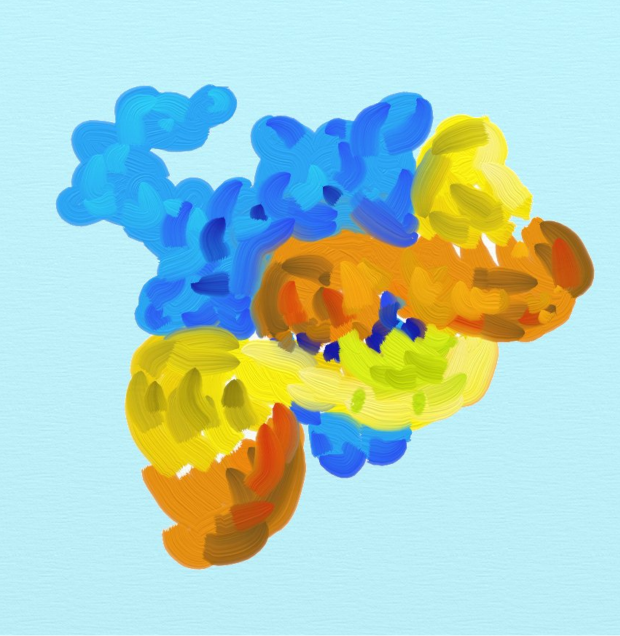
PREDICTION OF DNA-BINDING PROPENSITY OF PROTEINS USING MACHINE LEARNING



ANDREA SZABÓOVÁ

Department of Cybernetics
Faculty of Electrical Engineering
Czech Technical University in Prague

A thesis submitted for the degree of Doctor of Philosophy (Ph.D.)
Study Programme No. P2612-Electrotechnics and Informatics
Branch No. 3902V035 - Artificial Intelligence and Biocybernetics

May 2013

SUPERVISOR:
Doc. Ing. Filip Železný, PhD.

I would like to dedicate this thesis to my loving parents.

ABSTRACT

The process of protein-DNA interaction has been an important subject of recent bioinformatics research. The goal of this thesis is to find an accurate model for the prediction of DNA-binding propensity. A characterisation of proteins which are able to bind to DNA, can help biologists to better understand various processes that occur in living cells. In this thesis, we contribute novel methods based on relational machine learning and novel methods based on distribution-based approaches.

The thesis has two main parts. The first part presents our relational learning approaches for the problem of DNA-binding propensity prediction. We introduce three novel relational learning methods for prediction. These methods are based on automatic construction of patterns capturing local configurations of amino acids in DNA-binding proteins. The automatically constructed patterns are used as attributes for predictive classification using established machine learning algorithms. This is in contrast to most existing works, where attributes are constructed manually by domain experts. In principle, our novel methods can find patterns which may be new for domain experts. In addition, in this part, we describe a method for preprocessing of relational learning examples.

The second part of the thesis presents our novel distribution-based approaches. The distribution-based approaches aim at modelling the probability that if we randomly pick a region of a learning example, it will match a given pattern. This enables us to capture distribution of certain substructures defined by a given pattern. We introduce distribution-based methods able to learn only from primary structures or from spatial structures of proteins.

The methods described in this thesis perform better than state-of-the-art methods based on physicochemical features of protein structures. Moreover, they provide us with interpretable structural patterns characterising local configurations of amino acids in protein structures.

Another field of application of our novel methods is the prediction of antimicrobial activity of peptides. Antimicrobial peptides are molecules responsible for defence against microbial infections in the first stages of the immunological response. Recently antimicrobial peptides have been recognized as a potential replacement of conventional antibiotics. An accurate model for antimicrobial activity prediction could help in the process of design of new peptides for medical application. We show that our novel methods originally designed for DNA-binding propensity prediction are able to obtain comparable or even better results than state-of-the-art methods for the antimicrobial activity prediction. In addition, since antimicrobial peptides are typically much smaller than DNA-binding proteins, they are excellent for validation of the predictive ability of our novel methods in the domain of smaller structures.

ABSTRAKT

Proces interakcie proteínov a DNA je dôležitým predmetom súčasného výskumu v bioinformatike. Cieľom tejto práce je nájsť presný model pre predikciu schopnosti proteínov viazať sa na DNA. Charakterizácia proteínov, ktoré sú schopné viazať sa na DNA, môže pomôcť biológom lepšie pochopiť rôzne procesy, ktoré sa odohrávajú v živých bunkách. V tejto práci uvádzame naše nové metódy založené na relačnom strojovom učení a metódy založené na distribučnom prístupe na riešenie tejto problematiky.

Práca má dve hlavné časti. Prvá časť predstavuje naše metódy založené na relačnom strojovom učení k riešeniu problému predikcie schopnosti proteínu viazať sa na DNA. Predstavíme tri nové metódy relačného strojového učenia pre predikciu. Tieto metódy sú založené na automatickom konštruovaní vzorov zachytávajúcich miestnu konfiguráciu aminokyselín v proteínoch schopných viazať sa na DNA. Automaticky konštruované vzory sú používané ako atribúty pre prediktívnu klasifikáciu s využitím existujúcich algoritmov strojového učenia. Väčšina existujúcich prác využíva atribúty ručne konštruované doménovými expertmi. V zásade, naše nové metódy môžu nájsť také vzory, ktoré budú nové pre doménových odborníkov. Naviac, v tejto časti popíšeme našu metódu na predspracovanie relačných trénovacích príkladov.

Druhá časť práce predstavuje naše nové metódy založené na distribučnom prístupe. Tieto metódy sú zamerané na modelovanie pravdepodobnosti, s ktorou náhodne vybraná oblasť v proteíne bude odpovedať zadanému vzoru. Toto nám umožňuje zachytiť distribúciu niektorých podštruktúr definovaných daným vzorom. Predstavíme metódy, ktoré sú schopné sa učiť len z primárnych štruktúr alebo z priestorových štruktúr bielkovín.

Metódy popísané v tejto práci fungujú lepšie ako najlepšie známe metódy založené na fyzikálno-chemických vlastnostiach proteínových štruktúr. Navyše, poskytnú nám interpretovateľné štrukturálne vzory charakterizujúce miestne konfigurácie aminokyselín v proteínových štruktúrach.

Ďalšou oblasťou možného použitia našich nových metód je problém predikcie antimikrobiálnej aktivity peptidov. Antimikrobiálne peptidy sú molekuly zodpovedné za obranu proti mikrobiálnej infekcii v prvých fázach imunitnej reakcie. Antimikrobiálne peptidy sú považované za potenciálnu náhradu konvenčných antibiotík. Presný model pre predikciu antimikrobiálnej aktivity by mohol pomôcť v procese navrhovania nových peptidov pre medicínske použitie. Ukázali sme, že naše nové metódy pôvodne určené pre predikciu schopnosti proteínov viazať sa na DNA sú schopné získať porovnateľné, alebo dokonca lepšie výsledky ako najlepšie známe metódy pre predikciu antimikrobiálnej aktivity. Naviac, pretože antimikrobiálne peptidy sú zvyčajne oveľa menšie ako DNA-viažúce proteíny, sú vynikajúce pre validáciu prediktívnej schopnosti našich nových metód v oblasti menších štruktúr.

PUBLICATIONS

Andrea Szabóová, Ondřej Kuželka, Filip Železný and Jakub Tolar. Prediction of DNA-binding Propensity of Proteins by the Ball-Histogram Method using Automatic Template Search. *BMC Bioinformatics*, 13, Sup 10 , 2012.

Andrea Szabóová, Ondřej Kuželka, Filip Železný and Jakub Tolar. Prediction of DNA-binding proteins from relational features. *Proteome Science*, 10 , 2012.

Andrea Szabóová, Ondřej Kuželka and Filip Železný. Prediction of Antimicrobial Activity of Peptides using Relational Machine Learning. *IEEE International Conference on Bioinformatics and Biomedicine (BIBM 2012)*, 2012.

Ondřej Kuželka, Andrea Szabóová and Filip Železný. Bounded Least General Generalization. *ILP 2012*, 2012.

Ondřej Kuželka, Andrea Szabóová and Filip Železný. Extending the Ball-Histogram Method with Continuous Distributions and an Application to Prediction of DNA-Binding Proteins. *IEEE International Conference on Bioinformatics and Biomedicine (BIBM 2012)*, 2012.

Ondřej Kuželka, Andrea Szabóová and Filip Železný. Reducing Examples in Relational Learning with Bounded-Treewidth Hypotheses. *Proceedings of the Workshop on New Frontiers in Mining Complex Patterns (NFMCP 2012)*, 2013.

Ondřej Kuželka, Andrea Szabóová and Filip Železný. Relational Learning with Polynomials. *IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2012)*, 2012.

Andrea Szabóová, Ondřej Kuželka, Filip Železný and Jakub Tolar. Searching for Important Amino Acids in DNA-binding Proteins for Histogram Methods (poster paper). *The 7th International Symposium on Bioinformatics Research and Applications*, 2011.

Andrea Szabóová, Ondřej Kuželka, Sergio Morales E., Filip Železný and Jakub Tolar. Prediction of DNA-binding Propensity of Proteins by the Ball-Histogram Method. *The 7th International Symposium on Bioinformatics Research and Applications*, 2011.

Ondřej Kuželka, Andrea Szabóová and Filip Železný. Gaussian Logic and Its Applications in Bioinformatics (poster paper). *ACM-BCB 2011: ACM Conference on Bioinformatics, Computational Biology and Biomedicine*, 2011.

Ondřej Kuželka, Andrea Szabóová, Matěj Holec and Filip Železný. Gaussian Logic for Predictive Classification. *ECML/PKDD 2011: European Conference on*

Machine Learning and Principles and Practice of Knowledge Discovery in Databases, 2011.

Ondřej Kuželka, Andrea Szabóová, Matěj Holec and Filip Železný. Gaussian Logic for Proteomics and Genomics. *MLSB 2011: the 5th International Workshop on Machine Learning in Systems Biology*, 2011.

Andrea Szabóová, Ondřej Kuželka, Filip Železný and Jakub Tolar. Mining Frequent Spatial Docking Patterns in Zinc Finger - DNA Complexes. *Proceedings of the 7th European Symposium on Biomedical Engineering*, 2010.

Andrea Szabóová, Ondřej Kuželka, Filip Železný and Jakub Tolar. Prediction of DNA-Binding Proteins from Structural Features. *Proceedings of the Fourth International Workshop on Machine Learning in Systems Biology*, 2010.

"A mind is like a parachute. It doesn't work if it is not open."

— Frank Zappa

ACKNOWLEDGEMENTS

I would like to thank to my supervisor *Doc. Filip Železný* for the opportunity to work in the field of data mining applied in the area of bioinformatics and also for his encouraging guidance.

I would also like to thank to *Prof. Jakub Tolar* for motivating this work from the biological point of view and to my colleague *Ondřej Kuželka* for the great collaboration.

I would like to thank to our exchange students from Costa Rica who cooperated on the development of a user interface for some of our algorithms: *Laura Vásquez*, *Keylor Chaves*, *Beatriz González* and *Hugo Trejos*.

I acknowledge support by the grant provided by the *Czech Science Foundation: P202/12/2032*, by the grant provided by the *Czech Ministry of Education: ME10047*, and by the grants provided by the *Grant Agency of the Czech Technical University in Prague: #10-801940* and *161-811550C000*.

My thanks also belong to my family and friends for their love and support.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

The process of protein-DNA interaction has been an important subject of recent bioinformatics research, however, it has not been completely understood yet. DNA-binding proteins have a vital role in the biological processing of genetic information like DNA transcription, replication, maintenance and the regulation of gene expression. The goal of this thesis is to find an accurate model for the prediction of DNA-binding propensity. This is an important task for the following reasons. A characterisation of proteins which are able to bind to DNA, can help biologists to better understand various processes that occur in living cells at the microscopic level. This can be also useful for designing new proteins for emerging gene therapies. To design proteins of sufficient binding specificity, it is important to understand the patterns underlying DNA binding. It is unlikely that a prediction model of DNA-binding propensity will be deductively inferred from biochemical laws. The approaches presented in this thesis thus take the opposite way, where binding rules are learned by generalization from sets of known DNA-binding proteins. Here, we contribute novel methods based on relational machine learning and novel methods based on what we call *distribution-based approaches*. These methods perform better than state-of-the-art methods based on physicochemical features of protein structures. Moreover, they provide us with interpretable structural patterns characterising local configurations of amino acids in protein structures.

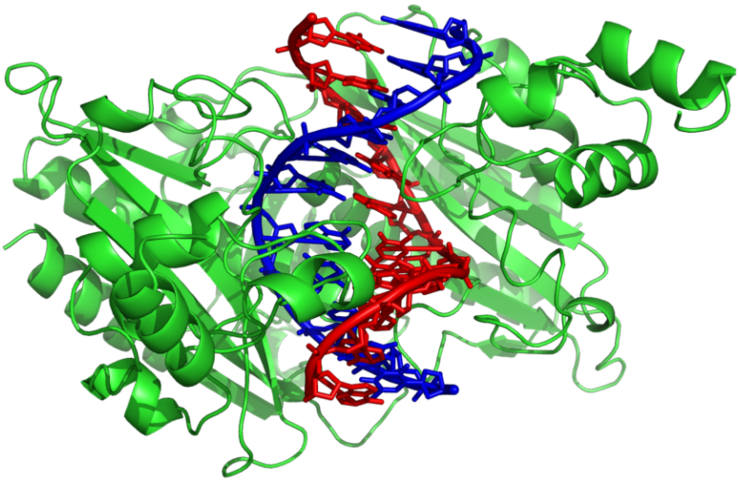


Figure 1: DNA-binding protein in complex with DNA. Adapted from *en.wikipedia*.

Another field of application of our novel methods is the prediction of antimicrobial activity of peptides. Antimicrobial peptides are molecules responsible for defence against microbial infections in the first stages of the immunological response. While in the case of DNA-binding process we are interested in the characterisation of proteins which are able to bind to the DNA, in the case of antimicrobial peptides (which are also sequences of amino acids) we try to characterise peptides involved in bacteria-killing processes. Since antimicrobial peptides are typically much smaller than DNA-binding proteins, they are excellent for validation of predictive ability of our novel methods in the domain of smaller structures. Furthermore, recently antimicrobial peptides have been recognized as a potential replacement of conventional antibiotics for which some microorganisms had already acquired resistance. An accurate model for antimicrobial activity prediction could help in the process of design of new peptides for medical application. We show that our novel methods are able to obtain comparable or even better results than state-of-the-art methods.

1.1 PROBLEM STATEMENT

This thesis aims to solve the problem of the prediction of DNA-binding propensity of proteins using relational machine learning from protein data describing their spatial structure. Our goal is to learn predictive models which would accurately predict proteins' DNA-binding function, while providing insights into the underlying DNA-binding process. Naturally, the more information is incorporated into the models, the more accurate results can be obtained. On the other hand, in most cases a limited amount of information is available about the protein structures. A typical example is evolutionary conservation, which is a very informative attribute, but may be missing for some groups of proteins. Therefore, we are trying to find models which use a minimal necessary amount of information – primary structure information, primary and secondary structure information, primary structure information and information about the spatial structure of the protein, additional information about physicochemical properties of protein regions. Moreover, datasets of protein structures are large, thus it is important to develop scalable methods for the prediction problem.

1.2 OVERVIEW OF THE THESIS

In Part i we start by describing the theoretical background needed in the subsequent parts of the thesis. This part is divided into two chapters. A brief description of the biological background is presented in Chapter 2. The necessary background from relational learning and constraint satisfaction is presented in Chapter 3. This chapter also includes a review of state-of-the-art approaches to the prediction of DNA-binding propensity in Section 3.2 and to antimicrobial activity prediction of peptides in Section 3.3. The datasets used in this thesis are described in Part ii.

Part iii is the first main part of the thesis. It presents our relational learning approaches for the problem of DNA-binding propensity and antimicrobial activity prediction. In this part we introduce three relational learning methods for prediction. These methods are based on automatic construction of novel patterns of DNA-binding proteins or antimicrobial peptides. The patterns are utilized for predictive classification using established machine learning algorithms. This is in contrast to most existing works, where features are constructed manually by domain experts. Our methods can find not only patterns which may be new for domain experts, but they can also improve prediction accuracy, as indicated by our experimental results. In addition, in this part, we describe a method for preprocessing of learning examples.

The first method presented in Chapter 5 uses relational learning based on structural patterns for the prediction of DNA-binding propensity and the antimicrobial activity of peptides. The second method described in Chapter 6 is based on multivariate relational aggregation of numeric properties of proteins' regions. The last method presented in Chapter 7 is based on construction of large, complex structural patterns. Chapter 8 describes our advanced preprocessing techniques. Finally, we discuss the advantages and disadvantages of relational learning approaches in Chapter 9.

Part iv is the second main part of the thesis. It presents our distribution-based approaches, which use restricted form of patterns. The distribution-based approaches aim at modelling the probability that if we randomly pick a region of a learning example, it will match the given pattern. This enables us to capture distribution of certain substructures defined by a given pattern.

First, we present our motivation for developing the distribution-based approaches in Chapter 10. We start with a short study of statistical distributions of gaps between charged amino acids in proteins in Chapter 11. Then we describe four novel distribution-based methods, namely the tube-histogram method in Chapter 12, the ball-histogram method in Chapter 13, the ball-histogram method for regression in Chapter 14 and the ball-histogram method with polynomial features in Chapter 15. Finally, we discuss the presented distribution-based approaches in Chapter 16.

Part v concludes the dissertation thesis. Additionally, an open-source software package containing implementations of some of our novel methods is described in Appendix A.

1.3 KEY THESIS CONTRIBUTIONS

This thesis brings the following contributions. The work in Part iii focuses on our novel relational learning approaches. The main contributions presented in this part are the following. We introduce a novel relational representation of protein data suitable for learning. We show that in order to achieve high predictive accuracy it is useful to count occurrences of small structural patterns, rather than just detecting their presence. We upgrade an existing state-of-the-art pattern construction system to be able to support occurrence counting and we enrich it with sampling strategies, which turn out to be necessary for data of proteomics scale. From the biologists' point of view, the main contributions of this part are that our new methods outperform existing state-of-the-art methods based on physicochemical features for the prediction of DNA-binding propensity of proteins and antimicrobial activity of peptides. This work was already published in [86, 89, 91].

Next, we contribute a framework able to work efficiently in multi-relational domains with numerical data. The new framework exploits multi-variate polynomial aggregation functions. We show that we are able to obtain favourable results even when using just information about primary and secondary structures of proteins. This work was published in [43, 46, 44].

We also introduce a new relational learning approach which is able to search for large, complex relational patterns. This method redefines the notion of equivalence of patterns (first-order logic clauses) and parametrizes it. Based on this, we present a new operator for pattern search called bounded least general generalization. We show that the method is able to construct accurate classifiers using only a small number of patterns. Another advantage is that these small sets of patterns are potentially easier to interpret by biologists. A part of this work was already published in [47, 48].

The work in Part iv investigates approaches based on modelling distributions of certain substructures in proteins. We contribute three novel distribution-based methods for predictive classification and one distribution-based method for regression. An important advantage of these methods is their scalability, which is much better compared to the general relational learning methods presented in Part iii. The main idea of the distribution-based approaches is to capture distributions of certain properties in learning examples and to construct features based on these distributions.

When using only primary structure information, we already obtain predictive accuracies comparable with state-of-the-art methods based on physicochemical features derived from spatial structure information. This work relying on primary structure information – the tube-histogram method – was published in [88].

Adding information about spatial structure and numeric properties of protein regions further improves the predictive accuracy. The novel ball-histogram method, which we present in this part, is able to capture distributions of certain properties in protein structures. Importantly, this method not only does scale better than relational learning approaches, but also has higher predictive accuracy. In fact, this method has already come close to methods exploiting evolutionary information in terms of predictive accuracy, despite the fact that it does not use evolutionary information at all. This work was already published in [90, 87, 45].

Part I

THEORETICAL BACKGROUND

2 BIOLOGICAL BACKGROUND

2.1 PROTEINS

Proteins are large biological molecules consisting of one or more chains of amino acids which are bonded together by peptide bonds between the carboxyl and amino groups of adjacent amino acid residues. The sequence of amino acids in a protein is defined by the sequence of a gene, which is encoded in the genetic code. In general, the genetic code specifies 20 standard amino acids.

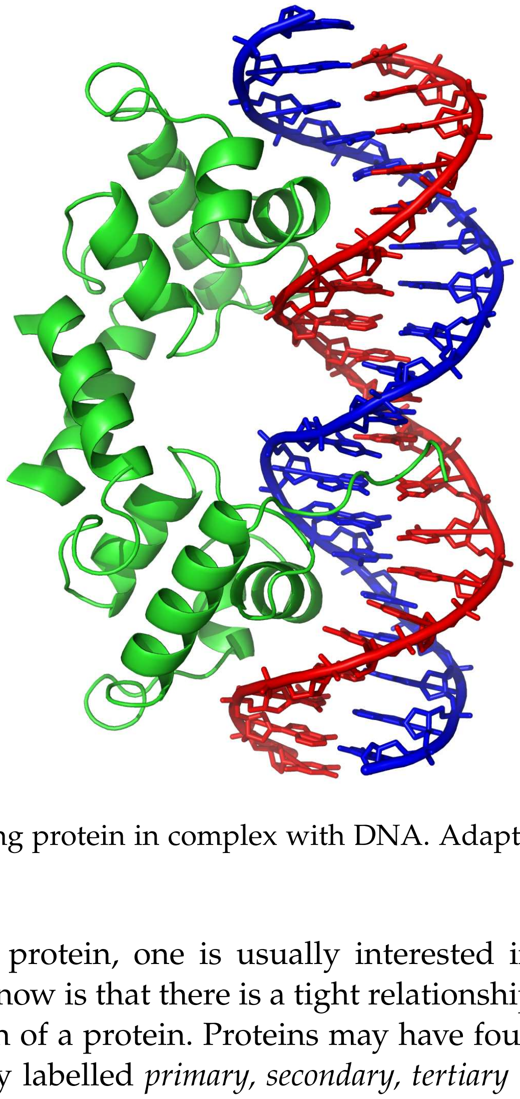


Figure 2: DNA-binding protein in complex with DNA. Adapted from *en.wikipedia*.

When studying a protein, one is usually interested in its function. One thing all biologists know is that there is a tight relationship between the structure and the function of a protein. Proteins may have four levels of structure. The levels are simply labelled *primary, secondary, tertiary* and *quaternary*. The primary structure of a protein is simply the sequence of amino acids comprising the molecule. The primary structure of a protein is the amino acid sequence within the molecule. Proteins fold locally into secondary structures such as α-helices, β-strands, and turns. Two or three adjacent secondary structures might combine into common local folds called *motifs* or *superssecondary* structures such as β-sheets. These building blocks then fold into the 3D or tertiary structure of a protein. Finally, one or more tertiary structures may be combined as subunits into a quaternary structure such as an enzyme or a virus.

2.2 DNA-BINDING PROTEINS

DNA-binding proteins (illustration in Figure 2) are proteins that contain DNA-binding domains and thus have a specific or general affinity for either single or double stranded DNA. A DNA-binding domain is an independently folded protein domain that contains at least one motif that recognizes double- or single-stranded DNA. A DNA-binding domain can recognize a specific DNA sequence (a recognition sequence) or have a general affinity to DNA.

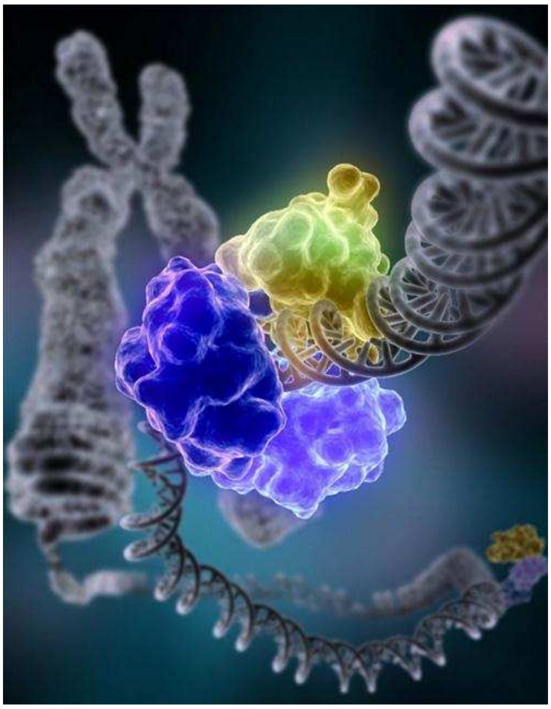


Figure 3: DNA repair. Adapted from *en.wikipedia*.

The function of DNA binding is either structural or involving transcription regulation, with the two roles sometimes overlapping. DNA-binding domains with functions involving DNA structure have biological roles in the replication, repair (illustration in Figure 3), storage, and modification of DNA. Many proteins involved in the regulation of gene expression contain DNA-binding domains. For example, proteins that regulate transcription by binding DNA are called transcription factors.

The DNA-binding domain interacts with the nucleotides of DNA in a DNA sequence-specific or non-sequence-specific manner, but even non-sequence-specific recognition involves some sort of molecular complementarity between protein and DNA. DNA recognition by the DNA-binding domain can occur at the major or minor groove of DNA (i.e. the spatial gaps between the two DNA strands), or at the sugar-phosphate DNA backbone. Each specific type of DNA recognition is tailored to the protein's function. Many DNA-binding domains must recognize specific DNA sequences, such as DNA-binding domains of transcription factors that activate specific genes, or those of enzymes that modify DNA at specific sites, like restriction enzymes and telomerase.

The specificity of the transcription and replication requires recognition on the molecular level between protein structures and nucleic acid structures. The genetic code has to be readable. The reading of the code is a conformationally specific interaction between amino acids and nucleic acids. Surface properties of the macromolecules involved are the essential key in the recognition process. Electrostatic interaction, hydrogen bonding capability and hydrophobic effects are of importance.

2.2.1 *Zinc Finger Proteins*

Zinc fingers (illustration in Figure 4) are small protein structural motifs that can coordinate one or more zinc ions to help stabilize their folds. They can be classified into several different structural families (zinc finger proteins) and typically function as interaction modules that bind DNA, RNA, proteins, or small molecules. The name "zinc finger" was originally coined to describe the finger-like appearance of a diagram showing the hypothesized structure of the repeated unit in Xenopus laevis transcription factor IIIA.

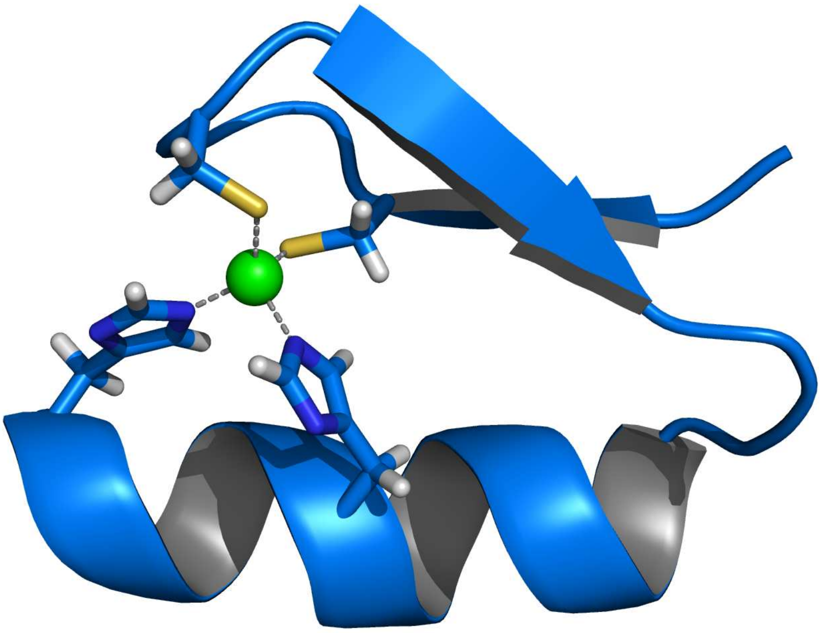


Figure 4: Cys_2His_2 zinc finger motif, consisting of an α helix and an antiparallel β sheet. The zinc ion (green) is coordinated by two histidine residues and two cysteine residues. Adapted from *en.wikipedia*.

Generating arrays of engineered Cys_2His_2 zinc fingers is one of the most developed methods for creating proteins capable of targeting desired genomic DNA sequences.

2.3 ANTIMICROBIAL PEPTIDES

Antimicrobial peptides are molecules responsible for defence against microbial infections in the first stages of the immunological response. Recently antimicrobial peptides have been recognized as a potential replacement of conventional antibiotics for which some microorganisms had already acquired resistance. Although, there are theories about the mechanisms by which antimicrobial peptides kill pathogenic microorganisms, the process has not been fully uncovered yet.

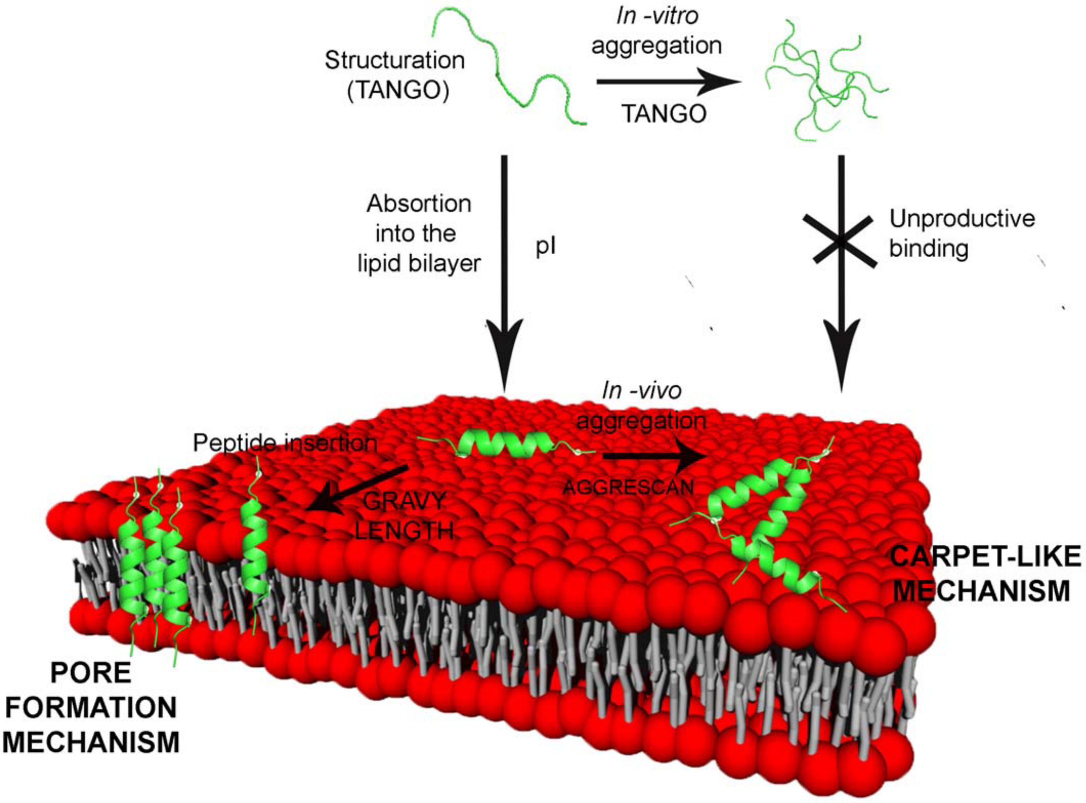


Figure 5: The figure depicts the main steps in the interaction and permeabilization of the bacterial membrane by AMPs. An arrow indicates each step. The peptide associated physicochemical parameters and the related prediction system used is also detailed. Adapted from [96].

Antimicrobial peptides (AMPs) have been actively researched for their potential therapeutic application against infectious diseases. AMPs are amino acid sequences of length typically from 6 to 100. They are produced by living organisms of various types as part of their innate immune system [73]. They express potent antimicrobial activity and are able to kill a wide range of microbes. In contrast to conventional antibiotics, AMPs are bacteriocidal (i.e. bacteria killer) instead of bacteriostatic (i.e. bacteria growth inhibitor). Most

AMPs work directly against microbes through a mechanism which starts with membrane disruption and subsequent pore formation, allowing efflux of essential ions and nutrients (illustration in Figure 5). According to current view this mechanism works as follows: AMPs bind to the cytoplasmic membrane and create micelle-like aggregates, which leads to disruption of the membrane. In addition, there may be complementary mechanisms such as intracellular targeting of cytoplasmic components crucial to proper cellular physiology. Thus, the initial interaction between the peptides and the microbial cell membrane allows the peptides to penetrate into the cell to disrupt vital processes, such as cell wall biosynthesis and DNA, RNA, and protein synthesis. A convenient property of AMPs is their selective toxicity to microbial targets, which makes them non-toxic to mammalian cells. This specificity is based on the significant distinctions between mammalian and microbial cells, such as composition, transmembrane potential, polarization and structural features.

Antimicrobial peptides are small, positively charged, amphipathic molecules. They include two or more positively charged residues and a large proportion of hydrophobic residues. Many AMPs exist in relatively unstructured conformations prior to interaction with target cells. Upon binding to pathogen membranes, peptides may undergo significant conformational changes to helical or other structures. These conformations of antimicrobial peptides may impact their selective toxicity [107]. The three-dimensional folding of the peptides results in the hydrophilic or charged amino acids segregating in space from the hydrophobic residues, leading to either an amphipathic structure, or a structure with two charged regions spatially separated by a hydrophobic segment. Such a structure can interact with the membrane [27]. The amphipathicity of the AMPs enables insertion into the membrane lipid bilayer.

3 MACHINE LEARNING BACKGROUND

Machine learning [28] is a branch of artificial intelligence, which deals with the construction and study of systems that can learn from data. The most important questions addressed in machine learning are how to represent data and how to guarantee *generalization* of learnt systems. Generalization corresponds to the ability of learnt systems to perform well on unseen data instances. Typically, there are two modes in which machine learning systems operate: *learning* and *prediction*. In the former mode, parameters of the system are tuned on *learning examples*. In the latter mode, the learnt system is used for predicting given target variables. A system that implements classification is known as a classifier and a system that implements prediction of a continuous variable is known as a regression model.

There are many types of machine learning algorithms:

- Decision tree learning algorithms,
- Artificial neural networks,
- Support vector machines,
- Ensemble methods (e.g. Ada-boost or Random forest),
- etc.

These types of algorithms have been already used for prediction of DNA-binding propensity (see Section 3.2), which is the problem we are interested in.

3.1 RELATIONAL MACHINE LEARNING

Relational machine learning is a subfield of artificial intelligence, machine learning and data mining that is concerned with models of domains that exhibit complex, relational structure. Typically, the knowledge representation formalisms developed in relational learning use first-order logic to describe relational properties of a domain in a general manner.

3.1.1 *Logic-based Relational Learning*

In this thesis, we focus on relational learning algorithms based on first-order-logic - on so-called inductive logic programming [76]. The basic inductive logic programming problem is given as follows. We are given background knowledge B in the form of a clausal theory and a set of examples, which are first-order-logic clauses. The task is to find a hypothesis H in the form of a clausal theory such that H *covers* all positive and no negative examples. More formally, the task is to find a theory H such that $B \wedge H \models e$ for all positive e and $B \wedge H \not\models e$ for all negative examples e. This setting is known as intensional inductive logic programming [13] and is used in classical ILP systems (Aleph or Progol [63]).

Example 1 (Intensional Inductive Logic Programming). *Let us have background knowledge*

$$B = \{\mathsf{hasBenzeneRing}(\mathsf{phenylalanine})\}$$

and a set of positive examples (containing just one example in this case)

$$E^+ = \{\mathsf{isAromatic}(\mathsf{phenylalanine})\}$$

and a set of negative examples (containing also only one example)

$$E^- = \{\mathsf{isAromatic}(\mathsf{glycine})\}.$$

Then a solution to the learning task under intensional inductive logic programming setting with the given sets of positive and negative examples is for example

$$H = \forall X : \mathsf{hasBenzeneRing}(X) \rightarrow \mathsf{isAromatic}(X).$$

We can easily verify that it holds

$$H \wedge B \models \mathsf{isAromatic}(\mathsf{phenylalanine})$$

and

$$H \wedge B \not\models \mathsf{isAromatic}(\mathsf{glycine}).$$

The crisp ILP approach for learning from structured data, as described above, is not very useful from the practical point of view. It does not deal well with uncertainty arising either from noise which is often present in real-life problems or from an inherent probabilistic nature of some learning problems. It also does not provide many ways to control the generalization performance of the learned theories. Both learning in the presence of uncertainty as well as controlling of the generalization performance are topics that have been thoroughly studied in the field of attribute-value learning. *Propositionalization* [55] is a general strategy developed in the field of relational learning in order to exploit multitude of methods from attribute-value learning. Propositionalization systems can, in principle, exploit any method from attribute-value machine learning. These systems get a relational learning problem on their input and produce an attribute-value representation of these examples - they create an attribute-value table in which rows correspond to examples and columns correspond to first-order-logic formulas. These formulas typically acquire a form such as the one below

$$F1(Mol) \leftarrow \mathsf{hasBenzeneRing}(Mol, C) \wedge \mathsf{connected}(C, A) \wedge \mathsf{isBromine}(A)$$

where Mol is used as an identifier (key) of the learning examples. The feature $F1$ acquires the value *true* for the examples (molecules) which contain a benzene ring connected to a bromine atom. A set of generated features then plays the role of the attribute set for the attribute-value representation. The range of all possible features is, for a given learning problem, usually constrained by declaring a language of admissible features. Furthermore, propositionalization systems usually provide means to further restrict the set of features. For example, it is possible to set a *minimum frequency* of features [15], maximum length of features etc. Once a sufficiently rich set of features is found, the attribute-value table is constructed and it is fed into an attribute-value learning system such as decision trees or support vector machines [8].

In this thesis, we assume learning examples to be ground Horn clauses all sharing the same predicate p with arity 0 in the head. Hypotheses are composed of function-free Horn clauses with predicate p in the head. We also assume that the clauses in hypotheses are non-recursive and that there is no background knowledge (i.e. $B = \emptyset$). A hypothesis H is said to *cover* an example e if $H \models e$. We demonstrate how the learning problem from Example 1 would be represented in this simplified setting.

Example 2. *Let us have a set of positive examples* E^+ *containing the example*

$$\mathrm{isAromatic} \leftarrow \mathrm{is}(\mathrm{phenylalanine}) \wedge \mathrm{hasBenzeneRing}(\mathrm{phenylalanine})$$

and a set of negative examples E^- *containing the example*

$$\mathrm{isAromatic} \leftarrow \mathrm{is}(\mathrm{glycine}) \wedge \mathrm{hasBenzeneRing}(\mathrm{phenylalanine}).$$

Then a solution to the learning task in this simplified setting with the given sets of positive and negative examples is for example

$$H = \forall X : \mathrm{isAromatic} \leftarrow \mathrm{is}(X) \wedge \mathrm{hasBenzeneRing}(X).$$

We can easily verify that it holds

$$H \models \mathrm{isAromatic} \leftarrow \mathrm{is}(\mathrm{phenylalanine}) \wedge \mathrm{hasBenzeneRing}(\mathrm{phenylalanine})$$

and

$$H \not\models \mathrm{isAromatic} \leftarrow \mathrm{is}(\mathrm{glycine}) \wedge \mathrm{hasBenzeneRing}(\mathrm{phenylalanine}).$$

In the case of this simplified setting, we will abstract from clause heads. For brevity of formulas, we will display clauses as sets of literals separated by commas. The hypothesis H and the positive example from Example 2 will be written as

$$H = \mathrm{is}(X), \mathrm{hasBenzeneRing}(X)$$

and

$$e_1 = \mathrm{is}(\mathrm{phenylalanine}), \mathrm{hasBenzeneRing}(\mathrm{phenylalanine}).$$

Another advantage of this simplified setting is that entailment can be checked for non-recursive function-free clauses using decidable θ-subsumption described in Section 3.1.2.

3.1.2 θ-*subsumption as Approximation of Implication*

A problem of using the covering relation $\models$ is that it is undecidable. Therefore, Plotkin [75] introduced the concept of θ-subsumption as an incomplete approximation of implication.

Definition 1 (θ-subsumption)**.** *Let* A *and* B *be clauses. The clause* A θ*-subsumes* B*, if and only if there is a substitution* θ *such that* $A\theta \subseteq B$*. If* $A \preceq_\theta B$ *and* $B \preceq_\theta A$*, we call* A *and* B θ*-equivalent (written* $A \approx_\theta B$*).*

It can happen that two clauses are logically equivalent, but one of them is much larger than the other. Therefore, it is useful to have tools for finding an equivalent, reduced clause for a given large clause. Next, we define θ-reduction of a clause, which can be used for this purpose.

Definition 2 (θ-Reduction)**.** *Let* A *be a clause. If there is another clause* R *such that* $A \approx_\theta R$ *and* $|R| < |A|$ *then* A *is said to be* θ*-reducible. A minimal such* R *is called* θ*-reduction of* A*.*

Both θ-subsumption and θ-reduction are *NP-hard problems* and both of them need to be evaluated many times during a single run of a relational learning system. *Constraint satisfaction* [14] with finite domains represents a class of problems closely related to the θ-subsumption problems. Constraint satisfaction algorithms can be used to compute θ-subsumption and θ-reduction relatively efficiently (though, still with *exponential runtime* in the worst case).

A constraint satisfaction problem is a triple $(\mathcal{V}, \mathcal{D}, \mathcal{C})$, where $\mathcal{V}$ is a set of variables, $\mathcal{D} = \{D_1, \ldots, D_{|\mathcal{V}|}\}$ is a set of domains of values (for each variable $v \in \mathcal{V}$), and $\mathcal{C} = \{C_1, \ldots, C_{|\mathcal{C}|}\}$ is a set of constraints. Every constraint is a pair (s, R), where s (*scope*) is an n-tuple of variables and R is an n-ary relation. An evaluation of variables θ satisfies a constraint $C_i = (s_i, R_i)$ if $s_i\theta \in R_i$. A solution is an evaluation that maps all variables to elements in their domains and satisfies all constraints.

CSP can be used for relatively efficient solving of θ-subsumption problems thanks to many heuristics and filtering algorithms existing in the field of CSP. The CSP representation of the problem of deciding $A \preceq_\theta B$ has the following form. There is one CSP variable X_v for every variable $v \in vars(A)$. The domain of each of these CSP variables contains all terms from $terms(B)$. The set of constraints contains one k-ary constraint $C_l = (s_l, R_l)$ for each literal $l = pred_l(t_1, \ldots, t_k) \in A$. We denote by $I_{var} = (i_1, \ldots, i_m) \subseteq (1, \ldots, k)$ the indexes of variables in arguments of l (the other arguments might contain constants). The scope s_l of the constraint C_l is $(X_{t_{i_1}}, \ldots, X_{t_{i_m}})$ (i.e. the scope contains all CSP variables corresponding to variables in the arguments of literal l). The relation R_l of the constraint C_l is then constructed in three steps.

1. A set L_l is created which contains all literals $l' \in B$ such that $l \preceq_\theta l'$ (note that checking θ-subsumption of two literals is a trivial linear-time operation).

2. Then a relation R_l^* is constructed for every literal $l \in A$ from the arguments of literals in the respective set L_l. The relation R_l^* contains a tuple of terms $(t_1', \ldots, t_k')$ if and only if there is a literal $l' \in L_l$ with arguments $(t_1', \ldots, t_k')$.

3. Finally, the relation R_l of the constraint C_l is the projection of R_l^* on indexes I_{var} (only the elements of tuples of terms which correspond to variables in l are retained).

Example 3 (Converting θ-subsumption to CSP)**.** *Let us have clauses* A *and* B *as follows*

$$A = hasAminoAcid(C), bond(C, L), charge(L, positive)$$

$$B = hasAminoAcid(c), bond(c, l_1), bond(c, l_2), charge(l_2, positive).$$

We now show how we can convert the problem of deciding $A \preceq_\theta B$ *to a CSP problem. Let* $\mathcal{V} = \{C, L\}$ *be a set of CSP-variables and let* $\mathcal{D} = \{D_C, D_L\}$ *be a set of domains of variables from* $\mathcal{V}$ *such that* $D_C = D_L = \{c, l_1, l_2\}$*. Further, let*

$$\mathcal{C} = \{C_{hasAminoAcid(C)}, C_{bond(C,L)}, C_{charge(L,positive)}\}$$

be a set of constraints with scopes (C)*,* (C, L) *and* (L) *and with relations* $\{(c)\}$*,* $\{(c, l_1), (c, l_2)\}$ *and* $\{(l_2)\}$*, respectively. Then the constraint satisfaction problem given by* $\mathcal{V}$*,* $\mathcal{D}$ *and* $\mathcal{C}$ *represents the problem of deciding* $A \preceq_\theta B$ *as it admits a solution if and only if* $A \preceq_\theta B$ *holds.*

There is a theory categorizing tractable subclasses (i.e. solvable in polynomial time) of constraint satisfaction problems. For example, CSPs with *bounded treewidth* represent such a tractable class. Tractability of CSPs can be naturally translated into tractability of θ-subsumption. Therefore, a possible strategy for fast relational learning is to limit admissible hypotheses to be from a given tractable subclass.

The treewidth of CSPs, which is a measure of their tractability, can be defined using the notion of Gaifman graphs. The Gaifman (or primal) graph of a clause A is the graph with one vertex for each variable $v \in vars(A)$ and an edge for every pair of variables $u, v \in vars(A)$, $u \neq v$ such that u and v appear in a literal $l \in A$. Similarly, we define Gaifman graphs for CSPs. The Gaifman graph of a CSP problem $\mathcal{P} = (\mathcal{V}, \mathcal{D}, \mathcal{C})$ is the graph with one vertex for each variable $v \in \mathcal{V}$ and an edge for every pair of variables which appear in a scope of some constraint $c \in \mathcal{C}$. Gaifman graphs can be used to define treewidth of clauses or CSPs.

Tree decomposition, Treewidth

A tree decomposition of a graph $G = (V, E)$ is a labelled tree T such that

- Every node of a tree T is labeled by a non-empty subset of vertices V.

- For every edge $(v, w) \in E$, there is a node of the tree T with label containing v, w.

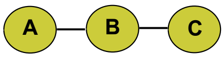
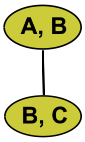
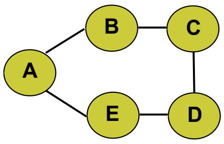
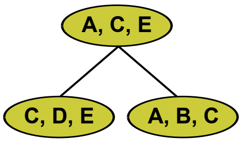
Clause	Gaifman graph	Tree decomposition
$\mathsf{atm}(A, h)$, $\mathsf{bond}(A, B, 1), \mathsf{atm}(B, c)$, $\mathsf{bond}(B, C, 2), \mathsf{atm}(C, o)$		
$\mathsf{bond}(A, B, 1)$, $\mathsf{bond}(B, C, 1), \mathsf{bond}(C, D, 1)$, $\mathsf{bond}(D, E, 1), \mathsf{bond}(E, A, 1)$		

Table 1: An illustration of Gaifman graphs and tree decompositions of clauses.

- For every vertex $v \in V$, the set of nodes of tree T with labels containing vertex v is a connected subgraph of T.

The width of a tree decomposition T is the maximum cardinality of a label in T minus 1. The treewidth of a graph G is the smallest number k such that G has a tree decomposition of width k. The treewidth of a clause is equal to the treewidth of its Gaifman graph. Analogically, the treewidth of a CSP is equal to the treewidth of its Gaifman graph.

An illustration of Gaifman graphs of two exemplar clauses and their tree decompositions is shown in Table 1. Note that tree decompositions are not unique. That is why treewidth is defined as the *maximum* cardinality of a label minus 1.

For instance, all trees have treewidth 1, cycles have treewidth 2, rectangular $n \times n$ grid-graphs have treewidth n. Treewidth is usually used to isolate tractable sub-classes of NP-hard problems.

Constraint satisfaction problems with treewidth bounded by k can be solved in polynomial time by the k-consistency algorithm. We now briefly describe the k-consistency algorithm [2]. Let us have a CSP $\mathcal{P} = (\mathcal{V}, \mathcal{D}, \mathcal{C})$ where $\mathcal{V}$ is the set of variables, $\mathcal{D}$ is the set of domains of the variables and $\mathcal{C}$ is the set of constraints. A partial solution ϑ is an evaluation of variables from $\mathcal{V}' \subseteq \mathcal{V}$ which is a solution of the sub-problem $\mathcal{P}' = (\mathcal{V}', \mathcal{D}, \mathcal{C})$. If ϑ and φ are partial solutions, we say that φ extends ϑ (denoted by $\vartheta \subseteq \varphi$) if $\mathsf{Supp}(\vartheta) \subseteq \mathsf{Supp}(\varphi)$ and $V\vartheta = V\varphi$ for all $V \in \mathsf{Supp}(\vartheta)$, where $\mathsf{Supp}(\vartheta)$ and $\mathsf{Supp}(\varphi)$ denote the sets of variables which are affected by the respective evaluations ϑ and φ. The k-consistency algorithm then works as follows:

k-consistency algorithm
Input: a constraint satisfaction problem $\mathcal{P} = (\mathcal{V}, \mathcal{D}, \mathcal{C})$ and a positive integer k

1. Let H be the collection of all partial solutions ϑ with $|\mathsf{Supp}(\vartheta)| < k+1$.

2. For every $\vartheta \in H$ with $|\mathsf{Supp}(\vartheta)| \leqslant k$ and every $V \in \mathcal{V}$, if there is no $\varphi \in H$ such that $\vartheta \subseteq \varphi$ and $V \in \mathsf{Supp}(\varphi)$, remove ϑ and all its extensions from H.

3. Repeat step 3 until H is unchanged.

4. If H is empty return *false*, else return *true*.

If the k-consistency algorithm returns *true* and $\mathcal{P}$ has treewidth bounded by k then $\mathcal{P}$ is guaranteed to have a solution [23]. For constraint satisfaction problems with generally unbounded treewidth, k-consistency is only a necessary but not a sufficient condition to have a solution. If the k-consistency algorithm returns *false* for a CSP problem $\mathcal{P}$ then $\mathcal{P}$ is guaranteed to have no solutions. If it returns *true* then the problem may or may not have some solutions. So, equivalently, if the problem is soluble then k-consistency always returns *true*. Another basic property of k-consistency that we will also need is the following. If the k-consistency algorithm returns *true* for a CSP problem then it will also return *true* for any problem created from the original problem by removing some variables and some constraints, i.e. with a *subproblem*.

It is easy to check that if a clause A has treewidth bounded by k then also the CSP representation of the problem of deciding $A \preceq_\theta B$ has treewidth bounded by k for any clause B. It is known that due to this and due to the equivalence of CSPs and θ-subsumption, the problem of deciding θ-subsumption $A \preceq_\theta B$ can be solved in polynomial time when clause A has bounded treewidth (which has been known for long time by the above mentioned widely known correspondence between θ-subsumption and CSP problems).

3.2 MACHINE LEARNING FOR DNA-BINDING PREDICTION

In the early 80's, when the first three-dimensional structures of protein-DNA complexes were studied, Ohlendorf and Matthew [69] noticed that the formation of protein-DNA complexes energetically driven by the electrostatic interaction of asymmetrically distributed charges on the surface of the proteins complement the charges on DNA. Large regions of positive electrostatic potentials on protein surfaces has been suggested to be a good indication of DNA-binding sites.

Stawiski et al. [85] proposed a methodology for predicting Nucleic Acid-binding function based on the quantitative analysis of structural, sequence and evolutionary properties of positively charged electrostatic surfaces. After defining the electrostatic patches they found the following features for discriminating the DNA-binding proteins from other proteins: secondary structure content, surface area, hydrogen-bonding potential, surface concavity, amino acid frequency and composition and sequence conservation. They used 12 parameters to train a neural network to predict the DNA-binding propensity of proteins.

Jones et al. [37] analysed residue patches on the surface of DNA-binding proteins and developed a method for predicting DNA-binding sites using a single feature of these surface patches. Surface patches and the DNA-binding sites were analysed for accessibility, electrostatic potential, residue propensity, hydrophobicity and residue conservation. They observed that the DNA-binding sites were amongst the top 10% of patches with the largest positive electrostatic scores.

Tsuchiya et al. [98] analysed protein-DNA complexes by focusing on the shape of the molecular surface of the protein and DNA, along with the electrostatic potential on the surface, and constructed a statistical evaluation function to make predictions of DNA interaction sites on protein molecular surfaces.

Ahmad and Sarai [1] trained a neural network based on the net charge and the electric dipole and quadrupole moments of the protein. It was found that the magnitudes of the moments of electric charge distribution in DNA-binding protein chains differ significantly from those of a non-binding control dataset. It became apparent that the positively charged residues are often clustered near the DNA and that the negatively charged residues either form negatively charged clusters away from the DNA or get scattered throughout the rest of the protein. The entire protein has a net dipole moment, because of the topological distribution of charges. The resulting electrostatic force may steer proteins into an orientation favorable for binding by ensuring that correct side of the protein is facing DNA.

Bhardwaj et al. [3] examined the sizes of positively charged patches on the surface of DNA-binding proteins. They trained a support vector machine classifier using positive potential surface patches, the protein's overall charge and its overall and surface amino acid composition. In case of overall composition, noticeable differences were observed between the binding and the non-binding case with respect to the frequency of Lys and Arg. These are positively charged amino acids, so their over-representation in DNA-binding proteins is evident.

A further advancement in DNA binding propensity prediction was presented by Szilágyi and Skolnick [92]. Their method was based on a logistic regression classifier with ten variables (physicochemical properties) to predict from sequence and low-resolution structure of a protein whether it is DNA-binding. To find features that discriminate between DNA-binding and non-DNA-binding proteins, they tested a number of properties. The best combination of parameters resulted in the amino acid composition, the asymmetry of the spatial distribution of specific residues and the dipole moment of the protein.

The above approaches rely exclusively on protein structure data (whether sequential or spatial). To our knowledge, the predictive accuracy achieved by the lastly mentioned strategy [92] was only improved by incorporating an additional source of background knowledge, in particular, information on evolutionarily conserved domains. Nimrod et al. [68] presented a random forest classifier for identifying DNA-binding proteins among proteins with known

3D structures. First, their method detects clusters of evolutionarily conserved regions on the surface of proteins using the PatchFinder algorithm. Next, a classifier is trained using features like the electrostatic potential, cluster-based amino acid conservation patterns, the secondary structure content of the patches and features of the whole protein, including all the features used by Szilágyi and Skolnick [92].

It is nevertheless important to continue improving methods that do not exploit evolutionary information. Such methods are valuable mainly due to their ability to predict DNA-binding propensity for engineered proteins for which evolutionary information is not available. Engineered proteins are highly significant for example in emerging gene-therapy technologies [10].

3.3 MACHINE LEARNING FOR ANTIMICROBIAL ACTIVITY PREDICTION

Several methods have been developed to predict antimicrobial activity of AMPs with potential therapeutic application. Some algorithms take advantage of data mining and high-throughput screening techniques and apply attribute-value approach to scan protein and peptide sequences [53, 96]. Similar strategies were proposed based on supervised learning techniques, such as artificial neural networks or support vector machines, in order to evaluate amounts of complex data [34]. Most attempts have been focused to the prediction of peptide's activity using quantitative structure-activity relationships (QSAR) descriptors together with artificial neural networks [35, 21, 11], linear discriminant [94] or principal component analysis [93]. A QSAR-based artificial neural network system was experimentally validated using SPOT high-throughput peptide synthesis, demonstrating that this methodology can accomplish a reliable prediction [20]. Recently, an artificial neural network approach based on the peptide's physicochemical properties has been introduced [97]. These properties were derived from the peptide sequence and were suggested to comprise a complete set of parameters accurately describing antimicrobial peptides.

Part II

DATA USED IN THE THESIS

4 DESCRIPTION OF THE USED DATASETS

4.1 DNA-BINDING PROTEINS

We worked with the following datasets of DNA-binding proteins in our experiments:

- PD138 - dataset of 138 DNA-binding protein structures in complex with DNA,
- UD54 - dataset of 54 DNA-binding protein structures in unbound conformation,
- BD54 - dataset of 54 DNA-binding protein structures in DNA-bound conformation corresponding to the set UD54
- APO104 - dataset of 104 DNA-binding protein structures in unbound conformation,
- ZF - dataset of 33 Zinc Finger protein structures in complex with DNA,

Dataset PD138 was created using the Nucleic Acid Database (NDB) by Szilágyi and Skolnick [92] - it contains a set of DNA-binding proteins in complex with DNA strands with a maximum pairwise sequence identity of 35% between any two sequences.

Both the protein and the DNA can alter their conformation during the process of binding. This conformational change can involve small changes in side-chain location, and also local refolding, in case of the proteins. Predicting DNA-binding propensity from a structural model of a protein makes sense if the available structure is not a protein-DNA complex, i.e. it does not contain a bound nucleic acid molecule. In order to find out how the results would change according to the conformation before and after binding, we used two other datasets (UD54, BD54). BD54 contains bound conformations of DNA-binding proteins, i.e. DNA-protein complexes. UD54 contains the same sequences in their unbound, free conformation. These datasets were also obtained from Szilágyi and Skolnick [92].

Another set of DNA-binding protein structures (APO104) determined in the absence of DNA was obtained from Gao et al. [25].

Thirty-three examples of Cys_2His_2 ZF-DNA complexes were sourced from Siggers et al. [82]. Their structural description was obtained from the *Protein Data Bank*.

From the structural description of each protein we extracted the list of all contained residues with information on their type and the list of pairwise spatial distances among all residues. As for the physicochemical features, we followed Szilágyi and Skolnick's work [92] and extracted features indicating the respective proportions of the Arg, Lys, Asp, Ala and Gly residues, the spatial asymmetry of Arg, Gly, Asn and Ser, and the dipole moment of the protein.

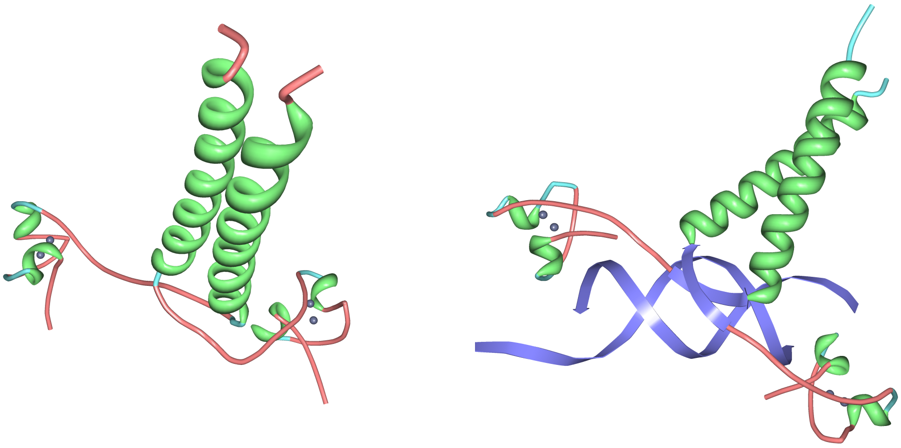


Figure 6: Crystal structures of protein PUT3 in its unbound versus bound conformation (in the Protein Database these conformations are listed as 1AJY and 1ZME).

4.2 NON-DNA-BINDING PROTEINS

We used the following datasets of non-DNA-binding proteins in our experiments:

- NB110 - dataset of 110 non-DNA-binding protein structures,
- NB843 - dataset of 843 non-DNA-binding protein structures.

Rost and Sander constructed a dataset (RS126) for secondary structure prediction. Ahmad & Sarai [1] removed the proteins related to DNA binding from it, thus getting a final dataset of non-DNA-binding proteins. As our negative dataset (NB110) we used this set of non-DNA-binding proteins.

We also used an extended dataset (NB843) by Nimrod et al. [68]. This dataset contains additional 733 structures of non-DNA-binding proteins. The additional structures were gathered using the PISCES server. Entries in this list include crystal structures with a resolution better than 3.0Å. The sequence identity between each pair of sequences is smaller than 25%.

4.3 ANTIMICROBIAL PEPTIDES

We used three datasets of antimicrobial peptides in our experiments:

- CAMEL,
- RANDOM,

- BEE.

The first dataset named CAMEL was described by Cherkasov et al. [11]. It is composed of 101 antimicrobial peptides with experimentally tested antimicrobial potency. These peptides are rich in leucine and it has been demonstrated that they exhibit high activity against various strains of bacteria. The minimal inhibitory concentrations for these peptides have been averaged over 13 microorganisms (*Bacteroides, Bordetella, Campylobacter, Corynebacterium, Klebsiella, Listeria, Moraxella, Pastuerella, Taylorella, Yersinia, Rhodococcus, Staphylococcus and Streptococcus*). The average minimal inhibitory concentrations (MIC) were used to calculate average potencies (which is our target variable) according to formula from [60]

$$\mathrm{Potency} = \log_2 \frac{1066}{\mathrm{MIC}}.$$

The second dataset named RANDOM was presented by Fjell et al. [20]. It contains 200 peptides with fixed length which are composed of a few amino acids (TRP, ARG and LYS and, more limitedly, LEU, VAL and ILE). Although antimicrobial peptides are actually enriched in these residues, a wide diversity in the amino acid content can be found in natural antimicrobial peptides [7]. The peptides were assayed for antimicrobial activity using a strain of *Pseudomonas aeruginosa*. Fjell et al. did not report absolute MIC values, but only MIC values divided by MIC of Bac2A peptide (to simplify the measurements). Using relative MIC values poses no problem, because it manifests itself only through addition of a constant to the potency values (due to the logarithm).

We named the last dataset BEE. We compiled it from three different sources: peptides from the venom of the eusocial bee Halictus sexcinctus and their analogs [61], peptides from the venom of the eusocial bee Lasioglossum laticeps [100] and peptides from the venom of the cleptoparasitic bee Melecta albifrons [99]. They contain peptides of length ca. 5 - 15 amino acids. The minimal inhibitory concentrations for these peptides were obtained for *Bacillus subtilis, Escherichia coli, Staphylococcus aureus, Pseudomonas aeruginosa*. We used the average of these values following the methodology of previous works [11, 20]. In some cases, when only lower bounds on MIC were available, we used these values.

Part III

RELATIONAL LEARNING APPROACHES

5 RELATIONAL LEARNING BASED ON STRUCTURAL PATTERNS

Relational machine learning is concerned with models of domains that exhibit complex, relational structure. Typically, the knowledge representation formalisms developed in relational learning use first-order logic to describe relational properties of a domain in a general manner. An advantage of general relational learning systems based on logic is the relative ease of setting up experiments. A disadvantage is that general systems usually scale poorly. In this chapter, we describe a relational learning method for prediction based on structural patterns, which builds upon the pattern construction algorithm RelF [42]. The presented relational learning method is based on our novel relational encoding of molecular structures. Although RelF had been shown to scale better than other relational learning systems, our preliminary experiments presented in Section 5.3.1 showed that it was still necessary to upgrade it for the domains of interest in this thesis. The necessary upgrades were twofold. First, we needed to introduce so-called counting patterns as opposed to existential patterns originally available in RelF. Second, we had to extend RelF with sampling strategies because RelF was unable to process datasets containing more than a few hundred proteins. With these upgrades we were able to achieve state-of-the-art accuracies.

This chapter is organized as follows. The novel relational representation of proteomics data is presented in Section 5.1. We describe our propositionalization-based learning method in Section 5.2. We present our experimental results in the domain of DNA-binding proteins in Section 5.3. In subsection 5.3.1 we describe the difference between existential and counting patterns and subject them to comparative experiments. These experiments show that counting patterns are superior to existential patterns (at least when patterns are small, which is the case for patterns constructed by RelF). Therefore, in the next subsection 5.3.2 only counting patterns are used and subjected to more in-depth experimental evaluation. We explore the obtained structural patterns and interpret them w.r.t. current biological knowledge. In Section 5.4 we describe our relational learning method modified for regression problems and show experimental results in the domain of antimicrobial peptides. In Section 5.5 we conclude this chapter. In the last section – Section 5.6 we present the algorithmic details of this chapter.

5.1 RELATIONAL REPRESENTATION OF DATA

The novel representation of examples that we use is rooted in the field of inductive logic programming [76] as introduced in Chapter 3. In the present experimental context, *patterns* which correspond to single-clause hypotheses (see Section 3.1.1), are used to capture spatial or other configurations of amino acids or individual atoms in protein structures. A pattern is a set of literals which, unlike examples, may contain variables. An example of a pattern is

$$p_1 = residue(A, X), distance(A, B, 10.0), residue(B, glu).$$

A pattern p is said to *cover* an example e when we are able to find a substitution θ to variables of p such that $p\theta \subseteq e$ (this is known as θ-subsumption in relational learning). For example the pattern p_1 covers the example e_1 because $p_1\theta \subseteq e_1$ for substitution $\theta = \{A/b, B/a\}$. We may be interested not only whether a pattern p covers a given example e but also how many *covering substitutions* there are, i.e. how many substitutions θ such that $p\theta \subseteq e$ there are. We call the number of covering substitutions of a pattern p its *value* and the patterns that count substitutions *counting patterns*.

What can be captured by patterns depends on the relational representation of the protein structures. In a pioneering work of Nassif et al. [67] a relational learning representation of hexose-binding proteins based on detailed atomic description was used. This representation did not involve description of amino acid types. The sole information available in their representation consists of the atom types and pairwise distances between these atoms. In principle, one would not need more information, but due to the large size of protein structures Nassif et al. had to subsample the relational descriptions using so-called *recall* mechanism in Aleph (in order to be able to process these structures). This is problematic, because a lot of information can be lost in this subsampling.

Our representation does not work on the atomic level. Instead it relies on the assumption that positions of individual amino acids in proteins can be more or less captured by positions of their α-carbons and directional vectors pointing from the α-carbons to the geometrical centroids of side-chains. Naturally, the very detailed atomic configurations can be lost in this representation, but first, this seems to hardly matter for DNA-binding propensity prediction (which we verify in our experiments in Section 5.3) and second, the detailed atomic configurations can be often captured implicitly in this representation anyway. Our representation is based on listing the amino acids and their types, pairwise spatial distances between these amino acids up to a maximum distance limit and, in some experiments, also angles between directional vectors of pairs of amino acids.

An illustration of a pattern in our relational learning representation is shown in Figure 7. Here, the pattern assumes the presence of two cysteines (*Cys* - A, B), two histidines (*His* - C, D) and one arginine (*Arg* - E), where the cysteine A is in distance 6Å from the cysteine B, 8Å from the histidine C, 10Å from the histidine D and 10Å from the arginine E. The listed distances do not refer to exact distances between the amino acids, but to their discretized values. Unless stated otherwise we use simple discretization with resolution 2Å in the experiments described in this thesis (recall that the datasets mostly include protein crystal structures with a resolution only better than 3Å).

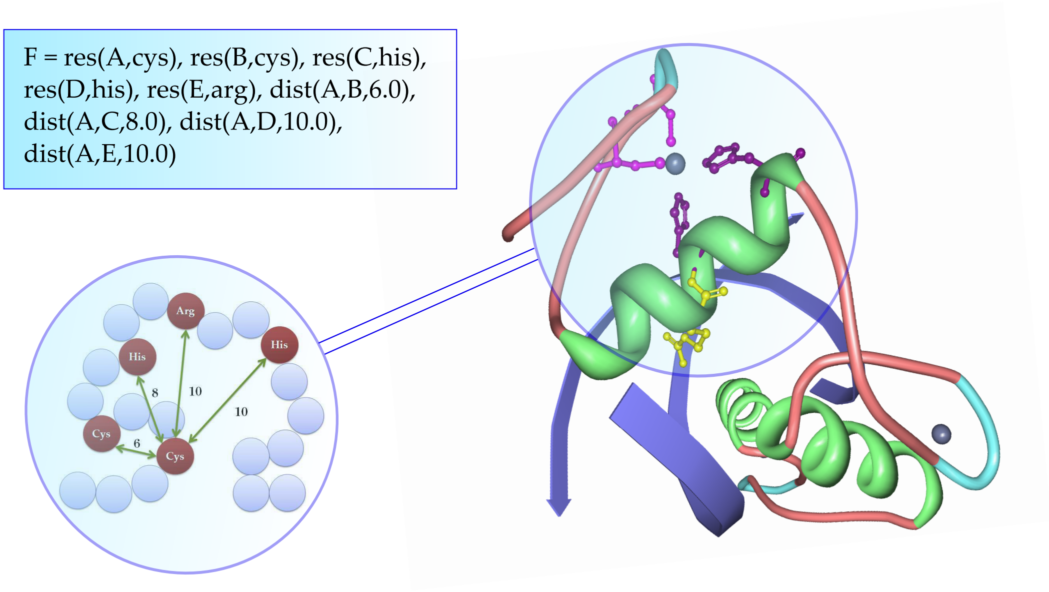

F = res(A,cys), res(B,cys), res(C,his), res(D,his), res(E,arg), dist(A,B,6.0), dist(A,C,8.0), dist(A,D,10.0), dist(A,E,10.0)
Arg
His
His
Cys
Cys
8
10
10
6

Figure 7: A match of a relational pattern shown in the left within a zinc finger protein (1A1F) shown in the right (using the protein viewer software [62]). Amino acids assumed by the pattern are indicated in the following way: CYS - pink, HIS - violet, ARG - yellow.

5.2 PROPOSITIONALIZATION-BASED LEARNING METHOD

Our method exploits techniques of relational machine learning [76] in conjunction with state-of-the-art attribute-value learning algorithms [28]. Our method can be viewed as proceeding in three steps. It starts with PDB files, which is a widely used format for proteins. Then it creates a relational representation of the proteins described in Section 5.1 (*step 1*). After that it tries to extract meaningful relational patterns from the relational structures describing proteins and uses them to create an approximate attribute-value representation of the proteins (*step 2*) which is then used for learning attribute-value classifiers (*step 3*).

Although the field of attribute-value machine learning is more mature than the field of relational machine learning, attribute-value learning algorithms, such as decision trees or support vector machines, suffer from the limitation that they can deal only with data which is in the form of data tuples (such as real-valued or boolean vectors) of fixed length. Attribute-value learning algorithms face problems when dealing with data in a more structured form, for example spatial structures of proteins. On the other hand, relational learning algorithms can directly learn from data expressed as relational structures such as graphs or the logic-based form which we adopt. Spatial structures of proteins, which is what we are interested in, can be represented very naturally within the relational-learning framework.

Propositionalization [55] (described in Chapter 3) is a general strategy which combines advantages of attribute-value learning algorithms (usually higher accuracy) and relational learning algorithms (ability to handle structured examples). In propositionalization, one tries to convert a relational learning problem to an attribute-value learning problem by *transforming* the original relational representation to an (approximate) attribute-value representation, i.e. to representation where learning examples are represented as vectors of fixed size, and then to train an attribute-value classifier for such data. Thus, roughly speaking, propositionalization corresponds to steps 2 and 3 of our method.

We use the relational representation of proteins described in Section 5.1 that consists of literals representing types of the amino acids and literals representing pair-wise distances between the amino acids. These distances are computed from α-carbon coordinates obtained from PDB. We also restrict the shapes of possible patterns by insisting that the patterns have to be tree-like. Despite these simplifications, some of the examples may contain, in the end, tens of thousands literals which would be very challenging for common relational learning systems such as Aleph, not to mention that these systems do not allow computing numbers of covering substitutions. Therefore we customized the pattern search algorithm RelF which is more appropriate for problems of this size due to its pruning mechanisms and strong structural language bias (it constructs only tree-like patterns). This pattern search algorithm prunes pattern space using two measures: *redundancy* (described by Kuželka et al. [42]) and *minimum frequency* which is a minimum number of examples that must be covered by a pattern. An example of a tree-like pattern is res(A,arg), res(B,arg), res(C,lys), dist(A,B,10.0), dist(A,C,10.0). This particular pattern assumes the presence of two arginines – A and B – and one lysine – C. The distance between the arginines is 10Å, the distance between the arginine A and the lysine C is also 10Å. A pattern like this can be used as an attribute, counting the number of occurrences of this particular spatial configuration of amino acids in proteins. Although counting the number of covering substitutions is not very common in ordinary propositionalization approaches, it makes perfect sense for the problem of predicting DNA-binding propensity of proteins, since ability to bind DNA is often connected with count or proportion of atom-groups with certain properties (e.g. charged residues [37]). We verify the contribution brought by using counting patterns instead of existential patterns experimentally in Section 5.3.1. The algorithmic details describing the necessary upgrade of RelF towards counting patterns are described in Section 5.6.

The generated patterns can be used for classification using state-of-the-art attribute-value learning algorithms. They can be also combined with other types of attributes, for example physicochemical attributes such as those devised by Szilágyi and Skolnick [92].

In some cases, not even RelF can construct patterns because of the enormous size of some proteomics datasets. Therefore, we designed a sampling strategy that allows RelF to quickly construct meaningful sets of structural patterns. Unlike the subsampling of Nassif et al., our sampling method does not subsample literals of individual protein structures, but it samples on the level of learning examples. This sampling strategy is described in more detail in Section 5.6.

5.3 EXPERIMENTS WITH DNA-BINDING PROTEINS

In this section we describe experimental results of our relational learning method based on the relational representation of protein structures described in Section 5.1. The purpose of the presented experiments is to answer the following questions: (i) are existential patterns sufficient for accurate predictions or do we need the counting patterns? (ii) can our relational learning method outperform methods based on features selected by human experts? (iii) can our relational learning method bring interpretable insights?

In order to answer these questions we performed experiments with several datasets of protein structures including DNA-binding (in their bound and unbound conformations) and non-DNA-binding proteins. We compared classifiers based on structural patterns discovered by our method (SP) with classifiers based on 10 physicochemical features (PF) identified as most predictive from a set of manually constructed features by Szilágyi and Skolnick [92] and with classifiers based on both structural patterns and the physicochemical features (PSP). In all the experiments, the three types of patterns (SP, PF, PSP) were used for classification using six state-of-the-art attribute-value learning algorithms detailed in Section 5.3.3. We performed experiments with more than one attribute-value learning algorithm in order to get objective assessment of the different types of patterns.

In Section 5.3.1 we start by comparing existential patterns with counting patterns on a dataset of DNA-binding protein structures in their unbound conformations. Since the results clearly show that counting patterns significantly outperform the using of existential patterns, we decided to further investigate only relational learning using counting patterns. In Section 5.3.2 we present a broader experimental evaluation of our relational learning method based on counting patterns.

5.3.1 *Relational Learning using Existential Patterns vs. Counting Patterns*

Here, we compare counting patterns with existential patterns while using them in our relational learning method based on structural patterns. We performed experiments on a *positive dataset* of DNA-binding proteins in their unbound conformation (UD54) and a *negative dataset* of non-DNA-binding proteins (NB110). As a result of structural pattern searching we obtained about 1500 patterns present in the unbound DNA-binding proteins. We made two sets of trainings (accuracies are shown in Tab. 2): i) considering just the occurrence of the structural patterns - columns marked with (E), i.e. Existential Patterns, ii) considering also the number of occurrences of each pattern - columns marked with (C), i.e. Counting Patterns. We compared classifiers based on our structural patterns (SP) with classifiers based on 10 physicochemical features (PF) obtained from Szilágyi et al. [92]. We also trained classifiers based on both our structural patterns and physicochemical features of Szilágyi et al. (PSP). As we can see, we got better results for classifiers considering the number of occurrences of each pattern, i.e. using counting patterns. For the most classifiers the accuracy was higher when they were based on our counting structural patterns than on physicochemical features of Szilágyi et al. However, we got the best results with combination of the two feature-sets.

Classifier	**PF**	**SP(E)**	**PSP(E)**	**SP(C)**	**PSP(C)**
Linear SVM	84.0 (2)	77.5 (5)	78.1 (4)	83.0 (3)	**84.2 (1)**
SVM with RBK	81.6 (3)	67.1 (4-5)	67.1 (4-5)	83.0 (2)	**85.4 (1)**
Simple log. regr.	81.6 (3)	73.9 (5)	78.8 (4)	**87.6 (1)**	82.3 (2)
L_2-regularized log. regr.	84.0 (2)	78.7 (5)	80.5 (4)	82.4 (3)	**84.2 (1)**
Ada-boost	77.4 (4)	73.2 (5)	83.0 (2)	79.3 (3)	**84.7 (1)**
Random forest	78.6 (4)	76.8 (5)	**83.6 (1)**	80.5 (2)	79.9 (3)
Average ranking:	3	4.92	3.25	2.33	**1.5**

Table 2: Accuracies obtained by stratified 10-fold crossvalidation using physicochemical features of Szilágyi et al. (PF), our structural patterns (SP) and combination of both of them (PSP). The numbers in parentheses correspond to ranking w.r.t. the obtained accuracies. The results shown in this table differ slightly from the result shown in Table 4. The reason is that the results in this table were obtained using an older version of the algorithm RelF which selected a different but equivalent set of non-redundant features on training data.

It is evident that the existential patterns are not able to capture the properties which determine whether a protein can bind to DNA or not. The most plausible explanation seems to be that unlike binding of small ligands where local geometry plays major role, binding to DNA can be better predicted using global properties of proteins such as their electric charge or dipole moment. These global properties cannot be captured by small existential patterns, but they can be to a large extent captured by small counting patterns. For this reason, we further investigated only our relational learning method using counting patterns. This is described in the next section.

These results answer the first question asked at the beginning of Section 5.3, whether existential patterns are sufficient for accurate predictions or whether we need the counting patterns, in favour of the latter type of patterns.

5.3.2 *Relational Learning using Counting Patterns*

In this section we present results achieved by our relational learning method using counting patterns for prediction of DNA-binding propensity of proteins. We performed five sets of experiments with datasets of DNA-binding proteins - PD138, UD54, BD54, APO104 and ZF - each one as a set of *positive examples* and dataset of non-DNA-binding proteins NB110 - as a set of *negative examples*. We used the RelF algorithm to construct patterns with minimum frequency 0.7 and obtained about 1400 frequent structural patterns for datasets PD138/NB110, approximately 1500 frequent structural patterns for datasets UD54/NB110, about 2400 frequent structural patterns for datasets BD54/NB110, about 2800 frequent structural patterns for datasets APO104/NB110 and approximately 6000 frequent structural patterns for datasets ZF/NB110. Accuracies and areas under the ROC curve (AUC) obtained on the respective datasets by stratified 10-fold cross validation using physicochemical features (PF), structural patterns (SP) and combination of both of them (PSP) are shown in Tables 3, 4, 5 and 6. The results for the method based on physicochemical features (PF) differ slightly from the results reported by Szilágyi and Skolnick [92], because we used 10-fold cross-validation whereas Szilágyi and Skolnick used leave-one-out cross-validation.

We computed average rankings (over several machine learning algorithms) for accuracies and AUCs. The average ranking (over several machine learning algorithms) of classifiers based on structural patterns (SP) was best on datasets UD54/NB110, APO104/NB110 (tie with PSF) and ZF/NB110 for accuracies and on datasets UD54/NB110 and ZF/NB110 in terms of AUC. The average ranking of classifiers based on combination of structural patterns and physicochemical features (PSP) was highest on datasets PD138/NB110, APO104/NB110 (tie with SP) and BD54/NB110 for accuracies and on datasets PD138/NB110, APO104/NB110 and ZF/NB110 (tie with SP) in terms of AUC. Classifiers based on physicochemical features (PF) obtained highest ranking only for AUCs on dataset BD54/NB110.

	Accuracy			*AUC*		
PD138 vs. NB110	*PF*	*SP*	*PSP*	*PF*	*SP*	*PSP*
Simple log. regr.	**83.4 (1)**	82.2 (2)	80.7 (3)	0.91 (2)	0.90 (3)	**0.94 (1)**
L_2-reg. log. regr.	81.4 (3)	83.5 (2)	**85.5 (1)**	**0.92 (1)**	0.91 (2)	0.91 (2)
SVM with RBF	81.8 (2)	79.9 (3)	**85.1 (1)**	0.92 (2)	0.90 (3)	**0.93 (1)**
Linear SVM	81.4 (3)	83.6 (2)	**83.9 (1)**	0.92 (2)	0.89 (3)	**0.93 (1)**
Ada-boost	80.6 (2)	78.6 (3)	**81.4 (1)**	**0.90 (1)**	**0.90 (1)**	**0.90 (1)**
Random forest	81.8 (3)	**83.5 (1)**	82.3 (2)	0.90 (3)	0.91 (2)	**0.93 (1)**
Average ranking	2.33	2.17	**1.5**	1.83	2.33	**1.17**

Table 3: Predictive accuracies and areas under the ROC curve (*AUC*) on datasets PD138 vs. NB110 achieved by 6 machine learning algorithms using physicochemical features (*PF*) as proposed by Szilágyi and Skolnick [92], structural patterns (SP) automatically constructed by our algorithm, and the combination of both feature sets (*PSP*).

The method based on purely structural patterns (SP) and the method based on the combination of structural patterns and physicochemical features (PSP) achieved higher predictive accuracies than the method based purely on physicochemical features (PF) - features introduced by Szilágyi and Skolnick [92]. The only exception was in case of the dataset BD54/NB110, where the method

	Accuracy			*AUC*		
UD54 vs. NB110	*PF*	*SP*	*PSP*	*PF*	*SP*	*PSP*
Simple log. regr.	81.0 (3)	**86.0 (1)**	82.8 (2)	**0.91 (1)**	0.89 (2)	0.89 (2)
L2-reg. log. regr.	82.2 (3)	82.4 (2)	**84.1 (1)**	0.89 (3)	**0.91 (1)**	0.90 (2)
SVM with RBF	81.0 (2)	**84.0 (1)**	80.4 (3)	**0.92 (1)**	0.88 (3)	0.91 (2)
Linear SVM	81.7 (2)	**82.4 (1)**	**82.4 (1)**	0.90 (2)	**0.91 (1)**	0.87 (3)
Ada-boost	76.2 (3)	78.0 (2)	**79.3 (1)**	0.88 (3)	0.89 (2)	**0.90 (1)**
Random forest	78.6 (3)	**79.3 (1)**	79.2 (2)	0.88 (3)	0.89 (2)	**0.90 (1)**
Average ranking	2.67	**1.34**	1.67	2.17	**1.67**	2

BD54 vs. NB110	*PF*	*SP*	*PSP*	*PF*	*SP*	*PSP*
Simple log. regr.	80 (3)	80.5 (2)	**81.8 (1)**	**0.91 (1)**	0.85 (2)	**0.91 (1)**
L2-reg. log. regr.	**83.1 (1)**	81.9 (2)	81.7 (3)	**0.92 (1)**	0.88 (3)	0.91 (2)
SVM with RBF	82.5 (2)	82.5 (2)	**83.6 (1)**	**0.91 (1)**	0.90 (2)	0.90 (2)
Linear SVM	81.4 (3)	82.3 (2)	**82.9 (1)**	0.93 (2)	0.90 (3)	**0.94 (1)**
Ada-boost	**84.2 (1)**	73.8 (3)	79.8 (2)	**0.91 (1)**	0.88 (2)	0.88 (2)
Random forest	**82.4 (1)**	75.0 (3)	79.4 (2)	0.89 (2)	0.89 (2)	**0.91 (1)**
Average ranking	1.83	2.33	**1.67**	**1.33**	2.33	1.5

Table 4: Predictive accuracies and areas under the ROC curve (*AUC*) on datasets UD54 vs. NB110 and BD54 vs. NB110 achieved by 6 machine learning algorithms using physicochemical features (*PF*) as proposed by Szilágyi and Skolnick [92], structural patterns (SP) automatically constructed by our algorithm, and the combination of both feature sets (*PSP*).

based on purely physicochemical features performed better than the method based on purely structural patterns. The results were not as definite in the case of AUC as in the case of predictive accuracy. The method based on structural patterns turned out to be better than the method based on physicochemical features on three datasets. Interestingly, these two datasets contain DNA-binding proteins in their unbound conformations. The method based on the combination of structural patterns and physicochemical features was better than the method based on purely physicochemical features on four datasets.

It may seem counter-intuitive that in some of the experiments, physicochemical features (PF) or structural patterns (SP) outperformed the combined feature set (PSP). However, this is a rather natural manifestation of the overfitting effect; expansion of the feature set may indeed be detrimental especially with small datasets [28].

It is interesting to compare the results for the datasets UD54 and BD54. Dataset UD54 contains DNA-binding proteins in unbound conformation, data-

	Accuracy			***AUC***		
APO104 vs. NB110	*PF*	*SP*	*PSP*	*PF*	*SP*	*PSP*
Simple log. regr.	80.7 (3)	**85.0 (1)**	80.8 (2)	0.89 (3)	**0.92 (1)**	0.91 (2)
L_2-reg. log. regr.	82.6 (3)	**84.5 (1)**	83.1 (2)	0.90 (2)	**0.91 (1)**	**0.91 (1)**
SVM with RBF	79.4 (3)	83.2 (2)	**84.1 (1)**	0.88 (3)	0.90 (2)	**0.91 (1)**
Linear SVM	79.4 (3)	**84.5 (1)**	84.1 (2)	0.89 (2)	0.89 (2)	**0.92 (1)**
Ada-boost	77.6 (3)	78.1 (2)	**79.1 (1)**	0.87 (2)	0.87 (2)	**0.89 (1)**
Random forest	**81.7 (1)**	78.5 (3)	79.4 (2)	0.88 (2)	0.87 (3)	**0.89 (1)**
Average ranking	2.67	**1.67**	**1.67**	2.33	1.83	**1.17**

Table 5: Predictive accuracies and areas under the ROC curve (*AUC*) on datasets APO104 vs. NB110 achieved by 6 machine learning algorithms using physicochemical features (*PF*) as proposed by Szilágyi and Skolnick [92], structural patterns (SP) automatically constructed by our algorithm, and the combination of both feature sets (*PSP*).

set BD54 contains the same DNA-binding proteins, but in bound conformation with DNA. Whereas the highest predictive accuracies and best AUCs were obtained by the method based on structural patterns on dataset UD54, this method performed worst on dataset BD54. Interestingly, the number of frequent structural patterns was significantly higher for dataset BD54 (approximately 2400 structural patterns) than for the dataset UD54 (approximately 1500 structural patterns). This suggests that conformational changes after DNA-binding give rise to greater variability of spatial arrangements of some amino acid groups. Moreover, conformational changes may be responsible for increase of spatial asymmetry of some amino acids or protein's dipole moment. This can explain the better performance of the method based on physicochemical features on the dataset BD54 (recall that these features were selected by experimenting on DNA-binding proteins in bound conformation with DNA by Szilágyi and Skolnick [92]). Also note that prediction of DNA-binding propensity from unbound conformations, for which our method performed best, is more important for practical applications.

We examined the best discovered patterns in detail. For each split of the dataset PD138 induced by 10-fold cross-validation we selected the ten most informative structural patterns according to the χ^2 criterion. Table 7 shows the number of occurrences of the ten best structural patterns. There are four structural patterns which are present in all ten folds. The first is res(A), residue(A,arg). This pattern counts the number of arginines in the protein. It is known that the arginine plays an important role in the DNA binding process. For now, we are interested in structural patterns. Since this pattern included no spatial information relating to other amino acids, we decided to analyse just the remaining three patterns.

We inspected how structural patterns are reflected in protein's primary structure. First, we examined whether amino acids matched by a pattern occur in a preferred order in the proteins' sequences. We calculated the dis-

	Accuracy			*AUC*		
ZF vs. NB110	*PF*	*SP*	*PSP*	*PF*	*SP*	*PSP*
Simple log. regr.	95.1 (3)	**98.7 (1)**	97.2 (2)	0.99 (2)	**1.0 (1)**	**1.0 (1)**
L_2-reg. log. regr.	95.9 (3)	99.3 (2)	**100 (1)**	0.99 (2)	**1.0 (1)**	**1.0 (1)**
SVM with RBF	95.8 (3)	**99.3 (1)**	98.6 (2)	0.99 (2)	**1.0 (1)**	**1.0 (1)**
Linear SVM	81.4 (3)	**99.3 (1)**	97.8 (2)	**1.0 (1)**	**1.0 (1)**	**1.0 (1)**
Ada-boost	95.9 (3)	99.3 (2)	**100 (1)**	0.98 (2)	**1.0 (1)**	**1.0 (1)**
Random forest	96.5 (3)	**97.9 (1)**	97.2 (2)	0.99 (2)	**1.0 (1)**	**1.0 (1)**
Average ranking	3	**1.33**	1.67	1.83	**1**	**1**

Table 6: Predictive accuracies and areas under the ROC curve (*AUC*) on datasets ZF vs. NB110 achieved by 6 machine learning algorithms using physicochemical features (*PF*) as proposed by Szilágyi and Skolnick [92], structural patterns (SP) automatically constructed by our algorithm, and the combination of both feature sets (*PSP*).

tribution of permutations of the amino acids matched by the first analysed structural pattern res(A,arg), res(B,lys), dist(A,B,4.0). The distribution of permutations on positive dataset was almost identical. Next, we were looking for relative positions of these amino acids in the sequences of DNA-binding proteins. Mostly the amino acids were situated next to each other in the proteins' sequences for both permutations of amino acids: [arg,lys] and [lys,arg], i.e. on positions n and n+1. We also obtained occurrences of this pattern, where the amino acids were on positions n and n+3 for permutation [arg, lys].

The next analysed structural pattern was res(A,arg), res(B,arg), res(C,lys), dist(A,B,10.0), dist(A,C,10.0). There were no prevailing permutations for this structural pattern and also no prevailing local arrangements of amino acids in sequence. It would be hard to express this pattern using only primary structure information, unlike in the case of the previous pattern.

The last analysed structural pattern was res(A,arg), res(B,arg), dist(A,B,6.0). The most frequent relative positions of the amino acids were [n, n+2], [n, n+3], [n, n+4], where the first relative positions were approximately two times more frequent than the other two.

It is interesting that, with the exception of the first pattern, each structural pattern corresponds to multiple sequential patterns in the proteins' primary structure. For instance, the second pattern corresponded to more than 20 different sequential patterns that appeared at least twice in the dataset. Naturally, it would be hard to identify these sequential patterns in isolation using only the information about the proteins' primary structure. This could be related to the fact that spatial configurations of amino acids tend to be more evolutionary conserved than configurations in primary structure.

Finally, we performed an additional experiment involving the method of Nimrod et al. [68] on the dataset PD138/NB843. The method of Nimrod et al. exploits also evolutionary information therefore it is interesting to see

	STRUCTURAL PATTERN	N
1	res(A,arg)	10
2	**res(A,arg), res(B,lys), dist(A,B,4.0)**	10
3	**res(A,arg), res(B,arg), res(C,lys), dist(A,B,10.0), dist(A,C,10.0)**	10
4	**res(A,arg), res(B,arg), dist(A,B,6.0)**	10
5	res(A,arg), res(B,lys), dist(A,B,6.0)	9
6	res(A,ile), res(B,arg), res(C,arg), dist(A,B,6.0), dist(A,C,10.0)	7
7	res(A,leu), res(B,glu), res(C,arg), dist(A,B,10.0), dist(A,C,6.0)	7
8	res(A,lys), res(B,arg), dist(A,B,10.0)	7
9	res(A,arg), res(B,arg), res(C,leu), dist(A,B,10.0), dist(A,C,6.0)	7
10	res(A,arg), res(B,arg), dist(A,B,10.0)	6

Table 7: The ten most informative structural patterns according to the χ^2 criterion for the dataset PD138. **N** is the number of folds, for which the actual pattern was one of the ten best patterns.

whether methods relying only on physicochemical features and/or structural patterns could come close to its predictive accuracy.

In this additional experiment, we used only random forest classifier because this classifier was also used by Nimrod et al. The AUC values of the approaches based on the physicochemical features (PF), structural patterns (SP), and their combination (PSP) were (respectively) 0.84, 0.82, and 0.82, whereas the method of Nimrod et al. achieved AUC of 0.9. This indicates that there is still a large gap between the structural pattern and physicochemical feature based approaches on one hand, and methods relying on evolutionary conservation information.

The performed experiments allow us to evaluate usability of the relational learning approach for prediction of DNA-binding propensity as well as its usability for discovery of interesting spatial patterns in proteins. Results of our experiments suggest that the method is suitable for both of the tasks (thus, answering questions (ii) and (iii) asked at the beginning of Section 5.3). In Section 5.3.2.1 we also discuss the factors influencing predictive performance and biological relevancy of discovered structural patterns in case of Zinc finger proteins.

5.3.2.1 *Discussion of the Results for Zinc finger proteins*

Zinc finger proteins are one of the most common DNA-binding proteins in eukaryotic transcription factors. Several studies [16, 17, 18, 66, 95, 72, 19, 104] have tried to determine the DNA recognition by these proteins. The sequence of three fingers of the protein Zif268, which served as the prototype for understanding DNA recognition by this family of proteins, is shown with the cysteines and histidines involved in zinc coordination indicated in *bold* font in Table 8 (reproduced from Wolfe et al. [104]). *Filled squares* below the sequences indicate the position of the conserved hydrophobic residues. *Filled circles* and *stars* indicate residue positions that are involved in phosphate and base contacts (respectively) in most of the fingers. We evaluated relevance of the discovered structural patterns matching them to observations in the paper of Wolfe et al. [104].

We made predictive classification experiments on dataset of zinc finger proteins (ZF). The best results, in terms of accuracy and AUC, were obtained by the method based on structural patterns. However, here the results were influenced by the fact that the zinc finger proteins were highly homologous. Therefore, we were more interested in the question whether the structural patterns were able to discover some basic characteristic of DNA-binding process shared by zinc finger proteins.

We inspected the best discovered patterns. We selected the ten most informative structural patterns according to the χ^2 criterion, following the same procedure as for the DNA-binding proteins in general. Table 9 shows the number of occurrences of the ten best patterns. There were three structural patterns present in all of the dataset splits. We show them in Figures 8 and 9.

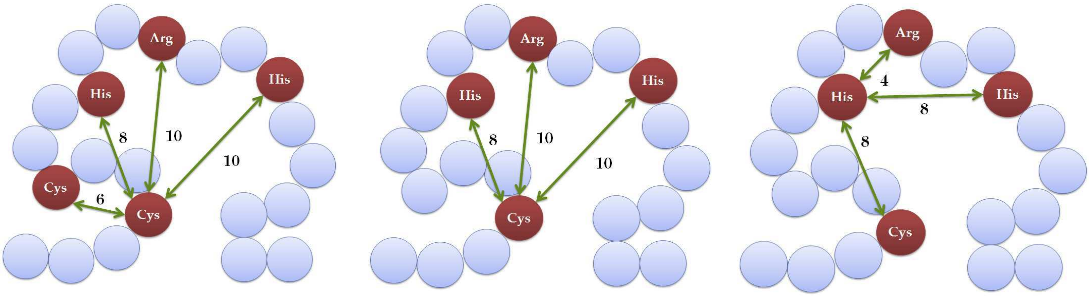


Figure 8: Most informative structural patterns according to the χ^2 criterion for the dataset of Zinc Fingers (edges not to scale).

															-1	1	2	3	4	5	6	7	8	9						
Finger 1	P	Y	A	**C**	P	V	E	S	**C**	D	R	R	F	S	R	S	D	E	L	T	R	**H**	I	R	I	**H**	T	G	Q	K
Finger 2	P	F	Q	**C**	R	I	-	-	**C**	M	R	N	F	S	R	S	D	H	L	T	T	**H**	I	R	T	**H**	T	G	E	K
Finger 3	P	F	A	**C**	D	I	-	-	**C**	G	R	K	F	A	R	S	D	E	R	K	R	**H**	T	K	I	**H**	L	R	Q	K
		■									●		■		⋆		⋆	⋆	■		⋆	●								●
				0	1	2	3	4	**5**	6	7	8	9	10	11	12	13	14	15	16	**17**	**18**	19	20	21	**22**				
P1				**C**					**C**												**R**	**H**				**H**				
P2				**C**																	**R**	**H**				**H**				
P3				**C**																	**R**	**H**				**H**				

Table 8: The sequence of the three fingers of Zif268 is shown with the cysteines and histidines involved in zinc coordination indicated in *bold* font. *Filled squares* below the sequences indicate the position of the conserved hydrophobic residues. *Filled circles* and *stars* indicate residue positions that are involved in phosphate and base contacts (respectively) in most of the fingers.

	Structural Pattern	N
1	**res(A,cys), res(B,cys), res(C,his), res(D,his), res(E,arg), dist(A,B,6.0), dist(A,C,8.0), dist(A,D,10.0), dist(A,E,10.0)**	10
2	**res(A,cys), res(B,his), res(C,his), res(D,arg), dist(A,B,8.0), dist(A,C,10.0), dist(A,D,10.0)**	10
3	**res(A,his), res(B,his), res(C,cys), res(D,arg), dist(A,B,8.0), dist(A,C,8.0), dist(A,D,4.0)**	10
4	res(A,cys), res(B,his), res(C,his), res(D,phe), dist(A,B,8.0), dist(A,C,10.0), dist(A,D,8.0)	9
5	res(A,his), res(B,cys), res(C,his), res(D,arg), dist(A,B,10.0), dist(A,C,8.0), dist(A,D,6.0)	9
6	res(A,his), res(B,cys), res(C,his), res(D,arg), dist(A,B,10.0), dist(A,C,8.0), dist(A,D,4.0)	8
7	res(A,cys), res(B,his), res(C,his), dist(A,B,8.0), dist(A,C,10.0)	8
8	res(A,cys), res(B,cys), res(C,his), res(D,his), res(E,phe), dist(A,B,6.0), dist(A,C,8.0), dist(A,D,10.0), dist(A,E,8.0)	8
9	res(A,his), res(B,cys), res(C,his), res(D,phe), res(E,cys), dist(A,B,10.0), dist(A,C,8.0), dist(A,D,10.0), dist(A,E,8.0)	5
10	res(A,his), res(B,cys), res(C,arg), res(D,his), dist(A,B,10.0), dist(A,C,10.0), dist(A,D,8.0)	3

Table 9: The ten most informative structural patterns according to the χ^2 criterion for the dataset of Zinc Finger proteins. **N** is the number of folds, for which the actual pattern was one of the ten best patterns.

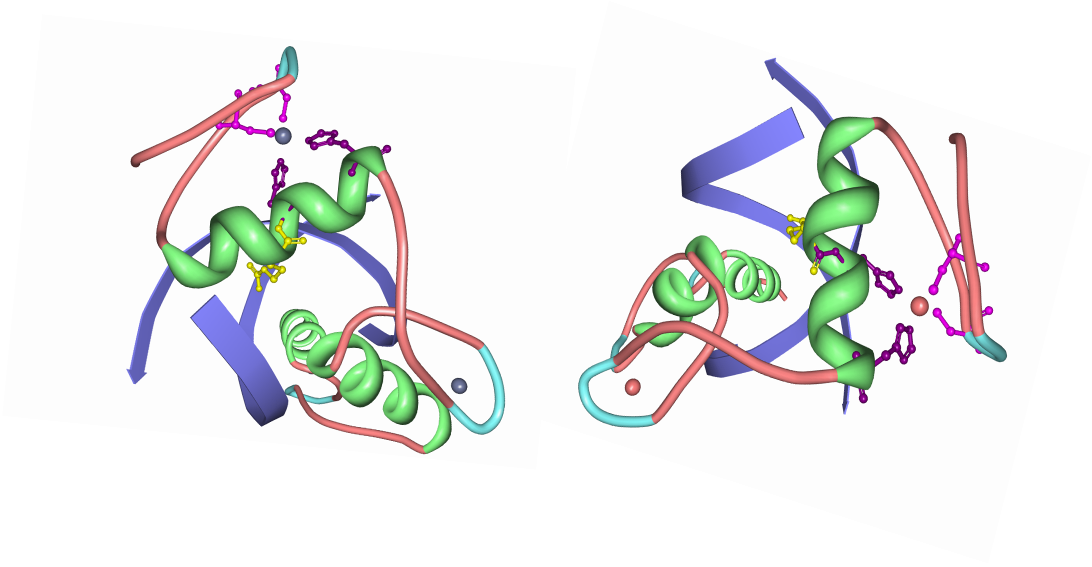


Figure 9: Example proteins (1A1F and 1AAY) containing one discovered pattern shown for the Zinc-finger proteins' dataset using the protein viewer software [62]. Residues assumed by the pattern are indicated in the following way: CYS - pink, HIS - violet, ARG - yellow.

We calculated the distribution of permutations of the amino acids matched by the first analysed structural pattern res(A,cys), res(B,cys), res(C,his), res(D,his), res(E,arg), dist(A,B,6.0), dist(A,C,8.0), dist(A,D,10.0), dist(A,E,10.0). The most frequent permutation was [cys, cys, arg, his, his]. We looked for the relative positions of these amino acids in zinc finger proteins' sequences. The most frequently occurring relative positions were: [n, n+5, n+17, n+18, n+22]. We compared this result with the observation described in the paper of Wolfe et al. [104] (reproduced in Table 8). This discovered structural pattern exactly matched the positions of some of the amino acids which are supposed to be directly involved in DNA-binding. In case of the second structural pattern res(A,cys), res(B,his), res(C,his), res(D,arg), dist(A,B,8.0), dist(A,C,10.0), dist(A,D,10.0) and the third structural pattern res(A,his), res(B,his), res(C,cys), res(D,arg), dist(A,B,8.0), dist(A,C,8.0), dist(A,D,4.0) the most frequent permutation was [cys, arg, his, his] and the resulting relative positions were [n, n+17, n+18, n+22]. Table 8 indicates that these two patterns (P2 and P3) cover the first pattern (P1).

While, as already commented, the discovered patterns matched the positions of some of the amino acids supposed to be directly involved in DNA-binding, they in fact do not capture specific properties of DNA-binding process but rather a consensus amino acid pattern known to be present in Cys_2His_2 zinc fingers [104]. One could be concerned whether the patterns discovered for DNA-binding proteins in general (datasets PD138, UD54, BD54, APO104) just captured conserved consensus patterns of different folds as well. However, this was not the case, because every discovered pattern was contained in at least 70% of DNA-binding proteins (note that minimum frequency 0.7 was used for feature construction). In order to assure validity of this claim we performed an additional experiment in which the relational learning model was always constructed for proteins from all but one protein group and then tested on this excluded group (see Section 5.3.2.2). Nevertheless, these observations indicate that caution should be exercised when applying our relational learning method on datasets with highly homologous proteins, because conserved consensus patterns not necessarily related to the function of the proteins could be discovered instead of the sought patterns responsible for the function.

5.3.2.2 *Evaluation of binding motif independence*

In order to further support our claim that the patterns discovered for DNA-binding proteins in general (datasets PD138, UD54, BD54, APO104) did not just capture the consensus patterns of particular folds, we performed an experiment in which the relational learning model was always constructed for proteins from all but one protein group and then tested on this excluded group. Proteins of the dataset PD138 were divided into seven groups following the work of Szilágyi and Skolnick [92]. They were the following: helix-turn-helix, zinc-coordinating, zipper-type, other α-helix, β-sheet, other and enzyme. We used linear SVM based on our structural patterns (SP), because SVM turned out to perform best in the experiments described in Section 5.3.2. We show both the predictive accuracies obtained by testing the learnt classifiers on the excluded groups and the cross-validated accuracies obtained by the classifiers on the remaining parts of the dataset in Table 10. The resulting accuracies on the excluded groups, which should correlate with the ability of our method to discover patterns characteristic for DNA-binding proteins in general, are reasonably high with the exception of the enzyme group. This agrees with the results of Szilágyi and Skolnick [92] and Stawiski et al. [85], who also noticed a drop in the ability of their method to detect DNA-binding proteins in the enzyme group. We can conclude that our method is indeed able to construct classifiers which can work accurately over various (non-enzyme) groups of proteins and that its ability to detect DNA-binding proteins is not due to discovery of conserved consensus patterns of different protein folds.

5.3.3 *Detailed Experimental Settings*

In all the experiments, the three types of patterns (PF, SP, PSP) were used for classification using six state-of-the-art attribute-value learning algorithms listed in Table 11. We used implementation of these learning algorithms present in the WEKA [103] open-source machine learning software. We performed experiments with more than one attribute-value learning algorithm in order to get objective assessment of the different types of patterns.

Parameters of the classifiers were tuned using internal cross-validation. When performing cross-validation, the set of patterns was created separately for each train-test split corresponding to iterations of cross-validation procedure. The number of trees for random forest and the number of iterations for

PROTEIN GROUP	ACC. ON EXCL. GROUP	CV ACC. ON TRAINING DATA
Helix-turn-helix	83.3	80.3
Zinc-coordinating	100	82.9
Zipper-type	88.9	83.1
Other α-helix	100	85.0
β-sheet	77.8	86.0
Other	100	82.5
Enzyme	58.1	90.4

Table 10: Predictive accuracies obtained by linear SVM classifiers trained on the datasets PD138/NB110 with protein groups excluded from PD138. The *accuracy on excluded group* is the percentage of correctly classified proteins from the protein group excluded from the training data. The *cross-validated accuracy on training data* is the accuracy of the learnt model estimated by 10-fold cross-validation on the training data.

CLASSIFIER	CATEGORY	REFERENCES
Linear support vector machine	kernel	[8]
SVM with RBF kernel	kernel	[8]
Simple logistic regression	regression/ensemble	[50]
L2-regularized logistic regression	regression	[30]
Ada-boost (with decision stamps)	ensemble	[24]
Random forest	ensemble	[6]

Table 11: State-of-the-art attribute-value learning algorithms used for classification.

Ada-boost was selected from the set $\{10, 20, 50, 100, 200, 500, 1000\}$. The complexity parameter c for linear support vector machine and for support vector machine with RBF kernel was selected from the set $\{1, 10, 10^2, 10^3, 10^4, 10^5, 10^6\}$. The regularization parameter of L2-regularized logistic regression was selected from the set $\{10^{-3}, 10^{-2}, 10^{-1}, 1, 10, 10^2, 10^3\}$. The minimum frequency of features on one of the classes was 0.7.

5.4 EXPERIMENTS WITH ANTIMICROBIAL PEPTIDES

In this section we present our relational machine learning techniques to predict antimicrobial activity of antimicrobial peptides (AMPs). To our best knowledge this is the first attempt to automatically discover common structural patterns present in antimicrobial peptides and to use them for prediction of antimicrobial activity. We utilized our relational learning method [42] which has already been used for DNA-binding propensity prediction of proteins [86] described in Section 5.3. There are two main differences between the work presented for the prediction of DNA-binding propensity of proteins and the prediction of antimicrobial acitivity of antimicrobial peptides. First, the problem that we tackled in Section 5.3 [86] dealt with classification, whereas here we built a regression model. Second, here only primary structures of peptides are available (therefore we have to rely on structure prediction), whereas we could use spatial structures obtained by X-ray crystallography in the case of the study with DNA-binding proteins.

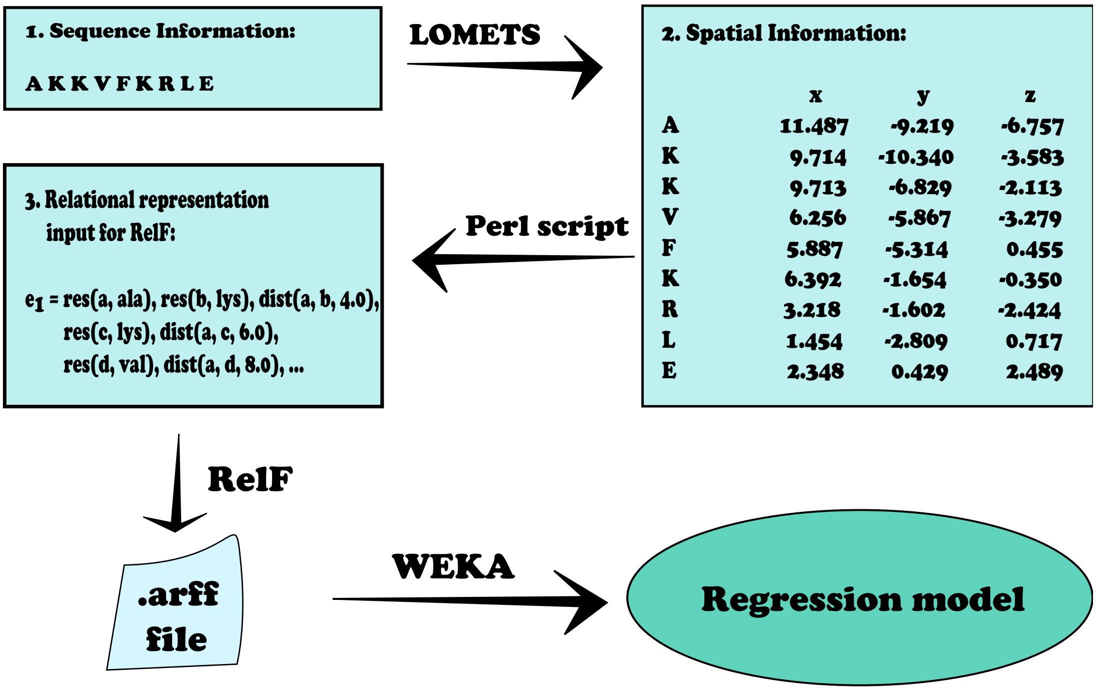
We modified the propositionalization-based method described in Section 5.2 for prediction of antimicrobial activity of AMPs. The modified method proceeds in four steps. It starts with AMP sequences, for which we obtain spatial models using LOMETS structure prediction software [105] (*step 1*). This gives us 3D information in PDB files. Then we create a relational representation of the peptides (*step 2*). After that we use our relational learning algorithm RelF to extract meaningful relational patterns from the relational structures describing peptides and convert them to an approximate attribute-value representation of the peptides (*step 3*). *Step 2* corresponds to *step 1* and *step 3* corresponds to *step 2* in Section 5.2. Finally, the output of RelF - *.arff* file readable by WEKA [103] is used for learning regression models (*step 4*) - in this case of antimicrobial activity prediction.

In the first step, 3D structures of peptides are computed using LOMETS software. LOMETS combines results of several threading-based structure prediction algorithms and returns several models with predicted coordinates of α-carbon atoms. We use only the best full-length model according to ordering given by LOMETS for each sequence.

In the second step, we create a representation of peptides' spatial structures suitable for relational learning as described in Section 5.1.

In the third step, we use RelF to construct a set of meaningful structural patterns. Since RelF had been designed for classification problems, we had to find a way to use it for regression problems. We decided to follow a straightforward approach. We enriched RelF with preprocessing in which the training data are split into two sets[1] according to antimicrobial activity - the first set containing peptides with lower-than-median activities, the second set containing peptides with higher-than-median activities. As soon as we have a dataset with at least two classes, RelF can be used for construction of discriminative patterns. The output of RelF is an attribute-value representation in WEKA format. We also added to these files additional information about dipole moment, proportions of amino acid types and their spatial asymmetries [92] which proved to be useful when added to relational patterns in Section 5.3 [86]. In the last step, we used implementation of SVM with RBF kernel present in the WEKA open-source machine learning software to train a regression model using the files generated in *step 3*. Parameters of the regression model are tuned using internal cross-validation. When performing cross-validation, the set of patterns is created separately for each train-test

[1] When performing cross-validation, we always split the data taking into account only the training set to avoid information leakage into the independent test set.


1. Sequence Information:
AKKVFKRLE
LOMETS
2. Spatial Information:
Perl script
3. Relational representation input for RelF:
e1 = res(a, ala), res(b, lys), dist(a, b, 4.0), res(c, lys), dist(a, c, 6.0), res(d, val), dist(a, d, 8.0), ...
RelF
.arff file
WEKA
Regression model

	x	y	z
A	11.487	-9.219	-6.757
K	9.714	-10.340	-3.583
K	9.713	-6.829	-2.113
V	6.256	-5.867	-3.279
F	5.887	-5.314	0.455
K	6.392	-1.654	-0.350
R	3.218	-1.602	-2.424
L	1.454	-2.809	0.717
E	2.348	0.429	2.489

Figure 10: The main steps of the method for prediction of antimicrobial activity of antimicrobial peptides.

split corresponding to iterations of the cross-validation procedure. The data transformation process is shown in Figure 10.

5.4.1 *Discussion of the Results for Antimicrobial Peptides*

In this section we present experiments performed on real-life data. We used the relational representation of peptides described in Section 5.1. The representation consisted of literals representing types of the amino acids and literals representing pair-wise distances between the amino acids up to 10 Å. These distances were computed from α-carbon coordinates obtained from PDB files computed by LOMETS. We used discretisation of distances with discretisation step 2 Å. We trained support vector machine [8] regression models with RBF kernel selecting optimal C (complexity constant) and $gamma$ (determines the kernel width parameter) for each fold by internal cross-validation. The estimated results are shown in Table 12.

We performed experiments on three datasets (CAMEL, RANDOM and BEE) which are described in Section 4.3. We compared the results of our relational learning method for regression with the results reported by Torrent et al. which is a state-of-the-art method. In the study by Torrent et al. [97], only cross-validated coefficients of determination were given. Coefficient of determination can be regarded as the proportion of variability in a dataset that is accounted for by the statistical model. In addition, we also report correlation coefficient (q) and root-mean-square error $(RMSE)$ for our regression method. On dataset CAMEL we achieved the same results as Torrent et al. On dataset RANDOM we improved upon the results of Torrent et al. in terms of coeffi-

	Torrent et al. [97]	**Our Regression Model**		
	q^2	q^2	q	RMSE
CAMEL	**0.65**	**0.65**	0.81	1.23
RANDOM	0.72	**0.74**	0.87	1.23
BEE	-	0.3	0.61	1.04

Table 12: Experimental results obtained by cross-validation, where q^2 is coefficient of determination, q is correlation coefficient and RMSE is root-mean-square error.

cients of determination. Since dataset BEE is a newly compiled dataset, there are no results to compare our approach with. It is also a harder dataset, than the other two, because it is composed of three different sources. Each of these sources is homogeneous on their own, but heterogeneous when joined into one big dataset. Also the variance of antimicrobial activity is lower in this dataset than in the other two. This explains why the coefficient of determination is so small as compared to the coefficients of determination obtained for the other datasets.

A problem of antimicrobial peptides as antibiotics is that they often have the ability to lyse eukaryotic cells, which is commonly expressed as hemolytic activity or toxicity to red blood cells. Unlike the other methods which use a pre-fixed set of physicochemical features, our method is not limited to one particular task. Since the sources from which we compiled the dataset BEE contained also information about the hemolytic activity, we decided to assess the potential of our method also for prediction of hemolytic activity. Because more than half of the reported hemolytic activities were given only by a lower-bound (200 μM) (i.e. it was not possible to measure the exact value), we decided to transform the problem to a two-class classification problem - the first class corresponding to peptides with activities below the lower-bound, the second class corresponding to peptides with activities higher than the lower-bound. We performed experiments following the same steps as in the prediction of antimicrobial activity, but with a random forest classifier instead of support vector machine classifier for regression. We obtained accuracy 60.83% and AUC (area under ROC curve) 0.725.

In addition, we can analyse the structural patterns used in the regression model which can give us insights about the process by which the antimicrobial peptides kill bacteria. We used the following methodology. First, we discretized the antimicrobial activity attribute, so that we could apply χ^2 criterion for ranking of patterns. Then, for each split of the datasets (CAMEL, RANDOM and BEE) induced by 10-fold cross-validation we selected the three most informative structural patterns according to the χ^2 criterion. We chose

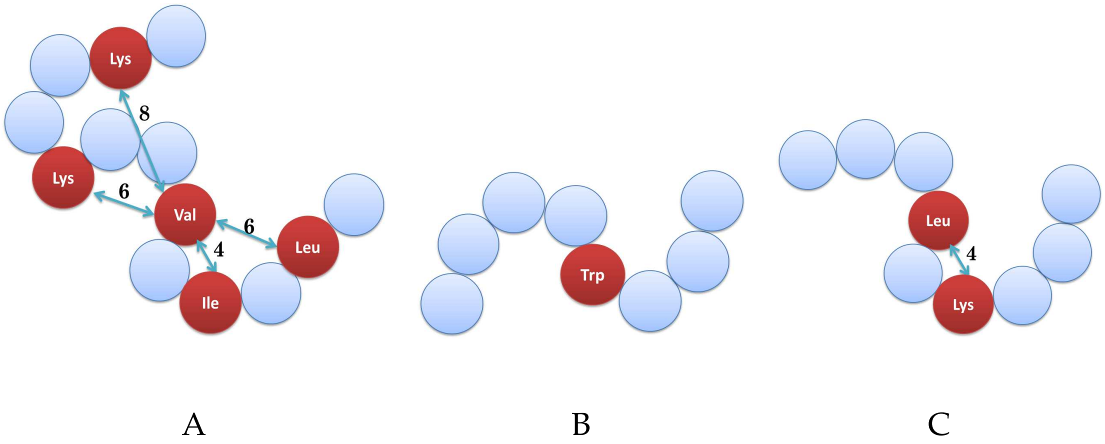


Figure 11: Most informative structural patterns according to the χ^2 criterion for the dataset of CAMEL (A), RANDOM (B) and BEE (C)(edges not to scale).

one pattern which was selected most often among the folds for each dataset for further discussion. These patterns are shown in Figure 11.

The selected pattern for the dataset CAMEL assumes presence of five amino acids: ILE, LEU, 2×LYS, VAL with distances between them as depicted in Figure 11. The positively charged lysines are known to correlate with antimicrobial activity and the presence of leucine can be explained by the fact that the dataset CAMEL contains mostly leucine-rich peptides. Interestingly, the remaining two amino acids - isoleucine and valine - and leucine are the only proteinogenic branched-chain amino acids - they each have a carbon chain that branches off from the amino acid's main chain, or backbone.

The selected pattern for the dataset RANDOM is very simple. It assumes presence of tryptophan. Since the patterns count the number of occurrences, it corresponds to proportion of TRP in peptides. This is not surprising, given that the peptides of the dataset RANDOM are composed mostly of TRP and some other amino acids.

Finally, the selected pattern for the dataset BEE assumes presence of two amino acids: LEU and LYS in the distance 4Å from each other. Again, the positively charged amino acid - lysine is known to correlate well with antimicrobial activity. Both leucine and lysine appeared also in the selected pattern for the dataset CAMEL.

In summary, we have shown that our relational learning approach for regression improves on a state-of-the-art approach to antimicrobial activity prediction in terms of predictive accuracy. Moreover, we have illustrated that our method is capable to also provide interpretable patterns describing spatial configurations of amino acids in peptide structures.

5.5 CONCLUSIONS

In this chapter, we designed a suitable relational representation of structural protein data. Then we used relational machine learning for construction of classifiers able to predict DNA-binding propensity of proteins. We found out that it is beneficial for accurate prediction to count occurrences of structural relational patterns in protein structures instead of relying just on the pattern's presence or absence. In the experiments we demonstrated that our relational learning approach is able to achieve higher predictive accuracies than methods based on features hand-crafted by experts. We also ensured that our approach is able to construct classifiers which can work accurately over various groups of proteins. We further utilized our relational learning approach for antimicrobial activity prediction of peptides. Unlike the DNA-binding propensity prediction, this is a regression problem. We showed that a modification of our relational learning approach tailored for regression improves on a state-of-the-art approach to antimicrobial activity prediction in terms of predictive accuracy. Furthermore, the patterns discovered by our relational learning methods can potentially uncover the underlying mechanism of the DNA-binding or the bacteria killing process.

5.6 ALGORITHMIC DETAILS

The original algorithm RelF was designed to construct existential patterns, but as we saw in this chapter, counting patterns are more appropriate for proteomics problems. Therefore, we extended RelF with support for counting patterns. The first problem that we had to solve was to implement an algorithm for computing the number of occurrences for a given pattern and a relational structure. Our implementation[2] essentially corresponds to the algorithm described by Pichler et al. [74]. A more difficult problem turned out to be the construction of counting patterns. The main difficulty was how to define redundancy of patterns and how to exploit it during the pattern construction process.

We extended the definition of redundancy used by RelF for counting patterns as follows: we defined two patterns to be mutually redundant if the number of occurrences of one pattern was a fixed multiple of the number of occurrences of the other pattern for all relational structures (examples) in the given dataset. With this definition of redundancy, we were able to prune redundant building blocks during the pattern construction process as follows. We extended the definition of *domain* used by RelF for pruning redundant building blocks by assigning a number to each element of the domain. This number corresponds to the number of occurrences of the building block in the given relational structure with the input variable of its root-literal substituted by the respective value from the domain. It can be shown that when extended domains of two building blocks are equal or *fixed multiples* of each other, then one of the blocks can be safely discarded while guaranteeing that a complete set of non-redundant patterns will be generated. While theoretically appealing, this approach unfortunately leads to construction of many patterns that are hard to interpret. Therefore, we used a more straightforward approach in

[2] We devised this algorithm independently, but it is just a small refinement of the well-known algorithm of Yannakakis [106].

the experiments[3]. This more straightforward version of RelF first constructs a set of patterns which is non-redundant w.r.t. the existential patterns and then computes the number of occurrences of these patterns.

It remains to explain the sampling strategy that we had to use on our large datasets. The strategy works as follows: it samples a subset of learning examples containing a pre-specified number of examples and then uses RelF to construct a non-redundant set of patterns. This process is repeated several times and then all the constructed patterns are evaluated on the complete dataset and filtered.

[3] The version of RelF which guarantees that all non-redundant counting patterns are always found is part of our suite of relational learning algorithms called TreeLiker (described in Appendix A).

6 RELATIONAL LEARNING WITH POLYNOMIALS

Modelling of relational domains which contain substantial part of information in the form of real-valued variables is an important problem with applications in bioinformatics. For proteins, knowing physicochemical properties of regions on their surfaces may be useful for prediction of their function.

So far there have not been many relational learning systems introduced in the literature that would be able to model multi-relational domains with numerical data efficiently. For example RelF used in Chapter 5 is unable to work efficiently in such domains. A framework able to work efficiently in multi-relational domains with numerical data is hybrid Markov logic [102]. However, there is currently no known approach for learning structure of hybrid Markov logic which is mainly due to their excessive complexity. Another type of systems that do not directly model the probabilities but are able to learn in domains rich in numerical data are systems based on relational aggregation [38, 101]. In this chapter we describe a relatively simple novel transformation-based framework for learning in rich relational domains containing numerical data.

The new framework exploits multi-variate polynomial aggregation functions, which is something that, surprisingly, has not been studied in the relational-learning literature yet. This chapter is organized as follows. We first define so-called *polynomial features*, then we study their properties in Section 6.1. A novel relational representation of proteomics data is presented in Section 6.2. Then, we show how to apply the developed approach in proteomics domain. We present our experimental results in the domain of DNA-binding proteins and other related domains in Section 6.3. In Section 6.4 we conclude this chapter. Finally, in Section 6.5 we present the algorithmic details of this chapter. There, we derive efficient algorithms for *tree-like polynomial features* and *polynomial features* with bounded tree-width in general. Unlike methods based on straightforward application of existing algorithms for conjunctive-query answering, our new algorithms run in time which depends only linearly on the maximum number of variables in monomials.

6.1 POLYNOMIAL RELATIONAL FEATURES

In this section we describe a simple framework for working with hybrid relational learning problems. We start by defining notation. Let $n \in \mathbf{N}$. If $\vec{v} \in \mathbf{R}^n$ then v_i $(1 \leqslant i \leqslant n)$ denotes the i-th component of $\vec{v}$. If $I \subseteq [1;n]$ then $\vec{v}_I = (v_{i_1}, v_{i_2}, \dots v_{i_{|I|}})$ where $i_j \in I$ $(1 \leqslant j \leqslant |I|)$. To describe training examples as well as learned models, we use a conventional first-order logic language $\mathcal{L}$ whose alphabet contains a distinguished set of constants $\{r_1, r_2, \dots r_n\}$ and variables $\{R_1, R_2, \dots R_m\}$ $(n, m \in \mathbf{N})$. An *r-substitution* ϑ is any substitution as long as it maps variables R_i only to terms r_j. For the largest k such that $\{R_1/r_{i_1}, R_2/r_{i_2}, \ldots, R_k/r_{i_k}\} \subseteq \vartheta$ we denote $I(\vartheta) = (i_1, i_2, \ldots i_k)$. A (Herbrand) *interpretation* is a set of ground atoms of $\mathcal{L}$. $I(H)$ ($I(\varphi)$) denotes the naturally ordered set of indexes of all constants r_i found in an interpretation H ($\mathcal{L}$-formula φ).

Our training examples have both *structure* and *real parameters*. An example may e.g. describe a protein structure; here the *structure* (part of the example) could describe the spatial structure of the protein (similarly as in Chapter 5) and the *parameters* would describe the numerical physicochemical properties of amino acids or groups of amino acids in the protein. The *structure* will be described by an interpretation, in which the constants r_i represent uninstantiated real parameters. The parameter values will be determined by a real vector. Formally, an example is a pair $\left(H, \vec{\theta}\right)$ where H is an interpretation, $\vec{\theta} \in \Omega_H$, and $\Omega_H \subseteq \mathbf{R}^{|I(H)|}$ (here, $\mathbf{R}$ denotes the set of real numbers).

A *feature* is simply a $\mathcal{L}$-formula. For each example $\left(H, \vec{\theta}\right)$, it *extracts* some components of $\vec{\theta}$ into a set of vectors. Given an example $e = (H, \vec{\theta})$ and a feature φ, the *sample set* of φ and e is the multi-set $\mathcal{S}(\varphi, e) = \{\vec{\theta}_{I(\vartheta)} | H \models \varphi\vartheta\}$ where ϑ are r-substitutions grounding all free variables[1] in φ, and $H \models \varphi\vartheta$ denotes that $\varphi\vartheta$ is true under H.

Example 4. *Let* $\varphi = a(X, R_1), e(X, Y), a(Y, R_2)$ *be a feature and* $e = a(a, 1), e(a, b), a(b, 2), e(b, c), a(c, 3), e(a, c)$ *be an example (formally:* $e = (a(a, r_1), e(a, b), a(b, r_2), e(b, c), a(c, r_3), e(a, c), (1, 2, 3)^T)$*). Then the sample-set of feature* φ *w.r.t. example* e *is* $\mathcal{S}(\varphi, e) = \{(1, 2)^T, (2, 3)^T, (1, 3)^T\}$.

Now, we can introduce *polynomial relational features* and discuss some of their properties. We start with *monomial relational features*. A *monomial relational feature* M is a pair $(\varphi, (d_1, \ldots, d_k))$ where φ is a feature with k distinguished variables and $d_1, \ldots, d_k \in \mathbf{N}$. *Degree* of M is $deg(M) = \sum_{i=1}^{k} d_i$. Given a non-empty sample set $\mathcal{S}(\varphi, e)$, we define the *value* of a monomial feature $M = (\varphi, (d_1, \ldots, d_k))$ w.r.t. example e as

$$M(e) = \frac{1}{|\mathcal{S}(\varphi, e)|} \sum_{\vec{\theta} \in \mathcal{S}(\varphi, e)} \vec{\theta}_1^{d_1} \cdot \vec{\theta}_2^{d_2} \cdot \cdots \cdot \vec{\theta}_k^{d_k}$$

where $\vec{\theta}_i$ is the i-th component of vector $\vec{\theta}$.

Sometimes, we will use a more convenient notation for monomial features motivated by this definition of *value*:

$$(\varphi, (d_1, \ldots, d_k)) \equiv^{def} R_1^{d_1} \cdot R_2^{d_2} \cdot \cdots \cdot R_k^{d_k}.$$

A *polynomial relational feature* is an expression of the form $P = \alpha_1 M_1 + \alpha_2 M_2 + \cdots + \alpha_k M_k$ where $M_1, \ldots, M_k$ are monomial features and $\alpha_1, \ldots, \alpha_k \in R$

[1] Note that an interpretation H does not assign domain elements to variables in $\mathcal{L}$. The truth value of a *closed* formula (i.e., one where all variables are quantified) under H does not depend on variable assignment. For a general formula though, it does depend on the assignment to its free (unquantified) variables.

(formally expressed as a pair of two ordered sets - one of monomials and one of the respective coefficients). *Value* of a polynomial feature $P = \alpha_1 M_1 + \cdots + \alpha_k M_k$ w.r.t to an example e is defined as $P(e) = \alpha_1 M_1(e) + \alpha_2 M_2(e) + \cdots + \alpha_k M_k(e)$. *Degree* of a polynomial relational feature P is maximum among the degrees of its monomials.

Polynomial relational features can be used for learning as follows. First, a set $\mathcal{F}$ of monomial relational features is constructed. Then, all monomial features are evaluated w.r.t. all examples in the dataset and the values are stored in a table. This gives us an attribute-value table which can be processed by any attribute-value learning algorithm such as SVM or random forest. This is an instance of *propositionalization* [55]. This approach is a generalization of existing aggregation-based systems that have been introduced in relational learning [38, 101] which surprisingly did not incorporate any multi-variate aggregate functions.

Polynomial relational features are closely related to hybrid Markov logic [102]. Hybrid Markov logic (HML) is an extension of Markov logic [77] to hybrid domains, i.e. to domains which contain both discrete relational data and numerical data. A hybrid Markov logic network is a set of pairs (F_i, w_i) where F_i is a first-order formula or a numeric term and $w_i \in R$ is a weight of the formula F_i. When one fixes a set of constants C then hybrid Markov logic defines probability distribution over possible worlds as follows:

$$p(X = x) = \frac{1}{Z} \exp\left(\sum_i w_i s_i(x)\right)$$

where $s_i(x)$ is either number of true groundings of feature F_i w.r.t x if F_i is not a numeric-feature, or the sum of *values* w.r.t. x when F_i is a numeric feature. If one has conditional distributions represented as HMLs for two classes and the task is to learn a classifier distinguishing examples from the two classes then the resulting decision boundary has equation (where $+$ and $-$ superscripts distinguish the weights and features of the models for the two classes)

$$\sum_i w_i^+ s_i^+(x) - \sum_i w_i^- s_i^-(x) = t$$

which defines a linear hyperplane. If one used polynomial numeric features in the HML, the resulting classifier would become very similar to what we obtain with polynomial relational features and a linear classifier, e.g. support vector machine, the main difference being that $s_i(x)$ would be an *average* of values of polynomials applied on the ground instances of feature F_i whereas it is a *sum* in the case of HML. A convenient property of averages used in the polynomial-feature framework is that they are not so sensitive to the size of the examples (possible worlds).

6.2 REPRESENTATION OF DATA FOR LEARNING WITH POLYNOMIALS

There are several representations of proteins which can harness the advantages of the framework of polynomial relational features. One of the simplest representations which is already powerful enough to achieve high predictive accuracies is based just on the combination of primary and secondary structure of proteins. In this representation, proteins are represented as sequences of consecutive segments. These segments are labelled by their secondary structure and by their numerical physicochemical properties. This is exemplified in Example 5.

Specifically, the sequence of segments is represented by literals of the type $n(x, y)$ which denote two segments x and y to be neighbours. Then, there are literals of the type $prop(x, name, value)$ which denote a numeric property $name$ to be equal to $value$ in segment x. The secondary structure is described by literals of the type $sec\_structure(x, type)$ which denote the secondary structure $type$ for the segment x. It can happen that there are multiple literals of the type $sec\_structure(x, type)$ for a single segment – if a segment contains amino acids corresponding to different secondary structures.

Physicochemical properties used in this representation can be for example: average charge of amino acids in the segment, average hydropathy index, dipole moment, polarity and so on.

Example 5. *Let*

$$\begin{aligned} e \;=\; & n(s_1, s_2), n(s_2, s_3), n(s_3, s_4), prop(s_1, charge, 0.5), prop(s_2, charge, 0.1), \\ & prop(s_3, charge, 0.4), prop(s_4, charge, 0.5), sec\_structure(s_1, helix), \\ & sec\_structure(s_2, helix), sec\_structure(s_3, sheet), \\ & prop(s_1, hydropathy, 0.2), prop(s_2, hydropathy, -3.5), \\ & prop(s_3, hydropathy, 2.8), prop(s_4, hydropathy, -0.4) \end{aligned}$$

be an example of a protein – it is divided into four segments s_1, s_2, s_3 and s_4.

Next, let $\varphi_1 = prop(X, charge, R_1), sec\_structure(X, sheet)$ be a feature and let $M_1^{\varphi_1} = R_1$ and $M_2^{\varphi_1} = R_1^2$ be monomial features based on the feature φ_1. Intuitively, the monomial feature $M_1^{\varphi_1} = R_1$ corresponds to the average charge in sheets of the protein. The monomial feature $M_2^{\varphi_1} = R_1^2$ captures the dispersion *of charge over the sheets of the protein. Indeed, let us have two proteins A and B and a monomial feature $M = (R_1)^2$ based on the relational feature $\varphi = prop(X, charge, R_1)$ and let us assume that A and B are composed of the same number of amino acids and that they contain the same number of positively charged amino acids and no negatively charged amino acids in the sheets. Finally, let us also assume that the positively charged amino acids are distributed more or less uniformly over the protein structure A but are mostly concentrated in one of the sheets of the protein structure B. Then it is not hard to see that for the values $M(A)$ and $M(B)$ it should hold $M(A) \leqslant M(B)$. However, variance as usually defined, is better captured by the following expression composed of monomial features $M_1^{\varphi_1}$ and $M_2^{\varphi_1}$: $M_2^{\varphi_1}(e) - (M_1^{\varphi_1}(e))^2$.*

Likewise, let us have a feature

$$\varphi_2 = prop(X, charge, R_1), n(X, Y), prop(Y, charge, R_2)$$

and monomial features $M_1 = R_1 \cdot R_2$, $M_2 = R_1$, $M_3 = R_2$ based on it. Then the monomial feature M_1 corresponds to agreement *of charges of neighbouring segments* *over a given protein structure, their* covariance *is better captured by the following expression involving all three monomial features:*

$$M_1(e) - M_2(e) \cdot M_3(e).$$

Note that this expression is not a polynomial feature but only an expression composed of values *of polynomial (monomial) features.*

6.3 EXPERIMENTS

We evaluated performance of the polynomial-feature-based method in three relational learning domains. We compared it with Tilde [5] and with the method of Szilágyi et al. [92]. We performed experiments with tree-like polynomial relational features with degree one, two and three in order to evaluate impact of degree of monomials on predictive accuracy. For each dataset, we used two types of relational descriptions with different complexity. We used random forest classifiers with 100, 500 and 1000 trees (see Table 13).

6.3.1 *DNA-binding proteins*

Our first set of experiments dealt with prediction of DNA-binding propensity of proteins using the representation of proteins described in Section 6.2. Recall, that it had already been shown – for example in the work of Szilágyi and Skolnick – that electrostatic properties of proteins are good features for predictive classification. Therefore, our first model for predicting whether a protein binds to DNA used only distributions of charged amino acids in fixed-size segments and the secondary structure of the proteins. Surprisingly, for the experiments using only electric charge, the highest accuracies were obtained by monomials of degree 1. Nevertheless, results obtained for all degrees of monomials were higher than 75.8% accuracy obtained by Tilde. The results for degree 1 were better than the results obtained by Szilágyi et al.

In our second model, we added also information about average *propensity* of amino acids in the fixed-size segments to bind to DNA which had been measured by Sathyapriya et al. [81]. The accuracies obtained by our method on the second model were consistently higher than 81.4% accuracy obtained by Szilágyi and Skolnick with logistic regression or 82.2% that we obtained using random forest on Szilágyi's and Skolnick's features. The best results were obtained for monomials of degree 3. It seems to be the case that the ability to capture the statistical dependence of values of these numerical properties is important for prediction of DNA-binding propensity of proteins.

The obtained results are encouraging given that we used only a limited amount of information about the proteins. We further study the possibility to use models capturing distribution of continuous numerical properties of protein regions in Chapter 15, where the most accurate method of this thesis is presented – which also utilises a type of polynomial features.

	Degree 1	**Degree 2**	**Degree 3**
	100/500/1000	100/500/1000	100/500/1000
PD138 - Ch	**84.7/82.3/82.3**	81.0/79.5/81.0	81.0/79.8/79.8
PD138 - Ch + P	82.7/84.7/84.3	83.9/84.7/84.3	**85.1/85.5/85.5**
Muta - Ch	88.8/88.3/88.3	88.8/88.8/88.3	**89.9/89.9/89.4**
Muta - A + Ch	87.8/88.3/88.3	88.3/88.8/88.8	**89.9/89.9/89.9**
NCI 786 - Ch	61.0/61.0/61.0	66.5/66.5/66.8	**67.2/68.0/68.0**
NCI 786 - A + Ch	70.3/70.1/69.9	70.5 /**70.8/70.8**	**70.6**/70.2/70.4

Table 13: Accuracies estimated by 10-fold cross-validation for learning using monomial features and random forests for degrees of monomial features: 1, 2 and 3.

6.3.2 *Other Related Problems*

Since the method based on polynomial relational features is not bound just to applications involving DNA-binding propensity prediction of proteins, we evaluated it also on two other datasets.

Our second set of experiments was done on the well-known Mutagenesis dataset [84], which consists of 188 organic molecules marked according to their mutagenicity. The problem of predicting mutagenicity relates closely to prediction of DNA-binding propensity of proteins, because mutagenicity regards the ability of molecules to damage DNA (this damage can be caused indirectly, for example through free radicals). We performed two experiments in this domain. In the first experiment, we used only information about bonds and their types (*single*, *double*, *triple*, *resonant*) and information about charge of atoms, but not about their types. In the second experiment, we also added information about atom types. The accuracies obtained by our method (Table 13) are consistently higher than the best accuracy 86% achieved by Tilde in [5]. The best results are obtained for monomial features of degree 3.

Our third set of experiments was performed on the NCI 786 dataset which contains 3506 molecules labelled according to their ability to inhibit growth of renal tumors. Again we performed two experiments in this domain. In the first experiment, we used only information about bonds and their types and information about charge of atoms and in the second experiment we also added information about atom types. Monomials of degree 3 turned out to be best for the first representation whereas monomials of degree 2 performed best for the second representation. Tilde did not perform well on this dataset, so at least, we compared our results with results reported in [42] for kFOIL (63.1), nFOIL (63.7) and RelF (69.6). The accuracies obtained with monomial features for the *atoms + charge* representation were consistently higher than these results.

6.4 CONCLUSIONS

In this chapter, we presented a conceptually simple framework for relational learning with multivariate polynomial functions suitable for complex domains (such as protein function prediction) involving real-valued parameters. We showed how polynomial relational features can be used for estimation of higher-order moments of distributions in the relational context. We also demonstrated how they can be used for DNA-binding propensity prediction where we were able to obtain state-of-the-art accuracies using only limited amounts of numerical information.

6.5 ALGORITHMIC DETAILS

In this section we describe an efficient algorithm (see Algorithm 1) for computing values of monomial tree-like features. The algorithm runs in time polynomial in the combined size of a feature and an example. Importantly, it scales polynomially also in the number of the distinguished numeric variables.

While we used the adjective *tree-like* informally so far, now we need to define it precisely. A first-order conjunction without quantifications C is *tree-like* if the iteration of the following rules on C produces the empty conjunction: (i) *Remove an atom which contains fewer than 2 variables.* (ii) *Remove a variable which is contained in at most one atom.* Intuitively, a tree-like conjunction can be imagined as a tree with the exception that whereas trees are graphs, conjunctions correspond in general to hypergraphs. Features based on tree-like formulas are called tree-like.

We will need some auxiliary definitions. Let φ be a tree-like feature. Let us suppose that $s_1, s_2, \ldots, s_k$ is a sequence of steps of the reduction procedure which produces an empty feature from φ. Let $\lhd$ be an order on the atoms of φ such that if an atom a_1 disappeared before an atom a_2 during the reduction process then $a_1 \lhd a_2$. Then we say that $\lhd$ is a topological ordering of φ's atoms. Let $A \subseteq \varphi$ be a maximal set of atoms having a variable v in common. We say that $a \in A$ is a parent of atoms from $A \setminus \{a\}$ if a has disappeared after all the other atoms from $A \setminus \{a\}$. An atom a is called root if it has no parents w.r.t. $\lhd$.

We use $(C, v) \in \mathtt{Children}(\varphi, \lhd)$ for the set of all features φ_C with roots equal to children of φ (w.r.t. $\lhd$) together with the respective shared variables v. Similarly, we use $C \in \mathtt{Children}(\varphi, v, \lhd)$ for the set of all features φ_C with roots equal to children of φ (w.r.t. $\lhd$). We define the *input* variable of a feature φ contained in some bigger feature ψ (denoted by $\mathtt{inp}(\varphi, \psi, \lhd)$) as the variable which is shared by $\mathtt{root}(\varphi, \lhd)$ with its parent in ψ. When a is a ground atom such that $\mathrm{root}(\varphi, \lhd)\theta = a$ then we define *input* operator $\mathrm{inp}(a, \varphi, \psi, \lhd)$ which will give us $\mathrm{inp}(\varphi, \psi, \lhd)\theta$. If $\varphi = \psi$ then

$$\mathrm{inp}(a, \varphi, \psi, \lhd) = \mathrm{inp}(\varphi, \psi, \lhd) = \emptyset.$$

Here, the empty set is used as a *dummy* input. It does not mean that the returned value of the *input* functions would be a set in general.

The evaluation algorithm (Algorithm 1) uses two auxiliary algorithms for computing so-called *domains* and *term-domains* of features. Let φ be a tree-like feature, $\lhd$ be a topological ordering of φ's atoms. Then we say that a set of atoms $A \subseteq e$ is domain of φ w.r.t. an example e (denoted by $A = \mathcal{D}(\varphi, e, \lhd)$) if A contains all ground atoms $a = \mathrm{root}(\varphi, \lhd)\theta$ such that $e \models \varphi\theta$.

Let $e = a(a, b), a(a, c), b(b)$ be an example and let $\varphi = a(X, Y), b(Y)$ and $\psi = a(X, Y), a(X, Z), b(Y)$ be features. Let $b(Y) \lhd a(X, Y)$. Then $\mathcal{D}(\varphi, e, \lhd) = \{a(a, b)\}$.

The algorithm for computing domains is derived from the well-known algorithm for answering acyclic conjunctive queries [106]. It runs in time polynomial in $|\varphi|$ and $|e|$. It computes not only domains corresponding to the roots of the given features but also domains corresponding to all the sub-features during one pass over a given feature. In the pseudocode of the parameter-estimation algorithms we will *call* the procedure for computing domains as $\mathcal{D}(\varphi, e, \lhd, T)$ where φ is the feature and e is the example for which we want to compute the domain, $\lhd$ is a topological ordering of φ's atoms and T is a table in which domains of all φ's sub-features should be stored.

By *sample parameters* we mean a 3-tuple (x, v, n). Here x can be either an empty set, an atom or a term, v is a real number and n is a natural number. Next, we define a concatenation operation for combining sample parameters. Let $A = (x, v, n_A)$ and $B = (y, w, n_B)$ be sample parameters. Then we define $A \otimes B$ as

$$A \otimes B = (x,\ v \cdot w, n_A \cdot n_B)\,.$$

Let $\varphi \subseteq \psi$ be features and $\lhd$ a topological ordering of ψ's atoms. Let $A = (x, v, n_A)$ and $B = (y, w, n_B)$ be sample parameters where x and y are logic atoms such that $\mathrm{inp}(x, \varphi, \psi, \lhd) = \mathrm{inp}(y, \varphi, \psi, \lhd)$. Then we define $A \oplus_{\lhd}^{\varphi,\psi} B$ as

$$A \oplus_{\lhd}^{\varphi,\psi} B = (\mathrm{inp}(x, \varphi, \psi, \lhd), \frac{1}{n_A + n_B}(n_A \cdot v + n_B \cdot w), n_A + n_B).$$

Let $\varphi \subseteq \psi$ be features and $\lhd$ a topological ordering of φ's atoms. Let $\mathcal{X} = \{(x_1, v_1, n_1), \dots, (x_k, v_k, n_k)\}$. Next, let $\mathcal{X}[t]$ denote the set of all sample parameters $(x, \dots) \in \mathcal{X}$ for which $\mathrm{inp}(x, \varphi, \psi, \lhd) = t$. Then $\bigoplus_{\lhd}^{\varphi,\psi} \mathcal{X}$ is defined as follows:

$$\bigoplus_{\lhd}^{\varphi,\psi} \mathcal{X} = \{(y_1, w_1, n_{w_1}), \dots, (y_m, w_m, n_{y_m})\}$$

where $(y_i, w_i, n_{y_i}) = x_1 \oplus_{\lhd}^{\varphi,\psi} x_2 \oplus_{\lhd}^{\varphi,\psi} \cdots \oplus_{\lhd}^{\varphi,\psi} x_p \oplus_{\lhd}^{\varphi,\psi} (\emptyset, 0, 0)$ for $\{x_1, \dots, x_p\} = \mathcal{X}[y_i]$.

The correctness of the algorithm follows from the next reasoning. Let us have a tree-like feature φ with k distinguished variables, a monomial feature

Algorithm 1 An algorithm for computing value $M(e)$ of a monomial feature $M = (\varphi, (d_1, \ldots, d_k))$ for connected tree-like φ

```
Procedure: $Eval(M = (\varphi, (d_1, \ldots, d_k)), e = (H, \vec{\theta}), \lhd)$
$T \leftarrow []$
$\mathcal{D}(\varphi, e, \lhd, T)$ /* This fills values into the table $T$ */
return $\bigoplus_{\lhd}^{\varphi,\varphi} Eval2(\varphi, \varphi, e, \lhd, T)$
Procedure: $Eval2(M = (\varphi, (d_1, \ldots, d_k)), \psi, e = (H, \vec{\theta}), \lhd, T)$
$SP \leftarrow []$ /* An associative array of sample parameters */
$D_\varphi \leftarrow T[\varphi]$ /* $D_\varphi$ is domain of $\varphi$ */
for $\forall a \in D_\varphi$ do
    $r_{i_1}, r_{i_2}, \ldots, r_{i_l} \leftarrow$ distinguished constants contained in $root(\varphi)$ (indexed
    by indexes of the corresponding distinguished variables from $root(\varphi)$)

    $SP[a] \leftarrow \{a, r_{i_1}^{d_{i_1}} \cdot r_{i_2}^{d_{i_2}} \ldots r_{i_1}^{d_{i_l}}, 1\}$
end for
for $(\varphi_C, \nu) \in Children(\varphi, \lhd)$ do
    $SP_{\varphi_C} \leftarrow \bigoplus_{\lhd}^{\varphi_C,\varphi} Eval2(\varphi_C, \varphi, e, \lhd)$
    for $\forall a \in D_\varphi$ do
        $SP[a] \leftarrow SP[a] \bigotimes SP_{\varphi_C}[\nu\vartheta]$ where $root(\varphi, \lhd)\vartheta = a$
    end for
end for
return $SP$
```

$M = (\varphi, (d_1, \ldots, d_k))$ and an example e. For simplicity, we assume that any literal $l \in \varphi$ can contain at most one distinguished variable[2]. Our task is to compute the value $M(e)$. We start by picking a literal $l \in \varphi$ containing distinguished variable R_1 and ground all its variables using a substitution $\vartheta : vars(l) \to constants(c)$ so that $e \models \varphi\vartheta$. We then create a new auxiliary monomial $M_\vartheta = (\varphi\vartheta, (d_1, \ldots, d_k))$. The problem of computing $M_\vartheta(e)$ can be decomposed as:

$$M_\vartheta(e) = (R_1\vartheta)^{d_1} \cdot \prod_i M_i(e) \tag{1}$$

where $M_1, \ldots, M_m$ are connected sub-features of $\varphi\vartheta$ which arise when we remove literal $l\vartheta$ from $\varphi\vartheta$. This follows from the next proposition.

Proposition 1. *Let $\varphi = (\psi) \wedge (\gamma)$ be a disconnected feature such that ψ and γ do not share any variable. Let $M = (\varphi, (d_1, \ldots, d_k))$ be a monomial. Then it is possible to find monomial features M_ψ and M_γ such that $M(e) = M_\psi(e) \cdot M_\gamma(e)$ for all examples e.*

Proof. It holds $\mathcal{S}(\varphi, e) = \mathcal{S}(\psi, e) \times \mathcal{S}(\gamma, e)$ where $\times$ denotes Cartesian product. We can therefore construct the monomial features as follows. First, we split

[2] This is without loss of generality because any representation with demanding more than one distinguished variable per literal can be rewritten into a representation where there is always only one distinguished variable per literal.

the set of distinguished variable of F to two (necessarily disjoint) sets $\mathcal{R}_\psi$, $\mathcal{R}_\gamma$ according to the formula in which they appear. We split also the respective exponents to ordered sets $(v_1, \dots, v_{k_\psi})$ and $(w_1, \dots, w_{k_\gamma})$.

$$M_\psi = (\psi, (d_1^\psi, \dots, d_{k_\psi}^\psi))$$

$$M_\gamma = (\gamma, (d_1^\gamma, \dots, d_{k_\gamma}^\gamma)).$$

The product of these monomial features gives rise to the original feature because

$$\begin{aligned}
&M_\psi(e) \cdot M_\gamma(e) = \left(\tfrac{1}{|\mathcal{S}(\psi,e)|} \textstyle\sum_{\vec{\theta} \in \mathcal{S}(\psi,e)} \vec{\theta}_1^{v_1} \dots \vec{\theta}_k^{v_{k_\psi}}\right) \cdot \\
&\cdot \left(\tfrac{1}{|\mathcal{S}(\gamma,e)|} \textstyle\sum_{\vec{\theta} \in \mathcal{S}(\gamma,e)} \vec{\theta}_1^{w_1} \dots \vec{\theta}_{k_\gamma}^{w_{k_\gamma}}\right) = \tfrac{1}{|\mathcal{S}(\psi,e)| \cdot |\mathcal{S}(\gamma,e)|} \cdot \\
&\cdot \left(\textstyle\sum_{\vec{\theta} \in \mathcal{S}(\psi,e)} \vec{\theta}_1^{v_1} \dots \vec{\theta}_{k_\psi}^{v_{k_\psi}}\right) \cdot \left(\textstyle\sum_{\vec{\theta} \in \mathcal{S}(\gamma,e)} \vec{\theta}_1^{w_1} \dots \vec{\theta}_{k_\gamma}^{w_{k_\gamma}}\right) \\
&= \tfrac{1}{|\mathcal{S}(\psi,e) \times \mathcal{S}(\gamma,e)|} \textstyle\sum_{\vec{\theta} \in \mathcal{S}(\gamma,e) \times \mathcal{S}(\gamma,e)} \vec{\theta}_1^{d_1} \dots \vec{\theta}_k^{d_k} = M(e)
\end{aligned}$$

□

The value $M(e)$ can be then computed as

$$M(e) = \frac{1}{\sum_{\vartheta \in \Theta} \alpha_\vartheta} \sum_{\vartheta \in \Theta} \alpha_\vartheta M_\vartheta(e)$$

where $\Theta = \{\vartheta : vars(l) \to constants(c) | e \models \varphi\vartheta\}$ is the set of all true groundings of literal l and α_ϑ are the numbers of true groundings of $\varphi\vartheta$.

The time-complexity of the algorithm is $\mathcal{O}(|\varphi| \cdot (|e| \cdot k + \log|\varphi|))$ where k is the number of distinguished variables. First, the complexity of computing domains of all literals on line 2 of procedure $Eval$ is $\mathcal{O}(|\varphi| \cdot |e|)$ [78]. Second, the procedure $Eval2$ can be imagined as proceeding from leaves to root of the feature. At each step, at most $|e|$ literals must be processed in the loop on line 3 and the complexity of each iteration is $\mathcal{O}(k)$ where k is the number of distinguished variables. Third, the $\bigoplus_{\lhd}^{\varphi_C, \varphi}$ can be performed in time $\mathcal{O}(|\varphi| \log|\varphi|)$. Fourth, although the for-loop on line 7 has a nested loop, the nested loop is executed at most once for each literal $l \in \varphi$ and the inner loop is executed $\mathcal{O}(|e|)$-times. So the overall complexity of the algorithm is indeed $\mathcal{O}(|\varphi| \cdot (|e| \cdot k + \log|\varphi|))$.

7 RELATIONAL LEARNING WITH BOUNDED LGG

The relational learning approaches described in previous chapters of this part of the thesis focused on searching for relatively small structural patterns. The numbers of occurrences of these patterns were used as attributes for predictive classification. The advantage of small patterns is that when found as discriminative, it is probably not due to overfitting. If a small pattern is frequently found in a sufficiently large dataset, then it will likely be present also in unseen structures. On the other hand, overfitting may still be an issue when a big number of small patterns is combined in a single classifier. Classifiers based on a smaller number of more complex patterns may be equally good or even better. What the relational learning methods presented in previous chapters are not able to achieve is to capture more complex common substructures in proteins or peptides. In this chapter, we introduce a relational learning approach which is able to search for large, complex relational patterns.

A natural way to search for complex relational patterns could be based on Plotkin's least general generalization (LGG) [75]. A least general generalization of a set of clauses is a maximally specific clause which covers all the examples from this given set. Unfortunately, if we do not attempt to reduce the clauses that we obtain as LGGs, their size may grow very quickly, which makes the use of this method impractical. Even if we try to replace LGGs by smaller equivalent clauses, it may still be the case that their size will be too big. Moreover, finding the smaller equivalent clauses is an NP-hard problem. The novel method presented in this chapter redefines the notion of equivalence and parametrizes it by sets of clauses. Two clauses are considered to be equivalent if they are *indistinguishable* by the clauses from the given parametrization set. Finding a smaller equivalent clause w.r.t. this new notion of equivalence may be done in polynomial time for many practically relevant parametrization sets of clauses. In addition, the smallest clause equivalent to a given clause w.r.t. the new notion of equivalence may be actually smaller than the smallest clause equivalent to this clause in the classical sense. The presented new LGG operator based on the new parametrized notion of equivalence is named *bounded LGG*. Our relational learning approach with bounded LGG always guarantees to find a clause which is at least as *good* as the best clause from the parametrization set.

This chapter is organized as follows. In Section 7.1 we introduce a generalization of θ-subsumption, a generalization of θ-reduction and using these generalized notions, we introduce so-called bounded least general generalization, which is a generalized version of conventional least general generalization. Then, we present a generic bottom-up learning algorithm. We subject the novel method to experimental evaluation in Section 7.2. We conclude this chapter in Section 7.3. We review necessary material regarding

θ-subsumption, θ-reduction, least general generalization, constraint satisfaction, tree-decompositions and treewidth in Section 7.4. Also all proofs are located in this section.

7.1 BOUNDED LEAST GENERAL GENERALIZATION

In this section, we present a novel bottom-up learning algorithm based on a generalized notion of conventional LGG. We start by introducing *bounded* versions of θ-subsumption. If A and B are clauses, then we say that the clause A θ-subsumes the clause B (denoted by $A \preceq_\theta B$), if and only if there is a substitution θ such that $A\theta \subseteq B$. If $A \preceq_\theta B$ and $B \preceq_\theta A$, we call A and B θ-equivalent (written $A \approx_\theta B$). We define x-subsumption and x-equivalence which are weaker versions of θ-subsumption and θ-equivalence.

Definition 3 (x-subsumption, x-equivalence)**.** *Let X be a possibly infinite set of clauses. Let A, B be clauses not necessarily from X. We say that A x-subsumes B w.r.t. X (denoted by $A \preceq_X B$) if and only if $(C \preceq_\theta A) \Rightarrow (C \preceq_\theta B)$ for every clause $C \in X$. If $A \preceq_X B$ and $B \preceq_X A$ then A and B are called x-equivalent w.r.t. X (denoted by $A \approx_X B$). For a given set X, the relation $\preceq_X$ is called x-subsumption w.r.t. X and the relation $\approx_X$ is called x-equivalence w.r.t. X.*

Definition 3 provides no efficient way to decide x-subsumption between two clauses as it demands θ-subsumption of an infinite number of clauses to be tested in some cases. However, for many practically relevant sets of clauses X, there is a relation called *x-presubsumption* which implies x-subsumption.

Definition 4 (x-presubsumption)**.** *Let X be a set of clauses. If $\preceq_X$ is the x-subsumption w.r.t. X and $\lhd_X$ is a relation such that for any clauses A and B:*

1. *$(A \preceq_\theta B) \Rightarrow (A \lhd_X B)$*
2. *$(A \lhd_X B) \Rightarrow (A \preceq_X B)$*

then we say that $\lhd_X$ is an x-presubsumption w.r.t. the set X.

The relation x-presubsumption can be used to check whether two clauses are x-equivalent w.r.t. a given set of clauses X. It can be therefore used to search for clauses which are smaller than the original clause but are still x-equivalent to it.

Definition 5 (x-reduction)**.** *Let X be a set of clauses. We say that a clause $\widehat{A}$ is an x-reduction of clause A if and only if $\widehat{A} \preceq_\theta A$ and $A \preceq_X \widehat{A}$ and if this does not hold for any $B \subsetneq \widehat{A}$ (i.e. if there is no $B \subsetneq \widehat{A}$ such that $B \preceq_\theta A$ and $A \preceq_X B$).*

For a given clause, there may be even smaller x-equivalent clauses than its x-reductions. There may also be multiple x-reductions differing by their lengths for a single clause.

In order to be able to compute x-reductions, we would need to be able to decide x-subsumption. However, we very often have only an efficient x-presubsumption. Importantly, if there is an x-presubsumption $\lhd_X$ w.r.t. a set

X decidable in polynomial time then there is a polynomial-time algorithm for computing possibly larger clauses with the same properties as x-reductions. We call this algorithm *literal-elimination algorithm*. It works as shown in the pseudo-code below.

Literal-elimination algorithm:

```
1. Given a clause $A$ which should be reduced.
2. Set $A' := A$.
3. Select a literal $L$ from $A'$ such that $A' \lhd_X A' \setminus \{L\}$. If there is no such literal, return $A'$ and finish.
4. Set $A' := A' \setminus \{L\}$
5. Go to step 3.
```

The next proposition states formally the properties of the literal-elimination algorithm. It also gives a bound on the size of the reduced clause which is output of the literal-elimination algorithm. This bound is given in terms of lengths of conventional θ-reductions.

Proposition 2. *Let us have a set X and a polynomial-time decision procedure for checking $\lhd_X$ which is an x-presubsumption w.r.t. the set X. Then, given a clause A on input, the literal-elimination algorithm finishes in polynomial time and outputs a clause $\widehat{A}$ satisfying the following conditions:*

1. *$\widehat{A} \preceq_\theta A$ and $A \preceq_X \widehat{A}$ where $\preceq_X$ is an x-subsumption w.r.t. the set X.*
2. *$|\widehat{A}| \leqslant |\widehat{A}_\theta|$ where $\widehat{A}_\theta$ is θ-reduction of a subset of A's literals with maximum length.*

So, the output $\widehat{A}$ of the literal-elimination algorithm has the same properties as x-reduction ($\widehat{A} \preceq_\theta A$ and $A \preceq_X \widehat{A}$) with one difference and that is that it may not be minimal in some cases.

Next, we show how x-reductions in general, and the literal-elimination algorithm in particular, can be used in bottom-up approaches to relational learning. We introduce a novel concept which we term *bounded least general generalization*. This concept generalizes Plotkin's *least general generalization* of clauses [75], which we formally define in Section 7.4.

Definition 6 (Bounded Least General Generalization). *Let X be a set of clauses. A clause B is said to be a bounded least general generalization w.r.t. the set X of clauses A_1, A_2, ..., A_n (denoted by $B = LGG_X(A_1, A_2, \ldots, A_n)$) if and only if $B \preceq_\theta A_i$ for all $i \in \{1, 2, \ldots, n\}$ and if for every other clause $C \in X$ such that $C \preceq_\theta A_i$ for all $i \in \{1, 2, \ldots, n\}$, it holds $C \preceq_\theta B$.*

Note that neither the clauses A_1, A_2, ... A_n nor the resulting bounded least general generalization have to be from the set X. The set X serves only to specify the clauses which, if they θ-subsume the clauses A_1, A_2, ... A_n, must

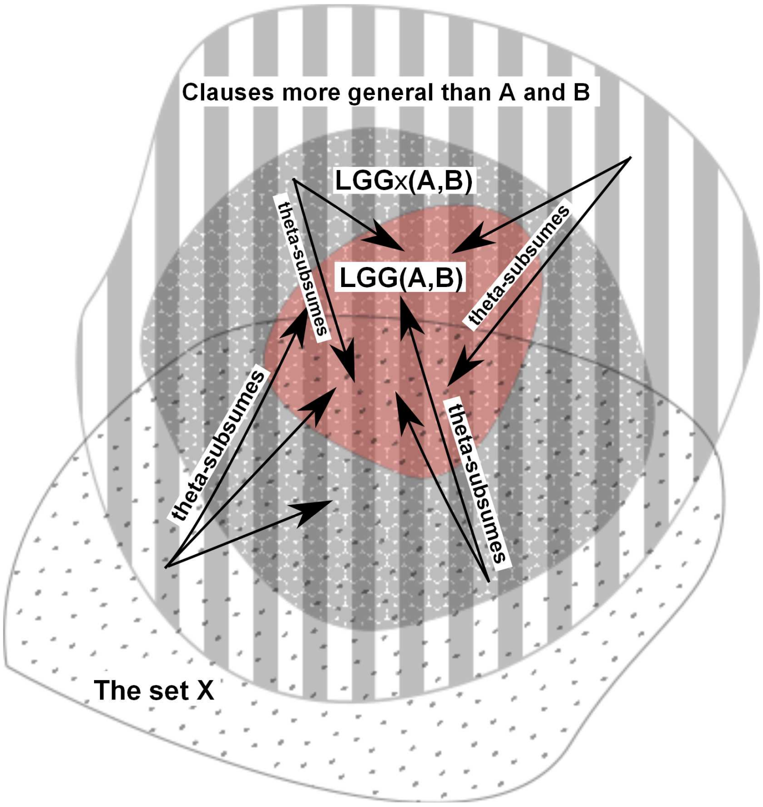


Figure 12: The relationship between the conventional least general generalization and the bounded least general generalization w.r.t. a set X.

be more general than the resulting bounded least general generalization. The set of all bounded least general generalizations of clauses $A_1, A_2, \ldots A_n$ w.r.t. a set X is a superset of the set of conventional least general generalizations of these clauses. This set of all bounded least general generalizations of clauses $A_1, A_2, \ldots A_n$ is also a subset of the set of all clauses which θ-subsume all A_1, $A_2, \ldots A_n$. The relationship between bounded and conventional least general generalization is depicted in Fig. 12.

We have already mentioned that reduced forms of bounded least general generalizations can often be computed in polynomial time using the literal-elimination algorithm. The method to accomplish this is based on application of x-reductions. This is formalized in the next proposition.

Proposition 3. *Let X be a set of clauses and let $\lhd_X$ be an x-presubsumption w.r.t. the set X then the clause*

$$B_n = \mathsf{litelim}_X(\mathsf{LGG}(A_n, \mathsf{litelim}_X(\mathsf{LGG}(A_{n-1}, \mathsf{litelim}_X(\mathsf{LGG}(A_{n-2}, \ldots)))))$$

is a bounded least general generalization of clauses $A_1, A_2, \ldots, A_n$ w.r.t. the set X (here, $\mathsf{litelim}_X(\ldots)$ denotes calls of the literal-elimination algorithm using $\lhd_X$ and LGG denotes Plotkin's least general generalization operator).

Conventional least general generalization can be used as an operator in searching for hypotheses [64, 32]. Basically, the search can be performed by iteratively applying LGG operation on examples or on already generated reduced LGGs. Application of reduction procedures, which compute smaller but equivalent least general generalizations, is usually necessary for keeping sizes of clauses constructed by LGG operator reasonably small. θ-reduction cannot guarantee that the sizes of least general generalizations would not grow exponentially in the worst case, but it is able to reduce the sizes of the clauses (and therefore also runtime and memory consumption) significantly in non-pathological cases. A problem of approaches based on conventional least general generalization is that computing θ-reduction is an NP-hard problem, which is especially problematic when the size of least general generalizations is large. In general, bounded reduction also cannot guarantee that the size of the constructed clauses will not be too large (in the end, θ-reduction is a special type of bounded reduction). Nevertheless, if we have a set of clauses X w.r.t. which there exists a polynomial-time literal-elimination algorithm then, at least time complexity of the reduction algorithms is not an issue. For these reasons, bounded least general generalization seems to be a suitable method for bottom-up learning.

There are only a few problems which can be solved by finding a single clause cleanly splitting positive and negative examples. More often, it is the case that we need to find a set of clauses which together cover as many positive examples and as few negative examples as possible (which is typically expressed through maximization of a scoring function). The existing systems such as Aleph [83] or ProGolem [65] usually tackle this task by an iterative covering approach in which a single clause obtaining good score is found in each iteration and the positive examples covered by it are removed from the dataset so that the clauses found in the subsequent iterations would cover the other, not yet covered, positive examples. Next, we describe an iterative covering approach using bounded least general generalization called BULL.

The single-clause learning component of the main algorithm called S-BULL[1] (Algorithm 2) is based on best-first search [79] in which each new candidate clause is constructed by computing bounded least general generalization of an already constructed clause and an example not yet $\mathcal{X}$-subsumed by it. This way, the algorithm would be exactly the best-first search algorithm where the used scoring function is the difference of the number of covered positive and negative examples. However, unlike straightforward implementation of the best-first-search algorithm, S-BULL contains also a step in which bounded least general generalization of the newly constructed clause and the positive examples covered by it w.r.t. the $\mathcal{X}$-presubsumption $\lhd_X$ is computed (the innermost repeat-until loop starting on line 15). This step ensures that the positive examples covered by a constructed hypothesis w.r.t. the $\mathcal{X}$-presubsumption $\lhd_X$ will be covered by it also w.r.t. the ordinary θ-subsumption.

The S-BULL procedure is used as a part of the iterative covering algorithm BULL which works similarly as Golem [64] or the iterative variant of ProGolem [65]. The covering algorithm starts with the full set of positive examples. It randomly picks a positive example and uses the S-BULL procedure

[1] **S**ingle-Clause **B**ottom-**U**p **L**earner (**L**)

Algorithm 2 S-BULL: the clause-learning component of the main algorithm

```
Input: Set of positive examples $\mathcal{E}^+$, set of negative examples $\mathcal{E}^-$, seed example $S$.
Open := () /* Open is a list with elements sorted by score. */
Closed := {} /* Closed is a set. */
$PosExCovBySeed := \{E \in \mathcal{E}^+ | S \lhd_X E\}$
$NegExCovBySeed := \{E \in \mathcal{E}^- | S \lhd_X E\}$
$BestScore := |PosExCovBySeed| - |NegExCovBySeed|$
$BestClause := Seed$
Store the triple $(S, PosExCovBySeed, NegExCovBySeed)$ in the list Open.
Store the pair $(PosExCovBySeed, NegExCovBySeed)$ in the set Closed.
while $Open \neq \emptyset$ do
    $(H, \{E^+_{i_1}, E^+_{i_2}, \dots, E^+_{i_n}\}, \{E^-_{j_1}, E^-_{j_2}, \dots, E^-_{j_m}\})$ := get the item with the highest score from Open and remove it from Open.
    for all $E^* \in \mathsf{CandidateExamples}(H, \mathcal{E}^+)$ do
        $H^* := LGG_X(H, E^*)$
        $PosCovered := \{E^+_{i_1}, E^+_{i_2}, \dots, E^+_{i_n}\} \cup \{E^+\}$
        repeat
            $NewPosCovered := \{E \in (\mathcal{E}^+ \setminus PosCovered) | H^* \lhd_X E\}$
            $PosCovered := PosCovered \cup NewPosCovered$
            $H^* := LGG_X(H^*, A_1, \dots, A_k)$ where $A_i \in NewPosCovered$
        until $NewPosCovered = \emptyset$
        $NewNegCovered := \{E \in (\mathcal{E}^- \setminus \{E^-_{j_1}, \dots, E^-_{j_m}\}) | H^* \lhd_X E\}$
        $NegCovered := NewNegCovered \cup \{E^-_{j_1}, \dots, E^-_{j_m}\}$
        if $(PosCovered, NegCovered \cup NewNegCovered) \notin Closed$ then
            Store the triple $(S, PosCovered, NegCovered)$ in the list Open.
            Store the pair $(PosCovered, NegCovered)$ in the set Closed.
            $Score := |PosCovered| - |NegCovered|$
            if $Score > BestScore$ then
                $BestClause := H^*$
                $BestScore := Score$
            end if
        end if
    end for
end while
```

with the randomly picked example as a seed example to find a hypothesis (a clause) maximizing the difference of covered positive and negative examples. Then it stores the found clause in a list of already found clauses and removes the set of examples covered by the found clause from the set of examples. Then it repeats this process using the remaining set of positive examples until all positive examples are removed. The result is a set of clauses.

7.2 EXPERIMENTS

We performed experiments with three datasets of antimicrobial peptides (CAMEL, RANDOM and BEE – which are class-labelled according to their antimicrobial activity: higher-than-median or lower-than-median) in order to evaluate the practical potential of the novel concept of bounded least general generalization. We implemented a basic version of the BULL algorithm as described in Section 7.1, using x-presubsumption w.r.t. the intersection of the set of all clauses with treewidth 1 and the set of clauses restricted by the user-definable language bias (described in Section 7.4). We used the AC-3 arc-consistency algorithm [58] for checking x-presubsumption. We implemented the version of the literal-elimination algorithm presented in Section 7.1 with the following improvement. If a clause consists of several independent components[2] then the algorithm first compares the components in a pairwise manner using the x-presubsumption and if some component x-subsumes another component then the algorithm removes it. Before reducing the components, the algorithm sorts them by their lengths. If the number of literals of a component encountered during the reduction is greater than a given limit (set to infinity for the experiments reported in Section 7.2.1 and to 1000 literals in the experiments reported in Section 7.2.3) then the component is replaced by a clause composed of a randomly selected subset of its literals. The implemented algorithm also allows the user to set the maximum number of hypotheses expanded in a single run of the S-BULL procedure and the number of examples that should be sampled as candidates for being used for generalizing the current hypothesis in the S-BULL component of the algorithm. In addition, it is possible to set the maximum number of negative examples covered by a learned clause. Since we are mainly interested in the practical potential of the bounded least general generalization operation and not in the performance of heuristic strategies how to select the best theory from a set of clauses, the implemented version of BULL uses only the basic iterative covering strategy described in Section 7.1 which is repeated for the given number of times (set to 3 in all the experiments).

7.2.1 *Bounded Reductions*

The first question that we addressed was how much x-reductions of clauses (w.r.t. the set of all treelike clauses) differ from the respective θ-reductions

[2] We say that a clause C consists of more than one connected component if it can be rewritten as $C = C_1 \cup C_2$ where $C_1 \neq \emptyset$, $C_2 \neq \emptyset$ and $vars(C_1) \cap vars(C_2) = \emptyset$.

and how much faster they can be computed as compared to θ-reductions. We implemented a θ-reduction algorithm using the CSP representation of θ-subsumption problems. The underlying CSP solver was based on backtracking search using *maintaining-arc-consistency* algorithm and the *min-domain heuristic* [78]. The θ-reduction algorithm was implemented with the same level of sophistication as the literal-elimination algorithm. Similarly as the literal-elimination algorithm, first, it performed elimination of connected components and only after that it performed elimination of individual literals. We compared this θ-reduction algorithm with the literal-elimination algorithm w.r.t. the set of all treelike clauses. We performed experiments with clauses from the datasets of antimicrobial peptides CAMEL, RANDOM and BEE. We did not perform experiments on any dataset of DNA-binding proteins, because they are still beyond the reach of any bottom-up learning algorithm – including BULL, because of the large size of these datasets. The clauses for reduction were created by sampling 1000 pairs of clauses from each dataset and then computing LGGs of these pairs of clauses w.r.t. the same language bias as used in Section 7.2.3. We measured the runtime for reduction, sizes of reduced clauses and also for each clause whether its θ-reduction and the output of the literal-elimination algorithm were isomorphic. The results are shown in Table 14.

	Literal elimination		**θ-reduction**		
	Runtime [ms]	Average size	Runtime [ms]	Average size	Isomorphic
CAMEL	11.9	162.7	406.0	162.7	100 %
RANDOM	4.0	52.17	27.6	52.18	99.7 %
BEE	5.2	80.5	48.7	80.5	100 %

Table 14: Runtime in milliseconds and average sizes, measured as number of literals, of reduced clauses for the literal-elimination and the θ-reduction algorithm.

As can be seen from the results, the literal-elimination algorithm is not only substantially faster in most cases than the θ-reduction algorithm, but it also outputs clauses which are isomorphic (i.e. identical up to renaming of variables) to θ-reductions in majority of cases. Naturally, there are examples where the literal elimination must return a clause non-isomorphic to the respective θ-reduction but it is still interesting that isomorphic clauses were returned in most cases by the literal-elimination algorithm and the θ-reduction algorithm for the three real-life datasets used in this experiment. Interestingly, when we disabled the initial stage in which components are reduced in pairwise manner, the literal-elimination algorithm started to return clauses smaller than θ-reductions (and thus non-isomorphic to it) more often which might be attributed partly to sorting of components by their sizes before their pairwise reduction.

7.2.2 *Comparison to a Baseline Bottom-up Learning Algorithm*

The results presented in the previous section indicate that the literal-elimination algorithm w.r.t. the set of treelike clauses can compute reductions of clauses significantly faster than the ordinary θ-reduction algorithm and that the sizes of the reduced clauses are usually equal or even smaller than those of the respective θ-reductions. These results do not answer another question and that is how much faster bottom-up relational learning can be and how much of its accuracy is lost when bounded least general generalization is used instead of the ordinary least general generalization. In order to answer this question, we compared BULL using the x-presubsumption w.r.t. the set of treelike clauses (T-BULL) and BULL using θ-subsumption as the x-presubsumption w.r.t. the set of all clauses (θ-BULL). In θ-BULL, we disabled the loop which ensures that all clauses covered w.r.t. a chosen x-presubsumption will be covered also w.r.t. θ-subsumption (because it is unnecessary when the x-presubsumption is θ-subsumption). We used T-BULL and θ-BULL with the following settings for all datasets. We set the maximum number of expanded hypotheses to 30 and the maximum number of covered negative examples to 0. The results of 10-fold cross-validation are shown in Table 15. It can be seen from the results that the accuracies are almost identical but that T-BULL is substantially faster than θ-BULL – for instance, twenty-times faster for the dataset CAMEL which contains the longest peptides from the three datasets.

	T-BULL		**θ-BULL**	
	Runtime [s]	Accuracy	Runtime [s]	Accuracy
CAMEL	462	86.2 ± 8.3	9078	86.2 ± 8.3
RANDOM	179	88.0 ± 10.1	335	88.0 ± 10.1
BEE	47	57.0 ± 17.5	101	57.0 ± 17.5

Table 15: Runtime in seconds and accuracy estimated by 10-fold cross-validation for BULL using x-presubsumption w.r.t. the set of treelike clauses (T-BULL) and for BULL using θ-subsumption (θ-BULL).

7.2.3 *Comparison with Existing Relational Learning Algorithms*

What is very important for judging the practical utility of bounded least general generalization is whether algorithms based on it can be competitive with existing relational learners in terms of predictive accuracy. For this reason, we performed predictive-classification experiments with the datasets CAMEL, RANDOM and BEE, in which we measured predictive accuracy (estimated by 10-fold cross-validation) of BULL and relational learning systems Aleph, nFOIL and ProGolem. The reason why we did not use these algorithms for comparison in Chapter 5 is that we showed that counting patterns are necessary for accurate DNA-binding propensity prediction and the algorithms considered in this chapter work only with existential patterns.

We set parameters of all four systems so that their runtime would be in the same orders of magnitude (tens of minutes per fold at most). We used BULL with the following settings for all the datasets. We set the maximum number of expanded hypotheses to 30 and the maximum number of covered negative examples to 0. For Aleph, we set the number of searched nodes to 50000, the *noise* parameter to 5 % and the maximum clause length to 100. nFOIL was used with maximum clause length set to 20 and beam size set to 100 for all datasets. For ProGolem, we used most of the parameters with their default values, except the clause-evaluation function, which we set to use the Subsumer algorithm [80], and the maximum number of covered negative examples. We used two different settings of the lastly mentioned parameter, giving rise to two columns in Table 16: ProGolem1 and ProGolem2. ProGolem1 refers to ProGolem with the default setting for the maximum number of covered negative examples, whereas ProGolem2 refers to ProGolem with the maximum number of covered negative examples set to 0.

	T-BULL	**Aleph**	**ProGolem1**	**ProGolem2**	**nFOIL**
CAMEL	**86.2** ± **8.3**	72.4 ± 14.5	80.5 ± 13.1	78.3 ± 10.1	83.3 ± 11.3
RANDOM	**88.0** ± **10.1**	86.0 ± 6.1	**88.0** ± **6.3**	81.0 ± 9.4	83.0 ± 5.9
BEE	57.0 ± 17.5	53.8 ± 16.5	56.8 ± 16.3	**57.5** ± **20.8**	47.7 ± 18.7

Table 16: Accuracy estimated by 10-fold cross-validation for the following systems: T-BULL, Aleph, ProGolem with default maximum number of covered negative examples (ProGolem1), ProGolem with zero maximum number of covered negative examples and nFOIL.

Results estimated by 10-fold cross-validation are shown in Table 16. The results indicate that BULL outperforms existing relational learning systems, including two recent ones: nFOIL and ProGolem.

7.2.4 *A discovered set of relational patterns*

The output of the algorithm BULL is a set of relational patterns which cover as many positive examples as possible and each of them covers at most a given number of negative examples. Typically, the output set of patterns tends to be small and therefore potentially more interpretable. Here, we show a set of relational patterns obtained for the dataset CAMEL in detail. It contains three relatively complex patterns shown in Figures 13, 14 and 15.

The first relational pattern P1 (Figure 13) assumes the presence of the following amino acids: lysine (LYS), leucine (LEU), phenylalanine (PHE) and valine (VAL) in certain distances from each other. A match of the relational pattern P1 is shown in the right of the figure schematically, and in the left of the figure within an antimicrobial peptide. The pattern covers 10 of 50 positive examples, and none of 51 negative examples.

P1 = residue(A,val), residue(B,leu), residue(C,lys), residue(D,phe), residue(E,val), residue(F,leu), residue(G,leu), residue(H,lys), residue(I,lys), residue(J,lys), residue(K,leu), residue(L,lys), dist(A,val,B,leu,4.0), dist(A,val,C,lys,6.0), dist(C,lys,D,phe,6.0), dist(D,phe,E,val,10.0), dist(E,val,F,leu,4.0), dist(D,phe,G,leu,4.0), dist(G,leu,H,lys,4.0), dist(D,phe,H,lys,6.0), dist(G,leu,I,lys,6.0), dist(J,lys,K,leu,6.0), dist(K,leu,L,lys,10.0)

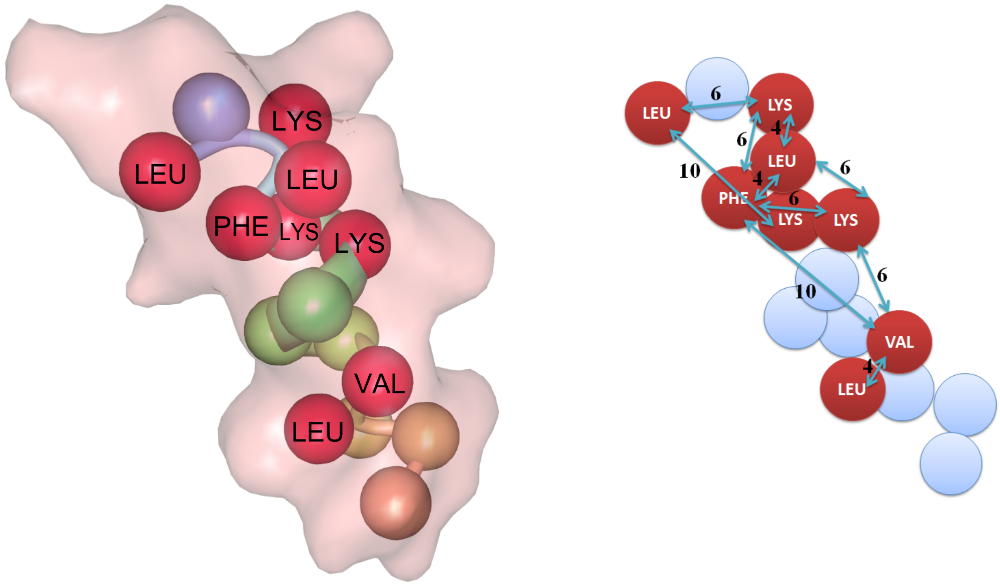


Figure 13: A match of a relational pattern P1 shown in the right schematically, and in the left within an antimicrobial peptide (using the protein viewer software [62]). Amino acids assumed by the pattern are indicated in red.

The second relational pattern P2 (Figure 14) assumes the presence of tryptophan (TRP), lysine, leucine, isoleucine (ILE) and valine in certain distances from each other. Similarly as for pattern P1, a match of this pattern is shown in the right of the figure schematically, and in the left of the figure within an antimicrobial peptide. The pattern covers 34 of 50 positive examples, and none of 51 negative examples.

The third relational pattern P3 (Figure 15) assumes the presence of tryptophan, lysine, leucine, glycine (GLY) and valine in distances from each other as shown in the right of the figure. The pattern covers 21 of 50 positive examples, and none of 51 negative examples.

The patterns in total cover 49 of 50 positive examples and none of the negative examples. This type of patterns seems to be suitable for synthetic development of antimicrobial peptides, in which a chemist could use a discovered set of patterns for designing new peptides as follows. He would synthesize antimicrobial peptides following the obtained patterns, measure their antimicrobial activity and could get a result confirming or disproving the hypothesis implied by the patterns. He could pick a peptide with high antimicrobial activity and match the discovered patterns against this peptide.

P2 = residue(A,val), residue(B,leu), residue(C,lys), residue(D,ile), residue(E,trp), residue(F,leu), residue(G,lys), residue(H,leu), residue(I,val), residue(J,ile), dist(A,val,B,leu,6.0), dist(B,leu,C,lys,6.0), dist(D,ile,E,trp,10.0), dist(E,trp,G,lys,8.0), dist(D,ile,F,leu,6.0), dist(H,leu,I,val,6.0), dist(H,leu,J,ile,6.0), dist(I,val,J,ile,10.0)

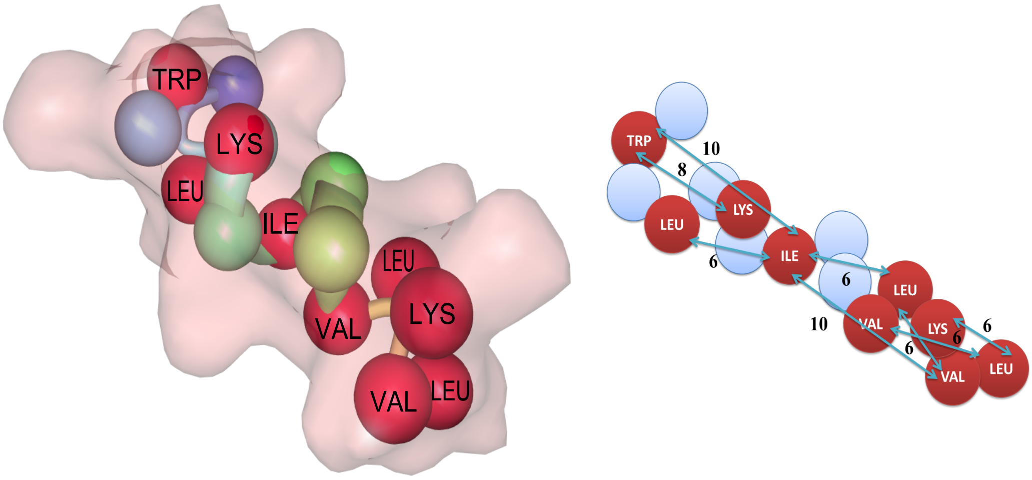


Figure 14: A match of a relational pattern P2 shown in the right within an antimicrobial peptide shown in the left (using the protein viewer software [62]). Amino acids assumed by the pattern are indicated in red.

Then he would have two possible options: if he was satisfied with the patterns, he could try to replace any of the amino acids not involved in the match and measure the antimicrobial activity of the new peptide. If he was not satisfied, he could try to replace any of the matched amino acids and measure the antimicrobial activity of the new peptide. According to the measured activity he could add the new peptide to the set of positive or negative learning examples. A new positive learning example could be used for generalization of the original set of patterns, a new negative example could prune the original set of patterns. Sometimes, when a sufficient number of new peptides is synthesized, it may be the case that the whole set of patterns needs to be recomputed from scratch.

7.3 CONCLUSIONS

In this chapter, we introduced the first bottom-up relational learning algorithm based on least general generalization which can exploit structural tractability bias while not restricting the form of learning examples. The algorithm is generic and can be used w.r.t. various sets X, representing the language bias. The returned clause itself does not have to be from the set X. If the set X contains only clauses for which θ-subsumption can be decided in polynomial time and if there is a suitable polynomial-time decidable x-presubsumption then the algorithm can use polynomial-time procedures for computing reduced forms of least general generalizations of clauses and for computing their coverage. Prior to this work, the only way structural tractability bias

P3 = residue(A,val), residue(B,leu), residue(C,leu), residue(D,lys), residue(E,trp), residue(F,lys), residue(G,lys), residue(H,leu), residue(I,lys), residue(J,gly), dist(A,val,B,leu,6.0), dist(A,val,C,leu,4.0), dist(A,val,D,lys,6.0), dist(E,trp,F,lys,4.0), dist(G,lys,H,leu,6.0), dist(H,leu,I,lys,4.0), dist(H,leu,J,gly,8.0)

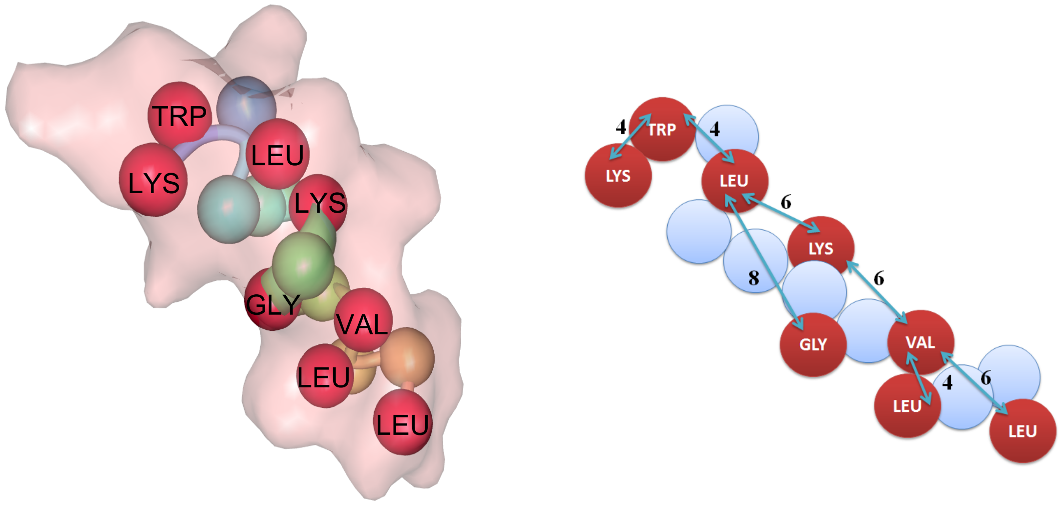


Figure 15: A match of a relational pattern P3 shown in the right within an antimicrobial peptide shown in the left (using the protein viewer software [62]). Amino acids assumed by the pattern are indicated in red.

could be exploited in bottom-up learning systems was to restrict the learning examples to be from a set for which θ-subsumption could be decided efficiently and which would be closed under formation of least general generalizations. Such an approach was pursued e.g. by Horváth et al. [32] for treelike clauses. Importantly, when subjected to comparative experimental evaluation on datasets of antimicrobial peptides, the new algorithm turned out to be very competitive to state-of-the-art relational learning systems.

7.4 ALGORITHMIC DETAILS

In this section, we describe algorithmic details of this chapter. In Section 7.4.1 we present necessary background concerning least general generalization. In Section 7.4.2 we describe practically relevant instantiations of the framework of the generic algorithm BULL. We show that bounded least general generalizations can be computed efficiently w.r.t. classes of clauses with bounded treewidth in Section 7.4.2.1 and w.r.t. to clauses complying with a simple language bias in Section 7.4.2.2. Finally, we provide proofs of propositions stated in this chapter in Section 7.4.3.

7.4.1 *Least General Generalization*

An important tool exploited in this chapter, which can be used for learning clausal theories, is Plotkin's least general generalization (LGG) of clauses.

Definition 7. *A clause* C *is said to be a least general generalization of clauses* A *and* B *(denoted by* $C = \mathrm{LGG}(A, B)$*) if and only if* $C \preceq_\theta A$, $C \preceq_\theta B$ *and for every clause* D *such that* $D \preceq_\theta A$ *and* $D \preceq_\theta B$ *it holds* $D \preceq_\theta C$.

A least general generalization of two clauses C, D can be computed in time $\mathcal{O}(|C| \cdot |D|)$.

7.4.2 *Instantiations of the Framework*

Here, we describe practically relevant instantiations of the framework of the generic algorithm BULL. We show that bounded least general generalizations can be computed efficiently w.r.t. classes of clauses with bounded treewidth and w.r.t. to clauses complying with a simple language bias.

7.4.2.1 *Clauses of Bounded Treewidth*

One of the classes of clauses w.r.t. which the reduced forms of bounded LGGs can be computed efficiently is the class of clauses with bounded treewidth. The notion of treewidth of clauses is explained in Section 3.1. What we need to show in order to demonstrate that the framework can be efficiently applied in the case of bounded-treewidth clauses is that there is a polynomial-time decidable x-presubsumption relation. In the next proposition, we show that k-consistency algorithm [2] can be used to obtain such an x-presubsumption.

Proposition 4. *Let* $k \in \mathbb{N}$ *and let* $\lhd_k$ *be a relation on clauses defined as follows:* $A \lhd_k B$ *if and only if the* k*-consistency algorithm run on the CSP-encoding (described in Section 3.1) of the* θ*-subsumption problem* $A \preceq_\theta B$ *returns true. The relation* $\lhd_k$ *is an* x*-presubsumption w.r.t. the set* X_k *of all clauses with treewidth at most* k*.*

Example 6. *Let us have the following four clauses which are all mutually* x*-equivalent.*

$$\begin{aligned}
A &= e(A,B), e(B,C), e(C,E), e(D,B), e(D,E), e(E,F), e(F,D)\\
B &= e(B,C), e(C,E), e(D,B), e(D,E), e(E,F), e(F,D)\\
C &= e(B,C), e(C,E), e(D,B), e(D,E)\\
D &= e(D,E), e(E,F), e(F,D)
\end{aligned}$$

These clauses are depicted graphically as graphs in Figure 16 together with the θ*-subsumption relations among them. The clause* B *is a* θ*-reduction of the clause* A*. The clauses* C *and* D *are* x*-reductions of* A *(and consequently also of* B*).*

Low-treewidth clauses can lead to highly accurate classifiers. In previous studies [49, 42], it was observed that all clauses learned by the ILP systems Progol, nFOIL [52] and kFOIL [51] in all the conducted experiments had treewidth 1 (after the removal of the variable formally identifying the learning example) although this had not been stipulated by the language bias. In a similar spirit, Horváth and Ramon [31] note that more than 99.9 percent of molecules in the NCI repository have treewidth lower than four. The classical ILP systems produce low-treewidth clauses mostly because their top-down

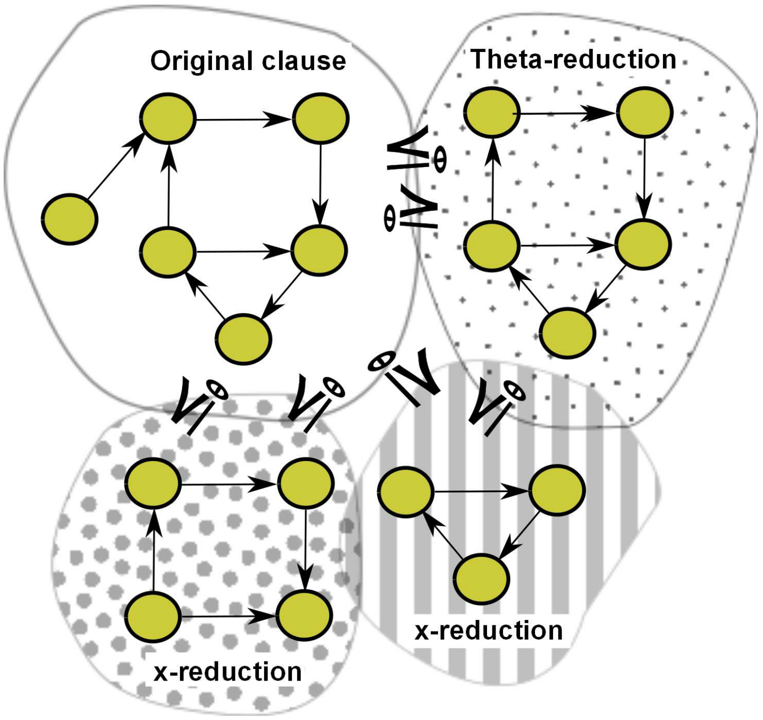


Figure 16: Illustration of θ-reduction, reduction by *literal-elimination algorithm* and reduction by *literal-substitution algorithm*.

search strategy does not allow them to reach long clauses of higher treewidth. Learning that produces clauses as good as those with treewidth bounded by some number is therefore of considerable practical importance.

7.4.2.2 *Clauses Constrained by a User-definable Language Bias*

In machine learning, we often need to introduce a bias corresponding to apriori knowledge which we have about a problem domain at hand. In this section, we introduce a simple language bias that can be combined with the framework of bounded least general generalization.

Definition 8 (Constant Language Bias). *Constant language bias is a set* $\mathcal{LB} = \{(p_i/arity_i, \{a_{i_1}, \ldots, a_{i_k}\})\}$ *where* p_i *are predicate symbols,* $a_i \in \mathbf{N}$ *and*

$$\{a_{i_1}, \ldots, a_{i_k}\} \subseteq \{1, \ldots, arity_i\}.$$

A literal $l = p_i(t_1, \ldots, t_k)$ *is said to comply with language bias* $\mathcal{LB}$ *if it contains constants in all arguments* $a_{i_1}, \ldots, a_{i_k}$*. A clause* C *is said to comply with language bias* $\mathcal{LB}$ *if all its literals comply with it.*

Informally, the *constant language bias* requires certain arguments of some literals to contain constants and not variables. We will use a simpler notation for constant language bias inspired by mode declarations known from Progol [63]. So, for example, we will write $atom(x, \#)$, $bond(x, x, \#)$ for a constant language bias $\{(atom/2, \{2\}), (bond/3), \{3\}\}$ which specifies clauses describing molecules where we require that if there is an atom in a learned hypothesis we want to know its type and, similarly, if there is a bond we also want to know whether it is a *single bond*, a *double bond* etc. For instance, the clause

$$\mathrm{atom}(X, c), \mathrm{bond}(X, Y, \mathrm{double}), \mathrm{atom}(Y, h)$$

complies with this language bias whereas the clause

$$\mathrm{atom}(X, c), \mathrm{bond}(X, Y, Z), \mathrm{atom}(Y, h)$$

does not comply with it because of the variable Z in the argument where a constant should appear.

Proposition 5. *Let $\mathcal{LB}$ be a language bias and let $X_{\mathcal{LB}}$ be the set of all clauses complying with $\mathcal{LB}$. Let A be a clause. If $A_{\mathcal{LB}} \subseteq A$ is a clause composed of exactly the literals from A complying with $\mathcal{LB}$ then $A_{\mathcal{LB}} \preceq_\theta A$ and $A \preceq_X A_{\mathcal{LB}}$ w.r.t. the set $X_{\mathcal{LB}}$.*

The language bias can be used to define a set of clauses w.r.t. which bounded least general generalizations can be computed. This is a simple corollary of Proposition 5 because any clause $C_{\mathcal{LB}}$ obtained by dropping literals which do not comply with the given language bias has the same properties as an x-reduction except that it might be non-minimal.

Example 7. *Let us have a language bias $\mathcal{LB} \approx e(x, x, \#)$ and two clauses*

$$\begin{aligned} A &= e(a, b, 1), e(b, a, 2) \\ B &= e(c, d, 1), e(d, e, 1), e(e, c, 1) \end{aligned}$$

The ordinary LGG of these clauses is

$$\begin{aligned} \mathrm{LGG}(A, B) &= e(A, B, 1), e(B, C, X), e(C, D, 1), \\ &\quad\; e(D, E, X), e(E, F, 1), e(F, A, X) \end{aligned}$$

The bounded LGG w.r.t. the set $X_{\mathcal{LB}}$ is much smaller

$$\mathrm{LGG}_{X_{\mathcal{LB}}}(A, B) = e(A, B, 1)$$

7.4.3 *Theoretical Details*

This section provides proofs of propositions stated in this chapter.

Here, we prove formally the properties of the literal-elimination algorithm described in Section 7.1.

Proposition 6. *Let us have a set X and a polynomial-time decision procedure for checking $\lhd_X$ which is an x-presubsumption w.r.t. the set X. Then, given a clause A on input, the literal-elimination algorithm finishes in polynomial time and outputs a clause $\widehat{A}$ satisfying the following conditions:*

1. *$\widehat{A} \preceq_\theta A$ and $A \preceq_X \widehat{A}$ where $\preceq_X$ is an x-subsumption w.r.t. the set X.*

2. $|\widehat{A}| \leqslant |\widehat{A}_\theta|$ *where* $\widehat{A}_\theta$ *is* θ*-reduction of a subset of* A*'s literals with maximum length.*

Proof. We start by proving $\widehat{A} \preceq_\theta A$ and $A \preceq_X \widehat{A}$. This can be shown as follows. First, $A \preceq_X A'$ holds in any step of the algorithm which follows from $(A' \lhd_X A' \setminus \{L\}) \Rightarrow (A' \preceq_X A' \setminus \{L\})$ and from transitivity of x-subsumption. Consequently we also have $A \preceq_X \widehat{A}$ because $\widehat{A} = A'$ in the last step of the algorithm. Second, $\widehat{A} \preceq_\theta A$ because $\widehat{A} \subseteq A$. Now, we prove the second part of the proposition. What remains to be shown is that the resulting clause $\widehat{A}$ will not be bigger than $\widehat{A}_\theta$. Since $\widehat{A} \subseteq A$, it suffices to show that $\widehat{A}$ cannot be θ-reducible. Let us assume, for contradiction, that it is θ-reducible. If $\widehat{A}$ was θ-reducible, there would have to be a literal $L \in \widehat{A}$ such that $\widehat{A} \preceq_\theta \widehat{A} \setminus \{L\}$. The relation $\lhd_X$ satisfies $(A \preceq_\theta B) \Rightarrow (A \lhd_X B)$ therefore it would also have to hold $A' \lhd_X A' \setminus \{L\}$. However, then L should have been removed by the literal-elimination algorithm which is a contradiction with $\widehat{A}$ being output of it. The fact that the literal-elimination algorithm finishes in polynomial time follows from the fact that, for a given clause A, it calls the polynomial-time procedure for checking the relation $\lhd_X$ at most $|A|^2$ times (the other operations of the literal-elimination algorithm can be performed in polynomial time as well). □

Proposition 3. Let X be a set of clauses and let $\lhd_X$ be an x-presubsumption w.r.t. the set X then the clause

$$B_n = \mathsf{litelim}_X(\mathsf{LGG}(A_n, \mathsf{litelim}_X(\mathsf{LGG}(A_{n-1}, \mathsf{litelim}_X(\mathsf{LGG}(A_{n-2}, \dots))))))$$

is a bounded least general generalization of clauses $A_1, A_2, \dots, A_n$ w.r.t. the set X (here, $\mathsf{litelim}_X(\dots)$ denotes calls of the literal-elimination algorithm using $\lhd_X$).

Proof. First, we show that $B \preceq_\theta A_i$ for all $i \in \{1, 2, \dots, n\}$ using induction on n. The base case $n = 1$ is obvious since then $B_1 = A_1$ and therefore $B_1 \preceq_\theta A_1$. Now, we assume that the claim holds for $n - 1$ and we will show that then it must also hold for n. First, $B_n = \mathsf{LGG}(A_n, B_{n-1})$ θ-subsumes the clauses $A_1, \dots, A_n$ which can be checked by recalling the induction hypothesis and definition of LGG. Second, $\mathsf{litelim}_k(\mathsf{LGG}(A_n, B_{n-1}))$ must also θ-subsume the clauses $A_1, \dots, A_n$ because $\mathsf{litelim}_k(\mathsf{LGG}(A_n, B_{n-1})) \subseteq \mathsf{LGG}(A_n, B_{n-1})$.

Again using induction, we now show that $C \preceq_\theta B_n$ for any $C \in X$ which θ-subsumes all A_i where $i \in \{1, \dots, n\}$. The base case $n = 1$ is obvious since then $B_1 = A_1$ and therefore every C which θ-subsumes A_1 must also θ-subsume B_1. Now, we assume that the claim holds for $n - 1$ and we prove that it must also hold for n. That is we assume that

$$C' \preceq_\theta B_{n-1} = \mathsf{litelim}_X(\mathsf{LGG}(A_{n-1}, \mathsf{litelim}_X(\mathsf{LGG}(A_{n-2}, \mathsf{litelim}_X(\mathsf{LGG}(A_{n-3}, \dots))))))$$

for any $C' \in X$ which θ-subsumes the clauses $A_1, A_2, \dots, A_{n-1}$. We show that then it must also hold $C \preceq_E B_n = \mathsf{litelim}_X(\mathsf{LGG}(A_n, B_{n-1}))$ for any $C \in X$ which θ-subsumes the clauses $A_1, A_2, \dots, A_n$. We have $C \preceq_\theta \mathsf{LGG}(A_n, B_{n-1})$

because $C \preceq_\theta B_{n-1}$ which follows from the induction hypothesis and because any clause which θ-subsumes both A_n and B_{n-1} must also θ-subsume $\mathsf{LGG}(A_n, B_{n-1})$ (from the definition of LGG). It remains to show that C also θ-subsumes $\mathsf{litelim}_X(\mathsf{LGG}(A_n, B_{n-1}))$. This follows from

$$\mathsf{LGG}(A_n, B_{n-1}) \preceq_X \mathsf{litelim}_X(\mathsf{LGG}(A_n, B_{n-1}))$$

(which is a consequence of Proposition 6) because if

$$\mathsf{LGG}(A_n, B_{n-1}) \preceq_X \mathsf{litelim}_X(\mathsf{LGG}(A_n, B_{n-1}))$$

then

$$(C \preceq_\theta \mathsf{LGG}(A_n, B_{n-1})) \Rightarrow (C \preceq_\theta \mathsf{litelim}_X(\mathsf{LGG}(A_n, B_{n-1})))$$

for any clause $C \in X$ (this is essentially the definition of x-subsumption). □

The next lemma giving a sufficient condition for a relation to be an x-presubsumption will be useful for proving that certain procedures are x-presubsumptions.

Lemma 1. *Let X be a set of clauses and $\lhd_X$ be a relation satisfying the following conditions:*

1. *If $A \lhd_X B$ and $C \subseteq A$ then $C \lhd_X B$.*
2. *If $A \in X$, ϑ is a substitution and $A\vartheta \lhd_x B$ then $A \preceq_\theta B$.*
3. *If $A \preceq_\theta B$ then $A \lhd_X B$.*

Then $\lhd_X$ is an x-presubsumption w.r.t. the set X.

Proof. The implication $(A \preceq_\theta B) \Rightarrow (A \lhd_X B)$ is already contained in the conditions in the statement of the lemma. We need to show that $(A \lhd_X B) \Rightarrow (A \preceq_X B)$, i.e. that if $A \lhd_x B$ then $(C \preceq_\theta A) \Rightarrow (C \preceq_\theta B)$ for all clauses $C \in X$. First, if $A \lhd_x B$ and $C \not\preceq_\theta A$ then the implication holds trivially. Second, $C \preceq_\theta A$ means that there is a substitution ϑ such that $C\vartheta \subseteq A$. This implies $C\vartheta \lhd_X B$ using the condition 1. Now, we can use the second condition which gives us $C \preceq_\theta B$ (note that $C \in X$ and $C\vartheta \lhd_X B$). □

Proposition 4. Let $k \in \mathbb{N}$ and let $\lhd_k$ be a relation on clauses defined as follows: $A \lhd_k B$ if and only if the k-consistency algorithm run on the CSP-encoding (described in Section 3.1) of the θ-subsumption problem $A \preceq_\theta B$ returns true. The relation $\lhd_k$ is an x-presubsumption w.r.t. the set X_k of all clauses with treewidth at most k.

Proof. We need to verify that $\lhd_k$ satisfies the conditions stated in Lemma 1

1. *If $A \lhd_k B$ and $C \subseteq A$ then $C \lhd_k B$.* This holds because if the k-consistency algorithm returns *true* for a problem then it must also return *true* for any of its subproblems. It is easy to check that if $C \subseteq A$ are clauses then the CSP problem encoding the θ-subsumption problem $C \preceq_\theta B$ is a subproblem of the CSP encoding of the θ-subsumption problem $A \preceq_\theta B$. Therefore this condition holds.

2. *If* $A \in X$, ϑ *is a substitution and* $A\vartheta \lhd_k B$ *then* $A \preceq_\theta B$. The CSP encoding of the problem $A \preceq_\theta B$ is a subproblem of the problem encoding $A\vartheta \preceq_\theta B$, in which there are additional constraints enforcing consistency with the substitution ϑ (because the set of constraints of the former is a subset of the constraints of the latter). Therefore if $A\vartheta \lhd_k B$ then also $A \lhd_k B$ and, since $A \in X$, it also holds $A \preceq_\theta B$.

3. *If* $A \preceq_\theta B$ *then* $A \lhd_k B$. This is a property of k-consistency.

□

Proposition 5. Let $\mathcal{LB}$ be a language bias and let $X_{\mathcal{LB}}$ be the set of all clauses complying with $\mathcal{LB}$. Let A be a clause. If $A_{\mathcal{LB}} \subseteq A$ is a clause composed of exactly the literals from A complying with $\mathcal{LB}$ then $A_{\mathcal{LB}} \preceq_\theta A$ and $A \preceq_X A_{\mathcal{LB}}$ w.r.t. the set $X_{\mathcal{LB}}$.

Proof. The first part of the proposition is obvious. If $A_{\mathcal{LB}} \subseteq A$ then $A_{\mathcal{LB}} \preceq_\theta A$. We show the validity of the second part by contradiction. We assume that $A \not\preceq_X A_{\mathcal{LB}}$. This means that there is a clause $C \in X_{\mathcal{LB}}$ such that $C \preceq_\theta A$ and $C \not\preceq_\theta A_{\mathcal{LB}}$. Let ϑ be a substitution such that $C\vartheta \subseteq A$. The substitution ϑ can map only literals complying with $\mathcal{LB}$ to literals also complying with $\mathcal{LB}$ (because constants cannot be mapped to variables) so $C\vartheta \subseteq A_{\mathcal{LB}}$ which also means $C \preceq_\theta A_{\mathcal{LB}}$. This is a contradiction with $C \not\preceq_\theta A_{\mathcal{LB}}$. □

8 ADVANCED PREPROCESSING TECHNIQUES

Relational representation of protein data are mostly huge. Reducing the complexity of such input data should be beneficial for learning. In attribute-value learning, a wide range of *feature selection* methods is available [56]. These methods try to select a strict subset of the original example attributes while maintaining or even improving the performance of the model learned from it with respect to that learned from the original attribute set. For binary classification tasks with Boolean attributes, the REDUCE [54] algorithm has been proposed that removes so called *irrelevant* attributes. For any model learned with the original attribute set, a model with same or better fit on the learning examples may be expressed without the irrelevant attributes.

As we have seen in previous chapters, examples are not expressed as tuples of attribute values in relational learning but rather take the form of relational constructs such as first-order clauses. Feature selection methods are thus not applicable to simplify such learning examples. Here we are interested to see whether also relational examples can somehow be reduced while guaranteeing that the set of relational patterns which can be learned from such reductions would not be affected.

An obvious approach would be to look for θ-reductions [75] of the input clauses. A θ-reduction of a clause is a smaller, but subsumption-equivalent (and thus also logically equivalent) clause. An approach based on θ-subsumption was explored before in [41], achieving learning speed-up factors up to 2.63. However, the main problem of θ-reduction is that finding it is an NP-hard problem, rendering the approach practically infeasible in domains with large examples such as those describing protein structures.

Here, we follow the key idea that the complexity curse can be avoided by sacrificing part of the generality of θ-reduction. In particular, we will look for reductions which may not be equivalent to the original example in the logical sense, but which are equivalent *given the language bias* of the learning algorithm. In other words, if the learning algorithm is not able to produce a hypothesis covering the original example but not covering its reduction (or vice versa), the latter two may be deemed equivalent. For instance, consider the clausal example $\leftarrow \mathsf{atom(a1), carbon(a1), bond(a1, a2)}, \ldots$, whose entire structure is shown in the left of Figure 17. Assume that all terms in hypotheses are variables and hypotheses must have *treewidth* at most 1. Then the learning example is equivalent to the simpler one shown in the right of Figure 17.

Our first main contribution is a formal framework for example reduction based on the given language bias for hypotheses. In this framework, we prove two propositions which can be used for showing that certain procedures which transform learning examples always produce reductions equivalent to the original examples under the given bias.

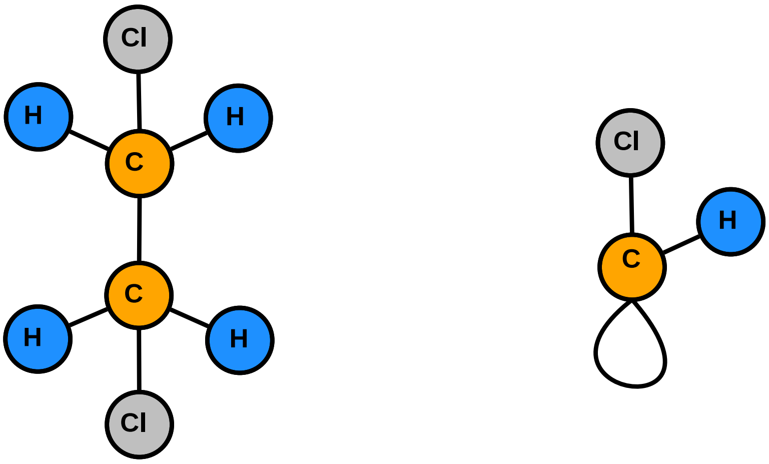


Figure 17: A learning example and its reduction.

Our second main contribution is the application of the above framework to the specific bias of *bounded treewidth* clauses. We show that in this case, interestingly, learning examples can be reduced in polynomial time, and moreover, that, in some cases, they can be reduced even more than they would be using the NP-hard θ-reduction.

As the previous paragraph indicates, the benefits gained from the bounded treewidth assumption are significant. Notably though, the price we pay for them is not too high in that the bias would be over-restrictive. First remind that, as a hypothesis bias, it only constrains the learned clauses, and not the learning examples. Second, low treewidth is in fact characteristic of clauses induced in typical ILP experiments. In [49] it was observed that all clauses learned by the ILP system Progol in all the conducted experiments had treewidth 1 although this had not been stipulated by the language bias. Similarly, in experiments in another study [42], all clauses learned by the systems nFOIL and kFOIL were of treewidth 1 after the removal of the variable formally identifying the learning example.

A salient feature of our approach is its general application scope. Indeed, example reduction can take place independently of the type of ILP learner employed subsequently. While we evaluate the approach in the standard ILP setting of learning from entailment, it is also relevant to propositionalization [39], which aims at the construction of attribute-based descriptions of relational examples. Interestingly, propositionalization can simultaneously benefit from both the relational example reduction step employed before the construction of patterns, and any feature selection algorithm applied subsequently on the constructed attribute set. In this sense, our approach is complementary to standard feature selection methods.

This chapter is structured as follows. In Section 8.1 we refer to the preliminaries for the study, namely θ-subsumption and reduction, their correspondence to the constraint satisfaction problem, and the concepts of tree decomposition and treewidth. Section 8.2 presents the framework for safe reduction of relational examples under a given hypothesis language bias. Section 8.3 instantiates the framework to the langauge bias of bounded-treewidth clauses and shows that reduction under this bias can be conducted effectively. Finally, we experimentally evaluate our method in Section 8.4.

8.1 PRELIMINARIES

Most of the concepts used in this chapter are defined in Chapter 7. This includes θ-subsumption, θ-reduction, CSP representation of θ-subsumption, k-consistency algorithm, tree decomposition and treewidth.

8.2 SAFE REDUCTION OF LEARNING EXAMPLES

The learning task that we consider in this chapter is fairly standard. We are given labelled learning examples encoded as first-order-logic clauses and we would like to find a classifier predicting the class labels of examples as precisely as possible. This task could be solved by numerous relational-learning systems. We aim at finding a reduction procedure that would allow us to reduce the number of literals in the examples while guaranteeing that the coverage of any hypothesis from a pre-fixed hypothesis language $\mathcal{L}$ would not be changed.

There are several settings for logic-based relational learning. In this chapter we will focus on the *learning from entailment setting* [13].

Definition 9 (Covering under Learning from Entailment)**.** *Let $\mathcal{H}$ be a clausal theory and e be a clause. Then we say that $\mathcal{H}$ covers e under entailment if and only if $\mathcal{H} \models e$.*

The basic learning task is to find a clausal theory $\mathcal{H}$ that covers all positive examples and no negative examples and contains as few clauses as possible.

Definition 10 (Safe Equivalence and Safe Reduction under Entailment)**.** *Let e and $\hat{e}$ be two clauses and let $\mathcal{L}$ be a language specifying all possible hypotheses. Then $\hat{e}$ is said to be safely equivalent to e if and only if $\forall \mathcal{H} \in \mathcal{L} : (\mathcal{H} \models e) \Leftrightarrow (\mathcal{H} \models \hat{e})$. If e and $\widehat{e}$ are safely equivalent and $|\hat{e}| < |e|$ then $\widehat{e}$ is called safe reduction of e.*

Clearly, if we have a hypothesis $\mathcal{H} \in \mathcal{L}$ which splits the examples to two sets X and Y then this hypothesis $\mathcal{H}$ will also split the respective set of safely reduced examples to the sets $\widehat{X}$, $\widehat{Y}$ containing the safely reduced examples from the sets X and Y, respectively. Also, when predicting classes of test-set examples, any deterministic classifier that bases its decisions on the queries using the covering relation $\models$ will return the same classification even if we replace some of the examples by their safe reductions. The same is also true for propositionalization approaches that use the $\models$ relation to construct boolean vectors which are then processed by attribute-value-learners.

In this chapter, we focus on hypothesis languages in the form of *non-resolving* clausal theories. Recall that we do not put any restrictions on the learning examples. The only restrictions are those put on hypotheses. A *non-resolving* clausal theory is a set of clauses such that no predicate symbol which appears in the head of a clause appears also in the body of any clause. The main reason why we start with non-resolving clausal theories is that logical entailment $\mathcal{H} \models A$, for a non-resolving clausal theory $\mathcal{H}$ and a clause A, can be checked using θ-subsumption. If there is a clause $H \in \mathcal{H}$ such that $H \preceq_\theta A$ then $\mathcal{H} \models A$, otherwise $\mathcal{H} \not\models A$.

We will use the notions of x-subsumption and x-equivalence introduced in Chapter 7 which are weaker versions of θ-subsumption and θ-equivalence. We may notice that x-presubsumption introduced in Chapter 7 can be used to check if two learning examples e and $\widehat{e}$ are equivalent w.r.t. hypotheses from a fixed hypothesis language. It can be therefore used to search for safe reductions of learning examples. This is formalized in the next proposition. Note that this proposition does not say that e and $\widehat{e}$ are equivalent. It merely says that they are equivalent when being used as learning examples in the *learning from entailment* setting with hypotheses drawn from a fixed set.

Proposition 7. *Let $\mathcal{L}$ be a hypothesis language containing only non-resolving clausal theories composed of clauses from a set X and let $\lhd_X$ be an x-presubsumption w.r.t. X. If e and $\widehat{e}$ are learning examples (not necessarily from X), $e \lhd_X \widehat{e}$ and $\widehat{e} \lhd_X e$ then for any $\mathcal{H} \in \mathcal{L}$ it holds $(\mathcal{H} \models e) \Leftrightarrow (\mathcal{H} \models \widehat{e})$. Moreover, if $|\widehat{e}| < |e|$ then $\widehat{e}$ is a safe reduction of e under entailment.*

Proof. First, $e \lhd_X \widehat{e}$ and $\widehat{e} \lhd_X e$ imply $e \approx_X \widehat{e}$ (where $\approx_X$ denotes x-equivalence w.r.t. the set X). Then for any non-resolving clausal theory $\mathcal{H} \in \mathcal{L}$ we have $(\mathcal{H} \models e) \Leftrightarrow (\mathcal{H} \models \widehat{e})$ because for any clause $A \in X$ we have $(A \preceq_\theta e) \Leftrightarrow (A \preceq_\theta \widehat{e})$ (from $e \approx_X \widehat{e}$). This together with $|\widehat{e}| < |e|$ means that $\widehat{e}$ is a safe reduction of e under entailment w.r.t. hypothesis language $\mathcal{L}$. □

We will use Proposition 7 for showing that certain procedures which transform learning examples always produce safe reductions of these examples. Specifically, we will use them to show that bounded reduction for bounded-treewidth clauses introduced in Chapter 7 based on k-consistency algorithm can be used for computing safe reductions of learning examples w.r.t. hypothesis sets composed of clauses with bounded treewidth.

We start with two simpler transformation methods. For the first transformation method, we assume to have a fixed hypothesis language $\mathcal{L}_\mathcal{U}$ consisting of non-resolving clausal theories which contain only constants from a given set $\mathcal{U}$. The transformation then gets a clause A on its input and produces a new clause $\tilde{A}$ by *variabilizing* constants in A which are not contained in $\mathcal{U}$. It is easy to check that for any such A and $\tilde{A}$ it must hold $A \approx_X \tilde{A}$ w.r.t. the set of clauses containing only constants from $\mathcal{U}$. Therefore A and $\tilde{A}$ are safely equivalent w.r.t. $\mathcal{L}$. We can think of the constants not used in a hypothesis language $\mathcal{L}$ as identifiers of objects whose exact identity is not interesting for us. Such constants can appear e.g. when we describe molecules and we want to give names to atoms in the molecules with no actual meaning.

Another simple transformation which produces safely equivalent clauses is based on θ-reduction. In this case the set of clauses X can be arbitrary. The transformation gets a clause A on its input and returns its θ-reduction. The

x-equivalence of the clause A and its θ-reduction follows from the fact that θ-subsumption is an x-subsumption w.r.t. the set of all clauses.

Importantly, transformations which produce x-equivalent clauses w.r.t. a set X can be chained due to transitivity of x-subsumption. So, for example, if we have a hypothesis language $\mathcal{L}_{\mathcal{U}}$ consisting of non-resolving clausal theories which contain only constants from a pre-fixed set $\mathcal{U}$ and we want to safely reduce a clause A then we can first variabilize it and then reduce it using θ-reduction.

Example 8. *Let us have an example*

$$e = \mathsf{edge}(a, b, 1), \mathsf{edge}(b, a, 2), \mathsf{edge}(b, c, 2), \mathsf{edge}(c, d, 1), \mathsf{edge}(d, a, 2)$$

and a hypothesis language $\mathcal{L}$ containing arbitrary non-resolving clausal theories with the set of allowed constants $\mathcal{U} = \{1, 2\}$. We variabilize e and obtain clause

$$\tilde{e} = \mathsf{edge}(A, B, 1), \mathsf{edge}(B, A, 2), \mathsf{edge}(B, C, 2), \mathsf{edge}(C, D, 1), \mathsf{edge}(D, A, 2).$$

Now, e and $\tilde{e}$ are safely equivalent w.r.t. hypotheses from $\mathcal{L}$. Next, we obtain a safe reduction of e by computing θ-reduction of $\tilde{e}$ which is $\widehat{e} = \mathsf{edge}(A, B, 1), \mathsf{edge}(B, A, 2)$.

8.3 REDUCTION UNDER THE BOUNDED TREEWIDTH ASSUMPTION

Here, we describe a transformation method which assumes the hypothesis languages to consist only of clauses with bounded treewidth. Unlike the exponential-time method based on θ-reduction, this method runs in time polynomial in the size of the reduced clause (though, with a multiplicative factor exponential in the fixed maximum treewidth of allowed hypotheses). Interestingly, it does not need any restrictions (e.g. bounded treewidth) on the learning examples which are reduced.

The reduction method is based on x-subsumption w.r.t. the set X_k of clauses with treewidth at most k. The safe reduction method based on x-subsumption w.r.t. the set X_k works as follows. We suppose that there is a set $\mathcal{U}$ of constants which are allowed in the hypothesis language $\mathcal{L}_k$ and that the hypotheses in $\mathcal{L}_k$ consist only of clauses with treewidth at most k. The method gets a clause A and variabilizes all constants not contained in $\mathcal{U}$. The result is a clause $\tilde{A}$ which is also safely equivalent to A w.r.t. the hypothesis language $\mathcal{L}_k$. This clause is then reduced by the *literal-elimination algorithm* which was described in Chapter 7. This algorithm runs in time polynomial in the size of the reduced clause (though, with a multiplicative factor exponential in the fixed maximum treewidth of allowed hypotheses) and always produces a clause $\widehat{A}$ which is safely equivalent to A w.r.t. $\mathcal{L}_k$ as Proposition 8 shows.

Proposition 8. *Let $\mathcal{L}_k$ be a set of non-resolving hypotheses containing only clauses with treewidth at most k. Let C be a clause and $\widehat{C}_\theta$ be the maximal θ-reduction of a subset of the literals in C. We can find a clause $\widehat{C}_k$ such that $C \approx_X \widehat{C}_k$ w.r.t. to $\mathcal{L}_k$ and $|\widehat{C}_k| \leqslant |\widehat{C}_\theta|$ in time $\mathcal{O}\left(|C|^{2k+3}\right)$ by the "literal-elimination algorithm".*

Proof. First, it follows from transitivity of k-equivalence that $\widehat{C}_k \approx_k C$. Therefore, $\widehat{C}_k$ and C are safely equivalent w.r.t. $\mathcal{L}_k$ which follows from Proposition 7. The bound on the size of the resulting clause follows from Proposition 6 stated in Chapter 7. □

We can find even smaller safely equivalent clauses w.r.t. $\mathcal{L}_k$ for a clause C by a *literal-substitution algorithm* with just a slightly higher runtime $\mathcal{O}(|C|^{2k+4})$. This algorithm first runs the *literal-elimination algorithm* and then tries to further reduce its output C' as follows: For each pair of literals $l, l' \in C'$ it constructs a substitution $\theta : vars(l) \to vars(l')$ and checks if $C'\theta \lhd_k C'$ and if so, it sets $C' \leftarrow C'\theta$. It is easy to check that it always holds $C \approx_X C'$ w.r.t. the set of clauses with treewidth at most k. The algorithm runs in time $\mathcal{O}(|C|^{2k+4})$ as it performs $\mathcal{O}(|C|^2)$ k-consistency checks.

The clauses with bounded treewidth are not the only ones for which efficient safe reduction can be derived. For example, it is possible to derive a completely analogical safe reduction w.r.t. acyclic clauses, which can have arbitrary high treewidth but despite that admit a polynomial-time θ-subsumption checking algorithm. The only difference would be the use of generalized arc-consistency algorithm [78] instead of the k-consistency test.

8.4 EXPERIMENTAL EVALUATION OF SAFE REDUCTION

In this section we evaluate usefulness of safe reduction on real-life datasets. We implemented *literal-elimination* and *literal-sbstitution* algorithms for treewidth 1, i.e. for tree-like clausal theories. We used the efficient algorithm AC-3 [58] for checking 1-consistency[1]. We forced nFOIL and Aleph to construct only clauses with treewidth 1 using their *mode declaration* mechanisms.

8.4.1 *DNA-binding Proteins and Antimicrobial Peptides*

One of the main motivations for studying example preprocessing techniques for relational data was the huge size of most protein datasets. Therefore, we started with experiments in which we tested how much relational descriptions of protein data can be reduced. We experimented on two datasets: APO104/NB110 and CAMEL described in Section 4. We used the representation described in Section 5.1 which represents proteins or peptides on the amino acid level. The reduction of the complete dataset CAMEL took about 11 seconds, whereas the reduction of the dataset APO104/NB110 took hours. However, there was no compression for either of the datasets. This is not so surprising given the structure of the relational representation. Moreover, it is good news for the relational representation that we devised, because if it was reducible then some substructures could not be distinguished by the type of patterns that we used, even in theory. Naturally, we expect that some substructures cannot be captured anyway. As we show in the next section, many

[1] Note again the terminology used in this paper following [2]. In CSP-literature, it is often common to call 2-consistency what we call 1-consistency.

similar problems – including problems involving proteins using different representation – can be reduced substantially.

8.4.2 *Other Related Problems*

We further experimentally evaluated usefulness of the *safe reduction of learning examples* with other real-life datasets and two relational learning systems – the popular system Aleph and the state-of-the-art system nFOIL [52]. One of these datasets is a dataset of proteins labelled according to whether they bind hexose. Unlike the datasets tested in Section 8.4.1 this dataset can be reduced substantially. The reason is that this particular dataset contains proteins described on the atomic level, whereas the datasets tested in Section 8.4.1 contain proteins and peptides described on the amino acid level using our relational learning representation described in Chapter 5.

We used three datasets in these experiments: *hexose-binding proteins* [67], *predictive toxicology challenge* (PTC) [29] and *CAD* [108]. The hexose-binding dataset contains 80 hexose-binding and 80 non-hexose-binding protein domains. Following [67] we represent the protein domains by atom-types and atom-names (each atom in an amino acid has a unique name) and pair-wise distances between the atoms which are closer to each other than some threshold value. We performed two experiments with the last mentioned dataset for cut-off set to 1 Angstrom (Hexose ver. 1) and 2 Angstroms (Hexose ver. 2). The PTC dataset contains descriptions of 344 molecules classified according to their toxicity for male rats. The molecules are described using only *atom* and *bond* information. Finally, the CAD dataset contains descriptions of 96 class-labelled product-structure designs.

We applied the *literal-elimination* algorithm followed by *literal-substitution* algorithm on these three datasets. The compression rates (i.e. ratios of number of literals in the reduced learning examples divided by the number of literals in the original non-reduced examples) are shown in the left panel of Figure 18. The right panel of Figure 18 then shows the time needed to run the reduction algorithms on the respective datasets. We note that these times are generally negligible compared to runtimes of nFOIL and with the exception of Hexose ver. 2 also to runtimes of Aleph.

8.4.2.1 *Experiments with nFOIL*

We used nFOIL to learn predictive models and evaluated them using 10-fold cross-validation. For all experiments with the exception of the hexose-binding dataset with cut-off value 2 Angstroms, where we used beam-size 50, we used beam-size 100. From one point of view, this is much higher than the beam-sizes used by [52], but on the other hand, we have the experience that this allows nFOIL to find theories which involve longer clauses and at the same time have higher predictive accuracies. The runtimes of nFOIL operating on reduced and non-reduced data are shown in the left panel of Figure 19. It can be seen that the reduction was beneficial in all cases but that the most significant speed-up of more than an order of magnitude was achieved on

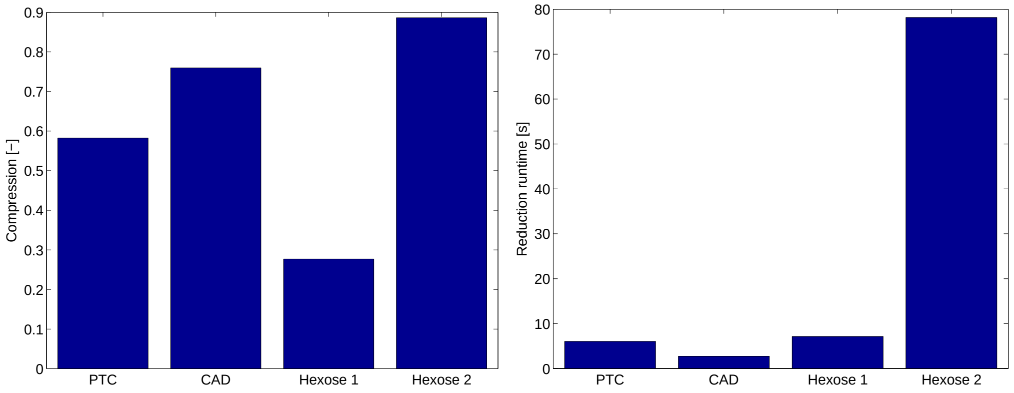


Figure 18: **Left:** Compression rates achieved by *literal-substitution algorithm* on four datasets (for treewidth 1). **Right:** Time for computing reductions of learning examples on four datasets (for treewidth 1).

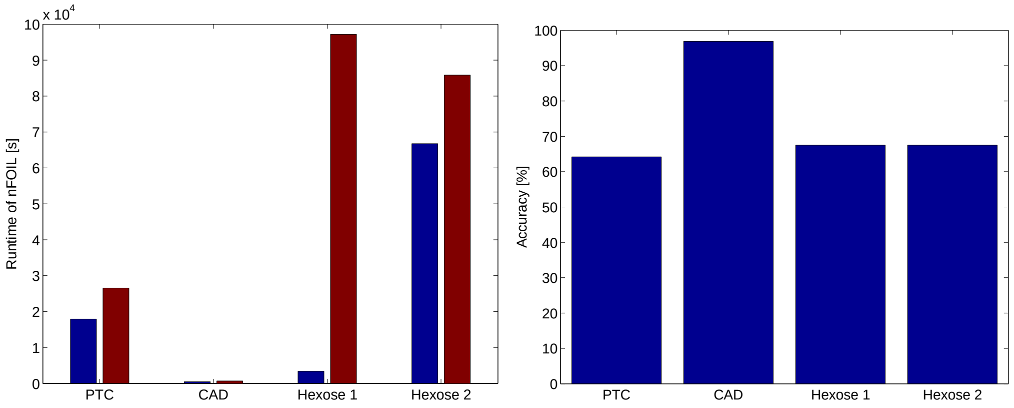


Figure 19: **Left:** Runtime of nFOIL on reduced (blue) and non-reduced (red) datasets. **Right:** Predictive accuracies of nFOIL on four datasets estimated by 10-fold cross-validation.

Hexose data. This could be attributed to the fact that nFOIL constructed long clauses on this dataset and the covering test used by it had not probably been optimized. So, in principle, nFOIL could be made faster by optimizing the efficiency of its covering test. The main point, however, is that we can speed-up the learning process for almost any relational learning algorithm merely by preprocessing its input. The right panel of Figure 19 shows nFOIL's predictive accuracies (estimated by 10-fold cross-validation). The accuracies were not affected by the reductions. The reason is that (unlike Aleph) nFOIL exploits learning examples only through the entailment queries.

8.4.2.2 *Experiments with Aleph*

We performed another set of experiments using the relational learning system Aleph. Aleph restricts its search space by bottom-clauses. After constructing a bottom-clause it searches for hypotheses by enumerating subsets of literals of the bottom-clause. When we reduce learning examples, which also means reduction of bottom-clauses, we are effectively reducing the size of Aleph's search space. This means that Aleph can construct longer clauses earlier than if it used non-reduced examples. On the other hand, this also implies that, with the same settings, Aleph may run longer on reduced data than on non-reduced data. That is because computing coverage of longer hypotheses is more time-consuming. Theories involving longer clauses may often lead to more accurate predictions. For these reasons, we measured not only runtime and accuracy, but also the average number of learnt rules and the average number of literals in these rules on reduced and non-reduced data.

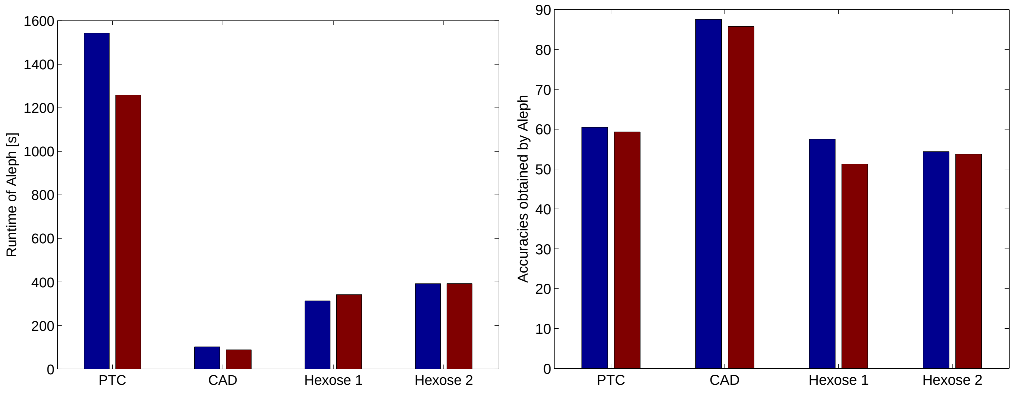


Figure 20: **Left:** Runtime of Aleph on reduced (blue) and non-reduced (red) datasets. **Right:** Predictive accuracies of Aleph on reduced (blue) and non-reduced (red) datasets estimated by 10-fold cross-validation.

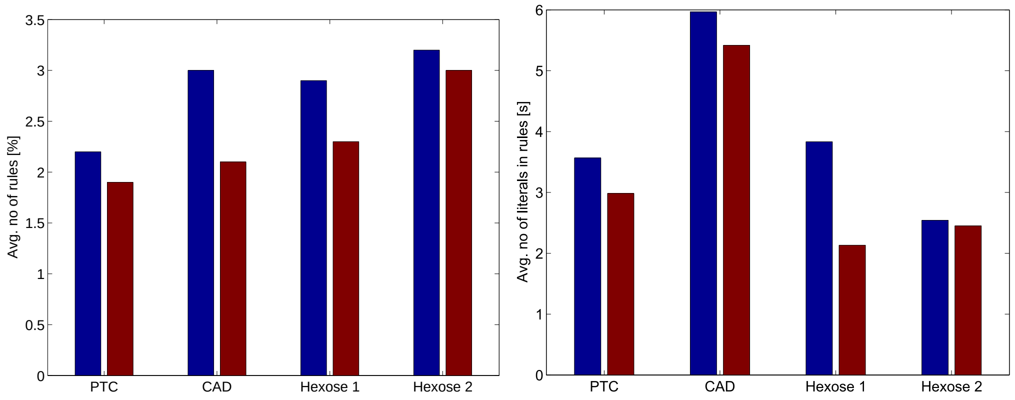


Figure 21: **Left:** Average number of rules generated by Aleph on reduced (blue) and non-reduced (red) datasets. **Right:** Average number of literals in rules generated by Aleph on reduced (blue) and non-reduced (red) datasets.

We ran Aleph on reduced and non-reduced versions of the datasets and evaluated it using 10-fold cross-validation. We used the literal-elimination algorithm for reducing examples. We set the maximum number of explored *nodes* to 50000, the *noise* parameter to 1% of the number of examples in the respective datasets. The runtime in the performed experiments was higher for reduced versions of datasets PTC and CAD, the same for Hexose ver. 2 and lower for Hexose ver. 1 than for their non-reduced counterparts (see left panel of Figure 20). The accuracies were higher for reduced versions of all four datasets (see right panel of Figure 20). Similarly, the average number of rules, as well as the average number of literals in the rules, was higher for the reduced versions of all four datasets (see Figure 21). These results confirm the expectation that Aleph should be able to construct longer hypotheses on reduced datasets which, in turn, should result in higher predictive accuracies.

8.5 conclusions

Reducing the complexity of huge relational data (such as descriptions of protein structures) can be beneficial for learning. We introduced a novel concept called *safe reduction*. We showed how it can be used to safely reduce learning examples (without affecting learnability) which makes it possible to speed-up many relational learning systems by merely preprocessing their input. The methods that we introduced run in polynomial time for hypothesis languages composed of clauses with treewidth bounded by a fixed constant. Our experiments confirm that these techniques can bring significant speed-up in some domains.

9 DISCUSSION OF RELATIONAL LEARNING APPROACHES

In this part, we presented several relational learning approaches for machine learning in molecular biology. These approaches are based on the relational representation of protein structures that we introduced in Section 5.1. Unlike other, previously introduced relational representations, e.g. the representation of Nassif et al. [67], our representation describes proteins on the amino acid level. The first two approaches presented in this part (relational learning based on structural patterns constructed by RelF and relational learning with polynomials) are based on combining a large number of relatively small structural patterns for building a prediction model. The third approach (relational learning with bounded LGG) is based on constructing a relatively small set of complex structural patterns which are used for predictive classification.

The advantage of the approaches based on a large number of small structural patterns is that small patterns can be mined efficiently even from large protein structures. Relational learning based on structural patterns described in Chapter 5 can be used for predictive classification and also for regression. As we have shown in experiments reported in Section 5.3.1, counting the occurrences of characteristic relational patterns in protein or peptide structures is crucial for accurate prediction. In the case of prediction of DNA-binding propensity of proteins, we achieved higher predictive accuracies than a state-of-the-art method of Szilágyi et al. [92]. In the case of prediction of antimicrobial activity of peptides, we were also able to outperform a state-of-the-art method of Torrent et al. [97]. A disadvantage of this approach is that it cannot handle real-valued variables efficiently. The second approach – relational learning with polynomials – described in Chapter 6 is designed to efficiently model multi-relational domains with numerical data. Unfortunately, this method does not scale as good as the former method. The reason is that the number of polynomial relational features is higher than the number of *non-aggregation* patterns. On the other hand, this method (using only information about primary and secondary structures) is able to obtain similar predictive accuracies as the method based on structural patterns. What is important, the idea of multivariate polynomial aggregation underlying this method serves as a motivation for the most accurate method of this thesis – presented in Chapter 15. The last approach – relational learning with bounded LGG – presented in Chapter 7, is able to construct large, complex relational patterns. Due to the nature of these complex patterns it is not necessary to count their occurrences in order to obtain high predictive accuracies. This was confirmed by the experimental results on datasets of antimicrobial peptides in Section 7.2. The disadvantage of this method is that it cannot handle protein structures with hundreds of amino acids.

Relational representation of protein data consists of thousands of literals. In order to reduce the complexity of such input data, relational counterparts of feature selection methods would be required. In Chapter 8 we presented advanced preprocessing techniques for reduction of relational learning examples. Interestingly, when applied on protein datasets in our representation described in Section 5.1, the reduction in size was small, which suggests that our representation is not redundant. It is good news for our relational representation, because if it was reducible then some substructures could not be distinguished by the type of patterns that we used, even in theory. When we applied the reduction method on protein structures represented on the atomic level, we obtained substantial reduction rates.

Part IV

DISTRIBUTION-BASED APPROACHES

10 MOTIVATION FOR DISTRIBUTION-BASED APPROACHES

We have shown that general relational learning methods can be used for learning from spatial structures of proteins (see Part iii). Nevertheless, they cannot exploit specific properties of these domains. This exhibits itself in decreased ability to scale. Therefore, it is useful to develop methods for relational data embedded in Euclidean space. In general, the problem would be computationally very hard, but if we restrict the possible pattern form, it can become tractable. The methods that we present in this part (Part iv) use restricted form of patterns and differ from conventional relational learning approaches also in their semantics. Here, we are not interested in whether a given pattern is present in an example or how many times it occurs in it, but we are interested in the probability that if we randomly pick a region of the learning example, it will match the given pattern. This enables us to capture distribution of certain substructures defined by a given pattern. This is why we call these methods *distribution-based approaches*.

Unlike, in the case of relational learning approaches with counting patterns, we cannot simply count the number of regions in which a given pattern is matched, because there are usually infinitely many of them. In the case of distribution-based approaches the value of a pattern is rather computed by a multidimensional integral over a given learning example. In practise, we approximate these integrals using Monte-Carlo integration.

From the biological point of view, very simple global properties of proteins (or peptides) can work well as predictors of their function, for example the physicochemical properties described by Szilágyi and Skolnick [92] turned out to be good predictors of DNA-binding function of proteins. Even the simplest of their features, i.e. proportion of some amino acids, could predict the DNA-binding function with surprisingly good accuracy[1]. The distribution-based approaches enable us to generalize the methods based on global properties of proteins (or peptides) by aggregating them over the regions of these proteins (or peptides).

We start this part by studying distributions of gaps between charged amino acids in DNA-binding proteins in Chapter 11. Here, the main question is whether the presence of charged amino acid patches in proteins' spatial structures reflects itself in differences of distributions of charged amino acids in proteins' sequences. Then, we introduce the tube-histogram method in Chapter 12. This method constructs classifiers based on a systematic exploration of the distribution of certain amino acids in the proteins' sequences. Next, we present the ball-histogram method in Chapter 13, which generalizes the tube-histogram method by lifting it to 3D space. The ball-histogram method is further extended for regression problems in Chapter 14. Finally, we intro-

[1] Though, with lower accuracy than accuracies reached by our methods

duce ball-histogram method with polynomial features in Chapter 15, which is able to work with continuous properties of protein regions, unlike the plain ball-histogram method.

11 DISTRIBUTIONS OF GAPS BETWEEN CHARGED AMINO ACIDS

Charged amino acids are known to be critical for the DNA-binding function of proteins. What is crucial is the spatial configuration of these amino acids, which is usually conserved by evolution. The question is: How does this reflect in the primary structure of proteins (i.e. in their sequence)? To answer it, we study distributions of charged amino acids in primary structure of DNA-binding proteins. In this chapter we give an evidence that the statistical laws governing the scaled distributions of positively charged amino acids[1] in proteins' primary structures differ between DNA-binding proteins on one hand, and non-DNA-binding proteins on the other hand. Clearly, since DNA-binding proteins typically contain larger fractions of positively charged amino acids, the respective *unscaled* distributions of gaps between them are very different from those of non-DNA-binding proteins. However, this tells us nothing more than what is already known, i.e. that the average proportion of positive and negative amino acids is different in DNA-binding proteins and non-DNA-binding proteins. In order to learn more about the distributions we need to cancel out the dependence on the proportion of charged amino acids. To this end we adopt methods used for analysis of intervals between consecutive events pursued by Goh et al. [26]. With these methods we show that there is a noticeable difference between distributions of positively charged amino acids in DNA-binding and non-DNA-binding proteins whereas there is only a small difference between the respective distributions of negatively charged amino acids. The results presented here serve as motivation for developing a predictive classification method described in the next chapter.

This chapter is organized as follows. We describe the method for studying distributions of gaps in amino acid sequences between charged amino acids in Section 11.1 and discuss the results in Section 11.2. In Section 11.3 we conclude this chapter.

11.1 METHOD

For each protein from given DNA-binding and non-DNA-binding datasets, we computed the set of lengths of all positive gaps, and an analogical set for all negative gaps. A positive gap in a protein with amino acid sequence[2] $a_1, a_2, \dots, a_L$ is a substring $a_i, a_{i+1}, \dots, a_j$, such that $i = 1$ or another positive gap ends in a_{i-1}, and j $(i < j \leqslant l)$ is the smallest index such that a_j is

[1] Under normal circumstances, arginine and lysine are positively charged, whereas Glu and Asp are charged negatively.

[2] In the entire chapter, smaller indices are closer to the N-terminus of the protein.

positively charged or $j = L$ if no such index exists. Negative gaps are defined analogically. Both concepts are illustrated in Figure 22.

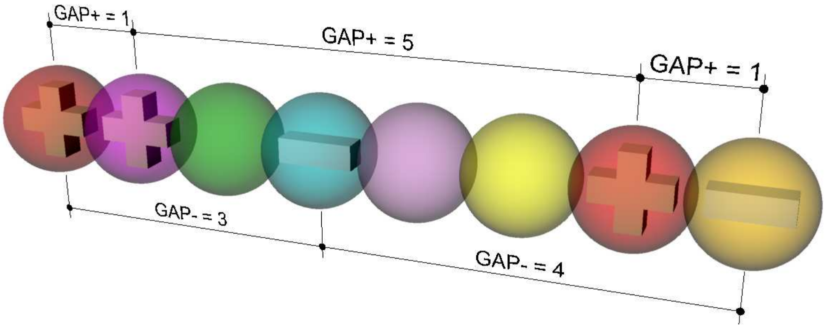


Figure 22: Positive (*GAP+*) and negative (*GAP-*) gaps in a protein primary sequence.

From the union of lists of gap lengths for all proteins in a single dataset we could get an estimate of the unscaled distribution of gap lengths. As we have noted above we are not interested in the unscaled distributions. Therefore we follow a trick from [26] which lies in computing the mean length l_P of gaps for every protein P in the dataset and then dividing each gap length associated with P by l_P. We compare variances of distributions which, as we will see, turn out to be reliable indicators of the differences between distributions of gap lengths of positively charged amino acids in DNA-binding and non-DNA-binding proteins. Finally, we also use plots of complementary distribution functions in logarithmic coordinates to allow visual inspection of the distributions.

11.2 EXPERIMENTS

Here, we present results concerning the differences between distributions of charged amino acids in DNA-binding proteins from the dataset PD138, and the non-DNA-binding proteins from the dataset NB110.

First, we computed cumulative distributions of gap lengths of positive amino acids and gap lengths of negative amino acids on the whole datasets which are shown in Figure 23. From these figures it is apparent that there is a much bigger difference between the gap-length distributions of positively charged amino acids in DNA-binding and non-DNA-binding proteins than the difference between the respective distributions for negatively charged amino acids. Next, we computed also variances of the distributions. Intuitively, the more slowly decaying is the tail of a distribution the bigger variance the distribution has, therefore variance is a good measure for the types of differences similar to those observed in Figure 23. For positively charged

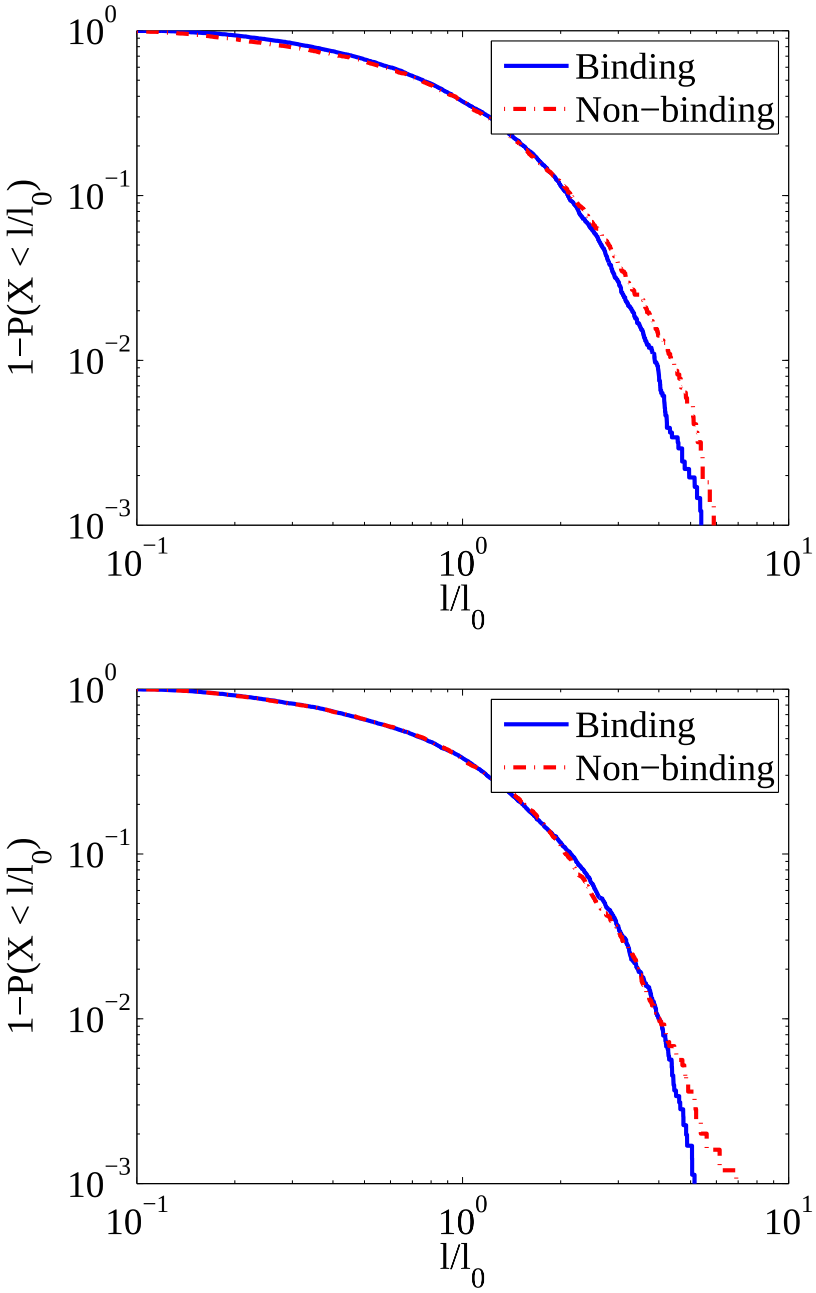


Figure 23: Complementary distributions of gap-lengths of positively charged amino acids (top panel) and negatively charged amino acids (bottom panel).

amino acids we obtained variance 0.67 in DNA-binding proteins and 0.82 in non-DNA-binding proteins. For negatively charged amino acids we obtained variance 0.71 in DNA-binding proteins and 0.75 in non-DNA-binding proteins. This is another piece of evidence supporting the claim that distributions of positively charged amino acids differ more significantly between DNA-binding and non-DNA-binding proteins than the negatively charged amino acids.

The differences observed in Figure 23 could be just due to the fact that proteins with low fraction of positively charged amino acids are more frequent

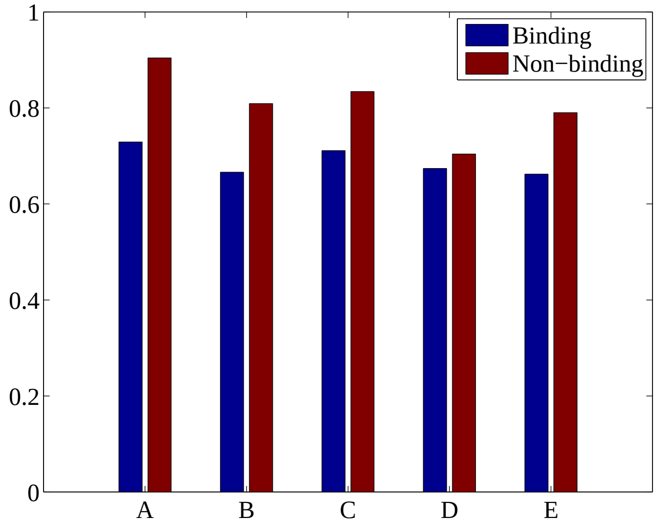


Figure 24: Variances of gap-length distributions for sets of proteins from PD138 and NB110 grouped according to proportion of positively charged amino acids (see main text).

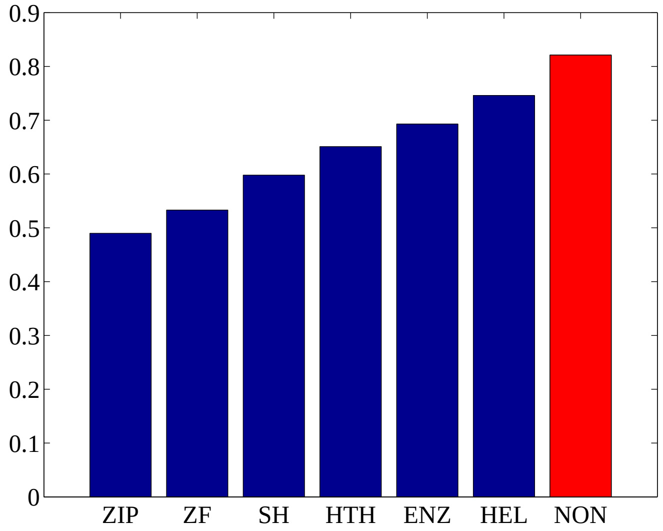


Figure 25: Variances of gap-length distributions for sets of proteins from PD138 and NB110 grouped according to their binding motifs: *helix-turn-helix (HTH), zinc coordinating (ZF), zipper-type (ZIP), other alpha helix (HEL), beta sheet (SH), enzyme (ENZ)* and non-binding proteins (NON).

in the dataset NB110 than in the dataset PD138. Thus, the positively charged amino acids in the non-DNA-binding proteins could have higher variance simply because the physical constraints on the distances of amino acids far away from each other in a sequence could allow for more variance in their distribution. In order to rule out this possibility we have split the datasets into several smaller sub-datasets according to the fraction of positively charged amino acids in the proteins. We have removed proteins from both sides of the spectra which corresponded to fractions too low for DNA-binding proteins and too high for non-DNA-binding proteins. Then we divided the datasets in such a way that each class (*DNA-binding* or *non-DNA-binding*) in a sub-dataset would contain at least ten proteins. This produced a breakdown into the following intervals: $A = (0.0675, 0.1055)$, $B = (0.1055, 0.118)$, $C = (0.118, 0.1289)$, $D = (0.1289, 0.134)$, $E = (0.134, 0.183)$. The variances of the gap-length distributions for the respective sub-datasets are shown in Figure 24. We can see that the variances corresponding to DNA-binding proteins are consistently smaller than the respective variances for non-DNA-binding proteins.

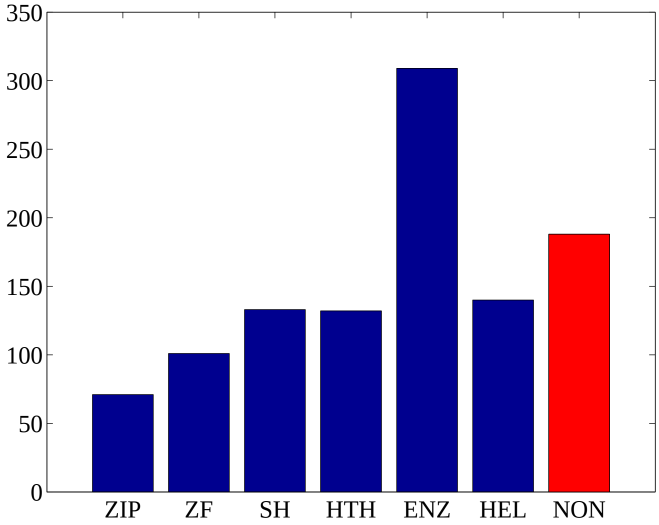


Figure 26: Average lengths of proteins from PD138 and NB110 grouped according to their binding motifs: *helix-turn-helix (HTH)*, *zinc coordinating (ZF)*, *zipper-type (ZIP)*, *other alpha helix (HEL)*, *beta sheet (SH)*, *enzyme (ENZ)* and non-binding proteins (NON).

Another question is whether the variance of gap-lengths depends on binding motifs of DNA-binding proteins; for example, whether there is a difference in distributions of positively charged amino acids in zinc finger proteins on one hand, and proteins having a helix-turn-helix motif on the other hand. Therefore we used a categorization of the proteins in dataset PD138 established by Szilagyi et al. [92], following the methodology of [57], into the following protein groups: *helix-turn-helix (HTH)*, *zinc coordinating (ZF)*, *zipper-type (ZIP)*, *other alpha helix (HEL)*, *beta sheet (SH)*, *enzyme (ENZ)*. For these groups we again computed variances of their gap-length distributions (shown in Figure 25). The displayed variances clearly differ among the protein groups.

Yet another potential confounding factor is that the observed differences may be largely due to the variable lengths of the respective proteins' chains. Indeed, the average lengths of proteins' chains in the groups vary substantially. However, as can be seen from Figure 26, the average lengths of chains do not correlate with the variances substantially. For example, the average length of chains in the group *enzyme* is by far the biggest, however, the variance is only medium in comparison with the other groups.

11.3 CONCLUSIONS

To summarize our findings in this chapter, we have given an evidence that the laws governing distributions of positively charged amino acids in DNA-binding proteins are different from those governing distributions of positively charged amino acids in non-DNA-binding proteins. We have shown that these differences probably cannot be attributed to different proportions of positively charged amino acids or to different average lengths of proteins present in the studied datasets PD138 and NB110.

Although, the results are interesting and suggest that distributions of positively and negatively charged amino acids are governed by different laws, they need further validation. An objective measure that lends itself readily is predictive accuracy. Therefore, we developed a predictive classification method which is able to capture these distributions. We present this method in the next chapter, where we also show that it can achieve high predictive accuracies using only limited information.

12 THE TUBE-HISTOGRAM METHOD

The results of experiments described in the previous chapter show that distributions of positively and negatively charged amino acids can be used to distinguish DNA-binding proteins from non-DNA-binding proteins. This was the motivation for the development of a novel predictive classification method entitled the *tube-histogram* method. This method is based on a systematic exploration of the distribution of certain amino acids in primary structures – for example the distribution of charged amino acids. For this purpose we employ so-called tube histograms, which are capable of capturing joint probabilities of specified amino acids occurring in certain distances in amino acid sequences from each other. The method consists of four main steps. First, so-called *templates* are found, which determine amino acids whose distributions should be captured by tube histograms. In the second step tube histograms are constructed for all proteins in a training set. Third, a transformation method is used to convert these histograms to a form usable by standard machine learning algorithms. Finally, a random forest classifier [6] is learned on this transformed dataset and then it is used for classification. We validate this method in prediction experiments, achieving favourable accuracies.

This chapter is organized as follows. First, we explain the basic principles of the tube-histogram method in Section 12.1. In Section 12.2 we describe a method for construction of templates capable to identify amino acids which are critical for DNA-binding propensity prediction. In Section 12.3 we describe the results obtained by application of the novel tube-histogram method. In Section 12.4 we conclude this chapter.

12.1 METHOD

Here, we explain the basic principles of the tube-histogram method. To describe *tube histograms*, we first define a few auxiliary terms. A *template* is a list of names of some Boolean amino acid properties. Given a template and a location in the primary structure of a protein, we infer a list of binary values indicating the truth values of the respective properties in the template for the amino acid at the position. For example, the template $(\mathsf{Arg}, \mathsf{Lys}, \mathsf{Positive}, \mathsf{Negative}, \mathsf{Neutral})$ acquires the value $(1, 0, 1, 0, 0)$ if the amino acid at the inquired position is an arginine. A *sampling-tube of size* s represents a part of an amino acid sequence containing s consecutive amino acids (Figure 27).

Given a protein, a template $\tau = (f_1, \dots, f_k)$ and a sampling-tube size s, a tube histogram is a k-dimensional histogram constructed as follows. Starting by placing the sampling tube on the first s amino acids we get the first *sample* for the histogram. When a sample is collected the numbers of amino acids complying with the particular properties listed in the given template are ex-

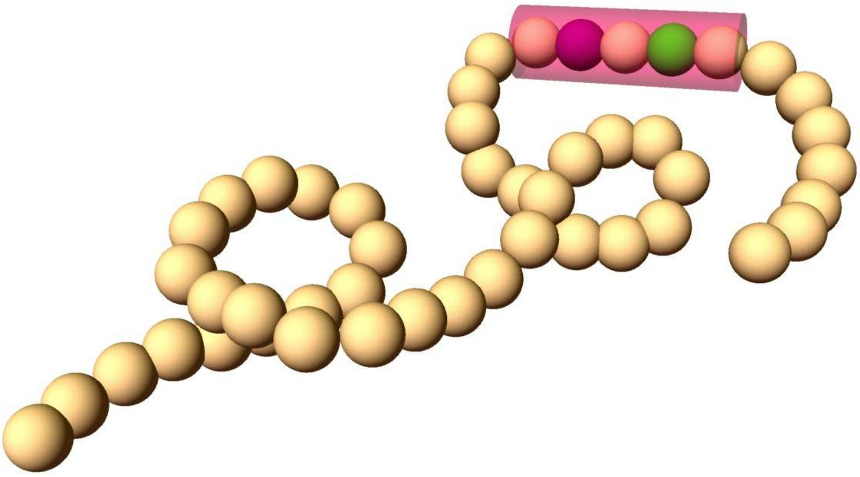


Figure 27: Illustration of the tube-histogram method - Amino acids are shown as small balls in sequence forming an amino acid chain. A tube is shown in pink. Some of the amino acids which comply with properties of an example template are highlighted inside the tube area. They have different colors according to their type.

tracted from it and stored. In further steps the tube is moved by one amino acid at time along the protein sequence and the samples are continuously stored for subsequent histogram construction. This process ends when the last amino acid is reached. Finally, the histogram constructed from the collected samples is normalized. Intuitively, tube histograms capture the joint probability that a randomly picked *sampling tube* (Figure 27) will contain exactly t_1 amino acids complying with f_1, t_2 amino acids complying with f_2 etc.

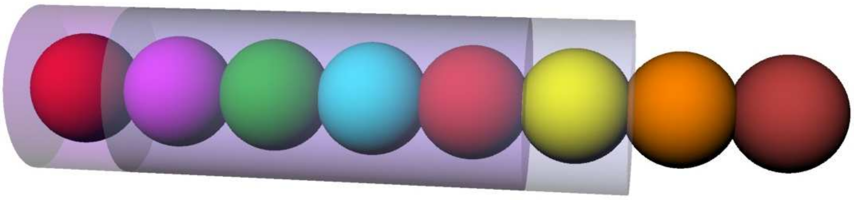


Figure 28: An example sequence with two sampling tubes placed on it (pink cylinders). Amino acid types are indicated by colors of the balls as follows: Arg - red, Cys - violet, His - green, Leu - blue, Lys - yellow, Glu - orange, Ile - brown.

Example 9. *Let us illustrate the process of histogram construction for a small example amino acid sequence shown in Figure 28. Let us have a template* $(\mathrm{Arg}, \mathrm{Lys})$ *and the tube size set to* 5*. The algorithm starts by placing the sampling tube at the beginning of the given protein sequence. The first sampling tube contains the following amino acids:* 2 arginines, 1 cysteine, 1 histidine and 1 leucine *therefore we increment a counter in the histogram associated with vector* $(2, 0)$*. Then in the second* *sampling tube, we get* 1 cysteine, 1 histidine, 1 leucine, 1 arginine and 1 lysine *so we increment a counter associated with vector* $(1,1)$*. We continue in this process until reaching the end of the protein sequence. In the end we normalize the histogram. An example histogram for a real protein 1AIS (chain B) is shown in Figure 29.*

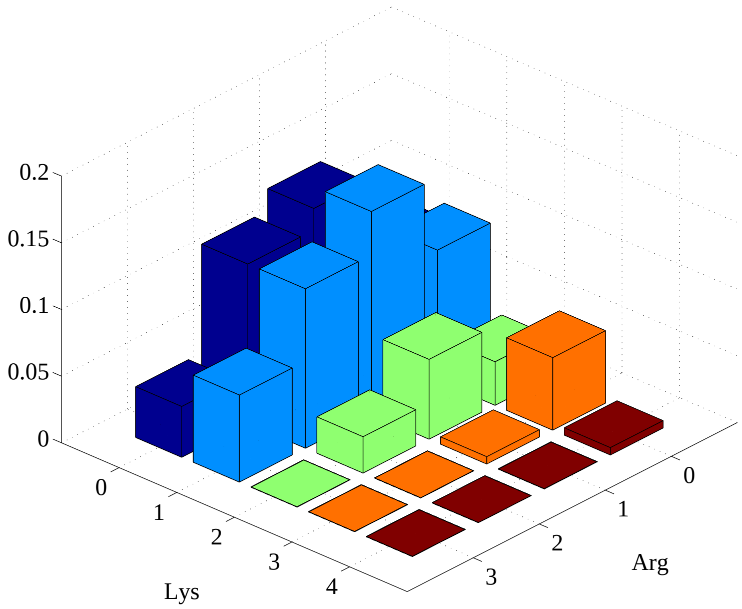


Figure 29: Example tube histogram with template (Arg, Lys) and tube size 10 constructed for protein 1AIS (chain B) from PD138.

We now explain how to use the constructed tube histograms for predictive classification. One possible approach would be to define a metric on the space of normalized histograms and then use either a nearest neighbour classifier or a nearest-centroid classifier. Since our preliminary experiments with these classifiers did not give us satisfying predictive accuracies, we decided to follow a different approach inspired by *propositionalization* – see Chapter 3.

The transformation method is quite straightforward. It looks at all histograms generated from the proteins in a training set and creates a numerical *attribute* for each vector of property occurrences which is non-zero at least in one of the histograms. After that an attribute vector is created for each training example using the collected attributes. The values of the entries of the attribute-vectors then correspond to heights of the bins in the respective histograms. After this transformation a random forest classifier is learned on the attribute-value representation. This random forest learning algorithm is then used for predictive classification. In practice, there is a need to select an optimal tube size. This can be done by creating several sets of histograms and their respective attribute-value representations corresponding to different sizes and then selecting the optimal parameters using an internal cross-validation procedure[1].

Note that tube histograms are able to represent patterns which would be hard to capture by conventional subsequence patterns. For example, the bin at coordinates *(2,1)* of a tube histogram with template *(Arg, Lys)* and tube size 5 represents the proportion of the following subsequence patterns in a protein's sequence: *(Arg, Arg, Lys, ?, ?)*, *(Arg, Arg, ?, Lys, ?)*, *(Arg, Arg, ?, ?, Lys)*, . . . , *(Lys, Arg, ?, ?, Arg)*, . . . , etc. In certain sense, the tube-histogram method can also be viewed as a generalization of protein-classification methods based on amino acid composition, because it concisely represents amino acid compositions in small windows (tubes).

12.2 CONSTRUCTION OF TEMPLATES

A question that we left unanswered so far in the description of our method is how to construct appropriate templates, which would allow us to accurately predict DNA-binding propensity. It is obvious that an *all-inclusive* strategy where the template would simply list all possible properties is infeasible. A template with n properties will generate training samples with a number of attributes d that is exponential in n. Furthermore, machine learning theory [28] indicates that the number of training samples needed to preserve accurate classification grows exponentially in d. In effect, the requested number of training samples grows doubly-exponentially with the size of the template. It is thus crucial that the template consists of only a small number of relevant properties. On the other hand, omitting some amino acids completely might be a problem as well. A possible solution is to use more templates of bounded size instead of one big template, because the number of attributes d grows only linearly with the number of templates.

But how to select the templates? One possibility could be to use templates with sets of amino acids believed to play an important role in the DNA-binding process according to literature. This could mean, for example, using the four charged amino acids - *Arg*, *Lys*, *Asp*, *Glu*, which are known to often interact with the negatively charged backbone as well as with the bases of the DNA [70, 59, 36] or other amino acids identified as important, e.g. the eight amino acids used in [92]. In this section we follow a different strategy. We develop an automated method for construction of templates. The basic idea of the method is to find templates which maximize *distance* between average histograms from the two classes (DNA-binding and non-DNA-binding proteins). Intuitively, such templates should allow us to construct classifiers with good discriminative ability. We construct the templates in a heuristic way using best-first search algorithm (Algorithm 3) to maximize the distance be-

[1] When evaluating the classifiers' performance using 10-fold cross-validation, we optimize the tube size parameter always on the nine training folds and then use it for the remaining testing fold, which is a standard way to obtain an unbiased estimate of the predictive performance of a classifier with tunable parameters.

tween the average histograms from the two classes. We use the Bhattacharyya distance [4] which is defined as

$$D_B(h_a, h_b) = -\ln\left(\sum_{x\in X}\sqrt{h_a(x)\cdot h_b(x)}\right),$$

where h_a and h_b are histograms and X is their support-set. Other types of *distances* could be used as well.

The following example shows that the templates cannot be constructed greedily. Although we do not directly prove hardness of the *template-search* problem here, the next example gives an intuition why the problem is probably hard.

Example 10. *We assume that we have histograms for DNA-binding proteins and non-DNA-binding proteins as shown in Table 17 and we want to find an optimal template with length 2. It can be easily verified that greedy search starting with an empty template would construct either the template* (Arg, Gly) *or* (Lys, Gly)*, but not the optimal template* (Arg, Lys).

	Lys		
Arg	**0.5**	**0**	0.5
	0	**0.5**	0.5
	0.5	0.5	

	Gly		
Arg	**0.4**	**0.1**	0.5
	0.2	**0.3**	0.5
	0.6	0.4	

	Gly		
Lys	**0.4**	**0.1**	0.5
	0.2	**0.3**	0.5
	0.6	0.4	

	Lys		
Arg	**0**	**0.5**	0.5
	0.5	**0**	0.5
	0.5	0.5	

	Gly		
Arg	**0.1**	**0.4**	0.5
	0.3	**0.2**	0.5
	0.4	0.6	

	Gly		
Lys	**0.1**	**0.4**	0.5
	0.3	**0.2**	0.5
	0.4	0.6	

Table 17: Histograms used as counter-examples for greedy search. The first row contains histograms for DNA-binding proteins, the second row contains histograms for non-DNA-binding proteins.

Let us elaborate on this in more detail. We start with the case of the search starting from the empty template. In the first step, the template (Gly) *is constructed because it maximizes distance between the histograms for the two classes. Both* (Arg) *and* (Lys) *would give rise to identical histograms for the two classes. In the next step,* Arg *or* Lys *is added to this template. However, the resulting template is clearly not optimal as can be checked by routine calculation of the Bhattacharyya distance which is finite for the discovered sub-optimal template but which would be infinite for the template* (Arg, Lys).

Similarly, if we wanted to construct an optimal template of length 1 *and if we started with the maximal template* (Arg, Lys, Gly) *and then tried to iteratively remove its elements while greedily maximizing the distance between the histograms of the two classes then we would end up with a sub-optimal template as before. In the first step,* Gly *would be removed and we would get the optimal template of size* 2 (Arg, Lys)*. However, in this case we want to construct a template of length* 1*. Therefore, in the next step we would create either the template* (Arg) *or* (Lys) *but not the optimal one* (Gly) *(note that the distances for templates* (Arg) *or* (Lys) *are* 0*).*

In order to avoid repeated construction of histograms from the whole datasets, we construct a histogram corresponding to the biggest possible template (containing all amino acid properties), then, during the best-first search, we construct histograms for the other templates by marginalising this maximal histogram.

While searching for a single template using best-first search is quite straightforward, searching for several templates is more complicated, because we need to find not only a set of templates making the distances between the average histograms as big as possible, but also these templates should be sufficiently diverse. Although, there are multiple ways to introduce diversity to the set of templates, we decided to use a heuristic approach. During the template search we penalise all candidate templates which are subsets of some templates already discovered in previous runs of the procedure *Best-FirstSearch(Templates)*. In order to direct the search early to the most promising regions of the search-space, we first initialize the set *Open* with all pairwise intersections of the already discovered templates. The intuition is that sub-templates which appear in more templates constitute some kind of *core*, which is shared by the most informative templates.

12.3 EXPERIMENTS

In this section we present experiments performed on real-life datasets of DNA-binding proteins – PD138 and non-DNA-binding proteins – NB110 described in Chapter 4. We compare the accuracy of our method with the accuracy of two other methods. Then we evaluate meaningfulness of the discovered templates from biological perspective. We also show here some proteins with the occurrence of the most informative template patterns.

We constructed histograms with automatically discovered templates (with maximum length 5) and three different sampling-tube sizes: 5, 10 and 15. We trained random forest classifiers selecting optimal sampling-tube size and an optimal number of trees for each fold by internal cross-validation. The estimated accuracy and area under ROC is shown in Table 18. As we can see, the accuracy of our method exceeds the accuracy obtained by the method used in [92].

We can see that the method which had higher accuracy than our tube-histogram method, is the method of Nimrod et al. [68]. This method detects clusters of evolutionarily conserved regions on the surface of proteins and trains a classifier using features like the electrostatic potential, cluster-

Algorithm 3 Template Search

```
function TemplateSearch()
Templates ← {}
for $i = 1$ to NumberOfTemplates do
    $i \leftarrow i + 1$
    Templates ← Templates ∪ BestFirstSearch(Templates)
end for

function BestFirstSearch(Templates, $\lambda$)
$E^+$ - set of positive examples (DNA-binding Proteins)
$E^-$ - set of negative examples (non-DNA-binding Proteins)
Open ← {()}
Open ← Open ∪ $\{t_i \cap t_j | t_i, t_j \in \text{Templates}\}$
Closed ← ∅
BestTemplate ← ()
Scores ← HeuristicScore(Open)
while Open ≠ ∅ do
    Template ← Remove best template from Open according to Scores
    if $(D_x(\text{Template}, E^+, E^-) > D_x(\text{BestTemplate}, E^+, E^-)) \wedge (\text{Template} \notin \text{Templates})$ then
        BestTemplate ← Template
    end if
    for T ∈ Expand(Template) do
        if T ∉ Closed then
            Closed ← Closed ∪ {T}
            Open ← Open ∪ {T}
            if $\exists T' \in \text{Templates} : T \subseteq T'$ then
                Score ← $\lambda \cdot$ HeuristicScore(T)
            else
                Score ← HeuristicScore(T)
            end if
            Scores ← Scores ∪ Score
        end if
    end for
end while
```

based amino acid conservation patterns, the secondary structure content of the patches and features of the whole protein, including all the features used by Szilágyi and Skolnick [92]. This is much more information than our tube-histogram method used. Clearly, their classifier is more accurate. However, when removing evolutionary information (or information about the secondary structure), their classifier's misclassification error increased by 3.5. Even without this information their classifier still used significantly more information than our method. Speaking about limited amount of information, we performed an additional experiment with our tube-histogram method, in

Method	Accuracy	AUC
Szilágyi et al. [92]	81.4	0.92
Nimrod et al. [68]	90.0	0.96
Tube histogram	86.3	0.94

Table 18: Accuracies and AUCs estimated by 10-fold cross-validation on PD138/NB110.

which we used a fixed template (Arg, Lys, Glu, Asp) in order to support the results about distributions of charged amino acids presented in Chapter 11. The predictive accuracy obtained in this experiment was 85.5%, which is close to the result obtained using automatic template search.

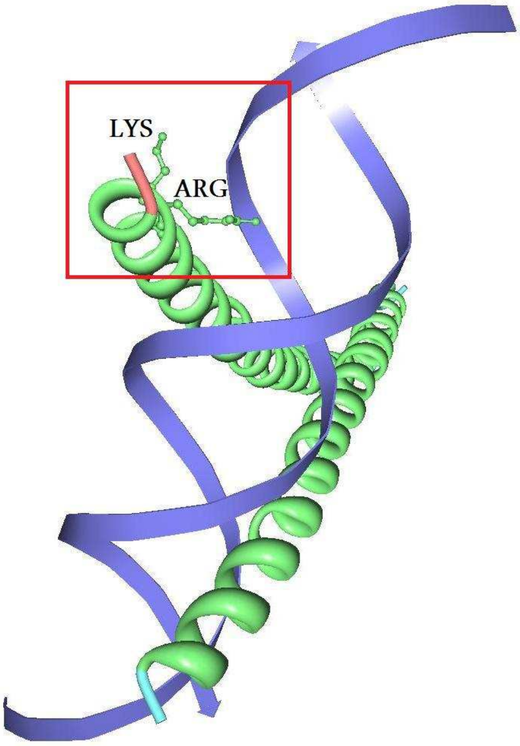


Figure 30: Protein 1DH3 containing the discovered pattern $(\mathrm{Arg}, \mathrm{Cys}, \mathrm{Lys}, \mathrm{Gly}, \mathrm{Ala}) = (1, 0, 1, 0, 0)$ shown using the protein viewer software [62]. Amino acids assumed by the pattern are indicated.

We already made experiments based only on primary structure of proteins in Chapter 6. The method based on polynomial relational features achieved as high accuracies as the tube-histogram method. The specialized tube-histogram method is more scalable than the method based on polynomial relational features.

The four most informative (according to χ^2 criterion) automatically selected templates are: *(Arg, Cys, Lys, Gly, Ala)*, *(Arg, Cys, Lys, Gly, Asp)*, *(Arg, Cys, Lys, Gly, Glu)*, *(Arg, Cys, Lys, Gly, Leu)*. It is noteworthy that each charged amino acid is contained in at least one of these templates. This confirms results of previous studies ([70, 59, 36]), where charged amino acids were identified as critical for DNA-binding. Furthermore, it is interesting that unlike the negatively charged amino acids the positively charged amino acids are present in all discovered templates.

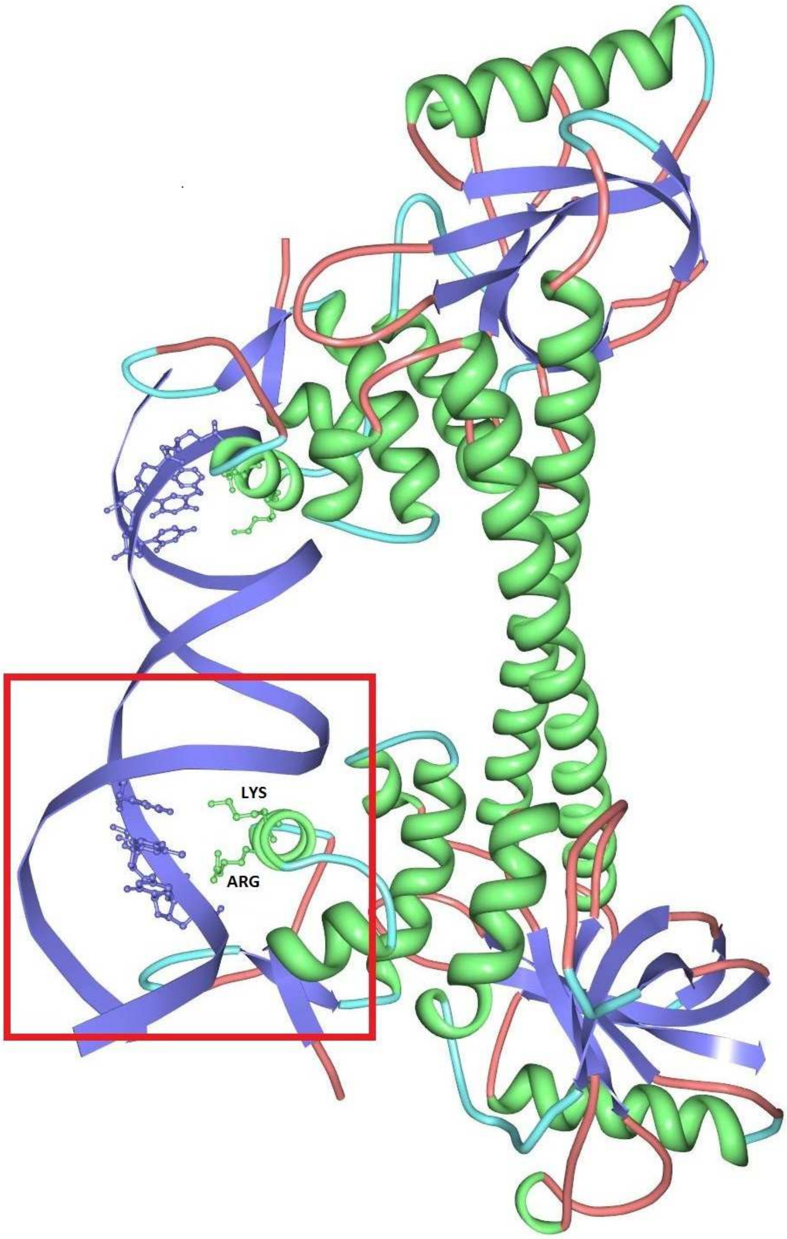


Figure 31: Protein 1R8E containing the discovered pattern $(\mathrm{Arg}, \mathrm{Cys}, \mathrm{Lys}, \mathrm{Gly}, \mathrm{Asp}) = (1, 0, 1, 0, 0)$ shown using the protein viewer software [62]. Amino acids assumed by the pattern are indicated.

In addition to improved accuracy, our method provides us with interpretable patterns involving distributions of selected amino acids in primary structures. The most informative *patterns* according to the χ^2 criterion assumed presence of one arginine and one lysine in a tube of size 5. Given a protein, each pattern captures the fraction of sampling tubes, which contain the specified numbers of amino acids of given types. Arginine and lysine are known to often interact with the negatively charged backbone as well as with the bases [70, 59, 36]. We show two example occurrences of this pattern in DNA-binding proteins with the highlighted amino acids in Figure 30 and Figure 31.

12.4 CONCLUSIONS

We presented a novel tube-histogram method for the task of prediction of DNA-binding propensity of proteins. We have shown that this method is capable to identify important amino acids for predicting DNA-binding propensity. The tube-histogram method is able to capture joint probabilities of specified amino acids occurring within certain distances in amino acid sequences from each other. We validated this method in prediction experiments using only proteins' primary structure, achieving favourable accuracies. We compared the method with two other state-of-the-art methods, from which one was less accurate than our method. The second method by Nimrod et al. [68] outperformed our method, but it required much more information to classify DNA-binding proteins accurately.

13 THE BALL-HISTOGRAM METHOD

The tube-histogram method described in the previous chapter is both fast and accurate despite the fact that it uses only information about the primary structure of proteins. Primary structure refers to the sequence of amino acids in the protein chains. We cannot overlook the fact that a protein is not only a linear sequence of amino acids, but it is a complex molecule with intricate 3D shape. Proteins fold into spatial structures that reflect their biological roles. We can expect that we could obtain better accuracies for predicting DNA-binding function, when taking proteins' spatial structure into account. Therefore, we decided to generalize the tube-histogram method to 3D space. The tube-histogram method uses a sampling tube to gather information about the amino acid distribution in a protein sequence. What would be a generalization of the sampling tube (which is essentially 1D) in 3D space? Naturally, we thought about a sampling ball which as it turned out has many convenient properties, for instance it is possible to use it for sampling without having to consider its rotation. If we chose a different shape, we would have to consider its rotation in order to ensure invariance of the result to the orientation of the protein. This would make the sampling more difficult.

In this section we describe our novel method for predictive classification of DNA-binding propensity of proteins using so-called *ball histograms*. We propose the following approach based on the tube-histogram method. Similarly as the tube-histogram method, this method also consists of four main steps. First, *templates* are found, which determine amino acids whose distributions should be captured by *ball histograms*. In the second step *ball histograms* are constructed for all proteins in a training set. Third, a transformation method is used to convert these histograms to a form usable by standard machine learning algorithms. Finally, a random forest classifier [6] is learned on this transformed dataset and then it is used for classification. The reason why we chose the random forest learning algorithm is that it is known to be able to cope with large numbers of attributes such as in our case of *ball histograms* [9].

This chapter is organized as follows. We describe the ball-histogram method in Section 13.1. We subject the method to experimental evaluation in the domain of DNA-binding proteins in Section 13.2. In Section 13.3 we conclude this chapter.

13.1 METHOD

Here, we describe the ball-histogram method. We first define a few auxiliary terms. Similarly as in the case of the tube-histogram method, a *template* is a list of some Boolean amino acid properties. A property may, for example, refer to the charge of the amino acid (e.g. *Positive*), but it may also directly stipulate the amino acid type (e.g. *arginine*). An example of a template is $(\mathrm{Arg}, \mathrm{Lys}, \mathrm{Polar})$ or $(\mathrm{Positive}, \mathrm{Negative}, \mathrm{Neutral})$. A *bounding sphere* of a protein structure is a sphere with center located in the geometric center of the protein structure and with radius equal to the distance from the center to the farthest amino acid of the protein plus the radius of the *sampling ball* which is a parameter of the method. We say that an amino acid *falls* within a sampling ball if the alpha-carbon of that amino acid is contained in the sampling ball in the geometric sense.

Given a protein structure, a template $\tau = (f_1, \ldots, f_k)$, a sampling-ball radius R and a bounding sphere S, a *ball histogram* is defined as:

$$H_\tau(t_1, \ldots, t_k) = \frac{\int\int\int_{(x,y,z)\in S} I_{T,R}(x, y, z, t_1, \ldots, t_k) dxdydz}{\sum_{(t_1', \ldots, t_k')} \int\int\int_{(x,y,z)\in S} I_{T,R}(x, y, z, t_1', \ldots, t_k') dxdydz}, \quad (2)$$

where $I_{T,R}(x, y, z, t_1, \ldots, t_k)$ is an indicator function which we will define in turn. The expression $\sum_{(t_1', \ldots, t_k')} \int\int\int_{(x,y,z)\in S} I_{T,R}(x, y, z, t_1', \ldots, t_k') dxdydz$ is meant as a normalization factor - it ensures that $\sum_{(t_1, \ldots, t_k)} H_T(t_1, \ldots, t_k) = 1$. In order to define the indicator function $I_{T,R}$ we first need to define an auxiliary indicator function $I_{T,R}'(x, y, z, t_1, \ldots, t_k)$:

$$I_{T,R}'(x, y, z, t_1, \ldots, t_k) = \begin{cases} 1 & \text{if there are exactly } t_i \text{ amino acids complying with property } f_i\ (1 \leqslant i \leqslant k) \text{ in the sampling ball with center } x, y, z \text{ and radius } R, \\ 0 & \text{otherwise.} \end{cases}$$

Notice that $I_{T,R}'(x, y, z, 0, \ldots, 0)$ does not make any distinction between a sampling ball that contains no amino acid at all and a sampling ball that contains some amino acids of which none complies with the parameters in the template T. Therefore if we used $I_{T,R}'$ in place of $I_{T,R}$ the histograms would be affected by the amount of empty space in the bounding spheres. Thus, for example, there might be a big difference between histograms of otherwise similar proteins where one would be oblong and the other one would be more curved. In order to get rid of this unwanted dependence of the indicator function $I_{T,R}$ on proportion of empty space in sampling spheres we define $I_{T,R}$ in such a way that it ignores the empty space. For $(t_1, \ldots, t_k) \neq 0$ we set:

$$I_{T,R}(x, y, z, t_1, \ldots, t_k) = I_{T,R}'(x, y, z, t_1, \ldots, t_k).$$

In the cases when $(t_1, \ldots, t_k) = 0$ we set $I_{T,R}(x, y, z, t_1, \ldots, t_k) = 1$ if and only if $I_{T,R}'(x, y, z, t_1, \ldots, t_k) = 1$ and if the sampling ball with radius R at (x, y, z) contains at least one amino acid.

Ball histograms capture the joint probability that a randomly picked *sampling ball* (See Figure 32) containing at least one amino acid will contain exactly t_1 amino acids complying with property f_1, t_2 amino acids complying with property f_2 etc. They are invariant to rotation and translation of protein structures which is an important property for classification. Also note that the histograms would not change if we increased the size of the bounding sphere.

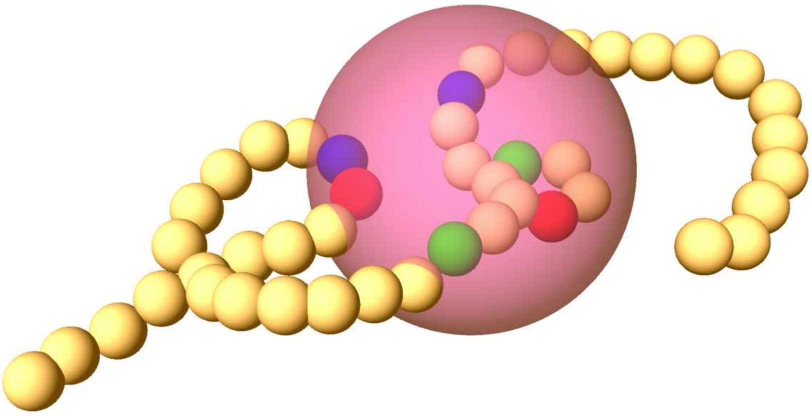


Figure 32: Amino acids are shown as small balls in sequence forming an amino acid chain. A *sampling ball* is shown in violet. Some of the amino acids which comply with properties of an example template are highlighted inside the *sampling ball* area. They have different colors according to their type.

The indicator function $I_{T,R}$ makes crisp distinction between the case where an amino acid falls within a sampling ball on one hand, and the case where it falls out of it, on the other hand. This could be changed towards capturing a more complex case by replacing the value 1 by the fraction of the amino acid that falls within the sampling ball.

Computing the integral in (2) precisely is infeasible therefore we decided to use a Monte-Carlo method. The method starts by finding the bounding sphere. First, the geometric center C of all amino acids of a given protein P is computed (each amino acid is represented by coordinates of its alpha-carbon). The radius R_S of the sampling sphere for the protein structure P is then computed as:

$$R_S = \max_{Res \in P}(distance(Res, C)) + R,$$

where R is a given sampling-ball radius. After that the method collects a predefined number of samples from the bounding sphere. For each sampling ball the algorithm counts the number of amino acids in it, which comply with the particular properties contained in a given template and increments a corresponding bin in the histogram. In the end, the histogram is normalized.

Example 11. *Let us illustrate the process of histogram construction. Consider the template* (Arg, Lys) *and assume we already have a bounding sphere. The algorithm starts by placing a sampling ball randomly inside the bounding sphere. Assume the first such sampling ball contained the following amino acids:* 2 arginines and 1 leucine *therefore we increment (by 1) the histogram's bin associated with vector* $(2,0)$. *Then, in the second sampling ball, we get* 1 histidine and 1 aspartic acid, *so we increment the bin associated with vector* $(0,0)$. *We continue in this process until we have gathered a sufficient number of samples. In the end we normalize the histogram. Examples of such histograms are shown in Figure 33 and 34.*

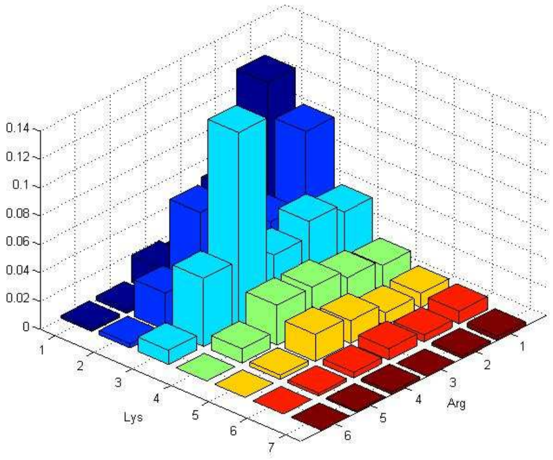


Figure 33: Example ball histogram with template $(\mathrm{Arg}, \mathrm{Lys})$ and sampling-ball radius $R = 12$ Åconstructed for proteins 1A31 from PD138.

Templates can be constructed automatically using the same template construction method as in the case of tube-histogram method in Section 12.2.

Then, the ball-histogram method can be used for predictive classification similarly as the tube-histogram method. First, the method creates a numerical *attribute* for each bin which is non-zero at least in one of the histograms. The values of the attribute-vectors correspond to heights of the bins in the respective histograms. After this transformation a classifier is learned on the attribute-value representation. This classifier is used for the predictive classification. It is necessary to estimate the optimal sampling-ball radius. This can be done by selecting the optimal parameters using an internal cross-validation procedure.

13.2 EXPERIMENTS

In this section we present experiments performed on DNA-binding and non-DNA-binding proteins described in Chapter 4. We constructed histograms with automatically discovered templates and three different sampling-ball radii: 4, 8 and 12 Å. We trained random forest classifiers selecting optimal sampling-ball radii and optimal numbers of templates (1, 3, 5 or 7 templates) for each fold by internal cross-validation. The estimated AUCs (area under

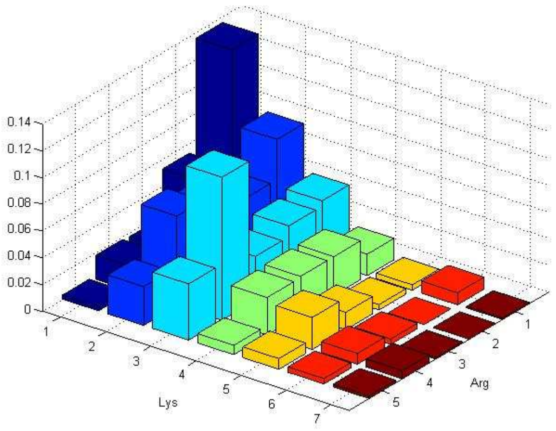


Figure 34: Example ball histogram with template (Arg, Lys) and sampling-ball radius $R = 12$ Åconstructed for proteins 1A3Q from PD138.

curve) are shown in Table 19 and the estimated accuracies are shown in Table 20. We performed two sets of experiments. In the first experiment we tested the ball-histogram method (see Ball histograms in Table 19 and Table 20). In the second experiment we used only the coarse-grained features of Szilágyi and Skolnick [92]. For both of these experiments we trained two types of classifiers: random forests [6] and linear support vector machines [8] in order to determine the extent to which the choice of classifiers matters. In addition, we learnt a logistic regression classifier [33] for the experiment with the coarse-grained features from Szilágyi and Skolnick [92] since it was the classifier used originally by the authors. (We also tried to learn logistic regression classifiers for the ball-histogram method but logistic regression turned out to be too slow with the high number of attributes generated by the ball-histogram method.) We can see from the experimental results that the choice of the classifier has low influence on the performance of the Szilágyi's and Skolnick's method whereas it has slightly bigger impact on the ball-histogram method. A possible explanation is that random forest classifier is able to cope with large numbers of attributes [9] such as in our ball-histogram method.

We report both AUC and accuracy for the two combinations of datasets (PD138/NB110 and PD138/NB843). In case of datasets PD138 and NB110, the ball-histogram method achieved the best results in terms of accuracy and AUC. The best results for ball-histogram methods were obtained by random forest learning algorithm. In case of datasets PD138 and NB843 accuracy is not very meaningful measure of classification quality because the dataset is highly class-skewed. However, if we have a look at the AUC value, we can see that again the ball-histogram method performs best.

	Classifier	**PD138/NB110**	**PD138/NB843**
Ball histograms	Random forest	**0.94 ± 0.05**	**0.87 ± 0.04**
	SVM	0.92 ± 0.04	0.83 ± 0.03
Szilágyi et al.	Log. regression	0.92 ± 0.04	0.84 ± 0.04
	Random forest	0.90 ± 0.05	0.82 ± 0.04
	SVM	0.92 ± 0.05	0.83 ± 0.05

Table 19: AUCs estimated by 10-fold cross-validation.

	Classifier	**PD138/NB110**	**PD138/NB843**
Ball histograms	Random forest	**0.87 ± 0.08**	**0.88 ± 0.01**
	SVM	0.84 ± 0.07	0.87 ± 0.01
Szilágyi et al.	Log. regression	0.81 ± 0.05	0.87 ± 0.01
	Random forest	0.82 ± 0.07	0.87 ± 0.02
	SVM	0.81 ± 0.05	0.87 ± 0.01

Table 20: Accuracies estimated by 10-fold cross-validation.

In order to see whether the ball-histogram method, which uses only structural information, could come close to the results of methods which exploit also information about the evolutionary conservation of regions on protein surfaces, we compared our results with the results of Nimrod et al. [68]. The AUC 0.96 and accuracy 0.90 reported by Nimrod et al. for the datasets PD138 and NB110 differs only slightly (0.02, 0.03 respectively) from our best results. The AUC 0.90 obtained for the datasets PD138 and NB843 differs by 0.03 from our best results. These results are encouraging given how important evolutionary information turned out to be according to experiments of Nimrod et al. When removing evolutionary information, their classifier's misclassification error increased by 0.035. Even without this information their classifier used significantly more information than our method (e.g. secondary structure information). It is important to continue improving methods that do not exploit evolutionary information. Such methods are valuable mainly due to their ability to predict DNA-binding propensity for engineered proteins for which evolutionary information is not available. Engineered proteins are highly significant for example in emerging gene-therapy technologies [10].

In addition to improved accuracy, we try to interpret discovered features in protein structures. The three most frequent automatically selected templates are: *(Arg, Cys, Gly)*, *(Cys, Gly, Positive)*, *(Cys, Polar, Positive)*. Recall that positively charged amino acids are critical for DNA-binding function ([70, 59, 36]). This is probably the reason why the property *Positive* is contained in two out of three most informative templates. In the third one we have the explicitly listed positively charged amino acid - *arginine*. The remaining amino acid properties listed in the three most informative templates also fit well with the results of Sathyapriya et al. [81], who studied protein-DNA interactions through structure network analysis. According to their results the polar, non-negative amino acids have high DNA-binding propensity which supports meaningfulness of the template *(Cys, Polar, Positive)*. Furthermore, they also show that *cysteine* is one of the amino acids with the lowest DNA-binding propensity. This again fits well with the discovered templates, since *cysteine* appears in all of them.

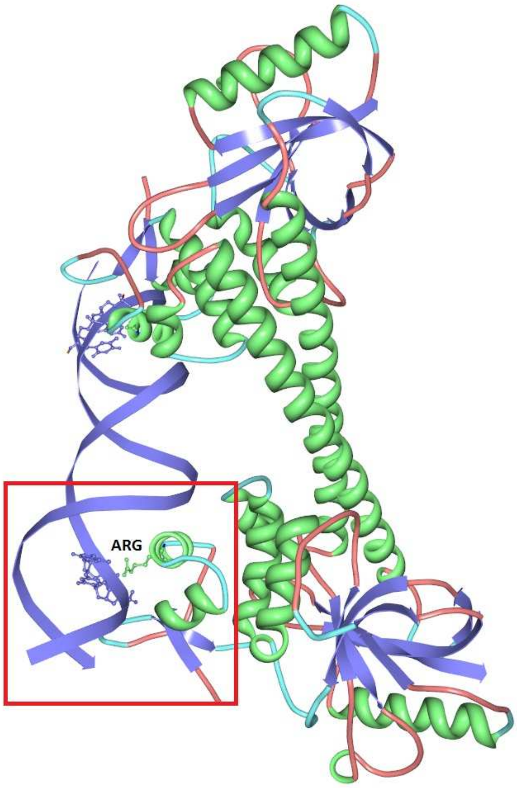


Figure 35: Protein 1R8E containing the discovered pattern $(\mathrm{Arg}, \mathrm{Cys}, \mathrm{Gly}) = (1, 0, 0)$ shown using the protein viewer software [62]. Amino acid assumed by the pattern is indicated.

Each template gives rise to a set of features which correspond to individual bins in the respective multi-dimensional histogram. It is therefore interesting to evaluate also the particular features from the point of view of predictive information which they carry. We evaluated the features corresponding to the automatically selected templates using χ^2-criterion. The most informative *feature* according to the χ^2-criterion assumed presence of one arginine, no cysteine and no glycine in a ball with radius 8 Å. Given a protein structure, each feature captures the fraction of sampling balls, which contain the specified numbers of amino acids complying with given properties. The next two most informative features assumed presence of two, respectively three positively charged amino acids, no cysteine and no glycine. All the three most informative features correspond to the above-mentioned observations of Sathyapriya et al. We show an example occurrence of the first feature in a DNA-binding protein with the highlighted amino acid in Figure 35. This figure may look familiar. It depicts the same protein as Figure 31 from Section 12.3, where we discuss the results of the tube-histogram method. The same protein region was detected by the two different patterns – one assuming the presence of 1 arginine and 1 lysine and no cysteine, no glycine and no aspartic acid in sequence, and the other one assuming the presence of 1 arginine, no cysteine and no glycine in a ball with radius 8 Å. We can see that these two patterns overlap.

13.3 CONCLUSIONS

In this chapter, we extended the tube-histogram method from 1D to 3D. We improved on state-of-the-art accuracies in the prediction of DNA-binding propensity of proteins from structure data through an innovative *ball-histogram method*. The method is based on systematic exploration of the distribution of automatically-selected amino acid properties in protein structures, yielding a predictive model based on features amenable to direct interpretation.

14 THE BALL-HISTOGRAM METHOD FOR REGRESSION

The ball-histogram method as presented in the previous chapter is suitable only for predictive classification tasks. This is because the automatic template search assumes the learning examples (proteins) to be class-labelled. The rest of the ball-histogram method does not depend on availability of class-labels. However, many problems require prediction of continuous variables, i.e. regression problems. That is why we upgraded the *ball-histogram* method for predictive classification to a method for regression. The main contribution of this chapter is the introduction of a method for automatic *template search* for regression problems.

This chapter is organized as follows. In Section 14.1 we describe how to upgrade the ball-histogram method for classification to regression problems. Next, we present an application of this method for antimicrobial activity prediction in Section 14.2. We compare the results of the ball-histogram method for regression to a state-of-the-art method for prediction of antimicrobial activity [97]. In Section 13.3 we conclude this chapter.

14.1 AUTOMATIC TEMPLATE SEARCH FOR REGRESSION

Here, we present a ball-histogram method adapted for regression. The main problem that we have to solve is to develop a method for construction of high-quality templates. As we have already indicated in Section 12.2, it is crucial that the template consists of only a small number of relevant properties. Omitting some amino acids or properties completely might be a problem as well. The solution is to use more templates of bounded size instead of one big template.

Here, we advance the strategy by developing an automated method for template construction for regression. The basic idea of the method is to construct templates which give rise to histograms with bins highly correlated with a target variable. Intuitively, such templates should allow us to construct regression models with good predictive accuracies. We construct the templates in a heuristic way using a variant of best-first search algorithm (Algorithm 4 - Template search).

First, we explain how to construct a single template and after that we explain how to construct a set of templates. When searching for a single best template we proceed as follows. We create an empty template and we search the space of all possible templates using a variant of best-first search algorithm. The heuristic, guiding the search, is the maximum correlation between the target variable and the histogram bins *(best-bin correlation)* measured using Pearson's correlation coefficient [22]. Specifically, given a template and a set of examples, we create histograms for all the examples using the template and compute correlations between the values of each bin (e.g. [arg, lys] = [1, 0]) and the target variable and select the maximal value. This is a heuristic measure of the template's quality. We construct a histogram corresponding to the largest possible template – in order to avoid repeated construction of histograms from the whole datasets, then we construct histograms for the other templates by marginalising this largest histogram.

Algorithm 4 Template search

```
// E - set of examples (peptides), A - activities
function TemplateSearch(E, A)
// T - templates
$T \leftarrow \{\}$
for $i = 1$ to NumberOfTemplates do
    $i \leftarrow i + 1$
    $t \leftarrow BFS(T, E, A)$
    $T \leftarrow T \cup \{t\}$
    /* Performs regression using most correlated bin and subtracts the estimates from the original values */
    $A \leftarrow BestBinResiduals(A, t, E)$
end for

// Best First Search
function BFS(Templates, Examples, Activities)
$Open \leftarrow \{()\}$
/* Initialization with all pairwise intersections of already discovered templates */
$Open \leftarrow Open \cup \{t_i \cap t_j | t_i, t_j \in Templates\}$
$Closed \leftarrow \emptyset$
$BestTemplate \leftarrow ()$
$BestScore \leftarrow -\infty$
$Scores \leftarrow HeuristicScore(Open)$
while $Open \neq \emptyset$ do
    $(Template, Score) \leftarrow$ Remove best template from Open according to Scores
    if $(Score > BestScore)$ then
        $BestTemplate \leftarrow Template$
    end if
    for $T \in Expand(Template)$ do
        if $T \notin Closed$ then
            $Closed \leftarrow Closed \cup \{T\}$
            $Open \leftarrow Open \cup \{T\}$
            $NewScore \leftarrow HeuristicScore(T, E, Activities)$
            $Scores \leftarrow Scores \cup NewScore$
        end if
    end for
end while
```

While searching for a single template using best-first search is quite straightforward, searching for several templates is more complicated. Once we have a method for constructing one template, we can use it as a basis of a method for constructing a set of templates. The biggest problem when searching for a set of templates is avoiding construction of redundant templates - i.e. templates very similar to templates already discovered. This is because we need to find not only a set of templates giving rise to histograms with bins highly correlated with the target variable, but also these templates should be sufficiently diverse. We decided to follow a fast heuristic approach. After a template is constructed, we perform univariate regression using the most correlated bin and subtract the estimates from the original values of the target variable. Then we search for the next template maximizing best-bin correlation using the modified values of the target variable. Intuitively, the new template should not be too similar to the previous one. In order to direct the search early to the most promising regions of the search-space, we first initialize the set *Open* with all pairwise intersections of the already discovered templates. Sub-templates which appear in more templates constitute a kind of a *core* shared by the most informative templates. This helps the algorithm visit the most promising parts of the search space.

14.2 EXPERIMENTS

Prediction of antimicrobial activity of peptides is a problem that involves prediction of continuous variables, i.e. regression. Here, we apply the ball-histogram method with automatic template search for regression to this problem. Our approach for prediction of antimicrobial activity of peptides exploits structure prediction methods and ball-histogram method in conjunction with state-of-the-art attribute-value learning algorithms. Our method can be imagined as proceeding in the following way. It starts with AMP sequences, for which we obtain spatial models using LOMETS structure prediction software [105]. Then we construct a set of templates using the method described in Section 14.1, which we use to create ball histograms for all peptides. We find a pre-specified number of templates using a given sampling ball radius and then we construct the corresponding ball histograms for all peptides from the dataset. This gives us *number_of_templates* × *number_of_peptides* histograms. When performing cross-validation, the set of histograms is created separately for each train-test split corresponding to iterations of the cross-validation procedure. Then, we save these histograms into a WEKA file [103], in which every bin occurring in a histogram gives rise to an attribute and each peptide corresponds to a learning example. We also add additional information about dipole moment, proportions of amino acid types and their spatial asymmetries [92]. In the last step, we use implementation of SVM with RBF kernel present in the WEKA open-source machine learning software to train a regression model using the generated WEKA files. Parameters of the regression model are tuned using internal cross-validation.

We describe experiments performed on two datasets of antimicrobial peptides described in Chapter 4: CAMEL and RANDOM. In the experiments we constructed histograms with automatically discovered templates and sampling-ball radius 10Å separately for each train-test split (selecting the templates always only on training data). We trained support vector machine [8] regression models with RBF kernel selecting optimal C (complexity constant) and $gamma$ (kernel width parameter) for each fold by internal cross-validation. The estimated results are shown in Table 21. We compared the results of our ball-histogram method for regression with the results reported by Torrent et al. which is a state-of-the-art method. Only cross-validated coefficients of determination (q^2) were given by Torrent et al. [97]. In addition, we also report correlation coefficient (q) and root-mean-square error ($RMSE$) for our regression method. On both datasets we improved upon the results of Torrent et al. in terms of coefficients of determination. Coefficient of determination can be regarded as the proportion of variability in a dataset that is accounted for by the statistical model.

	Torrent et al. [97]	**Ball Histogram for Regression**		
	q^2	q^2	q	$RMSE$
CAMEL	0.65	**0.69**	0.85	1.14
RANDOM	0.72	**0.73**	0.87	1.28

Table 21: Experimental results obtained by cross-validation, where q^2 is coefficient of determination, q is correlation coefficient and $RMSE$ is root-mean-square error.

In addition to improved prediction quality, our method provides us with interpretable features involving distributions of selected amino acids in peptide structures. We used the following methodology. For each split of the datasets (CAMEL and RANDOM) induced by 10-fold cross-validation we recorded the automatically selected templates. Then we chose templates which appeared most often among the folds for each dataset. There were two templates which appeared in all of the ten folds in case of the dataset CAMEL: [ASN, ILE] and [LEU, positively_charged_aa]. In case of the dataset RANDOM the best template [GLY, TRP] appeared in 8 from 10 folds. In order to get a global view on the differences between distributions of selected amino acids in templates in peptides with high antimicrobial activity and peptides with low antimicrobial activity (we discretized the antimicrobial activity attribute, so that we could split the datasets into two classes according to their activity (low/high)), we computed the average ball histograms for these two classes of peptides. The histogram obtained by subtracting the average ball histogram for low-activity peptides from the average ball histogram for high-activity peptides is shown in Figures 36 and 37. We can notice a remarkable difference of distributions of the selected amino acid pairs between low-

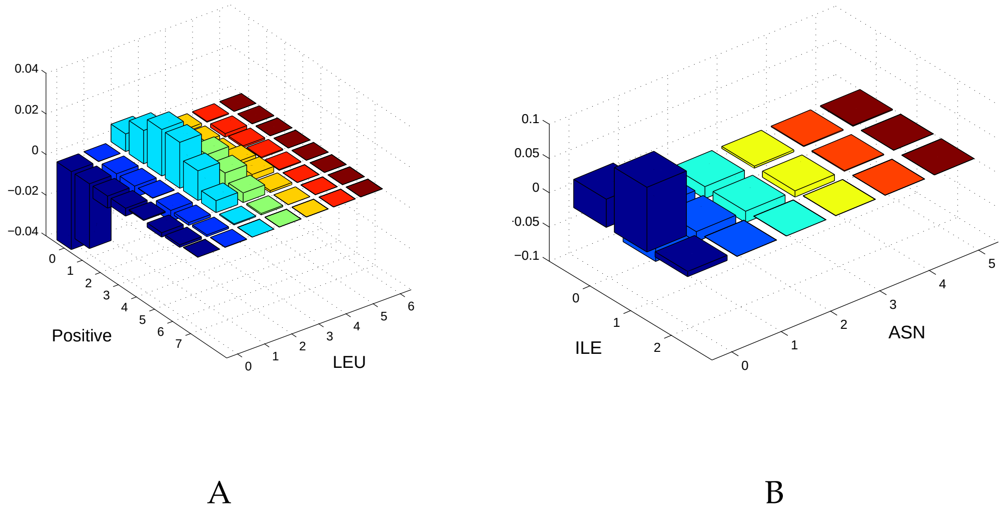


Figure 36: The histogram obtained by subtracting the average ball histogram for low-activity peptides from the average ball histogram for high-activity peptides for the dataset CAMEL (A - template [LEU, positively_charged_aa], B - template [ASN, ILE]).

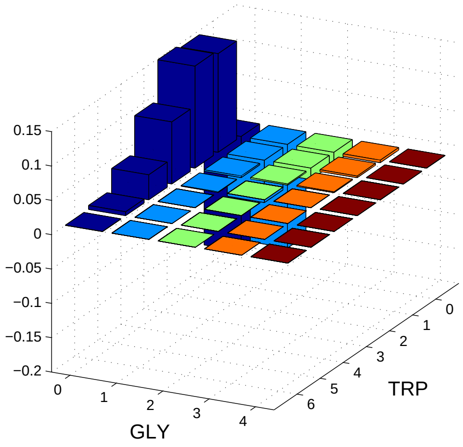


Figure 37: The histogram obtained by subtracting the average ball histogram for low-activity peptides from the average ball histogram for high-activity peptides for the dataset RANDOM (template [GLY, TRP]).

and high-activity peptides. As for the template [LEU, positively_charged_aa] found for the dataset CAMEL, the positively charged amino acids are known to correlate with antimicrobial activity and the presence of leucine can be explained by the fact that the dataset contains mostly leucine-rich peptides. In case of the dataset RANDOM the template [GLY, TRP] was automatically selected most probably, because the peptides of this dataset are composed mostly of TRP and some other amino acids. Moreover, glycine is known to have one of the lowest helix propensities [71] which together with the fact that helix is a frequent formation of amino acids of high-activity peptides, could explain negative bins for non-zero counts of glycine in Figure 37.

14.3 CONCLUSIONS

We upgraded the ball-histogram method presented in Chapter 13 to a method for regression and applied it to antimicrobial activity prediction of peptides. There are two main differences between the work presented in Chapter 13 and the work presented here. First, the problem that we tackled in Chapter 13 dealt with classification, whereas here we built a regression model. Second, here only primary structures of peptides are available (therefore we had to rely on structure prediction), whereas we could use spatial structures obtained by X-ray crystallography in our previous study with DNA-binding proteins. We have shown that our ball-histogram method for regression improves on a state-of-the-art approach to antimicrobial activity prediction in terms of coefficient of determination.

15 THE BALL-HISTOGRAM METHOD WITH POLYNOMIAL FEATURES

As we have already seen in the experiments using relational learning with polynomials in Chapter 6, physicochemical properties of proteins' regions can be important for prediction of proteins' function. These properties are often represented as continuous quantities, for instance Van der Waals volume, hydropathy index or isoelectric point. In this chapter, we are interested in how continuous variables could be used in the ball-histogram framework. The discrete ball-histogram method presented in Chapter 13 is not suitable for work with continuous attributes. Therefore, we adapt the idea of multivariate polynomial aggregation features described in Chapter 6 for the ball-histogram method. We show that with the approach based on multivariate polynomial aggregation we are able to capture characteristics (e.g. variance, covariance, mean, etc.) of distributions of continuous physicochemical properties of proteins' regions. This enables the ball-histogram method to achieve even higher predictive accuracies than those reported in the previous chapters.

This chapter is organized as follows. We introduce so-called *polynomial aggregation features* in Section 15.1 and show how they can be used in a ball histogram-based approach to predictive classification in Section 15.2. We evaluate the ball-histogram method with polynomial features for prediction of DNA-binding propensity and also discuss the results in Section 15.3. In Section 13.3 we conclude this chapter.

15.1 MULTIVARIATE POLYNOMIAL AGGREGATION FEATURES

A drawback of the original ball-histogram method described in Chapter 13 is that it is ill-suited for work with continuous variables. It can model the *distributions* of certain amino acids. For example, it is able to construct an empirical estimate of the probability $P(\mathrm{Arg} = i, \mathrm{Lys} = j)$ *that there are exactly* i *arginines and* j *lysines in a ball randomly sampled from a given protein structure* (which can be regarded as the desired *model of the distributions of amino acids* for our predictive purposes). However, if we tried to model distributions of e.g. *hydropathy* and *volume* of amino acids in a given protein structure in the very same way, we would face serious difficulties stemming from combinatorial explosion of the number of histograms' bins - attributes as the next example indicates.

Example 12. *Let us have a protein structure* P *and a sampling-ball radius* R *such that any ball can contain at most 6 amino acids. Let us have a template* $\tau = [\mathrm{Arg}, \mathrm{Lys}]$. *Then there is less than* $7^2 = 49$ *non-zero bins in the respective histogram for values* $[\mathrm{Arg}, \mathrm{Lys}] = [0, 0]$, $[\mathrm{Arg}, \mathrm{Lys}] = [1, 0]$, $\dots$, $[\mathrm{Arg}, \mathrm{Lys}] = [6, 0]$, $\dots$, $[\mathrm{Arg}, \mathrm{Lys}] = [0, 6]$. *Let us now have a template* $\tau_2 = [\mathrm{Hydropathy}, \mathrm{Volume}]$ *and note that both* *hydropathy and volume are properties of amino acids which acquire real values. In theory, we can construct a protein structure such that the number of bins will be at least* $\begin{pmatrix} 20 \\ 6 \end{pmatrix} = 38760$ *bins. This is certainly an impractically high number. Moreover, if we followed the transformation-based approach of the basic ball-histogram method which represents each bin as a real-valued attribute, we would be effectively discarding much of the information about which bins are close to each other.*

The above example illustrates the problems that would arise if we tried to use the original ball-histogram method for continuous properties. A possible approach to cope with these problems would be to use discretization of the values. However, discretization is known to perform poorly when the number of dimensions of the problem increases [12] which may very often be the case with ball histograms. Instead of relying on discretization, we use *multivariate polynomial aggregation* - a similar strategy to what we have introduced in Chapter 6.

Here, we introduce *multivariate polynomial aggregation features*. We start by defining *monomial* and *polynomial features* for sampling balls and then use them to define values of monomial and polynomial features on protein structures.

A *monomial feature* M is a pair $(\tau, (d_1, \ldots, d_k))$ where τ is a template with k properties and $d_1, \ldots, d_k \in \mathbf{N}$. *Degree* of M is $deg(M) = \sum_{i=1}^{k} d_i$. Given a sampling ball B placed on a protein structure P, we define the *value* of a monomial feature $M = (\tau, (d_1, \ldots, d_k))$ as $M(B) = \tau_1^{d_1} \cdot \tau_2^{d_2} \cdot \cdots \cdot \tau_k^{d_k}$ where τ_i is the average value of the i-th property of template τ averaged over the amino acids contained in the sampling ball B. We use the convention that $0^0 = 1$. We use a more convenient notation for monomial features motivated by this definition of *value*:

$$(\tau = (\tau_1, \ldots, \tau_k), (d_1, \ldots, d_k)) \equiv^{def} \tau_1^{d_1} \cdot \tau_2^{d_2} \cdot \cdots \cdot \tau_k^{d_k}$$

Example 13. *Let us have a template*

$$\tau = [hydropathy, volume],$$

a monomial feature

$$M = hydropathy \cdot volume^2$$

and a sampling ball containing two leucines ($hydropathy = 3.8$, $volume = 124$*) and one arginine (*$hydropathy = -4.5$, $volume = 148$*). Then*

$$M(B) = \frac{2 \cdot 3.8 - 4.5}{3} \cdot \left(\frac{2 \cdot 124 + 148}{3}\right)^2 \approx 1.8 \cdot 10^4$$

A *multivariate polynomial feature* is an expression of the form

$$N = \alpha_1 M_1 + \alpha_2 M_2 + \cdots + \alpha_k M_k$$

where $M_1, \ldots, M_k$ are monomial features and $\alpha_1, \ldots, \alpha_k \in R$. *Value* of a polynomial feature $N = \alpha_1 M_1 + \cdots + \alpha_k M_k$ w.r.t a sampling ball B placed on a protein structure P is defined as

$$N(B) = \alpha_1 M_1(B) + \alpha_2 M_2(B) + \cdots + \alpha_k M_k(B).$$

Degree of a polynomial aggregation feature P is maximum among the degrees of its monomials.

Now, we extend the definitions of values of monomial and polynomial features for protein structures. Given a polynomial aggregation feature N and a sampling-ball radius R, we define the *value* $N(P)$ w.r.t. a protein structure P as:

$$N(P) = \frac{\int_{\widehat{P}} N(B) dB}{\int_{\widehat{P}} dB}$$

where $\widehat{P}$ is the set of all sampling balls with radius R which contain at least one amino acid of the protein structure P. The integral $\int_{\widehat{P}} dB$ in the denominator is used as a normalization constant. Intuitively, the integral computes the average value of a polynomial feature N over balls located on a given protein structure.

It can be seen quite easily that polynomial aggregation features on protein structures share convenient properties with the discrete ball histograms. They are invariant to rotation and translation of protein structures which is important for predictive classification tasks. Intuitively, a monomial feature $M = \tau_i$ corresponds to the average value of property τ_i (in sampling balls of a given radius) over a given protein structure. A monomial feature $M = \tau_i^2$ captures the *dispersion* of the values of property τ_i over a given protein structure. Indeed, let us have two proteins A and B and a monomial feature $M = \mathtt{charge}^2$ and let us assume that A and B are composed of the same number of amino acids and that they contain the same number of positively charged amino acids and no negatively charged amino acids. Finally, let us also assume that the positively charged amino acids are distributed more or less uniformly over the protein structure A but are concentrated in a small region of the protein structure B. Then it is not hard to see that for the values $M(A)$ and $M(B)$ it should hold $M(A) \leqslant M(B)$. Analogically, a monomial feature $M = \tau_i \cdot \tau_j$ corresponds to *agreement* of values of properties τ_i and τ_j over a given protein structure but the *covariance* of these values is better captured by the following expression involving monomial features:

$$M_1(P) - M_2(P) \cdot M_3(P)$$

where $M_1 = \tau_i \cdot \tau_j$, $M_2 = \tau_i$ and $M_3 = \tau_j$. Note that this expression is not a polynomial aggregation feature but only an expression composed of polynomial (monomial) aggregation features. This can be seen when we expand $M_1(P)$, $M_2(P)$ and $M_3(P)$ and obtain

$$M_1(P) - M_2(P) M_3(P) = \frac{\int_{\widehat{P}} \tau_i \cdot \tau_j dB}{\int_{\widehat{P}} dB} - \frac{\int_{\widehat{P}} \tau_i dB}{\int_{\widehat{P}} dB} \cdot \frac{\int_{\widehat{P}} \tau_j dB}{\int_{\widehat{P}} dB}$$

which is not a value of a polynomial aggregation feature. However, it can be easily constructed from some polynomial aggregation features.

Values of polynomial aggregation features can be further decomposed into so called k-*values* computed only from balls containing exactly k amino acids. Given a polynomial feature N and a positive integer k, the k-value of N w.r.t. a protein P is given as

$$N(P|k) = \frac{\int_{\widehat{P}_k} N(B)dB}{\int_{\widehat{P_k}} dB}$$

where $\widehat{P}_k$ is the set of all sampling balls which contain exactly k amino acids. The value of a polynomial feature can then be expressed using k-values as

$$N(P) = \sum_i \beta_i \cdot N(P|i)$$

where $\beta_i = \int_{\widehat{P_i}} dB / \int_{\widehat{P}} dB$.

In summary, polynomial aggregation features can be expressed using combinations of monomial aggregation features and values of monomial aggregation features can, in turn, be computed using simple expressions involving k-values of monomial aggregation features and proportions of balls containing exactly a given number of amino acids. This implies that when using polynomial features for construction of attributes for machine learning, we can rely solely on the k-values and the few proportions and let the machine learning algorithms compute the values of monomial or polynomial aggregation features from these values if needed.

15.2 METHOD

Polynomial aggregation features can be used for predictive classification in a way completely analogical to discrete ball histograms. Given a template τ, sampling-ball radius R, a maximum degree d_{max} and a protein structure P, we construct all monomials containing the continuous variables from τ and having degree at most d_{max}. After that we construct the attribute-table. The rows of this table correspond to examples and the columns (attributes) correspond to k-values of the constructed monomial features. There is an attribute for every k-value such that there is at least one protein structure in the dataset which contains a set of k amino acids fitting into a ball of radius R.

The integrals used in definitions of values (or k-values) of monomial aggregation features are difficult to evaluate precisely therefore we use a Monte-Carlo-based approach similar to the case of discrete ball histograms. The set of k-values of monomial aggregation features for a protein P is computed as follows. First, a bounding sphere is found for the protein structure (with geometric center located in the geometric center of the protein structure and with radius $R_S = \max_{Res \in P}(distance(Res, C)) + R$, where R is a specified sampling-ball radius). After that the method collects a pre-defined number of samples containing at least one amino acid from the bounding sphere. For each sampling ball B the algorithm computes k_B-values (where k_B is the number of amino acids contained in B) of all monomial features complying with a given template and with a given maximum degree and stores them. In the end, the collected k-values of sampling balls are averaged to produce *approximate* k-values for the protein structure P.

After the attribute-table is constructed, it can be used to train an attribute-value classifier such as random forest or support vector machine which can be used for prediction on unseen proteins.

15.3 EXPERIMENTS

In this section, we describe results obtained in experiments with two datasets of DNA-binding proteins: PD138 and UD54, and two datasets of non-DNA-binding proteins: NB110 and NB843. We used monomial aggregation features with maximum degree 3, the following basic chemical properties of amino acids:

- Amino acid charge (under normal conditions),
- Amino acid Van der Waals volume,
- Amino acid hydropathy index,
- Amino acid isoelectric point (pI),
- Amino acid dissociation constants pK1 and pK2

and the following three properties related to DNA-binding derived by Sathyapriya et al. [81]

- Amino acid base-contact propensity,
- Amino acid sugar-contact propensity,
- Amino acid phosphate-contact propensity.

We trained random forest classifiers using only the attributes having non-zero information gain-ratio on training set. When performing cross-validation, this attribute selection was performed separately on the respective training sets induced by cross-validation so that no information could leak from a training set to a testing set. We compared the ball-histogram method with polynomial features with the original discrete ball-histogram method and with the method of Szilágyi and Skolnick [92]. The estimated accuracies and AUCs are shown in Table 22. The ball-histogram method with polynomial features performed best in terms of accuracy in all cases and in terms of AUC in all but one case where the discrete ball-histogram method performed best. We also tested the original ball-histogram method with random forest classifiers enriched with attribute-selection but it did not improve the performance.

In addition, we performed experiments with the method of Szilágyi and Skolnick where we replaced logistic regression by random forests (the classifier originally used in their paper was logistic regression for which the obtained accuracy is shown in Table 22). We also compared the obtained results

	C. ball histograms		**D. ball histograms**		**Szilágyi et al.**	
	Acc	**AUC**	**Acc**	**AUC**	**Acc**	**AUC**
PD138/NB110	**0.89**	**0.95**	0.87	0.94	0.81	0.92
PD138/NB843	**0.89**	0.86	0.88	**0.87**	0.87	0.84
UD54/NB110	**0.87**	**0.90**	0.81	0.89	0.82	0.89
UD54/NB843	**0.95**	**0.83**	0.94	0.81	0.94	0.78

Table 22: Experimental results obtained by cross-validation for the continuous ball-histogram method (C. ball histograms), the original discrete ball-histogram method (D. ball histograms) and the method of Szilágyi and Skolnick (Szilágyi et al.).

with results reported by Szilágyi and Skolnick in [92]. When using random forest classifier with features of Szilágyi and Skolnick, accuracy increased to 0.82 for the dataset PD138/NB110, which is still lower than 0.89 obtained by the ball-histogram method with polynomial features, and remained unchanged for dataset PD138/NB843 and AUC actually decreased for both of the datasets by 0.02. Szilágyi and Skolnick [92] reported AUC 0.93 for the dataset PD138/NB110 which is still lower than 0.95 obtained by the ball-histogram method with polynomial features. They also reported AUC 0.91 for the dataset UD54/NB110 which is higher by 0.01 than the result obtained by the ball-histogram method with polynomial features. However, this value of AUC was obtained on the dataset UD54/NB110 by a logistic regression classifier trained on the dataset PD138/NB110. The reported value is probably overoptimistic because the proteins from the dataset NB110 were used both in the training set and in the test set.

It can be interesting to compare the results of this method using only structural information, with the results of methods which exploit also information about evolutionary conservation of regions on protein surfaces. Therefore, we decided to confront our results with the results of Nimrod et al. [68] (already mentioned in previous chapters). The AUC 0.96 and accuracy 0.90 reported by Nimrod et al. [68] for the datasets PD138 and NB110 differs only slightly (by 0.01) from our best results. The AUC 0.90 obtained for the datasets PD138 and NB843 differs by 0.04 from our best results. As already noted in Chapter 13, evolutionary information is very important for the prediction of protein's function. When removing evolutionary information, Nimrod et al.'s misclassification error on the dataset PD138/NB110 increased by 0.035 which corresponds to lower predictive accuracy than obtained by our method. Even without the evolutionary information the classifier of Nimrod et al. used significantly more information than our method.

In addition to improved accuracy, our method provides us with to-some-extent interpretable features involving distributions of regions with certain

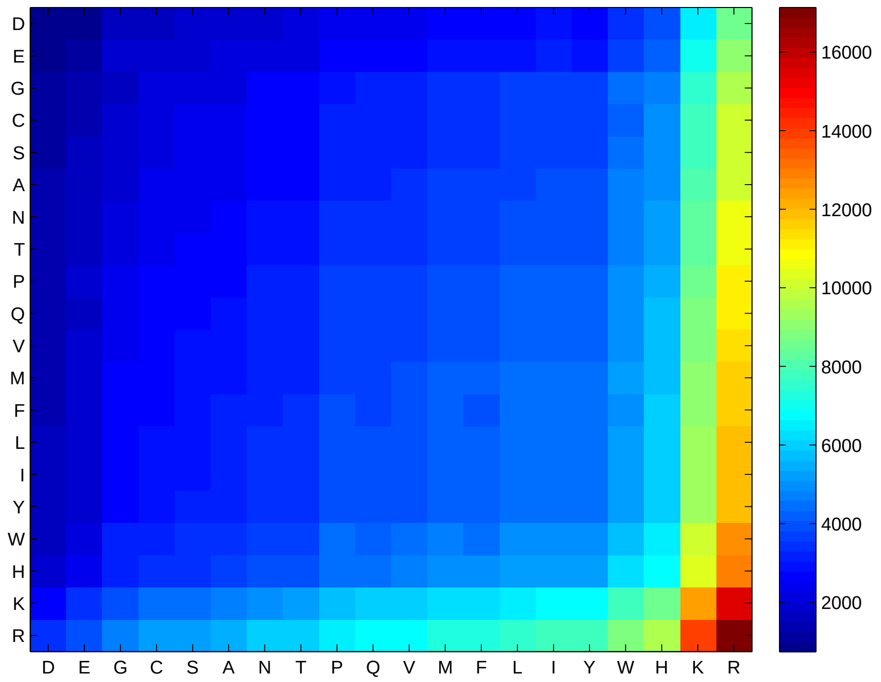


Figure 38: Values of feature $volume \cdot pI^2$ ($k = 3$) for combinations of amino acids in the form $\{AA_1, AA_1, AA_2\}$. The x-axis corresponds to AA_1 and the y-axis corresponds to AA_2.

chemical properties. We used information-gain attribute selection method to select three most informative attributes on the dataset PD138/NB843 for further inspection. The selected attributes (i.e. k-values of monomial features) were: $volume \cdot pI^2$ ($k = 3$), $P\_p \cdot volume \cdot charge$ ($k = 2$) and $P\_p \cdot volume \cdot pI$ ($k = 2$). It is interesting to note that the first (best) monomial did not involve any of the propensities P_p, P_B or P_S and that only the propensity P_p appeared in the remaining two of the three best monomials. The best feature $volume \cdot pI^2$ ($k = 3$) defines one value for every combination of three amino acids, which can appear in a sampling ball, and these values are then used to compute aggregated values over protein structures. In Figure 38 we show these values for combinations of amino acids in the form $\{AA_1, AA_1, AA_2\}$. It is interesting to note that the balls corresponding to the highest values of this feature are those containing positively-charged amino acids - arginine and lysine and that, analogically, the balls corresponding to the lowest values are those containing the negatively-charged amino acids - aspartic acid and glutamic acid. This is of course mainly caused by the pI^2 term because pI is the pH at which an amino acid carries no electrical charge. However, it is interesting that the monomial $volume \cdot pI^2$ is a better predictor of DNA-binding propensity than monomials pI, pI^2 or pI^3.

15.4 CONCLUSIONS

We extended the ball-histogram method presented in Chapter 13 by incorporation of polynomial aggregation features which are able to capture distributions of continuous properties of proteins' regions. The method achieved higher predictive accuracies than the original ball-histogram method as well as an existing state-of-the-art method.

16 DISCUSSION OF DISTRIBUTION-BASED APPROACHES

We introduced a novel type of approaches – so-called distribution-based approaches – for machine learning. The main idea is to capture distributions of certain properties in learning examples and to construct features based on these distributions. We developed methods which are able to use distributions of certain properties for construction of predictive classifiers – in primary structures using tube histograms and in 3D space using ball histograms. We used the ball-histogram method both for predictive classification and for regression. The ball-histogram method was also extended to be able to work with continuous properties of proteins' regions.

The advantage of the tube-histogram method presented in Chapter 12 is that it does not need structural information and despite relying only on the primary structure information it can achieve predictive accuracies higher than a method of Szilágyi et al. [92] which utilizes structural information. A natural generalization of this approach to 3D space is the ball-histogram method introduced in Chapter 13. This method incorporating structural information further improves predictive accuracy. This is not surprising given how important structural information is. In fact, what is more surprising is the relatively small difference between the accuracies achieved by the tube-histogram method and the ball-histogram method. However, the highest predictive accuracies were achieved by the ball-histogram method with polynomial features presented in Chapter 15. This method is able to work with continuous properties of proteins' regions, such as isoelectric point, Van der Waals volume or hydropathy index. It is able to capture higher order moments of distributions of these properties over protein structures. There are several possible explanations why this method performed best. First of all, the advantage of representing amino acids by their physicochemical properties and not by their types is that similar amino acids contribute similarly to the aggregate feature values, whereas relying only on the types of amino acids does not take into account the structural similarities of certain amino acids. Second, bins close to each other in ball histograms without polynomial features are treated in the same way as bins which are not close to each other. This problem is reduced in case of the ball-histogram method with polynomial features, because polynomial features can implicitly capture higher order moments. Finally, the simplest explanation is that polynomial features represent a more appropriate learning bias for predicting DNA-binding function.

In addition to predictive classification problems, the ball-histogram method can be used also for regression problems. The ball-histogram method for regression is described in Chapter 14. We had to adjust the template search algorithm, which required the learning examples to be class-labelled. The performed experiments indicate that the ball-histogram method obtains state-of-the-art results even for regression problems.

Part V

CONCLUSIONS

17 CONCLUSIONS

The objective of this thesis was to develop predictive models which would accurately predict proteins' DNA-binding function, while providing insights into the underlying DNA-binding process. We presented several novel methods which can be divided into two categories: relational learning approaches and distribution-based approaches.

In the first part, we presented the relational learning approaches for DNA-binding (and also antimicrobial activity) prediction. These approaches are based on our novel relational representation of protein structures. The first two approaches – relational learning based on structural patterns constructed by RelF and relational learning with polynomials – are based on combining a large number of relatively small structural patterns for building a prediction model. The third approach – relational learning with bounded LGG – is based on constructing a relatively small set of complex structural patterns which are used for predictive classification. The advantage of the approaches based on a large number of small structural patterns is that small patterns can be mined efficiently even from large protein structures. As we have shown experimentally, counting the occurrences of characteristic relational patterns in protein or peptide structures is crucial for accurate prediction. In the case of prediction of DNA-binding propensity of proteins, we achieved higher predictive accuracies than a state-of-the-art method of Szilágyi et al. [92]. In the case of prediction of antimicrobial activity of peptides, we were also able to outperform a state-of-the-art method of Torrent et al. [97]. The second approach – relational learning with polynomials – was designed to efficiently model multi-relational domains with numerical data. Unfortunately, this method does not scale as good as the former method. The reason is that the number of polynomial relational features is higher than the number of *non-aggregation* patterns. On the other hand, this method (using only information about primary and secondary structures) is able to obtain similar predictive accuracies as the method based on structural patterns. The last approach – relational learning with bounded LGG – is able to construct large, complex relational patterns. Due to the nature of these complex patterns it is not necessary to count their occurrences in order to obtain high predictive accuracies. This was confirmed by the experimental results on datasets of antimicrobial peptides. Relational representation of protein data consists of thousands of literals. In order to reduce the complexity of such input data, relational counterparts of feature selection methods would be required. Here, we also presented advanced preprocessing techniques for reduction of relational learning examples. Interestingly, when applied on protein datasets in our representation, the reduction in size was small, which suggests that our representation is not redundant.

In the second part, we presented the distribution-based approaches for DNA-binding (and also antimicrobial activity) prediction. The main idea of these approaches is to capture distributions of certain properties in learning examples and to construct features based on these distributions. We developed methods which are able to use distributions of certain properties for construction of predictive classifiers – in primary structures using the tube-histogram method and in 3D space using the ball-histogram method. The advantage of the tube-histogram method is that it does not need structural information and despite that it can achieve predictive accuracies higher than state-of-the-art methods which utilize structural information. A natural generalization of this approach to 3D space is the ball-histogram method. We used the ball-histogram method both for predictive classification and for regression. The ball-histogram method was also extended to be able to work with continuous properties of proteins' regions. The highest predictive accuracies were achieved by the ball-histogram method with polynomial features. This method is able to capture higher order moments of distributions of these properties over protein structures.

Part VI

APPENDIX

A TREELIKER

TreeLiker is a suite of algorithms enabling to analyze complex-structure data with standard machine-learning algorithms. It automatically identifies structural features of given data instances to produce an attribute-value data representation processible by common machine-learning systems. TreeLiker contains the implementations of relational learning algorithms used in this thesis, suitable for different kinds of data analysis problems. TreeLiker integrates these algorithms into a unified environment with joint data structures and user control. The open-source JAVA implementation licensed under GNU GPL is based on a common set of classes and is optimized for scalability as required for real-life bioinformatics problems. It includes both a graphical and a scripting interface and its output is readable by the popular machine-learning suite WEKA. TreeLiker is a software tool enabling to apply standard attribute-value machine learning algorithms on structured data such as molecular structures or biochemical networks.

The software package *TreeLiker* contains implementations of three feature-construction and propositionalization algorithms: HiFi [40], RelF [49, 42], and Poly [43, 46]. All three produce tree-like features, use the same mechanism (so-called *templates*) for specifying a particular language bias, and exploit a special, block-wise strategy to construct features. HiFi and RelF generate Boolean-valued or numerical (integer-valued) features. In the latter case, the value is the number of different matching substitutions for the example. Unlike HiFi, RelF assumes that examples are class-labeled and thus is especially suitable for supervised-learning analysis. Using class labels, RelF can filter out *redundant* and *irrelevant* features. Poly generates numerical (real-valued) features and does not require class labels. Its main distinguishing feature is its focus on domains which contain large amounts of information in the form of numerical data. Poly is based on a technique of multivariate polynomial aggregation, generalizing the concepts of Gaussian Logic [43].

A.1 IMPLEMENTATION

The algorithms are implemented in Java and all three are based on the same core set of underlying classes. These classes provide support for the syntactical generation of features, for their filtering based on syntactical and redundancy constraints, and for multivariate aggregation. The core classes are compact, comprising of approximately 15 thousand lines of code. The implementation is intended to be easily extendible. For example, it is easy to add new types of multivariate aggregation features. The core code of the algorithms is documented using JAVADOC.

Efficiency of the implemented algorithms was an important objective during the development of TreeLiker. The implementation contains substantial parts which are parallelized in order to harness the power of multi-core processors. Furthermore, many sub-results are cached using intelligent mechanism of so-called *soft-references* provided by the Java virtual machine. Soft references allow us to let the virtual machine decide when the cached results should be discarded in order to free the memory.

A.1.1 *Representation of Input Data*

Learning examples are composed of ground facts which are expressions not involving variables, for example: hasCharge(arginine). Two instances of learning examples are shown below:

DNA-binding aminoacid(a), is(a, histidine), aminoacid(b), is(b, cysteine), distance(a, b, 6.0), distance(b, a, 6.0)

non-DNA-binding aminoacid(a), is(a, tryptophan), aminoacid(b), is(b, tyrosine), distance(a, b, 4.0), distance(b, a, 4.0)

Here, the first *word* on each line denotes class of the example. The rest of the line is then the set of true facts - the description of the example. In this simple case, the first learning example is labelled as a *DNA-binding* protein (that is the class of this example) and the protein has an amino acid Histidine and an amino acid Cysteine which are in distance 6.0 Å from each other. The second learning example is labelled as a *non-DNA-binding* protein and has an amino acid Tryptophan and an amino acid Tyrosine which are in distance 4.0 Å from each other. In a realistic setting, the description of a protein would consist of thousands of facts.

A.1.2 *Types of Relational Features*

The algorithms contained in TreeLiker (RelF, HiFi and Poly) are intended for construction of relational features. Relational features are conjunctions of literals. For example,

$$F = \mathsf{aminoacid}(A), \mathsf{distance}(A, B, 6.0), \mathsf{is}(B, \mathsf{cysteine})$$

is a feature stipulating the presence of an untyped amino acid and a cysteine in the mutual distance of 6Å. A feature F matches example e if and only if there is a substitution θ to the variables of F such that $F\theta \subseteq e$. So for example, our feature F matches the first learning example in the previous section. This can be also seen as checking whether a conjunctive database query (feature) succeeds for a given relational database (learning example).

There are three settings in which the three feature construction algorithms can work. The first is the existential setting in which we are only interested in whether a given feature matches an example. As a result of matching, the feature thus receives a Boolean value. The second setting is the counting setting. Here, we count how many substitutions θ there are such that $F\theta \subseteq e$

for feature F and example e. The feature then receives an integer value. The counting interpretation showed some significant advantages over the existential interpretation in analysis of DNA-binding proteins (see Section 5.3.1).

Finally, there is also a third setting based on multivariate relational aggregation, designed especially for representing structures annotated with numerical data. The method is described in detail in Chapter 6. Briefly, in this setting, a feature may contain several distinguished variables which are used as *extractors* of numerical information. Every substitution θ such that $F\theta \subseteq e$ gives us one sample of the numerical variables which is a vector of real numbers. This vector can be used as input to a multivariate function (which may compute e.g. correlations of the variables). The result is then computed by averaging outputs of the multivariate function over all samples for the given example e.

A.1.3 *Language Bias - Templates*

Through *templates*, the user constrains the syntax of generated features.[1] Formally, templates are sets of literals. For example, the following expression is a template:

$$\begin{aligned}\tau_1 \quad = \quad & \mathsf{aminoacid}(-a), \mathsf{is}(+a, \#\mathsf{str}), \mathsf{distance}(+a, -b, \#\mathsf{num}), \\ & \mathsf{aminoacid}(+b), \mathsf{is}(+b, \#\mathsf{str}).\end{aligned}$$

Literals in templates have typed arguments. In template τ_1 above, the types are: a and b. Only same-typed arguments may contain the same variable in a correct feature. For example, the next *feature* complies with the *typing* from template τ_1:

$$F_1 = \mathsf{aminoacid}(X), \mathsf{distance}(X, Y, 4), \mathsf{is}(Y, \mathsf{histidine})$$

which can be checked easily: variable X appears only in arguments which can be marked by type a and variable Y appears only in arguments which can be marked by type b. On the other hand, the next expression is not a valid feature according to the typing in template τ_1:

$$F_2 = \mathsf{aminoacid}(X), \mathsf{is}(Y, X)$$

because X appears in arguments marked by two different types: a and str.

Arguments of literals in templates also have *modes*. The most important types of modes are the following (signs, shown in parentheses, are used in templates to denote the particular modes): *input* (+), *output* (-), *constant* (#) and *aggregation* (*).

We start by explaining the two most important types of modes: *input* and *output* modes. Any variable in a valid feature must appear exactly once as an *output* (i.e. in an argument marked by mode -) and at least once as an *input* (i.e. in an argument marked by mode +). For example, F_1 (shown above) complies

[1] Templates are similar in spirit to *mode-declarations* used in inductive logic programming systems Aleph or Progol [63].

with this condition w.r.t. the respective templates. On the other hand, for example, F_2 and F_3 and F_4 shown below do not comply with modes specified in template τ_1:

$$F_3 = \mathsf{aminoacid}(A), \mathsf{distance}(A, B, 6)$$

$$F_4 = \mathsf{distance}(A, B, 6), \mathsf{is}(B, \mathsf{histidine}).$$

The feature F_3 is not valid because variable B appears as an *output* but not as an *input*. The feature F_4 is not valid because variable A appears as an *input* but not as an output.

A simple way to understand how templates specify features is to imagine them procedurally as defining a process for constructing valid features. The process can be visualized as follows. We find a literal in the given template which does not contain any input-argument (i.e. none of its arguments is marked by +) and create the first literal of the feature to be constructed from it, e.g.

$$\mathsf{aminoacid}(A) \tag{3}$$

from the template-literal $\mathsf{aminoacid}(-a)$. Then we search for literals in the template which have an input-argument with such type that can be connected to $\mathsf{aminoacid}(A)$. Such a literal is $\mathsf{distance}(+a, -b, \#\mathsf{num})$ or $\mathsf{is}(+a, \#\mathsf{str})$. So, for example, we can create a literal $\mathsf{distance}(A, B, 8)$ according to $\mathsf{distance}(+a, -b, \#\mathsf{num})$ and connect it to $\mathsf{aminoacid}(A)$ which gives us

$$\mathsf{aminoacid}(A), \mathsf{distance}(A, B, 8). \tag{4}$$

Now, we have several options to extend the partially constructed feature (4). One possibility is to connect another literal to the variable A. We can add another literal based on $\mathsf{is}(+a, \#\mathsf{str})$ or $\mathsf{distance}(+a, -b)$, because there may be multiple input-occurrences of one variable, or we can add a literal based on $\mathsf{is}(+b, \#\mathsf{str})$ or $\mathsf{aminoacid}(+b)$ and connect it to variable B. Let us assume that we decided to follow the last option. Then we can get e.g. the following expression:

$$\mathsf{aminoacid}(A), \mathsf{distance}(B, C, 6), \mathsf{is}(B, \mathsf{histidine}) \tag{5}$$

We could continue in this process indefinitely and create larger and larger expressions. However, as was shown in [42], there is only a finite number of non-reducible tree-like features. Moreover, there are usually even fewer non-reducible and non-redundant features. RelF, HiFi and Poly are able to search through all these possible features exhaustively and efficiently.

Any template-literal can contain at most one input-argument. For example, the next template is not valid

$$\tau_2 = \mathsf{atom}(-a, \#\mathsf{atomType}), \mathsf{bond}(+a, +a)$$

because the literal $\mathsf{bond}(+a, +a)$ has two input arguments.

Mode and type declarations must not contain cycles. There is an additional technical requirement on valid templates. Let us define an auxiliary graph. In this graph, we have one vertex for each type (of arguments) contained in the template. There is an edge from a vertex V to a vertex W if and only if there is a literal which contains the type associated to the vertex V in an input argument and the type associated to the vertex W in an output argument. This graph must not contain oriented cycles.

This means that the next template

$$\tau_3 = \mathsf{atom}(+a), \mathsf{bond}(+a, -a)$$

is not valid because there is a cycle (loop in this case) from a to a. Similarly, the template

$$\tau_4 = \mathsf{atom}(-a), \mathsf{bond}(+a, -b), \mathsf{bond}(+b, -a)$$

is also not valid because there is a cycle $a - b - a$.

Very often, we need not only variables but also constants. Templates can be used to denote which arguments may contain only constants. For example for the next template

$$\tau_5 = \mathsf{aminoacid}(-a), \mathsf{is}(+a, \#\mathsf{const})$$

one of the possible valid features could be

$$F_5 = \mathsf{aminoacid}(A), \mathsf{is}(A, \mathsf{histidine}).$$

where his is a constant. Another example of a template using constants is shown next:

$$\tau_6 = \mathsf{atom}(-a, \#\mathsf{atomType}), \mathsf{bond}(+a, -b), \mathsf{atom}(+b, \#\mathsf{atomType})$$

which specifies features such as:

$$F_6 = \mathsf{atom}(X, \mathsf{carbon}), \mathsf{bond}(X, Y), \mathsf{atom}(Y, \mathsf{carbon}), \mathsf{bond}(X, Z), \mathsf{atom}(Z, \mathsf{hydrogen}).$$

The feature-construction algorithm Poly is able to construct multi-variate polynomial relational features. These are polynomial aggregation features which generalize μ-vectors and σ-matrices from Gaussian logic (see [43]). We need to be able to select which arguments can contain variables that should be used to *extract* the numerical values from the learning examples. We use so-called *aggregation modes* (denoted by *) for this. For example the next template:

$$\tau_7 = \mathsf{charge}(-a, *\mathsf{chrg}), \mathsf{bond}(+a, -b), \mathsf{charge}(+b, *\mathsf{chrg})$$

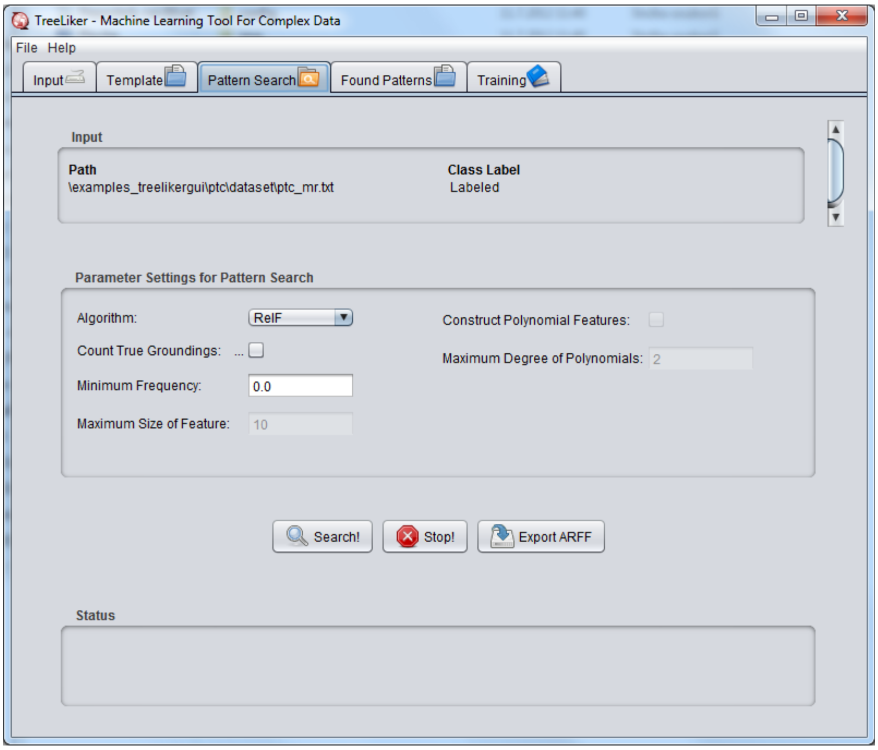
defines features which are able to construct multivariate polynomial features involving charges of atoms in molecules such as:

$$F_7 = \mathsf{charge}(X, \mathsf{CH1}), \mathsf{bond}(X, Y), \mathsf{charge}(Y, \mathsf{CH2}), \mathsf{bond}(X, Z), \mathsf{charge}(Z, \mathsf{CH3})$$

which can in turn be used to construct the polynomial aggregation features such as $\mathsf{AVG}(\mathsf{CH1} \cdot \mathsf{CH2} \cdot \mathsf{CH3})$ or $\mathsf{AVG}(\mathsf{CH1}^2)$.

A.1.4 *TreeLiker GUI*

The user interacts with TreeLiker either through a graphical interface or through a scripting interface. The former provides only a rather limited access to WEKA's learning algorithms and is meant mainly to assist the user in rapid assessment of the usefulness of the propositionalized representation in the iterations of template tuning. As soon as reasonable settings have been established, the user may employ TreeLiker through the scripting interface within more intricate experimental workflows.



Input

Path
\examples_treelikergui\ptc\dataset\ptc_mr.txt

Class Label
Labeled

Parameter Settings for Pattern Search

Algorithm: RelF

Count True Groundings: ... □

Minimum Frequency: 0.0

Maximum Size of Feature: 10

Construct Polynomial Features: □

Maximum Degree of Polynomials: 2

Status

Figure 39: A screenshot of the graphical user interface of TreeLiker.

The application consists of six main modules: Input Module, Template Module, Pattern Search Module, Found Patterns Module and Training Module. The Input Module allows the user to select the dataset directories or the specific files that should be used as input data. The user can add as many datasets as desired. The Template Module permits the user to introduce the template specifying the language bias that should be used in the execution of the algorithms. The Pattern Search Module enables the user to construct relational patterns for the datasets selected in the Input Module. The language bias is taken from the Template Module. The Found Patterns Module uses the results provided by the Pattern Search Module. It shows the structural patterns that were found. The Training Module allows the user to train a classifier based on the patterns generated in the Pattern Search Module. The available classifiers are Zero Rule, SVM with Radial Basis Kernel, J48 Decision Tree, One Rule, Ada-boost, Simple Logistic Regression, Random Forest, L2-Regularized Logistic Regression and Linear SVM.

A screenshot of the graphical user interface is shown in Figure 39.

A.1.5 *TreeLiker from Command Line*

The command line interface of TreeLiker provides full access to all features of TreeLiker (it provides access to some advanced functionalities which are not accessible from TreeLiker GUI). It allows to set-up more complicated experiments through TreeLiker-batch files.

TreeLiker can be run from command line once we have a TreeLiker-batch file with settings of the experiment. Here, we assume that we already have a TreeLiker-batch file called experiment.treeliker. Then, TreeLiker can be run using the following command:

```
java -Xmx1G -jar TreeLiker.jar -batch experiment.treeliker
```

A.1.5.1 *TreeLiker-Batch Files*

TreeLiker-batch files specify the data to be processed, algorithms with which they should be processed and the detailed settings of the algorithms. The content of a sample TreeLiker-batch file is shown below:

```
set(algorithm, relf) % the algorithm
set(output_type, single) % type of output (single = one file)
set(output, 'proteins.arff') % where to save the results
set(examples, 'proteins.txt') % the learning examples
% the template
set(template, [aminoacid(-a), is(+a, #aa_type), aminoacid(+b), is(+b, #aa_type),
distance(+a, -b, #num)])
work(yes) % tells TreeLiker to run the selected algorithm
  % with the selected parameters
```

The first line set(algorithm, relf) sets the algorithm to be used by TreeLiker. In this case, it is RelF which works in *existential mode*.

The second line set(output_type, single) sets the type of output. There are three types: 1. single which constructs one file using all the examples given in the training data, 2. cv which creates the given number (10 by default) of pairs of training and testing .arff (WEKA) files which can be used to perform cross-validation, 3. train_test which creates two files from the given training and testing data.

The third line set(output, 'proteins.arff') sets the output file in which the constructed relational features and the propositionalized table should be stored. TreeLiker uses .arff file format which is used in WEKA.

The fourth line set(examples, 'proteins.txt') sets path to the training examples which should be used.

The fifth line set(template, [aminoacid(-a), is(+a, #aa_type), aminoacid(+b), is(+b, #aa_type), distance(+a, -b, #num)]) specifies the template which should be used to constrain the space of possible features.

Finally, the sixth line work(yes) tells TreeLiker to run the selected feature construction algorithm.

If the above TreeLiker-batch file is run on the following file of training examples (proteins.txt):

DNA-binding aminoacid(a), is(a, his), aminoacid(b), is(b, cys), aminoacid(c), is(c, arg), distance(a, b, 6.0), distance(b, a, 6.0), distance(a, c, 4.0), distance(c, a, 4.0)

non-DNA-binding aminoacid(a), is(a, his), aminoacid(b), is(b, cys), aminoacid(c), is(c, arg), distance(a, b, 4.0), distance(b, a, 4.0), distance(a, c, 4.0), distance(c, a, 4.0)

non-DNA-binding aminoacid(a), is(a, trp), aminoacid(b), is(b, tyr), distance(a, b, 4.0), distance(b, a, 4.0)

then it outputs the following .arff file:

```
@relation propositionalization
@attribute 'aminoacid(A), distance(A, B, 4.0), aminoacid(B)' {'+'}
@attribute 'aminoacid(A), distance(A, B, 4.0), is(B, cys)' {'+','-'}
@attribute 'aminoacid(A), distance(A, B, 6.0), aminoacid(B)' {'+','-'}
@attribute 'aminoacid(A), is(A, arg)' {'+','-'}
@attribute 'aminoacid(A), is(A, trp)' {'+','-'}
@attribute 'classification' {'DNA-binding','non-DNA-binding'}

@data
'+', '+', '-', '+', '-', 'non-DNA-binding'
'+', '-', '+', '+', '-', 'DNA-binding'
'+', '-', '-', '-', '+', 'non-DNA-binding'
```

In the more realistic settings of our previous experimental evaluations, TreeLiker would construct tens of thousands features for thousands of learning examples. More involved ways of using TreeLiker are described in the user manual.

A.1.6 *Output*

All the three algorithms store their output as .arff files which can be read by WEKA [103]. As exemplified in the previous section, the output file contains both the definitions of the produced features and the attribute-value table consisting of evaluations of each feature on each example.

A.2 AVAILABILITY AND REQUIREMENTS

Project name: TreeLiker
Project home page: http://sourceforge.net/projects/treeliker
Operating system: Platform independent
Programming language: Java
Other requirements: Java 1.6 or higher

License: GNU GPL
Any restrictions to use by non-academics: none

BIBLIOGRAPHY

[1] S Ahmad and A Sarai. Moment-based prediction of dna-binding proteins. *Journal of Molecular Biology*, 341(1):65 – 71, 2004.

[2] A Atserias, A Bulatov, and V Dalmau. On the power of k-consistency. In *Proceedings of ICALP-2007*, pages 266–271, 2007.

[3] N Bhardwaj, RE Langlois, G Zhao, and H Lu. Kernel-based machine learning protocol for predicting dna-binding proteins. *Nucleic Acids Research*, 33(20):6486–6493, 2005.

[4] A Bhattacharyya. On a measure of divergence between two statistical populations defined by their probability distributions. *Bulletin of the Calcutta Mathematical Society 35*, pages 99–109, 1943.

[5] H Blockeel, L De Raedt, and J Ramon. Top-down induction of clustering trees. In *ICML '98: International Conference on Machine Learning*, pages 55–63, 1998.

[6] L Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. ISSN 0885-6125.

[7] P Bulet, R Stöcklin, and L Menin. Anti-microbial peptides: from invertebrates to vertebrates. *Immunological Reviews*, 198(1), 2004.

[8] CJC Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[9] R Caruana, N Karampatziakis, and A Yessenalina. An empirical evaluation of supervised learning in high dimensions. In *International Conference on Machine Learning (ICML)*, pages 96–103, 2008.

[10] T Cathomen and JK Joung. Zinc-finger nucleases: The next generation emerges. *Molecular Therapy*, 16, 2008.

[11] A Cherkasov and B Jankovic. Application of 'inductive' qsar descriptors for quantification of antibacterial activity of cationic polypeptides. *Molecules*, 9(12):1034–1052, 2004.

[12] J Choi, E Amir, and D Hill. Lifted inference for relational continuous models. In *Proceedings of the Twenty-Sixth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-10)*, pages 126–134, 2010.

[13] L De Raedt. Logical settings for concept-learning. *Artificial Intelligence*, 95(1):187–201, 1997.

[14] R Dechter. *Constraint Processing*. Morgan Kaufmann Publishers, 2003.

[15] L Dehaspe and H Toivonen. Discovery of frequent datalog patterns. *Data Mining and Knowledge Discovery*, 3(1):7–36, 1999.

[16] JR Desjarlais and JM Berg. Toward rules relating zinc finger protein sequences and dna binding site preferences. *Proceedings of the National Academy of Sciences*, 89(16):7345–7349, 1992.

[17] JR Desjarlais and JM Berg. Use of a zinc-finger consensus sequence framework and specificity rules to design specific dna binding proteins. *Proceedings of the National Academy of Sciences*, 90(6):2256–2260, 1993.

[18] JR Desjarlais and JM Berg. Length-encoded multiplex binding site determination: application to zinc finger proteins. *Proceedings of the National Academy of Sciences*, 91(23):11099–11103, 1994.

[19] M Elrod-Erickson, MA Rould, L Nekludova, and CO Pabo. Zif268 protein-dna complex refined at 1.6å: a model system for understanding zinc finger-dna interactions. *Structure*, 4(10):1171–1180, 1996.

[20] CD Fjell, H Jenssen, K Hilpert, WA Cheung, N Pante, REW Hancock, and A Cherkasov. Identification of novel antibacterial peptides by chemoinformatics and machine learning. *Journal of Medicinal Chemistry*, 52(7):2006–2015, 2009.

[21] V Frecer. Qsar analysis of antimicrobial and haemolytic effects of cyclic cationic antimicrobial peptides derived from protegrin-1. *Bioorganic & Medicinal Chemistry*, 14(17):6065 – 6074, 2006.

[22] D Freedman, R Pisani, and R Purves. *Statistics*. W. W. Norton & Company, 2007.

[23] EC Freuder. Complexity of k-tree structured constraint satisfaction problems. In *Proceedings of the eighth National conference on Artificial intelligence - Volume 1*, AAAI'90, pages 4–9. AAAI Press, 1990.

[24] Y Freund and RE Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *EuroCOLT '95: Proceedings of the Second European Conference on Computational Learning Theory*, pages 23–37, London, UK, 1995. Springer-Verlag. ISBN 3-540-59119-2.

[25] M Gao and J Skolnick. Dbd-hunter: a knowledge-based method for the prediction of dna-protein interactions. *Nucleic Acids Research*, 36(12): 3978–92, 2008.

[26] KI Goh and AL Barabási. Burstiness and memory in complex systems. *EPL (Europhysics Letters)*, 2008.

[27] REW Hancock and A Rozek. Role of membranes in the activities of antimicrobial cationic peptides. *FEMS Microbiology Letters*, 206(2):143–149, 2002.

[28] T Hastie, R Tibshirani, and J Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York: Springer-Verlag, 2001.

[29] C Helma, RD King, S Kramer, and A Srinivasan. The predictive toxicology challenge 2000-2001. *Bioinformatics*, 17(1):107–108, 2001.

[30] JM Hilbe. *Logistic Regression Models*. Texts in statistical science. New York: Taylor & Francis, Inc., 2009. ISBN 9781420075755.

[31] T Horváth and J Ramon. Efficient frequent connected subgraph mining in graphs of bounded tree-width. *Theoretical Computer Science*, 411(31-33):2784 – 2797, 2010.

[32] T Horváth, G Paass, F Reichartz, and S Wrobel. A logic-based approach to relation extraction from texts. In *ILP*, pages 34–48, 2009.

[33] DW Hosmer and S Lemeshow. *Applied logistic regression (Wiley Series in probability and statistics)*. Wiley-Interscience Publication, 2000.

[34] H Jenssen, T Lejon, K Hilpert, CD Fjell, A Cherkasov, and REW Hancock. Evaluating different descriptors for model design of antimicrobial peptides with enhanced activity toward p. aeruginosa. *Chemical Biology & Drug Design*, 70(2), 2007.

[35] H Jenssen, CD Fjell, A Cherkasov, and REW Hancock. Qsar modeling and computer-aided design of antimicrobial peptides. *Journal of Peptide Science*, 14(1), 2008.

[36] S Jones, P van Heyningen, HM Berman, and JM Thornton. Protein-dna interactions: a structural analysis. *Journal of Molecular Biology*, 287(5): 877–896, 1999.

[37] S Jones, HP Shanahan, HM Berman1, and JM Thornton. Using electrostatic potentials to predict dna-binding sites on dna-binding proteins. *Nucleic Acids Research*, 31:7189 – 7198, 2003.

[38] M Krogel and S Wrobel. Transformation-based learning using multirelational aggregation. In *ILP '01: Inductive Logic Programming*, 2001.

[39] MA Krogel, S Rawles, F Zelezny, P Flach, N Lavrac, and S Wrobel. Comparative evaluation of approaches to propositionalization. In *ILP*. Springer, 2003.

[40] O Kuželka and F Železný. HiFi: Tractable propositionalization through hierarchical feature construction. In *Inductive Logic Programming Conference - Late Breaking Papers*, pages 69–74, 2008.

[41] O Kuželka and F Železný. Seeing the world through homomorphism: An experimental study on reducibility of examples. In *ILP*, pages 138–145. Springer, 2011.

[42] O Kuželka and F Železný. Block-wise construction of tree-like relational features with monotone reducibility and redundancy. *Machine Learning*, 83:163–192, 2011.

[43] O Kuželka, A Szabóová, M Holec, and F Železný. Gaussian logic for predictive classification. In *Proceedings of the 2011 European conference on Machine learning and knowledge discovery in databases - Part II*, ECML PKDD, pages 277–292. Springer-Verlag, 2011.

[44] O Kuželka, A Szabóová, and F Železný. Gaussian logic and its applications in bioinformatics. In *BCB*, pages 496–498. ACM, 2011.

[45] O Kuželka, A Szabóová, and F Železný. Extending the ball-histogram method with continuous distributions and an application to prediction of dna-binding proteins. In *BIBM*, pages 1–4, 2012.

[46] O Kuželka, A Szabóová, and F Železný. Relational learning with polynomials. In *Proceedings of IEEE International Conference on Tools with Artificial Intelligence*, ICTAI, pages 1145–1150, 2012.

[47] O Kuželka, A Szabóová, and F Železný. Bounded least general generalization. In *ILP*, 2012.

[48] O Kuželka, A Szabóová, and F Železný. Reducing examples in relational learning with bounded-treewidth hypotheses. In *New Frontiers in Mining Complex Patterns*, volume 7765 of *Lecture Notes in Computer Science*, pages 17–32. Springer Berlin Heidelberg, 2013.

[49] Ondřej Kuželka and Filip Železný. Block-wise construction of acyclic relational features with monotone irreducibility and relevancy properties. In *ICML 2009: The 26th International Conference on Machine Learning*, 2009.

[50] N Landwehr, M Hall, and E Frank. Logistic model trees. *Machine Learning*, 59(1-2):161–205, 2005.

[51] N Landwehr, A Passerini, L De Raedt, and P Frasconi. kFOIL: learning simple relational kernels. In *AAAI'06: Proceedings of the 21st national conference on Artificial intelligence*, pages 389–394. AAAI Press, 2006.

[52] N Landwehr, K Kersting, and L DeRaedt. Integrating naïve bayes and FOIL. *Journal of Machine Learning Research*, 8:481–507, 2007.

[53] S Lata, N Mishra, and G Raghava. Antibp2: improved version of antibacterial peptide prediction. *BMC Bioinformatics*, 11, 2010.

[54] N Lavrac, D Gamberger, and V Jovanoski. A study of relevance for learning in deductive databases. *J. Log. Program.*, 40(2-3):215–249, 1999.

[55] N Lavrač and PA Flach. An extended transformation approach to inductive logic programming. *ACM Transactions on Computational Logic*, 2: 458–494, October 2001. ISSN 1529-3785.

[56] H Liu, H Motoda, R Setiono, and Z Zhao. Feature selection: An ever evolving frontier in data mining. *Journal of Machine Learning Research - Proceedings Track*, 10:4–13, 2010.

[57] NM Luscombe, SE Austin, HM Berman, and JM Thornton. An overview of the structures of protein-dna complexes. *Genome Biology*, 1, 2000.

[58] A Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8(1):99–118, 1977.

[59] Y Mandel-Gutfreund, O Schueler, and H Margalit. Comprehensive analysis of hydrogen bonds in regulatory protein dna-complexes: in search of common principles. *Journal of Molecular Biology*, 253(2):370–382, 1995.

[60] RP Mee, TR Auton, and PJ Morgan. Design of active analogues of a 15-residue peptide using d-optimal design, qsar and a combinatorial search algorithm. *The Journal of Peptide Research*, 49(1), 1997.

[61] L Monincová, M Buděšínský, J Slaninová, O Hovorka, J Cvačka, Z Voburka, V Fučík, L Borovičková, L Bednárová, J Straka, and V Čeřovský. Novel antimicrobial peptides from the venom of the eusocial bee halictus sexcinctus (hymenoptera: Halictidae) and their analogs. *Amino Acids*, 39:763–775, 2010.

[62] JL Moreland, A Gramada, OV Buzko, Q Zhang, and PE Bourne. The molecular biology toolkit (mbt): A modular platform for developing molecular visualization applications. *BMC Bioinformatics*, 6(1):21+, 2005. ISSN 1471-2105.

[63] S Muggleton. Inverse entailment and Progol. *New Generation Computing, Special issue on Inductive Logic Programming*, 13(3-4):245–286, 1995.

[64] S Muggleton and C Feng. Efficient induction of logic programs. In *ALT*, pages 368–381, 1990.

[65] S Muggleton, J Santos, and A Tamaddoni-Nezhad. Progolem: A system based on relative minimal generalisation. In *ILP*, pages 131–148, 2009.

[66] J Nardelli, T Gibson, and P Charnay. Zinc finger-dna recognition: analysis of base specificity by site-directed mutagenesis. *Nucleic Acids Research*, 20(16):4137–4144, 1992.

[67] H Nassif, H Al-Ali, S Khuri, W Keirouz, and D Page. An Inductive Logic Programming approach to validate hexose biochemical knowledge. In *Proceedings of the 19th International Conference on ILP*, pages 149–165, Leuven, Belgium, 2009.

[68] GG Nimrod, A Szilágyi, C Leslie, and N Ben-Tal. Identification of dna-binding proteins using structural, electrostatic and evolutionary features. *Journal of molecular biology*, 387(4):1040–53, 2009.

[69] DH Ohlendorf and JB Matthew. Electrostatics and flexibility in protein-dna interactions. *Advances in Biophysics*, 20:137 – 151, 1985.

[70] CO Pabo and RT Sauer. Transcription factors: structural families and principles of dna recognition. *Annual review of biochemistry*, 61:1053–1095, 1992.

[71] CN Pace and JM Scholtz. A helix propensity scale based on experimental studies of peptides and proteins. *Biophysical journal*, 75(1):422–427, 1998.

[72] NP Pavletich and CO Pabo. Zinc finger-dna recognition: crystal structure of a zif268-dna complex at 2.1å. *Science*, 252(5007):809–17, 1991.

[73] BM Peters, ME Shirtliff, and MA Jabra-Rizk. Antimicrobial peptides: Primeval molecules or future drugs? *PLoS Pathogens*, 6, 10 2010.

[74] R Pichler and S Skritek. Tractable counting of the answers to conjunctive queries. In *AMW*, volume 749 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2011.

[75] GD Plotkin. A note on inductive generalization. *Machine Intelligence*, 5: 153–163, 1970.

[76] Luc De Raedt. *Logical and Relational Learning*. Springer, 2008.

[77] M Richardson and P Domingos. Markov logic networks. In *Machine Learning*, page 2006, 2006.

[78] F Rossi, P van Beek, and T Walsh, editors. *Handbook of Constraint Programming*. Elsevier, 2006.

[79] S Russell and P Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, Englewood Cliffs, NJ, 2nd edition edition, 2003.

[80] J Santos and S Muggleton. Subsumer: A prolog theta-subsumption engine. In *Technical communications of the 26th Int. Conference on Logic Programming, Leibniz International Proc. in Informatics, Edinburgh, Scotland*, 2010.

[81] R Sathyapriya, MS Vijayabaskar, and S Vishveshwara. Insights into protein-dna interactions through structure network analysis. *PLoS Computational Biology*, 4(9), 09 2008.

[82] TW Siggers and B Honig. Structure-based prediction of C2H2 zinc-finger binding specificity: sensitivity to docking geometry. *Nucleic Acids Research*, 35(4), 2007.

[83] A Srinivasan. The aleph manual, 4th edition. available online: http://www.cs.ox.ac.uk/activities/machlearn/aleph/aleph.html, 2007.

[84] A Srinivasan and SH Muggleton. Mutagenesis: Ilp experiments in a non-determinate biological domain. In *ILP '94: Inductive Logic Programming*, pages 217–232, 1994.

[85] EW Stawiski, LM Gregoret, and Y Mandel-Gutfreund. Annotating nucleic acid-binding function based on protein structure. *Journal of Molecular Biology*, 326:1065–1079, 2003.

[86] A Szabóová, O Kuželka, F Železný, and J Tolar. Prediction of dna-binding proteins from structural features. In *MLSB 2010: 4th International Workshop on Machine Learning in Systems Biology*, pages 71–74, 2010.

[87] A Szabóová, O Kuželka, SE Morales, F Železný, and J Tolar. Prediction of dna-binding propensity of proteins by the ball-histogram method. In *ISBRA*, pages 358–367, 2011.

[88] A Szabóová, O Kuželka, F Železný, and J Tolar. Searching for important amino acids in dna-binding proteins for histogram methods (poster paper). In *The 7th International Symposium on Bioinformatics Research and Applications*, 2011.

[89] A Szabóová, O Kuželka, and F Železný. Prediction of antimicrobial activity of peptides using relational machine learning. In *BIBM Workshops*, pages 575–580, 2012.

[90] A Szabóová, O Kuželka, F Železný, and J Tolar. Prediction of dna-binding propensity of proteins by the ball-histogram method using automatic template search. *BMC Bioinformatics*, 13(S-10):S3, 2012.

[91] A Szabóová, O Kuželka, F Železný, and J Tolar. Prediction of dna-binding proteins from relational features. *Proteome Science*, 10(1):66, 2012. ISSN 1477-5956.

[92] A Szilágyi and J Skolnick. Efficient prediction of nucleic acid binding function from low-resolution protein structures. *Journal of Molecular Biology*, 358(3):922–933, 2006.

[93] O Taboureau, OH Olsen, JD Nielsen, D Raventos, PH Mygind, and H Kristensen. Design of novispirin antimicrobial peptides by quantitative structure-activity relationship. *Chemical Biology & Drug Design*, 68(1):48–57, 2006.

[94] S Thomas, S Karnik, RS Barai, VK Jayaraman, and S Idicula-Thomas. Camp: a useful resource for research on antimicrobial peptides. *Nucleic Acids Research*, 38:D774–D780, 2010.

[95] SK Thukral, ML Morrison, and ET Young. Mutations in the zinc fingers of adr1 that change the specificity of dna binding and transactivation. *Molecular and Cellular Biology*, 12(6):2784–2792, 1992.

[96] M Torrent, V Nogués, and E Boix. A theoretical approach to spot active regions in antimicrobial proteins. *BMC Bioinformatics*, 10(1), 2009.

[97] M Torrent, D Andreu, VM Nogués, and E Boix. Connecting peptide physicochemical and antimicrobial properties by a rational prediction model. *PLoS ONE*, 6, 02 2011.

[98] Y Tsuchiya, K Kinoshita, and H Nakamura. Structure-based prediction of dna-binding sites on proteins using the empirical preference of electrostatic potential and the shape of molecular surfaces. *Proteins: Structure, Function, and Bioinformatics*, 55(4):885–894, 2004.

[99] V Čeřovský, O Hovorka, J Cvačka, Z Voburka, L Bednárová, L Borovičková, J Slaninová, and V Fučík. Melectin: A novel antimicrobial peptide from the venom of the cleptoparasitic bee melecta albifrons. *ChemBioChem*, 9(17):2815–2821, 2008.

[100] V Čeřovský, M Buděšínský, O Hovorka, J Cvačka, Z Voburka, J Slaninová, L Borovičková, V Fučík, L Bednárová, I Votruba, and J Straka. Lasioglossins: Three novel antimicrobial peptides from the venom of the eusocial bee lasioglossum laticeps (hymenoptera: Halictidae). *ChemBioChem*, 10(12):2089–2099, 2009.

[101] C Vens, A Van Assche, H Blockeel, and S Dzeroski. First order random forests with complex aggregates. In *ILP: Inductive Logic Programming*, pages 323–340, 2004.

[102] J Wang and P Domingos. Hybrid markov logic networks. In *Proceedings of the 23rd national conference on Artificial intelligence - Volume 2*. AAAI Press, 2008.

[103] IH Witten and E Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2nd edition, 2005.

[104] SA Wolfe, L Nekludova, and CO Pabo. DNA recognition by Cys-2-His-2 zinc finger proteins. *Annual review of biophysics and biomolecular structure*, 29:183–212, 2000.

[105] S Wu and Y Zhang. Lomets: A local meta-threading-server for protein structure prediction. *Nucleic Acids Research*, 35(10):3375–3382, 2007.

[106] M Yannakis. Algorithms for acyclic database schemes. In *International Conference on Very Large Data Bases (VLDB '81)*, pages 82–94, 1981.

[107] MR Yeaman and NY Yount. Mechanisms of antimicrobial peptide action and resistance. *Pharmacological Reviews*, 55(1):27–55, 2003.

[108] M Žáková, F Železný, J Garcia-Sedano, CM Tissot, N Lavrač, P Křemen, and J Molina. Relational data mining applied to virtual engineering of product designs. In *ILP06*, volume 4455 of *LNAI*, pages 439–453. Springer, 2007.