluted among similar attributes; the second missing data mechanism is actually learned from the probabilistic model of real-world user behavior datasets.

4.3 Missing Data Mechanism 1: Uniform Dilution

In this section, we describe several observations about the missing patterns of large sparse user behavior data. Inspired by these observations, we propose the *Uniform Dilution* (UD) missing data mechanism, which belongs to MAR. We then describe two concrete data generation methods (*Bernoulli-trial Expansion* and *Just-1 Expansion*) for UD.

4.3.1 Observations on Sparse User Behavior Data

Given a user behavior dataset, to find out the exact missing data mechanisms among a large number of users and items is not an easy task. However, the reason for sparsity is clear – the number of available items is huge but the behavioral budget of each user is quite limited. To find out its missing data mechanism then amounts to finding out how users distribute their behavior budgets over the items. Fortunately, the large number of items are not all distinct. Many items are similar, e.g., different episodes of a TV opera, books of a very similar topic and writing style. Here we simply denote a group of similar attributes (items) as a *category*. It is reasonable to assume that such categories implicitly exist in real-world user behavior datasets and items within a category are similar enough so that user behaviors (information) are distributed almost uniformly among them.

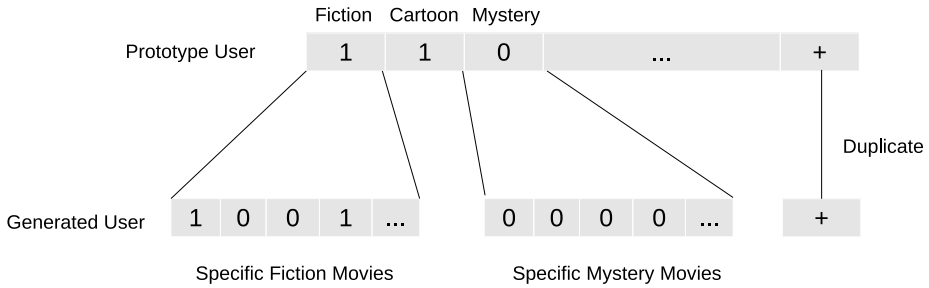


Figure 4.4: The user-movie example of Uniform Dilution. Each data entry is either 1 (watched) or 0 (not-watched). The label is a binary-valued user feature, such as the gender. For illustration purpose, we here use different genres (i.e., *Fiction*, *Cartoon*) to denote non-overlapping attribute categories. But in practice, attribute categories are mostly implicit.

Based on this typical observation, we propose an MAR missing data mechanism, *Uniform Dilution*. We describe it with an example in Figure 4.4. For a real user behavior dataset about users and movies, we assume that there exists an underlying *Prototype* dataset, which contains data instances about categories of movies watched by each user. Each of its attributes represents a movie category instead of each specific movie, and the value is still either *watched* (1) or *not-watched* (0).[2] The Uniform Dilution process assumes that user behaviors on specific items is actually diluted from the underlying prototype with a uniform probability. Notice that UD is MAR since the missingness does not depend on actual data value but does depend on the information in the prototype dataset.

This UD process is reasonable and practical. On the one hand, it is unavoidable to make assumptions about real data in order to generate synthetic data. On the other hand, compared to a real user behavior data, the assumption made by Uniform Dilution only ignores fine-grained differences among attributes within a category.

Given a prototype dataset, we can generate an arbitrary number of i.i.d. (independently and identically distributed) samples following UD process. We next describe two concrete UD processes for binary data which allow us to generate large sparse data and change m and s systematically.

[2]In practice, the underlying item categories are data-dependent and may not explicitly map to any known item labels such as movie genres.

4.3.2 Bernoulli-trial Expansion

The first UD process is *Bernoulli-trial Expansion* (BTE). In BTE, if a prototype attribute is non-missing ($x_j = 1$), then each expanded attribute $x_{j,k}(k \in \{1, 2, ..., t\})$ has uniform chance $\theta \in [0, 1]$ to be non-missing. Its intuition is that information across a category of items should be dispersed uniformly if without further prior knowledge. For example, if we know a user has watched horror movies, without further knowledge we should assume the chance that she/he watched each horror movie to be the same:

$$x_{j,k} = \mathbb{I}(x_j)\mathbf{Ber}(\theta). \tag{4.1}$$

In this setting, suppose a prototype dataset has m_0 attributes and its sparsity is s_0, with expansion rate t, the number of generated attributes m and the expectation of data sparsity s is (when infinite data are generated):

$$m = m_0 t \tag{4.2}$$

and

$$\mathbb{E}(s) = 1 - (1 - s_0)\theta. \tag{4.3}$$

In this respect, we could change the value of t and θ in order to vary m and s systematically.

4.3.3 Just-1 Expansion

We consider the other typical uniform dilution process for user behavior datasets. In this process called *Just-1 Expansion* (JE), we assume the information contained in one attribute is only retained in one of the expanded attributes, and the others are missing:

$$x_{j,k} = \mathbb{I}(x_j), \text{ if } k = \mathbf{Cat}(t), \tag{4.4}$$

and

$$x_{j,k} = 0, \text{ otherwise}, \tag{4.5}$$

where $\mathbf{Cat}(t)$ is the t-way categorical distribution with identical probability $\frac{1}{t}$. This setting is simplified from the observation that a user could only have a certain number of actions over a given group of items, no matter how large the group is. For example, Amazon users usually purchase at most one product of a certain type, e.g., a TV.

Again, we do not introduce further prior knowledge within each category of attributes but assume each fine-grained attribute has identical chance of getting picked. Using this just-1

expansion method, both m and s are changed simultaneously as we change the expansion rate t:

$$m = m_0 t \tag{4.6}$$

and

$$\mathbb{E}(s) = 1 - \frac{1 - s_0}{t}. \tag{4.7}$$

Both BTE and JE use the MAR assumption and are suitable for modeling the missing data mechanism of sparse binary user behavior data. With either BTE and JE, we are now able to generate an arbitrarily large dataset, and change m and s systematically to study the learning behavior of a classifier.

4.4 Experiment with the Uniform Dilution Approach

In this section, we describe experiments with the Uniform Dilution approach. To generate large and sparse synthetic data with UD , we start from relatively smaller and denser prototype datasets. We use two datasets from the UCI data repository[3] as the prototypes. These two datasets are popular benchmark datasets, and they are relatively small and dense thus suitable for expansion and dilution. We convert all attributes and labels into binary ones. Table 4.1 shows the properties of the prototype datasets after preprocessing.

Table 4.1: Prototype datasets used for expansion

Prototype	Attr.	class +/-	Sparsity	AUC
(1).Breast Cancer Wisconsin	9	241/458	0.6865	0.9865
(2).Abalone	7	2081/2096	0.5081	0.7988

We conduct a series of experiments under BTE and JE with a wide range of m and s, by tuning parameters t and θ of Eq. (4.1) and (4.4). Our experiments use 5 fold cross validation AUC to measure classification performance. For different settings of m and s, we always generate a synthetic dataset large enough to see the learning curve convergence. The sample sizes are chosen from $l = 2^i, i \in \{0.25, 0.5, 0.75, ...\}$, until we see obvious learning curve convergence. We run the data expansion, generation and cross validation jobs in parallel on a shared cluster. The largest synthetic dataset (10.3M instances and 90K attributes) takes 20 hours on a 2.4 GHz core with 32 GB allocated job memory.

[3] http://archive.ics.uci.edu/ml/

Here we present our experiments and observations about how sparsity and the number of attributes affect the learning behavior, a detailed theoretical study of these observations will be given in Section 4.7.

4.4.1 Learning Curve Behaviors of Bernoulli-trial Expansion

When using BTE, we are able to vary m and s independently. The values of m and s are deliberately chosen to cover a wide range, from a small m and s to the level of real-world user-item data. When varying one parameter, the other is fixed at a large or sparse setting similar with the datasets used in Junqué de Fortuny et al. [40]. Specifically, we set m as $70K$ and $90K$ respectively (expansion rate $t = 10K$) for each prototype dataset when varying s (Figure 4.5). When changing m systematically, we set s to be 0.9997 (approximately) so that the data stays sparse (Figure 4.6).

From Figure 4.5, we find that when m is fixed, learning curves of higher sparsity converge at a 'later' time (more instances are needed to see convergence). This agrees with the intuition that sparser data leads to inferior classification performance. Meanwhile, sparser data tend to have lower upper bounds. However, this observation is only obvious at an extremely high s.

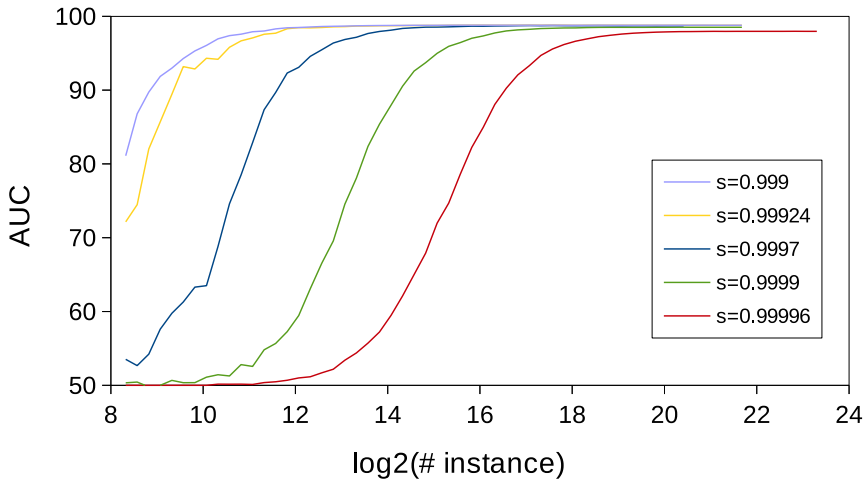
From Figure 4.6, we see that when sparsity s is fixed, more attributes results in higher AUC upper bound. To sum up, we have the following observation about the learning curve behavior under BTE:

Observation 1. *For large and sparse datasets generated by BTE, higher sparsity leads to lower AUC convergence rate and upper bound; less attributes leads to lower AUC upper bound.*

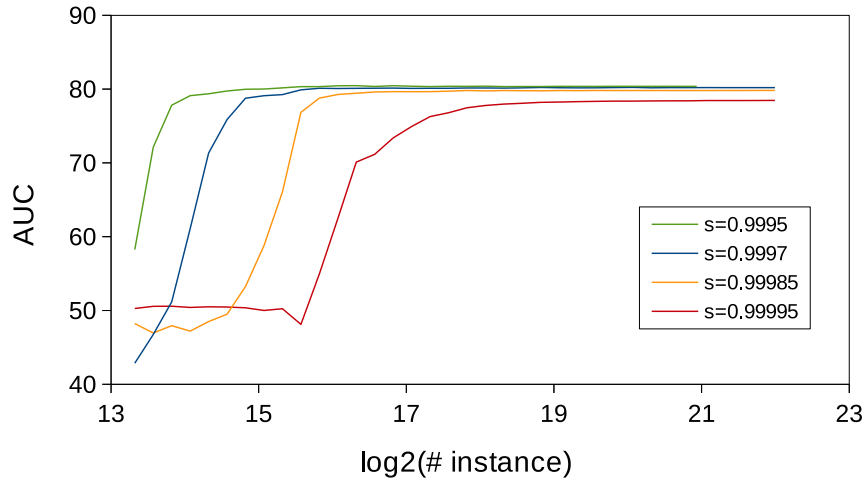
A detailed theoretical study of this observation is given in Section 4.7 (Lemmas 4.7.1 and 4.7.2). In Section 4.8, we also give a decision flowchart and a guideline on how to apply this empirical finding in practice.

4.4.2 Learning Curve Behaviors of Just-1 Expansion

When using JE, m and s change simultaneously as we vary expansion rate t. Larger t results in more attributes and higher sparsity (Figure 4.7). We find that the AUC upper bound is not affected by different t's. Meanwhile, we see that learning curves of smaller expansion rate converge faster, which is quite intuitive. We can imagine that when expansion rate is extremely large, a huge number of data instances will be needed to see the convergence. These behaviors can be summarized as the following observation:

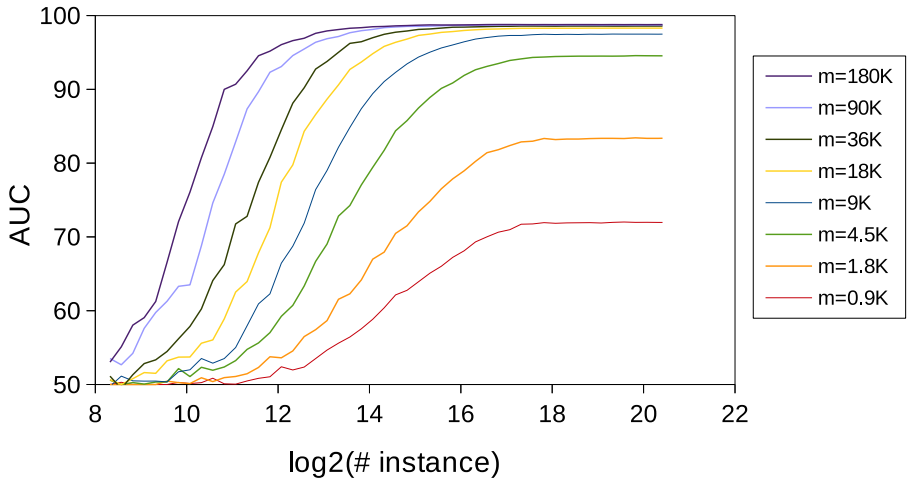


Prototype 1, BTE, m=90K

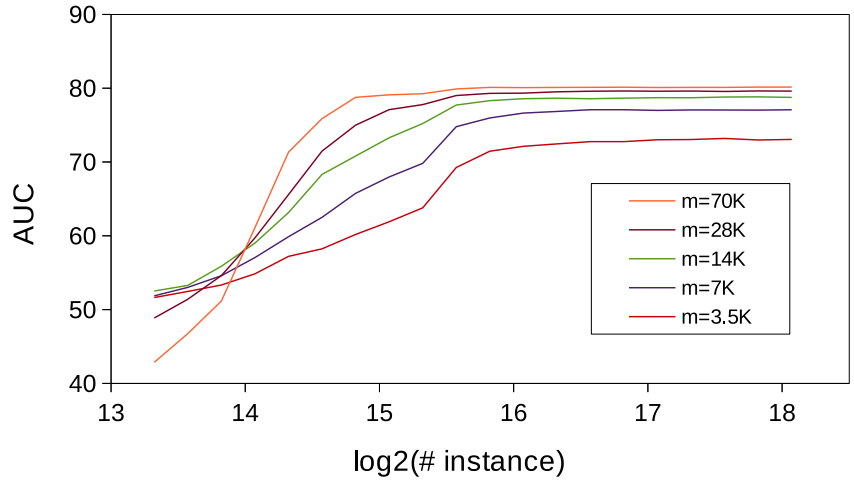


Prototype 2, BTE, m=70K

Figure 4.5: Learning curves of Bernoulli-trial expansion with fixed m and different s. To ensure that we get smooth and stable learning curves, all our experiments on synthetic data uses sampling sizes of $l = 2^i, i \in \{8, 8.25, 8.5, 8.75, \ldots\}$. Moreover, data sampling and 5-fold cross validation is exhaustively repeated 50 times for each value of l. *Observation: higher sparsity leads to slower convergence and lower AUC upper bound.*

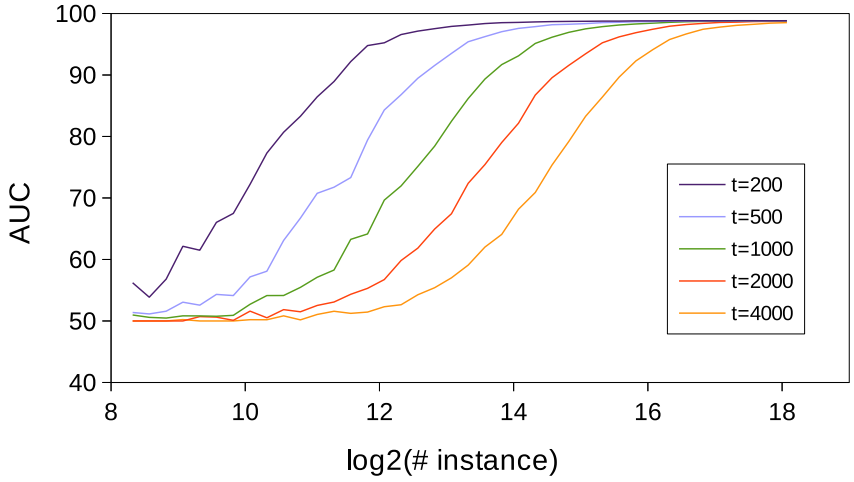


Prototype 1, BTE, s=0.9997

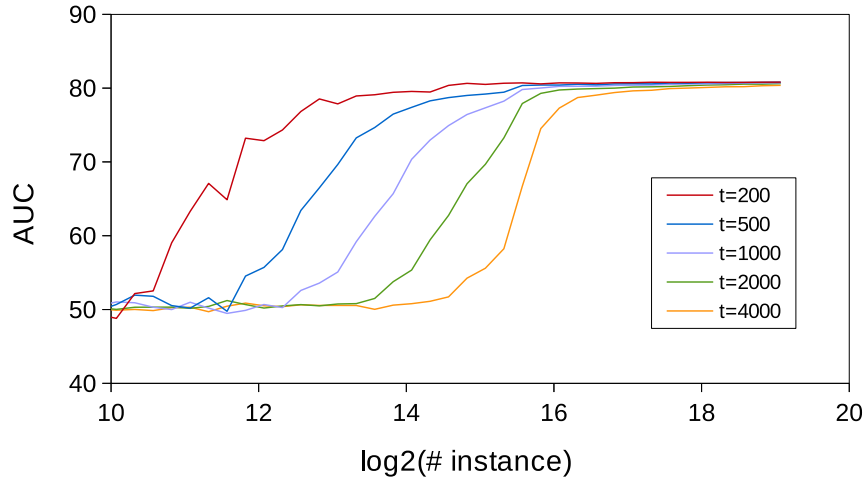


Prototype 2, BTE, s=0.9997

Figure 4.6: Learning curves of Bernoulli-trial expansion with fixed s and different m. Observation: *more attributes leads to higher AUC upper bound.*



Prototype 1, JE



Prototype 2, JE

Figure 4.7: Learning curves of just-1 expansion with different expansion rate t. The number of attributes and sparsity are not displayed in the legend of the figures, but can be computed easily from Eq. (4.7). *Observation: larger t leads to lower convergence rate; however, different t does not change the AUC upper bound.*

Observation 2. *For large and sparse datasets that are generated by JE, larger t makes the AUC learning curve converge slower. However, changing t does not change AUC upper bound.*

A detailed theoretical study of this observation will be given in Section 4.7 (Lemmas 4.7.1 and 4.7.4). For practical classification problems, an intuitive decision flowchart on how this observation can help is also given in Section 4.8.

4.4.3 Comparing BTE and JE

When comparing the AUC upper bounds of different expansion methods and the corresponding prototype datasets in Table 4.1, we have the following comparative conclusions:

Comparison 1. *For BTE, the AUC upper bound could be lower than the AUC of the prototype data; for JE, the AUC upper bound always equals to the AUC of the prototype data.*

This contrast contains no surprise. In BTE, it is possible all the attributes $x_{j,k}$ expanded from $x_j = 1$ become 0 in some data instances, which means the information of $x_j = 1$ could be completely lost. In JE, such cases are avoided. We believe this might be the main reason for Comparison 1. Moreover, the curves in Figure 4.5 and 4.6 imply that we can not completely compensate this probabilistic loss of information in BTE even when enough data instances are used. A detailed study is given Lemma 4.7.2 in Section 4.7.2.

On the one hand, it is intuitive that sparser data may always lead to inferior classification performance; on the other hand, m may play almost contradictory roles in different missing data mechanisms. Consider a prototype attribute x_j that equals to 1. When using JE, this information ($x_j = 1$) is diluted across t attributes. The more attributes we have, the harder for a naïve Bayes classifier to "aggregate" this diluted information. We provide the detailed study for this effect in Lemma 4.7.1 of Section 4.7.2. For BTE, however, the same information emerges with a fixed probability. The more attributes we have the more information a classifier will get; See Lemma 4.7.3 in Section 4.7.3. We can also verify it from the experiment learning curves:

Comparison 2. *For both Uniform Dilution mechanisms, higher sparsity always leads to slower learning curve convergence rate. More attributes lead to lower convergence rate for JE, but higher AUC upper bound for BTE.*

In Section 4.8, we show how the above observations of Uniform Dilution experiments can provide practical guidance to classifying large sparse datasets with naïve Bayes. Besides, we also provide an intuitive decision flowchart (Figure 4.10) for readers to easily apply our results in solving real-world classification problem.

4.5 Missing Data Mechanism 2: Probabilistic Modeling

In previous sections, we consider the missing data mechanism inferred from domain knowledge (observation) of user behavior data. We now take a probabilistic modeling approach in interpreting the generative story of the user behavior dataset and its missingness. We use a Probabilistic Matrix Factorization model [67] to decode the generative process and missing data mechanism of a dataset. Moreover, we propose the corresponding data generation method which essentially samples synthetic data instances from the inferred PMF model posterior.

For binary attributes, [32] have proposed a Bayesian PMF model which has good predictive performance on several real datasets. Given a sparse binary matrix $\mathbf{X}$, the posterior of their model is:

$$p(\mathbf{U}, \mathbf{V}, z|\mathbf{X}) = \frac{p(\mathbf{X}|\mathbf{U}, \mathbf{V}, z)p(\mathbf{U})p(\mathbf{V})p(z)}{p(\mathbf{X})}, \tag{4.8}$$

where $p(\mathbf{X})$ is a constant and $p(\mathbf{U})$, $p(\mathbf{V})$ are fully factorized Gaussian priors for the two low-rank matrices, $z \in \mathbb{R}$ is the constant that controls global sparsity. The data generation process can be modeled as:

$$p(\mathbf{X}|\mathbf{U}, \mathbf{V}, z) = \prod_{(i,j)} p(x_{i,j}|\mathbf{u}_i, \mathbf{v}_j, z) = \prod_{(i,j)} \text{Ber}(x_{i,j}|\sigma(\mathbf{u}_i\mathbf{v}_j^T + z)), \tag{4.9}$$

where $\sigma(\cdot)$ denotes the logistic function to squash a real number into $[0, 1]$.

For data generation, we assume that data are sampled according to the PMF model posterior, and sparsity can be changed by choosing a different prior for z. We denote this data generation methods as G_{PMF} (Generation using PMF), to distinguish from attribute dilution methods such as JE and BTE. Notice that G_{PMF} also uses the MAR assumption since the missing likelihood does not depend on the explicit value of each data entry.

To compute the model posterior, we use the efficient inference algorithm proposed in Hernandez et al. [32]. After Bayesian inference, we could then generate an arbitrarily large dataset using the model posterior. In order to sample an infinite number of synthetic users, we employ the stochastic process as discussed in Section 3.2.3, i.e., whenever we need to generate a new synthetic user, we first randomly choose a user $\mathbf{X}_i$ from the users (rows) that exist in $\mathbf{X}$ as a prototype user. We then generate a new user using model posterior of this user prototype, i.e., $\mathbf{u}_i$ along with posteriors of other parameters ($\mathbf{V}$ and z). This approach will be further discussed in Section 5.4.2 as we extend our study to real-valued data.

Using this method, the generated data can well approximate the original data in a distributional sense. Besides, in Section 4.6, we will also verify that classifying the generated data could 'replicate' the naïve Bayes learning curve of classifying the original data in our experiments.

4.6 Experiment with the Probabilistic Modeling Approach

To evaluate the probabilistic modeling data generation approach G_{PMF} , we choose Yahoo movie data (*ymovie*) and one of the Movie lens data (*ml-1m*) where attributes refer to movies and each data instance corresponds to a user's ratings on those movies. If a user i has rated on movie j, then $\mathbf{X}_{i,j} = 1$, otherwise 0. The class label is the gender of each user. There are $2,184$ females and $5,436$ males in the original *ymovie* data. To eliminate class skewness, we randomly select $2,184$ instances from both classes. We use $9,489$ out of the original $11,916$ movies which are rated at least once by those $4,368$ users. The overall sparsity for the preprocessed *ymovie* data is 0.9977. The same preprocess is used for *ml-1m* dataset (originally $6,040 \times 3,952$). The preprocessed *ml-1m* dataset is $3,418 \times 3,647$ with balanced classes and an overall sparsity of 0.9550. For the preprocessed *ymovie* and *ml-1m*, we use the inference algorithm proposed in Hernandez et al. [32] to estimate the posteriors of the latent variables $(\mathbf{U}, \mathbf{V}, z)$ separately for data of each class label.

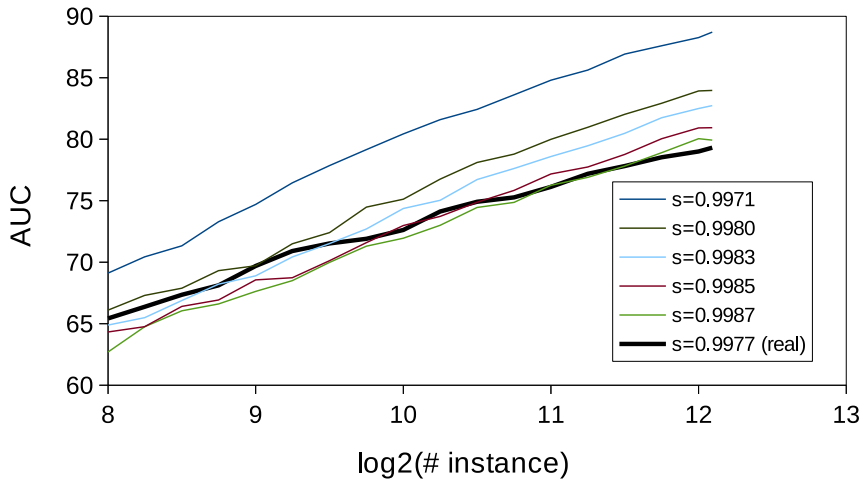
4.6.1 Replicating Learning Curves with G_{PMF}

Hernandez et al. [32] have already evaluated the performance of their PMF model using both synthetic and real-world data experiments. After removing a set of 1 entries from the training data, their evaluation task is to find out the location of those removed entries. Here we further evaluate how well we can reconstruct the whole binary data matrix using only the estimated model posterior in terms of naïve Bayes learning behavior. To be specific, we focus on whether we could 'replicate' the AUC learning curve behavior of the real data using generated data of the same size.

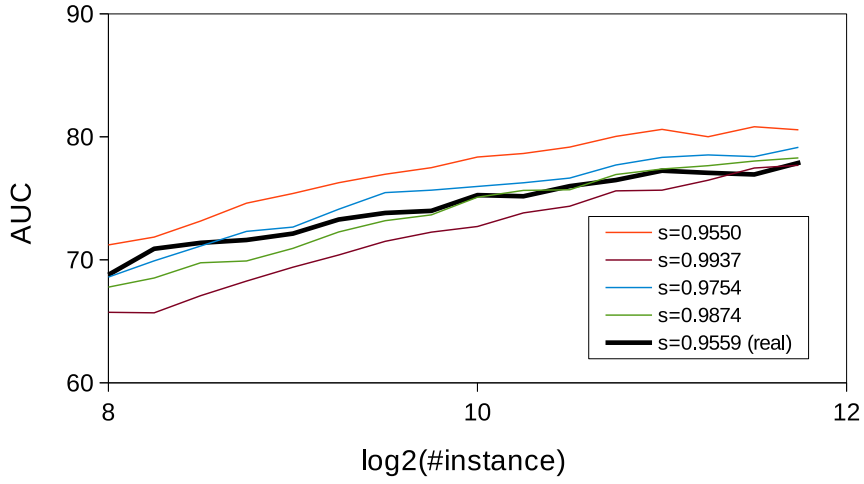
To do so, we sample a synthetic data matrix according to the size of *ymovie* and *ml-1m* from the posterior distributions of each corresponding real dataset. The global sparsity of the synthetic data matrices are very close to the corresponding real data. Table 4.2 shows a comparison of sparsity between real and reconstructed datasets with different z biases.

Using these datasets, we test the 5-fold cross-validation AUC of naïve Bayes with different sample sizes. To be consistent with experiments in Section 4.4, sample sizes are also chosen from the series $\{2^i\}$ with the increment of i being 0.25. The result is shown in Figure 4.8. In the figure, each AUC value is the average of 50 runs of repeated random sampling. We see that the learning curve behavior is similar for real data and synthetic data: they all increase log-linearly with the number of data instances, though with slightly different slopes.

The result of this experiment implies that G_{PMF} provides a reasonable synthetic setting for



ymovie



ml-1m

Figure 4.8: Comparison of the learning curves on real datasets (the thick lines) and the generated data. *Observation: the learning curve behavior is similar for real data and synthetic data.*

Table 4.2: Datasets for G_{PMF} experiment. Numbers in the parentheses are the biases added to the posterior mean of z.

data	sparsity	data	sparsity
ymovie-*real*	0.9977	ml1m-*real*	0.9559
ymovie-*synth*(0)	0.9971	ml1m-*synth*(0)	0.9550
ymovie-*synth*(-0.5)	0.9980	ml1m-*synth*(-1)	0.9754
ymovie-*synth*(-0.7)	0.9983	ml1m-*synth*(-2)	0.9874
ymovie-*synth*(-0.9)	0.9985		
ymovie-*synth*(-1.1)	0.9987		

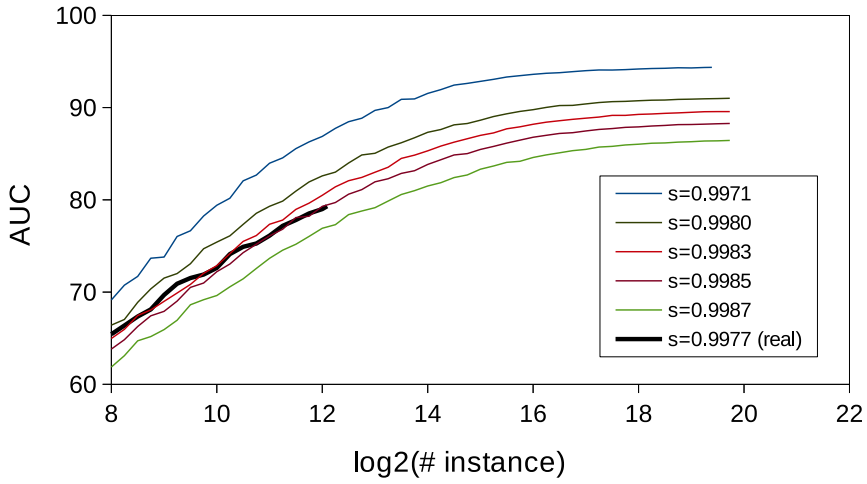
studying naïve Bayes learning curve behavior on sparse datasets.

4.6.2 Data Generation Experiment

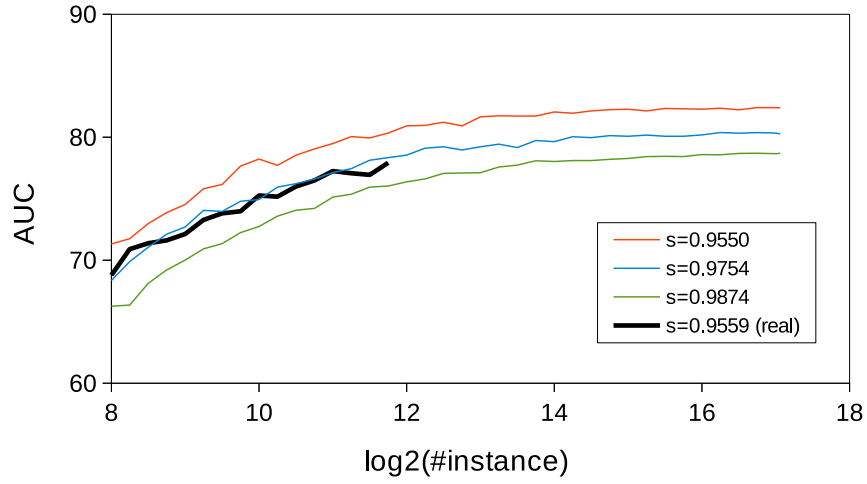
In order to see the entire learning curves, we have sampled more than 0.8 million synthetic data instances from the inferred the PMF model of *ymovie* and > 0.13 million for *ml-1m*. We vary the data sparsity s by changing the PMF parameter z systematically. Just as the settings of previous experiments, we record the 5 fold cross-validation AUC over different sample sizes.

We also speedup the experiments by running the data expansion, sampling and cross validation jobs in parallel on a shared cluster. To speed up the data generation process, we split the generation of a l-by-m data matrix vertically by a factor of f ($f = 10$ for *ymovie* and $f = 8$ for *ml-1m*), i.e., we generate l/f instances in each single job. The parameters $\mathbf{V}$ and z are sampled before data generation and shared across jobs of each specific setting (u_i have to be sampled on-the-fly for each new instance). We exhaustively run random sampling for 50 repeats for all the PMF experiments. We also split the random sampling workload into 10 parallelized jobs, each for 5 repeated random subsampling. Naïve Bayes cross-validation is performed right after the generation of each subsample. Using this strategy, the slowest data generation job takes no more than 8K seconds and the slowest random subsampling and cross-validation job takes no more than 6K seconds (both correspond to the largest expansion experiment of *ymovie* with no negative bias of z).

Figure 4.9 shows the learning curves of naïve Bayes on the genereated data. We see that the curves increase log-linearly at the beginning, but gradually tend to converge. Since the G_{PMF} experiment does not change sparsity s significantly compared to the BTE and JE experiments, the rate of learning curve convergence does not differ much for different s. However, we have the following observation about the learning curve upper bound:



ymovie



ml-1m

Figure 4.9: The AUC learning curves of the real data (the thick black line) and synthetic data generated by G_{PMF} with different sparsity. *Observation: higher s leads to lower upper bound.*

Observation 3. *For datasets generated by PMF, higher sparsity leads to lower AUC upper bound.*

This result is consistent with our observations on both JE and BTE. The theoretical reason can also be found in Section 4.7.3. Just like our other observations, readers can refer to our decision flowchart and guideline in Section 4.8 to know how and when to apply this result in practice.

4.7 Theoretical Study for Experiment Observations

In this section, we provide the theoretical reason for our experiment observations. Specifically, we consider the missing data mechanisms described in Section 4.3 and 4.5. We study how data sparsity s and the number of attributes m affect binary classification problem. We focus on the convergence rate of learning and the asymptotic generalization AUC. We first describe the problem definition.

4.7.1 Problem Definition

Consider the space $\mathcal{X} = \{0,1\}^m$ for inputs $\mathbf{x}$, and an unknown distribution $\mathcal{D} = p(\mathbf{x}, y)$ over $\mathcal{X} \times \{-1, +1\}$. When we use BTE, we denote the resultant data space as $\mathcal{D}_{BTE}(t, \theta)$; similarly, we use $\mathcal{D}_{JE}(t)$ to denote the data space of JE. Sparsity refers to the percentage (probability) of 0 values. The set of attributes are denoted as $F = \{x_1, x_2, ..., x_m\}$. For ease of illustration, we also use $\mathbf{x}_{-j}$ to denote all attributes except x_j. In an asymptotic setting (infinite data samples), a naïve Bayes classifier predicts by:

$$f_{\mathcal{NB}}(\mathbf{x}) = \arg\max_y \Big(\prod_{j=1}^{m} p(x_j|y)p(y) \Big), \tag{4.10}$$

where the following naïve independence assumption is used:

$$p(\mathbf{x}, y) = \prod_{j=1}^{m} p(x_j|y)p(y). \tag{4.11}$$

For naïve Bayes, AUC is based on the scoring of test instance $\mathbf{x}$, given by

$$s_{\mathcal{NB}}(\mathbf{x}) = \log \frac{p(+|\mathbf{x})}{p(-|\mathbf{x})} = \log \frac{p(+)}{p(-)} + \sum_{j=1}^{m} \log \frac{p(x_j|+)}{p(x_j|-)}. \tag{4.12}$$

The (asymptotic) risk of AUC is defined as follows [76]:

$$\mathcal{R}(\mathcal{D}) = \Pr_{(\mathbf{x},y),(\mathbf{x}',y')\sim\mathcal{D}}\Big\{[s_{\mathcal{NB}}(\mathbf{x}) - s_{\mathcal{NB}}(\mathbf{x}')](y - y') \leq 0, y \neq y'\Big\}. \tag{4.13}$$

Now suppose there is a training (empirical) dataset $\mathcal{S}$ sampled i.i.d. from $\mathcal{D}$, estimated values of $p(x_j|y)$ and $p(y)$ in $\mathcal{S}$ are $\tilde{p}(x_j|y) = \frac{N_{y\wedge x_j}}{N_y}$ and $\tilde{p}(y) = \frac{N_y}{|\mathcal{S}|}$, respectively; where $N_{y\wedge x_j}$ denotes the number of instances in $\mathcal{S}$ that have label y and attribute value x_j. We first study how the size of $\mathcal{S}$ affects the convergence of AUC risk.

4.7.2 Convergence Rate Analysis

We first bound the convergence of $s_{\mathcal{NB}}(\mathbf{x})$

Lemma 4.7.1 *Let $\mathcal{S} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_l\}$ be the empirical set sampled i.i.d. from distribution $\mathcal{D}$, $s_{\mathcal{NB}}$ be the naïve Bayes AUC score function defined in Eq.(4.12) whose empirical estimate is $\tilde{s}_{\mathcal{NB}}$. For any unseen (test) instance $\mathbf{x} \sim \mathcal{D}$, to ensure*

$$Pr(|\tilde{s}_{\mathcal{NB}}(\mathbf{x}) - s_{\mathcal{NB}}(\mathbf{x})| \geq \epsilon) \leq \delta, \tag{4.14}$$

it suffices to have a training size

$$l \geq \frac{3\log\frac{4m+4}{\delta}}{p_{min}(\mathbf{x})(1-\exp(-\frac{\epsilon}{2m+2}))^2}, \tag{4.15}$$

where $p_{min}(\mathbf{x}) = \min\left\{p(x_j|y)\middle| j \in \{1, ..., m\}, y \in \{+, -\}\right\}$.

Proof We first upper bound the error between estimated and true score. From Eq. (4.12), with some simple algebraic manipulation, we have

$$|\tilde{s}_{\mathcal{NB}}(\mathbf{x}) - s_{\mathcal{NB}}(\mathbf{x})| \leq \sum_y \left(\sum_{j=1}^{m} |\log\frac{\tilde{p}(x_j|y)}{p(x_j|y)}| + \left|\log\frac{\tilde{p}(y)}{p(y)}\right|\right). \tag{4.16}$$

Following the analysis of Ng and Jordan [69], to study the convergence of $\tilde{s}_{\mathcal{NB}}(\mathbf{x}) \to s_{\mathcal{NB}}(\mathbf{x})$, we bound each of the $2m+2$ terms in the right of Eq. (4.16) individually with Chernoff bound and use Union bound afterwards. Chernoff bound gives

$$\Pr\left(|\frac{\tilde{p}}{p} - 1| \geq \epsilon_1\right) \leq 2\exp(-\frac{l\epsilon_1^2 p}{3}) \tag{4.17}$$

We know:

$$\begin{aligned}
&\Pr\left(|\frac{\tilde{p}}{p} - 1| \geq \epsilon_1\right) \geq \Pr\left(1 - \epsilon_1 \geq \frac{\tilde{p}}{p}\right) \\
&= \Pr\left(\log(1-\epsilon_1) \geq \log\frac{\tilde{p}}{p}\right) \geq \Pr\left(\log(1-\epsilon_1) \geq |\log\frac{\tilde{p}}{p}|\right) \\
&= \Pr\left(\log\frac{1}{1-\epsilon_1} \leq |\log\frac{\tilde{p}}{p}|\right)
\end{aligned} \tag{4.18}$$

(given that $\epsilon_1 \in (0,1)$). Thus

$$\Pr\left(|\log\frac{\tilde{p}}{p}| \geq \log\frac{1}{1-\epsilon_1}\right) \leq \Pr\left(|\frac{\tilde{p}}{p} - 1| \geq \epsilon_1\right) \leq 2\exp(-\frac{l\epsilon_1^2 p}{3}) \tag{4.19}$$

or equivalently,

$$\Pr\left(|\log\frac{\tilde{p}}{p}| \geq \epsilon_1\right) \leq 2\exp\left(-\frac{l(1-\exp(-\epsilon_1))^2 p}{3}\right). \tag{4.20}$$

which entails that $\Pr\left(|\log\frac{\tilde{p}}{p}| \geq \epsilon_1\right) \leq \delta_1)$ holds when

$$l \geq \frac{3\log\frac{2}{\delta_1}}{(1-\exp(-\epsilon_1))^2 p}. \tag{4.21}$$

To bound Eq. (4.16), we set $\epsilon = (2m+2)\epsilon_1$ and $\delta = (2m+2)\delta_1$. Now with the Union bound we know that it suffices to have:

$$l = \frac{3\log\frac{4m+4}{\delta}}{p_{min}(\mathbf{x})(1-\exp(-\frac{\epsilon}{2m+2}))^2}, \tag{4.22}$$

where

$$p_{min}(\mathbf{x}) = \min\left\{p(y), p(x_j|y)\middle| j \in \{1,...,m\}, y \in \{+,-\}\right\}. \tag{4.23}$$

When the data are not extremely unbalanced in terms of the distribution of $p(y)$, $p_{min}(\mathbf{x})$ equals $\min\left\{p(x_j|y)\middle| j \in \{1,...,m\}, y \in \{+,-\}\right\}$.

Remarks. It is obvious that $p_{min}(\mathbf{x})$ is the lowest per-attribute density in either class. With Eq.(4.15), we have now proved that when sparsity increases ($p_{min}(\mathbf{x})$ decreases), the minimum training size l needed for convergence at high probability increases. In other words, convergence rate slows down. This explains the observations in all our experiments that sparser data always lead to slower learning curve convergence (Figures 4.5, 4.7). For curves in Figure 4.9, the convergence rate does not change significantly. This is because sparsity does not vary significantly as we change z in the PMF experiments.

4.7.3 Upper Bound Analysis

Next we study the asymptotic AUC (upper bound) of the learning problem. We first give the qualitative relationship between data sparsity and asymptotic AUC risk. The following lemma is proved through our analysis (we defer its proof to Appendix)

Lemma 4.7.2 *If each attribute x_j has missing probability $\theta \in (0,1)$, then the AUC risk $\mathcal{R}(\mathcal{D})$ decreases with θ monotonically.*

This lemma is important since it explains qualitative observation about AUC upper bound and data sparsity (Figures 4.5 and 4.9), i.e., why higher sparsity always leads to lower upper bound. Notice that Lemma 4.7.2 cannot give and exact quantitative relationship between θ and AUC upper bound. In fact, the exact quantitative relationship should depend on the distribution $\mathcal{D}$ and it may be hard to find a universal answer.

Next we study how the number of attributes affect asymptotic AUC. In the BTE experiment, we have varied the number of attributes m and keep sparsity s unchanged. For this setting, we derive the following lemma (see Appendix for its proof).

Lemma 4.7.3 *When the data distribution of existing attributes does not change, more attributes lead to lower AUC risk.*

Remarks. This lemma gives the qualitative relationship between AUC risk and the number of attributes. From the definition of BTE (Eq. (4.1)), we see that the distribution of each attribute is irrelevant to expansion rate t. Thus, when θ does not change, increasing the number of attributes would not change the distribution of existing attributes. Lemma 4.7.3 is applicable in this situation and explains why more attributes always lead to a lower AUC upper bound (see Figure 4.6).

For JE (Eq. (4.4)), when we increase the expansion rate the data distribution of existing attributes will also change. Lemma 4.7.3 does not apply here. We next study AUC upper bound under JE.

4.7.4 Upper Bound Analysis for Just-1 Expansion

The asymptotic value of JE is special in that the number of attributes and data distribution of existing attributes change simultaneously. However, we could prove that the following lemma holds.

Lemma 4.7.4 *For JE, AUC risk does not change with t.*

Proof Assume that instance $\mathbf{x}$ in space $\mathcal{D}_{JE}(t)$ is expanded from $\tilde{\mathbf{x}}$ in $\mathcal{D}_{JE}(1)$. Since we will consider the change of data space, we use notations $p(\cdot|\mathcal{D})$, $s_{\mathcal{NB}}(\cdot|\mathcal{D})$ for probabilities and naïve Bayes AUC scores in space $\mathcal{D}$. From Eq. (4.4) and (4.12) we have

$$s_{\mathcal{NB}}(\mathbf{x}|\mathcal{D}_{JE}(t)) = \underbrace{\sum_{\tilde{x}_j=0} \log \frac{p(x_j = 0|+; \mathcal{D}_{JE}(1))^t}{p(x_j = 0|-; \mathcal{D}_{JE}(1))^t}}_{\text{sum over missing (0-valued) attributes of instance } \tilde{\mathbf{x}}} +$$

$$\underbrace{\sum_{\tilde{x}_j=1} \log \frac{p\left(x_j=1|+;\mathcal{D}_{JE}(1)\right) p\left(x_j=0|+;\mathcal{D}_{JE}(1)\right)^{t-1}}{p\left(x_j=1|-;\mathcal{D}_{JE}(1)\right) p\left(x_j=0|-;\mathcal{D}_{JE}(1)\right)^{t-1}}}_{\text{sum over observed (1-valued) attributes of instance } \tilde{\mathbf{x}}} + \log \frac{p(+)}{p(-)}$$

$$= (t-1)s_{\mathcal{NB}}\Big(\tilde{\mathbf{x}}|\mathcal{D}_{JE}(1)\Big) + (t-1)s_{\mathcal{NB}}\Big(\mathbf{0}|\mathcal{D}_{JE}(1)\Big) + \log \frac{p(+)}{p(-)}. \tag{4.24}$$

Given $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{x}}'$ from the original space $\mathcal{D} = \mathcal{D}_{JE}(1)$, whenever $s_{\mathcal{NB}}\Big(\tilde{\mathbf{x}}|\mathcal{D}_{JE}(1)\Big) < s_{\mathcal{NB}}\Big(\tilde{\mathbf{x}}'|\mathcal{D}_{JE}(1)\Big)$, we always have that $s_{\mathcal{NB}}(\mathbf{x}|\mathcal{D}_{JE}(t)) < s_{\mathcal{NB}}(\mathbf{x}'|\mathcal{D}_{JE}(t))$ because of (4.24). Thus from the definition of AUC risk Eq. (4.13), we know that JE always keeps AUC risk unchanged. This also explains our observation in Figure 4.7.

4.8 A Practical Guide

In this section, we show how our results can help classify real-world large sparse datasets with naïve Bayes. First, we summarize our empirical observations and theoretical results for different missing data mechanisms and various settings of m and s in Table 4.3.

Table 4.3: upper bound (U) and convergence rate (V) of AUC learning curves

Missing Mechanisms	Attributes $m \uparrow$	Sparsity $s \uparrow$	Notations
BTE	$U \uparrow$	$U \downarrow, V \downarrow$	$\uparrow$ / $\downarrow$ / $\rightarrow$: increase/decrease/unchanged
JE	$U \rightarrow, V \downarrow$	$U \rightarrow, V \downarrow$	(ns): not studied
G_{PMF}	ns	$U \downarrow, V \downarrow$	

We show how these results help answer the following practical questions mentioned in the introduction of this section:

Question 1. *How does data sparsity affect the convergence rate of AUC as data increases?*

Our Answer: Higher data sparsity leads to slower AUC convergence rate; see Figure 4.5 and 4.7 in our experiments. Lemma 4.7.1 can be used as a PAC bound for the relation between sparsity and the convergence rate of naïve Bayes AUC.

Question 2. *How does data sparsity affect the asymptote (upper bound) of AUC?*

Our Answer: For most missing data mechanisms including BTE and G_{PMF}, sparser data leads to a lower asymptotic AUC; see Figures 4.5, 4.9 and Lemma 4.7.2. Besides, we also identify a special missing data mechanism JE where sparsity does not affect asymptotic AUC; see Figure 4.6 and its analysis in Lemma 4.7.4.

Question 3. *To improve AUC, do we need (a) more data, (b) more attributes or (c) to reduce the sparsity of existing data?*

Our Answer: To decide which actions to take, practitioners should consider the learning curve behavior as well as the missing data mechanism. Specifically, we provide the following guideline, as outlined in Figure 4.10.

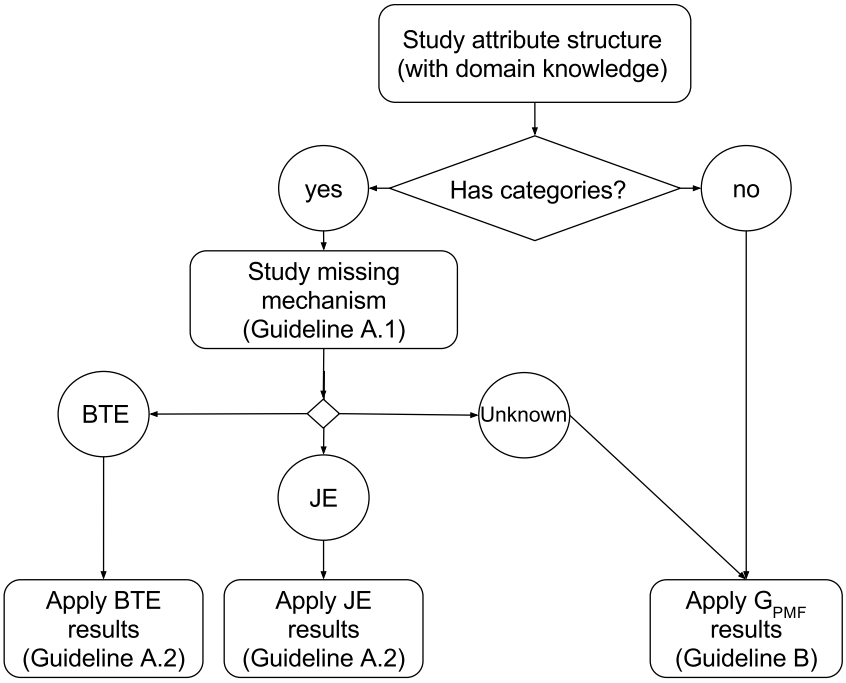


Figure 4.10: The decision flowchart of our practical guideline.

Given a sparse user behavior dataset, we first identify if there are attribute categories using domain knowledge. If there are such categories then go to Guideline A, otherwise see Guideline B.

Guideline A.1. We continue to identify which expansion assumption (BTE or JE) holds based on domain knowledge. For the user-movie example introduced in Section 4.3, since a user could watch several movies of a category, we can assume that data are generated by BTE from category-level information. For cases where at most one attribute of a category is active, we should use the JE assumption. For instance, JE is suitable when each attribute corresponds the most recent cellphone purchased by a user. If neither BTE nor JE applies, then go to Guideline B, otherwise A.2.

Guideline A.2. After identifying the missing data mechanism, we are able to apply our results about BTE and JE in Table 4.3 to predict the learning curve behavior after certain candidate actions we take. For example, adding more attributes usually gives better generalization AUC for both BTE and JE; however, the learning curve behavior can also be different for BTE and JE, thus Figures 4.6, 4.7 and Lemmas 4.7.3 and 4.7.4 can be used as a guideline.

Guideline B. For datasets where no attribute categories exist, we can use the probabilistic modeling approach to answer the above question. Specifically, we first infer the probabilistic model of the sparse dataset, and use G_{PMF} to generate a much larger synthetic dataset and predict the AUC learning curve behavior. This is what we have done in Section 4.6 on real-world movie rating datasets. For example, according to Figure 4.9, we could predict how changing sparsity affects the asymptotic convergence behavior.

4.9 Relation to Previous Work

In this section, we mainly review: (1) previous research that applies or studies naïve Bayes using large sparse datasets; (2) other machine learning methods for dealing with large sparse user behavior data; (3) related research which also uses synthetic datasets for verification.

Though simple, naïve Bayes was reported to perform surprisingly well on many classification tasks [78, 26, 47, 71]. Ng and Jordan [69] have analyzed the asymptotic behavior of a multivariate naïve Bayes classifier. The authors discover that a naïve Bayes classifier needs a training set of size $l = \Omega(\log(m))$ to obtain near-optimal generalization error. However, this argument has not been systematically evaluated on large and sparse datasets with different mechanisms of missing.

For datasets with a large number of attributes, many attributes might be irrelevant, redundant or noisy [39, 48, 38]. In dealing with such datasets, naïve Bayes classifiers are often used with attribute selection algorithms [37, 4, 84]. However, for user behavior data where each attribute often contains a small but non-negligible amount of information, selecting a subset of the most informative attributes could not be optimal, as demonstrated in Junqué de Fortuny et al. [40].

Recommender systems often have to deal with sparse user behavior datasets. For example, in the Netflix Prize contest [3], contestants are provided with a dataset containing millions of movie ratings given by anonymous users, a very large proportion of the ratings are missing. Early systems [80, 46] use imputation to fill the missing values. However, inaccurate imputation could distort the true (unknown) data distribution and harm the predictors' performance. Matrix-Factorization [49] is a way to model the data with only the observed values while using regularization to prevent over-fitting. Many variants of Matrix-Factorization are later introduced, of which, Steck [86] has analyzed and made assumptions on the patterns of data missing and designed a novel objective function for MF. Instead of trying to fill the missing values or using the observed to predict the unknown entries, in this thesis we focus on

studying how data sparsity affects classification performance.

There is a popular branch of MF that uses Bayesian probabilistic models to fit the data, latent variables and their relationships. A complex generation process is assumed in these methods. In Mnih and Salakhutdinov [67], the term Probabilistic Matrix Factorization is proposed to describe such methods. Earlier, Meeds et al. [64] have provided a Bayesian probabilistic MF model that assumes data $\mathbf{X}$ is generated by the product of $\mathbf{UWV}$, where $\mathbf{U}, \mathbf{V}$ are binary matrices, and $\mathbf{W}$ is a real-valued weight matrix. Salakhutdinov and Mnih [79] have proposed a fully Bayesian treatment for PMF, which leads to better predictive performance when evaluated on the Netflix dataset. Hernandez et al. [32] have proposed a PMF model for binary sparse datasets and its variational Bayes inference algorithm. It outperforms recent similar methods when tested on several large binary datasets. In this chapter, we use it to generate sparse datasets. A modified version of this model is also used in the missing data model (MDM) of the MF-MNAR algorithm [31], which as a whole is a PMF model for data that is MNAR.

There are previous works in the field of PMF that generates synthetic dataset for evaluation purposes. Mohamed [68] has used a prototype based approach to generate binary matrix with coin-flipping probabilities. The performance of the PMF model is evaluated by reconstructing the synthetic dataset from different sizes of sampling. Sutherland et al. [87] have generated synthetic matrix of certain patterns of observation (missing) to evaluate the performance of their PMF-based active learning algorithm. Synthetic random matrix of ordinal user ratings are also used to evaluate RMSE (root-mean-square error). Hernandez et al. [32] have created synthetic data using the PMF data generation process with pre-specified hyper-parameters. In our G_{PMF} method, synthetic data are created with only the posterior parameters estimated from a known real-world data.

4.10 Summary

Naïve Bayes has good scalability on large sparse datasets. In this chapter, we have studied the convergence behavior of naïve Bayes on large sparse user behavior datasets. Specifically, we use generated datasets to explore how different mechanisms of missing, data sparsity and the number of attributes systematically affect the AUC learning curve behavior. We consider several realistic Missing At Random mechanisms for binary user behavior data. We propose the corresponding data generation approaches using uniform dilution (BTE, JE) and probabilistic modeling (G_{PMF}). We generate very large synthetic datasets for each missing data mechanism and study the entire learning curve behavior when systematically changing sparsity and

attribute number. Several useful observations have been made, and we provide detailed theoretical studies for those observations. We also provide an intuitive decision flowchart and a guideline on how our results could be used in practice.

Chapter 5

Convergence Behavior of Linear Classifiers on Large Sparse Data

Large sparse datasets are very common in product recommendation applications. For scalability reasons, linear classifiers are preferred in classification tasks on such datasets. The previous two chapters have studied how data sparsity affects linear SVM and naive Bayes classification under typical missing data mechanisms for binary inputs. In this chapter, we will extend our study to all discriminative linear classifiers.[1] We will also generalize the missing data mechanisms and synthetic data generation methods to real-valued datasets. Using real-world and synthetic experiments, we observe several important learning curve behaviors under different missing mechanisms. We also study the theoretic reasons for all our observations. Our studies provide a practical guideline to determine if or when obtaining more data and/or obtaining missing values in the data is worthwhile or not. This can be very valuable in many applications.

5.1 Introduction

It is known that more training data can lead to lower generalization risk and asymptotically the risk converges to the optimal for a given class of classifiers [89]. For a specific data distribution and a family of classifiers, the asymptotic classification risk and how fast learning converges when increasing the training size are fundamental problems of learning. As the sizes of real-world datasets increase, these convergence behaviors become more practically

[1] Linear SVM is a discriminative linear classifier. While the multinomial naive Bayes model has been shown to be a (generative) linear classifier, other naive Bayes variants such as the Bernoulli naive Bayes classifier considered in the previous chapter are not linear classifiers.

important nowadays. Chapter 3 has studied how data sparsity affects linear SVM and its convergence behavior. For naive Bayes classifier, Chapter 4 has also answered how sparsity affects the convergence rate and asymptote of AUC under typical missing data mechanisms. However, the above studies are still limited to specific classifiers and the experiments and analysis were focusing on binary input data.

In this chapter, we significantly extend our work to address discriminative linear classifiers and generalized loss functions. Instead of studying the missing mechanisms of binary input data, we consider common missing mechanisms for large sparse datasets of real-value inputs here: *Uniform Missing* (UM) and *Uniform Dilution* (UD), which are more general. Section 5.3 of this chapter describes the details of the two missing mechanisms.

To empirically demonstrate the classification behavior in an asymptotic setting, we generate arbitrarily large i.i.d. data using the probabilistic model [79] of real-world large sparse data and introduce the missing mechanisms afterwards. Our experiments cover different discriminative linear classifiers and loss functions. We have several important observations that consistently hold for different classifiers/losses. To confirm our results and provide an in-depth explanation, we prove all our observations under the framework of Statistical Learning Theory [89]. As with the previous chapter, our results help answer the following questions for applying discriminative linear classifiers with different loss measures on real-valued large sparse data:

1. How does data sparsity affect the *convergence rate* of learning as sample size increases?

2. How does data sparsity affect the *asymptotic* classification risk?

3. To improve classification performance, do we need (a) *more data samples*, (b) *more attributes* or (c) *to reduce the sparsity of existing data*?

This can have an economic impact in real applications, because either to impute the missing values statistically or to query the actual missing values comes at a cost. To easily apply our results in practice, we provide an intuitive decision flowchart and its guidelines in Section 5.6.

The rest of this chapter is organized as follows. Section 5.2 and 5.3 establish our problem settings and describe the missing data models. Section 5.4 describes experiments and observations on both real-world and synthetic datasets. Section 5.5 presents our theoretic studies. Section 5.6 gives a guideline for using our results in practice. Section 5.7 reviews related works; and the final section concludes the chapter.

5.2 Linear Classification and Asymptotic Risk

For scalability reason, linear classifiers are often used in mining large sparse data. In this section, we introduce the basic problem setting for discriminative linear classification.

5.2.1 Linear Classification

Without loss of generality, we study the classification problem with binary label as specified below. Given an unknown distribution $\mathcal{D} = p(\mathbf{x}, y)$ over $\mathbb{R}^d \times \{1, -1\}$, the full set of input attributes for the task is denoted as $F = \{x_1, x_2, ..., x_d\}$. For user feedback data, each data instance $\mathbf{x} \in \mathbb{R}^d$ corresponds to feedbacks given by a user to d items, such as ratings (ordinal values), 'like's (binary values), etc. The goal is to learn a function f within some hypothesis class $\mathcal{H}$ to minimize classification risk:

$$R_{\mathcal{D}}(f) = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\Big[l(f(\mathbf{x}), y)\Big], \tag{5.1}$$

where $l(f(\mathbf{x}), y)$ is a loss function. In this chapter, $\mathcal{H}$ can be all linear classifiers $\mathcal{H}_l = \{f : f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x}, \mathbf{w} \in \mathbb{R}^d\}$. The solution to this classification problem amounts to finding $f^* = \arg\min_{f\in\mathcal{H}_l} \{R_{\mathcal{D}}(f)\}$. We can then use f^* to predict unseen data. In this chapter, we consider commonly used loss functions including the 0-1 loss and also its convex surrogates such as hinge and logistic loss [2].

5.2.2 Asymptotic Risk and Convergence Rate

In practice, the empirical risk of a classifier:

$$R^{\text{emp}}_{\mathcal{S}\sim\mathcal{D}}(f) = \frac{1}{N}\sum_{i=0}^{N} l(f(\mathbf{x}^{(i)}), y^{(i)}) \tag{5.2}$$

is used to approximate generalization risk in the above learning problem, where $\mathcal{S} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1,...,N}$ is an empirical dataset sampled from $\mathcal{D}$. By the Uniform Convergence of Empirical Risk Minimization [89], when $|\mathcal{S}|$ continues to increase, empirical risk and true risk will converge asymptotically, so long as $\mathcal{H}$ has bounded VC-dimension (true for linear classifiers $\mathcal{H}_l$). We consider the Empirical Risk Minimization [89] setting which does not include the regularization term; this is reasonable because we are dealing with large sample sizes and simple classifiers.

We denote $\hat{R}_{\mathcal{D}}$ as this asymptotic empirical/true risk. We hereafter use *asymptotic risk* for brevity:

$$\hat{R}_{\mathcal{D}} := \lim_{|\mathcal{S}|\to\infty} R^{\text{emp}}_{\mathcal{S}\sim\mathcal{D}}(f^*_{\mathcal{S}}) \approx R_{\mathcal{D}}(f^*), \tag{5.3}$$

where the approximation holds with high probability in the asymptotic setting, and $f^*_{\mathcal{S}}$ is the optimal linear classifier learnt from empirical set $\mathcal{S}$:

$$f^*_{\mathcal{S}} := \arg\min_{f \in \mathcal{H}_l} \left\{ R^{\text{emp}}_{\mathcal{S}\sim\mathcal{D}}(f) \right\}. \tag{5.4}$$

As we increase $|\mathcal{S}|$, the *convergence rate* of learning describes how fast the empirical risk converges to the asymptotic risk $\hat{R}_{\mathcal{D}}$. Given the measure of convergence $\epsilon > 0$, the convergence rate can be calculated as the smallest empirical size $\hat{N}$ that guarantees high probability convergence between the (optimized) empirical risk and asymptotic risk:

$$\hat{N} := \arg\min_{|\mathcal{S}|} \left\{ \Pr(|R^{\text{emp}}_{\mathcal{S}\sim\mathcal{D}}(f^*_{\mathcal{S}}) - \hat{R}_{\mathcal{D}}| > \epsilon) < \delta \right\}. \tag{5.5}$$

Practically, it is impossible to have an infinitely large dataset. However, the increasing scales of real-world applications are pushing linear classifiers towards this idealistic frontier. As data are becoming larger and larger, on the one hand, it is important to study complicated models with a large learning capacity [52]. On the other hand, it is also practically important to know how different factors might affect the convergence behaviors of simpler data mining models. This thesis focuses on the later perspective, and approaches the data sparsity problem by studying how it affects the convergence of discriminative linear classifiers.

5.3 Sparsity and Missing Data Models

When using linear classifiers on large sparse data, missing values in the input $\mathbf{x}$ are treated as 0s (0-imputation) by convention, and *sparsity* refers to the percentage (probability) of 0s. As before, we assume that data sparsity is caused by some missing mechanisms that can be modeled mathematically. In Chapter 3 and 4, we have introduced missing data mechanisms for binary data. Here, we will study missing mechanisms for large sparse data of real-valued inputs, which is more general. We consider two simple missing mechanisms each of which addresses a major cause of sparsity: (1). limited user feedback budget; (2) large number of items. According to the missing data theory [56, 77], these two missing mechanisms are MAR (Missing At Random) models, i.e., we do not make explicit assumptions about how attribute values affect the likelihood of missing.

5.3.1 Uniform Missing

The first missing data model *Uniform Missing* (UM) addresses how user activeness (behavior budgets) affects the sparsity of a user feedback dataset. Figure 5.1 gives a graphical explanation. Basically, UM assumes that each attribute has a uniform likelihood of missing.

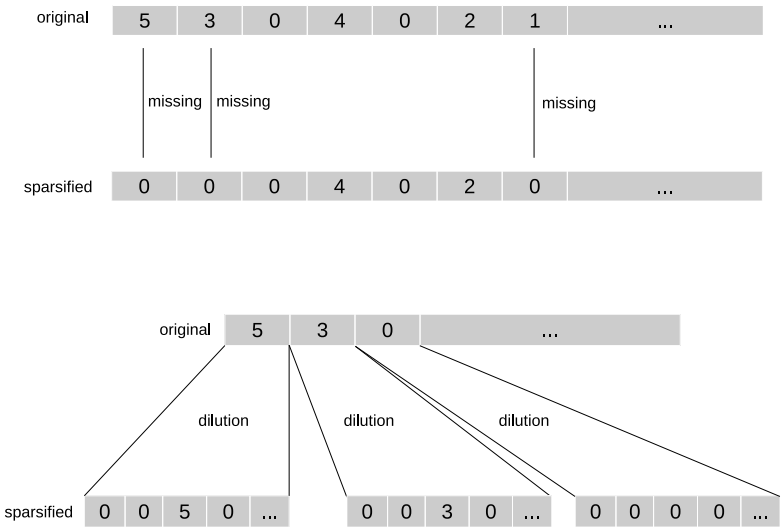


Figure 5.1: A graphical illustration of Uniform Missing (upper figure) and Uniform Dilution (lower figure). 0 denotes missing.

Notice that the binary version of this missing mechanism has been used in our study of linear SVM (see Section 3.2.4), here we override its definition for the case of real-valued inputs. Given a set of attributes (items) $\Phi \subseteq F$, UM models the change of user activeness over Φ using a latent multiplicative factor $s \in [0, 1]$. Formally, given an original data distribution $\mathcal{D}$ over $\mathbb{R}^d \times \{1, -1\}$, Uniform Missing assumes that all attribute values in Φ are subject to a uniform likelihood of missing $s \in [0, 1]$. See Figure 5.1 for a graphical illustration. We denote the resultant data distribution as $\mathcal{D}_{miss}(\Phi, s)$, and its joint distribution as $p_{miss}(\mathbf{x}, y|\Phi, s)$. Notice that this missing data model is additive, in the sense that the original data $\mathcal{D}$ could be either dense or sparse.

With its additive property, UM can generalize to the case where different attributes have different probabilities of missing. For example, if some attributes $\Phi_1 \subseteq F$ are missing with probability s_1, while others $\Phi_2 \subseteq F$ are missing with s_2, we can recursively define $\mathcal{D}'_{miss}(\Phi_2, s_2)$ where $\mathcal{D}' = \mathcal{D}_{miss}(\Phi_1, s_1)$. In this chapter, we omit such complications and only study $\mathcal{D}_{miss}(\Phi, s)$.

5.3.2 Uniform Dilution

The major cause of sparsity in a feedback dataset is actually two-fold – user activeness (behavior budget) is limited and the number of available items is huge. The aforementioned Uniform

Missing model addresses how user activeness affects sparsity. To model sparsity caused by the large number of items, we use the *Uniform Dilution* (UD) mechanism which was also introduced in the previous chapter for the case of binary data. Basically, UD dilutes each non-missing value using attribute expansion, as can be seen in Figure 5.1. Here we briefly review the practical meaning of UD and generalize it to real-valued input data.

For many user feedback data, there can be latent clusters of items such that items of the same cluster are highly similar [53, 91, 66], e.g., different episodes of a TV opera. It is reasonable that *a cluster of similar items often have similar user feedbacks*. For example, if a user gives a high rating to *The Lord of the Rings I*, (s)he will likely to rate *The Lord of the Rings II* high as well. In order to systematically study the sparsity caused by the number of items, UD considers the case where ratings are sparsified (diluted) evenly across the items of each cluster. In this manner, we could use an existing real-world dataset as the prototype and expand it to generate the synthetic large sparse data.

Formally, given an original data distribution $(\mathbf{x}, y) \sim \mathcal{D}$ over $\mathbb{R}^d \times \{1, -1\}$, a dilution rate t and a set of attributes $\Phi \in F$, the Uniform Dilution mechanism specifies a data distribution $\mathcal{D}_{dilute}(\Phi, t)$ over the expanded space $\mathbb{R}^{d-|\Phi|+t|\Phi|} \times \{1, -1\}$. From a generative perspective, to sample an instance $(\mathbf{x}, y)$ from $\mathcal{D}_{dilute}(\Phi, t)$ is equivalent to sample $(\tilde{\mathbf{x}}, y)$ from $\mathcal{D}$ first and dilute $\tilde{\mathbf{x}}$ to $\mathbf{x}$ afterwards. We still use the two dilution processes introduced in the previous chapter, i.e., Bernoulli-trial Expansion (BTE) and Just-1 Expansion (JE). The only difference is that we here consider real-valued attribute values instead of binary values; see Figure 5.1.

We can apply Uniform Dilution to any subset of attributes Φ of a real-world large sparse dataset. Essentially, UD expands item set Φ for t times and dilutes the user feedback information in a uniform manner. In this respect, UD can model the sparsity caused by the increasing number of items.

In the previous chapter, we have also considered the missing mechanism derived from a Probabilistic Matrix Factorization model [32]. However, the PMF model is limited to data matrices of binary inputs and not suitable for real-valued inputs. Besides, changing the sparsity by tuning the parameter z in the binary PMF model (see Eq.(4.9)) is equivalent to varying the global user activeness, which is already addressed by the UM mechanism. In the next section, we use experiments to study how UD and UM affect the convergence behaviors of linear classifiers.

5.4 Empirical Study

To demonstrate how different sparsity rates and missing data mechanisms affect the learning behavior, we first describe our findings in real-world and synthetic experiments.

5.4.1 Experiments with Real-world Data

We experiment with real-world large sparse datasets introduced in Table 1.1. We apply the two missing mechanisms UM and UD on each dataset with different degrees of sparsity. 20% of each dataset are held out for testing. In order to study the learning curve behaviors as training size grows, we increase training sample sizes according to $\{2^8, 2^{8.25}, 2^{8.5}, \dots\}$ until the whole training set is used. For each training sample, we compute the optimized model and its training and testing errors. To apply the UM models, we vary data sparsity by changing the probability of missing for both training and testing data according to $s \in \{0.1, 0.2, 0.5, 0.7\}$; for UD, we used the Just-1 Expansion strategy and vary the rate of dilution according to $t \in \{1, 4, 8, 16\}$. We use linear SVM as our target classifier with a very small regularization ($\lambda = 10^{-5}$) in this experiment. The experiment results of *ymovie, Epinions, book* datasets are depicted in Figure 5.2.

We observe that these real-world data are not large enough to reveal how learning curves converge. The curves keep changing even at their ends. To overcome this challenge, we next describe the approach of synthetic data generation using probabilistic models of real-world large sparse datasets.

5.4.2 Synthetic Data Generation

To generate arbitrarily large datasets, our approach is to sample i.i.d. data from the statistical distribution of real-world datasets. Specifically, we use Probabilistic Matrix Factorization [67, 79] to model and infer the empirical distribution of real-world user-item data. Different from previous chapters, we select two versions of PMF models which could generate binary data (implicit feedback) and ordinal data (explicit feedback), respectively. The first PMF only supports binary data [32], as introduced in Section 3.2.3 and 4.5. The other PMF model is catered for ordinal rating matrix [31]. These two models are recent variants of PMF that have well-designed probabilistic modeling framework and have been tested to have good predictive performance on user-item datasets. The details of these two models can be found in their original papers [31, 32]. Here we focus on how to generate large sparse i.i.d. data after we have inferred the model posteriors from a real-world user rating matrix.

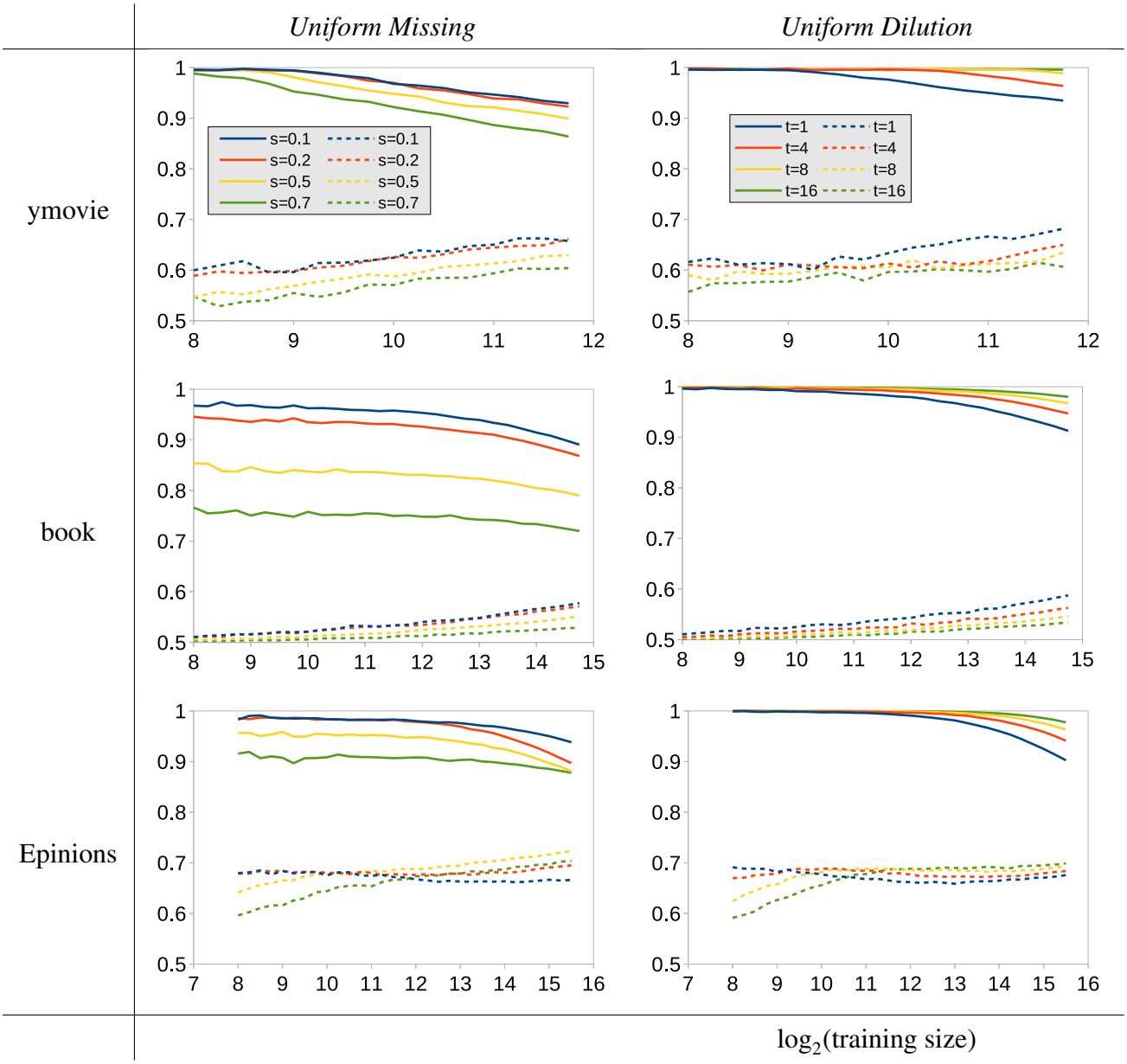


Figure 5.2: Classification accuracy on **real-world** data with different missing data mechanisms. For UM and UD, sparsity rate increases with s and t, respectively. Solid and dashed lines indicate training and testing accuracies, respectively.

For either of the PMF models, we denote the data (users) belonging to class y as $\mathbf{D}_y$, the latent parameters for a PMF as Ξ and the corresponding posterior as $p(\Xi|\mathbf{D}_y)$. [2] Again, the meaning of y is problem specific, for example, y could be the gender of a user. The distribution of synthetic data is $p(\mathbf{x}, y) = p(\mathbf{x}|y)p(y)$. For $p(y)$, we simply use a balanced class prior, $p(+) = p(-) = 0.5$; $p(\mathbf{x}|y)$ is the posterior predictive distribution after we infer the model posterior:

$$p(\mathbf{x}|y) = p(\mathbf{x}|\mathbf{D}_y) = \int p(\mathbf{x}|\Xi)p(\Xi|\mathbf{D}_y)d\Xi. \tag{5.6}$$

Notice that $p(\mathbf{x}|\Xi)$ in Eq. (5.6) should be different from the original posterior predictive of a PMF, which is often given in the form of $p(\mathbf{X}|\Xi)$, where $\mathbf{X}$ is the data matrix that are used for inferring Ξ. In other words, the posterior predictive distribution by default only has support for *in-matrix* prediction and could only be used to predict (or reconstruct) entries in the original $\mathbf{X}$, as is used in most collaborative filtering literature.

In order to generate an arbitrary number of users, we still use the prototype-based sampling strategy: whenever we need to generate/sample a new synthetic instance $\mathbf{x}$, we first randomly choose a user $\mathbf{X}_i$ from the users (rows) that exist in $\mathbf{X}$ as a user prototype. We then generate the new synthetic user using the model posterior of this user prototype, i.e., Ξ_i. The following pseudo code illustrates this idea:

Algorithm 2 Data Generation from a generalized PMF model

Infer the posteriors $p(\Xi|\mathbf{D}_+)$ and $p(\Xi|\mathbf{D}_-)$;
for each new instance $(\mathbf{x}, y)$ **do**
  Sample y from $p(+) = p(-) = 0.5$;
  Pick a user u_i randomly from $\mathbf{D}_y$ as the prototype;
  Sample Ξ_i from $p(\Xi_i|\mathbf{D}_y)$;
  Sample $\mathbf{x}$ from $p(\mathbf{x}|\Xi_i)$;
end for

This algorithm allows us to sample infinite instances from distributions of real-world datasets. The advantage of this data generation methodology is two-fold: (1). by generating an arbitrarily large dataset, we could see the convergence behavior of classification. (2). from a statistical point of view, we are studying distributions of real large sparse data.

Next we describe our experiment results on synthetic datasets generated using Algorithm 2.

[2] With this notation, the data generation distribution used in Section 3.2.3 and 4.5 becomes a special case, where $\Xi = \{U, V, z\}$.

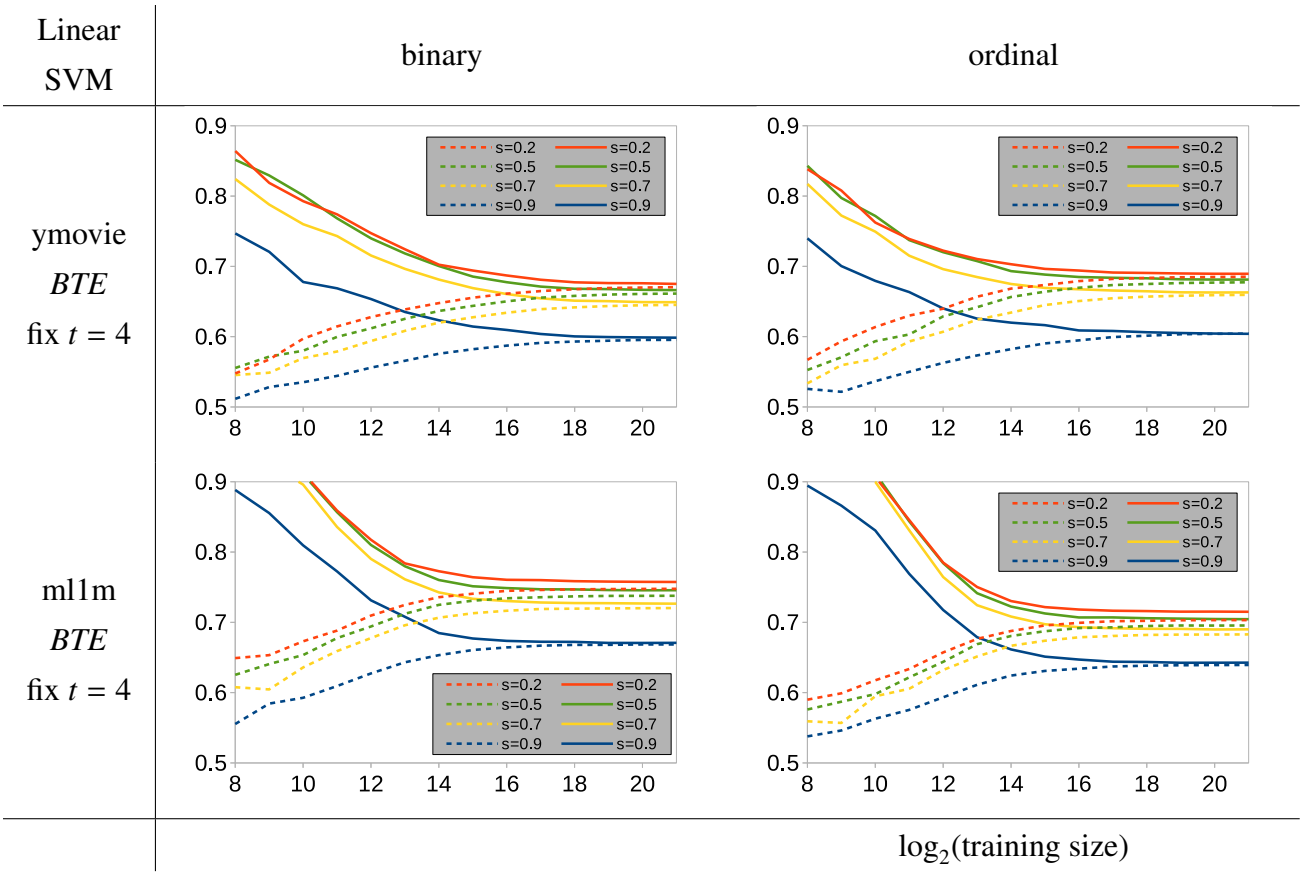


Figure 5.3: Linear SVM classification accuracy on synthetic large sparse data generated from BTE with fixed expansion rate t and various missing likelihood s. Solid and dashed lines indicate training and testing accuracies, respectively. The rate of convergence can be measured by the training size needed to approach convergence.

5.4.3 Experiments on Synthetic Data

In this section, we describe how we use the proposed data generation process to evaluate our theoretic results in an asymptotic setting.

Data and Experiment Settings we use two movie rating datasets that are widely studied in the collaborative filtering literature, i.e., *ml1m* and *ymovie*, which were also mentioned previously in Table 1.1. We have not used larger datasets such as *epinions* and *book* to do Bayesian inference, because it is costly on data matrices of such a large scale. We consider the gender of a user as the class label y.

For each real-world dataset, we infer its Bayesian PMF model posterior Eq. (5.6) and generate samples of various sizes using Algorithm 2. Initially, we find that learning curves converge very slowly (a huge number of synthetic data are needed), especially for the UD mechanism.

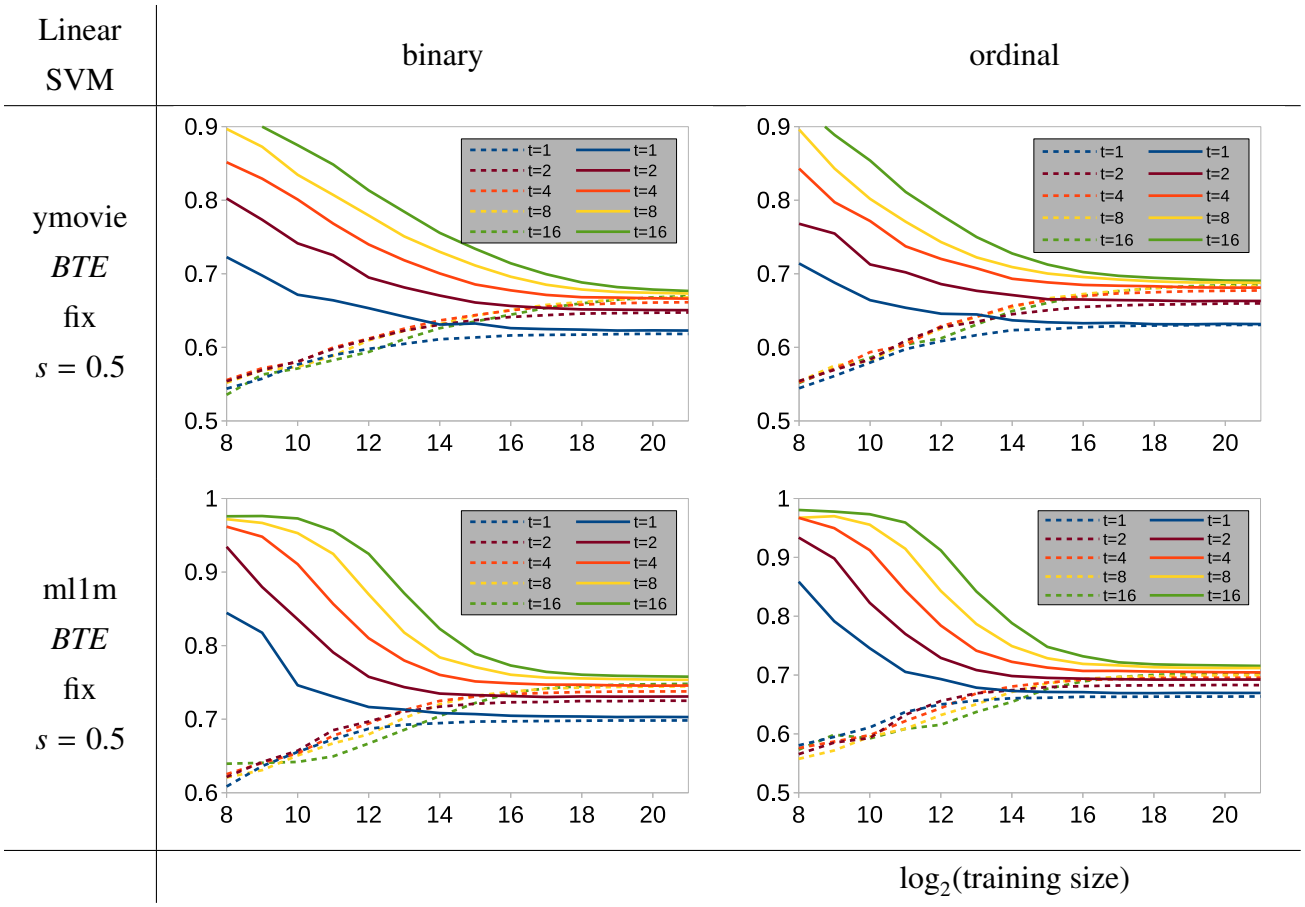


Figure 5.4: Linear SVM classification accuracy on synthetic large sparse data generated from BTE with fixed missing likelihood s and various expansion rate t. Solid and dashed lines indicate training and testing accuracies, respectively. The rate of convergence can be measured by the training size needed to approach convergence.

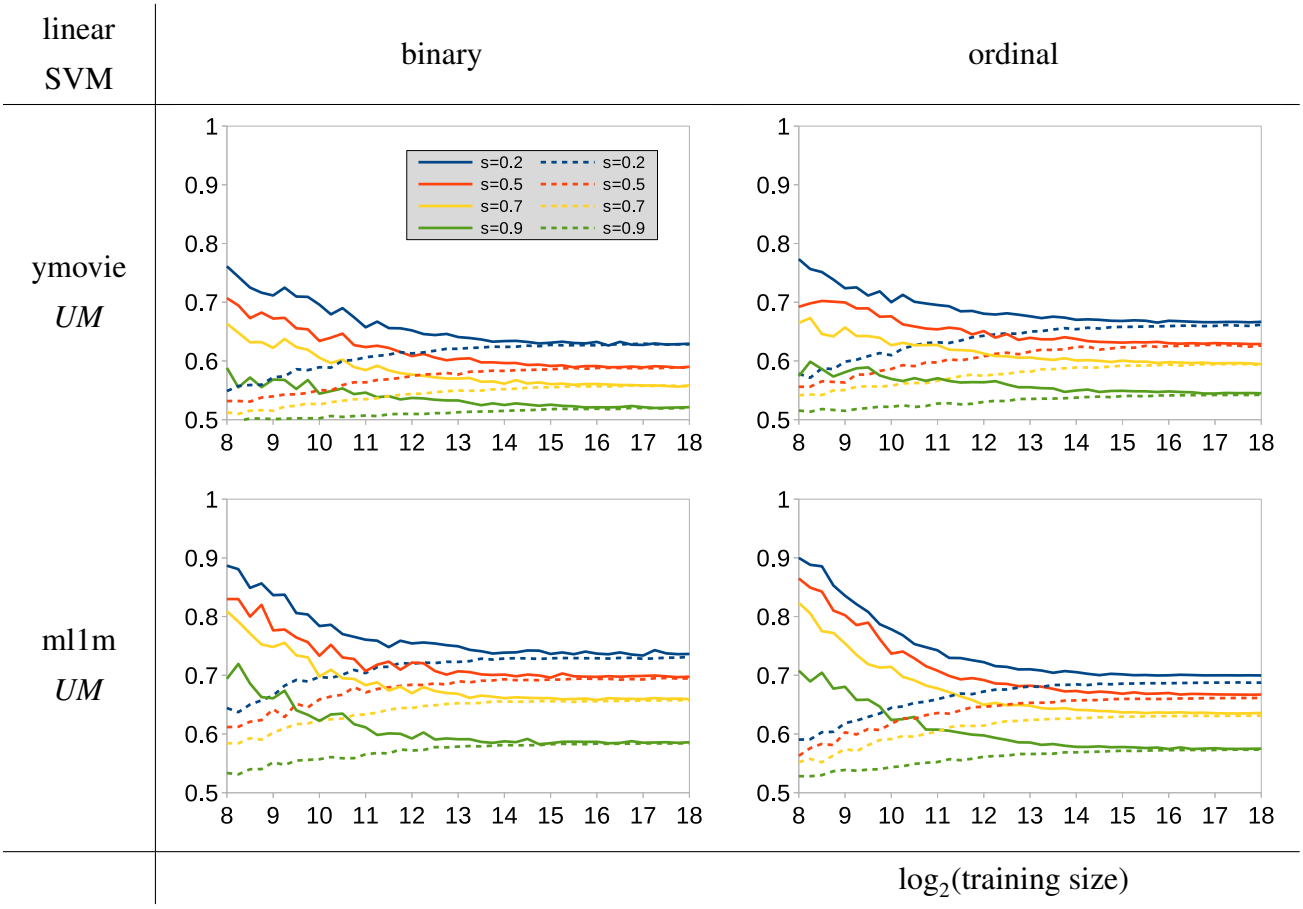


Figure 5.5: Linear SVM classification accuracy on synthetic large sparse data with UM missing mechanism. Solid and dashed lines indicate training and testing accuracies, respectively. The rate of convergence can be measured by the training size needed to approach convergence.

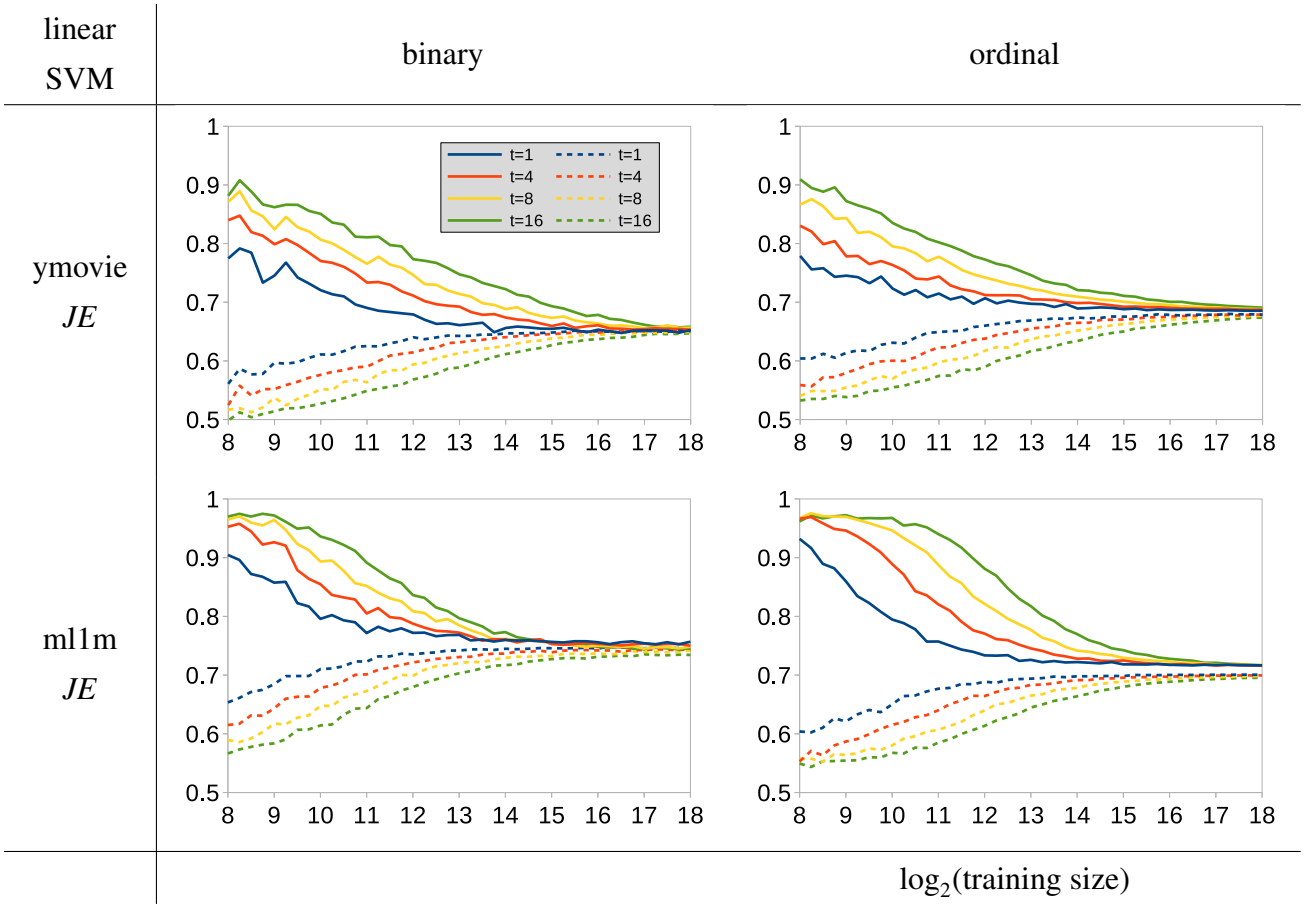


Figure 5.6: Linear SVM classification accuracy on synthetic large sparse data with JE missing mechanism. Solid and dashed lines indicate training and testing accuracies, respectively. The rate of convergence can be measured by the training size needed to approach convergence.

In order to improve efficiency, we randomly choose a subset of items (200 items in *ml1m*, 800 items in *ymovie*) beforehand. This item selection process helps reveal the convergence behavior with less data generation time.

To study our research questions systematically, we employ both missing mechanisms on the synthetic data. We introduce various data missing likelihoods $s \in \{0.2, 0.5, 0.7, 0.9\}$ for UM, while for UD, we set the dilution rates to be $t \in \{1, 2, 4, 8, 16\}$. We use a held-out synthetic dataset of size 2^{18} for testing, which is generated from the same probabilistic model with the corresponding missing mechanism.

We use linear SVM and Logistic Regression as our target classifiers, which are popular in practice and have different loss functions (hinge and logistic loss). The two types of classifiers show very similar learning curves in our experiments.

Experiment Results. For each missing mechanism, training size and sparsity, we record the training and testing accuracy for the learned classifier. The experiment results are given in Figures 5.3, 5.4, 5.5 and 5.6. We have several important observations:

Observation 1. *For BTE with fixed t and UM*[3]*, adding sparsity increases asymptotic classification risk.*

Observation 2. *For BTE with fixed s, adding features also decreases asymptotic classification risk.*

Observation 3. *For JE, adding features and sparsity*[4] *slows down learning convergence.*

Observation 4. *For JE, adding features and sparsity does not affect the asymptotic classification risk.*

Compared to our experiment results on naive Bayes and binary datasets of Chapter 4, these four observations are mostly consistent with only a minor difference: in our empirical and theoretical results (Lemma 4.7.1) for naive Bayes, we find that adding sparsities in BTE slows down the convergence rate, this is not significant in the figures here. This is no surprise, as Lemma 4.7.1 is derived based on the naive Bayes inference process over binary inputs, while the classifiers studied in this chapter are discriminatively trained with real-valued inputs.

To summarize, observations 1-4 indicate that how sparsity affects convergence of learning depends on the underlying mechanism of missing. Section 5.6 will discuss how these results can help in classifying large sparse data in practice. To confirm our findings, we first prove that the observations are theoretically correct.

[3]If we fix $t = 1$ and vary the missing likelihood s, BTE will be equivalent to UM

[4]remember that features and sparsity will increase simultaneously with JE

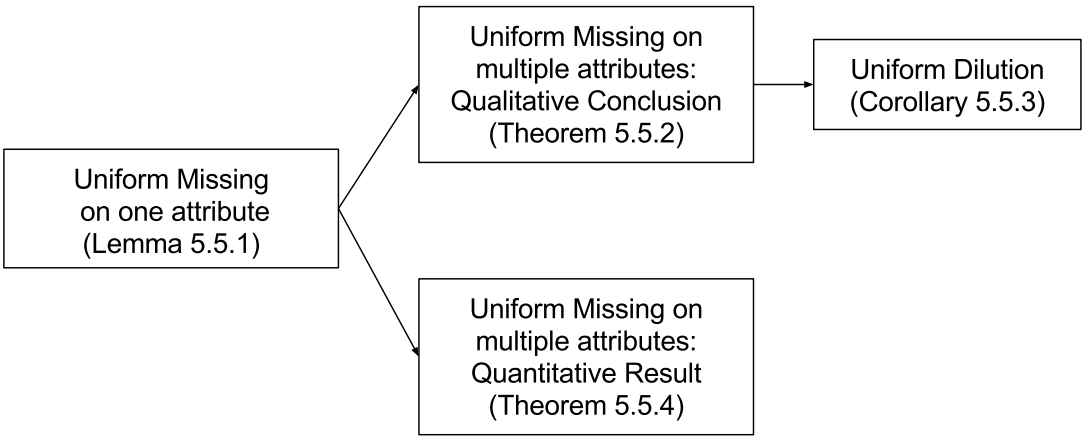


Figure 5.7: The road map of our theoretic study about asymptotic risk.

5.5 Theoretical Study for Experiment Observations

In this section, we present the theoretical studies for our experiment observations. For each missing mechanism, we study how sparsity affects asymptotic risk and rate of learning convergence: we explain why higher sparsity always increases the asymptotic risk (Theorem 5.5.2) under UM and BTE with fixed t, but not under JE (Corollary 5.5.3). Moreover, we identify the quantitative relationship between asymptotic risk and the likelihood of missing in UM (Theorem 5.5.4). Finally, we study why the missing mechanisms lead to different behaviors of learning convergence rate.

5.5.1 Notations

We denote the asymptotic risk under the UM, JE and BTE models as $\hat{R}_{\mathcal{D}_{miss}(\Phi,s)}$, $\hat{R}_{\mathcal{D}_{je}(\Phi,t)}$ and $\hat{R}_{\mathcal{D}_{bte}(\Phi,t,s)}$, respectively. Their definitions can be found by simply replacing distribution $\mathcal{D}$ in Equations (5.3) and (5.4) with the corresponding distributions. In UM, attributes in Φ are completely missing if $s = 1.0$, which is equivalent as if the classifier only accesses attributes in $F - \Phi$ (set subtraction). We denote a classifier that only accesses $J \subseteq F$ as f_J. In such a case, the asymptotic risk is denoted as

$$\hat{R}_{\mathcal{D},J} := \lim_{|\mathcal{S}|\to\infty} R^{\text{emp}}_{\mathcal{S}\sim\mathcal{D}}(f^*_{\mathcal{S},J}), \tag{5.7}$$

where $f^*_{\mathcal{S},J}$ is the empirically optimized linear classifier using loss $l(f_J(\mathbf{x}), y)$:

$$f^*_{\mathcal{S},J} := \arg\min_{f\in\mathcal{H}_l} \left\{R^{\text{emp}}_{\mathcal{S}\sim\mathcal{D}}(f_J)\right\}. \tag{5.8}$$

5.5.2 Asymptotic Risk for Different Missing Mechanisms

Our proofs on asymptotic risk follow the road map shown in Figure 5.7. Specifically, we first derive the accurate value and a lower bound of asymptotic risk when UM is introduced to only one attribute (Lemma 5.5.1). This lemma also enables us to derive our major qualitative (Theorem 5.5.2) and quantitative results (Theorem 5.5.4) about UM. These results are also applicable to BTE with fixed t and varying s. As a corollary to Theorem 5.5.2, we also find why JE does not change asymptotic risk (Corollary 5.5.3).

We start by considering UM and if Φ contains only one attribute, x_j. We use $\{\mathbf{x}_{-j}\}$ to denote the set of attributes except for x_j. In this case, Lemma 5.5.1 gives the exact asymptotic risk and also a lower bound when Uniform Missing is introduced to only one attribute. Intuitively, it indicates that the asymptotic risk equals to the minimum of weighted average risk of: (1). totally ignoring attribute $\{x_j\}$; (2). having full observation of $\{x_j\}$.

Lemma 5.5.1

$$\begin{aligned}\hat{R}_{\mathcal{D}_{miss}(\{x_j\},s)} &= \min_f \{(1-s)R_{\mathcal{D}}(f) + sR_{\mathcal{D}}(f_{\{\mathbf{x}_{-j}\}})\} \\ &\geq (1-s)\hat{R}_{\mathcal{D}} + s\hat{R}_{\mathcal{D},\{\mathbf{x}_{-j}\}} \end{aligned} \tag{5.9}$$

Proof According to the Uniform Missing model, we could compute $\mathcal{D}_{miss}(\{x_j\}, s) = p_{miss}(\mathbf{x}, y|\{x_j\}, s)$ from $\mathcal{D} = p(\mathbf{x}, y)$. If $x_j \neq 0$,

$$p_{miss}(\underline{\mathbf{x}_{-j}, x_j}, y|\{x_j\}, s) = (1-s) \cdot p(\underline{\mathbf{x}_{-j}, x_j}, y) \tag{5.10}$$

and

$$\begin{aligned} p_{miss}(\underline{\mathbf{x}_{-j}, 0}, y|\{x_j\}, s) &= s\int_{\tau\neq 0} p(\underline{\mathbf{x}_{-j}, \tau}, y)\,\mathrm{d}\tau + p(\underline{\mathbf{x}_{-j}, 0}, y) \\ &= s\int_{\tau} p(\underline{\mathbf{x}_{-j}, \tau}, y)\,\mathrm{d}\tau + (1-s)p(\underline{\mathbf{x}_{-j}, 0}, y) \\ &= sp(\mathbf{x}_{-j}, y) + (1-s)p(\underline{\mathbf{x}_{-j}, 0}, y). \end{aligned} \tag{5.11}$$

Using the above two equations, we can rewrite the risk of $\mathcal{D}_{miss}(\{x_j\}, s)$ as:

$$\begin{aligned} R_{\mathcal{D}_{miss}(\{x_j\},s)}(f) &= \int_y \int_{\mathbf{x}_{-j}} \int_{x_j} l(f(\mathbf{x}), y) p_{miss}(\mathbf{x}, y|\{x_j\}, s)\,\mathrm{d}\mathbf{x}\,\mathrm{d}y \\ &= \int_y \int_{\mathbf{x}_{-j}} \Big\{ \int_{x_j\neq 0} l(f(\mathbf{x}), y) p_{miss}(\mathbf{x}, y|\{x_j\}, s)\,\mathrm{d}x_j + \\ &\quad l(f(\underline{\mathbf{x}_{-j}, 0}), y) p_{miss}(\underline{\mathbf{x}_{-j}, 0}, y\big|\{x_j\}, s)\Big\}\,\mathrm{d}\mathbf{x}_{-j}\,\mathrm{d}y. \end{aligned} \tag{5.12}$$

Using Eqs. (5.11, 5.10) we have

$$
\begin{aligned}
R_{\mathcal{D}_{miss}(\{x_j\},s)}(f) = \int_y \int_{\mathbf{x}_{-j}} \Big\{ (1-s) \int_{x_j \neq 0} l(f(\mathbf{x}), y) p(\mathbf{x}, y)\, \mathrm{d}x_j \\
+ l(f(\underline{\mathbf{x}_{-j}, 0}), y) \Big[(1-s) p(\underline{\mathbf{x}_{-j}, 0}, y) + s p(\mathbf{x}_{-j}, y) \Big] \Big\}\, \mathrm{d}\mathbf{x}_{-j}\, \mathrm{d}y \\
= \int_y \int_{\mathbf{x}_{-j}} \Big\{ (1-s) \int_{x_j} l(f(\mathbf{x}), y) p(\mathbf{x}, y) + s l(f(\underline{\mathbf{x}_{-j}, 0}), y) p(\mathbf{x}_{-j}, y) \Big\}\, \mathrm{d}\mathbf{x}_{-j}\, \mathrm{d}y \\
= (1-s) R_{\mathcal{D}}(f) + s R_{\mathcal{D}}(f(\mathbf{x}_{-j})).
\end{aligned} \tag{5.13}
$$

Hence the asymptotic empirical risk of $\mathcal{D}_{miss}(\{x_j\}, s)$ is:

$$
\hat{R}_{\mathcal{D}_{miss}(\{x_j\},s)} \approx \min_f R_{\mathcal{D}_{miss}(\{x_j\},s)}(f) = \min_f \{(1-s) R_{\mathcal{D}}(f) + s R_{\mathcal{D}}(f(\mathbf{x}_{-j})\}. \tag{5.14}
$$

Because $\min_u \{h(u) + g(u)\} \geq \min_u h(u) + \min_u g(u)$, we have

$$
\hat{R}_{\mathcal{D}_{miss}(\{x_j\},s)} \geq (1-s) \hat{R}_{\mathcal{D}} + s \hat{R}_{\mathcal{D},\{\mathbf{x}_{-j}\}}. \tag{5.15}
$$

∎

Next, we use this lemma to find the reason why higher sparsity always increases asymptotic risk under Uniform Missing (Observation 1 of our experiments).

Qualitative Results For the optimal classifier f_J^* that only accesses attribute set J, it is obvious that

$$
\hat{R}_{\mathcal{D}} \leq R_{\mathcal{D}}(f_J^*) = \hat{R}_{\mathcal{D},J}. \tag{5.16}
$$

This actually explains the **Observation 2**, i.e., *for BTE with fixed s, adding features also decreases asymptotic classification risk*. In this particular BTE experiment, we have kept the missing probability fixed and increased the expansion rate, this is equivalent to adding more features while keeping the distribution of existing features unchanged.

When $J = \{\mathbf{x}_{-j}\}$ we have $\hat{R}_{\mathcal{D}} \leq \hat{R}_{\mathcal{D},\{\mathbf{x}_{-j}\}}$, and with Eq. (5.9) we know that:

$$
\hat{R}_{\mathcal{D}_{miss}(\{x_j\},s)} \geq \hat{R}_{\mathcal{D}}, \tag{5.17}
$$

which indicates that adding Uniform Missing to one attribute increases asymptotic risk. Notice that

$$
\hat{R}_{\mathcal{D}_{miss}(\{x_j\},s)} = \min_f \{R_{\mathcal{D}}(f) + s[R_{\mathcal{D}}(f_{\{\mathbf{x}_{-j}\}}) - R_{\mathcal{D}}(f)]\}. \tag{5.18}
$$

Thus when $s_1 \leq s_2$, we have

$$
\hat{R}_{\mathcal{D}_{miss}(\{x_j\},s_1)} \leq \hat{R}_{\mathcal{D}_{miss}(\{x_j\},s_2)}, \tag{5.19}
$$

which proves that when Uniform Missing is only introduced to one attribute, higher sparsity leads to higher asymptotic risk.

If we apply this result one attribute at a time for all attributes in Φ, we can prove our **Observation 1** that *higher sparsity leads to higher asymptotic classification risk for Uniform Missing*:

Theorem 5.5.2

$$\hat{R}_{\mathcal{D}_{miss}(\Phi,s_1)} \leq \hat{R}_{\mathcal{D}_{miss}(\Phi,s_2)}, \tag{5.20}$$

where $0 \leq s_1 \leq s_2 \leq 1$.

Based on this theorem, the following corollary explains **Observation 4** of our experiments, i.e., *JE does not change the asymptotic risk*:

Corollary 5.5.3 *For Just-1 Expansion, we have:*

$$\hat{R}_{\mathcal{D}_{je}(\Phi,t)} = \hat{R}_{\mathcal{D}}. \tag{5.21}$$

Proof Notice that diluting multiple attributes is equivalent to diluting each of the attributes sequentially, one after another. We can thus assume $\Phi = \{x_j\}$ and try to prove:

$$\hat{R}_{\mathcal{D}_{je}(\{x_j\},t)} = \hat{R}_{\mathcal{D}} \tag{5.22}$$

instead, without loss of generality.

Suppose $f^*(\mathbf{x}) = \mathbf{w}^* \cdot \mathbf{x}$ is the minimizer of $R_{\mathcal{D}}(f)$ among all linear classifiers $\mathcal{H}_l$. We construct a linear classifier $f'(\mathbf{x})$ for data in the diluted space as:

$$f'(\mathbf{x}) = \sum_{i=1, i\neq j}^{d} w_i^* \cdot x_i + \sum_{q=1}^{t} w_j^* \cdot x_{j_q}, \tag{5.23}$$

where x_{j_q} are attributes diluted from x_j. By the definition of asymptotic risk and Uniform Dilution, we have:

$$R_{\mathcal{D}_{je}(\{x_j\},t)}(f') = R_{\mathcal{D}}(f^*) = \hat{R}_{\mathcal{D}}. \tag{5.24}$$

Thus

$$\hat{R}_{\mathcal{D}} \geq \min_{f\in\mathcal{H}_l} R_{\mathcal{D}_{je}(\{x_j\},t)}(f) = \hat{R}_{\mathcal{D}_{je}(\{x_j\},t)}. \tag{5.25}$$

We next show that $\hat{R}_{\mathcal{D}} \leq \hat{R}_{\mathcal{D}_{je}(\{x_j\},t)}$ is also true. Consider a special distribution: starting from the original distribution $\mathcal{D}$, we only copy the attributes in x_j for t times as $\{x_{j_1}, \ldots, x_{j_t}\}$ but do

not dilute them. We denote this resultant distribution as $\mathcal{D}_{copy}(\{x_j\}, t)$. From the definition of Just-1 Expansion, we know that

$$\mathcal{D}_{je}(\{x_j\}, t) = \mathcal{D}'_{miss}\left(\{x_{j_1}, \ldots, x_{j_t}\}, 1 - \frac{1}{t}\right), \tag{5.26}$$

where $\mathcal{D}' = \mathcal{D}_{copy}(\{x_j\}, t)$. In other words, *Just-1 Expansion on an attribute is equivalent to duplicate the attribute t times and apply Uniform Missing on the duplicated attributes*. It is obvious that duplicating attributes does not change asymptotic classification risk: $\hat{R}_{\mathcal{D}} = \hat{R}_{\mathcal{D}'}$. From Theorem 5.5.2, we now have:

$$\hat{R}_{\mathcal{D}} = \hat{R}_{\mathcal{D}'} \leq \hat{R}_{\mathcal{D}'_{miss}(\{x_{j_1},\ldots,x_{j_t}\},1-\frac{1}{t})} = \hat{R}_{\mathcal{D}_{je}(\{x_j\},t)}. \tag{5.27}$$

The proof can be completed after combining Equations (5.25) and (5.27). ∎

To summarize, the above results indicate that higher sparsity would surely lead to worse classification performance for UM (and BTE with fixed t, varying s); while for JE, the increased risk can be alleviated by having larger data, and asymptotically the risk will not be affected by dilution. These two conclusions are consistent with the theoretical studies on naive Bayes and AUC risk in the previous chapter. However, the analysis presented here can apply to discriminative linear classifiers and generalized loss functions, which has much broader application.

Quantitative Results From a quantitative perspective, we use the next theorem to answer how much the asymptotic risk would change, given the uniform missing likelihood s. Intuitively speaking, Eq. (5.28) entails that the asymptotic risk on $\mathcal{D}_{miss}(\Phi, s)$ equals to weighted average risk of training over each attribute subspace $F - J$, where J is every possible subspace of Φ, and each J gets picked with a probability controlled by a binomial distribution. Eq. (5.29) further concludes that the optimized risk over $\mathcal{D}_{miss}(\Phi, s)$ is higher than the averaged risk of separately training each subspace:

Theorem 5.5.4 *$\forall \Phi \subseteq F$, let $K = |\Phi|$, the asymptotic risk for distribution under Uniform Missing $\mathcal{D}_{miss}(\Phi, s)$ satisfies*

$$\hat{R}_{\mathcal{D}_{miss}(\Phi,s)} = \min_f R_{\mathcal{D}_{miss}(\Phi,s)}(f) = \min_f \{\mathbb{E}_{k\sim Bin(K,s)}\mathbb{E}_{J\in\mathcal{P}_k(\Phi)}R_{\mathcal{D}}(f_{F-J})\} \tag{5.28}$$

$$\geq \mathbb{E}_{k\sim Bin(K,s)}\{\mathbb{E}_{J\in\mathcal{P}_k(\Phi)}\hat{R}_{\mathcal{D},F-J}\}, \tag{5.29}$$

where $Bin(N, p)$ denotes a binomial distribution, $\mathcal{P}_k(\Phi)$ denotes all subsets of set Φ that has k elements.

Proof The basic idea of this proof is to apply Lemma 5.5.1 additively, one attribute at a time. Consider the following process which introduces sparsity one attribute at a time:

$$\Phi_0 = \emptyset; \Phi_k = \Phi_{k-1} \cup \{x_k\}; \Phi_K = \Phi. \tag{5.30}$$

Because of Eq. (5.13) in Lemma 5.5.1, we have :

$$R_{\mathcal{D}_{miss}(\Phi_k,s)}(f) = (1-s)R_{\mathcal{D}_{miss}(\Phi_{k-1},s)}(f) + sR_{\mathcal{D}_{miss}(\Phi_{k-1},s)}(f_{\{\mathbf{x}_{-k}\}}), \tag{5.31}$$

where we can apply the expansion rule (5.13) again:

$$R_{\mathcal{D}_{miss}(\Phi_{k-1},s)}(f) = (1-s)R_{\mathcal{D}_{miss}(\Phi_{k-2},s)}(f) + sR_{\mathcal{D}_{miss}(\Phi_{k-2},s)}(f_{\{\mathbf{x}_{-(k-1)}\}}) \tag{5.32}$$

and

$$R_{\mathcal{D}_{miss}(\Phi_{k-1},s)}(f_{\{\mathbf{x}_{-k}\}}) = (1-s)R_{\mathcal{D}_{miss}(\Phi_{k-2},s)}(f_{\{\mathbf{x}_{-k}\}}) + sR_{\mathcal{D}_{miss}(\Phi_{k-2},s)}(f_{\{\mathbf{x}_{-(k,k-1)}\}}). \tag{5.33}$$

After applying the above two expansion rules sequentially for $k = \{K, K-1, \ldots, 2\}$, we arrive at:

$$\begin{aligned} R_{\mathcal{D}(\Phi,s)}(f) &= (1-s)^K R_{\mathcal{D}}(f) + \sum_{k=1}^{K} \sum_{J \in \mathcal{P}_k(\Phi)} s^k (1-s)^{K-k} R_{\mathcal{D}}(f_{F-J}) \\ &= \sum_{k=0}^{K} \sum_{J \in \mathcal{P}_k(\Phi)} s^k (1-s)^{K-k} R_{\mathcal{D}}(f_{F-J}). \end{aligned} \tag{5.34}$$

Notice that since $|\mathcal{P}_k(\Phi)| = \binom{K}{k}$, we have

$$\begin{aligned} R_{\mathcal{D}_{miss}(\Phi,s)}(f) &= \sum_{k=0}^{K} \binom{K}{k} s^k (1-s)^{K-k} \frac{\sum_{J \in \mathcal{P}_k(\Phi)} R_{\mathcal{D}}(f_{F-J})}{|\mathcal{P}_k(\Phi)|} \\ &= \mathbb{E}_{k \sim \text{Bin}(K,s)} \{ \mathbb{E}_{J \in \mathcal{P}_k(\Phi)} R_{\mathcal{D}}(f_{F-J}) \}, \end{aligned} \tag{5.35}$$

where we have used the fact that $\binom{N}{x} p^x (1-p)^{N-x}$ is the probability mass function of a binomial distribution $x \sim \text{Bin}(N, p)$. Because $R_{\mathcal{D}}(f_{F-J})$ is always positive for all f and J:

$$\begin{aligned} \hat{R}_{\mathcal{D}_{miss}(\Phi,s)} &= \min_f \{ \mathbb{E}_{k \sim \text{Bin}(K,s)} \mathbb{E}_{J \in \mathcal{P}_k(\Phi)} R_{\mathcal{D}}(f_{F-J}) \} \\ &\geq \mathbb{E}_{k \sim \text{Bin}(K,s)} \mathbb{E}_{J \in \mathcal{P}_k(\Phi)} \min_f R_{\mathcal{D}}(f_{F-J}) \\ &= \mathbb{E}_{k \sim \text{Bin}(K,s)} \{ \mathbb{E}_{J \in \mathcal{P}_k(\Phi)} \hat{R}_{\mathcal{D},F-J} \}, \end{aligned} \tag{5.36}$$

Notice though we have specified the order of choosing attribute at each step k; however, the result is irrelevant to this ordering.

The above theorem is useful for understanding how much the classification risk will change as sparsity is reduced/increased with UM.

5.5.3 Learning Convergence Rate

From the PAC theory [89] we know that the convergence rate of learning depends on the VC-dimension [5] of the hypothesis space. If d is the number of attributes, the VC-dimension of a linear classifier equals to $d + 1$, which is $\min([\frac{R^2}{\Delta^2}], d) + 1$ for Δ-margin linear classifiers, where R is the input radius. When we apply JE, the VC-dimension increases with d, which leads to slower convergence rate (**Observation 3** in our experiments). Meanwhile, we argue that Uniform Missing does little affect on the VC-dimension,[6] thus we do not observe change of convergence rate in experiments of UM or BTE with fixed t. In the next subsection, we will discuss how to apply our empirical and theoretical results in practice.

5.6 A Practical Guideline

After detailed empirical and theoretic studies, Table 5.1 summarizes our major conclusions about learning behaviors with respect to different missing mechanisms, which can be used as a practical reference. Compared to the results in the previous chapter, this table has a broader application because it works for the whole family of discriminative linear classifiers on real-valued data inputs.

Table 5.1: Asymptotic risk ($\hat{R}$) and convergence rate (V) of discriminative linear classification.

Missing Mechanism	Attributes ↑	Sparsity ↑	Notations
UM	ns	$\hat{R} \uparrow$	$\uparrow$: *increasing* $\downarrow$: *decreasing*
JE	$\hat{R} \rightarrow, V \downarrow$	$\hat{R} \rightarrow, V \downarrow$	$\rightarrow$: *keep unchanged*
BTE	$\hat{R} \downarrow$	$\hat{R} \uparrow$	ns : not supported

To easily apply our results in real applications, we also provide a guideline to help make decisions such as whether to collect more data instances or to reduce the current missing values. The flowchart and guidelines are very similar to the one we presented in Section 4.8. Basically, the first step is still to determine which expansion assumption (UM, BTE or JE) is likely to hold using domain knowledge. Then we refer to each of the following detailed guidelines. See the decision flowchart (Figure 5.8) and guidelines below:

[5] VC-dimension is not the only theoretic measurement for learning convergence rate, but it does provide a good explanation for our experiment observations.

[6] As far as we know, Long and Servedio [57] have made an initial step in understanding how data sparsity affects the margin of hyperplanes, but how sparsity affects the VC-dimension of a problem can still be an open problem.

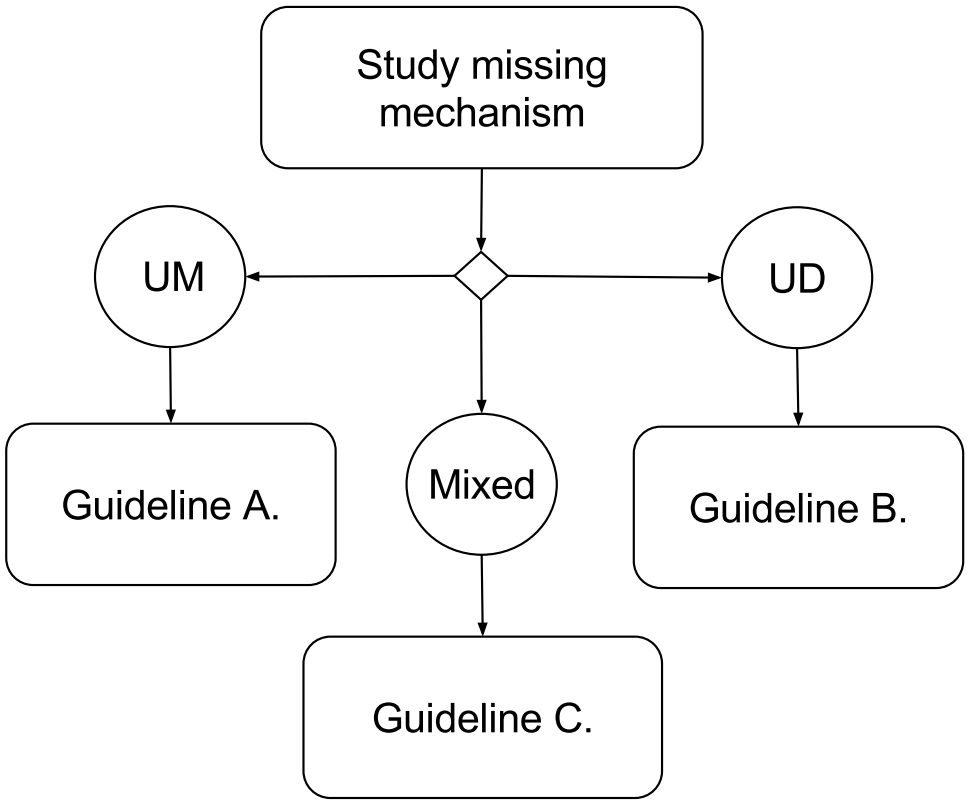


Figure 5.8: The decision flowchart of our practical guideline.

Guideline A. In the case of UM (or BTE with fixed t), higher sparsity always decreases learning performance. This is true irrespective of data size. In practice, no matter how large the sample size we have, reducing the sparsity rate will always help us build a better predictive model. For example, we can apply certain strategies to increase the activeness of the users or to interactively query the users for the missing values. If or not this is worthwhile depends on the cost of reducing sparsity and the benefit of improved performance for the specific application.

Guideline B. This guideline is useful when we find clusters of similar items. In the case of BTE, we know that adding more attributes (items) will help improve classification performance. This motivates collecting more attributes when the extra effort is affordable.

In the case of JE, our results indicate that sparsity does not change the asymptotic risk, but does affect the rate of convergence. In our experiments, when the data size is small, we observe that sparsity causes classification performance to deteriorate. However, this deterioration can be gradually *compensated* after seeing more training instances. In this case, we do not necessarily need to reduce the sparsity of existing data by querying users to give more feedbacks. In fact, we can actually gain better predictive performance as the user population grows.

In the case where we cannot have more data, or when obtaining more data is too expensive

for a specific application, it is still useful to identify the similar items (latent item clusters) and aggregate them together, e.g., combining *The Lord of the Rings* trilogy into one movie in a movie recommendation dataset. This amounts to reducing the expansion rate t in our JE experiments, which is helpful especially when data size is small, see Figure 5.6.

Guideline C. A real-world dataset is more likely to have more than one missing data mechanisms. Still takes a movie recommendation dataset as an example: for a certain group of users, we may have none or too few feedbacks on some items (UM), meanwhile we may have clusters of items which are very similar (BTE or JE) and can be aggregated. In this case, we need to apply different strategies accordingly.

5.7 Related Works

Classifying data with missing or corrupted values has been a classic topic for the machine learning and data mining community [17, 56]. In this section, we mainly review some of the research efforts in this direction.

To deal with missing data in classification, imputation is a widely used strategy. The goal of imputation is to reduce a problem with missing data to the simpler problem where all data could be observed. Commonly used imputation strategies include unconditional and conditional mean imputation, maximum likelihood and multiple imputation. Readers are referred to Little [55] for a review. However, advanced imputation methods often involve inferring the Bayesian probabilistic model of the observed and missing data, which is computationally inefficient for large datasets. Besides, the effectiveness of these methods also rely on selecting the right model or prior, which is also a complicated issue.

Apart from imputation, there is a large body of work devoted to learning with missing or corrupted attributes without an explicit imputation step. For example, Pelckmans et al. [70] have proposed to minimize an unbiased estimate of the empirical risk when missing values are present, and proposes an ERM algorithm based on Least-Square SVM. Our study also relates to the field of budgeted learning [59, 44, 7], where a learner only has access to a limited number of attributes at training [28, 51] or testing [16], and the budget constraint can be either on a per-instance or global basis. However, budgeted learners get the chance to actively choose which attributes to observe, while for the problem we studied, the missingness of a dataset is fixed and before learning.

Apart from these related works, our formulation of Uniform Missing (Section 5.3) also connects to dropout learning with marginalized corrupted features [85, 58, 11, 10]. By artificially

corrupting attributes with some known distribution, dropout learning aims to train robust classifiers by minimizing the expected loss under corruption, thus to mimic the idealistic setting of training with infinite (corrupted) examples [58]. However, the purpose of dropout learning is not to study the convergence performance, since the corruption is only introduced to the limited training dataset.

There are several recent studies [25, 29] that address the problem of classifying datasets with missing values by using the low-rank assumption. The intuition is that large sparse data might reside in a lower dimension as implicitly evidenced by the existence of prototypical users in many user item datasets. With this assumption, Hazan et al. [29] have proposed a kernel-based algorithm that could classify provably as well as the best classifier that has access to the full data. It is very interesting to study how such kernel classifiers perform on large sparse datasets. However, scalability and accuracy is always a trade-off among different choices of classifiers. Complicated models such as kernels and matrix completion based methods usually lead to higher time complexity, thus harder to evaluate with large scale data. In the next chapter, we will address the scalability issue of applying the kernel method [29] for classifying large sparse data.

5.8 Summary

There is a growing need for mining large sparse data. For such datasets, how sparsity systematically affects linear classification is an important data mining problem. We assume that data sparsity is caused by certain missing mechanisms that can be modeled mathematically. The previous chapter has studied how typical missing mechanisms affect the convergence of naive Bayes classifiers with the AUC measure. In this chapter, we have greatly extended our study by considering all linear classifiers and generalized loss functions. With real-world and synthetic experiments, we observe different learning curve behaviors under different missing data mechanisms. We further prove all our observations theoretically. Our results provide a practical guideline to determine if or when obtaining more data and/or obtaining missing values in the data is worthwhile or not. This can be very valuable in applications of classifying large sparse datasets.

Chapter 6

Scalable and Effective Methods for Classifying Large Sparse Data

Classifying large sparse data is important in many real-world applications, such as to predict users' gender and profitability based on product ratings. Traditionally, such datasets are often classified with linear models such as logistic regression and linear SVM for scalability, but the predictive accuracy may suffer. Previous studies suggest that large sparse datasets often have a low-rank structure. By finding the polynomial approximation to the low-rank space, Hazan et al. [29] developed a kernel algorithm (KARMA) for classifying such datasets with a higher accuracy. However, their algorithm does not scale well to large datasets. In this paper, we develop novel scalable feature mappings to efficiently approximate the kernels used in KARMA. In experiments, our method is comparable with KARMA on medium-sized data and scales well to larger datasets that KARMA does not. Our method also significantly outperforms linear classifiers on datasets of various sizes. Overall, our method finds a good trade-off between effectiveness (better accuracy) and scalability, which is very useful for classifying large sparse data in practice.

6.1 Introduction

As we have described in earlier chapters, many user-item datasets in real-world applications are extremely sparse and very high in dimensionality. For example, the Flickr dataset [8], as one of the datasets used in this chapter has a feature dimensionality of $d = 497,470$ and sparsity $s = 99.9994\%$. Again, classifying such large sparse data is important in many applications, such as to predict users' gender and profitability based on the product ratings.

Traditionally, linear models (such as logistic regression and linear SVM) are preferred in such a learning task because of their efficiency in dealing with sparsity. Because of the high feature dimensionality, many complicated models (such as neural network methods) may fail to scale to large datasets. We have analyzed the learning behavior of linear models and data sparsity in previous chapters. However, an important question that remains unanswered is that whether a linear model is our best option for classifying the large sparse data in practice. It is reasonable to worry that the predictive accuracy may suffer, because the linear models do not have a rich learning capacity and hence cannot exploit the non-linear data structures.

In recent years, large sparse datasets have been the focus of collaborative filtering research. Many popular algorithms in this field rely on the assumption that the dataset is of a low rank, e.g., the Probabilistic Matrix Factorization models [67, 32, 31] we have extensively used in previous chapters for generating synthetic data. This low-rank assumption is quite intuitive as items/users can often be well-described by a small number of latent factors. For classification tasks, there are also research efforts to take advantage of this low-rank property. For example, Goldberg et al. [25] attempt to treat the classification label as an additional column to the input feature matrix, and assume that the augmented matrix is of low rank. This formulation allows the use of matrix completion algorithms to solve the classification problem. However, optimizing the matrix completion loss function is neither necessary nor sufficient for ensuring good classification performance [29]. Meanwhile, this strategy is also hardly scalable to datasets of a very large dimension.

More recently, Hazan et al. [29] have developed a classification framework for low-rank and missing *input* data. Compared to Goldberg et al. [25], the new framework considers the low-rank property of the input features only, which is more natural for practical classification tasks. By using polynomial approximation of the low-rank input space, they demonstrate that the framework is possible to compete with the oracle linear classifiers which have access to the full input data (i.e., without missing) during training. Based on the framework, the authors provide a learning algorithm, KARMA (Kernelized Algorithm for Risk-minimization with Missing Attributes), which is a kernel classifier and is optimized with stochastic gradient descent. While KARMA outperforms previous methods on medium-sized sparse data, it still does not scale well to larger datasets because of the expensive kernel computation; see Section 6.2.2.

In this chapter, we develop a scalable learning algorithm based on KARMA [29] which provides a better trade-off between scalability and classification accuracy. Instead of using kernels which are computationally expensive, we propose to use feature mappings that satisfy

the following criteria:

- **Compactness**. The feature mapping can be computed and stored efficiently.
- **Expressiveness**. Similar to the KARMA kernels, the feature mapping should capture the linear and also non-linear (low-rank) structures in the sparse data.

To achieve this goal, we first study the empirical performance of KARMA with different degrees of polynomial approximation. We find that degree-two polynomial approximation often leads to improved generalization performance in most cases, even though the KARMA kernel supports approximation of an arbitrary degree. It represents a good tradeoff between model expressiveness and complexity, while higher degree approximations often over-fit the data. Based on this, we propose three novel feature mappings to efficiently approximate the degree-two polynomials; see Section 6.3.

Our methods are evaluated on several real-world large sparse datasets. When classifying medium-sized sparse data (with $< 10^5$ instances), our method can achieve similar accuracy to that of the KARMA kernel classifier. For large scale classification problems where it is too expensive to compute the KARMA kernel, our methods scale efficiently to datasets of 10^7 data points with over 10^5 sparse features. Compared to the original KARMA algorithm, our method is also much more efficient in terms of memory usage. Training such large scale datasets can be done within several hours on a normal desktop computer. Most importantly, as our methods can learn from both the linear and non-linear (low-rank) structures in the data, it leads to significant accuracy improvement over the linear classifiers; see Section 6.4. For the above reasons, our method is well-suited for classifying large sparse datasets in real-world applications.

The rest of the chapter is organized as follows. In Section 6.2, we describe the problem setting and review previous works of classifying sparse data. In Section 6.3, we provide empirical studies on the KARMA classifier with real-world large sparse datasets. We further describe our novel feature mappings in details. Section 6.4 describes our experiments. Section 6.5 concludes this chapter.

6.2 Classify Large Sparse Data

In this section, we introduce the problem setting of classifying large sparse data. We review several techniques of classifying datasets with missing values.

6.2.1 Problem Formulation

Suppose $p(\mathbf{x}, y)$ is the underlying distribution for a fully observed dataset $\mathcal{S}_{dense} = \{\mathbf{x}^{(i)}, y^{(i)}\}_{i=1...N}$, where $\mathbf{x} = [x_1, x_2, \ldots, x_d] \in \mathbb{R}^d$ is the input vector and $y \in \mathcal{Y}$ is a one-dimensional label. We study the case where the majority of the feature values in $\mathbf{x}$ are missing. In other words, instead of learning from the dataset $\mathcal{S}_{dense}$, the learner is only given an extremely sparsified dataset $\mathcal{S}_{sparse} = \{\mathbf{x_o}^{(i)}, y^{(i)}\}_{i=1...N}$ where $\mathbf{o} = [o_1, o_2, \ldots, o_d] \in [0, 1]^d$ a is a d-dimensional observability vector indicating whether each feature is missing or not.

Using $*$ to denote a missing value, we have that $\mathbf{x_o} \in \{\mathbb{R} \cup \{*\}\}^d$, in particular,

$$(\mathbf{x_o}^{(i)})_j = \begin{cases} x_j^{(i)} & \text{if } o_j^{(i)} = 1 \\ * & \text{if } o_j^{(i)} = 0 \end{cases} . \tag{6.1}$$

Given a Lipschitz loss function $l(\cdot)$, the goal of classifying large sparse data is to learn from $\mathcal{S}_{sparse}$ and induces a solution f^* that has small risk on future (sparse) data:

$$f^* = \min_{f \in \mathcal{H}} \mathbb{E}_{(\mathbf{x_o}, y) \sim p(\mathbf{o}, \mathbf{x}, y)} \Big[l(f(\mathbf{x_o}), y) \Big]. \tag{6.2}$$

We denote the distribution $p(\mathbf{o}|\mathbf{x}, y)$ as the *missing data mechanism*. As mentioned before, according to the missing data theory [56], the dataset is MCAR (Missing Completely At Random) when the observability does not depend on the feature values, i.e., $p(\mathbf{o}|\mathbf{x}, y) = p(\mathbf{o})$; MAR (Missing At Random) if observability only depends on the observed feature values $p(\mathbf{o}|\mathbf{x}, y) = p(\mathbf{o}|\mathbf{x_o}, y)$, and MNAR (Missing Not At Random) in all other cases. The study in this paper does not restrict the missing data mechanism. It is applicable to any of MAR, MNAR or MCAR.

6.2.2 Previous Works

To classify data with missing values, one can simply skip the missing values if using a generative classifier such as naive Bayes. For (discriminative) linear classifiers such as support vector machines, one of the common strategies is to use 0-imputation where all $*$s are replaced with 0s. For 0-imputation, we have

$$(\mathbf{x_o}^{(i)})^{\text{0-imp}} = \mathbf{x}^{(i)} \circ \mathbf{o}^{(i)}, \tag{6.3}$$

where $\circ$ denotes element-wise product.

Apart from imputation, Chechik et al. [9] propose to maximize the classifier margin computed according to geometries of the non-missing dimensions of a particular instance, i.e.:

$$\max_{\mathbf{w}} \left(\min_i \frac{y^{(i)} \mathbf{w} \mathbf{x}^{(i)}}{||\mathbf{w} \circ \mathbf{o}^{(i)}||} \right). \tag{6.4}$$

However, this method has not been tested on the classification tasks of large sparse user-item datasets where the sparsity could be extremely high. In the field of budgeted learning, there are also much work on learning from data with a limited number of observed features [7, 28, 51, 6]. However, they all assume that the learner can actively choose which features to observe, which is different from our setting.

The success of low-rank matrix factorization algorithms [79, 49] on many collaborative filtering tasks suggest that many large sparse datasets have a low rank structure. To leverage this low-rank property, Goldberg et al. [25] formulate missing data classification as a matrix completion task where the class label is treated as an additional column to the sparse input matrix. This formulation can handle multi-label classification, semi-supervised classification and matrix completion at the same time. However, instead of optimizing the classification loss, this model needs to optimize the matrix completion loss as an indirect step.

To address this challenge, Hazan et al. [29] have proposed a learning framework that only assumes the input $\mathbf{X}$ is of low-rank. By studying the polynomial approximation of the linear models in the low-rank data space, authors provide a learning framework for classifying low-rank data that directly optimizes classification loss. Here we briefly review their results.

(Theorem 1 [29]) When distribution $\mathcal{D}$ is of low rank, let $\Gamma = \sum_{j=1}^{\gamma} d^j$ be the number of unique sequences s of feature index $\{1, 2, \ldots, d\}$ with length $1 \leq |s| \leq \gamma$. Using γ-degree polynomial approximation to the inner product $(\mathbf{w} \cdot \mathbf{x})$ in the low-rank space, authors have proved that a certain feature mapping $\phi_\gamma : \{\mathbb{R} \cup \{*\}\}^d \to \mathbb{R}^\Gamma$ can be constructed, such that there exists a classifier $\mathbf{v}^*$ in the ambient space $\mathbb{R}^\Gamma$ which satisfies

$$\mathbb{E}[l(\mathbf{v}^* \cdot \phi_\gamma(\mathbf{x_o}), y)] \leq \min_{\mathbf{w}} \mathbb{E}[l(\mathbf{w} \cdot \mathbf{x}, y)] + \epsilon. \tag{6.5}$$

In other words, the classifier $\mathbf{v}^*$ can compete with the best linear classifier that accesses the full data during training.

Instead of computing ϕ_γ directly, which is extremely high-dimensional, its inner product can be computed using the following kernel trick (Theorem 2 [29]):

$$\langle \phi_\gamma(\mathbf{x_o}^{(i_1)}), \phi_\gamma(\mathbf{x_o}^{(i_2)}) \rangle = k_\gamma(\mathbf{x_o}^{(i_1)}, \mathbf{x_o}^{(i_2)}) = \frac{|\mathbf{o}^{(i_1)} \cap \mathbf{o}^{(i_2)}|^\gamma - 1}{|\mathbf{o}^{(i_1)} \cap \mathbf{o}^{(i_2)}| - 1} \langle \mathbf{x_o}^{(i_1)}, \mathbf{x_o}^{(i_2)} \rangle, \tag{6.6}$$

where $|\mathbf{o}^{(i_1)} \cap \mathbf{o}^{(i_2)}|$ denotes the size of common non-missing features of $\mathbf{x_o}^{(i_1)}$ and $\mathbf{x_o}^{(i_2)}$. Based on this kernel trick, Hazan et al. [29] further provide a kernelized stochastic gradient descent learning algorithm, KARMA, to find the optimal solution in the space of ϕ_γ.

Restricted by the time and space complexity of kernel computation, KARMA is not scalable on large sparse datasets. Meanwhile, notice that Eq. (6.6) is essentially a dot product kernel in

the ultra-high dimensional ($\Gamma = \sum_{j=1}^{\gamma} d^j$) space. It is too expensive even to compute its random feature approximation [73]. For example, to get N random features for a dot-product kernel, we need to generate $d \times N$ random numbers [45].

Next, we describe our method which benefits from the theoretic results of Hazan et al. [29] while skipping the kernel computation step. Our approach is motivated by the following thought:

Given that the dimensionality of ϕ_γ is too high to compute directly, is it possible to approximate ϕ_γ using relatively lower dimensional feature mappings which we can compute and store explicitly?

6.3 Approximate Feature Mappings

To approximate ϕ_γ, we first review its definition (Definition 4 [29]). Given γ, each dimension of ϕ_γ encodes a unique sequence s of feature index $\{1, 2, \ldots, d\}$, $1 \leq |s| \leq \gamma$:

$$\left(\phi_\gamma(\mathbf{x_o})\right)_s = \begin{cases} \mathbf{x}_{s_{end}} & \text{if } s \in \mathbf{o} \\ 0 & \text{else} \end{cases}, \tag{6.7}$$

where s_{end} denotes the last element of s. We could see that when $\gamma = 1$, $\phi_1(\mathbf{x_o}) = (\mathbf{x_o})^{\text{0-imp}}$; while ϕ_2 is the concatenation of $\phi_1(\mathbf{x_o})$ and the co-occurrence information of each feature pair:

$$\phi_2(\mathbf{x_o}) = [\phi_1(\mathbf{x_o}), A(\mathbf{x_o})], \tag{6.8}$$

where the operator $A : \{\mathbb{R} \cup \{*\}\}^d \rightarrow \mathbb{R}^{d^2}$ denotes the flattening of outer-product:

$$A(\mathbf{x_o}) := ((\mathbf{x_o})^{\text{0-imp}} \otimes \mathbf{o})_{\text{flatten}}, \tag{6.9}$$

where $\otimes$ denotes the outer product of two vectors. Equation (6.9) encodes the second order (co-occurrence) information among the non-missing features. For example, for data point $\mathbf{x_o} = [2 * 3]$, we have $\phi_2(\mathbf{x_o}) = [203202000303]$. where $\phi_2(\mathbf{x_o})_{[1:3]} = (\mathbf{x_o})^{\text{0-imp}}$ and $\phi_2(\mathbf{x_o})_{[4:12]}$ equals the flattening of $[203] \otimes [101]$. It is obvious that larger γ provides higher learning capacity but also increases the risk of over-fitting.

To see how the polynomial degree γ affects classification performance on sparse data, we use several real-world benchmark datasets for empirical evaluation, which have been introduced in Table 1.1. Again, the classification task is to classify user gender from movie rating vectors (*ml100k, ml1m*[1]*, ymovie*[2]) or age according to grocery shopping records (*tafeng*[3]). We

[1] http://grouplens.org/datasets/movielens/

[2] http://webscope.sandbox.yahoo.com

[3] http://recsyswiki.com/wiki/Grocery_shopping_datasets

Table 6.1: Best test performance with different γ values. Each dataset is split as 4:1 for training and testing. The last column indicates the percentage of hyper-parameters for which Algorithm 3 solved the Kernel SDCA [82] optimization problem.

Data	$\gamma = 1$	$\gamma = 2$	$\gamma = 3$	$\gamma = 4$	$\gamma = 5$	Solved
ml100k	0.8349	**0.8459**	0.8202	0.8183	0.8018	19/150
ml1m	0.8416	**0.8501**	0.8457	0.8407	0.8348	18/150
ymovie	0.7883	**0.8383**	0.8380	0.8353	0.8328	44/150
tafeng	0.7423	**0.7680**	0.7567	0.7391	0.7306	150/150

use hinge loss with $l2$-regularization in our experiments. For fast optimization, we use the kernelized version of Stochastic Dual Coordinate Ascent [82]. We compute the generalization performance under a series of parameter settings, i.e., $\gamma = \{1.0, 2.0, 3.0, 4.0, 5.0\}$ and regularization parameter $C = \{2^{-15}, 2^{-14}, \ldots, 2^{14}, 2^{15}\}$. This is a parameter grid of 150 vertices. To speedup the exhaustive grid search process, we use the approximate parameter path tracking approach [5]. The basic idea is that when a dual solution α found by SDCA is optimal (with a duality-gap less than ϵ) at regularization parameter C_1, then $\frac{C_2}{C_1}\alpha$ will also be an optimal solution for the problem at C_2 if C_2 is in the vicinity of C_1. The pseudo code in Algorithm 3 implements this idea for hyper-parameter selection of KARMA. With this approximate tracking procedure, we only need to solve the SDCA problems for a small number of grid vertices; see the last column of Table 6.1. [4]

An important observation from these real-world experiments is that $\gamma = 2.0$ gives significant performance boost compared to 0-imputation ($\gamma = 1.0$). Meanwhile, in many of our experiments, $\gamma > 2.0$ leads to over-fitting rather than better generalization performance. While it is too early to conclude that $\gamma = 2.0$ is always the best setting, it is promising that ϕ_2 could help improve classification accuracy in many tasks. Besides, as introduced in the previous section, higher γ corresponds to a more complex model structure and thus is harder to compute or approximate efficiently. Because of the above reasons, we focus on degree-two polynomial approximation, which is promising to provide a good trade-off between model expressiveness and complexity.

However, even ϕ_2 still has $d + d^2$ dimensions, which is intractable for large datasets, where the feature dimension $d > 10^5$. Next we discuss several practical strategies to construct a sufficiently compact feature mapping to approximate ϕ_2.

Notice that the first d-dimensions in ϕ_2 equals to the 0-imputation of the original feature

[4] the *tafeng* dataset is an exception, for which the approximate path does not lead to any speedup

Algorithm 3 Approximate Path Tracking to evaluate the Generalization Performance of KARMA

Input: matrices $A_{[\cdot,\cdot]}$, $G_{[\cdot,\cdot]}$ for storing the solution and duality gap at each grid vertex;

Output: $A_{[\cdot,\cdot]}$, optimal solutions for all grid vertices

```
(γ', C') ← (γ_min, C_min); /*the next grid vertex to solve*/
allconverged ← False
while not allconverged do
    Compute KARMA Kernel K_γ';
    A_[γ',C'], G_[γ',C'] ← SolveKernelSDCA(K_γ', y, C') until duality gap < ε/4;
    for every (γ, C) in grid do
        /*scan the whole grid with A_[γ',C']*/
        Update kernel to be K_γ;
        α ← A_[γ',C'] · C/C';
        g ← EvaluateDualGap(α, K_γ, y, C)
        if g > ε then
            (γ', C') ← (γ, C); /*an unconverged solution*/
            Break;
        else if g < G_[γ,C] then
            (G_[γ,C], A_[γ,C]) ← (g, α); /*update the solution*/
        end if
    end for
    allconverged ← True;
end while
return A
```

vector, which needs no computation and is tractable for storage. We only need to approximate the d^2 dimensions that encode the second-order terms in the polynomial (feature co-occurrence information). We propose the following strategies.

6.3.1 Density-based strategy

For many real-world large sparse data, an important observation is that density often follows a long-tailed distribution, so that a small fraction of features contribute the majority of non-missing values. Naturally, a large number of co-occurrence information would be contributed by a small number of dense features only.

Let $F \subseteq \{j_1, j_2, \ldots, j_k\}$ be the top-k features in terms of data density. We propose to approximate the second order terms $A(\cdot)$ in $\phi_2(\cdot)$ by only representing the co-occurrence among the k dense features, i.e., we approximate ϕ_2 (see Eq.(6.8)) by

$$\tilde{\phi}_2(\mathbf{x_o}) = [(\mathbf{x_o})^{\text{0-imp}}, A((\mathbf{x_o})_F)]$$

$$= [(\mathbf{x_o})^{\text{0-imp}}_{-F}, \phi_2((\mathbf{x_o})_F)], \tag{6.10}$$

where $-F$ denotes features except for F, and $\phi_2((\mathbf{x_o})_F) = [(\mathbf{x_o})^{\text{0-imp}}_{F}, A((\mathbf{x_o})_F)]$ is the degree-two polynomial approximation to the low rank space F. The intuition is that we care more about the low-rankness in the denser features because learning from the extremely sparse features may not be worthwhile in terms of information gain v.s. computational cost. The procedure is also described in Algorithm 4.[5]

Algorithm 4 Feature Mapping with Density-based Strategy

Input: $\mathbf{x_o}$, the original input vector (sparse);

Input: k, the dimensionality of the selected subspace;

Select top-k dense features to form the subspace F;

Compute $A((\mathbf{x_o})_F)$ and $\tilde{\phi}_2(\mathbf{x_o})$ using (6.9) and (6.10);

Use $\tilde{\phi}_2(\mathbf{x_o})$ to train / test with a linear classifier;

According to Theorem 1 and 2 of [29], the classifier trained with $\tilde{\phi}_2(\mathbf{x_o})$ can still compete with the semi-oracle classifier that can observe all feature values in the subspace F without any missing. (the subspace of the less dense features $-F$ are handled with 0-imputation.)

With this strategy, the overall dimension of $\tilde{\phi}_2$ will be $d+k^2$ and can be computed efficiently in $O(k^2_{nnz})$ time using simple outer product operations, where k_{nnz} denotes the average number of non-zeroes in the selected k features. The memory consumption will be $O(d_{nnz}+k^2_{nnz})$. When $k \ll d$, it is possible to directly compute and store $\tilde{\phi}_2$.

6.3.2 Feature-selection strategy

Alternatively, we could also select the subspace F using feature selection algorithms. In our current implementation, we simply rank the feature importance according to $|w_j|$, where $\mathbf{w}$ is the weight vector of a pre-trained linear SVM. More advanced feature selection algorithm is possible to lead to better performance, which can be studied in future work.

[5]The algorithm for the other two feature-mapping strategies are very similar. We omit for brevity.

6.3.3 Clustering-based strategy

Finally, instead of selecting a small subspace for low-rank learning, we could also aggregate the sparse and high dimensional data space to form the more compact subspace. In this clustering-based strategy, we use K-means clustering [43] to aggregate the d features into k feature clusters, which forms the space F of cluster-level features. We set the value of each cluster-level feature as the mean of its member features; see Figure 6.1. This method is also intuitive because latent item clusters do exist in many large sparse data (e.g., different episodes of a TV opera), several previous methods [53, 91, 66] have successfully exploited this observation.

With this strategy, F is no longer a subspace of the original feature space. However, same as the previous two strategies, using $\tilde{\phi}_2$ (Eq. (6.10)) can still take advantage of the low-rank structure of space F according to the framework of [29].

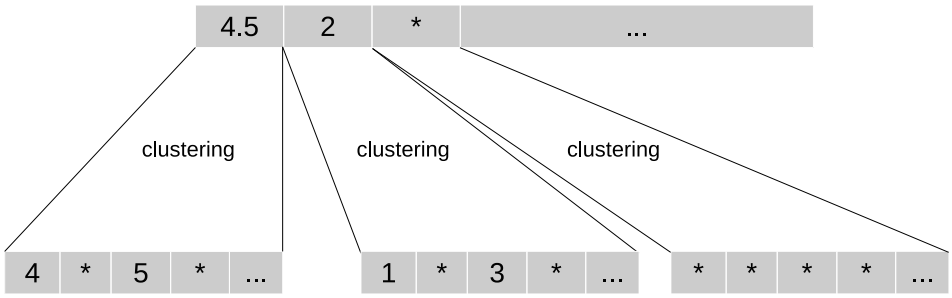


Figure 6.1: A graphical illustration of the clustering-based feature mapping strategy. $*$ denotes missing. We set the value of each cluster-level feature as the mean of its member features.

6.3.4 Combining feature mapping strategies

The aforementioned strategies use feature selection or clustering to form a low-dimensional space F and approximate the second-order information $A(\mathbf{x_o})$ with $A((\mathbf{x_o})_F)$. It is possible that the space formed by different strategies can capture complementary information. For this reason, we also propose to combine different feature mapping strategies, which could be implemented simply by concatenating their corresponding subspaces when computing F. In our experiments, we will show that combining the density-based strategy with the clustering-based strategy helps improve classification performance.

In the next section, we compare the predictive performance of these different strategies.

6.4 Experiments

We use the large sparse real-world datasets described previously in Table 1.1. For simplicity, all our datasets address binary classification. Apart from the *ymovie*, *ml-1m*, *ml-100k* and *tafeng* datasets introduced in the previous section, we use *book* [93] dataset to predict user age from highly-sparse book rating records. For the *epinions* [65] dataset, we classify each user as trustworthy or not based on the rating records. For the *flickr* [8] dataset, we classify whether a photo is heavily commented according to the likes given by a large population of users. Apart from the whole *flickr* data, we also populate the *flickr-5m* and *flickr-500k* data by randomly subsampling 5 million and 500,000 data instances, respectively.

To deal with imbalanced data, we use under-sampling to reduce the over-represented class. We split each balanced dataset into 6:1:3 as training, validation and test set. The validation set is used to tune the best hyper-parameters. For the regularization parameter C, we use grid search within $[10^{-5}, 10^{-4}, \ldots, 10^4, 10^5]$ and we tune the best γ within $[1, 2, 3, 4, 5]$. We repeat the whole process (undersampling, splitting, training, parameter-tuning and testing) for 10 times with different random seeds.

We evaluate the classification accuracy of the feature mappings proposed in Section 6.3, which include *DenseFM, WeightFM, ClusterFM* and a mixed strategy (*MixFM*) that combines *DenseFM* and *ClusterFM*[6]. For the baseline methods, we consider linear classification with 0-imputation, geometric margin adjustment (geom) [9] and the KARMA kernel classifier [29].

All our experiments are carried on a desktop computer with 24 GB of memory. Since the feature selection strategies are simple and efficient, we try to make k as large as possible so long as it fits into the memory. For *DenseFM* and *WeightFM* strategies we select the dimension of F to be $k = 40000$. For the *ClusterFM* strategy the number of clusters is set as 1000 for efficient clustering.

With these settings, the memory consumption for each method is given in Table 6.3. For KARMA, we need to store the kernel matrix which is $O(n^2)$. For other methods, we only need to store the non-zero entries in the feature or feature mapping vectors. It is obvious that it is impractical to compute and store the KARMA kernel for the larger datasets, while the proposed feature mappings are much more memory-saving.

All classifiers are implemented with the SDCA optimizer [82], and the duality gap $\epsilon < 10^{-4}$ is used as the stopping condition. For smaller datasets, we evaluate all classification methods mentioned above. For datasets with more than 10^5 instances, it is computationally expensive

[6]Notice that we have also tried combining *ClusterFM* and *WeightFM*, the accuracy and time/space complexity is similar.

to compute the KARMA kernel matrices. Thus we do not evaluate KARMA classifier on those large-scale data. The averaged test accuracy[7] is shown in Table 6.2. For the KARMA experiments, we only record the best performance among different settings of γ. We have several experiment observations:

Observation 1. On the small to medium-sized data (*ml100k, ml1m, ymovie* and *tafeng*) where it is possible to compute the KARMA kernel, the proposed feature mapping strategies can achieve similar performance to that of KARMA. For large scale classification tasks where we cannot compute the KARMA kernel matrix (*book, epinions, flickr*), all our feature mapping strategies significantly outperform 0-imputation and the Geom algorithm, especially for the *MixFM* strategy.

Table 6.2: Testing accuracy on different large sparse data. Each dataset is split as 6:1:3 for training, validation and testing. The best accuracy is in **bold** while the second best is marked with *. Underlined results indicate significantly better than linear classification under McNemar's test ($p = 0.05$). We have not evaluated KARMA on datasets with more than 10^5 instances, because it is too expensive to store the kernel matrix.

Datasets	*0-imp*	*Geom*	*KARMA*	*DenseFM*	*WeightFM*	*ClusterFM*	*MixFM*
ml-100k	0.7806	0.7667	0.7885*	0.7739	0.7752	**0.7909**	0.7703
ml-1m	0.8150	0.8156	**0.8219**	0.8216*	0.8146	0.8160	0.8161
ymovie	0.7736	0.7484	**0.7976**	0.7913	0.7826	0.7732	0.7956*
tafeng	0.7158	0.7038	0.7333	0.7390*	**0.7391**	0.7192	0.7354
book	0.7276	0.6881	**0.7355**	0.7308	0.7297	0.7309	0.7337*
epinions	0.7211	0.6960	-	0.7308	0.7301	0.7548*	**0.7664**
flickr-500k	0.6645	0.6278	-	0.6657	0.6659	0.6873*	**0.6937**
flickr-5m	0.7090	0.6455	-	0.7231*	0.7209*	0.7191	**0.7447**
flickr-all	0.7164	0.6477	-	0.7480*	0.7437	0.7263	**0.7659**

Observation 2. Comparing different feature mapping methods, we notice that *DenseFM* and *WeightFM* have very similar behavior in terms of classification accuracy, runtime and space complexity. This is no surprise, as both of them select subsets of features to form the low dimensional space. Meanwhile, *ClusterFM* does behave differently. Finally, *MixFM* leads to the best performance especially on the larger datasets. This indicates that *DenseFM* and *ClusterFM* do provide complementary information. For this reason, we recommend *MixFM* to

[7]Notice that the accuracy for KARMA on the smaller datasets is different from the results in Table 6.1. The reason is that in Table 6.1, we do not need a held-out set for tuning the hyper-parameter, so we split each data into $1:4$ training and testing, instead of $6:1:3$.

be the first-choice in real-world applications.

Observation 3. As shown in Table 6.3, the proposed feature mapping strategies need far less memory than the KARMA kernels, which makes them tractable using a normal desktop computer even for the largest dataset *flickr-all* with more than ten million instances.

Table 6.3: Memory consumption (in **millions** of nonzero data entries) on the large datasets. For KARMA, we need to store the kernel matrices. For other methods, we only need to store the non-zeroes in the feature (mapping) vectors.

Datasets	*0-imp & Geom*	*KARMA*	*DenseFM*	*WeightFM*	*ClusterFM*	*MixFM*
book	0.25	604.9	53.6	10.9	1.7	55.3
epinions	0.95	2178.6	71.5	41.5	10.5	82
flickr-500k	1.8	1.15×10^5	40.0	39.5	4.7	44.7
flickr-5m	17.7	1.15×10^7	358.4	369.4	46.1	404.5
flickr-all	31.5	3.6×10^7	634.5	654.1	81.9	716.4

Observation 4. Notice that all methods can be solved with stochastic linear optimizer such as SDCA and SGD which scales linearly with the number of non-zeroes in a feature or feature mapping vector. Compared to 0-imputation, the proposed methods do need the extra time to pre-compute the feature mappings. However, as can be seen in Table 6.4, the proposed methods are able to train with the largest dataset within several hours, which is practical.

Table 6.4: Training time in seconds (including feature mapping computation). The proposed methods can handle the *flickr-all* dataset (more than 10^7 instances) within several hours of training.

Datasets	*0-imp*	*Geom*	*KARMA*	*DenseFM*	*WeightFM*	*ClusterFM*	*MixFM*
book	0.1	0.1	-	30	30	280	310
epinions	0.3	0.3	-	90	70	1.0×10^3	1.0×10^3
flickr-500k	2	1	-	190	200	3.5×10^3	3.6×10^3
flickr-5m	25	12	-	1.9×10^3	2.0×10^3	5.3×10^3	6.4×10^3
flickr-all	46	33	-	3.1×10^3	3.2×10^3	7.0×10^3	8.2×10^3

Observation 5. Comparing different feature mapping strategies, the *ClusterFM* method uses a much smaller k value than the *DenseFM* and *WeightFM* while achieves comparable results. This indicates that the subspace constructed from clustering is more informative than simply selecting subsets of features. However, *ClusterFM* requires a clustering step which involves a considerable amount of computation.

To summarize, the above observations indicate that the proposed feature mappings are efficient and effective for classifying large sparse datasets in practice.

6.5 Conclusion

In this chapter, we have developed an efficient and effective method for classifying large sparse datasets. Our method efficiently approximates the KARMA kernel [29] which can learn from the low-rank structure of a large sparse dataset but does not scale well to very large datasets. To achieve this, we first analyze the empirical performance of KARMA, and find that the degree-two polynomials often lead to superior performance. We then propose several novel scalable feature mappings to efficiently approximate the degree-two polynomials. In experiments, our method is comparable with KARMA on medium-sized data and scales well to larger datasets that KARMA does not. Our method also significantly outperforms linear classifiers on datasets of various sizes. Overall, our method is well-suited for classifying large sparse datasets in real world applications.

Chapter 7

Conclusion

Large and sparse datasets are pervasive in this big data era. Classifying these data effectively and efficiently is very important in many applications. In this thesis, we have studied how data sparsity would affect the convergence behavior of classification and how we could learn the non-linear structure in the large sparse data efficiently. In this final chapter, we give concluding remarks and indicate possible future directions.

7.1 Summary of research questions and results

To summarize, this thesis has addressed two important questions related to classifying sparse data in scale, which were proposed at the beginning of this thesis.

1. Sparse data and convergence behavior. *How different properties of a dataset, such as the sparsity rate and the mechanism of missing data systematically affect convergence behavior of classification?*

This is the major topic of this thesis. In Chapter 3, we have focused on the data sparsity issue in linear SVM, which is a popular algorithm for classifying large sparse datasets. In order to understand how data sparsity affects the convergence behavior of linear SVM, we have proposed a novel approach to generate large and sparse data from real-world datasets using statistical inference and the data sampling process of the PAC framework. This method allows us to have arbitrarily large synthetic data efficiently and also enables us to study the learning behavior systematically. From the experiments we have observed: 1. Higher sparsity will lead to larger asymptotic generalization error rate, while convergence rate of learning is almost unchanged for different sparsity. We have also proved these findings theoretically.

We then move on to study the convergence behavior of naive Bayes on large sparse datasets.

Naive Bayes has good scalability on large sparse datasets and sometimes even outperform linear classifiers. In Chapter 4, we continue to use generated datasets to explore how different mechanisms of missing data, data sparsity and the number of attributes systematically affect the AUC learning curve of naive Bayes. We have considered several realistic Missing At Random mechanisms for binary large and sparse data. We propose the corresponding data generation methods using uniform dilution (BTE, JE) and also probabilistic modeling (G_{PMF}). We have generated very large synthetic datasets for each missing mechanism and study the entire learning curve behavior when systematically changing data sparsity and the attribute dimension. Several useful observations have been made, and we have provided in-depth theoretic studies for those observations. We have also provided an intuitive decision flowchart and a guideline on how our results could be used in practice.

In Chapter 5 of the thesis, we further extend our study on naive Bayes model by considering all discriminative linear classifiers and generalized loss functions. Moreover, we also extend the missing data mechanisms and data generation methods from binary input to real-valued input. With real-world and synthetic experiments, we have observed several meaningful learning curve behaviors under different missing data mechanisms. We further prove all our observations theoretically. Our results provide a practical guideline to determine if or when obtaining more data and/or obtaining missing values in the data is worthwhile or not. This can be very valuable in applications of classifying large sparse datasets.

2. Handling sparse data with non-linear model. *How to efficiently learn non-linear data structures when classifying large sparse data?*

As the second topic of this thesis, we consider how to build a better model to improve the classification performance of a given large sparse dataset. Traditionally, we often use linear models to classify these large-scale and high-dimensional datasets for the sake of scalability. However, the predictive accuracy may suffer because of linear classifiers cannot capture the non-linearity in the data. Previous studies of the collaborative filtering problem suggest that large sparse datasets often have a low-rank structure. To exploit this low-rankness and achieve a higher classification accuracy, Hazan et al. [29] have developed a kernel algorithm (KARMA) for classifying such datasets by finding the polynomial approximation to the low rank space. However, KARMA does not scale well to large datasets because of the expensive kernel computation. In Chapter 6 of this thesis, we have developed novel scalable feature mappings to efficiently approximate the kernels used in KARMA. In experiments, our method is comparable with KARMA on medium-sized data and scales well to larger datasets that KARMA does not. Our method also significantly outperforms linear classifiers on datasets of various

sizes. Overall, our method finds a good trade-off between effectiveness (better accuracy) and scalability, which can be useful for classifying large sparse data in practice.

7.2 Suggestions of future directions

Finally, we identify possible future research related to classifying large sparse data.

- *Active Learning for classifying large sparse data*. Our studies (Chapter 3-5 of this thesis) indicate that reducing sparsity often helps in learning large sparse data. However, in real-world applications, either data collection or imputation comes at a cost. It is thus important to know which data value to collect/impute to be most efficient, for example, when dealing with the *cold-start users* who have very few feedbacks, the recommendation system has to decide which item query to send. Previous work [42, 24] has applied active learning to classification with partially observed data. However, for large and sparse data which are high-dimensional, we may have new challenges for applying such active learning methods, e.g., for each user the number of candidate values to be queried is huge but many of them are invalid since users only have limited knowledge. Recent studies of online Matrix Factorization [92, 33] have demonstrated effective and efficient ways of interactively and sequentially query the missing values to improve recommendation. It is interesting to explore whether similar strategies could be used to improve classification accuracy.

- *Kernel Methods for classifying large sparse data*. Chapter 6 of our thesis has proposed efficient algorithm of learning low-rank structures in the data. Meanwhile, it is also useful to explore whether learning other non-linear structures would further improve accuracy of large sparse data classification. For example, the commonly used non-linear kernels such as Gaussian kernel, Laplacian kernel and so on. Recent studies [13, 35] have shown that it is possible to scale-up kernel methods to datasets with a large number of samples. It is interesting to study whether these methods help improve the classification of large sparse data. However, the scalability of these methods is still limited by the very high feature dimensionality. It is meaningful to find new kernel approximation algorithms to scale only with the number of non-missing features rather than the total number of feature dimensions.

Bibliography

[1] Peter Bartlett and John Shawe-Taylor. Generalization performance of support vector machines and other pattern classifiers. *Advances in Kernel Methods¢Support Vector Learning*, pages 43–54, 1999.

[2] Peter L Bartlett, Michael I Jordan, and Jon D McAuliffe. Large margin classifiers: Convex loss, low noise, and convergence rates. In *NIPS*, 2003.

[3] James Bennett and Stan Lanning. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35, 2007.

[4] Pablo Bermejo, José A Gámez, and José M Puerta. Speeding up incremental wrapper feature subset selection with naive bayes classifier. *Knowledge-Based Systems*, 55:140–147, 2014.

[5] Katharina Blechschmidt, Joachim Giesen, and Soeren Laue. Tracking approximate solutions of parameterized optimization problems over multi-dimensional (hyper-) parameter domains. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 438–447, 2015.

[6] Brian Bullins, Elad Hazan, and Tomer Koren. The limits of learning with missing data. In *Advances in Neural Information Processing Systems*, pages 3495–3503, 2016.

[7] Nicolo Cesa-Bianchi, Shai Shalev-Shwartz, and Ohad Shamir. Efficient learning with partially observed attributes. *The Journal of Machine Learning Research (JMLR)*, 12:2857–2878, 2011.

[8] Meeyoung Cha, Alan Mislove, and Krishna P Gummadi. A measurement-driven analysis of information propagation in the flickr social network. In *Proceedings of the 18th international conference on World Wide Web (WWW)*, pages 721–730. ACM, 2009.

[9] Gal Chechik, Geremy Heitz, Gal Elidan, Pieter Abbeel, and Daphne Koller. Max-margin classification of data with absent features. *The Journal of Machine Learning Research (JMLR)*, 9:1–21, 2008.

[10] Ning Chen, Jun Zhu, Jianfei Chen, and Ting Chen. Dropout training for svms with data augmentation. *arXiv preprint arXiv:1508.02268*, 2015.

[11] Ning Chen, Jun Zhu, Jianfei Chen, and Bo Zhang. Dropout training for support vector machines. In *Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI)*, 2014.

[12] Corinna Cortes and Mehryar Mohri. Auc optimization vs. error rate minimization. *Advances in neural information processing systems*, 16(16):313–320, 2004.

[13] Bo Dai, Bo Xie, Niao He, Yingyu Liang, Anant Raj, Maria-Florina F Balcan, and Le Song. Scalable kernel methods via doubly stochastic gradients. In *Advances in Neural Information Processing Systems*, pages 3041–3049, 2014.

[14] Brian Dalessandro, Daizhuo Chen, Troy Raeder, Claudia Perlich, Melinda Han Williams, and Foster Provost. Scalable hands-free transfer learning for online advertising. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1573–1582. ACM, 2014.

[15] Sofie De Cnudde, Julie Moeyersoms, Marija Stankova, Ellen Tobback, Vinayak Javaly, David Martens, et al. Who cares about your facebook friends? credit scoring for microfinance. Technical report, 2015.

[16] Ofer Dekel, Ohad Shamir, and Lin Xiao. Learning to classify with missing and corrupted features. *Machine learning*, 81(2):149–178, 2010.

[17] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[18] Pedro Domingos and Michael Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine learning*, 29(2-3):103–130, 1997.

[19] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research (JMLR)*, 12:2121–2159, 2011.

[20] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. John Wiley & Sons, 2012.

[21] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research (JMLR)*, 9:1871–1874, 2008.

[22] Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

[23] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[24] Tianshi Gao and Daphne Koller. Active classification based on value of classifier. In *Advances in Neural Information Processing Systems*, pages 1062–1070, 2011.

[25] Andrew Goldberg, Ben Recht, Junming Xu, Robert Nowak, and Xiaojin Zhu. Transduction with matrix completion: Three birds with one stone. In *Advances in neural information processing systems (NIPS)*, pages 757–765, 2010.

[26] David J Hand and Keming Yu. Idiot's bayes not so stupid after all? *International statistical review*, 69(3):385–398, 2001.

[27] James A Hanley and Barbara J McNeil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36, 1982.

[28] Elad Hazan and Tomer Koren. Linear regression with limited observation. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, pages 807–814, 2012.

[29] Elad Hazan, Roi Livni, and Yishay Mansour. Classification with low rank and missing data. *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 257–266, 2015.

[30] Daniel F Heitjan and Srabashi Basu. Distinguishing missing at random and missing completely at random. *The American Statistician*, 50(3):207–213, 1996.

[31] Jose M Hernandez-lobato, Neil Houlsby, and Zoubin Ghahramani. Probabilistic matrix factorization with non-random missing data. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pages 1512–1520, 2014.

[32] Jose M Hernandez-lobato, Neil Houlsby, and Zoubin Ghahramani. Stochastic inference for scalable probabilistic modeling of binary matrices. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, pages 379–387, 2014.

[33] Neil Houlsby, Jose M Hernandez-lobato, and Zoubin Ghahramani. Cold-start active learning with robust ordinal matrix factorization. In *Proceedings of the 31st International Conference on Machine Learning (ICML)*, 2014.

[34] Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S Sathiya Keerthi, and Sellamanickam Sundararajan. A dual coordinate descent method for large-scale linear svm. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 408–415. ACM, 2008.

[35] Cho-Jui Hsieh, Si Si, and Inderjit Dhillon. A divide-and-conquer solver for kernel support vector machines. In *International Conference on Machine Learning*, pages 566–574, 2014.

[36] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 263–272. Ieee, 2008.

[37] Kashif Javed, Haroon Atique Babri, and Mehreen Saeed. Feature selection based on class-dependent densities for high-dimensional binary data. *Knowledge and Data Engineering, IEEE Transactions on*, 24(3):465–477, 2012.

[38] Luis O Jimenez and David A Landgrebe. Supervised classification in high-dimensional space: geometrical, statistical, and asymptotical properties of multivariate data. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 28(1):39–54, 1998.

[39] George H John, Ron Kohavi, Karl Pfleger, et al. Irrelevant features and the subset selection problem. In *ICML*, volume 94, pages 121–129, 1994.

[40] Enric Junqué de Fortuny, David Martens, and Foster Provost. Predictive modeling with big data: Is bigger really better? *Big Data*, 1(4):215–226, 2013.

[41] Enric Junqué de Fortuny, Marija Stankova, Julie Moeyersoms, Bart Minnaert, Foster Provost, and David Martens. Corporate residence fraud detection. In *Proceedings of the*

20th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 1650–1659. ACM, 2014.

[42] Pallika Kanani and Prem Melville. Prediction-time active feature-value acquisition for cost-effective customer targeting. 2008.

[43] Tapas Kanungo, David M Mount, Nathan S Netanyahu, Christine D Piatko, Ruth Silverman, and Angela Y Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE transactions on pattern analysis and machine intelligence*, 24(7):881–892, 2002.

[44] Aloak Kapoor and Russell Greiner. Learning and classifying under hard budgets. In *Machine Learning: ECML 2005*, volume 3720, pages 170–181. Springer Berlin Heidelberg, 2005.

[45] Purushottam Kar and Harish Karnick. Random feature maps for dot product kernels. In *Proceedings of The 15th International Conference on Artificial Intelligence and Statistics(AISTATS)*, 2012.

[46] Dohyun Kim and Bong-Jin Yum. Collaborative filtering based on iterative principal component analysis. *Expert Systems with Applications*, 28(4):823–830, 2005.

[47] Sang-Bum Kim, Kyoung-Soo Han, Hae-Chang Rim, and Sung Hyon Myaeng. Some effective techniques for naive bayes text classification. *Knowledge and Data Engineering, IEEE Transactions on*, 18(11):1457–1466, 2006.

[48] Daphne Koller and Mehran Sahami. Toward optimal feature selection. In *In 13th International Conference on Machine Learning*, 1996.

[49] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

[50] Michal Kosinski, David Stillwell, and Thore Graepel. Private traits and attributes are predictable from digital records of human behavior. *Proceedings of the National Academy of Sciences*, 110(15):5802–5805, 2013.

[51] Doron Kukliansky and Ohad Shamir. Attribute efficient linear regression with distribution-dependent sampling. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 153–161, 2015.

[52] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[53] Bin Li, Qiang Yang, and Xiangyang Xue. Transfer learning for collaborative filtering via a rating-matrix generative model. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 617–624. ACM, 2009.

[54] Xiang Li, Huaimin Wang, Bin Gu, and Charles X Ling. Data sparseness in linear svm. In *Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI)*, pages 3628–3634. AAAI Press, 2015.

[55] Roderick JA Little. Regression with missing x's: a review. *Journal of the American Statistical Association (JASA)*, 87(420):1227–1237, 1992.

[56] Roderick JA Little and Donald B Rubin. *Statistical analysis with missing data*. John Wiley & Sons, 2014.

[57] Philip M Long and Rocco A Servedio. Low-weight halfspaces for sparse boolean vectors. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, pages 21–36. ACM, 2013.

[58] Laurens Maaten, Minmin Chen, Stephen Tyree, and Kilian Q Weinberger. Learning with marginalized corrupted features. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pages 410–418, 2013.

[59] Omid Madani, Daniel J Lizotte, and Russell Greiner. Active model selection. In *Proceedings of the 20th conference on Uncertainty in Artificial Intelligence (UAI)*, pages 357–365. AUAI Press, 2004.

[60] Benjamin M. Marlin, Richard S. Zemel, Sam Roweis, and Malcolm Slaney. Collaborative filtering and the missing at random assumption. In *In Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence (UAI)*, 2007.

[61] David Martens, Foster Provost, and Jessica Clark. Mining massive fine-grained behavior data to improve predictive analytics. *MIS quarterly*, 40(4):869–888, 2016.

[62] Andrew McCallum, Kamal Nigam, et al. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48, 1998.

[63] H Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, et al. Ad click prediction: a view from the trenches. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1222–1230. ACM, 2013.

[64] Edward Meeds, Zoubin Ghahramani, Radford M Neal, and Sam T Roweis. Modeling dyadic data with binary latent factors. In *Advances in neural information processing systems*, pages 977–984, 2006.

[65] Simon Meyffret, Emmanuel Guillot, Lionel Mdini, and Frdrique Laforest. RED: a Rich Epinions Dataset for Recommender Systems. Technical Report RR-LIRIS-2012-014, LIRIS UMR 5205 CNRS/INSA de Lyon/Universit Claude Bernard Lyon 1/Universit Lumire Lyon 2/cole Centrale de Lyon, October 2012.

[66] Nima Mirbakhsh and Charles X Ling. Clustering-based factorized collaborative filtering. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 315–318. ACM, 2013.

[67] Andriy Mnih and Ruslan Salakhutdinov. Probabilistic matrix factorization. In *Advances in neural information processing systems (NIPS)*, pages 1257–1264, 2007.

[68] Shakir Mohamed. *Generalised Bayesian matrix factorisation models*. PhD thesis, University of Cambridge, 2011.

[69] Andrew Y Ng and Michael I Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Advances in neural information processing systems*, 2:841–848, 2002.

[70] Kristiaan Pelckmans, Jos De Brabanter, Johan AK Suykens, and Bart De Moor. Handling missing values in support vector machine classifiers. *Neural Networks*, 18(5):684–692, 2005.

[71] Anita Prinzie and Dirk Van den Poel. Random multiclass classification: Generalizing random forests to random mnl and random nb. In *Database and Expert Systems Applications*, pages 349–358. Springer, 2007.

[72] Foster J Provost, Tom Fawcett, et al. Analysis and visualization of classifier performance: comparison under imprecise class and cost distributions. In *KDD*, volume 97, pages 43–48, 1997.

[73] Ali Rahimi, Benjamin Recht, et al. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, 2007.

[74] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 452–461. AUAI Press, 2009.

[75] Jason D Rennie, Lawrence Shih, Jaime Teevan, David R Karger, et al. Tackling the poor assumptions of naive bayes text classifiers. In *ICML*, volume 3, pages 616–623. Washington DC), 2003.

[76] James Ridgway, Pierre Alquier, Nicolas Chopin, and Feng Liang. Pac-bayesian auc classification and scoring. In *Advances in Neural Information Processing Systems*, pages 658–666, 2014.

[77] Donald B Rubin. Inference and missing data. *Biometrika*, 63(3):581–592, 1976.

[78] Mehran Sahami. Learning limited dependence bayesian classifiers. In *KDD*, volume 96, pages 335–338, 1996.

[79] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 880–887. ACM, 2008.

[80] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John T. Riedl. Application of dimensionality reduction in recommender system – a case study. In *WEBKDD 2000 Workshop*. ACM, 2000.

[81] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, 127(1):3–30, 2011.

[82] Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss. *The Journal of Machine Learning Research (JMLR)*, 14(1):567–599, 2013.

[83] Shai Shalev-Shwartz and Tong Zhang. Accelerated proximal stochastic dual coordinate ascent for regularized loss minimization. *Mathematical Programming*, pages 1–41, 2014.

[84] Qinbao Song, Jingjie Ni, and Guangtao Wang. A fast clustering-based feature subset selection algorithm for high-dimensional data. *Knowledge and Data Engineering, IEEE Transactions on*, 25(1):1–14, 2013.

[85] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research (JMLR)*, 15(1):1929–1958, 2014.

[86] Harald Steck. Training and testing of recommender systems on data missing not at random. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 713–722. ACM, 2010.

[87] Dougal J Sutherland, Barnabás Póczos, and Jeff Schneider. Active learning and search on low-rank matrices. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 212–220. ACM, 2013.

[88] Vladimir N Vapnik. An overview of statistical learning theory. *Neural Networks, IEEE Transactions on*, 10(5):988–999, 1999.

[89] Vladimir Naumovich Vapnik. *Statistical learning theory*, volume 1. Wiley New York, 1998.

[90] Stefan Wager, William Fithian, Sida Wang, and Percy S Liang. Altitude training: Strong bounds for single-layer dropout. In *Advances in Neural Information Processing Systems*, pages 100–108, 2014.

[91] Bin Xu, Jiajun Bu, Chun Chen, and Deng Cai. An exploration of improving collaborative recommender systems via user-item subgroups. In *Proceedings of the 21st international conference on World Wide Web*, pages 21–30. ACM, 2012.

[92] Xiaoxue Zhao, Weinan Zhang, and Jun Wang. Interactive collaborative filtering. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 1411–1420. ACM, 2013.

[93] Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web (WWW)*, pages 22–32. ACM, 2005.

Appendix A

Proofs of Theorems

Proof of Eqs. (3.25, 3.26, 3.27) in Lemma 3.4.1

Proof From (3.16 ,3.17), we notice that:

$$p^{(j)}(\mathbf{x}_{-j}, 0|y, s) + p^{(j)}(\mathbf{x}_{-j}, 1|y, s) = p(\mathbf{x}_{-j}, 0|y) + \cdot p(\mathbf{x}_{-j}, 1|y) \tag{1}$$

A. when Case 1 (3.22) holds, we have

$$\min_{y\in\{+,-\}} p(\mathbf{x}_{-j}, 0|y) = p(\mathbf{x}_{-j}, 0|y_l) \tag{2}$$

(2)(3.18)(3.16) lead to

$$\min_{y\in\{+,-\}} p^{(j)}(\mathbf{x}_{-j}, 0|y, s) = p^{(j)}(\mathbf{x}_{-j}, 0|y_l, s) \tag{3}$$

(2)(3.19) lead to

$$Z(\mathbf{x}_{-j}) = p(\mathbf{x}_{-j}, 0|y_l) + p(\mathbf{x}_{-j}, 1|y_l) \tag{4}$$

(3)(3.20)(1)(4) lead to

$$\Delta_Z(s) = 0$$

B. when Case 2 (3.23) holds, we have

$$\min_{y\in\{+,-\}} p(\mathbf{x}_{-j}, 0|y) = p(\mathbf{x}_{-j}, 0|y_h) \tag{5}$$

(3.16)(3.23) lead to

$$\min_{y\in\{+,-\}} p^{(j)}(\mathbf{x}_{-j}, 0|y, s) = p^{(j)}(\mathbf{x}_{-j}, 0|y_l, s) \tag{6}$$

(5)(3.19) lead to

$$Z(\mathbf{x}_{-j}) = p(\mathbf{x}_{-j}, 0|y_h) + p(\mathbf{x}_{-j}, 1|y_l) \tag{7}$$

(6)(3.20)(1) lead to

$$Z^{(j)}(\mathbf{x}_{-j}, s) = p^{(j)}(\mathbf{x}_{-j}, 0|y_l, s) + p^{(j)}(\mathbf{x}_{-j}, 1|y_l, s)$$

$$= p(\mathbf{x}_{-j}, 0|y_l) + p(\mathbf{x}_{-j}, 1|y_l) \tag{8}$$

(5)(7)(8) lead to

$$\Delta_Z(s) = p(\mathbf{x}_{-j}, 0|y_l) - p(\mathbf{x}_{-j}, 0|y_h) > 0$$

C. when Case 3 (3.24) holds, we know that Eq.(5) still holds.

(3.16)(3.18)(3.24)(3.21) lead to

$$\min_{y\in\{+,-\}} p^{(j)}(\mathbf{x}_{-j}, 0|y, s) = p^{(j)}(\mathbf{x}_{-j}, 0|y_h, s) > 0 \tag{9}$$

(5)(3.19) lead to

$$Z(\mathbf{x}_{-j}) = p(\mathbf{x}_{-j}, 0|y_h) + p(\mathbf{x}_{-j}, 1|y_l) \tag{10}$$

(3.20)(9)(3.16)(3.17)(3.21) lead to

$$Z^{(j)}(\mathbf{x}_{-j}, s) = p^{(j)}(\mathbf{x}_{-j}, 0|y_h, s) + p^{(j)}(\mathbf{x}_{-j}, 1|y_l, s)$$

$$= Z(\mathbf{x}_{-j}) + s \cdot [p(\mathbf{x}_{-j}, 1|y_h) - p(\mathbf{x}_{-j}, 1|y_l)] \tag{11}$$

(3.18)(11) lead to

$$\Delta_Z(s) = s \cdot [p(\mathbf{x}_{-j}, 1|y_h) - p(\mathbf{x}_{-j}, 1|y_l)] > 0$$

Proof of Lemma 4.7.2

Proof From the definition of AUC risk (Eq. (4.13)), we have (after simple algebraic manipulation)

$$\mathcal{R}(\mathcal{D}) = \Pr_{(\mathbf{x},y),(\mathbf{x}',y')\sim\mathcal{D}}\Big\{ \sum_{j=1}^{m} \log \frac{p(x_j|y)p(x'_j|y')}{p(x_j|y')p(x'_j|y)} \leq 0, y \neq y' \Big\} \tag{12}$$

Assuming that the naive independence assumption holds, Eq. (4.13) further becomes

$$\mathcal{R}(\mathcal{D}) = \Pr_{(\mathbf{x},y),(\mathbf{x}',y')\sim\mathcal{D}}\Big\{ \log \frac{p(\mathbf{x}, y)p(\mathbf{x}', y')}{p(\mathbf{x}, y')p(\mathbf{x}', y)} \leq 0, y \neq y' \Big\}$$

$$= \Pr_{(\mathbf{x},y),(\mathbf{x}',y')\sim\mathcal{D}}\Big\{ p(\mathbf{x}, y)p(\mathbf{x}', y') \leq p(\mathbf{x}, y')p(\mathbf{x}', y), y \neq y' \Big\}$$

$$= \sum_{(\mathbf{x},y),(\mathbf{x}',y')} p(\mathbf{x}, y)p(\mathbf{x}', y') \cdot \mathbb{I}\Big\{ p(\mathbf{x}, y)p(\mathbf{x}', y') \leq p(\mathbf{x}, y')p(\mathbf{x}', y), y \neq y' \Big\}$$

$$= \sum_{(\mathbf{x},y),(\mathbf{x}',y')} \mathbb{I}(y \neq y') \min_{(y,y')} p(\mathbf{x}, y)p(\mathbf{x}', y') = \sum_{(\mathbf{x},\mathbf{x}')} \min_{y\in\{+,-\}} p(\mathbf{x}, y)p(\mathbf{x}', -y) \tag{13}$$

where we have leveraged the fact that $(\mathbf{x}, y)$ and $(\mathbf{x}', y')$ live in discrete space $\mathcal{X} \times \{-1, +1\}$. Suppose attribute x_j has chance θ being missing, i.e.:

$$p(x_j = 1|y; \theta) = \theta p(x_j = 1|y) \tag{14}$$

$$p(x_j = 0|y; \theta) = 1 - \theta p(x_j = 1|y) \tag{15}$$

where $p(x_j = 1|y)$ is the original data sparsity on attribute x_j given label y. Now we study the change of AUC risk as:

$$\begin{aligned}\mathcal{R}(\mathcal{D}) &= \sum_{(\mathbf{x}_{-j}, \mathbf{x}'_{-j})} \sum_{(x_j, x'_j)} \min_{y \in \{+,-\}} p(\mathbf{x}, y) p(\mathbf{x}', -y) \\ &= \sum_{(\mathbf{x}_{-j}, \mathbf{x}'_{-j})} \sum_{(x_j, x'_j)} \min_{y \in \{+,-\}} p(\mathbf{x}_{-j}, y) p(\mathbf{x}'_{-j}, -y) p(x_j|y) p(x'_j| - y)\end{aligned} \tag{16}$$

Define

$$Y := \arg\min_{y \in \{+,-\}} p(\mathbf{x}_{-j}, y) p(\mathbf{x}'_{-j}, -y) \tag{17}$$

$$V := p(\mathbf{x}_{-j}, Y) p(\mathbf{x}'_{-j}, -Y) \tag{18}$$

$$\bar{V} := p(\mathbf{x}_{-j}, -Y) p(\mathbf{x}'_{-j}, Y) \tag{19}$$

thus

$$\bar{V} \geq V \tag{20}$$

we further denote

$$P := p(x_j = 1|Y) \tag{21}$$

$$\bar{P} := p(x_j = 1| - Y) \tag{22}$$

$$g(x_j, x'_j) := p(x_j|Y) p(x'_j| - Y) \tag{23}$$

$$\bar{g}(x_j, x'_j) := p(x_j| - Y) p(x'_j|Y) \tag{24}$$

$$Z(x_j, x'_j) := \min\{V g(x_j, x'_j), \bar{V} \bar{g}(x_j, x'_j)\} \tag{25}$$

$$W(\mathbf{x}_{-j}, \mathbf{x}'_{-j}) := \sum_{(x_j, x'_j)} Z(x_j, x'_j) \tag{26}$$

The following holds

$$\bar{g}(0, 1) = g(1, 0) = \theta P(1 - \theta \bar{P}) \tag{27}$$

$$\bar{g}(0, 0) = g(0, 0) = (1 - \theta P)(1 - \theta \bar{P}) \tag{28}$$

$$\bar{g}(1, 1) = g(1, 1) = \theta^2 P \bar{P} \tag{29}$$

$$\bar{g}(1,0) = g(0,1) = \theta\bar{P}(1-\theta P) \tag{30}$$

and

$$\mathcal{R}(\mathcal{D}) = \sum_{(\mathbf{x}_{-j},\mathbf{x}'_{-j})} W(\mathbf{x}_{-j},\mathbf{x}'_{-j}) \tag{31}$$

the proof of lemma 4.7.2 now amounts to prove:

Proposition. $W(\mathbf{x}_{-j},\mathbf{x}'_{-j})$ decreases monotonically with θ .

To verify the validity of this proposition, we discuss all possible configurations of $Z(x_j, x'_j)$ in an exhaustive manner.

Case 0. we consider the special case when $\mathbf{x}_{-j} = \mathbf{x}'_{-j}$. This forces $V = \bar{V}$, and

$$Z(0,1) = Z(1,0) = V\min\{g(0,1), g(1,0)\} \tag{32}$$

$$Z(0,0) = Vg(0,0) = (1-\theta P)(1-\theta\bar{P})V \tag{33}$$

$$Z(1,1) = Vg(1,1) = \theta^2 P\bar{P}V \tag{34}$$

which leads to

$$W(\mathbf{x}_{-j},\mathbf{x}'_{-j}) = V(1-\theta|P-\bar{P}|) \tag{35}$$

and the lemma holds in this special case.

Case 1. when $\mathbf{x}_{-j} \neq \mathbf{x}'_{-j}$, and $Z(0,1) = Vg(0,1) \leq \bar{V}\bar{g}(0,1)$, $Z(1,0) = Vg(1,0) \leq \bar{V}\bar{g}(1,0)$. We have

$$V\theta\bar{P}(1-\theta P) \leq \bar{V}\theta P(1-\theta\bar{P}) \tag{36}$$

$$V\theta P(1-\theta\bar{P}) \leq \bar{V}\theta\bar{P}(1-\theta P) \tag{37}$$

if $V < \bar{V}$, then

$$\theta \leq \min\{\frac{\bar{V}\bar{P} - VP}{P\bar{P}(\bar{V}-V)}, \frac{\bar{V}P - V\bar{P}}{P\bar{P}(\bar{V}-V)}\} \tag{38}$$

if $V = \bar{V}$, then:

$$P = \bar{P} \text{ and } \theta \in [0,1] \tag{39}$$

In either case,

$$W(\mathbf{x}_{-j},\mathbf{x}'_{-j}) = (1-\theta P)(1-\theta\bar{P})V +$$

$$\theta^2 P\bar{P}V + V\theta\bar{P}(1-\theta P) + V\theta P(1-\theta\bar{P}) = V \tag{40}$$

thus in case 1, $W(\mathbf{x}_{-j},\mathbf{x}'_{-j})$ would not change with θ.

Case 2. when $\mathbf{x}_{-j} \neq \mathbf{x}'_{-j}$, and $Z(0,1) = Vg(0,1) \leq \bar{V}\bar{g}(0,1)$, $Z(1,0) = \bar{V}\bar{g}(1,0) < Vg(1,0)$. We have

$$V\theta\bar{P}(1-\theta P) \leq \bar{V}\theta P(1-\theta\bar{P}) \tag{41}$$

$$V\theta P(1-\theta\bar{P}) > \bar{V}\theta\bar{P}(1-\theta P) \tag{42}$$

if $V < \bar{V}$, then

$$\bar{P} < P \text{ and } \frac{\bar{V}\bar{P} - VP}{P\bar{P}(\bar{V} - V)} < \theta \leq \frac{\bar{V}P - V\bar{P}}{P\bar{P}(\bar{V} - V)} \tag{43}$$

if $V = \bar{V}$, then

$$\bar{P} < P \text{ and } \theta \in [0, 1] \tag{44}$$

In either case,

$$W(\mathbf{x}_{-j}, \mathbf{x}'_{-j}) = -(P\bar{P})(V + \bar{V})\theta^2 + (\bar{V}\bar{P} - VP)\theta \tag{45}$$

which strictly monotonically decreases with respect to θ, given Eq. (43, 44).

Case 3. when $\mathbf{x}_{-j} \neq \mathbf{x}'_{-j}$, and $Z(0,1) = \bar{V}\bar{g}(0,1) < Vg(0,1)$, $Z(1,0) = Vg(1,0) \leq \bar{V}\bar{g}(1,0)$. This case is similar with case 2, we also find $W(\mathbf{x}_{-j}, \mathbf{x}'_{-j})$ monotonically decreases with respect to θ. Because, this case entails

$$V\theta\bar{P}(1-\theta P) > \bar{V}\theta P(1-\theta\bar{P}) \tag{46}$$

$$V\theta P(1-\theta\bar{P}) \leq \bar{V}\theta\bar{P}(1-\theta P) \tag{47}$$

if $V < \bar{V}$, then

$$\bar{P} > P \text{ and } \frac{\bar{V}P - V\bar{P}}{P\bar{P}(\bar{V} - V)} < \theta \leq \frac{\bar{V}\bar{P} - VP}{P\bar{P}(\bar{V} - V)} \tag{48}$$

if $V = \bar{V}$, then

$$\bar{P} > P \text{ and } \theta \in [0, 1] \tag{49}$$

in either case,

$$W(\mathbf{x}_{-j}, \mathbf{x}'_{-j}) = -(P\bar{P})(V + \bar{V})\theta^2 + (\bar{V}P - V\bar{P})\theta \tag{50}$$

which strictly monotonically decreases with respect to θ, given Eq. (48, 49).

Case 4. when $\mathbf{x}_{-j} \neq \mathbf{x}'_{-j}$, and $Z(0,1) = \bar{V}\bar{g}(0,1) < Vg(0,1)$, $Z(1,0) = \bar{V}\bar{g}(1,0) < Vg(1,0)$. This configuration is impossible, given Eq. (20, 27, 30).

From the discussion of all possible cases, we proved that $W(\mathbf{x}_{-j}, \mathbf{x}'_{-j})$ always monotonically decreases with θ, which in turn proves lemma 4.7.2.

Proof of Lemma 4.7.3

Proof From case 0-4 in the above proof we have

$$W(\mathbf{x}_{-j}, \mathbf{x}'_{-j}) \leq V \tag{51}$$

The AUC risk of having only the $j-1$ attributes is

$$\mathcal{R}(\mathcal{D}) = \sum_{(\mathbf{x}_{-j}, \mathbf{x}'_{-j})} W(\mathbf{x}_{-j}, \mathbf{x}'_{-j}) < \sum_{(\mathbf{x}_{-j}, \mathbf{x}'_{-j})} V = \mathcal{R}(\mathcal{D}_{-j}) \tag{52}$$

where $\mathcal{D}_{-j}$ is the data space with only attributes $\mathbf{x}_{-j}$. Thus we have proved that adding attribute will lead to lower AUC risk.

Curriculum Vitae

Name:	Xiang Li
Post-Secondary Education and Degrees:	University of Western Ontario London, ON 2013 - 2017 Ph.D. National University of Defense Technology Changsha, Hunan, China 2010 - 2012 M.Sc. National University of Defense Technology Changsha, Hunan, China 2006 - 2010 B.Sc.
Honours and Awards:	First prize award in Machine Learning and AI Session of UWO Research in Computer Science, UWORCS 2016
Scholarships	Western Graduate Research Scholarship 2013 - 2017 China Scholarship Council Scholarship 2013 - 2017

Related Work	Research Assistant
Experience:	University of Western Ontario
	2013 - 2017
	Teaching Assistant
	University of Western Ontario
	2013 - 2017

Publications:

Xiang Li, Huaimin Wang, Bin Gu, and Charles X Ling. Data sparseness in linear SVM. In Proceedings of the 24th International Conference on Artificial Intelligence (IJCAI), pp. 3628-3634. AAAI Press, 2015.

Xiang Li, Charles X. Ling, and Huaimin Wang. The convergence behavior of naive bayes on large sparse datasets. In Proceedings of the 2015 IEEE International Conference on Data Mining, pp. 853-858.

Xiang Li, Charles X. Ling, and Huaimin Wang. The convergence behavior of naive bayes on large sparse datasets. ACM Transactions on Knowledge Discovery from Data (TKDD), 11(1):10:110:24, July 2016.

Xiang Li, Bin Gu, Shuang Ao, Huaimin Wang, and Charles X. Ling. Triply Stochastic Gradients on Multiple Kernel Learning. 2017 International Conference on Uncertainty in Artificial Intelligence, accepted.

Xiang Li, Huaimin Wang, Bin Gu, and Charles X Ling. The Convergence of Linear Classifiers on Large Sparse Data. Neurocomputing, under review.

Shuang Ao, **Xiang Li**, Charles X. Ling. Effective Multiclass Transfer for Hypothesis Transfer Learning. In Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 64-75. Springer, Cham, 2017.

Shuang Ao, **Xiang Li**, Charles X. Ling. Fast Generalized Distillation for Semi-Supervised Domain Adaptation. In Proceedings of the 21th AAAI Conference on Artificial Intelligence, pp. 1719-1725. AAAI Press, 2016.