Fall 11-25-2009

Classification, Clustering and Data-Mining of Biological Data

Thomas Triplet
University of Nebraska at Lincoln, thomastriplet@gmail.com

Classification, Clustering and Data-Mining of Biological Data

by

Thomas Triplet

A Dissertation

Presented to the Faculty of
The Graduate College at the University of Nebraska
In Partial Fulfillment of Requirements
For the Degree of Doctor of Philosophy

Major: Computer Science
(Bioinformatics)

Under the Supervision of Professor Peter Revesz

Lincoln, Nebraska

November, 2009

Classification, Clustering and Data-Mining of Biological Data

Thomas Triplet, Ph.D.

University of Nebraska, 2009

Advisor: Peter Revesz

The proliferation of biological databases and the easy access enabled by the Internet is having a beneficial impact on biological sciences and transforming the way research is conducted. There are currently over 1100 molecular biology databases dispersed throughout the Internet. However, very few of them integrate data from multiple sources. To assist in the functional and evolutionary analysis of the abundant number of novel proteins, we introduce the PROFESS (PROtein Function, Evolution, Structure and Sequence) database that integrates data from various biological sources. PROFESS is freely available at `http://cse.unl.edu/~profess/`. Our database is designed to be versatile and expandable and will not confine analysis to a pre-existing set of data relationships. Using PROFESS, we were able to quantify homologous protein evolution and determine whether bacterial protein structures are subject to random drift after divergence from a common ancestor.

After relevant data have been mined, they may be classified or clustered for further analysis. Data classification is usually achieved using machine-learning techniques. However, in many problems the raw data are already classified according to a set of features but need to be reclassified. Data reclassification is usually achieved using data integration methods that require the raw data, which may not be available or sharable because of privacy and legal concerns. We introduce general *classification integration* and *reclassification* methods that create new classes by combining in a flexible way the existing classes without requiring access to the raw data. The flexibility is achieved by representing any linear classification in a constraint database. We also considered *temporal data classification* where the input is a temporal database that describes measurements over a period of time in history while the predicted class is expected to occur in the future. We experimented the proposed classification methods on five datasets covering the automobile, meteorological and medical areas and showed significant improvements over existing methods.

To my loving fiancée Chloë

Acknowledgements

I would like to express my sincere gratitude to my thesis advisor Professor Peter Revesz who first introduced bioinformatics to me. Without his brilliant ideas and guidance I would not have been able to complete this dissertation. He has helped me to expand the breadth and depth of my knowledge and research by providing many insights into my research problems.

I would like to thank Dr. Jitender Deogun, Dr. Mark Griep and Dr. Robert Powers for serving on my thesis committee. I sincerely appreciate their inestimable feedback on my research and their valuable comments on my dissertation. A special thanks to Dr. Jean-Jack Riethoven for giving me the opportunity to work with him at the Bioinformatics Core Research Facility. I would also like to thank Matt Shortridge for answering so many of my questions in biology.

I am grateful to the Milton E. Mohr Fellowship at the University of Nebraska-Lincoln and the Department of Computer Science & Engineering for their financial support related to this work. I am also especially grateful to Dr. Gregory Butler from Concordia University (Montreal, QC) for his understanding and his support while I was finishing my dissertation, in particular during the ADBIS 2009 conference in Riga.

No special thanks to Dell, whose customer service proved to be remarkably worthless when my laptop burnt three weeks before submitting this dissertation (for the second time in one year!). The numerous overnight experiments I ran on my laptop to complete this work must have been too intensive...

Finally, I would like to thank my parents who have provided great support and encouragement throughout my education. Last, I would like to thank my adorable fiancée Chloë for her understanding, love, and support throughout all these years and short nights.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

"*Biological databases have been invaluable for making [protein data] accessible. But, although they are architecturally similar, so far their integration has proved problematic.*"

Stein, 2003[1]

The progress of the techniques in molecular biology, genomics and medicine have enabled the scientific community to collect an increasing amount of biological and medical data. However, the greatest value of the data is in its analysis, which may be challenging when dealing with huge amounts of data if a data structure is not properly defined. Bioinformatics is the new interdisciplinary area at the intersection of biological and computer, and information sciences necessary to manage, process, and understand those large amounts of data, for instance from genome sequencing or from large databases containing information about plants and animals for use in discovering and developing new drugs.

Current biological databases present many types of problems, from poorly defined user-interfaces[2] to limited and error-prone search options[3]. A common problem with databases are the unique, randomly assigned accession numbers (COG number, Pfam number, PDB-ID, etc) that are the primary means to retrieve data, but are generally extremely difficult to obtain or decipher. Therefore text-based searches against predefined fields are the major means for a biologist to retrieve the desired data from databases[4,5,6]. However, current implemented text-based queries are fundamentally flawed and routinely return inaccurate and redundant results[3].

For example, a text search of the Protein Data Bank[7] for structures of the Bid apoptosis protein yielded nine structures, but only three Bid structures are in the PDB. The other six structures were retrieved by mistake, which included such completely unrelated structures as a parvovirus-DNA complex (PDB-ID 4DPV) and an actin-inhibtor complex (PDB-ID 2FXU). While this short list can be manually filtered, it clearly identifies a problem with searches that result in large lists of answers, which cannot be manually edited and are probably unreliable. Extending a search across multiple databases requires correlating answers from several different queries; this will simply compound the number of errors. The query results will probably be meaningless since a majority of the data had been incorrectly selected. How data can be retrieved, classified and analyzed is greatly restricted. The remaining of this chapter gives an overview the author's main contributions.

The PROFESS database

To address the above issues, we first propose in Chapter 3, the PROFESS (PROtein Function, Evolution, Structure and Sequence) database, a genome biology database system to assist in the functional and evolutionary analysis of the abundant number of novel proteins. To achieve this goal, it is essential for the database structure to be versatile and expandable and not confine analysis to a pre-existing set of data relationships. A fundamental component of this approach is based on an intuitive query system that incorporates a variety of similarity functions capable of generating data relationships not conceived during the creation of the database. PROFESS is freely available at `http://cse.unl.edu/~profess/`. It was presented at the 11th International Congress on Amino Acids, Peptides and Proteins[8] and has been submitted for publication in *Database*[9].

Phylum Dependent Divergence

In Chapter 4, we present in details a concrete example of how PROFESS can be utilized to quantify homologous protein evolution and determine whether bacterial protein structures are subject to random drift after divergence from a common ancestor. The clusters of orthologous groups (COG) classification system was used to annotate homologous bacterial protein structures in the Protein Data Bank (PDB). The structures and sequences of proteins within each resulting COG were then compared against each other to establish their relatedness. As expected, the analysis showed there was a sharp structural divergence between the bacterial phyla *Firmicutes* and *Proteobacteria*. Additionally, we showed that although each COG had a distinct sequence/structure relationship – suggesting that different evolutionary pressures affect the degree of structural divergence – the relative drift rate between sequence identity and structure divergence remains constant. Our analysis has been submitted for publication in PLoS ONE[10].

Once some relevant data have been retrieved, they may be classified or clustered for further analysis. Data classification is usually achieved using decision trees[11], support vector machines[12] or other machine learning algorithms. Researchers or medical experts may conduct similar studies, leading to redundant classifications. Intuitively, those studies may be combined to produce more accurate results. Most data integration methods will fail because the raw data may not be available for privacy and legal concerns. Integrating the classification results of the former studies may still be valuable. How can those classifications be integrated? How does this classification integration compare with standard data integration methods?

Classification Integration

Chapter 7 introduces *classification integration*[13] as a new method for data integration from different sources. Classification integration is a new method that represents classifiers using constraints. The flexibility is achieved by utilizing our novel representation of any linear classification in a constraint database[14,15,16] (Chapter 6). Our method does not depend on any particular linear classifier nor any specific constraint database system. Our experiments using support vector machines and decision trees on two medical datasets show that the proposed classification integration method is significantly more accurate than current data integration methods when there are many missing values in the data. Beside these particular cases, our general method is also appropriate for many other application areas and may yield similar accuracy improvements. Our classification integration method has been accepted for publication in *Artificial Intelligence in Medicine*[13].

Classification Reclassification

The classification integration problem is further complicated when two classifiers classify different characteristics. In many problems the raw data is already classified according to a variety of features using some linear classification algorithm but needs to be reclassified. In Chapter 8, we introduce a novel *reclassification* method[17] that creates new classes by combining in a flexible way the existing classes without requiring access to the raw data. The flexibility is achieved by representing the results of the linear classifications in a linear constraint database and using the full query capabilities of a constraint database system. The method was successfully tested on two datasets that were already classified using support vector machines and decision trees. Our reclassification method was presented at the 5th Midwest Database Research Symposium[18] and was published in the proceedings of the 12th East-European Conference on Advances in Databases and Information Systems[17].

Temporal Data Reclassification

The classification integration and reclassification proposed above considered the case where data was collected at the same time. Chapter 9 describes an extension to those methods by considering *temporal data classification*[19] where the input is a temporal database that describes measurements over a period of time in history while the predicted class is expected to occur in the future. We describe a new temporal classification method that improves the accuracy of standard classification methods. The benefits of the method are tested on weather forecasting using the meteorological database from the Texas Commission on Environmental Quality. We also used the Google Flu Trends data to show that the proposed method outperforms traditional spatio-temporal interpolation method. Our temporal data classification method was published in the proceedings of the 13th East-European Conference on Advances in Databases and Information Systems[19] and has been selected for submission in *Information Systems*[20].

Finally, Chapter 10 concludes this work and and suggests possible future extensions to solve open problems.

Chapter 2

Background and Related Material

Bioinformatics (noun): *The collection, organization and analysis of large amounts of biological data, using networks of computers and databases.*

Biology (from Greek $\beta\iota o\lambda o\gamma o\varsigma$ - $\beta\iota o\varsigma$, "life"; -$\lambda o\gamma o\varsigma$, -logos, study of) is the science of living organisms. Biology at a microscopic scale began in 1665 when Robert Hooke discovered that organisms are composed of small individual units called *cells*. This discovery marked an important milestone as it turned biology into a science beyond the reach of the naked eye. After describing the basic concepts of biology (Section 2.1), we review some of the major bioinformatics tools (Section 2.2) that have been developed for more than twenty years to assist biologists. Section 2.3 explains how machines can acquire some knowledge in a field of interest and automatically build taxonomies to classify items. Section 2.4 introduces constraint databases, which we use to improve machine-learning algorithms. Finally, Section 2.5 presents spatio-temporal interpolation methods.

2.1 Basic Concepts of Biology

A great diversity of cells exists in nature. Despite those differences, they also share a great number of common characteristics. Cells are a complex mechanical system. Not only do all cells store all the *information* necessary to make a complete replica of itself*, they

*Except haploid cells or gametes

also contain the *machinery* necessary to store energy, manufacture their components, etc... Despite their complexity, cells also share structural components that are conserved across most – if not all – organisms.

The remaining of this review of biological principles is organized as follows: section 2.1.1 explains in more details the four classes of biomolecules. Sections 2.1.2 and 2.1.3 gives a general description of the processes that allow the synthesis of proteins from DNA. Finally, section 2.1.4 describes in details the four levels of protein structures.

2.1.1 Biomolecules

All known forms of life depends on fours classes of biomolecules[*]:

- **Carbohydrates** or saccharides are the most abundant biomolecules. They play many roles in cells, from energy storage to structural components.
- **Lipids**, which are used in cell membranes and as efficient energy source.
- **Nucleic acids** carry the genetic information of the organism.
- **Proteins** include *enzymes* that perform most biochemical reactions for cell regulation.

2.1.1.1 Carbohydrates

Carbohydrates are simple organic compounds that have the empirical formula $(CH_2O)_n$. The simplest carbohydrates are called *monosaccharides* or sugars. Glucose is the six-carbon monosaccharide used as a basic source of energy by most heterotrophic organisms, that is, organisms that use organic carbon for growth. Ribose and deoxyribose are the five-carbon sugars that serve a structural role in the nucleic acids RNA and DNA respectively. Sucrose (see Figure 2.1[†]) is a disaccharide composed of glucose and fructose (an isomer of glucose) and is the major sugar transported between cells in plants, whereas glucose is the primary sugar transported in animal cells. Most carbohydrate molecules in nature are composed of hundreds of sugar units and are referred as *polysaccharides*.

[*]Other types of molecules, which are beyond the scope of this work, also play a critical role in cells.

[†]From http://www.bio.miami.edu/∼cmallery

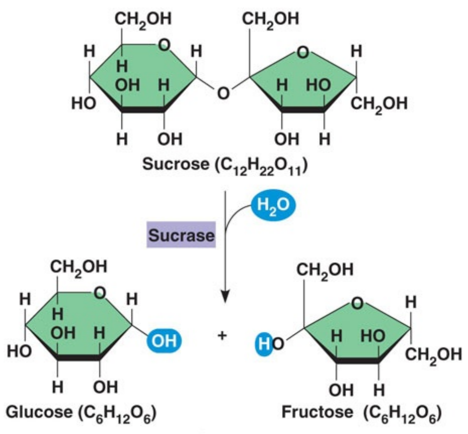


Figure 2.1: Carbohydrate sucrose is a disaccharide composed of glucose and fructose. In presence of water, sucrose can be hydrolyzed by an enzyme called *sucrase*.

Carbohydrates play several major functions in living organisms. For example, monosaccharides serve as readily utilizable energy sources. Carbohydrates also performs structure roles, such as cellulose in plant cell walls and chitin in the exoskeletons of arthropods. Surface carbohydrates often form complexes with proteins known as glycoproteins, or with lipids. The great potential for structural diversity and thus, specificity, makes these molecules very useful as cell markers in cellular communication.

2.1.1.2 Lipids

Lipids are small non-polar molecules that are insoluble in water. The most important feature of lipids is their ability to form sheetlike membranes. Membranes in both prokaryotic and eukaryotic cells separate the cellular content from the external environment, thus allowing the cell to function as a unit. Lipids also serve as highly efficient energy storage molecules.

2.1.1.3 Nucleic Acids

Nucleic acids occur in two forms: *deoxyribonucleic acid* (DNA) and *ribonucleic acid* (RNA). Both are linear unbranched polymers of subunits called *nucleotides*. DNA is found in the nucleus of eukaryotes and the cytoplasm or nucleoid of prokaryotes, and is the molecule that contains the genetic material of the organism. RNA molecules are synthesized on DNA templates (see Section 2.1.2) and participate in protein synthesis in the cytoplasm (see

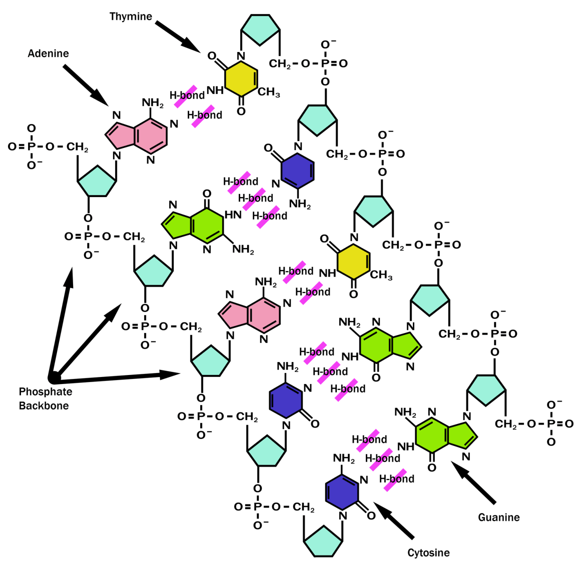


Figure 2.2: DNA is composed of a phosphate group, a deoxyribose sugar (cyan) and four nitrogen-containing organic bases: adenine (pink), cytosine (purple), guanine (green) and thymine (yellow). The DNA molecule is composed of two complementary strands, where adenine bonds to thymine, and cytosine bonds to guanine.

Section 2.1.3). DNA is usually found in the form of a helix composed of two complementary strands whereas RNA is single-stranded.
Each nucleotide consists of three major parts:

1. A five-carbon carbohydrate (pentose).

2. A negatively charged phosphate group, which gives the polymer its acidic property.

3. A nitrogen-containing organic base.

The sugar β-D-ribose is found in the ribonucleotide monomers of RNA. The pentose in the deoxyribonucleotide monomers of DNA differ only by the absence of oxygen at the #2 carbon and is thus called 2-deoxy-β-D-ribose. The organic bases are of two types: single-ringed *pyrimidines* and double-ringed *purines*. The purines are *adenine* (A) and *guanine*. The pyrimidines are *cytosine* (C), *thymine* (T) and *uracil* (U). Thymine is primarily found in DNA, whereas uracil is found in RNA only.

Historically, DNA was first discovered in 1869 when Johann Friedrich Miescher isolated a substance he called "nuclein" from nuclein of white blood cells. In the early 1900s, the four organic bases of DNA were known. In the 1920s, nucleic acids were classified into two classes: DNA and RNA. Interestingly, for nearly eighty years, little attention if any was given to DNA, because it was thought to be a simple polymer incapable of encoding sophisticated genetic information as hypothesized by the accepted Schrödinger's code script*. Schrödinger's theory was proved to be incorrect in 1944 by Oscar *et al.* when they demonstrated that genes indeed reside on DNA. In 1950, Chargaff discovered an exact one-to-one ratio of the adenine/thymine and cytosine/guanine content[21]. A year later, Wilkins[22] and Franklin[23] obtained the first X-ray images of DNA, which suggested a helical structure. The actual double helical structure the the DNA molecule was determined in 1953 by James Watson and Francis Crick[24].

"*It has not escaped our notice that the specific pairing we have postulated immediately suggests a possible copying mechanism for the genetic material.*"

Watson and Crick, 1953

Each type of base on one strand forms a bond with just one type of base on the other strand (see Figure 2.2). This is called *complementary base pairing.* Purines form hydrogen bonds to pyrimidines, with adenine pairing only with thymine, and cytosine only with guanine. This arrangement of two nucleotides binding together across the double helix is called a *base pair.* As hydrogen bonds are not covalent, they can be broken and rejoined relatively easily. DNA strands start at the 5'-end of the nucleotide chain and terminates at the 3'-end of the molecule. The sugar-phosphate backbone is on the outside of the helix where the polar phosphate groups interact with the environment. The nitrogen containing bases are inside, stacking perpendicular to the helix axis. On the contrary, RNA molecules are synthesized as single stranded molecules. The single RNA strand may however fold onto itself and form complementary base pairs to make unique secondary structures. Such

*Schrödinger suggested that the rules defining life was encrypted in a complex *instruction book.*

RNA secondary structures proved to be critical in the synthesis of some proteins, such as selenocysteine-containing proteins[25].

As noticed by Watson and Crick[24], an important property of DNA is that it can self-replicate, that is, make copies of itself through a process known as *replication*[*]. Each strand of DNA in the double helix can serve as a pattern for duplicating the sequence of bases. This is critical when cells divide because each daughter cell needs to have an exact copy of the DNA present in the mother cell.

There exists three major types of RNA molecules:

- **Messenger RNA** (mRNA), the most common type of RNA, are generated from DNA templates (section 2.1.2), and used to synthesize proteins.
- **Ribosomal RNA** (rRNA), are a central component of ribosomes, which are used during RNA translation (section 2.1.3).
- **Transfert RNA** (tRNA), are small RNA strands (typically less than 100 nucleotides) that bind individual amino-acids and are used to grow the polypeptide chain.

2.1.1.4 Proteins

Proteins – or polypeptides – are molecular devices, at the atomic scale, where biological functions are performed. They are the fundamental bricks of cells in our bodies and in all living organisms. Although the genetic information necessary for life to exist is encoded in DNA, the actual processes for life maintenance, cell regulation, replication and reproduction are carried out by proteins.

Proteins are classified at different levels in the Gene Ontology database[26]. The *Molecular Function* level describes the tasks performed by proteins and encompasses twelve categories: cellular processes, metabolism, DNA replication/modification, transcription/translation, intracellular signaling, cell-cell communication, protein folding/degradation, transport, multi-functional proteins, cytoskeletal/structural and defense and immunity. Structural proteins for instance are responsible for the cell physical integrity. The immune system, which is

[*]The details of DNA replication are beyond our scope but may be found in numerous textbooks.

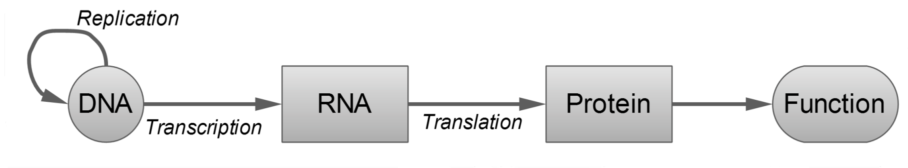


Figure 2.3: The central dogma of molecular biology stipulates that the genetic information in DNA, which can self-replicate, is used to make RNA molecules through a process known as *transcription*, and that the information in RNA is used to synthesize proteins which have a function by a process called *translation*.

responsible for our body's defense, is based on specific structure recognition. At a molecular level, such recognition processes consist of protein-protein interactions on the surface of immune system' cells.

Proteins typically contains thousands of atoms, that may be difficult to decipher. In order to simplify protein description, four structural levels are usually documented. These hierarchical levels describe increasingly complex protein structure level and are referred as primary, secondary, tertiary and quaternary structures of proteins. These structure levels are described in greater details in Section 2.1.4.

2.1.2 Transcription: from DNA to RNA

Shortly after the double helical structure of DNA was discovered, the hypothesis that nucleic DNA functions as the template for mRNA molecules, which subsequently move to the cytoplasm where they are used to determine the amino-acid sequence of proteins, was accepted. This pathway for the flow of genetic information was referred by Crick & Watson [27] in 1956 as the *central dogma* of biology (Figure 2.3). Note that the arrows in this pathway are unidirectional, meaning that polypeptide templates are never used to synthesize mRNA strands and mRNA templates are not used to synthesize DNA strands*. More than 50 years later, the central dogma remains essentially valid.

The genetic information in DNA – or gene – is used to synthesize mRNA molecules through a process known as *transcription*, which is carried out by RNA polymerase enzymes.

*The synthesis of genetic material from RNA strands may actually occur in rare occasions. Retro-viruses such as the Human Immunodeficiency Virus are examples of this process, also known as *retro-transcription*.

The information for synthesizing a specific mRNA strand is located in only one of the two DNA strands. The strand that actually contains the usable genetic information to make the mRNA molecule is called the *template* – or *sense* – strand. Its complementary strand is usually called the nonsense strand, because it contains no useful information for the synthesis of this specific mRNA. Note that DNA templates coding for mRNAs are not all on the same DNA strand.

Transcription in prokaryotic cells differs from transcription in eukaryotic cells. A major difference is that the genetic material in eukaryotic cells is located in a well-defined nucleus. Transcription in eukaryotic cells hence occur in the nucleus. mRNA is subsequently transported to the cytoplasm to be translated.

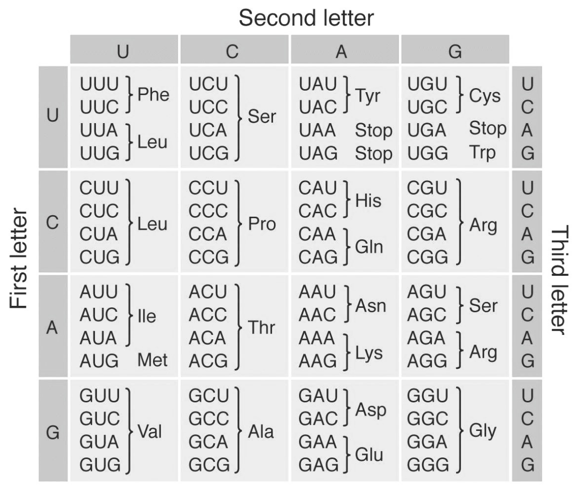
Another major difference is that eukaryotic genes usually alternate coding sequences called *exons* and non-coding sequences called *introns*. Transcription in eukaryotic cells hence produces pre-mRNA strands rather than mRNA. After transcription, pre-mRNA undergoes significant processing before being transported in the cytoplasm. The 5' end is capped with a 7-methylguanine, which ensures stability during translation. The 3' end is polyadenylated. This addition of a poly-A tail at the 3' end plays various roles, from enzymatic degradation protection to transcription termination. Finally, the pre-mRNA is converted into mRNA by the excision of introns and the splicing of the remaining exons.

2.1.3 Translation: from RNA to Proteins using the Genetic Code

Translation is the production of proteins by decoding mRNA produced during the transcription (Section 2.1.2) of DNA. Translation is performed in the cytoplasm by a complex biological machinery, which includes in particular large rRNA molecules called *ribosomes*. Ribosomes are made of two subunits which surround the messenger RNA (mRNA) and produce a specific polypeptide according to the rules specified by the *genetic code*.

The *genetic code*, which defines the codons coding for a specific amino-acid, was determined experimentally. The problem to solve was the following: given the existence of twenty amino-acids and only four bases, how to group nucleotides to encode amino-acids? Pairs of two nucleotides would only specify 16 ($= 4 * 4$) amino-acids are therefore insufficient. Hence, as soon as 1954, focus was given to triplets of three nucleotides because they allows 64 ($= 4 * 4 * 4$) possible permutations, enough to encode the 20 amino-acids. This

Table 2.1: The genetic code establishes the universal correspondence between codons and amino-acids. Most amino-acids are encoded by more than one codon. Protein chains always start with a methionine. Three stop codons may terminate protein chains.



First letter	Second letter: U	Second letter: C	Second letter: A	Second letter: G	Third letter
U	UUU Phe	UCU Ser	UAU Tyr	UGU Cys	U
U	UUC Phe	UCC Ser	UAC Tyr	UGC Cys	C
U	UUA Leu	UCA Ser	UAA Stop	UGA Stop	A
U	UUG Leu	UCG Ser	UAG Stop	UGG Trp	G
C	CUU Leu	CCU Pro	CAU His	CGU Arg	U
C	CUC Leu	CCC Pro	CAC His	CGC Arg	C
C	CUA Leu	CCA Pro	CAA Gln	CGA Arg	A
C	CUG Leu	CCG Pro	CAG Gln	CGG Arg	G
A	AUU Ile	ACU Thr	AAU Asn	AGU Ser	U
A	AUC Ile	ACC Thr	AAC Asn	AGC Ser	C
A	AUA Ile	ACA Thr	AAA Lys	AGA Arg	A
A	AUG Met	ACG Thr	AAG Lys	AGG Arg	G
G	GUU Val	GCU Ala	GAU Asp	GGU Gly	U
G	GUC Val	GCC Ala	GAC Asp	GGC Gly	C
G	GUA Val	GCA Ala	GAA Glu	GGA Gly	A
G	GUG Val	GCG Ala	GAG Glu	GGG Gly	G

hypothesis was proven in 1961 by Crick *et al.*[28] when they established that groups of three nucleotides called *codons* are used to specify individual amino-acids.

The completion of the code in 1966 revealed that 61 codons corresponds to amino-acids, with most amino-acids being encoded by more than one codon (see Table 2.1). Methionine (AUG) always starts a polypeptide chain. The three remaining codons – UAA, UAG and UGA nicknamed ochre, amber and opal respectively – are called as *stop codons* and serve as translational stop signals. In very rare cases, UGA can code for selenocysteine and UAG can code for pyrrolysine depending upon associated signal sequences in the mRNA[25,29,30,31]. A recent study[32] showed that UAA is unlikely to code for a twenty-third amino-acid.

Surprisingly, the genetic code is *universal*, that is, the same codons encode the same amino-acids in all organisms. It should be noted that the genetic code is redundant, but not ambiguous*. One benefit of such redundancy is that many changes – or *mutations* – in the genetic code will have no effect on the amino-acid composition of the protein. For example, the codon CUU codes for leucine. It can bee seen from Table 2.1 that any of the four possible mutations of the third letter of this codon will have no effect on amino-acid sequence because the codon will be translated in leucine in all four cases.

*Turanov *et al.*[33] showed that one codon may code for two amino-acids under certain conditions

2.1.4 Protein Structures

To simplify protein description, four structural levels are usually documented (see Figure 2.4). These hierarchical levels describe increasingly complex protein structure levels and are referred as primary, secondary, tertiary and quaternary structures of proteins.

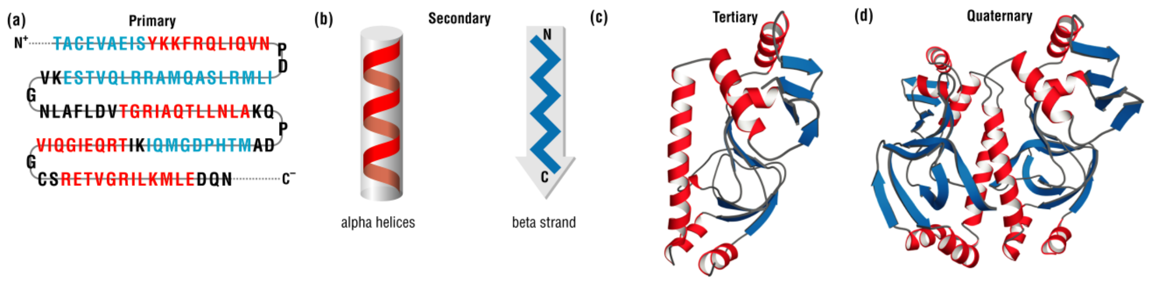


Figure 2.4: The four levels of protein structures. Helices are in red, strands in blue. From *Protein Structure and Function*[34], reproduced with *BioMed Central*'s authorization.

2.1.4.1 Primary Structure

A polypeptide chain is an organic compound composed of monomeric organic residues called *amino acids* or *peptide*. The first amino-acid, asparagine, was discovered in 1806 by Louis-Nicolas Vauquelin and Pierre Jean Robiquet[35]. By the early 1900s, all twenty standard amino-acids had been discovered and their chemical structure identified.

The amino-acids are arranged as a linear chain and the sequence of the amino acids in the protein determines the function of that protein. An average polypeptide chain contains about 300 amino-acids. A large number of amino-acids naturally occur on earth. However, only twenty different amino-acids are used to synthesize polypeptides*.

All amino-acids have the same general structure (Figure 2.5). A carbon atom, known as the α-carbon, lies at the center of each amino-acid. To its left is an amine group (NH_3^+). To the right of the α-carbon, a carboxyl group (COO^-). The carbon in the carboxyl group is referred as the β-carbon. A hydrogen atom forms the third bond of the α-carbon. The fourth bond connects to a side chain (R) which depends on the amino-acid. Table 2.2

*Two additional peptides – selenocysteine and pyrrolysine – may be found in proteins, but are very rare.

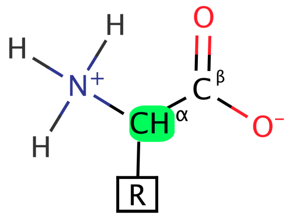


Figure 2.5: General structure of an amino-acid in its zwitterionic form. Each of the 20 amino-acids is composed of an amine group NH_3^+ (left), a carboxylic acid group COO^- (right) and a side chain R (red). The central carbon atom (green) is referred as the α-carbon. The carbon in the carboxyl group is referred as the β-carbon.

shows the structure of the twenty amino-acids, and summarizes their main structural and functional role in proteins.

Amino-acids may be classified according to the nature and the chemical properties of the side chain R. One way to classifiy amino-acids is to draw the Venn diagram (Figure 2.6), which groups peptides according to their chemical properties[36]. The Venn diagram assigns multiple properties to each amino acid. For example, lysine has the property hydrophobic because of its long side chain as well as the properties polar, positive and charged. Chemical properties of amino-acids and their representation in a diagram are essential for efficient sequence alignments[37,38,39].

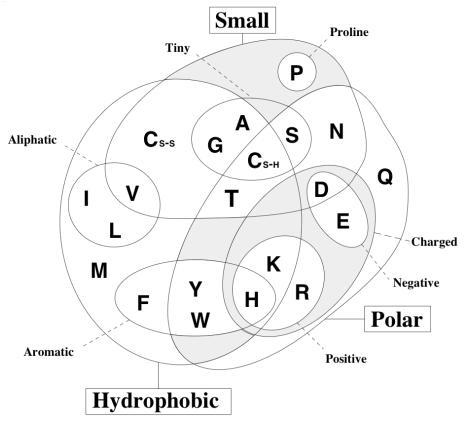
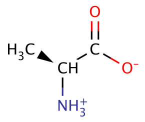
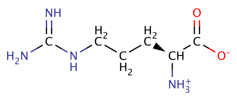
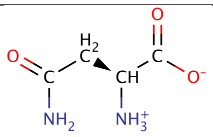
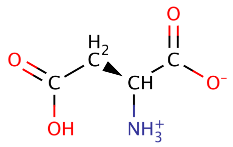
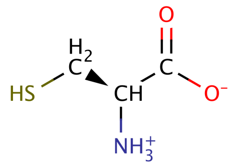
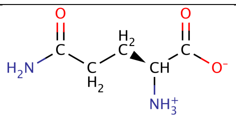


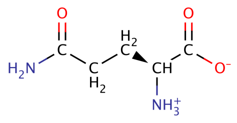
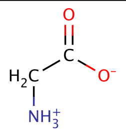
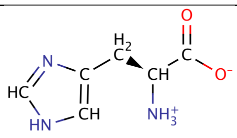
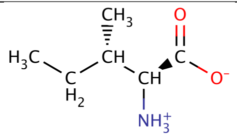
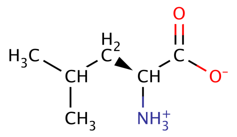
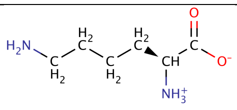
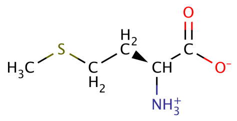
Figure 2.6: Classification of amino-acids properties using a Venn diagram. Shaded areas highlight sets of properties possessed by none of the common amino acids. Cysteine occurs at two different positions depending on the oxidation state of the disulphide bridge.

Table 2.2: Main structural and functional roles of the 20 standard amino-acids

Name	Abbr.	Structure	Main structural and functional roles in proteins
Alanine	A - Ala		The side chain is very non-reactive, and is thus rarely directly involved in protein function. However it can play a role in substrate recognition or specificity, particularly in interactions with other non-reactive atoms such as carbon.
Arginine	R - Arg		Arginines frequently play an important role in structure because of their long amphipathic side chain. Arginines are also frequently involved in salt-bridges, where they pair with a negatively charged amino acid to create stabilising hydrogen bonds. They are quite frequent in protein active or binding sites where their positive charge can interact with anions.
Asparagine	N - Asn		Asparagines are quite frequently involved in protein active or binding sites at the surface of proteins. The polar side-chain is good for interactions with other polar or charged atoms.
Aspartate	D - Asp		Aspartates are quite frequently involved in protein active or binding sites at the surface of proteins. Their negative charge means that they can interact with positively charged non-protein atoms, such as cations. When buried within the protein, aspartates are frequently involved in salt-bridges, where they pair with a positively charged amino acid to create stabilising hydrogen bonds.
Cysteine	C - Cys		The role of cysteines in structure is dependent on the cellular location of the protein. Within extracellular proteins, cysteines are frequently involved in disulphide bonds, where pairs of cysteines are oxidised to form a covalent bond, which serves mostly to stabilise the protein structure. In the intracellular environment, cysteines can play a key structural role as their sulfydryl side-chain is excellent for binding to metals.
Glutamate	E - Glu		Similar to Aspartate.

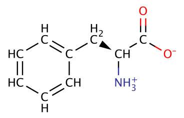
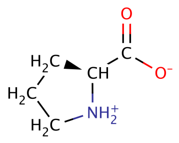
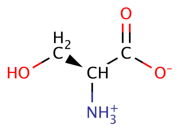
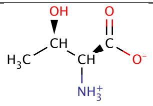
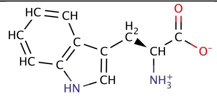
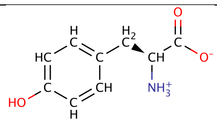
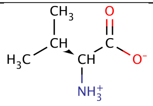
Continued on next page...

Table 2.2 (continued): Main structural and functional roles of the 20 standard amino-acids

Name	Abbr.	Structure	Main structural and functional roles in proteins
Glutamine	Q - Gln		Similar to Asparagine.
Glycine	G - Gly		Glycine is a unique amino acid because it contains a hydrogen as its side chain (instead of a carbon). This allows a greater conformational flexibility of the protein. Glycine can reside in parts of protein structures that are forbidden to all other amino acids and use its sidechain-less backbone to bind to phosphates.
Histidine	H - His		Histidines are the most common amino acids in protein active or binding sites because it has a pKa near to that of physiological pH. They are very common in metal binding sites, often acting together with cysteines.
Isoleucine	I - Ile		Similar to leucine. In addition, like threonine and valine, isoleucine is C-beta branched. It is therefore more restricted in the conformations the main-chain can adopt. It often lies within beta-sheets.
Leucine	L - Leu		The hydrophobic leucine is more likely to be buried in protein hydrophobic cores.The side chain is fairly non-reactive, and is thus rarely directly involved in protein function, though it can play a role in substrate recognition. In particular, phenylalanine can be involved in binding/recognition of hydrophobic ligands such as lipids.
Lysine	K - Lys		Similar to Arginine
Methionine	M - Met		Similar to leucine. Methionine is always the first peptide of a protein. Methionine also contains a sulphur atom, that can be involved in binding to atoms such as metals. It is however connected to a methyl group making less reactive than the sulphur in cysteines.

Continued on next page...

Table 2.2 (continued): Main structural and functional roles of the 20 standard amino-acids

Name	Abbr.	Structure	Main structural and functional roles in proteins
Phenylalanine	F - Phe		Phenylalanines prefer to be buried in protein hydrophobic cores. The aromatic side chain can also mean that tryptophans are involved in stacking interactions with other aromatic side-chains.
Proline	P - Pro		Proline is unique because its side chain is connected to the protein backbone twice. This important difference means that proline is unable to occupy many of the main chain conformations. Hence, proline is often found in very tight turns in protein structures. It can also function to introduce kinks into alpha helices, since it is unable to adopt a normal helical conformation.
Serine	S - Ser		Serine can reside both within the interior of a protein, or on the protein surface. Its small size means that it is relatively common within tight turns on the protein surface, where its side-chain hydroxyl oxygen to form a hydrogen bond with the protein backbone. The hydroxyl group is fairly reactive, being able to form hydrogen bonds with a variety of polar substrates.
Threonine	T - Thr		Similar to serine. In addition, like valine and isoleucine it is C-beta branched, which restricts the conformations the protein can adopt.
Tryptophan	W - Trp		Similar to phenylalanine. Tryptophans' nitrogens can play a role in binding to non-protein atoms, but such instances are rare.
Tyrosine	Y - Tyr		Similar to phenylalanine. However, unlike phenylalanine, tyrosine contains a reactive hydroxyl group, thus making it much more likely to be involved in interactions with non protein atoms.
Valine	V - Val		Similar to leucine. In addition, like isoleucine and threonine it is C-beta branched, which restricts the conformations the protein can adopt.

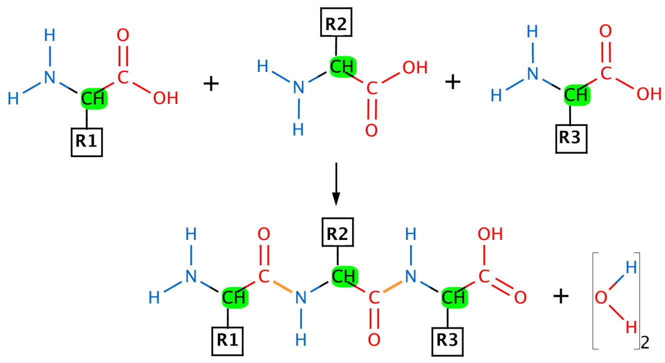


Figure 2.7: The dehydration synthesis reaction between three amino-acids. Amine groups are in blue, carboxyl groups in red. α-carbons are marked in green. The two peptide bonds linking the three amino-acids are highlighted in orange.

The amino acids are joined together by the peptide bonds between the carboxyl and amino groups of adjacent amino acid residues. Peptide bonds linking amino-acids are enzymatically formed by *dehydration synthesis* (see Figure 2.7). An oxygen atom is removed from the carboxyl group of one peptide, together with two hydrogens from the amine of the second peptide.

2.1.4.2 Secondary Structure

In structural biology, the *secondary structure* of a segment of polypeptide chain is the local spatial arrangement of its main-chain atoms without regard to the conformation of its side chains or to its relationship with other segments[40]. Secondary structures are defined by patterns of hydrogen bonds between backbone amide and carboxyl groups.

Polypeptides follow standard stereochemistry rules, which define the possible tridimensional conformations of molecules: all bond lengths and angles are fixed, varying only minimally around their standard values. Due to the chemical nature of the amide bond, the peptide linkage between two amino acids is subject to the phenomenon of resonance, meaning that it acquires the characteristics of a partial double bond. As a consequence, the $C - N$ distance is shorter than a normal single bond and longer than a normal double bond and this partial double bond introduces rigidity into the structure as the bond is no longer freely rotatable. This rigidity forces α-carbons of two adjacent peptides, the carbonyl

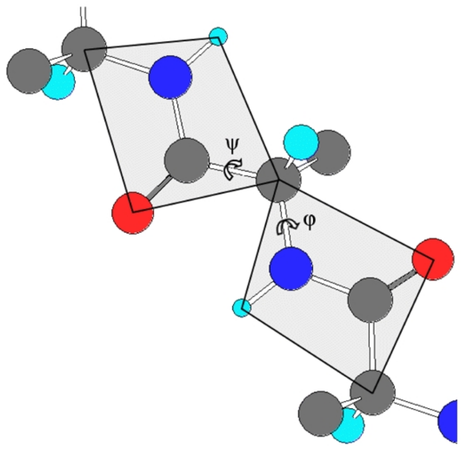


Figure 2.8: Short section of polypeptide chain showing the planar peptide groups and identifying the torsion angles ϕ and ψ.

and the amine of the peptide group to lie on the same plane (see Figure 2.8*). Therefore, the only source of conformational freedom that the polypeptide possesses comes from the torsional rotation around its single bonds. There are only two remaining single bonds per residue along the main chain, and from these bonds one may associate the torsion angles ϕ and ψ. Not all combinations of ϕ and ψ are stereochemically possible, since many lead to steric hindrance. Ramachandran[41,42] showed that only about one third of the ϕ/ψ space is stereochemically accessible to amino acid residues in a real polypeptide. The ϕ/ψ space can be represented in a two-dimensional hyperspace, also known as the Ramachandran plot (see Figure 2.9), where the two variables ϕ and ψ may vary between -180 and $+180$. It should be noted that glycine side chain is composed of a hydrogen atom. Hence it is less restrictive. This can be visualized in the Ramachandran plot where the allowable area is considerably larger when glycine is part of the polypeptide (lighter shade).

Pauling *et al.*[43,44,45,46,47] analysed the geometry and dimensions of the peptide bonds in the crystal structures of molecules containing either one or a few peptide bonds. Two main classes of patterns in proteins: *α-helices* and *β-sheets*. Any pattern that does not fall in one of those two classes is called *random coil*. On average, about 50% of the amino acids are in a secondary structure, among which 54% are in an α-helix and 46% are in a β structure. Table 2.3 summarizes the values of the torsion angles for the main secondary structures.

*From http://www.ncbi.nlm.nih.gov/

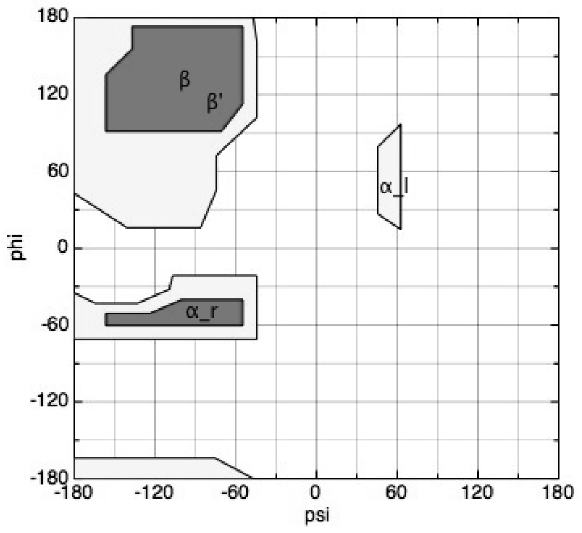


Figure 2.9: A Ramachandran plot represents the stereochemically allowable values for ϕ/ψ. Light-shaded regions show possible angle formations that include glycine, while darker areas are for conformations that do not include glycine. Standard secondary protein structures may be precisely located on the plot: β refers to antiparallel β-sheets, β' to parallel β-sheets, α_r to right-handed α-helices and α_l to the rare left-handed α-helices.

Myoglobin is the first protein whose structure was solved by X-ray crystallography by Perutz and Kendrew in 1958. Their discovery was rewarded by the Nobel Prize in Chemistry in 1962. The analysis revealed that this oxygen-carrying enzyme's structure is made up of eight helices, which represent about 70% of the amino-acids (see Figure 2.11).

A α-helix is a helical structure where each amino acid corresponds to a $100\,^\circ$ turn in the helix, that is, the helix has 3.6 residues per turn (a full turn is $360\,^\circ$), and a translation of 1.5 Å along the helical axis. The pitch of the helix (the vertical distance between two points on the helix) is $1.5 * 3.6 = 5.4\AA$. The most important structural characteristic of α-helices, which may as well define them, is that the N-H group of an amino acid forms a hydrogen bond with the C=O group of the amino acid four residues earlier. The length of α-helices ranges from four to forty amino-acids. However, most helices contain around ten residues, which correspond to about three turns. The 3_{10}-helix is a variant of the standard α-helix, where the N-H group forms a hydrogen bond with the C=O group three residues earlier. Similarly, the π-helix is a variant where the N-H group forms a hydrogen bond with the C=O group five residues earlier. However, those two variants occur rarely. Pace &

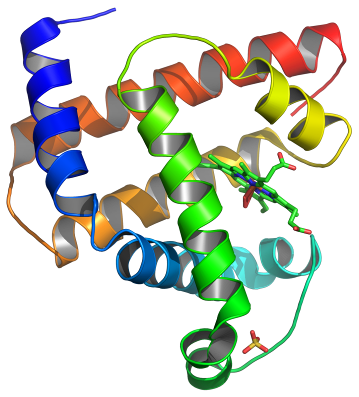


Figure 2.10: A representation of the 3D structure of the myoglobin protein. The eight α-helices are shown in color and represent 70% of the protein's structure.

Scholtz[48] showed that methionine, alanine, leucine, uncharged glutamate, and lysine have high helix-forming propensities, whereas proline, glycine and negatively charged aspartate have poor helix-forming propensities. Proline tends to kink – or break – helices because it has no amide hydrogen to donate. However, proline is often seen as the first residue of a helix.

A *β-strand* is a polypeptide segment where the torsion angle ψ is about $120\,°$. As a result, the sidechains of two neighboring residues in this segment point in the opposite direction from the backbone. A *β-sheet* is composed of two or more strands, linked together by hydrogen bonds between amine groups of one strand and carbonyl groups on the other strand. If the strands all run in one direction, the sheet is called *parallel β-sheet* whereas

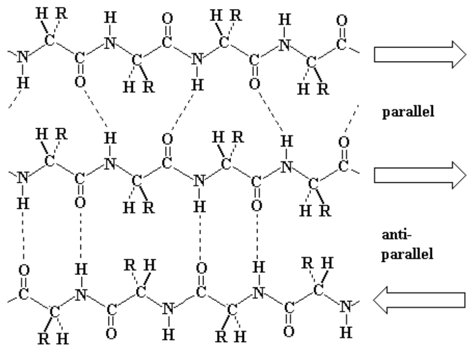


Figure 2.11: Comparison of parallel β-sheets and anti-parallel β-sheets. Hydrogen bonds between strands in parallel β-sheets are not straight and thus weaker.

in *anti-parallel sheets* they all run in opposite directions. In mixed sheets some strands are parallel and others are anti-parallel. In the classical Pauling-Corey[49] models, hydrogen bonds between strands in parallel β-sheets are not straight and are consequently weaker. It should however be noted that the backbone hydrogen bonds of α-helices are slightly weaker than those found in β-sheets, and are thus more likely to be oxidized by ambient water molecules.

2.1.4.3 Ternary Structure

The tertiary structure of a protein molecule is defined by the Commission on Biochemical Nomenclature as the arrangement of all its atoms in space, without regard to its relationship with neighbouring molecules or subunits[40]. The tertiary structure may be considered as the final result of the folding process of the polypeptide chain. This folding depends on the amino acid sequence and on the atomic details of the structure. For soluble proteins this will usually result in the formation of a hydrophobic core in which apolar residues tend to cluster their side chains within the protein's interior, leaving hydrophilic residues exposed to solvents.

Most protein structures been solved with the experimental technique of *X-ray crystallography*, which typically provides data of high resolution but provides no time-dependent information on the protein's conformational flexibility. A second common way of solving protein structures uses *Protein Nuclear Magnetic Resonance spectroscopy*, which provides lower-resolution data in general and is limited to relatively small proteins, but can provide time-dependent information about the motion of a protein in solution.

Table 2.3: Main parameters of regular secondary structures. ϕ and ψ are the torsion angles in degrees, n the number of residues per helical turn, p the helical pitch in angstroms.

Structure	ϕ	ψ	n	p
α-helix	-57	-47	3.6	5.4
3_{10}-helix	-74	-4	3.0	6.0
π-helix	-57	-70	4.4	5.0
Parallel beta strand	-119	113	2.0	6.4
Antiparallel beta strand	-39	135	2.0	6.8

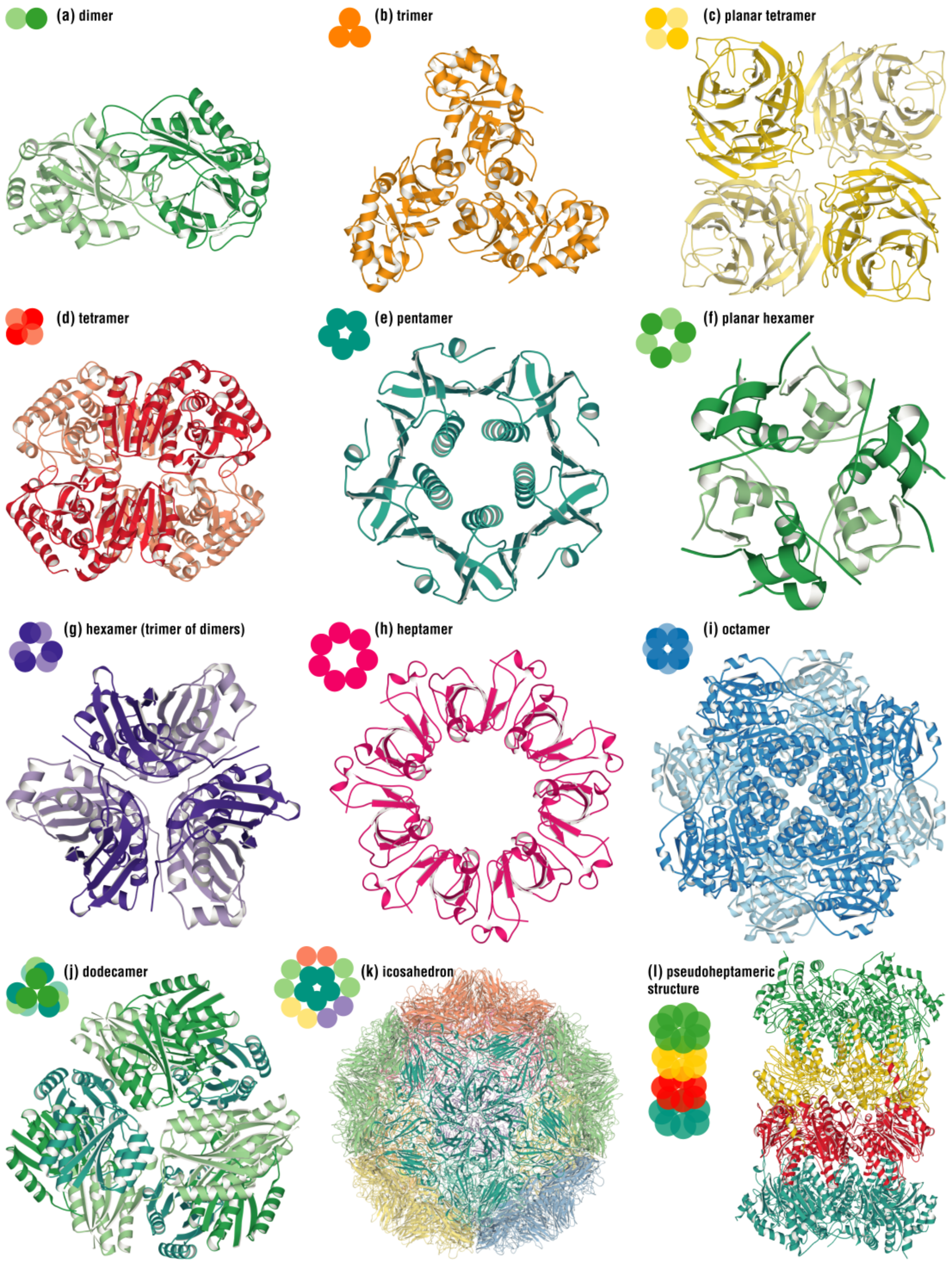


Figure 2.12: Examples of typical protein quaternary structures. Proteins may be composed of two (dimers), three (trimers) or more polypeptides, giving the protein its final shape. From *Protein Structure and Function*[34], reproduced with *BioMed Central*'s authorization.

Many classifications of protein folds exist. The two most widely used are the Structural Classification of Proteins[50] (SCOP) and the CATH Protein Structure Classification[51], both of which use a hierarchical approach. The Protein Data Bank[7] (PDB) is the main repository for protein structures. The knowledge of the crystallographic structure of the protein is a critical piece of data as it may help determining the function of the protein by structural alignment with proteins whose functions are known. It may also provide valuable information for drug design[52,53].

2.1.4.4 Quaternary Structure

Most proteins are composed of several polypeptide chains or subunits. The quaternary structure of a protein molecule is defined as the arrangement of its subunits in space and the ensemble of its intersubunit contacts and interactions, without regard to the internal geometry of the subunits[40]. The subunits in a quaternary structure must be in noncovalent association. The association of these individual polypeptide chains gives the protein its final shape and thus its function. Figure 2.12 illustrates a few typical protein polymers.

2.2 Review of Bioinformatics Databases and Tools

2.2.1 Protein Sequence Alignments

The central dogma of biology[27] stipulates that the function of a protein can be inferred from the tridimensional structure which can be obtained from its primary structure, that is, the amino-acid sequences. A corollary of the central dogma is that proteins that have similar sequences are expected to have similar functions. Hence, analyzing protein sequence similarities can provide valuable information towards the knowledge of the protein function.

Sequence similarity analyzes are usually performed by *sequence alignments* where amino-acids substrings common to both proteins are matched together. Hence, a pairwise sequence alignment can identify conserved residues and mutations. Then, one needs to decide which alignments are more likely to have occurred because the sequences are indeed related, or just by chance. This process, described in Section 2.2.1.1, is usually done by assigning a score to each individual mutation. Section 2.2.1.2 presents a dynamic programming algorithm developed by Smith & Waterman[54] which outputs an exact solution, at the cost of a relatively slow running time. Section 2.2.1.3 presents a very popular heuristic called *Basic Local Alignment Search Tool*[55] (BLAST) which dramatically increases the throughput.

2.2.1.1 Scoring Schema

When comparing protein sequences, one is looking for evidence that the sequences have diverged from a common ancestor by a process of mutation/selection. Three basic types of mutations are considered: *substitutions*, when a residue is changed for another one, *insertions* and *deletions*, when a residue is added or removed from one of the two sequences. Insertions and deletions are referred as *gaps*.

The total score of the alignment is the sum of each of the mutations. As mentioned in Section 2.1.4.1, some amino-acids substitutions are more likely to occur than others, based on their chemical properties. For example, both serine and threonine have a reactive hydroxyl group, which easily forms hydrogen bonds with a variety of polar substrates. Hence, *effective* substitutions of serine for threonine are *expected* to occur quite frequently. On the contrary, arginine usually interacts with anions whereas aspartate usually interacts with cations. Hence, such a substitution is not expected to happen often. These substitution likelihoods are usually represented in the form of a table, called *substitution matrix*. Each matrix is twenty-by-twenty (for the twenty standard amino acids); the value in a given cell represents the probability of a substitution of one amino acid for another.

Point Accepted Mutation (PAM) matrices

Dayhoff *et al.*[56] developed one of the first substitution matrix referred as the *Point Accepted Mutation* (PAM) matrix. The PAM matrices were derived from 1,572 observed mutations in 71 families of closely related proteins. The PAM matrices are normalized so that the PAM_1 matrix has one mutation per hundred amino acids, and is appropriate for scoring sequences which are very similar. PAM matrices for comparing sequences of lower similarity are calculated from repeated multiplication of the PAM_1 matrix by itself. PAM_2 is equivalent to two substitutions per hundred amino acids and is defined by $PAM_2 = PAM_1^2$. PAM_{30} and PAM_{70} are commonly used in practice.

BLOcks of Amino Acid SUbstitution (BLOSUM) matrices

Another popular substitution matrices family – BLOcks of Amino Acid SUbstitution – was introduced by Henikoff & Henikoff[57], who scanned the BLOCKS database[58] for gapless conserved regions of protein families and then counted the relative frequencies of amino acids and their substitution probabilities. BLOSUM matrices are based on observed alignments, without considering closely related proteins like the PAM matrices. Several BLOSUM matrices were built, using different degrees of protein conservation: the conservation percentage used was appended to the name. For example, BLOSUM80 corresponds to the matrix built with sequences that were more than 80% identical. Raw substitution probabilities are in average 1/20. The *log-odds* score s_{ij} for each of the 210 possible substitutions of the twenty standard amino acids were calculated using Equation 2.1.

$$s_{ij} = \frac{1}{\lambda} log(\frac{p_{ij}}{q_i q_j}) \tag{2.1}$$

where, p_{ij} is the probability of two amino acids i and j replacing each other in a homologous sequence, and q_i and q_j are the background probabilities of finding the amino acids i and j

Table 2.4: The BLOSUM62 substitution matrix. Highest scores represent most conservative substitutions. The matrix is symmetric and that higher scores are in the main diagonal.

	C	**S**	**T**	**P**	**A**	**G**	**N**	**D**	**E**	**Q**	**H**	**R**	**K**	**M**	**I**	**L**	**V**	**F**	**Y**	**W**
C	**9**	-1	-1	-3	0	-3	-3	-3	-4	-3	-3	-3	-3	-1	-1	-1	-1	-2	-2	-2
S	-1	**4**	1	-1	1	0	1	0	0	0	-1	-1	0	-1	-2	-2	-2	-2	-2	-3
T	-1	1	**4**	1	-1	1	0	1	0	0	0	-1	0	-1	-2	-2	-2	-2	-2	-3
P	-3	-1	1	**7**	-1	-2	-1	-1	-1	-1	-2	-2	-1	-2	-3	-3	-2	-4	-3	-4
A	0	1	-1	-1	**4**	0	-1	-2	-1	-1	-2	-1	-1	-1	-1	-1	-2	-2	-2	-3
G	-3	0	1	-2	0	**6**	-2	-1	-2	-2	-2	-2	-2	-3	-4	-4	0	-3	-3	-2
N	-3	1	0	-2	-2	0	**6**	1	0	0	-1	0	0	-2	-3	-3	-3	-3	-2	-4
D	-3	0	1	-1	-2	-1	1	**6**	2	0	-1	-2	-1	-3	-3	-4	-3	-3	-3	-4
E	-4	0	0	-1	-1	-2	0	2	**5**	2	0	0	1	-2	-3	-3	-3	-3	-2	-3
Q	-3	0	0	-1	-1	-2	0	0	2	**5**	0	1	1	0	-3	-2	-2	-3	-1	-2
H	-3	-1	0	-2	-2	-2	1	1	0	0	**8**	0	-1	-2	-3	-3	-2	-1	2	-2
R	-3	-1	-1	-2	-1	-2	0	-2	0	1	0	**5**	2	-1	-3	-2	-3	-3	-2	-3
K	-3	0	0	-1	-1	-2	0	-1	1	1	-1	2	**5**	-1	-3	-2	-3	-3	-2	-3
M	-1	-1	-1	-2	-1	-3	-2	-3	-2	0	-2	-1	-1	**5**	1	2	-2	0	-1	-1
I	-1	-2	-2	-3	-1	-4	-3	-3	-3	-3	-3	-3	-3	1	**4**	2	1	0	-1	-3
L	-1	-2	-2	-3	-1	-4	-3	-4	-3	-2	-3	-2	-2	2	2	**4**	3	0	-1	-2
V	-1	-2	-2	-2	0	-3	-3	-3	-2	-2	-3	-3	-2	1	3	1	**4**	-1	-1	-3
F	-2	-2	-2	-4	-2	-3	-3	-3	-3	-3	-1	-3	-3	0	0	0	-1	**6**	3	1
Y	-2	-2	-2	-3	-2	-3	-2	-3	-2	-1	2	-2	-2	-1	-1	-1	-1	3	**7**	2
W	-2	-3	-3	-4	-3	-2	-4	-4	-3	-2	-2	-3	-3	-1	-3	-2	-3	1	2	**11**

in any protein sequence at random. The constant λ is a scaling factor, such that the matrix contains easily computable integer values.

Log-odd ratios are a convenient way to represent small probabilities in a "human-readable" format. BLOSUM62* (see Table 2.4) turned out to give best results in practice. Note that the matrix is symmetric, meaning that the probability that the amino-acid i mutates into j is equal to the probability that j mutates into i.

2.2.1.2 Smith-Waterman Algorithm

Given a scoring scheme and two sequences, we need to find the optimal alignment. Assuming both sequences are composed of n residues, there are†:

$$\binom{2n}{n} = \frac{2n!}{n!^2} \approx \frac{2^{2n}}{\sqrt{\pi n}} \tag{2.2}$$

possible alignments between two sequences of length n. The average length of amino-acids is about 300 residues, leading to $1.4 * 10^{179}$ alignments to consider. Optimistically assuming that 1,000,000 alignments can be computed every second, it would take $4.3 * 10^{165}$ years to enumerate all the possible alignments, that is, $3.1 * 10^{155}$ times the age of the known universe. A brute force approach, consisting of enumerating all possible alignments and then choosing the best one, is clearly not possible. Dynamic programming algorithms must be used instead.

The first dynamic programming algorithm for *global* sequence alignment was devised by Needleman & Wunsch[61]. The algorithm was later improved by Gotoh[62]. However, protein sequences usually contain a number of irrelevant residues at the extremities of the polypeptide chain. Hence, Smith & Waterman[54] proposed a *local* sequence alignment algorithm, also based on a dynamic programming approach, which compute the best alignment of subsequences from both protein. Local alignments are usually favored because they are more sensitive to capture specific conserved domains. When using global alignment

*Styczynski *et al.*[59] showed in 2008 that the BLOSUM62 used as a standard since 1996 is not exactly accurate according to the algorithm described by Pietrokovski *et al.*[58].

†Using Stirling's approximation of large factorials[60]

methods, such domains may be more difficult to detect because of noisy mutations in the remaining of the sequence.

The idea of the algorithm is to build an optimal alignment using previous known optimal alignments of subsequences. The final alignment is hence computed by (1) solving the problem for shorter – and easier to compute – subsequences, (2) combining the individual solutions of the smaller alignments. The alignment is computed by constructing a matrix D, indexed by $i \in [1..n]$ and $j \in [1..m]$, one index representing each of the two sequences of length n and m respectively. Let $x_{1..i}$ be the subsequence of x from the first to the i^{th} residue. The value $D(i,j)$ represents the best alignment between $x_{1..i}$ and $y_{1..j}$ and can be computed recursively as shown in Equation 2.3.

$$D(i,j) = max \begin{cases} 0 \\ D(i-1,j-1) + s(x_i,y_j) \\ D(i-1,j) - d \\ D(i,j-1) - d \end{cases} \tag{2.3}$$

where $s(x_i, y_j)$ is the score for the substitution of residue i from x by residue j of y given by the scoring substitution matrix and d the linear cost for gaps. The option $'0'$ corresponds to starting a new alignment: if the best alignment becomes negative, it is better to start a new one instead of extend the old one. The initial conditions for $i = 0$ or $j = 0$ are defined by $D(i,j) = 0$. Given D, the alignment can then be easily obtained by retrieving the path in the matrix that was necessary to follow to calculate $F(n,m)$. This process is called backtracking.

For example, consider $x =$ `PAWHEAE` and $y =$ `HEAGAWGHEE`. Using the BLOSUM62 scoring matrix and a linear gap penalty of 5. The resulting matrix D is shown in Figure 2.13. Using the matrix D, we can infer the local alignment:

```
AWGHE
AW-HE
```

The complexity of the Smith-Waterman is quadratic – $O(nm)$, with n and m, the length of the two input sequences –, which greatly improves the exponential complexity of the brute force approach. In addition, this algorithm is “correct” in the sense that it guaranties to find *the* optimal alignment given a scoring scheme.

D(i,j)		H	E	A	G	A	W	G	H	E	E
	^{0} 0	$^{-5}$ −5	$^{-5}$ −10	$^{-5}$ −15	$^{-5}$ −20	$^{-5}$ −25	$^{-5}$ −30	$^{-5}$ −35	$^{-5}$ −40	$^{-5}$ −45	$^{-5}$ −50
P	$^{-5}$ −5	$^{-2}$ 0	$^{-1}$ 0	$^{-1}$ 0	**$^{-2}$ 0**	$^{-1}$ 0	$^{-4}$ 0	$^{-2}$ 0	$^{-2}$ 0	$^{-1}$ 0	$^{-1}$ 0
A	$^{-5}$ −10	$^{-2}$ 0	$^{-1}$ 0	^{4} 4	^{0} 0	**^{4} 4**	$^{-3}$ 0	^{0} 0	$^{-2}$ 0	$^{-1}$ 0	$^{-1}$ 0
W	$^{-5}$ −15	$^{-2}$ 0	$^{-3}$ 0	$^{-3}$ 0	$^{-2}$ 2	$^{-3}$ 0	**^{11} 15**	**$^{-2}$ 10**	$^{-2}$ 5	$^{-3}$ 0	$^{-3}$ 0
H	$^{-5}$ −20	^{8} 0	^{0} 0	$^{-2}$ 0	$^{-2}$ 0	$^{-2}$ 0	$^{-2}$ 10	$^{-2}$ 13	**^{8} 18**	^{0} 13	^{0} 8
E	$^{-5}$ −25	^{0} 0	^{5} 5	$^{-1}$ 0	$^{-2}$ 0	$^{-1}$ 0	$^{-3}$ 5	$^{-2}$ 8	^{0} 13	**^{5} 23**	^{5} 18
A	$^{-5}$ −30	$^{-2}$ 0	$^{-1}$ 0	^{4} 9	^{0} 4	^{4} 4	$^{-3}$ 0	^{0} 5	$^{-2}$ 8	**$^{-1}$ 18**	$^{-1}$ 22
E	$^{-5}$ −35	^{0} 0	^{5} 5	$^{-1}$ 4	$^{-2}$ 7	$^{-1}$ 3	$^{-3}$ 1	$^{-2}$ 0	^{0} 5	^{5} 13	**^{5} 23**

Figure 2.13: Dynamic programming matrix for the Smith-Waterman alignment of sequences `PAWHEAE` and `HEAGAWGHEE` using the BLOSUM62 scoring matrix and a linear gap penalty of 5. $D(i,j)$ represents the best alignment between $x_{1..i}$ and $y_{1..j}$. The backtrack path used to construct the local alignment is shaded.

Variants of the original pairwise algorithm have been proposed for multiple sequence alignment, which allows one to align several sequences together. However, the multiple sequence alignment problem proved to be NP-completed[63,64]. To reduce the complexity of the problem, heuristics have been proposed and implemented in ClustalW[65,66], which we used in our structural comparison of functional orthologs (see Chapter 4).

2.2.1.3 Basic Local Alignment Search Tool: BLAST

However, in some cases, the complexity improvement may not be sufficient. In particular, sequence alignments are typically used to match a sequence of interest with a sequence in a database of known proteins. Such databases usually contains millions of protein sequences. In that case, one must perform pairwise alignments between the protein of interest and each of the proteins within the database. For this reason, there has been many attempts to produces non-optimal but faster algorithms[67,68,69]. However, those algorithms did not perform well when used with standard scoring matrices. Hence, new heuristics were developed, in particular FAST-All[70] and the Basic Local Alignment Search Tool[55] (BLAST). The latest was used to built our PROFESS database (see Chapter 3).

The idea behind the BLAST heuristic is that true alignments are likely to contain short highly conserved subsequences or identities. BLAST will hence look for such subsequences – or seeds – which then can be extended. The seeds are normally kept short so that a table with all possible seeds can be preprocessed and used later as a lookup table. Then, BLAST will try to extend seeds with a word of given length (3 by default for protein sequences, 11 for nucleic acid sequences) that matches the query sequence with a score higher than a given threshold. The process is referred as the *hit extension*. The algorithm terminates whenever the score drops below a parametrized expectation threshold. The original BLAST algorithm only found ungapped alignments. However, more recent versions are able to output gapped alignments[71,72,73] and greatly improved performance when querying large sequence databases[74].

2.2.2 Protein Structure Alignments

In addition to aligning protein sequences, there have been a number of attempts to align the tridimensional structure of proteins. Structural alignment provide valuable additional information as a single mutation in the amino-acids sequence can dramatically change the corresponding 3D structure of the protein. For example, leucine is often found in alpha helices. A mutation of this amino-acid into proline is likely to kink the alpha helix because of the unique structure of proline.

These algorithms take as input the atomic coordinates of the proteins to align, and output the superposed atomic coordinates. The algorithms also output the minimal root mean square deviation (RMSD) between the structures, which measures the structural divergence of aligned proteins. When aligning structures with very significantly divergent sequences, most structural alignment methods consider only the backbone atoms included in the peptide bond. The coplanarity of the peptide bond is used to maximize throughput and the α-carbon coordinates alone are usually considered for the alignment. The remaining atoms, in particular the side chains, are used to generate the final alignment only when the RMSD drops below a given threshold, that is, when protein structures are similar enough. Like multiple sequence alignments, the multiple structure alignment proved to be NP-Complete. Hence, approximate polynomial-time solution have been devised by Kolodny & Linial[75] and Zhu[76]. Ye & Godzik[77] also proposed a solution that utilizes graph theory.

Several pieces of software for structural alignment have been implemented. Most popular program include *Dali*[78,79], *Combinatorial Extension*[80] (CE), *MAtching Molecular Models Obtained from THeory*[81] (MAMMOTH) and its multiple alignment extension MAMMOTH-mult[82], and *Sequential Structure Alignment Program*[83] (SSAP) which was used to build the *Class, Architecture, Topology, Homology* (CATH) database[51]. Our PROFESS database was constructed in part by using Dali and MAMMOTH-mult.

2.3 Supervised Machine Learning Algorithms

In many problems, we need to classify items, that is, we need to predict some characteristic of an item based on several of its parameters. Each parameter is represented by a variable which can take a numerical value. Each variable is called a *feature* and the set of variables is called the *feature space*. The number of features is the *dimension* of the feature space. The actual characteristic of the item we want to predict is called the *label* or *class* of the item.

To make the predictions, we use *classifiers*. Each classifier maps a feature space X to a set of labels Y. Classifiers are built using machine learning algorithms, which are able to automatically improve by the analysis of data sets, i.e. they learn by experience. Speech or handwriting recognition are typical applications of machine learning approaches. Also in computational biology various machine learning techniques have been successfully used, for example neural networks for detection of signal peptides in proteins[84], Hidden Markov models for protein homology detection[85] and stochastic context free grammars for modeling and prediction of RNA secondary structures[25,86].

In this work, we are interested in *linear classifiers*, that is, a classifier that can be mathematically defined by a linear equation. We are also assuming that the set of labels used during the training stage is known *a priori*, which is also known as *supervised learning*. Hence, this work does not consider the variety of unsupervised learning algorithms, which include in particular clustering and segmentation techniques.

After briefly introducing linear classifiers in section 2.3.1, we describe two linear classifiers: *Support Vector Machines* in section 2.3.2 and *Decision Trees* in section 2.3.3.

2.3.1 Linear Classifiers

A linear classifier maps a feature space X to a set of labels Y by a linear function. In other words, a linear classifier computes the label of an item to classify using a linear function. In general, a linear classifier $f(\overrightarrow{x})$ can be expressed as follows:

$$f(\overrightarrow{x}) = \langle \overrightarrow{w} \cdot \overrightarrow{x} \rangle + b = \sum_i w_i x_i + b \tag{2.4}$$

where $w_i \in \mathbb{R}$ are the *weights* of the classifiers and $b \in \mathbb{R}$ is a constant. The value of $f(\overrightarrow{x})$ for any item $\overrightarrow{x}$ directly determines the predicted label, usually by a simple rule. For example, in binary classifications, if $f(\overrightarrow{x}) \geq 0$, then the label is $+1$ else the label is -1. Note that the knowledge of the weights w_i is necessary and sufficient to define the linear classifier f.

Example 2.1 Suppose that a disease is conditioned by two antibodies A and B. The feature space is $X = \{Antibody\_A, Antibody\_B\}$ and the set of labels is $Y = \{Disease, No\_Disease\}$, where *Disease* corresponds to $+1$ and *No_Disease* corresponds to -1. A linear classifier is:

$$f(\{Antibody\_A, Antibody\_B\}) = w_1 Antibody\_A + w_2 Antibody\_B + b$$

where $w_1, w_2 \in \mathbb{R}$ are constant weights and $b \in \mathbb{R}$ is a constant. We can use the value of $f(\{Antibody\_A, Antibody\_B\})$ as follows:

- If $f(\{Antibody\_A, Antibody\_B\}) \geq 0$ then the patient has *Disease.*
- If $f(\{Antibody\_A, Antibody\_B\}) < 0$ then the patient has *No_Disease.*

The set of linear classification methods includes a number of algorithms, ranging from generative probabilistic models such as Naive Bayes algorithms, where a model is *built* using some data during the training stage, to discriminative models such as Logistic Regression algorithms, Decision Trees or Support Vector Machines, where the model is obtained by applying *constraints* derived from the training data. The methods proposed in this work are applicable to any linear classifier in general. During our experiments, we used support vector machines and decisions trees, which are described in sections 2.3.2 and 2.3.3 respectively.

2.3.2 Support Vector Machines

In the past decade, Support Vector Machines (SVM), a new class of learning algorithms, have become increasingly popular in computational biology because of their attractive features:

1. During the training, the algorithm optimizes a convex cost function, avoiding local maxima issues: given a model, the SVM algorithm computes the optimal solution.
2. SVMs support high dimensional data sets.
3. SVMs performs well on noisy data.
4. The algorithms are computationally efficient and their modular design makes them easy to implement.

There exists a large body of literature describing Support Vector Machines (SVM) algorithms[12,87,88]. The remaining of this section briefly describes the theoretical principles of SVMs, in particular SVMs with a linear kernel.

Suppose that numerical values can be assigned to each of the n features in the feature space. Let $\overrightarrow{x_i} \in \mathbb{R}^n$ with $i \in [1..l]$ be a set of l training examples. Each training example $\overrightarrow{x_i}$ can be represented as a point in the n-dimensional feature space.

SVMs classify the items by constructing a hyperplane of dimension $n-1$ that will split all items into two sets of classes +1 and -1. As shown in Figure 2.14, several separating hyperplanes may be suitable to split correctly a set of training examples. In this case, an SVM will construct the *maximum-margin hyperplane*, that is, the hyperplane which maximizes the distance to the closest training examples.

Suppose first that the data is linearly separable, that is, there is a hyperplane that separates the items of labels +1 and -1. Then, the maximum-margin hyperplane is defined in Equation 2.5:

$$f(\overrightarrow{x}) = 0 \tag{2.5}$$

To find the hyperplane with the largest margin, $\overrightarrow{w}$ and b can be scaled so that the closest training examples $\overrightarrow{x_i}$ to the hyperplane satisfy $|f(\overrightarrow{x_i})| = 1$. In this case, the margin

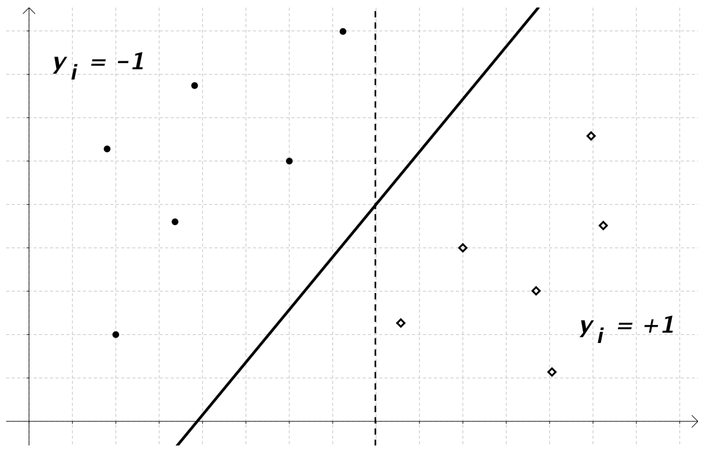


Figure 2.14: A set of training examples with labels $+1$ ($\lozenge$) and -1 ($\bullet$). This set is linearly separable because a linear decision function in the form of a hyperplane can be found that classifies all examples without error. Two possible hyperplanes that both classify the training set without error are shown (solid and dashed lines). The solid line is expected to be a better classifier than the dashed line because it has a wider *margin*, which is the distance between the closest points and the hyperplane.

equals $2/\|\overrightarrow{w}\|$ as shown in Figure 2.15. Maximizing the margin is then equivalent to solving the following optimization problem:

$$\min_{\overrightarrow{w}\in\mathbb{R}^n, b\in\mathbb{R}} \frac{1}{2}\|\overrightarrow{w}\|^2 \tag{2.6}$$

subject to:

$$|f(\overrightarrow{x_i})| \geq 1 \tag{2.7}$$

Equation 2.6 leads to the following solution:

$$f(\overrightarrow{x}) = \sum_{i=1}^{l} \alpha_i y_i \langle \overrightarrow{x_i} \cdot \overrightarrow{x} \rangle + b \tag{2.8}$$

where α_i are positive real coefficients and $y_i \in \{+1, -1\}$ is the label of $\overrightarrow{x_i}$.

However, in practice, data is rarely linearly separable, and Equation 2.8 may not be used. Vapnik[12] combined in 1995 the technique described above with a mathematical method called the *kernel trick* introduced in 1964 by Aizerman *et al.*[89]. The kernel trick consists in a mapping function ϕ (see Figure 2.16) such that data, which is not linearly separable in the input feature space, may be linearly separable in a higher dimension feature space.

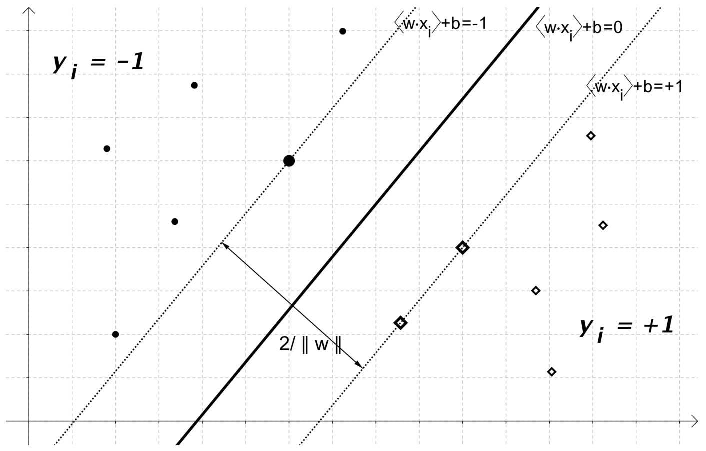

$\langle w\cdot x_i\rangle$+b=-1
$\langle w\cdot x_i\rangle$+b=0
$\langle w\cdot x_i\rangle$+b=+1
y_i = -1
y_i = +1
2/ ‖ w ‖

Figure 2.15: Building the optimal – or maximum margin - hyperplane. The points nearest to the optimal separating hyperplane are called the "support vectors" (drawn larger than the other points). When w and b are properly scaled, the margin is $2/\|w\|$.

Using the kernel trick, Equation 2.8 can be reformulated as:

$$f(\overrightarrow{x}) = \sum_{i=1}^{l} \alpha_i y_i \langle \phi(\overrightarrow{x_i}) \cdot \phi(\overrightarrow{x}) \rangle + b \tag{2.9}$$

We call the function $K(\overrightarrow{a}, \overrightarrow{b}) = \langle \phi(\overrightarrow{a}) \cdot \phi(\overrightarrow{b}) \rangle$ the *kernel* of the SVM. The following standard kernels are commonly used:

- Linear: $K(\overrightarrow{a}, \overrightarrow{b}) = \langle \overrightarrow{a} \cdot \overrightarrow{b} \rangle$
- Polynomial: $K(\overrightarrow{a}, \overrightarrow{b}) = (\gamma \langle \overrightarrow{a} \cdot \overrightarrow{b} \rangle + c_0)^d$
- Radial Basis Function: $K(\overrightarrow{a}, \overrightarrow{b}) = exp(-\gamma \| \overrightarrow{a} - \overrightarrow{b} \|^2)$

Other kernels may be used to solve a specific problem. Riemannian geometry-based kernels[90] were developed by Amari and Wu to analyze breast cancer data. Xiong *et al.* used data-dependent kernel machines to classify microarray data[91]. Liu applied Gabor-based kernel to face recognition[92]. In this work, we are interested in linear classifications. Hence, we choose a linear kernel defined as in Equation 2.10:

$$K(\overrightarrow{a}, \overrightarrow{b}) = \langle \overrightarrow{a} \cdot \overrightarrow{b} \rangle = \sum_{j=1}^{n} a_j b_j \tag{2.10}$$

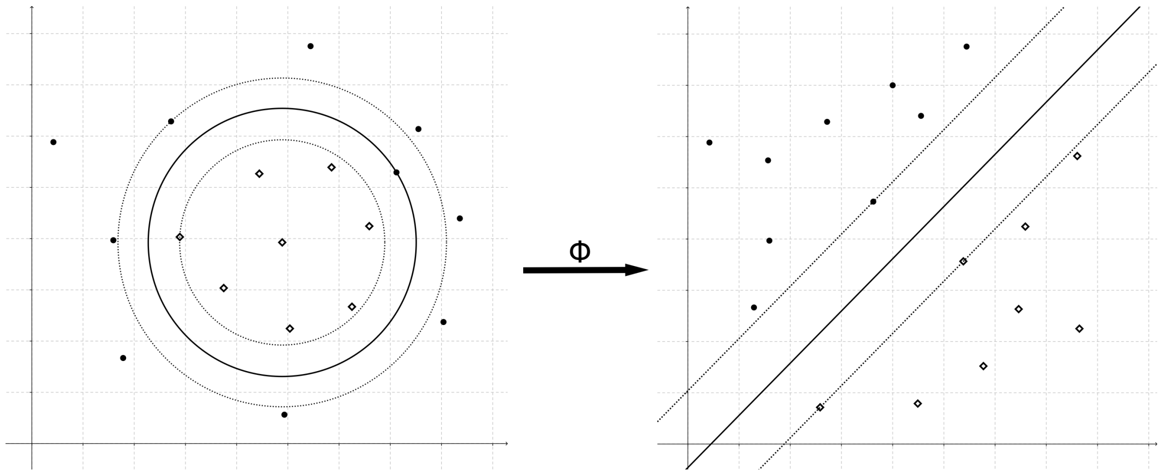


Figure 2.16: Mapping input data in to a higher dimensional features space. Problems that are not linearly separable in input space can be linearly separable in feature space.

Let $\overrightarrow{x} = (x_1, x_2, ..., x_n)$. Combining Equations 2.9 and 2.10, $f(\overrightarrow{x})$ becomes:

$$\begin{aligned} f(\overrightarrow{x}) &= \sum_{i=1}^{l} \left(\alpha_i y_i \sum_{j=1}^{n} x_{ij} x_j \right) + b \\ f(\overrightarrow{x}) &= \sum_{j=1}^{n} \left(\sum_{i=1}^{l} \alpha_i y_i x_{ij} \right) x_j + b \\ f(\overrightarrow{x}) &= \sum_{j=1}^{n} w_j x_j + b \end{aligned} \tag{2.11}$$

with $w_j = \sum_{i=1}^{l} \alpha_i y_i x_{ij}$.

2.3.3 Decision Trees

Decision trees were frequently used in the nineties by artificial intelligence experts because they can be easily implemented and they provide an *explanation* of the result. A decision tree is a tree with the following properties:

- Each internal node tests an attribute,
- Each branch corresponds to the value of the attribute,
- Each leaf assigns a classification.

The output of decision trees is a set of logical rules in the form of disjunctions of conjunctions. There exists a number of algorithms to train a decision tree, for example Quinlan's Iterative Dichotomiser 3[11] and C4.5[93] algorithms or Breiman's Random forests[94]. In our experiments, we used the Iterative Dichotomiser 3 (ID3) algorithm, which is is defined as follows:

1. The best attribute, A, is chosen for the next node. The best attribute maximizes the *information gain* as defined below in Equation 2.12.

$$gain(S, A) = entropy(S) - \sum_{v \in values(A)} \frac{|S_v|}{|S|} entropy(S_v) \tag{2.12}$$

   where S is a sample of the training examples and A is a partition of the parameters. Like in thermodynamics, the entropy measures the *impurity* of S, purer subsets having a lower entropy:

$$entropy(S) = -\sum_{i=0}^{n} p_i log_2(p_i) \tag{2.13}$$

   where S is a sample of the training examples, p_i is the proportion of *i-valued* examples in S, and n is the number of attributes.

2. We create a descendant for each possible value of the attribute A.

3. For each non-perfectly classified branch repeat steps (1) and (2).

ID3 is a greedy algorithm, without backtracking. This means that this algorithm is sensible to local optima. Furthermore, ID3 is inductively biased: the algorithm favors short trees and high information gain attributes near the root. At the end of the procedure, the decision tree perfectly suits the training data including noisy data. This leads to complex trees, which usually lead to problems in classifying new data.

Ockham's razor stipulates that "entities must not be multiplied beyond necessity", that is, if two theories make the same predictions the simplest one should be preferred. Following this general principle, to avoid over-fitting, complex decision trees are usually pruned after the training stage by minimizing $size(tree) + error\_rate$, where $size(tree)$ is the number of leaves in the tree, and $error\_rate$ is the ratio of the number of misclassified instances and the total number of instances. Finally, $accuracy$ is defined as $accuracy = 1 - error\_rate$.

"*Entia non sunt multiplicanda praeter necessitatem*"

William of Ockham (1883)

It is interesting to compare decision trees with SVMs. Unlike decision trees, SVMs are not subject to local optimal issues because SVMs are guaranted to find the global optimal. Hence, SVMs are expected to generalize better than other machine learning algorithms such as decision trees. On the other hand, decision trees have the advantage of providing an explanation of their classifications, while SVMs do not provide any explanations.

2.4 Constraint Databases

Constraint databases, which were initiated by Kanellakis *et al.*[14], have many applications ranging from spatial databases[95,96] through moving objects[97,98] to epidemiology[99]. Geist[100] and Johnson *et al.*[101] applied them to classsification problems. In particular, both discussed the representation of decision trees by constraint databases. However, they did not consider the case of Support Vector Machines, nor solved the reclassification problem.

Constraint databases[14,15,16] form an extension of relational databases[102] where the database can contain variables that are usually constrained by linear or polynomial equations. Constraint databases are able to represent infinite amounts of data, as long as it is finitely representable using constraints. For example, spatio-temporal data, which are common in the medical and geographic areas, typically contains real-valued objects that change over time. Constraint databases provide an elegant means to represent such objects and can be queried using both Datalog and SQL queries[103,104,105]. Constraint database systems include CCUBE[106], DEDALE[107], IRIS[108], and MLPQ[109]. We used IRIS and MLPQ during our experiments.

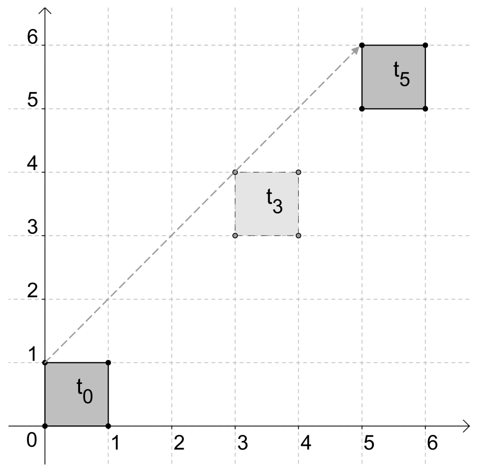


Figure 2.17: Moving square that may be efficiently represented in a constraint database.

Example 2.2 For example, Figure 2.17 shows a moving square, which at time $t = 0$ starts at the first square of the first quadrant of the plane and moves to the northeast with a speed of one unit per second to the north and one unit per second to the east.

When $t = 0$, then the constraints are $x \geq 0, x \leq 1, y \geq 0, y \leq 1$, which is the unit square in the first quadrant. We can calculate similarly the position of the square at any time $t > 0$ seconds. For example, when $t = 5$ seconds, then the constraints become $x \geq 5, x \leq 6, y \geq 5, y \leq 6$, which is another square with lower left corner $(5, 5)$ and upper right corner $(6, 6)$. In general, for any time t, the constraints describing the moving square in a constraint database can be expressed as in Table 2.5.

2.5 Inverse Distance Weighted Interpolation

Most scientific experiments are based on data collection. A data set thus formed is usually composed of discrete values obtained after some data sampling method. For example, in the case of meteorological studies, the temperature may be measured every hour. However, although this meteorological data set will give accurate values when the temperature was measured, it does not provide any information about the temperature evolution between

Table 2.5: Constraint database representation of a moving square

X	Y	T	
x	y	t	$x \geq t, x \leq t+1, y \geq t, y \leq t+1, t \geq 0$

two measurements. In addition, data may be collected only in a few particular locations. In the case of weather data collection, the temperature is typically measured in weather stations only. Hence, although the temperature is known at those few locations, there is no data for other locations.

In order to estimate values between two measurements, a number of *regression* algorithms have been devised. Regression is a statistical method that aims at building the *continuous* curve function r that will fit best the known values. Note that in general, for any known measure y collected at time t, $r(t) \neq y$, that is, the known values do not belong to the best-fit curve. However, the most important characteristic of regression curves is their ability to estimate values between two measurements. The earliest form of regression was the *method of least squares*[110], which was used by the French mathematician Legendre in 1805 to estimate orbits of comets whose positions could be measured only at night. *Interpolation* is a special case of regression where the known measurement belong to the best-fit curve. As a result, interpolated curves return exact values when data is known, but at a cost of a slight increase of the overall error rate.

One way to achieve interpolation is to average all the measured values. This makes good sense when the measurements are really independent of location. However, a variable defined

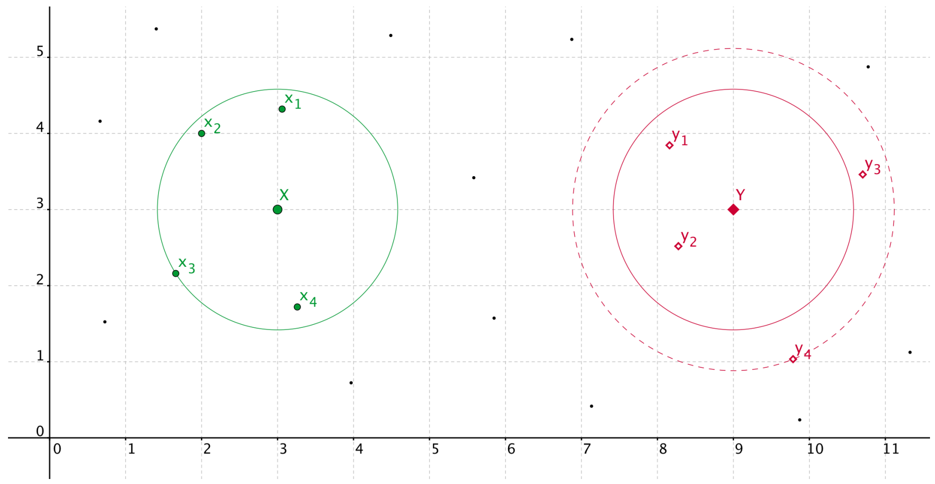


Figure 2.18: The surrounding area of an interpolated point may be obtained using a fixed radius (solid circle), resulting in various number of surrounding points ; or using a fixed number of surrounding points, resulting in various surroundings radii (dashed circle).

continuously over part of a surface does not normally change wildly from point to point. We can reasonably expect the values at unmeasured points to be related to the values at nearby measured points. The interpolated value is then obtained by averaging values of surrounding points only (see Figure 2.18). The method is called the *k-nearest-neighbor interpolation*. Surrounding points may be obtained by defining a fixed radius (solid circles) around the interpolated value. Therefore, the set of surrounding points may contain different numbers of points, possibly zero points, in which case, the interpolated value cannot be obtained. Another method is to consider a fixed number of points to calculate the interpolated value. In the example of Figure 2.18, 4 points are used to calculate X and Y. As a result, the radius of the surrounding area around Y must be increased (dashed circle). Using this method, the interpolated value can always be computed. However, it may not be always meaningful as surrounding points may be at a large distance.

A natural extension of this simple algorithm is based on the assumption that the interpolated values should be mostly influenced by nearby points and less by the more distant points. Hence, the value of neighboring values can be weighted based on their distance to the interpolated point. The interpolation algorithm is known as the Inverse Distance Weighted (IDW) interpolation and is one of the most commonly used techniques for interpolation of scatter points. This method was used during our experiments on temporal classification in Chapter 9 to compare and evaluate the proposed temporal classification. The value v of the interpolated point x is the weighted average of the k-nearest points x_i as defined in Equation 2.14, where v_i is the value of x_i and $w_i(x)$ is the weight of x_i based on its distance to x.

$$v(x) = \frac{\sum\limits_{i=0}^{k} w_i(x) v_i}{\sum\limits_{i=0}^{k} w_i(x)} \tag{2.14}$$

The weight function is normally largest at zero distance and decrease as the distance increases. The most commonly used weighting function (defined in Equation 2.15) is based on the power function and was introduced by Shepard[111], where $d(x, x_i)$ is the distance between x and x_i and $p \in \mathbb{R}^+$ is called the *power parameter*.

$$w_i(x) = \frac{1}{d(x, x_i)^p} \tag{2.15}$$

Combining Equations 2.14 and 2.15 gives Equation 2.16

$$v(x) = \frac{\sum_{i=0}^{k} \frac{1}{d(x, x_i)^p} v_i}{\sum_{i=0}^{k} \frac{1}{d(x, x_i)^p}} \tag{2.16}$$

Let $(a, b) \in \mathbb{R}^2$ be two real constants. We defined $v_{a,b}(x)$ as the value of the interpolated point x obtained after multiplying the individual values v_i by a and adding b. We obtain:

$$\begin{aligned}
v_{a,b}(x) &= \frac{\sum_{i=0}^{k} \frac{1}{d(x, x_i)^p}(av_i + b)}{\sum_{i=0}^{k} \frac{1}{d(x, x_i)^p}} \\
v_{a,b}(x) &= \frac{a\sum_{i=0}^{k} \frac{1}{d(x, x_i)^p} v_i + b\sum_{i=0}^{k} \frac{1}{d(x, x_i)^p}}{\sum_{i=0}^{k} \frac{1}{d(x, x_i)^p}} \\
v_{a,b}(x) &= a\frac{\sum_{i=0}^{k} \frac{1}{d(x, x_i)^p} v_i}{\sum_{i=0}^{k} \frac{1}{d(x, x_i)^p}} + b \\
v_{a,b}(x) &= av(x) + b \qquad (2.17)
\end{aligned}$$

Equation 2.17 proves that the IDW interpolation is *linear*. The linear property of IDW interpolation was used by Revesz *et al.*[112,113] to give a constraint database representation of the interpolator, which was used for spatiotemporal reasoning about epidemiological data[99]. In this work, we evaluated our *temporal classification* (see Chapter 9) by comparing it with the constraint database representation of IDW.

Chapter 3

PROFESS: PROtein Function, Evolution, Structure and Sequence

“*With our query, you type the query once,*
and it will federate the results from all these sources together.”

Lubor Ptacek

3.1 Introduction

Life sciences techniques made significant improvements over the past decades, resulting in huge amounts of data, collected over the years by the scientific community. In order to facilitate the organization and the subsequent analyses of this valuable data, *databases* have been developed very early. Databases may be broadly defined as organised collections of data, structured so that data can be automatically retrieved or manipulated. One of the first biological databases was the ”Atlas of Protein Sequences and Structures” by Margaret Dayhoff[114]. It was first published as a book in 1965. It contained the protein sequences determined at the time, and updated editions of the book were published in the 1970s[115]. Its data became the foundation for the Protein Information Resource database[116] (PIR). At that time, databases existed in the form of a set of index cards, each containing the information for one of the entities. Since then, the development and rapid advances in electronics led to the development of database models that are far more efficient for dealing with large volumes of information than flat databases. The most notable advance is the relational model, which was proposed by Codd in 1970[102], making databases an efficient tool to support the research community. Since then, the number of databases has dramatically increased[117], and the 2009 Molecular Biology Database Collection[118] includes 1,170 databases, each containing thousands, if not millions, of entries. These databases constitute the extent of our knowledge related to genomics, proteomics, metabolomics and structural genomics. However, most serve only as data warehouses with simple interfaces for data retrieval[3].

3.2 Database Integration Problem

To address increasingly complex questions, biologists are routinely required to develop new databases by filtering information from existing databases[119]. Even though this is extremely inefficient, there are a growing number of specialized databases designed around single topics. Unfortunately, this simply propagates the underlying inability to utilize the data outside the constraints imposed by the database designers[120].

As an example, the Protein FAMilies[121] (PFAM) database contains large collection of high quality, manually curated families, which are helpful to identify conserved domains that occur within proteins and therefore provide insights into the proteins functions. Besides sequence-based queries using BLAST[74], the primary means to mine data is based on a randomly assigned accession number or keywords. There exist only very limited interactions with other databases. As a consequence, the set of possible queries available to mine the data is limited as well. In particular, it is not possible to search PFAM for enzyme classes[122] that are common to a specific family for example.

Capitalizing on the potential of biological information requires the development of a next-generation database that enables biologists to explore biological data in new ways. The key to solving this problem is to move the design focus from the database structure to a fluid association that can be adapted to a biologists questions[123] without re-designing the underlying data structure. However, there are barriers to linking individual databases because of different data formats and structure[124,125].

Thus, it was essential to this effort to implement a new approach to integrate diverse biological databases[1,126]. Satisfying, general and practical solutions are still lacking for the integration of business or spatio-temporal data[127] and are rarely able to handle the additional complexities of biological data. The most versatile of these solutions is to use a separate adapter – or wrapper – program around each source database[128]. The wrappers provide a simplified *view* of the source database, presented in a form that is understandable and easier-to-use than the original source database. In fact, some parts of the source data may be completely omitted in this repacked presentation, leaving only the parts of the data that are needed for the user who wants to use it.

The advantage of the "answering queries using views" approach to the database integration problem is that it reduces the integration problem to the two steps of:

1. Building wrappers of the source databases, thereby providing simple views,
2. Applying standard database queries on the views.

Thus, implementing wrappers will also enable a robust query system that incorporates a variety of similarity functions capable of generating data relationships not conceived during the creation of the database, allowing the user-interface to move beyond the simple text-based queries that dominate existing databases.

3.3 Overview of PROFESS

The PROFESS (PROtein Function, Evolution, Structure and Sequence) database was initially developed as a resource to measure structure plasticity of orthologous families upon taxonomic divergence (see Chapter 4). It was containing over 65,000 pairwise protein structure comparisons generated using DaliLite and more than 1,000 clusters of orthologous proteins[129] (COG). To understand the biological significance of structure divergence, fourteen additional sources of data were added to PROFESS (Table 3.1, see Section 3.4 for details) using a Local-As-View (LAV) modular approach (Section 3.5).

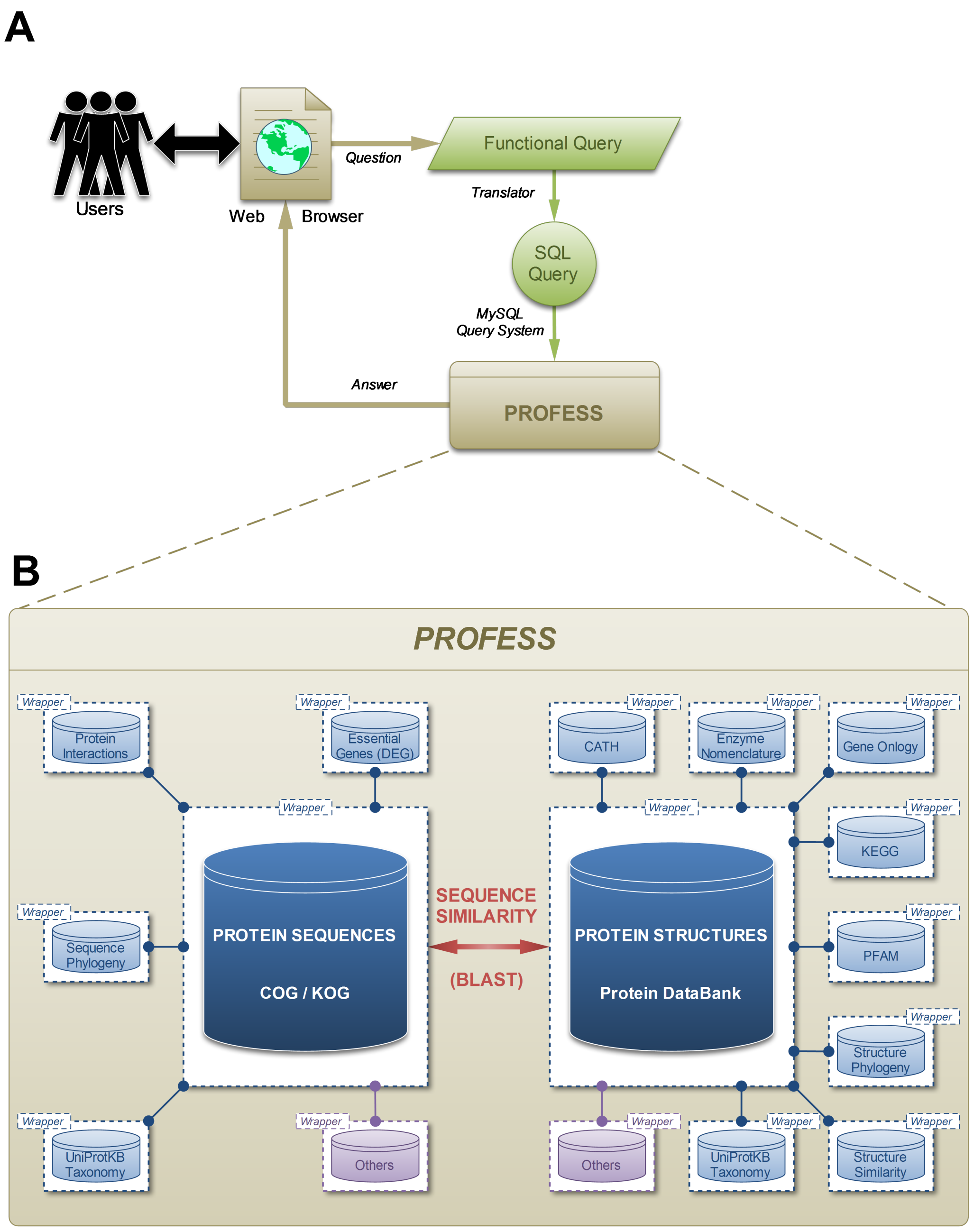


Figure 3.1: Outline of the PROFESS database. A) the relationship of the user interface to the functional query system (green) to the PROFESS databases; and B) the core databases integrated in PROFESS. The central COG-PDB linkage is shown in red, double arrows indicate intensive interactions, blue boxes represent databases available on the Internet, and purple boxes denote other databases to be integrated in the future. Each additional data set interacts with the PROFESS core through the use of wrapper programs to make the query language uniform.

Table 3.1: Core databases currently integrated in PROFESS (last update May 2009)

Name	Code	Level	Link	Ref.
CATH database	CATH	Structure	`http://www.cathdb.info/`	51
Clusters of Orthologous Groups of proteins	COG	Function	`http://www.ncbi.nlm.nih.gov/COG/`	129
Enzyme Classification	EC	Function	`http://www.chem.qmul.ac.uk/iubmb/enzyme/`	122
Database of Essential Genes	DEG	Evolution	`http://www.essentialgene.org/`	130
Database of Interaction Proteins	DIP	Function	`http://dip.doe-mbi.ucla.edu/`	131
Functional structure/sequence-based phylogeny		Evolution	Proposed in Chapter 4	
Functional structure similarity comparisons		Structure	Proposed in Chapter 4	
Gene Ontology	GO	Function	`http://www.geneontology.org/`	132
GenBank	GENBANK	Sequence	`http://www.ncbi.nlm.nih.gov/Genbank/`	133
KEGG Ligands	KEGG	Function	`http://www.genome.jp/kegg/ligand.html`	134
Protein Data Bank	PDB	Structure	`http://www.rcsb.org/`	7
Protein Families database	PFAM	Function	`http://pfam.sanger.ac.uk/`	121
Protein/Protein interactions in *E. coli*	PIN	Function	`http://genome.cshlp.org/content/16/5/686`	135
Structural Classification of Proteins	SCOP	Structure	`http://www.bio.cam.ac.uk/scop/`	50
UniProtKB Taxonomy	NEWT	All	`http://www.uniprot.org/taxonomy/`	136

PROFESS was designed to assist in the functional and evolutionary analysis of proteins continually identified from whole-genome sequencing. Hence, a requirement for PROFESS was to be both versatile and extendable. For these reasons, the PROFESS database was created using wrappers of a Local-As-View (LAV) approach (Section 3.5). The web interface was also designed using a modular approach (Section 3.7), each module representing a unique view of the data.

Figure 3.1 outlines the structure of our system: users interact with PROFESS through a web interface using a functional-style query language that is translated to SQL for mining PROFESS (A). The core of PROFESS (B) consists of the COG-PDB relationship (Section 3.4.1). Other databases interact with PROFESS core through the use of wrappers. As a consequence, all integrated databases can be uniformly queried (Section 3.6). The flexible design of PROFESS coupled with user friendly searching capabilities makes PROFESS particularly useful for asking a range of questions about the sequence, structure and functional relationship of orthologous proteins.

PROFESS is freely accessible through the URL `http://cse.unl.edu/~profess`. Data can be downloaded as parseable files in Comma Separated Values (CSV) format from the web-interface or using HTTP GET requests that may be batched in scripts.

Table 3.2: List of modules available in the user interface of PROFESS

Name	Level	Data sources
Essential genes	Evolution	COG, DEG, NEWT, PDB
Functions	Function	COG, EC, GO, KEGG, NEWT, PDB, PFAM
Functions summary	Function	COG, EC, GO, PDB, PFAM
Ligands	Function	COG, KEGG, PDB
Protein interactions	Function	COG, PIN
Sequences	Sequence	COG, GENBANK, NEWT, PDB
Sequence-based phylogeny	Evolution	COG, GENBANK, PDB
Sequence similarities	Sequence	COG, GENBANK, PDB
Structures	Structure	CATH, COG, NEWT, PDB, SCOP
Structure-based phylogeny	Evolution	COG, PDB
Structural comparisons	Structure	COG, NEWT, PDB

3.4 Database Content

3.4.1 Functional Annotation of the Protein Data Bank

The core of PROFESS established a relationship between the Protein Data Bank[7] (PDB) and the cluster of orthologous groups[129] (COG) databases (Figure 3.1B). The Protein Data Bank is the main repository for protein structures and contains 55,159 structures*. The most recent COG database was created by finding the genome-specific best-hit for each gene in 66 unicellular genomes (50 bacteria, 13 archaea, and 3 eukaryota). Specifically, the orthologs present in three or more genomes were detected automatically and then multidomain proteins were manually split into component domains to eliminate artifactual lumping. The COG database contains 192,987 sequences distributed among 4,876 COGs, accounting for 75% of genes in these 66 genomes.

Assignment of each bacterial protein in the PDB to a COG number in the clusters of orthologous groups database required downloading the complete sequence lists from both databases. The BLAST algorithm[74] implemented with the Protein Mapping and Comparison Tool[137] (PROMPT) was used to match sequences in the PDB with COG and KOG sequences using the BLOSUM62 matrix with a gap penalty of 11, a gap extension penalty of 1 and a word size of 5 and a BLAST relatively low expectation threshold (E-value) of 10^{-9}. This low E-value was used to unambiguously match genes in the COG database with proteins in the PDB instead of simply match homologous sequences.

As of April 28, 2009, the BLAST search gave a hit rate of 74.8% of total PDB protein sequences (41,273 of 55,159 total sequences) matching the COG or KOG databases. Of the 41,273 PDB/COG hits, 54.2% matched with greater than 50% sequence identity with 17% giving 100% identity. All BLAST hits are displayed in the *Sequence Similarities* module of the web interface. The BLAST required 3 days of intensive calculations running on a 2.80 GHz Intel CPU with 1.5 GB of RAM. The output from the BLAST is reformatted and stored in our database. This functional annotation of the Protein DataBank is also the core component of our *structural comparison of functional orthologs*, which is described in details in Section4.2.

*As of April 2009

3.4.2 Functional Level

The Function level of PROFESS contains three unique modules that describe the primary biological function of a cluster of orthologs, in addition to a "summary" module which provides an overview of most relevant classifications.

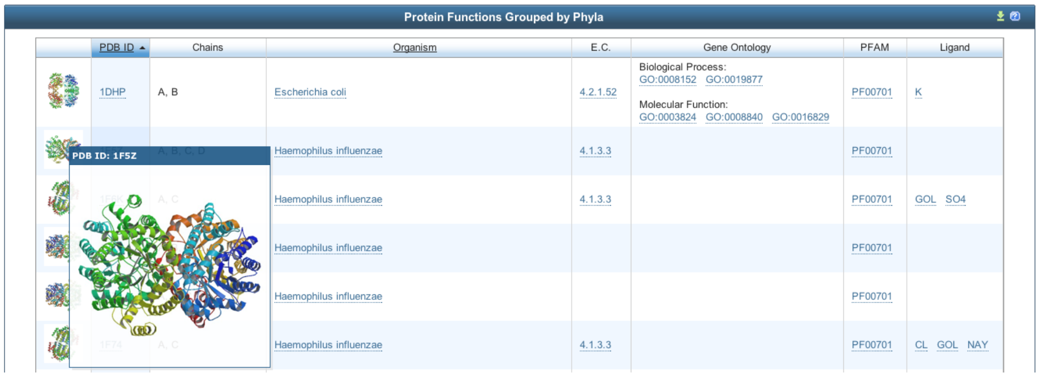
Module Functions

The module *Functions* (Figure 3.2a) is a table of various functional annotations for a protein structure in the Protein DataBank including the PFAM, the Gene Ontology, and Enzyme Classification. The PFAM database contains large collection of high quality, manually curated families (version 23.0 includes 10,340 families), which are helpful to identify conserved domains that occur within proteins. Families are described by multiple sequence alignments in addition to probabilistic models called hidden Markov models. The Gene Ontology provides an ontology of 27,606 defined terms representing gene properties covering three domains: *cellular components*, the parts of a cell or its extracellular environment; *molecular functions*, the elemental activities of a gene at the molecular level; and *biological processes*, sets of molecular events essential for the functioning of organisms. The Enzyme Nomenclature defines a hierarchical numerical classification for enzymes, based on the chemical reactions they catalyze. The current sixth edition contains 3,196 different classes.

Protein function can also be described by protein interaction partners, therefore two additional modules (Ligands and Protein Interactions) list the ligands and proteins experimentally shown to interact with members of the COG family.

Module Ligands

In biochemistry, a *ligand* is a substance that is able to bind to and form a complex with a biomolecule to serve a biological purpose. For example, it may act as a signal triggering molecule, binding to a site on a target protein. The *Ligands* module (Figure 3.2b) displays details about ligands known to bind a protein based on ligand bound structures from the Protein DataBank as well as cross-references to the Kyoto Encyclopedia of Genes and Genomes[134] (KEGG). In particular, we provide the list of protein structures within the orthologous cluster that bind each of the listed ligands. The Protein DataBank provides a

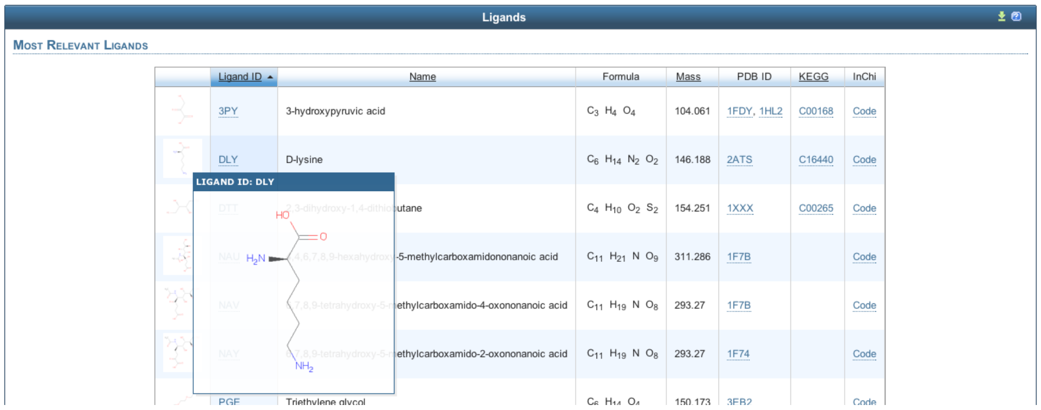


Protein Functions Grouped by Phyla

	PDB ID	Chains	Organism	E.C.	Gene Ontology	PFAM	Ligand
	1DHP	A, B	Escherichia coli	4.2.1.52	Biological Process: GO:0008152 GO:0019877 Molecular Function: GO:0003824 GO:0008840 GO:0016829	PF00701	K
	1F5Z	A, B, C, D	Haemophilus influenzae	4.1.3.3		PF00701	
	1F6K	A, C	Haemophilus influenzae	4.1.3.3		PF00701	GOL SO4
			Haemophilus influenzae			PF00701	
			Haemophilus influenzae			PF00701	
	1F74	A, C	Haemophilus influenzae	4.1.3.3		PF00701	CL GOL NAY

PDB ID: 1F5Z

(a) Module Functions



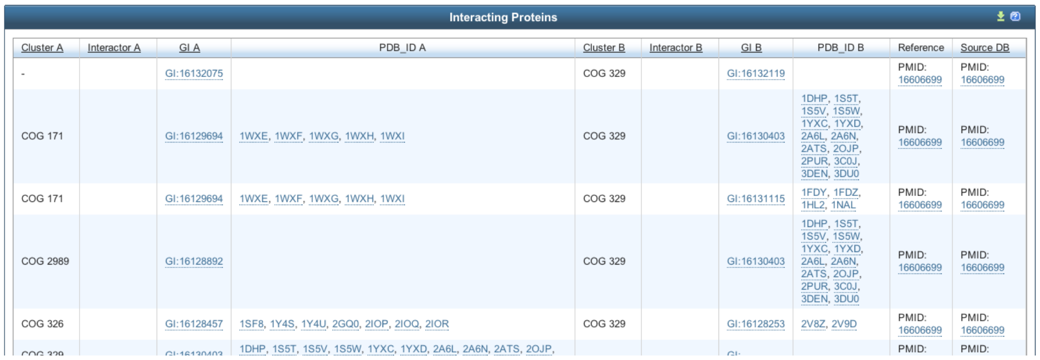
Ligands

Most Relevant Ligands

	Ligand ID	Name	Formula	Mass	PDB ID	KEGG	InChi
	3PY	3-hydroxypyruvic acid	C_3 H_4 O_4	104.061	1FDY, 1HL2	C00168	Code
	DLY	D-lysine	C_6 H_{14} N_2 O_2	146.188	2ATS	C16440	Code
	DTT	2,3-dihydroxy-1,4-dithiobutane	C_4 H_{10} O_2 S_2	154.251	1XXX	C00265	Code
	NAU	4,6,7,8,9-hexahydroxy-5-methylcarboxamidononanoic acid	C_{11} H_{21} N O_9	311.286	1F7B		Code
	NAV	6,7,8,9-tetrahydroxy-5-methylcarboxamido-4-oxononanoic acid	C_{11} H_{19} N O_8	293.27	1F7B		Code
	NAY	6,7,8,9-tetrahydroxy-5-methylcarboxamido-2-oxononanoic acid	C_{11} H_{19} N O_8	293.27	1F74		Code
	PGE	Triethylene glycol	C_6 H_{14} O_4	150.173	3FB2		Code

LIGAND ID: DLY

(b) Module Ligands



Interacting Proteins

Cluster A	Interactor A	GI A	PDB_ID A	Cluster B	Interactor B	GI B	PDB_ID B	Reference	Source DB
-		GI:16132075		COG 329		GI:16132119		PMID: 16606699	PMID: 16606699
COG 171		GI:16129694	1WXE, 1WXF, 1WXG, 1WXH, 1WXI	COG 329		GI:16130403	1DHP, 1S5T, 1S5V, 1S5W, 1YXC, 1YXD, 2A6L, 2A6N, 2ATS, 2OJP, 2PUR, 3C0J, 3DEN, 3DU0	PMID: 16606699	PMID: 16606699
COG 171		GI:16129694	1WXE, 1WXF, 1WXG, 1WXH, 1WXI	COG 329		GI:16131115	1FDY, 1FDZ, 1HL2, 1NAL	PMID: 16606699	PMID: 16606699
COG 2989		GI:16128892		COG 329		GI:16130403	1DHP, 1S5T, 1S5V, 1S5W, 1YXC, 1YXD, 2A6L, 2A6N, 2ATS, 2OJP, 2PUR, 3C0J, 3DEN, 3DU0	PMID: 16606699	PMID: 16606699
COG 326		GI:16128457	1SF8, 1Y4S, 1Y4U, 2GQ0, 2IOP, 2IOQ, 2IOR	COG 329		GI:16128253	2V8Z, 2V9D	PMID: 16606699	PMID: 16606699
COG 329		GI:16130403	1DHP, 1S5T, 1S5V, 1S5W, 1YXC, 1YXD, 2A6L, 2A6N, 2ATS, 2OJP,			GI:		PMID:	PMID:

(c) Module Protein/Protein Interactions

Figure 3.2: Three modules (top rows only) from the *Function* level of the PROFESS database for Dihydrodipicolinate synthases (COG 329). Moving the cursor over structure or ligand thumbnails activates a popup with a larger image.

similar functionality. However, a major difference with PROFESS is that a search on the PDB for a specific ligand will return the list of *all* bound protein structures. PROFESS only returns those which are orthologous thus expected to have similar functions. To provide rapid access to biologically relevant data, common buffers, detergents, ions and solvents are listed separately.

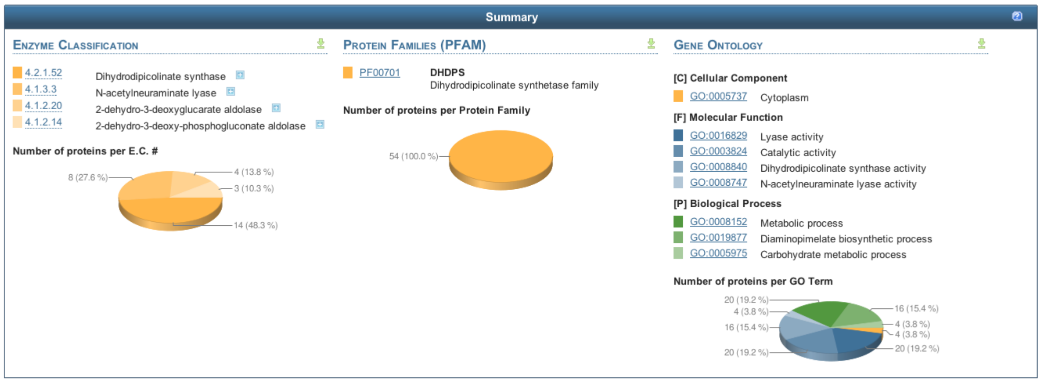
Module Protein/Protein Interactions

The interactions between proteins are critical for most biological functions. For example, signal transduction is based on interactions between extra-cellular signaling molecules and membrane proteins, and plays a fundamental role in many biological processes and in many diseases. The *Protein Interactions* module (Figure 3.2c) lists protein interactions found in *E. coli.* The data was extracted from Arifuzzaman *et al.*[135] and interactions were correlated to the corresponding protein structures by matching bait and prey genes to their representative COG. The Protein Interactions module also integrates the 55,692 manually curated protein/protein interactions (as of August 2009) in 243 organisms from the Database of Interacting Proteins[131].

Module Functional Summary

The Function level of PROFESS also summarizes the main biological functions of an orthologous cluster (example of *Dihydrodipicolinate synthases* (COG 329) shown in Figure 3.3). For three primary descriptions of protein function – the Protein Families[121], the Enzyme Classification[122] and the Gene Ontology[132] – , the numbers of proteins within each class (within the current orthologous cluster) are computed and the distributions are represented as pie charts. This synthetic view allows the user to quickly differentiate relevant classes from outliers. Classes are sorted by decreasing number of proteins. The darker the color in the pie chart, the higher the number of proteins. We implemented PHP scripts to generate the raw data whereas the pie chart are generated using the Google Chart API*. While the webpage displays chart thumbnails, the pie charts can also be downloaded in high-definition.

*Documentation available at `http://code.google.com/apis/chart/`



ENZYME CLASSIFICATION

🟧	4.2.1.52	Dihydrodipicolinate synthase
🟧	4.1.3.3	N-acetylneuraminate lyase
🟧	4.1.2.20	2-dehydro-3-deoxyglucarate aldolase
🟧	4.1.2.14	2-dehydro-3-deoxy-phosphogluconate aldolase

Number of proteins per E.C. #

PROTEIN FAMILIES (PFAM)

🟧	PF00701	DHDPS Dihydrodipicolinate synthetase family

Number of proteins per Protein Family

GENE ONTOLOGY

[C] Cellular Component

🟧	GO:0005737	Cytoplasm

[F] Molecular Function

🟦	GO:0016829	Lyase activity
🟦	GO:0003824	Catalytic activity
🟦	GO:0008840	Dihydrodipicolinate synthase activity
🟦	GO:0008747	N-acetylneuraminate lyase activity

[P] Biological Process

🟩	GO:0008152	Metabolic process
🟩	GO:0019877	Diaminopimelate biosynthetic process
🟩	GO:0005975	Carbohydrate metabolic process

Number of proteins per GO Term

Figure 3.3: The module *Function Summary* represents the distribution of the proteins in all PFAM, E.C. and GO classes within the current orthologous cluster.

3.4.3 Phylogenetic Level

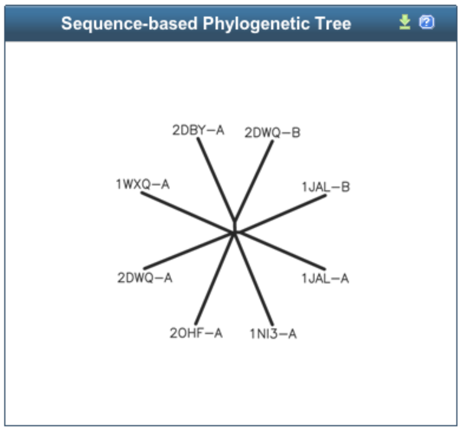
Phylogeny is the study of the evolution of organisms and describes how they evolved together. The evolution and relations between organisms or proteins of interested are usually represented in an evolution – or phylogenetic – tree. The Evolution level of PROFESS displays a table of essential genes, and sequence and structure-based phylogenetic trees.

Module Sequence-based Phylogeny

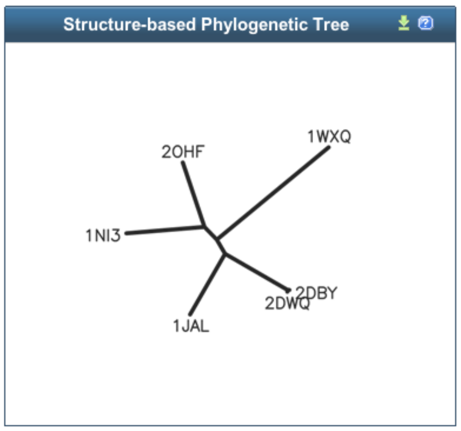
The *Sequence Tree* (Figure 3.4a) shows the unrooted phylogenetic tree generated using protein chain sequences from the PDB. First, the sequences were aligned using ClustalW2[65]. Second, the tree was computed using ClustalW2 using the multiple sequence alignment as a guide. The final image was generated using DrawTree from the PHYLIP package[138].

Module Structure-based Phylogeny

The *Structure Tree* (Figure 3.4b) module shows the unrooted phylogenetic tree generated using protein structures from the PDB. To increase throughput, the structures were aligned using MAMMOTH-mult[82], a tool for multiple structure alignments. The tree was computed by ClustalW2 using the multiple structure alignment as a guide. The final image was generated using DrawTree. The two phylogenetic trees allow a quick visual comparison of the divergence between sequences and structures of orthologous proteins. In both cases, the tree can be downloaded as an ASCII file in PHYLIP format as well as in high-definition picture.



(a) Sequence-Based Phylogeny



(b) Structure-Based Phylogeny

Figure 3.4: The modules *Structure* and *Sequence-based Phylogeny* for Predicted GTPase (COG 12).

Module Essential Genes

The *Essential Genes* module of the Evolution level shows whether the protein in the orthologous cluster is essential and was obtained from the Database of Essential Genes[130] (DEG). As of version 5.2, DEG includes 5,260 essential prokaryotic genes and 4,808 eukaryotic genes extracted from the literature. Genes are displayed with corresponding protein structures from the PDB.

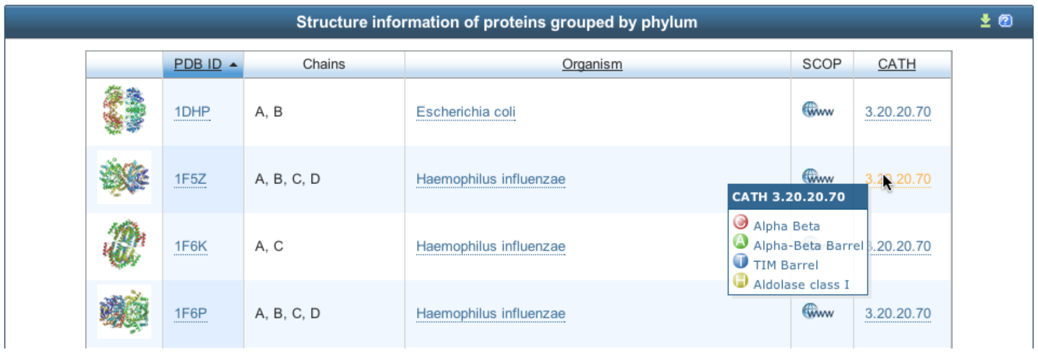
3.4.4 Structural Level

Module Structures

The Structure level of PROFESS contains an aggregates table of data from the CATH Protein Structure Classification[51], the Protein Data Bank and the Structural Classification of Proteins[50] (SCOP) databases and was annotated with the source organism using the UniProtKB taxonomy (Figure 3.5a). The Structure level is particularly useful for finding other orthologous clusters with a particular structure fold.

The CATH (Class, Architecture, Topology and Homologous superfamily) classification is a semi-automatic, hierarchical classification of protein domains. The domains are automatically sorted into classes and clustered on the basis of sequence similarities. These clusters form the H levels of the classification. The topology level is formed by structural comparisons of the homologous groups, the architecture level is assigned manually based on particular structural features. The four CATH classes are defined based on the secondary structures of the proteins: (1) mostly-alpha, (2) mostly-beta, (3) alpha and beta, (4) few secondary structures.

SCOP also describes four levels of hierarchic structural classification: the *class*, which defines eleven general structural architectures of the domain, the *fold* for similar arrangement of regular secondary structures but without evidence of evolutionary relatedness, the *superfamily* where sufficient structural and functional similarity can infer a divergent evolutionary relationship but not necessarily detect sequence homology and the *family* where sequence similarities can be detected. Unlike CATH, SCOP is largely a manual classification. Note that due to copyright issues, we only provide links to retrieve data from SCOP

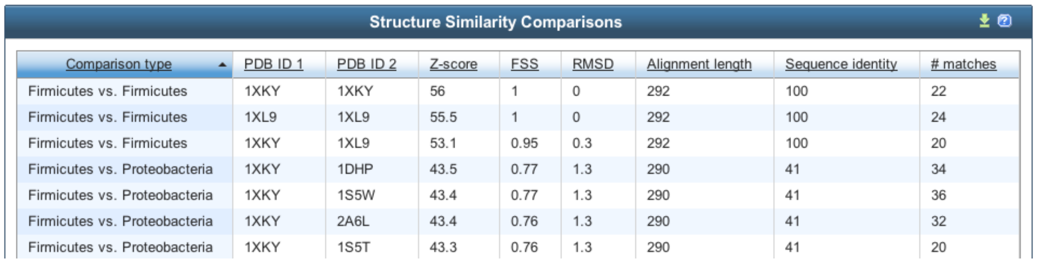


Structure information of proteins grouped by phylum

	PDB ID	Chains	Organism	SCOP	CATH
	1DHP	A, B	Escherichia coli	www	3.20.20.70
	1F5Z	A, B, C, D	Haemophilus influenzae	www	3.2 20.70
	1F6K	A, C	Haemophilus influenzae		.20.20.70
	1F6P	A, B, C, D	Haemophilus influenzae	www	3.20.20.70

CATH 3.20.20.70
- Alpha Beta
- Alpha-Beta Barrel
- TIM Barrel
- Aldolase class I

(a) Module Structure

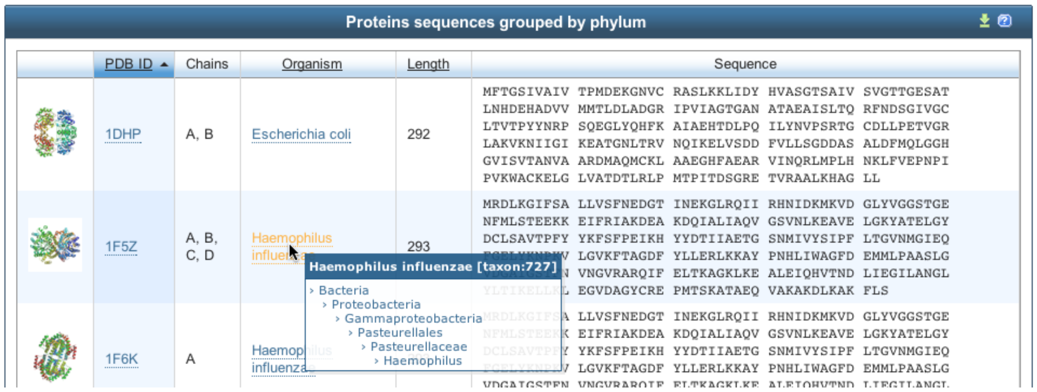


Structure Similarity Comparisons

Comparison type	PDB ID 1	PDB ID 2	Z-score	FSS	RMSD	Alignment length	Sequence identity	# matches
Firmicutes vs. Firmicutes	1XKY	1XKY	56	1	0	292	100	22
Firmicutes vs. Firmicutes	1XL9	1XL9	55.5	1	0	292	100	24
Firmicutes vs. Firmicutes	1XKY	1XL9	53.1	0.95	0.3	292	100	20
Firmicutes vs. Proteobacteria	1XKY	1DHP	43.5	0.77	1.3	290	41	34
Firmicutes vs. Proteobacteria	1XKY	1S5W	43.4	0.77	1.3	290	41	36
Firmicutes vs. Proteobacteria	1XKY	2A6L	43.4	0.76	1.3	290	41	32
Firmicutes vs. Proteobacteria	1XKY	1S5T	43.3	0.76	1.3	290	41	20

(b) Module Structure Comparisons

Figure 3.5: The two modules (top rows only) from the *Structure* level of the PROFESS database for Dihydrodipicolinate synthases (COG 329). Moving the cursor over cross-reference activates tooltips that provide additional information.

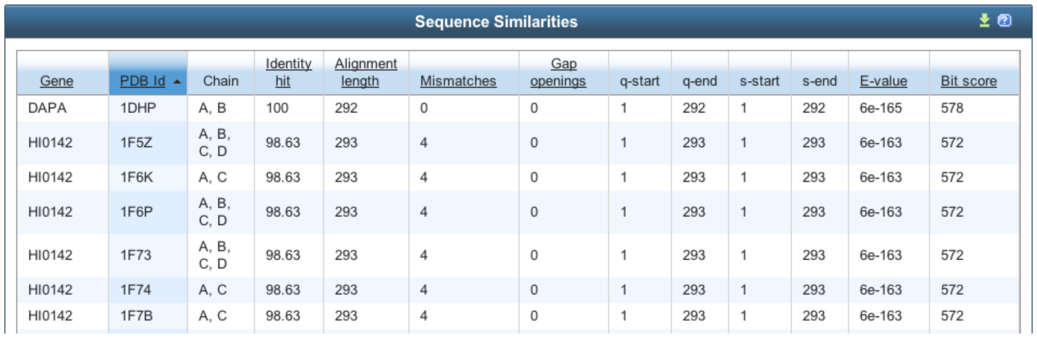


Proteins sequences grouped by phylum

	PDB ID ▲	Chains	Organism	Length	Sequence
	1DHP	A, B	Escherichia coli	292	MFTGSIVAIV TPMDEKGNVC RASLKKLIDY HVASGTSAIV SVGTTGESAT LNHDEHADVV MMTLDLADGR IPVIAGTGAN ATAEAISLTQ RFNDSGIVGC LTVTPYYNRP SQEGLYQHFK AIAEHTDLPQ ILYNVPSRTG CDLLPETVGR LAKVKNIIGI KEATGNLTRV NQIKELVSDD FVLLSGDDAS ALDFMQLGGH GVISVTANVA ARDMAQMCKL AAEGHFAEAR VINQRLMPLH NKLFVEPNPI PVKWACKELG LVATDTLRLP MTPITDSGRE TVRAALKHAG LL
	1F5Z	A, B, C, D	Haemophilus influenzae	293	MRDLKGIFSA LLVSFNEDGT INEKGLRQII RHNIDKMKVD GLYVGGSTGE NFMLSTEEKK EIFRIAKDEA KDQIALIAQV GSVNLKEAVE LGKYATELGY DCLSAVTPFY YKFSFPEIKH YYDTIIAETG SNMIVYSIPF LTGVNMGIEQ FGELYKNPKV LGVKFTAGDF YLLERLKKAY PNHLIWAGFD EMMLPAASLG VDGAIGSTIN VNGVRARQIF ELTKAGKLKE ALEIQHVTND LIEGILANGL YLTIKELLKL EGVDAGYCRE PMTSKATAEQ VAKAKDLKAK FLS
	1F6K	A	Haemophilus influenzae		MRDLKGIFSA LLVSFNEDGT INEKGLRQII RHNIDKMKVD GLYVGGSTGE NFMLSTEEKK EIFRIAKDEA KDQIALIAQV GSVNLKEAVE LGKYATELGY DCLSAVTPFY YKFSFPEIKH YYDTIIAETG SNMIVYSIPF LTGVNMGIEQ FGELYKNPKV LGVKFTAGDF YLLERLKKAY PNHLIWAGFD EMMLPAASLG VDGAIGSTIN VNGVRARQIF ELTKAGKLKE ALEIQHVTND LIEGILANGL

Haemophilus influenzae [taxon:727]
› Bacteria
› Proteobacteria
› Gammaproteobacteria
› Pasteurellales
› Pasteurellaceae
› Haemophilus

(a) Module Sequence



Sequence Similarities

Gene	PDB Id ▲	Chain	Identity hit	Alignment length	Mismatches	Gap openings	q-start	q-end	s-start	s-end	E-value	Bit score
DAPA	1DHP	A, B	100	292	0	0	1	292	1	292	6e-165	578
HI0142	1F5Z	A, B, C, D	98.63	293	4	0	1	293	1	293	6e-163	572
HI0142	1F6K	A, C	98.63	293	4	0	1	293	1	293	6e-163	572
HI0142	1F6P	A, B, C, D	98.63	293	4	0	1	293	1	293	6e-163	572
HI0142	1F73	A, B, C, D	98.63	293	4	0	1	293	1	293	6e-163	572
HI0142	1F74	A, C	98.63	293	4	0	1	293	1	293	6e-163	572
HI0142	1F7B	A, C	98.63	293	4	0	1	293	1	293	6e-163	572

(b) Module Sequence Similarities

Figure 3.6: The two modules (top rows only) from the *Sequence* level of the PROFESS database for Dihydrodipicolinate synthases (COG 329). Moving the cursor over cross-reference activates tooltips that provide additional information.

website rather than reproducing SCOP data on our pages.

Module Structure Comparisons

Additionally, the Structure level contains all pairwise structure alignments of an orthologous cluster (Figure 3.5b). The pairwise structure comparison tool DaliLite was used to measure the backbone structure similarity of proteins within each orthologous cluster defined by the COG database. All-against-all pairwise structural comparisons were carried out for all COGs that were represented by a minimum of two organisms. The Dali Z-scores were normalized to calculate a *Fractional Structure Similarity* (FSS) score:

$$FSS = \frac{Z_{AB}}{max(Z_{AA}, Z_{AA})} \tag{3.1}$$

where Z_{AB} is the Dali Z-score when protein B is compared to protein A and Z_{AA} is the Z-score when protein A is compared to itself. Thus, Z_{AA} represents the maximum Z-score that can be achieved for perfect similarity. FSS provides a simple normalized and quantitative measure of the distance the two proteins have diverged in their structures.

3.4.5 Sequence Level

The Sequence level of PROFESS lists all protein chain sequences within the orthologous cluster from the corresponding PDB structure (Figure 3.6a). The sequences provide the link between the PDB and the COG database as well as cross-references to GenBank[133]. The similarity scores from the PDB to COG link (see Section 3.4.1 for details) are shown in the Sequence Similarities module (Figure 3.6b).

3.5 Local-As-View Data Integration and Database Design

Traditional data integration methods involves data warehousing, where the warehouse database extracts, transforms and loads (ETL) data from various sources into a single schema that is easy to query (Figure 3.8a). However ETL methods lack flexibility because they require the warehouse schema to be tightly coupled with the data sources. As a result, integrating new data sources requires considerable efforts as the entire warehouse and subsequent queries need to be redefined. The warehouse schema may also have to be redesigned in case one of the data sources schema changes after an update.

To address the flexibility issues of widely-used ETL methods, the PROFESS database was designed using a flexible Local-As-View (LAV) method[128,139] as shown in Figure 3.8b. LAV methods involve wrappers that provide a data abstraction layer of the databases to be integrated. Wrappers are software that translate the data sources and provide an abstract, simplified view of the integrated data sources. Although there have been prior integration efforts of structural data and functional data sources, the PROFESS system has a unique approach because it creates two internal wrappers, one for the integrated functional data and another for the integrated structural data and then applies novel functions for the association between these two wrappers. This multi-step integration approach merges the easier-to-integrate data sources first, and then the harder-to-integrate data sources.

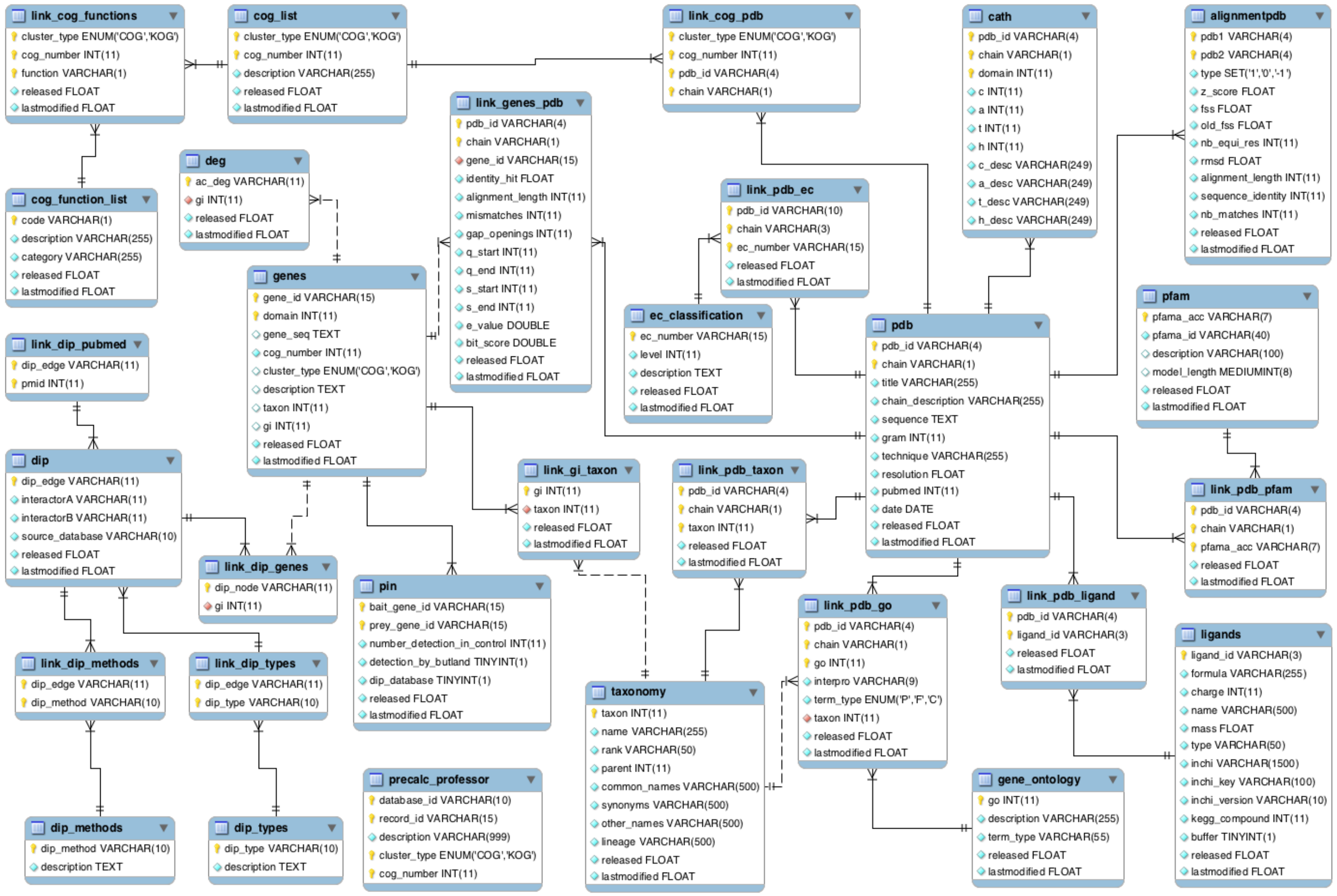


Figure 3.7: The Entity-Relationship diagram of the PROFESS database.

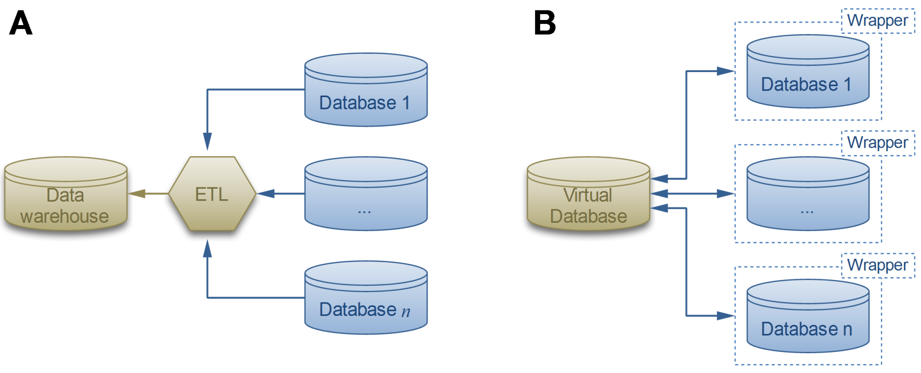


Figure 3.8: Two solutions for the data integration problem. A) The ETL software extracts, transforms and loads the data sources into the warehouse. B) The more flexible *Local-As-View* (LAV) method defines a virtual database that interacts with data sources through wrappers, which provide simplified views of the original databases.

Incomplete and incorrect information in the data source is one of the major difficulties with data integration. By first merging together closely related data sources, our method increases the likelihood that data from different sources will complete and correct each other. In this way, PROFESS will help users overcome such problems as incomplete and misleading data annotations. Structural and functional data are often difficult to integrate because of different identification numbers, different functional definitions, and the absence of a direct link between the two data sources. These problems are solved by using sequence and other related information to link together structural and functional data. Our multi-level integration approach first links all intermediate information to either the functional wrapper or to the structural wrapper. This association then makes it easier to find links between the functional and the structural wrappers using the intermediate information.

The final step to achieve our flexibility and extendibility goals was to normalize our database structure. *Database normalization* was introduced by Codd[102]. It is a systematic process to ensure that a database structure is will not be subject to anomalies after insertion, update, and deletion, that could lead to a loss of data integrity[140]. Data normalization is also useful to reduce the need for restructuring the collection of relations as new types of data are introduced. There are currently five normal forms*, numbered from one to five. The

*There exists a sixth normal form proposed by Date *et al.*[141] for temporal data higher the normal form, the more robust the database structure is against inconsistencies. PROFESS was designed using the fifth normal form proposed by Fagin[142]. The resulting Entity-Relationship diagram is shown in Figure 3.7.

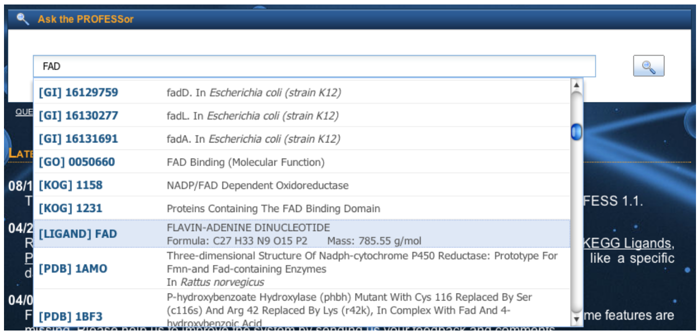
However, selective denormalization was subsequently performed for performance reasons[143]. In particular, the PROFESSor (see Section 3.6.1) queries data from a unique table (precalc_professor) that includes pre-computed joins between relations. To maintain the data consist, routines were implemented along with the wrappers to regenerate the table after data is inserted in PROFESS.

3.6 Functional-Style Query System

3.6.1 The PROFESSor

The primary search tool to query the database is the *PROFESSor* (Figure 3.6.1), a unified text field that will assist the user to easily refine complex queries by dynamically suggesting entries from any integrated database.

The PROFESSor assists the user by correcting for spelling errors using DamerauLevenshtein metrics[144]. The Damerau-Levenshtein distance between two sequences of characters is the minimum number of basic operations (substitution, insertion, deletion and transposition) necessary to transform one sequence into the other one. For example, the Damerau-



[GI] 16129759	fadD. In *Escherichia coli (strain K12)*
[GI] 16130277	fadL. In *Escherichia coli (strain K12)*
[GI] 16131691	fadA. In *Escherichia coli (strain K12)*
[GO] 0050660	FAD Binding (Molecular Function)
[KOG] 1158	NADP/FAD Dependent Oxidoreductase
[KOG] 1231	Proteins Containing The FAD Binding Domain
[LIGAND] FAD	FLAVIN-ADENINE DINUCLEOTIDE Formula: C27 H33 N9 O15 P2 Mass: 785.55 g/mol
[PDB] 1AMO	Three-dimensional Structure Of Nadph-cytochrome P450 Reductase: Prototype For Fmn-and Fad-containing Enzymes In *Rattus norvegicus*
[PDB] 1BF3	P-hydroxybenzoate Hydroxylase (phbh) Mutant With Cys 116 Replaced By Ser (c116s) And Arg 42 Replaced By Lys (r42k), In Complex With Fad And 4-hydroxybenzoic Acid

Figure 3.9: The PROFESSor is a search tool generated from the core databases. It displays interactive suggestions to help the user to refine complex query. Using the PROFESSor, users can quickly and accurately find all functional, structure and sequence information about a particular protein and its relation to other protein functions and folds.

Levenshtein distance between "homo sapiens" and "homu spaiens" is 2: *u* is changed to *o*, and *p* and *a* need to be switched. This is an effective means to detect spelling errors as Damerau[145] showed that 80% of misspellings can be corrected with one basic operation only, which corresponds to a distance of one.

It also provides a user defined focused browsing feature. For instance, upon typing in the query FAD (Flavin-Adenine Dinucleotide) the PROFESSor returns a drop down list of protein folds and functions that have known relation with the FAD ligand (Figure 3). If a user selects the ligand suggestion, PROFESS will return all functional clusters known to interact with FAD. The PROFESSor searches all other data sources within PROFESS in the same manner. A user can rapidly identify other protein functions that have the same fold, bind similar ligands, or identify cellular localizations all in one search. In addition to the PROFESSor, a traditional search form is also available.

The PROFESSor may also be queried using many keywords from several databases using boolean logic. Using regular expressions, the general syntax for queries is defined as:

$$\texttt{([KEY]}^{0,1}\ \texttt{\textbackslash w}^{*}\ \texttt{([OR] \textbackslash w}^{*}\texttt{)}^{*}\texttt{)([OR]}^{0,1}\texttt{[KEY]}^{0,1}\ \texttt{\textbackslash w ([OR] \textbackslash w)}^{*}\texttt{)}^{*}$$

KEY depends on the database and may be one of the following (note that this list will grow with the number of core databases): ALL, COG, EC, GO, LIGAND, PDB, PFAM. By default, all keywords after a `[KEY]` are considered as a unique string for the query. This behavior can be altered by prefixing the keywords with `[OR]`. The wildcard characters % (any number n of characters, with $n \geq 0$) and _ (exactly one character) may be used in a query. A logical `AND` is performed between different keys.

3.6.2 Functional-Style Query System

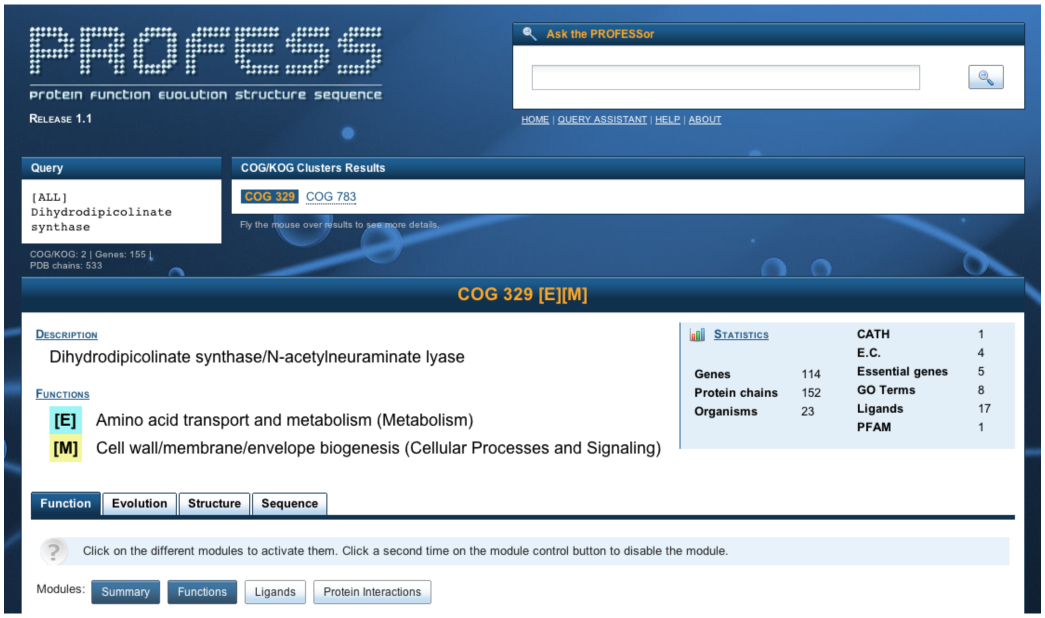
A fundamental component of our approach required the development of an intuitive functional-style query system that incorporated a variety of similarity functions[53] capable of generating data relationships not conceived during the creation of the database. For example, one may query for a relationship between the PFAM and the COG databases, although this relation is not explicitly defined in the database. Functional-style queries are composed of a set of atomic functions provided to the users. Each function takes as input a set of parameters and gives as output a well-defined value or set of values. A full query is defined as a pipeline of any of the atomic functions, where the output of a function serves as input of the next function in the pipeline. This is particularly useful to build incremental filters.

In addition to the possibilities offered by the PROFESSor, the user has the possibility to use BLAST to query our database using a protein sequence. In that case, the BLAST result is parsed and reformatted, and the function outputs the list of COG clusters that contain proteins homologous to the input sequence.

3.7 Web User Interface

The primary means for users to interact with PROFESS is through webpages over the Internet. PROFESS was built a flexible and extendable framework for genomics and proteomics data. Hence, in addition to an adequate underlying database structure (Section 3.4), the system must offer users a flexible interface that goes beyond simple static web pages.

Instead, the client-side software was designed to be interactive and user friendly. To achieve our goal, we utilized *Asynchronous Javascript and XML* (AJAX) web development techniques, which allows web applications to download data from the server asynchronously



[ALL] Dihydrodipicolinate synthase

COG/KOG: 2 | Genes: 155 | PDB chains: 533

Fly the mouse over results to see more details.

COG 329 [E][M]

Description
Dihydrodipicolinate synthase/N-acetylneuraminate lyase

Functions
[E] Amino acid transport and metabolism (Metabolism)
[M] Cell wall/membrane/envelope biogenesis (Cellular Processes and Signaling)

Statistics

		CATH	1
		E.C.	4
Genes	114	Essential genes	5
Protein chains	152	GO Terms	8
Organisms	23	Ligands	17
		PFAM	1

Click on the different modules to activate them. Click a second time on the module control button to disable the module.

Figure 3.10: Overview of the user interface highlighting the four levels to study proteins.

using the *XMLHttpRequest object* (XHR) in the background without interfering with the display and general behavior of the existing page. XHR objects enable web pages to effectively download data on demand from a server and return the data directly to the program for additional processing. The server's response is usually return in XML (Extensible Markup Language) or in JSON (Javascript Object Notation) format. Data in XML format have the advantage of being self-descriptive, meaning that the structure of the data is contained within the data, which facilitates interoperability. However, the self-descriptivity property of XML comes at the cost of a significant data overhead compared to the same data encoded in JSON format. Hence, because of the large amounts of data, JSON was favored to reduce latency and keep the system reactive.

XHR requests are the basis of the interactivity of our web interface. The PROFESSor (see Section 3.6) is the component that benefits the most from this technique: each time a user types a character in the text field, a XHR request is sent to the server. The server then queries the database and sends the response back to the client in JSON format. Finally, the response is processed and formated by the client to populate the drop-down menu which displays seemingly "live" suggestions to the user based on his input. The suggestions can help the user to refine his queries and detect typos more easily.

Because of the large amounts of data, the interface could quickly become confusing, hence diminishing the benefits of PROFESS. To keep the interface simple and user-friendly, each PROFESS entry has four tabs containing protein information about: Function, Evolution, Structure and Sequence (Figure 3.10). Each level of the PROFESS database mines pieces of information from all the integrated databases and provide the user with comprehensive tables highlighting annotations. The tables are defined as independent modules, each providing a unique representation of the integrated data. Table 3.2 gives an overview of available modules and the data sources that were necessary for their development. Each module can be activated or deactivated, depending on the specific needs of the user. For example, on Figure 3.10, the modules *Summary* and *Functions* are activated whereas *Ligands* and *Protein Interactions* are not. The interface also summarized the content of the current COG cluster using statistics.

The front-end user interface was implemented in dynamic HTML (Hyper Text Markup Language) and in Javascript. We also used the general AJAX frameworks developed by Yahoo! (YUI v2.7)[*] and ExtJS (v3.0)[†]. The server back-end program of PROFESS was implemented in PHP v5.2[‡] and relies on the free MySQL database management system[§]. To keep the system as open as possible, wrappers were implemented in Java 1.6 and thus platform independent. PROFESS is running under Open SuSE Linux 11.0 on the SunFire x4600 server of the Computer Science and Engineering department of the University of Nebraska, which features 8 AMD quad-core processors (32 cores) and 64 GB of memory.

[*] Available at `http://developer.yahoo.com/yui/`
[†] Available at `http://extjs.com`
[‡] Available at `http://www.php.net/`
[§] Available at `http://www.mysql.com/`

Chapter 4

Structural Comparison of Functional Orthologs

"*We are finding that things that once appeared to be biologically independent are closely connected.*"

Peter K. Sorger

4.1 Introduction

Quantifiable models of protein evolution are useful for developing robust tools to identify suitable drug-binding sites, to predict increases in susceptibility to a human genetic disease, and to predict and modify organismal niches. Some of the strongest arguments in favor of biological evolution draw from studies on protein evolution using sequence homology[146]. Multiple sequence alignments are routinely used to create phylogenetic relationships[147,148], which highlights sequence variability between organisms. The accepted view of protein evolution is that changes to the proteins gene sequence are selected and modulated by a number of factors that includes protein structures[149,150].

What is the impact on protein structure as its sequence undergoes genetic drift? Maintaining the correct protein fold is fundamental to preserving its function[151], but evolving the sequence would also be expected to result in structural changes[152,153]. The resulting paradox is that sequence determines a proteins structure, but the structure is relatively invariant over a large range of sequences. This paradox is highlighted by the tremendous difference between the number of known protein structures versus protein folds[154]. Even though the Protein Data Bank contains 61,086 protein structures[*], there are only 1,110 unique topologies[†] and 1,195 unique folds[‡] in the CATH and SCOP structure classification databases, respectively. The significant reduction in the number of protein folds relative to the number of protein sequences implies a much stronger correlation between structure and function. Protein structures are generally viewed as more conserved relative to its sequence. Evolution rates of protein sequences and structures were quantified by Illergård *et al.*[155].

An intuitive explanation is that substitutions of residues with similar chemical properties are relatively frequent, but that many mutations are silent, that is, they do not lead to any significant structural modifications. However, the explicit reason for the reduction in fold space remains unclear although some have suggested that the protein fold space is more likely to be represented as a continuum instead of a collection of discreet folds[156]. Using a continuous representation the fold space, a protein fold should be considered as being plastic, where sequence changes are accommodated by local perturbations in the structure while maintaining the general characteristics of a particular fold[155,157,158]. Correspondingly, the genetic drift in a proteins sequence may imply a similar gradual divergence in structure instead of a sudden dramatic transition to a new fold. If this perspective is accurate, then a comparative analysis of homologous proteins should identify correlated rates of structure and sequence divergence. Previous studies have looked at homologous structure similarity before but the datasets did not try to show phylogenetic consequences of structure divergence[155,157,158]. To help understand how protein plasticity affects organism divergence we compared 48 sets of homologous protein families annotated in the COG database for two bacterial phyla, *Firmicutes* and *Proteobacteria*.

[*]As of October 29, 2009

[†]As of version 3.2.0 released in August 2008

[‡]As of version 1.75 released in February 2009

4.2 Functional Annotation of Protein Structures

Current functional annotation tools available in the PDB include the Gene Ontology[132] (GO) and Enzyme Classification[122] (EC). Unfortunately, due to potential for convergence of function, this functional information is not useful for the study of homologous structures. Among the 20 resources for structural classification of proteins, the clusters of orthologous groups[129] (COGs) scheme is the only one that attempts to identify orthology. Therefore, each sequence and structure in the PDB was annotated with one COG number. This was achieved by developing the PROFESS (PROtein Function, Evolution Sequence and Structure) database which is described in details in Chapter 3.

The two best-represented bacterial phyla, which account for nearly one-fourth of all structures in the PDB, were selected for annotation. The PDB contains 8,298 *Proteobacteria* protein structures and 3,416 *Firmicutes* structures. PROFESS contains the PDB to COG annotations among other biological relevant information. This includes associating each structure with its taxonomic classification, which allowed for the structures from *Firmicutes* and *Proteobacteria* to be easily selected. To carry out our comparative study, we implemented a script to extract only those COGs that contained a minimum of two *Firmicutes* organisms and two *Proteobacteria* organisms. This requirement gave 281 unique COGs with a total of 3,047 bacterial proteins, including 1,066 *Firmicutes* and 1,981 *Proteobacteria*.

4.3 Pairwise Structure Similarity

4.3.1 Methods

The pairwise structure comparison tool DaliLite[159][*] was used to perform 63,504 pairwise comparisons between all of the proteins in our dataset. In total, the backbone structure similarity corresponded to 31,542 Proteobacteria-Proteobacteria comparisons (-/-), 12,674 Firmicutes-Firmicutes comparisons (+/+), and 19,288 Proteobacteria-Firmicutes comparisons (-/+). DaliLite was installed on Powers lab[†] Beowulf cluster, which runs CentrOS 4.4 and features 32 cores (16 dual-core AMD Athlon @2.13 GHz), 1 GB of RAM and 2.25TB for storage.

[*]Available at `http://www.ebi.ac.uk/Tools/dalilite/`

[†]Powers lab website: `http://bionmr-c1.unl.edu/`

Table 4.1: The 48 COG structure families with two *Firmicutes* and two *Proteobacteria* organisms after manual curation. "Split" means the *Firmicutes* and *Proteobacteria* proteins were strongly separated, "Starburst" means there was no evidence for a split according to phyla, and "Split +1" means there was strong evidence for a split with the exception of one protein.

COG	Function	Tree	CATH
28	Thiamine pyrophosphate requiring enzymes	Split	3.40.50.970
39	Malate/lactate dehydrogenases	Split	3.40.50.720
394	Protein-tyrosine-phosphatase	Split	3.40.50.270
446	Uncharacterized NAD (FAD)-dependent dehydrogenases	Split	3.30.390.30
604	NADPH:quinone reductase and related Zn-dependent oxidoreductases	Split	3.40.50.720
605	Superoxide dismutase	Split	3.20.20.80
742	N6-adenine-specific methylase	Split	3.40.50.150
813	Purine-nucleoside phosphorylase	Split	3.40.50.1580
1012	NAD-dependent aldehyde dehydrogenases	Split	3.40.309.10
1057	Nicotinic acid mononucleotide adenylyltransferase	Split	3.40.50.620
1075	Predicted acetyltransferases and hydrolases with the alpha/beta hydrolase fold	Split	3.40.50.1820
1607	Acyl-CoA hydrolase	Split	3.40.0.1820
1940	Transcriptional regulator/sugar kinase	Split	3.30.420.40
2124	Cytochrome P450	Split	1.10.630.10
2188	Transcriptional regulators	Split	3.40.1410.10
242	N-formylmethionyl-tRNA deformylase	Split +1	3.90.45.10
1052	Lactate dehydrogenase and related dehydrogenases	Split +1	3.40.50.720
2141	Coenzyme F420-dependent N5,N10-methylene tetrahydromethanopterin reductase and related flavin-dependent oxidoreductases	Split +1	3.20.20.30
3832	Uncharacterized conserved protein	Split +1	3.30.530.20
110	Acetyltransferase (isoleucine patch superfamily)	Starburst	2.160.10.10
171	NAD synthase	Starburst	3.40.50.620
251	Putative translation initiation inhibitor, yjgF family	Starburst	3.30.1330.40
346	Lactoylglutathione lyase and related lyases	Starburst	3.10.180.10

Continued on next page...

Table 4.1 (continued): The 48 COG structure Families with two *Firmicutes* and two *Proteobacteria* organisms after manual curation

COG	Function	Tree	CATH
366	Glycosidases	Starburst	2.60.40.1180
454	Histone acetyltransferase HPA2 and related acetyltransferases	Starburst	3.40.630.30
491	Zn-dependent hydrolases, including glyoxylases	Starburst	3.60.15.10
500	SAM-dependent methyltransferases	Starburst	3.40.50.150
526	Thiol-disulfide isomerase and thioredoxins	Starburst	3.40.30.10
590	Cytosine/adenosine deaminases	Starburst	3.40.140.10
637	Predicted phosphatase/phosphohexomutase	Starburst	1.10.164.10
664	cAMP-binding proteins	Starburst	1.10.10.10
745	Response regulators consisting of a CheY-like receiver domain and a winged-helix DNA-binding domain	Starburst	3.40.50.2300
753	Catalase	Starburst	3.30.63.10
778	Nitroreductase	Starburst	3.40.109.10
784	FOG: CheY-like receiver	Starburst	3.40.50.2300
796	Glutamate racemase	Starburst	3.40.50.1860
1028	Dehydrogenases with different specificities	Starburst	3.40.50.720
1151	6Fe-6S prismane cluster-containing protein	Starburst	1.20.1270.30
1309	Transcriptional regulator	Starburst	1.10.357.10
1396	Predicted transcriptional regulators	Starburst	1.10.260.40
1404	Subtilisin-like serine proteases	Starburst	3.40.50.200
1733	Predicted transcriptional regulators	Starburst	1.10.510.10
1846	Transcriptional regulators	Starburst	1.10.10.10
2159	Predicted metal-dependent hydrolase of the TIM-barrel fold	Starburst	3.20.20.140
2367	Beta-lactamase class A	Starburst	3.40.710.10
2730	Endoglucanase	Starburst	3.20.20.80
3693	Beta-1,4-xylanase	Starburst	3.20.20.80
4948	L-alanine-DL-glutamate epimerase and related enzymes of enolase superfamily	Starburst	3.20.20.120

We designed a C-shell script that matches the PDB files from each *Proteobacteria-Proteobacteria* comparison (-/-), *Firmicutes-Firmicutes* comparison (+/+) and *Proteobacteria-Firmicutes* comparison (-/+) and then submits the job to the program DaliLite. Each structural comparison takes approximately 2-10 min, depending on the relative similarity of structures. The total time to run all 63,504 comparisons was approximately seven weeks.

The shell script then extracted all structural comparison information reported by DaliLite (comparison files, rmsd, sequence identity, Z-score) on a per chain basis. A single PDB file may contain multiple protein chains, where each chain may have a separate COG assignment. These data represent the core of the *Structure Comparisons* module of our PROFESS database. The data were then parsed to find the largest Z-score for each pairwise structure comparison. The largest Z-score represents the best structure comparison for a pair of proteins and was used to calculate the Fractional Structure Similarity score (FSS) defined in Equation 3.1.

All comparisons were manually filtered within their respective COG to remove all but one redundantly solved structure, multiple or non-functionally relevant conformations (mutant protein, non-native experimental conditions, inhibited ligand complex), and the shorter of two protein structures. The final dataset contained 48 COGs (Table 4.1) with a total of 1,713 structural comparisons among 147 *Firmicutes* proteins from 58 unique organisms and 176 *Proteobacteria* proteins from 84 unique organisms.

4.3.2 Results

The resulting Dali Z-scores from the pairwise structure comparisons were plotted against sequence identity (Figure 4.1) to reveal a saturating relationship as the percent identity rose to 100%. The lowest observed Z-score was $Z = 5.7$ with a corresponding 16% sequence identity. This Z-score was well above the minimum cutoff of 2.0 (dashed line) for matches that were two standard deviations above a random match. This lowest Z-score came from the comparison of two Firmicutes proteins in COG0346 (lactoylglutathione lyase and related lyases): 2QH0 (Clostridium acetobutylicum); and 2QQZ (Bacillus anthracis). The average Z-score for all comparisons was $Z = 27 \pm 13$, indicating that all structural matches were very significant even at sequence identities below 20%.

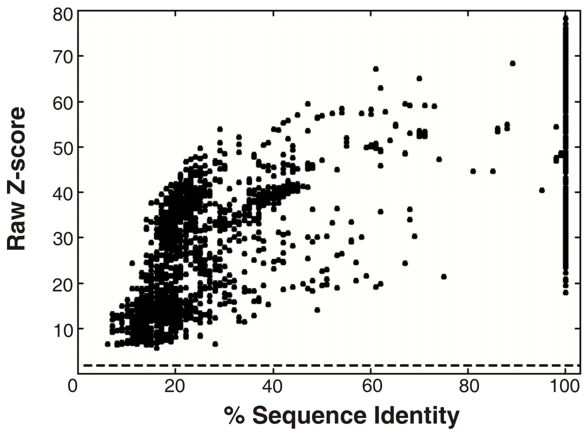


Figure 4.1: The relationship between structure similarity and sequence identity for 48 COGs. Structure similarity is given as the raw Z-score, which increases as the protein length increases. The comparisons were for all proteins against all proteins, and include those for each protein against itself. The dashed line identifies a Dali Z-score of 2, which is the minimal limit for inferring structural similarity.

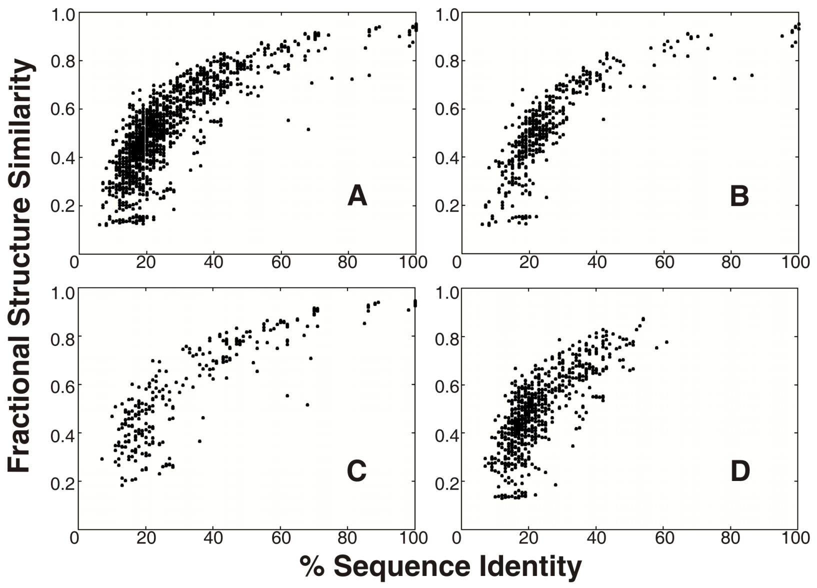


Figure 4.2: The fractional structure similarity (FSS) and sequence identity for 48 COGs. The FSS values were plotted against sequence identity for (A) all the pairwise comparisons, (B) only *Proteobacteria-Proteobacteria* comparisons, (C) only *Firmicutes-Firmicutes* comparisons and (D) only *Proteobacteria-Firmicutes* comparisons.

Since Z-scores increase as a function of the protein length, we calculated the Fractional Structure Similarity (FSS) score (see Section 4.3.1). When the pairwise FSS scores were plotted against sequence identity (Figure 4.2), a hyperbolic curve was obtained with all FSS values below an upper-limit at each percent identity. In fact, 20% sequence identity yielded a maximal FSS of 60%. This FSS limit was observed when all of the data were used (Figure 4.2A), when only the pairwise comparisons within either phyla were used (Figure 4.2B and C), or when only the pairwise comparisons between the two phyla were used (Figure 4.2D). The pairwise comparison plot between the two phyla (Figure 4.2D) showed an abrupt cutoff at 61% sequence identity and a 0.84 FSS score.

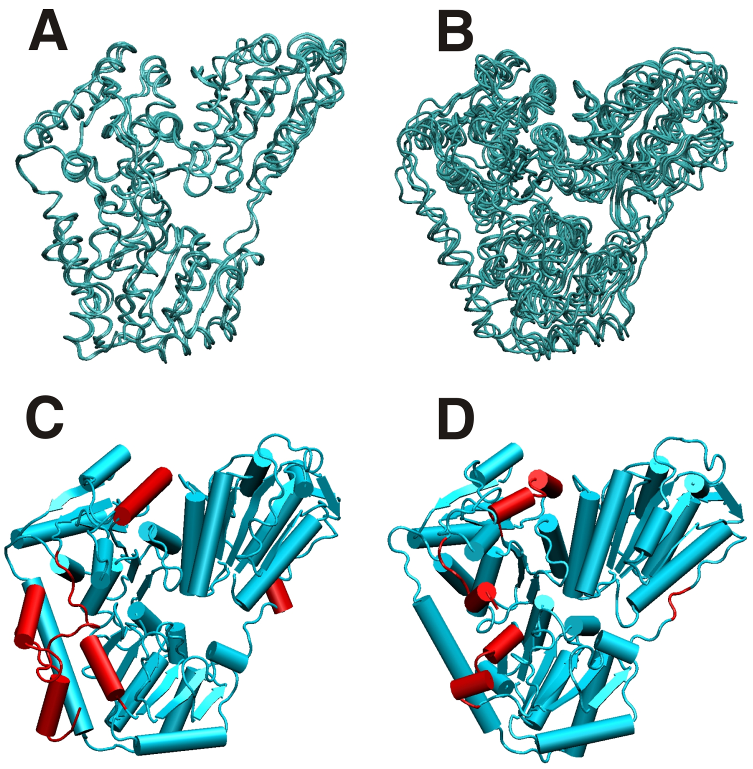


Figure 4.3: The protein structures for COG28 (thiamine pyrophosphate requiring enzymes) show (A) that the two *Firmicutes* structures have highly overlapping structures and (B) that the four *Proteobacteria* structures are very similar to each another. The major structural differences between the *Firmicutes* and *Proteobacteria* are highlighted in red on a representative *Firmicutes* (C) structure from *L. plantarum* (PDB ID: 1POW) and a representative *Proteobacteria* structure (D) from *P. fluorescens* (PDB ID: 2AG0).

The protein structures in COG 28 (thiamine pyrophosphate requiring enzymes) provides a useful example of the structural divergence that occurred after the *Firmicutes* and *Proteobacteria* phyla split. The overall fold is conserved between the phyla while there are discrete localized structural elements that are unique to each phylum. The two *Firmicutes* structures (Figure 4.3A) yield a Z-score of 59.6 and an FSS of 0.83, indicating very high structural conservation. The four *Proteobacteria* structures (Figure 4.3B) yield an average Z-score of 37.7 ± 1.6 and an average FSS of 0.58 ± 0.03. Again, the structures share a similar fold despite the slightly lower scores.

The major structural differences between the *Firmicutes* and *Proteobacteria* are highlighted in red on a representative *Firmicutes* (Figure 4.3C) structure from *L. plantarum* (PDB ID: 1POW) and a representative *Proteobacteria* structure (Figure 4.3D) from *P. fluorescens* (PDB ID: 2AG0). The comparison of protein structures between phyla yields an average Z-score of 34.8 ± 1.2 and an average FSS of 0.49 ± 0.02, which is significantly lower than the comparisons within each phylum. This suggests a divergence in structural details while conserving the overall fold. A detailed analysis reveals localized differences between the structures from the two phyla. In the *Firmicutes* representative structure, there is a continuous helix compared to helical breaks and loop insertions in the *Proteobacteria* structure. This is similar to the C-terminal domain of primase, where a long continuous helix found in the *E. coli* structure is broken by a loop region in *B. stearothermophilus*[160,161,162,163].

4.4 Phylogenetic Analysis of Functional Orthologs

4.4.1 Methods

Additionally to pairwise structural alignment, all the protein structures from each COG were simultaneously aligned using the multiple structure alignment program MAMMOTH-multi[*][82]. We implemented a script utilizing the resulting aligned structures and the structure-based sequence alignment to calculate an all-versus-all matrix of per-residue α-Carbon (C_α) distances. Standard bootstrapping techniques[164] were then applied to the all-versus-all matrix of per-residue C_α distances to generate 100 distance-matrices. Columns of structure-based sequence alignments with the corresponding C_α distances were randomly selected

[*]Software available at `http://ub.cbm.uam.es/mammoth/mult/`

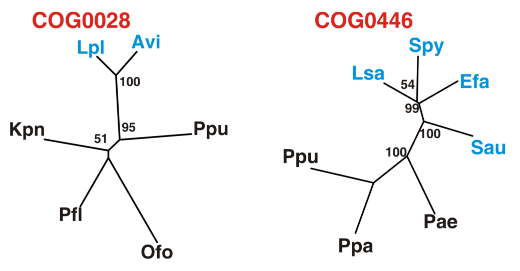
until the total number of columns in the original sequence alignment was reached. The resulting set of C_α distances were then used to calculate a root mean square deviation (rmsd) between each pair of structures in the matrix. The 100 distance-matrices were imported into PHYLIP v3.68[138] to generate a consensus phylogenetic tree and bootstrap confidence levels.

Each set of 100 bootstrapped distance-matrices were analyzed by the Fitch-Margoliash method[165] implemented in PHYLIP. Each matrix was jumbled with 100 replicates. This resulted in 10000 ($= 100 * 100$) unique and random distance matrices for each COG. The best tree was identified with the program *Consense* implemented in PHYLIP using the extended majority rule conservation. Since the bootstrapped trees do not show distance relationship, the original distance matrix generated by MAMMOTH-multi was used to generate a distance based phylogenetic tree. Each original distance matrix was jumbled with 100 replicates. The distance based phylogenetic tree was drawn using the program *Drawtree* implemented in PHYLIP. Each tree was visually inspected and compared with the DaliLite analysis using the bootstrap values to determine if a tree fit the star, split or undetermined classification.

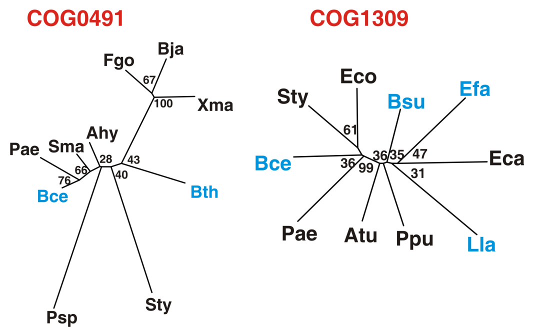
4.4.2 Results

Structure based phylogenies were created from root-mean square differences (rmsd) in per residue C_α positions for optimally aligned protein structures using MAMMOTH-multi (see Section 2.2.2). A separate phylogenetic tree was generated for each COG, where three distinct patterns were observed as reported in Table 4.1. Fifteen trees exhibited a strong split at the phylum level, 29 exhibited a starburst pattern suggesting little to no evidence for a split according to phyla, and 4 exhibited a strong split at the phylum level but with the exception of a single structure (split +1).

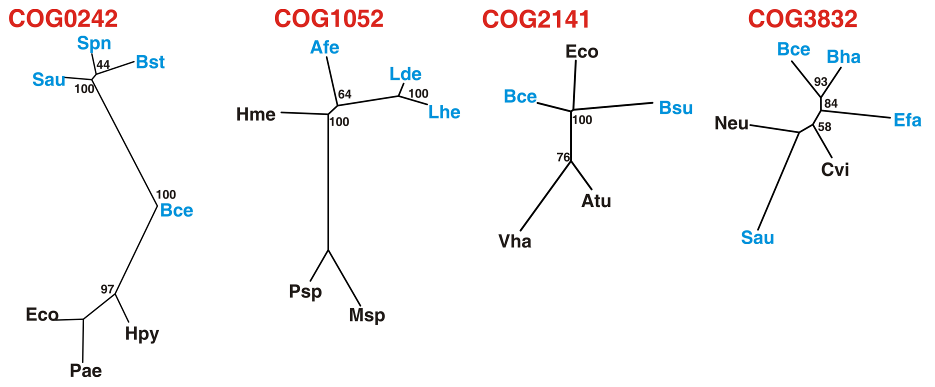
The fifteen COG phylogenies with strong phylum-splitting patterns had two branches, one with closely related *Firmicutes* structures and the other with closely related *Proteobacteria* structures. Figure 4.4a shows COG28 (Thiamine pyrophosphate requiring enzymes) and COG446 (Uncharacterized NAD(FAD)-dependent dehydrogenases) as examples. The



(a) Split pattern, illustrated by COG28 and COG446



(b) Starburst pattern, illustrated by COG491 and COG1309



(c) Split+1 pattern, illustrated by COG242, COG1052, COG2141 and COG3832

Figure 4.4: Protein structure based phylogenetic trees highlighting the *split*, *split+1* and *starburst* patterns.

structures for both of these COGs are classified in the CATH system as α/β 3-layer sandwiches, but differ in that COG28 proteins have a Rossmann fold topology and COG446 proteins have a FAD/NAD (P)-binding domain topology.

The twenty-nine COGs with phylogenetic starburst patterns (listed in Table 4.1) showed no evidence for the separation of structures according to phyla. Figure 4.4b shows COG491 (Zn-dependent hydrolases) and COG1309 (Transcriptional regulator) as examples. The CATH classification for COG491 *Bacillus cereus* Zinc-dependent beta-lactamase (PDB ID: 1BC2) describes it as an α/β 4-layer sandwich with metallo-beta-lactamase Chain A topology. Beta-lactamases constitutes a large collection of enzymes that can be derived from any one of a group of proteins that bind, synthesize, or degrade peptidoglycans. The protein structures assigned to COG491 gave FSS scores with large standard deviations, consistent with the separated clusters within the *Proteobacteria* arm of the phylogenetic tree. The COG1309 structural family falls into one of two CATH topologies, Arc Repressor Mutant (subunit A) or Tetracycline Repressor (domain 2). Only those structures similar to the Arc Repressor Mutant topology were used for the pairwise comparison, since it was the dominant fold in this COG. The protein structures in the COG1309 structure family gave low FSS scores. However, even with low overall FSS the average absolute Z-score was 13 ± 2 indicating that it has significant overall structure similarity. The high FSS deviations of COG491 structural family and the low average FSS scores of COG1309 structural family both indicate rapid structural divergence following the phyla split, consistent with the observed starburst phylogenetic patterns.

Four COG structure phylogenetic trees (listed in Table 4.1) showed a strong split pattern with a single outlier (Figure 4.4c). This result provides further evidence for the observation of phyla split based on structure similarity. The presence of the outlier in a clear split pattern suggests a horizontally transferred gene or potential paralog. For all four families, there was a large and significant average absolute Z-score for all comparisons along with strong BLAST E-values indicating that the correct match was made between COG and PDB. For COG242, the *Bacillus cereus* gene *def* that encodes the N-formylmethionyl-tRNA deformylase protein (PDB ID: 1WS0) has been previously identified by Garcia-Vallve *et al.*[166] as a gene that has undergone horizontal gene transfer.

4.5 Structure Divergence Rates across Phyla

To quantify the relationship between structure difference and sequence difference, each phylogenetic tree was assigned a simple score by calculating a *structure similarity ratio* θ_{FSS} and a *sequence identity ratio* θ_{SeqID}.

The structure similarity ratio θ_{FSS} (Equation4.1) was determined for all 48 COGs by calculating an average FSS score for the *Proteobacteria-Firmicutes* structure comparisons and dividing by the sum of the average *Proteobacteria-Proteobacteria* and *Firmicutes-Firmicutes* comparisons.

$$\theta_{FSS} = \frac{Avg(FSS_{+/-})}{\frac{Avg(FSS_{+/+})}{2} + \frac{Avg(FSS_{-/-})}{2}} \tag{4.1}$$

Similarly, the sequence identity ratio θ_{SeqID} (Equation4.2) was determined by calculating an average sequence identity for the *Proteobacteria-Firmicutes* structure comparisons, and dividing by the sum of the average *Proteobacteria-Proteobacteria* and *Firmicutes-Firmicutes* comparisons.

$$\theta_{SeqID} = \frac{Avg(SeqID_{+/-})}{\frac{Avg(SeqID_{+/+})}{2} + \frac{Avg(SeqID_{-/-})}{2}} \tag{4.2}$$

In general, most starburst phylogenetic trees (see Figure 4.4b) had a branch length between members of different phyla that was much shorter than the branch lengths between members within the same phyla. That is, a starburst phylogeny was expected to have θ_{FSS} and θ_{SeqID} values greater than one. Likewise, most split phylogenies had longer branches between phyla than within each phyla (see Figure 4.4a) and were expected to yield θ_{FSS} and θ_{SeqID} of less than one.

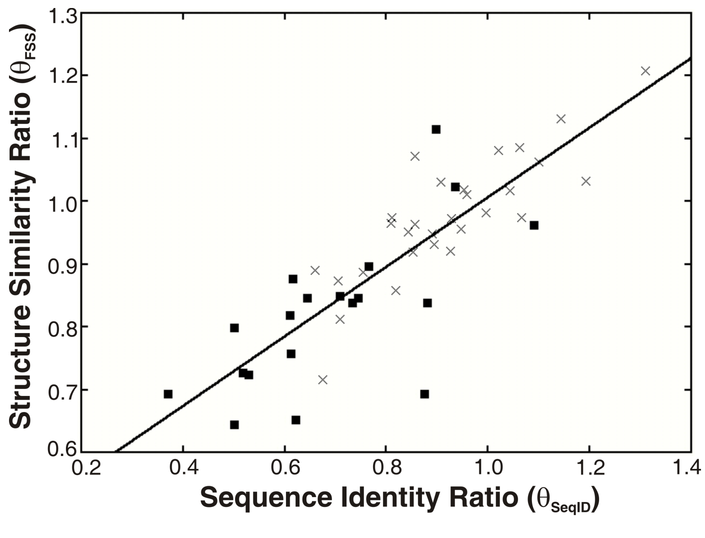


Figure 4.5: The relationship between structure and sequence change was constant regardless of the phylogenetic starburst ($\times$) or split (■) pattern. The best-fit line is defined by $\theta_{FSS} = 0.55\theta_{SeqID} + 0.45$ and yields a correlation factor $R^2 = 0.7$.

When θ_{FSS} and θ_{SeqID} for all 48 COGs were plotted versus one another (Figure 4.5), the starburst phylogenies clustered around unity for both structure and sequence whereas the split phylogenies clustered around 0.85 for structure and 0.70 for sequence. This indicated that split phylogenies occur when the structure differences are significantly less than their sequence differences. In addition, the plot of θ_{FSS} versus θ_{SeqID} conformed to a linear relationship regardless of the shape of the phylogenetic tree indicating that all homologous protein structure differences are constant with respect to homologous protein sequence differences ($\theta_{FSS} = 0.55\theta_{SeqID} + 0.45$; $R^2 = 0.7$). Thus, this curve represents the relative structural drift rate for each COG structural family between the two phyla. The slope indicates that structure branch lengths change approximately half as fast as sequence branch lengths.

4.6 Fold dependency on Structure Similarity

A plot of FSS vs. sequence identity for the two most populated CATH families in our dataset (Figure 4.6) was used to investigate if particular protein architectures are more amenable to structural changes. Half of our data set (24 out of 48 COGs), is represented by CATH 3.40 (α/β, 3-Layer ($\alpha\beta\alpha$) sandwich). Within CATH 3.40, 12 of 24 COGs (50%) are

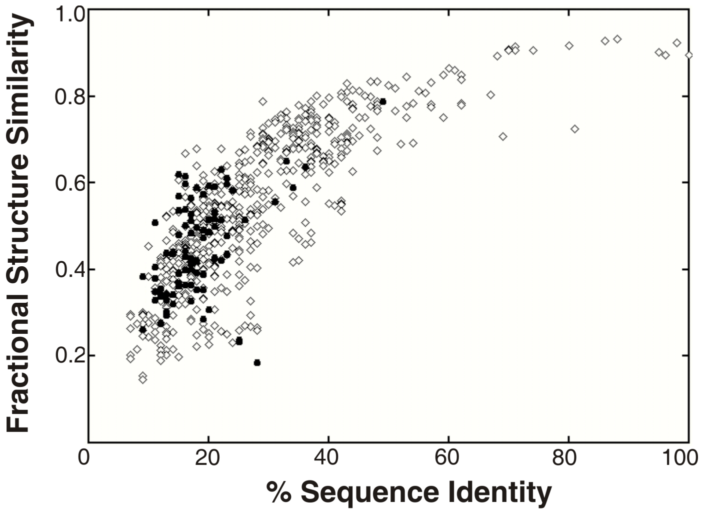


Figure 4.6: Fold dependency on fractional structure similarity (FSS) and sequence comparisons for CATH 1.10 (●) and CATH 3.40 (◇).

represented by the starburst phylogenetic tree pattern. The remaining 12 COGs correspond to 11 splits and 1 split +1 phylogenetic tree patterns.

The second most populous CATH family is CATH 1.10 (mainly α, orthogonal bundle) with 15% of our COGs belonging to this CATH family. Most (85.7%) of the COGs (6 of 7) in the CATH 1.10 family are represented by the starburst phylogenetic tree pattern with only one COG represented by a split pattern. There appears to be a limit in structure similarity at approximately 0.6 FSS and a corresponding sequence identity limit at 40% for CATH 1.10 (●). This limit is not observed in the CATH 3.40 family (⋄). The sequence and structure similarity limit for CATH 1.10 combined with a larger percentage of COGs assigned to the starburst family suggests that CATH 1.10 is more susceptible to mutations that affect the protein structure. The results suggest a faster evolutionary rate leading to a higher structural divergence relative to other CATH architectures.

4.7 Discussion

The comparison of homologous protein structures with the same function provides quantitative evidence that protein structures diverged following the speciation events that created the modern bacterial phyla of *Firmicutes* and *Proteobacteria*. The abrupt cutoff at 61% sequence identity and 0.84 fractional structure similarity observed between *Firmicutes* and *Proteobacteria* proteins was mirrored by an approximate 60% protein sequence identity between these two phyla observed by 16S rRNA sequence similarity[167,168]. Thus, this maximum observed sequence identity imparts limits to the maximum possible structure similarity between homologus proteins from these two phyla. This is consistent with prior observations that sequence identity 40-50% sometimes results in significant structural and functional differences[152,153,169]. Furthermore, the results imply an inherent allowable structural plasticity that does not perturb function. Additionally, the random drift after speciation inexorably leads to non-identical structures despite maintenance of function. There are a number of cases where FSS was below 0.20 indicating a significant structural change. Proteins with completely different folds but the same function are extreme examples of the plasticity of the structure-function relationship and include such proteins as peptidyl-tRNA hydrolases[170], pantothenate kinase[171], polypeptide release factors[172] and lysyl-tRNA synthetases[173].

Forty percent of the COGs we examined have evolved slowly enough that it was possible to generate phylogenetic trees consistent with this ancient split. The other COGs have either evolved too rapidly or are otherwise subject to few evolutionary constraints to provide evidence for this split. This distinction between the COGs is clearly apparent from the comparison of θ_{FSS} and θ_{SeqID} from Figure 4.5. The linear relationship implies a *fixed relative structure drift rate*, where structure changes half as fast as sequence across phyla. This correlation in the divergence of protein sequences and protein structures has additional ramifications beyond bacterial evolution. Our analysis implies a continuum of protein folds that adapt to large sequence changes by incurring local structural modifications[155,156,157,158]. This continuum of protein folds makes it challenging to apply protein structural classification to identify function, as has been previously noted[174,175].

Does the nature of the proteins three-dimensional structure play a role in protein structure divergence? Our analysis demonstrates that some proteins evolve slowly and maintain high sequence identity ($\geq$80%) and structure similarity ($\geq$0.80 FSS) while other proteins exhibit rapid evolution rates where the sequence identity is lower than 20% and the FSS below 0.40. This implies that the underlying architecture of a particular protein may be more or less amenable to amino-acid substitutions in order to maintain functional activity. A specific protein fold may have a higher intrinsic plasticity that enables it to readily accommodate sequence changes through local conformational changes without a detrimental impact on activity. This is exactly what was observed, structural variations were localized to specific regions as illustrated by the comparison of the COG 28 protein structures see (Figure 4.3). This is consistent with the observation that there are different structure divergence rates within a protein[176,177]. Regions of the protein that do not impact biological activity are expected to yield a higher divergence rate and incur larger local structural changes[152,178]. As a result, a fold with a relatively high plasticity would experience an elevated structural diversity between phyla, where the rate of change may closely parallel the mutation rate[155]. Conversely, another fold may be extremely sensitive to amino-acid substitutions, where minor sequence perturbations may result in a decrease in structural integrity and a corresponding loss of activity. As a result, the sequence and structure of this protein class would be relatively conserved. This analysis is consistent with the known range of protein thermodynamic stabilities[179], and the general observation that most mutations destabilize protein structures[180].

Chapter 5

Experimental Datasets

"It is a capital mistake to theorize before one has data."

Sir Arthur Conan Doyle

In order to experimentally test and evaluate the various classification methods proposed in Chapters 6, 7, 8 and 9, we used five different data sets, namely CRCARS (Section 5.1), FLU (Section 5.2), HDD (Section 5.3), PBC (Section 5.4), TCEQ (Section 5.5). Table 5.1 summarizes the utilization of the datasets during the experimental evalution of the proposed methods. These datasets will also be used through the remaining of this dissertation to explain the classification methods and illustrate the successive steps of the algorithms.

During our experiments, we used five different data sets: CRCARS (Section 5.1), FLU (Section 5.2), HDD (Section 5.3), PBC (Section 5.4), TCEQ (Section 5.5). These data sets are based on real data (non-synthetical) and cover broad knowledge domains, in particular

Table 5.1: Usage of the datasets during the experimental evaluation of the proposed methods.

Method	CRCARS	FLU	HDD	PBC	TCEQ
Classification Integration			■		
Reclassification	■			■	
Temporal Classification		■			■

Table 5.2: Main characteristics of the CRCARS, FLU, HDD, PBC and TCEQ data sets used during our experiments. The data sets are based on real, non-synthetical, data.

Dataset	Domain	Records*	Features*	Classes*	Temporal	Ref.
CRCars	Automobile	406	5	3/3	No	181
Google Flu Trends	Medical	15,352	4	2	Yes	182
Heart Disease Diagnostic	Medical	720	10	2	No	183
Primary Biliary Cirrhosis	Medical	314	17	3/2/4	No	184
TCEQ Ozone	Meteorological	2,534	40	2	Yes	185

* Datasets were remapped during the experiments. As a result, the number of actual records, features and classes used during the experiments may vary from the above numbers. See relevant sections for details.

the automobile (CRCARS), the meteorological (TCEQ) and the medical (HDD, FLU and PBC) fields. Unlike FLU and TCEQ, the CRCARS, PBC and HDD databases contain static data, that is, data that do no vary over time. It should be noted that the PBC, HDD and the TCEQ databases contain a relatively large number of features used by the classification algorithms. The main characteristics of the databases are summarized in Table 5.2.

5.1 CRCars Database

The *CRCars database* was used during the *Second Exposition of Statistical Graphics Technology* organized by the Committee on Statistical Graphics of the American Statistical Association in 1983. This data set was collected by Donoho *et al.*[181] in 1982. It comprises 406 observations on the following 7 measurements:

- **acceleration**: from 0 to 60 mph, measured in seconds (between 8 and 24.8 seconds)
- number of **cylinders** of the engine (between 3 and 8 cylinders),
- engine **displacement** in cubic inches (between 68 and 455 cubic inches),
- **horsepower**: power of the engine (between 46 and 230 horsepower),
- vehicle **weight**, measured in pounds (between 732 and 5140 lbs.),
- **mpg**: fuel efficiency (between 8.0 and 24.6 miles per gallon),
- **origin** of car (American, European or Japanese).

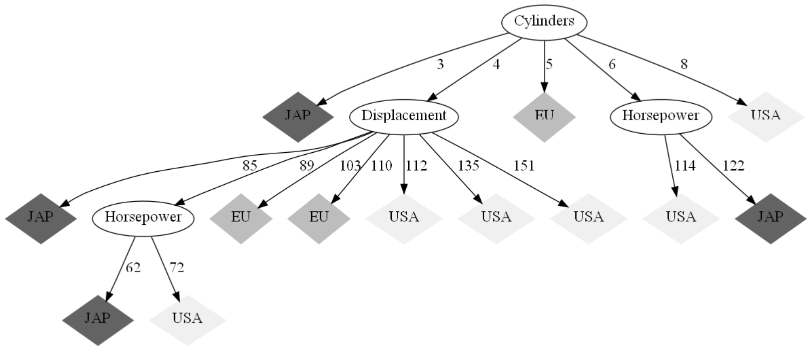


Figure 5.1: Decision Tree for the prediction of *ORIGIN* using the CRCARS dataset.

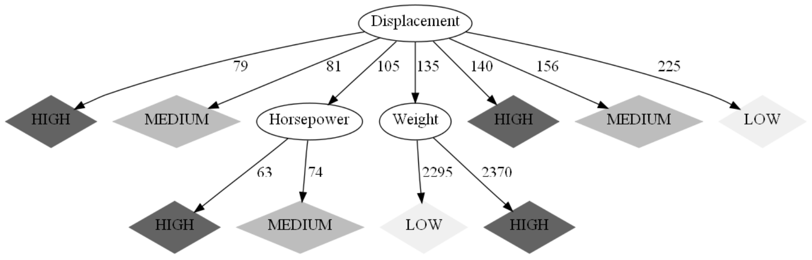


Figure 5.2: Decision Tree for the prediction of *EFFICIENCY* using the CRCARS dataset.

During our experiments, we used the first five measurements to define the feature space, and the last two variables as labels for the prediction. Note that in order to better accommodate the ID3 algorithm, we discretized the real-valued fuel efficiency in three classes (*low*, *medium*, *high*), and remapped the values so that each of them was adequately classified. Two example decision trees built using the ID3 algorithm for the prediction of the origin of cars and their fuel efficiency are shown in Figures 5.1 and 5.2 respectively.

5.2 Google Flu Trends Dataset

Seasonal influenza is a major public health concern, causing millions of respiratory illnesses and 250,000 to 500,000 deaths worldwide each year. If a new strain of influenza virus emerges, a pandemic could ensue with the potential to cause millions of deaths. The ongoing H1N1 ("swine flu") pandemic is responsible for the hospitalization of nearly 500,000 people worldwide, causing the death of over 5,000 patients*. The United-States alone count over

*Statistics from `http://www.flucount.org/` as of October, 12th 2009.

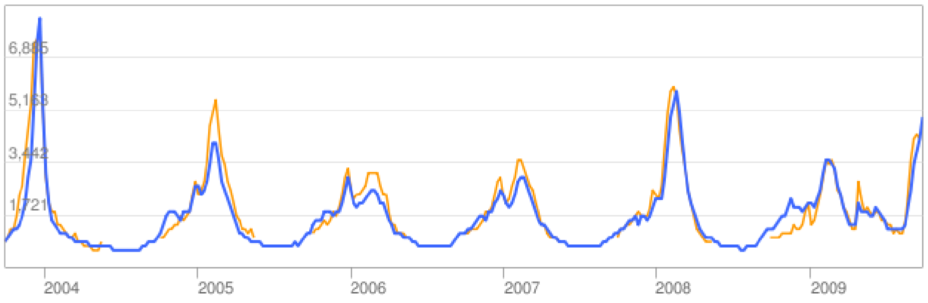


Figure 5.3: The comparison of the historical query-based flu estimates for the United States from Google Flu Trends (blue) against official influenza surveillance data from the Centers for Disease Control (orange) reveals a strong correlation, showing that query-based flu estimates can effectively measure current flu activity around the world in near real-time.

60,000 patients, including 413 cases in Nebraska, and the death of nearly 1,000 of patients*. These statistics highlight the need for early disease prediction, which can potentially greatly reduce the number of people affected by developing vaccines or some other medicine.

As one might expect, there are more flu-related searches during flu seasons. Ginsberg *et al.*[182] proved that certain search terms are good indicators of flu activity. In particular, they established a strong correlation between the number of *influenza-like illness* (ILI) searches and the data collected by the *Centers for Disease Control and Prevention* as shown on Figure 5.3†. Google Flu Trends is a novel service from Google that exploits this correlation and uses aggregated Google search data to estimate current flu activity around the world in near real-time.

The data set provides weakly ILI measurements since 2003 for each American states as well as national average measurements‡. The raw values provided by Flu Trends correspond to the number of ILI cases per 100,000 physician visits. The database describes the following two variables:

1. **t**, the date the measurement was recorded,

2. **ili**, the number of ILI cases per 100,000 physician visits.

*Statistics from the *Centers for Disease Control and Prevention* as of October, 16th 2009
†Data Source: Google Flu Trends (http://www.google.org/flutrends)
‡Since October 2009, data for 19 other – mostly European – countries were added to Flu Trends.

Because of its historical nature, this data set is particularly suitable for temporal classification. Hence we used it to evaluate the proposed *Temporal linear classification* discussed in Chapter 9.

5.3 Heart Disease Databases

The heart disease diagnosis data set[183] describes the medical records of patients regarding their heart condition and was collected by Robert Detrano, Andras Janosi and William Steinbrunn from the following three hospitals:

- Cleveland Clinic Foundation, USA (303 patients),
- Institute of Cardiology, Budapest, Hungary (294 patients),
- University Hospital of Zürich, Switzerland (123 patients).

The dataset contains no missing value. Each hospital records the following ten features:

1. **age**: years
2. **gender**: (0 = female, 1 = male)
3. **cp**: chest pain (1=typical angina, 2=atypical, 3=non-anginal, 4=asymptomatic)
4. **trestbps**: resting blood pressure (in mmHg on admission to the hospital)
5. **chol**: serum cholestoral in mg/dl
6. **restecg**: resting electrocardiographic results
7. **thalach**: maximum heart rate achieved
8. **exang**: exercise induced angina (0 = no, 1 = yes)
9. **oldpeak**: ST depression induced by exercise relative to rest
10. **disease**: presence of heart disease (-1 = not present, 1 = present)

We also simulated missing values by generating the three following subsets from the original data set by removing two features from each source:

- BUDAPEST with features (1, 2, 3, 4, 5, 7, 9, 10)
- CLEVELAND with features (1, 2, 3, 5, 7, 8, 9, 10)
- ZÜRICH with features (1, 2, 3, 5, 6, 7, 9, 10)

In both cases, the class to predict was the tenth measurement, that is, the presence of heart disease. This classification is binary in the sense that **disease** may take only two values: −1 is the patient is in good health, +1 otherwise.

5.4 Primary Biliary Cirrhosis Trial

A study of *Primary Biliary Cirrhosis* (PBC) of the liver conducted between 1974 and 1984 by the Mayo Clinic. A total of 312 PBC patients, who met eligibility criteria for the randomized placebo controlled trial of the drug D-penicillamine, participated in the trial. The database was adapted and formated by Fleming & Harrington[184] and contains the following twenty features:

1. **case** number,
2. **days** between registration and the earliest of death, transplantion, or analysis time,
3. **age** in days,
4. **sex** (0=male, 1=female),
5. **asictes** present (0=no or 1=yes),
6. **hepatomegaly** present (0=no or 1=yes),
7. **spiders** present (0=no or 1=yes),
8. **edema** (0 = none, 0.5 = edema resolved, 1 = edema with diuretics),
9. serum **bilirubin** in mg/dl,

10. serum **cholesterol** in mg/dl,

11. **albumin** in mg/dl,

12. **urine** copper in μg/day,

13. **alkaline** phosphatase in Units/liter,

14. **SGOT** in Units/ml,

15. **triglicerides** in mg/dl,

16. **platelets** per cubic ml/1000,

17. **prothrombin** time in seconds,

18. **status** (0=alive, 1=transplanted, or 2=dead),

19. **drug** (1=D-penicillamine or 2=placebo), and

20. histologic stage of **disease** (1, 2, 3, 4).

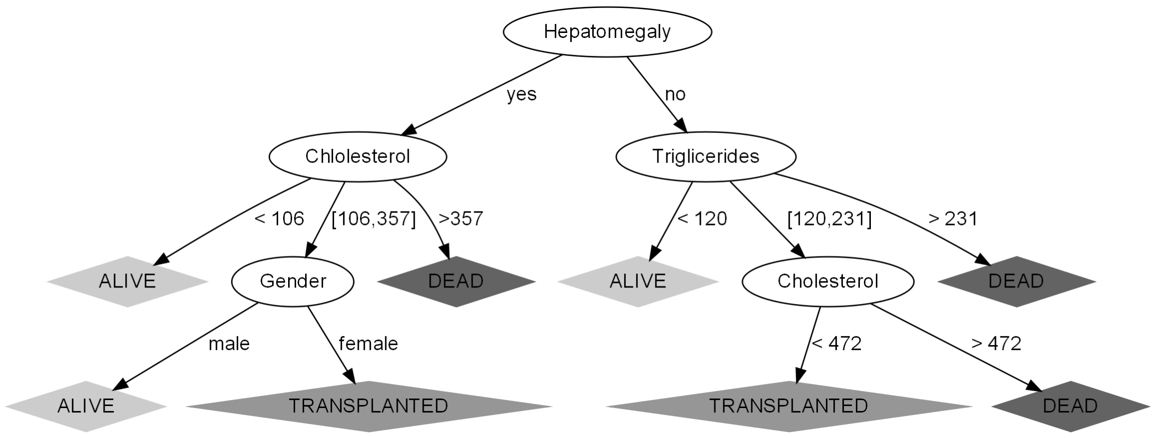


Figure 5.4: ID3-Interval Decision tree for the prediction of the status of a patient using *Primary Biliary Cirrhosis* data.

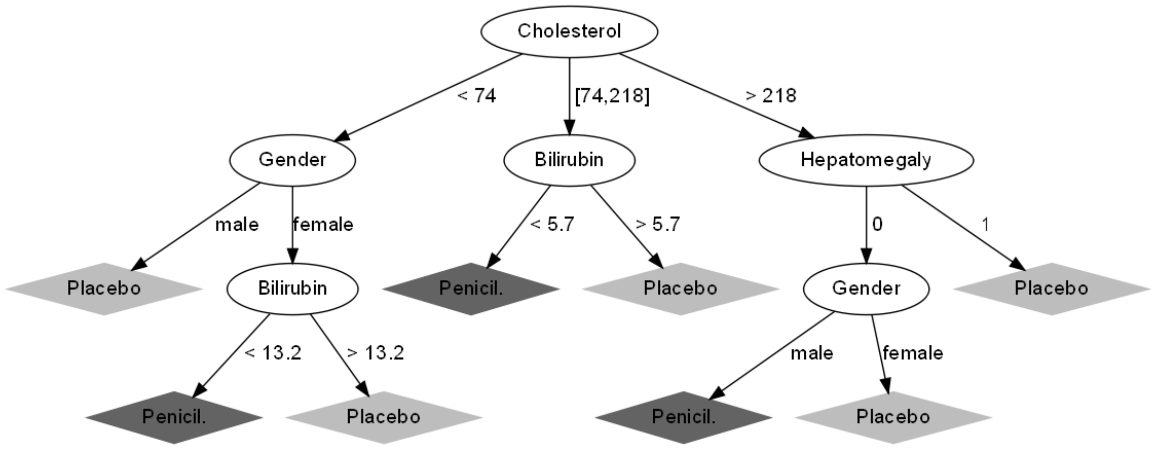


Figure 5.5: ID3-Interval Decision tree for the prediction of the drug efficiency using *Primary Biliary Cirrhosis* data.

We generated the following three subsets from the original data set:

- DISEASE with features (3, 4, 5, 7, 8, 9, 10, 13, 14, 16, 17, 20),
- DRUG with features (3, 4, 6, 7, 8, 9, 10, 11, 13, 16, 17, 19), and
- STATUS with features (3, 4, 5, 6, 7, 8, 9, 10, 11, 16, 17, 18).

In each subset, we used the first eleven features to predict the the twetfth, that is, the last feature.

5.5 Texas Commission on Environmental Quality Dataset

Weather forecasting is a challenging task. It is also natural to study because the major interest is in the prediction of the weather ahead of time instead of describing the current conditions. We tested our temporal classifier on a meteorological database of the Texas Comission on Environmental Quality.

The experiments used the Texas Commission on Environmental Quality (TCEQ) database[185], which recorded meteorological data between 1998 and 2004. From the TCEQ database, we used only the data for Houston, Texas*.

*Available from the UCI Machine Learning Repository[183].

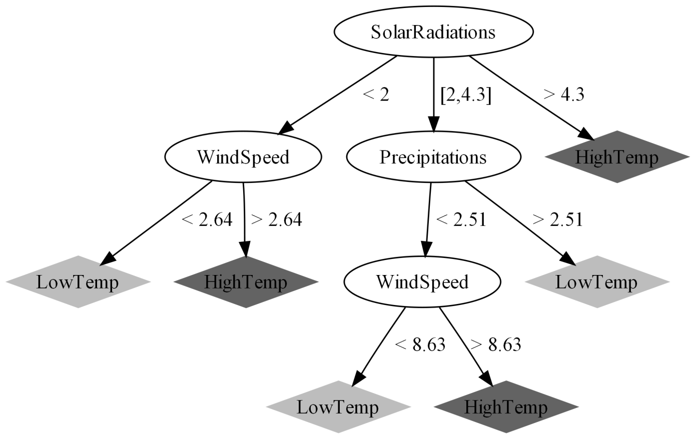


Figure 5.6: ID3-Interval decision tree for the prediction of the *temperature* using TCEQ.

The database describes the following fourty features and the class to predict:

1-24. **sr**: hourly solar radiation measurements

25. **asr**: average solar radiation

26. **ozone**: ozone pollution (0 = no, 1 = yes)

27. **tb**: base temperature where net ozone production begins

28-30. **dew**: dew point (at 850, 700 and 500 hPa)

31-33. **ht**: geopotential height (at 850, 700 and 500 hPa)

34-36. **wind-SN**: south-north wind speed component (at 850, 700 and 500 hPa)

37-39. **wind-EW**: east-west wind speed component (at 850, 700 and 500 hPa)

40. **precp:** precipitation

41. **T**: temperature class to predict

For **sr, dew, ht, wind-SN, wind-EW** we use a subscript to indicate the hour or the hPa level. Figure 5.6 shows a sample decision tree trained using the ID3-interval algorithm using six random records from the TCEQ database (see Table 6.1).

The data was normalized by making for each feature the lowest value to be -1 and the highest value to be $+1$ and proportionally mapped into the interval $[-1, +1]$ all the other values. This normalization is a precaution against any bias by the classifications. The normalization also allows a clearer comparison of the SVM weights of the features.

Chapter 6

Representation and Querying of Linear Classifiers

"*The goal is to transform data into information.*"

Carly Fiorina

6.1 Introduction

This chapter describes our novel representation of existing linear classifiers which is used as a first step in the proposed *classification integration*, *reclassification* and *temporal classification* methods presented in this dissertation. Although it can be extended to any linear classifier, we illustrate the main steps of our representation using SVMs in Section 6.2, ID3 decision trees in Section 6.3 and ID3-Interval trees in Section 6.4. In addition, the proposed constraint database representation is not limited to a specific database system and may be applied to *any* linear constraint database system. During our experiments, we used the IRIS Datalog Interpreter[108] and the MLPQ constraint database system[109]. We also describe a few typical queries that are useful for classifying new data. Queries are expressed in both Datalog and SQL languages.

6.2 Representation and Querying of SVMs

We proved in Section 2.3.2 that Support Vector Machines with linear kernels are linear classifiers, that is, they can be mathematically represented using Equation 2.4. Moreover, constraint databases, whose principles were described in Section 2.4, are particularly useful to handle and represent data that can be expressed in terms of linear constraints. Hence, we would like to take advantage of the advanced query capabilities of constraint databases to give an efficient and reusable representation of the knowledge within trained SVMs.

To illustrate the main steps of our method, we use the Texas Commission on Environmental Quality (TCEQ) database (see Section 5.5 for details) which contains daily meteorological measurements collected for over 7 years. For simplicity, consider the following smaller version *Texas_Weather* shown in Table 6.1, with only six consecutive days, where for each day D, the features are: Precipitation P, Solar Radiation R, and Wind Speed (north-south component) W, and the label is Temperature T, which is "High" or "Low".

Table 6.1: Sample data from the simplified *Texas_Weather* database.

D	P	R	W	T
1	1.73	2.47	-1.3	Low
2	0.95	3.13	9.32	High
3	3.57	3.56	4.29	Low
4	0.24	1.84	1.51	Low
5	0.0	1.19	3.77	High
6	0.31	4.72	-0.06	High

To classify the above data, we can use a SVM linear classifier. First, we need to assign a numerical value to symbolic features because SVMs are unable to handle non-numerical values. For instance, we assign the value $t = -1$ whenever $t =' low'$ and $t = +1$ whenever $t =' high'$. Then, we use the popular $LibSVM^{186}$ library to build a linear classification using a SVM. Like most other SVM packages, *LibSVM* does not output the equation of the maximum-margin separating hyperplane. Instead, it returns the coordinates of the support vectors in the n-dimensional feature space. Hence, we used Equation 2.11 to implement routines that calculate the equation of the separating hyperplane given the support vectors. That would result in a linear classifier, which can be represented by the linear constraint relation *Texas_SVM* shown in Table 6.2.

Table 6.2: Linear constraint representation *Texas_SVM* of the SVM trained using the *Texas_Weather* data.

P	R	W	T	
p	r	w	t	$-0.442838p + 0.476746r + 2.608779w - 0.355809 = t$

Given the $Texas\_Weather(d, p, r, w)$ and the $Texas\_SVM(p, r, w, t)$ relations, the following Datalog query computes for each day d the distance t to the hyperplane separating the two temperature classes:

```
Temp_SVM(d, t) :- Texas_Weather(d, p, r, w), Texas_SVM(p, r, w, t).
```

Finally, we can use the SVM relation to do the predictions, based on whether we are above or below the hyperplane.

```
Predict(d, y) :- Temp_SVM(d, t), 'high' = y, t >= 0.
Predict(d, y) :- Temp_SVM(d, t), 'low' = y, t < 0.
```

Instead of the above Datalog queries, one can use the logically equivalent SQL query:

```
CREATE VIEW Predict AS
  SELECT D.d, "High"
  FROM  Texas_Weather as D, Texas_SVM as T
  WHERE D.p = T.p  AND  D.r = T.r  AND  D.w = T.w  AND  T.t >= 0
UNION
  SELECT D.d, "Low"
  FROM  Texas_Weather as D, Texas_SVM as T
  WHERE D.p = T.p  AND  D.r = T.r  AND  D.w = T.w  AND  T.t < 0
```

6.3 Representation and Querying of ID3 Decision Trees

Figure 6.1 shows the ID3 decision tree for the *Texas_Weather_Data* in Section 6.2. Note that in this ID3 decision tree only the *Precipitation* feature is used. That is because the value of *Precipitation* is sufficient to classify the data for each day in the small database. For a larger database some precipitation values are repeated and other features need to be looked at to make a classification.

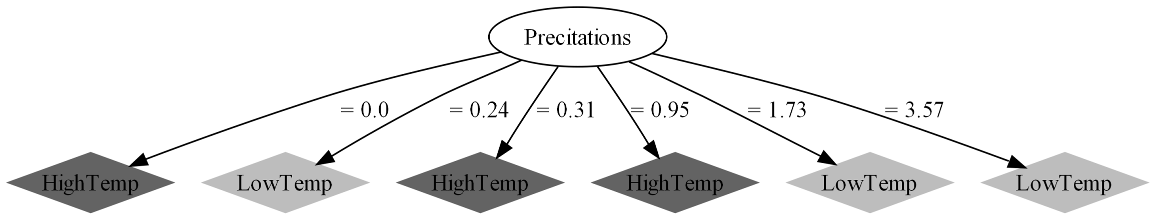


Figure 6.1: ID3 decision tree for the prediction of the temperature using the *weather* dataset.

A straightforward translation from the ID3 decision tree in Figure 6.1 to a linear constraint database yields the set of linear constraints *Texas_ID3* shown in Table 6.3. Given the $Texas\_Weather(d, p, r, w)$ and the $Texas\_ID3(p, r, w, t)$ relations, the following Datalog query can be used to predict the temperature for each day:

```
Predict(d, t) :- Texas_Weather(d, p, r, w), Texas_ID3(p, r, w, t).
```

Table 6.3: Linear constraint representation *Texas_ID3* of the ID3 decision tree trained using the *Texas_Weather* data.

P	R	W	T	
p	r	w	t	$p = 1.73,\ t =' Low'$
p	r	w	t	$p = 0.95,\ t =' High'$
p	r	w	t	$p = 3.57,\ t =' Low'$
p	r	w	t	$p = 0.24,\ t =' High'$
p	r	w	t	$p = 0.0,\ t =' Low'$
p	r	w	t	$p = 0.31,\ t =' High'$

Instead of Datalog queries, one can use the logically equivalent SQL query:

```
CREATE VIEW Predict AS
SELECT D.d, T.t
FROM  Texas_Weather as D, Texas_ID3 as T
WHERE D.p = T.p  AND  D.r = T.r  AND  D.w = T.w
```

6.4 Representation and Querying of ID3-Interval Decision Trees

A straightforward translation from the original decision tree to a linear constraint database does not yield a good result for problems where the attributes can have real number values instead of only discrete values. Real number values are often used when we measure some attribute like the wind speed in miles-per-hour or the temperature in degrees Celsius. In this case, the ID3 algorithm will generate trees with large number of branches to perfectly fit the training data. The phenomenon, as known as *overfitting*, prevents the tree from generalizing well when classifying unforeseen data.

Hence we improve the naive translation by introducing comparison constraints $>, <, \geq, \leq$ to allow continuous values for some attributes. That is, we translate each node of the decision tree by analyzing all of its children. First, the children of each node are sorted based on the possible values of the attribute. Then, we define an interval around each discrete value based on the values of the previous and the following children. The lower

Table 6.4: Linear constraint representation *Texas_ID3-Interval* of the ID3 decision tree based on inequalities trained using the *Texas_Weather* data.

P	R	W	T	
p	r	w	t	$r < 2,\ w < 2.64,\ t =' Low'$
p	r	w	t	$r < 2,\ w \geq 2.64,\ t =' High'$
p	r	w	t	$r \geq 2,\ r < 4.3,\ p < 2.51,\ w < 8.63,\ t =' Low'$
p	r	w	t	$r \geq 2,\ r < 4.3,\ p < 2.51,\ w \geq 8.63,\ t =' High'$
p	r	w	t	$r \geq 2,\ r < 4.3,\ p \geq 2.51,\ t =' Low'$
p	r	w	t	$r \geq 4.3,\ t =' High'$

bound of the interval is defined as the median value between the value of the current child and the value of the previous child. Similarly, the upper bound of the interval is defined as the median value of the current and the following children. For instance, assume we have the values $\{10, 14, 20\}$ for an attribute for the children. This will lead to the intervals $\{(-\infty, 12], (12, 17], (17, +\infty)\}$.

Figure 5.6, which shows a modified decision tree, based on the above heuristic. Translating that modified decision tree yields the *Texas_ID3-Interval* constraint relations as defined in Table 6.4. The querying of ID3-Interval decision tree representations can be done like the querying of ID3 decision tree representations after replacing $Texas\_ID3$ with $Texas\_ID3 - Interval$.

Chapter 7

Data and Classifier Integration

"*This data has expanded without any significant structure or classification. While it is secure at basic levels, much needs to be done.*"

Mayur Raichura

7.1 Introduction

Classifications are usually done by classifiers such as support vector machines[12], decision trees[11,93], or other machine learning algorithms. After being trained on some sample data, these classifiers can be used to classify new data. However, many challenging applications require the reuse of the old classifiers to derive a new classifier. This later problem emerges in various settings, such as, the following.

Figure 7.1 shows several hospitals. Suppose that the hospitals keep the records of cardiology patients. Further suppose that their analysis of their own patients leads each of them to their own different classification of cardiology patients *for the same characteristic*, in this case, heart disease risk. Intuitively, the three hospitals could get a potentially better classification of cardiology patients if they could combine their databases. Combining their databases, called *data integration*[187,188], could be followed by running a new classification algorithm on the integrated data.

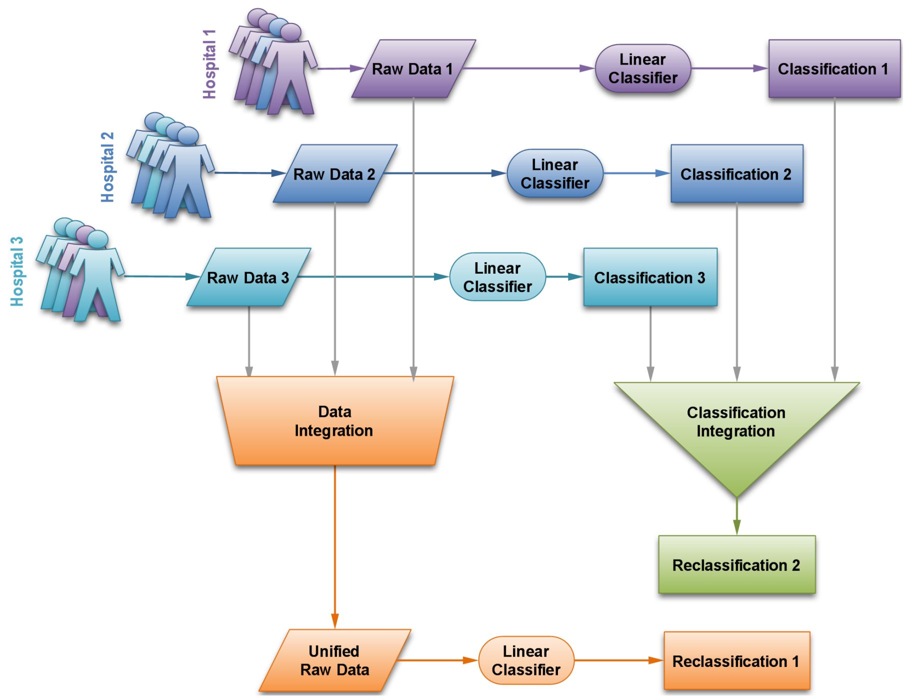


Figure 7.1: Comparison of *data integration* and *classification integration* methods.

However, the above simplistic approach often fails for several reasons. First, it is possible that the hospitals tested different variables on their own patients. This would yield many *null* values in the integrated data set. Most classification algorithms perform poorly when there are many *null* values in the data. Second, even in the absence of *null* values, the hospitals may be restricted in sharing their data due to privacy and legal concerns.

We propose the following alternative approach as a solution that avoids privacy problems: *Let the hospitals share only their classifiers.* Sharing classifiers leads to the new concept of *classification integration*, which yields another new classifier that may be better than any of the individual classifiers.

The rest of the chapter is organized as follows. Section 7.2 describes our novel *classification integration* method that relies on constraint databases. Section 7.3 describes some computer experiments and discusses their significance.

7.2 The Classification Problem with Multiple Sources

A common problem in classification arises when we have several different sources of data. For example, the Cleveland Clinic Foundation collected a heart disease data set, which is described in detail in Section 5.3. For simplicity, in this example, we consider the following smaller version (Table 7.1a) with only six randomly selected patients, where the features are: chest pain P, cholesterol C, gender G, and resting blood pressure B, and the label is disease D, which is "Yes" or "No" according to whether a patient has a heart disease. The Institute of Cardiology in Budapest, Hungary also collected data set for heart disease patients. From this data set, we may select randomly seven patients as shown on Table 7.1b.

The main problem is how to combine the two data sources and use them together. We describe two different solutions in the following two sections. In the description of those two solutions, we assumed that SVMs were used as original classifications. The method is however applicable to *any* linear classifier.

7.2.1 Data Integration

To solve the classification problem raised above, we can use *data integration*[187,188]. In data integration, we first take the union of the two data sets, yielding the integrated data set with thirteen patients shown in Table 7.2.

Table 7.1: Sample Patients from Cleveland and Budapest hospitals from the simplified *Heart Disease* database.

(a) Cleveland Patients

P	C	G	B	D
1	233	1	145	No
3	250	1	130	Yes
3	275	0	110	No
4	230	1	117	Yes
2	198	0	105	No
4	266	1	124	Yes

(b) Budapest Patients

P	C	G	B	D
1	237	0	170	No
2	219	0	100	No
4	270	1	120	Yes
2	198	0	105	No
4	246	0	100	Yes
1	156	1	140	Yes
2	257	1	110	Yes

Table 7.2: Integrated *Budapest-Cleveland Patients* data set.

P	C	G	B	D
1	233	1	145	No
3	250	1	130	Yes
3	275	0	110	No
4	230	1	117	Yes
2	198	0	105	No
4	266	1	124	Yes
1	237	0	170	No
2	219	0	100	No
4	270	1	120	Yes
2	198	0	105	No
4	246	0	100	Yes
1	156	1	140	Yes
2	257	1	110	Yes

Second, we use the integrated data set from Table 7.2 to build a linear classification using a SVM. That would result in a linear classifier, which can be represented by the linear constraint relation defined in Table 7.3. We obtained the constraint relation from the SVM classification by applying our method described in Section 6.2.

Table 7.3: New SVM classification derived from the integrated *Cleveland-Budapest* data.

P	C	G	B	D	
p	c	g	b	d	$0.8696p - 0.0024c + 1.7055g + 0.0052b - 3.5154 = d$

7.2.2 Classifier Integration with Constraint Databases

Considering the *Cleveland Patients* table, the Cleveland Clinic Foundation may build the following linear classifier using a SVM. Using our method described in Section 6.2, the SVM classifier may be represented in a linear constraint relation as follows in the Table 7.4a. Similarly, the Institute of Cardiology in Budapest, Hungary may build based on its data set the linear classification using a SVM (Table 7.4b).

Table 7.4: SVM-based *Cleveland Classification* and *Budapest Classification* trained using the *Cleveland Patients* and *Budapest Patients* data sets respectively.

(a) Cleveland Classification

P	C	G	B	D_1	
p	c	g	b	d_1	$1.1568p - 0.0172c + 0.4278g + 0.0333b - 3.404 = d_1$

(b) Budapest Classification

P	C	G	B	D_2	
p	c	g	b	d_2	$1.0213p - 0.0016c + 1.9091g + 0.015b - 4.2018 = d_2$

Table 7.5: Integrated Classification of the *Cleveland* and *Budapest* SVM Classifications.

(a) Integrated Classification

P	C	G	B	D	
p	c	g	b	d	$d = 0.5d_1 + 0.5d_2$, $1.1568p - 0.0172c + 0.4278g + 0.0333b - 3.404 = d_1$, $1.0213p - 0.0016c + 1.9091g + 0.015b - 4.2018 = d_2$

(b) Integrated Classification after Simplification

P	C	G	B	D	
p	c	g	b	d	$1.0891p - 0.0094c + 1.1685g + 0.0242b - 3.8029 = d$

As an alternative to data integration (see Section 7.2.1), we propose the use of *classification integration* by taking the natural join of the two linear constraint relations and then selecting the value $d = 0.5d_1 + 0.5d_2$. That yields the classification described in Table 7.5. Since the integrated classification is based on two data sets, it can be expected to be better than either the Cleveland classification or the Budapest classification in itself.

7.3 Experimental Comparison of the Data Integration and Classifier Integration Methods

7.3.1 Experimental Protocol and Results

We compare the *data integration* and the *classification integration* methods assuming that both methods use either SVMs or ID3-Interval decision trees as the original linear classifiers. In our experiments, we used the SVM implementation from the LibSVM[186] and our implementation of ID3-Interval as described in Section 6.4. The experiments were conducted using the *Heart Disease Diagnosis* data set, which includes the records of 10 features for 720 patients collected in three different hospitals (see Section 5.3 for details). We used the following protocol, where n is a parameter controlling the size of the training set:

1. `Randomly select 60 patients from the three hospitals as a testing set.`
2. `Randomly select` n `percent of the remaining patients as a training set.`
3. `Build a SVM classification` f_{123} `on the union of the training data.`
4. `Build a SVM classification` f_1, f_2, f_3 `for each training data source.`
5. `Integrate` f_1, f_2, f_3 `using` ***`classification integration`*** `to find` f_{class_integ}.
6. `Test the accuracy of` f_{123} `and` f_{class_integ} `on the testing set.`

Figures 7.2 and 7.3 report the average results of repeating the above procedure ten times for each n equal to 5%, 15%, 25%, …, 95% using SVMs and ID3 decision trees respectively as original classifiers. The statistical significance of the results was analyzed using Student's two-tailed paired T-test[189].

We also simulated missing values by generating the three following subsets from the original data set by removing two features from each source:

- BUDAPEST with features (1, 2, 3, 4, 5, 7, 9, 10)
- CLEVELAND with features (1, 2, 3, 5, 7, 8, 9, 10)
- ZÜRICH with features (1, 2, 3, 5, 6, 7, 9, 10)

In each subset, we used the first seven features to predict the last one. We used those three subsets in a similar manner as described in the above procedure. Figures 7.4 and 7.5 reports the average results of repeating the above procedure ten times for each n equal to 5%, 15%, 25%, …, 95% using SVMs and ID3 decision trees respectively as original classifiers as well as the two-tailed p value of Student's paired T-test.

The *classification integration* method was significantly (T-test $p = 0.01\%$) more accurate than the *data integration* method when both used SVMs (Figures 7.2 and 7.4). No significant accuracy improvement (T-test $p = 8.81\%$) was noted when both used decision trees when no data was missing (Figure 7.3) whereas some improvement (T-test $p = 0.35\%$) was reported in the case where the dataset is sparse (Figure 7.5). Surprisingly, the improvement δ obtained using classification integration was more important when the data set contains missing values ($\delta = 5.20\%$) than when considering a complete dataset ($\delta = 0.45\%$). That surprising result seems to be because the *classification integration* method relies on constraint databases, which can perform efficient joins on sparse data. That is another benefit of *classification integration* in addition to its simplicity of relying on only the classifiers instead of the raw data.

7.3.2 Discussion

The experiments suggest that classification integration has an advantage in preserving relationships found by the original classifiers. As a hypothetical example, if chest pain is strongly related to heart attack, it will be recognized as such in a hospital where the chest pain of each patient is recorded. However, if the data from that hospital is aggregated with data from other hospitals that did not measure chest pain, then the relationship of chest pain and heart attack may be less clear and recognizable. In that case, a classifier used on the integrated data may miss some of the relationships. That seems to be the reason for the increase of the relative performance of the classification integration over data integration when there are missing values. If the amount of missing information is further increased, then classification integration will likely outperform data integration even with decision trees.

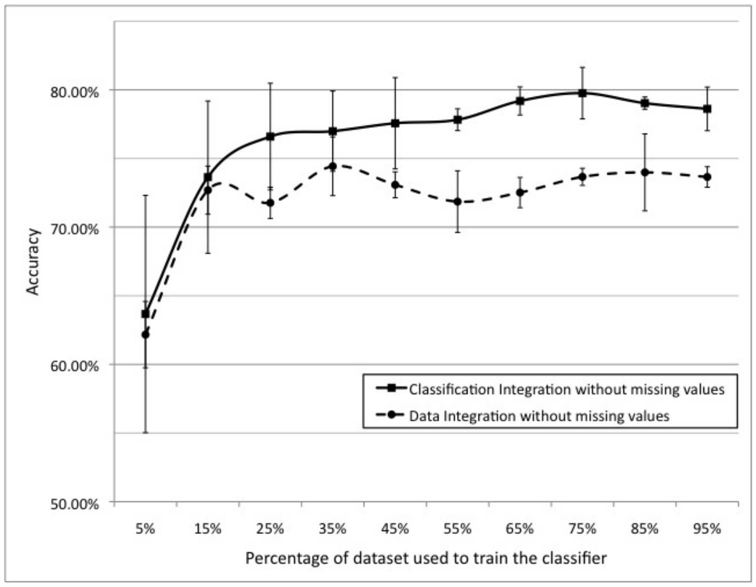


Figure 7.2: Comparison of *data integration* with *classification integration* using SVMs without missing values in the training set. T-test $p = 0.01\%$. $\delta = 4.31\%$.

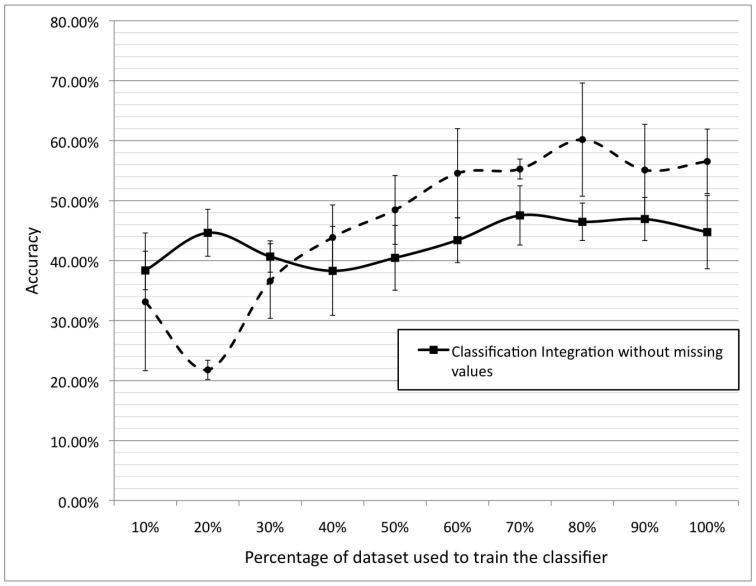


Figure 7.3: Comparison of *data integration* with *classification integration* using ID3 decision trees without missing values in the training set. T-test $p = 8.81\%$. $\delta = -3.40\%$.

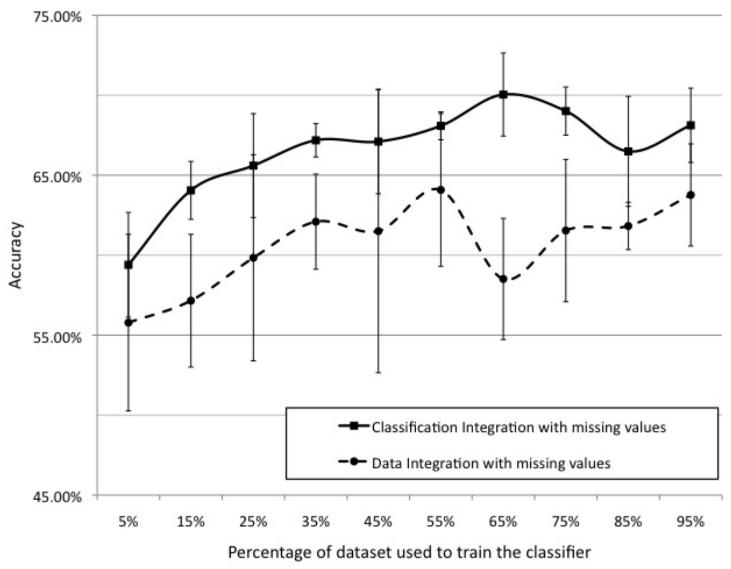


Figure 7.4: Comparison of *data integration* and *classification integration* using SVMs with missing values in the training set. T-test $p = 0.01\%$. $\delta = 5.90\%$.

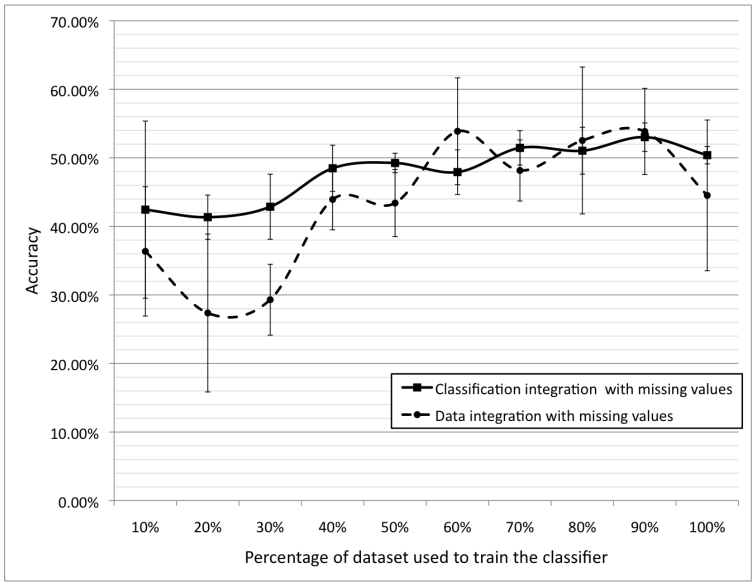


Figure 7.5: Comparison of *data integration* and *classification integration* using ID3 decision trees with missing values in the training set. T-test $p = 0.35\%$. $\delta = 4.49\%$.

Chapter 8

Data Reclassification

"The classification process will also be covered,
because too much information is classified and more should be reclassified [...]
We probably have some overclassification situations of our own."

Peter Hoekstra

8.1 Introduction

The *Classification Integration* problem is further complicated when two classifiers classify *different characteristics*. For example, suppose that one classifier tests whether patients have disease A and another classifier tests whether they have disease B. If one wants to combine these two classifiers, then one would need a classifier for patients who (1) have both diseases, (2) have only disease A, (3) have only disease B, and (4) have neither disease. In general, when combining n classifiers, there are 2^n combinations to consider. Hence, many applications would be simplified if a single combined classifier could be found.

For example, Figures 5.4 and 5.5 present two different ID3 decision tree classifiers for the status of patients and the efficiency of a drug. For simplicity, the status of patients is classified as *alive, dead*, or *transplanted*, and the drug efficiency is classified as *penicillamine* and *placebo*. Analysis of such decision trees is difficult. For instance, just by looking at

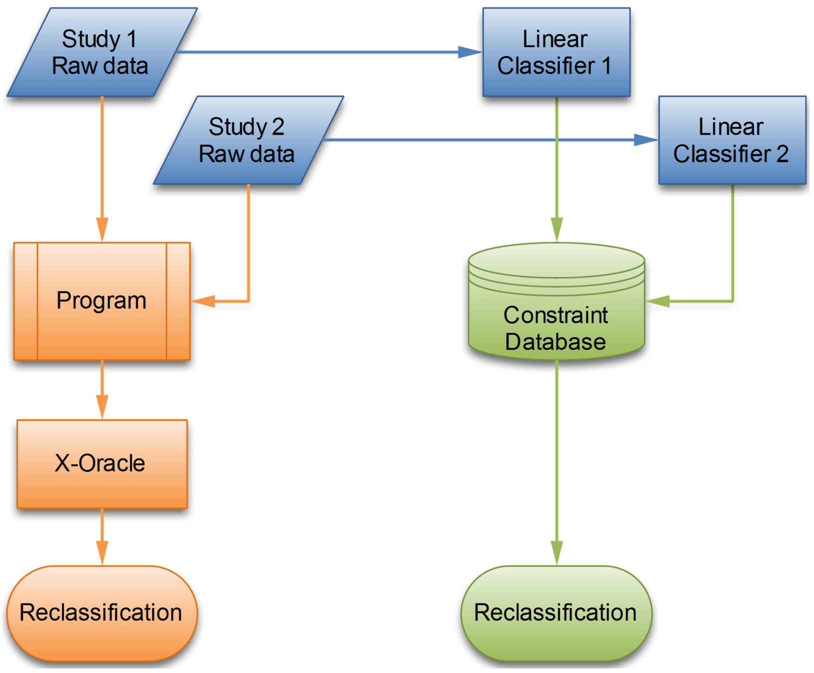


Figure 8.1: Comparison of the *reclassification with an oracle* and *reclassification with constraint databases* methods.

these decision trees, it is hard to tell whether any patient died after taking a placebo, or whether any patient who used the drug is still alive. The problem with the above simple-looking queries is that no decision tree contains both *patient status* and *drug efficiency*. Finding a single decision tree that contains both *patient status* and *drug efficiency* would provide a convenient solution to the above queries. This chapter introduces a novel constraint database-based reclassification method that results in a single classifier and enables to answer many challenging queries on medical data.

We first review the reclassification problem by giving an example in Section 8.2. Then we introduce several new reclassification methods. Section 8.3 describes the *Reclassification with an oracle* method. While oracle-based methods do not exist in practice, those methods give a theoretical limit to the best possible practical methods. Section 8.4 describes the practical *Reclassification with constraint databases* method. A comparison of these two methods is given later in Section 8.5.

8.2 The Reclassification Problem

One study found a classifier for the status of patients with *primary biliary cirrhosis* (see Section 5.4 for a detailed description of the dataset) using:

$$X_1 = \{Cholesterol(C), Gender(G), Hepatomegaly(H), Triglicerides(T)\}$$

and the class for patient status:

$$S = \{Alive, Dead, Transplanted\}$$

where cholesterol is the serum cholesterol in mg/dl, hepatomegaly is '1' if the liver is enlarged and '0' otherwise, gender is '0' for male and '1' for female, and triglicerides level is measured in mg/dl. Figure 5.4 shows an example decision tree for the status of patients obtained after training by 50 random examples. A sample training data is shown in Table 8.1a.

Another study found a classifier to distinguish between a real drug and a placebo using:

$$X_2 = \{Bilirubin(B), Cholesterol(C), Gender(G), Hepatomegaly(H)\}$$

and the class for drug:

$$D = \{Penicillamine, Placebo\}$$

where the serum bilirubin of the patient is measured in mg/dl.

Figure 5.5 shows an example of decision tree for the efficiency of a drug obtained after training by 50 random examples. A sample training data for the second study is shown in Table 8.1b.

Table 8.1: Example of medical records for the status of patients and the efficiency of penicillamine.

(a) Patient Status

C	G	H	T	S
261	1	1	172	Alive
200	0	0	143	Transplanted

(b) Drug Efficiency

B	C	G	H	D
3.6	244	0	0	Placebo
7.4	54	0	0	Penicillamine

Building a new classifier for (X, Y) seems easy, but the problem is that there is no database for (X, Y). Finding such a database would require a new study with more data collection, which would take a considerable time. That motivates the need for reclassification. As Section 8.3 shows, a classifier for (X, Y) can be built by an efficient reclassification algorithm that uses only the already existing classifiers for (X_1, S) and (X_2, D).

Suppose we need to find a classifier for

$$X = X_1 \cup X_2 = \{Bilirubin, Cholesterol, Gender, Hepatomegaly, Triglicerides\}$$

and

$$\begin{aligned} Y = S \times D \ = \ & \{Alive\_Penicillamine,\ Alive\_Placebo, \\ & Dead\_Penicillamine,\ Dead\_Placebo, \\ & Transplanted\_Penicillamine,\ Transplanted\_Placebo\} \end{aligned}$$

The next two sections present two different solutions to this problem.

8.3 Reclassification with an Oracle

In theoretical computer science, researchers study the computational complexity of algorithms in the presence of an oracle that tells some extra information that can be used by the algorithm. The computational complexity results derived using oracles can be useful in establishing theoretical limits to the computational complexity of the studied algorithms.

Similarly, in this section we study the reclassification problem with a special type of oracle. The oracle we allow can tell the value of a missing attribute of each record. That allows us to derive essentially a theoretical upper bound on the best reclassification that can be achieved. The reclassification with oracle method extends each of the original relations with the attributes that occur only in the other relation. Then one can take a union of the extended relations and apply any of the chosen classification algorithms. We illustrate the idea behind the *Reclassification with an oracle* method using an extension of the example described in Section 8.2 and the ID3 algorithm.

First, we add a bilirubin measure and a drug class to each record in the *Patient Status* relation using an oracle. Suppose we obtain the relation shown in Table 8.2a. Second,

Table 8.2: Modification of the *Patient Status* and *Drug Efficiency* relations using an oracle.

(a) Patient Status

B	C	G	H	T	S	D
14.1	261	1	1	172	Alive	Placebo
5.3	200	0	0	143	Transplanted	Penicillamine

(b) Drug Efficiency

B	C	G	H	T	S	D
3.6	244	0	0	114	Alive	Placebo
7.4	54	0	0	189	Transplanted	Penicillamine

we add *Triglicerides* measurements and a *Status* class to each record in the *Drug Efficiency* relation using an oracle. The resulting relation is shown in the relation shown in Table 8.2b.

After the union of these two relations, we can train an ID3 decision tree to yield a reclassification as needed to complete the example from Section 8.2. The above reclassification method is not in general a practical method because oracles do not exist.

8.4 Reclassification with Constraint Databases

The *Reclassification with Constraint Databases* method has two main steps:

1. **Translation to Constraint Relations:** We translate the original linear classifiers to a constraint database representation. Our method does not depend on any particular linear classification method.

2. **Join:** The linear constraint relations are joined together using a constraint database join operator[15,16].

We give first an example to illustrate the *Translation to Constraint Relations.* Assume the following variables in the feature space:

- b, the bilirubin serum,

- c, the cholesterol rate,

- g, the gender of the patients,
- h, the hepatomegaly,
- t, the triglicerides rate.

Suppose we try to predict the drug efficiency d on each patient. Then we use the decision tree shown in Figure 5.5 to classify the efficiency of the drug. The decision tree may be translated into linear constraint as shown in Table 8.3. We also try to predict the status s. Similarly, the decision tree represented in Figure 5.4 is used to generate the constraint relation in Table 8.4.

The reclassification problem can be solved by a constraint database join of the *Drug* and *Status* relations, expressed in non-recursive Datalog as follows:

```
Reclass(b,c,g,h,t,d,s) :- Drug(b,c,g,h,d), Status(c,g,h,t,s).
```

The evaluation of the above query requires a constraint database join, which is explained in detail, for example in the textbook[16]. To find the value of the *Reclass* relation, the constraint join operator will try to combine every constraint tuple of *Drug* with every constraint tuple of *Status*. If the combination of the two tuples yields an unsatisfiable set of constraints, then it is discarded. Only those combinations are kept that are satisfiable

Table 8.3: Constraint database representation of the *Drug Efficiency* classification obtained using the decision tree shown in Figure 5.5. Each row describes a path from the root to a leaf in the decision tree.

B	C	G	H	D	
b	c	g	h	d	$c < 74,\ g = 1,\ d =$ 'Placebo'
b	c	g	h	d	$c < 74,\ g = 0,\ b < 13.2,\ d =$ 'Penicil.'
b	c	g	h	d	$c < 74,\ g = 0,\ b > 13.2,\ d =$ 'Placebo'
b	c	g	h	d	$c > 74,\ c < 218,\ b < 5.7,\ d =$ 'Penicil.'
b	c	g	h	d	$c > 74,\ c < 218,\ b > 5.7,\ d =$ 'Placebo'
b	c	g	h	d	$c > 218,\ h = 0,\ g = 1,\ d =$ 'Penicil.'
b	c	g	h	d	$c > 218,\ h = 0,\ g = 0,\ d =$ 'Placebo'
b	c	g	h	d	$c > 218,\ h = 1,\ d =$ 'Placebo'

Table 8.4: Constraint database representation of the *Patient Status* classification obtained using the decision tree shown in Figure 5.4. Each row describes a path from the root to a leaf in the decision tree.

C	G	H	T	S	
c	g	h	t	s	$h = 1,\ c < 106,\ s =$'Alive'
c	g	h	t	s	$h = 1,\ c > 106,\ c < 357,\ g = 1,\ s =$'Alive'
c	g	h	t	s	$h = 1,\ c > 106,\ c < 357,\ g = 0,\ s =$'Transp.'
c	g	h	t	s	$h = 1,\ c > 357,\ s =$'Dead'
c	g	h	t	s	$h = 0,\ t < 120,\ s =$'Alive'
c	g	h	t	s	$h = 0,\ t > 120,\ t < 231,\ c < 472,\ s =$'Transp.'
c	g	h	t	s	$h = 0,\ t > 120,\ t < 231,\ c > 472,\ s =$'Dead'
c	g	h	t	s	$h = 0,\ t > 231,\ s =$'Alive'

Table 8.5: First five satisfiable constraint tuples for the reclassification of the *Drug* and *Status* relation. The complete reclassification leads to 64 possible combinations to consider.

B	C	G	H	T	D	S	
b	c	g	h	t	d	s	$c < 74,\ g = 1,\ d =$ 'Placebo', $h = 1,\ c < 106,\ s =$'Alive'
b	c	g	h	t	d	s	$c < 74,\ g = 1,\ d =$ 'Placebo' $h = 0,\ t < 120,\ s =$'Alive'
b	c	g	h	t	d	s	$c < 74,\ g = 1,\ d =$ 'Placebo' $h = 0,\ t > 120,\ t < 231,\ c < 472,\ s =$'Transplanted'
b	c	g	h	t	d	s	$c < 74,\ g = 1,\ d =$ 'Placebo' $h = 0,\ t > 231,\ s =$'Alive'
b	c	g	h	t	d	s	$c < 74,\ g = 0,\ b < 13.2,\ d =$ 'Penicillamine' $h = 1,\ c < 106,\ s =$'Alive'
							⋮

for some input values for the features. As there are eight constraint tuples in both *Drug* and *Status*, there are 64 possible combinations to consider. Table 8.5 shows the first five satisfiable constraint tuples of *Reclass*.

In the *Reclass* relation, the first tuple is a combination of the first tuple of *Drug* with the first tuple of *Status*. A constraint database would usually not only check satisfiability but also simplify the constraints. That would mean deleting unnecessary constraints. For example, $c < 106$ is unnecessary because it is already implied by $c < 74$, which is the first constraint in the first tuple of *Reclass*.

Next, the first tuple of *Drug* and the second tuple of *Status* are combined. However, that is unsatisfiable because one contains $c < 74$ while the other contains $c > 106$. Hence this combination is not part of *Reclass*. Similarly, we also discard the combination of the first tuple of *Drug* with the third and fourth tuples of *Status*. Therefore, the next combination that is satisfiable is the first tuple of *Drug* and the fifth tuple of *Status*. This satisfiable combination is recorded as the second constraint tuple of *Reclass*. The other tuples are determined similarly. With n constraint tuples in both input relations, the overall time complexity of the constraint join operator is $O(n^2)$[16], that is, it is as efficient as a relational database join operator.

Let *Patients(id,b,c,g,h,t)* be a relation that records the identification number and the features of each patient. Then the following non-recursive Datalog query can be used to predict the drug efficiency and the status for each patient:

```
Predict(d,s) :- Patients(id,b,c,g,h,t), Reclass(b,c,g,h,t,d,s).
```

Instead of Datalog queries, one can use the logically equivalent SQL query shown below. The SQL query is evaluated similarly to the Datalog query. It is largely a stylistic preference whether one uses SQL or Datalog queries.

```
CREATE VIEW Predict AS
SELECT R.d, R.s
FROM  Patients as P, Reclass as R
WHERE P.b = R.b AND P.c = R.c AND P.g = R.g AND
      P.h = R.h AND P.t = R.t
```

8.5 Comparison of the Reclassification with an Oracle and Constraint Databases Methods

We tested our methods using both SVMs and decision trees. However, our goal is not to compare SVMs with decision trees but to show the versatility of our methods. Note that each method can use either SVMs or decision trees, but comparing a method that uses SVM with another method that uses decision trees cannot decide which method is better because the difference may be due only to the differences between SVMs and decision trees. The experiments were performed using the CRCARS dataset as reported in Section 8.5.1 and with the PBC database in Section 8.5.2. The significance of the observed results was established using a two-tailed paired T-test.

8.5.1 Experimental Results with the CRCARS data set

In this section, we use the CRCARS data set described in Section 5.1 to evaluate our *Reclassification with Constraint Database* method. Our comparison of the accuracies of the original ID3 classifier with the reclassified individual classifiers for the predictions of the fuel efficiency (Figure 8.2) and the origin of cars (Figure 8.3) revealed the proposed method significantly (T-test $p = 0.01\%$ and $p = 0.07\%$ respectively) improved the reclassification with an oracle method. Second, we reclassified and combined the two ID3 classifications using the *reclassification with an oracle* and the *reclassification with constraint database* methods. The result of the experiment is shown in Figure 8.4. No significant difference between the two reclassification methods was identified (T-test $p = 44.56\%$).

8.5.2 Experimental Results with the PBC database and Discussion

In this section, we compare the *reclassification with an oracle* and the *reclassification with constraint databases* methods. We also compare *reclassification with constraint databases* with the original linear classification for the DISEASE and the DRUG classes.

Figures 8.5 and 8.6 show the results of our experiments using SVMs as original classifications for the prediction of DISEASE and DRUG respectively. The constraints defining the separating hyperplane of the SVMs can be exactly represented in the constraint database; hence the accuracies of the original SVMs and their constraint database representation are

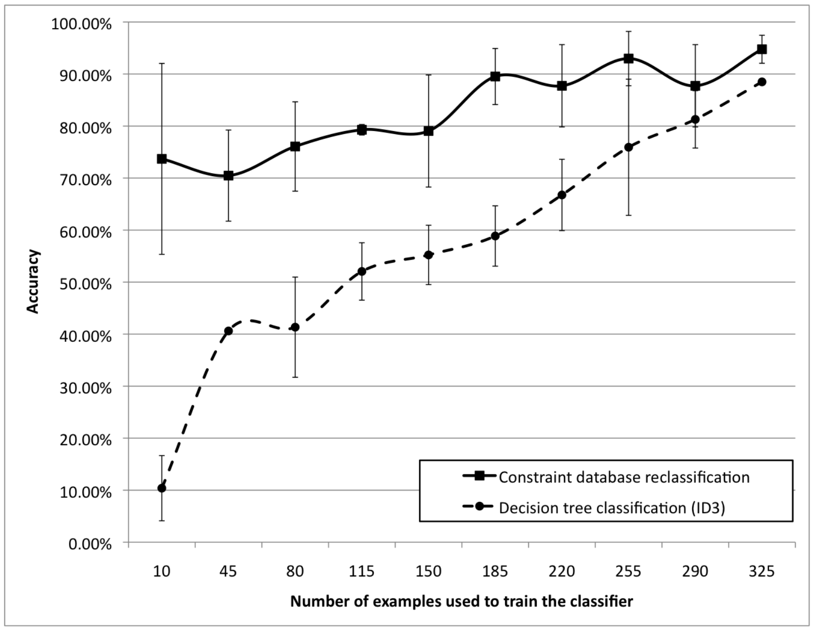


Figure 8.2: Comparison of the *Reclassification with constraint databases* and the original *Classification with a decision tree (ID3)* for the prediction of the class MPG efficiency of the cars. T-test $p = 0.01\%$. $\delta = 26.04\%$.

identical. Similarly Figures 8.7 and 8.8 show the results with ID3 decision trees for the prediction of DISEASE and DRUG respectively. Constraint databases provide a more flexible representation of the original classifications. The experimental results show that constraint databases may significantly (T-test $p = 0.01\%$) improve the accuracy of the classification depending on the choice of the original classifier.

Figures 8.9 and 8.10 compares the accuracies of the *reclassification with an oracle* and the *reclassification with constraint databases* for SVMs and decision trees respectively. The results show that both perform similarly for both SVMs (T-test $p = 32.61\%$) and decision trees ($p = 13.15\%$). That proves that our practical *reclassification with constraint databases* method achieves the theoretical limit represented by the *reclassification with an oracle* method. By theoretical limit we mean in this example only the maximum achievable with the use of the original linear classification algorithm.

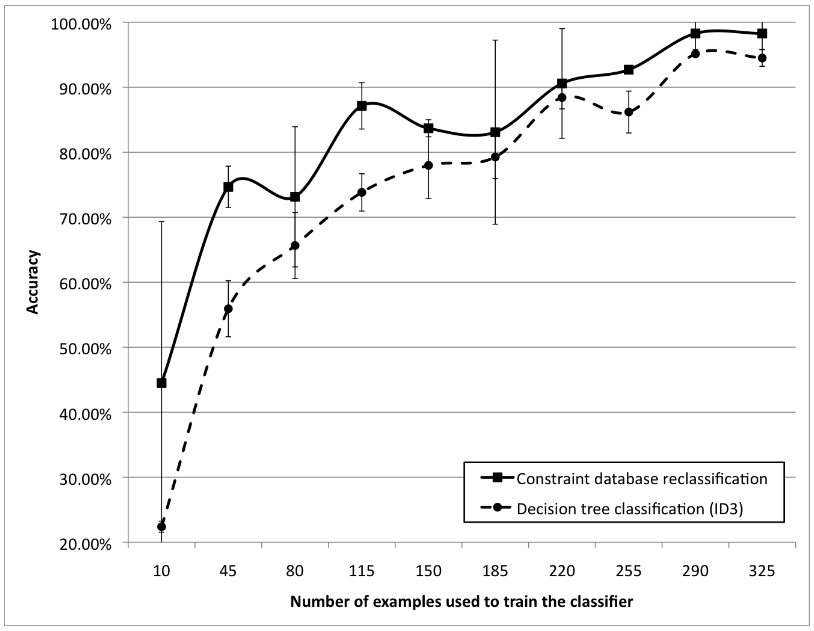


Figure 8.3: Comparison of the *Reclassification with constraint databases* and the original *Classification with a decision tree (ID3)* for the prediction of the class ORIGIN of the cars. T-test $p = 0.07\%$. $\delta = 8.68\%$.

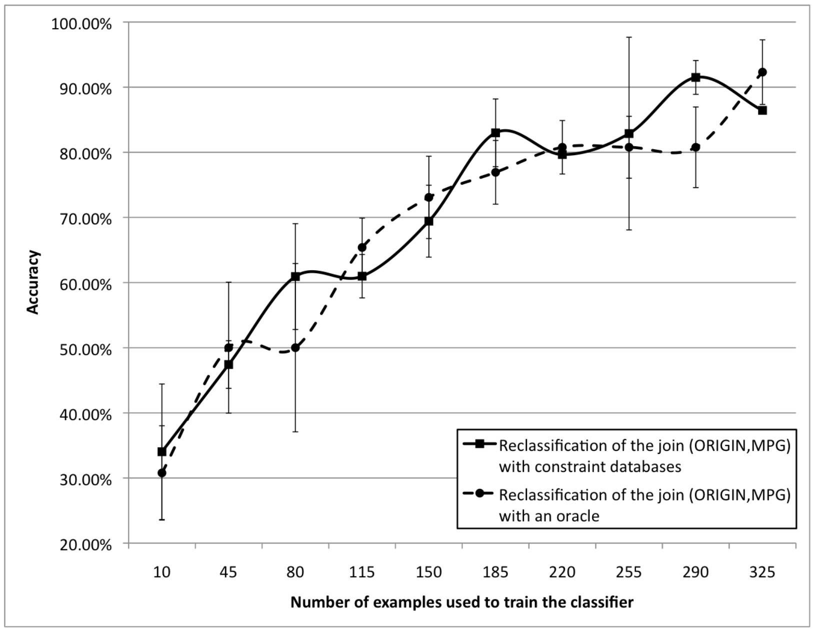


Figure 8.4: Comparison of the *Reclassification with constraint databases* and the *Reclassification with an oracle* using the CRCARS database. T-test $p = 44.56\%$. $\delta = 1.54\%$.

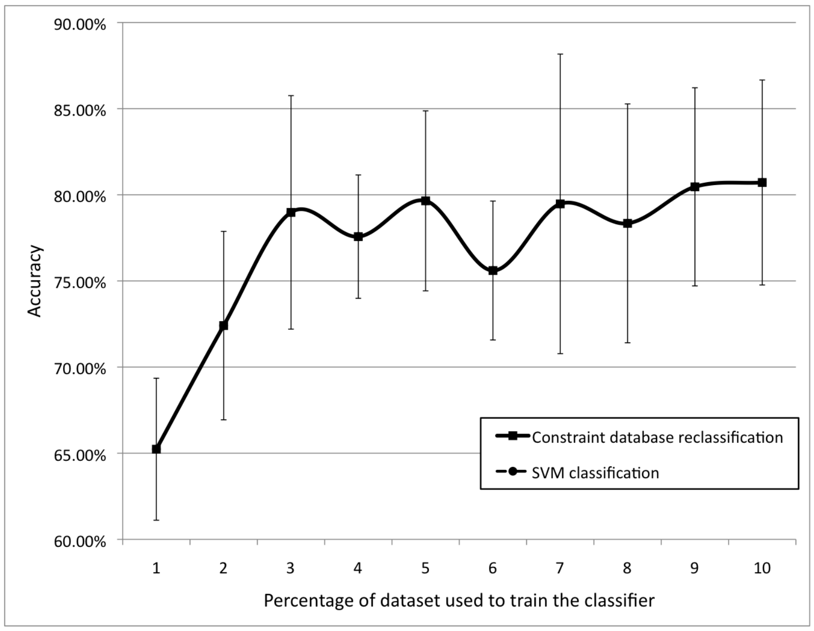


Figure 8.5: Prediction of DISEASE from the primary biliary cirrhosis data using SVMs. Note that the SVMs and their constraint database representations are identical.

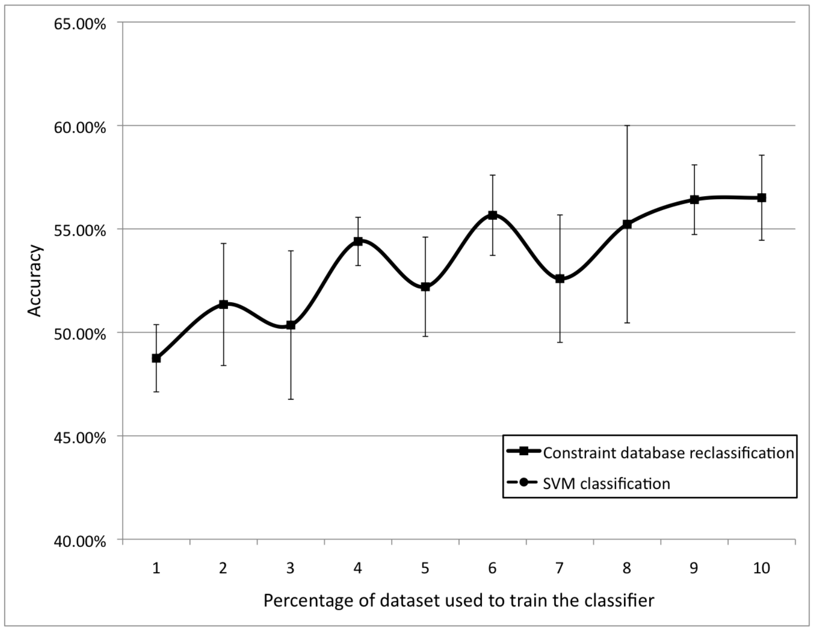


Figure 8.6: Prediction of DRUG from the primary biliary cirrhosis data using SVMs. Note that the SVMs and their constraint database representations are identical.

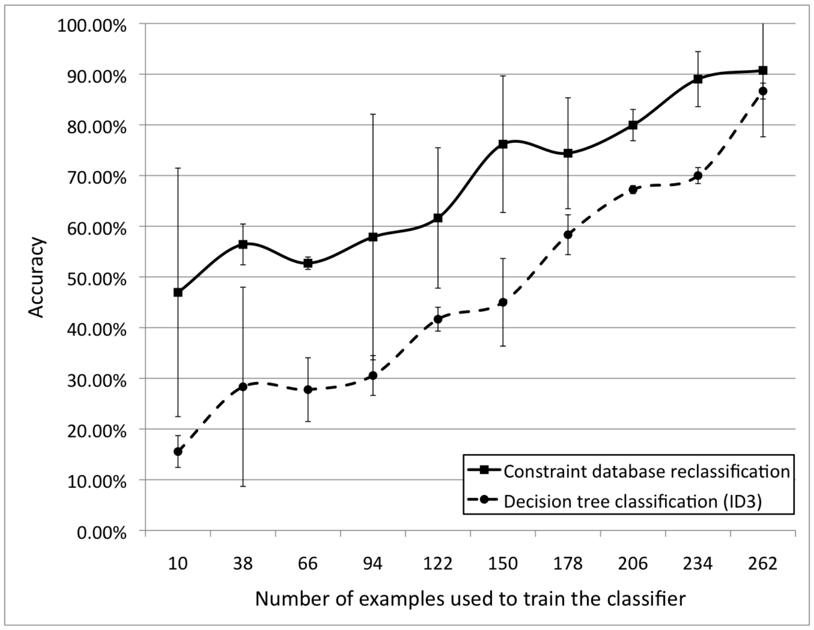


Figure 8.7: Prediction of DISEASE from the primary biliary cirrhosis data using ID3 decision trees. T-test $p = 0.01\%$. $\delta = 21.48\%$.

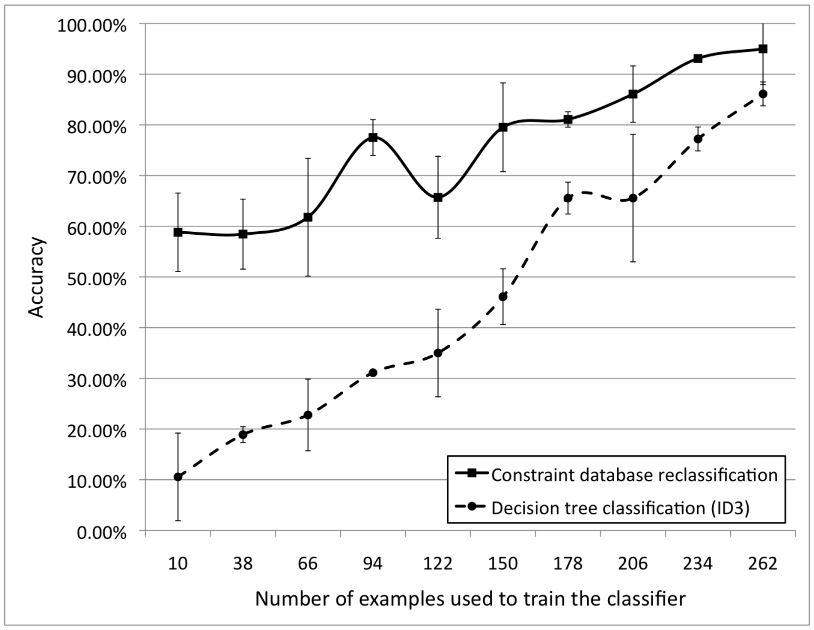


Figure 8.8: Prediction of DRUG from the primary biliary cirrhosis data using ID3. T-test $p = 0.01\%$. $\delta = 29.81\%$.

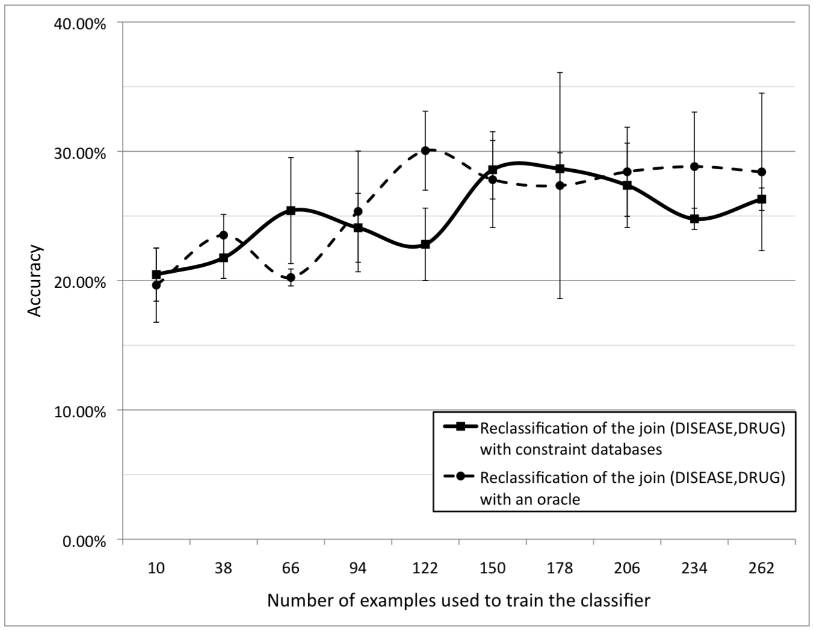


Figure 8.9: Prediction of DISEASE_DRUG with the primary biliary cirrhosis data using SVMs. T-test $p = 32.61\%$. $\delta = -0.94\%$.

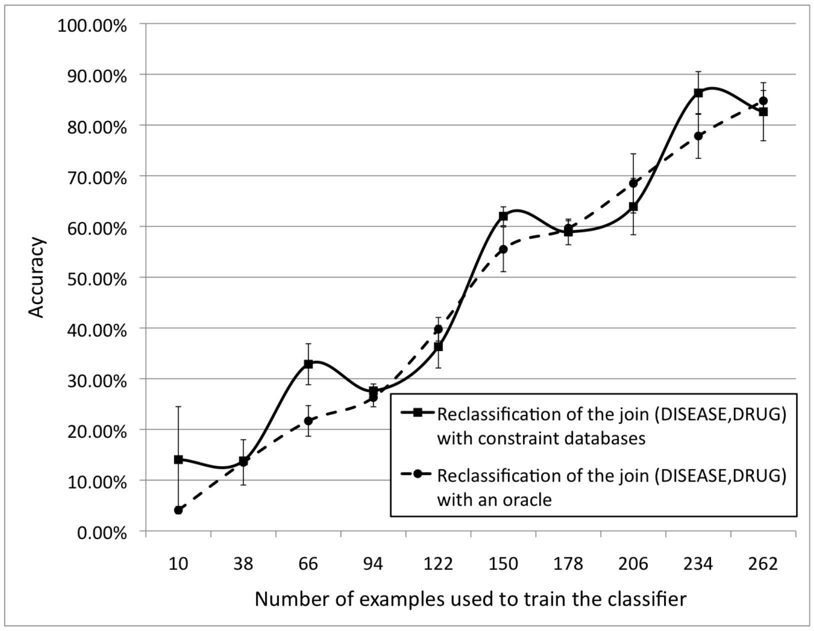


Figure 8.10: Prediction of DISEASE_DRUG with primary biliary cirrhosis data using decision trees. T-test $p = 13.15\%$. $\delta = 2.69\%$.

Chapter 9

Temporal Classification

"*We look at historical data and do some averaging. That number is our best guess.*"

John McDowell

9.1 Introduction

Data classifiers, such as SVMs (see Section 2.3.2), decision trees (see Section 2.3.3), or other machine learning algorithms, are widely used. However, they are used to classify data that occur in the same time period. For example, a set of cars can be classified according to their fuel efficiency. That is acceptable because the fuel efficiency of cars is not expected to change much over time. Similarly, we can classify a set of people according to their current heart condition. However, people's heart condition can change over time. Therefore, it would be more interesting to classify people using the current information according to whether they are likely to develop serious heart condition some time in the future.

Consider a patient who transfers from one doctor to another. The new doctor may give the patient a set of tests and use the new results to predict the patient's prospects. The question arises whether this prediction could be enhanced if the new doctor would get the older test results of the patient. Intuitively, there are cases where the old test results could be useful for the doctor. For example, the blood pressure of a patient may be 130/80, which

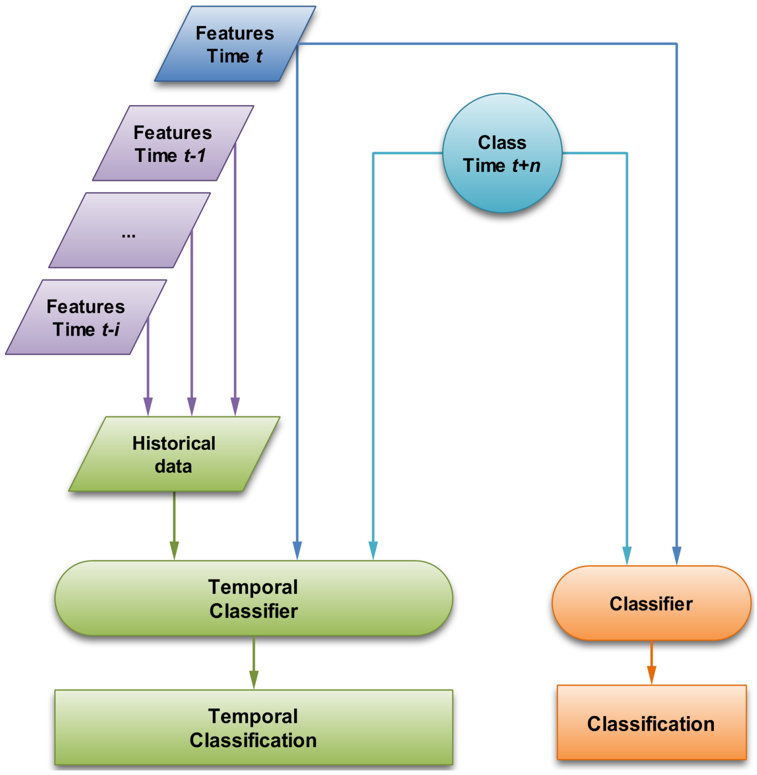


Figure 9.1: Comparison of the standard and the temporal classification methods.

may be considered within normal. However, if it was 120/80 last year and 110/80 the year before, then the doctor may be still concerned about the steady rise of the patient's blood pressure. On the other hand, if the patient's blood pressure used to be around 130/80, then the doctor may be more confident of predicting the patient to be in good health. Therefore, the history of the patient is important in distinguishing between these two cases.

Nevertheless, the temporal history of data is usually overlooked in the machine learning area. There are only a few previous works that combine some kind of spatio-temporal data and classification algorithms. Seidel *et al.*[190] used linear regression with historical meteorological and hydrological data for the reconstruction of historical ood events. Qin & Obradovic[191] are interested in incrementally maintaining an SVM classifier when new data is added to a database. Therefore, their method is not useful to predict the future health of a patient or other classes that one may want to predict for the future. Tseng & Lee[192] classify temporal data using probabilistic induction.

Previous chapters of this work considered only data integration and reclassification by classifiers when all the data was measured at the same time. In this chapter, we propose a new temporal classification method that instead of probabilistic induction[192] extends existing linear classifiers to deal with temporal data. Figure 9.1 compares the standard classifiers and the new temporal classifier method. The standard classifiers take as input the current (at time t) values of the features in the feature space and the class label some n time units ahead (at time $t+n$). The temporal classifiers take as input in addition to the current features and the class, the *history*, that is, the old values of the features up to some i time units back in time (that is, from time $t-i$ to $t-1$).

Weather forecasting is a challenging task. It is also natural to study because the major interest is in the prediction of the weather ahead of time instead of describing the current conditions. We tested our temporal classifier on a meteorological database of the Texas Comission on Environmental Quality. At a first glance it would seem useless to look at the weather history back more than a couple of days. Surprisingly, we discovered that the history does matter more than expected and the classification can be improved if one looks back 15 days back in time.

We were also surprised that the history of some features were considerably more useful than the history of the others. Moreover, the features that are the most important when looking at only time t are not the same as the features that are important when one looks at the weather history. That happens because the different the features have different permanency. For example, wind direction may change greatly form one hour to another. On the other hand, ozone levels are fairly constant.

Sections 2.3 and 2.4 reviewed classifiers and constraint databases respectively. Chapter 6 described our database representation and querying of linear classifiers. These representations are used in our implementations. The rest of this chapter is organized as follows. Section 9.2 presents the new temporal classification method and a corresponding data mapping. Section 9.3 describes computer experiments with the TCEQ database and discusses the results. Section 9.4 compares the proposed temporal classification method with the IDW interpolation (reviewed in Section 2.5) using the FLU dataset.

9.2 Temporal Classifications with Historical Data

The $Texas\_Weather$ database described in Section 5.5 is an atypical data for linear classifiers because it involves a temporal dimension. Although one may consider each day as an independant instance and simply ignore the temporal dimension, as we did earlier, it probably would not be the best solution. Instead, we propose below a temporal classification method for dealing with temporal data. The temporal classification method is based on an alternative representation of the database. As an example, the $Texas\_Weather(d, p, r, w, t)$ relation (Table 6.1) can be rewritten into the temporal relation:

$$Texas\_Weather\_History(d, p_{d-2}, r_{d-2}, w_{d-2}, p_{d-1}, r_{d-1}, w_{d-1}, p_d, r_d, w_d, t)$$

where for any feature $f \in \{p, r, w\}$ the f_i indicates the day i when the measurements are taken.

Note that even though we did not use in $Texas\_Weather$ any subscript, the implicit subscript for the features was always d. Now the subscripts go back in time, in this particular representation two days back to $d - 1$ and $d - 2$. The $Texas\_Weather\_History$ relation is shown in Table 9.1.

The $Texas\_Weather\_History$ relation uses the same set of feature measures as the $Texas\_Weather$ relation because the data in the $P_{d-2}, R_{d-2}, W_{d-2}$ and the $P_{d-1}, R_{d-1}, W_{d-1}$ columns are just shifted values of the P_d, R_d, W_d columns. However, when the $Texas\_Weather\_History$ relation is used instead of the $Texas\_Weather$ relation to generate one of the linear classifiers, then represented and queried as in Chapter 6, then there is a potential for improvement because each training data includes a more complete set of features.

Table 9.1: The *Texas_Weather_History* relation illustrates how the original temporal data can be remapped to take advantage of the historical data.

D	P_{d-2}	R_{d-2}	W_{d-2}	P_{d-1}	R_{d-1}	W_{d-1}	P_d	R_d	W_d	T
3	1.73	2.47	-1.3	0.95	3.13	9.32	3.57	3.56	4.29	Low
4	0.95	3.13	9.32	3.57	3.56	4.29	0.24	1.84	1.51	Low
5	3.57	3.56	4.29	0.24	1.84	1.51	0.0	1.19	3.77	High
6	0.24	1.84	1.51	0.0	1.19	3.77	0.31	4.72	-0.06	High

For example, if today's precipitation is a relevant feature in predicting the temperature a week ahead, then it is likely that yesterday's and the day before yesterday's precipitations are also relevant features in predicting the temperature a week ahead. That seems to be the case because the precipitation from any particular day tends to stay in the ground and affect the temperature for many more days. Moreover, since the average precipitation of three consequtive days varies less than the precipitation on a single day, the former may be more reliable than the latter for the prediction of the temperature a week ahead. These intuitions lead us to believe that the alternative representation is advantageous for classifying temporal data. Although this seems a simple idea, it was not tried yet for decision trees or SVMs.

In general, the alternative representation allows one to go back i number of days and look ahead n days, as outlined in Figure 9.1. The original representation is a representation that looks back 0 days and looks ahead the same number n of days. Therefore, the transformation from a basic to an alternative representation, which we denote by $\Longrightarrow$, can be described as:

$$Texas\_Weather^{0,n} \Longrightarrow Texas\_Weather\_History^{i,n}$$

where for any relation the first superscript is the days of historical data and the second superscript is the days predicted in the future.

9.3 Experimental Evaluation of the Temporal Classification Method

9.3.1 Experimental Results with *TCEQ* Data

We experimentally compared the regular classification and the temporal classification methods. In some experiments both the regular and the temporal classification methods used SVMs and in some other experiments both methods used decision trees. In particular, we used the SVM implementation from the *LibSVM*[186] library and our implementation of the ID3-Interval algorithm described in Section 6.4.

The experiments used the Texas Commission on Environmental Quality (TCEQ) database (described in detail in Section 5.5), which recorded 40 meteorological measurements nearby Houston, Texas between 1998 and 2004. From the TCEQ database, we used only the data for Houston, Texas and the following fourty features and the class to predict. We conducted the experiments using the following procedure to predict the temperature T, where n is a training set size control parameter:

1. Normalize the dataset.
2. Randomly select 60 records from the dataset as a testing set.
3. Randomly select n percent of the remaining records as a training set.
4. Build a SVM, ID3, or ID3-Interval classification using the training data.
5. Test the accuracy of the classification on the testing set.

In step (1), the data was normalized by making for each feature the lowest value to be -1 and the highest value to be $+1$ and proportionally mapped into the interval $[-1, +1]$ all the other values. This normalization was a precaution against any bias by the classifications. The normalization also allowed a clearer comparison of the SVM weights of the features.

For testing the regular classifiers, we used the above procedure with $\text{TCEQ}^{0,2}$, which we obtained from the original $\text{TCEQ}^{0,0}$ database by shifting backwards by two days the T column values. For testing the temporal classifiers, we made the transformation

$$\text{TCEQ}^{0,2} \Longrightarrow \text{TCEQ}^{15,2}$$

as described in Section 9.2.

Figure 9.2 reports the average results of repeating the above procedure twelve times for n equal to %5, 15%, 25%, ..., 95% using the SVM algorithm. Similarly, Figure 9.3 reports the average results using ID3 decision trees. The experiments show that adding the historical data significantly (T-test $p = 0.01\%$ and $p = 0.04\%$ respectively) improves the temperature predictions using both the ID3 and the SVM algorithms. Moreover, the SVM algorithm performed better than the original ID3 algorithm, although the ID3-Interval algorithm (not shown) gave some improvements.

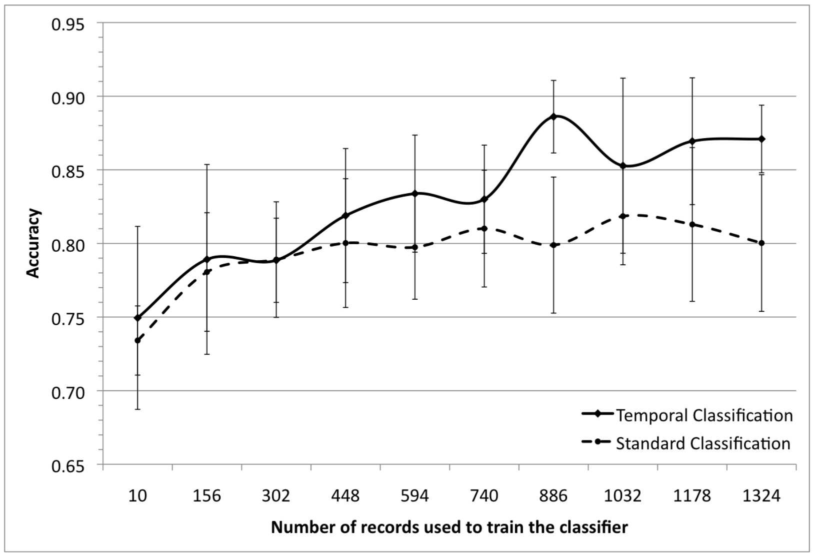


Figure 9.2: Comparison of regular and temporal classification using 40 features and SVMs. T-test $p = 0.04\%$. $\delta = 4.93\%$.

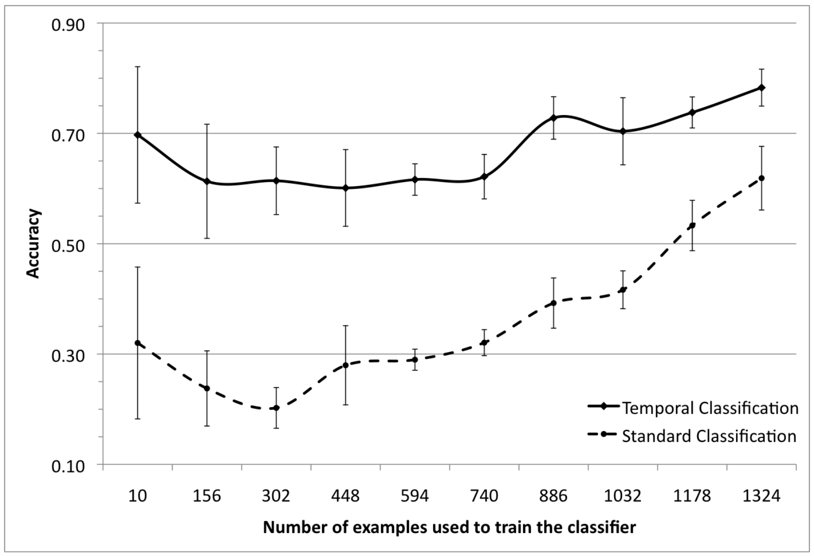


Figure 9.3: Comparison of regular and temporal classification using 40 features and ID3. T-test $p = 0.01\%$. $\delta = 31.10\%$.

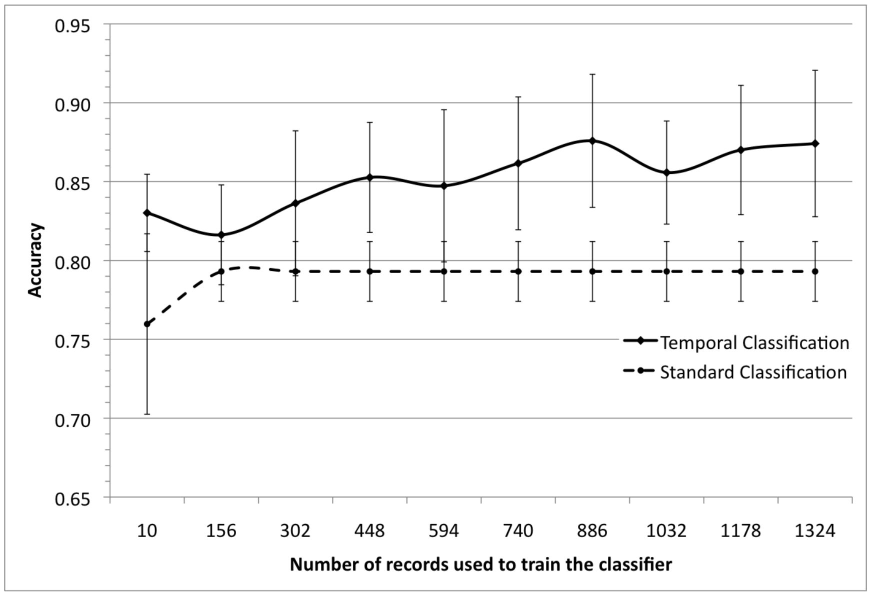


Figure 9.4: Comparison of regular and temporal classification using 3 features and SVMs. T-test $p = 0.01\%$. $\delta = 6.02\%$.

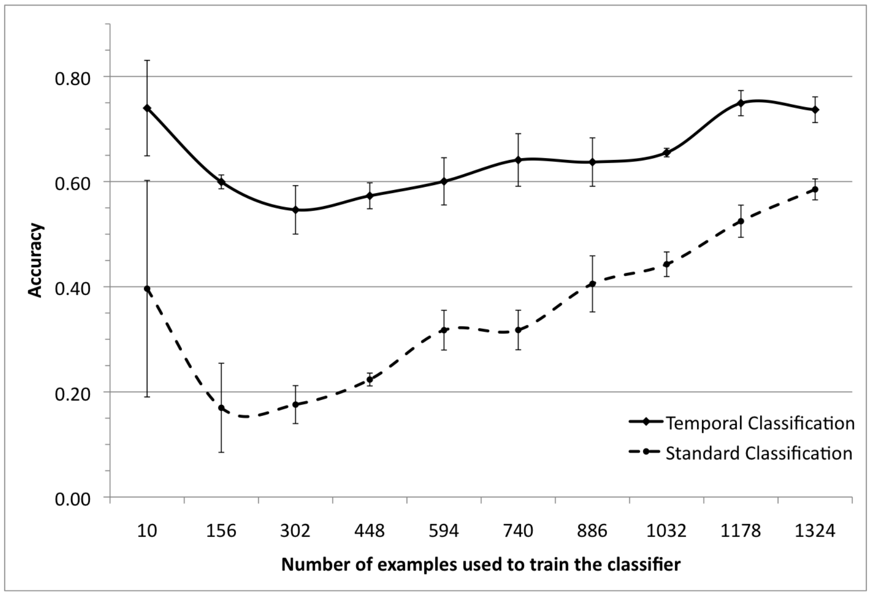


Figure 9.5: Comparison of regular and temporal classification using 3 features and ID3. T-test $p = 0.01\%$. $\delta = 29.17\%$.

9.3.2 Experimental Results with Reduced *TCEQ* Data

Databases with a large number of features often include many noisy variables that do not contribute to the classification. The TCEQ database also appears to include many noisy variables because the SVM placed small weights on them. Since we normalized the data, the relative magnitudes of the SVM weights correspond to the relative importance of the features. In particular, the following numerical features had the highest weights:

25. **asr**: average solar radiation

35. **wind-SN$_{700}$**: south-north wind speed component at 700 hPa

40. **precp:** precipitation

How accurate classification can be obtained using only these three selected features? These features have some interesting characteristics that make them better than other features. For example, wind-SN$_{700}$, the south-north wind speed component, is intuitively more important than wind-EW$_{700}$, the east-west wind speed component, in determining the temperature in Houston, Texas. In addition, the precipitation can stay in the ground for some time and affect the temperature a longer period than most of the other features. Hence our hypothesis was that these three features can already give an accurate classification.

To test this hypothesis, we conducted another set of experiments by applying the experimental procedure described in Section 9.3.1 to the reduced three-feature TCEQ database. The results of these experiments are shown in Figures 9.4 and 9.5. The accuracies of the classifiers based on only three features were surprisingly similar to the accuracies of the classifiers based on all fourty features. In this experiment the temporal classification was again more accurate ($p = 0.01\%$) than the traditional classification.

9.4 Comparison of the Temporal Classification Method and the IDW Interpolation

9.4.1 Experimental Results with Temporal *FLU* Data

Unlike interpolation methods, SVMs and decision trees are not designed to estimate a value in the future, but assign a label to records. The IDW interpolation and temporal classification methods thus cannot be compared directly. Hence, we derived a new variable **alert** by applying a fixed threshold to **ili** values. The binary **alert** variable determines whether the current flu activity is within normal levels given the ILI value or whether an alert should be raised. The threshold was arbitrarily choose after analyzing the distribution of ILI values. The interpolated value could be used to predict one of the two possible alert states and could therefore be compared against the temporal classification methods using the following procedure:

1. Normalize the dataset.
2. Randomly select 10% of records from the dataset as a testing set.
3. Build a IDW, SVM or ID3-Interval classifier using the remaining data.
4. Analyze the accuracy of the classifier using the testing set.

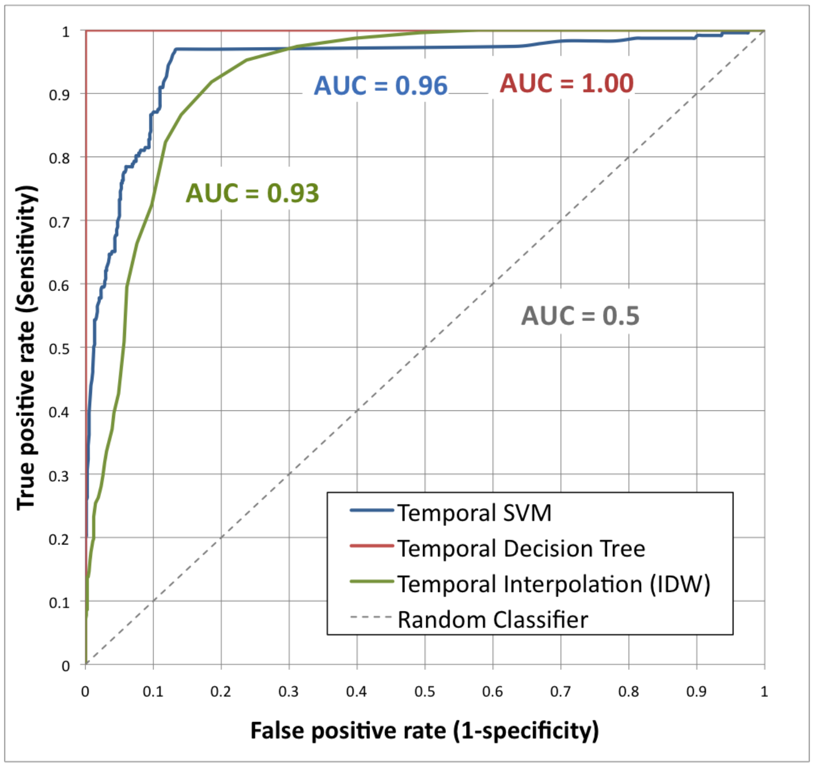


Figure 9.6: ROC analysis using temporal data of the IDW interpolation and the temporal classification method applied to SVMs and decision trees.

As mentioned earlier, the prediction of disease outbreaks is critical to raise an *alert* state so that appropriate measures to prevent the disease from spreading may be taken. On the other hand, the alert should be temporary and raised only when necessary because maintaining an alert state indefinitely would be meaningless. The analysis of the overall accuracies of the classifiers is therefore not satisfying and the classifier should be evaluate in terms of *sensitivity* and *specificity*. The sensitivity – or true positive rate – is the probability that an alert is predicted when there is actually an alert state. The specificity – or false positive rate – is the probability that the alert state will not be raised when it is not necessary.

In order to distinguish the two alert states, the predictions of the two methods were analyzed using a Receiver Operating Characteristic (ROC) curve analysis[193,194]. A ROC curve represents the sensitivity as a function of the specificity. The ROC curve analysis is specially useful when the distribution of the classes to predict is unbalanced as it is the case with the flu dataset. The overall performance of a specific classifier is defined as the area under its ROC curve (AUC). This simple value is representative of both the sensitivity and the specificity of the classifier. The AUC normally ranges from 0.5, which corresponds to a classifier that randomly classify items, to 1 which corresponds to a perfect classifier.

For testing the three methods, we used the data of the past six weeks to predict the alert state two weeks ahead of time. Hence, we made the following transformation:

$$\mathrm{FLU}^{0,0} \Longrightarrow \mathrm{FLU}^{6,2}$$

as described in Section 9.2.

The weight $w = 0.3$ of the IDW interpolation was empirically chosen to maximize the performance of the interpolation method. Figure 9.6 shows the ROC curves for the IDW interpolation algorithm and the temporal classification method applied to SVMs and decision trees.

9.4.2 Experimental Results with Spatio-Temporal *FLU* Data

The *Google Flu Trends* database does not provide spatial data directly. Instead, it gives the ILI measurement for each American state. We added a spatial dimension to the database by adding for each state the coordinates (latitude and longitude) of the capital city of the state being examined. Hence, the database include the following four variables:

1. **t**, the date the measurement was recorded,
2. **x**, the longitude of the state capital city,
3. **y**, the latitude of the state capital city,
4. **ili**, the number of *influenza-like illnesses* cases per 100,000 physician visits.

We compared the proposed temporal classification methods using both SVMs and decision trees against the IDW spatio-temporal interpolation method on $FLU^{6,2}$ (see Section 9.2) using the same procedure as in Section 9.4.1. The resulting ROC analysis is show in Figure 9.7.

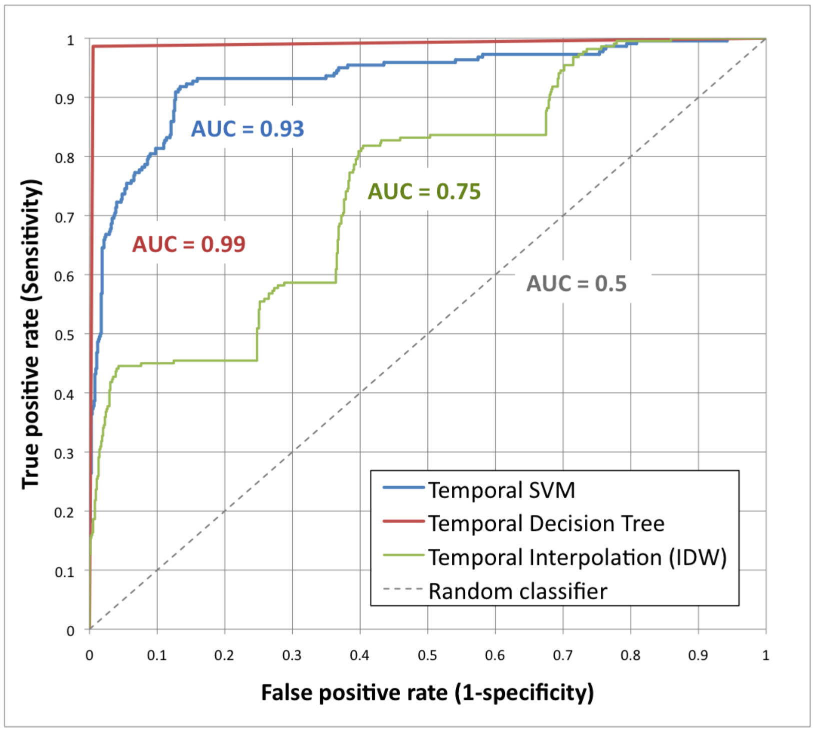


Figure 9.7: ROC analysis using spatio-temporal data of the IDW interpolation and the temporal classification method applied to SVMs and decision trees.

9.4.3 Discussion

The results of the experiments when using temporal data only (Figure 9.6) show that the temporal classification method applied to decision trees was able to perfectly predict the alert state. Overall, when applied to SVMs, the temporal classification also improves the IDW interpolation because AUC_{SVM} is greater than AUC_{IDW}. In the details, the ROC analysis reveals that the IDW interpolation has a higher sensitivity. In other words, this algorithm is more suitable when one needs to accurately predict when there is a risk of influenza pandemic. On the other hand, the temporal classification with SVM has a higher specificity, which means the temporal SVM classifier is more able to predict when the flu activity is within acceptable levels.

When considering the spatio-temporal data (Figure 9.7), the feature space has a higher dimension. In this case, we first notice that all three classifiers have a lower AUC. However, although the performance of the two temporal classifiers is only slightly lower, the performance (AUC) of IDW interpolation has significantly diminished from 0.92 to 0.75. This result confirms the intuition that decision trees and SVMs are more capable of handling highly-dimensional feature spaces. As a result, when considering the spatio-temporal dataset, our method performs significantly better than the IDW interpolation algorithm. This result may be explained by the local variability of the flu activity in each state. If the activity varies frequently, unlike both temporal classifiers, the interpolation algorithm, which is based on the few past values only, will not able to capture those variations, and accurately predict future values. At a federal scale, when using the temporal data only, the local variations are smoothened and the interpolation performs comparably to the temporal classification.

Chapter 10

Conclusion

"And now, the conclusion..."

Teal'c

The PROFESS database already proved to be useful in its current state. The modular functionality of PROFESS coupled with user friendly searching capabilities makes PROFESS particularly useful for asking a range of questions about the sequence, structure and functional relationship of orthologous proteins. However, its development and future extensions of this long-term project will continue much beyond the scope of this dissertation. First, data from the integrated databases are increasing on a daily basis. Hence, the first task to keep PROFESS useful will be to perform data updates on a regular basis, typically, several times per year.

In addition, the current core databases are of great value. However, they represent only a fraction of the data made available to the community. Hence, we plan to integrate new databases as the need will arise. In particular, users feedback will help us to prioritize the next databases to be integrated with our system. New features will be added as well. For example, the sketcher *MarvinSketch*[*] developed by ChemAxon will be integrated soon. It will enable users to query PROFESS by drawing the chemical structure of ligands instead of typing its formula or some identification number. The integration of new similarity functions such as the *Comparison of protein active site structure*[53] is also planned.

[*]Available at `http://www.chemaxon.com/product/msketch.html`

Our work on phylum-dependent structure divergence illustrates the inherent value in solving structures for functionally identical proteins from multiple organisms. A major challenge in creating our COG-to-PDB dataset was the fundamental requirement to have structures from at least two Firmicutes organisms and two Proteobacteria organisms. Only 48 (1%) of the 4,876 COGs meet this restrictive condition. The limited number of multiple homologous structures has partly occurred because structural biology efforts are focused on obtaining single representative structures for each functional class or protein fold[195] and understandably biased toward therapeutically relevant proteins[196]. If we are to achieve a more accurate understanding of the relationship between the evolution of protein fold, protein sequence, and the organisms in which they function, the fields of bioinformatics and structural biology must expand their focus to include efforts to obtain a more diverse set of homologous protein structures.

Our experiments using support vector machines and decision trees on *heart disease diagnosis* and *primary biliary cirrhosis* data show that our *classification integration* method is more accurate than existing *data integration* methods when there are many missing values in the data. The *reclassification* problem also can be solved using constraint databases without requiring access to the raw data. Our experiments on the TCEQ database show two major results: (1) Significant accuracy improvements are obtained by using histories, and (2) No accuracy improvement is obtained by using more than three features. The experiments with the FLU database also proved that the proposed temporal classification using historical can also greatly improves existing spatio-temporal interpolation methods such as the *Inverse Distance Weighted* algorithm.

Beside these particular cases, our general method is also appropriate for many other application areas and may yield similar accuracy improvements. An interesting question is whether these results also hold for other databases. There are some other remaining questions. For example, would non-linear temporal classifiers also be better than regular non-linear classifiers? Experiments with other data sets and use non-linear classifiers in addition to SVMs and decision trees may also be be interesting.

References

1. Stein, L. D. Integrating biological databases. *Nat Rev Genet* **4**, 337–45 (2003). 1, 46

2. Goodman, N. A plethora of protein data, a shortage of solutions. *Genome Technol.* **22**, 82–83 (2002). 1

3. Navarro, J. D., Niranjan, V., Peri, S., Jonnalagadda, C. K. & Pandey, A. From biological databases to platforms for biomedical discovery. *Trends Biotechnol* **21**, 263–268 (2003). 1, 45

4. Lewitter, F. Text-based database searching. *Trends Guide to Bioinformatics* 3–5 (1998). 1

5. Wang, J., Li, Z., Cai, C. & Chen, Y. Assessment of approximate string matching in a biomedical text retrieval problem. *Computers in Biology and Medicine* **35**, 717–724 (2005). 1

6. Hull, R., Jenkins, A. & Zweerink, M. Text-based information systems for drug discovery. *Current Opinion in Drug Discovery & Development* **2(3)**, 186–196 (1999). 1

7. Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalov, I. N. & Bourne, P. E. The Protein Data Bank. *Nucleic Acids Res* **28**, 235–42 (2000). 2, 25, 48, 50

8. Triplet, T., Shortridge, M., Griep, M., Powers, R. & Revesz, P. 11th International Congress on Amino Acids, Peptides and Proteins. 95 (Vienna, Austria, 2009). 2

9. Triplet, T., Shortridge, M., Griep, M., Stark, J. L., Day, M., Powers, R. & Revesz, P. PROFESS: A PROtein Function, Evolution, Structure and Sequence database. *Database (submitted)* (2009). 2

10. Shortridge, M., Triplet, T., Revesz, P., Griep, M. & Powers, R. Bacterial Protein Structures Reveal Phylum Dependent Divergence. *PLoS ONE (submitted)* (2009). 3

11. Quinlan, J. Induction of decision trees. *Machine Learning* **1**, 81–106 (1986). 3, 38, 99

12. Vapnik, V. *The Nature of Statistical Learning Theory* (Springer-Verlag, 1995). 3, 34, 35, 99

13. Revesz, P. & Triplet, T. Classification Integration and Reclassification using Constraint Databases. *Artificial Intelligence in Medicine (accepted with minor revisions)* (2009). 3

14. Kanellakis, P. C., Kuper, G. M. & Revesz, P. Constraint Query Languages. *Journal of Computer and System Sciences* **51**, 26–52 (1995). 3, 39

15. Kuper, G. M., Libkin, L. & Paredaens, J. *Constraint Databases* (Springer-Verlag, 2000). 3, 39, 112

16. Revesz, P. *Introduction to Constraint Databases* (Springer-Verlag, 2002). 3, 39, 112, 113, 115

17. Revesz, P. & Triplet, T. Reclassification of Linearly Classified Data Using Constraint Databases. In *12th East European Conference on Advances of Databases and Information Systems*, 231–245 (2008). 4

18. Triplet, T. & Revesz, P. Reclassification of Linear Classifiers. In *5th Midwest Database Research Symposium* (Chicago, USA, 2008). 4

19. Revesz, P. Z. & Triplet, T. Temporal Data Classification Using Linear Classifiers. In *ADBIS* (eds. Grundspenkis, J., Morzy, T. & Vossen, G.), vol. 5739 of *Lecture Notes in Computer Science*, 347–361 (Springer, 2009). 4

20. Revesz, P. & Triplet, T. Temporal Data Classification Using Linear Classifiers. *Information Systems (submitted)* (2009). 4

21. Chargaff, E., Zamenhof, S. & Green, C. Composition of human desoxypentose nucleic acid. *Nature* **165**, 756–7 (1950). 9

22. Wilkins, M. H. F., Gosling, R. G. & Seeds, W. E. Physical studies of nucleic acid. *Nature* **167**, 759–60 (1951). 9

23. Franklin, R. E. & Gosling, R. G. Molecular configuration in sodium thymonucleate. *Nature* **171**, 740–1 (1953). 9

24. Watson, J. D. & Crick, F. H. Molecular structure of nucleic acids; a structure for deoxyribose nucleic acid. *Nature* **171**, 737–8 (1953). 9, 10

25. Kryukov, G. V., Castellano, S., Novoselov, S. V., Lobanov, A. V., Zehtab, O., Guigó, R. & Gladyshev, V. N. Characterization of mammalian selenoproteomes. *Science* **300**, 1439–43 (2003). 10, 13, 32

26. Ashburner, M. *et al.* Gene ontology: tool for the unification of biology. *Nature Genetics* **25**, 25–29 (2000). 10

27. Crick, F. H. & Watson, J. D. Structure of small viruses. *Nature* **177**, 473–5 (1956). 11, 25

28. Crick, F. H., Barnett, L., Brenner, S. E. & Watts-Tobin, R. J. General nature of the genetic code for proteins. *Nature* **192**, 1227–32 (1961). 13

29. Walczak, R., Westhof, E., Carbon, P. & Krol, A. A novel RNA structural motif in the selenocysteine insertion element of eukaryotic selenoprotein mRNAs. *RNA* **2**, 367–79 (1996). 13

30. Atkins, J. F. & Gesteland, R. Biochemistry. The 22nd amino acid. *Science* **296**, 1409–10 (2002). 13

31. Zhang, Y., Baranov, P. V., Atkins, J. F. & Gladyshev, V. N. Pyrrolysine and selenocysteine use dissimilar decoding strategies. *J Biol Chem* **280**, 20740–51 (2005). 13

32. Lobanov, A. V., Kryukov, G. V., Hatfield, D. L. & Gladyshev, V. N. Is there a twenty third amino acid in the genetic code? *Trends Genet* **22**, 357–60 (2006). 13

33. Turanov, A. A., Lobanov, A. V., Fomenko, D. E., Morrison, H. G., Sogin, M. L., Klobutcher, L. A., Hatfield, D. L. & Gladyshev, V. N. Genetic code supports targeted insertion of two amino acids by one codon. *Science* **323**, 259–61 (2009). 13

34. Petsko, G. A. & Ringe, D. *Protein Structure and Function* (New Science Press, Ltd., Sunderland, 2008). 14, 24

35. Vauquelin, L.-N. & Robiquet, P. J. The discovery of a new plant principle in Asparagus sativus. *Ann Chim* **57**, 88–93 (1806). 14

36. Taylor, W. R. The classification of amino acid conservation. *J Theor Biol* **119**, 205–18 (1986). 15

37. Zvelebil, M. J., Barton, G. J., Taylor, W. R. & Sternberg, M. J. Prediction of protein secondary structure and active sites using the alignment of homologous sequences. *J Mol Biol* **195**, 957–61 (1987). 15

38. Parry-Smith, D. J. & Attwood, T. K. SOMAP: a novel interactive approach to multiple protein sequences alignment. *Comput Appl Biosci* **7**, 233–5 (1991). 15

39. Livingstone, C. D. & Barton, G. J. Protein sequence alignments: a strategy for the hierarchical analysis of residue conservation. *Comput Appl Biosci* **9**, 745–56 (1993). 15

40. IUPAC-IUB. Commission on Biochemical Nomenclature. Abbreviations and symbols for the description of the conformation of polypeptide chains. *J Mol Biol* **52**, 1–17 (1970). 19, 23, 25

41. Ramachandran, G. N., Venkatachalam, C. M. & Krimm, S. Stereochemical criteria for polypeptide and protein chain conformations. 3. Helical and hydrogen-bonded polypeptide chains. *Biophys J* **6**, 849–72 (1966). 20

42. Ramakrishnan, C. & Ramachandran, G. N. Stereochemical criteria for polypeptide and protein chain conformations. II. Allowed conformations for a pair of peptide units. *Biophys J* **5**, 909–33 (1965). 20

43. Pauling, L. & Corey, R. B. Configuration of polypeptide chains. *Nature* **168**, 550–1 (1951). 20

44. Pauling, L. & Corey, R. B. Atomic coordinates and structure factors for two helical configurations of polypeptide chains. *Proc Natl Acad Sci U S A* **37**, 235–40 (1951). 20

45. Pauling, L. & Corey, R. B. The structure of synthetic polypeptides. *Proc Natl Acad Sci U S A* **37**, 241–50 (1951). 20

46. Pauling, L. & Corey, R. B. The pleated sheet, a new layer configuration of polypeptide chains. *Proc Natl Acad Sci U S A* **37**, 251–6 (1951). 20

47. Pauling, L., Corey, R. B. & Branson, H. R. The structure of proteins; two hydrogen-bonded helical configurations of the polypeptide chain. *Proc Natl Acad Sci U S A* **37**, 205–11 (1951). 20

48. Pace, C. N. & Scholtz, J. M. A helix propensity scale based on experimental studies of peptides and proteins. *Biophys J* **75**, 422–7 (1998). 22

49. Pauling, L. & Corey, R. B. A Proposed Structure For The Nucleic Acids. *Proc Natl Acad Sci U S A* **39**, 84–97 (1953). 23

50. Andreeva, A., Howorth, D., Chandonia, J.-M., Brenner, S. E., Hubbard, T. J. P., Chothia, C. & Murzin, A. G. Data growth and its impact on the SCOP database: new developments. *Nucleic Acids Res* **36**, D419–25 (2008). 25, 48, 55

51. Cuff, A. L., Sillitoe, I., Lewis, T., Redfern, O. C., Garratt, R., Thornton, J. & Orengo, C. A. The CATH classification revisited–architectures reviewed and new ways to characterize structural divergence in superfamilies. *Nucleic Acids Res* **37**, D310–4 (2009). 25, 32, 48, 55

52. Powers, R., Mercier, K. A. & Copeland, J. C. The application of FAST-NMR for the identification of novel drug discovery targets. *Drug Discov Today* **13**, 172–179 (2008). 25

53. Powers, R., Copeland, J. C., Germer, K., Mercier, K. A., Ramanathan, V. & Revesz, P. Comparison of protein active site structures for functional annotation of proteins and drug design. *Proteins* **65**, 124–35 (2006). 25, 62, 135

54. Smith, T. F. & Waterman, M. S. Identification of common molecular subsequences. *J Mol Biol* **147**, 195–7 (1981). 25, 28

55. Altschul, S. F., Gish, W., Miller, W., Myers, E. W. & Lipman, D. J. Basic local alignment search tool. *J Mol Biol* **215**, 403–10 (1990). 26, 30

56. Dayhoff, M., Schwartz, R. & Orcutt, B. A model of evolutionary change in proteins. *Atlas of Protein Sequence and Structure* **5**, 45–352 (1978). 26

57. Henikoff, S. & Henikoff, J. G. Amino acid substitution matrices from protein blocks. *Proc Natl Acad Sci U S A* **89**, 10915–9 (1992). 27

58. Pietrokovski, S., Henikoff, J. G. & Henikoff, S. The Blocks database–a system for protein classification. *Nucleic Acids Res* **24**, 197–200 (1996). 27, 28

59. Styczynski, M. P., Jensen, K. L., Rigoutsos, I. & Stephanopoulos, G. BLOSUM62 miscalculations improve search performance. *Nat Biotechnol* **26**, 274–5 (2008). 28

60. Stirling, J. *Methodus differentialis, sive tractatus de summation et interpolation serierum infinitarium* (London, 1730). 28

61. Needleman, S. B. & Wunsch, C. D. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J Mol Biol* **48**, 443–53 (1970). 28

62. Gotoh, O. An improved algorithm for matching biological sequences. *J Mol Biol* **162**, 705–8 (1982). 28

63. Lathrop, R. H. The protein threading problem with sequence amino acid interaction preferences is NP-complete. *Protein Eng* **7**, 1059–68 (1994). 30

64. Wang, L. & Jiang, T. On the complexity of multiple sequence alignment. *J Comput Biol* **1**, 337–48 (1994). 30

65. Larkin, M., Blackshields, G., Brown, N., Chenna, R., McGettigan, P., McWilliam, H., Valentin, F., Wallace, I., Wilm, A., Lopez, R., Thompson, J., Gibson, T. & Higgins, D. Clustal W and Clustal X version 2.0. *Bioinformatics* **23**, 2947–2948 (2007). 30, 54

66. Thompson, J., Higgins, D. & Gibson, T. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic acids research* **22**, 4673–4680 (1994). 30

67. Wu, S. & Manber, U. Fast Text Searching Allowing Errors. *Commun. ACM* **35**, 83–91 (1992). 30

68. Chang, W. I. & Lawler, E. L. Approximate String Matching in Sublinear Expected Time. In *FOCS*, vol. I, 116–124 (IEEE, 1990). 30

69. Myers, E. W. A Sublinear Algorithm for Approximate Keyword Searching. *Algorithmica* **12**, 345–374 (1994). 30

70. Pearson, W. R. & Lipman, D. J. Improved tools for biological sequence comparison. *Proc Natl Acad Sci U S A* **85**, 2444–8 (1988). 30

71. Altschul, S. F. & Gish, W. Local alignment statistics. *Methods Enzymol* **266**, 460–80 (1996). 31

72. Gertz, E. M., Yu, Y.-K., Agarwala, R., Schäffer, A. A. & Altschul, S. F. Composition-based statistics and translated nucleotide searches: improving the TBLASTN module of BLAST. *BMC Biol* **4**, 41 (2006). 31

73. Altschul, S. F., Gertz, E. M., Agarwala, R., Schäffer, A. A. & Yu, Y.-K. PSI-BLAST pseudocounts and the minimum description length principle. *Nucleic Acids Res* **37**, 815–24 (2009). 31

74. Altschul, S. F., Madden, T. L., Schäffer, A. A., Zhang, J., Zhang, Z., Miller, W. & Lipman, D. J. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res* **25**, 3389–402 (1997). 31, 45, 50

75. Kolodny, R. & Linial, N. Approximate protein structural alignment in polynomial time. *Proc Natl Acad Sci U S A* **101**, 12201–6 (2004). 31

76. Zhu, B. Protein local structure alignment under the discrete Fréchet distance. *J Comput Biol* **14**, 1343–51 (2007). 31

77. Ye, Y. & Godzik, A. Multiple flexible structure alignment using partial order graphs. *Bioinformatics* **21**, 2362–2369 (2005). 31

78. Holm, L. & Sander, C. Mapping the protein universe. *Science* **273**, 595–603 (1996). 32

79. Holm, L., Kääriäinen, S., Rosenström, P. & Schenkel, A. Searching protein structure databases with DaliLite v.3. *Bioinformatics* **24**, 2780–1 (2008). 32

80. Shindyalov, I. N. & Bourne, P. E. Protein structure alignment by incremental combinatorial extension (CE) of the optimal path. *Protein Eng* **11**, 739–47 (1998). 32

81. Kedem, K., Chew, L. P. & Elber, R. Unit-vector RMS (URMS) as a tool to analyze molecular dynamics trajectories. *Proteins* **37**, 554–64 (1999). 32

82. Lupyan, D., Leo-Macias, A. & Ortiz, A. R. A new progressive-iterative algorithm for multiple structure alignment. *Bioinformatics* **21**, 3255–63 (2005). 32, 54, 74

83. Orengo, C. A. & Taylor, W. R. SSAP: sequential structure alignment program for protein structure comparison. *Methods Enzymol* **266**, 617–35 (1996). 32

84. Bendtsen, J. D., Nielsen, H., von Heijne, G. & Brunak, S. Improved prediction of signal peptides: SignalP 3.0. *J Mol Biol* **340**, 783–95 (2004). 32

85. Karplus, K., Barrett, C. & Hughey, R. Hidden Markov models for detecting remote protein homologies. *Bioinformatics* **14**, 846–56 (1998). 32

86. Eddy, S. R. & Durbin, R. RNA sequence analysis using covariance models. *Nucleic Acids Res* **22**, 2079–88 (1994). 32

87. Zhang, T. An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods. *AI Magazine* **22**, 103–104 (2001). 34

88. Weston, J., Mukherjee, S., Chapelle, O., Pontil, M., Poggio, T. & Vapnik, V. Feature Selection for SVMs. In *NIPS* (eds. Leen, T. K., Dietterich, T. G. & Tresp, V.), 668–674 (MIT Press, 2000). 34

89. Aizerman, M., Braverman, E. & Rozonoer, L. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control* **25**, 821–837 (1964). 35

90. Amari, S. & Wu, S. Improving support vector machine classifiers by modifying kernel functions. *Neural Network* **12**, 783–789 (1999). 36

91. Xiong, H., Zhang, Y. & Chen, X.-W. Data-Dependent Kernel Machines for Microarray Data Classification. *IEEE/ACM Trans. Comput. Biol. Bioinformatics* **4**, 583–595 (2007). 36

92. Liu, C. Gabor-Based Kernel PCA with Fractional Power Polynomial Models for Face Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **26**, 572–581 (2004). 36

93. Quinlan, J. Improved use of continuous attributes in C4.5. *Journal of Artificial Intelligence Research* **4**, 77–90 (1996). 38, 99

94. Breiman, L. Random Forests. *Machine Learning* **45**, 5–32 (2001). 38

95. Rigaux, P., Scholl, M. & Voisard, A. *Introduction to Spatial Databases: Applications to GIS*, 355–396 (Morgan Kaufmann, 2000). 39

96. Chomicki, J., Haesevoets, S., Kuijpers, B. & Revesz, P. Classes of Spatiotemporal Objects and their Closure Properties. *Annals of Mathematics and Artificial Intelligence* **39**, 431–461 (2003). 39

97. Güting, R. & Schneider, M. *Moving Objects Databases* (Morgan Kaufmann, 2005). 39

98. Anderson, S. & Revesz, P. Efficient MaxCount and threshold operators of moving objects. *GeoInformatica* **13** (2009). 39

99. Revesz, P. & Wu, S. Spatiotemporal Reasoning about Epidemiological Data. *Artificial Intelligence in Medicine* **38**, 157–70 (2006). 39, 43

100. Geist, I. A Framework for Data Mining and KDD. In *Proc. ACM Symposium on Applied Computing*, 508–13 (ACM Press, 2002). 39

101. Johnson, T., Lakshmanan, L. V. & Ng, R. T. The 3W Model and Algebra for Unified Data Mining. In *International Conference on Very Large Data Bases*, 21–32 (2000). 39

102. Codd, E. F. A Relational Model for Large Shared Data Banks. *Communications of the ACM* **13**, 377–87 (1970). 39, 45, 60

103. Abiteboul, S., Hull, R. & Vianu., V. *Foundations of Databases* (Addison-Wesley, 1995). 39

104. Ramakrishnan, R. *Database Management Systems* (McGraw-Hill, 1998). 39

105. Ullman, J. D. *Principles of Database and Knowledge-Base Systems* (Computer Science Press, 1989). 39

106. Brodsky, A., Segal, V., Chen, J. & Exarkhopoulo, P. The CCUBE Constraint Object-Oriented Database System. *Constraints* **2**, 245–77 (1997). 39

107. Grumbach, S., Rigaux, P. & Segoufin, L. The DEDALE System for Complex Spatial Queries. In *ACM SIGMOD International Conference on Management of Data*, 213–24 (1998). 39

108. Bishop, B., Fischer, F., Keller, U., Steinmetz, N., Fuchs, C. & Pressnig, M. Integrated Rule Inference System (2008). URL `http://www.iris-reasoner.org/`. 39, 93

109. Revesz, P., Chen, R., Kanjamala, P., Li, Y., Liu, Y. & Wang, Y. The MLPQ/GIS Constraint Database System. In *ACM SIGMOD International Conference on Management of Data* (2000). 39, 93

110. Legendre, A. *Nouvelles méthodes pour la détermination des orbites des comètes* (Courcier, Paris, 1805). 41

111. Shepard, D. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM national conference*, 517–524 (ACM, New York, NY, USA, 1968). 42

112. Li, L. & Revesz, P. Interpolation Methods for Spatiotemporal Geographic Data. *Computers, Environment, and Urban Systems* **28**, 201–27 (2004). 43

113. Li, L. & Revesz, P. A Comparison of Spatio-Temporal Interpolation Methods. In *Proc. Second International Conference on Geographic Information Science*, vol. 2478 of *Lecture Notes in Computer Science*, 145–160 (Springer-Verlag, 2002). 43

114. Dayhoff, M. O. *Atlas of protein sequence and structure* (National Biomedical Research Foundation, 1965). 44

115. Dayhoff, M. O. *Atlas of protein sequence and structure* (National Biomedical Research Foundation, 1973). 44

116. Wu, C. H., Yeh, L.-S. L., Huang, H., Arminski, L., Castro-Alvear, J., Chen, Y., Hu, Z., Kourtesis, P., Ledley, R. S., Suzek, B. E., Vinayaka, C. R., Zhang, J. & Barker, W. C. The Protein Information Resource. *Nucleic Acids Res* **31**, 345–7 (2003). 44

117. Babu, P. A., Udyama, J., Kumar, R. K., Boddepalli, R., Mangala, D. S. & Rao, G. N. DoD2007: 1082 molecular biology databases. *Bioinformation* **2**, 64–67 (2007). 45

118. Galperin, M. Y. & Cochrane, G. R. Nucleic Acids Research annual Database Issue and the NAR online Molecular Biology Database Collection in 2009. *Nucleic Acids Res* **37**, D1–4 (2009). 45

119. Stein, L. Creating a bioinformatics nation. *Nature* **417**, 119–120 (2002). URL `http://dx.doi.org/10.1038/417119a`. 45

120. Horn, F., Vriend, G. & Cohen, F. E. Collecting and harvesting biological data: the GPCRDB and NucleaRDB information systems. *Nucleic Acids Res* **29**, 346–349 (2001). 45

121. Finn, R. D., Tate, J., Mistry, J., Coggill, P. C., Sammut, S. J., Hotz, H.-R., Ceric, G., Forslund, K., Eddy, S. R., Sonnhammer, E. L. L. & Bateman, A. The Pfam protein families database. *Nucleic Acids Res* **36**, D281–8 (2008). 45, 48, 53

122. Webb, E. *Enzyme Nomenclature 1992: Recommendations of the Nomenclature Committee of the International Union of Biochemistry and Molecular Biology on the No (Enzyme Nomenclature)* (Academic Press, San Diego, 1992). 45, 48, 53, 68

123. Stevens, R., Goble, C., Baker, P. & Brass, A. A classification of tasks in bioinformatics. *Bioinformatics* **17**, 180–8 (2001). 45

124. Davidson, S. B., Overton, C. & Buneman, P. Challenges in integrating biological data sources. *J Comput Biol* **2**, 557–572 (1995). 45

125. Wong, L. Technologies for integrating biological data. *Brief Bioinform* **3**, 389–404 (2002). 45

126. Joyce, A. R. & Palsson, B. Ø. The model organism as a system: integrating 'omics' data sets. *Nat Rev Mol Cell Biol* **7**, 198–210 (2006). 46

127. Chen, Y. & Revesz, P. Querying Spatiotemporal XML Using DataFox. In *Proc. IEEE International Conference on Web Intelligence*, 301–9 (IEEE Press, 2003). 46

128. Pottinger, R. & Halevy, A. MiniCon: A scalable algorithm for answering queries using views. *The VLDB Journal* **10**, 182–198 (2001). 46, 58

129. Tatusov, R. L. *et al.* The COG database: an updated version includes eukaryotes. *BMC Bioinformatics* **4**, 41 (2003). 46, 48, 50, 68

130. Zhang, R. & Lin, Y. DEG 5.0, a database of essential genes in both prokaryotes and eukaryotes. *Nucleic Acids Res* **37**, D455–8 (2009). 48, 55

131. Salwinski, L., Miller, C. S., Smith, A. J., Pettit, F. K., Bowie, J. U. & Eisenberg, D. The Database of Interacting Proteins: 2004 update. *Nucleic Acids Res* **32**, D449–51 (2004). 48, 53

132. Consortium, T. G. O. The Gene Ontology (GO) project in 2006. *Nucleic Acids Research* **34**, D322–D326 (2006). 48, 53, 68

133. Benson, D. A., Karsch-Mizrachi, I., Lipman, D. J., Ostell, J. & Sayers, E. W. GenBank. *Nucleic Acids Res* **37**, D26–31 (2009). 48, 58

134. Kanehisa, M., Araki, M., Goto, S., Hattori, M., Hirakawa, M., Itoh, M., Katayama, T., Kawashima, S., Okuda, S., Tokimatsu, T. & Yamanishi, Y. KEGG for linking genomes to life and the environment. *Nucleic Acids Res* **36**, D480–4 (2008). 48, 51

135. Arifuzzaman, M. *et al.* Large-scale identification of protein-protein interaction of Escherichia coli K-12. *Genome Research* **16**, 686–691 (2006). 48, 53

136. UniProt Consortium. The Universal Protein Resource (UniProt) 2009. *Nucleic Acids Res* **37**, D169–74 (2009). 48

137. Schmidt, T. & Frishman, D. PROMPT: a protein mapping and comparison tool. *BMC Bioinformatics* **7**, 331 (2006). 50

138. Felsenstein, J. PHYLIP- Phylogeny Inference Package (Version 3.2). *Cladistics* **5**, 164–166 (1989). 54, 75

139. Halevy, A. Answering queries using views: A survey. *VLDB Journal* **10**, 270–294 (2001). 58

140. Codd, E. F. *The Relational Model for Database Management, Version 2* (Addison-Wesley, 1990). 60

141. Date, C., Darwen, H. & Lorentzos, N. *Temporal Data and the Relational Model* (Morgan Kaufmann, 2002). 60

142. Fagin, R. Normal Forms and Relational Database Operators. In *SIGMOD Conference* (ed. Bernstein, P. A.), 153–160 (ACM, 1979). 61

143. Date, C., Darwen, H. & Lorentzos, N. *Database in Depth: Relational Theory for Practitioners* (O'Reilly, 2005). 61

144. Levenshtein, V. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady* **10**, 707–710 (1966). 61

145. Damerau, F. A technique for computer detection and correction of spelling errors. *Commun. ACM* **7**, 171–176 (1964). 62

146. Do, C. B. & Katoh, K. Protein multiple sequence alignment. *Methods Mol Biol* **484**, 379–413 (2008). 66

147. Feng, J.-a. Improving pairwise sequence alignment between distantly related proteins. *Methods Mol Biol* **395**, 255–68 (2007). 66

148. Chang, G. S., Hong, Y., Ko, K. D., Bhardwaj, G., Holmes, E. C., Patterson, R. L. & van Rossum, D. B. Phylogenetic profiles reveal evolutionary relationships within the "twilight zone" of sequence similarity. *Proc Natl Acad Sci U S A* **105**, 13474–9 (2008). 66

149. Pál, C., Papp, B. & Lercher, M. J. An integrated view of protein evolution. *Nat Rev Genet* **7**, 337–48 (2006). 66

150. Rocha, E. P. C. The quest for the universals of protein evolution. *Trends in Genetics* **22**, 412–416 (2006). 66

151. Forouhar, F. *et al.* Functional insights from structural genomics. *J Struct Funct Genomics* **8**, 37–44 (2007). 66

152. Chothia, C. & Lesk, A. M. The relation between the divergence of sequence and structure in proteins. *The EMBO Journal* **5**, 823–826 (1986). 66, 80, 81

153. Rost, B. Twilight zone of protein sequence alignemnts. *Protein Engineering* **12**, 85–94 (1999). 66, 80

154. Sadreyev, R. I. & Grishin, N. V. Exploring dynamics of protein structure determination and homology-based prediction to estimate the number of superfamilies and folds. *BMC Structural Biology* **6**, No pp given (2006). 67

155. Illergård, K., Ardell, D. H. & Elofsson, A. Structure is three to ten times more conserved than sequence–a study of structural response in protein cores. *Proteins* **77**, 499–508 (2009). 67, 81

156. Kolodny, R., Petrey, D. & Honig, B. Protein structure comparison: implications for the nature of 'fold space', and structure and function prediction. *Curr Opin Struct Biol* **16**, 393–8 (2006). 67, 81

157. Panchenko, A. R., Wolf, Y. I., Panchenko, L. A. & Madej, T. Evolutionary plasticity of protein families: coupling between sequence and structure variation. *Proteins* **61**, 535–44 (2005). 67, 81

158. Williams, S. G. & Lovell, S. C. The effect of sequence evolution on protein structural divergence. *Mol Biol Evol* **26**, 1055–65 (2009). 67, 81

159. Holm, L. & Park, J. DaliLite workbench for protein structure comparision. *Bioinformatics* **16**, 566–567 (2000). 68

160. Bailey, S., Eliason, W. K. & Steitz, T. A. Structure of hexameric DnaB helicase and its complex with a domain of DnaG primase. *Science* **318**, 459–63 (2007). 74

161. Oakley, A. J., Loscha, K. V., Schaeffer, P. M., Liepinsh, E., Pintacuda, G., Wilce, M. C. J., Otting, G. & Dixon, N. E. Crystal and Solution Structure of the Helicase-Binding Domain of Escherichia coli Primase. *The Journal of Biological Chemistry* **280**, 11495–11504 (2005). 74

162. Su, X. C., Schaeffer, P. M., Loscha, K. V., Gan, P. H., Dixon, N. E. & Otting, G. Monomeric solution structure of the helicase-binding domain of Escherichia coli DnaG primase. *Febs J* **273**, 4997–5009 (2006). 74

163. Syson, K., Thirlway, J., Hounslow, A. M., Soultanas, P. & Waltho, J. P. Solution Structure of the Helicase-Interaction Domain of the Primase DnaG: A model for Helicase Activation. *Strcuture* **13**, 609–616 (2005). 74

164. Efron, B., Halloran, E. & Holmes, S. Bootstrap confidence levels for phylogenetic trees. *Proc Natl Acad Sci U S A* **93**, 13429–34 (1996). 74

165. Fitch, W. M. & Margoliash, E. Construction of phylogenetic trees. *Science* **155**, 279–84 (1967). 75

166. Garcia-Vallve, S., Romeu, A. & Palau, J. Horizontal gene transfer in bacterial and archaeal complete genomes. *Genome Res* **10**, 1719–25 (2000). 77

167. Konstantinidis, K. T. & Tiedje, J. M. Towards a Genome-Based Taxonomy for Prokaryotes. *Journal of Bacteriology* **187**, 6258–6264 (2005). 80

168. Konstantinidis, K. T. & Tiedje, J. M. Genomic insights that advance the species definition for prokaryotes. *Proc Natl Acad Sci U S A* **102**, 2567–72 (2005). 80

169. Rost, B. Enzyme Function Less Conserved than Anticipated. *Journal of Molecular Biology* **318**, 595–608 (2002). 80

170. Powers, R., Mirkovic, N., Goldsmith-Fischman, S., Acton, T. B., Chiang, Y., huang, Y. J., Ma, L., Rajan, P., Cort, J. R., Kennedy, M. A., Liu, J., Rost, B., Honig, B., Murray, D. & Montelione, G. T. Solution structure of Archaeglobus fulgidis peptidyl-tRNA hydrolase (Pth2) provides evidence for an extensive conserved family of Pth2 enzymes in archea, bacteria, and eukaryotes. *Protein Science* **14**, 2849–2861 (2005). 81

171. Yang, K., Eyobo, Y., Brand, L. A., Martynowski, D., Tomchick, D., Strauss, E. & Zhang, H. Crystal structure of a type III pantothenate kinase: insight into the mechanism of an essential coenzyme A biosynthetic enzyme universally distributed in bacteria. *Journal of Bacteriology* **188**, 5532–5540 (2006). 81

172. Kisselev, L. Polypeptide Release Factors in Prokaryotes and Eukaryotes. Same Function, Different Structure. *Structure (Cambridge, MA, United States)* **10**, 8–9 (2002). 81

173. Ibba, M., Morgan, S., Curnow, A. W., Pridmore, D. R., Vothknecht, U. C., Gardner, W., Lin, W., Woese, C. R. & Soll, D. A euryarchaeal lysyl-tRNA synthetase: resemblance to class I synthetases. *Science* **278**, 1119–22 (1997). 81

174. Hadley, C. & Jones, D. T. A systematic comparison of protein structure classifications: SCOP, CATH and FSSP. *Structure* **7**, 1099–112 (1999). 81

175. Pascual-García, A., Abia, D., Ortiz, A. R. & Bastolla, U. Cross-over between discrete and continuous protein structure space: insights into automatic classification and networks of protein structures. *PLoS Comput Biol* **5**, e1000331 (2009). 81

176. Lin, Y.-S., Hsu, W.-L., Hwang, J.-K. & Li, W.-H. Proportion of solvent-exposed amino acids in a protein and rate of protein evolution. *Molecular Biology and Evolution* **24**, 1005–1011 (2007). 81

177. Chirpich, T. P. Rates of protein evolution. Function of amino acid composition. *Science (Washington, DC, United States)* **188**, 1022–3 (1975). 81

178. Lesk, A. M. & Chothia, C. How different amino acid sequences determine similar protein structures: the structure and evolutionary dynamics of the globins. *Journal of Molecular Biology* **136**, 225–70 (1980). 81

179. Robertson, A. D. & Murphy, K. P. Protein Structure and the Energetics of Protein Stability. *Chem Rev* **97**, 1251–1268 (1997). 82

180. Sánchez, I. E., Tejero, J., Gómez-Moreno, C., Medina, M. & Serrano, L. Point mutations in protein globular domains: contributions from function, stability and misfolding. *J Mol Biol* **363**, 422–32 (2006). 82

181. Donoho, D. & Ramos, E. *The CRCARS dataset.* Exposition of Statistical Graphics Technology (American Statistical Association, Toronto, 1983). 84

182. Ginsberg, J., Mohebbi, M. H., Patel, R. S., Brammer, L., Smolinski, M. S. & Brilliant, L. Detecting influenza epidemics using search engine query data. *Nature* **457**, 1012–4 (2009). 84, 86

183. Asuncion, A. & Newman, D. UCI Machine Learning Repository (2009). URL `http://archive.ics.uci.edu/ml/`. 84, 87, 90

184. Fleming, T. R. & Harrington, D. P. *Counting Processes and Survival Analysis* (Wiley, New York, 1991). 84, 88

185. Kun, Z. & Wei, F. Forecasting skewed biased stochastic ozone days: analyses, solutions and beyond. *Knowledge and Information Systems* **14**, 299–326 (2008). 84, 90

186. Chang, C. C. & Lin, C. J. LIBSVM: A library for support vector machines (2001). URL `http://www.csie.ntu.edu.tw/~cjlin/libsvm`. 94, 104, 126

187. Halevy, A. Y., Ashish, N., Bitton, D., Carey, M. J., Draper, D., Pollock, J., Rosenthal, A. & Sikka, V. Enterprise information integration: Successes, challenges and controversies. In *ACM SIGMOD International Conference on Management of Data*, 778–787 (2005). 99, 101

188. Lenzerini, M. Data Integration: A Theoretical Perspective. In *ACM Symposium on Principles of Database Systems*, 233–246 (2002). 99, 101

189. Student (W. Gosset). On the probable error of the mean. *Biometrika* **6**, 1–25 (1908). 104

190. Seidel, J., Imbery, F., Dostal, P., Sudhaus, D. & Bürger, K. Potential of historical meteorological and hydrological data for the reconstruction of historical flood events – the example of the 1882 flood in southwest Germany. *Natural Hazards and Earth System Sciences* **9**, 175–183 (2009). 123

191. Qin, Y. & Obradovic, Z. Efficient Learning from Massive Spatial-Temporal Data Through Selective Support Vector Propagation. In *17th European Conference on Artificial Intelligence*, 526–530 (2006). 123

192. Tseng, V. S. & Lee, C.-H. Effective temporal data classification by integrating sequential pattern mining and probabilistic induction. *Expert Systems with Applications* **36**, 9524–9532 (2009). 123, 124

193. Metz, C. Basic principles of ROC analysis. *Seminars in nuclear medicine* **8**, 283–298 (1978). 132

194. Zweig, M. & Campbell, G. Receiver-operating characteristic (ROC) plots: a fundamental evaluation tool in clinical medicine. *Clinical Chemistry* **39**, 561–577 (1993). 132

195. Chandonia, J.-M. & Brenner, S. E. Implications of structural genomics target selection strategies: Pfam5000, whole genome, and random approaches. *Proteins Structure, Function, and Bioinformatics* **58**, 166–179 (2004). 136

196. Mestres, J. Representativity of target families in the Protein Data Bank: impact for family-directed structure-based drug discovery. *Drug Discov Today* **10**, 1629–37 (2005). 136