DISCOVERING DISCRIMINATIVE AND CLASS-SPECIFIC SEQUENCE AND STRUCTURAL MOTIFS IN PROTEINS

by

CEM MEYDAN

Submitted to the Graduate School of Engineering and Natural Sciences
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

Sabancı University

June 2013

DISCOVERING DISCRIMINATIVE AND CLASS-SPECIFIC SEQUENCE AND STRUCTURAL MOTIFS IN PROTEINS

APPROVED BY:

Prof. Uğur Sezerman, (Dissertation Supervisor)

[illegible] ....

Prof. Canan Atılgan

Canan Atılgan ....

Prof. Selim Çetiner

[illegible] ....

Asst.Prof. Murat Çokol

[illegible] ....

Prof. Özlem Keskin

Ö. Keskin ....

DATE OF APPROVAL: 21.06.2013

Abstract

Finding recurring motifs is an important problem in bioinformatics. Such motifs can be used for any number of problems including sequence classification, label prediction, knowledge discovery and biological engineering of proteins fit for a specific purpose. Our motivation is to create a better foundation for the research and development of novel motif mining and machine learning methods that can extract class-specific and discriminative motifs using both sequence and structural features.

We propose the building blocks of a general machine learning framework to act on a biological input. This thesis present a combination of elements that are aimed to be applicable to a variety of biological problems. Ideally, the learner should only require a number of biological data instances as input that are classified into a number of different classes as defined by the researchers. The output should be the factors and motifs that discriminate between those classes (for reasonable, non-random class definitions). This ideal workflow requires two main steps. First step is the representation of the biological input with features that contain the significant information the researcher is looking for. Due to the complexity of the macromolecules, abstract representations are required to convert the real world representation into quantifiable descriptors that are suitable for motif mining and machine learning. The second step of the proposed workflow is the motif mining and knowledge discovery step. Using these informative representations, an algorithm should be able to find discriminative, class-specific motifs that are over-represented in one class and under-represented in the other.

This thesis presents novel procedures for representation of the proteins to be used in a variety of machine learning algorithms, and two separate motif mining algorithms, one based on temporal motif mining, and the other on deep learning, that can work with the given biological data. The descriptors and the learners are applied to a wide range of computational problems encountered in life sciences.

Özet

Biyolojik motiflerin keşfi biyoinformatik için önemli problemlerden biridir. Bu tür motifler, dizilerin sınıflandırılması, veri madenciliği ve rasyonel protein mühendisliği gibi amaçlarla kullanılabilir. Bu tez, proteinlerin dizi ve yapısal özelliklerinden ayrımcı motiflerin bulunması ve makine öğrenimi yöntemlerinin araştırma ve geliştirilmesinde kullanılmak üzere daha iyi bir temel oluşturma amacı barındırmaktadır.

Bu tez, çeşitli biyolojik problemlere uygulanabilirliği olan makine öğrenim yapı blokları önermektedir. Öğrenim algoritmalarının girdisi ideal olarak yalnızca biyolojik veri örneklemleri ve bu örneklerin ait olduğu sınıf verileri olmalıdır. Bu girdiye denk gelen çıktı ise bu sınıfları ayıran faktör ve motifler olmalıdır (rastgele olmayan, makul sınıf tanımları için). Bu ideal iş akışı iki ana adıma ihtiyaç duyar. Birinci adım, biyolojik örneklerin araştırma için önem arz eden özelliklerle temsil edilmesidir. Makromoleküller kompleks üç boyutlu yapılar olduğu için, bu komplike gösterimin soyutlaştırılarak makine öğrenimi ve motif keşfi için kullanmaya daha uygun sayısal ve simgesel temsillere dönüştürülmesi gerekmektedir. İkinci adım ise bu temsili gösterimler üzerinde kullanılmaya uygun motif keşfi ve makine öğrenimi algoritmalarının geliştirilmesidir. Bir algoritma ilk adımda çıkartılan tanıtıcı temsilleri kullanalarak sınıflandırıcı ve ayırt edici motifleri keşfedebilmelidir.

Bu çalışma ile çeşitli makine öğrenimi yöntemlerinde kullanılmak üzere bir çok yeni protein temsil yöntemleri; ve bu temsil sistemleri ile çalışmak üzere iki ayrı motif keşif yöntemi (zamana bağlı motif madenciliği ve derin öğrenim temelli motif keşfi) geliştirilmiştir. Bu temsil ve öğrenim algoritmaları yaşam bilimlerinde karşılaşılan çeşitli hesaplamalı problemlere uygulanmıştır.

Acknowledgements

It is a pleasure to express my humble gratitude to several individuals who in one way or another contributed and extended their valuable assistance in the preparation and completion of this dissertation.

I gratefully thank to my thesis advisor Prof. Dr. Uğur Sezerman, for his guidance and support throughout thesis. I also wish to convey my gratitude to my thesis progress jury members; Prof. Dr. Canan Atılgan and Prof. Dr. Selim Çetiner for sharing their exceptional scientific backgrounds; and also to all of my thesis jury members for their constructive comments on this dissertation. I am grateful that in the midst of all their activity, they accepted to participate.

I also wish to acknowledge the financial support of TUBITAK BIDEB.

I convey my sincere thanks to my dear friends and colleagues, Alper Küçükural, Aydın Albayrak, Begüm Topçuoğlu, Emel Durmaz, Günseli Akçapınar and Sinan Yavuz for their advice and their willingness to share their bright thoughts with me on every kind of subject and for the scientific discussions which greatly helped on forming this thesis. I also thank the great number of friends I made here, Batuhan Yenilmez, Can Timuçin, Çağrı Bodur, Ebru Kaymak, Özgür Gül, Tuğsan Tezil, Yasin Bakış, and including the ones I forgot to mention, I thank to all friends and fellows for providing the necessary motivation to take the load off my shoulders.

Last but not least; I would like to thank my family for their support and being there when I needed them to be.

TABLE OF CONTENTS

List of Tables

List of Figures

1 INTRODUCTION

1.1 Motivation

Finding recurring motifs is an important problem in bioinformatics. Such motifs can be used for any number of problems including sequence classification, label prediction, knowledge discovery and biological engineering of proteins fit for a specific purpose. In the biological context, the word motif usually connotes the concept of a sequence motif. However, the concept that discriminates between a set of macromolecules from others can be in any form, whether it is based on sequence identity, structural homology or functional similarity is irrelevant from a global point of view. The fact is, the concept of similarity is an abstract idea that is defined subjectively by the viewer. A protein can be assigned an arbitrary number of labels based on any of its features, thus two proteins that have identical labels when classified from a specific standpoint (e.g. having the same catalytic activity) can be assigned completely different labels when analyzed by another aspect (e.g. functional efficiency).

Due to this abstract concept of class, a formal definition of what a motif is and what type of information it should utilize cannot be defined. For this reason, there are large number of motif mining algorithms that deal with different aspects of the biological problems. The biological problems usually carry significant similarities within themselves, but are marginally different in one aspect to necessitate a slightly different approach during the development of the algorithm. This causes the number of available tools to approach the number of biological sub-problems. It can be argued that this approach is inefficient from the point of both the method development (since the developers frequently try to solve problems which were already solved for another case) and the use of those methods in research (large number of available but under-utilized tools, each with their own strengths and limitations).

Our motivation is to create a better foundation for the research and development of novel motif mining and machine learning methods. Thus, we propose the elements of a machine learning framework to act on biological input. This thesis present a combination of elements that are aimed to be applicable to a variety of biological problems. As mentioned, the class is an abstract concept, therefore a researcher may be looking for a specific feature when trying to find a motif. Ideally, the algorithm should only require a number of biological data instances as input that are classified into a number of different classes as defined by the researchers. The output should be the factors and motifs that discriminate between those classes (for reasonable, non-random class definitions). This ideal workflow requires two main steps, representation of the data and the extraction of the information from the input.

First step is the representation of the biological input with features that contain the significant information the researcher is looking for. Due to the complexity of the macromolecules, abstract representations are required to convert the real world representation into quantifiable descriptors that are suitable for motif mining and machine learning. By its definition, motif mining and machine learning experiments require the input representation to have a high generalization capability; the representation should be able to ignore slight noises in the input during the learning and decision making processes.

To give an example, we usually ignore or tolerate the presence of mutations and insertion/deletions while comparing two sequences, minor or even major differences between two sequences can be ignored to find the most common elements during motif extraction. However, the opposite also holds true, a slight mutation can have paramount effects on the whole protein, e.g. one mutation can significantly alter the fold, function and the stability of a protein.

As a result, the relationship between the input (e.g. primary structure for this case) and the output (3D structure, function, dynamics etc.) cannot be characterized linearly: Two perturbations of the same magnitude applied to an input data can result in two outputs that are significantly different. Ideally, the protein representation should be able to capture such a non-linear relationships. Then, the problem becomes, which slight changes in the input are noise and can be discarded, and which changes are informative. Unfortunately, this is not a problem that can be solved with the knowledge and technology of today. However, we try to find a very diverse set of higher-level representations that can be used in conjunction with machine learning methods to learn to approximate the relationship between the input and the output.

The second step of the proposed workflow is the motif mining and knowledge discovery step. Using these informative representations, an algorithm should be able to find discriminative, class-specific motifs that are over-represented in one class and under-represented in the other. The resulting motifs can be analyzed for the discovery of biological knowledge, or used with other machine learning tools to predict the label of unknown instances, characterize and quantify the relationship between other data sources or any combination of tasks.

1.2 Background

1.2.1 Biological Background

1.2.1.1 Amino Acids

Proteins are polymers that composed of amino acids linked through amide bonds (also called peptide bond). The peptide bonded polymer that forms the backbone of polypeptide structure is called the main chain. The peptide bonds of the main chain are rigid planar units formed by the dehydration reaction of the carboxyl of one amino acid with the amino group of another releasing one molecule of H_2O in the process. The carbonyl-amino amide bond has partial double bond character and also possesses no rotational freedom [1].

The physiochemical properties of each amino acid in a protein sequence ultimately determine its structure, reactivity, and function. Each amino acid is composed of an amino group and a carboxyl group bound to a central carbon, called the C_α. Also bound to the C_α are a hydrogen atom and a side chain that determines the physiochemistry of each amino acid. The side chains are not directly involved in the formation of the polypeptide backbone and are free to interact with their environment [1].

Amino acids may be grouped based on their side chain characteristics. There are 20 standard common amino acids found throughout nature, each containing a side chain with particular size, structure, charge, hydrogen bonding capacity, polarity, and reactivity. There are seven amino acids that contain aliphatic side chains, which are relatively non-polar and hydrophobic in character: glycine, alanine, valine, leucine, isoleucine, methionine, and proline. Glycine (Gly) is the simplest amino acid with its side chain consisting of only a hydrogen atom. Alanine (Ala) possesses a single methyl group for its side chain. Valine (Val), leucine (Leu), and isoleucine (Ile) are slightly more complex with three or four carbon branched-chain constituents. Methionine (Met) contains a thioether (-S-CH3) group at the terminus of its hydrocarbon chain. Proline (Pro) is actually the only imino acid and its side chain forms a ring structure with its amino group resulting in two covalent linkages to its Cα atom. Due to its unique structure, Pro often causes severe turns in a polypeptide chain and cannot be accommodated in normal α-helical structures, except at the ends where it may create a turning point for the chain [2].

Phenylalanine (Phe) and tryptophan (Trp) contain aromatic side chains that, like the aliphatic amino acids, are also relatively non-polar and hydrophobic. All of the aliphatic and aromatic hydrophobic residues are usually encountered at the interior of protein structure or in areas that are not readily accessible to water or other hydrophilic molecules.

Tyrosine (Tyr) contains a phenolic side chain with a pKa of about 9.7-10.1. Although the amino acid is only slightly soluble in water, the ionizable nature of the phenolic group makes it often appear in hydrophilic regions of a protein [1].

There are four amino acids which have relatively polar side chains and are hydrophilic: asparagine (Asn), glutamine (Gln), threonine (Thr), and serine (Ser). They are usually found at or near the surface where they can have favorable interactions with the surrounding hydrophilic environment. There is also another group of hydrophilic amino acids that contain ionizable side chains: aspartic acid (Asp), glutamic acid (Glu), lysine (Lys), arginine (Arg), cysteine (Cys), histidine (His), and tyrosine (Tyr). Both Asp and Glu contain carboxylate groups with similar ionization properties as the C-terminal carboxylate. The theoretical pKa of the carboxyl of Asp (3.7-4.0) and the carboxyl of Glu (4.2-4.5) are somewhat higher than the carboxyl groups at the C-terminal of a polypeptide chain (2.1-2.4). At pH values above their pKa, these groups are generally ionized to form negatively charged carboxylates. Thus at physiological pH, they contribute to the overall negative charge of a protein [1].

Lys, Arginine, and His have ionizable amine containing side chains that, similar to the N-terminal amine, contribute to a protein's overall net positive charge. Lys contains an unbranched four-carbon chain terminating in a primary amine group. The theoretical pKa of Lys amine is around 9.3-9.5 and at pH values lower than the pKa of this group, Lys is generally protonated to have a positive charge. At pH values greater than the pKa, Lys is unprotonated and has no net charge. Arg contains a strongly basic group on its side chain called a guanidino group. The ionization point of this residue is so high (pKa of 12.0) that keeps Arg always protonated with a positive charge. The side chain of His is an imidazole ring that is potentially protonated at slightly acidic pH values (pKa of 6.7-7.1). Thus, at physiological pH, these residues contribute to the overall net positive charge of an intact protein molecule. The amine containing side chains in Lysine, Arginine, and Histidine typically are located at the surface of proteins and can be involved in salt bridges through their interactions with the aspartic and glutamic acids [2].

Cys is the only amino acid containing a thiol group (-S-H). At physiological pH, this residue is normally protonated and possesses no charge. Ionization only occurs at high pH (pKa = 8.8-9.1) and results in a negatively charged thiolate group. The most important reaction of Cys residues in proteins is the formation of disulfide crosslinks with another Cys residue. Cys disulfides (also called cystine or disulfide bridges) often are key points in stabilizing protein structure and conformation. They frequently occur between polypeptide subunits, creating a covalent linkage to hold two chains together. Cysteines are relatively hydrophobic due to the small electronegativity difference (i.e., 2.58 vs. 2.20) between the sulfur and hydrogen atoms and usually can be found within the core of a protein [1]. For this reason, strong deforming agents may be needed to open up the protein core to fully reduce the disulfides bonds within structure.

1.2.1.2 Secondary and Tertiary Structures

Amino acids are linked through peptide bonds to form long polypeptide chains. The primary structure of protein molecules is simply the linear sequence of each amino acid residue along the main chain. Each amino acid in the chain can form various interactions with surrounding groups through its unique side chain functionalities. Noncovalent forces such as hydrogen bonding and ionic and hydrophobic interactions work together to create each protein's unique shape. The sequence and types of amino acids and the shape that they are folded into is the main factor which provides protein molecules with specific structure, activity, and function. Ionic charge, hydrogen bonding capability, and hydrophobicity are the major determinants for the resulting three-dimensional structure of protein molecules.

The main chain is twisted, folded, and formed into structural units called secondary structures based upon the intramolecular interactions such as H-bonds between the different parts of the peptide backbone. Major secondary structures of proteins such as α-helices and β-sheets are held together solely through a network of hydrogen bonding created through the carbonyl oxygens of peptide bonds interacting with the hydrogen atoms of other peptide bonds. Other minor secondary structures can also be found in the proteins such as 3_{10} helix, α-helix, turns, and β-bridges.

In addition, negatively charged residues may become bonded to positively charged groups through ionic interactions. Non-polar side chains may attract other non-polar residues and form regions of hydrophobicity to the exclusion of water and other ionic groups. Occasionally, disulfide bonds also are found holding different regions of the polypeptide chain together. All of these forces combine to create the secondary structure of proteins, which is the way the polypeptide chain folds in local areas to form larger, sometimes periodic structures.

On a larger scale, the unique folding and structure of one complete polypeptide chain is termed the tertiary structure of protein molecules. The difference between local secondary structure and complete polypeptide tertiary structure is arbitrary and sometimes of little practical difference. Larger proteins often contain more than one polypeptide chain. These multi-subunit proteins have a more complex shape, but are still formed from the same forces that twist and fold the local polypeptide. The unique three-dimensional interaction between different polypeptides in multi-subunit proteins is called the quaternary structure. Subunits may be held together by noncovalent contacts, such as hydrophobic or ionic interactions, or by covalent bonds formed by the cysteine residue of one polypeptide chain to another [2].

Aside from the covalently polymerized main chain itself, the protein structure is dominated by weaker, noncovalent interactions that are extremely susceptible to the environmental changes such that protein structure can be disrupted or denatured by fluctuations in pH, temperature, or by small amounts of chemicals that interferes with the inter-molecular interactions within a protein.

1.2.2 Machine Learning

Machine learning is a branch of artificial intelligence that deals with systems that can learn from the data. A machine learning system will "learn" the relationship within a training data, and can predict the outcome of an input using these known properties. The core of machine learning deals with representation and generalization. Representation of data instances and functions evaluated on these instances is the main basis of the learning process. Generalization is the ability of an algorithm to perform accurately on new, unseen examples after having trained on a learning data set. The core objective of a learner is to generalize from its experience. The training examples come from some generally unknown probability distribution and the learner has to extract a general trend within that distribution that allows the learner to produce useful predictions in new cases.

Machine learning methods can be separated into two main classes, supervised and unsupervised. In supervised (or discriminative) learning, we have a labelled vector of attributes Y, and an unlabelled vector of attributes X. Discriminative learning is the task of finding model parameters Θ such that the conditional probability $P(Y|X,\Theta)$ matches the trend seen in the X and Y values of the training set. In unsupervised (or generative) learning, there is no difference between the labelled and unlabelled attributes in the representation of the model. What is built is a joint probability model $P(X,Y|\theta)$. This means that all attributes, both labelled or unlabelled, can be predicted from the values of the model's parameters Θ. However, it is possible to use the joint model for classification. By conditioning the joint probability, we can obtain the conditional model for the labelled attributes Y given the unlabelled ones.

In a general context, the term "machine learning" is used to usually represent discriminative learning which focuses on prediction based on known properties learned from the training data. The term data mining is used when the task focuses on the discovery of previously unknown properties on the data. However, these distinctions are not clear and are usually based not on the approach but the goal in mind. Machine learning also employs data mining methods as unsupervised learning, which we'll focus on in the later chapters.

Classification is a supervised learning technique which deals with nominal labels on the instances. The learner will try to find attributes which discriminate between the inputs taken from different classes, and will use this knowledge to predict unknown class labels. Classification of proteins is an important process in many areas of bioinformatics including drug target identification, drug design, protein family characterization, and protein annotation. In a biological context, classification of proteins refers to the determination of the class of a protein or the assignment of a protein into a predefined category based on the existence of certain similarities to other members of the same category. Proteins can be classified based on their structural components, catalytic function, cellular location, pH and optimum working temperature and so on.

In classification, it is often interest to determine the class of a novel protein using features extracted from raw sequence or structure data rather than directly using the raw data. For example, a typical manual annotation of a novel protein can be carried out against a database which contains expert annotated proteins with other secondary attributes. The best match in the database can be used as a template and its properties may be transferred to the novel protein. The search would take the raw sequence information as input and find sequences that are similar to the given query sequence at a given similarity threshold. However, in a machine learning framework, the same process may be carried out as follows: i) obtain representative sequences from the database, ii) extract features from these sequences such as number and kind of domains, motif, signal regions, length of proteins, and post-translational modification sites, iii) utilize machine learning classifiers to learn from this training data, and iv) generate a model that can be used to predict the class of a new sample by testing the model on it.

Classification starts with the definition of a class and class properties that make it unique or different from other classes. Class boundaries may sometimes be difficult to establish due to following reasons: i) Class definition process is abstract in nature and does not represent underlying classes. ii) Established classes are not applicable to all proteins because of non-discovered classes. To eliminate boundary-related problems, a classification scheme may need to be updated with the availability of more data.

Previously, machine learning algorithms have been used in many classification problems particularly protein interaction prediction [3], cluster analysis of gene expression data [4], annotation of protein sequences by integration of different sources of information [5], automated function prediction [6], protein fold recognition and remote homology detection [7], SNP discovery [8], prediction of DNA binding proteins [9], and gene prediction in metagenomic fragments [10]. In the absence of experimental validation, similarity searches are routinely employed to transfer function or attribute of a known protein to a novel protein if the similarity is above a certain threshold. However, similarity searches may not necessarily perform well when similar proteins belong to different classes or families are used and significant mis-annotations can occur even at high sequence identity levels. In such cases, machine learning approaches can predict the class of a novel protein using features derived from raw sequence or structure data. In many cases, classification with machine learning approaches provides simple and yet advantageous solutions over more traditional, laborious and sometimes error-prone means that employ protein similarity measures.

1.3 Organization

This thesis is organized into six chapters. Chapter 1 (this chapter) is a general introductory chapter. Chapters 2, 3, 4 and 5 are organized as roughly self-sufficient individual units with their own Introduction, Methods, Results, and Conclusions sections. Each chapter is organized to address a different aspect of the motif mining and classification problems encountered in biological molecules. Chapters 2 and 3 deal with novel representations of the protein sequence and structure, and contain applications of the developed representations in real world problems. Building upon the works presented in Chapter 2 and 3, Chapters 4 and 5 present two novel robust motif mining methods suitable for extracting discriminative motifs in a variety of biological data. Those methods can work in both sequence and structure-level information using the feature representations described in Chapters 2 and 3. The algorithm definitions are supplemented with their applications to real world problems. Finally, Chapter 6 is a general conclusion chapter that summarizes the results of the studies.

Chapter 2 deals with representing the sequence and the structure of proteins in a local fashion. In this context, local means the sub-units of a protein that may be used for alignment, similarity and disparity calculations, database searches, biologically relevant motif discoveries and so on. To compare and contrast between two proteins, or to define reoccurring motifs, it is important to be able to match and quantify the relationships between the sub-units of different proteins. This quantification step requires the simplification of the physical molecule of the protein into more abstract parts.

The most common abstraction is the primary structure of the protein, defining the molecule as a collection of residue labels. This simplification allows us to represent the protein as a sequence of an alphabet of 20 amino acids. By defining a quantitative metric for the comparison of such an alphabet, such as the BLOSUM and PAM similarity scores between any two residues, it becomes trivial to check whether two proteins sequences are "equivalent". After this abstraction, we can define complex concepts such as similarity of two sequences and their alignment, database search of a sequence, DNA and protein domain detection/search/prediction, and a great number of biological problems as a mathematical problem that can be solved by a number of algorithms and heuristics. Therefore, it is important to be able to conceptualize the biological molecules into more abstract classes, nominal labels or numerical vectors.

Even though the abstraction process increases the signal-to-noise ratio of the input, and thus the generalization power, too much abstraction can filter out the information that we want to extract from the raw data. From there on, it becomes a trade-off to optimize the amount of task-relevant information versus irrelevant noise that is embedded in the abstraction. Therefore, it is important to be able to define a descriptor that contains enough discriminative power (but only the specific information we are looking for and no more), is robust to changes and random fluctuations in the input data that we are not interested in, allows the definition of similarity/dissimilarity metrics within different descriptors, and finally, computationally easy to calculate and compare. To this end, Chapter 2 investigates the use of different novel representation schemes suitable for use in machine learning methods. These descriptors focus on the different aspects of the sequence and structural information contained within a protein, from physicochemical features, contact and neighborhood information, relative angles and orientation between the residues and so on. The novel representations we introduce are then tested on two specific biological problems to test whether they are feasible in real world problems of local motif mining. The results of those tests as well as their comparisons with other techniques in the literature are given.

Whereas Chapter 2 focuses on local representations, **Chapter 3** deals with the problem of finding descriptors in a global scale, descriptors that can be used to find the similarities or differences of multiple proteins. Such global features can help on finding similar clusters in data sets by unsupervised learning, or can be used to learn factors that differentiate between two sub-classes of the data. Finding descriptors that are informative even when averaged over the whole protein is a challenging but important task. Chapter 3 describes a variety of known and novel descriptors that use information from a wide range of domains; from coding nucleic acid sequence, amino acid sequence, secondary and tertiary structure, physicochemical data, catalytic and active site information, residue interaction and mechanics/dynamics data, and finally 3D surface patches and hot spots. We use those features for predicting mRNA and protein expression levels from an input sequence and the expression host. This enables us to predict whether the protein will be expressed or not and an approximate level of steady-state protein abundance within the host, the solubility or aggregation of the final gene product, and whether it will correctly fold or be degraded. Using a very comprehensive data set collected from the literature consisting of 19 independent studies from 5 different organisms (both homologous/autologous and heterologous expression), a comprehensive statistical analysis was done on the features, which were further used to build a novel machine learning tool for prediction of protein abundance. The studies resulted in descriptors that explain a significant portion of the variance within the protein levels, some of which are organism-independent. The developed descriptors and the prediction tool can both be used to better understand the inner mechanisms of the cellular machinery. We also show that our prediction tool can help on the identification of the rate limiting step during translation and can be used for codon optimization to increase protein yield in experimental studies.

Chapter 4 describes a motif mining method that can find discriminative gapped short motifs that are highly variable within their composition. Such weak motifs usually cannot be extracted due to variable key elements which are interrupted by long segments of non-specific residues. Therefore, those motifs are can only be found on sequences with elements specific anchor positions, which creates dependance on fixed-length input. This chapter explains the partial periodic pattern mining algorithm, a length-independent and alignment-free motif mining method which can also be used to find discriminative, class-specific motifs. Given a set of sequences, our algorithm will give a list of over-represented motifs (compared to a background set, or in a discriminative manner). These motifs can be used in conjunction with machine learning methods for the prediction of any label or quantitative value that is correlated with the sequence motifs. We apply our algorithm to the MHC class I and II peptide binding prediction problem where the majority of the methods in the literature require a fixed length input. We show that our algorithm outperforms the state-of-the-art methods in different data sets.

Further, the method doesn't require the removal of the unwanted sequences that cannot be used for either training or prediction using the conventional methods.

Chapter 5 introduces a variety of deep learning techniques. Deep learning is an area of machine learning which utilizes a set of hierarchical learners that operate in a sequential fashion. The motivation behind the idea is inspired by the hierarchical architecture of the neocortex in the mammalian brains. This kind of layered, "deep" approach allows learning new representations of the raw input data, which are then fed to the next learner. Since the higher levels use the processed, informative features extracted from the input instead of its raw form, they can perform decision making in a much more abstract level. The recently developed architecture, Deep Belief Networks [11] can also utilize unlabeled data and perform learning in an unsupervised fashion. In the last few years, deep belief networks and similar approaches became the state-of-the-art machine learning methods for image and sound based learning. However, they haven't been applied to the biological problems, and with good reasons. The first problem is, proteins are not of fixed length. In the nature of the classification, fixed length inputs are nearly a universal requirement. Proteins are also highly variable in their composition and structure, and can be either very robust or very fragile to slight changes in their make-up depending on the context, e.g. some proteins can conserve their fold and function despite a great number of mutations, some proteins can retain their overall 3D structure but lose their activity with very few number of mutations, and some proteins will completely misfold even with one mutation. This disproportionate relationship between the input (e.g. primary structure for this case) and the output (3D structure, function, dynamics etc.) makes it very hard to create a non-case-specific machine learning method to find motifs in protein structures.

We propose some workarounds and solutions to some of the problems. Combining the representations from Chapters 2 with our approach, we performed deep learning methods which can be used to classify, cluster, and finally, find discriminative length-independent motifs in any set of input protein data for both sequence and structural representations. Due to the very general approach presented here, our algorithm was not developed for a specific problem or representation. We present its application to a set of very diverse problems to show its feasibility and performance.

In **Chapter 6**, important findings of this thesis are summarized along with remarks for future research topics.

2 LOCAL DESCRIPTORS FOR PROTEINS

2.1 Introduction

Structural studies of proteins for motif mining and other pattern recognition techniques require the abstraction of the structure into simpler elements for robust matching. To compare and contrast between two proteins, or to define reoccurring motifs, it is important to be able to match and quantify the relationships between the sub-units of different proteins. This quantification step requires the simplification of the physical molecule of the protein into more abstract parts. In analysis protein structures, different models of representations on various levels of structural details are used. From coarse-grained to all-atom models, simplified lattice to continuous representations, each model can be used in different areas of research.

The need for abstraction in computational methods (such as structure search and comparison, fold matching, structural motif mining and other areas of pattern recognition) is especially high. The very high amount of data and precision in the 3D coordinates makes computational analysis very complex and very rigid in its applicability. Simplified models capture relevant information and hide unimportant details through abstraction, conferring the ability to group complex 3D information into manageable clusters that can be searched for, compared and "learned" by machine-learning algorithms in a flexible fashion.

The most common abstraction is the primary structure of the protein, defining the molecule as a collection of residues. This simplification allows us to represent the protein as a sequence of an alphabet of 20 amino acids. By defining a quantitative metric for the comparison of such an alphabet, such as the BLOSUM and PAM similarity scores between any two residues, it becomes trivial to check whether two proteins sequences are "equivalent". After this abstraction, we can define complex concepts such as similarity of two sequences and their alignment, database search of a sequence, DNA and protein domain detection/search/prediction, and a great number of biological problems as a mathematical problem that can be solved by a number of algorithms and heuristics. Therefore, it is important to be able to conceptualize the biological molecules into more abstract classes, nominal labels or numerical vectors.

However, this abstraction comes with its own cost; too much abstraction can filter out the information that we want to extract from the raw data. From there on, it becomes a trade-off to optimize the amount of task-relevant information versus irrelevant noise that is embedded in the abstraction. Generally, the simpler the abstraction, the more generalized and noise-resistant it is. While more complex and specific descriptors can include much more information in their representations, the relevant information we are looking for may be lost in the ocean of data, and even if that information can be extracted, it is usually much less robust to slight variations in the input, making it harder to find generalized motifs.

Therefore, it is important to be able to define a descriptor that contains enough discriminative power (but only the specific information we are looking for and no more), is robust to changes and random fluctuations in the input data that we are not interested in, allows the definition of similarity/dissimilarity metrics within different descriptors, and finally, computationally easy to calculate and compare. The descriptor can use any number of information; sequence data, physicochemical properties, secondary or tertiary structure, information on the local neighborhood, dynamics based data, domain information, any number of tagging/modification related information, and so on.

A local descriptor can define any size of a sub-unit. The most commonly used abstraction level is residue based; it defines a nominal label or a feature vector on each and every residue in a protein (or in the case of nucleotide sequences, for every nucleotide). It is also common to use coarse-graining; grouping a number of residues as a single entity. If the coarse-graining is done carefully, this adds the neighborhood information into each sub-unit and can increase the specificity of the descriptor. While fixed size coarse-graining is the most common, adaptive, dynamically-sized units are entirely possible [12, 13]. In the further end, we have all-atom models; each atom, bond and even dynamics data can be embedded into a descriptor of an atom. Due to the sheer number of data and the noise, such embeddings are usually not preferred.

In this chapter, we investigate measures that convey information about the protein. Using these measures, we define a collection of possible protein representations that are suitable for use in machine learning algorithms. Since the exact information contained in a measure is as important as how it's represented, we selected a wide range of measures that can be useful during motif learning or classification tasks.

2.2 Residue-specific Representations

In a residue-specific representation, a collection of descriptors are created for every residue in a protein chain. The relationship between different residues is not taken into account (except for the information contained in the representation itself).

2.2.1 Amino acid Sequence

As we already mentioned, the primary structure of a protein is the most common abstraction. Even though representing them as nominal labels from an alphabet of 20 amino acids are sufficient for many motif mining algorithms, most machine learning methods cannot directly deal with such sequences and require the representation of the sequence by other means.

Binary representation

Direct counterpart of the nominal labels, this approach uses a 20xN matrix of binary values for a sequence of length N. Each residue is represented as a "1" in the corresponding index in the 20-long vector based on their amino acid labels, with the rest of the vector being "0".

Similarity-score representation

While the binary representation can correctly identify each and every amino acid, it requires further knowledge of the similarities between different amino acids during the calculation of similarities for two vectors. Since addition of this meta-information requires (generally not-trivial) modifications to the machine learning algorithm, it is better to embed this knowledge into the data vector itself.

In the similarity-score representation, the sequence is also a $20 \times N$ matrix. However, different than the binary representation, instead of using 1 or 0 values to denote an amino acid, it embeds the 20×1 similarity score between that amino acid and all the other amino acids. Therefore, similar amino acids will have similar feature vectors, whereas the binary representation will penalize any two different amino acids equally.

During our experiments, we use BLOSUM-62 matrix (see Table A.1) with every column normalized within itself into [0, 1] range.

Profile representation

The profile representation is a further step for embedding meta-information into the data itself. To create the $20 \times N$ matrix, we first search the sequence using PSI-BLAST [14] in a predefined database (task-specific or global). The results of the PSI-BLAST search will result in a collection of similar sequences, which are used to create a position-specific scoring matrix (PSSM) from the local alignment of the search results. The PSSM motif is placed into the $20 \times N$ matrix. In case there are no matches to the database query, the resulting PSSM will be equivalent to the BLOSUM scores (and therefore the similarity-score representation). Thus, instead of embedding the static background information about the amino acid similarity, it will actively try to find the general knowledge about the sequence motif in the databases.

Profile representation is very useful when the amount of available data is too low for the machine learning algorithm to ignore the noise in the input set and learn generalizations. However, such generalizations may also reduce the specificity of the input data since the information contained therein is being diluted by the background database. Coupled with the high computational overhead for the PSI-BLAST search, it is generally used if embedding the knowledge contained in the database is likely to increase the performance of the machine learning method.

2.2.2 Secondary Structure

The most common simplified representation of the protein states are the secondary structural assignments to the coordinates, which can be overlaid onto the sequence to create a 1D representation. We used STRIDE [15] to predict and label the secondary structural elements from the 3D protein structure. Secondary structures assigned to protein segments by STRIDE are represented in a 3-class and 7-class fashion. The 7 classes are α helix, 3_{10} helix, π helix, β-sheet, coil, turn and bridge. Those 7 classes can be simplified to 3 classes as "Helix", "Sheet" and "Loop".

The 3-class labels can be represented in a binary matrix similar to mentioned above. 7-class labels can also be represented in a binary form, or using a more suitable score matrix that takes the similarity between the helices into account.

2.2.3 Protein Blocks

While secondary structure is enough for describing many local folds, the simplification can result in losing too much information to abstraction. For example, representing the structure with two states (α-helix and β-sheet) causes the diversity of helices and sheets to be lost, as α-helices are frequently curved (58%) or kinked (17%) [16].

There have been studies with aims to create local structural alphabets to represent the structure as a 1D sequence of structural blocks [17]. A structural alphabet is defined as a set of small prototypes that can approximate each part of the backbone. Creating such an alphabet requires the identification of a set of recurrent blocks that can identify all possible backbone conformations. A commonly used structural alphabet is Protein Blocks (PB) [18], which uses the backbone dihedral angles of the 5 consecutive amino acids (resulting in 10 ϕ and ψ angles). The study by de Brevern found that majority of the protein structures found in the nature can be represented by a combination of only 16 different local 5-residue folds. Using the definition of folds, a 3D structure can be converted into a PB sequence by matching the dihedral angles of 5 residues in a sliding window to match the segment to one of 16 pre-defined blocks by choosing the block that has the lowest angle between the 5 residue unit in question.

The dihedral angles used during the block matching process is given in Table A.3, and a a graphical representation of the 16 protein blocks is shown in Figure A.1.

2.2.4 Other quantitative measures

It is possible to define a number of additional metrics for a given residue or subunit. Although it is possible to use extra information about the residue label itself, such as its hydrophobicity, aromaticity, charge and so on, such additional information is not guaranteed to add increase the information already available in the sequence data itself. That is, motifs and relationships that are based on such measures that utilize look-up values for a residue labels (hydrophobicity etc.) can be extracted from the data itself with enough data points.

For those reasons, it is best to add structural features instead of those that are based on sequence. Some of the features that are used during Chapter 5 are;

- B-factor, a measure of local flexibility of residues within a protein

- Solvent accessible portion of that residue
- Number and type of contacts, H-bonds, salt-bridges for that residue

During our experiments, proteins without known structures were homology modeled using templates and therefore were missing experimental B-factors. For those cases, we predicted the auto-correlation values (mean squared displacements) of the protein by using a Gaussian Network Model [19, 20] to approximate the flexibility of that residue.

2.3 Pairwise representation for Amino acids

While the protein structure can be approximated as a vector of similarity scores for Protein Blocks or any other structural alphabet, this inherently bins the possible values into discrete number of classes, i.e. 16 for PB. The residue-centric view also discards most of the information contained in the relationship of two residues. Addition of such relational information is entirely possible.

For a protein of length N, we can think of the residue-specific representations as a list of N feature vectors, whereas the pairwise representation will be some sort of NxN matrix. The most common example to pairwise representation is the contact map of a protein.

It is possible to use the N^2 feature vectors by themselves; however, the amount of redundant information and noise contained in such a large number of features may hinder the motif mining process. To limit the effects of the non-linear relationship, we can convert the NxN matrix into a NxM matrix, where $M << N$ is a constant denoting the number of consecutive neighbors to include in the feature vector. Thus, instead of using the all-versus-all matrix, we just take a M-wide band along the diagonal.

In both cases, NxN, and NxM, we can define the relationship between any residues in multiple ways.

2.3.1 Label similarity between amino acid pairs

Pairwise score of two residues might be taken as the distance between their labels.

In the case of nominal labels, a similarity measure must be defined beforehand. For example, in the case of amino acid labels, BLOSUM (Table A.1) or any analogous matrix can be used to find the similarity between the two residues. Note that the use of the BLOSUM matrix is not comparable to the similarity-score representation for single residues. In a single residue, we take the similarity of residue i with all of the 20 amino acids and return this 20-feature vector. In pairwise similarity, we look at the similarity of residue i and residue j and repeat this for every consecutive neighbor we wish to take (all j for $|j - i| \leq M$).

A similar approach can be taken when dealing with Protein Blocks instead of the residues. A similarity matrix of the 16 PB elements are defined in Table A.4 [21].

If we are representing the protein with continuous values, any distance metric can be used to find their similarities. Depending on the application, most commonly used alternatives are;

- Minkowski distance measures; Euclidean, Manhattan, Chebyshev etc.
- A Mahalanobis distance metric [22] based on the joint probability distribution of the features

- Correlation of the feature vectors; Pearson, Spearman
- The "angle" between the feature vectors; Cosine-similarity, Tanimoto coefficient

2.3.2 Interaction between amino acid pairs

Distance matrix

The most continuous representation of the residue contacts will be the distances between them, which will give us a matrix filled with all pairwise distances between the central atoms of all residues. To find motifs within directly or semi-directly interacting protein pairs, it is important to put a cap on this distance value; as the distance between two residues grow larger, their contributions to a specific local structural motif diminishes. To capture this non-linear relationship, we use a sigmoidal transformation function on the actual distance. As a final step, we take the negative of the distance (shifted to make the minimum 0), such that the nearest contacts (small distance) get a larger value, and further residues get scores that are lower than the distance ratio between them.

Contact map

We can further simplify the distance matrix by creating a specific cut-off distance, and marking every residue pair as contacting ("1") if the distance between them is less than this cut-off, and "0" otherwise. Contact maps are powerful and very simple to work with; however the sharp cut-off can reduce the performance of the probabilistic machine learning algorithms.

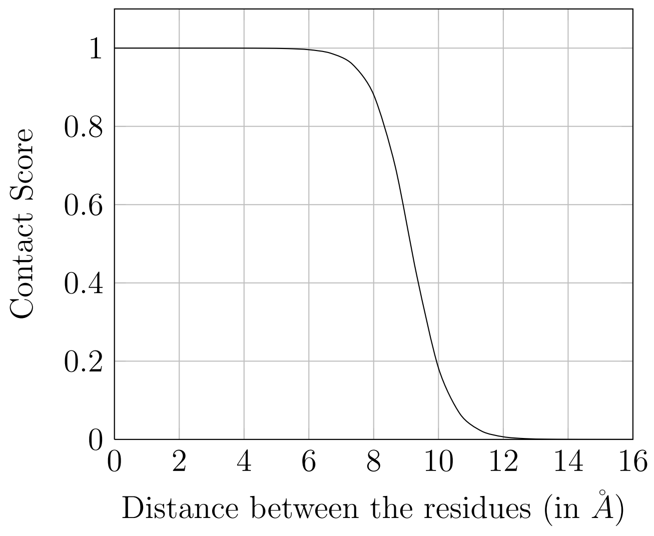


Figure 2.1: The probabilistic contact map function defined in Eq.2.1

Probabilistic contact map

We can mix the continuous nature of the distance matrix and the simplicity of contact maps by defining a new metric. We define a probabilistic contact value between two residues with distance of d between them as:

$$C = \frac{1}{1 + e^{(d \times A - B)}} \tag{2.1}$$

where A and B are scaling factors. We use $A = 1.75$ and $B = 16$ to approximate the probability of two residues "interacting". The resulting function can be seen in Figure 2.1. The function is bounded between 0 and 1, is continuous on the range we expect the residues to be in contact, but returns to 1 or 0 for too small or too large distances. These features make this metric very suitable for Restricted Boltzmann Machines defined in Chapter 5.

Contact potential

Contact potential is a measure of the interaction potential between two residues and is a combination of the previously defined metrics. The BLOSUM similarity scores are based on the interchangeability of two residues in the databases, but it does not take the interaction probability of those residues within themselves. The contact measures look at the distance physically separating two residues, but they lack any information about the label of the residues. However, it is known that the energy pairing between different residues are not equivalent [23]. Contact potential is an energy-like quantity of the interaction potentials between different residues. An example of such measure is the Thomas-Dill contact potential [24], which were calculated statistically from the inter-residue interaction potentials from a protein database. While the Dill contact potentials given in Table A.2 are fixed, the energy potential can be modified based on the distance between the residues as discussed in [24].

Relative orientation

In the protein backbone, the dihedral angles are defined are calculated from 4 consecutive atoms (ϕ: dihedral angle between C'-N-C$_\alpha$-C', ψ: N-C$_\alpha$-C'-N). However, defining the angles between two non-consecutive residues is more problem-prone. We define the relative orientation of residue j with respect to residue i as the angle(s) between them. To make the measure rotationally invariant, we assume residue i to be the origin and define the coordinate system with the "up" vector as the vector (C$_\beta$ - C$_\alpha$), the "right" vector as the vector (C' - C$_\alpha$), and the "back" vector as the cross product of those two vectors. By using this i-centric coordinate space, we find the location of residue j relative to the orientation of residue i by converting the cartesian coordinates of its center atom into the spherical coordinate system and taking the two polar angles. We can opt to take the angle value as signed ($[-\pi, \pi]$) or unsigned ($[0, \pi]$).

It should be noted that this method is not perfect. The "up" and "right" vectors are not exactly perpendicular due to the structure of the amino acids. Another point is, we are looking at the angle for a **point** j. The relative orientation of the sidechain of j is not taken into account; as long as the center stays the same, whether the sidechains point towards each other or not cannot be deduced.

2.4 Graph Properties

Another common approach for structure abstraction is to convert the protein structure into a graph from distance or contact maps. In this representation, each residue is coarse-grained into one center node that is connected to other nodes on the graph on the basis of distance (or other criteria). This allows each amino acid to be represented with its contacts and the topology of the network around it. Representing the structure as a graph allows for sub-graph matching to find reoccurring common motifs in a data set [25] and use of elastic network models for normal mode analysis [26]. Here, we explore the use of the graph theoretical properties that convey information about a residues contacts, local neighborhood, its centrality and importance on the global scale. Such connectivity information can capture the interactions between unconnected (physically separated) residues. We use a large number of conventional measures of network analysis as well as novel, modified indices to better capture the structural information of the proteins.

2.4.1 Graph Properties

2.4.1.1 Contact map

In the creation of the graph from the protein structure, each residue is taken as a node. To connect the nodes between themselves, we first calculate the contact map of the protein and use it as the adjacency matrix. For each residue pair i, j in the protein, we calculate the distance between their central atoms, r_{ij}. If the distance between the two residues are less than a pre-defined cutoff value, we add an edge between the nodes i and j.

In the adjacency matrices, for a given residue pair i, j, A_{ij} value is usually either taken as 0 or 1. However, it is known that the energy pairing between different residues are not equivalent [23]. For this purpose, we are using the interresidue interaction potentials defined by Thomas and Dill [24] which were calculated statistically from a protein database as an energy-like quantity.

The contact potentials matrix contains negative values, which are problematic in the calculation of the shortest paths. As defined in the Shortest Path section, our results do not change for any operation that modify the weights monotonically, therefore to make all of the edge weights positive, contact potentials are shifted as to make the minimum contact potential value 1 between any residue pair. This allows us to define the most favourable negative contact potential as 1, and other less favourable potentials as having greater distance than 1. The shifted contact potentials table is given in Table A.2.

As a result, our adjacency matrix becomes:

$$A_{ij} = \begin{cases} P(Seq[i], Seq[j]) & i \neq j \text{ and } r_{ij} < r_{\text{cutoff}} \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

where $Seq[i]$ is the i^{th} amino acid, $P(X, Y)$ is the shifted Dill contact potential between the amino acids X and Y, and r_{ij} is the distance between the center points of the i^{th} and j^{th} residues.

Center of a residue is usually taken as the coordinate of C_α or C_β atom of that residue. Even though C_α distances are commonly used in the literature, they can not differentiate between the cases where the sidechains are oriented away from each other and another case where sidechains face each other (loosely defined as an "interaction" for our case). For our experiment, we are using C_β atoms as the center, since they can capture the relative orientation and interaction of the side chains better than the C_α atoms. Glycine residues, which lack the C_β atoms are represented as their C_α coordinates. The cutoff distance r_{cutoff} can span a wide range depending on the problem. For our experiments, the cutoff distance is taken as 8 Å unless noted otherwise.

2.4.1.2 Shortest Path

The total weight of a path is the sum of the weights of the links along the path. The "optimal path" between a pair of nodes is the path of minimal total weight between these vertices. While we defined the edges between our nodes (amino acids) with the Thomas-Dill contact potentials, due to the different interaction types within a macromolecule, we use three different path distance measures and find the shortest paths between nodes with respect to them.

If a path between the vertices v to v' is P, and $P = (v_1, v_2, \ldots, v_n)$ where $v_1 = v$ and $v_n = v'$, then the weight of P will be $|P|$, which can be defined three different ways as below. The shortest path between v and v' will be the P over all possible paths that minimize $|P|$.

Homogeneous Shortest Path

If the weights in the network are drawn from a relatively narrow distribution, the weight of a path will be closely related to its hop number, since every edge will contribute a similar weight. In that case, the length of the optimal path will be proportional to the length of the shortest path. To simulate this behaviour, the network can be constructed without edge weights, in which the shortest path will be the path with the minimum number of hops.

$$|P_h| = n - 1 \tag{2.3}$$

For our implementation, homogeneous paths are calculated using Dijkstra's algorithm [27] using no edge weights.

Weak Disorder Shortest Path

The weak disorder case is similar to the homogeneous path such that the edge weights come from a narrow distribution. However, we no longer restrict ourselves to a 0, 1 weight scheme and take the actual edge weights (contact potentials in our case). The weak-disorder weight for P will be:

$$|P_w| = \sum_{i=1}^{n-1} e_{i,i+1} \tag{2.4}$$

where $e_{i,j}$ is the edge weight between the nodes i and j.

For our implementation, weak disorder paths are calculated using Dijkstra's algorithm using the contact potentials as described above.

Strong Disorder Shortest Path

If the distribution of weights is broad enough, the total weight of a path will be determined by the highest weight along the path, and is almost independent of all other weights. In this case, paths can be compared by the highest weight on them. This case is called strong disorder. Shortest path in a strong disorder graph can be interpreted as the shortest path that causes minimal maximum disturbance along the path, i.e. the path that minimizes the maximum edge weight that is crossed.

In strong disorder networks, the optimal path is very different compared to weak disorder. The optimal paths will attempt to avoid high weight links whenever possible. Thus, the optimal path may travel a longer total distance to avoid passing through high weight edges. Atilgan et al. [28] found that the interactions in biological networks can be more accurately defined by their strong-disorder distances, which emphasize the largest barrier to be crossed along the way and define the bottleneck residue.

The strong disorder weight for P is defined as:

$$|P_s| = max(e_{i,i+1} \text{ for } i = 1 \text{ to } n - 1) \tag{2.5}$$

If different paths share the highest weight edge, the lower weight path can be either determined by comparing the highest weight between the non-shared links, or by taking the one with the lowest hops. We opted to use the latter definition.

Strong disorder paths are calculated using two passes of Dijkstra's algorithm; the first will determine the maximum edge weight that needs to be traversed to reach v' from v, and the second will find the path with the minimum number of hops that do not exceed the given maximum weight.

2.4.1.3 Centrality Measures

Centrality measures determine the relative importance of a vertex within the graph [29, 30, 31, 32]. These measures can quantify the key nodes which are important in information flow/propagation within the network, or the bottleneck vertices that may be improved. Due to the projection of the entire global network topology to an easily quantifiable value for each node, they can be used to map the important nodes of different graphs in subgraph matching heuristics.

The centrality measures, with the exception of Degree and 2^{nd} Degree, depend on the shortest path calculations. For those reasons, they are calculated independently for both weak- and strong- disorder shortest paths.

Degree Centrality

Degree centrality is defined as the number of neighbors of a node. In directed graphs, it can be separated into two separate measures for indegree and outdegree. For undirected graphs, they are equivalent.

Degree centrality of a node i is defined as:

$$C_D(i) = \deg(i) \tag{2.6}$$

Second Degree Centrality

A more generalized version of the degree centrality is the second degree, which measures the number of distinct second neighbors of a node. that is, nodes that can be reached from i in 2 hops.

Closeness

For a graph, there is a distance metric between all pairs of nodes, defined by the length of their shortest paths. The farness of a node i is defined as the sum of its distances to all other nodes, and its closeness is defined as the inverse of the farness. Thus, central nodes will have a lower distance to all other nodes in the network. Closeness can be regarded as a measure of how fast it will take to propagate information/perturbation from node i to all other nodes sequentially.

Closeness can be defined as:

$$C_C(i) = \frac{1}{\sum_{j=1}^{N} d(i,j)} \tag{2.7}$$

where $d(i,j)$ is the shortest distance between the nodes i and j. $C_C(i)$ can also be normalized as:

$$C'_C(i) = C_C(i)(n-1) \tag{2.8}$$

Notice that closeness centrality for all the nodes will be 0 for an unconnected graph. To prevent this behavior, only the distances between the reachable nodes are taken. However, since our protein networks are calculated in a way to disallow unconnected graphs, this correction was not necessary.

Betweenness

Betweenness centrality quantifies the number of times a node acts as a bridge along the shortest path between two other nodes. It can be said that nodes with high betweenness values are critical in preserving the flow of the network and can act as bottlenecks.

Betweenness centrality of a node i is the sum of the fractions of all the shortest paths for each pair of nodes (j,k) that pass through that node i. To find the betweenness of a node i, shortest path (i.e. geodesic) between all pairs of nodes (j,k) are calculated, denoted as g_{jk}. From those paths, the number of geodesics that pass through i are counted to find $g_{jk}(i)$. Then, the betweenness centrality $C_B(i)$ of a node i becomes:

$$C_B(i) = \sum_{\substack{j<k \\ i \neq j \neq k}} \frac{g_{jk}(i)}{g_{jk}} \tag{2.9}$$

It is usually normalized as:

$$C'_B(i) = \frac{C_B(i)}{\frac{(n-1)(n-2)}{2}} \tag{2.10}$$

Distinct Betweenness and Non-redundant Betweenness

We define two further measures of betweenness. Distinct betweenness is a normalized version of betweenness. It is the ratio of the number of distinct occurrences of a node on the total number of distinct shortest paths between all pairs of nodes. If there are multiple shortest paths between a given pair of nodes, only the first occurrence of a node i is counted towards its betweenness.

$$C_{B_D}(i) = \frac{\sum\limits_{\substack{j<k \\ i \neq j \neq k}} min(g_{jk}(i), 1)}{\frac{(n-1)(n-2)}{2}} \tag{2.11}$$

We define another special case of distinct betweenness, non-redundant betweenness as the number of node pairs j, k that will be affected by the removal of node i. That is, if all shortest paths between the pairs j, k pass through i, then i acts as a bottleneck and there is no redundancy in the case i is removed from the network; the shortest path distance will increase for j and k.

$$C_{B_{nr}}(i) = \frac{\sum\limits_{\substack{j<k \\ i \neq j \neq k}} \begin{cases} 1 & g_{jk}(i) = g_{jk} \\ 0 & otherwise \end{cases}}{\frac{(n-1)(n-2)}{2}} \tag{2.12}$$

Stress Centrality

Stress centrality of a node is the fraction of all shortest paths that pass through that node. Note that this is different than betweenness; betweenness is the sum (for all pairs) of the fraction of the shortest paths between a given pair that pass through a node.

Stress centrality is defined as:

$$C_S(i) = \frac{\sum_{i \neq j \neq k} g_{jk}(i)}{\sum_{i \neq j \neq k} g_{jk}} \tag{2.13}$$

2.4.1.4 Cliques

In graph theory, a clique in an undirected graph is a subset of its nodes such that every pair of nodes in the subset are connected by an edge. In small-world networks, some nodes tend to create groups with relatively higher density of connection within themselves, compared to the probability of establishing an edge between two random nodes. In such networks, cliques or near-cliques give information about tightly connected groups within the graph.

A measure of local cliquishness is the clustering coefficient. Clustering coefficient is a measure of degree to which nodes in a graph tend to cluster together and quantifies a node on how close its neighbors are to being a clique. For an undirected graph $G = (V, E)$, the neighbourhood N_i for a vertex $v_i \in V$ is defined as its immediately connected neighbours. The clustering coefficient C_i for a vertex v_i is given by the proportion of links between the vertices within its neighbourhood divided by the number of links that could possibly exist between them. Thus;

$$N_i = \{v_j : e_{ij} \in E\} \tag{2.14}$$

$$C_i = \frac{|\{e_{jk} : v_j, v_k \in N_i, e_{jk} \in E\}|}{k_i(k_i - 1)} \tag{2.15}$$

2.4.2 Multiple Sequence Alignment using Graph Properties

The feasibility and the performance of those graph properties are tested by using them in a biological problem of multiple structural alignment. Finding the similarities between proteins has been one of the important issues in biological sciences. Multiple sequence alignment allows the similar and conserved proteins to be aligned within each other, essentially defining a motif. While the conventional multiple alignment algorithms deal with sequence information, structural similarity is much more important since the function of a protein follows its shape. Therefore, the ability to extract the structural homology between proteins even in the case of very low sequence identity is important for discovering class-specific motifs.

We propose the use of the graph properties to obtain a multiple alignment based on structural data. This approach differs from the structural superposition algorithms in the literature: Superposition algorithms try to align the 3D structures by rotating and translating the structures to obtain a suitable overlay betweem them, whereas our method will produce a sequence alignment using the structural information.

The overall idea is to calculate the graph properties for every residue. We can measure the similarity between different residues by applying a distance function on the graph properties, essentially replacing the sequence similarity matrices such as BLOSUM during the multiple alignment algorithm. Using our metric, we performed multiple structural alignment to capture over-represented structural folds that occur in a set of proteins. The results are compared within the literature and show that the graph properties can capture the global information about a residue better than the conventional methods.

2.4.2.1 Alignment algorithm

Alignment with affine gap penalty

During the pairwise alignment of two proteins, S and T, each protein is represented as a sequence where each node contains the graph properties as their inherent labels.

In order to account for affine gap penalties (to differentiate between first position of a gap and further gap extensions), three distinct matrices of size $[nxm]$ are created where n and m are the sequence lengths. The matrices A, B and C are defined as:

$A[i, j]$ = max. score of an alignment between $S[1 \ldots i]$ and $T[1 \ldots j]$ that ends in $S[i]$ matched with $T[j]$.

$B[i, j]$ = max. score of an alignment between $S[1 \ldots i]$ and $T[1 \ldots j]$ that ends in a gap matched with $T[j]$.

$C[i, j]$ = max. score of an alignment between $S[1 \ldots i]$ and $T[1 \ldots j]$ that ends in $S[i]$ matched with a gap.

These matrices are initialized as follows:

$$\begin{aligned}
A[0, 0] &= 0 & \\
A[i, 0] &= -\infty, & \text{for } 1 \leq i \leq m \\
A[0, j] &= -\infty, & \text{for } 1 \leq j \leq n \\
B[i, 0] &= -\infty, & \text{for } 0 \leq i \leq m \\
B[0, j] &= G_{\text{open}} + j.G_{\text{ext}}, & \text{for } 1 \leq j \leq n \\
C[i, 0] &= G_{\text{open}} + j.G_{\text{ext}}, & \text{for } 0 \leq i \leq m \\
C[0, j] &= -\infty, & \text{for } 1 \leq j \leq n
\end{aligned}$$

where G_{open} is the gap opening penalty and G_{ext} is the gap extension penalty for affine gap penalty function.

Then, the matrices A, B and C are filled for all $1 \leq i \leq m$ and $1 \leq j \leq n$ by a dynamic programming algorithm, defined by the following equations:

$$A[i,j] = p(i,j) + max \begin{cases} A[i-1,j-1] \\ B[i-1,j-1] \\ C[i-1,j-1] \end{cases} \tag{2.16}$$

$$B[i,j] = G_{\text{ext}} \quad + max \begin{cases} G_{\text{open}} & +A[i,j-1] \\ 0 & +B[i,j-1] \\ G_{\text{open}} & +C[i,j-1] \end{cases} \tag{2.17}$$

$$C[i,j] = G_{\text{ext}} \quad + max \begin{cases} G_{\text{open}} & +A[i,j-1] \\ 0 & +C[i,j-1] \\ G_{\text{open}} & +B[i,j-1] \end{cases} \tag{2.18}$$

where $p(i,j)$ is the score of a matching i^{th} and j^{th} residues, calculated by the node similarity scoring function that is defined in the following section.

To get the final result, the optimal alignment is constructed by tracing back from the maximum among the matrices $A[m,n]$, $B[m,n]$ and $C[m,n]$, from (m,n) to $(0,0)$. During back tracing, each previous step is the same one followed during the selection of the maximum scoring path. This will give us the both the optimal alignment between S and T, and their alignment score.

Pairwise Match Scoring

We define $p(Z,X)$, the score of aligning a residue Z with another residue X as;

$$p(Z,X) = \sqrt{\sum_{i=1}^{N} w_i (Z_i - X_i)^2} \tag{2.19}$$

where Z_i denotes the i^{th} graph property of Z, and w_i is the weight vector to control the importance of each property to the similarity.

To optimize w_i, we use a genetic algorithm. The fitness function of the genetic algorithm is taken as the SP-score of alignments our method produces and the reference alignments set. As the reference, we opted on the curated benchmark set of structure alignments from CATH [33] for 3 families.

Guide tree construction

Using the alignment algorithm described above, all proteins are aligned with each other. This results in a similarity matrix of all proteins, where the similarity is the obtained pairwise alignment score. Using these scores, we create a hierarchical clustering tree using the neighbor joining algorithm [34]. We first create the Q matrix:

$$Q(i,j) = (r-2)d(i,j) - \sum_{k=1}^{r} d(i,k) - \sum_{k=1}^{r} d(j,k) \tag{2.20}$$

where $d(i,j)$ is the distance between two proteins. Then the minimum element of the Q matrix, $Q(f,g)$ is found. This means that the element pair (f,g) will be clustered together to create the new element u which will be added to the tree. The distances between the newly created u node and other nodes are calculated as:

$$d(f,u) = \frac{1}{2}d(f,g) + \frac{1}{2(r-2)}\left[\sum_{k=1}^{r} d(f,k) - \sum_{k=1}^{r} d(g,k)\right] \tag{2.21}$$

$$d(g,u) = d(f,g) - d(f,u) \tag{2.22}$$

$$d(u,k) = \frac{1}{2}[d(f,k) + d(g,k) - d(f,g)] \text{ for all } k \neq f,g,u \tag{2.23}$$

Afterwards, the new Q matrix is calculated (with the addition of u and removal of f, g), and the process is repeated until no more nodes are left.

Multiple Alignment

Using the order and the topology of the guide tree, the multiple alignment of sequences are created in a hierarchical manner. The lowest layer of the tree will contain single proteins that will be aligned in a pairwise manner as shown above. The resulting alignment will be added to the tree as the parent element of the two proteins. When aligning an element that contains multiple proteins with another element, the similarity score between aligning the multiple elements are calculated by sum-of-pairs scoring (SP-score). SP-score of aligning the i^{th} position of an element A with the j^{th} position of an element B is the sum of the similarity scores between all proteins of A_i and B_j.

This process is continued by merging the alignments of different proteins until all of the proteins are aligned.

2.4.2.2 Data set

Use of an objective and reliable benchmark is needed to better understand the weak points and problems associated with a proposed alignment method. BALIBASE [35, 36] is the most commonly used alignment benchmark set for sequence alignment. It contains 217 reference alignments with 6255 sequences. Balibase focuses on "subfamily specific features, motifs in disordered regions, the effect of fragmentary or otherwise erroneous sequences on MSA quality" [36]. The reference alignments are manually curated based on 3D structural superpositions with detailed annotations.

Sequences in BALIBASE provide challenging test cases for both global and local alignment programs, as not only all of the alignments are provided with full-length sequences, 168 of them have one additional version where sequences are trimmed so that the focus is more on homologous regions. BALIBASE typically focuses on sequences with lower identities. Since sequences with higher identities are easier to align, low identity sequences -so called twilight zone sequences- are much more informative when assessing the performance of an alignment.

We used the 67 families given in reference set 1.0 of BALIBASE.

Evaluation Metric

For all of the data sets, a reference alignment is given. We compare the graph-based alignment results with the reference alignment using either sum-of-pairs score (SPS), or a column score (CS) [37]. In this context, SPS compares the alignments by checking the fraction of elements that are aligned correctly in the test alignment compared to the reference alignment. For all pairwise matches between the residues that are present in a specific position in the reference set, SPS will check whether that residue pair is aligned together in the test set (in any position). If they are aligned together, the score is incremented by 1. The final score will be divided by the total number of pairs, thus resulting in a value between 0.0 and 1.0, where 1.0 means perfect replication of the reference alignment.

CS follows a similar route. Only, instead of looking at all the specific residue pairs for a position, it considers the whole column at that position. If the residues present in a specific position are also aligned as a column in the test set (in any position), the column is considered to be aligned. For the whole alignment, CS will give the fraction of columns that are conserved between the two alignments.

SPS is beneficial when determining the percentage of correctly aligned residues for each individual sequence in the alignment. CS measures the performance of the algorithm in correctly aligning all sequences together.

BALIBASE database also includes the core columns for a family of alignments. Core columns are the columns where a specific domain, fold or any conserved important region is located. Thus, the core segments depict which columns are described as being reliably aligned, whereas other regions are annotated as "ambiguous". Therefore, the optimal aim of an alignment algorithm should be to align those core regions. Two other metrics, SPS_{core} and CS_{core} are defined to check the performance of an alignment on those core regions. Instead of looking at the whole of the alignment, Core-SP and -CS will compare only the residues present in the columns marked as "core" in the data and discard the rest of the alignment.

2.4.2.3 Results

The comparison of the average scores over the whole 67 families with the results for other alignment algorithms is given in Table 2.1 [38, 39]. Individual results for the given reference families selected in BALIBASE are given in Table 2.2.

Looking at Table 2.1, we can see that our alignment performs quite well when compared to the state-of-the-art algorithms. GraphAlign ranks 4th by SPS and 3rd by CS out of the 10 algorithms compared. It is important to note that our alignment algorithm is the same one as used by CLUSTAL [40], without the sequence weighting and gap optimization techniques of CLUSTALW2 [41]. Using the same alignment algorithm but scoring with graph properties instead of the sequence information outperforms CLUSTAL in both SPS and CS. This is an important distinction; the state-of-the-art MAFFT [42] evaluates a much more complicated scoring scheme with a very high complexity and running time. However, since our scoring method is not based on the alignment algorithm, we can implement graph property scoring into the main idea of MAFFT to utilize its powers in sequence alignment, with our strengths in capturing structural similarity.

A detailed view of individual scores for 12 families are given at Table 2.2. We can see a great variance within the results. 4/12 of the families were predicted with very high CS and SP scores including one family with a perfect alignment, another 3/12 of the families were aligned with about CS of 0.4 and SPS of 0.65. However, the rest 5/12 of the families are very problematic with regards to their column scores of 0 even though they have an average SP-score of 0.41. This means that, even though a relatively high percent of the residues were aligned correctly, none of the families have any columns with elements aligned perfectly in a row.

It should be noted that the reference alignments in the BALIBASE were criticized by several studies [43, 37] sequence alignment based on the BAliBASE benchmark]. The studies show that there are many problems in the reference alignments; Many sets align regions that are definitively not homologous or structurally similar, even in families that contain structural similarity, the structural alignments used by BALIBASE are wrong. Another big problem is that the core blocks should have unambiguously conserved secondary structure to ensure that structure and sequence alignments agree, however for many cases they don't. And finally, many sets are not globally alignable because they contain different domains in different orders. Often, only a single, small domain is common to all proteins in the input set.

Therefore, the problems with the families resulting in a column score of $\tilde{0}$ might either result from our method, or the reference alignment itself. It has been reported that most alignment algorithms such as MUSCLE, T-COFFEE, PROB-

CONS etc. also face problems with BALIBASE families [43]. It is possible that some families also result in significantly low column scores using other methods as well. However, we cannot test this claim since the benchmark studies do not report their findings in a family specific manner.

Table 2.1: Comparison of the MSA results to other alignment algorithms.

Algorithm	SP_{core}	CS_{core}
DIALIGN-TX	0.515	0.265
DIALIGN-T 0.2.2	0.493	0.253
DIALIGN 2.2	0.507	0.265
CLUSTAL W2	0.501	0.227
T-COFFEE 5.56	0.582	0.313
POA V2	0.38	0.153
MAFFT	0.671	0.446
MUSCLE 3.7	0.579	0.33
PROBCONS 1.12	0.67	0.417
GraphAlign	0.55	0.33

2.4.2.4 Discussion

This section explores the feasibility of using network information in capturing the local information of proteins. We propose the use of a variety conventional and novel graph theoretical measures as a similarity metric between local sub-units of proteins. We then use these descriptors in a multiple sequence and structural alignment method.

From the results of our experiments, we show that such descriptors are viable and can give comparable results even though the actual 3D structural information is converted into a more abstract form by the graph properties. An important

Table 2.2: Comparison of the multiple sequence alignment results with the benchmark alignments.

Data set	#	CS_{core}	SP
BB11001	10	1.00	1.00
BB11002	10	0	0.422
BB11003	10	0.45	0.78
BB11017	10	0.83	0.895
BB11020	10	0	0.704
BB11021	10	0.4	0.582
BB11022	10	0	0.318
BB11027	10	0.32	0.592
BB11028	10	0	0.623
BB11030	10	0.02	0.439
BB12020	10	0.98	0.99
BB12036	10	0.91	0.958

advantage of the descriptors presented here is the ability to obtain alignments that capture the similarities between the folds and domains of multiple sequences even in the absence of sequence similarity. It is known that structural information is mostly conserved through the evolutionary processes, or reach similar structures due to convergent evolution. This results in a variety of protein families that have very low sequence similarity but high structural similarity. Instead of using the amino acid label, we can add the structural information to perform better alignments.

Structural alignment of multiple proteins is problematic in the case of proteins that are dissimilar in overall shape but carry a common similar domain. This is an important problem, but a problem that has already been tackled and remedied in the case of multiple sequence alignment algorithms [39]. A very important part to focus is that our descriptors are mainly a similarity metric; the features do not depend on a specific implementation. Even though we used the alignment approach used by CLUSTAL, it is trivial to implement the distance matrix described here into any other sequence alignment algorithm. Since our results improve upon the classical CLUSTAL, we speculate that our results will improve the baseline of performance if applied to a better alignment technique.

2.5 Bond-Orientational Order Parameters

The previous section has shown the power of graph-based measures in capturing the global information about a protein. Even though this simplification is very powerful, it has a distinct disadvantage: the loss of the 3D topology. By definition, the graph construction will project the 3D structure into a flat presentation that only contains the distance information. That is, as long as the 2+ points of a 3+ point sytem are close together to form a contact, the relative topology and the orientation of the non-contacting points relative to the others is lost. It can be argued that the contact information, when analyzed in a global scale, contains enough implicit information to approximate topology, as shown by the algorithms that can correctly predict a 3D structure based only on the contact map [44, 45]. However, mining this implicit information require the analysis of the graph as a whole, i.e. use of the contact information of a residue as a constraint for another, irrelevant residue. Since the machine learning algorithm will operate on a local scale, the relative orientation information between the residues cannot be captured.

As a result, the graph properties contain information about the centrality in a global scale, but they lose the structural fidelity. And in contrast, while the structural alphabet approaches (such as Protein Blocks) can capture the local geometry of a fold specifically, they lose the position and the locale of the residue in the global scale of the protein. To approximate both the local structural information with a relatively high degree of certainty and the neighborhood information including the directionality of those contacts with a single model, we propose the use of bond-orientational order parameters as local descriptors of proteins.

Bond-orientational order is a well-established metric that is used in analysis and comparison of the crystal structures packing of atoms [46]. Due to the use of spherical harmonics, they can capture the directional information around each residue, and since they are invariant of the rotations of the reference frame, matching two structures require only the comparison of numbers, instead of the more computationally costly and problem-prone structural alignment methods.

As a first step, we wanted to test whether the number and placement (angle and distance) of neighboring atoms around each residue show a repeating pattern in average protein structures. If there is such a pattern, we can use the protein descriptors to approximate the local structure around a center point. To test the feasibility of representing the protein structure with such orientational order descriptors, we tried to use those descriptors to capture and differentiate the secondary structural elements from each other. Recognizing and assigning secondary structures to atomic coordinates is a complex task [47] and require the ability to recognize both the local structure (for helices) and contact information (between β strands). If orientational order descriptors can predict secondary structural elements, it shows that they capture the necessary information and can be evaluated further for more complex motif discovery purposes.

2.5.1 Method

2.5.1.1 Bond-orientational Order

The bond-orientational order parameter is previously described by Steinhardt et al. [46] in the study of packed spheres. It has also been employed in the analysis of protein structures by means of local connectivity around each residue [48]. The bond-orientational order parameters are given as

$$\bar{Q}_{\ell m}(i) = \frac{1}{N_b(i)} \sum_{n=1}^{N_b(i)} Y_{\ell m}\left[\theta(\vec{r}_n - \vec{r}_i), \phi(\vec{r}_n - \vec{r}_i)\right] \tag{2.24}$$

$$Q_\ell(i) = \left(\frac{4\pi}{2\ell+1} \sum_{m=-\ell}^{1} \left|\bar{Q}_{\ell m}(i)\right|^2 \right)^{1/2} \tag{2.25}$$

$$W_\ell(i) = \sum_{\substack{m_1,m_2,m_3 \\ m_1+m_2+m_3=0}} \begin{bmatrix} \ell & \ell & \ell \\ m_1 & m_2 & m_3 \end{bmatrix} \bar{Q}_{\ell m1}\bar{Q}_{\ell m2}\bar{Q}_{\ell m3} \tag{2.26}$$

where ℓ is the bond orientational parameter. $\vec{r}_n$ denotes the position vector of the n^{th} residue. $(\vec{r}_n - \vec{r}_i)$ is the bond vector from residue i to n, and θ, ϕ are the polar angles of this bond, measured with respect to an arbitrary reference frame. $Y_{lm}[\theta(\vec{r}_n - \vec{r}_i), \phi(\vec{r}_n - \vec{r}_i)]$ are Laplaces spherical harmonic functions [49] for the given angles. $N_b(i)$ is the total number of contacts of i that are below a given cutoff distance. The coefficients shown as a matrix in Equation 2.26 are the Wigner-3-j symbols [50].

While the spherical harmonics of the bonds for a given l can change drastically by rotating the coordinate system, combining the $Q_{\ell m}$ values into a quadratic invariant Q_ℓ (Equation 2.25) and third-order invariant W_ℓ (Equation 2.26) will result in a rotationally invariant parameter. These order parameters are invariant under reorientations of the external coordinate system. For $l = 2n$, spherical harmonics are also invariant under inversion and therefore independent of reference frame.

In research of the crystal packing, most commonly used parameter is the Q_6 [46, 51, 52] as $\ell = 6$ is the smallest value of ℓ that can capture both cubic (simple, face centered, and body centered) and icosahedral orders (whereas Q_4 will miss icosahedral and Q_2 will miss both) [48].

2.5.1.2 Data set

For experimentation, a total of 120 protein structures were collected from the Protein Data Bank [53]. Protein structures belonging to different SCOP [54] classes and folds were selected for more even representation of different folds in the dataset. On top of those, the benchmark set of non-homologous (<30% sequence identity) PDB proteins of Zhang et al. were also added to the final dataset [55].

For each residue, Q_ℓ (Equation 2.25) and W_ℓ (Equation 2.26) values are calculated from the contacts of that residue, where contact is defined as residues with distance between the Cα atoms that is less than a predefined cutoff threshold. During the calculation, different cutoff distances and l values were tried. Resulting $Q_{\ell m}$ and W_ℓ values were merged in a feature vector by using sliding window on the backbone.

The secondary structure of each protein was calculated using STRIDE [15]. The secondary structure values of the windows were assigned as a class value on the basis of occurring in the majority of the segment ($> 60\%$) in a continuous fashion in the sliding window. The transition regions between different secondary structures that contain two or more different secondary structure classes in the protein segment were removed from the dataset since there is no clear secondary structure to be used in learning and prediction. After those removals, extracting the features from the 120 proteins (using a window size of 5) results in 15273 rows (protein segments) in the final dataset.

2.5.1.3 Secondary structure prediction

Secondary structures assigned to protein segments by STRIDE [15] are represented in a 3-class and 7-class fashion. The 7 classes are α helix, 3_{10} helix, π helix, β-sheet, coil, turn and bridge. Those 7 classes were simplified to 3 classes as Helix, Sheet and Loop. The final dataset was created using both 3-class and 7-class representations. However, in the resulting dataset the classes bridge, 3_{10} helix and π helix had only few copies, as either they are uncommon or are rarely found consecutively. Also, STRIDE is believed to underpredict π helices [56], possibly lowering their count even further. Due to very low sample size, 3_{10} helix and π helix classes were merged with the α helix class, and bridge regions were removed completely, resulting in a 4-class (helix, sheet, coil, turn) data.

In the feature vector, Q_ℓ values always result in a value between 0 and 1, while W_ℓ values can take arbitrary values. To overcome this, W_l values were normalized to the $[0-1]$ range before the prediction.

Using the calculated Q_ℓ and normalized W_ℓ values from the sliding windows as the feature vector, and assigned secondary structure as the class value (for both 3-class and 4-class), a classification was performed using the SVM implementation libsvm [57] inside the Orange data mining software [58].

Optimization of the window size, l-values and the cutoff distance was carried out on a smaller independent set consisting of 15 proteins. The optimal results were obtained using a cutoff of 7 Å in conjunction with $\ell = 2$ to 10, with a window size of 5.

Training and prediction was done on separate data sets, created from independent proteins (i.e. no protein segment was predicted with a classifier that was trained with a segment belonging to the same protein). The data was split in a

50-50% fashion (of the PDBs) to create the training and the testing sets.

2.5.2 Results

2.5.2.1 Prediction Results

The accuracy and the AUC (area under the receiver-operating-characteristic curve) of the predictions of the test set are given in 2.3. Accuracy of the prediction is 92.3% and the AUC is 0.993. AUC gives the probability that a randomly selected positive instance will score higher than a random negative instance, and is a more robust performance measure than accuracy itself [59].

Looking at the confidence table, helices (sensitivity of 0.99) can be represented exceptionally well by the bond-orientational order parameters, followed by sheet structures (sensitivity of 0.91). In 4-class representation, coils and turns have lower sensitivity (respectively 0.71 and 0.75). However, as can be expected, they are more likely to be mistaken as each other than a sheet or helix. In 3-class representation, assigning the class value of loop-region to coils and turns will result in a significantly higher sensitivity of 0.87.

2.5.2.2 Feature Analysis and Clustering

Due to the very high accuracy and AUC values, we investigated whether the high accuracy was because of the high predictive performance of SVM due to the use of non-linear kernels, or whether the accuracy could be replicated with a simple, human-understandable method.

We first investigated the effects of different Q_ℓ and W_ℓ features for each residue in the segment to the corresponding secondary structure. To see the importance

Table 2.3:

Area under the ROC curve, accuracy and confusion matrix of the test set predictions. In the confusion matrix, number of predicted instances and ratio of the correct predictions are given.The last row (C+T) represents Coil and Turns being classified as Loop-region in the 3-class prediction.

AUC	**Accuracy**
0.993	**92.3%**

		Predicted				
		Helix	**Sheet**	**Coil**	**Turn**	**Sensitivity**
Actual	**Helix**	3768 (98.9%)	12 (0.3%)	0 (0.0%)	26 (0.7%)	0.990
	Sheet	3 (0.2%)	1199 (90.70%)	12 (0.9%)	105 (7.9%)	0.907
	Coil	3 (0.7%)	27 (6.1%)	316 (71.0%)	99 (22.2%)	0.710
	Turn	71 (10.1%)	43 (6.1%)	48 (6.9%)	527 (75.3%)	0.753
	C+T	74 (6.5%)	70 (6.1%)	990 (87.3%)		0.873

of each feature and a visual representation of their relationship with samples, we created a 2D linear projection [60] of the data using 6 features, selected by running the VizRank heuristic [61] for 2000 generations on the training set. The rotation of the axes and the final projection was optimized using the FreeViz algorithm [62] to optimize separation of data points. The result is given in Figure 2.2. From the perspective of the Q_3 and W_3 parameters, sheets and coils form the opposing ends of the spectrum. Notice that the classes show a non-perfect but distinct separation even on a linear projection.

To further investigate the quantitative importance of each feature to the prediction, we looked at the information gain and the linear SVM weight of the features. The features that have the highest information gain are the Q_3 and Q_4 values for the middle 3 residues of the window of 5 aminoacids. When ranked by their SVM weights, Q_9 values of the middle 3 residues were also selected as well as the Q4 values. Not surprisingly, the center portion of the window was ranked higher than the boundary portions. No W_ℓ values were selected as informative. We can conclude that Q_3, Q_4 and Q_9 are the most important features for classification, since they were all selected at least 3 times for that center portion without exception.

Using the top 6 features from the SVM weights (Q_4 and Q_9 for the 3 center residues of each window), we performed unsupervised k-means clustering on the dataset. The distance between each row was calculated as the distance between their vectors. Euclidean, Manhattan, Hamming distances and Pearson and Spearman correlation values were tried during the clustering. The optimal distance measure was found to be the Manhattan distance. Results for clustering with $k = 6$ in k-means algorithm are given in Figure 2.3 and Table 2.4. Figure 2.3 shows the frequency of the secondary structural elements in the resulting clusters,

Table 2.4:
Relative assignment of each class to the clusters. Cluster representations show which class is more likely to be in that cluster.

	# Helix	# Sheet	# Turn	# Coil	Representation
Cluster 1	98.3%	1.1%	0.5%	0.2%	Helix
Cluster 2	90.4%	4.1%	3.8%	1.7%	Helix
Cluster 3	90.1%	5.1%	3.1%	1.7%	Helix
Cluster 4	19.7%	68.9%	8.1%	3.3%	Sheet
Cluster 5	5.8%	20.9%	47.3%	26.0%	Loop region ~ Turn
Cluster 6	0.0%	0.7%	35.9%	63.4%	Loop region ~ Coil

Clustering Accuracy	84.6%
AUC	0.932

and Table 2.4 gives the clustering accuracy and relative assignments of each class to each cluster.

As we can see, even after discretizing the feature vectors to only 6 clusters with an unsupervised method, the clustering has 84.6% accuracy and 0.932 AUC. The clusters show relatively high sensitivity. That is, clusters 1,2 and 3 can represent helix structures with high certainty, cluster 4 is mostly sheet structures and the cluster 5, 6 is commonly loop regions, with most of the errors are due to misclassifying Turns as Coils and vice versa.

2.5.3 Discussion

In our study, we tested the feasibility of using bond-orientational order parameters as descriptors of protein structure in predicting secondary structure from the coordinates Cα atoms. This resulted in 92.3% accuracy and 0.993 AUC. The helices

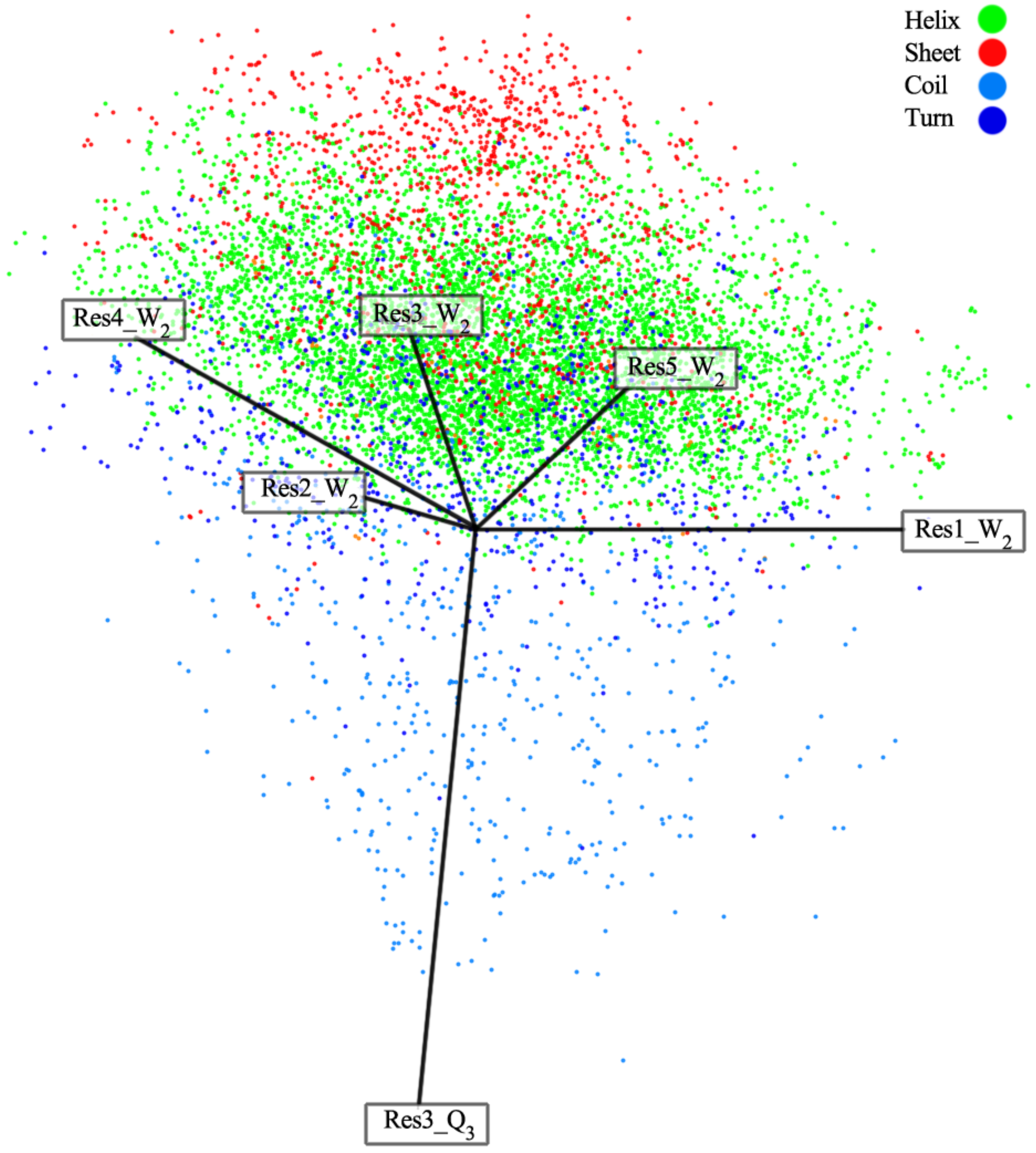


Figure 2.2:
Distribution of the class values with respect to different features in 2D linear projection. ResX_Y represents the feature Y of the residue X (out of 5) in the protein segment.

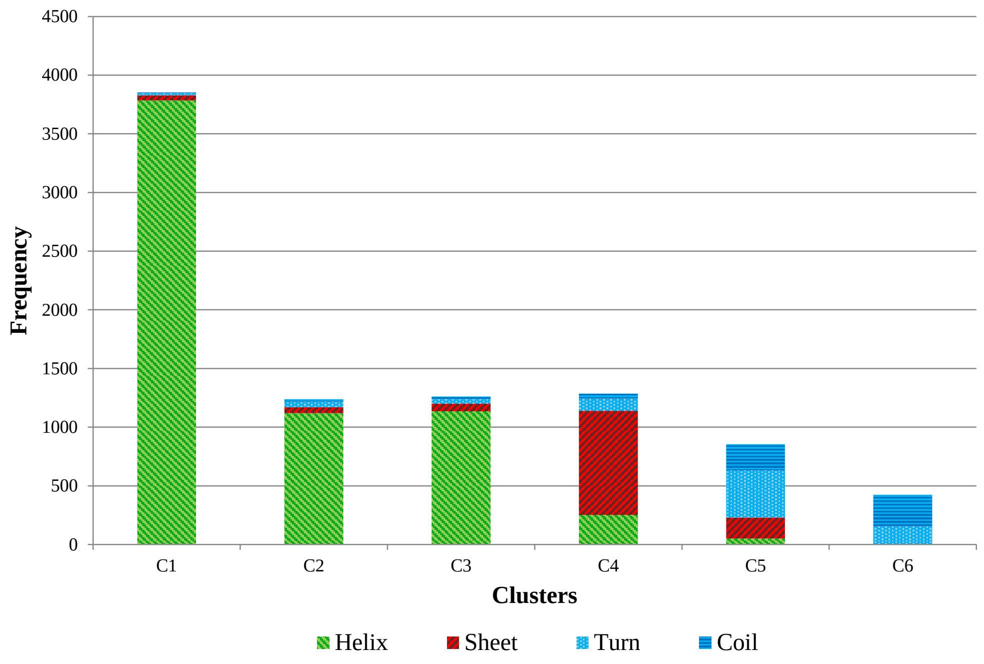


Figure 2.3: The number of elements in each cluster by their secondary structural elements.

can be predicted at 99% sensitivity. Since helices are formed by local interactions that are established within the close vicinity of each amino acid, we can conclude that this structure can easily be captured by the orientational order parameters.

While helices can be predicted quite easily using backbone dihedral angles, this is not the case for sheet structures due to non-local, long range interactions. We show that orientational order parameters can capture the representation of β-sheets equally well (91% sensitivity) since strands stand parallel to each other to form the sheets. There is less information coming from the sequentially adjacent residues forming the sheet in comparison to helices (which makes it difficult to predict them in secondary structure prediction algorithms) but the orientational order descriptors can still capture the necessary local and neighbor information. Addition of orientational order parameters with higher cutoff distance values may help in this regard.

Turns and coils are more difficult to predict in comparison to helices and sheets, (75% and 71% sensitivity respectively). This is expected as they are short, can be found in different local environments (i.e. buried in the core or exposed to water) and lack a rigid structure. Turns are easier to predict than random coils since they are more structured and may have conserved hydrogen bonds between the backbone residues. Some coil structures can be mistakenly classified as turns (22.2%) but the rate of misclassification of turns as coils is not as high (6.9

While the continuous features are shown to be enough to capture secondary structure, we also investigated the applicability of comparing two orientational order feature vectors to evaluate structural similarity (i.e. whether a vector can be assigned to a class based on just a distance value and not by a complex rule learned by the SVM). By using an unsupervised clustering method with a simple

Manhattan distance metric, we have obtained 6 clusters that correctly predict the secondary structure with 84.6% accuracy and 0.932 AUC, showing that similar structures definitely have similar vector characteristics, which is very important for use in structural alphabets. We can also see this effect in Figure 2.2; the classes have distinctive characteristics in their features that can be recognized even on a linear projection with few features.

We also looked at the relative importance of each feature in the descriptor vector. Q_3, Q_4 and Q_9 seem to be the most important features in prediction of the secondary structure elements, but a more through experimentation is needed.

We conclude that there is very strong potential application of orientational order parameters, especially in establishment of a new structural alphabet that takes local backbone structure as well as contact information from the neighboring regions into account. Such an alphabet can be exploited to identify structural motifs in a protein family that cannot be captured with other methods.

3 GLOBAL DESCRIPTORS FOR PROTEINS

3.1 Introduction

This chapter deals with the problem of finding descriptors in a global scale, descriptors that can be used to find the similarities or differences of multiple proteins. Such global features can help on finding similar clusters in data sets by unsupervised learning, or can be used to learn factors that differentiate between two subclasses of the data. Finding descriptors that are informative even when averaged over the whole protein is a challenging but important task. Due to the complexity of the macromolecules, abstract representations are required to convert the real world representation into quantifiable descriptors suitable for machine learning.

This chapter describes a variety of known and novel descriptors that use information from a wide range of domains; from coding nucleic acid sequence, amino acid sequence, secondary and tertiary structure, physicochemical data, catalytic and active site information, residue interaction and mechanics/dynamics data, and finally 3D surface patches and hot spots. Our motivation is that such a combination of elements will be applicable to a wide range of biological problems without requiring much user input. Using information from such different biological domains can help in finding features that can differentiate between the user-defined classes for a list of input data, and may result in automatic exploratory analysis of the vast amounts of data without very little user intervention.

To test the plausibility of combining our features with novel machine learning algorithms, we apply those features to the problem of predicting mRNA and protein expression levels. This enables us to predict whether the protein will be expressed or not and an approximate level of steady-state protein abundance within the host, the solubility or aggregation of the final gene product, and whether it will correctly fold or be degraded.

Using a very comprehensive data set consisting of 19 independent studies from 5 different organisms (both homologous/autologous and heterologous expression), a comprehensive statistical analysis was carried out, which were further used to build a novel machine learning tool for prediction of protein abundance. This results in descriptors that explain a significant portion of the variance within the protein levels, some of which are organism-independent. The developed descriptors will help the understanding of the inner mechanisms of the cellular machinery.

3.2 Features from Protein Coding Nucleotide Sequences

3.2.1 Raw features of the coding sequence

These features are organism independent and rely only on the nucleotide sequence of the gene in question.

Codon Frequency and Relative Synonymous Codon Usage (RSCU)

In its simplest version, codon frequency is the number of occurrences of a specific triplet, divided by the number of all codons in a gene. Even though it is very basic, it contains a great deal of information about the codon usage patterns of the gene in question. However, since the frequency of each codon is dependent and limited by the frequency of amino acids that they code for, that bias can be removed by looking not only at the counts of a specific codon, but its synonyms as well. To this end, Relative Synonymous Codon Usage [63] is a measure of uniformity in the selection of synonymous codons of a gene . RSCU values are the number of times a particular codon is observed, relative to the number of times that the codon would be observed for a uniform synonymous codon usage (if all the synonyms for a given amino acid had the same probability). A codon that is used less frequently than expected will have an RSCU value ¡ 1 and a codon that is preferred more often will have RSCU of ¿ 1.

G+C content of the gene (GC)

The frequency of nucleotides that are guanine or cytosine. Even though the genomic GC content is highly variable within organisms, it is known that genes have higher GC content compared to the whole genome, and length of the coding sequence is directly proportional to higher GC content [64].

G+C content 3rd position of synonymous codons (GC3s)

This the fraction of codons that are synonymous at the third codon position, which contain either a guanine of cytosine at that third codon position.

Base composition at silent sites (G3s, C3s, A3s, T3s)

These 4 features quantify the usage of each base at synonymous third codon positions. Although correlated with GC3s, they are not directly comparable. For the calculation of GC3s, each synonymous amino acid has at least one synonym with G or C in the third position. Therefore, amino acids with 2 or 3 synonyms do not have an equal probability when choosing a base in the synonymous third position. Thus, A3s is the frequency that codons have an A at their synonymous third position, relative to the amino acids that could have a synonym with A in the synonymous third codon position.

3.2.2 Indices of Codon Usage

Due to the multivariate nature of the codon usage, statistical analyses are required to analyze such data in its entirety. While the frequency of each codon within a gene carries some part of the information about the selection of codons, that information is spread over 61 different features. To tackle this problem, multiple measures and codon usage indices were developed throughout the years. These indices attempt to summarise, simplify and explain the bias within the codon usage of a gene by unifying the information contained within those 61 frequency values. While these measures can be organism independent, it is usually necessary to add a priori knowledge about the preferred codons of an organism to identify the major trends in the variation of the data. Using the codon selection trends of an organism, we can summarize the codon usage of a gene in question by positioning it in accordance with these trends.

Here we define some of the indices that can be used to analyze the codon sequence. An overview of the methods is given in Table 3.4.

Effective number of codons (Nc)

The effective number of codons [65] is a measure that quantifies the deviation of the codon selection of a gene from the random usage of synonymous codons. Nc is related to the amount of entropy in the codon usage of a sequence. It reaches is maximal value of 61 when all codons are used equally, and its minimal value of 20 when only one codon is used per amino acid. Since the selection bias results in a decrease of the entropy of codon usage in a sequence, Nc provides a reliable way of testing this effect. Nc requires no knowledge of the optimal codons or a reference set of highly expressed genes, and therefore applicable to any organism with minimal effort.

Since the silent G+C content in the third position (GC3s) of a gene g (shown as x_g for brevity) affects its Nc value, Wright et al. [65] proposed an equation to approximate this relationship (Nc_g) under the hypothesis of no selection. dos Reis et al. [66] proposed a modification of the general formula to optimize this metric to better capture the selection bias within a gene (Nc_{opt}) using the external variables a, b, c. The a, b, c values that optimize the function for *E.coli* and *H.sapiens* gene data sets (as well as the average values we used in this study) are given in Table 3.1.

$$Nc_g = 2 + x_g + \frac{29}{x_g^2 + (1 - x_g)^2} \tag{3.1}$$

$$Nc_{opt} = a + x_g + \frac{b}{x_g^2 + (c - x_g)^2} \tag{3.2}$$

Table 3.1: Parameters for Nc_{opt}

par	*E.coli K12*	*H.sapiens*	Used
a	-6.459	-6.650	-6.0
b	34.01	34.43	34.0
c	1.023	1.028	1.025

Codon Adaptation Index (CAI)

Codon Adaptation Index [67] is a measurement of the relative adaptiveness of the codon usage of a gene towards the codon usage of highly expressed genes for that organism. CAI is the most widespread technique for analyzing codon usage bias. While the Nc index measures the deviation of the codon selection from a uniform bias, CAI measures the deviation with respect to a reference set of genes.

CAI is defined as the geometric mean of the relative adaptiveness score w of each codon (predefined for an organism) over the length of the gene sequence.

$$\mathrm{CAI} = \exp\left(1/L\sum_{l=1}^{L}\log\left(w_i(l)\right)\right) \tag{3.3}$$

The relative adaptiveness of each codon is the observed frequency of each codon f_i to the usage ratio of the most abundant codon for the same amino acid.

$$w_i = \frac{f_i}{\max(f_j)} \qquad i, j \in [\text{synonymous codons for amino acid}] \tag{3.4}$$

Calculated codon adaptation indices for *E.coli* and *S.cerevisiae* are given in Table A.5.

Frequency of Optimal Codons (Fop)

Frequency of Optimal Codons [68] is the ratio of optimal codons to synonymous codons that are predefined for an organism. Fop values are calculated as

$$\mathrm{Fop} = \frac{N_{\text{optimal}} - N_{\text{non-optimal}}}{N_{\text{optimal}} + N_{\text{common}} + N_{\text{non-optimal}}} \tag{3.5}$$

where N is the number of codons that are defined as either "optimal", "common" or "non-optimal (rare)". Thus, Fop values are always between 0 (where no optimal codons are used) and 1 (where only optimal codons are used).

Calculated codon optimality indices for *E.coli* and *S.cerevisiae* are given in Table A.5.

Codon Bias Index (CBI)

Codon bias index [69] is a measure of directional codon bias. CBI measures the extent to which a gene uses a subset of optimal codons defined for an organism. While it is similar to Fop, CBI uses expected usage factor for scaling instead of using equal weights. Using the list of preferred triplets for an organism, CBI measures the distance between any given mRNA sequence and the preferred mRNA sequence that could code for that particular protein, i.e. fraction of codon choices that are biased towards the preferred triplets.

In a gene with an extreme codon bias (in which only the preferred codons are used for all of the triplets in the gene), CBI will be 1. In a gene with random codon usage CBI will be 0. However, CBI can also take on negative values, which means the number of optimal codons is less than what can be expected in a random selection. If CBI is significantly lower than zero, this shows a bias towards rare, non-preferred triplets.

Calculated codon optimality indices for *E.coli* and *S.cerevisiae* are given in Table A.5.

3.2.3 tRNA Based Indices

tRNA Adaptation Index (tAI)

The tRNA adaptation index [66] is a measure of the tRNA usage by coding sequences. tRNA gene copy number across some genomes has a high and positive correlation with tRNA abundance within the cell [68, 70]. It is fair to assume tRNA abundance as the driving force for translational selection. Therefore, measuring the tRNA usage of a gene may provide an indirect way for detecting translational selection. In this context, tAI will quantify how well the gene in question is adapted to the tRNA gene pool of an organism.

The absolute adaptiveness value W_i for each codon i is defined as:

$$W_i = \sum_{j=1}^{n_i} (1 - s_{ij})\mathrm{tGCN}_{ij} \tag{3.6}$$

where n_i is the number of tRNA isoacceptors that recognize the i^{th} codon, tGCN_{ij} is the gene copy number of the j^{th} tRNA that recognizes the i^{th} codon, and s_{ij} is a selective constraint on the efficiency of the codonanticodon coupling (Table 3.2).

The 64 codons that comprise the genetic code can be clustered into groups of four elements, which reflect the natural way in which tRNAs recognize them (by Crick's wobble rules [71] for codonanticodon pairing). To calculate W_i values, it is best to sort the codons as $\{T, C, A, G\}\{3\}$, resulting in {TTT, TTC, TTA, TTG, TCT, TCC, TCA, TCG ...}. Then, the formulas in Table 3.3 can be used to calculate all W_i values by incrementing i from 1 to 61 by 4.

From the W_i values the relative adaptiveness value w_i of a codon is obtained

Table 3.2: s values for codon-anticodon pairing. (I=Inosine, L=Lysidine)

s	value
$s_{I:U}$	0
$s_{G:C}$	0
$s_{U:A}$	0
$s_{C:G}$	0
$s_{G:U}$	0.41
$s_{I:C}$	0.28
$s_{I:A}$	0.9999
$s_{U:G}$	0.68
$s_{L:A}$	0.89

Table 3.3: Formulas for calculating W according to Crick's wobble rules

n	Anticodon	Codon	W
i	INN	NNU	$(1 - s_{I:U})\text{tGCN}_i + (1 - s_{G:U})\text{tGCN}_{i+1}$
$i + 1$	GNN	NNC	$(1 - s_{G:C})\text{tGCN}_{i+1} + (1 - s_{I:C})\text{tGCN}_i$
$i + 2$	UNN	NNA	$(1 - s_{U:A})\text{tGCN}_{i+2} + (1 - s_{I:A})\text{tGCN}_i$
$i + 3$	CNN	NNG	$(1 - s_{C:G})\text{tGCN}_{i+3} + (1 - s_{U:G})\text{tGCN}_{i+2}$

as:

$$w_i = \begin{cases} W_i/W_{max} & \text{if } W_i \neq 0, \\ w_{mean} & \text{else} \end{cases} \tag{3.7}$$

where W_{max} is the maximum W_i value and w_{mean} is the geometric mean of all w_i with $W_i \neq 0$.

Finally, tRNA adaptation index tAI_g of a gene g is defined as the geometric mean of the relative adaptiveness values of its codons.

$$\text{tAI}_g = \left(\prod_{k=1}^{l_g} w_{i_{kg}} \right)^{1/l_g} \tag{3.8}$$

where i_{kg} is the codon defined by the k^{th} triplet in gene g and l_g is the length of the gene in codons (except the stop codon). The measure tAI_g estimates the amount of adaptation of a gene g to its genomic tRNA pool.

Table A.6 contains the tRNA gene counts for several organisms that were used in this study.

Translation Efficiency Bottleneck Calculations

tRNA adaptation index of a gene is the geometric mean of the adaptiveness value of all of its codons. Since it measures the overall adaptation of the gene to the tRNA pool, the effects of locally unfit regions may be averaged out when the whole sequence is used during the calculation. To capture the information about the "bottleneck" residues that may have negative effects on the translation efficiency, we developed several new indices based on the tAI measure.

tAI bottleneck value uses a sliding window of size m over all of the codons of the gene g, where the geometric mean of the relative adaptiveness values of the m codons are calculated, and the minimum value over all of the gene is taken.

$$\text{tAI Bottleneck}(g, m) = min_{j=1}^{l_g-m+1} \left(\left(\prod_{k=j}^{j+m-1} w_{i_{kg}} \right)^{1/m} \right) \tag{3.9}$$

We use $m = \{1, 3, 6, 9, 12, 15, 20, 25\}$ during calculations.

Our other measures look at the length and adaptiveness value of consecutive "unfit" codons. For a pre-defined threshold t, we find the longest segment S in g where geometric mean of the w_i values are below the threshold t. This measure can be defined in two separate ways:

- The longest segment S_a where $\forall\ i \in S_a,\ \ w_i < t$
- The longest segment S_b where $\bar{w_i} < t$ (for $i \in S_b$)

That is, all elements in S_a will be $< t$, whereas S_b may contain elements greater than t as long as the mean of the segment is less than t. Calculation of S_a is trivial. We used the linear-time algorithm defined in [72] for the calculation of S_b.

From those segments, we can use the absolute and relative length of the segment (compared to the length of the whole gene) and the geometric mean of the w values inside the segment.

During the runs, we used $t = \{0.1, 0.2, 0.3, 0.4, 0.5\}$.

Ribosome Flow Model (RFM)

Ribosome Flow Model [73] is a complex probabilistic model of the translation elongation process. RFM and tAI are similar since they are both based on codon adaptation to the tRNA pool. However, in contrast to the RFM, tAI is not sensitive to the order of codons or to the effect caused by ribosome jamming. RFM considers the stochastic nature of the translation process and the excluded volume interactions between ribosomes and aims to capture the effect of codon order and composition on translation rates. RFM is based on the Totally Asymmetric Exclusion Process (TASEP) [74]. In the TASEP model, initiation time as well as the time a ribosome spends translating each codon is exponentially distributed (mean translation times are codon dependent). In addition, ribosomes span over several codons, and if two ribosomes are adjacent, the trailing one is delayed until the ribosome in front of it has proceeded onwards.

RFM has two free parameters, where λ is the initiation rate and the C is the "size" of the ribosome. mRNA molecules are coarse-grained into sites of C codons. A C value of 25 is suggested by the authors, and we tried C=10, 15, 25, 35, 50 during the final experiments.

It is possible to estimate the translation rate of single codons based on tAI or similar measures. RFM uses the absolute adaptiveness value W_i to calculate p_i, which is the probability that a tRNA will be coupled to the codon i.

$$p_i = \frac{W_i}{\sum_{j=1}^{61} tCGN_j} \tag{3.10}$$

The expected time on codon i, t_i is defined as:

$$t_i = 1/p_i \tag{3.11}$$

Table A.7 contains the codon time values for several organisms that were used in this study. The expected time on a site spanning C residues is the sum of times of all the codons in that site.

Using these definitions, Ribosome Flow Model is defined as

$$\frac{dp_1(t)}{dt} = \lambda[1 - p_1(t)] - \lambda_1 p_1(t)[1 - p_2(t)] \tag{3.12}$$

$$\frac{dp_i(t)}{dt} = \lambda_{i-1} p_{i-1}(t)[1 - p_i(t)] - \lambda_i p_i(t)[1 - p_{i+1}(t)] \quad 1 < i < n \tag{3.13}$$

$$\frac{dp_n(t)}{dt} = \lambda_{n-1} p_{n-1}(t)[1 - p_n(t)] - \lambda_n p_n(t) \tag{3.14}$$

where $p_i(t)$ as the probability that the i^{th} site will be occupied at time t. The rate of ribosome flow into/out of the system is given by: $\lambda[1 - p_1(t)]$ and $\lambda_n p_n(t)$ respectively. The rate of ribosome flow from site i to site $i + 1$ is given by $\lambda_i p_i(t)[1 - p_{i+1}(t)]$. The steady-state solutions of the equations results can be used to simulate the rate of protein production.

An important feature that emerges from the RFM model is the fact that the translation rate converges to a constant value as λ increases. This means that each gene has a different translation elongation capacity. This capacity is the maximal translation rate of the gene, achievable for infinitely large λ. In reality, the translation rate increases sharply and converges to a constant value even for small λ, the limiting capacity can be easily achieved for finite and biologically feasible values of λ.

3.3 Features from Amino Acid Sequences

Amino acid usage

Amino acid composition is a simple but usually powerful measure. We include the following indices for analyzing the overall amino acid usage of a protein.

Table 3.4: Traditional measures of translation elongation efficiency.

	Index Name		Model	Explicitly factors in/discriminates between: tRNA availability	Aminoacid composition	Efficiency of individual codons
Fop	Freq. of optimal codons	[68]	Freq. of optimal codons in a gene	Yes	No	Low
CBI	Codon Bias Index	[69]	Freq. of preferred codons relative to random usage of synonymous codons	Yes	No	Low
CAI	Codon Adaptation Index	[67]	Geo. mean of the ratios of the frequency of each codon in highly expressed genes to the frequency of its most abundant synonymous codon	No	Partially	Partially
Nc	Effective number of codons	[65]	Divergence of codon usage from equal usage of synonymous codons	No	No	None
tAI	tRNA Adaptation Index	[66]	Geo. mean of the availability of the tRNAs that serve a codon	Yes	Yes	High
RFM	Ribosome Flow Model	[73]	Physical model of the translation elongation process	Yes	Yes	Very High

Protein frequency indices give the absolute count and the relative frequency of each amino acid in a given protein sequence. These 20 frequency values can be merged to give a summary about specific classes of amino acids within the protein sequence. We define the such indices for the frequency of Hydrophobic, Polar and Charged amino acids. We further include the count of (+) charged and (-) charged residues, as well as the total length of the protein sequence.

IVYWREL

IVYWREL corresponds to the total number of IVYWREL residues in a protein normalized by the length of the protein sequence.

Hydropathicity of protein

The general average hydropathicity (GRAVY) index is an estimate of the overall hydrophobicity of the protein. Each amino acid has a hydrophobicity score that ranges between -4.6 and 4.6 with negative and positive values indicating hydrophilic and hydrophobic residues, respectively. GRAVY is calculated by taking the average of all hydrophobicity scores in a given protein sequence according to the hydropathic indices of by Kyte-Dolittle [52].

Aromaticity score

Aromaticity [75] is calculated using the frequency of aromatic amino acids. It is total number of Phe, Trp, and Tyr residues divided by the total number of residues in a protein sequence.

Aliphatic index

The aliphatic index [76] of a protein is defined as the relative volume occupied by aliphatic side chains (Ala, Val, Ile, and Leu).

Instability Index

Instability index is an estimate of the stability of a protein in a test tube based on the instability potential for each of the 400 dipeptides as defined by [77]. It is calculated using [78].

Isoelectric Point (pI)

pI is the pH value at which net charge of the protein is equal to 0. Theoretical pI value of the protein is calculated using [78].

Molecular Weight (MW)

Theoretical molecular weight for each protein is calculated using [78].

Helix, sheet, turn propensity

Secondary structural propensity values are calculated by counting the total number of residues which are more likely to be included in a given secondary structural element and dividing by protein sequence length. Residues that are more likely to be in each of these structures are:

- Helix: V, I, Y, F, W, L
- Turn : N, P, G, S
- Sheet: E, M, A, L

When the protein contains 3D structure information, these features become redundant due to the addition of the actual frequency of secondary structural elements.

FoldIndex

FoldIndex [79] is an algorithm which predicts if a given protein sequence is intrinsically unfolded. It is based on the average residue hydrophobicity and net charge of the sequence and uses a sliding window to identify large regions within a protein that possess folding propensities different from those of the whole protein.

We utilize the FoldIndex web server to predict the measures for unfoldability, charge and phobic potentials along the protein, number of unordered regions, length of the longest unordered segment and the total length of the unordered segment. The length segments are taken as both absolute and relative to the total protein length.

PEST Motif

A PEST sequence is a peptide sequence which is rich in Pro (P), Glu (E), Ser (S), and Thr (T). This sequence is associated with proteins that have a short intracellular half-life and it is hypothesized that the PEST sequence acts as a signal peptide for protein degradation [80]. Using the EMBOSS/ePESTFind server [81], we predict the potential PEST motifs within a protein, and we summarize the results as the following features: the maximum PEST score and the longest region among all the predicted sites, number of potential PEST sites, sum of the lengths and scores for all potential PEST regions.

3.4 Secondary Structural Features

Secondary structure content

Secondary structure content is defined as the percentage of total number of amino acids taking the form of a particular secondary structural element in a protein. We use both of the 3-class and 7-class secondary structural element definitions explained in Chapter 2. The secondary structural makeup of a protein is useful for characterizing the overall type of the protein fold, such as those defined in the SCOP [82] and CATH [33] databases.

Amino acid content in secondary structures

Amino acid content in secondary structures is calculated for each amino acid by finding the frequency of a specific residue with a specific secondary structure. This results in 60 features (20 amino acids $\times$ 3 secondary structure classes)

Secondary structure embedded sequence alphabet (SSESA)

SSESA corresponds to a structural sequence alphabet that is composed as a combination of 3 features, resulting in 180 triplets. The three conditions making up the alphabet are;

- Amino acid: 20 amino acids
- Secondary Structure: Helix, Sheet, Loop
- Relative Solvent Accessibility: Buried, Partially buried, Exposed

Relative Solvent Accessibility (RSA) is calculated by finding the solvent accessible surface area of each amino acid in a protein and dividing this value by the maximum surface area of that particular amino acid according to standard values provided in [83]. An amino acid is considered Buried if RSA ¡ 0.09, Partially-buried if 0.09 ¡ RSA ¡ 0.36, and Exposed if RSA ¿ 0.36.

3.5 Structural Features

3.5.1 Active Site Composition

The region around the active site of a protein is usually of more importance. To analyze the makeup of this region, we look at a number of features.

To calculate the following measures, we first check the Catalytic Site Atlas [84] for any records that protein might have. If there is no known data about the active site of the protein, we use the metaPocket algorithm [85] to predict the cavities and the most probable region that is biologically active. The center coordinates of the residues that are predicted to be functional is taken as the coordinate of the active site. Then, we define spheres of radii 8, 12, 20 Å centered on the active site. We analyze the residues within those spheres and extract the following features.

- Amino acid composition within the sphere (20 features)
- Amino acid+Secondary structural composition within the sphere (20x3 features)
- Hydrophobic (H), Polar (P) and Charged (C) amino acid composition (3 features)
- Number of contacting HPC residues within the sphere (6 features; Polar-Polar (P-P), P-C, P-H, H-H, H-C, C-C)
- Secondary structural composition, Helix (H), Sheet (S), Loop (L) (3 features)
- Number of contacting HSL within the sphere (6 features; Helix-Helix (H-H), H-S, H-L, S-S, S-L, L-L)

We repeat this process to find these features for all sphere diameters.

3.5.2 Structural rigidity

The B-factor (also called B-value, DebyeWaller factor, or temperature factor) is used to measure local flexibility of residues [86]. B-factor values are reported from experimental atomic-resolution structures. High values indicate higher mobility of residues in crystal structures.

We use the mean B-value of the protein, as well as the mean B-values for the residues belonging to each of the 3 secondary structural elements.

3.5.3 Hinge region related features

We use the HingeProt [87] algorithm for predicting rigid parts of proteins and the flexible hinge regions connecting them in the native topology of protein chains. HingeProt utilizes two elastic network models, Gaussian Network Model (GNM)[19, 20] and Anisotropic Network models (ANM) [88]. Using normal mode analysis, it will predict a list of rigid parts and hinge regions for the two slowest modes as well as a list of short flexible fragments for the two slowest modes which correspond to rigid segments with less than 15 amino acid residues.

We include the number of rigid fragments, the length of the longest rigid fragment and the number of short flexible fragments for both slow mode 1 and slow mode 2.

3.6 Chemical Bond and Interactions

3.6.1 Disulfide bonds

Disulfide bridges are formed through the coupling of thiol groups of two Cys residues. Disulfide bridges exert their stabilizing effects by reducing the entropy of the protein's unfolded state. The bridge usually brings different parts of a polypeptide chain to close proximity and reduces the size of the allowable conformational space, termed entropic effect [89].

We use two approaches in calculating the Disulfide features. First feature is the number of disulfide bridges present in the PDB file if the file contains linkage information. Second feature is a simple predictive approach based on distance. A disulfide bridge or disulfide bond is defined between two residues if the distance between their sulfur atoms is less than 2.3Å.

According to the study by Zhang et al. [89], the magnitude of the entropic effect of a disulfide bridge is proportional to the logarithm of the number of residues separating the two Cys residues involved in the formation of the bridge. In other words, the higher the number of residues separating two Cys residues forming the disulfide bridge, the higher the magnitude of entropic stabilization. To this end, we also add the longest distance (i.e. highest number of residues separating any Cys pair) within the experimental and predicted disulfide bridges if there are any.

Therefore, we define 6 features for disulfide bridges as:

- Number of disulfide bridges (experimental)
- Number of disulfide bridges (predicted)
- Number of CYS residues
- Number of metal bound CYS residues
- Number of free CYS residues
- Longest distance between the indices of CYS any disulfide bridge

3.6.2 Salt bridges

A salt bridge is an ionic interaction between the oppositely charged amino acids of a protein. A salt bridge is a combination of two noncovalent interactions, hydrogen bonding and electrostatic interactions. Different criteria exist in the literature for the definition of a salt bridge. For this study, the criterion for determining salt bridges is that the distance between any of the two carboxyl oxygen atoms on the side chain of Glu or Asp and nitrogen atoms on the side chain of Arg or Lys is $\leq$

4.00 Å. Histidine is excluded as a potential partner in a salt-bridge or ion pair due to the fact that it is very sensitive to pH changes in the physiological range. We add the total number of predicted salt bridges and the number of ions as features.

3.6.3 Cation-π interactions

Proteins contain amino acids that are called aromatic because they contain ring structures with delocalized conjugated π systems which allow the movement of electrons over the entire ring structure providing resonance stabilization. Aromatic amino acids are phenylalanine, tyrosine, tryptophan and histidine. Two different approaches exist in the literature to define cation-π interactions. In the first approach, a cation-π pair is considered interacting if the distance between them is less than 6 Å. In the second approach, only energetically significant cation-π pairs are considered interacting.

We use ExPASy [78] to extract the number of C/P interacting pairs and the number of energetically significant C/P cation-pi interactions where C (cation) takes values from ARG, LYS and P (π) takes values from PHE, TYR, TRP, resulting in $12 = 2 \times 3 \times 2$ features.

3.7 Surface Features

Protein surfaces serve as an interface with the molecular environment and is extremely important for its function. On the surface, geometric and chemical interactions with other molecules provides interaction specificity for ligand binding, docking of macromolecules, and enzymatic catalysis. Ability to characterize and represent the informative and unusual features that occur in a protein is very important for those reasons.

The surface of the protein can be defined in two ways. Molecular surface area (MSA) is the physical surface created by the union of spherical atom surfaces defined by the Van der Waals radius of each atom in the molecule representation. The solvent accessible surface area (SASA or ASA) is the surface that is exposed to the solvent. The difference between ASA and MSA is called Excluded surface area (ESA), regions on the surface which are not accessible by the solvents.

To find ASA, we can take a solvent atom (sphere of size 1.4 Å for Hydrogen in aqueous environments) and roll that atom along the Van der Waals surface of the protein, marking the regions it touches and triangulating the surface to create a topological mesh of the surface. The cavities that are smaller than that solvent atom will not be reached, therefore not accessible.

We calculate the surface of a protein using SurfaceRacer [90] with water as the solvent. The list of features extracted from this surface is shown below.

- Solvent excluded volume and surface area
- Total ASA/MSA
- Total Backbone ASA/MSA
- Polar ASA/MSA
- Polar Backbone ASA/MSA
- Polar Sidechain ASA/MSA
- Nonpolar ASA/MSA
- Nonpolar Backbone ASA/MSA

- Nonpolar Sidechain ASA/MSA
- (+) Charged ASA/MSA
- (-) Charged ASA/MSA
- Total number of cavities

3.7.1 Surface Patches

The features given above calculate the total ASA and MSA on a global scale. However, research has shown that specific surface patches affect the folding and activity of a protein [91]. Averaging the surface information around the whole surface area causes the information about the patches to be lost. It is therefore important to distinguish between a protein with multiple small hydrophobic patches versus another one with a large hydrophobic patch, even though the total nonpolar surface areas for the two proteins may be equal.

To find such patches, we first take the Van der Waals area of each atom and represent it with a sphere. The molecular surface of the protein is found by creating the union surface of those spheres using the marching cubes algorithm [92], resulting in a mesh created from triangle polygons. For each vertex in the final mesh, the two measures Lipophilic Potential (LP) and Electrostatic Potential (ESP) of an atom along a surface point are calculated by [93]. Those potentials are interpolated along the vertices by taking the weighted average of the LP and ESP for specific atoms neighboring that vertex [94].

After the creation of the mesh with the node-embedded LP and ESP measures, the mesh is converted into a graph structure. Each vertex v_i in the mesh is connected to its 6 neighboring vertices v_j (6 due to the triangular polygons) with an edge weight of either $|LP_i - LP_j|$ or $|ESP_i - ESP_j|$. Using this graph, we perform the spectral clustering method ClusterX [95] to cluster the network into a collection of subgraph clusters, based on the LP or ESP properties. The resulting subgraphs will be surface patches that are significantly different than the neighboring surface features.

Using those patches, we calculate the following measures.

- Mean LP and total area of the patch with the:
  - largest mean LP
  - smallest mean LP
  - largest max LP
  - smallest min LP
  - largest area
- Mean ESP and total area of the patch with the:
  - largest mean ESP
  - smallest mean ESP
  - largest max ESP
  - smallest min ESP
  - largest area

To give an example, among all the patches in P, we find the patch with the "largest mean LP" p_m as

$$p_m = argmax_m \frac{\sum_{(i \forall v_i \in p_m)}^{n} LP_i}{n}$$

and the patch with the "largest max LP" p_k as

$$p_k = argmax_k \left(max_{(i \forall v_i \in p_k)} LP_i \right)$$

The rest of the features follow similar definitions. We calculate the mean LP or ESP and the total surface area for the selected patches.

3.8 Application: Prediction of protein abundance

3.8.1 Introduction

Recombinant protein production is an important area of research especially for pharmaceutical and biotechnology industry [96, 97]. Up to date various approaches have been established to express proteins in a variety of host organisms ranging from bacteria, yeast to mammalian cells. Even though there are great advantages of heterologous protein production, such as high production yield potential and low cost, there are also great challenges since the production of a functional protein is highly related to the cellular machinery of the host organism. The quantitative relationship between change in absolute protein concentrations in response to biological stimuli are still not completely understood [98]. The complexity of this problem mainly arises from the imperfect understanding regulation of proteome models [99, 100]. Despite many studies investigating gene expression from a global perspective, there is still lack of information on the synthetic potential of target genes [101, 102]. Understanding the factors that are involved in protein expression efficiency of the host would enable optimization of protein production process, enhancing the yields and thus driving down the cost of production. Particularly, predictive models for gene expression would be beneficial for the production of proteins of biotechnological interest at low costs [103].

Gene expression occurs through a sophisticated machinery which is initiated by the mRNA synthesis and finalized by the formation of functional protein. Protein levels within the cell are controlled and regulated at many levels during transcription and translation, and can be affected by factors such as the mRNA stability, post-translational modifications and degradation of the expression products [101]. Hence, to understand how and why proteins accumulate or degrade to reach their absolute concentrations, one should consider each step involved in the synthesis of mRNA and functional protein [98]. Protein production process can be divided into two stages, the expression stage and the post-translational stage and each may influence the production yield of the physiologically active protein.

Transcriptional regulation is the first controlling step of gene expression since it determines when and how much mRNA will be synthesized. However, control of the transcription rate alone is not sufficient to decide on the expressed protein abundance [104], and less than 50% of variation in protein abundance can be explained by transcriptional regulation [105]. Such high variations between mRNA and protein abundance points to further levels of regulation than that can be accounted by the variation in the transcription rates. After eukaryotic transcription, mRNA maturation occurs through multiple modifications such as capping, adenylation of mRNA ends and splicing of the intronic segments. Primarily the longevity of eukaryotic mRNA survival depends on such modifications that protect mature mRNA from degradation [105]. Moreover, the localization of mRNA also affects stability of mRNA and thus protein abundance [106]. Such kind of extensive spatial and post-transcriptional modifications may impact the steady-state levels of protein abundance leading to low correlation between mRNA and protein levels [107]. In this sense, post-transcriptional regulation, albeit it is an integral part of gene expression, might rival the sophistication and importance of transcriptional control.

Parallel to post-transcriptional control, translational regulation also affects stability of mRNA and thus protein abundance in many ways through initiation, elongation and termination [108, 109, 110]. mRNA sequences may contain specific sequences and secondary structural elements in their untranslated regions (UTR) [107, 111] that can lower the translation rate of the main ORF [112]. In addition to the fact that mRNA poly-adenylation influences mRNA stability, it also affects translation as the length of the poly (A) tail usually correlates with translation efficiency [113].

Translation efficiency may also be limited by the availability of the tRNAs specific to the codon composition of the mRNA sequence. Since organisms contain unequal number of tRNA molecules for each codon-anticodon pair, some codons are favored more than the rare codons in the sense that they have more tRNAs available than infrequent codons [114]. Thus, codon usage and tRNA adaptation of the coding sequence relative to the host organism can deeply impact the absolute protein levels within the cell.

On top of translational regulation, protein degradation also affects protein abundance. The traits that cause some proteins to be degraded rapidly in vivo apparently include surface features and the nature of the amino and carboxyl termini [115, 80, 116]. The ubiquitin and ATP-dependent proteolytic system is the best characterized means of degradation of soluble proteins and it does not readily hydrolyze substrates with blocked NH2-terminal groups. In addition, it preferentially degrades peptides that contain basic or bulky, hydrophobic NH2-terminal amino acids, relative to peptides that contain other groups [116]. Moreover several degradation signals have been related with protein turnover. For example, PEST sequences [117] that contain proline, glutamate, serine and threonine amino acids were shown to lead protein degradation through ubiquitin-proteasome pathway [118, 119]. Lastly, unstructured protein regions that cannot have a distinct three-dimensional (3D) fold are also found to lower protein stability [120, 121, 122].

For a detailed review of variables (gene, vector, strain and the fermentation) involved in all the stages of recombinant protein production, refer to [123].

The studies in the literature up to now may be broadly divided into two overlapping camps; studies that research the control and regulation mechanisms of the cellular machinery due to different stimuli in the global context of the cell, and the studies that try to quantify the effects of such factors into the translation efficiency of the coding sequence to better control and optimize gene expression. In these respects global protein expression analyses carry a particular importance for today's science and technology. However it is a complicated task to utilize bulk information for shedding light on the expression capabilities [98].

In this study, we aim to find factors that are consistently shown to affect protein expression by combining a large number of data sources in different organisms and experimental sets, consisting of results from both homologous/autologous and heterologous expression. We develop and combine different descriptors to explain the relationship between the coding sequence and the resulting mRNA/protein yield in a very diverse set. Analyzing these factors can help us gain further insights into the workings, limitations and regulation of cellular mechanisms present (and vice versa, lacking) in protein expression. We show that the usually preferred codon usage indices such as codon adaptation index (CAI) and frequency of optimal codons (Fop) show little to no relation to the protein yield in heterologous expression and are not good measures of codon optimization, and suggest more robust indices to use in future studies.

Furthermore, we apply these features to predict the translation efficiency and steady-state abundance of expressed protein for a given coding sequence and the expression host. Such a prediction tool can simplify the technical aspects of protein expression, and can be used to filter out infeasible targets, select suitable hosts for a target and can be used for codon optimization by detecting the rate limiting factors of translation efficiency.

3.8.2 Methods

3.8.2.1 Data sets

Experimental quantification of protein abundance is an essential component for implementation of predictive models of gene expression behavior. Various approaches employing spectroscopy, flow cytometry or western blotting are used for measuring expressed protein abundances in either homologous or heterologous host systems. We used 12 experimental datasets for homologous expression systems and 7 for heterologous systems. A review of the data sets used in this study is given in Table 3.5 . The data sets can be classified under two main categories; Homologous and Heterologous expression. For easier comprehension, the data sets are named with a prefix that defines their content.

Homologous expression data

Homologous expression data contains a number of large scale experiments carried out on *E. coli* and *S. cerevisiae.*

Lu et al. (2007) has utilized absolute protein expression (APEX) profiling method that utilizes the proportionality between the fractions of peptides expected and observed from a given protein using experimental analyses of the fragmentation spectra (MS/MS)[124]. They applied APEX methodology to quantify the homologously expressed 430 proteins in *E. coli*, 833 in *S. cerevisiae* and 782 in *H. sapiens* to estimate the relative contributions of transcriptional- and translational-level regulation of gene expression in single cells.

Newman et al. used flow cytometry in a high throughput manner for measurement of a collection of *S. cerevisiae* proteins (a total of 1942) in which each protein is expressed as a carboxy-terminal GFP fusion protein [98]. In this approach flow cytometry allowed monitoring abundances of expressed proteins from single cells and following changes in the concentrations in response to environmental influences.

Similar to APEX and flow cytometry approach, Ishihama et al. also has quantified protein abundances for 731 *E. coli* proteins [125]. Using a combination of LC-MS/MS analyses, they presented a measure of abundance value for each protein based on the emPAI approach that takes into account the number of sequenced peptides per protein.

Ghaemmaghami et al. have used a library of *S. cerevisiae* genes whose ORF is tagged with a high-affinity epitope and expressed this fusion library from their original chromosamal location in *S. cerevisiae* [126]. They were able to analyze almost 98% of the all ORFs (a total of 2874) in the Saccharomyces genome database [127]. This particular study used western blotting for detecting fused proteins and measuring their abundances. Additionally, they compared the protein abundances with mRNA levels from an ealier microarray study [128].

More recently, Niwa et al. [129] developed a method to evaluate abundance of individual proteins using a cell-free translation system that was reconstituted to contain molecular machinery of *E. coli* responsible for protein synthesis [130, 131]. They carried out a comprehensive analysis, in which the complete *E. coli*

ORF library (ASKA library) [132] was translated in the reconsituted system in a chaperone-free manner [133] and soluble cell lysates for 2963 different proteins were fractionated by by SDS/PAGE and the band intensities were quantified by autoradiography.

Other than microbial host systems, Vogel et al. has used human cell line to express 1000 homologous genes [107] and quantified absolute protein levels and corresponding mRNA concentrations using shotgun proteomics and microarrays, respectively.

Homologous expression data sets are prefixed only with an organism identifier (”EC” for *E. coli*, ”SC” for *S. cerevisiae*, ”PP” for *P. pastoris*, ”HS” for *H. sapiens* or ”CFS” for cell-free synthesis).

Heterologous expression data

Heterologous expression data can further be divided into two sub-classes: *i)* The data sets prefixed with ”HET” (plus an identifier for the host organism) contain absolute quantification data for the heterologously expressed product. *ii)* Data sets with the prefix ”OPT” contain no absolute abundance values, instead they reported relative change in the abundance values. In those data sets, the same amino acid sequences were expressed twice using a pair of nucleic acid sequences differing only in synonymous codon usage.

Data Pre-processing and Sanitization

During calculation, author provided nucleotide sequences were used where possible. For the other studies, provided gene/protein ID values were cross-referenced in various databases to get a one-to-one mapping between IDs. In the studies where only the protein identifier (instead of the coding sequence/transcript ID) were available, the coding sequence for that protein from the specified organism/strain that matches the amino acid sequence were used. Due to the very large number of sequences spanning so many different studies, it is inevitable that some of the inputs are not clean (e.g. wrong nucleotide sequence due to inexact ID conversions between databases, deprecated IDs and so on). To minimize such errors, the data sets were validated within themselves wherever possible, making sure that the intrinsic features such as coding length, molecular weight and so on match between the input sequences and the author provided values.

In all the data sets, the inputs were filtered to exclude nucleotide sequences i-) that are not divisible by 3, ii-) contain stop codons in the middle, and iii-) that do not have a stop codon in the end. Furthermore, for the *S.cerevisiae* data sets, only the genes with verified open reading frames in Saccharomyces Genome Database (SGD, [127]) were used.

The resulting number of data rows for each data set after the data sanitization step is given in Table 3.5. The inter-correlation of experiments for *S.cerevisiae* and for *E.coli* are given in respectively Table 3.6 and Table 3.7. While the majority of SC experiments agree on some level, SC-Western has very low correlation with other data sets. EC data shows very little inter-correlation, except for EC-emPAI and EC-APEX.

3.8.2.2 Learning and Prediction

For prediction of protein abundance, we opted on a deep neural network architecture for our final tests, instead of PLS (partial least squares) regression as used by [145]. While PLS regression can give acceptable results as well, it is known to

Table 3.5: Details of the datasets used for learning/prediction.

Dataset Identifier	Source Organism	Target Organism	#	Protein Yield	mRNA	Solubility	Details	References
Homologous Expression								
CFS-PURE	E.coli	Cell Free Synthesis	2963	+		+		[129]
EC-emPAI	E.coli	E.coli	731	+				[125]
EC-APEX	E.coli	E.coli	430	+				[124]
EC-mRNA	E.coli	E.coli	378		+		Normalized mean of different mRNA studies. See Text.	[134, 135, 136, 137]
HS-APEX	H.sapiens	H.sapiens	843	+	+			[107]
SC-Spectral	S.cerevisiae	S.cerevisiae	1415	+				[138]
SC-APEX	S.cerevisiae	S.cerevisiae	833	+			Data available for growth in both YPD and YMD media.	[124]
SC-GFP	S.cerevisiae	S.cerevisiae	1942	+			Data available for growth in both YPD and YMD media.	[98]
SC-Western	S.cerevisiae	S.cerevisiae	2874	+				[126]
SC-Combined	S.cerevisiae	S.cerevisiae	3114	+			Non-linear combination of 4 different studies.	[139]
SC-mRNA	S.cerevisiae	S.cerevisiae	3961		+		Combination of 36 different mRNA studies	[140]
SC-ProdRate	S.cerevisiae	S.cerevisiae	3974	*	*		Protein production rate based on mRNA & Ribosome density	[140, 141]
Heterologous Expression								
HET-EC-SYN	(Mixed)	E.coli	62	+			Heterologous expression of 2 proteins (40 copies each) differing only in synonymous codon usage.	[142]
HET-EC-Pfalc	P.falciparum	E.coli	984	+*		+*	62 expressed/soluble, 270 expressed/non-soluble (non-quantified), 652 non-expressed	[143]
HET-PP	H.sapiens	P.pastoris	72	+*			*Categorized Protein expression (No/Low/Medium/High)	[144]
OPT-Gileadi	H.sapiens	E.coli	30 x 2	%*			*Relative change in PA after optimization (+/0/-)	[145]
OPT-Schafer	H.sapiens	E.coli	73 x 2	%*			*Relative change in PA after optimization (%)	[146]
OPT-Wagner	H.sapiens	H.sapiens	44 x 2	%*			*Relative change in PA after optimization (%)	[147]
OPT-SGDB	(Mixed)	(Mixed)	65 x 2	%*			*Relative change in PA after optimization (+/0/-)	[148]

Table 3.6:
Correlation of the *S.cerevisiae* data sets within themselves.

Pearson / Spearman	SC-Western	SC-Combined	SC-GFP$_{YPD}$	SC-GFP$_{YMD}$	SC-APEX$_{YPD}$	SC-APEX$_{YMD}$	SC-Spectral	SC-2Dgel	SC-Gygi	SC-ProdRate	SC-mRNA
SC-Western		0.84	0.39	0.37	0.36	0.46	0.26	0.13	0.23	0.40	0.46
SC-Combined	0.90		0.30	0.29	0.54	0.60	0.40	0.60	0.67	0.48	0.51
SC-GFP$_{YPD}$	0.34	0.38		0.96	0.80	0.78	0.73	0.60	0.71	0.64	0.57
SC-GFP$_{YMD}$	0.33	0.37	0.76		0.78	0.78	0.71	0.59	0.70	0.63	0.56
SC-APEX$_{YPD}$	0.37	0.39	0.43	0.43		0.95	0.87	0.65	0.74	0.71	0.69
SC-APEX$_{YMD}$	0.39	0.43	0.43	0.46	0.64		0.86	0.76	0.70	0.70	0.70
SC-Spectral	0.31	0.42	0.43	0.45	0.51	0.56		0.71	0.73	0.51	0.48
SC-2Dgel	0.49	0.48	0.54	0.55	0.73	0.73	0.61		0.56	0.52	0.60
SC-Gygi	0.46	0.46	0.52	0.53	0.71	0.71	0.59	0.95		0.77	0.74
SC-ProdRate	0.48	0.55	0.33	0.32	0.42	0.44	0.33	0.48	0.46		0.93
SC-mRNA	0.45	0.53	0.34	0.32	0.41	0.41	0.33	0.48	0.46	0.93	

Table 3.7: Correlation of the *E.coli* data sets within themselves.

Pearson Spearman	CFS-PURE	EC-emPAI	EC-APEX	EC-mRNA	CFS-Solubility
CFS-PURE		0.26	0.16	0.12	0.15
EC-emPAI	0.29		0.62	0.39	0.29
EC-APEX	0.35	0.60		0.62	0.29
EC-mRNA	0.36	0.56	0.81		0.22
CFS-Solubility	0.14	0.29	0.36	0.34	

be very sensitive to outliers and feature selection [149] and was unreliable within different datasets in our own experiments. Due to the very diverse nature of our datasets, we opted on a neural network architecture with dropout, which was more robust to changes and the selection of input features.

Training was carried through a 4 layer deep feed-forward neural network. A symbolic view of the network architecture is given in Figure 3.1 The hidden layer neurons use a rectified linear activation function (ReLU) instead of the common sigmoid activations (logistic function or hyperbolic tangent variations) due to their better performance in our tests. The top layer uses linear activations to apply a linear regression on the non-linear output of the previous layers. The network was trained using stochastic gradient descent with a minibatch size of 100. To prevent the weights from getting too large, we employed ℓ^2-norm regularization. During learning, momentum was increased linearly from 0.5 to 0.9 in increments of 0.1. An equal learning rate was used for all layers, and the learning rate was scaled down every epoch with 0.98. Due to the difficulty of learning deep networks using backpropagation, we initialized the weights to semi-optimal values using layer-wise prelearning.

While neural networks require extensive regularization and stopping criteria to prevent overfitting, the newly developed dropout technique [150] is shown to greatly limit over-training and ensures sparsity, even in the absence of non-sparsity penalties [151]. With a dropout value of p, each input and hidden unit has its output zeroed with probability of p. This random dropping out prevents the complex co-adaptation of neurons, since a neuron cannot rely on other hidden units for its output, and learns to perform better on its own. Using a dropout value of 0.5 and N hidden units, the network can be thought of as sampling 2^N different networks and performing model averaging on them, which greatly reduces overfitting and is shown to improve upon the state-of-the-art [150, 152]. This network architecture consistently outperformed PLS regression.

3.8.3 Results

5-fold cross validation results for within-set evaulation is given in Table 3.8. A representation of the predictions in *E.coli* data sets are given in Figures 3.2, 3.3 and 3.4. We can see that, with a minimum Pearson correlation of 0.6 among all the different sets, we conclude that the features are strong enough to capture patterns in the expression levels. The results indicate that the protein levels *S.cerevisiae* are much more correlated with our features. The predictions in SC have 0.71 to 0.78 Pearson correlation with the experimental data, whereas EC predictions range from 0.65 to 0.73. *H.sapiens* set also results in this range, with 0.65. We can explain a greater portion of the variance in the expression levels in *S.cerevisiae*

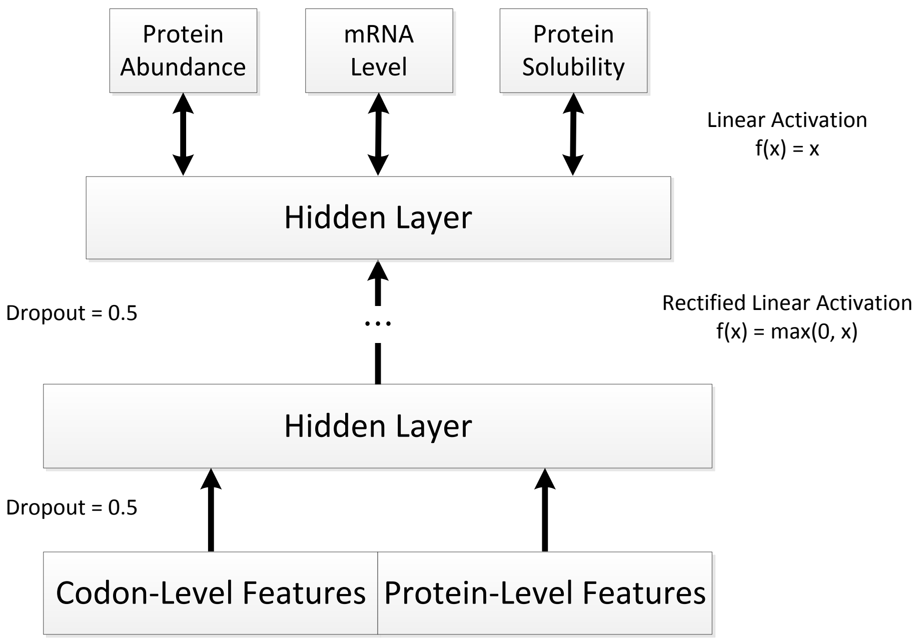


Figure 3.1: Schematic of the Artificial Neural Network used during prediction.

compared to the other organisms.

Results of the inter-experiment predictions are given in Table 3.10 for *S.cerevisiae* across many different experiments. Since the input instances overlap to a certain extent within the different experimental sets, we take care not to introduce any bias. During training and testing, genes that are exclusive to the training set are always included during training. The intersection set of training and test sets are evaluated in a 5-fold manner (training in 4 parts in the training set and testing the remaining part in the test set). Instances that are exclusive to the test set use all of the available training data. Also given in Table 3.10 are the correlation values of the experimental data (similar to Table 3.6, but log-log correlation). The predictions match with the experimental results consistently. Although, interestingly some predictions result in better correlation compared to the experimental results; we reason that since our method will eliminate the experimental noise during prediction, it agrees better with the two experimental sets by acting as a consensus between the two experiments.

The same cross-set predictions are repeated for *E.coli* as well, reported in Table 3.9. Unlike the SC results, EC sets have lower correlation in their predictions, maybe with the exception of EC-emPAI and EC-APEX. In fact, the model trained on EC-emPAI performs better on EC-APEX compared to the 5-fold cross-validation result of EC-APEX itself.

Table 3.11 gives the prediction of heterologous expression using our model. The first top part of the table looks at the prediction capabilities of different homologous sets when applied to heterologous expression. The middle segment gives the inter-correlation of the heterologous data within themselves, and the bottom segment gives the results in reverse; trained in heterologous data and tested in homologous sets. Results that are greater than 0.3 are marked with bold. The first results are not very promising with the exception of HET-EC-SYN. Cell-free expression data from CFS-PURE can capture the heterologous expression levels with a Pearson correlation of 0.66 (higher than 5-fold in CFS-PURE itself), but other EC sets fail to show such good predictive capabilities. However, HET-EC-Pfalc and HET-PP cannot be predicted with confidence, having highest correlation of 0.28 and 0.22. We show the results for the *P.pastoris* in Figure 3.5 as well; since it lacks the absolute protein quantifications (having the categories no/low/medium/high expression instead), the given correlation values in Table 3.11 may not show the true relationship. Since the predictions don't show a normal distribution, we opt on using kernel density estimation. Notice that there is a pattern of increasing average prediction as we move through the empirical categories from low to medium to high. But; the "no expression" set have predictions that range from the whole range. We discuss the importance of these in the next section.

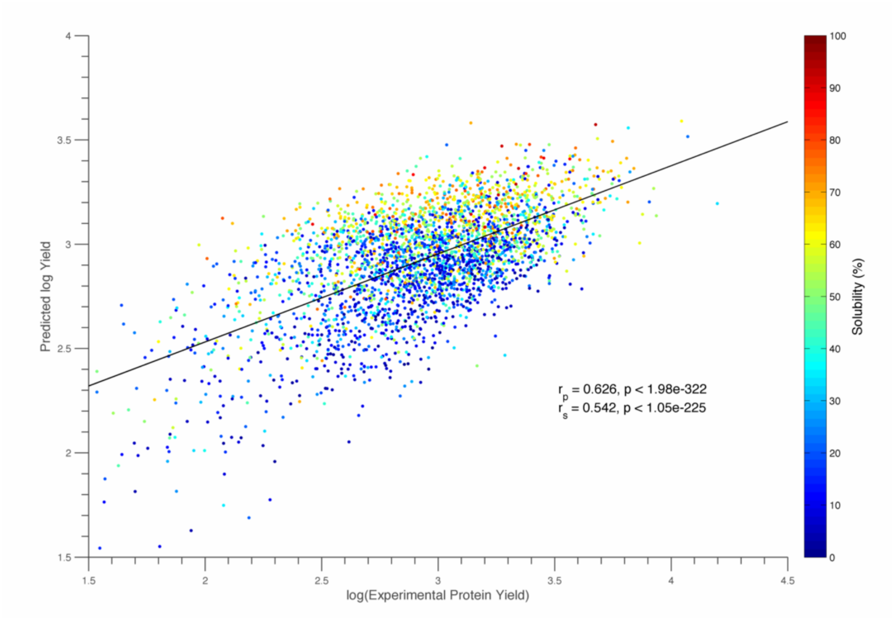
As shown in the middle segment of Table 3.11, heterologous sets don't agree within themselves either, again with the exception of HET-EC-SYN. However, the bottom part of the table is interesting; training with only the 62 instances in HET-EC-SYN (which lack the amino acid features since the proteins are synonymous), we can predict the 2956 genes in CFS-PURE with a correlation 0.54, just below the 5-fold results in CFS-PURE with 0.6, using very limited data. HET-EC-Pfalc can predict the SC sets with a correlation of 0.6, much higher than the reverse of 0.28.

We also give cross-dataset results without regards to species, i.e. cross-species prediction. The prediction results can be seen in Table 3.12. Some data sets exhibit correlations that are quite acceptable. We can predict HS-APEX using CFS-PURE

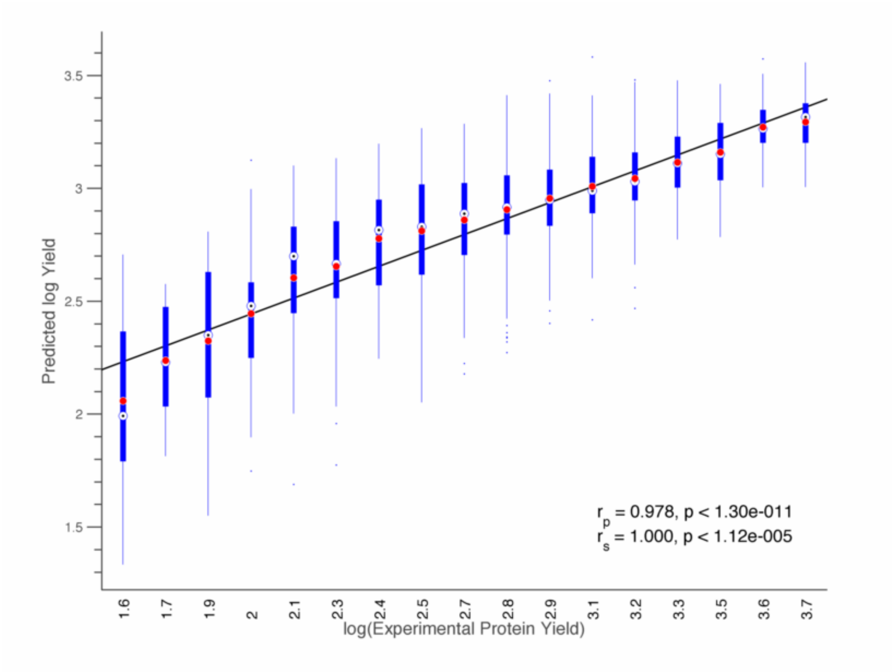
(r_p =0.48) or SC-GFP (0.48); predict EC-APEX using HS-mRNA (0.56) or SC-mRNA (0.58); predict EC-emPAI using SC-APEX (0.64); and predict SC-APEX using either HS (0.75) or EC data (EC-emPAI: 0.71, EC-APEX: 0.66). These results are promising, since most of the time we won't have enough data for ”less popular” organisms.

Table 3.8: Pearson and Spearman correlation values for prediction with 5-fold cross-validation.

		Correlation of Prediction	
Dataset	#	r_p	r_s
CFS-PURE	2956	0.602	0.520
CFS-PURE$_{\text{Solubility}}$	2956	0.646	0.653
EC-APEX	430	0.664	0.638
EC-emPAI	731	0.727	0.690
EC-mRNA	378	0.598	0.541
HS-APEX	843	0.657	0.660
SC-GFP$_{\text{YPD}}$	1940	0.747	0.670
SC-GFP$_{\text{YMD}}$	1873	0.768	0.691
SC-Spectral	1415	0.792	0.731
SC-Combined	3112	0.735	0.712
SC-APEX$_{\text{YPD}}$	833	0.771	0.604
SC-APEX$_{\text{YMD}}$	828	0.792	0.675
SC-Western	2872	0.688	0.641
SC-mRNA	3958	0.717	0.650
SC-ProdRate	3971	0.751	0.722
HET-EC-SYN	62	0.721	0.725

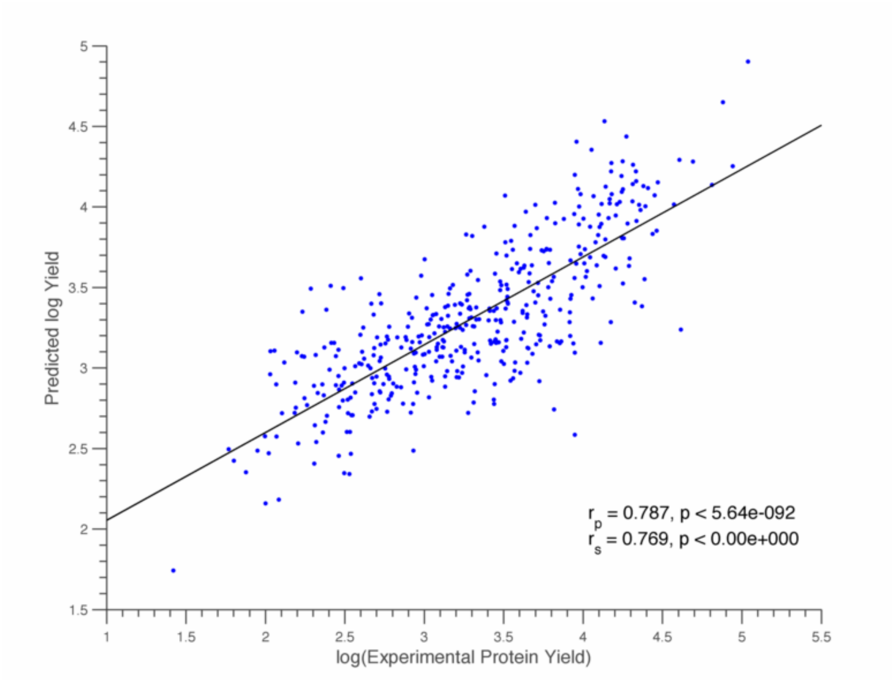


(a) Scatter plot of prediction vs experimental protein yields, colored by experimental solubility.

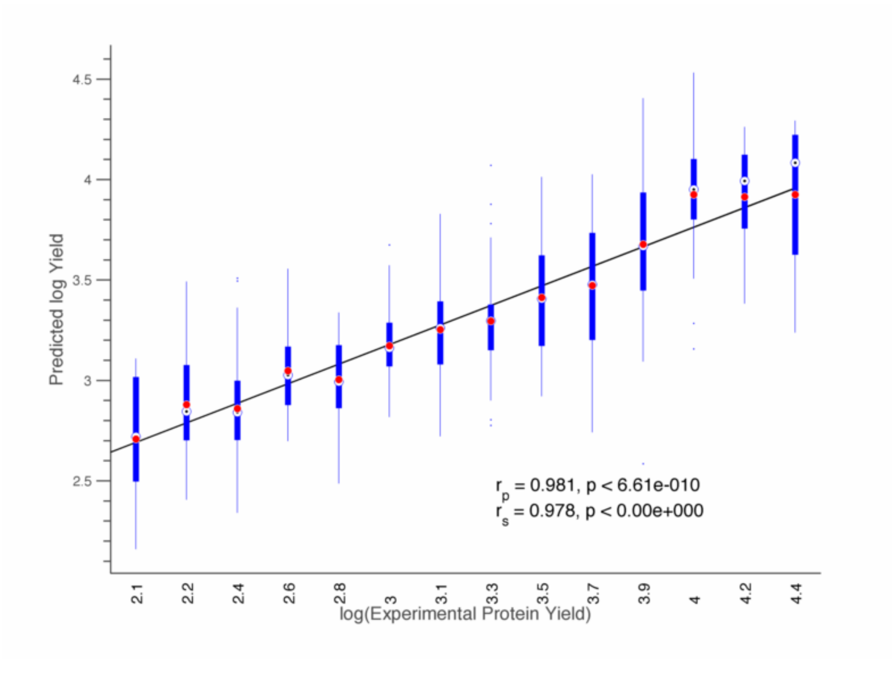


(b) Prediction vs binned experimental protein yields. Mean (red) and median (black/white) of the bins and the best fit line for mean of the bins are shown.

Figure 3.2: Prediction vs experimental protein yields for CFS-PURE.

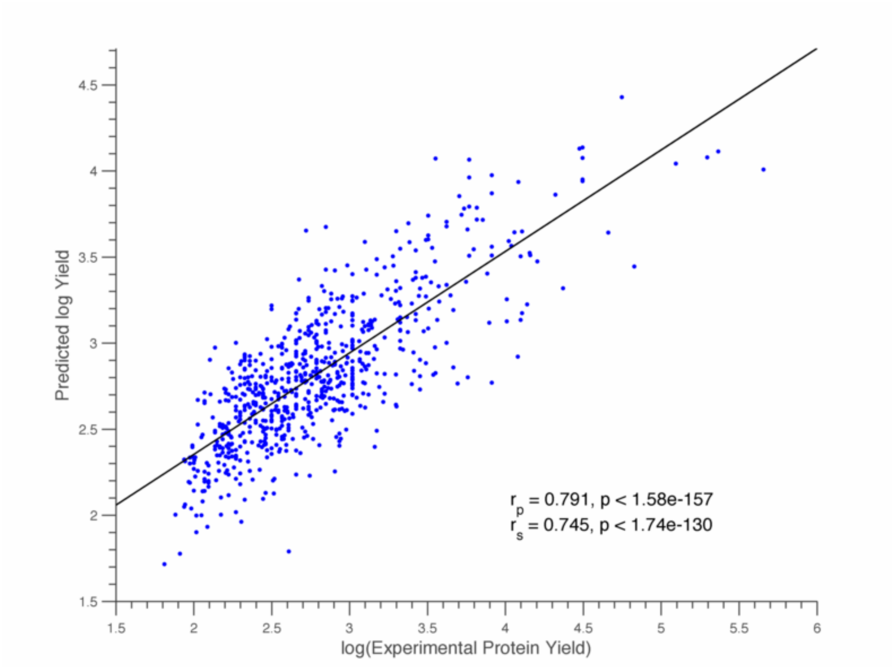


(a) Scatter plot of prediction vs experimental protein yields.

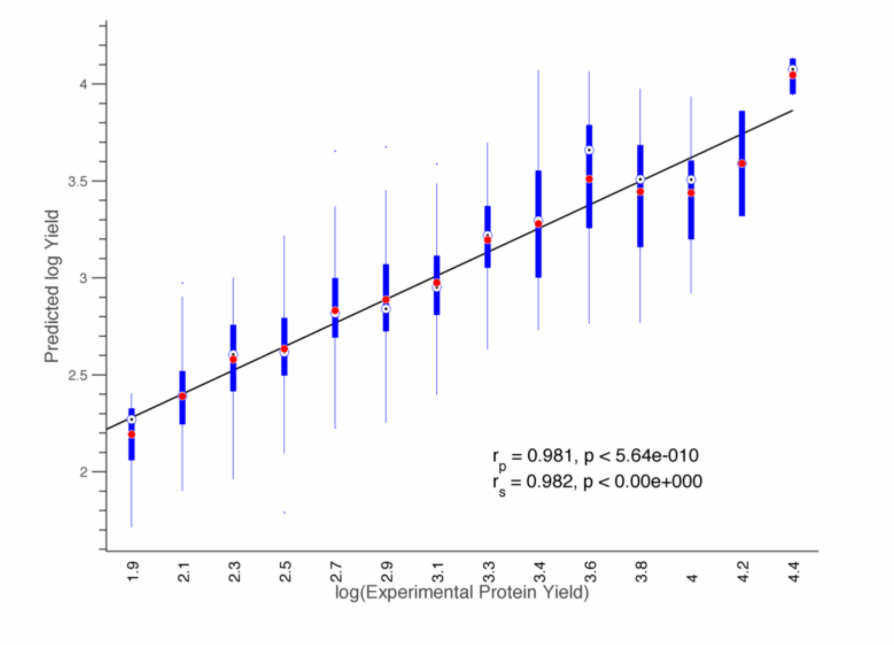


(b) Prediction vs binned experimental protein yields. Mean (red) and median (black/white) of the bins and the best fit line for mean of the bins are shown.

Figure 3.3: Prediction vs experimental protein yields for EC-APEX.



(a) Scatter plot of prediction vs experimental protein yields.



(b) Prediction vs binned experimental protein yields. Mean (red) and median (black/white) of the bins and the best fit line for mean of the bins are shown.

Figure 3.4: Prediction vs experimental protein yields for EC-emPAI.

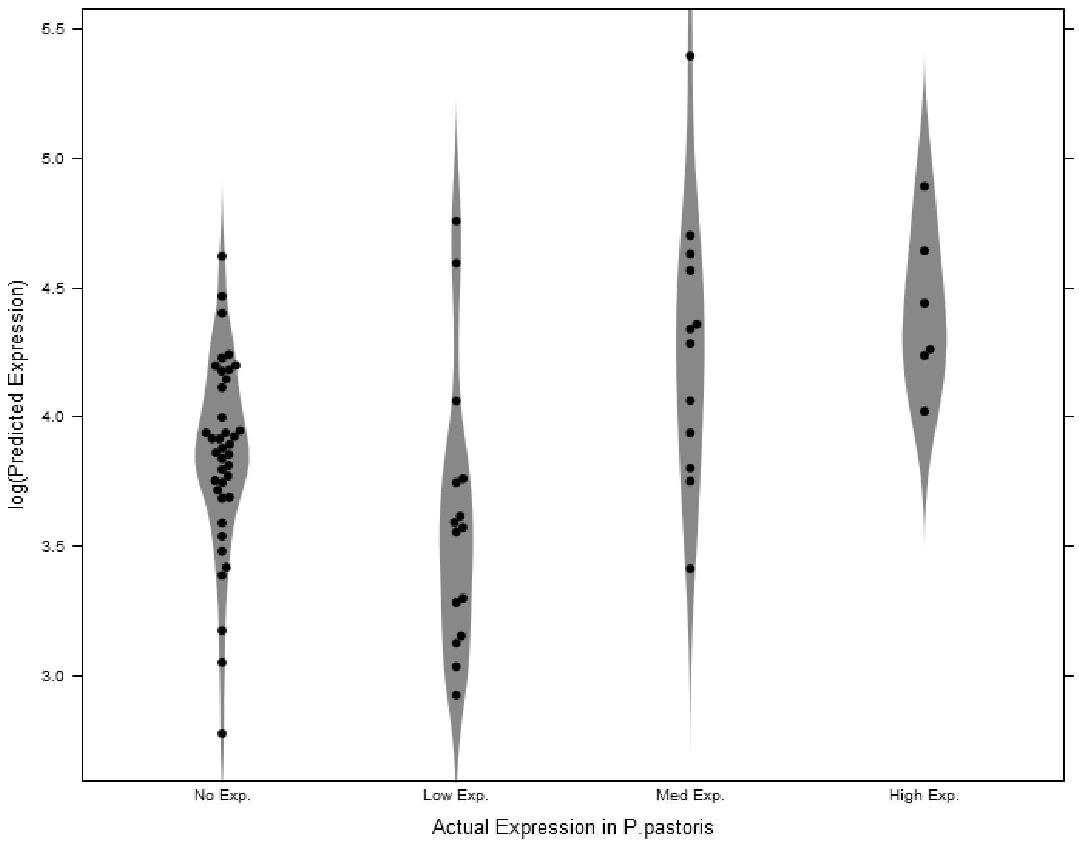


Figure 3.5: Violin plot of the predicted yield versus the experimental expression levels.

Table 3.9: Pearson and Spearman correlations for cross-dataset predictions in *E.coli*

		# of Rows		Correlation of prediction	
Training Set	Test Set	Train	Test	r_p	r_s
CFS-PURE	CFS-Solubility	2956	2956	0.452	0.485
	EC-APEX	2956	430	0.451	0.456
	EC-emPAI	2956	731	0.406	0.466
	EC-mRNA	2956	378	0.402	0.452
CFS-Solubility	CFS-PURE	2956	2956	0.438	0.409
	EC-APEX	2956	430	0.556	0.536
	EC-emPAI	2956	731	0.495	0.553
	EC-mRNA	2956	378	0.454	0.473
EC-APEX	CFS-PURE	430	2956	0.417	0.375
	CFS-Solubility	430	2956	0.468	0.481
	EC-emPAI	430	731	0.701	0.669
	EC-mRNA	430	378	0.613	0.603
EC-emPAI	CFS-PURE	731	2956	0.323	0.337
	CFS-Solubility	731	2956	0.466	0.496
	EC-APEX	731	430	0.685	0.657
	EC-mRNA	731	378	0.555	0.563
EC-mRNA	CFS-PURE	378	2956	0.239	0.216
	CFS-Solubility	378	2956	0.283	0.286
	EC-APEX	378	430	0.654	0.629
	EC-emPAI	378	731	0.553	0.514

3.8.3.1 Factors influencing protein expression and abundance

Each feature was correlated against the given expression level for a data set to find features that are correlated with protein abundance and mRNA levels. We show the Spearman correlation of the important and statistically significant features for the specific groups of experiments. Spearman correlation was preferred over the

Pearson correlation coefficient (unlike the prediction results in the previous step), because the former makes no assumptions about the underlying distributions of the variables. The significance tests use a base p-value of 0.01, divided by the number of features to correct for multiple testing (Bonferroni correction, [153]), resulting in a p-value of approximately $2 \times E^{-5}$.

In each of the tables, significantly correlated features are marked with bold. Table 3.13 shows the features that are globally important, significant in most of the experiments. Table 3.14 and Table 3.15 lists the features that are exclusively important for *S.cerevisiae* (significant in SC, but not in others) and *E.coli* (significant in EC, but not in others) respectively. Finally, we show the features that are significant in in vivo experiments among the homologous expression data, but not on cell-free synthesis or heterologous expression, in Table 3.16.

To better understand the correlation of the features among themselves, we show the representation of the Self-Organizing Map (SOM) trained on the SC databases. Figure 3.6 and 3.7 shows the features that are clustered with respect to the 4 data sets (given as the first 4 elements of Figure 3.6) with a very high weight (feature-class correlation), and with respect to each other (feature-feature correlation) in a weak fashion. We can see that majority of the indices can agree on the high- and low-expressed proteins on the average, and some features have very high inter-correlation within themselves.

The features that have the highest correlation are the "tAI bottleneck" and "tAI longest segment" descriptors proposed in this Chapter, along with RFM. These indices use the tRNA availability and consider the order of the codons.

On the other hand, indices such as Codon Adaptation Index (CAI) and Codon Bias Index (CBI) which are widely used in the literature show very good correlation among the SC and EC data sets; but they are not correlated within the cell-free expression, heterologous expression and *H.sapiens* data. This lack of agreement with heterologous expression levels have been reported as well in the literature [145]. Since CAI uses parameters that are calculated by looking at the proteins with the highest level of expression among an organisms own genes (i.e. homologous), it is correlated well with those results. However, this calculation is most likely biased toward the homologous data, and cannot predict the expression levels of a foreign gene. CBI, Fop, and CAI indices should only be calculated for those species where selection for the translational efficiency has overcome mutational drift. If these indices are calculated for genes where codon usage is determined by mutational bias, the resulting index value is essentially meaningless. For example, the CAI values of *H. sapiens* genes have no "meaning" because human codon usage is driven by mutational biases [154].

Looking at the results, features that give information about the size and length of the protein seem to be the descriptors that are most negatively correlated with protein expression levels near universally. This is apparent in Figure 3.8, where the Molecular Weight (MW) of the proteins are clustered together and the expression levels are shown. However, such indices are not very useful for heterologous expression or codon optimization.

Table 3.10: Pearson and Spearman correlations for cross-dataset predictions in *S.cerevisiae*

		# of Rows			Corr. of Prediction		Corr. of Experi-ments	
Training Set	Test Set	Train	Test	Common	r_p	r_s	r_p	r_s
SC-APEX	SC-Spectral	833	1415	437	0.64	0.67	0.71	0.73
	SC-Western	833	2872	632	0.51	0.58	0.49	0.55
SC-Combined	SC-APEX	3112	833	702	0.56	0.70	0.59	0.62
	SC-GFP	3112	1940	1834	0.66	0.73	0.65	0.69
	SC-mRNA	3112	3958	3104	0.60	0.68	0.64	0.66
	SC-ProdRate	3112	3971	3110	0.66	0.71	0.67	0.68
	SC-Spectral	3112	1415	1415	0.73	0.77	0.85	0.85
	SC-Western	3112	2872	2869	0.64	0.69	0.94	0.94
SC-GFP	SC-APEX	1940	833	472	0.57	0.74	0.60	0.69
	SC-Combined	1940	3112	1834	0.67	0.69	0.65	0.69
	SC-mRNA	1940	3958	1934	0.62	0.70	0.70	0.74
	SC-ProdRate	1940	3971	1939	0.65	0.70	0.68	0.72
	SC-Spectral	1940	1415	1054	0.71	0.75	0.62	0.68
	SC-Western	1940	2872	1764	0.60	0.66	0.60	0.64
SC-mRNA	SC-APEX	3958	833	831	0.58	0.75	0.56	0.69
	SC-Combined	3958	3112	3104	0.67	0.68	0.64	0.66
	SC-GFP	3958	1940	1934	0.66	0.72	0.70	0.74
	SC-ProdRate	3958	3971	3955	0.69	0.73	0.93	0.93
	SC-Spectral	3958	1415	1410	0.70	0.71	0.64	0.66
	SC-Western	3958	2872	2865	0.59	0.64	0.58	0.62
SC-Spectral	SC-APEX	1415	833	437	0.53	0.70	0.71	0.73
	SC-Combined	1415	3112	1415	0.69	0.72	0.85	0.85
	SC-GFP	1415	1940	1054	0.64	0.73	0.62	0.68
	SC-mRNA	1415	3958	1410	0.57	0.66	0.64	0.66
	SC-ProdRate	1415	3971	1414	0.61	0.66	0.66	0.66
	SC-Western	1415	2872	1177	0.58	0.65	0.56	0.58
SC-Western	SC-APEX	2872	833	632	0.57	0.71	0.49	0.55
	SC-Combined	2872	3112	2869	0.71	0.73	0.94	0.94
	SC-GFP	2872	1940	1764	0.65	0.73	0.60	0.64
	SC-mRNA	2872	3958	2865	0.61	0.68	0.58	0.62
	SC-ProdRate	2872	3971	2871	0.67	0.72	0.62	0.65
	SC-Spectral	2872	1415	1177	0.72	0.75	0.56	0.58

Table 3.11: Pearson and Spearman correlations for predictions in Heterologous expression sets

		# of Rows		Correlation of prediction	
Training Set	Test Set	Train	Test	r_p	r_s
CFS-PURE	HET-EC-SYN	2956	62	**0.659**	**0.674**
EC-APEX		430	62	-0.021	-0.020
EC-emPAI		731	62	0.175	0.162
EC-mRNA		378	62	-0.371	-0.415
HS-APEX		843	62	0.108	0.103
SC-APEX-YPD		833	62	0.303	0.311
CFS-PURE	HET-EC-Pfalc	2956	62	0.184	0.129
EC-APEX		430	62	0.226	0.149
EC-emPAI		731	62	0.282	0.139
EC-mRNA		378	62	0.113	0.166
HS-APEX		843	62	0.177	0.213
SC-APEX-YPD		833	62	0.279	0.237
CFS-PURE	HET-PP	2956	70	0.216	0.206
EC-APEX		430	70	-0.018	-0.018
EC-emPAI		731	70	0.172	0.178
EC-mRNA		378	70	-0.073	-0.056
HS-APEX		843	70	0.070	0.144
SC-APEX-YPD		833	70	0.197	0.126
HET-EC-Pfalc	HET-EC-SYN	62	62	**0.305**	**0.339**
HET-PP	HET-EC-SYN	70	62	**0.466**	**0.507**
HET-EC-SYN	HET-EC-Pfalc	32	62	0.265	0.123
HET-EC-Pfalc	HET-PP	62	70	0.066	0.075
HET-EC-SYN	HET-PP	62	70	0.049	0.131
HET-EC-SYN	CFS-PURE	30	2956	**0.541**	**0.449**
	EC-APEX	32	430	**0.363**	**0.395**
	EC-emPAI	32	731	0.285	0.368
	HS-APEX	30	843	**0.451**	**0.579**
	SC-APEX-YPD	32	833	**0.441**	**0.457**
HET-PP	HS-APEX	70	843	**0.436**	**0.573**
HET-EC-Pfalc	SC-APEX-YPD	62	833	**0.603**	**0.522**

Table 3.12: Pearson and Spearman correlations for cross-species predictions.

		# of Rows		Correlation of prediction	
Training Set	Test Set	Train	Test	r_p	r_s
CFS-PURE	EC-mRNA	2956	378	0.402	0.452
	SC-Combined	2956	3112	0.259	0.284
	SC-GFP-YPD	2956	1940	0.203	0.231
	SC-Western	2956	2872	0.279	0.276
	HS-APEX	2956	843	0.478	0.607
CFS-PURE-Sol	SC-mRNA	2956	3958	0.400	0.357
	HS-APEX	2956	843	0.465	0.555
EC-APEX	SC-APEX-YPD	430	833	0.664	0.593
	HS-APEX	430	843	0.427	0.542
EC-emPAI	CFS-PURE-Sol	731	2956	0.466	0.496
	HS-mRNA	731	843	0.104	0.134
	SC-APEX-YPD	731	833	0.712	0.605
EC-mRNA	SC-Western	378	2872	0.363	0.289
HS-APEX	CFS-PURE-Sol	843	2956	0.452	0.449
	EC-mRNA	843	378	0.541	0.511
	SC-APEX-YPD	843	833	0.752	0.585
	SC-Combined	843	3112	0.652	0.629
	SC-mRNA	843	3958	0.669	0.588
	SC-Spectral	843	1415	0.649	0.646
HS-mRNA	CFS-PURE	843	2956	0.112	0.118
	EC-APEX	843	430	0.555	0.536
	SC-GFP-YPD	843	1940	0.664	0.576
SC-APEX-YPD	EC-emPAI	833	731	0.636	0.634
	HS-APEX	833	843	0.325	0.300
	HS-mRNA	833	843	0.123	0.167
SC-Combined	CFS-PURE	3112	2956	0.163	0.161
SC-GFP-YPD	EC-emPAI	1940	731	0.612	0.600
	HS-APEX	1940	843	0.483	0.619
	HS-mRNA	1940	843	0.215	0.327
SC-mRNA	EC-APEX	3958	430	0.579	0.564
SC-Spectral	HS-APEX	1415	843	0.068	0.140
	EC-APEX	1415	430	0.457	0.457

Table 3.13: Important features common to all data sets

Features	CFS	EC-APEX	EC-mRNA	EC-emPAI	HET-EC-SYN	HET-PP	HET-EC-Pfalc	HS-APEX	HS-mRNA	SC-APEX	SC-mRNA
CodonCount_AAT	**−0.339**	**−0.493**	**−0.403**	**−0.440**	−0.192	0.166	−0.185	**−0.450**	**−0.169**	**−0.532**	**−0.369**
CodonCount_TAT	**−0.372**	**−0.477**	**−0.452**	**−0.385**	**−0.594**	−0.186	−0.168	**−0.415**	**−0.145**	**−0.445**	**−0.286**
CodonCount_TTT	**−0.402**	**−0.480**	**−0.468**	**−0.468**	**−0.645**	−0.034	−0.127	**−0.466**	**−0.171**	**−0.469**	**−0.284**
tAI bottleneck w-9	**0.219**	**0.438**	**0.341**	**0.465**	**−0.512**	−0.060	0.038	**0.422**	**0.206**	**0.536**	**0.512**
tAI longest seg.(all¡0.25)	**−0.228**	**−0.285**	**−0.257**	**−0.355**	**0.548**	−0.067	0.064	**−0.312**	**−0.154**	**−0.488**	**−0.395**
tAI longest seg.(avg¡0.5)	**−0.443**	**−0.402**	**−0.426**	**−0.362**	**−0.663**	−0.076	−0.059	**−0.576**	**−0.175**	**−0.595**	**−0.453**
tRNA adaptation index	**0.083**	**0.483**	**0.373**	**0.486**	**−0.511**	0.158	0.097	**0.300**	**0.314**	**0.530**	**0.567**

Table 3.14: *S. cerevisiae* exclusive features

Features	CFS	EC-APEX	EC-mRNA	EC-emPAI	HET-EC-SYN	HET-PP	HET-EC-Pfalc	HS-APEX	HS-mRNA	**SC-APEX**	**SC-mRNA**
CodonFreq_CAG	0.023	−0.102	−0.120	−0.031	0.370	0.031	0.123	−0.051	0.080	**−0.431**	**−0.266**
CodonFreq_CTG	−0.020	0.065	−0.058	0.139	−0.058	−0.155	−0.154	−0.007	0.075	**−0.435**	**−0.277**
CodonFreq_TAC	−0.029	0.001	−0.033	0.126	0.136	−0.020	−0.052	0.010	0.022	**0.273**	**0.128**
CodonRat_ACC	0.028	0.015	−0.003	0.125	−0.313	0.047	0.055	0.071	0.084	**0.375**	**0.279**
CodonRat_CCA	−0.028	−0.127	−0.143	−0.091	0.081	0.210	0.012	−0.095	−0.052	**0.442**	**0.324**
CodonRat_GCA	0.006	0.178	0.197	0.138	0.275	0.067	−0.183	−0.141	−0.101	**−0.490**	**−0.361**
CodonRat_GGC	0.034	−0.028	−0.059	0.040	−0.049	0.034	0.045	0.056	0.083	**−0.323**	**−0.161**
Unordered Segment Len.	−0.026	0.033	0.138	−0.019		0.148	−0.083	−0.100	0.035	**−0.235**	**−0.156**
FoldIndex	−0.071	−0.138	−0.190	−0.065		−0.167	0.070	0.003	−0.102	**0.144**	**0.107**
FoldIndex Phobic	−0.047	−0.061	−0.131	−0.035		−0.161	0.088	0.053	−0.089	**0.198**	**0.142**
Gravy	−0.045	−0.061	−0.131	−0.035		−0.161	0.088	0.053	−0.089	**0.198**	**0.142**

Table 3.15: *E. coli* exclusive features

Features	CFS	**EC-APEX**	**EC-mRNA**	**EC-emPAI**	HET-EC-SYN	HET-PP	HET-EC-Pfalc	HS-APEX	HS-mRNA	SC-APEX	SC-mRNA
AA Charged Ratio	**0.112**	**0.224**	**0.230**	0.101		0.166	0.078	0.069	**0.246**	−0.019	−0.025
AA Composition-LYS	**0.105**	**0.372**	**0.447**	**0.245**		−0.121	0.079	**0.161**	**0.183**	0.011	−0.024
AA Composition-TRP	**−0.105**	**−0.210**	−0.168	**−0.309**		−0.159	0.172	**−0.171**	−0.131	−0.040	−0.045
AA Count-GLY	**−0.364**	**−0.310**	**−0.356**	**−0.268**		**−0.007**	0.051	−0.457	−0.117	**−0.204**	−0.014
AA Count-VAL	**−0.389**	**−0.312**	**−0.355**	**−0.278**		**−0.055**	0.068	−0.506	−0.122	**−0.238**	−0.060
PEST All Count	**−0.060**	**−0.350**	**−0.328**	−0.139		0.045	0.080	0.001	−0.012	−0.088	**−0.096**
PEST All Len	**−0.065**	**−0.349**	−0.328	**−0.124**		0.013	0.051	0.001	−0.023	−0.076	**−0.078**

Table 3.16: Important features of homologous data

Features	**CFS**	**EC-APEX**	**EC-mRNA**	**EC-emPAI**	HET-EC-SYN	HET-PP	HET-EC-Pfalc	**HS-APEX**	**HS-mRNA**	**SC-APEX**	**SC-mRNA**
AA Count - GLN	**−0.363**	**−0.428**	**−0.421**	**−0.379**		−0.031	−0.111	**−0.547**	**−0.171**	**−0.401**	**−0.205**
AA Count - HIS	**−0.342**	**−0.349**	**−0.418**	**−0.353**		0.122	−0.051	**−0.576**	**−0.222**	**−0.377**	**−0.214**
AA Count - LEU	**−0.445**	**−0.449**	**−0.468**	**−0.429**		−0.069	−0.037	**−0.586**	**−0.192**	**−0.412**	**−0.235**
AA Count - PRO	**−0.415**	**−0.394**	**−0.424**	**−0.387**		−0.088	0.063	**−0.555**	**−0.199**	**−0.324**	**−0.177**
CAI	**0.088**	**0.555**	**0.418**	**0.532**	−0.178	0.114	0.238	**0.229**	**0.192**	**0.548**	**0.551**
CodonCount_CAT	**−0.318**	**−0.438**	**−0.469**	**−0.442**	−0.358	0.134	0.008	**−0.495**	**−0.222**	**−0.439**	**−0.252**
CodonCount_TCG	**−0.314**	**−0.484**	**−0.455**	**−0.505**	−0.064	0.115	−0.119	**−0.407**	**−0.144**	**−0.490**	**−0.308**
CodonCount_TTA	**−0.318**	**−0.490**	**−0.404**	**−0.523**	−0.192	−0.133	−0.031	**−0.405**	**−0.198**	**−0.430**	**−0.226**
Molecular Weight	**−0.446**	**−0.395**	**−0.422**	**−0.367**		0.035	−0.061	**−0.581**	**−0.176**	**−0.398**	**−0.211**
RFM-C50-A0.0008	**0.387**	**0.550**	**0.505**	**0.591**	0.092	0.000	0.148	**0.525**	**0.269**	**0.567**	**0.553**
Sequence Length	**−0.443**	**−0.391**	**−0.419**	**−0.360**		0.032	−0.059	**−0.576**	**−0.175**	**−0.389**	**−0.200**
tAI bottleneck w-20	**0.217**	**0.451**	**0.372**	**0.484**	−0.152	0.025	0.135	**0.422**	**0.221**	**0.546**	**0.528**
tAI longest seg.(avg¡0.3)	**−0.234**	**−0.441**	**−0.346**	**−0.427**	0.065	−0.110	−0.004	**−0.597**	**−0.216**	**−0.520**	**−0.470**

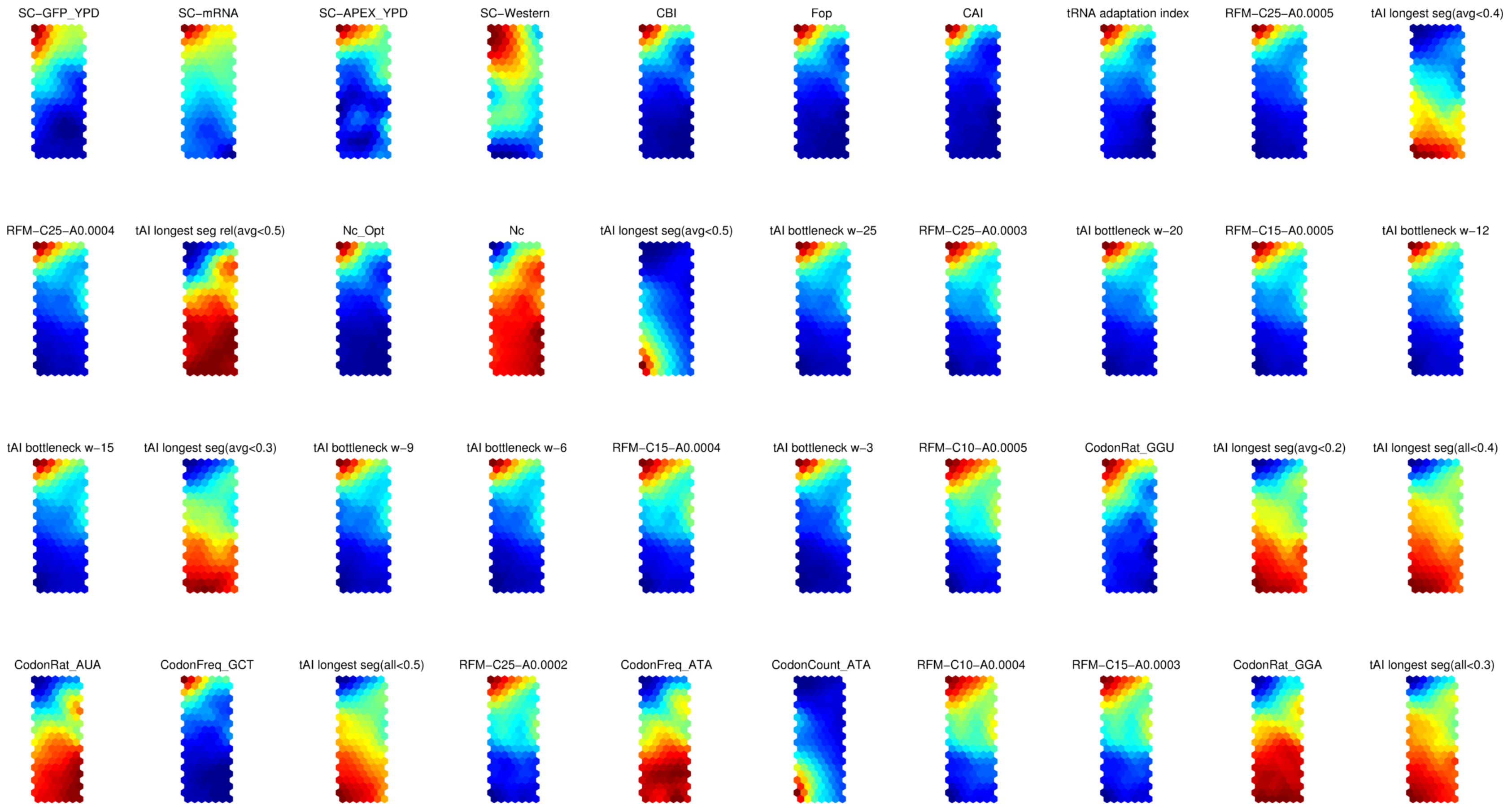


Figure 3.6:
Self-organizing map of the features in *S.cerevisiae* data set (Part 1)

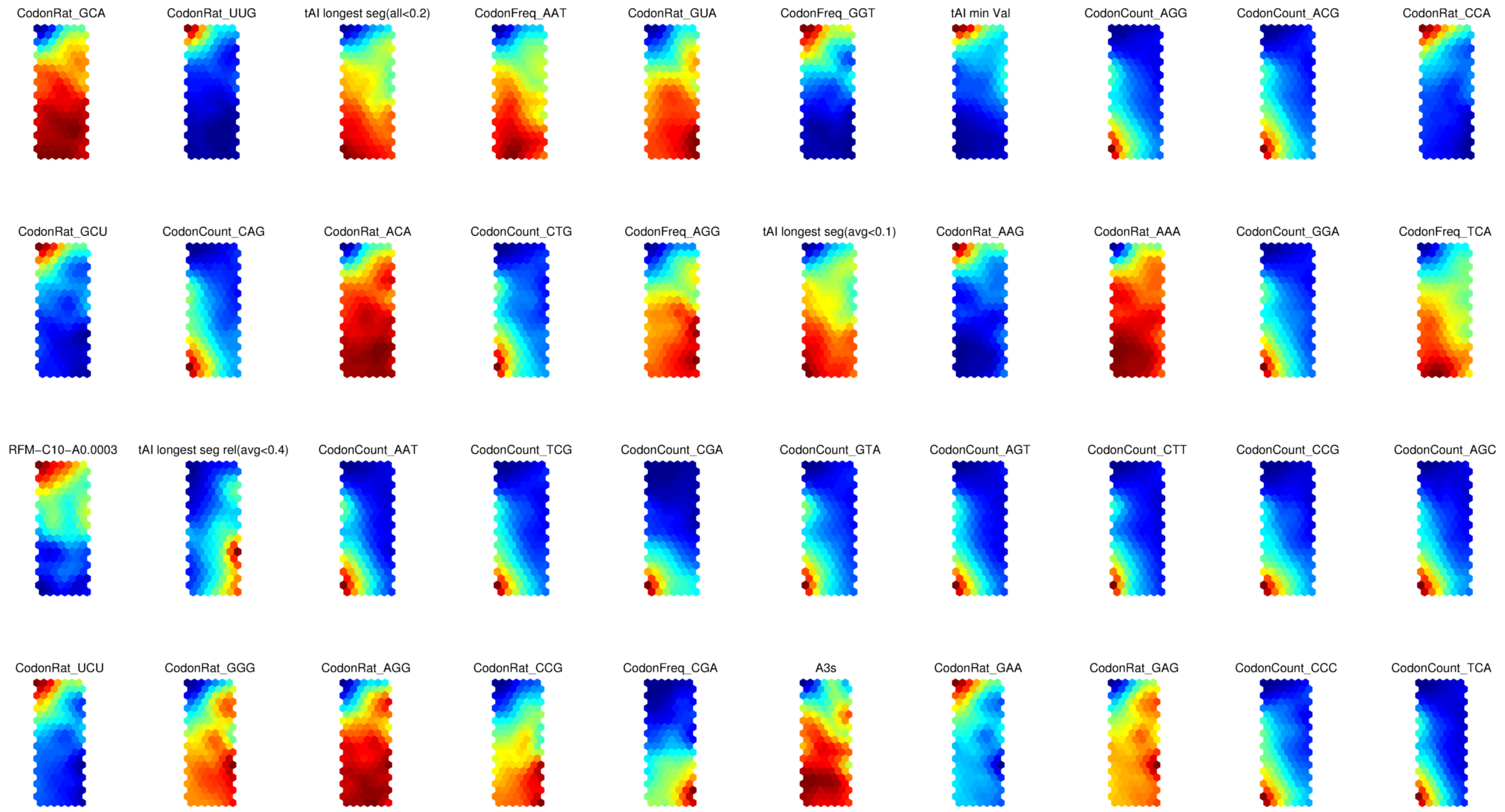


Figure 3.7:
Self-organizing map of the features in *S.cerevisiae* data set (Part 2)

3.8.4 Discussion

This study is the most comprehensive review of effects of different codon usage indices and amino acid features to the protein expression in literature. Whereas other studies analyze one to three data sets for a specific organism, we opted on using a variety of data sources distributed among different expression hosts, as well as including heterologous expression data, which are problematic to characterize and predict. Several studies focused on determining protein level factors that may be involved in active heterologous protein production stage. In a study by Kuratoni et al., they looked at the impact of 39 physiochemical features to active protein production stage [155]. They concluded that the length of the protein, hydrophobicity, PI, amino acid composition, solvent accessibility, presence of intrinsically disordered regions were significant factors. In a separate study, Tuller et al. studied the factors involved in protein aggregation [156] and summarized that proteins that have high negatively charged amino acids, Glu and Asp, tend to be more soluble whereas proteins with high ring structured amino acid content tend to aggregate more. Tokmakov et al. studied the impact of post-translational modification to heterologous protein production yields [157]. They determined that phosphorylation, sumoylation, ubiquitination and prenylation were associated with increased protein production yields of recombinant protein production.

We use a variety of features to quantify the effects of such different factors to the expression levels of protein product. Most commonly preferred codon usage indices such as codon adaptation index (CAI) and frequency of optimal codons (Fop) show little to no relation to the protein yield in heterologous expression and are not good measures of codon optimization. The newly developed Ribosome Flow Model (RFM) and our novel features ”tRNA Adaptation Index bottleneck”

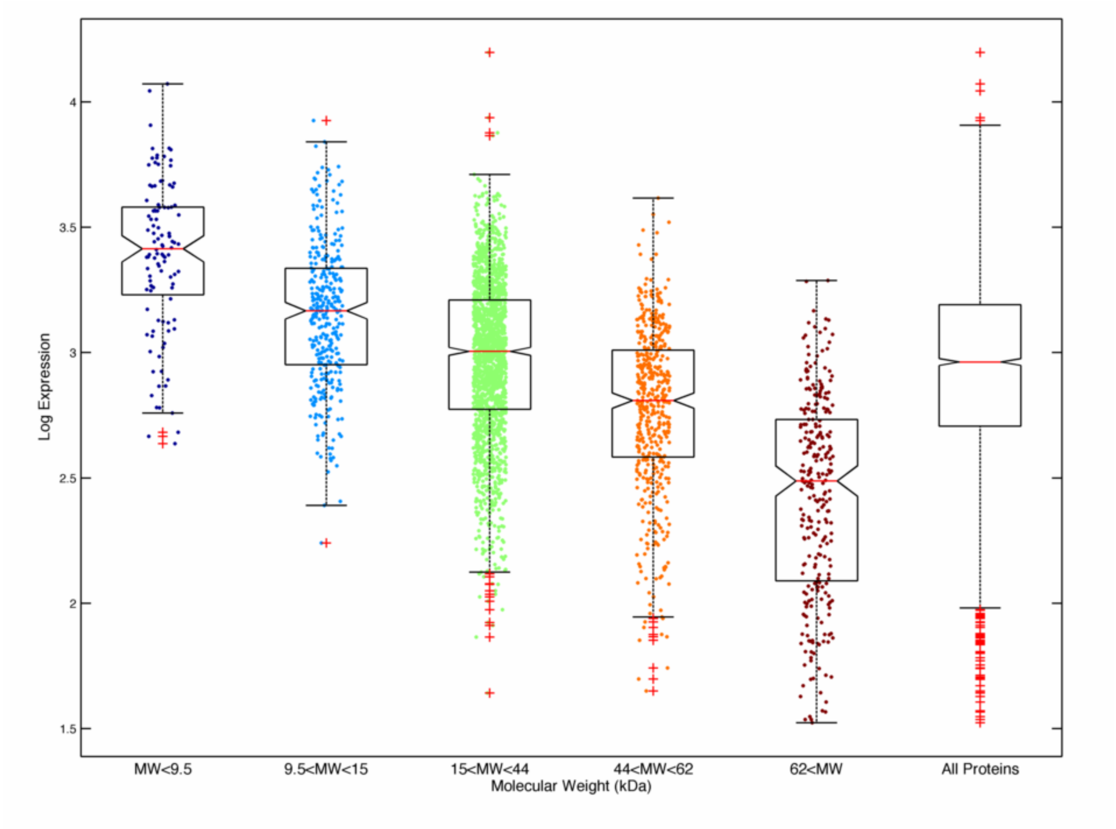


Figure 3.8: Expression vs Molecular Weight on the clustered results.

and "tAI longest rate limiting segment" are shown to be the best measures of protein production rate, without being affected by organism-specific instances. Even better is the fact that parameters required for tAI and RFM can be calculated much more easily for a novel organism compared to CAI (which requires the identification and quantification of the highly expressed genes). Since we show that cross-species predictions are possible to some extent, we suggest the use of such robust indices in future studies.

Generally speaking, protein abundance levels are determined by a balance between protein production and degradation rates. However, the protein degradation rates are unavailable in most of the analyzed cases. Consequently, any model to predict protein abundance must average out the effect of protein degradation and focus on the contribution of the production rate to the determination of protein abundance levels. Even though features based on PEST motifs and hydrophobic surface patches are shown to increase degradation rates, the predictive capabilities are not equal among all hosts. Further studies may be necessary, along with large-scale experiments that quantify the half-life of different proteins to be able to also capture the effects of degradation.

Finally, we propose the first available system capable of predicting protein abundance, with the ability to predict both homologous and heterologous expression for different hosts. Such a prediction tool can simplify the technical aspects of protein expression, and can be used to filter out infeasible targets, select suitable hosts for a target and can be used for codon optimization by detecting the rate limiting factors of translation efficiency. Coupling an optimization procedure to this prediction step can yield a tool that automatically optimizes a given nucleotide sequence for a given host.

4 PARTIAL PERIODIC PATTERN MINING FOR SHORT MOTIFS

4.1 Introduction

Finding recurring motifs is an important problem in bioinformatics. Such motifs can be used for any number of problems including sequence classification, label prediction, knowledge discovery and artificial sequence synthesis that fit a specific purpose.

However, discovery and representation of gapped short motifs is still an important problem. The main shortcoming of the existing prediction methods is their dependence on fixed length motifs. Most machine learning methods require the sequences to be of the same length (or aligned beforehand using multiple sequence alignment) to successfully discover the motifs. Multiple sequence alignment is not robust enough to capture short, weak motifs in a large set of relatively long sequences. While there are motif mining algorithms that can work with unaligned sequences [158, 159], the rigid constraints on the motif structure make it difficult to find gapped, weak partial motifs. In order to overcome this limitation, we propose the use of time-based motif mining methods that work position-independently.

This chapter explains the partial periodic pattern mining algorithm, a length-independent and alignment-free motif mining method which can also be used to find discriminative, class-specific motifs. Given a set of sequences, our algorithm will give a list of over-represented motifs (compared to a background set, or as shown in this chapter, in a discriminative manner). These motifs can be used in conjunction with machine learning methods for the prediction of any label or quantitative value that is correlated with the sequence motifs.

An important problem for motif-mining is the MHC binding prediction for a given peptide. MHC (Major Histocompatibility Complex) is a key player in the immune response of most vertebrates. The computational prediction of whether a given antigenic peptide will bind to a specific MHC allele is important in the development of vaccines for emerging pathogens, the creation of possibilities for controlling immune response, and for the applications of immunotherapy. One of the problems that make this computational prediction difficult is the detection of the binding core region in peptides, coupled with the presence of bulges and loops causing variations in the total sequence length. Due to the short peptide length, length variance within the peptides and very high promiscuity of the MHC molecules, defining a clear binding rule is usually not possible. These problems make the MHC binding prediction an excellent test case for our algorithm.

The partial periodic pattern mining method was tested on a benchmark set of 28 different alleles for MHC class I and 27 different alleles for MHC class II. The obtained results are comparable to the state of the art methods for both MHC classes, surpassing the published results for some alleles. The average prediction AUC values are 0.897 for class I, and 0.858 for class II.

We conclude that temporal motif mining using partial periodic patterns can capture information about the sequences well enough to predict the binding of the peptides and is comparable to state of the art methods in the literature. Unlike neural networks or matrix based predictors, our proposed method does not depend on peptide length and can work with both short and long fragments. This advantage allows better use of the available training data and the prediction of peptides of uncommon lengths.

Because the motif mining and prediction steps are uncoupled, the method can be used for different purposes. Other than MHC binding predictions, our algorithm can also be applied to find motifs in gapped sequences. The algorithm is presented here for use with amino acid sequences; however it can run on any linear sequence with arbitrary alphabets, such as nucleic acid sequences, reduced amino acid alphabets, and even alphabets from different representations (e.g. protein blocks and bond orientational order labels defined in Chapter 2).

4.2 Background

MHC (Major Histocompatibility Complex) is a large gene family with an important role in the immune system, autoimmunity, and reproduction. MHC molecules assume roles in the presentation of peptides, including self and non-self (antigenic) on their surface to T-cells. T-cells recognize antigenic peptides and trigger a cascade of events which leads to the destruction of pathogens and infected cells. Since MHCs have a key role in immune response, they are critical in many diseases, and can be used for controlling specific immunological processes by creating peptides to bind to specific MHC alleles. This binding affinity to specific peptides may be exploited for creating peptide vaccines for emerging pathogens [160], suppressing specific alleles in organ transplants [161, 162], and many other possible areas in immunotherapy.

MHC class I molecules bind short peptides, whose N- and C-terminal ends are anchored into the pockets located at the ends of the peptide binding groove

[163]. While the majority of the peptides are of length 9, longer peptides can be accommodated by the bulging of their central portion [164, 165], resulting in binding peptides of length 8 to 15 [166]. Peptides binding to class II proteins are not constrained in size [167, 168] and can vary from 11 to 30 amino acids long [169]. The peptide binding groove in the MHC class II molecules is open at both ends, which enables binding of peptides with relatively longer length. Though the core nine residues long segment contributes the most to the recognition of the peptide, the flanking regions are also important for the specificity of the peptide to the class II allele [170, 171]. MHC molecules bind peptides with high promiscuity; it is estimated that each HLA (human leukocyte antigen system) protein can bind between 1000 and 10,000 peptides for class I allotypes [172] and more than 2000 peptides for class II allotypes [173]. Thus, the large number of possible structures makes it unfeasible to find peptides that will bind to a specific allele using solely an experimental approach.

Computational methods for prediction of the binding affinity of a peptide to an MHC allele are based on three main artificial learning systems: statistical, structural, and neural methods [174, 175, 172]. The combination of these models is also common [176]. Computational approaches available for predicting MHC binding peptides from amino acid sequences include: (i) Motif-based methods such as methods that use a position weight matrix (PWM) to model a gapless multiple sequence alignment of MHC binding peptides, and a statistical approach based on Hidden Markov Models (HMMs); (ii) Machine learning methods based on Artificial Neural Networks (ANN) and Support Vector Machines (SVMs); (iii) Semi-supervised machine learning methods. Existing methods are reviewed in detail in [177, 178].

The formation of bulges and loops may allow peptides that are shorter or longer than 9 amino acids to bind to class I alleles. This length variance shifts the positions of amino acids in anchor locations, causing position-specific scoring matrices or other position-dependent methods to fail. Most existing methods enforce a length constraint of 9 peptides for class I prediction. ANN, quantitative matrices and similar methods require the peptides to be of the same length, with appropriate peptides aligned in the same location. Peptides of different lengths are either ignored or grouped into separate datasets by their length. This step may not always be feasible if the data is limited, especially for short and variable peptides.

Unlike MHC class I prediction methods, most of the MHC class II prediction methods can utilize peptides of variable length. However, the prediction strategy requires the determination of the core 9-mer region of the peptide. This core segment is assumed to be fixed-length and the possibility of longer binding core sequences is disregarded. Although peptides bind to MHC class II alleles mostly by the anchor residues, the interactions of the flanking regions may be important for specificity and therefore have to be taken into account [179].

In order to overcome these obstacles, we suggest a method using partial periodic pattern mining, which does not require the peptides to be of same length or the anchor positions to be specific. We propose a novel method for extracting the motifs on peptides with variable lengths by finding partial motifs in sequence data. Our method, called MHC-PPM, may capture aforementioned variations in peptides, without filtering or pre-processing the shorter/longer peptides or treating them as separate datasets. Additionally, the information in the flanking regions of the core 9-mers is taken into account without any information loss that may have arisen due to length constraints.

4.3 Methods

4.3.1 Dataset

We used 28 different alleles from the Immune Epitope Database (IEDB) benchmark dataset by Peters et al. [180, 181] for MHC class I prediction (total of 36,829 peptides). For MHC class II, we used two benchmark sets from Wang et al., 16 alleles containing 10,017 peptides [177] (referred to as Wang2008), and 26 alleles containing 44,541 peptides [182] (referred to as Wang2010). Wang 2010 contains data from several different human alleles, including HLA DR, DP and DQ. Wang2010 data also contains a similarity reduced subset (SR), where sequence similarity is minimized in order to reduce the overlap between cross-validation folds. In Peters and Wang2010 datasets, the same cross-validation folds are used for comparison to the benchmark values. 10-fold cross-validation was used in Wang2008 dataset.

The peptides from these alleles are assigned into positive and negative classes by the IC50 = 500 nM cut-off. Unlike other MHC prediction methods, no filtering was made with regard to length during the motif mining and prediction steps.

4.3.2 Motif Mining

4.3.2.1 Apriori Method

Our motif mining method is based on the apriori algorithm used in frequent association rule discovery [183]. An itemset is defined as a set of items or events that co-occur frequently. The Apriori algorithm uses the principle that all subsets of a frequent itemset must also be frequent. Accordingly, the algorithm has a bottom-up approach where the shorter frequent itemsets are extended to create longer candidates, which are then filtered by frequency of occurrence [183, 184, 185]. This iterative extension process continues until no frequent itemsets of a certain length can be found.

Due to the context difference, the formal statement of the problem in the Apriori algorithm [184] is slightly modified. Let I = i1,i2,...,im be an alphabet of items called *events* (amino acids in our case). Let D be a set of sequences, where each sequence S is an ordered set of items such that $S \subseteq I$. A sequence S contains *itemset* X, an ordered set of some items in I, if $X \subseteq S$. A rule is of the form $X \rightarrow Y$, where $X \subset I$, $Y \subset I$. With temporal information, $X \rightarrow Y$ also implies that the events in X occur before Y in a sequence S containing the rule.

The ratio of the sequences containing the association rule to all of the sequences is called the support of the rule. The ratio of the sequences containing a new rule created by the combination of two rules to the sequences containing the previous rule is called the confidence. That is,

$$Conf(X \rightarrow Y) = \frac{\text{Support}\,(X \bigcup Y)}{\text{Support}\,(X)} \tag{4.1}$$

Our motif mining method (MHC-PPM) is similar to temporal event mining in time-related databases [186]. In general, the partial periodic pattern mining algorithms for time series data will attempt to find frequently co-occurring events, or causality relationships between them. These methods try to capture the patterns which occur in an order which is not necessarily a consecutive one. In the domain of protein motifs, the amino acids become the events and the causality/future prediction aspects become the motifs that are sought [187].

In the proposed approach, each sequence is taken as a separate time series, with many parallel events occurring at the same time, with each event related only to the sequence upon which it is found. In these time series, if an event happens frequently after another one within a given time window, this frequent occurrences considered an episode of events, a motif. To exploit the apriori principle for performance, the motifs begin from length 1. A longer motif including a specific amino acid will have support less than or equal to the support of that amino acid. Hence, if an amino acid is infrequent, any motif that includes that amino acid will also be infrequent. Thus, iteratively L_N (frequent itemset of size N) is created from filtering of C_N, candidate itemset of size N by $C_N = L_{(N-1)} \rightarrow L_1$.

First L_1, the frequent itemsets of size 1 (i.e. amino acids) are found. The first step is straightforward: only the amino acids within the sequences are counted, and if an amino acids frequency (support of the rule) is below the given threshold, the amino acid is filtered out.

Then the candidate set of size 2, C_2 is created from the amino acids by $L_1 \rightarrow L_1$, that is, the combination of any two frequent itemsets of size 1. For example, if all of the 20 amino acids were frequent, we would have 400 candidate rules at C_2 for the given parameters. Those candidate rules would then be filtered according to the preset minimum support values, yielding L_2. Only a handful of those 400 rules would be frequent in the data. An example rule of size 2 would be $L \rightarrow V$, which represents Leucine followed by Valine in a window specified by parameters. The support of this candidate rule will be the ratio of occurrence of $L \rightarrow V$ to all of the sequences, and the confidence of the rule would be the ratio of occurrence of $L \rightarrow V$ to all of the sequences that contain L at some point. In other words, confidence would be the conditional probability of seeing Valine in the window, given that we observed a Leucine.

To account for the position variations in the alleles, a specific window should be defined. If an amino acid X is followed by Y after at least $MinS$ and at most $MaxS$ positions, then the rule $X \rightarrow Y$ is present in that sequence. If these amino acids co-occur within this window by this specific order at least minimum support times, then it is considered frequent.

In the motif mining context, the frequent rules are not simply association rules as in a shopping basket analysis; items also have a temporal value, which is used for relations such as "before" and "after" ("simultaneously" is not used in protein motifs since at each time point, that is a specific position in the sequence, only one amino acid can occur). The episode $A \rightarrow B$ then becomes, "whenever the events in the rule A occur in a given sequence, event B is likely to occur within n to m positions after A, with P($A \bigcap B$) as p (*support*) and P(B — A) as c (*confidence*)".

There are two parameters, the slack length (s), which is the length after an event within which we do not look for a rule, and the window size (w), in which the consequent event may occur. Thus, MinS = s and MaxS = $s + w - 1$, and the rule is given as $A \rightarrow B$ (p, c) for parameters (s, w). An example motif mining step is given in Figure 4.1 and 4.3, the pseudocode of the algorithm is given in Algorithm 1.

In our simulations, we used a window size of 1 to 3 and slack length of -8 to 8, producing different rulesets. Negative slack values are taken by reversing the input sequences and applying the algorithm with the absolute value of the slack.

For $s = 1$ and $s = 1$, the rules that consist of consecutive/nearby amino acids were mined whereas for the larger values of s, the motifs consisting of amino acids

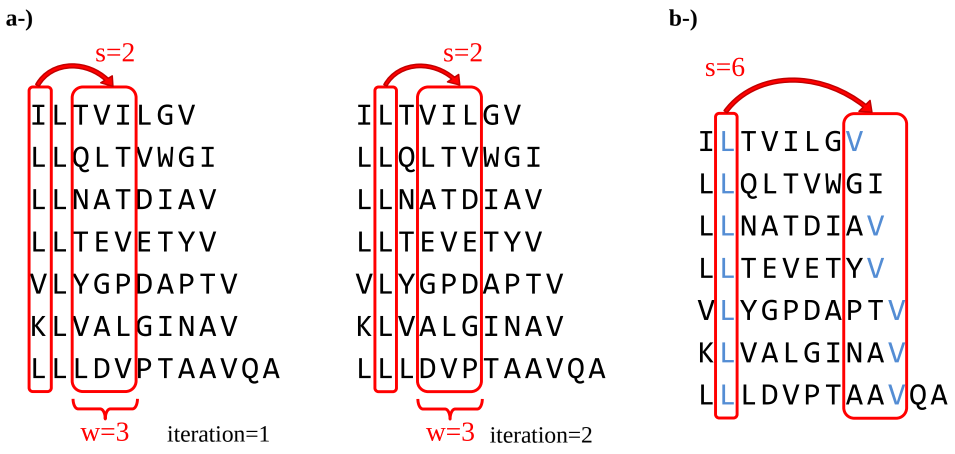


Figure 4.1:

An example of the temporal rule mining process.
a-) Schematic representation of the sliding windows approach on a sample set of sequences binding to MHC class I allele HLA A*0201. The windows are shifted with the given s and w values until all sequences are covered. The resulting rules are then filtered by their support. This process is repeated for all s values from -8 to -1 and 1 to 8.
b-) Representation of the L→V rule captured with the parameters s=6 and w=3. For HLA A*0201, Leucine in 2nd position and Valine around 9th position is a well-known binding motif [188]. Although this motif is present in 6 of the 7 sequences (support of 0.86), it is unlikely to be captured by position specific methods due to length variance and positional shifts.

at separate ends of the peptide were found. Since the anchor positions of MHC motifs may be different, different slack lengths are needed to mine them all.

4.3.3 Position dependent 1-rules

With the addition of position information, single amino acids can be employed as rules for anchors. When mining 1-rules, the position information is kept along with the window size. Thus, an example rule with window size of 2 may be L,

Algorithm 1 Partial Periodic Motif Mining algorithm

```
Input: $S \leftarrow$ List of sequences from which the motif will be mined
Input: $\varepsilon \leftarrow$ Minimum occurrence count of a rule to be considered frequent
Input: $s \leftarrow$ slack length
Input: $w \leftarrow$ window size
Output: All frequent rules inside $S$
  function TemporalApriori($S, \varepsilon, s, w$)
      $L_1 \leftarrow$ List of amino acids that appear more than $\varepsilon$ times in $S$
      $k \leftarrow 2$
      while $L_{k-1} \neq \varnothing$ do
          $C_k \leftarrow L_{k-1} + L_1$
          for all candidate $c$ in $C_k$ do
              for all sequence seq in $S$ do
                  if CheckMotif(seq, $c, s, w$) is true then
                      count[c] $\leftarrow$ count[c] $+ 1$
                  end if
              end for
          end for
          $L_k \leftarrow \{c : c \in C_k \wedge \text{ count[c]} \geq \varepsilon\}$
          $k \leftarrow k + 1$
      end while
      return $\bigcup_{k=2} L_k$
  end function

Input: $seq \leftarrow$ a sequence
Input: $c \leftarrow$ a rule that may or may not exist inside the sequence s
Input: $s \leftarrow$ slack length
Input: $w \leftarrow$ window size
Output: true if $c$ exists in $s$, false otherwise
  function CheckMotif(seq, $c, s, w$)
      $pos$[] $\leftarrow$ positions of $c[1]$ in seq
      for all position $k$ in $pos$[] do
          if CheckMotifRecursive($s, c, k + s, s, w$) then
              return true
          end if
      end for
      return false
  end function

  function CheckMotifRecursive(seq, $c, i, s, w$)
      $e \leftarrow c[i]$
      $pos$[] $\leftarrow$ positions of $e$ in $seq$
      for all  position $k$ in $pos$[] do
          if $e$ is the last event in $c$ then
              return true
          else if $k + s \leq$ length(seq) then
              $ss \leftarrow$ substring of $seq$ starting from $k + s$ to the end
              if CheckMotifRecursive($ss, c, i + 1, s, w$) is true then
                  return true
              end if
          end if
      end for
      return false
  end function
```

between positions 3-4 (support: p, confidence: c). This rule will be counted as present in a peptide which includes a Leucine between the positions 3 and 4.

4.3.4 Recursive Rule Mining on Training Set

In the rule mining process, the rules are mined for different slack lengths and finally 1-rules are added to the collection of rules. Following the rule mining process, all of the peptides in the training set are scored by the rules according to the Support-based prediction described below. After scoring every peptide in the training data, any peptide scoring below a predefined threshold is separated. Those separated peptides that are not sufficiently explained by the motifs are fed into the motif mining recursively.

This process can be thought of as mining rules for different clusters of sequences; the first iteration will try to capture the motifs for the cluster with the most sequences. After that, sequences that scored poorly will be used in motif mining again in the second iteration, and since the data is only a subset of the previous iteration, the limit for reaching minimum support will be lower. This process is repeated until the number of peptides that score lower than the threshold is below a predefined limit, until no more improvement can be gained by dividing the dataset or until a hard limit on iteration is reached. The supports for the newly mined rules are updated to reflect the support in all of the data, not the subset. An overall view of the recursive rule mining steps are given in Figure 4.2.

The recursive rule mining has advantages compared to setting the minimum support and confidence threshold to lower values and mining the rules in one pass. If the rules are mined in one pass with a very low support threshold, a greater number of rules will be found. Unless those rules are significant, the signal-to-noise ratio will decrease. By using a greater initial support value and progressively decreasing it on only a subset of data, the number of possible rules is reduced; if the first pass can capture motifs that are present in 70% of all sequences, we will only mine rules for explaining the remaining 30%, not the entire dataset. Hence, we end up with a lower number of more significant rules that explain the majority of the data.

4.3.5 Prediction

Before prediction, rules from both the binding and non-binding sequences are mined separately. During classification of an unknown peptide, the peptide is scored independently by both the binding and non-binding rules. The simplest classification method is the direct comparison of the scores for binding/non-binding rules. To calculate the scores, the support values of the rules that occur in the given peptide are summed for both classes. The peptide is predicted to belong to the class with a higher score. This naïve approach is called Support-based prediction and only used during the recursive rule mining step.

The presence of one significant negative motif can turn an otherwise strongly binding peptide to a non-binding one. For example, in the allele H-2Kd, charged or bulky amino acids inhibit binding when they are present at the 5^{th} position, even though the binding motif may also be present [189]. In a naïve prediction method, if the binding motifs are strong enough, the large number of binder rules will overpower the single negative motif, causing a false positive. Consequently, there is need for a way to predict these enhancing/inhibiting effects of the rules. Non-linear classification methods that intrinsically find the discriminant function on the feature space would fare better in such data.

For SVR-based prediction, motif mining is employed on the training data as described above. Using the motifs, a dataset is built by creating a binary matrix, where each row is a peptide and each column (feature) represents a motif. A cell has the value 1 if the peptide corresponding to that row includes the motif, otherwise 0. As additional columns, the sums of support and confidence scores for both the positive and negative classes are given. This data matrix is built for both the training and the test sets. Then, an SVR is trained on the training set, and the binding affinities of the peptides in the test set are predicted by the support vectors. The resulting binding affinity values can be converted into a binary class using an IC50 threshold where a binary class is required, such as feature selection methods or AUC calculation.

Since the training set is used in all of the rule mining, SVR training and parameter optimization steps, the prediction of the test set does not include any bias and represents the actual predictive performance of MHC-PPM.

The overall view of the prediction workflow can be seen in Figure 4.3.

4.4 Results and Conclusions

4.4.1 MHC class I

The prediction results of the proposed method are given in Table 4.1, along with the benchmark results for comparison [181]. The given values represent the area under the ROC curve (AUC) for the 5-fold cross validation using the same fold splits in the benchmark set.

Note that for some alleles (given in the top part of Table 4.1) the AUC values between the methods are not directly comparable because filtering of the data differs based on the prediction method in use. ANN [190] use only the 9-mers, the peptides of other lengths are filtered out. SMM uses 9-mers and 10-mers, but trained and tested independently (i.e. 9-mers belonging to an allele and 10-mers belonging to the same allele are taken as separate sets and are fed to different predictors). As stated, our method uses peptides of all lengths in the same classifier, without any filtering or separation. For comparison, in Table 4.1, the weighted average of AUC values using the 9-mer and 10-mer peptide counts are given for SMM [191] and ARB [192]. The results are directly comparable for alleles with only 9-mers.

Although superior in 9-mers, the main limitation of ANN is the need for the peptides to be of fixed length. The same constraint is also present in SMM and is overcome by using separate datasets for 9-mers and 10-mers. The main advantage of MHC-PPM is giving comparable and superior results to other methods without enforcing any constraints on peptide length. This flexible approach allows the use of information from all of the available data. Peptides that do not have enough representation in the dataset to train a separate classifier (e.g. 8 or 11 amino acids long) can still be predicted using the data from the 9 and 10-mers.

4.4.2 MHC class II

Results for Wang2008 [177] and Wang2010 [182] datasets are given in Table 4.2 and Table 4.3, respectively. Each method in the Table 4.3 has results for both all of the dataset (ALL) and a similarity-reduced version of the dataset (SR), used to decrease the sequence similarity between data folds.

In case of class II peptides, MHC-PPM is the top performer by the average score in the Wang2008 dataset benchmark results. However, as can be seen in the

Table 4.1:

Results of MHC-PPM in class I predictions in Peters dataset [181]. The best-performing method for each allele is underlined. The given AUC values for ARB and SMM are the weighted averages of the AUC values for 9-mers and 10-mers based on the given peptide counts for a specific allele. The alleles in the bottom part of the table were only trained & tested in 9-mers and are directly comparable.

	Allele	# Peptides		ANN	ARB	SMM	MHC-PPM
		9	10				
9-mers + 10-mers	HLA-A*0201	3089	1316	-	0.919	**0.939**	0.931
	HLA-A*0202	1447	1056	-	0.851	**0.879**	0.871
	HLA-A*0203	1443	1055	-	0.838	0.878	**0.882**
	HLA-A*0206	1437	1054	-	0.849	**0.890**	0.885
	HLA-A*0301	2094	1082	-	0.883	**0.915**	0.911
	HLA-A*1101	1985	1093	-	0.897	0.932	**0.937**
	HLA-A*2402	197	78	-	0.722	0.809	**0.833**
	HLA-A*3101	1869	1057	-	0.881	**0.903**	0.878
	HLA-A*3301	1140	1055	-	0.866	**0.888**	0.863
	HLA-A*6801	1141	1055	-	0.827	**0.874**	0.864
	HLA-B*0702	1262	205	-	0.925	0.952	**0.954**
	HLA-B*3501	736	177	-	0.833	**0.886**	0.866
	HLA-B*5101	244	177	-	0.782	0.875	**0.886**
	HLA-B*5301	254	177	-	0.758	**0.854**	0.847
9-mers	HLA-A*0101	1157	-	**0.982**	0.964	0.980	0.963
	HLA-A*2601	672	-	**0.956**	0.907	0.931	0.901
	HLA-A*2902	160	-	**0.935**	0.755	0.911	0.907
	HLA-A*6802	1434	-	**0.899**	0.865	0.898	0.867
	HLA-B*0801	708	-	**0.955**	0.936	0.943	0.926
	HLA-B*1501	978	-	0.941	0.900	**0.952**	0.922
	HLA-B*1801	118	-	0.838	0.573	0.853	**0.906**
	HLA-B*2705	969	-	0.938	0.915	**0.940**	0.938
	HLA-B*4002	118	-	0.754	0.541	0.842	**0.891**
	HLA-B*4402	119	-	0.778	0.533	0.740	**0.891**
	HLA-B*4403	119	-	0.763	0.461	0.770	**0.847**
	HLA-B*5401	255	-	0.903	0.847	**0.921**	0.883
	HLA-B*5701	59	-	0.826	0.428	0.871	**0.929**
	HLA-B*5801	988	-	0.961	0.889	**0.964**	0.944
	Average (All)			0.888	0.798	0.893	**0.897**
	Average (9mers)			0.888	0.751	0.894	**0.908**
	Weighted Avg			**0.932**	0.872	0.910	0.901

Wang2010 dataset, NN-align [193] outperforms all other methods when included in the comparison. Nonetheless, even though MHC-PPM is designed only to find position independent rules, and there are no external steps for core region detection (or any information about the core region length), it still performs exceptionally well with an average AUC value of 0.858, slightly above SMM-align [191] (AUC of 0.849) and only < 0.03 lower than NN-align (AUC of 0.882).

Unlike what has been observed in class I molecules, class II molecules are believed to bind only to the core 9-mer region of a peptide. Although the core region occupies the peptide binding groove, the non-bound N- and C-terminus residues that lie outside the MHC anchor residues, called peptide flanking residues (PFRs), have been shown to affect the binding affinity and stability [194, 170]. NN-align and SMM-align use the length and composition of the peptide flanking residues in addition to the peptide binding core sequence. However, to keep the same length of the input throughout the data, the flanking residues are encoded in a summarized form, decreasing the information content. Due to nature of our algorithm, the differences in affinity due to the PFRs can be captured without losing any information. To test that hypothesis, we used experimental affinity values of 9 sequences which have the same core sequence and differ only in the flanking regions [170] and tried to predict the binding affinity values from the sequence (Table 4.4). Although available data is limited, MHC-PPM has the lowest root mean squared error (RMSE). MHC-PPM also significantly outperforms ARB, SMM-align and NN-align in correlation of the predictions with the actual affinity values.

Table 4.2:

Results of MHC-PPM in class II predictions in Wang2008 dataset [177]. The (#) column gives the total number of peptides for the given allele. The best-performing method for each allele is underlined.

Allele	#	RANKPEP	ARB	PROPRED	SMM-align	MHCMIR	MHC-PPM
HLA-DRB1*0101	3882	0.700	0.760	0.740	0.770	0.810	**0.878**
HLA-DRB1*0301	502	0.670	0.660	0.650	0.690	0.640	**0.712**
HLA-DRB1*0401	512	0.630	0.670	0.690	0.680	**0.730**	0.666
HLA-DRB1*0404	449	0.660	0.720	0.790	0.750	0.730	**0.792**
HLA-DRB1*0405	457	0.620	0.670	**0.750**	0.690	0.730	0.734
HLA-DRB1*0701	505	0.580	0.690	0.780	0.780	0.830	**0.893**
HLA-DRB1*0802	245	-	0.740	0.770	0.750	0.740	**0.827**
HLA-DRB1*0901	412	0.610	0.620	-	0.660	0.620	**0.666**
HLA-DRB1*1101	520	0.700	0.730	0.800	0.810	0.810	**0.817**
HLA-DRB1*1302	289	0.520	**0.790**	0.580	0.690	0.720	0.679
HLA-DRB1*1501	520	0.620	0.700	0.720	0.740	0.730	**0.759**
HLA-DRB3*0101	420	-	0.590	-	0.680	-	**0.712**
HLA-DRB4*0101	245	0.650	0.740	-	0.710	0.760	**0.829**
HLA-DRB5*0101	520	0.730	0.700	0.790	0.750	0.710	**0.845**
H-2 IAb	500	0.740	**0.800**	-	0.750	0.690	0.786
H-2 IEd	39	0.830	-	-	-	-	**0.867**
Average		0.661	0.705	0.733	0.727	0.732	**0.779**
Weighted Avg		0.671	0.722	0.738	0.743	0.760	**0.780**

Table 4.3:

Results of MHC-PPM in MHC class II predictions in Wang2010 dataset [182]. Each method contains results from all of the peptides (ALL) and the similarity reduced data (SR). The best-performing method for each allele in ALL dataset is marked by bold and the best performing method in SR dataset is underlined.

Allele	# Peptides		ARB		SMM-align		NN-align		MHC-PPM	
	ALL	SR	ALL	SR	ALL	SR	ALL	SR	ALL	SR
HLA-DPA1*0103-DPB1*0201	1404	603	0.823	0.745	0.921	0.767	**0.943**	0.793	0.931	0.772
HLA-DPA1*01-DPB1*0401	1337	540	0.847	0.746	0.930	0.767	**0.947**	0.802	0.935	0.751
HLA-DPA1*0201-DPB1*0101	1399	604	0.824	0.743	0.909	0.786	**0.944**	0.818	0.938	0.806
HLA-DPA1*0201-DPB1*0501	1410	586	0.859	0.709	0.923	0.728	**0.956**	0.787	0.948	0.773
HLA-DPA1*0301-DPB1*0402	1407	602	0.821	0.771	0.932	0.818	**0.949**	0.828	0.935	0.815
HLA-DQA1*0101-DQB1*0501	1739	584	0.871	0.741	0.930	0.783	0.945	0.805	**0.949**	0.754
HLA-DQA1*0102-DQB1*0602	1629	593	0.777	0.708	0.838	0.734	**0.880**	0.762	0.842	0.730
HLA-DQA1*0301-DQB1*0302	1719	596	0.748	0.637	0.807	0.663	**0.851**	0.693	0.845	0.709
HLA-DQA1*0401-DQB1*0402	1701	585	0.845	0.643	0.896	0.761	**0.922**	0.742	0.920	0.778
HLA-DQA1*0501-DQB1*0201	1658	589	0.855	0.700	0.901	0.736	**0.932**	0.777	0.919	0.766
HLA-DQA1*0501-DQB1*0301	1689	602	0.844	0.756	0.910	0.801	**0.927**	0.811	0.915	0.771
HLA-DRB1*0101	6427	3504	0.770	0.710	0.798	0.756	**0.843**	0.763	0.821	0.758
HLA-DRB1*0301	1715	1136	0.753	0.728	0.852	0.808	**0.887**	0.829	0.828	0.747
HLA-DRB1*0401	1769	1221	0.731	0.668	0.781	0.721	**0.813**	0.734	0.763	0.711
HLA-DRB1*0404	577	474	0.707	0.681	0.816	0.789	0.823	0.803	**0.885**	0.717
HLA-DRB1*0405	1582	1049	0.771	0.716	0.822	0.767	**0.870**	0.794	0.831	0.734
HLA-DRB1*0701	1745	1175	0.767	0.736	0.834	0.796	**0.869**	0.811	0.846	0.804
HLA-DRB1*0802	1520	1017	0.702	0.649	0.741	0.689	**0.796**	0.698	0.752	0.687
HLA-DRB1*0901	1520	1042	0.747	0.654	0.765	0.696	**0.810**	0.713	0.762	0.671
HLA-DRB1*1101	1794	1204	0.800	0.777	0.864	0.829	**0.900**	0.847	0.858	0.811
HLA-DRB1*1302	1580	1070	0.727	0.667	0.797	0.754	**0.814**	0.732	0.768	0.717
HLA-DRB1*1501	1769	1171	0.763	0.696	0.796	0.741	**0.852**	0.756	0.813	0.745
HLA-DRB3*0101	1501	987	0.709	0.678	0.819	0.780	**0.856**	0.798	0.782	0.718
HLA-DRB4*0101	1521	1011	0.785	0.747	0.816	0.762	**0.870**	0.789	0.860	0.772
HLA-DRB5*0101	1769	1198	0.760	0.697	0.832	0.776	**0.886**	0.795	0.843	0.812
H-2-IAb	660	546	0.800	0.775	0.855	0.830	**0.858**	0.847	0.824	0.807
Average			0.785	0.711	0.849	0.763	**0.882**	0.782	0.858	0.755
WeightedAverage			0.784	0.709	0.843	0.762	**0.879**	0.778	0.853	0.754

Table 4.4:
Effect of flanking peptides on the binding affinity to HLA DRB1*1501 allele. Experimental affinity measurements are from [170]. Predictions of other values calculated from the IEDB website [180]. MHC-PPM has the lowest root mean squared error (RMSE) and has a correlation score approximately equal to the top performing method.

	Experimental	**ARB**	**NetMHCIIpan**	**SMM_align**	**NN_align**	**MHC-PPM**
Sequence	IC50(nM)	IC50(nM)	IC50(nM)	IC50(nM)	IC50(nM)	IC50(nM)
ENPVVHFFKNIVTPR	33	21.9	10	21	8	11
VVHFFKNIVHAAA	33	21.9	9.2	52	10.7	139
VVHFFKNIVTAAA	45	21.9	9.5	20	11.5	224
VVHFFKNIVT**K**AA	35	21.9	8.1	20	10.5	142
VVHFFKNIVTA**K**A	4	21.9	8.1	20	9.8	83
VVHFFKNIVTAA**K**	5	21.9	8.9	20	11.1	263
DAVVHFFKNITVA	326	82.5	23.6	25	23.8	316
A**D**VVHFFKNITVA	454	82.5	23.8	25	23.9	320
AA**D**VHFFKNITVA	264	1286.7	45	30	74.4	392
RMSE		371.90	190.78	191.84	187.13	**<u>134.35</u>**
Pearson's Corr.		0.349	**<u>0.728</u>**	0.041	0.540	0.721

4.5 Discussion

In this study we present a position independent motif mining method representing amino acid sequences as time series data to predict peptides binding to MHC class I and class II proteins.

In class I MHC-peptide complexes, peptides have been observed to bulge out of the binding groove [164, 165], shifting the peptide side chains in the binding pockets. The main shortcoming of the existing prediction methods is their dependence on fixed length motifs, even though peptides of various lengths are known to bind to class I molecules [166]. Although a separate predictor can be created by applying the same method on a dataset of peptides of a different length, there is usually not enough available data for uncommon sequence lengths. There have been methods that use random sampling of insertions and deletions to fit the peptide into the 9-length window for prediction [195], however the fixed length limitation still present in the core.

For MHC class I predictions, MHC-PPM has been shown to slightly outperform other methods on the average. However, all methods have very close scores and perform equally well. Our main advantage is the ability to use peptides of any length during both training and prediction phases. While the curated benchmark dataset contains only 9-mers and 10-mers for the given alleles, we expect MHC-PPM to fare better in a more diverse dataset.

Commonly used prediction servers give the consensus prediction of different algorithms. The addition of our predictions into a consensus-decision step with other state-of-the-art algorithms will almost certainly benefit the end-users; the overall accuracy for the 9-mers will increase, and longer peptides that would have been previously ignored (or treated as 9-mers) will also be evaluated.

On MHC class II molecules, MHC-PPM was the top performing one in Wang2008 dataset by average AUC and just below NN-align in Wang2010 dataset. Even though NN-align outperforms all other methods including ours, the difference in performance values are not as drastic. Due to the fixed size core region, length independence is not much of an issue during score calculation. On the other hand, the position independence allows the inherent detection of the core region and allows better representation of the peptide flanking residues. During the prediction of the effects of PFRs on binding affinity (Table 4.4), MHC-PPM resulted in the highest agreement with the experimental data, though more data is required for conclusive results.

On the subject of peptides binding to MHC class II molecules, the current view is that the peptides lie on a shallow groove with multiple contacts along the entire length of the peptide binding groove [196, 168]. This view does not address the possibility of peptides bulging out from the groove. There have been studies that proposed examples of peptide bulging (i.e. core binding region longer than 9 amino acids) in class II molecules for several alleles [197, 198, 199, 168]. Even though it is not known whether this is a general phenomenon for all class II alleles, it is possible that certain alleles can anchor peptides sufficiently at their N- and C-terminals to allow bulges, similar to class I molecules. If that is the case, a length insensitive method is required to correctly identify such examples, since NN-align and other methods require a fixed length core sequence.

The strength of MHC-PPM is its ability to capture length independent short motifs that are in close vicinity. Because the motif mining and prediction steps are uncoupled, the method can be used for different purposes. We have shown that the rules mined from the data can be used in conjunction with support vector machines or neural networks for non-linear prediction of any label (or quantitative value) that is correlated with the sequence motifs. However, the actual output of the algorithm is a collection of human-understandable rules and those motifs can be used as templates during sequence analysis or synthesis. Other than MHC binding predictions, the MHC-PPM method can also be applied to find motifs in gapped sequences, such as TCR recognition or receptor-ligand prediction problems. It is straightforward to extend the method to mine multiple groups of short sequence motifs (separated by relatively long distances) which co-occur frequently. We believe this approach can help uncover previously overlooked subtle sequence motifs in any large scale data.

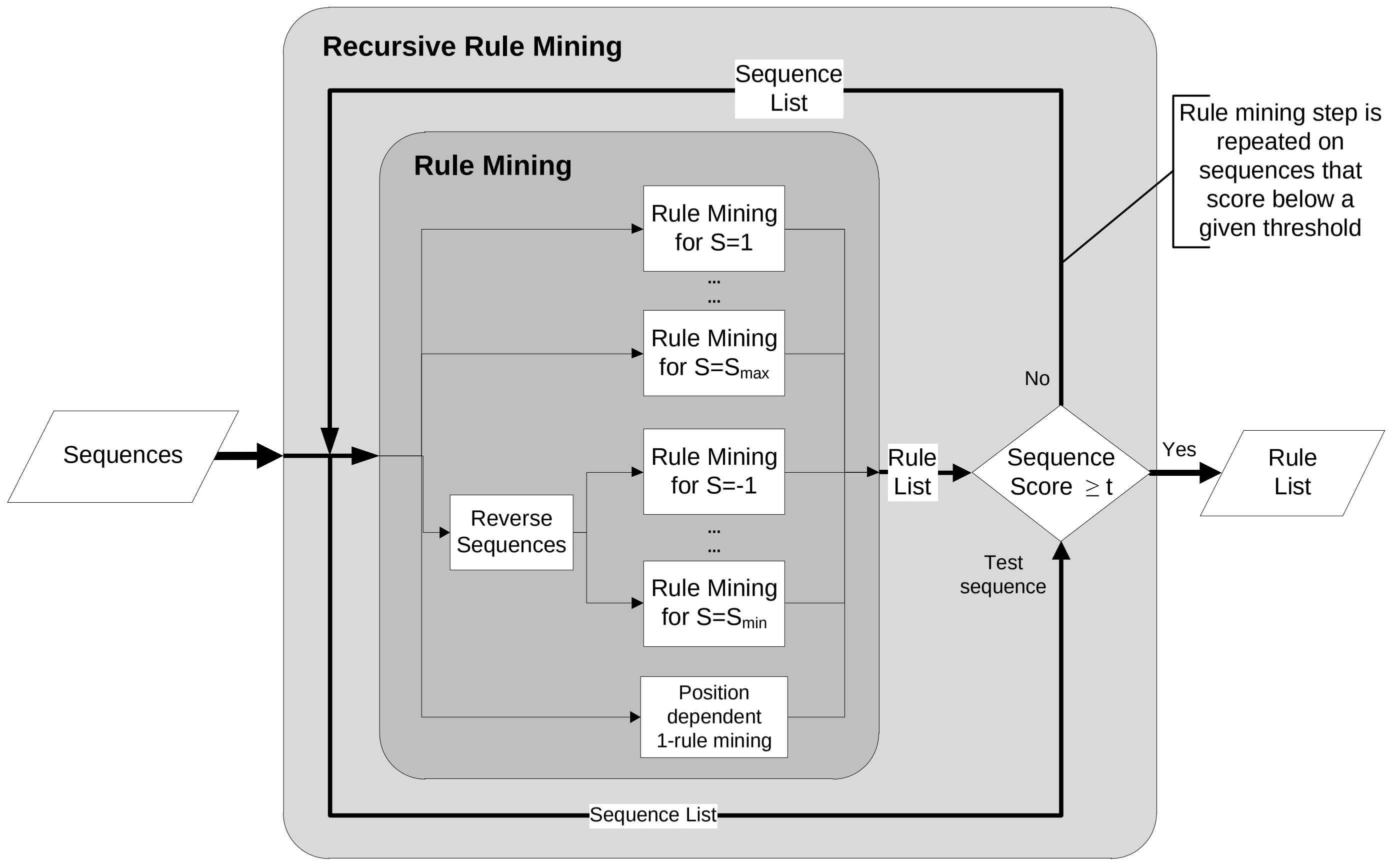


Figure 4.2:
Overall flow of the recursive rule mining step.

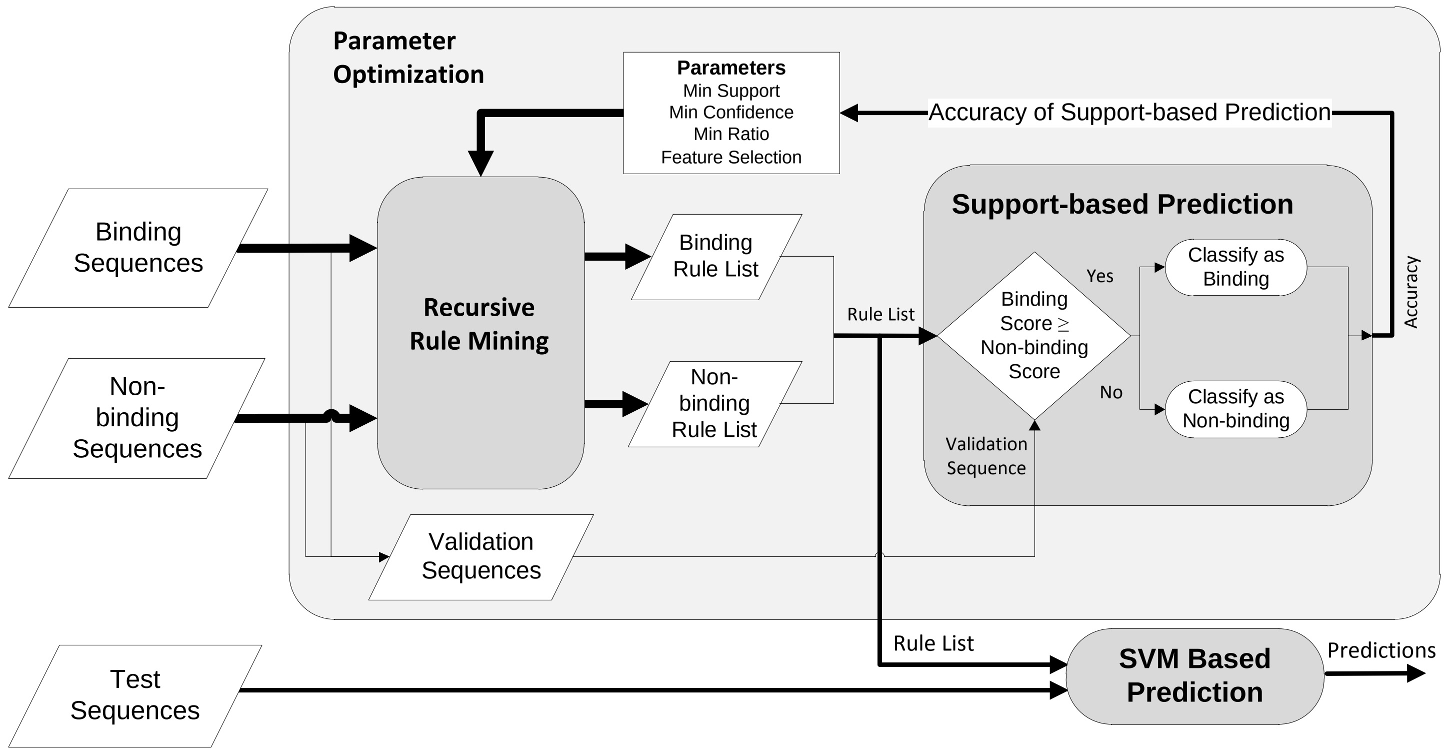


Figure 4.3:
Overview of the experimentation process.

$$C_1 = \begin{cases} \boldsymbol{L,} & \boldsymbol{Supp: 7/7} \\ \boldsymbol{V,} & \boldsymbol{Supp: 7/7} \\ \boldsymbol{T,} & \boldsymbol{Supp: 6/7} \\ \boldsymbol{G,} & \boldsymbol{Supp: 4/7} \\ \boldsymbol{I,} & \boldsymbol{Supp: 4/7} \\ \boldsymbol{A,} & \boldsymbol{Supp: 4/7} \\ \boldsymbol{D,} & \boldsymbol{Supp: 3/7} \\ Q, & Supp: 2/7 \\ N, & Supp: 2/7 \\ Y, & Supp: 2/7 \\ P, & Supp: 2/7 \\ E, & Supp: 1/7 \\ W, & Supp: 1/7 \\ K, & Supp: 1/7 \end{cases} \rightarrow \quad L_1 = \begin{cases} L, & Supp: 7/7 \\ V, & Supp: 7/7 \\ T, & Supp: 6/7 \\ G, & Supp: 4/7 \\ I, & Supp: 4/7 \\ A, & Supp: 4/7 \\ D, & Supp: 3/7 \end{cases}$$

$$C_2 = L_1 \rightarrow L_1 = \begin{cases} \boldsymbol{L \rightarrow L, Supp: 0.57} \\ \boldsymbol{L \rightarrow V, Supp: 0.57} \\ \boldsymbol{L \rightarrow T, Supp: 0.57} \\ \boldsymbol{L \rightarrow G, Supp: 0.43} \\ L \rightarrow I, Supp: 0.29 \\ \quad\quad\quad \dots \\ V \rightarrow L, Supp: 0.29 \\ V \rightarrow V, Supp: 0.29 \\ V \rightarrow T, Supp: 0.29 \\ \boldsymbol{V \rightarrow G, Supp: 0.57} \\ \quad\quad\quad \dots \end{cases} \rightarrow \quad L_2 = \begin{cases} L \rightarrow L, Supp: 0.57, Conf: 0.57 \\ L \rightarrow T, Supp: 0.57, Conf: 0.57 \\ L \rightarrow V, Supp: 0.57, Conf: 0.57 \\ V \rightarrow G, Supp: 0.57, Conf: 0.57 \\ T \rightarrow I, Supp: 0.43, Conf: 0.50 \\ T \rightarrow V, Supp: 0.43, Conf: 0.50 \\ I \rightarrow V, Supp: 0.43, Conf: 0.75 \\ L \rightarrow G, Supp: 0.43, Conf: 0.43 \\ L \rightarrow D, Supp: 0.43, Conf: 0.43 \end{cases}$$

$$C_3 = L_2 \rightarrow L_1 = \begin{cases} L - L \rightarrow L, Supp: 0.00 \\ \boldsymbol{L - L \rightarrow V, Supp: 0.43} \\ L - L \rightarrow T, Supp: 0.14 \\ \quad\quad\quad \dots \\ L - T \rightarrow L, Supp: 0.00 \\ \boldsymbol{L - T \rightarrow V, Supp: 0.43} \\ L - T \rightarrow T, Supp: 0.14 \\ \quad\quad\quad \dots \end{cases} \rightarrow \quad L_3 = \begin{cases} L - L \rightarrow V, Supp: 0.43, Conf: 0.75 \\ L - T \rightarrow V, Supp: 0.43, Conf: 0.75 \end{cases}$$

$$C_4 = L_3 \rightarrow L_1 = \begin{cases} L - L - V \rightarrow L, Supp: 0.00 \\ L - L - V \rightarrow V, Supp: 0.00 \\ L - L - V \rightarrow T, Supp: 0.14 \\ \quad\quad\quad \dots \\ L - T - V \rightarrow L, Supp: 0.00 \\ L - T - V \rightarrow V, Supp: 0.14 \\ L - T - V \rightarrow T, Supp: 0.14 \\ \quad\quad\quad \dots \end{cases} \rightarrow \quad L_4 = \emptyset$$

Figure 4.4:

A sample run of the algorithm. The candidate and frequent itemsets of all lengths for the given example sequences in Figure 4.1, for minimum support value of 0.4. Red/bold values represent rules above the support threshold. At each step a candidate set C_k is generated by extending the last frequent itemset L_{k-1}, then the candidates are filtered according to the support values to generate the frequent itemset L_k. This process is repeated until no frequent itemsets of a certain size can be found. Afterwards, the resulting frequent sets of different sizes (except L_1) are merged together and filtered according to a given minimum confidence boundary.

5 SEMI-SUPERVISED LEARNING OF DISCRIMINATIVE MOTIFS WITH DEEP BELIEF NETWORKS

5.1 Introduction

When dealing with a large number of long sequences, finding a relatively small region that are common among all the inputs becomes problematic due to mainly two reasons. One is, the motif can be in any position within the inputs, e.g. at the start of protein 1, at the middle of protein 2, near the end of protein 3 and so on. Second problem is, the motif can be of any form, the motif description matrix can potentially take on an astronomical number of possible states.

Because of these two problems, majority of the methods in literature make use of the position constraint to find motifs. To create the motif definition, as a PSSM (position specific scoring matrix) or pHMM (profile hidden Markov models), such methods require an input alignment. That is, given an alignment, we can create the motif easily since the alignment acts as anchoring the specific positions of the motif to be fixed along the input sequences. Or, for other methods, given a motif description, we can search it among all the sequences and find its positions.

The actual problem begins if we don't have the alignment nor the motif description. If the inputs do not have a strong global motif profile but a local one, multiple alignment will fail; unlike multiple global alignment, multiple local alignment is computationally intractable to be efficiently solved for anything more than a few sequences. This problem requires finding the motif position and definition concurrently.

There are a few algorithms that deal with this problem. Gibbs sampling is used in motif mining for such purposes [200], but it has its limitations; while it allows mutations in the motif, insertions/gaps are not allowed, it requires the exact motif length to be known, it assumes exactly 1 copy of motif is present in each sequence, and finally, it does not find discriminative motifs (motifs that differentiate between two classes). A popular method that solves many of these problems is MEME (Multiple EM for Motif Elicitation) [201, 158]. MEME can decide on the optimal motif length, can work with any number of occurrences in each sequence (exactly 1 copy, 0 or 1 copy, 0 to N copy per sequence) and can find discriminative motifs (for two classes). However, it still lacks the ability to find motifs that include insertion/deletions.

We propose a new approach based on deep learning to possibly extract discriminative motifs (for any number of classes) that can include short or long stretches of insertion/deletions in its definition, without requiring user input.

Deep learning is an area of machine learning which utilizes a set of hierarchical learners that operate in a sequential fashion. The motivation behind the idea is inspired by the hierarchical architecture of the neocortex in the mammalian brains. This kind of layered, "deep" approach allows learning new representations of the raw input data, which are then fed to the next learner. Since the higher levels use the processed, informative features extracted from the input instead of its raw form, they can perform decision making in a much more abstract level.

Even though the idea of deep, hierarchical learning dates back to 1980 [202], practical issues stopped it from gaining ground. Training neural networks with multiple layers is computationally very complex. Training such a network using backpropagation is futile, because the error gradient that is used to modify the weights get progressively more dilute as it travels down the network. With such a minimal correction signal, the network tends to stuck in a local minima. It has been shown that without proper weight initialization, deep networks perform worse than shallow networks [203].

However, recently Hinton et al. described a fast algorithm for training a network in an unsupervised manner [11]. Moreover, each layer can be trained separately in a greedy manner, and then stacked together to form what is called Deep Belief Networks (DBN). The unsupervised greedy layer-wise training step helps in initializing the network with stable weights. With those initial weights in place, backpropagation can train the network much more efficiently, without getting stuck in local minima as frequently.

In the last few years, deep belief networks and similar approaches became the state-of-the-art machine learning methods for image and sound based learning. However, they haven't been applied to the biological problems, and with good reasons. The first problem is, proteins are not of fixed length. In the nature of the classification, fixed length inputs are nearly a universal requirement. Proteins are also highly variable in their composition and structure, and can be either very robust or very fragile to slight changes in their make-up depending on the context, e.g. some proteins can conserve their fold and function despite a great number of mutations, some proteins can retain their overall 3D structure but lose their activity with very few number of mutations, and some proteins will completely misfold even with one mutation. This disproportionate relationship between the input (e.g. primary structure for this case) and the output (3D structure, function, dynamics etc.) makes it very hard to create a non-case-specific machine learning method to find motifs in protein structures.

We propose some work-arounds and solutions to these problems. Combining the representations from Chapters 2 with our approach, we performed deep learning methods to classify, cluster, and finally, find discriminative length-independent motifs between in any set of input protein data for both sequence and structural representations. Due to the very general approach presented here, our algorithm was not developed for a specific problem or representation. We present its application to a set of very diverse problems to show its feasibility and performance.

5.2 Background

5.2.1 Semi-supervised learning

We described supervised and unsupervised learning in Chapter 1. Semi-supervised learning is a mixture of the two. Semi-supervised learning includes the unlabeled instances during learning to settle on a better decision boundary. To see an example of how unlabeled data can help during classification, see Figure 5.1. Using only the supervised data, there is not enough information about the background probability distribution to fine-tune the decision boundary. By addition of the unlabeled instances, we can better estimate the probability distribution of the samples and find a much better decision boundary between the classes. Essentially, while the labeled instances are used with supervised learning to create class boundaries, the unlabeled instances give a priori information about the distribution of the data itself.

The importance of semi-supervised learning is apparent in machine learning when there is a lack of labeled data. Without a large number of training examples, even the best machine learning methods will perform poorly. In most domains, getting a sufficient number of correctly labeled training instances is a very problematic task within itself, e.g. structure data for protein that show a specific activity, genomics data for patients in a clinical setting, enzymes with a specific attribute we are interested in... Such data might be expensive, time consuming and potentially infeasible to collect in a large scale.

In such cases where the labeled data is limited, the ability to use unlabeled data becomes really important. The huge amount of data present in the global databases might be utilized to capture possible motifs, spatial and structural relationships that are present in the whole biological domain. Then, after learning about "what a protein is" in a global context, the machine learner might be able to differentiate the smaller number of labeled instances using the alphabet of descriptors (that are possible among all proteins) it learned beforehand.

5.2.2 Energy Based Learning

Energy-Based Models

Energy-Based Models (EBM) capture dependencies between variables by associating an energy value to each configuration of the variables [204]. Let us consider a model with two variables, X and Y. The energy-based model is an energy function $E(X, Y)$, which measures the "goodness" (or badness) of each possible conguration of X and Y. If the joint probability P(X, Y) is high, E(X, Y) will result in a small energy value. That is highly compatible congurations of the variables correspond to small energies whereas highly incompatible configurations of X and Y will result in large energy values.

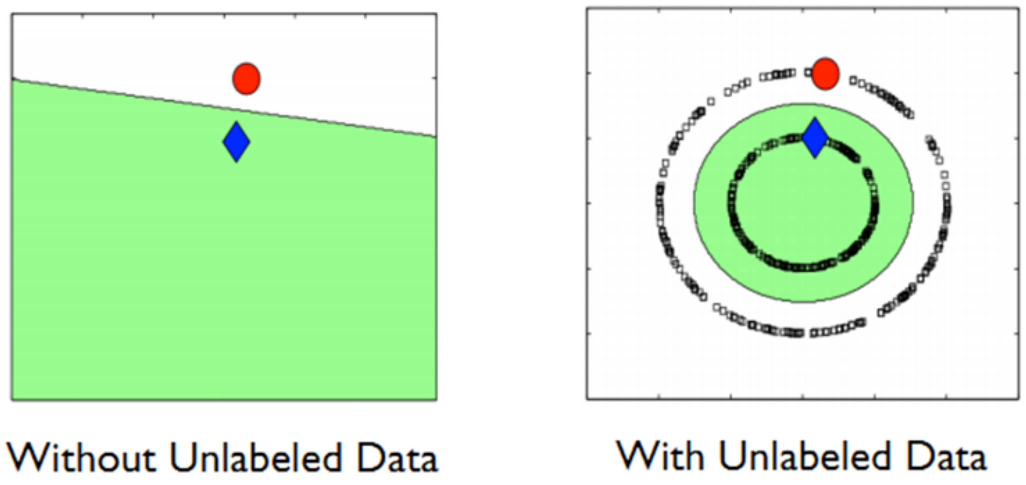


Figure 5.1:
An example case of semi-supervised learning where using unlabeled data would improve the decision boundary that is estimated from the labeled instances.

If we are given the value of X, the energy function can be used for prediction of the Y value Y^* that is most compatible with the given X, i.e. E(X, Y) is minimized.

$$Y^* = \text{argmin}_{Y \in \Upsilon} E(X, Y) \tag{5.1}$$

where Υ is the set of possible values Y can take. Since this approach compares the energy values within themselves, we don't need to normalize this energy value. But if required, we can define a partition function by integrating the energy values for all (X, Y) using Gibbs distribution, and use the partition function to convert the energies into a probability-like scale between [0, 1] [204].

Latent Variables

We can increase the descriptive power of the energy function by adding latent variables. That is, we may not be able to observe the actual "X" value, or assume there are non-observed variables that change the relationship (thus, energy) of X and Y. Using these "hidden" variables (Z), we will have an energy function defined as E(X, Y, Z).

To give an example, assume the Y variable to be the binding energy between two proteins (a and b) that are defined by the feature set X ($X = X_a \cup X_b$). If we have a set of experimental data that gives us the relationship between Y and X, we might learn an energy function E(X, Y) and use this EBM for prediction and inference. However, we might encounter problems finding E(X, Y) that can predict Y^* values that perfectly correlate with the experimental Y values. In such cases, the hidden variables act as the excluded or unseen/unobserved variables that are present in the data. For example, we can define the hidden variable Z as the folding state of protein a, which affects the binding energy, but is not directly observable.

Since Z is never observed, we can take the energy between X and Y by

$$E(X, Y) = \min_{Z \in \boldsymbol{z}} E(X, Y, Z) \tag{5.2}$$

The prediction will then become;

$$Y^* = \text{argmin}_{Y \in \Upsilon, Z \in \boldsymbol{z}} E(X, Y, Z) \tag{5.3}$$

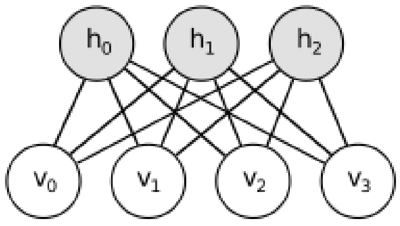


Figure 5.2: A visualization of a Restricted Boltzmann Machine layer

Learning in EBM

Learning the energy function can be carried out by minimizing the empirical negative log-likelihood of the training data using stochastic gradient descent. Due to the difficulty in calculating the gradient of the negative log-likelihood function when dealing with hidden variables, we consider a specific type of EBM called Restricted Boltzmann Machines (RBM).

Restricted Boltzmann Machines (RBM)

Boltzmann Machines (BMs) are a particular form of log-linear Markov Random Field, that is, their energy function is linear in its free parameters. We can add hidden variables to increase the modeling capacity of the BM enough to represent complicated distributions. We will represent the visible (observed) features as v, and the latent hidden variables as h. Restricted Boltzmann Machines further restrict BMs to those without visible-visible and hidden-hidden connections. A graphical depiction of an RBM is shown below.

The energy function $E(v, h)$ of an RBM is defined as:

$$E(v, h) = -b'v - c'h - h'Wv \tag{5.4}$$

where W represents the weights connecting hidden and visible units and b, c

are the offsets of the visible and hidden layers respectively.

Because of the specific structure of RBMs, visible and hidden units are conditionally independent given one-another. Using this property, we can write:

$$P(h|v) = \prod_i P(h_i|v) \tag{5.5}$$

$$P(v|h) = \prod_j P(v_j|h) \tag{5.6}$$

We can further simplify the input and output of the RBMs by restricting the visible and hidden states to take on binary values. Thus, the relationship between those hidden units define a Bernoulli distribution. The neuron activation function becomes a probabilistic function from Bernoulli distribution.

$$P(h_i = 1|v) = sigm(c_i + W_i v) \tag{5.7}$$

The free energy of an RBM with binary units (i.e. the backward pass) is

$$P(v_j = 1|h) = sigm(b_j + W'_j h) \tag{5.8}$$

These function are important; the forward pass (neuron activation function) is identical to the output of a standard perceptron layer with a sigmoid activation function. Consequently, this means that an RBM an be used in place of a neural network layer with the weight matrix of W. The effects of the forward pass using the input of v to get the output h will be mathematically identical. But, the RBM has an important distinction; the model is learned in a generative manner instead of the discriminative manner of neural networks. RBMs can utilize a backward pass for inference, i.e. generate the input features from the output by fixing the output values and sampling on the joint probability function. And since they are generative models, the stochastic gradient of the energy function does not depend on a label. Instead of learning by backpropagation on supervised data, RBMs work with unsupervised instances.

5.2.3 Deep Belief Networks

RBMs can be stacked in a deep manner to learn to deep hierarchical representation of the data. Hinton et al. proposed the Deep Belief Network architecture [11] which is trained in a greedy manner. The greedy pretraining is carried out in an unsupervised manner to model the joint distribution of the observed variables. Due to the minimization of the energy function in the RBM layers, the relationship between the input and output is formed in a way to allow the visible input to be generated from the output of the hidden variables. Thus, the output of the hidden layers are the learned features from the input data.

Since the RBMs can be used to model a perceptron layer as we mentioned, the deep belief network can behave as a multilayer neural network. After pretraining, we can convert the deep RBM architecture to use labelled instances, and train the DBN by backpropagation in a supervised fashion.

A representation of DBNs are given in Figure 5.3. DBNs model the joint distribution between observed vector x and the ℓ hidden layers h^k as:

$$P(x, h^1, \ldots, h^\ell) = \left(\prod_{k=0}^{\ell-2} P(h^k | h^{k+1}) \right) P(h^{\ell-1}, h^\ell) \qquad (5.9)$$

where $x = h^0$, $P(h^{k-1}|h^k)$ is a conditional distribution for the visible units

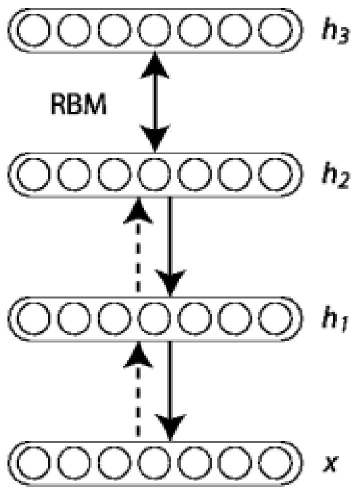


Figure 5.3: Stacking multiple layers of RBMs to create a Deep Belief Network

conditioned on the hidden units of the RBM at level k, and $P(h^{\ell-1}, h^{\ell})$ is the visible-hidden joint distribution in the top-level RBM.

The principle of greedy layer-wise unsupervised training for DBNs is as follows [205, 11].

1. Train the first layer as an RBM that models the raw input $x = h^{(0)}$ as its visible layer.

2. Perform a forward-pass on the visible units to obtain their hidden variables. The hidden units will be act as the representation of the input that will be used as data for the second layer.

3. Train the second layer as an RBM, taking the transformed data as training examples (for the visible layer of that RBM).

4. Repeat steps 2 and 3 for all of the layers, each time taking the output of the previous layer as input and propagating the data upward.

After step 4, we have a pre-trained network. We can then use backpropagation to fine-tune all the parameters of this deep architecture with respect to a supervised training criterion. The supervised learning can be a classifier, a regressor, or, an auto-encoder.

Stacked Auto-encoders

An auto-encoder is a model which learns to encode the data in such a way to allow reconstruction from that encoding. Auto-encoders can be though of as similar to dimensionality reduction using Principal Component Analysis (PCA). Whereas PCA is linearly defined, auto-encoders can create any non-linear representation of the data.

Using a DBN structure, a deep auto-encoder can be built by stacking multiple RBMs on top of each other. It is common to reduce the amount of hidden variables in each layer as the data is propagated upwards. After reaching the desired number of variables, the network can be fine-tuned by supervised learning to minimize the reconstruction error.

5.3 Method

5.3.1 Network structure

Convolutional layers

It is possible to use convolutional units during training. A convolution "filter" is a matrix of weights. A filter of size $A \times B$ will convolve the given input of size $X \times Y$, and the result of the operation will be the valid part of convolution of size $(X - A + 1) \times (Y - B + 1)$.

Convolution of a 2D signal can be represented as

$$o[m, n] = f[m, n] * g[m, n] = \sum_{u=-\infty}^{\infty} \sum_{v=-\infty}^{\infty} f[u, v]g[u - m, v - n] \tag{5.10}$$

Convolution operation allows sharing weights among all the input. Since the filter will be replicated across the entire input features can be detected regardless of their position in the input data. For example, if we assume the input data to be a $20 \times N$ matrix that represents a protein sequence of length N (as described in Chapter 2), we can define a sequence motif PSSM as a convolution filter of size $20 \times M, M \leq N$. Convolving the input with this filter will produce an output of size $1 \times (N - M + 1)$. This one dimensional vector will give us the "fitness" of that filter (motif) among all the positions in the sequence. The greatest score among this vector will give where this motif shows the strongest presence in the input amino acid sequence.

Therefore, instead of parameterizing each possible motif and each position combination separately, we will have one set of shared weights for each motif. The convolution operation correspond perfectly with our aim to find a motif regardless of its position in the input sequence, and greatly reduces the number of free parameters to learn.

Gradient descent can still be used to learn such shared parameters for convolutional filters, with only a small change to the original algorithm. The gradient of a shared weight is simply the sum of the gradients of the parameters being shared.

Max-pooling

Max-pooling is a form of non-linear down-sampling. Max-pooling partitions the input data into a set of non-overlapping regions, for each such sub-region outputs the maximum value. Max-pooling an input of size $X \times Y$ with a pool size of $a \times b$ will result in $\lceil (X/a) \rceil \times \lceil (Y/b) \rceil$ features.

Max-pooling is useful because it provides a form of position invariance and provides additional robustness to position. Since we will choose the maximum value among the axb, slight shifts in the input will not change the input to the higher layers. Consequently, by reducing the dimensionality of intermediate representations, it reduces the computational complexity for upper layers.

Network architecture

The networks used during our experiments can be summarized as follows. We will have a number of convolutional layers, each followed by a max-pooling layer as necessary, which takes the output of the previous layer as its input. These convolutional layers can be trained unsupervised in a greedy, layer-wise manner using Convolutional RBMs as described by Lee et al. [206].

During supervised learning, the output of the final convolution/maxpooling unit will be then fed through a fully-connected neural network layer with a number of hidden layers followed by a final logistic regression layer for classification, resulting in a prediction. The error (negative log-likelihood of the logistic regression layer) will be backpropagated through the fully-connected and convolutional layers, and this process will be repeated for every mini-batch of training examples and for a number of epochs until training stops.

The input will be of size $F \times N$, where F is the number of features for each residue(or coarse-grained unit), and N is the length of the protein. Since this data is linear and one-dimensional, our convolution filters for the first layer also need to be of size F. Assuming a convolutional unit of size FxC_1 and a total of U_1 filters, the first layer will produce a 1D output of size $1 \times (N - C_1 + 1) \times U_1$. The output will be downsampled using max-pooling of size $1 \times D_1$ to result in the output of first layer as $1 \times \left\lceil \frac{(N-C_1+1)}{D_1} \right\rceil \times U_1$. The second convolution layer will work with those U_1 linear outputs by using a filter length of $U_1 \times C_2$, and this process will be repeated as necessary.

5.3.2 Learning informative motifs

5.3.2.1 Prediction with variable length input

Convolution and max-pooling allows the extraction of informative, gapped and relatively position-independent motifs from the input sequences. However, RBMs and neural networks require a constant-length input. Considering proteins can theoretically be of any length, there is no way to represent these unbounded sequences in neural networks.

To overcome this limitation, we assume that the motif we are trying to extract is of length M, and M is relatively short and bounded. By giving an estimate for the maximum length of M (including possible insertions within the motif) as M', we can limit our search to M'-long segments at each step. By this approach, we may try to divide a length of protein N into overlapping segments of M', guaranteeing at least one of the segments completely cover the whole motif.

While this approach solves the representation of the input as a fixed size vector, it doesn't solve training and prediction. If we divide the protein P into s

overlapping segments of length M', each of one of the s segments will act as an instance by itself, and we will have s output predictions. The problem becomes on deciding which of those s predictions will be correct for P. If $M << |P|$, only a few segments of those segments will contain the motif in question; other segments may contain the motif partially, or not contain at all.

We propose 3 different approaches.

Maximum average probability

The classification of the input is carried out by the logistic regression layer at the top. A logistic regressor will give a vector Y of size C where C is the number of possible labels/outputs. Since logistic regression acts as a softmax of its input $((\sum_{i=1}^{C} Y_c) = 1)$, each one of those C predictions Y_i can be interpreted as the probability of the input belonging to the class i. Normally, the output prediction of a logistic regression layer is $y_{pred} = \text{argmax}_i Y_i$.

Utilizing this probabilistic nature, we can choose the class with the maximum average probability for the s different Y vectors:

$$y_{pred} = \text{argmax}_i(\sum_{\sigma=1}^{s} Y_i^{\sigma}) \tag{5.11}$$

It is possible that averaging over the whole of the sequence may cause the result of an informative motif to be lost due to noise. We can increase the weight of the few instances with very high predictions by taking the maximum squared average probability instead:

$$y_{pred} = \text{argmax}_i(\sum_{\sigma=1}^{s}(Y_i^{\sigma})^2) \tag{5.12}$$

Minimum entropy

Instead of averaging over the whole protein that consists of s segments, we may choose the segment s_i with the most confident prediction. If we assume C=3, there are 3-classes that we can assign an input to, logistic regression will give an output vector $[Y_1, Y_2, Y_3]$ where $Y_1 + Y_2 + Y_3 = 1$. Both of the predictions $[0.34, 0.33, 0.33]$ and $[1, 0, 0]$ have the first class probability as the maximum value, and will be classify the instance as class 1. However, the second prediction is much more confident about the prediction probabilities. In such cases, we'd like to find the segment with the most confident prediction.

We define "confidence" as the information content of a prediction; if there is a lot of ambiguity in the probabilities, then the information content, Shannon entropy [207] of that prediction will be maximized, whereas a prediction with high confidence will result in a lower ambiguity, and lower entropy.

Using the definition of Shannon entropy, we select the segment s_k with the most confident prediction results as:

$$s_k = \text{argmin}_i(-\sum_{j=1}^{C} Y_j^i \; log_2 Y_j^i) \tag{5.13}$$

and the prediction becomes

$$y_{pred} = \text{argmax}_i Y_i^{s_k} \tag{5.14}$$

Non-parametric class distribution

In this approach, we take all of the Y vectors from the inputs belonging to a specific class from the training data, and build a non-parametric model by kernel density estimation. Each class will have its own probability distribution that outputs a probability when given an input vector of size C. During prediction, we compare all the Y vectors of the test instance by the density function of every class, and assign the protein to the class with the maximum probability for all of its segments.

5.3.2.2 Learning with variable length input

The above section deals with predicting the class of a protein consisting of many input segments, but it assumes our model has correct weights. But to predict, we must first train the network on the input. Training on all of the segments will be problematic for the reasons given above, only a few of the segments will actually contain the motif we are looking for. If we backpropagate to minimize errors in uninformative segments, the network will not be able to find the correct weights for a motif.

Rational thought dictates that we must train the network using only the motif samples as an input. Our problem is, we don't know the location or the description of the motifs. Thus, the network should be able to find both the location and the structure of the motif together.

We propose the following solution. After pre-training the network using all of the segments, continue with supervised training for n epochs, still using all of the segments. After the initial n steps, we assume the network has reached a potentially good starting point. From then on, during training, predict all of the segments of a protein, and find the most informative segment by the minimum entropy method described above. Backpropagate using only that segment alone.

This approach is similar to the idea of Gibbs sampling. It presumes that the random weights in the initial state will start to converge to the actual definiton of the motif, which will in turn the segments containing the motif to be found with higher probability, which will cause the motif to be even more well-defined and so on, until the weights settle on a specific motif.

5.4 Experiments

5.4.1 Learning DNA Binding Domains from Sequence Information

The first test focuses on finding the protein sequence motifs that are found in different DNA-binding domains. From Uniprot [208, 209], we select approximately 8000 protein sequences that include any of the Zinc-Finger, Leucine-Zipper and Helix-Turn-Helix domains or related sub-domains. The sequences were filtered by minimum length of 50 and sequence identity of less than 90%. We built 3 different benchmark sets using further different maximum length and identity constraints, as well as sub-domain filtering. The sequences were converted into the residue-centered similarity-score representation explained in Chapter 2.

Since there is no comparable study that deals with alignment-free discriminative motif mining or giving the predictive performance of their motifs, we opted on using raw features with an SVM (support vector machine) classifier. The results of this approach are given in column 1 of Table 5.1.

Using the same data, a Convolutional RBM with two layers (a Convolutional Deep Belief Network, CDBN) were built on the sequences in an unsupervised manner; the information about which sequence belongs to which class was never utilized during the training process. First layer has 30 filters of size 20 $times 5$, and the second layer has 40 filters of size 1 $times 8$ (corresponding to an input of 20 $times 12$ without using any max-pooling).

The filters of the convolution layers after this unsupervised training are visualized in Figure 5.4. For the first layer, each "block" of 20 by 5 image represents the scores of 20 amino acids for the 5 positions in the motif, with brighter colors representing positive scores, dark colors representing negative scores and gray are uninformative/background. The amino acids are ordered from charged to polar to hydrophobic (from top to bottom) in the visualization.(In order: R, H, K, D, E, S, T, N, Q, C, G, P, A, V, I, L, M, F, Y, W)

As we can see, the first layer generally utilizes only one of the positions among the 5. While some of the filters recognize a specific residue (second row, second column recognizes only Proline), some of the filters act in a more general manner (first element in second row recognizes any hydrophobic, plus Cysteine with a low score). However, by combining these 1-position elements, the second convolution layer can find motifs that are much more variable as seen in Figure 5.4.

While these motifs may not be discriminative (since we trained them in an unsupervised manner), they should contain enough information to capture the general trends and feature clusters that may be present in the data. After converting our raw input into a deeper representation by feeding it through the network, we end up with a much smaller number of features that should be informative. To test this hypothesis, we classify the sequences by SVM using the preprocessed representation instead of the raw input. Results are shown in column 2 of Table 5.1. The results confirm our hypothesis, even though the classifier is identical, the noise and random variance across all the positions are converted into a more meaningful representation by the CDBN, resulting in a significantly higher accuracy among all three data sets.

We can test whether the representations contain enough information about their corresponding amino acid sequences by making a forward and backwards pass; by converting the sequences into a deep representation, and then reconstructing the AA sequence from this deep representation. A visualization of the reconstruction by the CDBN is given in Figure 5.5. While it is hard to compare and quantify such a complex structure, we see that the main informative parts of the sequence are preserved. If we probabilistically assign an amino acid label into a 20x1 vector for a specific position by comparing it to the BLOSUM vector, we are left with a sequence that has ¿ 0.7 identity (mean of 0.81) and ¿ 0.9 similarity (mean of 0.94) to the raw sequence. In fact, such modifications to the raw input is equivalent to filtering out uninformative and uncharacteristic parts of the sequence to better fit the overall structure of the motifs.

Finally, we add the label information to the training instances and further train the CDBN by connecting the output of the second convolutional RBM layer to 2 fully connected hidden layers and a final logistic regression layer, in essence creating a convolutional neural network (CNN). After this supervised training step, we end up with the results given in column 3 of Table 5.1. Out of the 3 methods, CNN is the best performer in all of the input sets, improving upon the base CDBN results.

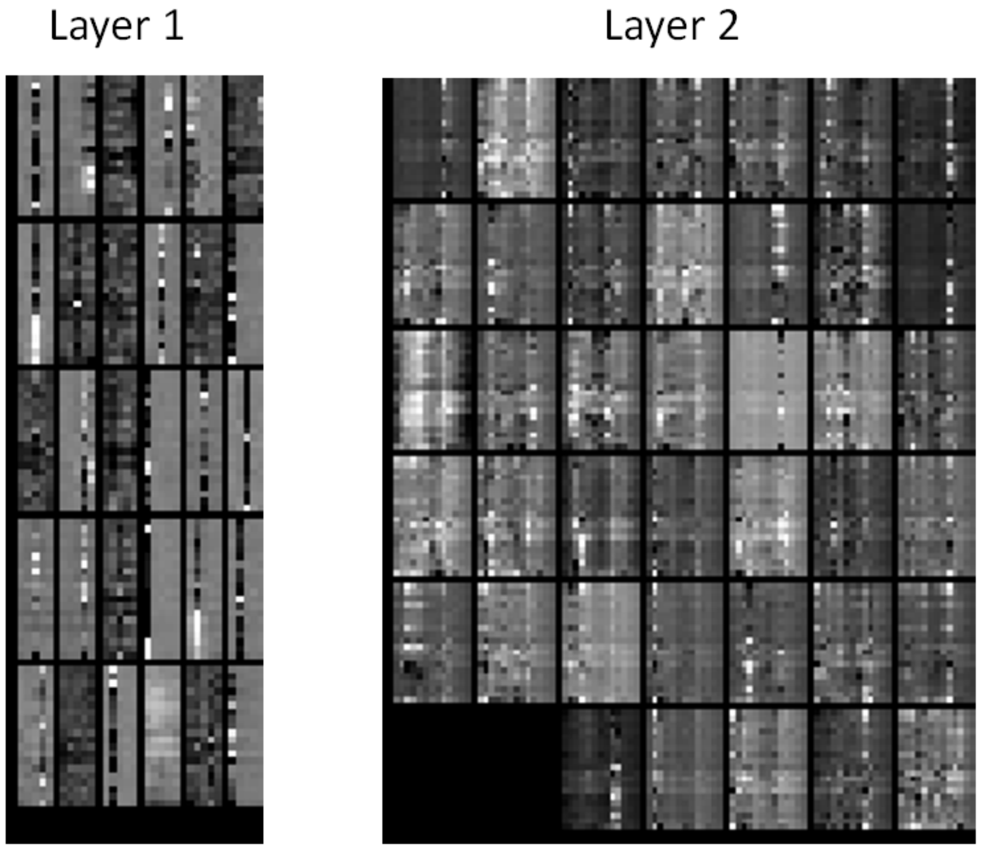


Figure 5.4:
Visualization of the sequence filters (first and second convolutional layers) for the DNA-binding motif data set.

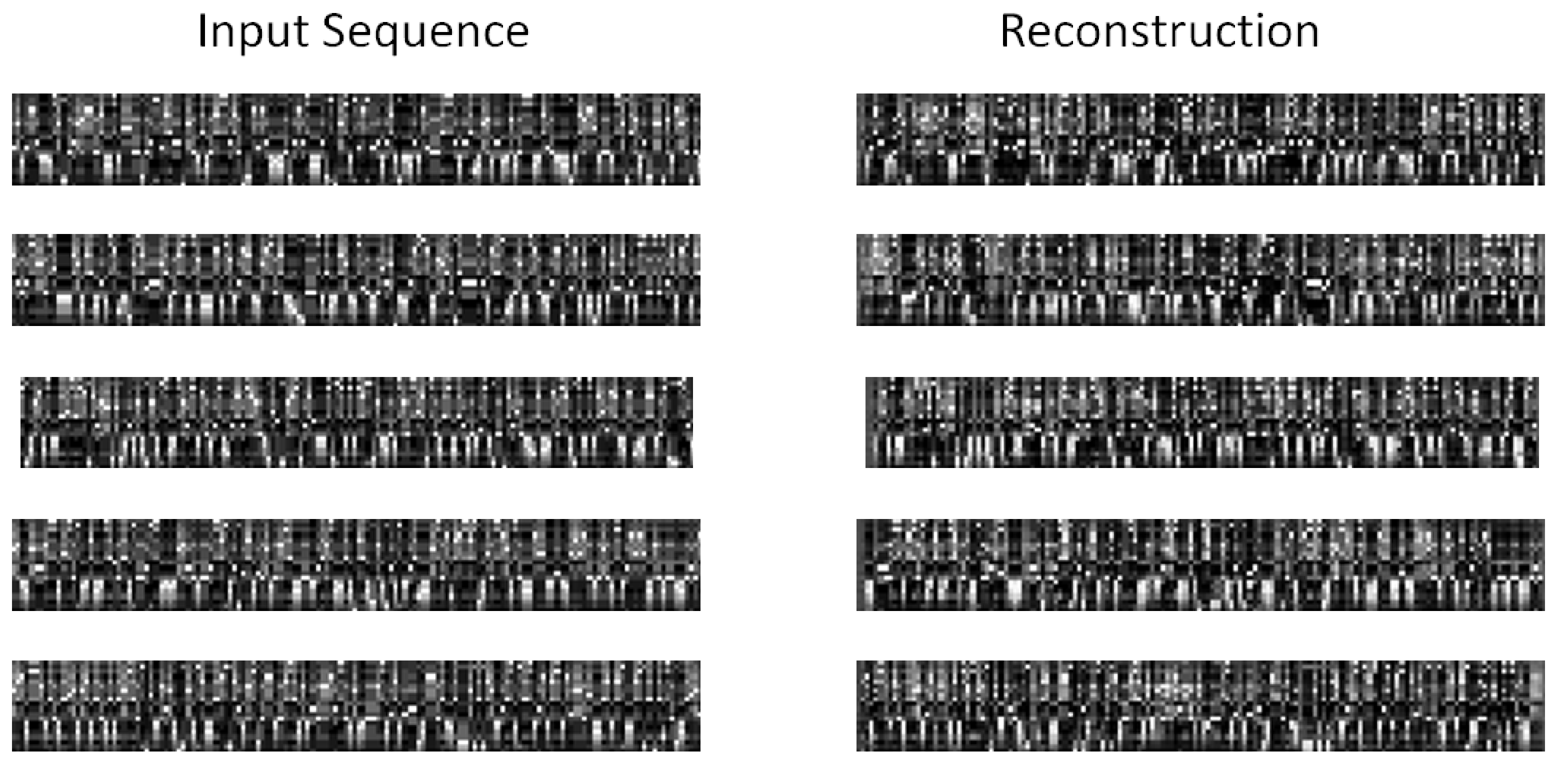


Figure 5.5:
Visualization of 4 random amino acid sequences and their reconstructions from the encoded values using the convolutional deep belief network.

Table 5.1: Results for the DNA-binding sequence motif data set

Data set	Raw Data SVM	CDBN + SVM	CNN
Benchmark Set 1	0.8	0.91	**0.93**
Benchmark Set 2	0.77	0.87	**0.89**
Benchmark Set 3	0.84	0.89	**0.91**

5.4.2 Learning CATH Folds from Structure Information

The second experiment focuses on the extraction of possible motifs and folds from the structural topology of the proteins.

Data set

All of the protein structures from the representative folds database of CATH [33] are downloaded. CATH database follows a hierarchical structure of Class - Architecture - Topology - Homologous superfamily. From the 3 main classes (”mainly alpha (helix)”, ”mainly beta”, ”alpha-beta”), we choose 7 architectures (out of 39) that contain at least 100 separate topologies. This results in 1015 proteins, each with a separate topology, from a total of 7 classes.

Deep Auto-encoder

An auto-encoder of the architecture ($2000 \rightarrow 500 \rightarrow 100 \rightarrow 2$) is used to represent 20 neighbor contact information of 100 residues by only 2 features. Whether such an unsupervised dimensionality reduction technique can represent the whole protein with only 2 numerical values is an important problem; such an approach can be used as global features to supplement the features we described in Chapter

3.

The resulting 2 features are plotted and colored by the architecture, shown in Figure 5.6. The results are clear; even in the absence of class information, the auto-encoder will try to learn the most efficient representation to be able to reconstruct a protein structure from just 2 inputs. Due to this strong bottleneck factor, most of the information that explains the general structure of the proteins are embedded in the network itself, and the encoder will represent only the most informative, distinguishing factors by those 2 features. This allows similar proteins to be grouped together while separating different classes.

Using 10 features instead of 2, we are able to classify a sequence by its class (3-class) by 86.3% accuracy, and by its architecture (7-class) by 67.2% accuracy.

Convolutional Neural Network

While an auto-encoder results in a very small number of informative global features, they are not as useful for finding discriminative motifs. Using a similar architecture described in the previous section, we build a CNN using different representations that are defined in Chapter 2.

The network is composed of a two to three convolutional layers, each followed by a max-pooling layer downsampling its output by 2, which is fed through a fully connected network and a logistic regression layer. Each protein is divided into segments of length 50 to 70 by sliding a window over the sequence. The parameter optimizations were carried out differently for each representation.

We train this network using the 1015 proteins in 5-fold cross validation. The results for the different representations are given in Table 5.2. Residue specific representations are amino acid sequence (AA), Protein Block sequence (PB) and

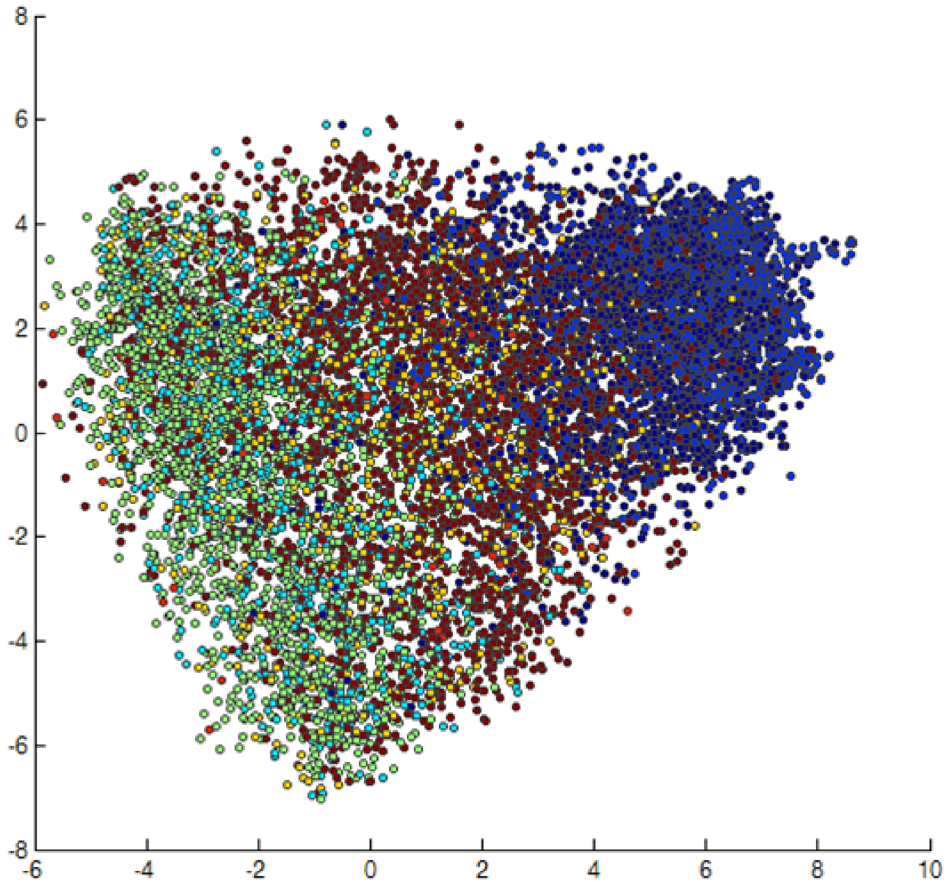


Figure 5.6:

Results of unsupervised dimensionality reduction using stacked auto-encoders. Blue hues: Mainly-alpha, Red hues: Alpha-Beta, Green hues:Mainly Beta

Bond-Orientational Order parameters. Pairwise representations with N consecutive neighbors are Probabilistic contact map (PCM), Relative Orientation angle and amino acid similarity between the contacts. We tried a large number of combinations for those features, and a representative set is shown.

The best results seem to come from PCM, with increasing number of neighbors decreasing the classification accuracy. This is somewhat problematic for our case. In theory, unsupervised learning methods should not be affected by the curse-of-dimensionality that causes the accuracy to decrease as the number of features grow beyond a point. However, a large number of features require learning more free parameters, which negatively affects the training process and requires longer computational time to reach the same level of maturity.

While the structural features like PCM (91%) and PB (81%) perform quite well, sequence information is also quite discriminative (77%). We see that bond orientational orders lack the discriminative power, though this may be causes by the large number of features denoting each residue (40) as discussed above.

Two samples from the learned filters are given in Figures 5.7 and 5.8. The filter represents the motif created by the probabilistic contact map representation for 10 neighbors. The elements in the top part of the filters represent further neighbors to the forward, the middle line represents the immediate neighbors of that residue and the bottom part represents the neighbors that are to the back of the center position.

If we keep the learning rate too high, we end up with features as shown in Figure 5.7. Since the alpha-helices dominate the input data (most common element, easier to distinguish, high predictive power), the convolution filters will learn to represent the helices; each of the filters in the second or the third layer describes different configurations of alpha-helices.

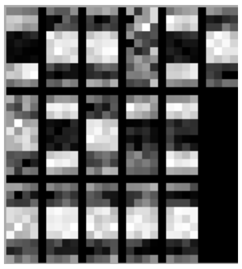
We can learn a much better representation by diminishing the learning rate of the network, resulting in Figure 5.8. While not immediately clear, the different diagonals represent separate curved structures and combination of sheet structures with helices.

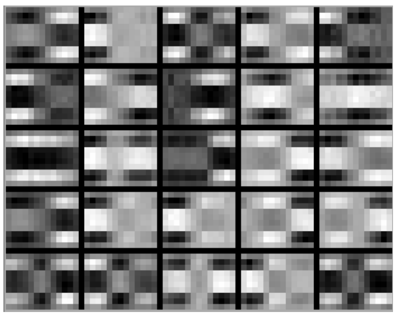
Table 5.2: Results of CNN in the CATH dataset

Data	**Accuracy**
AA Sequence	77.10%
PB Sequence	81.10%
Bond-Orientational Order	67.40%
PCM, N=5	**91.30%**
PCM, N=10	89.70%
PCM, N=20	86.40%
Relative Angle, N=5	73.20%
Relative Angle, N=10	67.40%
PCM + AA similarity, N=5	88.40%
PCM + AA similarity, N=10	83.50%
PCM + Relative Angle, N=5	85.30%
PCM + Relative Angle, N=10	80.10%

5.5 Discussion

We present a novel algorithm based on deep belief networks and convolutional neural networks to extract informative, discriminative motifs from sequence and structural information. The ability to work with unlabeled data is very important for studies where labeled data is scarce. By training the network on unlabeled instances, it will learn to represent the building blocks and background distribution


(a) Convolution Layer 1.


(b) Convolution Layer 2.

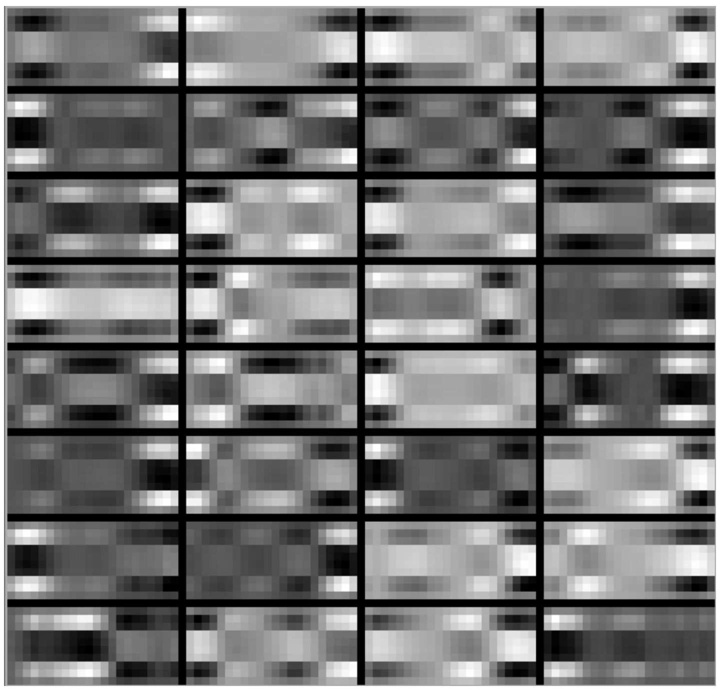

(c) Convolution Layer 3.

Figure 5.7: Visualization of the contact map filters showing the helix structures

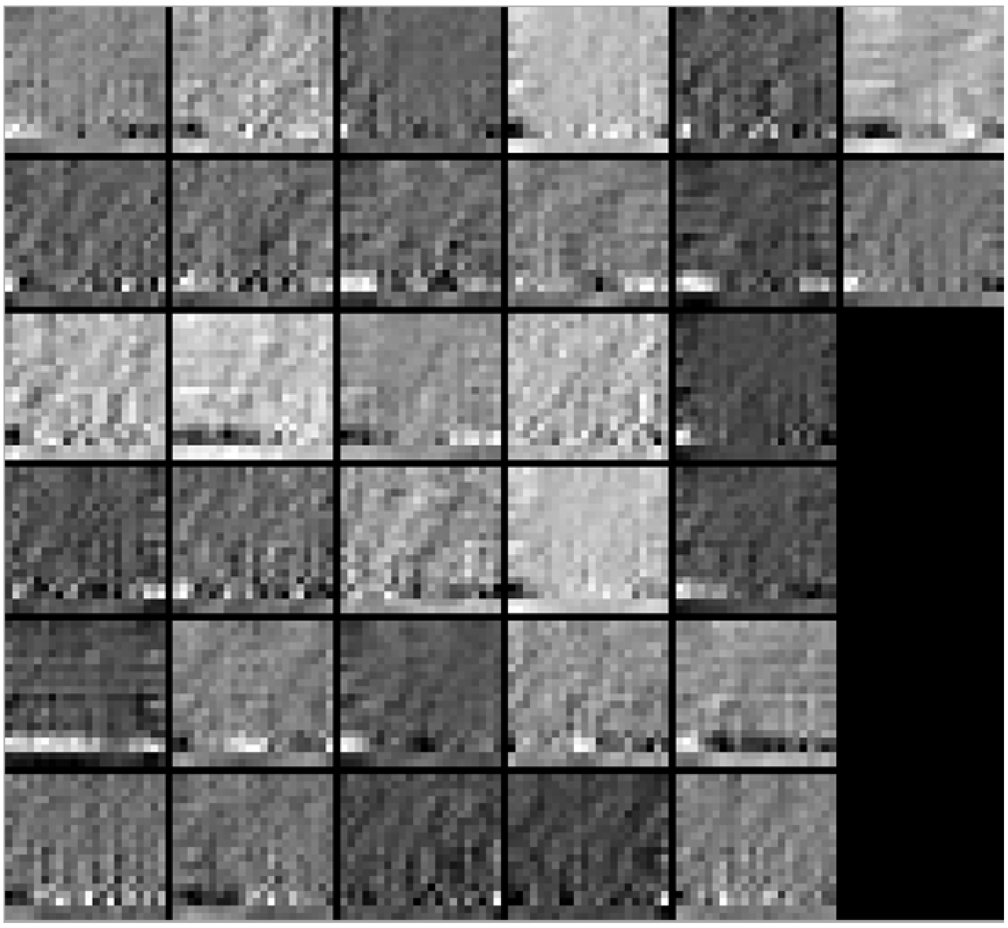


Figure 5.8: Visualization of the contact map filters showing different folds

of the possible configurations of features within themselves. After this pre-learning, even a small number of labeled instances can be used for discrimination by the use of the deviations from the background probability embedded into the network structure.

We further propose a solution to be able to use variable length inputs (such as biological data), into a deep learner. By this approach, it is possible to mine informative, discriminative motifs without knowing the sequence alignment of the motif description. Addition of the max-pooling step allows the motif to include up to 75% of its length in insertion/deletions for 2 convolution layers. Also, since higher layers combine the feature representations from below, the network is able to create super-motifs that can contain different domain/folds that can be separated by segments of variable length.

Our algorithm was not developed for a specific problem or representation. We present its application to both sequence and structural motif mining to show its feasibility and performance. However, it is not without its problems. We found that the network is limited by the number of features due to its computational complexity. However, it may be necessary to mix and match features from different domains to be able to capture an actually discriminative motif. This area requires further studies to make such alignment-free semi-supervised learning feasible.

Another point is, the problems presented here are "toy" problems; while there are no other such unsupervised method to compare our results against, we know that the CATH families and DNA-binding domains are well-characterized. An expert can manually surpass these results using a variety of tools and knowledge buried in the literature (multiple alignment, motif databases, sequence search and similarity measures). However, our approach is not aimed to perform better than a knowledgeable person that manually chooses the best possible way to approach the problem. This method is more suitable for cases when there is not enough domain information about the problem to decide on an approach, or when the user lacks the expertise, or when there is too much data to analyze, cluster and classify manually. If such un- and semi-supervised methods gain acceptance, the vast amount of raw data that are deposited into the biological databases might be searched and analyzed with little to no user input, presenting only the meaningful information to the researchers to analyze and comment on.

In conclusion, while the method we presented here requires further work, it can perform quite well even in its current form. It is possible that this approach may shift a big portion of the manual labor to the artificial learners, and change the workflow of many researchers in a positive way.

6 CONCLUSIONS

Proteins are complex biological molecules that perform a tremendous variety of functions in cells under diverse physiological conditions. Proteins can be classified based on their function, structural motifs that they possess, cellular location or adaptation to an external variable such as temperature, pH or salinity. All the diversity that is present in protein structure and function is encoded in protein sequence which determines a protein's structure, function or response to an external variable. Bridging the gap between protein sequence and structure or function is an active area of research in bioinformatics that can enable us to determine a target attribute of a novel protein (e.g. structural class, important binding sites, activity, half life and etc.) with high accuracy, and even allow the design of novel proteins fit for a specific purpose.

To that end, we introduced a variety of strategies to systematically represent a protein by a number of conventional and novel descriptors, and mine predictive and discriminative motifs using those representations along with novel machine learning algorithms.

Chapter 2 introduces and explains a number of techniques to describe a short local segment of a protein quantitatively using its sequence, local structural fold, its contacts, its centrality and importance in the global structure of the network by its graph properties, and by the relative orientation of its neighbors using bond-orientational order. The feasibility of the novel features are investigated by applying them to common problems such as multiple alignment and secondary structure and fold recognition.

Chapter 3 builds upon the local features to describe a number of features that contain and summarize informative regional features over the protein to represent the protein globally. We define a large number of features that contain information from the nucleic acid sequence, amino acid sequence, secondary and tertiary structure, physicochemical data, catalytic and active site information and the 3D surface patches and hot spots. To test the plausibility of combining our features with novel machine learning algorithms, we apply those features to the problem of predicting mRNA and protein expression levels. This enables us to predict the protein translation rate and abundance within the host, and the solubility or aggregation of the final gene product. Creating and using the most comprehensive data set published so far, we built a model of the transcription and translation process for different organisms. The model is created by a number of descriptors that explain a significant portion of the variance within the protein levels. We proposed the first available system capable of predicting protein abundance, with the ability to predict both homologous and heterologous expression for different hosts. Such a prediction tool can simplify the technical aspects of protein expression such as host-selection and codon optimization.

Chapter 4 presents a novel algorithm of partial periodic motif mining (PPM) that can mine discriminative gapped short motifs that are highly variable within their composition. The strength of PPM is in its ability to capture length independent short motifs that are in close vicinity. Because the motif mining and prediction steps are uncoupled, the method can be used for different purposes. We applied the algorithm to the MHC class I and II peptide binding prediction problem, and shown that the rules mined from the data can be used in conjunction with support vector machines or neural networks for non-linear prediction of any label (or quantitative value) that is correlated with the sequence motifs. When applied to the MHC problem, our algorithm outperformed the state-of-the-art methods in different data sets.

Finally, Chapter 5 describes a novel algorithm based on deep belief networks and convolutional neural networks to extract informative, discriminative motifs from sequence and structural information. We further propose a solution to be able to use variable length inputs like proteins in neural networks. By this approach, it is possible to mine informative, discriminative motifs without knowing the sequence alignment of the motif description. Our method is also possible to work with unlabeled instances when the labeled data is scarce.

One of the main problems of machine learning in bioinformatics is the lack of an obvious representation that explicitly defines a protein's structure. This problem is much more common than it is in other domains. For image recognition and similar tasks, representation of the visual information is obvious and standardized to a degree between the tasks. This allows the algorithm development to focus more on the machine learning side. However, the representation of sequence and especially 3D structural information for variable length input is not clearly defined. While we proposed some solutions to these problems, it is necessary to find more native forms of representation.

As future work, we suggest the descriptors and the representations to be more tightly integrated to allow more robust motif matching. The scoring methods for graph properties, bond-orientational order parameters and other quantitative measures can be modified to use adaptive weights based on the task at hand. As mentioned in Chapter 5, the deep learning step requires optimization and further work to mine motifs without as much user input. We hope that such unsupervised and semi-supervised learning methods will automate and reduce the manual work required during data analysis and prediction, shifting the focus of research to decision making on a higher-level.

References

[1] A. Lehninger, D. L. Nelson, and M. M. Cox, *Lehninger Principles of Biochemistry*, 2008.

[2] D. Jiang, C. Tang, and A. Zhang, "Cluster analysis for gene expression data: a survey," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 16, no. 11, pp. 1370–1386, 2004.

[3] Y. Qi, Z. Bar-joseph, and J. Klein-seetharaman, "Evaluation of different biological data and computational classification methods for use in protein interaction prediction," *Proteins*, vol. 63, pp. 490–500, 2006.

[4] D. Jiang, C. Tang, and A. Zhang, "Cluster analysis for gene expression data: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, pp. 1370–1386, 2004.

[5] I. V. Tetko, I. V. Rodchenkov, M. C. Walter, T. Rattei, and H.-W. Mewes.

[6] I. Friedberg, "Automated protein function prediction - the genomic challenge." *Briefings in Bioinformatics*, vol. 7, no. 3, pp. 225–242, 2006.

[7] T. Damoulas and M. Girolami, "Probabilistic multi-class multi-kernel learning: on protein fold recognition and remote homology detection," *Bioinformatics*, vol. 24, no. 10, pp. 1264–1270, 2008.

[8] L. K. Matukumalli, J. J. Grefenstette, D. L. Hyten, I.-Y. Y. Choi, P. B. Cregan, and C. P. Van Tassell, "Application of machine learning in SNP discovery." *BMC bioinformatics*, vol. 7, no. 1, pp. 4+, 2006.

[9] N. Bhardwaj, R. E. Langlois, G. Zhao, and H. Lu, "Kernel-based machine learning protocol for predicting DNA-binding proteins," *Nucleic Acids Research*, vol. 33, no. 20, pp. 6486–6493, 2005.

[10] K. J. Hoff, M. Tech, T. Lingner, R. Daniel, B. Morgenstern, and P. Meinicke, "Gene prediction in metagenomic fragments: a large scale machine learning approach." *BMC bioinformatics*, vol. 9, pp. 217+, 2008.

[11] G. E. Hinton and S. Osindero, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, p. 2006, 2006.

[12] M. Poursina, K. D. Bhalerao, S. C. Flores, K. S. Anderson, and A. Laederach, "Strategies for articulated multibody-based adaptive coarse grain simulation of rna," *Methods in enzymology*, vol. 487, p. 73, 2011.

[13] V. Tozzini, "Coarse-grained models for proteins," *Current opinion in structural biology*, vol. 15, no. 2, pp. 144–150, 2005. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0959440X05000515

[14] S. F. Altschul, T. L. Madden, A. A. Schffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman, "Gapped blast and psiblast: a new generation of protein database search programs," *NUCLEIC ACIDS RESEARCH*, vol. 25, no. 17, pp. 3389–3402, 1997.

[15] D. Frishman and P. Argos, "Knowledge-based protein secondary structure assignment," *Proteins*, vol. 23, no. 4, pp. 566–79, 1995.

[16] J. Martin, G. Letellier, A. Marin, J. F. Taly, A. G. de Brevern, and J. F. Gibrat, "Protein secondary structure assignment revisited: a detailed analysis of different assignment methods," *BMC Struct Biol*, vol. 5, p. 17, 2005.

[17] A. P. Joseph, G. Agarwal, S. Mahajan, J. C. Gelly, L. S. Swapna, B. Offmann, F. Cadet, A. Bornot, M. Tyagi, H. Valadie, B. Schneider, C. Etchebest, N. Srinivasan, and A. G. De Brevern, "A short survey on protein blocks," *Biophys Rev*, vol. 2, no. 3, pp. 137–147, 2010.

[18] A. G. de Brevern, C. Etchebest, and S. Hazout, “Bayesian probabilistic approach for predicting backbone structures in terms of protein blocks,” *Proteins*, vol. 41, no. 3, pp. 271–87, 2000.

[19] I. Bahar, A. R. Atilgan, and B. Erman, “Direct evaluation of thermal fluctuations in proteins using a single-parameter harmonic potential.” *Fold Des*, vol. 2, no. 3, pp. 173–181, 1997.

[20] T. Haliloglu, I. Bahar, and B. Erman, “Gaussian dynamics of folded proteins,” *Physical Review Letters*, vol. 79, no. 16, pp. 3090–3093, 1997. [Online]. Available: http://link.aps.org/doi/10.1103/PhysRevLett.79.3090

[21] M. Tyagi, V. S. Gowri, N. Srinivasan, A. G. de Brevern, and B. Offmann, “A substitution matrix for structural alphabet based on structural alignment of homologous proteins and its applications,” *PROTEINS: Structure, Function, and Bioinformatics*, vol. 65, no. 1, pp. 32–39, 2006. [Online]. Available: http://onlinelibrary.wiley.com/doi/10.1002/prot.21087/full

[22] P. C. Mahalanobis, “On the generalized distance in statistics,” in *Proceedings of the national institute of sciences of India*, vol. 2, no. 1. New Delhi, 1936, pp. 49–55.

[23] P. D. Thomas and K. A. Dill, “An iterative method for extracting energy-like quantities from protein structures.” *Proc Natl Acad Sci U S A*, vol. 93, no. 21, pp. 11 628–11 633, Oct 1996.

[24] ——, “Statistical potentials extracted from protein structures: how accurate are they?” *J Mol Biol*, vol. 257, no. 2, pp. 457–469, Mar 1996. [Online]. Available: http://dx.doi.org/10.1006/jmbi.1996.0175

[25] H. M. Grindley, P. J. Artymiuk, D. W. Rice, and P. Willett, "Identification of tertiary structure resemblance in proteins using a maximal common subgraph isomorphism algorithm," *J Mol Biol*, vol. 229, no. 3, pp. 707–21, 1993.

[26] A. R. Atilgan, S. R. Durell, R. L. Jernigan, M. C. Demirel, O. Keskin, and I. Bahar, "Anisotropy of fluctuation dynamics of proteins with an elastic network model," *Biophys J*, vol. 80, no. 1, pp. 505–15, 2001.

[27] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959. [Online]. Available: http://www.springerlink.com/index/uu8608u0u27k7256.pdf

[28] A. R. Atilgan, D. Turgut, and C. Atilgan, "Screened nonbonded interactions in native proteins manipulate optimal paths for robust residue communication," *Biophysical journal*, vol. 92, no. 9, p. 3052, 2007.

[29] L. C. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, pp. 35–41, 1977.

[30] M. Newman, *Networks: an introduction.* OUP Oxford, 2009.

[31] T. Opsahl, F. Agneessens, and J. Skvoretz, "Node centrality in weighted networks: Generalizing degree and shortest paths," *Social Networks*, vol. 32, no. 3, pp. 245–251, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378873310000183

[32] G. Sabidussi, "The centrality index of a graph," *Psychometrika*, vol. 31, no. 4, pp. 581–603, 1966. [Online]. Available: http://www.springerlink.com/index/u57264845r413784.pdf

[33] C. A. Orengo, A. Michie, S. Jones, D. T. Jones, M. Swindells, and J. M. Thornton, "Cath–a hierarchic classification of protein domain structures," *Structure*, vol. 5, no. 8, pp. 1093–1109, 1997.

[34] N. Saitou and M. Nei, "The neighbor-joining method: a new method for reconstructing phylogenetic trees." *Molecular biology and evolution*, vol. 4, no. 4, pp. 406–425, 1987. [Online]. Available: http://mbe.oxfordjournals.org/content/4/4/406.short

[35] J. D. Thompson, F. Plewniak, and O. Poch, "Balibase: a benchmark alignment database for the evaluation of multiple alignment programs." *Bioinformatics*, vol. 15, no. 1, pp. 87–88, 1999. [Online]. Available: http://bioinformatics.oxfordjournals.org/content/15/1/87.short

[36] J. D. Thompson, P. Koehl, R. Ripp, and O. Poch, "Balibase 3.0: latest developments of the multiple sequence alignment benchmark," *Proteins: Structure, Function, and Bioinformatics*, vol. 61, no. 1, pp. 127–136, 2005. [Online]. Available: http://onlinelibrary.wiley.com/doi/10.1002/prot.20527/full

[37] J. Błażewicz, P. Formanowicz, and P. Wojciechowski, "Some remarks on evaluating the quality of the multiple sequence alignment based on the balibase benchmark," 2009. [Online]. Available: http://www.degruyter.com/view/j/amcs.2009.19.issue-4/v10006-009-0054-y/v10006-009-0054-y.xml

[38] A. R. Subramanian, J. Weyer-Menkhoff, M. Kaufmann, and B. Morgenstern, "Dialign-t: an improved algorithm for segment-based multiple sequence alignment," *BMC bioinformatics*, vol. 6, no. 1, p. 66, 2005.

[39] A. R. Subramanian, M. Kaufmann, B. Morgenstern *et al.*, “Dialign-tx: greedy and progressive approaches for segment-based multiple sequence alignment,” *Algorithms Mol Biol*, vol. 3, no. 6, 2008.

[40] D. G. Higgins and P. M. Sharp, “Clustal: a package for performing multiple sequence alignment on a microcomputer,” *Gene*, vol. 73, no. 1, pp. 237–244, 1988. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0378111988903307

[41] J. D. Thompson, D. G. Higgins, and T. J. Gibson, “Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice,” *Nucleic acids research*, vol. 22, no. 22, pp. 4673–4680, 1994. [Online]. Available: http://nar.oxfordjournals.org/content/22/22/4673.short

[42] K. Katoh, K. Misawa, K.-i. Kuma, and T. Miyata, “Mafft: a novel method for rapid multiple sequence alignment based on fast fourier transform,” *Nucleic acids research*, vol. 30, no. 14, pp. 3059–3066, 2002. [Online]. Available: http://nar.oxfordjournals.org/content/30/14/3059.short

[43] R. C. Edgar, “Quality measures for protein alignment benchmarks,” *Nucleic acids research*, vol. 38, no. 7, pp. 2145–2153, 2010. [Online]. Available: http://nar.oxfordjournals.org/content/38/7/2145.short

[44] M. Vendruscolo, E. Kussell, and E. Domany, “Recovery of protein structure from contact maps,” *Folding and Design*, vol. 2, no. 5, pp. 295–306, 1997. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1359027897000412

[45] M. Vassura, L. Margara, P. Di Lena, F. Medri, P. Fariselli, and R. Casadio, "Ft-comar: fault tolerant three-dimensional structure reconstruction from protein contact maps," *Bioinformatics*, vol. 24, no. 10, pp. 1313–1315, 2008. [Online]. Available: http://bioinformatics.oxfordjournals.org/content/24/10/1313.short

[46] P. J. Steinhardt, D. R. Nelson, and M. Ronchetti, "Bond-orientational order in liquids and glasses," *Physical Review B*, vol. 28, no. 2, pp. 784–805, 1983.

[47] B. Offmann, M. Tyagi, and A. G. de Brevern, "Local protein structures," *Current Bioinformatics*, vol. 2, no. 3, pp. 165–202, 2007.

[48] C. Atilgan, O. B. Okan, and A. R. Atilgan, "How orientational order governs collectivity of folded proteins," *Proteins*, vol. 78, no. 16, pp. 3363–75, 2010.

[49] W. J. Sternberg and T. L. Smith, *The theory of potential and spherical harmonics*, [2nd ed., ser. Mathematical expositions. Toronto,: Univ. of Toronto Press, 1946.

[50] L. D. Landau and E. M. Lifshitz, *Quantum mechanics: non-relativistic theory*, 2nd ed., ser. Their Course of theoretical physics,. Oxford, New York,: Pergamon Press; sole distributors in the U.S.A., Addison-Wesley Pub. Co., Reading, Mass., 1965.

[51] T. M. Truskett, S. Torquato, and P. G. Debenedetti, "Towards a quantification of disorder in materials: distinguishing equilibrium and glassy sphere packings," *Phys Rev E Stat Phys Plasmas Fluids Relat Interdiscip Topics*, vol. 62, no. 1 Pt B, pp. 993–1001, 2000.

[52] S. Torquato, *Random heterogeneous materials : microstructure and macroscopic properties*, ser. Interdisciplinary applied mathematics. New York:

Springer, 2002.

[53] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne, "The protein data bank," *Nucleic Acids Res*, vol. 28, no. 1, pp. 235–42, 2000.

[54] A. G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia, "Scop: a structural classification of proteins database for the investigation of sequences and structures," *J Mol Biol*, vol. 247, no. 4, pp. 536–40, 1995.

[55] Y. Zhang and J. Skolnick, "Tm-align: a protein structure alignment algorithm based on the tm-score," *Nucleic Acids Res*, vol. 33, no. 7, pp. 2302–9, 2005.

[56] M. N. Fodje and S. Al-Karadaghi, "Occurrence, conformational features and amino acid propensities for the pi-helix," *Protein Eng*, vol. 15, no. 5, pp. 353–8, 2002.

[57] C.-C. Chang and C.-J. Lin, "Libsvm: a library for support vector machines," 2001.

[58] J. Demsar, B. Zupan, G. Leban, and T. Curk, "Orange: From experimental machine learning to interactive data mining," *Knowledge Discovery in Databases: Pkdd 2004, Proceedings*, vol. 3202, pp. 537–539, 2004.

[59] J. A. Hanley and B. J. McNeil, "A method of comparing the areas under receiver operating characteristic curves derived from the same cases," *Radiology*, vol. 148, no. 3, pp. 839–43, 1983.

[60] Y. Koren and L. Carmel, "Visualization of labeled data using linear transformations," *Infovis 2002: Ieee Symposium on Information Visualization 2003, Proceedings*, pp. 121–128 248, 2003.

[61] G. Leban, B. Zupan, G. Vidmar, and I. Bratko, “Vizrank: Data visualization guided by machine learning,” *Data Mining and Knowledge Discovery*, vol. 13, no. 2, pp. 119–136, 2006.

[62] J. Demsar, G. Leban, and B. Zupan, “Freeviz–an intelligent multivariate visualization approach to explorative analysis of biomedical data,” *J Biomed Inform*, vol. 40, no. 6, pp. 661–71, 2007.

[63] P. M. Sharp and W.-H. Li, “An evolutionary perspective on synonymous codon usage in unicellular organisms,” *Journal of Molecular Evolution*, vol. 24, no. 1-2, pp. 28–38, 1986. [Online]. Available: http://link.springer.com/article/10.1007/BF02099948

[64] U. Pozzoli, G. Menozzi, M. Fumagalli, M. Cereda, G. Comi, R. Cagliani, N. Bresolin, and M. Sironi, “Both selective and neutral processes drive gc content evolution in the human genome,” *BMC evolutionary biology*, vol. 8, no. 1, p. 99, 2008.

[65] F. Wright, “The effective number of codons used in a gene,” *Gene*, vol. 87, no. 1, pp. 23–29, 1990. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0378111990904919

[66] M. dos Reis, R. Savva, and L. Wernisch, “Solving the riddle of codon usage preferences: a test for translational selection,” *Nucleic acids research*, vol. 32, no. 17, pp. 5036–5044, 2004. [Online]. Available: http://nar.oxfordjournals.org/content/32/17/5036.short

[67] P. M. Sharp and W.-H. Li, “The codon adaptation index-a measure of directional synonymous codon usage bias, and its potential applications,”

Nucleic acids research, vol. 15, no. 3, pp. 1281–1295, 1987. [Online]. Available: http://nar.oxfordjournals.org/content/15/3/1281.short

[68] T. Ikemura, "Correlation between the abundance of escherichia coli transfer rnas and the occurrence of the respective codons in its protein genes," *Journal of molecular biology*, vol. 146, no. 1, pp. 1–21, 1981. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0022283681903636

[69] J. L. Bennetzen and B. Hall, "Codon selection in yeast." *Journal of Biological Chemistry*, vol. 257, no. 6, pp. 3026–3031, 1982. [Online]. Available: http://www.jbc.org/content/257/6/3026.short

[70] R. Percudani, A. Pavesi, and S. Ottonello, "Transfer rna gene redundancy and translational selection in saccharomyces cerevisiae," *Journal of molecular biology*, vol. 268, no. 2, pp. 322–330, 1997. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0022283697909426

[71] F. Crick, "Codonanticodon pairing: the wobble hypothesis," *Journal of molecular biology*, vol. 19, no. 2, pp. 548–555, 1966. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0022283666800220

[72] M. Csűrös, "A linear-time algorithm for finding the longest segment which scores above a given threshold," *arXiv preprint cs/0512016*, 2005. [Online]. Available: http://arxiv.org/abs/cs/0512016

[73] S. Reuveni, I. Meilijson, M. Kupiec, E. Ruppin, and T. Tuller, "Genome-scale analysis of translation elongation with a ribosome flow model," *PLoS computational biology*, vol. 7, no. 9, p. e1002127, 2011.

[74] L. B. Shaw, R. Zia, and K. H. Lee, "Totally asymmetric exclusion process with extended objects: A model for protein synthesis," *Physical*

Review E, vol. 68, no. 2, p. 021910, 2003. [Online]. Available: http://pre.aps.org/abstract/PRE/v68/i2/e021910

[75] J. Lobry and C. Gautier, "Hydrophobicity, expressivity and aromaticity are the major trends of amino-acid usage in 999 escherichia coli chromosome-encoded genes," *Nucleic Acids Research*, vol. 22, no. 15, pp. 3174–3180, 1994. [Online]. Available: http://nar.oxfordjournals.org/content/22/15/3174.short

[76] I. Atsushi, "Thermostability and aliphatic index of globular proteins," *Journal of Biochemistry*, vol. 88, no. 6, pp. 1895–1898, 1980. [Online]. Available: http://jb.oxfordjournals.org/content/88/6/1895.short

[77] K. Guruprasad, B. B. Reddy, and M. W. Pandit, "Correlation between stability of a protein and its dipeptide composition: a novel approach for predicting in vivo stability of a protein from its primary sequence," *Protein Engineering*, vol. 4, no. 2, pp. 155–161, 1990. [Online]. Available: http://peds.oxfordjournals.org/content/4/2/155.short

[78] E. Gasteiger, A. Gattiker, C. Hoogland, I. Ivanyi, R. D. Appel, and A. Bairoch, "Expasy: the proteomics server for in-depth protein knowledge and analysis," *Nucleic acids research*, vol. 31, no. 13, pp. 3784–3788, 2003. [Online]. Available: http://nar.oxfordjournals.org/content/31/13/3784.short

[79] J. Prilusky, C. E. Felder, T. Zeev-Ben-Mordehai, E. H. Rydberg, O. Man, J. S. Beckmann, I. Silman, and J. L. Sussman, "Foldindex©: a simple tool to predict whether a given protein sequence is intrinsically unfolded,"

Bioinformatics, vol. 21, no. 16, pp. 3435–3438, 2005. [Online]. Available: http://bioinformatics.oxfordjournals.org/content/21/16/3435.short

[80] S. Rogers, R. Wells, and M. Rechsteiner, "Amino acid sequences common to rapidly degraded proteins: the pest hypothesis," *Science*, vol. 234, no. 4774, pp. 364–8, 1986.

[81] P. Rice, I. Longden, A. Bleasby *et al.*, "Emboss: the european molecular biology open software suite," *Trends in genetics*, vol. 16, no. 6, pp. 276–277, 2000.

[82] A. G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia, "Scop: a structural classification of proteins database for the investigation of sequences and structures," *Journal of molecular biology*, vol. 247, no. 4, pp. 536–540, 1995. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0022283605801342

[83] U. Samanta, R. P. Bahadur, and P. Chakrabarti, "Quantifying the accessible surface area of protein residues in their local environment," *Protein engineering*, vol. 15, no. 8, pp. 659–667, 2002. [Online]. Available: http://peds.oxfordjournals.org/content/15/8/659.short

[84] C. T. Porter, G. J. Bartlett, and J. M. Thornton, "The catalytic site atlas: a resource of catalytic sites and residues identified in enzymes using structural data," *Nucleic acids research*, vol. 32, no. suppl 1, pp. D129–D133, 2004.

[85] Z. Zhang, Y. Li, B. Lin, M. Schroeder, and B. Huang, "Identification of cavities on protein surface using multiple computational approaches for drug binding site prediction," *Bioinformat-*

ics, vol. 27, no. 15, pp. 2083–2088, 2011. [Online]. Available: http://bioinformatics.oxfordjournals.org/content/27/15/2083.short

[86] P. Debye, "Interferenz von röntgenstrahlen und wärmebewegung," *Annalen der Physik*, vol. 348, no. 1, pp. 49–92, 1913. [Online]. Available: http://onlinelibrary.wiley.com/doi/10.1002/andp.19133480105/abstract

[87] U. Emekli, D. Schneidman-Duhovny, H. J. Wolfson, R. Nussinov, and T. Haliloglu, "Hingeprot: automated prediction of hinges in protein structures," *Proteins: Structure, Function, and Bioinformatics*, vol. 70, no. 4, pp. 1219–1227, 2008. [Online]. Available: http://onlinelibrary.wiley.com/doi/10.1002/prot.21613/full

[88] A. Atilgan, S. Durell, R. Jernigan, M. Demirel, O. Keskin, and I. Bahar, "Anisotropy of fluctuation dynamics of proteins with an elastic network model," *Biophysical Journal*, vol. 80, no. 1, pp. 505–515, 2001.

[89] T. Zhang, E. Bertelsen, and T. Alber, "Entropic effects of disulphide bonds on protein stability," *Nature Structural & Molecular Biology*, vol. 1, no. 7, pp. 434–438, 1994. [Online]. Available: http://www.nature.com/nsmb/journal/v1/n7/abs/nsb0794-434.html

[90] O. V. Tsodikov, M. T. Record, and Y. V. Sergeev, "Novel computer program for fast exact calculation of accessible and molecular surface areas and average surface curvature," *Journal of computational chemistry*, vol. 23, no. 6, pp. 600–609, 2002. [Online]. Available: http://onlinelibrary.wiley.com/doi/10.1002/jcc.10061/full

[91] G. Roi, K. Klara, K. Rachel, and K. Chen, "A library of protein surface patches discriminates between native structures and decoys generated by

structure prediction servers," *BMC Structural Biology*, vol. 11.

[92] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," in *ACM Siggraph Computer Graphics*, vol. 21, no. 4. ACM, 1987, pp. 163–169. [Online]. Available: http://dl.acm.org/citation.cfm?id=37422

[93] B. Honig and A. Nicholls, "Classical electrostatics in biology and chemistry," *Science*, vol. 268, no. 5214, pp. 1144–1149, 1995. [Online]. Available: http://www.sciencemag.org/content/268/5214/1144.short

[94] G. Steinkellner, R. Rader, G. G. Thallinger, C. Kratky, and K. Gruber, "Vasco: computation and visualization of annotated protein surface contacts," *BMC bioinformatics*, vol. 10, no. 1, p. 32, 2009.

[95] T. Nepusz, R. Sasidharan, and A. Paccanaro, "Scps: a fast implementation of a spectral method for detecting protein families on a genome-wide scale," *BMC bioinformatics*, vol. 11, no. 1, p. 120, 2010.

[96] D. B. Finkelstein and C. Ball, *Biotechnology of filamentous fungi: technology and products.* Butterworth-Heinemann, 1992.

[97] A. Banerjee, A. Kundu, and S. Ghosh, "Genetic manipulation of filamentous fungi," *New horizons in biotechnology. Kluwer, Dordrecht (Neth)*, pp. 193–198, 2003.

[98] J. R. Newman, S. Ghaemmaghami, J. Ihmels, D. K. Breslow, M. Noble, J. L. DeRisi, and J. S. Weissman, "Single-cell proteomic analysis of s. cerevisiae reveals the architecture of biological noise," *Nature*, vol. 441, no. 7095, pp. 840–6, 2006.

[99] S. P. Gygi, Y. Rochon, B. R. Franza, and R. Aebersold, "Correlation between protein and mrna abundance in yeast," *Mol Cell Biol*, vol. 19, no. 3, pp. 1720–30, 1999.

[100] M. P. Washburn, A. Koller, G. Oshiro, R. R. Ulaszek, D. Plouffe, C. Deciu, E. Winzeler, and r. Yates, J. R., "Protein pathway and complex clustering of correlated mrna and protein expression analyses in saccharomyces cerevisiae," *Proc Natl Acad Sci U S A*, vol. 100, no. 6, pp. 3107–12, 2003.

[101] E. Angov, "Codon usage: nature's roadmap to expression and folding of proteins," *Biotechnol J*, vol. 6, no. 6, pp. 650–9, 2011.

[102] H. Hieronymus and P. A. Silver, "A systems view of mrnp biology," *Genes Dev*, vol. 18, no. 23, pp. 2845–60, 2004.

[103] C. Gustafsson, J. Minshull, S. Govindarajan, J. Ness, A. Villalobos, and M. Welch, "Engineering genes for predictable protein expression," *Protein Expr Purif*, vol. 83, no. 1, pp. 37–46, 2012.

[104] V. L. MacKay, X. Li, M. R. Flory, E. Turcott, G. L. Law, K. A. Serikawa, X. L. Xu, H. Lee, D. R. Goodlett, R. Aebersold, L. P. Zhao, and D. R. Morris, "Gene expression analyzed by high-resolution state array analysis and quantitative proteomics: response of yeast to mating pheromone," *Mol Cell Proteomics*, vol. 3, no. 5, pp. 478–89, 2004.

[105] D. A. Day and M. F. Tuite, "Post-transcriptional gene regulatory mechanisms in eukaryotes: an overview," *J Endocrinol*, vol. 157, no. 3, pp. 361–71, 1998.

[106] S. Wang and T. Hazelrigg, "Implications for bcd mrna localization from spatial distribution of exu protein in drosophila oogenesis," *Nature*, vol. 369,

no. 6479, pp. 400–03, 1994.

[107] C. Vogel, S. Abreu Rde, D. Ko, S. Y. Le, B. A. Shapiro, S. C. Burns, D. Sandhu, D. R. Boutz, E. M. Marcotte, and L. O. Penalva, "Sequence signatures and mrna concentration can explain two-thirds of protein abundance variation in a human cell line," *Mol Syst Biol*, vol. 6, p. 400, 2010.

[108] D. A. Gay, S. S. Sisodia, and D. W. Cleveland, "Autoregulatory control of beta-tubulin mrna stability is linked to translation elongation," *Proc Natl Acad Sci U S A*, vol. 86, no. 15, pp. 5763–7, 1989.

[109] C. C. Oliveira and J. E. McCarthy, "The relationship between eukaryotic translation and mrna stability. a short upstream open reading frame strongly inhibits translational initiation and greatly accelerates mrna degradation in the yeast saccharomyces cerevisiae," *J Biol Chem*, vol. 270, no. 15, pp. 8936–43, 1995.

[110] M. E. Filbin and J. S. Kieft, "Toward a structural understanding of ires rna function," *Curr Opin Struct Biol*, vol. 19, no. 3, pp. 267–76, 2009.

[111] R. de Sousa Abreu, L. O. Penalva, E. M. Marcotte, and C. Vogel, "Global signatures of protein and mrna expression levels," *Mol Biosyst*, vol. 5, no. 12, pp. 1512–26, 2009.

[112] S. E. Calvo, D. J. Pagliarini, and V. K. Mootha, "Upstream open reading frames cause widespread reduction of protein expression and are polymorphic among humans," *Proc Natl Acad Sci U S A*, vol. 106, no. 18, pp. 7507–12, 2009.

[113] T. Preiss and M. W. Hentze, "Dual function of the messenger rna cap structure in poly(a)-tail-promoted translation in yeast," *Nature*, vol. 392, no.

6675, pp. 516–20, 1998.

[114] M. D. Ermolaeva, “Synonymous codon usage in bacteria,” *Curr Issues Mol Biol*, vol. 3, no. 4, pp. 91–7, 2001.

[115] A. Bachmair, D. Finley, and A. Varshavsky, “In vivo half-life of a protein is a function of its amino-terminal residue,” *Science*, vol. 234, no. 4773, pp. 179–86, 1986.

[116] A. Hershko, “Ubiquitin-mediated protein degradation,” *J Biol Chem*, vol. 263, no. 30, pp. 15 237–40, 1988.

[117] J. T. Rogers, A. I. Bush, H. H. Cho, D. H. Smith, A. M. Thomson, A. L. Friedlich, D. K. Lahiri, P. J. Leedman, X. Huang, and C. M. Cahill, “Iron and the translation of the amyloid precursor protein (app) and ferritin mrnas: riboregulation against neural oxidative damage in alzheimer’s disease,” *Biochem Soc Trans*, vol. 36, no. Pt 6, pp. 1282–7, 2008.

[118] M. L. Spencer, M. Theodosiou, and D. J. Noonan, “Npdc-1, a novel regulator of neuronal proliferation, is degraded by the ubiquitin/proteasome system through a pest degradation motif,” *J Biol Chem*, vol. 279, no. 35, pp. 37 069–78, 2004.

[119] M. Rechsteiner and S. W. Rogers, “Pest sequences and regulation by proteolysis,” *Trends Biochem Sci*, vol. 21, no. 7, pp. 267–71, 1996.

[120] P. Tompa, J. Prilusky, I. Silman, and J. L. Sussman, “Structural disorder serves as a weak signal for intracellular protein degradation,” *Proteins*, vol. 71, no. 2, pp. 903–9, 2008.

[121] H. J. Dyson and P. E. Wright, “Intrinsically unstructured proteins and their functions,” *Nat Rev Mol Cell Biol*, vol. 6, no. 3, pp. 197–208, 2005.

[122] J. Gsponer, M. E. Futschik, S. A. Teichmann, and M. M. Babu, "Tight regulation of unstructured proteins: from transcript synthesis to protein degradation," *Science*, vol. 322, no. 5906, pp. 1365–8, 2008.

[123] L. A. Palomares, S. Estrada-Moncada, and O. T. Ramírez, "Production of recombinant proteins," in *Recombinant Gene Expression*. Springer, 2004, pp. 15–51. [Online]. Available: http://link.springer.com/protocol/10.1385/1-59259-774-2

[124] P. Lu, C. Vogel, R. Wang, X. Yao, and E. M. Marcotte, "Absolute protein expression profiling estimates the relative contributions of transcriptional and translational regulation," *Nat Biotechnol*, vol. 25, no. 1, pp. 117–24, 2007.

[125] Y. Ishihama, T. Schmidt, J. Rappsilber, M. Mann, F. U. Hartl, M. J. Kerner, and D. Frishman, "Protein abundance profiling of the escherichia coli cytosol," *BMC Genomics*, vol. 9, p. 102, 2008.

[126] S. Ghaemmaghami, W. K. Huh, K. Bower, R. W. Howson, A. Belle, N. Dephoure, E. K. O'Shea, and J. S. Weissman, "Global analysis of protein expression in yeast," *Nature*, vol. 425, no. 6959, pp. 737–41, 2003.

[127] L. Issel-Tarver, K. R. Christie, K. Dolinski, R. Andrada, R. Balakrishnan, C. A. Ball, G. Binkley, S. Dong, S. S. Dwight, D. G. Fisk *et al.*, "Saccharomyces genome database." *Methods in enzymology*, vol. 350, p. 329, 2002. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/12073322

[128] P. T. Spellman, G. Sherlock, M. Q. Zhang, V. R. Iyer, K. Anders, M. B. Eisen, P. O. Brown, D. Botstein, and B. Futcher, "Comprehensive identifi-

cation of cell cycle-regulated genes of the yeast saccharomyces cerevisiae by microarray hybridization," *Mol Biol Cell*, vol. 9, no. 12, pp. 3273–97, 1998.

[129] T. Niwa, B. W. Ying, K. Saito, W. Jin, S. Takada, T. Ueda, and H. Taguchi, "Bimodal protein solubility distribution revealed by an aggregation analysis of the entire ensemble of escherichia coli proteins," *Proc Natl Acad Sci U S A*, vol. 106, no. 11, pp. 4201–6, 2009, niwa, Tatsuya Ying, Bei-Wen Saito, Katsuyo Jin, WenZhen Takada, Shoji Ueda, Takuya Taguchi, Hideki Research Support, Non-U.S. Gov't United States Proceedings of the National Academy of Sciences of the United States of America Proc Natl Acad Sci U S A. 2009 Mar 17;106(11):4201-6. doi: 10.1073/pnas.0811922106. Epub 2009 Feb 27.

[130] Y. Shimizu, A. Inoue, Y. Tomari, T. Suzuki, T. Yokogawa, K. Nishikawa, and T. Ueda, "Cell-free translation reconstituted with purified components," *Nat Biotechnol*, vol. 19, no. 8, pp. 751–5, 2001.

[131] Y. Shimizu, T. Kanamori, and T. Ueda, "Protein synthesis by pure translation systems," *Methods*, vol. 36, no. 3, pp. 299–304, 2005.

[132] M. Kitagawa, T. Ara, M. Arifuzzaman, T. Ioka-Nakamichi, E. Inamoto, H. Toyonaga, and H. Mori, "Complete set of orf clones of escherichia coli aska library (a complete set of e. coli k-12 orf archive): unique resources for biological research," *DNA Res*, vol. 12, no. 5, pp. 291–9, 2005.

[133] B. W. Ying, H. Taguchi, H. Ueda, and T. Ueda, "Chaperone-assisted folding of a single-chain antibody in a reconstituted translation system," *Biochem Biophys Res Commun*, vol. 320, no. 4, pp. 1359–64, 2004.

[134] T. E. Allen, M. J. Herrgard, M. Liu, Y. Qiu, J. D. Glasner, F. R. Blattner, and B. O. Palsson, “Genome-scale analysis of the uses of the escherichia coli genome: model-driven analysis of heterogeneous data sets,” *J Bacteriol*, vol. 185, no. 21, pp. 6392–9, 2003.

[135] R. W. Corbin, O. Paliy, F. Yang, J. Shabanowitz, M. Platt, J. Lyons, C. E., K. Root, J. McAuliffe, M. I. Jordan, S. Kustu, E. Soupene, and D. F. Hunt, “Toward a protein profile of escherichia coli: comparison to its transcription profile,” *Proc Natl Acad Sci U S A*, vol. 100, no. 16, pp. 9232–7, 2003.

[136] M. W. Covert, E. M. Knight, J. L. Reed, M. J. Herrgard, and B. O. Palsson, “Integrating high-throughput and computational data elucidates bacterial networks,” *Nature*, vol. 429, no. 6987, pp. 92–6, 2004.

[137] S. S. Fong, A. R. Joyce, and B. O. Palsson, “Parallel adaptive evolution cultures of escherichia coli lead to convergent growth phenotypes with different gene expression states,” *Genome Res*, vol. 15, no. 10, pp. 1365–72, 2005.

[138] H. Liu, R. G. Sadygov, and r. Yates, J. R., “A model for random sampling and estimation of relative protein abundance in shotgun proteomics,” *Anal Chem*, vol. 76, no. 14, pp. 4193–201, 2004.

[139] R. Brockmann, A. Beyer, J. J. Heinisch, and T. Wilhelm, “Posttranscriptional expression regulation: what determines translation rates?” *PLoS Comput Biol*, vol. 3, no. 3, p. e57, 2007.

[140] A. Beyer, J. Hollunder, H. P. Nasheuer, and T. Wilhelm, “Post-transcriptional expression regulation in the yeast saccharomyces cerevisiae on a genomic scale,” *Mol Cell Proteomics*, vol. 3, no. 11, pp. 1083–92, 2004, beyer, Andreas Hollunder, Jens Nasheuer, Heinz-Peter Wilhelm, Thomas

Research Support, Non-U.S. Gov't United States Molecular & cellular proteomics : MCP Mol Cell Proteomics. 2004 Nov;3(11):1083-92. Epub 2004 Aug 23.

[141] P. Shah and M. A. Gilchrist, "Explaining complex codon usage patterns with selection for translational efficiency, mutation bias, and genetic drift," *Proc Natl Acad Sci U S A*, vol. 108, no. 25, pp. 10 231–6, 2011.

[142] M. Welch, S. Govindarajan, J. E. Ness, A. Villalobos, A. Gurney, J. Minshull, and C. Gustafsson, "Design parameters to control synthetic gene expression in escherichia coli," *PLoS One*, vol. 4, no. 9, p. e7002, 2009.

[143] C. Mehlin, E. Boni, F. S. Buckner, L. Engel, T. Feist, M. H. Gelb, L. Haji, D. Kim, C. Liu, N. Mueller, P. J. Myler, J. T. Reddy, J. N. Sampson, E. Subramanian, W. C. Van Voorhis, E. Worthey, F. Zucker, and W. G. Hol, "Heterologous expression of proteins from plasmodium falciparum: results from 1000 genes," *Mol Biochem Parasitol*, vol. 148, no. 2, pp. 144–60, 2006.

[144] M. Boettner, C. Steffens, C. von Mering, P. Bork, U. Stahl, and C. Lang, "Sequence-based factors influencing the expression of heterologous genes in the yeast pichia pastoris–a comparative view on 79 human genes," *J Biotechnol*, vol. 130, no. 1, pp. 1–10, 2007.

[145] N. A. Burgess-Brown, S. Sharma, F. Sobott, C. Loenarz, U. Oppermann, and O. Gileadi, "Codon optimization can improve expression of human genes in escherichia coli: A multi-gene study," *Protein Expr Purif*, vol. 59, no. 1, pp. 94–102, 2008.

[146] B. Maertens, A. Spriestersbach, U. von Groll, U. Roth, J. Kubicek, M. Gerrits, M. Graf, M. Liss, D. Daubert, R. Wagner, and F. Schafer, "Gene opti-

mization mechanisms: a multi-gene study reveals a high success rate of full-length human proteins expressed in escherichia coli," *Protein Sci*, vol. 19, no. 7, pp. 1312–26, 2010.

[147] S. Fath, A. P. Bauer, M. Liss, A. Spriestersbach, B. Maertens, P. Hahn, C. Ludwig, F. Schafer, M. Graf, and R. Wagner, "Multiparameter rna and codon optimization: a standardized tool to assess and enhance autologous mammalian gene expression," *PLoS One*, vol. 6, no. 3, p. e17596, 2011.

[148] G. Wu, Y. Zheng, I. Qureshi, H. T. Zin, T. Beck, B. Bulka, and S. J. Freeland, "Sgdb: a database of synthetic genes re-designed for optimizing protein over-expression," *Nucleic Acids Res*, vol. 35, no. Database issue, pp. D76–9, 2007.

[149] M. Hubert and K. V. Branden, "Robust methods for partial least squares regression," *Journal of Chemometrics*, vol. 17, no. 10, pp. 537–549, 2003. [Online]. Available: http://onlinelibrary.wiley.com/doi/10.1002/cem.822/abstract

[150] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012. [Online]. Available: http://arxiv.org/abs/1207.0580

[151] N. Srivastava, "Improving neural networks with dropout," Ph.D. dissertation, University of Toronto, 2013.

[152] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, 2012, pp. 1106–1114.

[153] C. W. Dunnett, “A multiple comparison procedure for comparing several treatments with a control,” *Journal of the American Statistical Association*, vol. 50, no. 272, pp. 1096–1121, 1955. [Online]. Available: http://www.tandfonline.com/doi/pdf/10.1080/01621459.1955.10501294

[154] P. M. Sharp, M. Stenico, J. F. Peden, and A. T. Lloyd, “Codon usage: mutational bias, translational selection, or both?” *Biochem Soc Trans*, vol. 21, no. 4, pp. 835–841, Nov 1993.

[155] A. Kurotani, T. Takagi, M. Toyama, M. Shirouzu, S. Yokoyama, Y. Fukami, and A. A. Tokmakov, “Comprehensive bioinformatics analysis of cell-free protein synthesis: identification of multiple protein properties that correlate with successful expression,” *The FASEB Journal*, vol. 24, no. 4, pp. 1095–1104, 2010. [Online]. Available: http://www.fasebj.org/content/24/4/1095.short

[156] T. Tuller, Y. Y. Waldman, M. Kupiec, and E. Ruppin, “Translation efficiency is determined by both codon bias and folding energy,” *Proceedings of the National Academy of Sciences*, vol. 107, no. 8, pp. 3645–3650, 2010. [Online]. Available: http://www.pnas.org/content/107/8/3645.short

[157] A. A. Tokmakov, A. Kurotani, T. Takagi, M. Toyama, M. Shirouzu, Y. Fukami, and S. Yokoyama, “Multiple post-translational modifications affect heterologous protein synthesis,” *Journal of Biological Chemistry*, vol. 287, no. 32, pp. 27 106–27 116, 2012. [Online]. Available: http://www.jbc.org/content/287/32/27106.short

[158] T. L. Bailey, C. Elkan *et al.*, “Fitting a mixture model by expectation maximization to discover motifs in bipolymers,” 1994.

[159] M. C. Frith, N. F. Saunders, B. Kobe, and T. L. Bailey, "Discovering sequence motifs with arbitrary insertions and deletions," *PLoS computational biology*, vol. 4, no. 5, p. e1000071, 2008.

[160] S. A. Rosenberg, J. C. Yang, D. J. Schwartzentruber, P. Hwu, F. M. Marincola, S. L. Topalian, N. P. Restifo, M. E. Dudley, S. L. Schwarz, P. J. Spiess, J. R. Wunderlich, M. R. Parkhurst, Y. Kawakami, C. A. Seipp, J. H. Einhorn, and D. E. White, "Immunologic and therapeutic evaluation of a synthetic peptide vaccine for the treatment of patients with metastatic melanoma," *Nat Med*, vol. 4, no. 3, pp. 321–7, 1998.

[161] B. Murphy, K. S. Kim, R. Buelow, M. H. Sayegh, and W. W. Hancock, "Synthetic mhc class i peptide prolongs cardiac survival and attenuates transplant arteriosclerosis in the lewis–¿fischer 344 model of chronic allograft rejection," *Transplantation*, vol. 64, no. 1, pp. 14–9, 1997.

[162] N. Zavazava, F. Fandrich, X. Zhu, A. Freese, D. Behrens, and K. A. Yoo-Ott, "Oral feeding of an immunodominant mhc donor-derived synthetic class i peptide prolongs graft survival of heterotopic cardiac allografts in a high-responder rat strain combination," *J Leukoc Biol*, vol. 67, no. 6, pp. 793–800, 2000.

[163] K. Natarajan, H. Li, R. A. Mariuzza, and D. H. Margulies, "Mhc class i molecules, structure and function," *Rev Immunogenet*, vol. 1, no. 1, pp. 32–46, 1999.

[164] H. C. Guo, T. S. Jardetzky, T. P. Garrett, W. S. Lane, J. L. Strominger, and D. C. Wiley, "Different length peptides bind to hla-aw68 similarly at their

ends but bulge out in the middle," *Nature*, vol. 360, no. 6402, pp. 364–6, 1992.

[165] J. A. Speir, J. Stevens, E. Joly, G. W. Butcher, and I. A. Wilson, "Two different, highly exposed, bulged structures for an unusually long peptide bound to rat mhc class i rt1-aa," *Immunity*, vol. 14, no. 1, pp. 81–92, 2001.

[166] T. N. Schumacher, M. L. De Bruijn, L. N. Vernie, W. M. Kast, C. J. Melief, J. J. Neefjes, and H. L. Ploegh, "Peptide selection by mhc class i molecules," *Nature*, vol. 350, no. 6320, pp. 703–6, 1991.

[167] C. A. Nelson and D. H. Fremont, "Structural principles of mhc class ii antigen presentation," *Rev Immunogenet*, vol. 1, no. 1, pp. 47–59, 1999.

[168] M. Yassai, A. Afsari, J. Garlie, and J. Gorski, "C-terminal anchoring of a peptide to class ii mhc via the p10 residue is compatible with a peptide bulge," *J Immunol*, vol. 168, no. 3, pp. 1281–5, 2002.

[169] H. G. Rammensee, T. Friede, and S. Stevanoviic, "Mhc ligands and peptide motifs: first listing," *Immunogenetics*, vol. 41, no. 4, pp. 178–228, 1995.

[170] A. J. Godkin, K. J. Smith, A. Willis, M. V. Tejada-Simon, J. Zhang, T. Elliott, and A. V. Hill, "Naturally processed hla class ii peptides reveal highly conserved immunogenic flanking region sequence preferences that reflect antigen processing rather than peptide-mhc interactions," *J Immunol*, vol. 166, no. 11, pp. 6720–7, 2001.

[171] E. Y. Jones, L. Fugger, J. L. Strominger, and C. Siebold, "Mhc class ii proteins and disease: a structural perspective," *Nat Rev Immunol*, vol. 6, no. 4, pp. 271–82, 2006.

[172] V. Brusic, V. B. Bajic, and N. Petrovsky, "Computational methods for prediction of t-cell epitopes–a framework for modelling, testing, and applications," *Methods*, vol. 34, no. 4, pp. 436–43, 2004.

[173] S. G. E. Marsh, P. Parham, and L. D. Barber, *The HLA factsbook*, ser. Factsbook series. San Diego: Academic Press, 2000.

[174] M. W. Firebaugh, *Artificial intelligence: a knowledge-based approach.* Boston: Boyd and Fraser, 1988.

[175] R. J. Schalkoff, *Pattern recognition : statistical, structural, and neural approaches.* New York: J. Wiley, 1992.

[176] B. Trost, M. Bickis, and A. Kusalik, "Strength in numbers: achieving greater accuracy in mhc-i binding prediction by combining the results from multiple prediction tools," *Immunome Res*, vol. 3, p. 5, 2007.

[177] P. Wang, J. Sidney, C. Dow, B. Mothe, A. Sette, and B. Peters, "A systematic assessment of mhc class ii peptide binding predictions and evaluation of a consensus approach," *PLoS Comput Biol*, vol. 4, no. 4, p. e1000048, 2008.

[178] M. Nielsen, O. Lund, S. Buus, and C. Lundegaard, "Mhc class ii epitope predictive algorithms," *Immunology*, vol. 130, no. 3, pp. 319–28, 2010.

[179] M. Matsumura, D. H. Fremont, P. A. Peterson, and I. A. Wilson, "Emerging principles for the recognition of peptide antigens by mhc class i molecules," *Science*, vol. 257, no. 5072, pp. 927–34, 1992.

[180] B. Peters, J. Sidney, P. Bourne, H. H. Bui, S. Buus, G. Doh, W. Fleri, M. Kronenberg, R. Kubo, O. Lund, D. Nemazee, J. V. Ponomarenko, M. Sathiamurthy, S. Schoenberger, S. Stewart, P. Surko, S. Way, S. Wil-

son, and A. Sette, “The immune epitope database and analysis resource: from vision to blueprint,” *PLoS Biol*, vol. 3, no. 3, p. e91, 2005.

[181] B. Peters, H. H. Bui, S. Frankild, M. Nielson, C. Lundegaard, E. Kostem, D. Basch, K. Lamberth, M. Harndahl, W. Fleri, S. S. Wilson, J. Sidney, O. Lund, S. Buus, and A. Sette, “A community resource benchmarking predictions of peptide binding to mhc-i molecules,” *PLoS Comput Biol*, vol. 2, no. 6, p. e65, 2006.

[182] P. Wang, J. Sidney, Y. Kim, A. Sette, O. Lund, M. Nielsen, and B. Peters, “Peptide binding predictions for hla dr, dp and dq molecules,” *BMC Bioinformatics*, vol. 11, p. 568, 2010.

[183] R. Agrawal, T. Imielinski, and A. Swami, “Mining association rules between sets of items in large databases,” 1993.

[184] R. Agrawal and R. Srikant, “Fast algorithms for mining association rules in large databases,” 1994.

[185] R. Srikant and R. Agrawal, “Mining generalized association rules,” *Future Generation Computer Systems*, vol. 13, no. 2-3, pp. 161–180, 1997.

[186] J. F. Roddick and M. Spiliopoulou, “A bibliography of temporal, spatial and spatio-temporal data mining research,” *SIGKDD Explor. Newsl.*, vol. 1, no. 1, pp. 34–38, 1999.

[187] H. Mannila, H. Toivonen, and A. I. Verkamo, “Discovery of frequent episodes in event sequences,” *Data Mining and Knowledge Discovery*, vol. 1, no. 3, pp. 259–289, 1997.

[188] A. Solache, C. L. Morgan, A. I. Dodi, C. Morte, I. Scott, C. Baboonian, B. Zal, J. Goldman, J. E. Grundy, and J. A. Madrigal, “Identification of three

hla-a*0201-restricted cytotoxic t cell epitopes in the cytomegalovirus protein pp65 that are conserved between eight strains of the virus," *J Immunol*, vol. 163, no. 10, pp. 5512–8, 1999.

[189] V. Mitaksov and D. H. Fremont, "Structural definition of the h-2kd peptide-binding motif," *J Biol Chem*, vol. 281, no. 15, pp. 10 618–25, 2006.

[190] M. Nielsen, C. Lundegaard, P. Worning, S. L. Lauemoller, K. Lamberth, S. Buus, S. Brunak, and O. Lund, "Reliable prediction of t-cell epitopes using neural networks with novel sequence representations," *Protein Sci*, vol. 12, no. 5, pp. 1007–17, 2003.

[191] M. Nielsen, C. Lundegaard, and O. Lund, "Prediction of mhc class ii binding affinity using smm-align, a novel stabilization matrix alignment method," *BMC Bioinformatics*, vol. 8, p. 238, 2007.

[192] H. H. Bui, J. Sidney, B. Peters, M. Sathiamurthy, A. Sinichi, K. A. Purton, B. R. Mothe, F. V. Chisari, D. I. Watkins, and A. Sette, "Automated generation and evaluation of specific mhc binding predictive tools: Arb matrix applications," *Immunogenetics*, vol. 57, no. 5, pp. 304–14, 2005.

[193] M. Nielsen and O. Lund, "Nn-align. an artificial neural network-based alignment algorithm for mhc class ii peptide binding prediction," *BMC Bioinformatics*, vol. 10, p. 296, 2009.

[194] O. Rotzschke, K. Falk, J. Mack, J. M. Lau, G. Jung, and J. L. Strominger, "Conformational variants of class ii mhc/peptide complexes induced by n- and c-terminal extensions of minimal peptide epitopes," *Proc Natl Acad Sci U S A*, vol. 96, no. 13, pp. 7445–50, 1999.

[195] C. Lundegaard, O. Lund, and M. Nielsen, “Accurate approximation method for prediction of class i mhc affinities for peptides of length 8, 10 and 11 using prediction tools trained on 9mers,” *Bioinformatics*, vol. 24, no. 11, pp. 1397–8, 2008.

[196] C. A. Scott, P. A. Peterson, L. Teyton, and I. A. Wilson, “Crystal structures of two i-ad-peptide complexes reveal that high affinity can be achieved without large anchor residues,” *Immunity*, vol. 8, no. 3, pp. 319–29, 1998.

[197] H. Kropshofer, H. Max, C. A. Muller, F. Hesse, S. Stevanovic, G. Jung, and H. Kalbacher, “Self-peptide released from class ii hla-dr1 exhibits a hydrophobic two-residue contact motif,” *J Exp Med*, vol. 175, no. 6, pp. 1799–803, 1992.

[198] B. Fleckenstein, H. Kalbacher, C. P. Muller, D. Stoll, T. Halder, G. Jung, and K. H. Wiesmuller, “New ligands binding to the human leukocyte antigen class ii molecule drb1*0101 based on the activity pattern of an undecapeptide library,” *Eur J Biochem*, vol. 240, no. 1, pp. 71–7, 1996.

[199] K. J. Smith, J. Pyrdol, L. Gauthier, D. C. Wiley, and K. W. Wucherpfennig, “Crystal structure of hla-dr2 (dra*0101, drb1*1501) complexed with a peptide from human myelin basic protein,” *J Exp Med*, vol. 188, no. 8, pp. 1511–20, 1998.

[200] W. Thompson, E. C. Rouchka, and C. E. Lawrence, “Gibbs recursive sampler: finding transcription factor binding sites,” *Nucleic Acids Research*, vol. 31, no. 13, pp. 3580–3585, 2003.

[201] T. L. Bailey, M. Boden, F. A. Buske, M. Frith, C. E. Grant, L. Clementi, J. Ren, W. W. Li, and W. S. Noble, “Meme suite: tools for motif discovery

and searching," *Nucleic acids research*, vol. 37, no. suppl 2, pp. W202–W208, 2009.

[202] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological cybernetics*, vol. 36, no. 4, pp. 193–202, 1980. [Online]. Available: http://link.springer.com/article/10.1007/BF00344251

[203] Y. Bengio and Y. LeCun, "Scaling learning algorithms towards ai," *Large-Scale Kernel Machines*, vol. 34, 2007.

[204] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. Huang, "A tutorial on energy-based learning," *Predicting Structured Data*, 2006.

[205] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," *Advances in neural information processing systems*, vol. 19, p. 153, 2007.

[206] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the 26th Annual International Conference on Machine Learning.* ACM, 2009, pp. 609–616. [Online]. Available: http://dl.acm.org/citation.cfm?id=1553453

[207] C. E. Shannon and W. Weaver, "A mathematical theory of communication," 1948.

[208] R. Apweiler, A. Bairoch, C. H. Wu, W. C. Barker, B. Boeckmann, S. Ferro, E. Gasteiger, H. Huang, R. Lopez, M. Magrane *et al.*, "Uniprot: the universal protein knowledgebase," *Nucleic acids research*, vol. 32, no. suppl 1, pp. D115–D119, 2004.

[209] A. Bairoch, R. Apweiler, C. H. Wu, W. C. Barker, B. Boeckmann, S. Ferro, E. Gasteiger, H. Huang, R. Lopez, M. Magrane *et al.*, “The universal protein resource (uniprot),” *Nucleic acids research*, vol. 33, no. suppl 1, pp. D154–D159, 2005.

A PARAMETERS USED DURING CALCULATIONS

A.1 Tables for Chapter 2

Table A.1: BLOSUM62 similarity scores between the 20 amino acids

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

Table A.2:
Shifted Thomas-Dill Contact Potentials for 20 amino acids

	C	M	F	I	L	V	W	Y	A	G	T	S	Q	N	E	D	H	R	K	P
C	1	1.56	1.81	2.31	2.1	1.85	2.49	1.83	2.49	2.37	2.41	2.59	2.3	2.47	2.83	3.34	1.97	2.39	2.79	2.86
M		3.15	1.76	2.38	2.48	1.85	2.72	1.69	2.84	2.79	2.85	2.32	2.25	3.1	2.81	3.86	2.44	2.36	3.34	2.54
F			2.18	2.13	1.77	2.01	1.9	1.97	2.74	3	2.6	2.93	2.89	2.77	2.98	2.99	2.04	2.57	2.62	2.36
I				2.08	1.75	1.81	1.9	1.92	2.15	3.19	2.5	2.66	2.4	3.18	2.59	2.83	2.27	2.71	2.53	3.04
L					1.65	1.76	1.82	2.19	2.22	2.71	2.4	2.72	2.66	2.69	2.74	3.29	2.43	2.69	2.89	2.88
V						1.64	2.19	2.09	2.19	2.59	2.85	2.48	2.7	2.55	2.77	3.04	2.44	2.31	2.71	2.71
W							2.81	1.8	2.71	2.65	2.86	2.59	3.19	2.11	3.11	3.03	2.38	2.01	2.49	2.35
Y								3.14	2.42	2.47	2.56	3.04	2.4	2.05	3.01	2.9	2.12	3	2.59	2.34
A									2.71	2.7	2.57	2.78	2.68	2.65	2.82	2.89	2.64	2.86	2.79	3.2
G										2.83	2.92	2.75	2.91	2.61	3.19	2.73	2.79	2.64	2.89	3.19
T											3.05	2.84	2.62	2.52	2.94	2.76	2.52	2.62	2.88	3.15
S												2.66	3.19	3.16	3.09	2.7	2.2	3.4	2.97	3.23
Q													2.71	2.74	3.41	3.25	2.84	3.41	2.83	2.58
N														1.93	2.54	2.67	2.85	2.83	2.97	2.9
E															3	3.47	2.26	2.53	2.7	3.12
D																3.39	2.73	2.64	2.7	3.63
H																	2.93	2.78	2.93	2.57
R																		3.02	3.09	2.77
K																			4.24	3.3
P																				3.07

Table A.3:

Ideal values of ψ and ϕ dihedral angles that characterize the 16 Protein Blocks

Protein Block	Dihedral angles (in degrees)							
	ψ_{n-2}	ϕ_{n-1}	ψ_{n-1}	ϕ_n	ψ_n	ϕ_{n+1}	ψ_{n+1}	ϕ_{n+2}
a	41.14	75.53	13.92	-99.8	131.88	-96.27	122.08	-99.68
b	108.24	-90.12	119.54	-92.21	-18.06	-128.93	147.04	-99.9
c	-11.61	-105.66	94.81	-106.09	133.56	-106.93	135.97	-100.63
d	141.98	-112.79	132.2	-114.79	140.11	-111.05	139.54	-103.16
e	133.25	-112.37	137.64	-108.13	133	-87.3	120.54	77.4
f	116.4	-105.53	129.32	-96.68	140.72	-74.19	-26.65	-94.51
g	0.4	-81.83	4.91	-100.59	85.5	-71.65	130.78	84.98
h	119.14	-102.58	130.83	-67.91	121.55	76.25	-2.95	-90.88
i	130.68	-56.92	119.26	77.85	10.42	-99.43	141.4	-98.01
j	114.32	-121.47	118.14	82.88	-150.05	-83.81	23.35	-85.82
k	117.16	-95.41	140.4	-59.35	-29.23	-72.39	-25.08	-76.16
l	139.2	-55.96	-32.7	-68.51	-26.09	-74.44	-22.6	-71.74
m	-39.62	-64.73	-39.52	-65.54	-38.88	-66.89	-37.76	-70.19
n	-35.34	-65.03	-38.12	-66.34	-29.51	-89.1	-2.91	77.9
o	-45.29	-67.44	-27.72	-87.27	5.13	77.49	30.71	-93.23
p	-27.09	-86.14	0.3	59.85	21.51	-96.3	132.3	-92.91

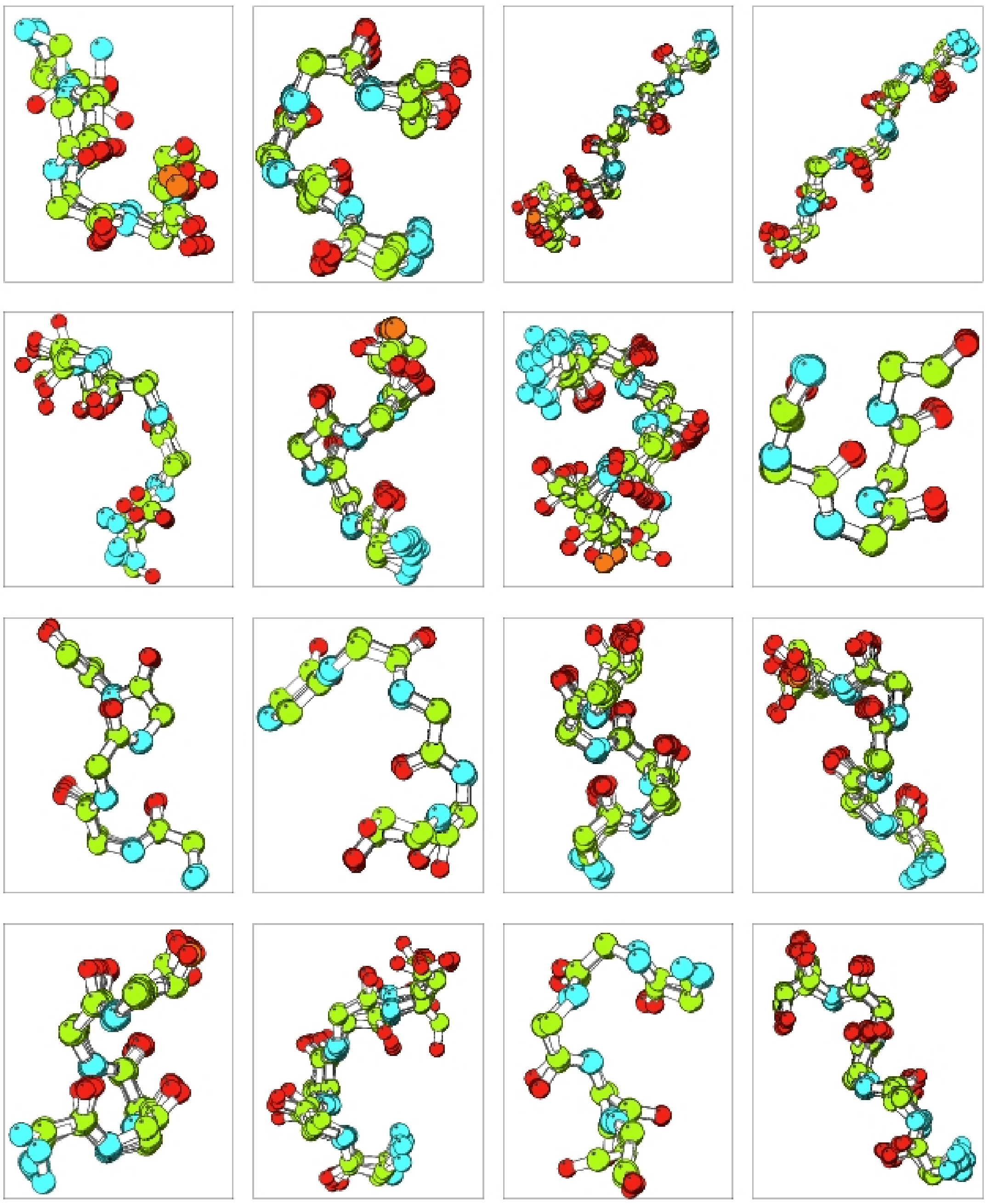


Figure A.1:
Visualization of the 16 protein blocks. Used with permission.

Table A.4:
Similarity matrix for the 16 Protein Blocks

	a	b	c	d	e	f	g	h	i	j	k	l	M	n	o	p
a	2.28	-0.12	0.54	-0.29	-1.59	-0.54	0.31	-1.14	0.39	-1.15	-1.75	-0.6	-2.4	-1.4	-0.54	-0.36
b		2.49	-0.21	-0.44	-0.48	-1.53	-0.73	0.2	0.24	0.32	-0.03	0.04	-2.98	-0.83	-0.55	0.33
c			1.69	0.17	-1.1	-0.39	0.18	-1.63	-1.11	-1.03	-2.45	-2.21	-2.7	-1.68	-0.65	-0.01
d				1.35	-0.36	-0.49	-1.29	-1.2	-1.12	-0.92	-2.63	-1.56	-5.2	-3.07	-2.66	-2.1
e					3.05	0.75	1.37	0.66	-1.15	-0.76	-0.38	-1.76	-4.75	-0.58	-2.48	-2.22
f						2.21	-0.33	-0.34	-1.07	-0.34	-0.04	-0.33	-2.14	-1.99	-1.41	-1.91
g							3.25	-0.74	-0.19	-0.51	-1.39	-0.74	-1.1	1.07	-0.01	0.47
h								3.07	-0.92	1.18	0.51	-0.36	-2.93	-1.07	0.96	-1.81
i									3.37	1.54	-0.15	-0.22	-3.15	-0.97	-0.89	1.32
j										3.74	0.07	-0.12	-2	-0.44	-0.48	0.6
k											2.52	0.19	-1.02	-0.56	-1.71	-1.35
l												2.24	-0.68	-0.27	0.06	-1.23
M													1.06	-0.77	-1.26	-1.1
n														3.65	0.26	0.36
o															3.36	0.24
p																2.83

A.2 Tables for Chapter 3

Table A.5:

Codon Adaptation Index (CAI) and Frequency of Optimal Codons (FOP)/Codon Bias Index (CBI) values for the codons. In FOP/CBI columns, 1 defines a non-optimal/rare codon, 2 defines a common codon and 3 defines an optimal codon.

	CAI		**FOP/CBI**	
	E.coli	***S.cerevisiae***	***E.coli***	***S.cerevisiae***
TTT	0.296	0.113	2	2
TCT	1	1	3	3
TAT	0.239	0.071	2	2
TGT	0.5	1	2	3
TTC	1	1	3	3
TCC	0.744	0.693	3	3
TAC	1	1	3	3
TGC	1	0.077	3	2
TTA	0.02	0.117	2	2
TCA	0.077	0.036	2	2
TAA	0	0	2	2
TGA	0	0	2	2
TTG	0.02	1	2	3
TCG	0.017	0.005	2	2
TAG	0	0	2	2
TGG	1	1	2	2
CTT	0.042	0.006	2	2
CCT	0.07	0.047	2	2
CAT	0.291	0.245	2	2
CGT	1	0.137	3	2
CTC	0.037	0.003	2	2
CCC	0.012	0.009	2	2
CAC	1	1	3	3
CGC	0.356	0.002	3	2
CTA	0.007	0.039	2	2
CCA	0.135	1	2	3
CAA	0.124	1	2	3
CGA	0.004	0.002	2	2
CTG	1	0.003	3	2
CCG	1	0.002	3	2
CAG	1	0.007	3	2
CGG	0.004	0.002	2	2
ATT	0.185	0.823	2	3
ACT	0.965	0.921	3	3
AAT	0.051	0.053	2	2
AGT	0.085	0.021	2	2
ATC	1	1	3	3
ACC	1	1	3	3
AAC	1	1	3	3
AGC	0.41	0.031	3	2
ATA	0.003	0.003	2	2
ACA	0.076	0.012	2	2
AAA	1	0.135	3	2
AGA	0.004	1	2	3
ATG	1	1	2	2
ACG	0.099	0.006	2	2
AAG	0.253	1	2	3
AGG	0.002	0.003	2	2
GTT	1	1	3	3
GCT	1	1	3	3
GAT	0.434	0.554	2	2
GGT	1	1	3	3
GTC	0.066	0.831	2	3
GCC	0.122	0.316	2	2
GAC	1	1	3	3
GGC	0.724	0.02	3	2
GTA	0.495	0.002	2	2
GCA	0.586	0.015	2	2
GAA	1	1	3	3
GGA	0.01	0.002	2	2
GTG	0.221	0.018	2	2
GCG	0.424	0.001	3	2
GAG	0.259	0.016	2	2
GGG	0.019	0.004	2	2

Table A.6:

tRNA copy numbers used in the calculation of tRNA Adaptation Index

	E.coli	***S.cerevisiae***	***S.pombe***	***P.pastoris***	***H.sapiens***
TTT	0	0	0	0	0
TTC	2	10	5	5	53
TTA	1	7	2	1	7
TTG	1	10	4	3	17
TCT	0	11	7	4	11
TCC	2	0	0	0	0
TCA	1	3	2	2	5
TCG	1	1	1	1	4
TAT	0	0	0	0	1
TAC	3	8	4	4	14
TAA	0	0	0	0	2
TAG	0	0	0	0	1
TGT	0	0	0	0	0
TGC	1	4	3	2	30
TGA	1	0	0	0	3
TGG	1	6	3	3	9
CTT	0	0	5	2	23
CTC	1	1	0	0	0
CTA	1	3	1	1	3
CTG	4	0	1	1	10
CCT	0	2	6	2	10
CCC	1	0	0	0	0
CCA	1	10	2	4	7
CCG	1	0	1	1	4
CAT	0	0	0	0	0
CAC	1	7	4	3	11
CAA	2	9	4	3	12
CAG	2	1	2	2	25
CGT	4	6	8	2	7
CGC	0	0	0	0	0
CGA	0	0	1	1	7
CGG	1	1	1	1	16
ATT	0	13	8	5	14
ATC	3	0	0	0	3
ATA	0	2	1	1	6
ATG	8	10	7	2	30
ACT	0	11	7	5	12
ACC	2	0	0	0	0
ACA	1	4	2	1	6
ACG	2	1	1	1	7
AAT	0	0	0	0	2
AAC	4	10	6	4	32
AAA	6	7	3	3	28
AAG	0	14	9	5	17
AGT	0	0	0	0	0
AGC	1	2	3	2	9
AGA	1	11	2	4	6
AGG	1	1	1	1	5
GTT	0	14	9	5	16
GTC	2	0	0	0	0
GTA	5	2	2	1	5
GTG	0	2	1	1	16
GCT	0	11	9	5	65
GCC	2	0	0	0	0
GCA	3	5	2	2	38
GCG	0	0	1	0	15
GAT	0	0	0	0	0
GAC	3	16	8	6	19
GAA	4	14	4	5	13
GAG	0	2	6	4	19
GGT	0	0	0	0	0
GGC	4	16	8	5	15
GGA	1	3	3	3	9
GGG	1	2	1	1	7

Table A.7:

Expected time of translation of a codon used in Ribosome Flow Model calculations

	E.coli	***S.cerevisiae***	***S.pombe***	***P.pastoris***	***H.sapiens***
TTT	72.8814	62.1868	77.9043	42.02152	97.3804
TTC	43	27.3	34.2	24.7927	42.75
TTA	86	39	85.5	123.9635	73.2857
TTG	65.1515	22.3039	36.8534	37.3384	55.5195
TCT	72.8814	24.8182	24.4286	30.99087	46.6364
TCC	43	34.4697	33.9286	43.04288	64.7727
TCA	86	90.9666	85.4701	61.96935	102.5774
TCG	65.1515	139.2857	104.2683	75.58749	91.6071
TAT	48.5876	77.7335	97.3804	52.5269	71.7884
TAC	28.6667	34.125	42.75	30.99087	34.8505
TGT	145.7627	155.467	129.8405	105.0538	38.9522
TGC	86	68.25	57	61.98175	17.1
TGG	65.1515	45.5	57	41.32116	51.506
CTT	145.7627	621.8679	34.2	61.98175	42.75
CTC	86	273	47.5	86.08576	59.375
CTA	86	91	170.9145	123.9387	170.9316
CTG	19.9074	284.375	129.5455	93.91173	46.8066
CCT	145.7627	136.5	28.5	61.98175	51.3
CCC	86	189.5833	39.5833	86.08576	71.25
CCA	86	27.2995	85.4744	30.98932	73.2752
CCG	65.1515	85.3125	104.2683	54.36995	82.2115
CAT	145.7627	88.8383	97.3804	70.03587	106.2332
CAC	86	39	42.75	41.32116	46.6364
CAA	43	30.3333	42.75	41.32116	46.6364
CAG	32.5758	70.3608	52.1341	41.87956	20.9217
CGT	21.5	45.5	21.375	61.98175	73.2857
CGC	58.1081	63.1944	29.6875	86.08576	101.7857
CGA	856.5737	1365	170.8633	123.9387	85.49
CGG	86	273	129.5455	93.91173	74.1329
ATT	48.5876	21	21.375	24.7927	29.2942
ATC	28.6667	29.1667	29.6875	34.4343	28.3739
ATA	573.3333	136.4113	170.8633	123.9015	102.5713
ATG	10.75	27.3	24.4286	61.98175	25.65
ACT	72.8814	24.8182	24.4286	24.7927	51.3
ACC	43	34.4697	33.9286	34.4343	71.25
ACA	86	68.2312	85.4701	123.9015	85.4858
ACG	37.069	119.7368	104.2683	93.91173	64.7727
AAT	36.4407	62.1868	64.9203	52.5269	34.0909
AAC	21.5	27.3	28.5	30.99087	15.6785
AAA	14.3333	39	57	41.32116	30.1763
AAG	44.7917	16.8103	17.1687	20.79924	22.861
AGT	145.7627	310.9339	129.8405	105.0538	146.0706
AGC	86	136.5	57	61.98175	64.125
AGA	86	24.8182	85.5	30.99087	85.5
AGG	65.1515	60.3982	104.2683	54.36995	74.1329
GTT	72.8814	19.5	19	24.7927	46.6364
GTC	43	27.0833	26.3889	34.4343	64.7727
GTA	17.2	136.4045	85.4615	123.9015	102.5774
GTG	53.75	103.4091	104.2683	93.91173	29.1477
GCT	72.8814	24.8182	19	24.7927	17.6897
GCC	43	34.4697	26.3889	34.4343	24.569
GCA	28.6667	54.588	85.4615	61.96625	56.9816
GCG	89.5833	170.625	104.2683	193.693	65.1015
GAT	48.5876	38.8667	48.6902	35.01794	61.5034
GAC	28.6667	17.0625	21.375	20.66058	27
GAA	21.5	19.5	42.75	24.7927	39.4615
GAG	67.1875	42.1296	23.489	22.13634	29.8951
GGT	36.4407	38.8667	48.6902	42.02152	77.9043
GGC	21.5	17.0625	21.375	24.7927	34.2
GGA	86	91	57	41.32116	57
GGG	65.1515	92.2297	87.2449	63.24668	51.9231