Support Vector Learning and Rule Induction for Knowledge Discovery in Biological Data

by

Tho Hoan Pham

submitted to
Japan Advanced Institute of Science and Technology
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

Supervisor: Associate Professor Kenji Satou

School of Knowledge Science
Japan Advanced Institute of Science and Technology

June 24, 2005

Abstract

In order to uncover the nature of life, it is important to reveal the relationships among sequences, structures, interactions, and functions of biomolecules such as DNAs, RNAs and proteins. Among these molecules, proteins constitute most of a cell's dry mass. They are not only the building blocks from which cells are built. Each protein consists of a precise sequence of amino acids that allows it to fold up into a particular three-dimensional shape, or conformation. This three-dimensional structure allows the protein to interact with other biomolecules (such as DNAs, RNAs, and other proteins) to perform specific functions. The ability to bind to other molecules enables proteins to act as catalysts, signal receptors, regulators, motors, tiny pumps, etc. Our research focuses on two fundamental issues of structures and interactions of proteins: (1) prediction and analysis of structure of proteins from their sequence, and (2) analysis of DNA-protein interactions.

Determining the three-dimensional fold of a protein is an extremely complex problem. Experimental approaches such as nuclear magnetic resonance (NMR) and X-ray crystallography are expensive and can take up to several months. As a result, there is a large gap between the number of known protein sequences and known three-dimensional protein structures. This gap has grown over the past decade (and is expected to keep growing) as a result of the various worldwide genome projects. Thus, computational methods which may give some indication of structure, interaction and/or function are becoming increasingly important.

We have developed a support vector machine (SVM)-based method to predict turn structures in a protein from its sequence of amino acids. Turns make the protein fold into a specific three-dimensional shape. They play an important role in globular proteins from structural, interactional and functional points of view. When compared with previous methods, our approach exhibits a superior performance. Moreover, our method can estimate the relevance of amino acids for the formation of turn structures depending on their position in a protein. This information is specially useful for defining template structures

when designing new molecules with certain desired characteristics (structure, interaction site or function, for instance).

With a specific three-dimensional structure, proteins can interact with other biomolecules to carry specific functions. In the second part of this thesis, we address issues concerning interactions between proteins and DNAs, which are important information to uncover gene regulatory mechanisms. In an organism, about 10% of all proteins (called transcription factors or regulators) have particular structure that allows them to bind to DNAs and their DNA-binding interactions make the set of downstream genes express. Both experimental and computational approaches have been proposed to establish mappings of DNA-binding locations of transcription factors, but while the former produces noisy results due to imperfect measuring methods, the latter often suffers from over-prediction problems. Also, interactions between transcription factors and DNA-binding sites are usually environment-dependent, with many regulators binding only under certain conditions. Even more, the presence of regulators at a promoter region indicates binding but not necessarily function: the regulator may act positively, negatively, or not act at all. Identifying true and functional interactions between transcription factors and genes under specific environment conditions is therefore an open and important problem in biology.

Thank to recent experimental advances such as microarray technique, we can collect expression data of a whole genome. This technique provides us an opportunity to combine DNA-protein interactions with expression profiles data in order to uncover complicated transcriptional regulatory mechanisms. We have developed a rule induction method that combines these two kinds of databases to discover both relevant transcription factors of a target gene from the set of potential ones, as well as the relationship between the expression behavior of a gene and its transcription factors. Our method can deal efficiently with noise present in both DNA-protein interactions and expression profiles. The results of our rule induction method are transcriptional regulatory rules, which describe the qualitative relationship between the expression of a target gene and its relevant regulators. Transcriptional regulatory rules reveal some regulatory circuits, which describe how a group of transcription factors regulates target genes. They also provide strong and comprehensive evidences of actual gene-regulator interactions, and of protein-protein interactions that could serve to identify transcriptional complexes.

Moreover, we introduced a new method to discover transcriptional regulatory patterns other than regulatory rules, also by combining DNA-protein interactions with expression profiles. The method finds expression patterns of a group of genes commonly bound by the same set of transcription factors (TFset). Our approach first clusters genes into modules based on a closed TFset lattice, which includes non-redundant combinations of transcription factors respective to a database of DNA-protein interactions; and then validates the expression profiles to confirm regulatory modules. Our method has been applied to yeast data for finding transcriptional regulatory modules (TRMs). The results agree with gene modules found by previous studies. Moreover, the obtained TRMs are more compact, concise and comprehensive to identify and interpret the transcriptional control of combinations of regulators.

Acknowledgments

The work presented in this dissertation has been demanding and challenging at times, but always exciting, instructive, and fun. Without help, support, and encouragement from several persons, I would never have been able to finish it.

First of all, I wish to express my sincere gratitude to my main advisor, Associate Professor Kenji Satou, for his constant encouragement, kind guidance and support during this work. His technical and editorial advice, as well as comments, were essential to the completion of this dissertation, and have taught me innumerable lessons and insights on the workings of academic research in general.

I would like to thank specially to my co-advisor, Professor Tu Bao Ho, for his helpful discussions, suggestions and support during my 3-year doctoral course.

I am grateful to my friends Jose Carlos Clemente and Duc Dung Nguyen for many interesting and good-spirited discussions related to this research. I also greatly appreciate Jose for his kind reviewing of almost all my publications.

My thanks go also to Professor Akihiko Konagaya, Associate Professor Xavier Defago and Associate Researcher Tomoyuki Yamamoto for their help at the beginning of my course.

I would like to express my deep gratitude to the Vietnamese government, and Ministry of Training and Education of Vietnam for the scholarship that covers all fees and my living cost for three years in Japan.

At last, to my father Tho Nam, my wife Thuy Hoa, my son Thai Duong, and other family members thanks for supporting me with your love and understanding.

This work is supported by Grant-in-Aid for Scientific Research on Priority Areas (C) "Genome Information Science" from the Ministry of Education, Culture, Sports, Science, and Technology of Japan; BIRD of Japan Science and Technology Agency (JST); and the COE project "Knowledge Creation from Data Mining" (JCP KS 1) from Japan Advanced Institute of Science and Technology.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Molecular Biology

Overview: from DNA to protein via RNA

The basic biological processes in all organisms are pretty much identical. In most living organisms, the genetic material (DNA: deoxyribonucleic acid) is replicated each time a cell divides, so that each daughter cell receives a copy of the DNA from its parent. The DNA is also passed on to progeny: for diploid organisms like humans, each progeny receives half a copy of each parent's DNA. The DNA contains the genes of an organism, which are used as templates for manufacturing RNA (ribonucleic acid). The RNA can then be used as instructions for manufacturing proteins.

Proteins

Proteins make up much of our bodies, playing a broad roles: some form the structural parts of our cells, while others can catalyze biochemical reactions. Proteins are polymer chains whose building blocks are any of the 20 different amino acids. Proteins have an orientation: one end of the chain contains an amino group, while the opposite end contains a carboxyl group. The structure of a protein is mainly determined by its amino acid sequence, although environmental conditions, association with other proteins, and chemical modifications also play an important role.

DNA

DNA is a polymer of nucleotides where each nucleotide can be one of four bases: adenine (A), guanine (G), thymine (T), and cytosine (C). Similar to proteins, DNA also has orientation, and its sequence is written from the 5' end to the 3' end (5' and 3' refer to the number of the carbon on the deoxyribose). The DNA in a cell usually consists of two strands of nucleotide chain, help together a double helix structure by hydrogen bonds between the bases. For the structure to be stable, the strands must be in opposite orientation and two strands must be complementary: T on a strand hydrogen bonds to A on the other strand, and G on a strand hydrogen bonds to C on the other strand. Hence, knowing the sequence of one strand allows one to infer the sequence of the other using the reverse complement (for example, the reverse complement of TGGAC is GTCCA). When the structure of DNA was first discovered by Watson and Crick in 1953, it suggested an obvious mechanism for DNA replication (which turned out to be true): one strand is used as a template to manufacture the other strand. In the cell, DNA is packaged into chromosomes with several special proteins. In humans, the genome (whole set of genes) is divided into 46 chromosomes: 22 pairs of homologous chromosomes plus two sex chromosomes (XX or XY). Pairs of homologous chromosomes are nearly identical to each other - they contain the same genes, but the sequence of the genes may be slightly different since one homolog comes from the mother and one comes from the father.

RNA, transcription, and translation

RNA is similar in structure to DNA: it is also found by the bases A, G, and C, but instead of T it uses the base U (uracil), which also hydrogen bonds with A. RNA is usually single-stranded. It is manufactured by using the DNA as a template. The DNA segment used to manufacture a strand of RNA is called a gene. This process is called transcription, since RNA is transcribed from DNA. In an organism, there are some special proteins (which are in turn products of certain genes) which have a transcriptional role, i.e., they bind to certain locations in DNA and make the process of transcription of the downstream DNA segments (genes) activated.

In eucaryote, the produced RNA is then transported out of the nucleus to the ribosome, where it is translated into specific proteins. The cell machinery uses the RNA sequence

to manufacture the sequence of amino acids of a protein. Starting with the first triplet AUG of the RNA (which encodes the amino acid methionine), the cell machinery reads triplets of bases that specify which amino acid to add to the growing chain of amino acids. Each triplet is called a codon, and the amino acid specified by each codon is called the genetic code. The genetic code is the same for almost all organisms. There are three special codons which do not encode amino acids, but function as stop codons - they tell the cell machinery to stop making the protein. The remaining 61 codons encode the 20 possible amino acids. Therefore the codons are redundant: more than one codon can encode the same amino acid. As a result, greatly different DNA sequences can encode the same protein. Note that since there are three possible reading frames to a sequence, if we add one or two bases right after the start codon AUG, the produced protein would be completely changed.

Proteins: sequence, structure, interaction, and function

Proteins constitute most of a cell's dry mass. Each protein consists of a precise sequence of amino acids that allows it to fold up into a particular three-dimensional shape, or conformation. This three-dimensional structure in turn allows protein to interact with other biomolecules (such as DNA, RNA, and protein) to do correctly their function. The ability to bind to other molecules enables proteins to act as catalysts, signal receptors, regulators, motors, tiny pumps, etc.

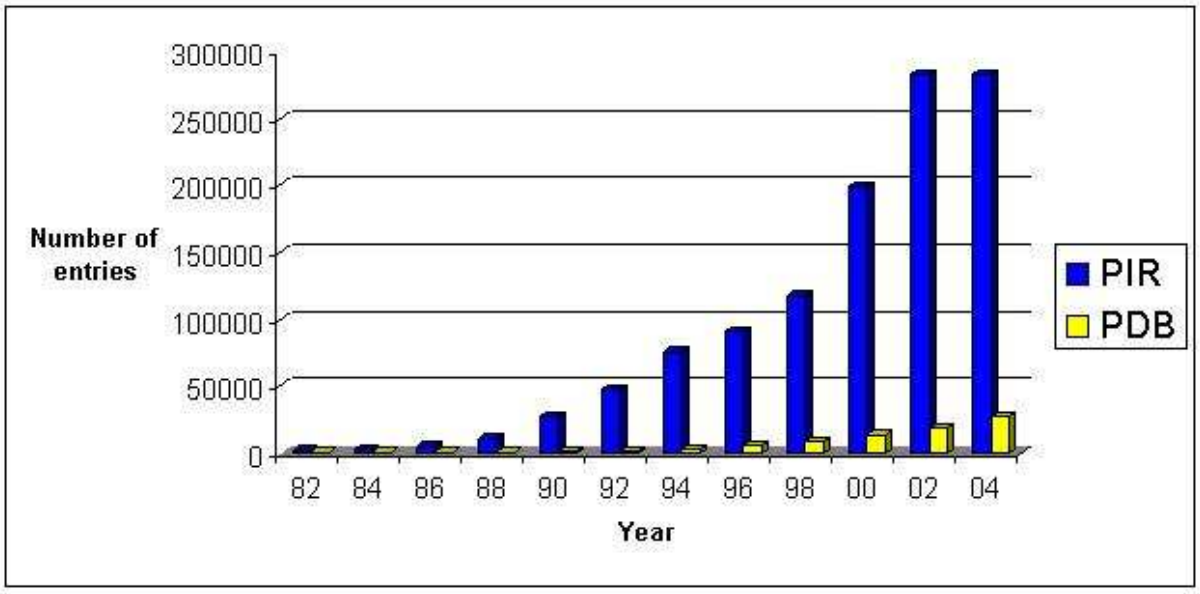
1.2 Protein structure

The structure or fold of a protein provides the key to understanding its interactional ability with other molecules and ultimately revealing its function in the body (i.e., enzymes, antibodies, transcription factors, etc.). Proteins may also be associated with particular human diseases, and thus, understanding protein structure may be used to better understand these diseases and to do rational drug design.

Unfortunately, determining the three-dimensional fold of a protein is difficult. Experimental approaches such as nuclear magnetic resonance (NMR) and X-ray crystallography are expensive and can take a long time (usually several months, even longer than a year

for some proteins). As a result, there is a large gap between the number of known protein sequences and that of known three-dimensional protein structures. This gap has grown over the past decade (and is expected to keep growing) because of the various worldwide genome projects (Fig. 1.1). Thus, computational methods which may give some indication of structure, interaction, and/or function are becoming increasingly important.

Figure 1.1: Growth of the number of protein sequences (PIR) and structures (PDB)



The structure of a protein is characterized not only by its amino acid sequence and full 3D structure, but also by other levels such as the secondary structure (local regular substructures: α-helix, β-sheet, and random coil); or the quaternary structure (the arrangement of several protein subunits in space) [1].

How can we try to predict the three-dimensional fold of a protein (either the exact or overall fold)? There are many approaches to this problem:

1. Model all the energy forces involved in protein folding, and try to find the structure with the lowest free energy. This is an extremely complex approach, both in terms of the modeling as well as in the searching of the vast conformational space.

2. Exploit high sequence similarity and use alignments. Two sequences that have just 25% sequence identity usually have the same overall fold. Alignments are probably the most widely used tool for understanding a protein's 3D structure; however, they are useful only when there are similar protein sequences for which structural

information is known.

3. Use the threading approach. This approach is based on the observation that many protein structures have similar folds, and assumes that there are just a limited number of distinct folds. For a protein sequence, the goal is to find the known protein structure which "best fits" it, according to some statistics-based potential function. This approach can not give structures of predicted proteins other than previously known folds.

Since predicting the 3D structure of a protein is difficult, many researchers have focused on trying to predict the secondary structure. That is, for each amino acid in a protein, can we predict whether the amino acid is in an α-helix, β-sheet, or random coil? Unfortunately, predicting the secondary structure of a protein is also a very difficult problem, especially random coil and β-sheet areas. Perhaps the secondary structure depends on the overall 3D structure of the fold. Most methods for predicting secondary structure are statistical or machine learning-based, and the overall three-state prediction accuracy is around 75%.

Another approach to structure prediction problem is recognizing structural motifs. Given a particular structural motif, how can we determine if it occurs in a given amino acid sequence, and if so, in what positions? The first part of this thesis will be devoted to this approach and will specially address *turn motifs*, which make up random coil areas in proteins. Turns make the protein fold into a specific three-dimensional shape. They play an important role in globular proteins from structural, interactional and functional points of view.

1.3 Molecular interactions

Proteins with appropriate structure can interact with other molecules to perform specific functions. Life is based on molecular interactions: underlying every biological process there is a multitude of proteins, nucleic acids, carbohydrates, hormones, lipids, and cofactors, binding to and modifying each other, forming complex frameworks and assemblies, and catalyzing reactions. Molecular interactions can be:

1. *protein-nucleic acid interactions:* proteins bind to DNA and RNA that mediate a number of processes, including regulation of gene expression, gene transcription, DNA replication, and mRNA intron splicing.

2. *protein-ligand interactions:* proteins bind to some target molecule or a set of target molecules, and perform some action: enzymes bind to substrate molecules and then catalyze chemical reaction that would otherwise occur too slowly to be biologically useful; some proteins involved in cellular signaling bind to a signal molecule and undergo a conformational change leading to further signaling or changes in cellular processes.

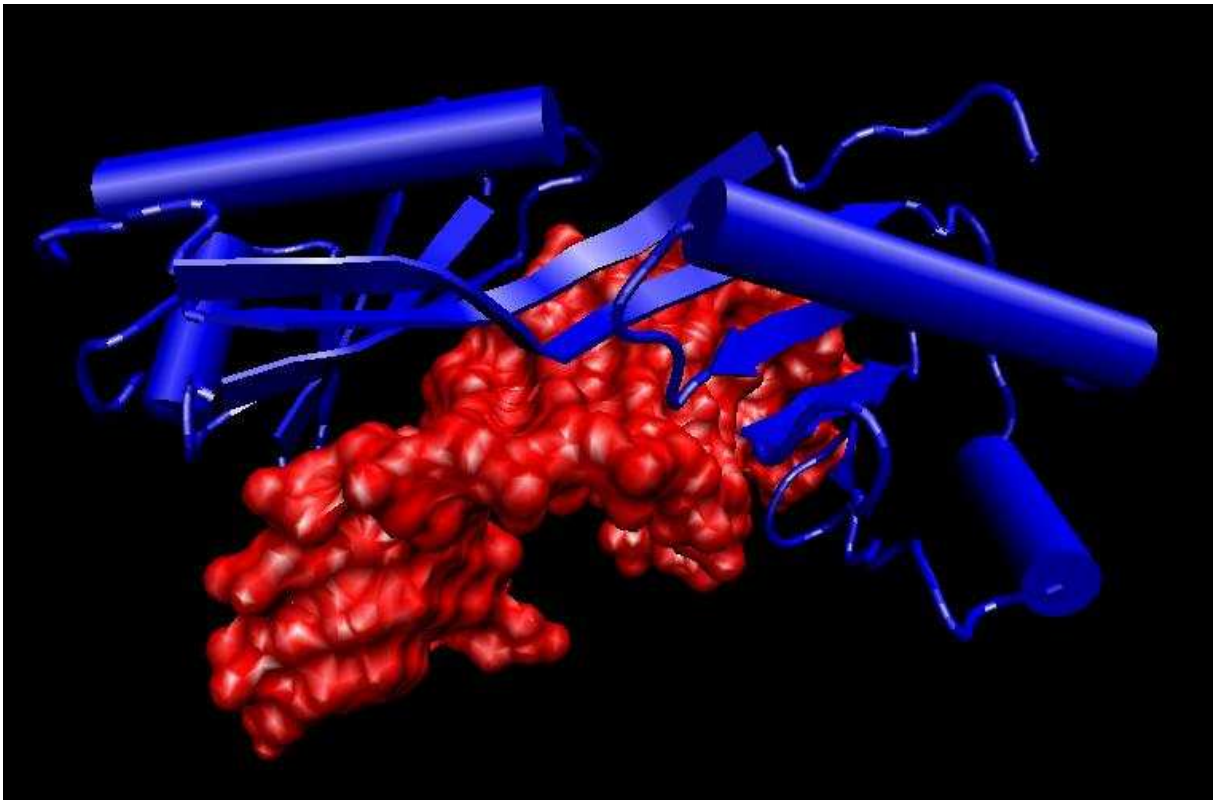
3. *protein-protein interactions:* many proteins function by forming active complexes with each other. The RNA polymerase II complex is an example of such an assembly. Protein-protein interactions are also involved in antibody-antigen binding, large scale organismal motion, and cell adhesion.

In the second part of this thesis, we focus on protein-nucleic acid interactions that regulate gene transcription. In an organism, there are about 10% of genes that can produce proteins having a transcriptional role. These special proteins are called transcription factors or regulators, and their DNA-binding interactions make the set of downstream genes express.

Fig. 1.2 shows an example of DNA-binding protein: TATA box-binding protein (TBP). TBP is responsible for initiating gene transcription on the chromosome. It specifically recognizes the promoter DNA sequence TATAAA. The promoter sequence lies about 25 base pairs upstream of a gene, and marks the location where an RNA polymerase complex must bind to transcribe that gene. Upon binding, TBP induces a kink in the DNA strands and forces open the minor groove of the DNA double helix, where most of its contacts with the DNA occur. Other transcription factors, as well as the RNA polymerase II complex, assemble around it. The TBP-DNA complex is slightly assymetrical, ensuring that transcription occurs on the correct strand of DNA.

Both experimental [5, 13, 39, 57] and computational [25, 10, 11, 4, 41] approaches have been proposed to establish mappings of DNA-binding locations of transcription factors. However, while location data obtained from experimental methods is noisy due to inherent

Figure 1.2: Schematic picture of TBP (blue, PDB structure ID 1tgh) bound to a promoter DNA sequence (red)



imperfections in the measuring methods [39], computational approaches often suffer from over-prediction problems due to the short length (less than 20 bases) of the sequence motifs bound by the transcription factors [7]. The second important problem is that interactions between transcription factors and DNA-binding sites are usually environment-dependent. Many regulators only bind to the promoter region of genes under specific environmental conditions. Even more, the presence of regulators at a promoter region indicates binding but not necessarily function: the regulator may act positively, negatively or not act at all. We aim to identify true and functional interactions between transcription factors and genes in specific environment conditions and to describe the relationship between them.

1.4 Objective

In the first part of this thesis, we will investigate the support vector learning to predict and analyze turn positions in proteins. There have been some previous approaches to specially address the problem of recognizing turns, such as statistical methods [31, 17, 29, 34] and

neural networks [9, 21, 20]. Support vector machine (SVM) is based on statistical learning theory and was developed by Vapnik [74]. In practice, SVM has a good performance and is easier to implement and train than neural networks. Two aspects of applying SVM to prediction and analysis of turn positions will be investigated. First, we will develop a SVM-based method that predicts turns in a protein from its sequence. Second, we will use support vectors-based classification functions to analyze the relevance of amino acids for the formation of turn positions depending on their position in a protein.

We will then proceed to analyze the data of interactions between proteins and DNAs (which are usually noisy and/or over-predicted) to uncover the gene regulatory mechanisms. We will use an inductive approach combining this data with expression profiles of genes to discover (1) transcription factors relevant for transcriptional regulation, and (2) transcriptional regulatory patterns. We assume that transcription factors regulating the expression of a gene must bind to its promoter, and the expression of the target gene must be consistent, in a specific way, to the expression behavior of these transcription factors. Three kinds of regulatory patterns will be studied with special interest in this work: *regulatory rules* that describe qualitatively relationships between the expression of target genes and their relevant regulators; *regulatory circuits* that describe clearly how a group of regulators controls taget genes; and *regulatory modules* that describe expression patterns of a group of genes commonly bound by a specific set of transcription factors.

1.5 Contributions

In this thesis, we develop inductive methods that can find global and/or local patterns and estimate the relevance of features for the formation of the patterns from biological data. We focus on two fundamental issues of structures and interactions of proteins: (1) prediction and analysis of turn structures of proteins from their sequence, and (2) analysis of DNA-protein interactions.

Prediction and analysis of turn structures in proteins

We investigated two aspects of applying support vector machine (SVM), a promising machine learning method for bioinformatics, for prediction and analysis of β-`turns` and

γ-turns.

1. First, we developed a SVM-based method to predict β-turns and γ-turns in a protein from its sequence. Our method has a superior performance when compared with previous approaches.

2. Second, we used SVMs with a linear kernel to estimate the support of amino acids for the formation of β-turns and γ-turns depending on their position in a protein. Our analysis results are more comprehensive and easier to use than the previous results in designing turns in proteins.

Analysis of DNA-protein interactions

We developed inductive methods that combine expression data with genomic location information to discover:

1. transcription factors relevant for expression of a gene from the set of noisy or over-predicted regulators binding to its promoter

2. transcriptional regulatory rules that qualitatively describe relationships between the expression of a target gene and its relevant transcription factors.

3. regulatory circuits that clearly describe how a group of regulators controls target genes.

4. expression patterns of a group of genes commonly bound by a specific set of regulators.

1.6 Thesis structure

This thesis includes 5 chapters and 2 appendices, and is structured as follows:

- Chapter 2 describes support vector machines for predicting β and γ-turns in a protein from its sequence. The idea of support vector machines is to use the optimal hyperplane in a feature space to separate turn positions from non-turn positions. In this chapter, we also use the optimal separating hyperplane to estimate the relevance of amino acids for the formation of turns depending on their position in protein.

- Chapter 3 presents a rule induction method which combines expression profiles data with a genomic locations data to identify DNA-protein interactions relevant for transcriptional regulation. The method also finds transcriptional regulatory rules and regulatory circuits.

- Chapter 4 introduces a new approach to cluster genes into modules based on database of DNA-protein interactions. Each module is a subset of genes commonly bound by a group of transcription factors (TFset). The method uses a closed TFset lattice, which is a concise representation of a DNA-TF interactions database. The method then uses expression profiles data to find an expression pattern in each module if it exists.

- Chapter 5 summarizes our work and provides future directions of study.

- Appendix A provides a detailed description of support vector learning, which is used in Chapter 2 of this thesis.

- Appendix B introduces some rule evaluation heuristics for knowledge discovery. We emphasize the appropriateness of the heuristics for description tasks in rule induction (similar to the problem in Chapter 3).

Chapter 2

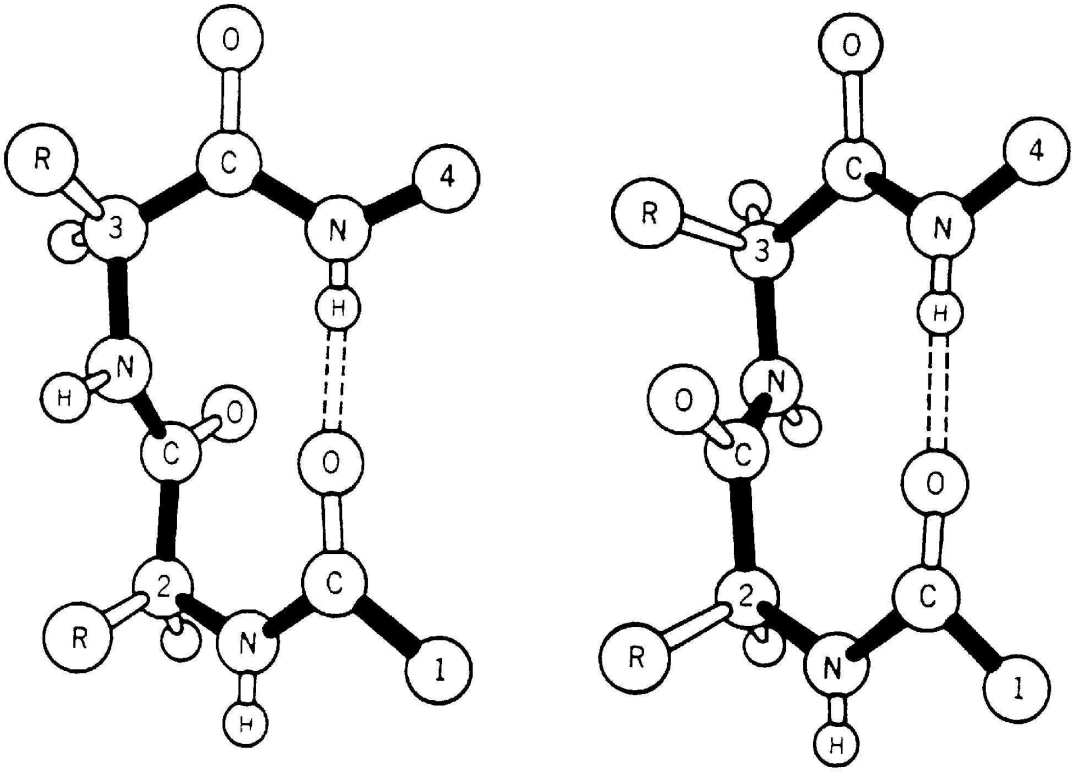
Prediction and analysis of turn structures in proteins

This chapter describes a support vector learning method to discriminate β- and γ-turn positions from non-turn positions in proteins. The method aims to find the globally optimal hyperplane (with the largest margin) to separate positive from negative points in a feature space, where each point corresponds to a turn position or non-turn position in proteins. The optimal separating hyperplane has been proved to have a good generalization ability, i.e., predictive ability for unseen data. It can also provide the relevance of amino acids for the formation of turn structures depending on their position in proteins.

2.1 Introduction

Tight turns [59] play an important role in protein folding and stability. Tight turns are classified as σ-turns, γ-turns, β-turns, α-turns, and π-turns. About 90% of tight turns in proteins constitute β-turns, and most of the remaining turns are γ-turns [49]. A β-turn is a four-residue reversal in a protein chain that is not in an α-helix, and the distance between $C\_\alpha(i)$ and $C\_\alpha(i+3)$ is less than $7\boldsymbol{r}A$ [71, 35] (Fig. 2.1). While β-turns may or may not be accompanied by the $NH(i+3) - CO(i)$ hydrogen bond connecting the main-chain atoms, a γ-turn consists of three consecutive residues at positions i, $i+1$, $i+2$ defined by the existence of a hydrogen bond between the $CO(i)$ group and $NH(i+2)$ group. β-turns and γ-turns provide useful information for defining template structures for the design of new molecules such as drugs, pesticides, and antigens [63].

Figure 2.1: Two examples of β-turns



There have been some attempts to predict and analyze β-turns and γ-turns. They can be divided into two categories: statistical and machine learning methods. The majority of statistical methods empirically employed the knowledge of amino acid preferences at individual positions in β-turns and γ-turns [31, 17, 29, 34]. Machine learning-based

methods have been recently developed for prediction of β-turns and γ-turns. They are BTPRED [9], BetaTPred2 [21] and GammaPred [20]. These methods all used neural networks with multiple sequence alignment, and significantly outperformed statistical approaches. However, the prediction and analysis results are still restricted due to the complexity of the problem and the unbalanced nature of the data (especially γ-turn data).

In this chapter, we introduce another machine learning approach using support vector machine (SVM) for both prediction and analysis of β-turns and γ-turns. SVM is based on statistical learning theory and was developed by Vapnik [74]. In practice, SVM has a good performance and is easier to implement and train than neural networks. SVM has also been successfully applied to some problems in bioinformatics, such as secondary structure prediction [22], microarray data analysis [27], protein-protein interactions [32], etc.

Two aspects of applying SVM to prediction and analysis of β-turns and γ-turns have been investigated. First, we developed a SVM-based method that predicts β-turns and γ-turns in a protein from its sequence. The prediction can be done with single sequence or multiple sequence alignment. The prediction results on a dataset of 426 non-homologous protein chains by sevenfold cross-validation, and on a dataset of 320 non-homologous protein chains by fivefold cross-validation, showed that our method performed well when compared to the other methods. Furthermore, the prediction results of our method were improved when combined with additional secondary structure information, which is in turn predicted by another high accuracy secondary structure prediction method PSIPRED [67]. Moreover, our method performed the prediction at the turn level, which makes our prediction results more comprehensive and easier to interpret.

Second, we analyzed β-turns and γ-*turns* by proposing the concept of *"the support of an amino acid position for the formation of* β-turns/γ-turns *under a linear SVM classification model"* (we will refer to it as *the support of an amino acid position*), which implies both the contribution and prevention of that amino acid position for the formation of β-turns/γ-turns. This information can be easily extracted from the "multivariable" classification model of a trained linear SVM. This model is more general than previously proposed models for prediction and analysis of β-turns and γ-turns such as

Site-Independent model [18], `1-4` and `2-3` Residue-Correlation model [34], and Sequence-Couple model [31].

The rest of this chapter is organized as follows. Section 2.2 describes our method in detail, where we present how to convert protein sequences into numerical vectors and how to assign positive and negative examples. A summary of learning support vectors for prediction and feature selection is also presented in this section. A more detailed description of support vector learning is presented in Appendix 1 of this thesis. Section 2.3 presents the main results of our method, together with experimental design. The final sections are devoted to discussion and conclusion.

2.2 Methods

2.2.1 Vector representations of a protein sequence

There are two basic ways to represent a protein sequence as a vector:

1. *Single sequence:* Each residue in the protein is represented by a 20-dimension vector of 0 and 1 coding for the corresponding amino acid at this residue. This binary representation can be extended by taking into account the general substitute abilities (scores) of amino acids, i.e., BLOSUM62. Therefore, each residue is represented by a 20-dimensional vector of the substitute scores of 20 amino acids for this residue.

2. *Multiple sequence alignment:* A protein sequence is firstly aligned with a non-redundant (NR) database (e.g., the version used in our work contains $1,109,366$ sequences) to find the family of sequences to which that protein belongs. The alignment can be expressed in a scoring matrix of probability estimates or scores [52]. Two kinds of such matrices are considered in our work: position-specific frequency matrices (PSFMs) and position-specific scoring matrices (PSSMs). A PSFM is a table that lists the frequencies of each amino acid in the alignment, while a PSSM gives the log-odds score for finding a particular matching amino acid in a target sequence (see the work of Gribskov et al. [2] and Altschul et al. [52, 12] for more details). The stand-alone version 2.2.6 of PSI-BLAST (ftp://ncbi.nlm.nih.gov/blast/executables/) has been used to generate PSFMs and PSSMs in this work with `E-value` threshold

of 0.001, three iterations and other parameters set to the respective default values.

Either of these representations, each protein sequence is represented as a bi-dimensional vector $L \times 20$, where L is the length of the sequence. In this work, all elements of bi-dimensional vectors are scaled into the interval $[-1, 1]$ by a simple linear transformation function. After having vector representations of proteins, we use a sliding window with a fixed length w along each protein to extract the dataset of vectors and input them into the machine learning system (i.e., support vector machine).

2.2.2 Assigning positive and negative examples

To predict and analyze β-turns and γ-turns, we use a sliding window along the protein representation to get examples in a vector format. How to define positive and negative examples is an important issue. There are two options (Fig. 2.2):

Figure 2.2: Assigning positive and negative windows at the residue level and turn level.

Sequence with 3 β-turns **nnnnnnTtttnnnTtTtttnnnnn**	
nnnnnnTtt → 0	nnnnnnTttt → 0
nnnnnTttt → 0	nnnnnTtttn → 0
nnnnTtttn → 1	nnnnTtttnn → 0
nnnTtttnn → 1	nnnTtttnnn → 1
nnTtttnnn → 1	nnTtttnnnT → 0
nTtttnnnT → 1	nTtttnnnTt → 0
TtttnnnTt → 0	TtttnnnTtT → 0
tttnnnTtT → 0	tttnnnTtTt → 0
ttnnnTtTt → 0	ttnnnTtTtt → 0
tnnnTtTtt → 1	tnnnTtTttt → 0
nnnTtTttt → 1	nnnTtTttn → 1
nnTtTtttn → 1	nnTtTtttnn → 0
nTtTtttnn → 1	nTtTtttnnn → 1
TtTttnnnn → 1	TtTtttnnnn → 0
tTtttnnnn → 1	tTtttnnnnn → 0
Ttttnnnnn → 0	
a) At residue level (Sliding window size=9)	*b) At turn level* (Sliding window size=10)

1. *Assigning positive and negative examples at a residue level:* A window will be considered as a positive or negative example if its central residue falls in a turn area or

not (Fig. 2.2a). That is, in the training phase, a window with the central residue falling in a turn area will be considered as a positive example, otherwise negative. In the testing phase, the prediction result of a window will conversely be assigned only for one central residue. In this way, the results of prediction may be invalid and unclear when the number of turn-predicted consecutive residues do not fit into a β-turn/γ-turn. For example, it is unrealistic to have three consecutive residues predicted as "ntn" or "tnt" (t for turn and n for non-turn). And it will be ambiguous to interpret the prediction result when more than five consecutive residues are predicted as β-turns/γ-turns, like "ttttttttt". How many β-turns or γ-turns are in this example? And where is the beginning of these turns?

2. *Assigning positive and negative examples at a turn level:* A window will be considered as a positive example if its four (or three with γ-turn) central residues form a β-turn/γ-turn, otherwise negative (Fig. 2.2b). In the training phase, a sliding window with four (or three with γ-turn) central residues forming a β-turn/γ-turn will be considered as a positive example, otherwise negative. In the testing phase, if a window is classified as a positive example, it means that its four (or three with γ-turn) central residues are predicted as a 4-residue-β-turn (3-residue-γ-turn). We used the signs "Tttt" for a 4-residue-β-turn and "Ttt" for a 3-residue-γ-turn, where T means the beginning of a turn and t means not-beginning of the turn. By using this approach in our work, we overcome the problems explained above.

2.2.3 Binary support vector machine

Support vector machine (SVM) is a learning technique based on statistical learning theory. The basic idea of applying SVM to binary pattern classification can be briefly stated as follows. First, map the input vectors into a feature space (often with a higher dimension), either linearly or non-linearly, which is relevant to the selection of the kernel function. Second, seek the optimal linear hyperplane (with the largest margin) to separate two classes within the feature space from the first step.

The implementation of SVM is as follows. Suppose that $(x_i, y_i), i = 1, .., l$ be a training dataset, where x_i is a vector and $y_i = 1$ or -1 is a class attribute. SVM training solves

the following problem:

$$\min_{w,b,\xi} \quad \frac{1}{2}w^T w + C\sum_{i=1}^{l}\xi_i$$

$$y_i(w^T\phi(x_i) + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0, i = 1, ..., l.$$

Its dual is a quadratic optimization problem:

$$\min_{\alpha} \quad \frac{1}{2}\alpha^T Q\alpha - e^T\alpha$$

$$0 \leq \alpha_i \leq C$$

$$y^T\alpha = 0$$

where e is the vector of all ones; $C > 0$ is a error penalty parameter, $y = \{y_i\}_{i=1,..,l}$, $Q_{ij} = y_i y_j K(x_i, x_j)$, $K(x_i, x_j) = \phi(x_i)^T\phi(x_j)$ is a kernel function; and $\phi(x_i)$ maps x_i into a higher (maybe infinite) dimensional space. So $K(x_i, x_j)$ is a symmetric positive definite function that reflects the similarity between the sample x_i and the sample x_j. In our research, we employed a linear function $K(x_i, x_j) = x_i.x_j$ and radial basis function (RBF) $K(x_i, x_j) = \exp(-\gamma(x_i - x_j)^2)$ as the kernel functions. The SVM classification function, after trained, has the following form:

$$f(x) = \sum_i \alpha_i y_i K(x, x_i) + b \tag{2.1}$$

where $\alpha = \{\alpha_i\}_{i=1,..,l}$ is the solution of the above dual problem and b is in the solution of the prime problem. Based on the Karush-Kuhn-Tucker theory, the solution of the prime problem and that of its dual satisfy the following equation:

$$\alpha_i\{y_i(w^T\phi(x_i) + b) - 1 + \xi_i\} = 0.$$

Therefore, if there is an i such that $\alpha_i \neq 0$, then $y_i(w^T\phi(x_i) + b) - 1 + \xi_i = 0$. In this case, x_i is called a "support vector".

SVM has a solid theoretical background, a good performance in practice, and a guaranteed global optimum. It can also handle a large dataset and is easier to implement and

train than a neural network. A more detailed description of SVM will be presented in Appendix A of this thesis, or can be found in the work of Vapnik [74] and Cristianini [26].

2.2.4 Support vector learning for discovering the support of features

Ranking informative (discriminant) attributes is of fundamental and practical interest in data mining and knowledge discovery. SVM has been successfully applied to this task [36, 27]. When SVM uses a linear kernel, it finds an optimal hyperplane that separates the positive from the negative class in the original space (not mapping into a higher dimensional space). This optimal hyperplane has then the following form (replacing K(x,y)=x.y in Eq. A.31):

$$f(X = (f_1, f_2, .., f_m)) = \sum_{i=1}^{m} w_i f_i + b \tag{2.2}$$

We can change the signs of the weights $w_i, i = 1, .., m$, and b in the above function such that if $f(X) > 0$ then X would be classified as a positive example, otherwise negative. It can be clearly seen that if w_i is positive, the attribute i would support the positive class; otherwise this attribute would support the negative class (or prevent the positive class); and the larger the absolute value of w_i, the stronger the support (or prevention) of attribute i. We therefore define the weight w_i as the *support of the feature* i.

2.2.5 BTSVM and GTSVM

We developed two support vector machine-based systems, BTSVM and GTSVM. BTSVM is used for predicting β-turns and analyzing the support of amino acids for the formation of β-turns, while GTSVM performs the same tasks for γ-turns. The settings of BTSVM and GTSVM for each task are presented in Table 2.1.

Table 2.1: Settings of BTSVM and GTSVM

	Task	Parameters
BTSVM	prediction	RBF kernel, PSSM, sliding_window_length=12
BTSVM_LIN	analysis	Linear kernel, PSFM, sliding_window_length=8
GTSVM	prediction	RBF kernel, PSSM, sliding_window_length=5
GTSVM_LIN	analysis	Linear kernel, PSFM, sliding_window_length=5

2.3 Experiments

2.3.1 Datasets

We used two datasets described in the work of Guruprasad and Rajkumar [49]. The first one (dataset B) consists of 426 non-homologous protein chains, while the second one (dataset G) consists of 320 non-homologous protein chains. These datasets have been used by Kaur and Raghava for assessing the performance of β-`turn` and γ-`turn` prediction methods [19, 20, 21]. In each dataset, there are no two protein chains having more than 25% sequence identity. The structure of these proteins is determined by X-ray crystallography at resolutions higher than $2.0\boldsymbol{r}A$. Each chain in the datasets contains at least one β-`turn` or γ-`turn`. The program PROMOTIF [30] has been used to assign β-`turns` and γ-`turns` in these proteins. The datasets are available at `http://genic.jaist.ac.jp/proteins`. The number of positive and negative examples at the residue level and turn level in the datasets of β-turns (B) and γ-turns (G) are reported in Table 2.2.

Table 2.2: The number of positive and negative examples of dataset B and G

Dataset	Level	#positive examples	#negative examples
Dataset B	residue	23555	72358
(426 proteins)	turn	7185	88728
Dataset G	residue	2669	79566
(320 proteins)	turn	904	81331

2.3.2 Performance measures

We use four criteria described in the work of Shepherd et al. [9]: (1) Q_{total} (prediction accuracy), the percentage of correctly predicted residues, (2) Matthew's Correlation

Coefficient (MCC), which accounts for both over- and under-prediction, (3) Q_{pred}, the percentage of correct prediction of turn residues (or probability of correct prediction), and (4) Q_{obs}, the percentage of observed turn residues that are correctly predicted (or percent coverage). These measures can be calculated using the following equations:

$$Q_{total} = (\frac{p+n}{t}) \times 100 \qquad Q_{pred} = (\frac{p}{p+o}) \times 100$$

$$MCC = \frac{pn - ou}{\sqrt{(p+o)(p+u)(n+o)(n+u)}} \qquad Q_{obs} = (\frac{p}{p+u}) \times 100$$

where p and n are the number of correctly predicted turn and non-turn residues, respectively; o and u are the number of incorrectly predicted turn and non-turn residues, and $t = p + n + o + u$ is the total of residues.

Following the work by Kaur and Raghava [19], in addition to the four criteria mentioned above, we used a threshold independent measure, AUC (area under the curve), for the comparison. A ROC curve is obtained by plotting all *sensitivity* values (true-positive fraction) on the y-axis against their equivalent $(1 - specificity)$ values (false-positive fraction) for all available thresholds on the x-axis, where $Sensitivity(Sn)$ and $specificity(Sp)$ are defined as:

$$Sn = \frac{p}{p+u} \qquad Sp = \frac{n}{n+o}$$

The AUC is taken as an important index because it provides a single measure of overall accuracy that is not dependent on a particular threshold [64]. Here we used trapezoidal intergration [61] to calculate the AUC of ROC curves produced by our prediction method.

2.3.3 K-fold cross-validation

To compare our method with other approaches, we employed the K-fold cross-validation described in the work of Kaur and Raghava [19, 20] ($K = 7$ and 5 for β-turns and γ-turns, respectively). The dataset is randomly divided into K subsets, each containing equal number of proteins. Each set is an unbalanced set that retains the naturally occurring proportion of turns and non-turns. $K - 1$ subsets are grouped into the training set. The last subset is for the testing set. This process is done K times to test the prediction result for each testing set. The final prediction results have been averaged over K testing sets.

2.3.4 Prediction of β and γ-turns

Table 2.3 shows the performance of BTSVM and 5 other methods on 426 non-homologous protein chains by sevenfold cross-validation; Table 2.4 shows the performance of some methods on 320 non-homologous protein chains by fivefold cross-validation. As it can be seen, BTSVM achieves a MCC score up to 0.43 when using PSSM and 0.45 when using additional secondary structure information, which is in turn predicted by PSIPRED; GTSVM has MCC of 0.11 when using PSSM and 0.13 when using additional predicted secondary structure information.

Table 2.3: Results of β-turn/non-β-turn prediction of some methods.

		Q_{total}	Q_{pred}	Q_{obs}	MCC	AUC
Chou-Fasman	Sin. seq.	74.9	46.1	16.9	0.16	
		(69.3)	(36.9)	(35.3)	(0.16)	
	Sin. seq. & sec. struct.	74.3	47.7	54.3	0.34	
		(75.3)	(49.6)	(47.5)	(0.32)	
Thornton	Sin. seq.	74.5	44.0	16.7	0.15	
		(70.1)	(36.7)	(30.5)	(0.14)	
	Sin. seq. & sec. struct.	75.2	49.3	44.9	0.31	
		(75.2)	(49.3)	(44.9)	(0.31)	
1-4 & 2-3 correlation model	Sin. seq.	63.2	35.3	60.4	0.21	
		(71.1)	(40.8)	(40.3)	(0.21)	
	Sin. sec. & seq. struct.	73.4	46.2	51.5	0.31	
		(74.8)	(48.0)	(39.8)	(0.28)	
Sequence couple model	Sin. seq.	50.6	31.7	88.4	0.23	
		(72.7)	(43.9)	(41.0)	(0.25)	
	Sin. seq. & sec. struct.	72.2	45.0	60.0	0.33	
		(75.4)	(49.6)	(40.0)	(0.28)	
BTPRED	Sin. seq.	71.6	44.1	57.3	0.31	
	Mul. seq.	**73.5**	**47.2**	**64.3**	**0.37**	**0.72**
	Mul. seq. & sec. struct.	75.5	49.8	72.3	0.43	0.77
BTSVM	Sin. seq.	74.2	47.6	49.2	0.31	
	Mul. seq.	78.4	55.9	58.6	0.43	**0.81**
		(**73.4**)	(**47.5**)	(**75.4**)	(**0.43**)	
	Mul. seq. & sec. struct.	79.8	59.2	58.0	0.45	0.82
		(76.0)	(50.9)	(72.0)	(0.45)	
BTSVM_LIN	Mul. seq. (PSFM)	73.1	46.0	55.0	0.32	

Note: the results of Chou-Fasman, Thornton, 1-4 & 2-3 correlation model and sequence couple model at original and new (in brackets) threshold values are from (Kaur, 2002) [19]. The results of BTPRED are from (Kaur, 2003) [21]. The results of BTSVM are sevenfold cross-validation accuracies obtained in the same way. BTSVM_LIN is used for analysis of β-turns.

Table 2.4: Results of γ-turn/non-γ-turn prediction of some methods.

		Q_{total}	Q_{pred}	Q_{obs}	MCC	AUC
Sequence couple model	Sin. seq.	66.3	2.8	50.1	0.05	
	Sin. seq. & sec. struct.	57.8	5.9	43.2	0.08	
GOR	Sin. seq.	62.1	4.7	54.4	0.06	
	Sin. seq. & sec. struct.	75.5	6.1	45.5	0.09	
WEKA (logistic regression)	Sin. seq.	61.7	4.7	56.2	0.06	
	Mul. seq.	62.7	5.5	63.9	0.10	
	Mul. seq. & sec. struct.	62.6	5.6	65.1	0.12	
WEKA (naive Bayes)	Sin. seq.	66.5	4.8	49.5	0.06	
	Mul. seq.	59.0	5.1	65.3	0.09	
	Mul. seq. & sec. struct.	57.4	5.0	65.4	0.11	
WEKA (J48 classifier)	Sin. seq.	89.6	4.3	10.4	0.02	
	Mul. seq.	92.5	5.0	7.2	0.02	
	Mul. seq. & sec. struct.	92.6	5.0	7.2	0.03	
SNNS	Sin. seq.	56.1	4.3	59.4	0.06	
	Mul. seq.	**76.6**	**5.1**	**58.6**	**0.12**	**0.69**
	Mul. seq. & sec. struct.	74.0	6.3	83.2	0.17	0.73
GTSVM	Sin. seq.	61.6	4.8	57.9	0.07	
	Mul. seq.	78.7 (**53.0**)	6.9 (**5.1**)	44.5 (**75.9**)	0.11 (**0.10**)	**0.70**
	Mul. seq. & sec. struct.	79.9 (67.4)	7.7 (6.3)	47.5 (64.7)	0.13 (0.12)	0.72
GTSVM_LIN	Mul. seq. (PSFM)	64.7	5.4	59.3	0.09	

Note: the results of sequence couple model, GOR, SNNS, WEKA are from (Kaur, 2003) [20]. The results of GTSVM are fivefold cross-validation accuracies obtained in the same way. GTSVM_LIN is used for analysis of γ-turns.

For the comparison, we set a new decision threshold for turn and non-turn classes such that Q_{pred} of our method is (nearly) equal to that of the best methods so far (BTPRED for β-turns and SNNS for γ-turns). The accuracy of our method at the new threshold are given in brackets in Table 2.3 and Table 2.4. As can be seen, for predicting β-turns, our method BTSVM has the best performance when compared to other single methods on the criteria Q_{pred}, Q_{obs} and MCC, while Q_{total} is still high enough. For predicting γ-turns, although our method GTSVM gives $MCC = 0.10, Q_{total} = 53.0$ that are lower than those of SNNS, our $Q_{obs} = 75.9$ is significantly higher.

We also calculated the threshold independent measure AUC for our prediction method by the trapezoidal method, which systematically underestimates the AUC [61]. The AUC of BTSVM and GTSVM (using PSSMs) are 0.81 and 0.70 respectively (see Table 2.3 and

Table 2.4), which are all greater than AUC of previous methods reported in [20, 21] (the AUC of BTPRED and SNNS using PSSMs are 0.72 and 0.69 respectively).

As in the work of Kaur and Raghava [20, 21], we tried to use additional secondary structure information, which is directly predicted by the PSIPRED method [67] without re-training it in the training dataset (which might be unfair for the comparison because PSIPRED might use a larger training dataset). Each protein sequence is then represented as a bi-dimensional vector $L \times 23$, where L is the length of its sequence, and each position in the protein is encoded by a group of 23 inputs, 20 units encoding for the amino acid at that position and the remaining three units being the probabilities of three states (helix, strand, and coil) provided in the output of the PSIPRED prediction. The performance of BTSVM is improved to $Q_{total} = 76.0, Q_{pred} = 50.9, Q_{obs} = 72.0, MCC = 0.45$ and $AUC = 0.82$ (Table 2.3) and that of GTSVM is improved to $Q_{total} = 67.4, Q_{pred} = 6.3, Q_{obs} = 64.7, MCC = 0.12$ and $AUC = 0.72$ (Table 2.4). As can be seen, BTSVM is still better than other methods, but GTSVM is worse than SNNS.

2.3.5 Support of amino acids for the formation of β and γ-turns

We used BTSVM_LIN and GTSVM_LIN with linear kernels and PSFMs (see Section 2.2.4 and Table 2.1 to estimate the support of amino acids at individual positions in the protein sequence (or, more briefly, the support of amino acid positions) for the formation of β-`turns` and γ-`turns`. In other words, we tried to find the w_i's in a linear SVM classification function (i.e., Eq. 2.2). In this task, first we used PSFMs for BTSVM_LIN and GTSVM_LIN since PSFMs emphasize clearly the occurrence of amino acids at an individual position in a protein sequence. While PSSMs (log-odds values), in addition to the information of occurence of amino acids, take account a general substitution matrix (i.e., BLOSUM62) and other information, they might be not as good as PSFMs in this task. We also tried to use single sequence for this task and found that the ranking of weights (w_i) is almost similar to the ranking of them generated by using PSFMs, although their values are different. Here we suppose that using PSFMs is more accurate because it gave a better performance (see Table 2.3 and Table 2.4). We chose the sliding window length of 8 for β-turns and 5 for γ-turns, because after having tried various experiments we found that these lengths make BTSVM_LIN and GTSM_LIN have the best performance.

After setting the parameters described above, we trained the BTSVM_LIN on the whole β-turn dataset B and GTSVM_LIN on the whole γ-turn dataset G to build the linear classification functions (Eq. 2.2) for turn/non-turn. From these classification functions, we extracted the supports (w_i's) of amino acid positions (see Section 2.2.4). Table 2.5 shows the supports of amino acids for the formation of β-turns depending on their position in the sliding window of length 8, and Table 2.6 shows the supports for the formation of γ-turns under the window of length 5.

In general, the support of an amino acid for the formation of β-turns/γ-turns varies from position to position in the window. We have marked in boldface positions where certain amino acids have a strong support, and underlined positions where they have a strong prevention.

Table 2.5: The supports of amino acid positions for the formation of β-turns

Amino acid	Position 1	2	3 (i)	4 (i+1)	5 (i+2)	6 (i+3)	7	8
Ala (A)	-0.346	-0.539	-1.047	0.223	-0.622	0.011	-0.435	-0.462
Arg (R)	-0.088	0.201	-0.788	0.349	0.275	0.271	0.377	-0.209
Asn (N)	-0.416	-0.164	0.122	0.400	**2.712**	0.267	**0.516**	-0.104
Asp (D)	-0.325	0.358	**0.589**	**0.690**	**1.542**	0.339	0.188	-0.367
Cys (C)	-0.131	0.138	-0.138	-0.472	0.067	0.286	0.069	0.098
Gln (Q)	-0.090	-0.227	-1.140	-0.083	0.106	**0.594**	0.122	-0.496
Glu (E)	-0.082	-0.286	-1.242	**0.798**	0.028	-0.400	0.285	-0.257
Gly (G)	-0.162	-0.044	-0.432	0.219	**2.207**	**1.061**	0.358	-0.278
His (H)	0.001	0.152	-0.412	0.250	**0.641**	0.499	0.382	0.186
Ile (I)	-0.094	-0.328	-1.250	-0.279	-0.489	-0.702	-0.202	-0.161
Leu (L)	-0.391	-0.311	-0.877	-0.299	-0.259	-0.242	0.002	-0.151
Lys (K)	-0.045	-0.221	-1.021	**0.944**	0.098	0.446	**0.741**	-0.211
Met (M)	-0.239	-0.312	-0.738	-0.279	-0.003	0.204	-0.009	-0.323
Phe (F)	-0.236	-0.150	-0.648	-0.409	0.368	-0.052	-0.048	-0.078
Pro (P)	0.077	-0.215	0.204	**1.982**	-0.254	0.263	**1.234**	0.227
Ser (S)	-0.206	-0.124	0.156	0.372	0.333	0.240	0.332	-0.282
Thr (T)	-0.214	-0.070	-0.427	-0.137	0.258	**0.502**	**0.724**	-0.052
Trp (W)	-0.263	-0.036	-0.801	-0.086	-0.044	-0.138	0.120	0.045
Tyr (Y)	0.177	0.089	-0.511	-0.164	0.230	-0.059	0.159	-0.125
Val (V)	0.034	0.044	-0.911	-0.326	-0.542	0.019	0.011	0.217

Note: amino acid positions with positive supports will contribute to the formation of β-turns, others will prevent the formation of β-turns. The larger the absolute value of the support, the stronger the contribution (or prevention if negative). Amino acid positions with the strongest supports (more than 0.50) are printed in boldface. Those with the lowest supports (less than -0.50) are underlined.

Table 2.6: The supports of amino acid positions for the formation of γ-turns

Amino acid	Position 1	2(i)	3(i+1)	4(i+2)	5
Ala	0.120	-0.960	-0.451	-1.221	-0.197
Arg	0.566	0.049	-1.067	0.581	0.287
Asn	0.693	0.280	**2.508**	0.162	0.879
Asp	0.733	-0.317	**2.040**	0.125	0.878
Cys	0.814	0.127	-0.206	0.324	-0.005
Gln	0.356	-0.473	-0.383	-1.040	-0.237
Glu	0.935	0.012	-1.276	-1.517	0.225
Gly	**1.479**	**1.080**	-0.691	-0.468	0.707
His	0.597	0.503	0.108	0.154	-0.130
Ile	0.609	0.250	-1.024	-0.559	-1.102
Leu	0.478	-0.066	-0.222	-0.800	-0.605
Lys	0.927	-0.518	-0.569	-0.403	-0.223
Met	0.874	0.326	**1.380**	0.133	-0.813
Phe	0.601	-0.182	-0.664	-0.388	-0.130
Pro	**1.413**	**1.197**	**1.929**	-1.024	**1.295**
Ser	**1.196**	-0.079	-1.278	0.751	0.605
Thr	0.631	-0.011	-2.074	0.598	0.024
Trp	0.694	-0.316	0.250	0.117	0.595
Tyr	0.402	0.411	-0.509	-0.216	-0.148
Val	0.154	-0.572	-1.751	0.332	0.478

Note: amino acid positions with positive supports will contribute to the formation of γ-turns, others will prevent the formation of γ-turns. The larger the absolute value of the support, the stronger the contribution (or prevention if negative). Amino acid positions with the strongest supports (more than 1.00) are printed in boldface. Those with the lowest supports (less than -1.00) are underlined.

There are some amino acids, of course at different positions, strongly supporting both the formation of β-turns and γ-turns. For example, Glycine (Gly) supports the β-turn formation at positions $i+2$ and $i+3$. It also supports the γ-turn formation at positions i and $i-1$. Especially, amino acid Asparagine (Asn) at position $i+2$ has the strongest support for the formation of β-turns; and it also has the strongest support for the formation of γ-turns when it occurs at position $i+1$. There are some amino acids, on the other hand, preventing both the formation of β-turns and γ-turns: Alanine (Ala), Isoleucine (Ile), etc. There are also some amino acids that, while their occurrence almost does not impact the β-turn formation (or γ-turn formation), their occurrence at specific positions strongly supports or prevents the formation of the other type of turns. For example, Serine (Ser) almost does not influence the β-turn formation, but it strongly

supports γ-turn formation at position $i-1$ and strongly prevents at position $i+1$.

2.4 Discussions

Prediction of β-turns and γ-turns

Our method gave clear prediction results and showed high performance. The reasons for these may be the following:

1. As explained in the work of Kaur [21, 20], our method, like BTPRED, BetaTPred2 and GammaPred, incorporates the evolutionary information of proteins by using multiple sequence alignment. The evolutionary information has been proved to significantly improve most structure prediction methods.

2. Like BTPRED, BetaTPred2 and GammaPred, our method can improve the prediction accuracy by using additional secondary structure, which is in turn predicted by a secondary structure prediction method with high accuracy, i.e., PSIPRED.

3. In our method, the prediction is performed at the turn level (see Section 2.2.2). This is different from previous work (PTPRED, BetaTPred2, and GammaPred), which performed the prediction at the residue level. Therefore, all β-turns/γ-turns predicted by our method, containing at least four residues with a β-turn and three residues with a γ-turn, are valid and clearer. In consequence, there is no need to go through a filtering process to exclude unrealistic β-turns/γ-turns.

4. Our method used SVM, which has many advantages over neural networks. For example, it always gives the global optimal solution with a particular kernel, it is easy to control the capacity, etc. [26, 74].

Supports of amino acid positions for the formation of β-turns and γ-turns

We introduced the term "support of an amino acid position to the formation of β-turns and γ-turns under the SVM classification model" that emphasizes the discriminative features. Our analysis results agree closely with those from previous statistical methods.

That is, amino acid positions with stronger positive supports for the formation of turns are often those with the higher amino acid positional potentials (preferences) for turns as previously reported in the work of Guruprasad and Rajkumar [49], and Chou [63]. Conversely, amino acid positions with stronger negative supports are often those with lower amino acid positional potentials (preferences).

However, there are at least four differences between our approach and others. First, our analysis and prediction are based on the "multivariable" classification model of SVM, which is more general than previous models, such as Site-Independent model [18], `1-4` and `2-3` Residue-Correlation model [34], and Sequence-Couple model [31]. Therefore, the supports of amino acid positions are not considered independently, but are mutually taken by a combinational linear. This explains why the order of amino acid positions sorted by their supports is different from the order when they are sorted by their potentials (or preferences).

Second, our method performed at the turn level by a window wider than the length of the β-`turn`/γ-`turn` itself. Some amino acids, although may not be in a turn region, have significant supports (or preventions) to the β-`turn` or γ-`turn` formation of the residues preceding or following them. This may explain why some previous statistical methods had low prediction performance, since they performed their prediction only under a window of size 4 with β-`turns` and 3 with γ-`turns`.

Third, as explained above, our approach emphasizes the discriminative features due to the discriminative character of SVM model.

Fourth, the analysis results of our approach are more comprehensive and therefore easier to use than previous studies. Amino acid positions with positive supports will contribute to the formation of turns; otherwise they will prevent it. The stronger the support, the stronger the effect of the amino acid position on the formation of turns.

2.5 Summary

In this chapter, we introduced a support vector learning method to the problem of predicting β- and γ-turn positions in a protein sequence. We first described how to convert turn and non-turn positions into numerical vectors, to which we can apply a machine learning method. Support vector learning can find a globally optimal hyperplane separating turn

vectors from non-turn vectors in a training data. The optimal separating hyperplane was then used to (1) predict unseen data and (2) estimate the support of amino acid for the formation of turn structures. If the training data has a distribution that is similar to the distribution of unseen data, the optimal separating hyperplane will have a good generalization ability (the ability of correct prediction for unseen data). In the next chapter, we will deal with the problem where the known data set follows a distribution that does not hold for unseen data.

Chapter 3

Discovery of transcriptional regulatory rules

This chapter describes a rule induction method to find protein-DNA interactions relevant for transcriptional regulation. The method uses a weighted covering strategy (an extension of separate-and-conquer), together with a rule evaluation heuristic that favors rules with high generality, to find local rules, which cover different (overlapped) subgroups of instances. Our method combines expression data with genomic location information to discover both (1) relevant transcription factors from the set of potential transcription factors of a target gene; and (2) transcriptional regulatory rules that describe the relationship between the expression behavior of a target gene and the expression behavior of its relevant transcription factors. These regulatory rules reveal some regulatory circuits that describe how some transcription factors regulate target genes.

3.1 Introduction

Even in a simple model organism like *Saccharomyces cerevisiae*, the mechanism of gene transcriptional regulation is extremely complex and uncovering it is one of the central problems in computational biology. In any organism, there are about 10% of genes that can produce proteins having a transcriptional role. These special proteins are called transcription factors (TFs) or regulators, and their DNA-binding interactions make the set of downstream genes express.

Mapping of DNA-binding locations of transcription factors has been proposed by using both experimental [5, 13, 39, 57] and computational approaches [25, 10, 11, 4, 41]. However, while genomic location data from experimental approaches is noisy due to imperfect measuring methods [39], computational approaches often suffer from over-prediction problems due to the short length (less than 20 bases) of the sequence motifs bound by the transcription factors. Harbison *et al.* [7] have constructed an initial map of yeast transcriptional regulatory code at different confidence levels by incorporating results from both experimental and computational methods. The frequency of false positives in genome-wide location data ranges from 6 to 10%, and about one-third of actual DNA-regulator interactions are not reported at the 0.001 *p-value* confidence level [39]. Nevertheless, increasing the *p-value* threshold (lowering the confidence) to include more true DNA-regulator interactions makes the rate of false positives increase. An additional problem we have to deal with when studying transcriptional regulation is the fact that interactions between regulators and DNA-binding sites are environment-dependent [7]. Many regulators only bind to the promoter of certain genes under specific environmental conditions. Even more, the presence of the regulators at a promoter region indicates binding but not necessarily function: the regulator may act positively, negatively or not act at all. Therefore, recognizing DNA-regulator interactions relevant for the transcription of a gene and how relevant regulators regulate the expression of that gene under specific environmental conditions are still important and open problems.

In this chapter, we propose a method that combines expression data with genomic location data to discover at the same time (1) relevant transcription factors from the set of potential transcription factors of a target gene; and (2) the relationship between the expression behavior of a target gene and the expression behavior of these relevant

transcription factors. We assume that transcription factors regulating the expression of a gene must bind to its promoter, and the expression of the target gene must be consistent, in a specific way, to the expression behavior of these transcription factors. Our method is based on rule induction, a machine learning technique which can efficiently deal with noisy domains like microarray data in our problem.

When applied to genomic location data with a relaxed confidence criterion (*p-value=0.005*) and three different expression datasets of yeast *Saccharomyces cerevisiae*, our method produced a set of regulatory rules comprehensively describing the relationship between the expression behavior of a specific target gene and the expression behavior of its relevant transcription factors. We were able to find the most frequent transcription factors occurring in regulatory rules under different conditions: response to environmental stress, response to DNA-damaging agents, and during the cell cycle. The resulting regulatory rules reveal some important regulatory circuits that clearly describe how a group of transcription factors regulates target genes.

We illustrate how the resulting regulatory rules provide strong evidences of true positive interactions between genes and regulators, as well as evidences of protein-protein interactions that could serve to identify transcriptional complexes.

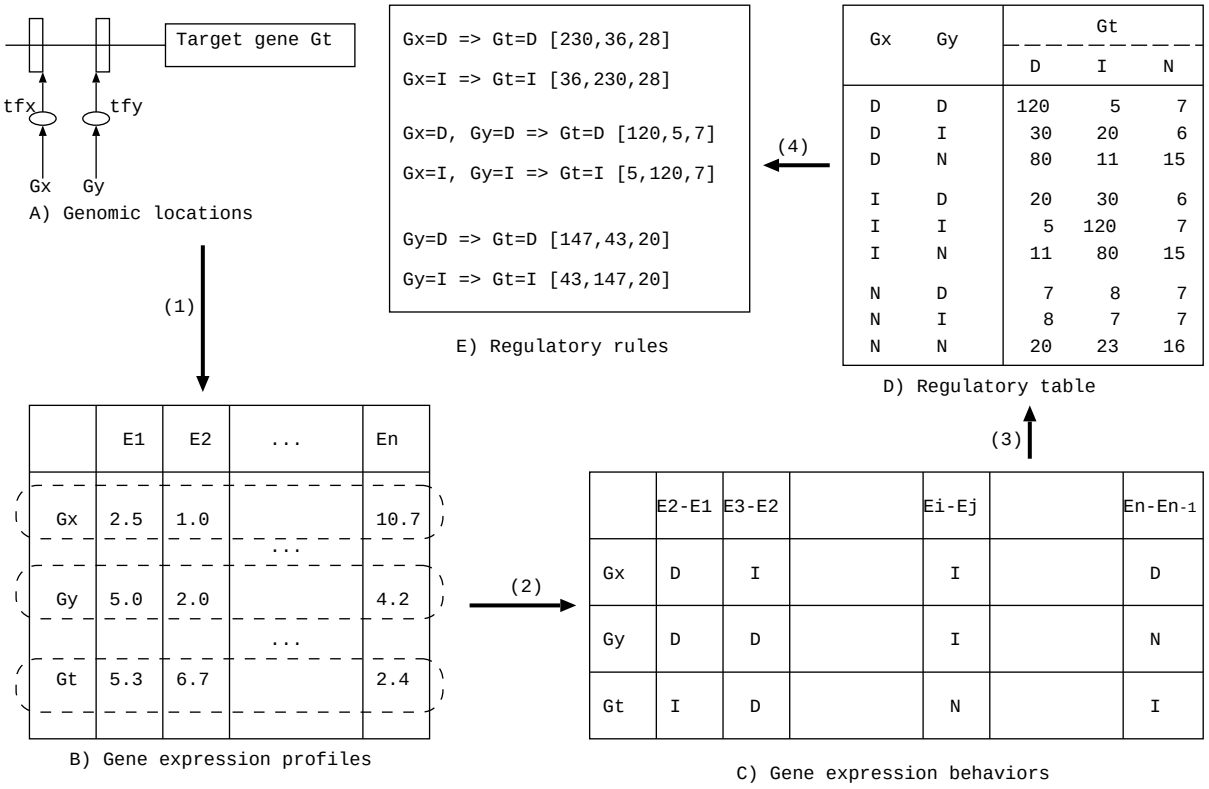
3.2 Methods

3.2.1 Overview of method

In this work we find regulatory rules that relate the expression profile of a gene with that of its regulators. Given n potential transcription factors $tf_1, tf_2, \ldots, tf_n$ binding to the promoter of a target gene G_t and assuming genes $G_1, G_2, \ldots, G_n$ are responsible for expressing those factors, (see Fig. 3.1A for an example with $n = 2$), we build a regulatory table as follows: we first determine the expression profiles of $G_1, G_2, \ldots, G_n$ and G_t (Fig. 3.1B) from the expression data. By comparing the results of pairs of experiments, we can determine if the expression of genes increased (I), decreased (D) or did not change (N) at the same time (Fig. 3.1C). With this information we can construct the regulatory table with instances of the form $(G_1 = v_1, G_2 = v_2, \ldots, G_n = v_n, G_t = v_t, count = k)$, with $v_i = I, D, or N$ (Fig. 3.1D). Section 3.2.2 provides more information on the regulatory

tables. From the regulatory tables, we then apply the CN2-SD rule induction system (see section 3.2.3) to produce a set of regulatory rules (Fig. 3.1E).

Figure 3.1: Approach overview



3.2.2 Regulatory tables

Given a gene and a set of its potential transcription factors, the regulatory table of this gene (the target gene) is a contingency table that describes the relation between the expression behavior of the gene and its regulators. If a gene has n potential regulators, its regulatory table consists of at most 3^n rows, since the expression behavior of each regulator has 3 states: I (upregulation), D (downregulation) and N (no change). For each set $(G_1, G_2, \ldots, G_n, G_t)$ of regulators and target gene, we then study their expression profiles. Every experiment is compared against all others to determine if the expression of a gene increased ($e_x - e_y > T$), decreased ($e_x - e_y < -T$) or did not change ($|e_x - e_y| \leq T$). Section 3.2.8 describes how the threshold T is determined. Fig. 3.1D is an example of regulatory table of a gene with two potential regulators.

These regulatory tables have two important characteristics to consider. First, they

contain noise from three different sources: (1) imperfect measurement methods to collect gene expression data, (2) uncertainty of interactions between transcription factors and the target gene (as explained in Section 3.1), and (3) method to obtain the threshold T to decide whether the expression increased, decreased or did not change. Due to these factors, the expression behavior of potential regulators turns out to be often inconsistent with the expression behavior of the target genes. To alleviate this inconsistency problem, we use the counts for each state of the expression behavior of the gene (Fig. 3.1D).

A second important fact to notice about the regulatory tables is that they might be sometimes incomplete. Since we construct these tables from expression data, some combinations of expression behavior of the set of regulators may have never happened under any conditions, or occurred with very low frequency as a result of noise.

Even though the regulatory tables can be incomplete and are constructed from noisy sources, consistent relationships between expression behavior of genes can nevertheless be uncovered from them. These relationships are represented in the form of a rule $G_{i_1} = v_{i_1}, \ldots, G_{i_k} = v_{i_k} \rightarrow G_t = v_t$, which takes account only of transcription factors $G_{i_1}, \ldots, G_{i_k}$ relevant for the expression behavior of the target gene G_t, and that ignores other non-relevant factors. In the following subsections, we will present a machine learning technique, rule induction, to efficiently discover such kinds of rules from regulatory tables.

3.2.3 Rule induction by CN2

Rule induction from examples is a machine learning technique that has been successfully used as a support tool for knowledge acquisition and prediction. The induced rules are usually expressed as $condition \rightarrow class$, where *condition* and *class* are logic expressions of the form $(variable_1 = value_1 \wedge variable_2 = value_2 \wedge \ldots \wedge variable_k = value_k)$.

There are three kinds of rule inducting algorithms: covering, decision tree-based and association rule-based. The first ones, covering algorithms, make use of a *separate-and-conquer* strategy over the search space to learn a rule set (see [65] for an overview). This *separate-and-conquer* strategy searches for a rule that explains (covers) part of its training instances, separates (or reassigns with lower weight) these examples, and recursively conquers the remaining examples by learning more rules until no examples remain. Decision

tree-based algorithms use a *divide-and-conquer* strategy [69, 70]. Much of the popularity of these algorithms stems from their efficiency in learning and classification. Decision trees can be easily turned into a rule set by generating one rule for each path from the root to a leaf. Finally, association rule-based algorithms use a *exhaustive search* strategy by exploring almost the whole search space [56, 40]. The basic idea is to use an association rule algorithm to gather all rules that predict the class attribute and also pass a minimum quality criterion.

By implementation, the divide-and-conquer strategy (in decision tree-based algorithms) is restricted to learn non-overlapping rules only. The exhaustive strategy (in association rule-based algorithms) has the problem of producing many redundant rules. The separate-and-conquer algorithms can partially avoid these disadvantages [65, 3], which is one of the main reasons for its popularity.

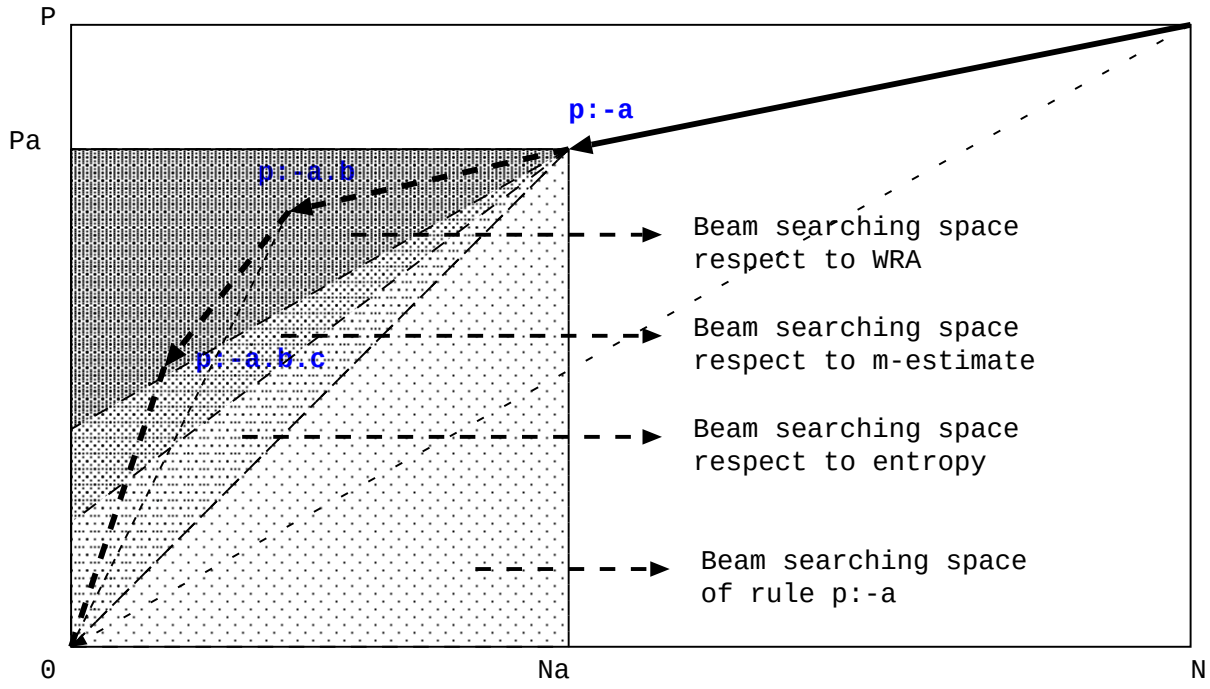
CN2 is a rule induction system implementing the separate-and-conquer strategy [54, 48]. It learns a rule set by iteratively adding rules one at a time. The system starts by using a general-to-specific search (described below) to learn the best rule according to some measures. Examples covered by this rule are removed from the search space before learning the next rule to add to the rule set. This is repeated until all examples are covered by at least one rule in the rule set or some stopping criteria is satisfied.

The general-to-specific search (Fig. 3.2) finds a single rule to be added into the set of learned rules. Beginning with a default rule (classify all examples as belonging to a positive class), it searches the space of possible rules by successively specializing the current best rule. Rules are specialized by greedily adding the condition which promises the highest gain according to a predefined heuristics.

3.2.4 Rule searching heuristics

During the general-to-specific search, CN2 must evaluate the rules it finds to decide which one is the best. Different measures (heuristics) have been proposed for this purpose, as well as for filtering out uninteresting rules and/or for stopping the refinement process at an appropriate point: accuracy, entropy h_{en} [54], Laplace $h_{laplace}$ [48], m-estimate $h_{m\_estimate}$ [62], weighted relative accuracy h_{WRA} [3], etc. The measures h_{en}, $h_{laplace}$,

Figure 3.2: Heuristic-dependent beam searching spaces



P, N: number of positive and negative examples. Pa, Na: number of positive and negative example covered by rule $p \leftarrow a$.

$h_{m\_estimate}$, and h_{WRA} are defined as follows:

$$h_{en} = -\sum_i p_i log_2(p_i) \tag{3.1}$$

$$h_{laplace} = \frac{n_c + 1}{n_{tot} + k} \tag{3.2}$$

$$h_{m\_estimate} = \frac{n_c + p_0(c)m}{n_{tot} + m} \tag{3.3}$$

$$h_{WRA}(condition \rightarrow class) = \frac{p(condition)}{p(class|condition) - p(class)} \tag{3.4}$$

where p_i denotes the probability distribution of examples among the k classes, n_c is the number of examples covered by the rule if the predicted class is c, n_{tot} represents the total number of examples covered by the rule, and $p_0(c)$ is the prior probability of class c.

In general, given a rule r, a heuristic h restricts the beam searching space for rules

more specific than r in the general-to-specific search [50]. Therefore, the larger the h-beam searching space is, the larger the probability that a more specific rule can be found respect to heuristic h. As it can be seen in Fig. 3.2, $bss_{h_{en}}(r) \geq bss_{h_{m\_estimate}}(r) \geq bss_{h_{WRA}}(r)$, where bss stands for beam searching space. This actually explains why h_{en} prefers specific rules [48], while h_{WRA} produces rules with a high generality. This also agrees with the opinions expressed in [3], where a measure like h_{WRA} was considered more suitable for description tasks (like the problem described in this chapter) rather than for prediction tasks. Therefore, in the rest of this chapter we will discuss results obtained by using the heuristic h_{WRA}.

3.2.5 CN2-SD for knowledge discovery

The original CN2 [54] works by learning a set of ordered rules. According to a certain heuristic measure, CN2 looks for the best rule in the set of training examples. Once a rule is found ("induced"), all examples covered by the induced rule are removed from the training set, and the system starts again to look for a new rule. The result of a typical CN2 session is a list of nested rules of the form "*if .. then .. else ..* ". The nested structure can become extremely complex when the number of rules is high, making this representation difficult to interpret. Subsequent versions of CN2 [48] allow the induction of unordered rule lists. Rules are learned for each class independently, and for each induced rule only covered examples belonging to the class are removed, instead of removing all covered examples. The produced final rules can therefore overlap, but at the same time can be interpreted independently. When using an unordered rule list to predict the class of new instances, several rules can contribute to the classification of this example, often resulting in an improved accuracy.

Unfortunately, and due to the way CN2 iteratively removes examples, in an unordered rule list only the first few induced rules are usually of interest. Subsequently induced rules are obtained from biased example subsets, i.e., subsets including only positive examples not covered by previously induced rules. This is not suitable for our description task (discovering the regulatory rules hidden in expression data), where desired rules may cover overlapped instances. CN2-SD [3], a modification of CN2 for subgroup discovery, solves this problem and will be the rule induction system used in the experiments reported

in this work. The basic idea is to generalize the covering algorithm by introducing example weights. Initially, all examples have a weight of 1.0. However, the weights of examples that are covered by a rule will not be set to 0.0 (as in CN2), but instead will be reduced by a certain factor. The resulting number of rules is typically higher than with CN2, since most examples will be covered by more than one rule. CN2-SD has therefore two complementary advantages: it can learn better local patterns because the influence of previously covered patterns is reduced, but not completely ignored; and, it can produce a better classifier by combining the evidence of more induced rules.

3.2.6 Filtering regulatory rules

As explained above, by using a weighted covering strategy, CN2-SD can restrict the redundancy of learned rules and guarantee the scanning of the whole search space. However, uninteresting regulatory rules are still produced, mainly due to noise in the microarray data we use as a source. In our system, in addition to the significance test, which ensures that the distribution of examples among classes covered by a rule is significantly different from the distribution that would be obtained by random assignment [54], we use two other heuristics to filter out trivial and inconsistent regulatory rules. Given a regulatory rule r: $TF_1 = v_1, \ldots, TF_n = v_m \rightarrow$ *Target_gene*=v $[n_D, n_I, n_N]$, where $v_i (i = 1, \ldots, m)$ and v are values in $\{D, I, N\}$, the expression behaviors (downregulation, upregulation, and no change) of genes or transcription factors; and $[n_D, n_I, n_N]$ are the class distribution of examples covered by this rule r.

1. *Removing trivial regulatory rules and irrelevant conditions in a rule*: r is called a trivial regulatory rule if its predictive value of *Target_gene* is N (no change). This regulatory rule can be interpreted as: "there is no relationship between the target gene and its transcription factors". This kind of rules is therefore trivial, and should be removed from the learned rule set. The remaining rules are those predicting *Target_gene* to be D or I.

   Moreover, if there is any transcription factor in the condition part of a rule appearing with value N (no change), the transcription factor has no role in regulating the expression of the target gene. We also remove these irrelevant factors in the condition part of regulatory rules, and update the class distribution for new rules.

2. *Removing inconsistent regulatory rules*: the consistence (*cons*) of a non-trivial regulatory rule r is defined as:

$$cons(r) = \frac{n_p}{n_D + n_I} * \frac{n_p}{n_D + n_I + n_N} \tag{3.5}$$

where n_p is equal to n_D if r is a classification prediction rule for *Target_gene* belonging to class D, and equal to n_I if r is a classification prediction rule for *Target_gene* belonging to class I. The consistence $cons(r)$ takes into account two factors: a *confidence-without-noise* ($\frac{n_p}{n_D+n_I}$) and *confidence-with-noise* ($\frac{n_p}{n_D+n_I+n_N}$), where n_N is the parameter to represent the noise in microarray data. Clearly, $0 \leq cons(r) \leq 1$, and the higher the value of $cons(r)$, the higher the confidence that regulatory rule r is true.

3.2.7 Datasets

In all our experiments we used genomic location data (as described in [7]) as a source for potential gene-transcription factor interactions. This dataset contains interactions between 106 transcription factors and about 6200 genes of yeast *Saccharomyces cerevisiae*, with a relaxed binding criterion of confidence *p-value* ≤ 0.005 (this relaxed confidence value is expected to increase the number of true and functional interactions that can be found), and conserved in at least one other yeast specie. Three expression datasets (see Gasch *et al*, 2000 [46], 2001 [37]; Spellman *et al.*, 1998 [16]) are also used to analyze the expression behavior of target genes as well as transcription factors response to environmental stresses, response to DNA-damaging agents, and during the cell cycle, respectively. The number of experiments of these three datasets is 172, 52, and 77, respectively.

3.2.8 Assigning expression behavior labels: upregulation, downregulation and no change

We compare the expression values e_i and e_j of a gene between any two microarray experiments i and j to determine its expression behavior. If $e_j - e_i > T$ the expression behavior of the gene is upregulated (I) from experiment i to experiment j; if $e_j - e_i < T$ the expression behavior of the gene is downregulated (D); otherwise it is non-changed (N).

When the threshold value T is large, our system will produce regulatory rules with high *confidence-without-noise* but low *confidence-with-noise* (see 3.2.6). These regulatory rules are often true positives although by using a high threshold we are also discarding some relevant regulatory rules. Inversely, if the value of T is small, our system will produce many irrelevant regulatory rules due to the noise in microarray data. To determine a reasonable threshold T for a microarray dataset, we first set an initial value T_0 large enough, then apply our method to find a set of true positive regulatory rules. This set is considered as previously known regulatory rules (since the set often includes true positives). We then tune the parameter T to get the highest value of the average measure *cons* (Eq. 3.5) over all the rules in this set. By using this method, we obtained threshold values T for Gasch *et al* (2000)'s data (1.3), Gasch *et al* (2001)'s data (0.75), and Spellman *et al* (1998)'s data (1.0).

3.3 Results

Results were obtained by using the datasets for gene expression profiles response to environment conditions Gasch *et al* (2000) [46], Gasch *et al* (2001) [37] and Spellman *et al.* (1998) [16], with the corresponding T threshold values calculated in section 3.2.8 (1.3, 0.75 and 1.0 respectively). These datasets represent the gene expression profiles response to stress conditions, response to DNA-damaging agents, and during the cell cycle, respectively. Genomic location data of yeast *Saccharomyces cerevisiae* with binding criterion relaxed to *p-value* ≤ 0.005 and conserved in at least one other yeast [7] is used to determine potential transcription factors of a gene. We removed genes that are bound by no regulator and where 95% of the total number of expression behaviors were N(non-changed). There are 1800, 2133, and 1172 remaining genes for the three expression datasets that are bound by at least one transcription factor and significantly expressed, i.e. they have a number of expression behaviors of classes D or I greater than 5% of their total number of behaviors. For each gene in these sets and each expression dataset, we constructed a regulatory table (see Fig. 3.1). As a result, we obtained 1800, 2133, and 1172 regulatory tables for the three datasets. The algorithm CN2-SD [3] with WRA heuristic (Eq.3.4) is then applied to find all regulatory rules from these regulatory tables. Finally, we filtered trivial rules, trivial conditions in rules (section 3.2.6), regulatory rules covering few

examples, and rules with consistence lower than 0.3.

Table 3.1: Summary of produced regulatory rules

	Resp. to env. changes[a]	Resp. to DNA-dama.[b]	Cell cycle[c]	all
# rules	2438	1974	506	3707
# genes	1002	889	288	1336
# found interactions	2206	1938	580	3033
# found interactions with p-value$\leq$0.001	1518 (68.8%)	1350 (69.7%)	401 (69.1%)	2103 (69.3%)
# p-value$\leq$0.001, no interaction found	475/1993 (23.8%)	557/1927 (28.9%)	389/790 (48.2%)	455/2558 (17.8%)
ten most frequent regulators	rap1,abf1, ste12,fhl1, reb1,nrg1, hsf1,swi6, ume6,cbf1	rap1,fhl1, hsf1,gcn4, abf1,ste12, cin5,msn4, cbf1,mbp1	swi6,ste12, swi4,mbp1, dig1,msn4, phd1,fkh1, fkh2,abf1	rap1,abf1, ste12,reb1, gcn4,hsf1, nrg1,cbf1, swi6,fhl1

Note: [a] Gasch *et al* (2000) [46]; [b] Gasch *et al* (2001) [37] and [c] Spellman *et al.* (1998) [16].

We found 3707 regulatory rules for predicting 1336 target genes to be D and the same number of rules for predicting target genes to be I. Since we analyze any pair of experiments without considering their order, with each regulatory rule for predicting the target gene to be D, there is an equivalent regulatory rule for predicting that target gene to be I where variables in the condition part of the rule received the opposite values ($I \leftrightarrow D$). For example, rules $Gx = D, Gy = D \rightarrow Gt = I$ and $Gx = I, Gy = I \rightarrow Gt = D$ are equivalent. For simplicity, we will refer only to regulatory rules for predicting target gene Gt belonging to class D as the representative ones.

Table 3.1 shows the number of regulatory rules, number of genes controlled by these rules and the ten most frequent transcription factors found from these three expression datasets. We found 1002 genes appearing in 2438 rules regulated in response to environmental changes; 889 genes appearing in 1974 rules response to DNA-damaging agents; 288 genes appearing in 506 rules regulated for the cell cycle; and a total of 1336 genes in 3707 rules in all three kinds of environments. We also found that the most frequent transcription factors occurring in regulatory rules in response to environmental stresses (RAP1, ABF1, STE12, FHL1, REB1, etc.) and in response to DNA-damaging agents (RAP1, FHL1, HSF1, GCN4, ABF1, etc.) are quite similar and agree with the function they have been annotated with in Gene Ontology [24], while the most frequent tran-

scription factors occurring in regulatory rules from the cell cycle dataset (SWI6, STE12, SWI4, MBP1, DIG1, etc.) have functions previously reported to control the cell cycle during growth [24]. The full, detailed set of regulatory rules can be obtained from *http://www.jaist.ac.jp/~h-pham/regulatory-rules*, files **regulatory-rules.txt*.

It should be noticed how one gene is often regulated by one or more transcription factors depending on environmental conditions. For example, gene YPR145W (ASN1) is regulated by different subsets of regulators (Table 3.2) under environmental changes. The transcription factors that most influenced the expression of YPR145W response to environmental changes are STE12, with regulatory rule YPR145W=D $\leftarrow$ STE12=D covering 2703 instances and consistence 0.82; and DAL82, which negatively regulates YPR145W. The transcription factors GCN4 and GLN3, when acting together with STE12 or DAL82, can increase activation ability of these factors. Conversely, a transcription factor (independently or co-operatively with others) can regulate many different genes at the same time. For example, MBP1 interacts with SWI6 in different ways to regulate the activity of 15 different genes (Table 3.3).

3.4 Discussions

3.4.1 Relevant interactions from genomic locations data

We analyzed relevant interactions between 94 transcription factors and 1336 genes occurring in 3707 regulatory rules found by our method from three microarray datasets. We found 3033 relevant interactions among them (Table 3.1), 2103 (69.3%) of which have been reported in the genomic locations data with *p-value* ≤ 0.001 [39]. Therefore 31.7% of the relevant interactions found in regulatory rules are from potential ones in the genomic locations with $0.001 <$ *p-value* ≤ 0.005. This result agrees with the work of Lee *et al* [39], where it was reported that about one-third of actual DNA-regulator interactions present in genomic locations data are missed at *p-value=0.001*. Details of interactions in regulatory rules and in genomic locations data for the 1336 genes can also be obtained from the complementary on-line material in our web site. 455 of the 2558 interactions involving the 1336 genes (Table 3.1) and all interactions involving other genes from this genomic locations data were not found in regulatory rules. The reasons are: (1) the

genomic locations data contains substantial noise; (2) in our experiments we only considered three kinds of environmental conditions, while the genomic locations data contains potential interactions between regulators and DNA that do not actually take place under the conditions we chose; and (3) many physically binding interactions are too weak or do not translate into a real function.

3.4.2 Regulatory circuits and transcription complexes

We used a clustering method based on a *closed itemset lattice*, which will be described in the next chapter, to group genes regulated by a common subset of transcription factors (here we consider each regulatory rule as a "transaction"). We define a regulatory circuit as a system including three components: a subset of genes, a subset of regulators and a subset of regulatory rules between them. Since a gene can be regulated by different subsets of regulators with different regulatory rules, it can belong to multiple regulatory circuits. In general, genes in each circuit often have similar or related functions that agree with the role of their transcription factors. We used *GO Term Finder* (http://www.yeastgenome.org/help/goTermFinder.html) to search for significant shared GO terms that are directly or indirectly associated with the genes in each regulatory circuit. To determine which terms are significant, the algorithm examines a group of genes to find GO terms to which a high proportion of the genes are associated, as compared to the number of times the term is associated with other genes in the genome. Table 3.4 describes regulatory circuits including some of the most frequent regulators. The complete set of regulatory circuits is available as supplementary material on-line. The regulatory circuits we obtained in this work are more detailed than those obtained in previous studies [47, 14, 15, 38, 33]: in addition to what genes and regulators are included in each module, our regulatory circuits also describe the regulatory relationship between them in the form of a rule under certain environmental conditions.

Table 3.3 shows a detailed description of one of the regulatory circuits appearing in Table 3.4. This circuit consists of 15 distinct genes commonly regulated by MBP1 and SWI6. These two regulators have been reported to form a complex involved in regulation of cell cycle progression [53, 55]. Out of 15 genes in this module, six of them (GIN4, MPT5, CLB6, SWE1, OPY2, and CLB5) are related to the cell cycle regulation ($P = 1.24E\text{-}07$),

as annotated in Gene Ontology.

This example suggests a possible use of our method to predict transcription complexes. We consider all regulatory circuits with more than two regulators and containing at least five genes. Regulators that co-activate or co-repress a specific set of genes are candidates to form transcription complexes. Table 3.5 shows candidate complexes that regulate 7 or more genes. For example, SWI4 and SWI6 co-regulate 16 distinct genes; TEC1 and STE12 co-regulate 15 distinct genes; INO2 and INO4 co-regulate 11 distinct genes; HAP2 and HAP4 co-regulate 10 distinct genes; FKH1 and FKH2 co-regulate 7 distinct genes; SWI4, SWI6 and STE12 co-regulate 4 distinct genes. These co-regulators have been also previously confirmed to interact in order to regulate genes (see Table 3.5 for references). There are some other pairs of co-regulators regulating 7 or more genes in the resulting regulatory rules for which we could not find any evidence in the BIND database [8]. For example, FHL1 and RAP1 co-regulate (almost always positively) up to 65 distinct genes, with most of these genes related to the process "protein biosynthesis". Until further experiments confirm or reject our results, we suggest that these pairs of co-regulators could be new transcription complexes. The complete list of co-regulators can be found on-line in our web site.

3.5 Summary

Data of DNA-protein interactions from experimental and computational methods is often noisy and contains information about physically binding interactions, although not necessarily functional ones. By combining this data with expression profiles data, our rule induction method can discover relevant transcription factors for a given target gene, as well as the relationship between the expression behavior of the target gene and that of its relevant regulators. When using a relaxed confidence value we were able to uncover interactions usually missed in other studies due to an excessively strict *p-value*. The use of three expression profiles obtained under different environments (stress response, DNA damage and cell cycle growth) allows us to establish not only if an interaction takes place, but also if it is functionally active and under what conditions it would happen.

Our method also provides evidence of transcription factors that commonly regulate different groups of genes. This result could be used to identify potential transcription

complexes, and we present examples of previously not reported complexes for which strong evidence was found.

Table 3.2: Examples of regulatory rules

Regulatory rules	Resp. environ. changes		Resp. DNA-damage		Cell cycle	
	classes distrib.	cons.	classes distrib.	cons.	classes distrib.	cons.
YPR145W=D ← STE12=D	[2198, 8, 479]	**0.82**	[36, 11, 130]	0.16	[0, 0, 102]	0.00
YPR145W=D ← GCN4=I,STE12=D	[300, 0, 33]	**0.90**	[0, 0, 0]	0.00	[0, 0, 5]	0.00
YPR145W=D ← DAL82=I	[820, 56, 435]	**0.59**	[19, 42, 155]	0.03	[0, 1, 6]	0.00
YPR145W=D ← DAL82=I,GCN4=I	[256, 15, 95]	**0.66**	[0, 1, 3]	0.00	[0, 0, 0]	0.00
YPR145W=D ← DAL82=I,GLN3=I	[356, 11, 92]	**0.75**	[6, 2, 14]	0.2	[0, 0, 0]	0.00
YBR067C=D ← ASH1=D	[1077, 111, 1102]	**0.43**	[111, 180, 303]	0.07	[408, 171, 696]	0.23
YBR067C=D ← ASH1=D,HSF1=I	[124, 0, 8]	**0.94**	[0, 7, 6]	0.00	[0, 3, 2]	0.00
YBR067C=D ← HSF1=D,NRG1=D	[74, 301, 228]	0.02	[24, 0, 6]	**0.80**	[0, 11, 16]	0.00
YBR067C=D ← HSF1=D	[139, 350, 360]	0.05	[51, 6, 41]	**0.47**	[26, 1, 51]	**0.32**

Table 3.3: Genes regulated by MBP1 and SWI6

Rule	Env.$^{+}$	GO terms for target gene
MBP1=D,SWI6=D → YBR070C=D	b,c	nuclear envelope-endoplasmic reticulum network
MBP1=D SWI6=I → YDR263C (DIN7) =D	a	DNA repair, mitochondrion
MBP1=I SWI6=I → YDR507C (GIN4) =D	c	protein amino acid phosphorylation*, protein kinase activity, bud neck
MBP1=D SWI6=I → YGL178W (MPT5) =D	a	cell wall organization and biogenesis*, mRNA binding,cytoplasm
MBP1=I SWI6=D → YGR109C (CLB6) =D	c	G1/S transition of mitotic cell cycle*, cyclin-dependent protein kinase regulator activity
MBP1=I SWI6=D → YGR152C (RSR1) =D	a,c	bipolar bud site selection*, GTPase activity*, plasma membrane*
MBP1=D SWI6=I → YGR180C (RNR4) =D	b	DNA replication, ribonucleoside-diphosphate reductase activity, cytoplasm*
MBP1=D SWI6=D → YJL187C (SWE1) =D	c	G2/M transition of mitotic cell cycle*, protein kinase activity, nucleus*
MBP1=D SWI6=D → YKL008C (LAC1)=D	a,c	ceramide biosynthesis*, sphingosine N-acyltransferase activity, endoplasmic reticulum
MBP1=I SWI6=D → YMR179W (SPT21)=D	c	regulation of transcription from Pol II promoter,nucleus
MBP1=I SWI6=D → YMR307W (GAS1) =D	a	cell wall organization and biogenesis,"1,3-beta-glucanosyltransferase activity", mitochondrion*
MBP1=D SWI6=D → YNR009W=D	a,c	unknown, cytoplasm*
MBP1=I SWI6=D → YNR009W=D	a	unknown, cytoplasm*
MBP1=I SWI6=D → YPL127C (HHO1) =D	c	"regulation of transcription, DNA-dependent*", DNA binding, nucleus*
MBP1=D SWI6=D → YPR075C (OPY2) =D	c	cell cycle arrest in response to pheromone, cytoplasm*
MBP1=I SWI6=D → YPR120C (CLB5) =D	c	G1/S transition of mitotic cell cycle*, cyclin-dependent, protein kinase regulator activity, nucleus

Note: $^{+}$Env = a, b, c ≡ regulatory rule is activated in response to environmental stresses, response to DNA-damaging agents, or during cell cycle, respectively.

Table 3.4: Description of some regulatory circuits

Regulators	Roles of regulators	#genes	significant shared GO terms
RAP1	transcriptional silencing of HML and HMR loci, activation of ribisomal glycolytic enzymes, ect.	111	(71/111) protein biosynthesis (P=4.99e-41) (21/111) ribosome biogenesis (P=1.37e-10)
ABF1	chromatin-reorganizing activity involved in transcriptional activation, gene silencing, and DNA replication and repair	132	(27/132) ribisome biogenesis (P=4.74e-14) (24/132) RNA processing (P=1.29e-08)
STE12	activates genes involved in mating or pseudohyphal/invasive growth pathways	78	(14/78) conjugation with cellular fusion (P=3.87e-12) (13/78) response to abiotic stimulus (P=3.69e-7)
FHL1	similarity to DNA-binding domain of Drosophila forkhead, required for rRNA processing	82	(66/82) protein biosynthesis (P=8.41e-49) (14/82) ribosomal subunit assembly (P=1.27e-15)
HSF1	heat shock transcription factor, activates multiple genes in response to hyperthermia.	99	(17/99) protein folding (P=3.57e-18) (17/99) response to stress (P=6.95e-06)
SWI6	G1/S transition, meiotic gene expression ocalization regulated by phosphorylation	67	(16/67) development (P=6.62e-6), (8/67) regulation of cell cycle (P=2.25e-5)
MBP1	involved in regulation of cell cycle progression from G1 to S phase	37	(6/37) DNA replication (1.35e-5), (10/37) DNA metabolism (0.00023) (5/37) DNA repair (0.0006)
MBP1 &SWI6	a complex regulates transcription at the G1/S transition	15	(6/15) regulation of cell cycle (P=1.24e-7), (3/15) regulation of cyclin dependent protein (3/15) kinase activity (P=3.91e-6)
SWI4	Involved in cell cycle dependent gene expression	36	(6/36) regulation of cell cycle (P=3.52e-5), (4/36) G1/S transition of mitotic cell cycle (P=8.71e-5), (3/36) G2/M transition of mitotic cell cycle (P=0.00059)
GCN4	amino acid biosynthetic genes in response to amino acid starvation	77	(21/77) amino acid and derivative metabolism (P=6.86e-16)

Table 3.5: Candidate complexes

Candidate complex	#genes	External evidences
FHL1 & RAP1	65	
DIG1 & STE12	21	Olson *et al.* (2000), BIND Id: 130453
SWI4 & SWI6	16	Siegmund & Nasmyth (1996), BIND Id: 24482
MBP1 & SWI6	15	Siegmund & Nasmyth (1996), BIND Id:24484
TEC1 & STE12	15	Kim *et al.* (2004)
CAD1 & YAP7	14	
TYE7 & CBF1	13	
MSN2 & MSN4	12	
YAP1 & YAP7	11	
DAL82 & GLN3	11	
INO2 & INO4	11	Wagner *et al.* (2001), BIND Id: 126362
HAP2 & HAP4	10	McNabb *et al.* (1997), BIND Id: 170195
UME6 & ABF1	9	
CBF1 & GCN4	9	
CBF1 & ABF1	9	
PHD1 & NRG1	8	
SOK2 & CIN5	8	
CIN5 & NRG1	8	
MBP1 & STE12	8	
TEC1 & DIG1	7	
STB1 & MBP1	7	
FKH2 & FKH1	7	Hollenhorst *et al.* (2000), BIND id: 172668
CAD1 & YAP1	7	
YAP7 & GCN4	7	
TEC1,DIC1,STE12	4	
SWI4,SWI6,STE12	4	Breeden & Nasmyth (1987)

Chapter 4

Discovery of regulatory modules

This chapter describes a different approach to uncover regulatory patterns other than regulatory rules. The method will find an expression profile pattern of a group of genes commonly regulated by a specific set of transcription factors. Our approach starts by clustering genes into modules based on a closed TFset lattice, which includes non-redundant combinations of transcription factors respective to a database of DNA-protein interactions; and then validate the expression profiles to confirm regulatory modules.

4.1 Introduction

In eukaryotes, gene expression is controlled by various transcription factors (TFs) that bind to promoter regions and can act in combination. With combinatorial control, a given transcription factor does not necessarily have a single, simply definable function as commander of a particular battery of genes or specifier of a particular cell type. Rather, transcription factors can be described in a similar way to the words of a language: they are used with different meanings in a variety of contexts and rarely alone. It is the well-chosen combination that conveys the information that specifies a gene regulatory event [1].

In Chapter 3 of this thesis, we developed a rule induction method to identify proteins binding to DNA that are relevant for the expression of some genes. Moreover, the method could find consistent relationships between the expression of a gene and relevant transcription factors binding to its promoter in a rule form. However, these regulatory rules only describe some local parts of expression data under some conditions. The relationships hidden in certain data collections may be more complicated and could not be represented in a rule form. Furthermore, expression patterns of genes are not always directly effected by the expression profiles of the factors regulating these genes. Indeed, in many cases transcription factors are post transcriptionally modified, and consequently we cannot examine their protein levels. Transcriptional regulatory rules described in Chapter 3 may not capture some regulatory patterns.

Another approach to uncover regulatory patterns that overcomes the restriction of regulatory rules is to discover a common expression pattern of genes bound by the same set of transcription factors. There exists two different approaches: genes can be clustered into modules based on the similarity of their expression profiles, and then common binding sites of transcription factors in the promoter region of genes in each module can be found [47, 14]. These methods have achieved various levels of success, but an intrinsic limitation is their over-reliance on expression data, which represent the result rather than the cause of genetic regulation. Alternatively, Bar-Joseph *et al.* proposed a method (GRAM) that first groups genes into modules where each module includes genes commonly bound by a set of transcription factors and then validates the expression profiles to confirm these modules [15].

The GRAM algorithm discovers gene modules that are a set of coexpress genes to

which the same set of transcription factors binds. Roughly, the GRAM algorithm scans all subsets of transcription factors (TFsets) and analyzes the expression profiles of genes to which each TFset binds. If the expression profiles of these genes are significantly similar, they would be controlled by the TFset. However, scanning all TFsets in the subset space is an extremely demanding task. GRAM uses Apriori properties to tackle this problem, i.e., if a gene i is included in a module controlled by a set F of TFs, it is likely to be included in any module controlled by a subset $F' \subset F$. Since we are interested in the complete set of factors controlling a gene, we do not gain anything by including i in module regulated by F'. Thus, prior to computing the set of genes contained in module regulated by F', they filter out any genes that have already been found to be included in a module controlled by a superset of F'. This reduces the number of overlapping modules, without reducing the explanatory power of this approach. But when the number of TFs, genes, and interactions between them are large, dealing with all the possible subsets becomes an infeasible task.

In this chapter, we introduce a new method to cluster genes into modules based on a closed TFset lattice, which includes non-redundant combinations of transcription factors respective to a database of DNA-protein interactions, and hence reduces redundant modules as many as possible. We also combine the closed TFset lattice with gene expression profiles to find out an expression pattern of genes in each module.

Our method has been applied to yeast data for finding transcriptional regulatory modules (TRMs). The results agree with gene modules found by previous studies. Moreover, TRMs are more compact, concise and comprehensive to identify and interpret the transcriptional control of combinations of regulators.

4.2 Mining frequent itemsets and closed itemsets

Frequent itemsets mining is the most important and demanding task in many data mining applications [51]. Let $\mathcal{I} = \{a_1, \ldots, a_M\}$ be a finite set of items and $\mathcal{D}$ be a finite set of transactions (the dataset) where each transaction $t \in \mathcal{D}$ is a list of distinct items $t = \{x_0, \ldots, x_T\}, x_i \in \mathcal{I}$. An ordered sequence of n distinct items $I = \{i_0, i_1, \ldots, i_n\} | i_j \in \mathcal{I}$ is called an itemset of length n, or n-itemset. The number of transactions in the dataset including an itemset I is defined as the *support* of I, denoted by $supp(I)$. Given a

threshold $MinSup$, an itemset is said to be frequent if its support is greater than or equal to $MinSup$, infrequent otherwise.

There are basically two kinds of algorithms for finding frequent itemsets. The first is Apriori algorithm [51] and its variants (see the work of Zaki and Hsiao [6] for an overview). They use the basic properties (Apriori properties) that all subsets of a frequent itemset are frequent and that all supersets of an infrequent itemset are infrequent in order to prune elements of the space of itemsets. These properties make it possible to effectively mine sparse datasets. However, with dense datasets, which contain strongly related transactions, it becomes much harder to mine since only a few itemsets can be pruned and the number of frequent itemsets grows very quickly while decreasing of $MinSup$ threshold. As a consequence, the mining task becomes rapidly intractable by these algorithms, which try to extract all the frequent itemsets.

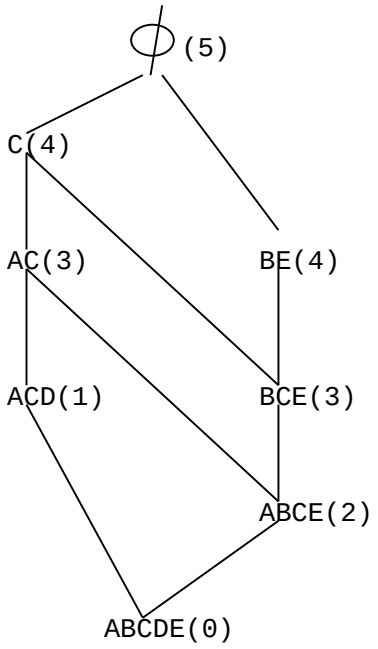
The second type of algorithms, which finds frequent closed itemsets, can avoid the above mentioned problem. A closed itemset is described as a maximal set of items common to a set of transactions. In other words, an itemset I is a closed itemset if there exists no itemset I' such that $I' \supset I$ and $supp(I') = supp(I)$. For example, in the transaction database $\mathcal{D}$ in Fig. 4.1, the itemset BCE is a closed itemset since it is the maximal set of items common to the transactions {2,3,5}. It is called a frequent closed itemset for $MinSup = 2$ as $supp(BCE) = 3 \geq MinSup$. The itemset BC is not a closed itemset since it is not a maximal group of items common to some transactions: all transactions including the items B and C also include the item E. All closed itemsets of a dataset form a lattice that is dually isomorphic to the Galois lattice [58]. In the figure, the lattice contains 8 closed itemsets. It is much smaller than the complete space of itemsets, which in this case includes up to 32 (5 items: 2^5) itemsets. The exact definition of closed itemsets and their useful properties have been described in the work of Pasquier et al. [58] and Zaki [75].

The set of closed itemsets is often much smaller than the set of all itemsets, but it presents exactly the same knowledge in a more succinct way. From the set of closed itemsets it is straightforward to derive both the identities and supports of all itemsets. Mining the frequent closed itemsets is thus semantically equivalent to mining all frequent itemsets, but with the great advantage that frequent closed itemsets are often orders of

Figure 4.1: closed itemsets from a database of transactions

TID	Items
1	A C D
2	B C E
3	A B C E
4	B E
5	A B C E

Transaction database



Closed itemset lattice

magnitude fewer than frequent ones. Using closed itemsets we implicitly benefit from data correlations which strongly reduce problem complexity [75].

Many algorithms for finding frequent closed itemsets have been developed such as CHARM [6], A-close [58], FPClose [28], etc. In our work, we used FPClose by Grahne and Zhu, implemented in C language.

4.3 Definition of closed sup-TRM and closed inf-TRM

Our purpose is to discover groups of transcription factors where each group (or $TFset$) binds to a set of genes ($geneset$) and regulates their expression. In other words, we want to find all transcriptional regulatory modules (TRMs) having a form $TFset \rightarrow geneset$, where $geneset$ is a set of genes that are bound by $TFset$ and similar in their expression profiles.

However, many TRMs are not informative since we cannot infer the biological meaning from them. For example, from the database of transcription factor binding sites in Fig. 4.2 we cannot infer that $TF2\_TF4 \rightarrow G1\_G3\_G8$ means "$TF2\_TF4$ transcriptionally regulates $G1\_G3\_G8$", because in addition to $TF2$ and $TF4$, there is another factor

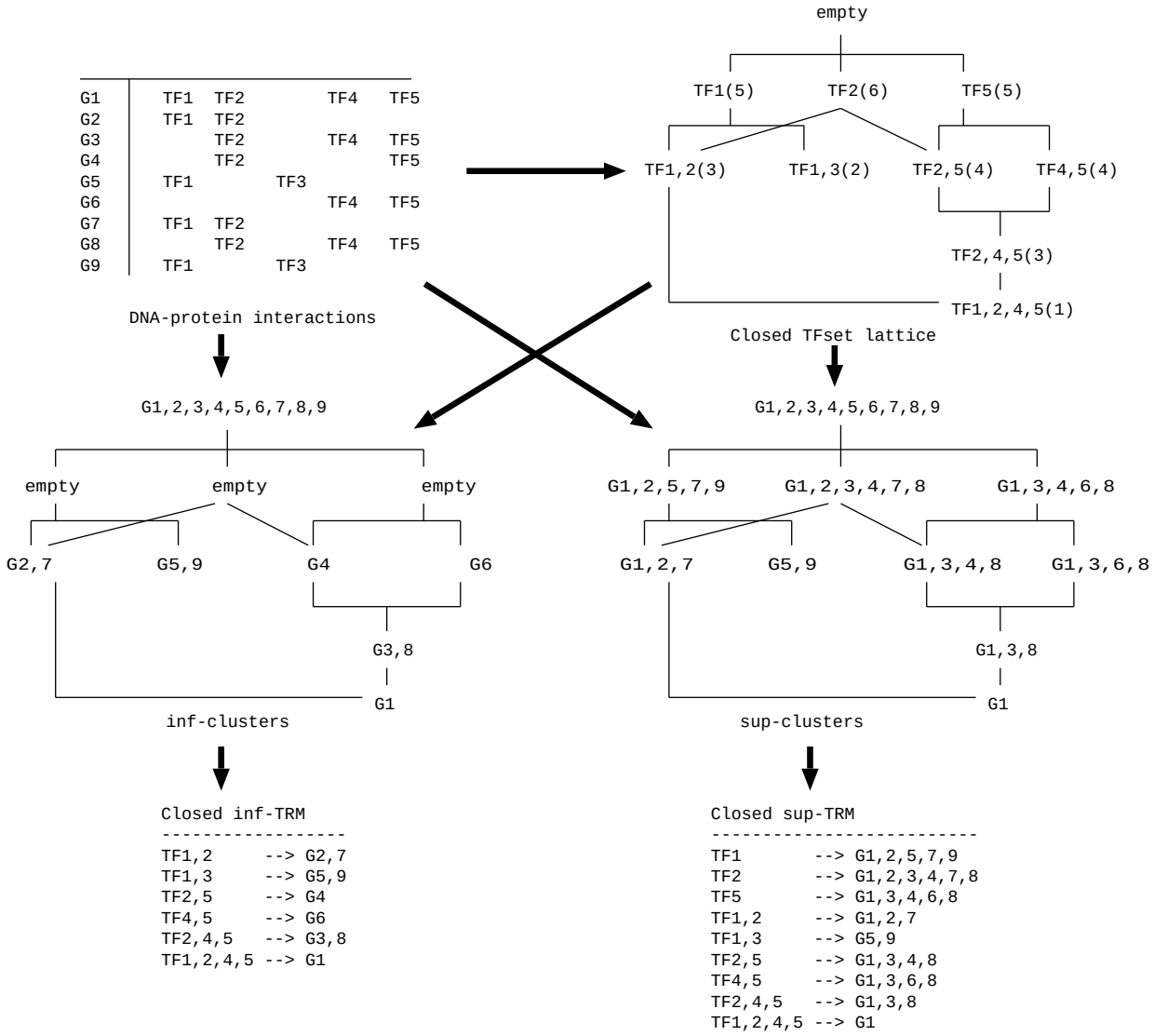


Figure 4.2: closed sup-TRMs and closed inf-TRMs

($TF5$) binding to all $G1, G3$ and $G8$. In other words, $TF2\_TF4$ is not a maximal set of factors commonly binding to {G1, G3, G8}. A TRM is informative only if its $TFset$ is a maximal set of factors (i.e.,*a closed TFset*) commonly binding to a set of genes. In this work, we focus only on informative TRMs, which have a form $closed\_TFset \rightarrow geneset$ (we will refer to this as *closed TRM*). As explained above, the set of closed TFsets is much smaller than the set of all TFsets, so the search space for closed TRMs is greatly reduced.

Moreover, we would like to find closed TRMs that emphasize the transcriptional role of their $closed\_TFset$. To do this, we can analyze the expression profiles of genes ($geneset$) that $closed\_TFset$ binds to. If they are similar we believe that $geneset$ is controlled by $closed\_TFset$. Here, there are two strategies to group genes into $geneset$ of a closed TRM.

First we can set *geneset* to be the maximal set of genes to which factors in $closed\_TFset$ commonly bind (we refer to it as *sup_geneset* and the corresponding TRM as *closed sup-TRM*). Second we can set *geneset* to be only genes bound exactly by $closed\_TFset$ (we refer to it as $inf\_geneset$ and the corresponding TRM as *closed inf-TRM*). For example, in Fig. 4.2 the maximal set of genes that the closed TFset $TF2\_TF4\_TF5$ commonly binds to is $\{G1, G3, G8\}$, therefore $TF2\_TF4\_TF5 \rightarrow G1\_G3\_G8$ is a closed sup-TRM. There are only 2 genes $G3$ and $G8$ that the exact closed TFset $TF2\_TF4\_TF5$ binds to, therefore $TF2\_TF4\_TF5 \rightarrow G3\_G8$ is a closed inf-TRM.

The reason for clarifying these two kinds of TRMs, closed sup-TRMs and closed inf-TRMs, is that the expression of some genes may be significantly changed (or may not) when one or more additional factors bind to their promoter. Closed inf-TRMs are useful to identify the transcriptional role of their $TFset$ without the impact from other factors, while closed sup-TRMs can include genes that are bound by additional factors other than their $TFset$ and these additional factors may have no transcriptional role. Furthermore, taking account both closed inf-TRMs and closed sup-TRMs is also useful to identify the regulators of not only a group of genes, but also an individual gene.

We also defined two measures for a TRM r: $support(r)$ and $similar\_ratio(r)$, where $support(r)$ is the number of genes in its *geneset* ($inf\_geneset$ with inf-TRM or *sup_geneset* with sup-TRM), and $similar\_ratio(r)$ is the rate of genes in its *geneset* whose expression profiles are significantly similar (see Section 4.4). Both $support(r)$ and $similar\_ratio(r)$ are important evidences to infer if r is a real transcriptional regulatory module or not, which emphasizes that $TFset$ transcriptionally cis-regulates *geneset*.

4.4 Mining closed sup-TRMs and closed inf-TRMs

Our method for mining closed sup-TRMs and closed inf-TRMs is based on the search on the space of closed TFsets and consists of two phases. The first phase is to find the list (or lattice) of all closed TFsets with *supp* greater than a threshold $MinSup$ from the database of transcription factor binding sites (Fig. 4.2). We generate a database of "transactions" where each transaction corresponds to a gene and contains a list of transcription factors that bind to it. We then used a software library (FPClose) provided by Grahne and Zhu [28] to find all closed TFsets with $supp \geq MinSup$ in this transaction database.

The second phase concerns how to choose genes included in the *geneset* of a TRM regulated by each $closed\_TFset$. As explained above, there are two ways to generate respectively a closed sup-TRM and a closed inf-TRM. If we get *geneset* to be the maximal set of genes to which the common factors in the $closed\_TFset$ bind, we will produce a candidate closed sup-TRM with the *support* equal to the number of these genes. If we get *geneset* to be only genes to which factors in the $closed\_TFset$ and only these factors bind, we will produce a candidate closed inf-TRM with the *support* equal to the number of these genes. Our method takes account of both kinds of TRMs (Fig. 4.2). If genes in the *geneset* of a candidate TRM have significantly similar expression profiles, the TRM $r = closed\_TFset \rightarrow geneset$ will be produced.

How can we determine if a group of genes has significantly similar expression profiles? Let $E_1 = (e_{11}, e_{12}, .., e_{1m}), E_2 = (e_{21}, e_{22}, .., e_{2m}), .., E_n = (e_{n1}, e_{n2}, .., e_{nm})$ be expression vectors of n genes $E = (E_1, E_2, .., E_n)$ under m experiments (after being standardized as described in Section 4.5); some e_{ij} may be *null*. We define $E_{avr} = (a_1, a_2, .., a_m)$ as the average expression profile (or the expression center) of the group of these genes ($a_j = average_{i=1,...,n}(e_{ij}|e_{ij} \neq null)$. The distance between a gene E_i and the average expression profile (expression center) is defined as follows:

$$distance(E_i, E_{avr}) = average_{j=1,...,m}(|e_{ij} - a_j| : e_{ij} \neq null)$$

As in the work of Bar-Joseph et al. [15], we determine a suitable distance threshold T_k to infer if the expression profiles of genes in a *k-geneset* are significantly similar ($P < 0.05$) based on randomization tests. Randomization tests have been extensively used in computational biology and provide good results. We select at random k genes, compute E_{avr} for this set, and determine the distance d of the 5% closest genes that were not included in the random sampled set. This process is repeated many times (it is actually performed as a pre-processing step, for different possible sizes of k), and we set the threshold T_k to be the median d obtained in these randomization tests.

Genes in the *k-geneset* of a TRM r are said to have significantly similar expression patterns ($P < 0.05$) if their expression vectors are all in the "sphere" centered at E_{avr} with radius T_k. Unfortunately, most TRMs do not satisfy this condition due to experimental errors in the expression profiles as well as in factor DNA-binding locations. Hence we

introduce the algorithm REVISION (Table 4.1) to find the subset of genes in *k-geneset* whose expression profiles are significantly similar. The idea of this algorithm is to remove outliers (one outlier for each loop), recalculate the expression center, and set new significantly similar threshold T_k. After removing outliers, we have *k′-geneset* whose expression profiles are significantly similar. We defined the *similar_ratio* of the TRM r as the ratio $\frac{k'}{k}$.

Table 4.1: REVISION algorithm

Input	*k-geneset*; $T[2,\ldots,n]$: $T[k]$ - threshold to infer *k-geneset* to be significantly similar (see the text); $E_1,\ldots,E_k$: $E_i = (e_{ij})_{j=1,\ldots,m}$ - expression profile;
Output	*k′-geneset*: subset of genes whose expression profiles are significantly similar.

1) **do**
2) $E_{aver} = expression\_center(E_1,\ldots,E_k)$; //see the text
3) for $i = 1$ to k $d_i = distance(Ei, E_{aver})$; //see the text
4) $j = max_{i=1,\ldots,k}(d_i)$;
5) if $(d_j > T_k)$
6) Report E_j as an outlier;
7) remove E_j from the list $E_1,\ldots,E_k$;
8) **while** $(d_j > T_k)$
9) Report $E_1,\ldots,E_k$ are significantly similar;

In summary, our method can discover the two most informative kinds of TRMs: closed sup-TRMs and closed inf-TRMs. *Support* and *similar_ratio* measures of each TRM are important evidences to infer if the TRM is a real transcriptional regulatory module or not.

4.5 Datasets

The data of factor DNA-binding sites is from the work of Lee et al. [39]. This data (updated December 5, 2003) presents profiles for location analysis experiments of 113 factors. A confidence value (p-value) for each factor DNA-binding interaction is calculated by using an error model [39]. From this data, we extracted a database of "transactions", where each transaction has an unique gene identifier (`geneid`) and contains a set of factors that binds to its promoter with the confidence less than a prespecified threshold

(0.001). We excluded all transactions that contain no factors. The number of remaining transactions in the database is 2363 (equal to the number of yeast genes that were bound by at least one of 113 transcriptional factors).

We used the ExpressDB from the work of Aach et al. [66] for gene expression data in our work. This data included 17.5 million pieces of data reported by 11 studies with three different kinds of high-throughput RNA assays and under 213 conditions. The data has been standardized as Estimated Relative Abundances (ERAs). We then normalized ERAs in the interval [0,1] by a simple linear transformation.

4.6 Results and discussions

Our method was applied on the yeast data of factor DNA-binding sites and ExpressDB (see Section 4.5). It produced 405 candidate closed sup-TRMs and 157 candidate closed inf-TRMs with *support* greater than 5. Among these candidates, there are 141 closed sup-TRMs and 40 closed inf-TRMs with *similar_ratio* greater than 0.5 (see Files "sup_TRMs.htm" and "inf_TRMs.htm" respectively. All files mentioned in this section are available at "http://www.jaist.ac.jp/~h-pham/regulation"). There are 13 overlapped TRMs among them. Therefore we have 168 most informative TRMs in total. We named each found TRM by its regulators ($TFset$). Table 4.2 shows an example of closed sup-TRM regulated by TFset $HAP2\_HAP3\_HAP5$. It contains 5 genes, in which 4 genes have significantly similar expression profile. The last one YHR051W (marked by a symbol $\star$) was considered as an outlier.

Table 4.2: An example of closed sup-TRM

#module: 10			
Regs: HAP2_HAP3_HAP5	*Support:* 5	*Similar_ratio:* 0.80	
0.048	YPL207W	similarity to hypthetical proteins from A. fulgidus, M.thermoau totrophicum and M. jannaschii	
0.050	YLR220W	involved in calcium regulation	
0.053	YLL027W	mitochondrial protein required for normal iron metabolism	
0.059	YER174C	member of the subfamily of yeast glutaredoxins (Grx3, GRX4, and Grx5)	
$\star$0.105	YHR051W	cytochrome-c oxidase subunit VI	

There are 13 TRMs (see File "overlapped_TRMs.htm") overlapped between closed

inf-TRMs and closed sup-TRMs. Taking into account both closed sup-TRMs and closed inf-TRMs will help us to understand more exactly the transcriptional role of regulators. For example, Closed inf-TRM No. 9 and Closed sup-TRM No. 10 have the same TFset $HAP2\_HAP3\_HAP5$. The former contains 4 genes $YPL207W, YLR220W, YLL027W$ and $YER174C$ to which the exact 3-TFset $HAP2\_HAP3\_HAP5$ bind. All these 4 genes have significantly similar expression profiles. The latter, in addition to the 4 above mentioned genes, it includes one more gene $YHR051W$. However, this gene has been considered as an outlier because its expression profile is different from that of the remaining genes in the module. When looking at factors that bind to this gene, we found that, in addition to 3 factors $HAP2, HAP3$ and $HAP5$, there are 2 other factors $HAP4$ and $ABF1$ binding to it. Therefore, we strongly believe that these two factors make $YHR051W$ to be expressed so differently from the others. This example proves that the distinction between two kinds of TRMs (sup-TRMs and inf-TRMs) is necessary and useful to identify the transcriptional regulators of not only a group of genes but also of an individual gene.

The 27 remaining closed inf-TRMs are those not found in the list of closed sup-TRMs. This proves that when one or more additional factors bind to a gene, its expression may be changed. For example, in Closed inf-TRM No. 36 regulated by $NRG1\_CIN5\_YAP6$ (see File "inf_TRMs.htm"), there are only 6 genes that this 3-TFset binds exactly to, and all these 6 genes are significantly similar in their expression profiles. But there are up to 24 genes that share these 3 factors (see Module 319 in File "sup_TRMs_revision.htm"), and their expression profiles are not similar, since in addition to these 3 factors mentioned above, there are some other factors that also bind to some genes in the module and make them expressed so differently. Therefore closed inf-TRMs are useful to identify a group of genes transcriptionally regulated by a group of factors by avoiding the impact from other factors.

Many TRMs (both closed sup-TRMs and closed inf-TRMs) found by our method include genes whose function are related and consistent with the regulators' known roles. For example, Closed sup-TRM No. 30 regulated by $HIR1\_HIR2$ contains 7 genes, with 6 of them playing a function related to "histone". Closed sup-TRM No. 98 and Closed inf-TRM No. 34 both regulated by the same TFset $PDR1\_FHL1\_RAP1$, include genes with function mainly related to "ribosomal protein", etc. We used *GO Term Finder* tool

(http://www.yeastgenome.org/help/goTermFinder.html) to search for significant shared GO terms that are directly or indirectly associated with the genes in each TRM. To determine significance, the algorithm examines the group of genes to find GO terms to which a high proportion of the genes are associated compared to the number of times that term is associated with other genes in the genome. As a result, 29 of 40 closed inf-TRMs and 81 of 141 closed sup-TRMs have significant ontology terms other than "biological_process unknown", "molecular_function unknown" and "cellular_component unknown". Therefore, TRMs found by our method are biologically meaningful and can be useful to infer the function of uncharacterized genes.

Our method identifies not only biologically related sets of genes, but also some factors that are interacting to regulate the genes; for example, $HAP2\_HAP3\_HAP5$ (Module 10 in the list of closed sup-TRMs), $HAP2\_HAP4$ (Module 27), $HAP2\_HAP3$ (Module 28), $HIR1\_HIR2$ (Module 30), $MBP1\_SWI4$, $SWI6\_SWI4$, and other interactions can be found in some modules. Some of these interactions have been confirmed by previous studies (collected in the work of Bar-Joseph et al. [15]).

Of 168 TRMs, 26 are very similar to some gene modules previously found by GRAM in the work of Bar-Joseph et al. [15], although their genes are not exactly same (see File "comparison.htm"). Some TRMs are overlapped with gene modules from their work. There are some differences between our method and GRAM. First, our method finds candidate TRMs based on the search on the closed TFset space that is much smaller than the space of all closed TFsets but does not loss any information. Second, our method for discovering closed sup-TRMs and inf-TRMs emphasizes the transcriptional control of combinations of regulators. This is the reason why TRMs generated by our method are different from gene modules generated by GRAM [15]. For example, Gene module No. 52 from the results of GRAM regulated by $FKH1\_FKH2$ is not found by our method, because the TFset $FKH1\_FKH2$ binds up to 13 genes and the expression profiles of these genes are very different (see Closed sup-TRM No. 233 in File "sup_TRMs_revision.htm" for the revision process of our method).

4.7 Summary

In this chapter, we proposed a new approach to cluster genes into regulatory modules based on a closed TFset lattice, which consists of only non-redundant TFsets from a DNA-TF interactions. Each regulatory module is a form: $TFset \rightarrow GeneSet$, where $GeneSet$ is a group of genes regulated by the same closed subset of transcription factors $TFset$. Because the DNA-TF bindings are noisy or overpredicted, we used expression profiles to validate these regulatory modules by finding common expression patterns of genes in each module.

Our method has been applied to yeast data to find transcriptional regulatory modules (TRMs). The results are consistent with those previouly found by other methods. Moreover, TRMs found by our method are more concise and comprehensive to identify and interpret transcriptional role of regulators. In this work, we used the data of factor DNA-binding sites proved by Lee et al. [39], which were harvested from a microarray method. Each gene-factor interaction was assigned with a confidence value. In future work, we will apply our method to factor DNA-binding sites data that will be computationally predicted (i.e., from the TRANSFAC database).

Chapter 5

Conclusion

5.1 Thesis summary

The ultimate aim of life sciences is to understand the functions of all biomolecules, where proteins constitute the largest portion. Understanding structures of proteins and interactions between them and biomolecules is the first step to understand their functions. This research aimed to (1) predict and analyze turn structures in proteins, and (2) identify DNA-protein interactions relevant for transcriptional regulation and discover transcriptional regulatory patterns.

Firstly, we developed support vector machine-based method to predict turn structures in proteins from their sequence. Support vector machines (SVMs) aim at learning the global optimal hyperplane (the hyperplane with the maximum margin) to discriminate positive from negative instances in a feature space. However, using support vector machines to predict turn positions in a protein sequence is a non-trivial task because protein sequences are not in a numerical vector form. We introduced two ways for representing a protein sequence in vector form: binary representation of the single sequence and real representation of the multiple sequence alignment. By using a sliding window along the vector representation of a protein sequence, we can convert turn and non-turn positions of proteins into positive and negative vectors. Moreover, we used the optimal hyperplane learned from known data set to estimate the support of amino acids for the formation of turn structures depending on their position in a protein. The predicting and analyzing results from our SVM method appeared good performance when compared with those

from previous methods.

Statistical theory has proved that when the distribution of training data set represents the distribution of the whole data space, the optimal separating hyperplane in support vector machines has a good generalization ability, i.e., correct classification ability for unseen data. However, in biology we often deal with biased data set where distribution of known data does not hold for unseen data. For example, under different conditions and at different time, the expression of a cell is greatly different. Measuring expression levels of genome for all possible conditions is impossible, so all available expression profiles are collected at specific conditions. Therefore, we need a different strategy to uncover local patterns in parts of the known data. The *separate-and-conquer strategy* (the other name is *covering strategy*) and its extension *(weighted covering strategy)* are appropriate to uncover local rules for incomplete data.

Secondly, we developed rule induction method that combines DNA-protein interactions data with expression profiles data to uncover (1) transcription factors relevant for transcriptional expression of a gene, and (2) qualitative relationship between the expression of these relevant transcription factors and the target gene. Our rule induction method uses a weighted covering strategy with a rule evaluation heuristic that favors rules with high generality and therefore can efficiently deal with noise in the datasets. When applied to different microarray datasets obtained under different environmental conditions, our method can produce a set of environment-dependent regulatory rules comprehensively describing the relationship between the expression behavior of genes and their relevant transcription factors. The resulting regulatory rules are useful to predict transcription complexes and to reveal regulatory circuits in the regulatory networks of an organism.

However, the regulatory rules only describe some local parts of expression data in some conditions. Relationships expressed by certain datasets may be more complicated and could not be represented in a rule form. Furthermore, patterns of genes are not always directly effected by the expression profiles of the factors regulating these genes. Indeed, in many cases transcription factors are post transcriptionally modified, and consequently we cannot examine their protein levels. Therefore, transcriptional regulatory rules may not capture some regulatory patterns.

Finally, we described a different approach to uncover regulatory patterns other than

regulatory rules. The approach finds an expression profile pattern of a group of genes commonly regulated by the same set of transcription factors. We developed a new method that first clusters genes into modules based on a closed TFset lattice, which includes non-redundant combinations of transcription factors respective to a database of DNA-protein interactions; and then validate the expression profiles to confirm regulatory modules. Our method produces transcriptional regulatory modules (TRMs), which are compact, concise and comprehensive to identify and interpret the transcriptional control of combinations of regulators.

5.2 Future work

The work in this thesis is a part of our long-term project, which aims at computationally analyzing biomolecular interactions. There are two main problems we would like to solve. First, from interaction databases (such as DNA-protein, RNA-protein, protein-protein interactions) collected by experimental approaches, which are often very noisy and only indicate physical binding but not function, we would like to identify actual and functional interactions. In the Chapter 3 of this thesis, we have developed a rule induction method to recognize actual DNA-protein interactions that regulate gene transcription. We also presented a new clustering approach in Chapter 4 to uncover regulatory modules, where reveal the functions of DNA-protein interactions. In the future, we will continue the work of identifying actual interactions from other types of data such as RNA-protein, protein-protein bindings; and discovering which kinds of functions these interactions play.

The second problem is whether we can predict interactions of a protein with other biomolecules? This is an extremely complex problem. We expect to improve our turn prediction method (presented in Chapter 2 of this thesis) to predict three-dimensional fold of a protein. This 3D fold will be the key information to understand and predict interaction sites of the protein with other biomolecules.

Appendix A

Support vector learning

A.1 Methods of separating hyperplanes

A.1.1 A simple classification problem

Given a set of training examples, i.e. pairs of patterns x_i and labels y_i,

$$(x_1, y_1), \ldots, (x_l, y_l) \in \mathbb{R}^N \times \{\pm 1\}, \tag{A.1}$$

the goal of a classification system is to find some decision function $g : \mathbb{R}^N \rightarrow \{-1, 1\}$ that accurately predicts the labels of unseen data points (x, y). A common approach to representing decision functions is to use hyperplanes (linear functions $f(x) = w.x + b$, Fig. A.1) with "good" statistical properties to separate examples belonging to class "+1" from those belonging to other (class "-1"). The problem of learning from data can be formulated as finding a set of parameters (w, b) such that $sgn(w.x_i + b) = y_i$ for all $1 \leq i \leq l$. In the rest of this section we assume that the training data is linearly separable without noise.

A.1.2 Perceptron for finding a separating hyperplane

At the end of the 1950s, F. Rosenblatt proposed a machine (Perceptron) for learning a hyperplane separating two classes from the training data. The Perceptron utilitizes a recurrent procedure (Fig. A.2) for learning the function (the coefficents w and b). The algorithm only updates (w, b) on a labelled example if the Perceptron misclassifies the

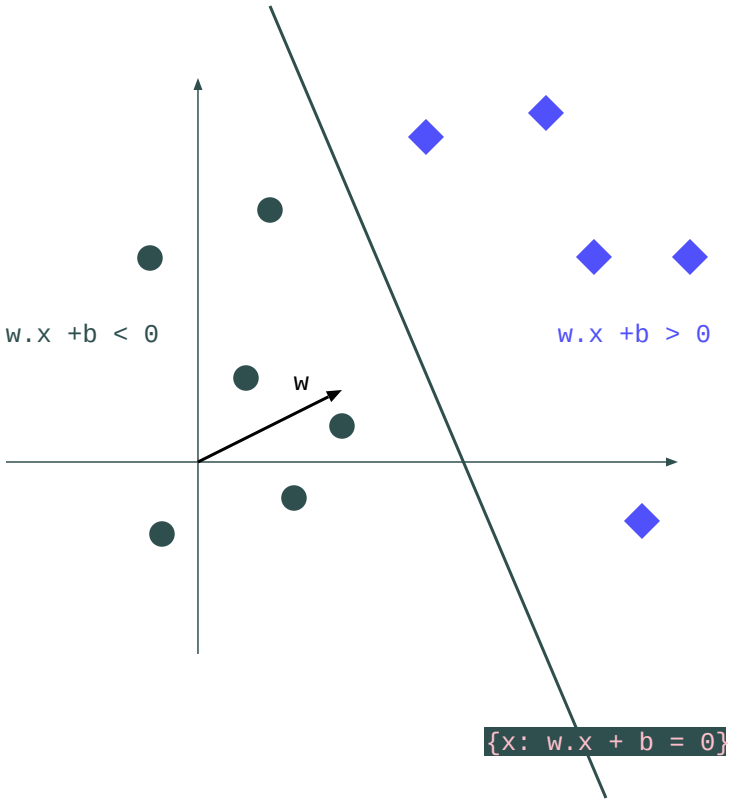


Figure A.1: Hyperplane separating positive and negative examples

example.

Figure A.2: Perceptron algorithm

Input	Training data $\{(x_1, y_1), \ldots, (x_l, y_l)\} \subseteq (\mathbb{R} \times \{0, 1\})^n$;
Output	Weight vector w and threshold b.

1) **initialize** $w, b = 0$
2) **repeat**
3) **for** $i = 1$ **to** l
4) **if** $y_i(w.x_i + b) < 0$
5) $w = w + y_i x_i$
6) $b = b + y_i$
7) **until** for all $1 \leq i \leq l$": $y_i(w.x_i + b) > 0$
8) **return** $f(x) = w.x + b$

Theorem 1. *suppose that there exists $\rho > 0$, a weight vector w^* satisfying $\|w^*\| = 1$, and a threshold b^* such that*

$$y_i(w^* x_i + b^*) \geq \rho \quad \textit{for all } 1 \leq i \leq l$$

. *Then Perceptron converges after no more than* $(b^{*2}+1)(R^2+1)/\rho$ *updates, where* $R = max_i\|x_i\|$.

A.1.3 Optimal hyperplane and support vectors

Definition 1. *Denote by* $f(x) = w.x + b$ *a linear function used for classification. Then*

$$\rho_f(x,y) := \frac{y(w.x+b)}{\|w\|}$$

is the margin (distance) by which the pattern x *is classified correctly (a negative value of* $\rho_f(x,y)$ *corresponds to an incorrect classification). Also denote by*

$$\rho_f := \min_{1\le i\le l} \quad 2\rho_f(x_i,y_i)$$

the minimum margin over the whole sample, determined by the "worst" classification on the whole training set $X \times Y$. *Fig. A.3 shows a way to calculate the margin of a training set respective to a hyperplane* $w.x + b$.

It would be desirable to have classifiers that achieve a large margin ρ_f (*the optimal hyperplane*) since we might expect that an estimate that is "reliable" on the training set will also perform well on unseen examples. This raises the question of whether there exists an estimator with maximum margin, i.e., whether there exists some f^* such that

$$f^* := argmax_f \quad \rho_f = argmax_f \min_i \quad \frac{y_i f(x_i)}{\|w\|}$$

This problem can be easily transformed into an equivalent constrained optimization task by conjecturing a lower bound on the margin, ρ, and maximizing ρ subject to the constraint that it actually is a lower bound:

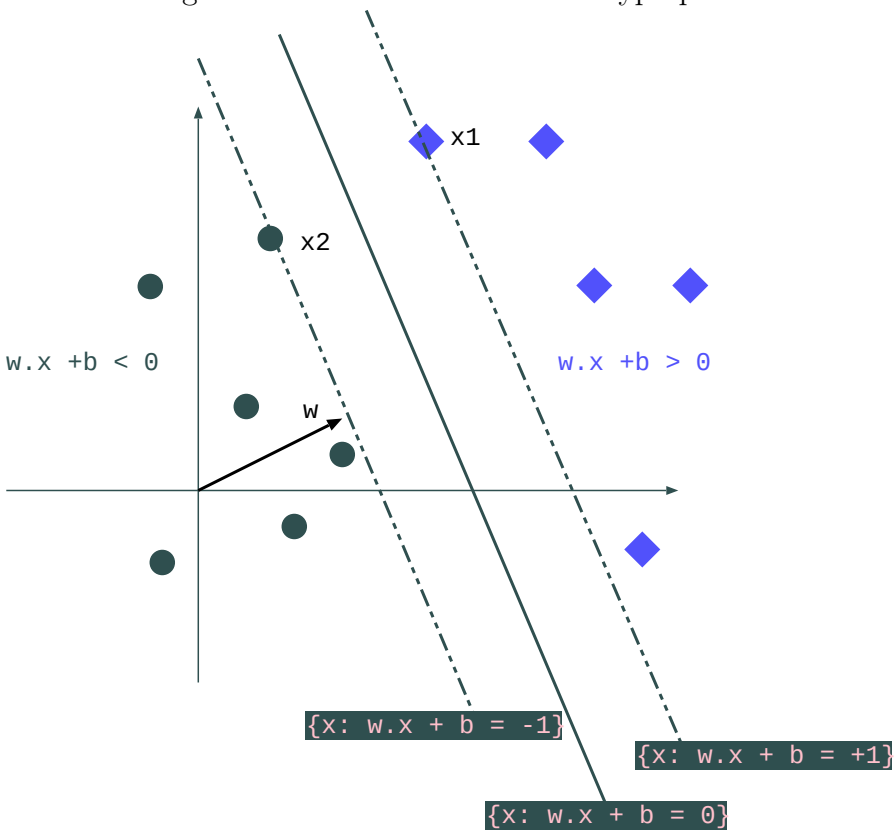
$$w^*, b^*, \rho^* \tag{A.2}$$

$$= \quad argmax_{w,b,\rho}\rho \ \text{ subject to } \frac{y_i(w.x_i+b)}{\|w\|} \ge \rho \text{ for } 1 \le i \le l \tag{A.3}$$

$$= \quad argmax_{w,b,\rho}\rho \ \text{ subject to } \|w\| = 1, \ \ y_i(w.x_i+b) \ge \rho \text{ for } 1 \le i \le l \tag{A.4}$$

$$= \quad argmin_{w,b}\|w\|^2 \ \text{ subject to } y_i(w.x_i+b) \ge 1 \text{ for } 1 \le i \le l \tag{A.5}$$

Figure A.3: Canonical form of a hyperplane



By requiring the scaling of w and b to be such that the point(s) x_i nearest to the hyperplane satisfy $\|w.x_i + b\| = 1$, we obtain a *canonical* form of a hyperplane. In this case, the margin, measured perpendicularly to the hyperplane equals $2/\|w\|$, which can also be obtained by considering two opposite points which precisely satisfy $\|w.x_i + b\| = 1$.

This last formulation is in the form of a quadratic programming, which can be easily handled using standard optimizers [68, 60]. One way is to introduce Lagrange multipliers $\alpha_i \geq 0$ and a Lagrangian

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_i \alpha_i \{y_i(x_i.w + b) - 1\} \tag{A.6}$$

The Lagrangian L has to be minimized with respect to the *primal variables* w and b and maximized with respect to the *dual variables* α_i. The solutions w^*, b^*, and α^* should satisfy the following conditions

$$\frac{\partial L(w^*, b^*, \alpha^*)}{\partial b} = 0 \tag{A.7}$$

$$\frac{\partial L(w^*, b^*, \alpha^*)}{\partial w} = 0 \tag{A.8}$$

Rewriting these equations in explicit form we obtain the following properties of the optimal hyperplane:

1. The coefficients α_i^* for the Optimal hyperplane should satisfy the constraints

$$\sum_{i=1}^{l} \alpha_i^* y_i = 0, \quad \alpha_i^* \geq 0, \quad i = 1, \ldots, l$$

2. The optimal hyperplane (vector w^*) is a linear combination of the vectors of the training set.

$$w^* = \sum_{i=1}^{l} y_i \alpha_i^* x_i, \quad \alpha_i^* \geq 0, \quad , i = 1, \ldots, l \tag{A.9}$$

Introducing the expression for w into the Lagrangian, we obtain the dual problem

$$\alpha^* = argmax_\alpha \quad = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j}^{l} \alpha_i \alpha_j y_i y_j (x_i . x_j) \tag{A.10}$$

$$\alpha_i \geq 0, \quad i = 1, \ldots, l \tag{A.11}$$

$$\sum_{i=1}^{l} \alpha_i y_i = 0 \tag{A.12}$$

Thus, to construct the optimal hyperplane, we should solve a simple quadratic programming problem: maximize the quadratic function A.10 with linear constrains A.11 and A.12. Let $\alpha^* = (\alpha_1^*, \ldots, \alpha_l^*)$ be a solution to this dual problem, then the parameter w^* of the optimal hyperplane can be calculated by A.9. Moreover, from Kuhn-Tucker theorem it follows that the solution α^* of the dual problem and the solution w^*, b^* of the primal problem must satisfy the conditions:

$$\alpha_i^* \{ y_i (x_i . w^* + b^*) - 1 \} = 0, \quad i = 1, \ldots, l. \tag{A.13}$$

Therefore, with each i such that $\alpha_i^* > 0$ we have $y_i(x_i.w^* + b^*) - 1 = 0$. These vectors x_i are called *support vectors*. The formula for calculating the parameter w^* of the optimal hyperplane is therefore

$$w^* = \sum_{\text{support vectors}} y_i \alpha_i^* x_i, \quad \alpha_i^* > 0 \tag{A.14}$$

The threshold b^* of the optimal hyperplane is then calculated by the formula:

$$b^* = \frac{1}{2}[w^*.x^*(1) + w^*.x^*(-1)] \tag{A.15}$$

where $x^*(1)$ denotes a (any) support vector belonging to the first class and $x^*(2)$ denotes for a support vector belonging belonging to the other class [74].

A.2 Statistical theory of the Optimal hyperplane

In this section, we will discuss why the optimal hyperplane is good to predict unseen examples. We first provide some basic concepts and results of statistical learning theory.

A.2.1 Statistical learning theory

Given a set of training examples as in A.1, i.e. pairs of patterns x_i and labels y_i ($i = 1, \ldots, l$), each one of them is generated from an unknown probability distribution $P(x, y)$ containing the underlying dependency. We want to learn a function f_{α^*} from a set of functions

$$\{f_\alpha : \alpha \in \Lambda\}, \quad f_\alpha : \mathbb{R}^N \to \{\pm 1\} \tag{A.16}$$

which provides the smallest possible value for the average error committed on independent examples randomly drawn from the same distribution P, called the *risk*

$$R(\alpha) = \int \frac{1}{2} \| f_\alpha(x) - y \| dP(x, y). \tag{A.17}$$

Unfortunately we cannot evaluate the risk because we do not know the probability distribution function $P(x, y)$. However, we can estimate the risk over the training set (called

the *empirical risk*)

$$R_{emp}(\alpha) = \frac{1}{l}\sum_{i=1}^{l}\frac{1}{2}\|f_\alpha(x_i) - y_i\|, \tag{A.18}$$

and the risk is bounded by the following inequality [74]:

$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\frac{h(log\frac{2l}{d}+1) - ln(\frac{\delta}{4})}{l}} \tag{A.19}$$

for any $\alpha \in \Lambda$ and $l > h$, with a probability of at least $1-\delta$, where h is the *VC dimension* of the function set. The VC dimension h is the maximum number of data points which can be separated into two classes in all possible 2^h ways by using functions in the function set, i.e. for each possible separation there exists a function which takes the value 1 on one class and -1 on the other class.

From the formulation in A.19, Vapnik [72] developed an *induction principle* for risk minimization: *structural risk minimization*. Given a fixed number l of training examples, we can control the risk by two quantities: $R_{emp}(\alpha)$ and $h(\{f_\alpha : \alpha \in \Lambda'\})$, where Λ' denotes some subset of the index set Λ. To control h, we introduces a structure of nested subsets $S_n := \{f_\alpha : \alpha \in \Lambda_n\}$ of $\{f_\alpha : \alpha \in \Lambda\}$,

$$S_1 \subset S_2 \subset \ldots \subset S_n \subset \ldots, \tag{A.20}$$

whose VC-dimensions, as a result, satisfy

$$h_1 \leq h_2 \leq \ldots \leq h_n \leq \ldots \tag{A.21}$$

The *structural risk minimization principle* chooses the function $f_{\alpha_i^n}$ in the subset $\{f_\alpha : \alpha \in \Lambda_n\}$ for which the guaranteed risk bound (the right hand of A.19) is minimal (Fig. A.4). The procedure of selecting the right subset for a given amount of observations is referred to as *capacity control*.

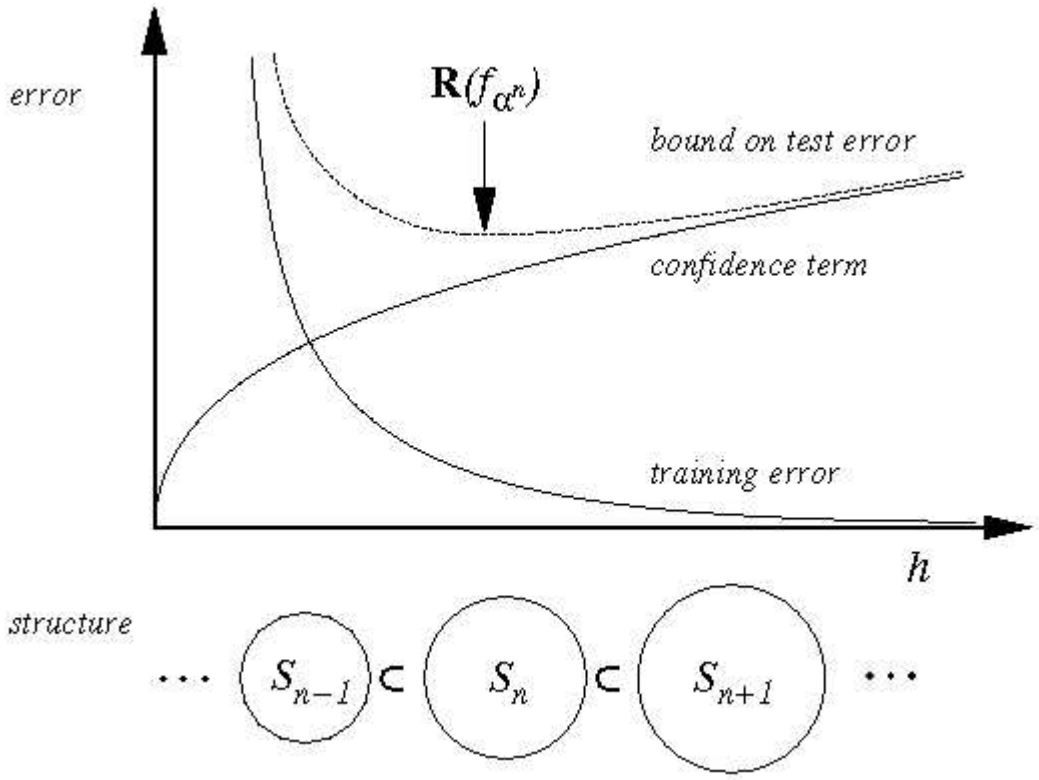


Figure A.4: Structural risk minimization principle

A.2.2 VC-dimension of some function sets

- *Set of linear functions:* the VC-dimension of the set of linear functions

$$Q(z, \alpha) = \sum_{p=1}^{n} \alpha_p z_p + \alpha_0, \quad \alpha_0, \ldots, \alpha_n \in \mathbb{R}$$

  in n-dimensional coordinate space $Z = (z_1, \ldots, z_n)$ is equal to $h = n + 1$, since by using functions from this set we can shatter at most $n + 1$ vectors.

- *Set of ρ-margin separating hyperplanes:* we call a hyperplane

$$w.x - b = 0, \quad \|w\| = 1$$

the ρ-margin separating hyperplane if it classifies vectors x as follows

$$y = \begin{cases} 1 & \text{if} \quad w.x - b \geq \rho \\ -1 & \text{if} \quad w.x - b \leq -\rho \\ undefined & \text{if} \quad -\rho < w.x - b < \rho \end{cases}$$

Theorem 2. *Let vectors $x \in X$ belong to a sphere of radius R, then the set of ρ-margin separating hyperplanes has VC dimension h bounded by the inequality*

$$h \leq min\left(\left[\frac{R^2}{\rho^2}\right], n\right) + 1 \tag{A.22}$$

We see that in general the VC-dimension of the set of hyperplanes equals $n+1$, where n is the dimensionality of the input space. However, the VC-dimension of the set of ρ-margin separating hyperplanes (with a large value of margin ρ) can be less than $n+1$. This fact, together with constraints A.4 and A.19, explains why the optimal hyperplane has often a high generalization ability (i.e. a small error on an independent test set).

A.3 Optimal hyperplane with kernels: support vector machines

A.3.1 Optimal hyperplane in linearly non-separable data

When the training examples are linearly non-separable (due to the presence of noise in the data), we introduce non-negative variables $\xi_i \geq 0$ and a penalty C for each missclassified example. The quadratic problem A.5 can be reformulated then as

$$w^*, b^* = argmin_{w,b,\xi}\|w\|^2 + C\sum_{i=1}^{l}\xi_i \quad \text{subject to } y_i(w.x_i + b) \geq 1 - \xi_i \text{ for } 1 \leq i \leq l. \tag{A.23}$$

Using the same formalism with Lagrange multipliers, we can obtain that the optimal hyperplane also has an expansion A.9 on support vectors. The coefficients α_i can be found by maximizing the same quadratic form as in the separating case A.10 under slightly

different constraints

$$0 \leq \alpha_i \leq C, \;\; i = 1, \ldots, l, \;\; \text{and} \;\; \sum_{i=1}^{l} \alpha_i y_i = 0. \tag{A.24}$$

A.3.2 Optimal hyperplane with kernels

To find a non-linear function that separates two classes, we can implement the following idea [73]: map the input vectors x_i into a higher dimensional feature space $\phi(x_i)$ through some non-linear mapping previouly chosen, ϕ; then construct an optimal separating hyperplane in this space.

The optimal hyperplane in the feature space is the solution of the quadratic optimization problem

$$\min_{w,b,\xi} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l} \xi_i \tag{A.25}$$

$$y_i\{w.\phi(x_i) + b\} \geq 1 - \xi_i \tag{A.26}$$

$$\xi_i \geq 0, i = 1, ..., l. \tag{A.27}$$

Its dual is a quadratic optimization problem:

$$\min_{\alpha} \quad \frac{1}{2}\alpha^T Q\alpha - e^T\alpha \tag{A.28}$$

$$0 \leq \alpha_i \leq C \tag{A.29}$$

$$y^T\alpha = 0 \tag{A.30}$$

where e is the vector of all ones; $C > 0$ is an error penalty parameter, $y = \{y_i\}_{i=1,..,l}$, $Q_{ij} = y_i y_j K(x_i, x_j)$, $K(x_i, x_j) = \phi(x_i)^T\phi(x_j)$ is a kernel function; and $\phi(x_i)$ maps x_i into a higher (maybe infinite) dimensional space. So $K(x_i, x_j)$ is a symmetric positive definite function that reflects the similarity between the sample x_i and the sample x_j. In our research, we employed a linear function $K(x_i, x_j) = x_i.x_j$ and radial basis function (RBF) $K(x_i, x_j) = \exp(-\gamma(x_i - x_j)^2)$ as the kernel functions. The SVM classification function, after trained, has the following form:

$$f(x) = \sum_i \alpha_i y_i K(x, x_i) + b \tag{A.31}$$

where $\alpha = \{\alpha_i\}_{i=1,..,l}$ is the solution of the above dual problem and b is in the solution of the prime problem. Based on the Karush-Kuhn-Tucker theory, the solution of the prime problem and that of its dual satisfy the following equation:

$$\alpha_i\{y_i(w^T\phi(x_i) + b) - 1 + \xi_i\} = 0.$$

Therefore, if there is an i such that $\alpha_i \neq 0$, then $y_i(w^T\phi(x_i) + b) - 1 + \xi_i = 0$. In this case, x_i is called a "support vector".

Appendix B

Descriptive rule induction

B.1 Descriptive and predictive rule induction

Rule induction from examples is a machine learning technique that has been successfully used as a support tool for knowledge acquisition and prediction. The induced rules are usually expressed as $condition \rightarrow class$, where *condition* and *class* are logic expressions of the form $(variable_1 = value_1 \wedge variable_2 = value_2 \wedge \ldots \wedge variable_k = value_k)$.

There are three kinds of rule inducting algorithms: covering, decision tree-based and association rule-based. The first ones, covering algorithms, make use of a *separate-and-conquer* strategy over the search space to learn a rule set (see [65] for an overview). This *separate-and-conquer* strategy searches for a rule that explains (covers) part of its training instances, separates (or reassigns with lower weight) these examples, and recursively conquers the remaining examples by learning more rules until no examples remain. Decision tree-based algorithms use a *divide-and-conquer* strategy [69, 70]. Much of the popularity of these algorithms stems from their efficiency in learning and classification. Decision trees can be easily turned into a rule set by generating one rule for each path from the root to a leaf. Finally, association rule-based algorithms use a *exhaustive search* strategy by exploring almost the whole search space [56, 40]. The basic idea is to use an association rule algorithm to gather all rules that predict the class attribute and also pass a minimum quality criterion.

By implementation, the divide-and-conquer strategy (in decision tree-based algorithms) is restricted to learn non-overlapping rules only. The exhaustive strategy (in association

rule-based algorithms) has the problem of producing many redundant rules. The separate-and-conquer algorithms can partially avoid these disadvantages [65, 3], which is one of the main reasons for its popularity.

CN2 is a rule induction system implementing the separate-and-conquer strategy [54, 48]. It was originally designed to solve classification and prediction tasks.

Recently, it has also been adapted for description tasks (CN2-SD) by using a weighted covering strategy (an extension to the separate-and-conquer strategy in CN2), combined with a rule evaluation heuristic (weighted relative accuracy [3]) that favors rules with higher generality. While the weighted covering strategy tends to find rules that explain overlapped subgroups of instances in the search space, the weighted relative accuracy heuristic produces highly general rules that express the knowledge contained in one specific subgroup.

Rule evaluation measures are the heart of rule learning systems. These evaluation measures are used as a rule searching heuristic, as well as for filtering out uninteresting rules and/or as a stopping criterion of the refinement process. Many measures have been proposed in the literature: accuracy, entropy [54], Laplace [48], m-estimate [62], weighted relative accuracy [3], etc. However, deciding which one is the best suited measure, especially for description tasks, is still an open problem [45, 44].

Furnkranz and Flach [42] have introduced a ROC-like tool, called *PN-space* (also *PN-graph* or *coverage space*) for analysis, evaluation and visualization of a set of rules. They showed that there is a surprising number of equivalences and similarities among commonly-used evaluation measures [42, 43, 44].

In this chapter, we use PN-space to analyze the generality of rules produced by CN2-SD with different rule evaluation heuristics and the appropriateness of these heuristics for description tasks or knowledge discovery, especially in noisy domains.

The rest of this chapter is organized as follows: section B.2 presents a formal definition of PN-space and isometrics of a rule evaluation measure. Section B.3 analyzes two algorithms by visualizing them in PN-space: the separate-and-conquer algorithm to learn a rule set and general-to-specific algorithm to learn a single rule. Section B.4 presents our main contribution, where we analyze the generality of rules produced by a general-to-specific search with different heuristics. Section B.5 will describe the differences between

CN2 and CN2-SD that are specially relevant for description tasks. Some experiments on 18 UCI datasets and microarray data are conducted to illustrate our analysis on section B.6. We finish by presenting some concluding remarks.

B.2 PN-space and isometrics

B.2.1 PN-space

Let P and N be the total number of positive and negative examples in a training set, while $p(r)$ and $n(r)$ denote the respective number of examples covered by a rule r. Measures of the rule r are two-dimensional functions of the form $h(p(r), n(r))$. For clarity, we will abridge $h(p(r), n(r))$ as $h(r)$, and omit the argument (r) from functions p, n, and h when it can be clearly deduced from the context.

Definition 2. ***PN-space (coverage space)*** *is a two-dimensional space of points (n, p), where $0 \leq n \leq N$ denotes the number of negative examples covered by a rule (false positives) and $0 \leq p \leq P$ denotes the number of positive examples covered by a rule (true positives).*

PN-space is quite similar to ROC space, a two-dimensional plane in which the operating characteristics of classifiers are visualized [43]. Each rule is represented by a point in the PN-space, with the point $(0, P)$ (corresponding to the rule that covers all positive examples and none of the negative) being an ideal point that every learning system tries to reach.

Definition 3. ***Compatible****: two measure h_1 and h_2 are compatible iff for all rules r, s: $h_1(r) > h_1(s) \Leftrightarrow h_2(r) > h_2(s)$*

Definition 4. ***Antagonistic****: two measure h_1 and h_2 are antagonistic iff for all rules r, s: $h_1(r) > h_1(s) \Leftrightarrow h_2(r) < h_2(s)$*

Definition 5. ***Equality-preserving****: two measure h_1 and h_2 are equality-preserving iff for all rules r, s: $h_1(r) = h_1(s) \Leftrightarrow h_2(r) = h_2(s)$*

Definition 6. ***Equivalence****: two measure h_1 and h_2 are equivalent ($h_1 \equiv h_2$) if they are either compatible or antagonistic, i.e. they order all rules in the same way.*

B.2.2 Isometrics

Definition 7. ***Isometrics****: an isometrics of a measure h is a line (or curve) in PN-space that connects, for some value c, all points for which $h(p,n) = c$.*

Fig. B.1 presents some examples of isometrics for some basic measures: accuracy h_{acc}, weighted relative accuracy h_{wra}, entropy h_{en}, and m-estimate h_m. The measures are defined as follows:

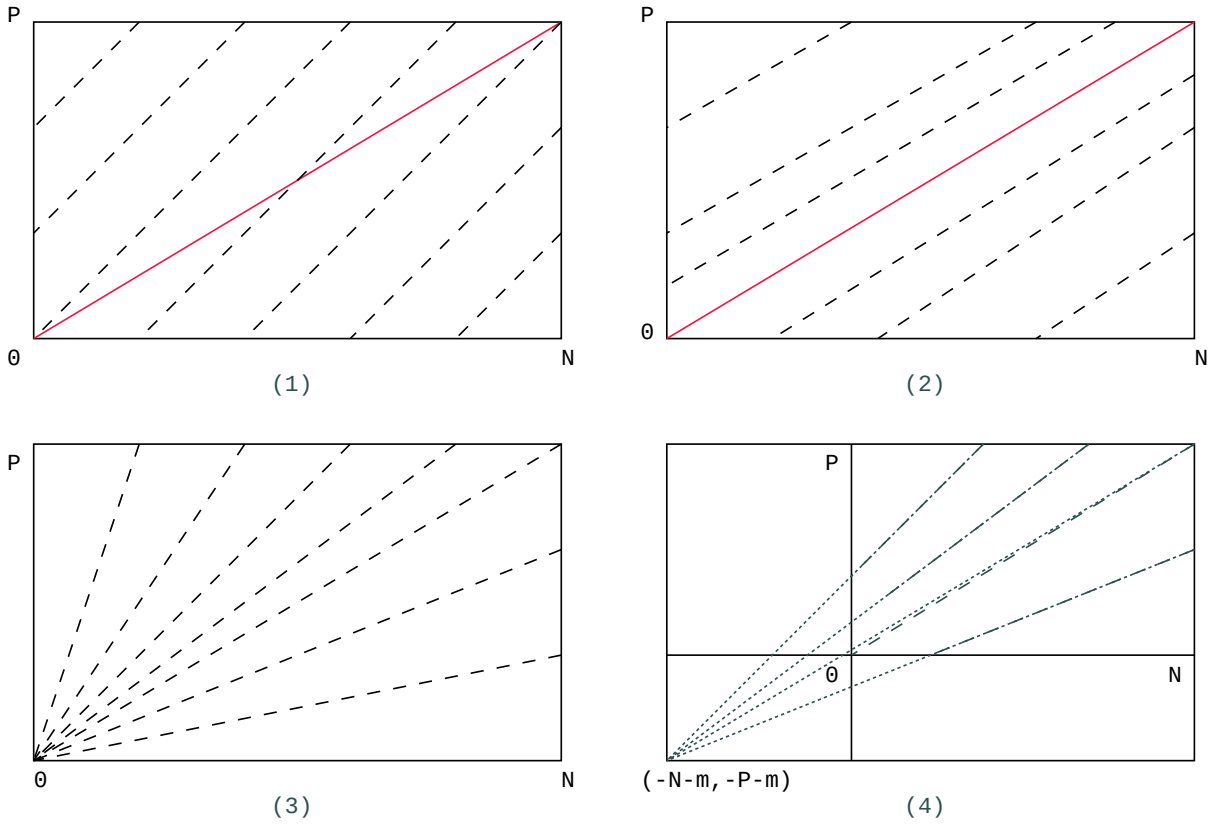


Figure B.1: Isometrics of h_{acc}, h_{wra}, h_{en}, and h_m.

$$h_{acc} = p - n \tag{B.1}$$

$$h_{wra} = \frac{p+n}{P+N}\left(\frac{p}{p+n} - \frac{P}{P+N}\right) \tag{B.2}$$

$$h_{en} = -\left(\frac{p}{p+n} log_2 \frac{p}{p+n} + \frac{n}{p+n} log_2 \frac{n}{p+n}\right) \tag{B.3}$$

$$h_m = \frac{p + m\frac{P}{P+N}}{p + n + m} \tag{B.4}$$

The isometrics of some other existing measures can be found in [44]. We propose two kinds of rule evaluation heuristics: h_{k_linear} (Eq. B.5) and $h_{P_1N_1\_quad}$ (Eq. B.6), which have linear and quadratic (non-linear) isometrics respectively.

$$h_{k\_linear}(r) = p - kn = n(\frac{p}{n} - k) \tag{B.5}$$

$$h_{P_1N_1\_quad}(r) = \frac{1}{\frac{(p-P)^2}{P_1^2} + \frac{n^2}{N_1^2}} \tag{B.6}$$

B.3 Rule learning in PN-space

B.3.1 Learning a rule set vs. learning a single rule

The separate-and-conquer algorithm learns a rule set by iteratively adding one rule at a time. The algorithm starts by performing a general-to-specific search (described below) to learn the best rule according to some measures. Examples covered by this rule are separated (or their weight is lowered) before learning the next rule. This is repeated until all examples are covered by at least one rule in the rule set or some stopping criteria are satisfactory.

The general-to-specific algorithm starts with a default rule (the rule that classifies all examples to be positive), and then searches the space of possible rules by successively specializing the current best rule. Rules are specialized by greedily adding the condition which promises the highest gain according to some evaluation measures (heuristics). We will analyze these rule search heuristics in the following sections.

B.3.2 PN-space path of learning a rule set vs. path of learning a single rule

Learning a rule set one rule at a time (separate-and-conquer algorithm) can be viewed as a path through PN-space, where each point on the path corresponds to an extra rule

added to the rule set. This path starts at (0,0), which corresponds to an empty rule set not covering any examples. Adding a rule makes the path "move" to a new point in the PN-space, which corresponds to a theory consisting of all rules that have been accumulatively learned so far. After the final rule has been learned, we can add another rule with the body that is always true to cover all remaining examples, which have not covered by any previous rules. This process, that starts at point (0,0) will eventually take us to the point (N,P). Fig. B.2.1 shows the coverage path for learning a specific rule set.

While learning a rule set (separate-and-conquer algorithm) corresponds to a coverage path from the point (0,0) to (N,P), learning an individual rule (general-to-specific algorithm) in each iteration of the learning process corresponds to a path from (N,P) toward down to the point (0,0). Learning a single rule, as explained above, is a general-to-specific search. From the default rule (rule with body that is always true), we move by adding the condition that is expected to get the highest gain according to specific measures, which will take us to a "better" point in the PN-space. Fig. B.2.2 shows the coverage path for learning a single rule by general-to-specific procedure. The best rule that would be added into the rule set corresponds to the "highest" point in the path.

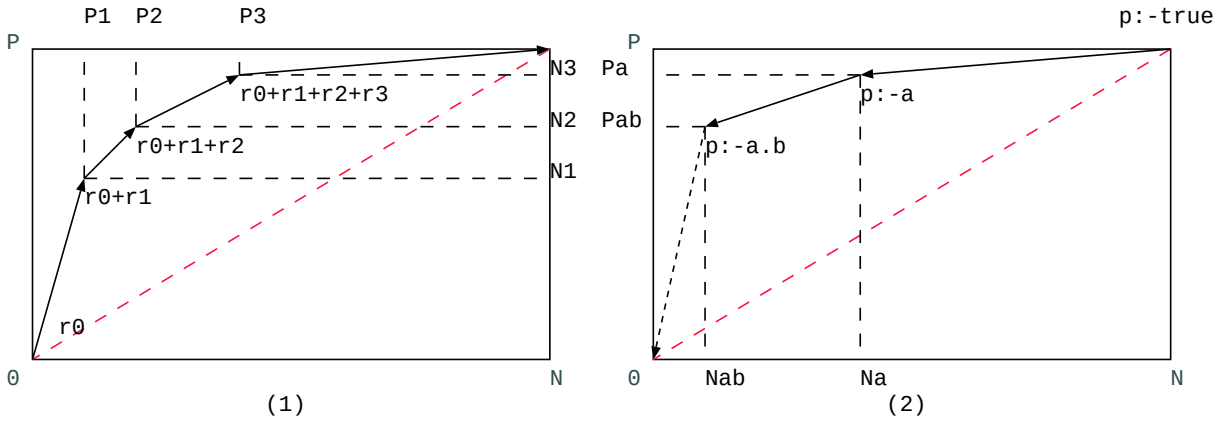


Figure B.2: Coverage path of learning a rule set (1) and learning a single rule (2).

B.4 Control of the generality of learned rules

During the learning process of an individual rule, the general-to-specific algorithm must decide which is the best available rule from candidate ones. Many measures (heuristics)

have been proposed for this purpose as well as for filtering out uninteresting rules and/or for stopping the refinement process at an appropriate point. Some of these measures produce compact rules, while others produce more specific rules. In this section, we will analyze the generality of learned rules using different heuristics and study what factors control the generality of these rules when working in a PN-space.

The generality of a rule r ($gen(r)$) is measured by the number of covered examples; and the generality of a rule set is evaluated by three measures: the number of rules ($\#rules$), the average number of conditions per a rule ($\#conds$), and the average generality of rules in this rule set ($avg\_gen(r)$).

Definition 8. ***Coverage space of a rule*** *r(n,p)****, (pn-space)*** *is a subspace of PN-space that includes all points (n_1, p_1), where $0 \leq n_1 \leq n$ and $0 \leq p_1 \leq p$.*

Definition 9. ***(descendant and ancestor)****: A rule $r_1(n_1, p_1)$ is a descendant of a rule $r(n, p)$, if r_1 is the result of adding one or more conditions to the rule r. In this case, rule r is called an ancestor of r_1*

We can easily see that if rule $r_1(n_1, p_1)$ is a descendant of a rule $r(n, p)$, the point (n_1, p_1) corresponding to r_1 in PN-space must be in the coverage space of r (or pn-space, see Fig. B.3), i.e., $0 \leq n_1 \leq n$ and $0 \leq p_1 \leq p$. The point $(0, p)$ in the pn-space corresponds to the best specific rule of r, where the general-to-specific procedure tries to reach. Points in the main diagonal of the pn-space correspond to rules r_1 with class distribution similar to the class distribution of r.

Definition 10. ***h-beam search space of a rule*** *$r(n, p)$* ***($bss_h(r)$)*** *(respective to an heuristic h) is the area of the coverage space of r that includes all points (n_1, p_1), where $h(n_1, p_1) > h(n, p)$.*

In other words, $bss_h(r)$ is the area of the coverage space of r that includes points above the h-isometric cutting through the point (n, p). Fig. B.3 presents the beam search space of a rule respective to some heuristics: entropy h_{en}, m-estimate h_m, and weighted relative accuracy h_{wra}.

Theorem 3. *for every rule $r(n, p)$: $bss_{h_{wra}}(r) \subseteq bss_{h_m}(r) \subseteq bss_{h_{en}}(r) = \frac{coverage\_space(r)}{2}$ with $m > 0$ (Fig. B.3).*

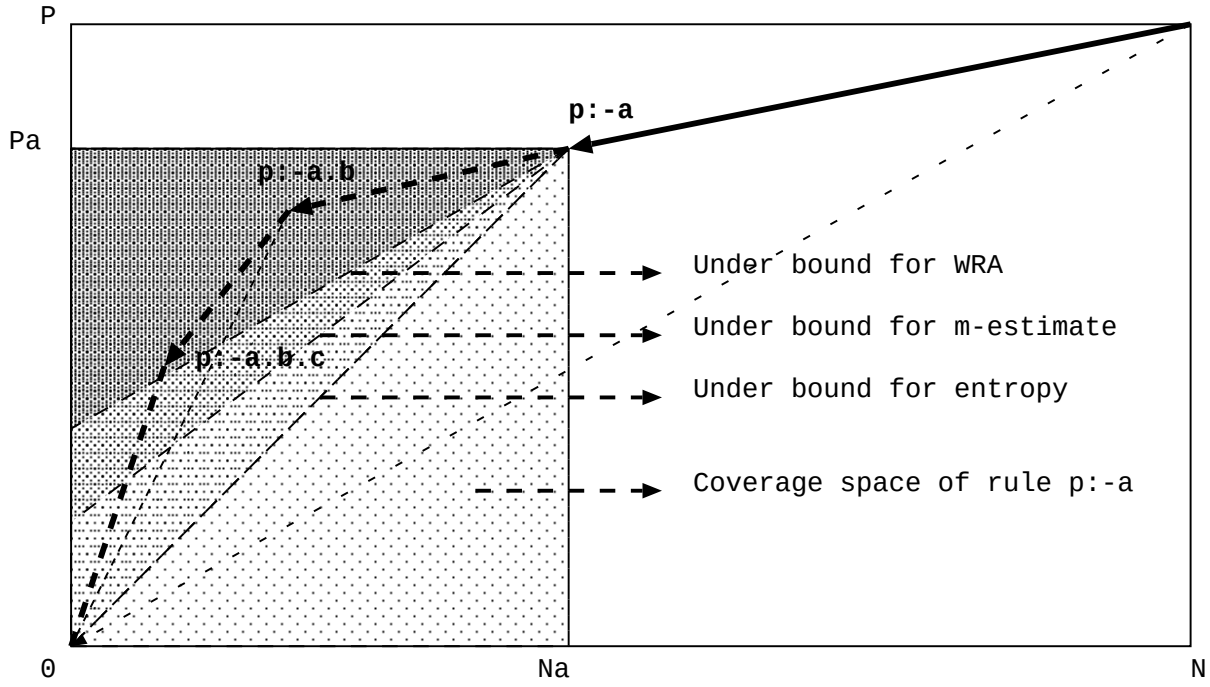


Figure B.3: Beam search space of rule $p \leftarrow a$ with different heuristics: entropy (en), m-estimate and weighted relative accuracy (WRA)

The rate between the area of the h-beam search space and that of the coverage space of r is an important factor to control the generality of learned rules as well as the generality of the learned rule set. The general-to-specific algorithm iteratively specializes the current best rule $r(n, p)$ by adding the condition that would produce the best rule according to heuristic h, i.e. it searches if there is a descendant of the best current rule $r(n, p)$ in the h-beam search space of r. If there is, this descendant will replace the current best rule $r(n, p)$. Therefore, the smaller the rate between the area of the h-beam search space and that of the coverage space of r, the lower the possibility that the specification is successful, and the more general the learned rule, as well as the sooner the procedure stop specialization process.

From Theorem 3, it can be seen that $bss_{h_{wra}}$ is smaller than bss_{h_m} and much smaller than $bss_{h_{en}}$, which explains why h_{wra} often produces too general rules [3]. With entropy heuristic h_{en} (or the equivalent accuracy heuristic h_{acc}), the area of the beam search space is relatively large (a half of the area of the coverage score of a rule), and the produced rules are often too specific [48]. The m-estimate heuristic h_m with $m = 0$ is equivalent to h_{en}; if $m \to \infty$, it becomes equivalent to h_{wra}. Therefore, the parameter m can control

the generality of learned rules.

To see how the different heuristics impact the generality of learned rules, we illustrate in Fig. B.4 the best learned rule with two heuristics entropy h_{en} and m-estimate h_m. There are two general-to-specific pathways ($beam - size = 2$). With the rule measure h_{en} (Fig. B.4.1), the best rules for each pathway are $p \leftarrow a.b.c$ and $p \leftarrow x.y$. The first one is better than the second, and in fact it is the best of all rules in any of the two pathways. Rule $p \leftarrow a.b.c$ will be added to the rule set. When we used m-estimate h_m for the rule evaluation measure instead of h_{en}, the best rule was $p \leftarrow a.b$ (Fig. B.4.2).

B.5 CN2 and CN2-SD: from prediction to description tasks

CN2 is a rule induction system implementing the separate-and-conquer strategy [54, 48]. It learns a rule set by iteratively adding rules one at a time. The system starts by using a general-to-specific search (described below) to learn the best rule according to some measures. Examples covered by this rule are removed from the search space before learning the next rule to add to the rule set. This is repeated until all examples are covered by at least one rule in the set or some stopping criteria is satisfied.

The original CN2 [54] works by learning a set of ordered rules. According to a certain heuristic measure, CN2 looks for the best rule in the set of training examples. Once a rule is found ("induced"), all examples covered by the induced rule are removed from the training set, and the system starts again to look for a new rule. The result of a typical CN2 session is a list of nested rules of the form "*if .. then .. else ..*". The nested structure can become extremely complex when the number of rules is high, making this representation difficult to interpret. Subsequent versions of CN2 [48] allow the induction of unordered rule lists. Rules are learned for each class independently, and for each induced rule only covered examples belonging to the class are removed, instead of removing all covered examples. The produced final rules can therefore overlap, but at the same time can be interpreted independently. When using an unordered rule list to predict the class of new instances, several rules can contribute to the classification of this example, often resulting in an improved accuracy.

Unfortunately, and due to the way CN2 iteratively removes examples, in an unordered rule list only the first few induced rules are usually of interest. Subsequently induced rules are obtained from biased example subsets, i.e., subsets including only positive examples not covered by previously induced rules. This is not suitable for description tasks, where desired rules may cover overlapped instances. CN2-SD [3], a modification of CN2 for subgroup discovery, solves this problem. The basic idea is to generalize the covering algorithm by introducing example weights. Initially, all examples have a weight of 1.0. However, the weights of examples that are covered by a rule will not be set to 0.0 (as in CN2), but instead will be reduced by a certain factor. The resulting number of rules is typically higher than with CN2, since most examples will be covered by more than one rule. CN2-SD has therefore two complementary advantages: it can learn better local patterns because the influence of previously covered patterns is reduced, but not completely ignored; and it can produce a better classifier by combining the evidence of more induced rules.

B.6 Experiments

B.6.1 UCI datasets

We conducted some experiments to illustrate the generality of rules produced by some search heuristics with the general-to-specific strategy. We use the rule learning systems CN2 [54, 48] and CN2-SD [3] with different heuristics: k-linear ($k = 0.2, 0.5, 1.0$), m-estimate ($m = 20, 5$), weighted relative accuracy, and quadratic ($P_1 = P, N_1 = N$). 18 UCI data sets were selected for the evaluation: Anneal, Audio, Balance, Car, Glass, Iris, Wine, Heart-Cleveland, Heart-Hungarian, Hepatitis, Australian, Breast, Echocardiogram, German, Ionosphere, Pole-and-card, Tic-tac-toe, Voting record. In addition to the generality of the learned rule set: the number of rules (*#rules*), the average number of conditions per a rule (*#conds*), and the average number of covered examples per rule (*covering*)), we also report the predictive performance on 18 data sets by 10×10-fold cross-validation (Table B.1).

In general, the number of rules and the generality of single rules produced by CN2 are much smaller than by CN2-SD when using the same rule evaluation heuristic. This

Table B.1: The average number of rules, the average number of conditions per a rule, and average accuracy in the rule set on 18 datasets produced by different heuristics with CN2 and CN2-SD.

		Lap.	20-est.	5-est.	0.2-lin.	0.5-lin.	1.0-lin.	WRACC	Quad.
#rules	CN2	39.5	25.3	35.7	10.1	12.7	16.2	13.3	13.9
	CN2-SD	78.1	53.7	67.1	39.9	37.8	36.0	39.7	42.3
#conds	CN2	3.10	2.95	3.03	2.80	2.66	2.49	2.71	3.04
	CN2-SD	3.21	3.06	3.14	3.27	2.86	2.69	2.85	3.58
#covering	CN2	24.3	29.2	24.3	124.4	61.9	48.3	57.3	68.0
	CN2-SD	27.1	32.5	26.8	100.9	58.2	46.3	46.6	79.0
acc.	CN2	82.9	82.3	83.5	74.0	79.3	79.6	78.6	78.8
	CN2-SD	83.3	84.3	85.0	78.5	80.9	80.1	81.3	80.0

explains why CN2-SD is more appropriate than CN2 for description tasks. The accuracy of CN2-SD is also often higher than that of CN2, because CN2-SD combines the evidence of more induced rules.

As can be seen in Table B.1, the number of rules produced by k-linear heuristic with $k = 0.2$ is often the smallest and each rule covers the largest number of examples. When increasing k, the returned number of rules is increased and the average number of covered example is inversely reduced. This is explained by the fact that the relative area of k-linear-beam search space is proportionally increased with the increase of k, hence more specific rules are preferred, and therefore the number of rules is increased. The same effect was also observed in m-estimates when m decreased.

While k-linear heuristic with $k = 0.2$ produces too general rules, Laplacian heuristic tends to prefer too specific rules. Heuristics WRA and *Quadratic* lay somewhere in between, producing rules not too specific but not too general either. The heuristic WRA has been used for the task of knowledge discovery in the work of Lavrac [3].

B.6.2 Microarray dataset

In this section, we illustrate the generalization ability of some heuristics in uncovering the transcriptional regulatory rules from microarray data. We used two heuristics, Laplacian and WRA, which represent two categories of heuristics: (1) preferring highly specific rules and (2) preferring highly general rules. We used regulatory tables as training data (a full description of these regulatory tables can be found in Chapter 3 of this thesis. Regulatory

tables are often incomplete and contain noise, which makes them an interesting subject of study to test the capability of CN2-SD to deal with such kind of data. Table B.2 is an example of some regulatory rules found by CN2 and CN2-SD with heuristics Laplacian and WRA from the regulatory table of gene YBR048W.

Table B.2: Some regulatory rules produced by CN2 and CN2-SD with heuristics Laplacian and WRA

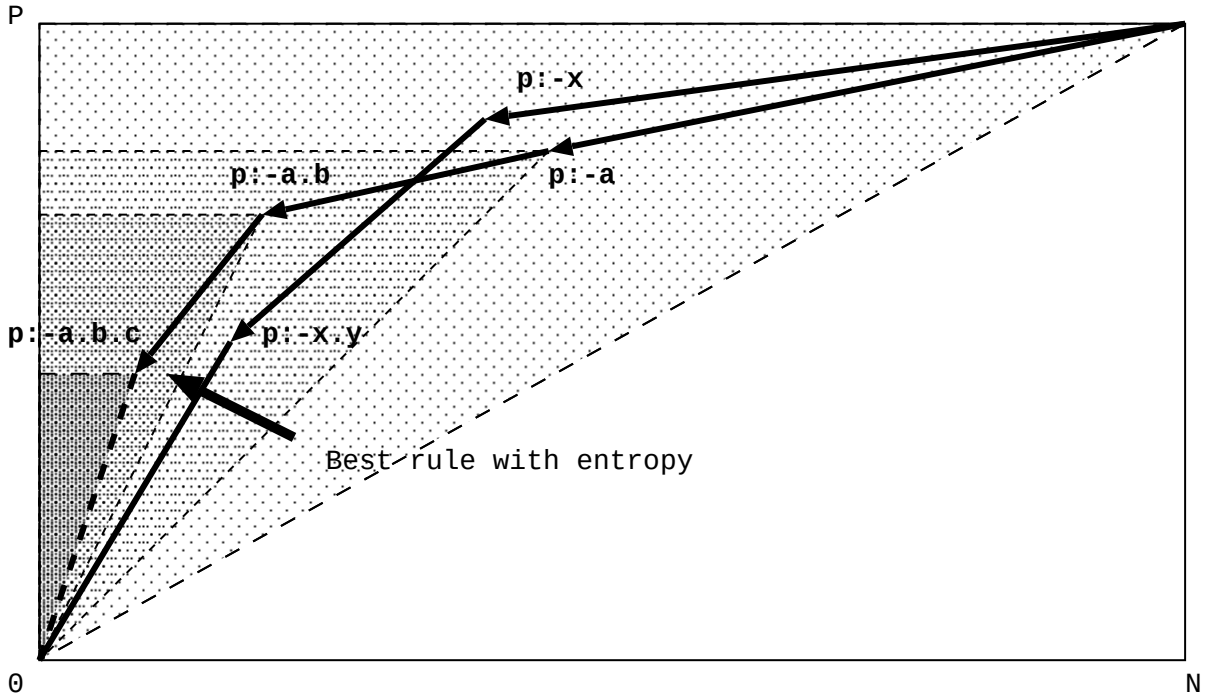
Regulatory rules	CN2		CN2-SD ($g = 0.5$)	
	Laplacian	WRA	Laplacian	WRA
HAP4=I → YBR048W=D [2497,234,1842]	No	No	No	Yes
MOT3=I, HAP4=I → YBR048W=D [138,0,5]	Yes	Yes	Yes	Yes
HAP4=I, FHL1=D → YBR048W=D [139,3,18]	No	No	No	Yes
HAP4=I, FHL1=D, RAP1=D → YBR048W=D [58,0,2]	Yes	Yes	Yes	No
UME1=I, HAP4=I, RSC1=D → YBR048W=D [19,0,5]	Yes	Yes	Yes	No
HAP4=I, RCS1=D, RAP1=D → YBR048W=D [18,0,5]	Yes	Yes	Yes	Yes
SFP1=D, RCS1=I → YBR048W=D [7,0,8]	No	No	No	Yes
RCS1=I, FHL1=D → YBR048W=D [4,0,1]	Yes	Yes	Yes	Yes

As it can be seen in table B.2, CN2-SD with WRA heuristic obtained a better regulatory rule set, including rules with different generality: from rules with very high generality such as HAP4=I → YBR048W=D (covering 4573 instances), to very specific rules as RCS1=I, FHL1=D → YBR048W=D (covering only 5 instances). Therefore, we use WRA heuristic for this work.

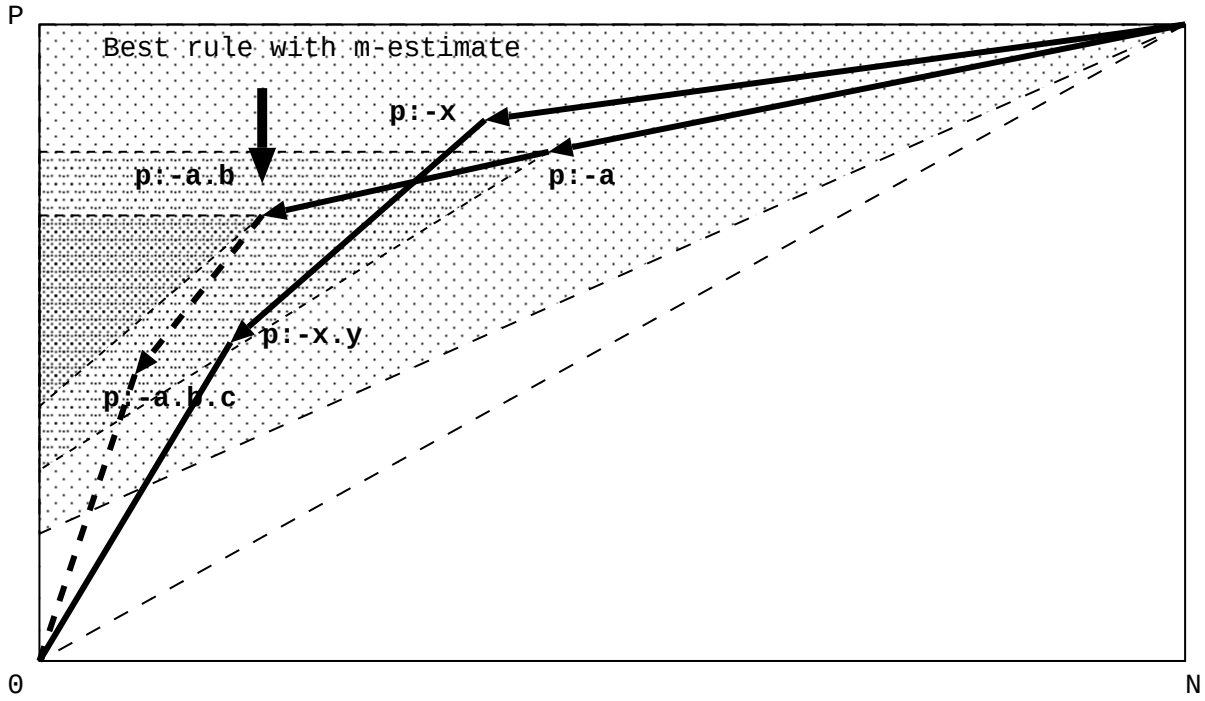
B.7 Concluding remarks

We have used PN-space to analyze the generality (or specificity) of rules produced by different heuristics with a general-to-specific learning strategy. We showed that the beam search space of a heuristic controls the generality of learned rules. We compared the generalization ability of some well-known heuristics like entropy, m-estimate, weighted relative accuracy. We also analyzed, under PN-space view, why entropy produces rules too specific, while weighted relative accuracy produces highly general rules. Our experiments showed that the weighted covering algorithm (in CN2-SD) combining with a rule evaluation heuristic with high generalization ability (like WRA) is the best suited for learning descriptive rules.

Compared to classification and prediction tasks, validating results of methods applied to description tasks (or knowledge discovery) is still quite difficult. The authors of CN2-SD have successfully applied CN2-SD with WRA heuristic in some real applications [3], and so we did to discover regulatory rules from microarray data [23].



(1)



(2)

Figure B.4: Impact of rule measures in learning a single rule. (1) with entropy measure. (2) with m-estimate measure.

Bibliography

[1] B. Alberts and A. Johnson and J. Lewis and M. Raff and K. Roberts and P. Walter. *Molecular Biology of the Cell, Fourth Edition.* Garland Science, Taylor&Francis Group, 2002.

[2] M. Gribskov and A.D. McLachlan and D. Eisenberg. Profile analysis: detection of distantly related proteins. *PNAS.*, 84(13):4355–4358, 1987.

[3] N. Lavrac and B. Kavsek and P. Flach and L. Todorovski. Subgroup discovery with cn2-sd. *Journal of Machine Learning Research*, 5:153–188, 2004.

[4] L. Timothy and Bailey and Charles Elkan. The value of prior knowledge in discovering motifs with meme. In *Proceedings of the 3rd International Conference on Intelligent Systems Molecular Biology*, pages 21–29, Menlo Park CA, USA, 1995.

[5] V. R. Iyer and C. E. Horak and C. S. Scafe and D. Botstein and M. Snyder and P. O. Brown. Genomic binding sites of the yeast cell-cycle transcription factors sbf and mbf. *Nature*, 409(6819):533–538, 2001.

[6] M. Zaki and C-J. Hsiao. Charm: An efficient algorithm for closed itemset mining. *2nd SIAM Inter. Conference on Data Mining*, 2002.

[7] C. T. Harbison and D. B. Gordon and et al. Transcriptional regulatory code of an eukaryotic genome. *Nature*, 431(7004):99–104, 2004.

[8] G. D. Bader and D. Betel and C. W. Hogue. Bind: the biomolecular interaction network database. *Nucleic Acids Res*, 31(1):248–250, 2003.

[9] A.J. Shepherd and D. Gorse and J.M. Thornton. Prediction of the location and type of β-turns in proteins using neural networks. *Protein Sci.*, 8:1045–1055, 1999.

[10] X. S. Liu and D. L. Brutlag and J. S. Liu. An algorithm for finding protein-dna binding sites with applications to chromatin-immunoprecipitation microarray experiments. *Nature Biotechnology*, 20, 2002.

[11] V. Matys and E. Fricke and R. Geffers and et al. Transfac: Transcriptional regulation, from patterns to profiles. *Nucleic Acids Research*, 31, 2003.

[12] S. Altschul and E. Koonin. Iterated profile searches with psi-blast - a tool for discovery in protein databases. *Trends Biochem. Sci.*, 23:444–447, 1998.

[13] B. Ren and F. Robert and et al. Genome-wide location and function of dna binding proteins. *Science*, 290, 2000.

[14] J. Ihmels and G. Friedlander and S. Bergmann and O. Sarig and Y. Ziv and N. Barkai. Revealing modular organization in the yeast transcriptional network. *Nature Genetics*, 31(4):370–377, 2002.

[15] Z. Bar-Joseph and G. K. Gerber and T. Lee and et al. Computational discovery of gene modules and regulatory networks. *Nature Biotechnology*, 21:1337–1342, 2003.

[16] P. T. Spellman and G. Sherlock and M. Q. Zhang and V. R. Iyer and K. Anders and M. B. Eisen and P. O. Brown and D. Botstein and B. Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization. *Mol Biol Cell*, 9(12):3273–3297, 1998.

[17] P.Y. Chou and G.D. Fasman. Conformational parameters for amino acids in helical, β-sheet and random coil regions calculated from proteins. *Biochemistry*, 13:211–222, 1974.

[18] P.Y. Chou and G.D. Fasman. Prediction of β-turns. *Biophys. J.*, 26:367–384, 1979.

[19] Harpreet Kaur and G.P.S Raghava. An evaluation of β-turn prediction methods. *Bioinformatics*, 18:1508–1514, 2002.

[20] Harpreet Kaur and G.P.S Raghava. A neural network based method for prediction of gamma-turns in proteins from multiple sequence alignment. *Protein Sci.*, 12:923–929, 2003.

[21] Harpreet Kaur and G.P.S Raghava. Prediction of β-turns in proteins from multiple alignment using neural network. *Protein Sci.*, 12:627–634, 2003.

[22] H. Kim and H. Park. Protein secondary structure prediction by support vector machines and position-specific scoring matrices. *Protein Engin.*, 2003.

[23] T. H. Pham and J. C. Clemente and K. Satou and T. B. Ho. Computational discovery of transcriptional regulatory rules. *Bioinformatics*, 21(suppl.2), 2005.

[24] M. A. Harris and J. Clark and A. Ireland and J. Lomax and et al. The gene ontology (go) database and informatics resource. *Nucleic Acids Res.*, 32:D258–D261, 2004.

[25] F. P. Roth and J. D. Hughes and P. Estep and G. M. Church. Finding dna regulatory motifs within unaligned noncoding sequences clustered by whole-genome mrna quantitation. *Nature Biotechnology*, 16, 1998.

[26] N. Cristianini and J. Shawe Taylor. *An Introduction to Support Vector Machines*. Cambridge, 2002.

[27] I. Guyon and J. Weston and S. Barnhill and V. Vapnik. Selection for cancer classification using support vector machines. *Machine Learning*, 46(1/3): 389, 2002.

[28] G. Grahne and J. Zhu. Efficiently using prefix-trees in mining frequent itemsets. In *Workshop on Frequent Itemset Mining Implementations*, Melbourne FL, USA, 2003.

[29] C.M. Wilmot and J.M. Thornton. Analysis and prediction of the different types of β-turns in proteins. *J. Mol. Biol.*, 203:221–232, 1988.

[30] E.G. Hutchinson and J.M. Thornton. A program to identify and analyze structural motifs in proteins. *Protein Sci.*, 5:212–220, 1996.

[31] K.C. Chou and J.R. Blinn. Classification and prediction of β-turn types. *J. Protein Chem.*, 16:575–595, 1997.

[32] Y. Minakuchi and K. Satou and A. Konagaya. Prediction of protein-protein interaction sites using support vector machines. In *Inter. Conference on Mathematics and Engineering Techniques in Medicine and Bilogical Sciences*, pages 22–28, 2003.

[33] T. H. Pham and K. Satou and T. B. Ho. Mining yeast transcriptional regulatory modules from factor dna-binding sites and gene expression data. *Genome Informatics*, 15(2):287–295, 2004.

[34] C.T Zhang and K.C. Chou. Prediction of β-turns in proteins by 1-4 & 2-3 correlation model. *Biopolymers*, 41:673–702, 1997.

[35] G.D. Rose and L.M. Gierasch and J.A. Smith. Turns in peptides and proteins. *Adv. Protein Chem.*, 37:100–109, 1985.

[36] J. Brank and M. Grobelnik and N. Milic-Frayling and D. Mladenic. Feature selection using support vector machines. In *3rd inter. Conference on Data Mining Methods and Databases for Engineering, Finance and Other Fields*, pages 261–273, 2002.

[37] A. Gasch and M. Huang and S. Metzner and D. Botstein and S. Elledge and P. O. Brown. Genomic expression responses to dna-damaging agents and the regulatory role of the yeast atr homolog mec1p. *Mol Biol Cell*, 12(10):2987–3003, 2001.

[38] E. Segal and M. Shapira and A. Regev and D. Peer and D. Botstein and D. Koller and N. Friedman. Module networks: Identifying regulatory modules and their condition-specific regulators from gene expression data. *Nature Genetics*, 34(2):166–176, 2003.

[39] T. I. Lee and N. J. Rinaldi and et al. Transcriptional regulatory networks in saccharomyces cerevisiae. *Science*, 298:799–804, 2002.

[40] B. Kavsek and N. Lavrac and V. Javanoski. Apriori-sd: Adapting association rule learning to subgroup discovery. In *Intelligent Data Analysis IDA 2003*, 2003.

[41] M. Kellis and N. Patterson and M. Endrizzi and B. Birren and E. S. Lander. Sequencing and comparison of yeast species to identify genes and regulatory elements. *Nature*, 423(6937):241–254, 2003.

[42] J. Furnkranz and P. Flach. An analysis of rule evaluation metrics. In *20th International Conference on Machine Learning (ICML-03)*, Washington, 2003.

[43] J. Furnkranz and P. Flach. An analysis of stopping and filtering criteria for rule learning. In *15th European Conference on Machine Learning (ECML-04)*, Italy, 2004.

[44] J. Furnkranz and P. Flach. Roc 'n' rule learning – towards a better understanding of covering algorithms. *Machine Learning*, 58(1):39–77, 2005.

[45] N. Lavrac and P. Flach and B. Zupan. Rule evaluation measures: A unifying view. In *Ninth International Workshop on Inductive Logic Programming (ILP'99)*, 1999.

[46] A. Gasch and P. Spellman and C. Kao and O. Carmel-Harel and M. Eisen MB and G. Storz and D. Botstein and P. Brown. Genomic expression programs in the response of yeast cells to environmental changes. *Mol Biol Cell*, 11(12):4241–4257, 2000.

[47] Y. Pilpel and P. Sudarsanam and G. M. Church. Identifying regulatory networks by combinatorial analysis of promoter elements. *Nature Genetics*, 29(2):153–159, 2001.

[48] P. Clark and R. Boswell. Rule induction with cn2: Some recent improvements. In *Proceedings of the 5th European Working Sessions on Learning*, pages 151–163, Porto, Portugal, 1991.

[49] K. Guruprasad and S. Rajkumar. β- and γ-turns in proteins revisited: a new set of amino acid dependent positional preferences and potential. *J. Biosci.*, 25:143–156, 2000.

[50] T. H. Pham and T. B. Ho and K. Satou. Rule evaluation heuristics for knowledge discovery. *Workshop on Knowledge Discovery and Data Management in Biomedical Science (KDDMBS), 9th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-05)*, 2005.

[51] R. Agrawal and T. Imielinski and and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD Inter. Conference on Management of Data*, pages 207–216, Washington DC, USA, 1993.

[52] S. Altschul and T. Madden and A. Shaffer and J. Zhang and Z. Zhang and et al. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucl. Acids Res.*, 25:3389–3402, 1997.

[53] C. Koch and T. Moll and M. Neuberg and H. Ahorn and K. Nasmyth. A role for the transcription factors mbp1 and swi4 in progression from g1 to s phase. *Science*, 261(5128):1551–1557, 1993.

[54] P. Clark and T. Nibblet. The cn2 induction algorithm. *Machine Learning*, 3(4):261–283, 1989.

[55] R. A. Bruin and W. H. McDonald and T. I. Kalashnikova and J. Yates and C. Wittenberg. Cln3 activates g1-specific transcription via phosphorylation of the sbf bound repressor whi5. *Cell*, 117(7):887–98, 2004.

[56] B. Liu and W. Hsu and Y. Ma. Integrating classification and association rule mining. In *4th International Conference on Knowledge Discovery and Data Mining KDD 98*, 1998.

[57] J. D. Lieb and X. Liu and D. Botstein and P. O. Brown. Promoter-specific binding of rap1 revealed by genomewide maps of protein-dna association. *Nature Genetics*, 28, 2001.

[58] N. Pasquier and Y. Bastide and R. Taouil and L. Lakhal. Efficient mining of association rules using closed itemset lattices. *Information Systems*, 24(1):25–46, 1999.

[59] K. Takano and Y. Yamagata and K. Yutani. Role of amino acid residues at turns in the conformational stability and folding of human lysozyme. *Biochemistry*, 39:8655–8665, 2000.

[60] D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, Belmont, MA, 1995.

[61] A.D. Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997.

[62] B. Cestnik. Estimating probabilities: a crucial task in machine learning. In *Proceedings of the Ninth European Conference on Artificial Intelligence*, pages 147–149, 1990.

[63] K.C. Chou. Prediction of tight turns and their types in proteins. *Analytical Biochem.*, 286:1–16, 2000.

[64] J. Deleo. Measuring classifier intelligence. *Second International Symposium on Uncertainty Modelling and Analysis*, pages 318–325, 1993.

[65] J. Furnkranz. Separate-and-conquer rule learning. *Artificial Intelligence Review*, 13(1):03–54, 1999.

[66] GM. Church J. Aach, W. Rindone. Systematic management and analysis of yeast gene expression data. *Genome Res.*, 10(4):431–445, 2000.

[67] D.T. Jones. Protein secondary structure prediction based on position-specific scoring matrices. *Mol. Biol.*, 292:195–202, 1999.

[68] D. G. Luenberger. *Introduction to Linear and Nonlinear Programming*. Addison-Wesley, Reading, MA, 1973.

[69] J. Quinland. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[70] J. Quinland. Generating production rules from decision trees. In *10th International Joint Conference on Artificial Intelligence (IJCAI-87)*, 1987.

[71] J.S. Richardson. The anatomy and taxonomy of protein structure. *Adv. Protein Chem.*, 34:167–339, 1981.

[72] V. Vapnik. *Estimation of Dependences Based on Empirical data*. English translation: Springer Verlag, N.Y. 1982, 1979.

[73] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.

[74] V. Vapnik. *Statistical Learning Theory*. Wiley N.Y., 1998.

[75] M. Zaki. Generating non-redundant association rules. *Data Mining and Knowledge Discovery: An International Journal*, 2004.

Publications

[1] T.H. Pham, J. C. Clemente, K. Satou, T.B. Ho, "Computational Discovery of Transcriptional Regulatory Rules", *Bioinformatics*, vol.21, suppl.2 (in press) (2005).

[2] T.H. Pham, K. Satou, T.B. Ho, "Support Vector Machines for Prediction and analysis of Beta-turns and Gamma-turns in Proteins", *Journal of Bioinformatics and Computational Biology*, vol.3, no.2, pp.343-358 (2005).

[3] T.H. Pham, K. Satou, T.B. Ho, "Mining Yeast Transcriptional Regulatory Modules from Factor DNA-binding Sites and Gene Expression Data", *Proc. Genome Informatics*, vol.15, no.2, pp.287-295 (2004).

[4] T.H. Pham, K. Satou, T.B. Ho, "Prediction and Analysis of β-turns in Proteins by Support Vector Machine", *Proc. Genome Informatics*, vol.14, pp.196-205 (2003).

[5] T.H. Pham, T.B. Ho, K. Satou, J. C. Clemente, "Rule Evaluation Heuristics for Knowledge Discovery", *Workshop on Knowledge Discovery and Data Management in Biomedical Science (KDDMBS), 9th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-05)*, 18-20 May 2005, Hanoi, Vietnam.

[6] T.H. Pham, K. Satou, T.B. Ho, "Computational Discovery of Yeast Transcriptional Regulatory Modules from DNA Sequences and Gene Expression Data", *5th International Symposium on Knowledge and Systems Sciences"(KSS)*, (2004).

[7] T.B. Ho, S. Kawasaki, T.H. Pham, S.Q. Le, V.N. Huynh: "Lessons learned in knowledge discovery from medical and biological data", *International Journal of Knowledge and Learning* (2005).