Protein Structure and Function Prediction using Kernel Methods.

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Huzefa Rangwala

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

George Karypis, Advisor

August 2008
To my mother.
Acknowledgements

This thesis has been accomplished due to the love, encouragement and support from many people. First and foremost, I would like to dedicate this thesis to my late mother, Fizza Rangwala. My father, Shabbir Rangwala and herself were instrumental in my graduate studies. They made countless sacrifices, supported me always, and showered me with all the needed love and luck towards this thesis. Mom and Dad, thank you.

Whatever I say about my advisor, Professor George Karypis will fall short. Over the past five years, he provided me with the best of learning environments, excellent guidance, support, and morale up-lifting talks. He has given me exposure to many opportunities, and I am very grateful for all that I have learned from him about being a researcher, teacher, mentor and a colleague.

In no short words, I would like to thank the enthusiasm and love that is showered upon me by my fiance, Ameena Bahrainwala. She has played a pivotal role in helping me maintain sanity, and calmness at various times during the thesis and graduate life process.

I owe several thanks to all my wonderful friends all over the world: Avni, Anuj, Shifra, Pallavi, Sid, Hushrav, Tara, Shalaka, Kevin, Ashwini, Mihir, Sameer, Jayant, Preeti, Shivani, Lavanya, and Mithila, as well as colleagues at the Karypis group: Selva, Ying, Michihiro, Kevin, Nikil, Chris, Sigve, Eugene, and Xia. Thank you all.

A special round of thanks to Professors Ravi Janardan, Vipin Kumar and Nevin Young, for their invaluable assistance, feedback and patience at different stages of this thesis. Their criticisms, and advises provide this thesis a solid background. Professor Nevin Young also helped me in establishing a strong research aspect in an inter-disciplinary area that this thesis focuses on.

Last but not the least, I would also like to thank my grandmother and all family members for their well wishes.
Abstract

Proteins play a crucial role in several life processes, and as such knowing the function and structure of proteins has several applications like drug design, and disease understanding. Knowing the three-dimensional structure of proteins is important to advances in biology as this structural information provides insights into how proteins operate.

Experimental methods to determine the structure and function of proteins have not been able to keep up with the high-throughput sequencing technologies. As a result, there is an over abundance of protein sequence information but only a fraction of these proteins have experimentally determined structure, and even a lesser fraction have experimentally determined function. Consequently, researchers are relying on computational methods to bridge this gap between sequence and structure, and between sequence and function.

In this dissertation we study and develop several algorithms that have significantly advanced the state-of-the-art computational methods for structural and functional characterization of proteins using sequence information only. Specifically, our contributions have led to the development of methods for remote homology detection, fold recognition, sequence alignment, prediction of local structure and function of protein, and a novel pairwise local structure similarity score estimated from sequence.

We approach the problem of classifying proteins into functional or structural classes by solving the remote homology detection and fold recognition as a multiclass classification problem. We aim to identify a particular class sharing similar evolutionary characteristics (i.e., remote homologs) and similar overall structural features and shapes (i.e., folds) using sequence information. Our technique is to use support vector machines to train one-versus-rest binary classifiers (one for each class), with the key contributions leading to the development of novel profile-derived kernel functions. These kernel functions use an explicit similarity measure that score a pair of sequences using ungapped alignment of high scoring subsequences or a standard local alignment. Our kernel functions have proven to be the state-of-the-art prediction methods on a common benchmark by a set of independent evaluators.

We also present and study algorithms to solve the k-way multiclass classification problem within the
context of remote homology detection and fold recognition. We show that a low error rate can be achieved by integrating the prediction outputs of the highly accurate one-versus-rest classifiers by learning weight parameters using large margin principles. We are also able to integrate hierarchical information prevalent in these structure databases effectively.

Motivated by the success of our string kernels, we also develop a new approach for sequence alignment that incrementally aligns the best profile-profile scored short subsequences. This algorithm shows comparable accuracy to the standard dynamic-programming based algorithms but also aligns several more residue-pairs classified as reliable aiding in transfer of functional and structural characteristics from known protein. This also helps in producing high quality homology-based modeled proteins.

In this thesis we also introduce a novel local structure similarity score estimated from sequence using a support vector machine framework. This score called $f_{\text{RMSD}}$ is the root mean square deviation between structure fragment pairs and forms the basis of several structure alignment algorithms. Sequence-based $f_{\text{RMSD}}$ estimation has several potential applications, one of which improves the accuracy of sequence alignment algorithms that leads to improved homology-based protein models. A case study presented in this thesis shows this predicted local structure similarity score effective in improving the accuracy of sequence alignments, especially when the identity between sequence pairs is less than 12%. One of the major contributions in prediction of $f_{\text{RMSD}}$ scores has been the development of a new kernel function that better captures pairwise interaction information within sequence and has shown superiority in comparison to the standard radial basis kernel function.

Over the last decade several prediction methods have been developed for determining structural and functional properties of individual protein residues using sequence and sequence-derived information. We also present a generalized protein sequence annotation toolkit (PROSAT) for solving classification or regression problems using support vector machines. The key characteristic of our method is its effective use of window-based information to capture the local environment of a protein sequence residue. This window information is used with several kernel functions available within our framework. We show the effectiveness of using the previously developed normalized second order exponential kernel function and experiment with local window-based information at different levels of granularity. PROSAT has shown comparable and even better performance to the competing custom-tailored methods for a wide range of annotation problems. PROSAT provides practitioners an efficient and easy-to-use tool, the results of which can be used to assist in solving the overarching 3D structure prediction problem.

The algorithms and methods presented here can be used to improve the various steps of a comparative modeling server ranging from template identification, alignment, and quality assessment.
# Contents

Acknowledgments  ii
Abstract  iii
List of Figures  x
List of Tables  xii

1 Introduction 1
  1.1 Key Contributions ........................................... 2
  1.2 Outline ......................................................... 5
  1.3 Related Publications ........................................ 6

2 Background 9
  2.1 Introduction to Protein Structures .......................... 9
      2.1.1 Protein Structure Levels ................................ 9
      2.1.2 Protein Sequence and Structure Databases ................. 12
  2.2 Protein Structure Prediction Methods ....................... 15
      2.2.1 Comparative Modeling ................................... 16
      2.2.2 Fold Prediction (Homologous) .......................... 16
      2.2.3 Fold Prediction (Analogous) ............................ 18
      2.2.4 Ab Initio ................................................. 18
  2.3 Learning from Data ........................................... 19
      2.3.1 Discriminative Learning: Support Vector Machines ....... 19
  2.4 Function and Structure Prediction - Capturing the right signals .... 21
      2.4.1 Protein Sequence and Subsequences .................... 21
5.4.3 PROSAT Prediction Performance ............................................. 106
5.5 MONSTER Web Server .......................................................... 113
  5.5.1 Server Implementation ....................................................... 114
5.6 Conclusions ................................................................. 116

6 Incremental Window-based Sequence Alignment .......................... 117
  6.1 Introduction ................................................................. 117
  6.2 Methods and Algorithms .................................................. 118
    6.2.1 Central Alignment Scheme (CA) .................................... 118
    6.2.2 Subset Alignment Scheme (SA) ..................................... 119
    6.2.3 Central and Subset Alignment Scheme (CSA) .................... 119
    6.2.4 Variable \( \omega \)mer Alignment Scheme .......................... 119
  6.3 Experimental Results ....................................................... 120
    6.3.1 Evaluation Methodologies and Metrics ............................ 120
    6.3.2 Dataset Description .................................................. 122
    6.3.3 Assessment of Incremental Window-based Alignments ........ 123
    6.3.4 Comparison with Earlier Results .................................. 126
  6.4 Conclusions ................................................................. 130

7 Conclusion ............................................................................. 133
  7.1 Thesis Summary ............................................................... 133
  7.2 Future Research Directions ................................................ 136

Bibliography ................................................................. 139

A Notations and Definitions .................................................... 157
# List of Figures

2.1 Overview of the Protein Structure Prediction Problem .......................... 11
2.2 Flowchart for the Comparative Modeling Process. .............................. 17

3.1 SCOP hierarchy tree showing the training and test instances setup for the remote homology detection problem. ......................................................... 48
3.2 SCOP hierarchy tree showing the training and test instances setup for the fold recognition problem. ................................................................. 48
3.3 Comparison of the different SVM-based methods for remote homology detection on the SCOP 1.53 benchmark dataset. The graph plots the total number of families for which a given method exceeds an ROC-50 score threshold. ................. 54
3.4 Comparison of the different SVM-based methods for fold detection on the SCOP 1.53 benchmark dataset. The graph plots the total number of superfamilies for which a given method exceeds an ROC-50 score threshold. ......................... 56

4.1 Scatter plot for test protein-pairs at all levels between estimated and actual $f_{RMSD}$ scores on the aligned dataset. The color coding represents the approximate density of points plotted in a fixed normalized area. .................................................. 86
4.2 Scatter plot for test protein-pairs at all levels between profile-to-profile scores and actual $f_{RMSD}$ scores on the aligned dataset. The color coding represents the approximate density of points plotted in a fixed normalized area. .................................................. 86
4.3 Relative CS Scores on the ce_ref dataset. For each segment we display the range of percent sequence identity, the number of pairs in the segment, and the average CS score of the baseline prof scheme. .......................................................... 92
4.4 Relative CS Scores on the mus_ref dataset. For each segment we display the range of percent sequence identity, the number of pairs in the segment, and the average CS score of the baseline prof scheme.

4.5 Speedup using the Seeding and Sampling Alignment Procedure on the ce_ref dataset.

5.1 Screenshot of MONSTER's INPUT FORM

5.2 Screenshot showing output for disorder prediction as generated by MONSTER.

5.3 Screenshot showing output for secondary structure prediction as generated by MONSTER.

6.1 Cline Score Comparison of SW-PSSM scheme against SA scheme for the 588 alignment pairs in the template-based dataset
List of Tables

2.1 Protein Sequence and Structure Databases. ........................................... 12

3.1 Kernel Methods for Remote Homology Detection and Fold Recognition. .... 30

3.2 Comparative performance of the window-based kernel functions that rely on sequence profiles. ................................................................. 49

3.3 Comparative performance of the window-based kernel functions that rely on BLOSUM62. ............................................................... 50

3.4 Comparative performance of the local alignment-based kernel functions that rely on sequence profiles. ...................................................... 51

3.5 Comparative performance of the local alignment-based kernel functions that rely on BLOSUM45 and BLOSUM62. .............................................. 53

3.6 Comparison against different schemes for the superfamily-level classification problem. 55

3.7 Comparison against different schemes for the fold-level classification problem. 57

3.8 Dataset Statistics. .................................................................................. 58

3.9 Zero-one and Balanced error rates for the remote homology detection problem optimized for the balanced loss function. ...................... 61

3.10 Zero-one and Balanced error rates for the remote homology detection problem optimized for the zero-one loss function. ......................... 62

3.11 Zero-one and Balanced error rates for the fold recognition problem optimized for the balanced loss function. ........................................ 63

3.12 Zero-one and Balanced error rates for the fold recognition problem optimized for the zero-one loss function. ........................................ 64

3.13 Error rates (topn₁, topn₃) for the remote homology detection problem. ....... 65

3.14 Error rates (topn₁, topn₃) for the fold recognition problem. ....................... 66
Introduction

Proteins have a vast influence on the molecular machinery of life. Stunningly complex networks of proteins perform innumerable functions in every living cell. Knowing the function and structure of proteins is crucial for the development of improved drugs, better crops, and even synthetic biofuels. As such, knowledge of protein structure and function leads to crucial advances in life sciences and biology.

With recent advances in large scale sequencing technologies, we have seen an exponential growth in protein sequence information. Protein structures are primarily determined using X-ray crystallography or NMR spectroscopy, but these methods are time consuming, expensive and not feasible for all proteins. The experimental approaches to determine protein function (e.g., gene knockout, targeted mutation, and inhibitions of gene expression studies) are low throughput in nature [146, 120]. As such, our ability to produce sequence information far outpaces the rate at which we can produce structural and functional information.

Consequently, researchers are increasingly rely on computational approaches to extract useful information from experimentally determined three-dimensional structures and functions of proteins. Unraveling the relationship between pure sequence information and three dimensional structure and/or function remains one of the fundamental challenges in molecular biology.

The research work presented in this thesis has been geared towards protein structural and functional bioinformatics, where the goal is to use amino acid sequence information to characterize various aspects of a protein’s structure and function.

Function prediction is generally approached by using inheritance through homology [120], i.e., proteins with similar sequences (common evolutionary ancestry) frequently carry out similar functions. However, several studies [120, 218, 42] have shown that a stronger correlation exists between structure conservation and function i.e., structure implies function, and a higher correlation exists between sequence conservation
1.1 Key Contributions

and structure i.e., sequence implies structure (sequence → structure → function). In this thesis, one of my contributions has been the development of sequence-based classification algorithms for identification of remote homologs (same ancestry) and folds (same overall structural shape). The successful solution of these problems characterizes proteins in terms of both function and structure, globally.

Certain parts of the protein structure may be conserved and interact with other biomolecules (e.g., proteins, DNA, RNA, and small molecules) and perform a particular function due to such interactions. In this thesis, we have developed algorithms for the local structural and functional annotation of protein residues using sequence information. We have also introduced a novel local structure similarity prediction problem. We have also developed sequence alignment algorithms to transfer structural and functional information between protein pairs.

All the methods and algorithms developed in this thesis can be used for performing the different steps for building a typical comparative modeling based protein structure prediction pipeline. Comparative modeling [13, 58] methods determine the structure of a new protein (called target) by (i) selecting one or more suitable structural templates, (ii) performing an accurate sequence-structure alignment, (iii) building the structural models for target using the sequence-structure alignments as reference, (iv) evaluating the quality of the various models to select the most promising, and (v) refining the promising models by taking into the account the placement of side-chains and loop modeling. Chapter 2 provides a comprehensive survey of the other structure prediction approaches, including a brief explanation of the comparative modeling steps.

1.1 Key Contributions

Profile-based Direct String Kernels for Remote Homology Detection and Fold Recognition

Remote homology detection aims to find protein pairs sharing the same evolutionary information, and fold recognition aims to find protein pairs with the same fold shape. Identifying homology relationships using sequence information allows structural and functional characterization of sequence in a global sense using the transfer approach. It also allows for better selection of templates for a give target in the comparative modeling process.

Our work on remote homology detection and fold recognition follows the approach of training one-versus-rest classification models. This methodology is very widely used, and several researchers (a complete review in Section 3.3.3 on page 29) have developed different kernel (or similarity) functions to better capture the similarity between protein sequences. These predictive models are evaluated with respect to how well each binary one-versus-rest classification models can identify proteins that belong to its own class (e.g.,
1.1 Key Contributions

Our key contribution to these problems builds upon previously developed string kernel function [123], and similarity-based kernel functions [177]. In particular, we introduced two novel classes of kernel functions that are derived from explicit sequence similarity measures, and utilize conservation information in the form of automatically constructed sequence profiles. One class determines the similarity between pairs of protein sequences by combining ungapped alignment scores of certain fixed-length subsequences, whereas the other class determines the similarity using standard local alignments with profile-to-profile scoring functions. Empirical results on a standard benchmark by ourselves [155], and an independent study [75] have shown these kernels to remain state-of-the-art three years after development.

Hierarchical Multiclass Classifiers for Remote Homology Detection and Fold Recognition

The second contribution to the problem of remote homology detection and fold recognition was development of \( k \)-way classification approaches by integrating the prediction results of the one-versus-rest profile based classification models [157]. This allows the practitioner to determine the membership of target sequence to a particular superfamily (remote homology detection) or fold (fold recognition) class. This was done using a two-level learning, by using the predictions from the state-of-the-art one-versus-rest classifier and training a second level model that learns the relationships between different classes using structured output spaces.

Within this two-level multiclass prediction framework, we were able to successfully integrate framework, hierarchical classification information of known protein folds, superfamilies, and families by developing binary classifiers for each level of hierarchy and then using a hierarchy-aware loss function to train the final classifier. The resulting two-level hierarchical classifier further improved the performance for the remote homology detection and fold recognition problems.

Local Structure Similarity Prediction

It is well established that predicted local structure information in the form of secondary structure or other structural alphabets plays a integral role in structure prediction methods [10], as well as functional annotation [120]. In particular, the accuracy of target-template alignment algorithms can be greatly improved, especially when there is very little sequence similarity between the target sequence and template, if information about the local structural compatibility between the fragments around each pair of aligned residues is taken into account.

This led to defining a novel problem of \( f_{RMSD} \) prediction [162]. \( f_{RMSD} \) for a pair of residues \((i, j)\) is the root mean square deviation between the pair of structural fragments of fixed length centered at residues \(i\) and \(j\). The novel prediction problem defined in this thesis is the estimation/prediction of \( f_{RMSD} \) scores between residues \(i\) and \(j\) using sequence or sequence-derived information only. Such a setup allows computation
of a local structure similarity (or compatibility) score using sequence information. The $f_{RMSD}$ predicted score allows for several applications in protein structure prediction, including improving sequence-based alignments for comparative modeling and evaluating the quality of alignments.

The $f_{RMSD}$ prediction problem was solved using a support vector regression framework, with the key contributions leading to the development of a normalized second order exponential kernel function. This kernel function was designed to capture pairwise interactions between the residue-pairs in consideration and was empirically shown to perform better at the estimation problem compared to the standard radial basis kernel function. Profile information derived from PSI-BLAST and predicted secondary structure information was encoded using a window around the central residue-pairs for the $f_{RMSD}$ prediction within the kernelized regression framework.

**Sequence Alignment Algorithms**  The accuracy of comparative modeling is highly dependent on the accuracy of the target-template alignment [208, 209]. Also, the standard dynamic programming-based alignment algorithms [138, 188] can be used to align structures, with the key-difference being scoring of structural fragments [113] (using a $f_{RMSD}$-like score), rather than scoring residue-pairs using profiles [134] or composition based substitution matrices like BLOSUM62.

Our contribution [161] has been the use of $f_{RMSD}$ predicted scores using profiles, as a way to score all residue-pairs between sequences to be aligned. Using such a $f_{RMSD}$-derived scoring matrix with a dynamic programming based framework lead to improvement in the accuracy of sequence alignments, especially when the sequence identity between the protein pairs was less than 12%. This work was similar to ProfNet [142] which uses a neural network approach to learn effective scoring functions for alignment. Our $f_{RMSD}$-based alignment method was engineered to be scalable by predicting the $f_{RMSD}$ scores for a sampling of residue-pairs, without much loss in the accuracy of the final alignment.

We also developed an alignment methodology [159], based on selection of high scoring subsequences iteratively. This method was primarily motivated from string kernel theory which was very successful for the problem of remote homology detection [155]. Also, from a comparative modeling standpoint the standard dynamic programming alignment algorithms miss aligning parts of the sequence-pairs that may have a higher score but do not fall within the optimal ordered solution they seek to achieve. Our alignment method, allows alignment of independent of order, shows comparable performance to the dynamic programming based solutions, but leads to alignments with higher quality residue-pairs identified between the target and template sequence from a comparative modeling perspective.
Framework for Local Structure and Function Prediction  We have developed a generalized support vector machine-based toolkit for protein sequence annotation or a residue-level prediction. Specifically, this toolkit called ProSAT uses a localized window-based scheme to encapsulate any sequence-derived features around the central residue (for which the annotation needs to be predicted). The framework [164,3,105,104] was tested on five classification problems: secondary structure prediction, transmembrane helix prediction, protein blocks prediction, ligand-binding site prediction, disorder region prediction, protein-dna binding site prediction and two estimation problems: contact order estimation and solvent accessible surface area estimation.

The key contributions of ProSAT [164] are use of the normalized second order exponential kernel function (introduced earlier [162,104]), and the extraction of information around the central residue using variable width windows. Depending on the problem, only rough information about distant sequence neighbors may be required for accurate predictions. We explore this issue by examining the performance trade-off between fine-grained near-neighbor and coarse-grained distant-neighbor information.

The comparable results of ProSAT in comparison to the custom-designed methods for the different prediction problems, led us to develop a web server called MONSTER [163], Minnesota prOteiN Sequence annoTation servER1. MONSTER provides practitioners an easy to use service for predict essential local structural and functional properties of residues using amino acid sequence information.

1.2 Outline

In this thesis, we develop effective computational methods for protein structure and function prediction. We have been successful at capturing the sequence signals using string kernels for solving various sub-problems associated with structure and function prediction. We believe that our contributions have moved the fields of computational biology as well as machine learning forwards, and as such the thesis is organized in the following chapters:

In Chapter 2, we provide an introduction to protein structures, detailing the different sub-tasks that aid in the protein structure prediction problems, and also surveying the breadth of proteins structure computational methods. We also review some key machine learning concepts, with a stronger emphasis on discriminatory learning using kernel-based techniques.

In Chapter 3, we introduce the problem of remote homology detection and fold recognition. First, we provide a complete survey of the different methods developed in the past two decades for solving this problem.

1MONSTER is available at http://bio.dtc.umn.edu/monster
Next, we introduce our method, and explain its novelty with respect to other methods and also compare its performance to other methods. We also introduce our method to extract hierarchical information from the homolog databases, and finally produce an effective multiclass solution.

In Chapter 4, we describe a novel local, structure-based similarity score that we estimate from sequence information only. Having motivated and explained the problem, we provide our methodology and contributions. We also perform a brief case study on one of the applications, sequence-based alignments for “orphan” sequences.

In Chapter 5, we describe in detail our generalized framework for predicting local structural and functional properties of proteins using any user-derived features. We provide a brief explanation, review and experimental evaluation on seven different residue-wise annotation problems. We also provide a brief summary of our web server for predicting these protein properties.

In Chapter 6 we introduce a alignment scheme developed using some of our successes with string kernel theory in remote homology detection. This alignment method introduces several heuristics to incrementally align subsequences, and shows comparable performance to dynamic-programming solutions, but has an edge in detection of residue-pairs classified to be reliable from a modeling perspective.

In Chapter 7, we present our conclusions, stating the contributions, discussing the limitations and also highlight future applications and directions.

1.3 Related Publications

The work in this thesis has been published in leading journals and conferences in computational biology and bioinformatics. We provide the readers a list of these publications for reference, and are freely available as technical reports at http://www.cs.umn.edu/research/technical_reports.php?page=author&author_id=404.

Related publications for Chapter 3 are:


Related publications for Chapter 4 include:


Related publications for Chapter 5 include:


• Huzefa Rangwala and George Karypis. *MONSTER: Minnesota prOteiN Sequence annotaTion servER*. Bioinformatics. (under review)

Related publications for Chapter 6 include:

Background

The aim of the chapter is to provide the background information regarding protein structures, and machine learning methods. Specifically, the work in this thesis has been directed towards development of computational methods for solving different problems that aid in the overarching 3D protein structure prediction problem. This chapter also provides the different databases that were used for experimentation in this thesis, as well as reviews the different granularities and methods involved in protein structure prediction.

2.1 Introduction to Protein Structures

In this section we introduce the basic definitions and facts about protein structure, the four different levels of protein structure as well as provide details about protein structure databases.

2.1.1 Protein Structure Levels

Within each structural entity called a protein there lies a set of recurring substructures, and within these substructures are smaller substructures still. As an example, consider hemoglobin, the oxygen-carrying molecule in human blood. Hemoglobin has four domains that come together to form its quaternary structure. Each domain assembles (i.e., folds) itself independently to form a tertiary structure. These tertiary structures are comprised of multiple secondary structure elements—in hemoglobin’s case α helices. Alpha helices (and their counterpart β sheets) have elegant repeating patterns dependent upon sequences of amino acids.
Primary Structure

Amino acids form the basic building blocks of proteins. Amino acids consists of a central carbon atom \((C_\alpha)\) attached by an amino \((\text{NH}_2)\), a carboxyl \((\text{COOH})\) group, and a side chain \((R)\) group. The side chain group differentiates the various amino acids. In case of proteins, there are primarily twenty different amino acids that form the building blocks. A protein is a chain of amino acids linked with peptide bonds. Pairs of amino acid form a peptide bond between the amino group of one and the carboxyl group of the other. This poly-peptide chain of amino acids is known as the primary structure or the protein sequence and is shown in Figure 2.1.

Secondary Structure

A sequence of characters representing the secondary structure of a protein describes the general three-dimensional form of local regions. These regions organize themselves into patterns of repeatedly occurring structural fragments independently from the rest of the protein. The most dominant local conformations of polypeptide chains are alpha helices and beta sheets. These local structures have a certain regularity in their form, attributed to the hydrogen bond interactions between various residues. An alpha helix has a coil-like structure, whereas a beta sheet consists of parallel strands of residues. (See Figure 2.1). In addition to regular secondary structure elements, irregular shapes form an important part of the structure and function of proteins. These elements are typically termed coil regions.

Secondary structure can be divided into several types, though usually at least three classes (alpha-helix, coils and beta-sheet) are used. No unique method of assigning residues to a particular secondary structure state from atomic coordinates exists, though the most widely accepted protocol is based on the DSSP algorithm [97]. DSSP uses the following structural classes: H (\(\alpha\)-helix), G (3\(_{10}\)-helix), I (\(\pi\)-helix), E (\(\beta\)-strand), B (isolated \(\beta\)-bridge), T (turn), S (bend), and – (other). Several other secondary structure assignment algorithms use a reduction scheme that converts this eight-state assignment down to three states by assigning H and G to the helix state (H), E and B to a the strand state (E), and the rest (I, T, S, and –) to a coil state (C). This is the format generally used in structure databases.

Tertiary Structure

The tertiary structure of the protein is defined as the global three-dimensional structure, represented by 3D coordinates for each atoms. These tertiary structures are comprised of multiple secondary structure elements, and the three-dimensional structure is a function of the interacting side chains between the different amino
2.1.1 Protein Structure Levels

Figure 2.1. Overview of the Protein Structure Prediction Problem

**Primary Protein Structure**
- sequence of amino acid residues

**Secondary Protein Structure**
- local structures

**Tertiary Protein Structure**
- three dimensional atomic co-ordinates

**Quaternary Protein Structure**
- interaction of several protein chains
acids. Hence, the linear ordering of amino acids forms secondary structure, arranging secondary structures yields tertiary structure.

**Quaternary Structure**

Quaternary structures represent the interaction between multiple polypeptide chains. The interaction between the various chains are due to the non-covalent interactions between the atoms of the different chains. Examples of these interactions include hydrogen bonding, van Der Walls interactions, ionic bonding, and disulphide bonding.

Research in computational structure prediction concerns itself mainly with predicting secondary and tertiary structure from known experimentally determined primary structure or sequence. This is due to the relative ease of determining primary structure and the complexity involved in quaternary structure.

**2.1.2 Protein Sequence and Structure Databases**

The large amount of protein sequence information, experimentally determined structure information, and structural classification information is stored in publicly available databases. In this section we review some of the databases that are used in this thesis, and provide their availability information in Table 2.1.

<table>
<thead>
<tr>
<th>Database</th>
<th>Information</th>
<th>Availability Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>UniProt</td>
<td>Sequence</td>
<td><a href="http://www.pir.uniprot.org/">http://www.pir.uniprot.org/</a></td>
</tr>
<tr>
<td>UniRef</td>
<td>Cluster Sequences</td>
<td><a href="http://www.pir.uniprot.org/">http://www.pir.uniprot.org/</a></td>
</tr>
<tr>
<td>PDB</td>
<td>Structure</td>
<td><a href="http://www.rcsb.org/">http://www.rcsb.org/</a></td>
</tr>
<tr>
<td>SCOP</td>
<td>Structure Classification</td>
<td><a href="http://scop.mrc-lmb.cam.ac.uk/scop/">http://scop.mrc-lmb.cam.ac.uk/scop/</a></td>
</tr>
<tr>
<td>CATH</td>
<td>Structure Classification</td>
<td><a href="http://www.cathdb.info/">http://www.cathdb.info/</a></td>
</tr>
<tr>
<td>FSSP</td>
<td>Structure Classification</td>
<td><a href="http://www.ebi.ac.uk/dali/fssp/">http://www.ebi.ac.uk/dali/fssp/</a></td>
</tr>
<tr>
<td>ASTRAL</td>
<td>Compendium</td>
<td><a href="http://astral.berkeley.edu/">http://astral.berkeley.edu/</a></td>
</tr>
</tbody>
</table>

The databases referred to in this table are used in this thesis.

**Sequence Databases**

The Universal Protein Resource, UniProt [33] is the most comprehensive warehouse containing information about protein sequences and their annotation. It is a database of protein sequences and their function that is
formed by aggregating the information present in the Swiss-Prot, TrEMBL, and PIR databases. The UniPro-
tKB 13.2 version of database (released on April 8, 2008) consists of 5,939,836 protein sequence entries
(Swiss-Prot providing 362,782 entries and TrEMBL providing 5,577,054 entries).

However several proteins have high pairwise sequence identity, and as such lead to redundant information.
The UniProt database [33] creates subset of sequences such that the sequence identity between all pairs
of sequences within the subset is less than a predetermined threshold. In essence, UniProt contains the
UniRef100, UniRef90, and UniRef50 subsets where within each group the sequence identity between pair of
sequences is less than 100%, 90% and 50% respectively.

NCBI also provide a non-redundant (nr) database of protein sequences using sequences from a wide
variety of sources. This database will have pairs of proteins with high sequence identity, but removes all the
duplicates. The NCBI nr version 2.2.18 (released on March 2, 2008) contains 6,441,864 protein sequences.

**Protein Data Bank (PDB)**

The RSCB Protein Data Bank (PDB) [15] stores experimentally determined three-dimensional structure
of biological macromolecules including nucleotides and proteins. As of April 20, 2008 this database consists
of 46,287 protein structures that are determined using X-Ray crystallography (90%), Nuclear Magnetic Res-
onance (NMR) (9%), and other methods like Cryo-electron microscopy (Cryo-EM). These experimental
methods are time consuming, expensive and need the protein to crystallize.

**Structure Classification Databases**

Various methods have been proposed to categorize protein structures. These methods are based on the pair-
wise structural similarity between the protein structures, as well as the topological and geometric arrangement
of atoms and predominant secondary structure like sub-units. SCOP [136], CATH [144], and FSSP [77] are
three widely used structure classification databases. The classification methodology involves breaking a pro-
tein chain or complex into independent folding units called domains, and then classifying these domains into
a set of hierarchical classes sharing similar structural characteristics.

**SCOP Database**

SCOP [136] is a manually curated database that provides a detailed and comprehen-
sive description of the evolutionary and structural relationships between proteins whose structure is known
(present in the PDB).

SCOP classifies proteins structures using visual inspection as well as structural comparison using a suite
of automated tools. The basic unit of classification is generally a domain. SCOP classification is based on
four hierarchical levels that encompass evolutionary and structural relationships [136]. In particular, proteins with clear evolutionary relationship are classified to be within the same family. Generally, protein pairs within the same family have pairwise residue identities greater than 30%. Protein pairs with low sequence identity, but whose structural and functional features imply probably common evolutionary information are classified to be within the same superfamily. Protein pairs with similar major secondary structure elements and topological arrangement of substructures (as well as favoring certain packing geometries) are classified to be within the same fold. Finally, protein pairs having a predominant set of secondary structures (e.g., all α-helices proteins) lie within the same class. The four hierarchical levels i.e., family, superfamily, fold, and class define the structure of the SCOP database.

The SCOP 1.73 version database (released on September 26, 2007) classifies 34,494 PDB entries (97,178 domains) into 1086 unique folds, 1777 unique superfamilies, and 3464 unique families.

**CATH Database** CATH [144], which denotes the Class, Architecture, Topology, and Homologous superfamilys database is a semi-automated protein structure classification database like the SCOP database. CATH uses a consensus of three automated classification techniques to break a chain into domains and classify them in the various structural categories [95]. Domains for proteins that are not resolved by the consensus approach are determined manually. These domains are then classified into the following hierarchical categories using both manual as well as automated methods in conjunction.

The first level membership, class is determined based on the secondary structure composition and packing within the structure. The second level, architecture clusters proteins sharing the same orientation of the secondary structure element but ignoring the connectivity between these sub-structural units. The third level, topology groups protein pairs with a high structure alignment score as determined by the SSAP [198] algorithm, and in essence share both overall shape and connectivity of secondary structures. The fourth level, homologous pairs share a common ancestor and identified by sequence alignment as well as the SSAP structure alignment method. Structures are further classified to be within the same sequence families if they share a high sequence identity.

The CATH 3.1.0 version database (released on January 19, 2007) classifies 30,028 (93885 domains) proteins from the PDB into 40 architecture-level classes, 1084 topology-level classes and 2091 homologous-level classes.

**FSSP Database** The FSSP [77], families of structurally similar proteins is a structure classification database. FSSP uses an automatic classification scheme that employs exhaustive structure-to-structure alignment of proteins using the DALI [76] alignment. FSSP does not provide a hierarchical classification like
the SCOP and CATH databases, but instead employs a hierarchical clustering algorithm using the pairwise structure similarity scores that can be used for the definition of fold classes, however not very accurate.

There have been several studies [70, 40] analyzing the relationship between the SCOP, CATH and FSSP database for representing the fold space for proteins. These major disagreement between the three databases lies in the domain identification step, rather than the domain classification step. A high percentage of agreement exists between the SCOP, CATH and FSSP databases especially at the fold-level with sequence identity greater than 25%. Due to the higher quality of SCOP database, in this thesis we use the SCOP database for our experimental studies. Though the algorithms developed are generally applicable across the CATH and FSSP databases as well.

**ASTRAL Compendium**  The ASTRAL [22, 26, 25] compendium is a set of database and tools used for analysis of protein structures and sequences. This database is partially derived from, and augments the SCOP [136] database. ASTRAL provides accurate linkage between the biological sequence and the reported structure in PDB, and identify the domains within the sequence using SCOP. Since the majority of domain sequences in PDB are very similar to others, ASTRAL tools reduce the redundancy by selecting high-quality representatives. Using the reduced non-redundant set of representation proteins allows for sampling of all the different structures in the PDB. This also removes bias due to over-represented structures. Subsets provided by ASTRAL are based on SCOP domains and use high quality structure files only. Independent subsets of representative proteins are identified using a greedy algorithm with filtering criterion based on pairwise sequence identity determined using BLAST [6], an e-value based threshold, or a SCOP level based filter.

### 2.2 Protein Structure Prediction Methods

One of the biggest goals in structural bioinformatics is the prediction of the three-dimensional (3D) structure of a protein from its one-dimensional (1D) protein sequence. The goal is to be able to determine the shape (known as a fold) that a given amino acid sequence will adopt. The problem is further divided based on whether the sequence will adopt a new fold or bear resemblance to an existing fold (template) in some protein structure database. Fold recognition is easy when the sequence in question has a high degree of sequence similarity to a sequence with known structure [19]. If the two sequences share evolutionary ancestry they are said to be homologous. For such sequence pairs we can build the structure for the query protein by choosing the structure of the known homologous sequence as template. This is known as comparative modeling.

In the case where no good template structure exists for the query, one must attempt to build the protein tertiary structure from scratch. These methods are usually called *ab initio* methods. In a third fold prediction
scenario, there may not necessarily be a good sequence similarity with a known structure, but a structural
template may still exist for the given sequence. To clarify this case, if one were aware of the target struc-
ture then they could extract the template using structure-structure alignments of the target against the entire
structural database. It is important to note that the target and template need not be homologous. These two
cases define the fold prediction (homologous) and fold prediction (analogous) problems during the CASP
competition.

2.2.1 Comparative Modeling

Comparative Modeling or homology modeling is used when there exists a clear relationship between the
sequence of a query protein (unknown structure) to that of a sequence of a known structure. The most basic
approach to structure prediction for such (query) proteins is to perform a pairwise sequence alignment against
each sequence in protein sequence databases. This can be accomplished using sequence alignment algorithms
such as Smith-Waterman [188] or sequence search algorithms (e.g. BLAST [6]). With a good sequence
alignment in hand, the challenge in comparative modeling becomes how to best build a three-dimensional
protein structure for a query protein using the template structure.

The heart of the above process is the selection of a suitable structural template based on sequence pair
similarity. This is followed by the alignment of query sequence to the template structure selected to build
the backbone of the query protein. Finally the entire modeled structure is refined by loop construction and
side-chain modeling. Several comparative modeling methods, more commonly known as modeler programs,
have been developed over the past several years [13, 58] focusing on various parts of the problem.

As seen in the various years of CASP [208, 209], the span of comparative modeling approaches [13, 58]
follow the five basic steps: (i) the selection of one or suitable templates, (ii) the utilization of sensitive
sequence-template alignment algorithms, (iii) building a protein model using the sequence-structure align-
ment as reference, (iv) model quality evaluation, and (v) model refinement. These typical steps for the
comparative modeling process are shown in Figure 2.2.

2.2.2 Fold Prediction (Homologous)

While satisfactory methods exist to detect homologs (proteins that share similar evolutionary ancestry) with
high levels of similarity, accurately detecting homologs at low levels of sequence similarity (remote homol-
ogy detection) remains a challenging problem. Some of the most popular approaches for remote homology
prediction compare a protein with a collection of related proteins using methods such as PSI-BLAST [7],
protein family profiles [63], hidden Markov models (HMMs) [115, 11], and SAM [100]. These schemes
Figure 2.2. Flowchart for the Comparative Modeling Process.
produce models that are generative, in the sense that they build a model for a set of related proteins and then check to see how well this model explains a candidate protein.

In recent years, the performance of remote homology detection has been further improved through the use of methods that explicitly model the differences between the various protein families (classes) by building discriminative models. In particular, a number of different methods have been developed that use support vector machines (SVM) [206] to produce results that are generally superior to those produced by either pairwise sequence comparisons or approaches based on generative models—provided there is sufficient training data. [85, 126, 123, 124, 79, 80, 177, 116].

2.2.3 Fold Prediction (Analogous)

Occasionally a query sequence will have a native fold similar to another known fold in a database, but the two sequences will have no detectable similarity. In many cases the two proteins will lack an evolutionary relationship as well. As the definition of this problem relies on the inability of current methods to detect sequential similarity, the set of proteins falling into this category remains in flux. As new methods continue to improve at finding sequential similarities as a result of increasing database size and better techniques, the number of proteins in question decreases. Techniques to find structures for such query sequences revolve around mounting the query sequence on a series of template structures, in a process known as threading [94, 91, 20]. An objective energy function provides a score for each alignment, and the highest-scoring template is chosen.

Obviously, if the correct template does not exist in the series then the method will not produce an accurate prediction. As a result of this limitation, predicting the structure of proteins in this category is as challenging as prediction of protein targets that are part of the new or rare folds.

2.2.4 Ab Initio

Techniques to predict novel protein structure have come a long way in recent years, though a definitive solution to the problem remains elusive. Research in this area can be roughly divided into fragment assembly [186, 101, 121] and first-principle based approaches, though occasionally the two are combined [169]. The former attempt to assign a fragment with known structure to a section of the unknown query sequence. The latter start with an unfolded conformation, usually surrounded by solvent, and allow simulated physical forces to fold the protein as would normally happen in vivo. Usually, algorithms from either class will use reduced representations of query proteins during initial stages to reduce the overall complexity of the problem.
Even in case of these \textit{ab initio} prediction methods, the state-of-the-art methods \cite{224,225,169} determine several template structures (using the template selection methods used in comparative modeling methods). The final protein is modeled using an assembly of fragments or substructures fitted together using a highly optimized approximate energy and statistics-based potential function.

2.3 Learning from Data

Supervised learning is the task of creating a function that maps a set of inputs to a particular set of outputs by examining labeled training data. This form of learning plays a vital role in several bioinformatics applications including protein structure prediction.

Several books \cite{37,206,31} cover the foundations of supervised learning in detail. The general framework of a supervised learning problem is as follows. Given an input domain $\mathcal{X}$ and output domain $\mathcal{Y}$, learn a function mapping each element of $\mathcal{X}$ to an element in domain $\mathcal{Y}$. In formal terms, given some training data $\left( X_1, Y_1 \right) \ldots \left( X_n, Y_n \right)$, we need to learn a function $h : \mathcal{X} \rightarrow \mathcal{Y}$ mapping each object $X_i \in \mathcal{X}$ to a classification label $Y_i \in \mathcal{Y}$.

It is assumed that there exists an underlying probability distribution $D(X, Y)$ over $\mathcal{X} \times \mathcal{Y}$. This distribution remains unchanged for the training and test samples, but this distribution is unknown. The training and test samples are assumed to be drawn independently, identically distributed from $D(X, Y)$.

Classifiers can be categorized as parametric models and distribution free models. Parametric models attempt to solve the supervised learning problem by explicitly modeling the joint distribution $D(X, Y)$ or conditional distribution $D(Y|X)$ for all $X$. Bayesian and Hidden Markov Models are examples of parametric models. Distribution-free models make no attempt to learn the distribution, but rather choose a function in a selected hypothesis space for classification purposes. Margin based learners like support vector machines \cite{207} are distribution-free classifiers. In the next section, we describe the theoretical foundations for support vector machines highlighting some of the key properties of this supervised learning technique.

2.3.1 Discriminative Learning: Support Vector Machines

Support vector machine (SVM) \cite{207} is a widely used supervised learning methodology based on large margin principles. In the past decade, SVMs have has found countless applications in bioinformatics \cite{181} ranging from gene prediction, RNA secondary structure prediction, and protein-protein interaction. SVMs have been widely accepted for solving problems in fields other than bioinformatics, a small sample of which
include isolated hand-written digit recognition [34], text categorization [87], and face detection in multimedia [183].

SVMs have found such large applicability because of a strong generalization performance i.e., error on the test set after the predictive system is trained. SVMs also have a well founded theory rooted in statistical learning theory [37, 180]. The SVM solutions are globally optimal (convex), unique, sparse and generalize for unseen data. There are several practical optimization solutions [88, 150] available that allow this method to scale to large datasets. However, the key features of SVM is the use of kernels [37] that allow the mapping of nonlinear separable data into a potentially separable hyper-dimensional feature space, and yet tractable computation.

Given a set of positive training examples $S^+$ and a set of negative training examples $S^-$, a support vector machine (SVM) learns a classification function $f(X)$ to maximize the distance or margin between the positive and negative training instances. This function $f(X)$ is of the form

$$f(X) = \sum_{X_i \in S^+} \lambda_i^+ \mathcal{K}(X, X_i) - \sum_{X_i \in S^-} \lambda_i^- \mathcal{K}(X, X_i),$$

(2.1)

where $\lambda_i^+$ and $\lambda_i^-$ are non-negative weights that are computed during training by maximizing a quadratic objective function, and $\mathcal{K}(\cdot, \cdot)$ is called the kernel function, which is computed over all training-set and test-set instances. Given this function, a new instance $X$ is predicted to be positive or negative depending on whether $f(X)$ is positive or negative. In addition, the value of $f(X)$ can be used to obtain a meaningful ranking of a set of instances, as it represents the strength by which they are members of the positive or negative class.

The kernel function, when computed over all pairs of training instances, produces a symmetric matrix. To ensure the validity of a kernel, it is necessary to ensure that it satisfies Mercer’s conditions, which require the pairwise matrix generated by the kernel function to be positive semidefinite. Formally, any function can be used as a kernel so long as for any number $n$, and any possible set of distinct instances $\{X_1, \ldots, X_n\}$, the $n \times n$ Gram matrix defined by $K_{i,j} = \mathcal{K}(X_i, X_j)$ is symmetric positive semidefinite.

A core component of an SVM is the kernel function, which measures the similarity between any pair of examples. Different kernels correspond to different notions of similarity and can lead to discriminative functions with different performance. Designing an efficient and accurate kernel function forms a critical part of SVM-based classification systems. A common approach for deriving a kernel function is to first choose an appropriate feature space, represent each sequence as a vector in that space, and then take the inner product (or a function derived from them) between these vector-space representations as a kernel for the sequences.
One of the major limitations of SVMs, even with its sparse solution is that the complexity during the testing phase is proportional to the number of support vectors, and a complex model with large support vectors will lead to an increased run-time during the prediction phase.

### 2.4 Function and Structure Prediction - Capturing the right signals

In this thesis we have looked at several problems within the larger context of protein structure and function prediction. An ideal solution to the structure prediction problem would correctly predict, from only sequence information, the complete native conformation of a protein in three-dimensional space. Due to the difficulty of developing such a grand solution, decomposing the problem has led to good solutions to smaller parts of the problem.

One of the fundamental steps in building good classification models is selecting features that fit the classification task well. The input domain $X$ for the protein structure prediction problems is the amino acid residues and their properties.

#### 2.4.1 Protein Sequence and Subsequences

A protein sequence $X$ of length $n$ is represented by a sequence of characters $X = (x_1, x_2, \ldots, x_n)$ such that each character corresponds to one of the 20 standard amino acids.

Quite often, the learning and prediction algorithms segment the sequence into short contiguous segments called $w$mers. Specifically, given a sequence $X$ of length $n$ and a user-supplied parameter $w$, the $w$mer at position $i$ of $X$ ($w < i \leq n - w$) is defined to be the $(2w + 1)$-length subsequence of $X$ centered at position $i$. That is, the $w$mer contains $x_i$, the $w$ amino acids before, and the $w$ amino acids after $x_i$. We will denote this subsequence as $w$mer$_X(i)$.

A pair of $w$mers are compared by computing their ungapped alignment scores. Given two sequence $X$ and $Y$, the ungapped alignment score, $wcore(x_i, y_j)$ between a pairs of $w$mers at positions $i$ and $j$ of $X$ and $Y$, respectively can be given by

$$wcore(x_i, y_j) = \sum_{k=-w}^{w} S_{X,Y}(i + k, j + k),$$  \hspace{1cm} (2.2)

where $S_{X,Y}(i + k, j + k)$ is the alignment score between $x_{i+k}$ and $y_{j+k}$ and can be computed using sequence profiles (Section 2.4.2 on the next page or substitution scoring matrices like BLOSUM62.)
2.4.2 Sequence Profiles

It is widely believed that a sequence of amino acids encodes a structural signal [8], and this belief forms the underlying premise of the protein structure prediction problem. Working under this assumption, researchers have tried to encapsulate protein sequence information in various forms for structure analysis. One common way to incorporate more information about the structure of a sequence is to consider similar (and hopefully, therefore, related) sequences. Using multiple sequence alignments one can infer structural information about conserved regions. Many classifiers take as input profiles constructed from such alignments.

PSI-BLAST Profiles

The profile of a protein \( X \) is derived by computing a multiple sequence alignment of \( X \) with a set of sequences \( \{Y_1, \ldots, Y_m\} \) that have a statistically significant sequence similarity with \( X \) (i.e., they are sequence homologs). Profiles are obtained using PSI-BLAST [7] as it combines both steps, is very fast, and has been shown to produce reasonably good results.

The profile of a sequence \( X \) of length \( n \) is represented by two \( n \times 20 \) matrices. The first is its position-specific scoring matrix \( \mathcal{P}_X \) that is computed directly by PSI-BLAST using the scheme described in [7]. The rows of this matrix correspond to the various positions in \( X \) and the columns correspond to the 20 distinct amino acids. The second matrix is its position-specific frequency matrix \( \mathcal{F}_X \) that contains the frequencies used by PSI-BLAST to derive \( \mathcal{P}_X \). These frequencies (also referred to as target frequencies [134]) contain both the sequence-weighted observed frequencies (also referred to as effective frequencies [134]) and the BLOSUM62 [73] derived-pseudocounts [7]. For each row, the frequencies were scaled so that they add up to one. In the cases in which PSI-BLAST could not produce meaningful alignments for certain positions of \( X \), the corresponding rows of the two matrices can be derived from the scores and frequencies of BLOSUM62.

Profile Hidden Markov Models

Profile Hidden Markov Models [49] (HMM) are a complementary approach of producing position-specific scoring matrices using multiple alignment of similar or homologous sequences. A standard profile HMM has three states: (i) match, (ii) insert, and (iii) delete states used for modeling the multiple sequence alignment. These three states model each column of the multiple alignment to produce a probabilistic profile, and compared to the standard PSI-BLAST profile is sensitive in terms of gap modeling, and also solves the position independent assumptions from PSI-BLAST.
Profile-based Sequence Similarity

Many different schemes have been developed for determining the similarity between profiles that combine information from the original sequence, position-specific scoring matrix, or position-specific target and/or effective frequencies [134, 215, 132].

A widely-used profile-to-profile scoring scheme uses a dot product between the profile columns. Specifically, the similarity score between the \(i\)th position of protein’s \(X\) profile and the \(j\)th position of protein’s \(Y\) profile is given by

\[
S_{X,Y}(i, j) = \sum_{k=1}^{20} \mathcal{P}_X(i, k) \mathcal{P}_Y(j, k)
\]  

(2.3)


where \(\mathcal{P}_X(i, k)\) and \(\mathcal{P}_Y(j, k)\) is the value corresponding to the \(k\)th amino acid at the \(i\)th position of \(X\)’s, and \(j\)th position of \(Y\)’s position-specific scoring matrix, respectively. We could use the position-specific frequency matrices as well to compute this profile-based sequence similarity score.

In this thesis we use a scheme that is derived from PICASSO [72, 134]. Specifically, the similarity score between the \(i\)th position of protein’s \(X\) profile, and the \(j\)th position of protein’s \(Y\) profile is given by

\[
S_{X,Y}(i, j) = \sum_{k=1}^{20} \mathcal{F}_X(i, k) \mathcal{P}_Y(j, k) + \sum_{k=1}^{20} \mathcal{F}_Y(j, k) \mathcal{P}_X(i, k),
\]  

(2.4)

where \(\mathcal{F}_X(i, k)\) and \(\mathcal{P}_X(i, k)\) are the values corresponding to the \(k\)th amino acid at the \(i\)th position of \(X\)’s position-specific frequency and score matrices, respectively. \(\mathcal{F}_Y(j, k)\) and \(\mathcal{P}_Y(j, k)\) are defined in a similar fashion.

Equation 2.4 determines the similarity between two profile positions by weighting the position-specific scores of one sequence according to the frequency at which the corresponding amino acid occurs in the second sequence’s profile. Note that by construction, Equation 2.4 leads to a symmetric similarity score. The key difference between the PICASSO function defined here and the corresponding scheme used in [134] (referred to as PICASSO3), is that our measure uses the target frequencies, whereas the scheme of [134] was based on effective frequencies.

The definitions and notations introduced in this chapter will be used consistently throughout the thesis and are also added as part of the appendix A.
Remote Homology Detection and Fold Recognition

3.1 Introduction

Remote homology detection and fold recognition methods play a critical role in characterizing the structural and functional nature of proteins. From a comparative modeling standpoint, classifying a target protein sequence using sequence information into a protein class sharing the same fold or shape, can be used as a precursor for template selection.

While satisfactory methods exist to detect homologs with high levels of similarity, accurately detecting homologs at low levels of sequence similarity (remote homology detection) still remains a challenging problem. Some of the most popular approaches for remote homology prediction compare a protein with a collection of related proteins using methods such as protein family profiles [63], hidden Markov models (HMMs) [115, 11], PSI-BLAST [7], and SAM [100]. These schemes produced models that are generative in the sense that they built a model for a set of related proteins and then check to see how well this model explained a candidate template protein.

In recent years, the performance of remote homology detection has been further improved through the use of methods that explicitly model the differences between the various protein families (classes) and build discriminative models. In particular, a number of different methods have been developed that build these dis-
3.2 Problem Definition

The remote homology detection problem is defined as the identification of protein pairs sharing the same evolutionary ancestry, but having less than 30% sequence identity. Fold recognition is defined as the identification of protein pairs having similar structural topology and shape but no guarantee on the sequence identity. The two problems can be solved by classification of proteins into a particular class of proteins that are remote homologs or folds. Section 2.2.2 on page 16 and Section 2.2.3 on page 18 provide the definitions of remote homology detection and fold recognition within the context of protein structure prediction, respectively.

3.3 Literature Review

Over the past few years, there have been a continual series of advances for methods to identify homologous relationships between protein pairs (both remote homologs as well as folds). More than 5000 research articles are indexed in PUBMED\(^1\) showing relevance to the term "fold recognition". Reviews by Fariselli et.

\(^1\)http://www.ncbi.nlm.nih.gov/sites/entrez
al [54], Wan et. al [213], Lindahl et. al [127], and Jones et. al [90] describe the widely used and developed computational approaches for remote homology detection and fold recognition.

Methods to identify remote homologs and folds can be categorized into two categories i.e., prediction-based approaches and comparative approaches. A further level of categorization of these methods is dependent on the information used i.e., sequence information only, or sequence-structure information also called as threading approaches [196]. However, the most popular and successful methods (as established by the CASP experiments [110, 209, 214]) are consensus-based methods that use a combination of different techniques. Such methods are also known as metaservers.

In this section we review the methods developed for remote homology detection and fold recognition with a particular emphasis on kernel-based methods that use sequence information only. In this thesis we study the development of effective sequence-based kernel techniques for remote homology detection and fold recognition, along with effective incorporation of hierarchical information present within the SCOP (Section 2.1.2) database.

### 3.3.1 Sequence-based Comparative Methods

Pairwise sequence alignments are the most straightforward set of techniques to identify homologous relationships. These methods use dynamic programming based alignments like Needleman-Wunsch [138] and Smith-Waterman algorithms [188], or heuristic based alignments like BLAST [6], and FASTA [148] with different substitution scoring matrices [17], and sophisticated gap modeling [69] for pairwise sequence comparisons.

Over the years, the sensitivity of sequence alignment have been improved by using PSI-BLAST profiles [7] or profile HMMs generated using HMMER [49] or SAM [100] to capture evolutionary information (See Section 2.4.2). Sequence-to-profile scoring schemes used in methods like DIALIGN [135], FPS [65] and profile-to-profile scoring schemes (described in Section 2.4.2) in methods like HHSearch [191], FORTE [201], HMAP [196], and PICASSO [72,134] have shown far superior performance than just sequence-based methods for identification of homologous pairs with sequence identity less than 30%. The review article by Wan et. al [213] lists all the different pairwise sequence-sequence, sequence-profile, and profile-profile alignment methods developed in the past several years.

### 3.3.2 Threading-based or Sequence-Structure Methods

Since the three-dimensional structures of proteins show better conservation during evolution in comparison to sequences, several methods use predicted protein structure information for remote homology detection
3.3.3 Sequence-based Prediction Methods

and fold recognition. In fact there are several cases where protein pairs are structurally similar but share no sequence identity.

Protein threading [94, 174, 220, 91] refers to approaches for structure prediction that "thread" a target sequence through the backbone structures of a collection of template protein structures, and compute a fitness function using the sequence-structure alignment. Such threading methods with varying scoring functions, template structure libraries, and sequence-structure alignment algorithms are used extensively for fold recognition [213].

Similar to the threading approaches are methods that build 3D structural profiles [20, 166, 182, 107], or even use secondary structure (2D profiles) together with primary sequence [174]. These 3D profiles extend the notion of sequence profile providing spatial as well as environmental specific information (in case of FUGUE [182]) for amino acid residues. Since secondary structure of proteins are more likely to be conserved than their sequences, several methods like TOPITS [174] use predicted secondary structure for the target, and true definitions for template as additional information for the alignment. All such methods that use structure information for the template have shown to improve the performance for remote homology detection and fold recognition in comparison to pure sequence-based methods [213, 54].

3.3.3 Sequence-based Prediction Methods

There are several machine-learning based approaches that use Hidden Markov Models (HMMs), Neural Networks (NNs), Support Vector Machines (SVMs) and Conditional Random Field (CRF). In particular, profile HMMs discussed earlier produce a probabilistic representation for profiles [49], but have shown to build accurate generative models using multiple track HMMs [100–102]. SAM-T04 [103] is a method that use neural networks to map the similarities between pairs of proteins, and also evaluate the alignment.

A large set of remote homology detection and fold recognition prediction approaches explicitly model the differences between the various protein families (classes) and build discriminative models. In particular, a number of different methods have been developed that build these discriminative models using support vector machines (SVM) [206] and have shown, provided there is sufficient data for training, to produce results that are in general superior to those produced by either pairwise sequence comparisons or approaches based on generative models [85, 126, 123, 124, 79, 80, 177, 116, 155, 157]. SVMs are primarily used to solve the binary classification problem and as such these methods build one-versus-rest classification models that are evaluated with respect to how well each binary classifier can identify the proteins that belong to its own modeled class. Different SVM-based methods primarily differ in the definition of kernel function (see Section 2.3.1 for specifics on kernel functions) between protein sequences and are reviewed in detail in Section 3.3.3.
Conditional random fields [117] based methods can also build discriminative models like SVMs, and have shown to be used for fold recognition [129].

A comparatively faster approach than using alignments and SVM-based models for remote homology detection is a recurrent neural network method called the Long Short-Term Memory (LSTM) [75]. LSTM automatically determines discriminating patterns and uses correlations between these patterns for the classification models. However, this method does not achieve comparable performance to the state-of-the-art SVM-based methods.

A couple of SVM-based and neural network methods are also used to improve the alignment between pairs of sequences by learning the parameters for gap modeling [195], and scoring matrices [89, 221, 142]. Our work [161, 162] on predicting a local structure similarity score using sequence information falls in this category (discussed in Chapter 4).

Ie et al. [83] developed schemes for combining the outputs of a set of binary SVM-based classifiers for primarily solving the remote homology prediction problem. This was in parallel to our work [157] that used the output of one-versus-rest classifiers in conjunction with hierarchical information from the SCOP [136] database for developing multiclass classification models for remote homology detection and fold recognition.

### Kernel Methods

In this section we review the methods that develop kernel function for use in a SVM-based one-versus-rest classifiers, and are also listed in Table 3.1. The kernel function can be thought of as a measure of similarity between sequences. Different kernels correspond to different notions of similarity and can lead to discriminative functions with different performance.

There are two widely used approaches for deriving kernel functions for protein sequences. The first approach constructs them by first choosing an appropriate vector representation for the sequences, and then taking the inner product (or a function derived from them) between these representations as a kernel for the sequences [85, 123, 124], whereas the second approach derives a valid kernel function from an explicit protein sequence similarity measure that has been shown to be biologically relevant [126, 177, 155, 157].

Our work [155, 157] fall within this space of SVM-based methods that use sequence-derived information in the form of profiles for remote homology detection and fold recognition.

One of the early attempts with such feature-space-based approaches is the SVM-Fisher method [85], in which a profile HMM model is estimated on a set of proteins belonging to the positive class and used to extract a vector representation for each protein. Another approach is the SVM-pairwise scheme [126], which represents each sequence as a vector of pairwise similarities between all sequences in the training
3.3.3 Sequence-based Prediction Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Key Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM-Fisher [85]</td>
<td>profile HMMs for families</td>
</tr>
<tr>
<td>SVM-Pairwise [126]</td>
<td>pairwise sequence similarity with sequences in training</td>
</tr>
<tr>
<td>Spectrum Kernel [123]</td>
<td>exactly identical short subsequences</td>
</tr>
<tr>
<td>Mismatch Kernel [124]</td>
<td>almost identical short subsequences</td>
</tr>
<tr>
<td>eMotif Kernel [14]</td>
<td>common local motifs in eMotif Database</td>
</tr>
<tr>
<td>SVM-Isites [79]</td>
<td>common local motifs in I-sites library</td>
</tr>
<tr>
<td>Cluster Kernel [216]</td>
<td>use of neighborhood sequences</td>
</tr>
<tr>
<td>LA Kernel [177]</td>
<td>direct local alignment using BLOSUM62</td>
</tr>
<tr>
<td>SVM-HMMSTR [80]</td>
<td>HMM information extracted from local motifs</td>
</tr>
<tr>
<td>Profile Kernel [116]</td>
<td>almost identical short subsequences scored using profiles</td>
</tr>
<tr>
<td>SW-PSSM [155]</td>
<td>profile-based direct local alignment</td>
</tr>
<tr>
<td>AF-PSSM [155]</td>
<td>direct profile-based subsequence scoring</td>
</tr>
<tr>
<td>Oligomer Kernel [128]</td>
<td>similar scoring oligomers</td>
</tr>
<tr>
<td>LSA Kernel [46]</td>
<td>LSA based selection of common subsequences</td>
</tr>
<tr>
<td>Genetic Kernel [82]</td>
<td>Genetic programming based motif selection</td>
</tr>
</tbody>
</table>

A relatively simpler feature space that contains all possible short subsequences ranging from 3–8 amino acids (k-mers) is explored in a series of papers (Spectrum kernel [123], Mismatch kernel [124], and Profile kernel [116]). All three of these methods represent a sequence \( X \) as a vector in this feature space and differ on the scheme they employ to actually determine if a particular dimension \( u \) (i.e., kmer) is present (i.e., has a non-zero weight) in \( X \)’s vector or not. The Spectrum kernel considers \( u \) to be present if \( X \) contains \( u \) as a substring, the Mismatch kernel considers \( u \) to be present if \( X \) contains a substring that differs with \( u \) in at most a predefined number of positions (i.e., mismatches), whereas the Profile kernel considers \( u \) to be present if \( X \) contains a substring whose PSSM–based ungapped alignment score with \( u \) is above a user-supplied threshold. An entirely different feature space is explored by the eMotif kernel [14], SVM-Isites [79] and SVM-HMMSTR [80] methods that take advantage of a set of local structural motifs (SVM-Isites) and their relationships (SVM-HMMSTR). Recently, genetic programming was used to determine motif features for developing remote homology detection methods [82], referred to as the GPKernel or Genetic-programming kernel. The Cluster kernel [216] is unique that it computes the features based on the sequence membership in a cluster or neighborhood.

An alternative to measuring pairwise similarity through a dot-product of vector representations is to calculate an explicit protein similarity measure. The recently developed LA-Kernel method [177] represents one
such example of a direct kernel function. This scheme measures the similarity between a pair of protein sequences by taking into account all the optimal local alignment scores with gaps between all of their possible subsequences. The experiments presented in [177] show that this kernel is superior to previously developed schemes that do not take into account sequence profiles and that the overall classification performance improves by taking into account all local alignments.

Our work [155] introduces kernel functions that are derived from explicit sequence similarity measures and utilized sequence profiles.

Another kernel called the oligomer kernel [128] uses similar ideas to our window-based profile [155] kernels and spectrum kernels [123] to compute the similarity between the protein sequences, and another method uses latent semantic analysis [46] for selective use of kmer to develop kernel functions.

### 3.3.4 Consensus Methods

Metaservers or methods that use a consensus of different approaches have found great success in the CASP competition [214, 27]. Pcons [210–212], 3D-Jury [61], Frankenstein’s monster [114], FAMS-ACE [199] and 3D-SHOTGUN [56] are examples of consensus based methods that vary in the methods that are combined and how they are combined. Linear programming [211] as well as neural network based methods [130] have been used to select which of the input predictions are the most reliable, and how they could be combined to produce the best results.

Within the kernel formulation framework, methods have been introduced to combine heterogeneous information using semidefinite programming [118, 119], second order cone programming [9], semi-infinite linear programming [165], and Bayesian framework [39]. Such multiple kernel learning approaches are not strictly consensus approaches but do intend to integrate multiple sources of information. However, such methods are not easily scalable and lead to very complex frameworks.

### 3.3.5 Pairwise Structure-based Methods

Using structure-structure approaches are reliable than the sequence-structure and sequence-sequence methods described earlier [213] in detecting homologs pairs in most cases, but this does depend on the pairs in consideration [54]. The limitation of such approaches is that knowledge of three dimensional structure is needed for the proteins (target as well as templates).

Structure alignment methods like DALI [76], ACE [184], MAMMOTH [131] and MUSTANG [113] are a popular way to identify homologs. There is no clear agreement regarding the best structure alignment method for the detection problems [54]. Recently, the MAMMOTH structure alignment program was used within the
3.4 Methods: Binary SVM Classifiers

In this thesis we introduce two classes of kernel functions for remote homology detection and fold recognition. These kernel functions are used to train one-versus-rest SVM-based classifiers, one per class. The first set of kernel functions, referred to as window-based, determines the similarity between a pair of sequences by using different schemes to combine ungapped alignment scores of certain fixed-length subsequences. The second, referred to as local alignment-based, determines the similarity between a pair of sequences using Smith-Waterman alignments and a position independent AF fine gap model, optimized for the characteristics of the scoring system. Both kernel-classes utilize profiles constructed automatically via PSI-BLAST and employ a profile-to-profile scoring scheme we develop by extending a recently introduced profile alignment method [134].

3.4.1 Window-based Kernels

The first class of profile-based kernel functions that we developed determines the similarity between a pair of sequences by combining the ungapped alignment scores of certain fixed length subsequences (referred to as \( w \)mers). Given a sequence \( X \) of length \( n \) and a user-supplied parameter \( w \), the \( w \)mer at position \( i \) of \( X \) (\( w < i \leq n - w \)) is defined to be the \( (2w + 1) \)-length subsequence of \( X \) centered at position \( i \). That is, the \( w \)mer contains \( x_i \), the \( w \) amino acids before, and the \( w \) amino acids after \( x_i \). We will denote this subsequence as \( w \)mer \( X(i) \).

Note that \( w \)mers are nothing more than the fixed-length windows used extensively in secondary structure prediction and in capturing local sequence information around a particular sequence position. Also, for some of the kernel functions described next, they also correspond to the \( k \)mers used by some of the feature-space derived kernel functions [123, 124, 116].

All Fixed-width \( w \)mers (AF-PSSM).

The first \( w \)mer-based kernel that we developed, referred to as AF-PSSM, can be considered as an extension of the fixed-length spectrum kernel [123] for the cases in which the sequences are represented by their profiles. Recall from Section 3.3.3 that in the fixed-length spectrum kernel, the similarity between two sequences \( X \)
and $Y$ is determined as the number of identical $w$mers that are present in both $X$ and $Y$. However, in the case of sequence profiles, whether two $w$mers match or not can be extended to allow for non-identical $w$mers as long as the alignment of these $w$mers is reasonably good.

Motivated by this observation, the AF-PSSM kernel computes the similarity between a pair of sequences $X$ and $Y$ by adding-up the alignment scores of all possible $w$mers between $X$ and $Y$ that have a positive ungapped alignment score. Specifically, if the ungapped alignment score between two $w$mers at positions $i$ and $j$ of $X$ and $Y$, respectively is denoted by $wscore_{X,Y}(i, j)$, $n$ and $m$ are the lengths of $X$ and $Y$, respectively, and $P_w$ is the set of all possible $w$mer-pairs of $X$ and $Y$ with a positive ungapped alignment score, i.e,

$$P_w = \{(wmer_X(i), wmer_Y(j)) \mid wscore_{X,Y}(i, j) > 0\}, \quad (3.1)$$

for $w + 1 \leq i \leq n - w$ and $w + 1 \leq j \leq m - w$, then the AF-PSSM kernel computes the similarity between $X$ and $Y$ as

$$AF-PSSM_{X,Y}(w) = \sum_{(wmer_X(i), wmer_Y(j)) \in P_w} wscore_{X,Y}(i, j). \quad (3.2)$$

The ungapped alignment score between two $w$mers is computed using the profile-to-profile scoring method of Equation 2.4 on page 23 rewritten below for ease of reference as follows:

$$wscore_{X,Y}(i, j) = \sum_{k=-w}^{w} S_{X,Y}(i + k, j + k). \quad (3.3)$$

where

$$S_{X,Y}(i, j) = \sum_{k=1}^{20} F_{X}(i, k) P_{Y}(j, k) + \sum_{k=1}^{20} F_{Y}(j, k) P_{X}(i, k), \quad (3.4)$$

where $F_{X}(i, k)$ and $P_{X}(i, k)$ are the values corresponding to the $k$th amino acid at the $i$th position of $X$’s position-specific frequency and score matrices, respectively. $F_{Y}(j, k)$ and $P_{Y}(j, k)$ are defined in a similar fashion.

Note that both the AF-PSSM kernel and the Profile kernel [116] determine the similarity between a pair of sequences by considering how all of their fixed-length subsequences are related in view of sequence profiles. However, unlike the feature-space based approach employed by Profile, the AF-PSSM kernels determine the $w$mer-based similarity of two sequences by comparing all of their possible $w$mers directly. This allows AF-PSSM to precisely determine whether two $w$mers are similar or not and provide better quantitative estimates of the degree to which two $w$mers are similar.
Best Fixed-width \textit{w}mer (BF-PSSM).

In determining the similarity between a pair of sequences $X$ and $Y$, the AF-PSSM kernel includes information about all possible \textit{w}mer-level local alignments between them. In light of this observation, it can be thought of as a special case of the LA kernels proposed by Saigo \textit{et al} [177], which compute the similarity between a pair of sequences as the sum of the optimal local alignment scores with gaps between all possible subsequences of $X$ and $Y$. The major differences are that the AF-PSSM kernel is profile-aware, only considers fixed-length \textit{w}mers, and uses ungapped alignments. The results reported in [177] show that taking into account all possible alignments leads to better results.

To see whether or not this is true in the context of the profile-derived \textit{w}mer-based kernels, we developed a scheme that attempts to eliminate this multiplicity by computing the similarity between a pair of sequences based on a subset of the \textit{w}mers used in the AF-PSSM kernel. Specifically, the BF-PSSM kernel selects a subset $\mathcal{P}_w'$ of $\mathcal{P}_w$ (as defined in Equation 6.1 on page 118) such that (i) each position of $X$ and each position of $Y$ is present in at most one \textit{w}mer-pair and (ii) the sum of the \textit{w}scores of the selected pairs is maximized. Given $\mathcal{P}_w'$, the similarity between the pair of sequences is then computed as follows:

$$BF\text{-}PSSM_{X,Y}(w) = \sum_{(\text{\textit{w}mer}(X,i),\text{\textit{w}mer}(Y,j)) \in \mathcal{P}_w'} \text{\textit{w}score}_{X,Y}(i,j).$$

(3.5)

The relation between $\mathcal{P}_w'$ and $\mathcal{P}_w$ can be better understood if the possible \textit{w}mer-pairs in $\mathcal{P}_w$ are viewed as forming an $n \times m$ matrix, whose rows correspond to the positions of $X$, columns to the positions of $Y$, and values correspond to their respective \textit{w}scores. Within this context, $\mathcal{P}_w'$ corresponds to a matching of the rows and columns [147] whose weight is high (bipartite graph matching problem). Since the selection forms a matching, each position of $X$ (or $Y$) contributes a single \textit{w}mer in Equation 3.5, and as such, eliminates the multiplicity present in the AF-PSSM kernel. At the same time, since we are interested in a highly weighted matching, we try to select the best \textit{w}mers for each position.

In our algorithm, we use a greedy algorithm to incrementally construct $\mathcal{P}_w'$ by including the highest weight \textit{w}mers that is not in conflict with the \textit{w}mers already in $\mathcal{P}_w'$. This algorithm terminates when we cannot include in $\mathcal{P}_w'$ any additional \textit{w}mers.

Note that an alternate way of defining $\mathcal{P}_w'$ is to actually look for the maximum weight matching (i.e., the matching whose weight is the highest among all possible matchings). However, the complexity of the underlying bipartite maximum weight matching problem is relatively high ($O(n^3 m + nm^2)$ [147]), and for this reason we use the greedy approach.
Best Variable-width \textit{w}mer (BV-PSSM).

In fixed-width \textit{w}mer-based kernels the width of the \textit{w}mers is fixed for all pairs of sequences and throughout the entire sequence. As a result, if \( w \) is set to a relatively high value, it may fail to identify positive scoring subsequences whose length is smaller than \( 2w + 1 \), whereas if it is set too low, it may fail to reward sequence-pairs that have relative long similar subsequences.

To overcome this problem, we developed a kernel, referred to as BV-PSSM, which is derived from the BF-PSSM kernel but operates with variable width \textit{w}mers. In particular, given a user-supplied width \( w \), it considers the set of all possible \textit{w}mer-pairs whose length ranges from one to \( w \), i.e.,

\[
P_{1\ldots w} = P_1 \cup \ldots \cup P_w,
\]

and among them, it uses the greedy scheme employed by BF-PSSM to select a subset \( P'_{1\ldots w} \) of \textit{w}mer-pairs that form a high weight matching. The similarity between the pair of sequences is then computed as follows:

\[
\text{BV-PSSM}_{X,Y}(w) = \sum_{(\text{mer}(X,i),\text{mer}(Y,j)) \in P'_{1\ldots w}} w \text{score}_{X,Y}(i, j).
\]

Since for each position of \( X \) (and \( Y \)), \( P'_{1\ldots w} \) is constructed by including the highest scoring \textit{w}mer for \( i \) that does not conflict with the previous selections, this scheme can automatically select the highest scoring \textit{w}mer whose length can vary from one up to \( w \); thus, achieving the desired effect.

3.4.2 Local Alignment-based Kernels (SW-PSSM)

The second class of profile-based kernels that we examine compute the similarity between a pair of sequences \( X \) and \( Y \) by finding an optimal alignment between them that optimizes a particular scoring function. There are three general classes of optimal alignment-based schemes that are commonly used to compare protein sequences. These are based on global, local, and global-local (also known as end-space free) alignments [69]. Our experiments with all of these schemes indicate that those based on optimal local alignments (also referred to as Smith-Waterman alignments [188]) tend to produce somewhat better results. For this reason we use this method to derive a profile-based alignment kernel, which is referred to as SW-PSSM.

Given two sequences \( X \) and \( Y \) of lengths \( n \) and \( m \), respectively, the SW-PSSM kernel computes their similarity as the score of the optimal local alignment in which the similarity between two sequence positions is determined using the profile-to-profile scoring scheme of Equation 3.4 on page 33, and a position independent affine gap model. The actual alignment is computed using the \( O(nm) \) dynamic programming algorithm developed by Gotoh [62].
Within this local alignment framework, the similarity score between a pair of sequences depends on the particular values of the affine gap model (i.e., gap-opening ($go$) and gap-extension ($ge$) costs) and the intrinsic characteristics of the profile-to-profile scoring scheme. In order to obtain meaningful local alignments, the scoring scheme that is used should produce alignments whose score must on average be negative with the maximum score being positive [188]. A scoring system whose average score is positive will tend to produce very long alignments, potentially covering segments of low biologically relevant similarity. On the other hand, if the scoring system cannot easily produce alignments with positive scores, then it may fail to identify any non-empty similar subsequences.

To ensure that the SW-PSSM kernel can correctly account for the characteristics of the scoring system, we modify the profile-to-profile scores calculated from Equation 3.4 on page 33 by adding a constant value. This scheme, commonly referred to as zero-shifting [215], ensures that the resulting alignments have scores that on the average are negative while allowing for positive maximum scores. In our scheme, the amount of zero-shifting, denoted by $zs$, is kept fixed for all pairs of sequences, as a limited number of experiments with sequence-pair specific $zs$ values did not produce any better results.

### 3.4.3 From Similarity Measures to Mercer Kernels

Any function can be used as a kernel as long as for any number $n$ and any possible set of distinct sequences $\{X_1, \ldots, X_n\}$, the $n \times n$ Gram matrix defined by $K_{i,j} = K(X_i, X_j)$ is symmetric positive semidefinite. These functions are said to satisfy Mercer’s conditions and are called Mercer kernels, or simply valid kernels.

The similarity based functions described in the previous sections can be used as kernel functions by setting $K(X_i, X_j)$ to be equal to one of AF-PSSM$_{X_i,X_j}$, BF-PSSM$_{X_i,X_j}$, BV-PSSM$_{X_i,X_j}$, or SW-PSSM$_{X_i,X_j}$. However, the resulting functions will not necessarily lead to valid Mercer kernels.

To overcome this problem we used the approach described in [177] to convert a symmetric function defined on the training set instances into positive definite by adding to the diagonal of the training Gram matrix a sufficiently large non-negative constant. Specifically, for each similarity-based training Gram matrix, we found its smallest negative eigenvalue and subtracted it from the diagonal. The resulting kernel matrix is identical to the similarity-based Gram matrix at all positions except those along the main diagonal. We also experimented with the empirical kernel map approach proposed in [180], but we find that the eigenvalue-based scheme produced superior results.
3.4.4 Runtime Complexity

The complexity of using the profile-based alignment or window kernel methods for remote homology detection and fold recognition is dominated by the time needed to compute a profile for the query or target sequence and performing profile-profile alignments between the query and sequences adjudged to be support vectors.

As such, the run-time complexity of computing a profile is $O(N)$ where $N$ is the number of sequences in the nr database, and the time complexity to compute a profile-profile alignment is $O(n^2)$ where $n$ is the average length of the protein sequence. Hence, given $|SV|$ support vectors during classification the run-time complexity would be $O(|SV|n^2)$.

An experiment by Hochreiter et al [75] showed that for a set of 20,000 sequences to be classified using the kernel-based formulation the dominant factor was computing the profile using the nr database. On average each sequence would take 90 seconds, and hence predicting the superfamily or fold using our method would be greater than 500 hours on a single Opteron 1.85 GHz machine.

However, the method proposed by us is certainly scalable and lends itself to an embarassingly parallel solution that could be effectively deployed within a distributed computing environment.

3.5 Methods: Multiclass Classifiers

Having designed highly accurate SVM-based binary classifiers, we studied the best way to combine the predictions of a set of SVM-based binary classifiers to solve the multiclass classification problem and assign a protein sequence to a particular superfamily or fold. We compared the multiclass classification performance between schemes that combine binary classifiers and schemes that directly build an SVM-based multiclass classification model.

In parallel research, Ie et al. [83] developed schemes for combining the outputs of a set of binary SVM-based classifiers for primarily solving the remote homology prediction problem. Specifically borrowing ideas from error-correcting output codes [43, 5, 35], they developed schemes that use a separate learning step to learn how to best scale the outputs of the binary classifiers such that when combined with a scheme that assigns a protein to the class whose corresponding scaled binary SVM prediction is the highest, it achieves the best multiclass prediction performance. In addition, for remote homology prediction in the context of the SCOP [136] hierarchical classification scheme, they also studied the extent to which the use of such hierarchical information can further improve the performance of remote homology prediction. Their experiments showed that these approaches lead to better results than the traditional schemes that use either the maximum
3.5.1 Methods and Algorithms

Given a set of \( m \) training examples \( \{(x_1, y_1), \ldots, (x_m, y_m)\} \), where example \( x_i \) is drawn from a domain \( X \subseteq \mathbb{R}^n \) and each of the label \( y_i \) is an integer from the set \( \mathcal{Y} = \{1, \ldots, K\} \), the goal of the \( K \)-way classification problem is to learn a model that assigns the correct label from the set \( \mathcal{Y} \) to an unseen test example. This can be thought of as learning a function \( f : X \rightarrow \mathcal{Y} \) which maps each instance \( x \) to an element \( y \) of \( \mathcal{Y} \).

**Direct SVM-based \( K \)-way Classifier Solution**

One way of solving the \( K \)-way classification problem using support vector machines is to use one of the many multiclass formulations for SVMs that were developed over the years [67, 68, 217, 4, 36]. These algorithms extend the notions of separating hyperplanes and margins and learn a model that directly separates the different classes.

In this study we evaluate the effectiveness of one of these formulations that was develop by Crammer and Singer [36], which leads to reasonably efficient optimization problems.

This formulation aims to learn a matrix \( W \) of size \( K \times n \) such that the predicted class \( y^* \) for an instance \( x \) is given by

\[
y^* = \arg \max_{i=1}^{K} \{ \langle W_i, x \rangle \},
\]

where \( W_i \) is the \( i^{th} \) row of \( W \) whose dimension is \( n \).

This formulation models each class \( i \) by its own hyperplane (whose normal vector corresponds to the \( i^{th} \) row of the matrix \( W \)) and assigns an example \( x \) to the class \( i \) that maximizes its corresponding hyperplane distance.
3.5.1 Methods and Algorithms

\( W \) itself is learned from the training data following a maximum margin with soft constraints formulation that gives rise to the following optimization problem [36]:

\[
\min \quad \frac{1}{2}\beta W^2 + \sum_{i=1}^{m} \xi_i,
\]

subject to:

\[
\forall i, z \quad \langle W y_i, x_i \rangle + \delta_{yiz} - \langle W z, x_i \rangle \geq 1 - \xi_i \tag{3.9}
\]

where \( \xi_i \geq 0 \) are slack variables, \( \beta > 0 \) is a regularization constant, and \( \delta_{yiz} \) is equal to 1 if \( z = y_i \), and 0 otherwise.

As in the binary support vector machines the dual version of the optimization problem and the resulting classifier depends only on the inner products, which allows us to use any of the recently developed protein string kernels.

**Merging \( K \) One-vs-Rest Binary Classifiers**

An alternate way of solving the \( K \)-way classification problem in the context of SVM is to first build a set of \( K \) one-versus-rest binary classification models \( \{ f_1, f_2, \ldots, f_K \} \), use all of them to predict an instance \( x \), and then based on the predictions of these base classifiers \( \{ f_1(x), f_2(x), \ldots, f_K(x) \} \) assign \( x \) to one of the \( K \) classes [43, 5, 190].

**Max Classifier** A common way of combining the predictions of a set of \( K \) one-versus-rest binary classifiers is to assume that the \( K \) outputs are directly comparable and assign \( x \) to the class that achieved the highest one-versus-rest prediction value; that is, the prediction \( y^* \) for an instance \( x \) is given by

\[
y^* = \arg\max_{i=1}^{K} \{ f_i(x) \}. \tag{3.10}
\]

However, the assumption that the output scores of the different binary classifiers are directly comparable may not be valid, as different classes may be of different sizes and/or less separable from the rest of the dataset- indirectly affecting the nature of the binary model that was learned.

**Cascaded SVM-Learning Approaches** A promising approach that has been explored in combining the outputs of \( K \) binary classification models is to formulate it as a cascaded learning problem in which a second level model is trained on the outputs of the binary classifiers to correctly solve the multiclass classification problem [83, 43, 5].

A simple model that can be learned is the scaling model in which the final prediction for an instance \( x \) is given by

\[
y^* = \arg\max_{i=1}^{K} \{ w_i f_i(x) \}, \tag{3.11}
\]
where \( w_i \) is a factor used to scale the functional output of the \( i^{th} \) classifier, and the set of \( K \) \( w_i \) scaling factors make up the model that is being learned during the second level training phase [83]. We will refer to this scheme as the scaling scheme (S).

An extension to the above scheme is to also incorporate a shift parameter \( s_i \) with each of the classes and learn a model whose prediction is given by

\[
y^* = \arg\max_{i=1}^{K} \{ w_i f_i(x) + s_i \}.
\]

(3.12)

The motivation behind this model is to emulate the expressive power of the z-score approach (i.e., \( w_i = 1/\sigma_i, s_i = -\mu_i/\sigma_i \)) but learn these parameters using a maximum margin framework. We will refer to this as the scale & shift (SS) model.

Finally, a significantly more complex model can be learned by directly applying the Crammer-Singer multiclass formulation on the outputs of the binary classifiers. Specifically, the model corresponds to a \( K \times K \) matrix \( W \) and the final prediction is given by

\[
y^* = \arg\max_{i=1}^{K} \{ \langle W_i, f(x) \rangle \},
\]

(3.13)

where \( f(x) = (f_1(x), f_2(x), \ldots, f_K(x)) \) is the vector containing the \( K \) outputs of the one-versus-rest binary classifiers. We will refer to this as the Crammer-Singer (CS) model.

Comparing the scaling approach to the Crammer-Singer approach we can see that the Crammer-Singer methodology is a more general version and should be able to learn a similar weight vector as the scaling approach. In the scaling approach, there is a single weight value associated with each of the classes. However, the Crammer-Singer approach has a whole weight vector of dimensions equal to the number of features per class. During the training stage, for the Crammer-Singer approach if all the weight values \( w_i, j = 0, \forall i \neq j \) the weight vector will be equivalent to the scaling weight vector. Thus we would expect the Crammer-Singer setting to fit the dataset much better during the training stage.

**Use of Hierarchical Information**

One of the key characteristics of remote homology prediction and fold recognition is that the target classes are naturally organized in a hierarchical fashion. This hierarchical organization is evident in the tree-structured organization of the various known protein structures that is produced by the widely used protein structure classification schemes of SCOP [136], CATH [144] and FSSP [77].

In our study we use the SCOP classification database to define the remote homology prediction and fold recognition problems. SCOP organizes the proteins into four primary levels (class, fold, superfamily,
and family) based on structure and sequence similarity. Within the SCOP classification, the problem of remote homology prediction corresponds to that of predicting the superfamily of a particular protein under the constraint that the protein is not similar to any of its descendant families, whereas the problem of fold recognition corresponds to that of predicting the fold (i.e., second level of hierarchy) under the constraint that the protein is not similar to any of its descendant superfamilies.²

The questions that arise are whether or not and how we can take advantage of the fact that the target classes (either superfamilies or folds) correspond to a level in a hierarchical classification scheme, so as to improve the overall classification performance?

The approach investigated in this study is primarily motivated by the different schemes presented in Section 3.5.1 on page 39 to combine the functional outputs of multiple one-versus-rest binary classifiers. A general way of doing this is to learn a binary one-versus-rest model for each or a subset of the nodes of the hierarchical classification scheme, and then combine these models using an approach similar to the CS-scheme described in Section 3.5.1 on page 38.

For example, assume that we are trying to learn a fold-level multiclass model with $K_f$ folds where $K_s$ is the number of superfamilies that are descendants of these $K_f$ folds, and $K_c$ is the number of classes that are ancestors in the SCOP hierarchy. Then, we will build $K_f + K_s + K_c$ one-versus-rest binary classifiers for each one of the folds, superfamilies, and classes and use them to obtain a vector of $K_f + K_s + K_c$ predictions for a test sequence $x$. Then, using the CS approach, we can learn a second level model $W$ of size $K_f \times (K_f + K_s + K_c)$ and use it to predict the class of $x$ as

$$y^* = \underset{i=1}{\text{argmax}} K \{W_i, f(x)\}, \quad (3.14)$$

where $f(x)$ is a vector of size $K_f + K_s + K_c$ containing the outputs of the binary classifiers.

Note that the output space of this model is still the $K_f$ possible folds, but the model combines information both from the fold-level binary classifiers as well as the binary classifiers for superfamily- and class-level models.

In addition to CS-type models, the hierarchical information can also be used to build simpler models by combining selective subsets of binary classifiers. In our study we experimented with such models by focusing only on the subsets of nodes that are characteristic for each target class and are uniquely determined by it. Specifically, given a target class (i.e., superfamily or fold), the path starting from that node and moving upwards towards the root of the classification hierarchy uniquely identifies a set of nodes corresponding to

²These two constraints are important because if they are violated, then we are actually solving either the family or remote homology prediction problems, respectively
higher level classes containing the target class. For example, if the target class is a superfamily, this path will identify the superfamily itself, its corresponding fold, and its corresponding class in the SCOP hierarchy.

We can construct a second level classification model by combining for each target class the predictions computed by the binary classifiers corresponding to the nodes along these paths. Specifically, for the remote homology recognition problem, let $K_s$ be the number of target superfamilies, $f_i(x)$ the prediction computed by the $i^{th}$ superfamily classifier, $f_{A_f}(x)$ the prediction of the fold classifier corresponding to the $i^{th}$ superfamily, and $f_{A_c}(x)$ the prediction of the class level classifier corresponding to the $i^{th}$ superfamily, then we can express the prediction for instance $x$ as

$$y^* = \arg\max_{i=1}^{K_s} \{w_i f_i(x) + w_{A_f} f_{A_f}(x) + w_{A_c} f_{A_c}(x)\}, \quad (3.15)$$

where $w_i$, $w_{A_f}$ and $w_{A_c}$ are scaling factors learned during training of the second level model. The notation $\wedge$ denotes the predecessor or ancestral relationship operator. In particular for the superfamily $i$, we say that it lies under fold $\wedge_{A_f}$ which is under the class $\wedge_{A_c}$.

Note that the underlying model in Equation 3.15 is essentially an extension of the scaling model of Equation 3.11 on page 39 as it linearly combines the predictions of the binary classifiers of the ancestor nodes.

In a similar fashion, we can use the scale and shift type approach for every node in the hierarchical tree. This allows for an extra shift parameter to be associated with each of the nodes being modeled. Note that similar approaches can be used to define models for fold recognition, where a weight vector is learned to combine the target fold level node along with its specific class level node. A model can also be learned by not considering all the levels along the paths to the root of the tree.

The generic problem of classifying within the context of a hierarchical classification system has recently been studied by the machine learning community and a number of alternative approaches have been developed [203, 194, 176].

### 3.5.2 Theoretical Foundations

In this section we describe how we learn the weight vectors that were introduced for integrating the binary classifiers. We learn the weight vector by a cross-validation set-up on the training set using either the ranking perceptron [32] or structured SVM algorithm [203] both of which work on the principles of large margin discriminative classifiers. We also introduce the notion of loss functions that are optimized for the different integration methods.
Structured Output Spaces

The various models introduced in Sections 3.5.1 and 3.5.1 can be expressed using a unified framework that was recently introduced for learning in structured output spaces [203, 32, 31, 197].

This framework [203] learns a discriminant function $F : \mathcal{X} \times \mathcal{Y} \rightarrow \mathcal{R}$ over input/output pairs from which it derives predictions by maximizing $F$ over the response variable for a specific given input $x$. Hence, the general form of the hypothesis $h$ is

$$
h(x; \theta) = \arg\max_{y \in \mathcal{Y}} \{ F(x, y; \theta) \},
$$

where $\theta$ denotes a parameter vector. Function $F$ is a $\theta$-parameterized family of functions that is designed such that $F(x, y; \theta)$ achieves the maximum value for the correct output $y$. Among the various choices for $F$, if we focus on those that are linear in a combined feature representation of inputs and outputs, $\psi(x, y)$, then Equation 3.16 can be rewritten as [203]:

$$
h(x; \theta) = \arg\max_{y \in \mathcal{Y}} \{ \langle \theta, \psi(x, y) \rangle \}.
$$

The specific form of $\Psi$ depends on the nature of the problem and it is this flexibility that allows us to represent the hypothesis spaces introduced in Sections 3.5.1 and 3.5.1 in terms of Equation 3.17.

For example, consider the simple scaling scheme for the problem of fold recognition (Equation 3.11 on page 39). The input space consists of the $f(x)$ vectors of the binary predictions and the output space $\mathcal{Y}$ consists of the set of $K_f$ folds (labeled from $1 \ldots K_f$). Given an example $x$ belonging to fold $i$ (i.e., $y = i$), the function $\psi(x, y)$ maps the $(x, y)$ pair onto a $K_f$-size vector whose $i$th entry (i.e., the entry corresponding to $x$’s fold) is set to $f_i(x)$ and the remaining entries are set to zero. Then, from Equation 3.17 we have that

$$
h(x; \theta) = \arg\max_{y \in \mathcal{Y}} \{ \langle \theta, \psi(x, y) \rangle \} = \arg\max_{i=1}^{K_f} \{ \theta_i f_i(x) \},
$$

which is similar to Equation 3.11 on page 39 with $\theta$ representing the scaling vector $w$.

Similarly, for the scale & shift approach (Equation 3.12 on page 40), the $\psi(x, y)$ function maps the $(x, y)$ pair onto a feature space of size $2K_f$, where the first $K_f$ dimensions are used to encode the scaling factors and the second $K_f$ dimensions are used to encode the shift factors. Specifically, given an example $x$ belonging to fold $i$, $\psi(x, y)$ maps $(x, y)$ onto the vector whose $i$th entry is $f_i(x)$, its $(2i)^{th}$ entry is one, and the remaining entries are set to zero. Then, from Equation 3.17 we have that

$$
h(x; \theta) = \arg\max_{i=1}^{K_f} \{ \langle \theta, \psi(x, i) \rangle \} = \arg\max_{i=1}^{K_f} \{ \theta_i f_i(x) + \theta_{2i} \},
$$

where $\theta$ represents the scaling vector $w$.
which is equivalent to Equation 3.12 on page 40, with the first half of θ corresponding the scale vector w, and the second half corresponding to the shift vector s.

Finally, in the case of the Cramer-Singer approach, the Ψ(x, y) function maps (x, y) onto a feature space of size $K_f \times K_f$. Specifically, given a sequence x belonging to fold $i$, Ψ(x, y) maps (x, y) onto the vector whose $K_f$ entries starting at $(i - 1)K_f$ are set to $f(x)$ (i.e., the fold prediction outputs) and the remaining $(K_f - 1)K_f$ entries are set to zero. Then, by rewriting Equation 3.17 on the previous page in terms of the above combined input-output representation, we get

$$h(x; \theta) = \max_{i=1}^{K_f} \{\theta, \Psi(x, i)\}$$

$$= \max_{i=1}^{K_f} \left\{ \sum_{j=1}^{K_f} \theta_{i-j+1} f_j(x) \right\}.$$

This is equivalent to Equation 3.13 on page 40, as θ can be viewed as the matrix W with $K_f$ rows and $K_f$ columns. equivalent to Equation 3.13 on page 40, where θ is equal to the matrix W.

**Ranking Perceptron.** One way of learning θ in Equation 3.17 on the preceding page, is to use the recently developed extension to Rosenblatt’s linear perceptron classifier [171], called ranking perceptron [32]. This is an online learning algorithm that iteratively updates θ for each training example that is misclassified according to Equation 3.17 on the previous page. For each misclassified example $x_i$, θ is updated by adding to it a multiple of $(\Psi(x_i, y_i) - \Psi(x_i, y_i^*))$, where $y_i^*$ is given from Equation 3.17 on the preceding page (i.e., the erroneously predicted class for $x_i$). This online learning framework is identical to that used in standard perceptron learning and is known to converge when the examples are linearly separable. However this convergence property does not hold when the examples are not linearly separable.

For our study, we have extended the ranking perceptron algorithm to follow a large margin classification principle whose goal is to learn θ that tries to satisfy the following $m$ constraints:

$$\forall i \ (\theta, \Psi(x_i, y_i)) - (\theta, \Psi(x_i, y_i^*)) \geq \beta \|\theta\|_2,$$

where $y_i$ is $x_i$’s true class and $y_i^* = \arg\max_{y \in Y \setminus y_i} \{\theta, \Psi(x_i, y)\}$. The idea behind these constraints is to force the algorithm to learn a model in which the correct predictions are well-separated from the highest scoring incorrect predictions (i.e., those corresponding to $y_i^*$). The degree of acceptable separation, which corresponds to the required margin, is given by $\beta \|\theta\|_2$, where $\beta$ is a user-specified constant. Note, the margin is expressed in terms of $\theta$’s length to ensure that the separation constraints are invariant to simple scaling transformations.
Algorithm 1 Learning Weight Vectors with the ranking perceptron algorithm

**Input:** $m$: Number of Training Samples.
$(x, y)$: Training Samples.
$\beta$: User constant to control separation constraints.
$\alpha$: Learning rate.

**Output:** $\theta$: Weight Vector.

$\theta \leftarrow 0$

while STOPPING CRITERION = false do
  for $i = 1$ to $m$ do
    $y_i^* = \arg\max_{y \in Y} \langle \theta, \Psi(x_i, y) \rangle$
    if $y_i^* = y_i$ then
      $y_i^* = \arg\max_{y \in Y / y_i} \langle \theta, \Psi(x_i, y) \rangle$
    end if
    if $\langle \theta, \Psi(x_i, y_i) \rangle - \langle \theta, \Psi(x_i, y_i^*) \rangle \leq \beta \| \theta \|_2$ then
      $\theta \leftarrow \theta + \alpha \Psi(x_i, y_i)$
      $\theta \leftarrow \theta - \alpha \Psi(x_i, y_i^*)$
    end if
  end for
end while

Return $\theta$

Algorithm 1 shows our extended ranking perceptron algorithm that uses the constraints of Equation 3.21 on the preceding page to guide its online learning. The key steps in this algorithm are lines 8–10 that update $\theta$ based on the satisfaction/violation of the constraints for each one of the $m$ training instances. Since the ranking perceptron algorithm is not guaranteed to converge when the examples are not linearly separable, Algorithm 1 incorporates an explicit stopping criterion that after each iteration it computes the training error-rate of $\theta$, and terminates when $\theta$’s error rate has not improved in 100 consecutive iterations. The algorithm returns the $\theta$ that achieved the lowest training error rate over all iterations.

**SVM-Struct.** Recently, an efficient way of learning the vector $\theta$ of Equation 3.17 on page 43 has been formulated as a convex optimization problem [203]. In this approach $\theta$ is learned subject to the following $m$ nonlinear constraints

$$\forall i : \max_{y \in Y / y_i} \langle \theta, \Psi(x_i, y) \rangle < \langle \theta, \Psi(x_i, y_i) \rangle.$$  

(3.22)

Note, that these constraints are similar in nature to those used in the ranking perceptron algorithm (Equation 3.21 on the preceding page).

The SVM-Struct [203] algorithm, is an efficient way of solving the above optimization problem in which the $m$ nonlinear inequalities are replaced by $|Y| - 1$ linear inequalities resulting in a total of $m(|Y| - 1)$
linear constraints and θ is learned using the maximum-margin principle leading to the following hard-margin problem [203]:

\[
\min_{\theta} \frac{1}{2} \|\theta\|_2^2 \\
\text{subject to } \langle \theta, \Psi(x_i, y_i) - \Psi(x_i, y) \rangle \geq 1 \quad (3.23)
\]

This hard-margin problem can be converted to a soft-margin equivalent to allow errors in the training set. This is done by introducing a slack variable, ξ, for every nonlinear constraint of Equation 3.22 on the preceding page. The soft-margin problem is expressed as [203]:

\[
\min_{\theta, \xi} \frac{1}{2} \|\theta\|_2^2 + \frac{C}{n} \sum_{i=1}^{n} \xi_i, \\
\text{subject to } \langle \theta, \Psi(x_i, y_i) - \Psi(x_i, y) \rangle \geq 1 - \xi_i \quad (3.24)
\]

The results of classification depend on the value C which is the misclassification cost that determines the trade-off between the generalization capability of the model being learned and maximizing the margin. It needs to be optimized to prevent under-fitting and over-fitting the data during the training phase.

**Loss Functions**

The loss function plays a key role while learning θ both the SVM-struct and ranking perceptron optimizations. Till now, our discussion focused on zero-one loss that assigns a penalty of one for a misclassification and zero for a correct prediction.

However, in cases where the class sizes vary significantly across the different folds, such a zero-one loss function may not be the most appropriate as it may lead to models where the rare class instances are often misclassified. For this reason, an alternate loss function is used, in which penalty for a misclassification is inversely proportional to the class size. This implies that the misclassification of examples belonging to smaller classes weigh higher in terms of the loss. This loss function is referred to as the balanced loss [83].

For the ranking perceptron algorithm (Algorithm 1) the update rules (statements 7 and 8) need to be scaled by the loss function. In case of the SVM-Struct formulation, the balanced loss can be optimized by re-weighting the definition of separation which can be done indirectly by rescaling the slack variables ξi in the constraint inequalities (Equation 3.24).

While using the hierarchical information in the cascaded learning approaches (Section 3.5.1 on page 40) we experimented with a *weighted* loss function where a larger penalty was assigned when the predicted label
did not share the same ancestor compared to the case when the predicted and true class labels shared the same ancestors. This variation did not result in an improvement compared to the zero-one and balanced loss.

3.6 Experimental Results: Binary Classifiers

3.6.1 Dataset Description

We evaluated the classification performance of the profile-based kernels on a set of protein sequences obtained from the SCOP database [136] (See Section 2.1.2 on page 13). We formulated two different classification problems. The first was designed to evaluate the performance of the algorithms for the problem of homology detection when the sequences have low sequence similarities (i.e., the remote homology detection problem), whereas the second was designed to evaluate the extent to which the profile-based kernels can be used to identify the correct fold when there are no apparent sequence similarities (i.e., the fold detection problem).

Remote Homology Detection (Superfamily Detection).

Within the context of the SCOP database, remote homology detection was simulated by formulating it as a superfamily classification problem. The same dataset and classification problems have been used in a number of earlier studies [126, 80, 177] allowing us to perform direct comparisons on the relative performance of the various schemes. The data consisted of 4352 sequences from SCOP version 1.53 extracted from the Astral database, grouped into families and superfamilies. The dataset was processed so that it does not contain any sequence pairs with an $E$-value threshold smaller than $10^{-25}$. For each family, the protein domains within the family were considered positive test examples, and protein domains within the superfamily but outside the family were considered positive training examples. This yielded 54 families with at least 10 positive training examples and 5 positive test examples. Negative examples for the family were chosen from outside of the positive sequences’ fold, and were randomly split into training and test sets in the same ratio as the positive examples. For example, we can visually represent the setup for the remote homology detection problem in terms of the test and training sets for a particular superfamily class (fold a.2.1) in Figure 3.1.

Fold Detection.

Employing the same dataset and overall methodology as in remote homology detection, we simulated fold detection by formulating as a fold classification within the context of SCOP’s hierarchical classification scheme.

---

Figure 3.1. SCOP hierarchy tree showing the training and test instances setup for the remote homology detection problem.

Figure 3.2. SCOP hierarchy tree showing the training and test instances setup for the fold recognition problem.
Table 3.2. Comparative performance of the window-based kernel functions that rely on sequence profiles.

<table>
<thead>
<tr>
<th>Superfamily-level</th>
<th>Fold-level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kernel</td>
<td>ROC</td>
</tr>
<tr>
<td>AF-PSSM (1)</td>
<td>0.965</td>
</tr>
<tr>
<td>AF-PSSM (2)</td>
<td>0.978</td>
</tr>
<tr>
<td>AF-PSSM (3)</td>
<td>0.976</td>
</tr>
<tr>
<td>AF-PSSM (4)</td>
<td>0.956</td>
</tr>
<tr>
<td>BF-PSSM (1)</td>
<td>0.967</td>
</tr>
<tr>
<td>BF-PSSM (2)</td>
<td>0.980</td>
</tr>
<tr>
<td>BF-PSSM (3)</td>
<td>0.977</td>
</tr>
<tr>
<td>BF-PSSM (4)</td>
<td>0.965</td>
</tr>
<tr>
<td>BV-PSSM (1)</td>
<td>0.965</td>
</tr>
<tr>
<td>BV-PSSM (2)</td>
<td>0.973</td>
</tr>
<tr>
<td>BV-PSSM (3)</td>
<td>0.966</td>
</tr>
<tr>
<td>BV-PSSM (4)</td>
<td>0.963</td>
</tr>
</tbody>
</table>

The parameter associated with each kernel corresponds to the width of the \( w_m \) used to define the kernel. The ROC50 of the best performing value of \( w \) for each kernel is shown in bold, and the overall best ROC50 is also underlined.

In this setting, protein domains within the same superfamily were considered to be as positive test examples, and protein domains within the same fold but outside the superfamily were considered as positive training examples. This yielded 23 superfamilies with at least 10 positive training and 5 positive test examples. Negative examples for the superfamily were chosen from outside of the positive sequences’ fold and split equally into test and training sets\(^4\). Since the positive test and training instances were members of different superfamilies within the same fold, this new problem is significantly harder than remote homology detection, as the sequences in the different superfamilies did not have any apparent sequence similarity \([136]\). For example, we can visually represent the setup for the fold recognition problem in terms of the test and training sets for a particular fold class (fold a.2) in Figure 3.2.

---

\(^4\)The classification problem definitions are available at http://bioinfo.cs.umn.edu-supplements/remote-homology/.
### 3.6.2 Evaluation Methodology

We measured the quality of the methods by using the receiver operating characteristic (ROC) scores, the ROC50 scores, and the median rate of false positives (mRFP). The ROC score is the normalized area under a curve that plots true positives against false positives for different possible thresholds for classification [64]. The ROC50 score is the area under the ROC curve up to the first 50 false positives. Finally, the mRFP is the number of false positives scoring as high or better than the median-scoring true positives.

Among these evaluation metrics, due to the fact that the positive class is substantially smaller than the negative class, the ROC50 is considered to be the most useful measure of performance for real-world applications [64]. For this reason, our discussions in the rest of this section will primary focus on ROC50-based comparisons.

<table>
<thead>
<tr>
<th>Kernel</th>
<th>Superfamily-level</th>
<th>Fold-level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ROC</td>
<td>ROC50</td>
</tr>
<tr>
<td>AF-GSM (1)</td>
<td>0.906</td>
<td>0.403</td>
</tr>
<tr>
<td>AF-GSM (2)</td>
<td>0.921</td>
<td>0.461</td>
</tr>
<tr>
<td>AF-GSM (6)</td>
<td>0.926</td>
<td>0.549</td>
</tr>
<tr>
<td>AF-GSM (7)</td>
<td>0.923</td>
<td><strong>0.557</strong></td>
</tr>
<tr>
<td>BF-GSM (1)</td>
<td>0.904</td>
<td>0.488</td>
</tr>
<tr>
<td>BF-GSM (2)</td>
<td>0.923</td>
<td>0.584</td>
</tr>
<tr>
<td>BF-GSM (6)</td>
<td>0.934</td>
<td><strong>0.669</strong></td>
</tr>
<tr>
<td>BF-GSM (7)</td>
<td>0.933</td>
<td>0.665</td>
</tr>
<tr>
<td>BV-GSM (1)</td>
<td>0.906</td>
<td>0.486</td>
</tr>
<tr>
<td>BV-GSM (2)</td>
<td>0.919</td>
<td>0.571</td>
</tr>
<tr>
<td>BV-GSM (6)</td>
<td>0.930</td>
<td><strong>0.666</strong></td>
</tr>
<tr>
<td>BV-GSM (7)</td>
<td>0.929</td>
<td>0.658</td>
</tr>
</tbody>
</table>

The parameter associated with each kernel corresponds to the width of the *w* *mer* used to define the kernel. The ROC50 of the best performing value of *w* for each kernel is shown in bold, and the overall best ROC50 is also underlined.
Table 3.4. Comparative performance of the local alignment-based kernel functions that rely on sequence profiles.

<table>
<thead>
<tr>
<th>Kernel</th>
<th>ROC</th>
<th>ROC50</th>
<th>mRFP</th>
<th>ROC</th>
<th>ROC50</th>
<th>mRFP</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0, 0.125, 0.0</td>
<td>0.972</td>
<td>0.784</td>
<td>0.014</td>
<td>0.867</td>
<td>0.377</td>
<td>0.111</td>
</tr>
<tr>
<td>2.0, 0.250, 0.0</td>
<td>0.972</td>
<td>0.791</td>
<td>0.014</td>
<td>0.873</td>
<td>0.334</td>
<td>0.114</td>
</tr>
<tr>
<td>3.0, 0.125, 0.0</td>
<td>0.971</td>
<td>0.796</td>
<td>0.013</td>
<td>0.860</td>
<td>0.382</td>
<td>0.133</td>
</tr>
<tr>
<td>3.0, 0.250, 0.0</td>
<td>0.960</td>
<td>0.771</td>
<td>0.027</td>
<td>0.852</td>
<td>0.395</td>
<td>0.138</td>
</tr>
<tr>
<td>3.0, 0.750, 1.5</td>
<td>0.982</td>
<td><strong>0.904</strong></td>
<td>0.015</td>
<td>0.933</td>
<td>0.530</td>
<td>0.052</td>
</tr>
<tr>
<td>3.0, 0.750, 2.0</td>
<td>0.979</td>
<td>0.901</td>
<td>0.017</td>
<td>0.936</td>
<td><strong>0.571</strong></td>
<td>0.054</td>
</tr>
</tbody>
</table>

The three parameters for each kernel correspond to the values for the gap opening, gap extension, and zero-shift parameters, respectively. The ROC50 of the best performing scheme is underlined.

3.6.3 Performance of the Window-based Kernels

Table 3.2 on page 49 summarizes the performance achieved by the window-based kernels for the superfamily- and fold-level classification problems across a range of \( w \) values.

These results show that for both the superfamily- and fold-level classification problems, the BV-PSSM kernel achieves the best results, the AF-PSSM kernel tends to perform the worst, whereas the BF-PSSM kernel’s performance is between these two. In the case of superfamily classification, the performance advantage of BV-PSSM over that of BF-PSSM is relatively small, whereas in the case of fold classification, the former has a clear advantage. It achieves an ROC50 value that is on average 16.3% better across the different window lengths.

Comparing the sensitivity of the three schemes based on the value of \( w \), we see that, as expected, their performance is worse for \( w = 1 \), as they only consider \( w \)-mers of length 3, and their performance improves as the value of \( w \) increases. In general, the BV-PSSM kernel performs better for larger windows, whereas the performance of the other kernels tends to degrade more rapidly as the length of the window increases beyond a point. Again, this result is consistent with the design motivation behind the BV-PSSM kernel. Also, the results show that the best value of \( w \) is also dependent on the particular classification problem. For most kernels, the best results for fold classification were obtained with longer windows compared to the superfamily classification.

To see the effect of using sequence profiles, we performed a sequence of classification experiments in which we used the same set of window-based kernel functions, but instead of scoring the similarity be-
between two amino acids using the profile-based scheme (Equation 3.4 on page 33), we used the BLOSUM62 position-independent scoring matrix. The results obtained from these experiments are summarized in Table 3.3 on page 50. In this table, AF-GSM, BF-GSM, and BV-GSM refer to the BLOSUM62-variants of the corresponding window-based kernels (GSM stands for *global scoring matrix*).

These results clearly illustrate the advantage of using sequence profiles in designing kernel functions for both remote homology detection and fold recognition. The profile-based kernel functions achieve significant improvements over their non-profile counterparts across all different kernel functions, classification problems, and metrics.

Comparing the performance of the profile-based kernel functions across the two classification problems, we see that their overall effectiveness in remote homology detection (superfamily-level classification) is much higher than that of fold recognition. This result is in line with the underlying complexity of the classification problem, as the sequence-based signals for fold recognition are extremely weak. This is also manifested by the relative improvement achieved by the profile-based kernel functions over their BLOSUM62-based counterparts (Tables 3.2 and 3.3). For fold recognition, the ROC50 values of the profile-based kernels are higher than those based on BLOSUM62 by a factor of two, whereas for remote homology prediction, the relative ROC50 values are higher by 25%–30%.

In light of the previously published results on LA-Kernels [177], the better results achieved by the BF-PSSM and BV-PSSM kernels over those achieved by the AF-PSSM kernel (which also hold for their corresponding BLOSUM62-based instances of these kernels) were surprising. One explanation for this discrepancy may be the fact that our window-based kernels consider only short-length ungapped alignments, and the results may be different when longer alignments with gaps are considered as well.

### 3.6.4 Performance of the Local Alignment-based Kernels

Table 3.4 on the previous page summarizes the performance achieved by the optimal local alignment-based kernel for the superfamily- and fold-level classification problems across a representative set of values for the gap-opening, gap-extension, and zero-shift parameters. These parameter values were selected after performing a study in which the impact of a large number of value combinations was experimentally studied, and represent some of the best performing combinations. Due to space constraints, this parameter study is not included in this paper.

The most striking observation from these results is the major impact that the zero-shift parameter has to the overall classification performance. For both the superfamily- and fold-level classification problems, the best results are obtained by the SW-PSSM kernel for which the zero shift parameter has been optimized (i.e.,
### Table 3.5. Comparative performance of the local alignment-based kernel functions that rely on BLOSUM45 and BLOSUM62.

<table>
<thead>
<tr>
<th>Kernel</th>
<th>Superfamily-level</th>
<th>Fold-level</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ROC</td>
<td>ROC50</td>
</tr>
<tr>
<td>B45, 3.0, 0.0</td>
<td>0.944</td>
<td>0.686</td>
</tr>
<tr>
<td>B45, 10.0, 0.0</td>
<td>0.940</td>
<td>0.687</td>
</tr>
<tr>
<td>B62, 3.0, 0.0</td>
<td>0.947</td>
<td>0.686</td>
</tr>
<tr>
<td>B62, 10.0, 0.0</td>
<td>0.912</td>
<td>0.599</td>
</tr>
<tr>
<td>B62, 5.0, 0.5</td>
<td>0.948</td>
<td>0.711</td>
</tr>
<tr>
<td>B62, 5.0, 1.0</td>
<td>0.946</td>
<td>0.711</td>
</tr>
</tbody>
</table>

The three parameters for each kernel correspond to the particular global scoring matrix (B45 for BLOSUM45 and B62 for BLOSUM62) and the values for the gap opening and zero-shift parameters, respectively. In all cases, the gap extension cost was set to 1.0. The ROC50 of the best performing scheme is underlined.

Comparing the classification performance of the SW-PSSM kernel against the window-based kernels (Table 3.2 on page 49) we see that the zero-shift optimized SW-PSSM kernel leads to better results than those obtained by the window-based kernels. Moreover, the relative performance advantage of SW-PSSM is higher for fold recognition over the superfamily classification problem. However, if the SW-PSSM kernel does not optimize the zero-shift parameter (i.e., \( z_s = 0.0 \)), the window-based kernels consistently outperform the SW-PSSM kernel. We also performed a limited number of experiments to see the extent to which the performance of the window-based kernels can be improved by explicitly optimizing the zero-shift parameter for them as well. Our results show that these kernels are not significantly affected by such optimizations.

To also see the impact of sequence profiles in the context of kernels derived from optimal local alignments, we evaluated the classification performance of a set of kernel functions that compute the optimal local sequence alignment using the BLOSUM45 and BLOSUM62 amino acid scoring matrices. Table 3.5 shows some of the results obtained with these kernel functions for a representative set of values for the gap opening, gap extension, and zero-shift parameters.

Comparing the results of Table 3.5 with those of Table 3.4 on page 51 we see that, as was the case with the window-based kernels, incorporating profile information leads to significant improvements in the overall classification performance. In addition, these results show that (i) the widely used value for the gap-opening
cost \((g_0 = 10)\) is not necessarily the best for either remote homology detection or fold recognition, and (ii) the classification performance achieved by local alignment kernels derived from the BLOSUM matrices can be further improved by explicitly optimizing the zero-shift parameter as well.

![Graph comparing different SVM-based methods for remote homology detection](image)

**Figure 3.3.** Comparison of the different SVM-based methods for remote homology detection on the SCOP 1.53 benchmark dataset. The graph plots the total number of families for which a given method exceeds an ROC-50 score threshold.

### 3.6.5 Comparisons with Other Schemes

Tables 3.6 and 3.7 compare the performance of the various kernel functions developed in this paper against that achieved by a number of previously developed schemes for the superfamily- and fold-level classification problems, respectively. In the case of the superfamily-level classification problem, the performance is compared against SVM-Fisher [85], SVM-Pairwise [126], and different instances of the LA-Kernel [177], SVM-HMMSTR [80], Mismatch [124], and Profile [116]. In the case of the fold-level classification problem, we only include results for the LA-Kernel and Profile schemes, as these results could be easily obtained from the publicly available data and programs for these schemes.

The results in these tables show that both the window- and local alignment-based kernels derived from sequence profiles (i.e., AF-PSSM, BF-PSSM, BV-PSSM, and SW-PSSM) lead to results that are in general better than those obtained by existing schemes. Comparing the ROC50 values obtained by our schemes, we see that each one of them outperforms all existing schemes. The performance advantage of these kernels is
3.6.5 Comparisons with Other Schemes

Table 3.6. Comparison against different schemes for the superfamily-level classification problem.

<table>
<thead>
<tr>
<th>Kernel</th>
<th>ROC</th>
<th>ROC50</th>
<th>mRFP</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVM-Fisher</td>
<td>0.773</td>
<td>0.250</td>
<td>0.204</td>
</tr>
<tr>
<td>SVM-Pairwise</td>
<td>0.896</td>
<td>0.464</td>
<td>0.084</td>
</tr>
<tr>
<td>LA-eig($\beta = 0.2$)</td>
<td>0.923</td>
<td>0.661</td>
<td>0.064</td>
</tr>
<tr>
<td>LA-eig($\beta = 0.5$)</td>
<td>0.925</td>
<td>0.649</td>
<td>0.054</td>
</tr>
<tr>
<td>SVM-HMMSTR-Ave</td>
<td>–</td>
<td>0.640</td>
<td>0.038</td>
</tr>
<tr>
<td>Mismatch</td>
<td>0.872</td>
<td>0.400</td>
<td>0.084</td>
</tr>
<tr>
<td>Profile(4,6)</td>
<td>0.974</td>
<td>0.756</td>
<td>0.013</td>
</tr>
<tr>
<td>Profile(5,7.5)</td>
<td>0.980</td>
<td>0.794</td>
<td>0.010</td>
</tr>
<tr>
<td>AF-PSSM(2)</td>
<td>0.978</td>
<td>0.816</td>
<td>0.013</td>
</tr>
<tr>
<td>BF-PSSM(2)</td>
<td>0.980</td>
<td>0.854</td>
<td>0.015</td>
</tr>
<tr>
<td>BV-PSSM(2)</td>
<td>0.973</td>
<td>0.855</td>
<td>0.018</td>
</tr>
<tr>
<td>SW-PSSM(3.0,0.750,1.50)</td>
<td>0.982</td>
<td>0.904</td>
<td>0.015</td>
</tr>
<tr>
<td>AF-GSM(6)</td>
<td>0.926</td>
<td>0.549</td>
<td>0.048</td>
</tr>
<tr>
<td>BF-GSM(6)</td>
<td>0.934</td>
<td>0.669</td>
<td>0.053</td>
</tr>
<tr>
<td>BV-GSM(6)</td>
<td>0.930</td>
<td>0.666</td>
<td>0.052</td>
</tr>
<tr>
<td>SW-GSM(B62,5.0,1,0.5)</td>
<td>0.948</td>
<td>0.711</td>
<td>0.039</td>
</tr>
</tbody>
</table>

The SVM-Fisher, SVM-Pairwise, LA-Kernel, and Mismatch results were obtained from [177]. The SVM-HMMSTR results were obtained from [80] and correspond to the best-performing scheme (the authors did not report ROC values). The Profile results were obtained locally by running the publicly available implementation of the scheme obtained from the authors. The ROC50 value of the best performing scheme has been underlined.
3.6.5 Comparisons with Other Schemes

Figure 3.4. Comparison of the different SVM-based methods for fold detection on the SCOP 1.53 benchmark dataset. The graph plots the total number of superfamilies for which a given method exceeds an ROC-50 score threshold.

greater over existing schemes that rely on sequence information alone (e.g., SVM-Pairwise, LA-Kernels), but still remains significant when compared against schemes that either directly take into account profile information (e.g., SVM-Fisher, Profile) or utilize higher-level features derived by analyzing sequence-structure information (e.g., SVM-HMMSTR). Also, the relative advantage of our profile-based methods over existing schemes is greater for the much harder fold-level classification problem over the superfamily-level classification problem. For example, the SW-PSSM scheme achieves ROC50 values that are 13.8% and 81.8% better than the best values achieved by existing schemes for the superfamily- and fold-level classification problems, respectively.

To get a better understanding of the relative performance of the various schemes across the different classes, Figures 3.3 and 3.4 plot the number of classes whose ROC50 was greater than a given threshold that ranges from 0 to 1. Specifically, Figure 3.3 shows the results for the remote homology detection problem, whereas Figure 3.4 shows the results for the fold detection problem. (Note that these figures contain only results for the schemes that we were able to run locally). These results show that our profile-based methods lead to higher ROC50 values for a greater number of classes than either the Profile or LA-kernels, especially for larger ROC50 values (e.g. in the range of 0.6 to 0.95). Also, the SW-PSSM tends to consistently outperform the rest of the profile-based direct kernel methods.

In addition, the results for the BF-GSM, BV-GSM, and SW-GSM kernels that rely on the BLOSUM
3.7 Experimental Results: Multiclass Classifiers

### Table 3.7. Comparison against different schemes for the fold-level classification problem.

<table>
<thead>
<tr>
<th>Kernel</th>
<th>ROC</th>
<th>ROC50</th>
<th>mRFP</th>
</tr>
</thead>
<tbody>
<tr>
<td>LA-eig($\beta = 0.2$)</td>
<td>0.847</td>
<td>0.212</td>
<td>0.129</td>
</tr>
<tr>
<td>LA-eig($\beta = 0.5$)</td>
<td>0.771</td>
<td>0.172</td>
<td>0.193</td>
</tr>
<tr>
<td>Profile(4,6)</td>
<td>0.912</td>
<td>0.305</td>
<td>0.071</td>
</tr>
<tr>
<td>Profile(5,7.5)</td>
<td>0.924</td>
<td>0.314</td>
<td>0.069</td>
</tr>
<tr>
<td>AF-PSSM(4)</td>
<td>0.911</td>
<td>0.374</td>
<td>0.067</td>
</tr>
<tr>
<td>BF-PSSM(4)</td>
<td>0.918</td>
<td>0.414</td>
<td>0.060</td>
</tr>
<tr>
<td>BV-PSSM(4)</td>
<td>0.941</td>
<td>0.481</td>
<td>0.043</td>
</tr>
<tr>
<td>SW-PSSM(3.0,0.750,2.0)</td>
<td>0.936</td>
<td>0.571</td>
<td>0.054</td>
</tr>
<tr>
<td>AF-GSM(6)</td>
<td>0.770</td>
<td>0.197</td>
<td>0.217</td>
</tr>
<tr>
<td>BF-GSM(6)</td>
<td>0.822</td>
<td>0.240</td>
<td>0.157</td>
</tr>
<tr>
<td>BV-GSM(7)</td>
<td>0.845</td>
<td>0.244</td>
<td>0.133</td>
</tr>
<tr>
<td>SW-GSM(B62,5,1.0,0,5)</td>
<td>0.826</td>
<td>0.223</td>
<td>0.176</td>
</tr>
</tbody>
</table>

The results for the LA-Kernel were obtained using the publicly available kernel matrices that are available at the author’s website. The Profile results were obtained locally by running the publicly available implementation of the scheme obtained from the authors. The ROC50 value of the best performing scheme has been underlined.

Scoring matrices show that these kernel functions are capable of producing results that are superior to all of the existing non-profile-based schemes. In particular, the properly optimized SW-GSM scheme is able to achieve significant improvements over the best LA-Kernel-based scheme (7.6% higher ROC50 value) and the best SVM-HMMSTR-based scheme (15.1% higher ROC50 value).

### 3.7 Experimental Results: Multiclass Classifiers

#### 3.7.1 Dataset Description

We assessed the performance of our multiclass classification schemes for solving the remote homology detection and fold recognition on four datasets. The first dataset, referred to as sf95 (superfamily - 95%), was created by Ie et al. [83] to evaluate the performance of the multiclass classification algorithms that they developed (sf95 was designed by Ie et al. [83], whereas the other three datasets, referred to as sf40 (superfamily...
Table 3.8. Dataset Statistics.

<table>
<thead>
<tr>
<th>Statistic</th>
<th>sf95</th>
<th>sf40</th>
<th>fd25</th>
<th>fd40</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASTRAL filtering</td>
<td>95%</td>
<td>40%</td>
<td>25%</td>
<td>40%</td>
</tr>
<tr>
<td>Number of Sequences</td>
<td>2115</td>
<td>1119</td>
<td>1294</td>
<td>1651</td>
</tr>
<tr>
<td>Number of Folds</td>
<td>25</td>
<td>25</td>
<td>25</td>
<td>27</td>
</tr>
<tr>
<td>Number of Superfamilies</td>
<td>47</td>
<td>37</td>
<td>137</td>
<td>158</td>
</tr>
<tr>
<td>Avg. Pairwise Similarity</td>
<td>12.8%</td>
<td>11.5%</td>
<td>11.6%</td>
<td>11.4%</td>
</tr>
<tr>
<td>Avg. Max. Similarity</td>
<td>63.5%</td>
<td>33.9%</td>
<td>32.2%</td>
<td>34.3%</td>
</tr>
<tr>
<td>Avg. Pairwise Similarity (within folds)</td>
<td>25.6%</td>
<td>17.9%</td>
<td>16.7%</td>
<td>17.4%</td>
</tr>
<tr>
<td>Avg. Pairwise Similarity (outside folds)</td>
<td>10.4%</td>
<td>11.03%</td>
<td>11.2%</td>
<td>11.0%</td>
</tr>
</tbody>
</table>

The percent similarity between two sequences is computed by aligning the pair of sequences using SW-GSM with a gap opening of 5.0 and gap extension of 1.0.

“Avg. Pairwise Similarity” is the average of all the pairwise percent identities, “Avg. Max. Similarity” is the average of the maximum pairwise percent identity for each sequence, i.e., it measures the similarity to its most similar sequence. The “Avg. Pairwise Similarity (within folds)” and “Avg. Pairwise Similarity (outside folds)” is the average of the average pairwise percent sequence similarity within the same fold and outside the fold for a given sequence.

Datasets, sf95 and sf40 are designed to evaluate the performance of remote homology detection and were derived by taking only the domains with less than 95% and 40% pairwise sequence identity according to Astral [22], respectively. This set of domains was further reduced by keeping only the domains belonging to folds that (i) contained at least three superfamilies and (ii) one of these superfamilies contained multiple families. For sf95, the resulting dataset contained 2115 domains organized in 25 folds and 47 superfamilies, whereas for sf40, the resulting dataset contained 1119 domains organized in 25 folds and 37 superfamilies.

Datasets, fd25 and fd40 were designed to evaluate the performance of fold recognition and were derived by taking only the domains with less than 25% and 40% pairwise sequence identity, respectively. This set of domains was further reduced by keeping only the domains belonging to folds that (i) contained at least three superfamilies and (ii) at least three of these superfamilies contained more than three domains. For fd25, the resulting dataset contained 1294 domains organized in 25 folds and 137 superfamilies, whereas for fd40, the
resulting dataset contained 1651 domains organized in 27 folds and 158 superfamilies.

### 3.7.2 Evaluation Methodology

The performance of the classification algorithms was assessed using the zero-one (ZE) and the balanced error rate (BE) [83]. The zero-one error rate treats the various classes equally and penalizes each misclassification by one. The balanced error rate accounts for classes of varying size and assigns a lower penalty for misclassifying a test instance belonging to a larger class. The motivation behind balanced error is that larger classes are easier to predict just by chance alone and it rewards a classifier if it can also correctly predict test instances belonging to smaller classes. Following the common practice [83], we set the error of each misclassification to be inversely proportional to its true class size.

In addition, the performance of the various classifiers was evaluated using the previously established approach for evaluating fold recognition methods introduced in [127, 182] that does not penalize for certain types of misclassification. For each test instance, this scheme ranks the various classes from the most to the least likely and a test instance is considered to be correctly classified if its true class is among the highest-ranked $n$ classes (i.e., $\text{top}_n$). The classes in the ranked list that are within the same next higher-level SCOP ancestral class are ignored and do not count towards determining the highest-ranked $n$ classes. That is, in the case of fold recognition, the folds that are part of the same SCOP class as the test instance are ignored and they do not count in determining the $n$ highest-ranked predictions. Similarly, in case of remote homology detection, this scheme ignores the superfamilies that are part of the same SCOP fold as the test sequence. Using a small value for $n$ that is greater than one, this measure assesses the ability of a classifier to find the correct class among its highest ranked predictions, and by penalizing only for the substantially wrong mispredictions (i.e., different SCOP classes or folds), it can assess the severity of the misclassification of the different schemes. In our experiments we computed the error rates for $n = 1$ and $n = 3$.

### Training Methodology

For each dataset we separated the proteins into test and training sets, ensuring that the test set is never used during any parts of the learning phase.

For sf95 and sf40 (fd25 and fd40), the test set is constructed by selecting from each superfamily (fold) all the sequences that are part of one family (superfamily). Thus during training, the dataset does not contain any sequences that are homologous (remote homologous) to the sequences in the test set and thus allows us to evaluate/assess remote homology prediction (fold recognition) performance. This is a standard protocol
for evaluating remote homology detection and fold recognition and has been used in a number of earlier studies [156, 177, 116, 85].

The models for the two-level approach can be learned in three phases by first splitting the training set into two sets, one for learning the first-level model and the other for learning the second-level model. In the first phase, the $k$ one-vs-rest binary classifiers are trained using the training set for the first level. In the second phase, each of these $k$ classifiers are used to predict the training set for the second level. Finally, in the third phase, the second-level model is trained using these predictions. However, due to the limited size of the available training set, we followed a different approach that does not require us to split the training set into two sets. This approach was motivated by the cross-validation methodology and is similar to that used in [83]. This approach first partitions the entire training set into ten equal-size parts. Each part is then being predicted using the $k$ binary classifiers that were trained on the remaining nine parts. At the end of this process, each training instance has been predicted by a set of $k$ binary classifiers, and these prediction outputs serve as training samples for the second-level learning (using the ranking perceptron or the structured SVM algorithm). Having learned the second-level model using the prediction values obtained from the first-level classifiers, we take the entire training set as a whole and retrain the first-level models.

During the evaluation stage, we compute the prediction for our untouched test dataset in two steps. In the first step, we compute the prediction values from the first level model, which are used as features to obtain the final prediction values from the second level model. These predictions are then evaluated using the zero-one and the balanced error.

**Model Selection**

As in all learning methods, the performance of SVM methods depends on the trade-off between the variance and bias in the model. Based on the width of the margin, the model learned may be complex but will not generalize to unseen test samples. The parameter $C$ in SVM-struct and $\beta$ for the ranking perceptron algorithm controls the complexity as well as generalizability, and hence the classification error on test samples.

As such, we perform a model selection or parameter selection step. To perform this exercise fairly, we split our test set into two equal halves of similar distributions, namely sets A and B. Using set A, we vary the controlling parameters and select the best performing model for set A. We use this selected model and compute the accuracy for set B. We repeat the above steps by switching the roles of A and B. The final accuracy results are the average of the two runs. While using the SVM-Struct program we let $C$ take values from the set \{0.0001, 0.001, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1.0, 2.0, 4.0, 8.0, 10.0, 16.0, 32.0, 64.0, 128.0\}. While using the perceptron algorithm we let the margin $\beta$ take values in the set \{0.0001, 0.005, 0.001, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1.0, 2.0, 4.0, 8.0, 10.0, 16.0, 32.0, 64.0, 128.0\}.
3.7.3 Performance Results

The performance of various schemes in terms of zero-one and balanced error is summarized in Tables 3.9 and 3.10 for remote homology detection and in Tables 3.11 and 3.12 for fold recognition. Note, the results in Tables 3.9 and 3.11 are obtained by optimizing the balanced loss function and the results in Tables 3.10 and 3.12 are obtained by optimizing the zero-one loss function. We use four datasets- sf95 and sf40 for remote homology detection, and fd25 and fd40 for fold recognition. We use the standard zero-one and balanced error rates for performance assessment (described in the methods section). The schemes that are included in these tables are the following: (i) the MaxClassifier, (ii) the direct $K$-way classifier, (iii) the two-level learning
Table 3.10. Zero-one and Balanced error rates for the remote homology detection problem optimized for the zero-one loss function.

<table>
<thead>
<tr>
<th></th>
<th>sf95 ZE</th>
<th>sf95 BE</th>
<th>sf40 ZE</th>
<th>sf40 BE</th>
</tr>
</thead>
<tbody>
<tr>
<td>MaxClassifier</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>14.7</td>
<td>30.0</td>
<td>21.0</td>
<td>29.7</td>
</tr>
<tr>
<td>Direct K-way Classifiers</td>
<td>13.5</td>
<td>24.8</td>
<td>20.5</td>
<td>26.5</td>
</tr>
<tr>
<td>Two-Level Approaches Without Hierarchy Information</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ranking Perceptron</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scaling</td>
<td>10.6</td>
<td>18.0</td>
<td>11.7</td>
<td>16.5</td>
</tr>
<tr>
<td>Scale &amp; Shift</td>
<td>13.2</td>
<td>24.5</td>
<td>10.9</td>
<td>13.4</td>
</tr>
<tr>
<td>Crammer Singer</td>
<td>17.0</td>
<td>34.3</td>
<td>14.2</td>
<td>19.4</td>
</tr>
<tr>
<td>SVM-Struct</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scaling</td>
<td>10.7</td>
<td>18.1</td>
<td>13.4</td>
<td>17.3</td>
</tr>
<tr>
<td>Scale &amp; Shift</td>
<td>12.4</td>
<td>23.7</td>
<td>13.4</td>
<td>17.3</td>
</tr>
<tr>
<td>Crammer Singer</td>
<td>12.7</td>
<td>25.2</td>
<td>15.5</td>
<td>19.8</td>
</tr>
<tr>
<td>Two-Level Approaches With Fold-level Nodes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVM-Struct</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scaling</td>
<td>10.4</td>
<td>18.7</td>
<td>14.7</td>
<td>20.0</td>
</tr>
<tr>
<td>Scale &amp; Shift</td>
<td>12.4</td>
<td>23.7</td>
<td>14.7</td>
<td>21.4</td>
</tr>
<tr>
<td>Crammer Singer</td>
<td>13.8</td>
<td>25.0</td>
<td>14.7</td>
<td>19.6</td>
</tr>
<tr>
<td>Two-Level Approaches With Class-level and Fold-level Nodes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVM-Struct</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scaling</td>
<td>10.9</td>
<td>19.1</td>
<td>12.6</td>
<td>17.7</td>
</tr>
<tr>
<td>Scale &amp; Shift</td>
<td>11.2</td>
<td>20.9</td>
<td>13.4</td>
<td>17.8</td>
</tr>
<tr>
<td>Crammer Singer</td>
<td>14.1</td>
<td>27.6</td>
<td>12.6</td>
<td>17.1</td>
</tr>
</tbody>
</table>

ZE and BE denote the zero-one error and balanced error percent rates respectively. The results were obtained by optimizing the zero-one loss function.

approaches based on either the superfamily- or fold-level binary classifiers, and (iv) the two-level learning approaches that also incorporate hierarchical information. For all two-level learning approaches (with and without hierarchical information) these tables show the results obtained by using the scaling (S), scale & shift (SS), and Crammer-Singer (CS) schemes to construct the second-level classifiers.

These tables also show the performance achieved by incorporating different types of hierarchical information in the two-level learning framework. For remote homology prediction they present results that combine information from the ancestor nodes (fold and fold+class), whereas for fold recognition they present results that combine information from ancestor nodes (class), descendant nodes (superfamily), and their combination (superfamily+class).
### Table 3.11. Zero-one and Balanced error rates for the fold recognition problem optimized for the balanced loss function.

<table>
<thead>
<tr>
<th>Model Type</th>
<th>fd25</th>
<th></th>
<th>fd40</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ZE</td>
<td>BE</td>
<td>ZE</td>
<td>BE</td>
</tr>
<tr>
<td><strong>MaxClassifier</strong></td>
<td>42.0</td>
<td>60.3</td>
<td>44.4</td>
<td>64.6</td>
</tr>
<tr>
<td><strong>Direct K-way Classifiers</strong></td>
<td>38.4</td>
<td>52.3</td>
<td>40.4</td>
<td>56.9</td>
</tr>
<tr>
<td><strong>Two-Level Approaches Without Hierarchy Information</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ranking Perceptron</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scaling</td>
<td>39.5</td>
<td>48.7</td>
<td>32.5</td>
<td>48.0</td>
</tr>
<tr>
<td>Scale &amp; Shift</td>
<td>38.8</td>
<td>51.0</td>
<td>29.0</td>
<td>43.0</td>
</tr>
<tr>
<td>Crammer Singer</td>
<td>37.7</td>
<td>49.6</td>
<td>36.0</td>
<td>49.6</td>
</tr>
<tr>
<td>SVM-Struct</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scaling</td>
<td>39.9</td>
<td>52.7</td>
<td>30.8</td>
<td>46.6</td>
</tr>
<tr>
<td>Scale &amp; Shift</td>
<td>39.9</td>
<td>52.5</td>
<td>28.1</td>
<td>42.8</td>
</tr>
<tr>
<td>Crammer Singer</td>
<td>41.3</td>
<td>50.5</td>
<td>31.1</td>
<td>43.3</td>
</tr>
<tr>
<td><strong>Two-Level Approaches With Class-level Nodes</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVM-Struct</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scaling</td>
<td>39.2</td>
<td>52.4</td>
<td>29.9</td>
<td>45.0</td>
</tr>
<tr>
<td>Scale &amp; Shift</td>
<td>38.1</td>
<td>51.6</td>
<td>29.0</td>
<td>41.7</td>
</tr>
<tr>
<td>Crammer Singer</td>
<td>41.7</td>
<td>50.9</td>
<td>29.9</td>
<td>41.7</td>
</tr>
<tr>
<td><strong>Two-Level Approaches With Superfamily-level Nodes</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVM-Struct</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scaling</td>
<td>40.2</td>
<td>52.6</td>
<td>30.5</td>
<td>44.5</td>
</tr>
<tr>
<td>Scale &amp; Shift</td>
<td>40.6</td>
<td>52.7</td>
<td>29.3</td>
<td>42.8</td>
</tr>
<tr>
<td>Crammer Singer</td>
<td>38.8</td>
<td>48.8</td>
<td>31.0</td>
<td>44.9</td>
</tr>
<tr>
<td><strong>Two-Level Approaches With Superfamily-level and Class-level Nodes</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVM-Struct</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scaling</td>
<td>41.0</td>
<td>50.9</td>
<td>33.7</td>
<td>44.6</td>
</tr>
<tr>
<td>Scale &amp; Shift</td>
<td>39.5</td>
<td>51.5</td>
<td>29.3</td>
<td>42.3</td>
</tr>
<tr>
<td>Crammer Singer</td>
<td>40.2</td>
<td>51.9</td>
<td>30.2</td>
<td>42.4</td>
</tr>
</tbody>
</table>

ZE and BE denote the zero-one error and balanced error percent rates respectively. The results were obtained by optimizing the balanced loss function.
### Table 3.12. Zero-one and Balanced error rates for the fold recognition problem optimized for the zero-one loss function.

<table>
<thead>
<tr>
<th>Method</th>
<th>fd25</th>
<th>fd40</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ZE</td>
<td>BE</td>
<td>ZE</td>
<td>BE</td>
</tr>
<tr>
<td>MaxClassifier</td>
<td>42.0</td>
<td>60.3</td>
<td>44.4</td>
<td>64.6</td>
</tr>
<tr>
<td>Direct K-way Classifiers</td>
<td>42.8</td>
<td>59.4</td>
<td>43.0</td>
<td>62.7</td>
</tr>
<tr>
<td>Two-Level Approaches Without Hierarchy Information</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ranking Perceptron</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scaling</td>
<td>39.9</td>
<td>52.9</td>
<td>32.2</td>
<td>50.6</td>
</tr>
<tr>
<td>Scale &amp; Shift</td>
<td>38.4</td>
<td>51.3</td>
<td>27.3</td>
<td>44.8</td>
</tr>
<tr>
<td>Crammer Singer</td>
<td>34.8</td>
<td>48.9</td>
<td>37.7</td>
<td>56.6</td>
</tr>
<tr>
<td>Two-Level Approaches With Class-level Nodes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVM-Struct</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scaling</td>
<td>41.3</td>
<td>55.2</td>
<td>33.7</td>
<td>50.0</td>
</tr>
<tr>
<td>Scale &amp; Shift</td>
<td>41.0</td>
<td>54.3</td>
<td>29.0</td>
<td>46.2</td>
</tr>
<tr>
<td>Crammer Singer</td>
<td>36.6</td>
<td>49.4</td>
<td>32.5</td>
<td>49.6</td>
</tr>
<tr>
<td>Two-Level Approaches With Superfamily-level Nodes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVM-Struct</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scaling</td>
<td>39.5</td>
<td>53.9</td>
<td>31.3</td>
<td>48.8</td>
</tr>
<tr>
<td>Scale &amp; Shift</td>
<td>39.9</td>
<td>53.4</td>
<td>31.3</td>
<td>48.4</td>
</tr>
<tr>
<td>Crammer Singer</td>
<td>37.7</td>
<td>52.1</td>
<td>33.4</td>
<td>51.0</td>
</tr>
<tr>
<td>Two-Level Approaches With Superfamily-level and Class-level Nodes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVM-Struct</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scaling</td>
<td>39.2</td>
<td>52.2</td>
<td>27.3</td>
<td>41.0</td>
</tr>
<tr>
<td>Scale &amp; Shift</td>
<td>39.9</td>
<td>53.9</td>
<td>28.4</td>
<td>44.1</td>
</tr>
<tr>
<td>Crammer Singer</td>
<td>38.8</td>
<td>54.7</td>
<td>31.3</td>
<td>48.0</td>
</tr>
</tbody>
</table>

ZE and BE denote the zero-one error and balanced error percent rates respectively. The results were obtained by optimizing the zero-one loss function.
Table 3.13. Error rates (topn<sub>1</sub>, topn<sub>3</sub>) for the remote homology detection problem.

<table>
<thead>
<tr>
<th></th>
<th>sf95</th>
<th></th>
<th>sf40</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>topn&lt;sub&gt;1&lt;/sub&gt;</td>
<td>topn&lt;sub&gt;3&lt;/sub&gt;</td>
<td>topn&lt;sub&gt;1&lt;/sub&gt;</td>
<td>topn&lt;sub&gt;3&lt;/sub&gt;</td>
</tr>
<tr>
<td>Two-Level Approaches</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Without Hierarchy Info.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVM-Struct</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scaling</td>
<td>7.5</td>
<td>2.6</td>
<td>10.1</td>
<td>3.8</td>
</tr>
<tr>
<td>Scale &amp; Shift</td>
<td>9.0</td>
<td>2.0</td>
<td>10.1</td>
<td>3.4</td>
</tr>
<tr>
<td>Crammer Singer</td>
<td>8.1</td>
<td>1.7</td>
<td>9.2</td>
<td>2.5</td>
</tr>
<tr>
<td>Two-Level Approaches</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>With Fold-level Nodes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVM-Struct</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scaling</td>
<td>4.6</td>
<td>0.9</td>
<td>6.3</td>
<td>1.7</td>
</tr>
<tr>
<td>Scale &amp; Shift</td>
<td>4.0</td>
<td>0.9</td>
<td>5.0</td>
<td>1.7</td>
</tr>
<tr>
<td>Crammer Singer</td>
<td>6.6</td>
<td>2.6</td>
<td>5.0</td>
<td>1.7</td>
</tr>
<tr>
<td>Two-Level Approaches</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>With Fold-level and C.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVM-Struct</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scaling</td>
<td>5.2</td>
<td>1.7</td>
<td>5.5</td>
<td>1.7</td>
</tr>
<tr>
<td>Scale &amp; Shift</td>
<td>5.8</td>
<td>2.3</td>
<td>4.2</td>
<td>2.1</td>
</tr>
<tr>
<td>Crammer Singer</td>
<td>6.6</td>
<td>2.0</td>
<td>5.0</td>
<td>1.7</td>
</tr>
</tbody>
</table>

The results shown in the table are optimized for the balanced loss function.

Zero-one Versus Balanced Loss Function

The direct K-way and the two-level learning approaches can be trained using either the zero-one or the balanced loss functions (the MaxClassifier scheme does not explicitly optimize a loss function). The zero-one loss function achieved consistently worse results than those achieved by the balanced loss function for both the remote homology detection (comparing Tables 3.9 and 3.10) and the fold recognition problem (comparing Tables 3.11 and 3.12). On the average, the zero-one and balanced error rates of the zero-one loss was 10% and 20% higher than balanced loss, respectively. For this reason, our evaluation of the various schemes focuses only on the results obtained by optimizing the balanced loss function (shown in Tables 3.9 and 3.11).

Performance of Direct K-way Classifier

Comparing the direct K-way classifiers against the MaxClassifier approach we see that the error rates achieved by the direct approach are smaller for both the remote homology detection and fold recognition problems. In many cases these improvements are substantial. For example, the direct K-way classifier achieves a 10.9% zero-one error rate for sf40 compared to a corresponding error rate of 21.0% achieved by MaxClassifier. In addition, unlike the common belief that learning SVM-based direct multiclass classifiers is computationally very expensive, we found that the Crammer-Singer formulation that we used, required time that is comparable
Table 3.14. Error rates (topn\(_1\), topn\(_3\)) for the fold recognition problem.

<table>
<thead>
<tr>
<th></th>
<th>fd25</th>
<th>fd40</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>topn(_1)</td>
<td>topn(_3)</td>
</tr>
<tr>
<td><strong>Two-Level Approaches Without Hierarchy Information</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVM-Struct</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scaling</td>
<td>38.5</td>
<td>24.5</td>
</tr>
<tr>
<td>Scale &amp; Shift</td>
<td>37.4</td>
<td>24.8</td>
</tr>
<tr>
<td>Crammer Singer</td>
<td>36.3</td>
<td>22.7</td>
</tr>
<tr>
<td><strong>Two-Level Approaches With Class-level Nodes</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVM-Struct</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scaling</td>
<td>36.7</td>
<td>21.9</td>
</tr>
<tr>
<td>Scale &amp; Shift</td>
<td>36.3</td>
<td>21.6</td>
</tr>
<tr>
<td>Crammer Singer</td>
<td>37.1</td>
<td>22.3</td>
</tr>
<tr>
<td><strong>Two-Level Approaches With Superfamily-level Nodes</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVM-Struct</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scaling</td>
<td>39.9</td>
<td>24.5</td>
</tr>
<tr>
<td>Scale &amp; Shift</td>
<td>39.6</td>
<td>23.4</td>
</tr>
<tr>
<td>Crammer Singer</td>
<td>40.6</td>
<td>27.3</td>
</tr>
<tr>
<td><strong>Two-Level Approaches With Superfamily-level and Class-level Nodes</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVM-Struct</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scaling</td>
<td>39.2</td>
<td>25.2</td>
</tr>
<tr>
<td>Scale &amp; Shift</td>
<td>38.5</td>
<td>23.0</td>
</tr>
<tr>
<td>Crammer Singer</td>
<td>37.1</td>
<td>23.7</td>
</tr>
</tbody>
</table>

The results shown in the table are optimized for the balanced loss function.

to that required for building the various binary classifiers used by the MaxClassifier approach.

**Non-Hierarchical Two-Level Learning Approaches**

Analyzing the performance of the various two-level classifiers that do not use hierarchical information we see that the scaling (S) and scale & shift (SS) schemes achieve better error rates than those achieved by the Crammer-Singer (CS) scheme. Since the hypothesis space of the CS scheme is a superset of the hypothesis spaces of the S and SS schemes, we found this result to be surprising at first. However, in analyzing the characteristics of the models that were learned we noticed that the reason for this performance difference is the fact that the CS scheme tended to overfit the data. This was evident by the fact that the CS scheme had lower error rates on the training set than either the S or SS schemes (results not reported here). Since CS’s linear model has more parameters than the other two schemes, due to the fact that the size of the training set for all three of them is the same and rather limited, such overfitting can easily occur. Note that these observations regarding these three approaches hold for the two-level approaches that use hierarchical information as well.
### Table 3.15. Comparative results for the remote homology detection problem on dataset sf95.

<table>
<thead>
<tr>
<th></th>
<th>Ie et al [83]</th>
<th>Scaling Model</th>
<th>Best Model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ZE</td>
<td>BE</td>
<td>ZE</td>
</tr>
<tr>
<td>Without Hierarchy Information</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ranking Perceptron</td>
<td>21.8</td>
<td>36.7</td>
<td>9.3</td>
</tr>
<tr>
<td>SVM-Struct</td>
<td>20.7</td>
<td>37.6</td>
<td>9.0</td>
</tr>
<tr>
<td>With Fold-level Nodes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SVM-Struct</td>
<td>20.4</td>
<td>37.5</td>
<td>11.2</td>
</tr>
</tbody>
</table>

The results for Ie et al were obtained from the supplementary website for the work [83], and represent the results obtained using the simple scaling model in their implementation. The results labeled “Scaling Model” correspond to the performance achieved by our two-level classifiers using the simple scaling model, whereas the results labeled “Best Model” correspond to the best performance achieved among the simple scaling, scaling & shift, and Cramer-Singer models. Both of these results were obtained from Table 3.9 on page 61. All results were obtained by optimizing the balanced loss function. ZE and BE denote the zero-one error and balanced error percent rates respectively.

Comparing the performance of the S and SS schemes against that of the direct $K$-way classifier we see that the two-level schemes are somewhat worse for sf40 and fd25 and considerably better for sf95 and fd40. In addition, they are consistently and substantially better than the MaxClassifier approach across all four datasets.

#### SVM-Struct versus Ranking Perceptron

For the two-level approaches that do not use hierarchical information, Tables 3.9 and 3.11 show the error-rates achieved by both the ranking perceptron and the SVM-struct algorithms. From these results we can see that for the S and SS schemes, the performance achieved by the ranking perceptron are comparable to and in some cases slightly better than those achieved by the SVM-struct algorithm. However, in the case of the CS scheme, SVM-struct is superior to the perceptron and achieves substantially smaller error rates.

This relative performance of the perceptron algorithm is both surprising as well as expected. The surprising aspect is that it is able to keep up with the considerably more sophisticated, mathematically rigorous, and computationally expensive optimizers used in SVM-struct, which tend to converge to a local minimum solution that is close to the global minimum. However, this behavior, especially when the results of the CS scheme are taken into account, was expected because the hypothesis spaces of the S and SS schemes are rather small (the number of variables in the S and SS models are $K$ and $2K$, respectively) and as such the
optimization problem is relatively easy. However, in the case of the CS scheme which is parameterized by $K^2$ variables, the optimization problem becomes harder, and SVM-struct’s optimization framework is capable of finding a better solution. Due to this observation we did not pursue the ranking perceptron algorithm any further when we considered two-level models that incorporate hierarchy information.

**Hierarchical Two-Level Learning Approaches**

The results for remote homology detection show that the use of hierarchical information does not improve the overall error rates. The situation is different for fold recognition in which the use of hierarchical information leads to some improvements for fd40, especially in terms of balanced error (Table 3.11). Also, these results show that adding information from ancestor nodes is in general better than adding information from descendant nodes, and combining both types of information can sometimes improve the classification performance.

Even though the use of hierarchical information does not improve the overall classification accuracy, as the results in Tables 3.13 and 3.14 show, it does reduce the severity of the misclassifications. Comparing the top$n_1$ and top$n_3$ error rates for the two sets of schemes, we see that by incorporating hierarchical information they achieve consistently lower error rates. For remote homology detection, there is more than 50% reduction in the error rate due to the addition of fold- and class-level information, whereas somewhat smaller gains (4%–20%) are obtained for fold recognition by incorporating class-level information. It is also interesting to note, that there is no reduction in error rates by addition of descendant node information i.e superfamiliy-level, in case of fold recognition problem.

**Comparison with Earlier Results**

As discussed in the introduction, our research in this paper was motivated by the recent work of Ie et. al. [83] in which they looked at the same problem of solving the $K$-way classification problem in the context of remote homology and fold recognition and presented a two-level learning approach based on the simple scaling model (S) with and without hierarchical information. Table 3.15 on the previous page shows the results reported in that work on the common dataset and classification problems (remote homology prediction for sf95). In addition, Table 3.15 on the preceding page shows the results obtained by our algorithms using the simple scaling model and the best results achieved among the three different models that were considered in this work (i.e., S, SS, and CS).

These results show that the zero-one and balanced error rates of our algorithms are in most cases less than half of that achieved by the previous algorithms. This performance advantage can be attributed to (i)
differences in the one-vs-rest binary classifiers ( [83] used the profile kernel [116] whereas our schemes used
the SW-PSSM kernel), (ii) our implementation of the ranking perceptron allows for a better specification
of the classification margin, and (iii) our results have been optimized by performing a model selection step,
described in detail in the methods section.

3.8 Conclusions

We have presented and experimentally evaluated a number of kernel functions for protein sequence clas-
sification that were derived by considering explicit measures of profile-to-profile sequence similarity. The
experimental evaluation in the context of a remote homology prediction problem and a fold recognition prob-
lem show that these kernels are capable of producing superior classification performance over that produced
by earlier schemes.

Three major observations can be made by analyzing the performance achieved by the various kernel
functions for developing the one-versus-rest binary classification models. First, as was the case with a number
of studies on the accuracy of protein sequence alignment [134, 215, 132], the proper use of sequence profiles
lead to dramatic improvements in the overall ability to detect remote homologs and identify proteins that
share the same structural fold. Second, kernel functions that are constructed by directly taking into account
the similarity between the various protein sequences tend to outperform schemes that are based on a feature-
space representation (where each dimension of the space is constructed as one of $k$-possibilities in a $k$-residue
long subsequence or using structural motifs (Isites) in the case of SVM-HMMSTR). This is especially evident
by comparing the relative advantage of the window-based kernels over the Profile kernel. Third, time-tested
methods for comparing protein sequences based on optimal local alignments (as well as global and local-
global alignments), when properly optimized for the classification problem at hand, lead to kernel functions
that are in general superior to those based on either short subsequences (e.g., Spectrum, Mismatch, Profile, or
window-based kernel functions) or local structural motifs (e.g., SVM-HMMSTR). The fact that these widely
used methods produce good results in the context of SVM-based classification is reassuring as to the validity
of these approaches and their ability to capture biologically relevant information.

We further integrate the outputs of the predictions obtained from our first level binary classification mod-
els, to solve the general $k$-way classification model in the context of remote homology detection and fold
recognition.

Our results show that direct $k$-way SVM-based formulations and algorithms based on the two-level learn-
ing paradigm are quite effective for solving these problems and achieve better results than those obtained
by using a set of binary one-vs-rest SVM-based classifiers. Moreover, our results and analysis showed that the two-level schemes that incorporate predictions from binary models constructed for ancestral categories within the SCOP hierarchy tend to not only lead to lower error rates but also reduce the number of errors in which a superfamily is assigned to an entirely different fold and a fold is predicted as being from a different SCOP class.

These classification methods lay the foundation for several other similar prediction problems in computation biology- protein subcellular localization, and function prediction. We used these SVM-based classifiers for selecting the best fold and searching within the predicted fold for a suitable template for a given target during the CASP 7 competition.
Local Structure Similarity Prediction

4.1 Introduction

The dynamic-programming-based algorithms [138, 188] used in sequence-sequence target-template alignment are also used by many methods to align a pair of protein structures. However, the key difference between these two problem settings is that, while the target-template alignment methods score a pair of aligned residues using sequence-derived information, the structure alignment methods use information derived from the structure of the protein. For example, structure alignment methods like CE [184] and MUSTANG [113] score a pair of residues by considering how well fixed-length fragments (i.e., short contiguous backbone segments) centered around each residue align with each other. This score is usually computed as the root mean squared deviation (RMSD) of the optimal superimposition of the two fragments.

Motivated by the alignment requirements of comparative modeling approaches and the operational characteristics of protein structure alignment algorithms, we focus on the problem of estimating the RMSD value of a pair of protein fragments by considering only sequence-derived information. Besides its direct application to target-template alignment, accurate estimation of these fragment-level RMSD values can also be used to solve a number of other problems related to protein structure prediction such as identifying the best template by assessing the quality of target-template alignments and identifying high-quality segments of an alignment.

We present algorithms to solve the fragment-level RMSD prediction problem using a supervised learning
framework based on support vector regression and classification that incorporates sequence-derived information in the form of position-specific profiles and predicted secondary structure [104]. This information is effectively encoded in fixed-length feature vectors. We develop and test novel second-order pairwise exponential kernel functions designed to capture the conserved signals of a pair of local windows centered at each of the residues and use a fusion-kernel-based approach to incorporate the profile- and secondary structure-based information.

4.2 Problem Definition: Fragment-Level RMSD Prediction

Given proteins $X$ and $Y$, we denote a residue pair formed by residues $x_i$ and $y_j$ by $\pi(x_i, y_j)$. Given a protein $X$ of length $n$ and a user-specified parameter $w$, we define $w\text{mer}(x_i)$ to be the $(2w + 1)$-length contiguous subsequence of $X$ centered at position $i$ ($w < i \leq n - w$) (Already defined in Section 2.4.1 on page 21). Similarly, given a user-specified parameter $v$, we define $v\text{frag}(x_i)$ to be the $(2v + 1)$-length contiguous substructure of $X$ centered at position $i$ ($v < i \leq n - v$). These substructures are commonly referred to as fragments [184, 113]. Without loss of generality, we represent the structure of a protein using the $C_\alpha$ atoms of its backbone. The $w\text{mers}$ and $v\text{frags}$ are fixed-length windows that are used to capture information about the sequence and structure around a particular sequence position, respectively.

Given a residue-pair $\pi(x_i, y_j)$, we define $f\text{RMSD}(x_i, y_j)$ to be the structural similarity score between $v\text{frag}(x_i)$ and $v\text{frag}(y_j)$. This score is computed as the root mean square deviation between the pair of substructures after optimal superimposition. A residue-pair $\pi(x_i, y_j)$ will be called reliable if its $f\text{RMSD}$ is below a certain value (i.e., there is a good structural superimposition of the corresponding substructures).

We aim to solve the following two problem related to predicting the local similarity of residue-pairs.

**Definition 1 (fRMSD Estimation Problem)** Given a residue-pair $\pi(x_i, y_j)$, estimate the $f\text{RMSD}(x_i, y_j)$ score by considering information derived from the amino acid sequence of $X$ and $Y$.

**Definition 2 (Reliability Prediction Problem)** Given a residue-pair $\pi(x_i, y_j)$, determine whether it is reliable or not by considering only information derived from the amino acid sequence of $X$ and $Y$.

It is easy to see that the reliability prediction problem is a special case to the $f\text{RMSD}$ estimation problem. As such, it may be easier to develop effective solution methods for it and this is why we consider it as a different problem in this work.
4.2.1 $f_{\text{RMSD}}$ Applications

The effective solution to these two problems has four major applications to protein structure prediction. First, given an existing alignment between a (target) protein and a template, a prediction of the $f_{\text{RMSD}}$ scores of the aligned residue-pairs (or their reliability) can be used to assess the quality of the alignment and potentially select among different alignments and/or different templates. Second, $f_{\text{RMSD}}$ scores (or reliability assessments) can be used to analyze different protein-template alignments in order to identify high-quality moderate-length fragments. These fragments can then be used by fragment-assembly-based protein structure prediction methods like TASSER [225] and ROSETTA [169] to construct the structure of a protein. Third, since residue-pairs with low $f_{\text{RMSD}}$ scores are good candidates for alignment, the predicted $f_{\text{RMSD}}$ scores can be used to construct a position-to-position scoring matrix between all pairs of residues in a protein and a template. This scoring matrix can then be used by an alignment algorithm to compute a high-quality alignment for structure prediction via comparative modeling. Essentially, this alignment scheme uses predicted $f_{\text{RMSD}}$ scores in an attempt to mimic the approach used by various structural alignment methods [113, 184]. Fourth, the $f_{\text{RMSD}}$ scores (or reliability assessments) can be used as input to other prediction tasks such as remote homology prediction and/or fold recognition.

In this work we study and evaluate the feasibility of solving the $f_{\text{RMSD}}$ estimation and reliability prediction problems for a given residue-pair. Our evaluation, also focuses on residue-pairs that are derived from optimal local sequence alignments. This allows evaluation of the first two applications discussed in the previous paragraph (assessment of target-template alignment and identification of high-confidence alignment regions). In Section 4.6 on page 88 we also show how estimated $f_{\text{RMSD}}$ scores can be used to improve sequence alignments in conjunction with other profile-to-profile schemes, especially for protein pairs with low sequence identity (less than 12%).

4.3 Literature Review

The problem of determining the reliability of residue-pairs has been visited before in several different settings. ProfNet [142, 141] uses artificial neural networks to learn a scoring function to align a pair of protein sequences. In essence, ProfNet aims to differentiate related and unrelated residue-pairs and also estimate the RMSD score between these residue-pairs using profile information. Protein pairs are aligned using STRUCTAL [60], residue-pairs within 3Å apart are considered to be related, and unrelated residue-pairs are selected randomly from protein pairs known to be in different folds. A major difference between our methods and ProfNet is in the definition of reliable/unreliable residue-pairs and on how the RMSD score between residue-
pairs is measured. We measure the structural similarity of two residues ($f_{	ext{RMSD}}$) by looking at how well their $v_{	ext{frags}}$ structurally align with each other. However, ProfNet only considers the proximity of two residues within the context of their global structural alignment. As such, two residues can have a very low RMSD and still correspond to fragments whose structure is substantially different. This fundamental difference makes direct comparisons between the results impossible. The other major differences lie in the development of order independent coding schemes and the use of information from a set of neighboring residues by using a $w_{	ext{mer}}$ size greater than zero.

The task of aligning a pair of sequences has also been cast as a problem of learning parameters (gap opening, gap extension, and position independent substitution matrix) within the framework of discriminatory learning [89, 221] and setting up optimization parameters for an inverse learning problem [195]. Recently, pair conditional random fields were also used to learn a probabilistic model for estimating the alignment parameters (i.e., gap and substitution costs) [44].

### 4.4 Methods and Algorithms

We approach the problems of distinguishing reliable/unreliable residue-pairs and estimating their $f_{	ext{RMSD}}$ scores following a supervised machine learning framework and use support vector machines (SVM) [87,207] to solve them.

Given a set of positive residue-pairs $A^+$ (i.e., reliable) and a set of negative residue-pairs $A^-$ (i.e., unreliable), the task of support vector classification is to learn a function $f(\pi)$ of the form

$$f(\pi) = \sum_{\pi_i \in A^+} \lambda_i^+ K(\pi, \pi_i) - \sum_{\pi_i \in A^-} \lambda_i^- K(\pi, \pi_i), \quad (4.1)$$

where $\lambda_i^+$ and $\lambda_i^-$ are non-negative weights that are computed during training by maximizing a quadratic objective function, and $K(., .)$ is the kernel function designed to capture the similarity between pairs of residue-pairs. Having learned the function $f(\pi)$, a new residue-pair $\pi$ is predicted to be positive or negative depending on whether $f(\pi)$ is positive or negative. The value of $f(\pi)$ also signifies the tendency of $\pi$ to be a member of the positive or negative class and can be used to obtain a meaningful ranking of a set of the residue-pairs.

We use the error insensitive support vector regression $\epsilon$-SVR [207, 189] for learning a function $f(\pi)$ to predict the $f_{	ext{RMSD}}(\pi)$ scores. Given a set of training instances $(\pi_i, f_{	ext{RMSD}}(\pi_i))$, the $\epsilon$-SVR aims to learn a function of the form

$$f(\pi) = \sum_{\pi_i \in A^+} a_i^+ K(\pi, \pi_i) - \sum_{\pi_i \in A^-} a_i^- K(\pi, \pi_i), \quad (4.2)$$
where $\Delta^+$ contains the residue-pairs for which $f \text{RMSD}(\pi_i) - f(\pi_i) > \epsilon$, $\Delta^-$ contains the residue pairs for which $f \text{RMSD}(\pi_i) - f(\pi_i) < -\epsilon$, and $a_i^+$ and $a_i^-$ are non-negative weights that are computed during training by maximizing a quadratic objective function. The objective of the maximization is to determine the flattest $f(\pi)$ in the feature space and minimize the estimation errors for instances in $\Delta^+ \cup \Delta^-$. Hence, instances that have an estimation error satisfying $|f(\pi_i) - f \text{RMSD}(\pi_i)| < \epsilon$ are neglected. The parameter $\epsilon$ controls the width of the regression deviation or tube.

In the current work we focused on several key considerations while setting up the classification and regression problems. In particular we explored different types of sequence information associated with the residue-pairs, developed efficient ways to encode this information to form fixed length feature vectors, and designed sensitive kernel functions to capture the similarity between pairs of residues in the feature spaces.

### 4.4.1 Sequence-based Information

For a given protein $X$, we encode the sequence information using profiles (Section 2.4.2) and predicted secondary structure. The profile of a sequence captures evolutionary information and has been described in detail in Section 2.4.2.

For a sequence $X$ of length $n$ we predict the secondary structure and generate a position-specific secondary structure matrix $S_X$ of length $n \times 3$. The $(i, j)$ entry of this matrix represents the strength of the amino acid residue at position $i$ to be in state $j$, where $j \in (0, 1, 2)$ corresponds to the three secondary structure elements: alpha helices (H), beta sheets (E), and coil regions (C).

Motivated by the work in [104], we also derived a feature space using the BLOSUM62 matrix. These scores correspond to the position non-specific score for the various residue positions. The primary motivation for the use of the BLOSUM62 derived feature vectors was to improve the classification accuracy in cases where the target sequence does not have a sufficiently large number of homologous sequences in nr, and/or PSI-BLAST fails to compute a correct alignment for some segments of the sequence [104]. By augmenting the input coding schemes to contain both PSSM- as well as BLOSUM62-based information, the SVM could potentially learn a model that is also partially based on the non-position specific information. This information will remain valid even in cases in which PSI-BLAST could not generate correct alignments. Our preliminary tests show that using the non-position specific information does not lead to an increase in the performance accuracy, and as such we do not experiment with them in this work.
4.4.2 Coding Schemes

The input to our prediction algorithms are a set of \( w \text{mer} \)-pairs associated with each residue-pair \( \pi(x_i, y_j) \). The input feature space is derived using various combinations of the elements in the \( P \) and \( S \) matrices that are associated with the subsequences \( w \text{mer}(x_i) \) and \( w \text{mer}(y_j) \).

We will use \( P_X(i - w \ldots i + w) \) to denote the \((2w + 1)\) rows of matrix \( P_X \) corresponding to \( w \text{mer}(x_i) \). A similar notation will be used for matrix \( S \).

Concatenation Coding Scheme

For a given residue-pair \( \pi(x_i, y_j) \), the feature-vector of the concatenation coding scheme is obtained by first linearizing the matrices \( P_X(i - w \ldots i + w) \) and \( P_Y(j - w \ldots j + w) \) and then concatenating the resulting vectors. This leads to feature-vectors of length \( 2 \times (2w + 1) \times 20 \). A similar representation is derived for matrix \( S \) leading to feature-vectors of length \( 2 \times (2w + 1) \times 3 \).

The concatenation coding scheme is order dependent as the feature-vector representations for \( \pi(x_i, y_j) \) and \( \pi(y_j, x_i) \) are not equivalent even though they represent the same residue-pair. This order-dependency can potentially reduce the accuracy of the classification/regression methods as there is no guarantee that the ordering used during prediction was the same with that used during learning.

We addressed this problem as follows. During training, we built a model using an arbitrary ordering of the \( w \text{mers} \) for each residue-pair. However, during model application, we classified/regressed both orderings of a residue-pair and used the average of the SVM/\( \epsilon \)-SVR outputs as the final classification/regression result. We denote this averaging method by \( \text{avg} \). Our empirical evaluation shows that the averaging scheme achieves up to 1% improvement over both representations for both the classification as well as regression problem. We present these results as part of the supplementary data, and always refer to the \( \text{avg} \) representation while using the concatenation coding scheme in this study.

Pairwise Coding Scheme

For a given residue-pair \( \pi(x_i, y_j) \), the pairwise coding scheme generates a feature-vector by linearizing the matrix formed by an element-wise product between \( P_X(i - w \ldots i + w) \) and \( P_Y(j - w \ldots j + w) \). The length of this vector is \((2w + 1) \times 20\) and is order independent. If we denote the element-wise product operation by “\( \odot \)”, then the element-wise product matrix is given by

\[
P_X(-w + i \ldots w + i) \odot P_Y(-w + j \ldots w + j).
\] (4.3)
A similar approach is used to obtain the pairwise coding scheme for matrix $S$, leading to feature-vectors of length $(2w + 1) \times 3$.

### 4.4.3 Kernel Functions

The general structure of the kernel function that we use for capturing the similarity between a pair of residue-pairs $\pi(x_i, y_j)$ and $\pi'(x_i', y_j')$ is given by

$$
K^{cs}(\pi, \pi') = \exp \left( 1.0 + \frac{K^c_1(\pi, \pi')}{\sqrt{K^c_1(\pi, \pi)K^c_1(\pi', \pi')}} \right),
$$

where $K^c_1(\pi, \pi')$ is given by

$$
K^c_1(\pi, \pi') = K^c_2(\pi, \pi') + (K^c_2(\pi, \pi'))^2,
$$

and $K^c_2(\pi, \pi')$ is a kernel function that depends on the choice of particular coding scheme ($cs$). For the concatenation coding scheme using matrix $P$ (i.e., $cs = P_{conc}$), $K^c_2(\pi, \pi')$ is given by

$$
K^c_{P_{conc}}(\pi, \pi') = \sum_{k=-w}^{k=w} (P_X(i+k), P_X'(i'+k)) + \sum_{k=-w}^{k=w} (P_Y(j+k), P_Y'(j'+k)).
$$

For the pairwise coding scheme using matrix $P$ (i.e., $cs = P_{pair}$), $K^c_2(\pi, \pi')$ is given as

$$
K^c_{P_{pair}}(\pi, \pi') = \sum_{k=-w}^{k=w} (P_X(i+k) \otimes P_Y(j+k), P_X'(i'+k) \otimes P_Y'(j'+k)).
$$

Similar kernel functions can be derived using matrix $S$ for both the pairwise and the concatenation coding schemes. We will denote these coding schemes as $S_{pair}$ and $S_{conc}$, respectively. Since the overall structure of the kernel that we used (Equations 4.4 and 4.5) is that of a normalized second-order exponential function, we will refer to it as $nsoe$.

The second-order component of Equation 4.5 allows the $nsoe$ kernel to capture pairwise dependencies among the residues used at various positions within each $wmer$, and we found that this leads to better results over the linear function. This observation is also supported by earlier research on secondary-structure prediction as well [104]. In addition, $nsoe$’s exponential function allows it to capture non-linear relationships within the data just like the kernels based on the Gaussian and radial basis function [207].
Fusion Kernels

We also developed a set of kernel functions that incorporate both profile and secondary structure information using an approach motivated by fusion kernels [118, 189]. Specifically, we constructed a new kernel function as the unweighted sum of the \textit{nsoe} kernel function for the profile and secondary structure information. For example, the concatenation-based fusion kernel function is given by

$$K^{(P + S)}_{\text{conc}}(\pi, \pi') = K^P_{\text{conc}}(\pi, \pi') + K^S_{\text{conc}}(\pi, \pi').$$

(4.8)

A similar kernel function can be defined for the pairwise coding scheme as well. We will denote the pairwise-based fusion kernel by $K^{(P + S)}_{\text{pair}}(\pi, \pi')$. Note that since these fusion kernels are linear combinations of valid kernels, they are also admissible kernels.

Cascaded Models

We also experimented with a second-level model that incorporates the predictions obtained from the first-level models and the original sequence information. This form of cascaded models has found success in secondary structure prediction algorithms like PHD [173], PSIPRED [92] and YASSPP [104].

In the two-level cascaded models, the fusion kernel (Equation 4.8) would be used to assign each position the $f_{\text{RMSD}}$ score. Since this first level model ($L_1$ model) treats each position independently, we would learn a second-level model ($L_2$ model) taking into account the predicted $f_{\text{RMSD}}$ score of adjacent positions. Specifically, such a kernel function in the cascaded level setting would be given by

$$K^{**(P + S)}_{\text{conc}}(\pi, \pi') = (1 - \beta) K^{L_1}(\pi, \pi') +$$

$$+ \beta (K^{(P + S)}_{\text{conc}}(\pi, \pi')),$$

(4.9)

where $K^{(P + S)}_{\text{conc}}$ represents the concatenation based coding scheme given by Equation 4.8, and $K^{L_1}$ is of the \textit{nsoe} kernel form, using the $(2w + 1)$ predictions from the first level, and the $\beta$ represents the weight parameter used in this fusion kernel setting. We use the same value of $w$ for both the levels, and for the second-level model, we use the $(2w + 1)$ predictions obtained from the $L_1$ model, along with the profile and secondary structure based information as used in the $L_1$ model.
## Table 4.1. Dataset Statistics.

<table>
<thead>
<tr>
<th></th>
<th>Reliability Prediction</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>all</td>
<td>fam</td>
<td>suf</td>
<td>fold</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total &lt; 0.75 (+) &gt; 0.75 (-)</td>
<td>Total &lt; 0.75 (+) &gt; 0.75 (-)</td>
<td>Total &lt; 0.75 (+) &gt; 0.75 (-)</td>
<td>Total &lt; 0.75 (+) &gt; 0.75 (-)</td>
<td></td>
</tr>
<tr>
<td>DSA</td>
<td>46803</td>
<td>16127</td>
<td>30676</td>
<td>12356</td>
<td>6691</td>
</tr>
<tr>
<td>DSR</td>
<td>28788</td>
<td>1977</td>
<td>26811</td>
<td>5311</td>
<td>341</td>
</tr>
</tbody>
</table>

Table 4.1. Dataset Statistics.

<table>
<thead>
<tr>
<th></th>
<th>RMSD Estimation</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>all</td>
<td>fam</td>
<td>suf</td>
<td>fold</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Total μ( fRMSD) σ( fRMSD)</td>
<td>Total μ( fRMSD) σ( fRMSD)</td>
<td>Total μ( fRMSD) σ( fRMSD)</td>
<td>Total μ( fRMSD) σ( fRMSD)</td>
<td></td>
</tr>
<tr>
<td>DSA</td>
<td>46803 1.84 1.28</td>
<td>12356 1.38 1.22</td>
<td>24616 2.01 1.26</td>
<td>9831 2.07 1.26</td>
<td></td>
</tr>
<tr>
<td>DSR</td>
<td>28788 2.61 1.0</td>
<td>5311 2.62 1.01</td>
<td>17360 2.62 1.0</td>
<td>6117 2.59 1.01</td>
<td></td>
</tr>
</tbody>
</table>

4.5 Experimental Results

### 4.5.1 Dataset Description

We evaluated the classification and regression performance of the various kernels on a set of protein pairs used in a previous study for learning a profile-to-profile scoring function [142]. These pairs of proteins were derived from the SCOP 1.57 database, classes a-e, with no two protein domains sharing greater than 40% sequence identity. The dataset is comprised of 158 protein pairs belonging to the same family, 394 pairs belonging to the same superfamily but not the same family, and 184 pairs belonging to the same fold but not the same superfamily.

We use a five-fold cross-validation framework to evaluate the performance of the various classifiers and regression models. We ensure that for each of the cross-validation runs, the sequence identity between sequences in the test and train set is at most 40%.

We created two datasets consisting of selected residue-pairs that allow us to evaluate the reliability prediction and fRMSD estimation problems within the context of the applications, discussed in Section 5.2. The first dataset, referred to as “DSR”, is designed to evaluate the fRMSD estimation accuracy for the application of creating position-to-position scoring matrices for generating sequence alignments. It contains 28788 residue-pairs that were selected randomly from each protein pair. The second dataset, refereed to as “DSA”, is designed to evaluate the fRMSD estimation accuracy for the application of assessing the target-template alignment quality and for selecting high-quality fragments from different alignments. It contains 46803 residue-pairs that correspond to the aligned positions produced by the Smith-Waterman [188] algorithm from
4.5.2 Evaluation Methodology

We measure the quality of the methods using the standard receiver operating characteristic (ROC) scores and the $ROC_5$ scores averaged across every protein pair. The $ROC$ score is the normalized area under the curve that plots the true positives against the false positives for different thresholds for classification [64]. The $ROC_n$ score is the area under the ROC curve up to the first $n$ false positives. We compute the $ROC$ and $ROC_5$ numbers for every protein pair and report the average results across all the pairs and cross-validation steps. We chose to report $ROC_5$ scores because each individual ROC-based evaluation is performed on a per protein-pair basis, which, on average, involves one to two hundred residue-pairs.

The regression performance is assessed by computing the standard Pearson correlation coefficient (CC) between the predicted and observed $f_{\text{RMSD}}$ values for every protein pair. We also compute the root mean square error $rmse$ between the predicted and observed $f_{\text{RMSD}}$ values for every protein pair. The results reported are averaged across the different protein pairs and cross-validation steps.

---

1http://bioinfo.cs.umn.edu/supplements/fRMSDPred/
4.5.3 Baseline Profile-to-Profile Scoring schemes

To assess the effectiveness of our supervised learning algorithms we compare their performance against that obtained by using two profile-to-profile scoring schemes to solve the same problems. Specifically, we use the profile-to-profile scoring schemes to compute the similarity between the aligned residue-pairs summed over the length of their wmers. To assess how well these scores correlated with the $f_{\text{RMSD}}$ score of each residue-pair we compute their correlation coefficients. Note that since residue-pairs with high-similarity score are expected to have low $f_{\text{RMSD}}$ scores, good values for these correlation coefficients will be close to -1. Similarly, for the reliability prediction problem, we sort the residue-pairs in decreasing similarity score order and assess the performance by computing $ROC$ and $ROC_5$ scores.

The two profile-to-profile scoring schemes that we used are based on the dot-product and the PICASSO score, both of which are used extensively and shown to produce good results [134, 215, 132] (described in Section 2.4.2 on page 23).

The dot-product similarity score is defined both for the profile- as well as the secondary-structure-based information, whereas the PICASSO score is defined only for the profile-based information. We will use $P_{\text{dotp}}$, $S_{\text{dotp}}$, and $P_{\text{pic}}$ to denote these three similarity scores, respectively.

4.5.4 $f_{\text{RMSD}}$ Prediction Results

We performed a comprehensive study evaluating the classification and regression performance of the various information sources, coding schemes, and kernel functions (Section 4.4 on page 74) and compared it against the performance achieved by the profile-to-profile scoring schemes (Section 4.5.3).

We performed a number of experiments using different length wmers for both the SVM/$\epsilon$-SVR and profile-to-profile-based schemes. These experiments showed that the supervised learning schemes achieved the best results when $5 \leq w \leq 7$, whereas in the case of the profile-to-profile scoring schemes, the best performing value of $w$ was dependent on the particular scoring scheme. For these reasons, for all the SVM/$\epsilon$-SVR-based schemes we only report results for $w = 6$, whereas for the profile-to-profile schemes we report results for the values of $w$ that achieved the best performance. For all our experiments we compared the performance of the concatenation coding scheme with the pairwise coding scheme, and found the concatenation coding scheme to always have better or comparable results to the pairwise coding scheme. Hence, we present results only for the concatenation coding scheme.
RBF versus NSOE Kernel Functions

Table 4.2 on the next page compares the classification and regression performance achieved by the standard \textit{rbf} kernel against that achieved by the \textit{nsoe} kernel described in Section 4.4.3 on page 77. The \textit{rbf} results were obtained after normalizing the feature-vectors to unit length, as it produced substantially better results over the unnormalized representation.

These results show that the performance achieved by the \textit{nsoe} kernel is consistently better than that of the \textit{rbf} kernel for the reliability prediction and $f_{\text{RMSD}}$ estimation problems. For example, for the concatenation coding scheme, the \textit{nsoe} kernel achieves 3\% and 1\% better $ROC_5$ values and its correlation coefficient is 3.5\% and 13.4\% higher than the \textit{rbf} kernel for the DSA and DSR datasets, respectively.

The key difference between the two kernels is that in the \textit{nsoe} kernel the even-ordered terms are weighted higher in the expansion of the infinite exponential series than the \textit{rbf} kernel. As discussed in Section 4.4.3, this allows the \textit{nsoe} kernel function to better capture the pairwise dependencies that exists at different positions of each \textit{w}mer.

In Table 4.2 we also present results for the pairwise coding scheme, and the regression and classification performance is always poorer compared to the concatenation coding scheme. This trend was verified across all the other experiments, and hence we report results for the concatenation coding scheme only.

Input Information

Table 4.3 on page 84 compares how the features derived from the profiles and the predicted secondary structure impact the performance achieved for the reliability prediction and $f_{\text{RMSD}}$ estimation problems on the DSA and DSR datasets. The table presents results for the SVM-based schemes using the concatenation coding scheme as well as results obtained by the dot-product-based profile-to-profile scoring scheme. (See Section 4.5.3 on the preceding page for a discussion on how these scoring schemes were used to solve the reliability prediction problem and $f_{\text{RMSD}}$ estimation problem). Also, as discussed in Section 4.5.3 on the previous page, the scores computed by the profile-to-profile scoring schemes should be negatively correlated with the $f_{\text{RMSD}}$; thus, negative correlations represent good estimations.

Analyzing the results across the different SCOP-derived test sets, it can be seen that the secondary structure coding schemes perform better compared to the profile-based coding schemes. Moreover, the relative performance gap between secondary-structure- and profile-based schemes tends to increase as we move from the superfamily- to the fold-derived set. These results should not be surprising as the secondary structure of two \textit{w}mers (assuming that it is predicted correctly) provides better information as to the structural similarity of their corresponding fragments than profiles alone. Also note that the only time that the profile-based cod-
The test and training set consisted of proteins from the *all* set. RP denotes the reliability prediction results, and EST denotes the $f_{\text{RMSD}}$ estimation results. $rbf$ denotes the standard radial basis kernel function, and $nsoe$ denotes the normalized second order exponential kernel function introduced in this work. The numbers in bold show the best performing schemes for each of the sub-tables.

Table 4.3 on the following page also highlights the superior performance of the SVM-based methods in comparison to the dot-product-based profile-to-profile scoring schemes. Comparing the $ROC_5$ value for the reliability prediction problem, the SVM-based scheme ($S_{\text{conc}}$) is about 43% and 8% better than the secondary structure based dot-product scoring schemes ($S_{\text{dotp}}$) on the DSA and DSR datasets, respectively.

The results obtained on the DSA and DSR datasets for the reliability prediction and $f_{\text{RMSD}}$ estimation problem, indicate that it is more challenging to perform the classification/estimation task for residue pairs that do not come from local alignments.

### Fusion Kernels

Table 4.3 on the next page also shows the performance achieved by the fusion kernels. For comparison purposes, this table also shows the best results that were obtained by using the profile-to-profile-based schemes...
### Table 4.3. Classification and regression performance of the individual and fusion kernels.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>ROC</th>
<th>ROC</th>
<th>CC</th>
<th>rmse</th>
<th>ROC</th>
<th>ROC</th>
<th>CC</th>
<th>rmse</th>
<th>ROC</th>
<th>ROC</th>
<th>CC</th>
<th>rmse</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\mathcal{P}_{\text{dotp}}) (6)</td>
<td>0.328</td>
<td>0.701</td>
<td>-0.331</td>
<td>-</td>
<td>0.465</td>
<td>0.762</td>
<td>-0.451</td>
<td>-</td>
<td>0.314</td>
<td>0.699</td>
<td>-0.322</td>
<td>-</td>
</tr>
<tr>
<td>(\mathcal{S}_{\text{dotp}}) (3)</td>
<td>0.432</td>
<td>0.814</td>
<td>-0.541</td>
<td>-</td>
<td>0.455</td>
<td>0.789</td>
<td>-0.523</td>
<td>-</td>
<td>0.428</td>
<td>0.824</td>
<td>-0.547</td>
<td>-</td>
</tr>
<tr>
<td>((\mathcal{P} + \mathcal{S})_{\text{dotp}}) (6)</td>
<td>0.339</td>
<td>0.713</td>
<td>-0.350</td>
<td>-</td>
<td>0.470</td>
<td>0.769</td>
<td>-0.460</td>
<td>-</td>
<td>0.327</td>
<td>0.712</td>
<td>-0.342</td>
<td>-</td>
</tr>
<tr>
<td>(\mathcal{P}<em>{\text{pic}} + \mathcal{S}</em>{\text{dotp}}) (3)</td>
<td>0.512</td>
<td>0.855</td>
<td>-0.602</td>
<td>-</td>
<td>0.541</td>
<td>0.835</td>
<td>-0.596</td>
<td>-</td>
<td>0.494</td>
<td>0.858</td>
<td>-0.601</td>
<td>-</td>
</tr>
<tr>
<td>(\mathcal{P}_{\text{conc}}) (all)</td>
<td>0.577</td>
<td>0.881</td>
<td>0.575</td>
<td>0.946</td>
<td>0.589</td>
<td>0.881</td>
<td>0.634</td>
<td>0.838</td>
<td>0.574</td>
<td>0.884</td>
<td>0.561</td>
<td>0.974</td>
</tr>
<tr>
<td>(\mathcal{S}_{\text{conc}}) (all)</td>
<td>0.622</td>
<td>0.898</td>
<td>0.712</td>
<td>0.776</td>
<td>0.589</td>
<td>0.875</td>
<td>0.692</td>
<td>0.747</td>
<td>0.628</td>
<td>0.907</td>
<td>0.715</td>
<td>0.790</td>
</tr>
<tr>
<td>((\mathcal{P} + \mathcal{S})_{\text{conc}}) (all)</td>
<td>0.666</td>
<td>0.914</td>
<td>(0.733)</td>
<td>0.744</td>
<td>0.657</td>
<td>0.902</td>
<td>(0.727)</td>
<td>0.699</td>
<td>0.665</td>
<td>0.919</td>
<td>0.733</td>
<td>0.762</td>
</tr>
<tr>
<td>((\mathcal{P} + \mathcal{S})_{\text{conc}}) (all)</td>
<td>0.633</td>
<td>0.896</td>
<td>0.715</td>
<td>0.723</td>
<td>0.623</td>
<td>0.863</td>
<td>0.665</td>
<td>0.677</td>
<td>0.667</td>
<td>0.914</td>
<td>0.738</td>
<td>0.767</td>
</tr>
<tr>
<td>(\mathcal{P}<em>{\text{pic}} + \mathcal{S}</em>{\text{conc}}) (all)</td>
<td>0.363</td>
<td>0.528</td>
<td>0.345</td>
<td>0.491</td>
<td>-0.416</td>
<td>-</td>
<td>0.355</td>
<td>0.551</td>
<td>-0.453</td>
<td>-</td>
<td>0.387</td>
<td>0.514</td>
</tr>
<tr>
<td>(\mathcal{P}_{\text{conc}}) (all)</td>
<td>0.152</td>
<td>0.355</td>
<td>-0.094</td>
<td>-</td>
<td>0.148</td>
<td>0.336</td>
<td>-0.083</td>
<td>-</td>
<td>0.143</td>
<td>0.369</td>
<td>-0.093</td>
<td>-</td>
</tr>
<tr>
<td>(\mathcal{S}_{\text{conc}}) (all)</td>
<td>0.403</td>
<td>0.541</td>
<td>-0.521</td>
<td>-</td>
<td>0.392</td>
<td>0.512</td>
<td>-0.499</td>
<td>-</td>
<td>0.398</td>
<td>0.561</td>
<td>-0.516</td>
<td>-</td>
</tr>
<tr>
<td>((\mathcal{P} + \mathcal{S})_{\text{conc}}) (all)</td>
<td>0.160</td>
<td>0.368</td>
<td>-0.120</td>
<td>-</td>
<td>0.155</td>
<td>0.346</td>
<td>-0.107</td>
<td>-</td>
<td>0.150</td>
<td>0.383</td>
<td>-0.119</td>
<td>-</td>
</tr>
<tr>
<td>((\mathcal{P} + \mathcal{S})_{\text{conc}}) (all)</td>
<td>0.363</td>
<td>0.528</td>
<td>-0.457</td>
<td>-</td>
<td>0.355</td>
<td>0.491</td>
<td>-0.416</td>
<td>-</td>
<td>0.355</td>
<td>0.551</td>
<td>-0.453</td>
<td>-</td>
</tr>
</tbody>
</table>

** denotes that a cascaded scheme was used for training the models. The numbers in bold show the best performing schemes for each of the sub-tables.

RP denotes the reliability prediction results, and EST denotes the \(f_{\text{RMSD}}\) estimation results. The test set consisted of proteins from the fam, suf, fold and all sets, whereas the training set used the all set. The numbers in parentheses for the profile-to-profile scoring schemes indicate the value of \(w\) for the \(w\)mers that were used. ** indicates that a cascaded scheme was used for training the models. The numbers in bold show the best performing schemes for each of the sub-tables.
Table 4.4. Comparing the classification and regression performance of the fusion kernels by training on different subsets of the datasets.

<table>
<thead>
<tr>
<th>Scheme</th>
<th>ROC</th>
<th>ROC</th>
<th>CC</th>
<th>rmse</th>
<th>ROC</th>
<th>ROC</th>
<th>CC</th>
<th>rmse</th>
</tr>
</thead>
<tbody>
<tr>
<td>(P+S)_{conc} - fam</td>
<td>0.632</td>
<td>0.892</td>
<td>0.707</td>
<td>0.728</td>
<td>0.621</td>
<td>0.905</td>
<td>0.695</td>
<td>0.829</td>
</tr>
<tr>
<td>(P+S)_{conc} - suf</td>
<td>0.611</td>
<td>0.886</td>
<td>0.686</td>
<td>0.780</td>
<td>0.651</td>
<td>0.916</td>
<td>0.722</td>
<td>0.776</td>
</tr>
<tr>
<td>(P+S)_{conc} - fold</td>
<td>0.542</td>
<td>0.832</td>
<td>0.686</td>
<td>0.827</td>
<td>0.608</td>
<td>0.884</td>
<td>0.625</td>
<td>0.916</td>
</tr>
<tr>
<td>(P+S)_{conc} - all</td>
<td>0.657</td>
<td>0.902</td>
<td>0.727</td>
<td>0.699</td>
<td>0.665</td>
<td>0.919</td>
<td>0.733</td>
<td>0.762</td>
</tr>
</tbody>
</table>

The test and training set consisted of proteins from fam, suf, and fold sets. RP denotes the reliability prediction results, and EST denotes the \( f_{\text{RMSD}} \) estimation results. The numbers in bold show the best performing schemes for each of the sub-tables.

to solve the reliability prediction and \( f_{\text{RMSD}} \) estimation problem. Specifically, we present dot-product-based results that score each \( w \)-mer as the sum of its profile and secondary-structure information \( (P+S)_{\text{dotp}} \) and results that score each \( w \)-mer as the sum of its PICASSO score and a secondary-structure-based dot-product score \( (P\text{F}_{\text{pic}}+S)_{\text{dotp}} \).

From the results in Table 4.3 on the facing page, we see that the SVM-based schemes, regardless of their coding schemes, consistently outperform the profile-to-profile scoring schemes. Comparing the best results obtained by the concatenation scheme against those obtained by the \( P\text{F}_{\text{pic}}+S_{\text{dotp}} \) scheme for the reliability prediction problem, we see that the former achieves 29\%–34\% and 20\%–26\% higher \( ROC_{5} \) scores on the DSA and DSR datasets, respectively. Similar improvements ranging between 20\%–21\% and 32\%–47\% were obtained for the \( f_{\text{RMSD}} \) estimation problem as well.

To further illustrate the performance difference between the two schemes on the DSA dataset, Figures 4.1 and 4.2 plot the actual \( f_{\text{RMSD}} \) scores against the estimated \( f_{\text{RMSD}} \) scores of \( (P+S)_{\text{conc-all}} \) and the similarity scores of \( P\text{F}_{\text{pic}}+S_{\text{dotp}} \), respectively. Comparing the two figures we can see that the \( f_{\text{RMSD}} \) estimations produced by the \( \epsilon \)-SVR-based scheme are significantly better correlated with those produced by \( P\text{F}_{\text{pic}}+S_{\text{dotp}} \).

Comparing the reliability prediction and \( f_{\text{RMSD}} \) estimation performance achieved by the fusion kernels with that achieved by the individual \( nsoe \) kernels (Table 4.3 on the preceding page) we see that by combing both profile and secondary structure information we can achieve improvements in the range of 3\% to 11\% in terms of \( ROC_{5} \) and 1.2\% to 7.0\% in terms of the \( rmse \) metrics. These performance improvements are consistent across the different test sets \( (all, fam, suf, and fold) \) and datasets (DSA and DSR).

Also, the simple dot-product based scoring of the secondary structure and profiles does not lead to an
improvement in the classification and regression result for the DSR dataset, but using the fusion kernel always leads to an improvement compared to the individual kernels.

Rather than using the unweighted combination of the profile-based and secondary structure based kernels (Equation 4.8), we also experimented changing the weighting on the two kernels. Our experiments indicated that the best performance for the reliability prediction classification problem and the \( f_{\text{RMSD}} \) estimation regression problem was achieved by an equal weighting of the two kernels.

![Figure 4.1](image1.png)

**Figure 4.1.** Scatter plot for test protein-pairs at all levels between estimated and actual \( f_{\text{RMSD}} \) scores on the aligned dataset. The color coding represents the approximate density of points plotted in a fixed normalized area.

![Figure 4.2](image2.png)

**Figure 4.2.** Scatter plot for test protein-pairs at all levels between profile-to-profile scores and actual \( f_{\text{RMSD}} \) scores on the aligned dataset. The color coding represents the approximate density of points plotted in a fixed normalized area.

### Cascaded Level Models

Table 4.3 on page 84 shows the results for the reliability prediction and \( f_{\text{RMSD}} \) estimation problem using the cascaded-level learning on the DSA and DSR datasets, marked in **. We use the \((P+S)^{conc}\) kernel as the base for the cascaded-level models, due to its superiority in comparison to other kernels and coding schemes.

The results were averaged across protein pairs after performing a five fold cross validation, where each pair of sequences were a part of the final test set. To prevent unwanted bias, we split the dataset such that three-fifths of the dataset was used for learning the first level model, one-fifth for learning the second-level model and one-fifth for the final testing.

We performed a parameter study to determine the best weights for the fusion kernel (Equation 4.9), and found that a weight of 0.1 (\( \beta = 0.9 \)) on the predictions obtained from the first-level model gave the best classification and estimation results.

Comparing the reliability prediction and \( f_{\text{RMSD}} \) estimation performance in Table 4.3, it can be noticed
that the cascaded-level models are generally outperformed or comparable to the single-level models. We have highlighted those results in Table 4.3 on page 84 by bold facing them. The complexity of the cascaded-level models makes them unsuitable in comparison the single level models.

**SCOP-level Trained Models**

To investigate the extent to which the classification and regression performance can be improved by building models that are specific to different levels of sequence similarity we performed an experiment in which we trained three different models using the *fam*, *suf*, and *fold* subsets of the DSA dataset. These results for \( \text{fRMSD} \) estimation and reliability prediction are presented in Table 4.4 on page 85.

Comparing the reliability prediction and \( \text{fRMSD} \) estimation performance of the \((P+S)^{\text{conc}}\) - *all* model i.e., trained using all the three levels of protein-pairs to the specific SCOP-level trained \((P+S)^{\text{conc}}\) - *fam*, \((P+S)^{\text{conc}}\) - *suf* and \((P+S)^{\text{conc}}\) - *fold* models the \((P+S)^{\text{conc}}\) - *all* trained model achieves the best or almost the best performance across different test sets. This suggests that there is little advantage to be gained by building models that are tuned to different levels of sequence similarity.

We also noticed that for the models trained on the different subsets of the data, the best performance for a particular subset is achieved by the model that was trained on the same subset, i.e., for a test residue-pair belonging to the *fam* subset, the best performance is achieved by a model trained on the *fam* subset as well (similar is the case for the *suf* and *fold* subsets).

**Additional Considerations**

We also experimented with variations in the input information, coding schemes and datasets. Our empirical results showed that the use of non-position specific information, decayed weighting function, cascading models, and SCOP-specific datasets did not improve the classification and estimation performance. These results are not included in this paper but are provided on our supplementary website for this work\(^2\).

**Position Independent Features** Motivated by the work in [104], we also derived a feature space using the BLOSUM62 matrix. These scores correspond to the position non-specific score for the various residue positions. The primary motivation for the use of the BLOSUM62 derived feature vectors was to improve the classification accuracy in cases where the target sequence does not have a sufficiently large number of homologous sequences in nr, and/or PSI-BLAST fails to compute a correct alignment for some segments of

\(^2\)http://bioinfo.cs.umn.edu/supplements/fRMSDPred/
the sequence [104]. By augmenting the input coding schemes to contain both PSSM- as well as BLOSUM62-based information, the SVM could potentially learn a model that is also partially based on the non-position specific information. This information will remain valid even in cases in which PSI-BLAST could not generate correct alignments.

Decay Weighted Features   In the coding schemes described above, the different positions within the womer contribute equally to the final similarity score. We also experimented with a linearly decaying weight function, such that the contribution of each position in $K^w_{c}(\pi, \pi')$ decreased linearly with respect to its distance from the central residue-pairs.

4.6 Case Study: Sequence Alignment

We study the effectiveness of using predicted $fRMSD$ scores for construction of a position-to-position scoring matrix between sequence pairs. This scoring matrix was then used by a dynamic-programming based alignment algorithm (Smith-Waterman local alignment [188]) to produce sequence alignments. We also derived position-to-position scoring matrices derived from a weighted combination of profile-based, predicted secondary structure and $fRMSD$-based scoring schemes.

In particular we use three position-specific scoring matrices: (i) prof which uses the PICASSO [72, 134] scoring function to compute similarity between two profile columns (described in Section 4.5.3), (ii) ss which uses a dot-product between the predicted secondary structure scores, (iii) frmsd which uses the predicted $fRMSD$ scores. The different alignments generated by the scoring matrices are referred by prof, ss, and frmsd.

For a given residue-pair $\pi(x_i, y_j)$, we use estimate the $fRMSD$ score, $fRMSD(x_i, y_j)$. Since this score is actually a distance, we convert it into a similarity score using the transformation: $\log(\alpha / fRMSD(x_i, y_j))$. This transformation assigns positive values to residue-pairs $\pi(x_i, y_j)$ having an estimated $fRMSD$ score that is less than $\alpha$. For the purposes of this study the $\alpha$ parameter was set to one, because we observed that the residue-pairs $\pi(x_i, y_j)$ with $fRMSD(x_i, y_j)$ score of less than one are more likely to be structurally aligned. This scoring scheme is referred as the frmsd.

Besides the above scoring schemes, we also investigated their combinations. We used a weighted combination of the profile-based, predicted secondary, and $fRMSD$-based scoring schemes to compute a similarity score for a residue pair $\pi(x_i, y_j)$. In this approach the similarity score for a residue-pair $\pi(x_i, y_j)$, using the $\mathcal{PF}_{pic}$ and frmsd scoring schemes is given by
$\theta \ast \frac{\mathcal{PF}_{pic}(i, j)}{\max P} + (1 - \theta) \ast \frac{frmsd(i, j)}{\max F}$ \hspace{1cm} (4.10)

where $\mathcal{PF}_{pic}(i, j)$ and $frmsd(i, j)$ represent the PICASSO and $fRMSD$ scores for $\pi(x_i, y_j)$, respectively. The value $\max P$ ($\max F$) is the maximum absolute value of all $\mathcal{PF}_{pic}$-based ($frmsd$-based) residue-pair scores between the sequences and is used to normalize the different scores prior to addition. The parameter $\theta$ defines the weighting for different parts of the scoring function after normalization. The optimal weight parameter $\theta$, was determined by varying $\theta$ from 0.0 to 1.0 with increments of 0.1. This parameter was optimized for the ce_ref dataset, and the same value was then used for the mus_ref dataset.

A similar approach is used to combine $\mathcal{PF}_{pic}$ with $ss$ and $frmsd$ with $ss$. In case of the $frmsd + \mathcal{PF}_{pic} + ss$ there are two weight parameters that need to be optimized.

We will denote the various combination schemes by just adding their individual components, e.g., $frmsd + \mathcal{PF}_{pic}$ will refer to the scheme that uses the scores obtained by $frmsd$ and $\mathcal{PF}_{pic}$.

### 4.6.1 Speedup Optimization

For a residue-pair, we can compute the PICASSO- and secondary structure-based scores using two and one dot-product operations, respectively. In comparison, the $fRMSD$ score needs $|SV|$ dot-product operations, where $|SV|$ is the number of support vectors determined by the $\epsilon$-SVR optimization method. Hence, the $frmsd$ alignment scheme has a complexity of at least $O(|SV|)$, which is significantly higher than that of the $\mathcal{PF}_{pic}$ and $ss$ alignment schemes. To reduce these computational requirements we developed two heuristic alignment methods that require the estimation of only a fraction of the total number of residue pairs.

#### Seeded Alignment

The first method combines the banded alignment approach and the seed alignment technique [69] and is performed in three steps. In the first step, we generate an initial alignment, referred to as the seed alignment, using the Smith-Waterman algorithm and the $\mathcal{PF}_{pic}$ + $ss$ scoring scheme. In the second step, we estimate the $fRMSD$ scores for all residue-pairs within a fixed number of residues from the seed alignment, i.e., a band around the seed alignment. Finally, in the third step, we compute the optimal local alignment in the restricted band around the initial seed alignment. The computed $frmsd$ alignment lies within a fixed band around the $\mathcal{PF}_{pic}$ + $ss$ alignment and will be effective if the original $frmsd$ alignment and the $\mathcal{PF}_{pic}$ + $ss$ alignments are not very far away from each other. The complexity of this method can be controlled by selecting bands of different sizes. We refer to this method as the seeded alignment technique.
4.6.2 Experimental Results - $f_{RMSD}$ Alignment

**Iterative Sampling Alignment**

The second method employs an iterative sampling procedure to optimize the speed of the $f_{RMSD}$ alignment. The basic idea is fairly similar to the seeded alignment. At iteration $i$, we estimate 1 out of $R_i$ $f_{RMSD}$ scores in the dynamic-programming matrix for those residue-pairs that lie within the banded region of size $K_i$ around the seed alignment generated in step $i - 1$. $K_i$ and $R_i$ denote the band size and the sampling rate at iteration $i$, respectively. Using the estimated $f_{RMSD}$ scores, an alignment is produced at step $i$ which serves as the seed alignment for step $i + 1$. The band size is reduced by half, whereas the sampling rate is doubled at each step (i.e., $R_i$ will be halved), effectively increasing the number of points in the dynamic-programming matrix to be estimated within a confined band. The first iteration can be assumed to have the initial seed as the main diagonal with a band size covering the entire dynamic-programming matrix.

**4.6.2 Experimental Results - $f_{RMSD}$ Alignment**

**Dataset Description**

We evaluate the accuracy of the alignment schemes on two datasets. The first dataset, referred to as the ce_ref dataset, was used in a previous study to assess the performance of different profile-profile scoring functions for aligning protein sequences [51]. The ce_ref dataset consists of 581 alignment pairs having high structural similarity but low sequence identity ($\leq 30\%$). The gold standard reference alignment was curated from a consensus of two structure alignment programs: FSSP [78] and CE [184]. The second dataset, referred to as the mus_ref dataset, was derived from the SCOP 1.57 database [136]. This dataset consists of 190 protein pairs with an average sequence identity of 9.6\%. Mustang [113] was used to generate the gold standard reference alignments.

To better analyze the performance of the different alignment methods, we segmented each dataset based on the pairwise sequence identities of the proteins that they contain. We segmented the ce_ref dataset into four groups, of sequence identities in the range of 6-12\%, 12-18\%, 18-24\%, and 24-30\% that contained 15, 140, 203, and 223 pairs of sequences, respectively. We segmented the mus_ref dataset into three groups, of sequence identities in the range of 0-6\%, 6-12\%, and 12-30\% that contained 76, 67, and 47 pairs of sequences, respectively. Note that the three groups of the mus_ref are highly correlated with the bottom three levels of the SCOP hierarchy, with most pairs in the first group belonging to the same fold but different superfamily, most pairs in the second group belonging to the same superfamily but different family, and most pairs in the third group belonging to the same family.
 Evaluation Metric

We evaluate the quality of the various alignment schemes by comparing the differences between the generated candidate alignment and the reference alignment generated from structural alignment programs [51, 178, 52]. As a measure of alignment quality, we use the Cline Shift score (CS) [30] to compare the reference alignments with the candidate alignments. The CS score is designed to penalize both under- and over-alignment and crediting the parts of the generated alignment that may be shifted by a few positions relative to the reference alignment [51, 30, 159]. The CS score ranges from a small negative value to 1.0, and is symmetric in nature. We also assessed the performance on the standard Modeler’s (precision) and Developer’s (recall) score [178], but found similar trends to the CS score and hence do not report the results here.

 Gap Modeling and Shift Parameters

For all the different scoring schemes, we use a local alignment framework with an affine gap model, and a zero-shift parameter [215] to maintain the necessary requirements for a good optimal alignment [69]. We optimize the gap modeling parameters (gap opening (go), gap extension (ge)), the zero shift value (zs), and weights on the individual scoring matrices for integrating them to obtain the highest quality alignments for each of the schemes. Having optimized the alignment parameters on the ce_ref dataset, we keep the alignment parameters unchanged for evaluation on the mus_ref dataset.

 Alignment Results

We performed a comprehensive study to evaluate the accuracy of the alignments obtained by the scoring scheme derived from the estimated \( f_{RMSD} \) values against those obtained by the \( prof \) and \( ss \) scoring schemes and their combinations. These results are summarized in Figures 4.3 and 4.4, which show the accuracy performance of the different scoring schemes on the ce_ref and mus_ref datasets, respectively. The alignment accuracy is assessed using the average CS scores across the entire dataset and at the different pairwise sequence identity segments. To better illustrate the differences between the schemes, the results are presented relative to the CS score obtained by the \( prof \) alignment and are shown on a log\(_2\) scale.

Analyzing the performance of the different scoring schemes we see that most of those that utilize predicted information about the protein structure (\( ss, f_{RMSD} \), and combinations involving them and \( prof \)) lead to substantial improvements over the \( prof \) scoring scheme for the low sequence identity segments. However, the relative advantage of these schemes somewhat diminishes for the segments that have higher pairwise sequence identities. In fact, in the case of the 12%–30% segment for mus_ref, most of these schemes perform worse than \( prof \). This result is not surprising, and confirms our earlier discussion in Section 4.2.1.
Comparing the ss and frmsd scoring schemes, we see that the latter achieves consistently and substantially better performance across the two datasets and sequence identity segments. For instance, for the first segment of ce_ref (sequence identities in the range of 6%–12%), frmsd’s CS score is 20% higher than that achieved by the ss scoring scheme. In the first segment of mus_ref dataset (sequence identity in the range of 0%–6%), frmsd’s CS score is 33% higher than achieved by the ss scoring scheme, and is 19% higher for the second segment (sequence identity in the range of 6%–12%).

Comparing most of the schemes based on frmsd and its combinations with the other scoring schemes we see that for the segments with low sequence identities they achieve the best results. Among them, the frmsd +prof scheme achieves the best results for ce_ref, whereas the frmsd +prof +ss performs the best for mus_ref. For the first segments of ce_ref and mus_ref, both of these schemes perform 6.1% and 27.8% better than prof +ss, respectively, which is the best non-frmsd-based scheme. Moreover, for many of these segments, the performance achieved by frmsd alone is comparable to that achieved by the prof +ss scheme. Also, comparing the results obtained by frmsd and frmsd +ss we see that by adding information about the predicted secondary structure the performance does improve. In the case of the segments with somewhat higher sequence identities, the relative advantage of frmsd +prof diminishes and becomes comparable to PFpic +ss.

Finally, comparing the overall performance of the various schemes on the ce_ref and mus_ref datasets, we see that frmsd +prof is the overall winner as it performs the best for ce_ref and similar to the best for mus_ref.
Comparison to Other Alignment Schemes  Since the ce_ref dataset has been previously used to evaluate the performance of various scoring schemes we can directly compare the results obtained here with those presented in [51]. In particular, according to that study, the best PSI-BLAST-profile based scheme achieved a CS score of 0.805, which is considerably lower than the CS scores of 0.854 and 0.845 obtained by the \textit{frmsd +prof} and \textit{prof + ss}, respectively.

Also, to ensure that the CS scores achieved by our schemes on the mus_ref dataset are reasonable, we compared them against the CS scores obtained by the state-of-the-art CONTRALIGN [44] and ProbCons [45] schemes. These schemes were run locally using the default parameters. CONTRALIGN and ProbCons achieved average CS scores of 0.197 and 0.174 across the 190 alignments, respectively. In comparison the \textit{frmsd} scheme achieved an average CS score of 0.299, whereas \textit{frmsd +prof} achieved an average CS score of 0.337.

Optimization Performance  We also performed a sequence of experiments to evaluate the extent to which the two runtime optimization methods discussed in Section 4.6.1 can reduce the number of positions whose \textit{fRMSD} needs to be estimated while still leading to high-quality alignments. These results are shown in Figure 4.5, which shows the CS scores obtained by the \textit{frmsd} scoring scheme on the ce_ref dataset as a function of the percentage of the residue-pairs whose \textit{fRMSD} scores were actually estimated. Also, the figure shows the average CS score achieved by the original (not sampled) \textit{frmsd} scheme.

These results show that both the seeded and iterative sampling procedures generate alignments close to the alignment generated from the original complete scheme. The average CS scores of the seeded and iterative sampling alignment by computing just 6\% of the original \textit{frmsd} matrix is 0.822 and 0.715, respectively. The average CS score of the original \textit{frmsd} scheme is 0.828. Hence, we get competitive scores by our sampling procedures for almost a 20 fold speedup. The seeded based technique shows better performance compared to the iterative sampling technique.

4.7 Conclusions  In this thesis we defined the \textit{fRMSD} estimation and the reliability prediction problems to capture the local structural similarity using only sequence-derived information. We developed a machine-learning approach for solving these problems by using a second-order exponential kernel function to encode profile and predicted secondary structure information into a kernel fusion framework. Our results showed that the \textit{fRMSD} values of any general residue-pairs can be predicted at a good level of accuracy.
Figure 4.5. Speedup using the Seeding and Sampling Alignment Procedure on the ce_ref dataset.
We believe that this lays the foundation for using estimated $f_{\text{RMSD}}$ values for producing sensitive and accurate sequence alignments. Our promising results shown on the aligned residue-pairs allows us to evaluate the quality of target-template alignments and refine them.

We also evaluated the effectiveness of using estimated $f_{\text{RMSD}}$ scores to aid in the alignment of protein sequences. Our results showed that the structural information encoded in these estimated scores are substantially better than the corresponding information in predicted secondary structures and when coupled with existing state-of-the-art profile scoring schemes, they lead to considerable improvements in aligning protein pairs with very low sequence identities.

Note, however sequence alignment may be one of the applications of our method. We would like to perceive our method to be used for protein modeling irrespective of the target difficulty (i.e whether it has an easy detectable known homolog). Our method, would be able to rank or assign the fragments from different set of templates per position which could be later assembled together using a fragment assembly method like TASSER [225], or Rosetta [169].
Local Structure and Function Prediction

5.1 Introduction

Determining the structural and functional characteristics of proteins is an important problem in computational biology. A wide variety of computational techniques have been developed to assign individual protein residues with property values in the form of a discrete label or continuous value. Familiar examples include secondary structure prediction [153, 104, 92] and solvent accessibility prediction [137, 152, 172]. Though specifics vary, the key feature of most methods is to use sequence conservation signals to generate predictions. From problem to problem, the amount of sequence information required to generate an accurate and general model may vary substantially.

Residue properties mostly take the form of either discrete label or continuous values and the task of assigning these properties is called protein sequence annotation.

Our work develops a generalized protein sequence annotation framework using kernel-based techniques. The generalized framework was motivated by the fact that there are countless problems in computational biology that use kernel-based methods [181]. As such, protein structure prediction problems at the residue-level followed a similar protocol. The key steps involved using sequence and sequence-derived information, feature extraction, use of a standard kernel function, and a cross-validation and evaluation framework. Our developed framework provides novice users to use any input information in the form of feature matrices and predict a discrete class label or estimate a continuous valued annotation for each residue.

Besides the generalized framework, this work also advocates the use of variable-width windows around
sequence positions. Depending on the problem, only rough information about distant sequence neighbors may be required for accurate predictions. We explore this issue by examining the performance trade-off between fine-grained near-neighbor and coarse-grained distant-neighbor information. We also study the performance of the normalized exponential kernel function developed in our previous studies [162] in comparison to the standard kernel function within the context of annotation problems.

As part of this work, we developed a protein sequence annotation toolkit, called ProSAT [164], that is applicable to any general annotation problem. We also developed a web-based interface called MONSTER [163], Minnesota prOteiN Sequence annotaTion servER to allow biologists to utilize these local structure and function prediction services easily. ProSAT was tested on a wide range of prediction problems including annotation for solvent accessibility [152, 172], local structure alphabet [99, 41], transmembrane helices [93], DNA-protein interaction sites [140], contact order [192], and disordered regions [29, 74]. We use previously established datasets that have pairwise sequence identity ranging from 25% to 40% for our evaluation. A report describing ProSAT and its empirical performance is available as part of a technical report1.

5.2 Problem Definitions

5.2.1 ProSAT Problem Definitions

The protein residue annotation problem is predicting a discrete label (using a classification framework) or a continuous value (using a regression framework) describing local structural and functional properties of proteins.

Given a protein sequence $X$ of length $n$, with it are associated derived features $F$, a $n \times d$ matrix where $d$ is the dimensionality of the feature space. The features associate with the $i$th residue $x_i$ are the $i$th row of the matrix $F$ denoted as $F_i$. When multiple types of features are considered, the $l$th feature matrix is specified by $F^l$. For example, we could use profiles as features in which case we a feature matrix $F$ could be defined as a position-specific scoring matrix, $p$ (previously detailed in Section 2.4.2 on page 22).

We capture local information around a residue $x_i$ using the $wme$r concept (See Section 2.4.1 on page 21). Specifically, the $(2w + 1)$-length subsequence around the central residue $x_i$ is referred to as a $wme$r and is denoted by $wme$(x_i), and is used to predict the discrete label or continuous value, $y_i$. The features associated with $wme$(x_i) are the rows of $F$, $F_{i-w}$ to $F_{i+w}$ and are denoted as $wme$(F_i).

5.2.2 Disorder Prediction

Some proteins contain regions which are intrinsically disordered in that their backbone shape may vary greatly over time and external conditions. A disordered region of a protein may have multiple binding partners and hence can take part in multiple biochemical processes in the cell which make them critical in performing various functions [48]. Accurate prediction of disordered regions can relieve some of the bottlenecks caused during high-throughput proteome analysis.

Several studies [170,205] have shown the differences in sequences for ordered and disordered regions. As such, a large number of computational approaches have been developed to predict the disordered segments using sequence information. Predicting disordered regions forms part of the biennial protein structure prediction experiment CASP\(^2\). Disorder region prediction methods mainly use physiochemical properties of the amino acids or evolutionary information. In particular IUPred [47] uses a pairwise energy function derived from amino acid composition, Poodle [74] employs a combination of different physiochemical properties as features for a SVM-based learning and prediction approach. Another disordered prediction tool, DISPro [29] utilizes a combination of evolutionary and sequence-derived features within a recurrent neural network.

5.2.3 Protein-DNA Interaction Site Prediction

When it is known that the function of a protein is to bind to DNA, it is highly desirable from an experimental point of view to know which parts of the protein are involved in the binding process. These interaction sites usually involve protein residues which come into contact with DNA and stabilize the complex due to favorable interactions with DNA. Sequence-based methods identify the most likely binding residues as the full structure of the protein is rarely known. Accurate methods that do so would allow an experimentalist to alter the protein behavior by mutating only a few residues.

The usual approach for a machine learning approach is to define a cutoff distance from DNA. If parts of a protein residue are within this cutoff, it is considered an interacting residue and is otherwise considered non-interacting, a binary classification problem. DISIS [140] uses support vector machines and a radial basis function kernel with PSSMs, predicted secondary structure, and predicted solvent accessibility as input features. This framework is directly comparable to our own along with neural network method of Ahmad and Sarai [2] which employs only PSSMs. Researchers have also utilized structure information such as the structural neighbors in DISPLAR [200] and the solvent accessibility using in the earlier work of Ahmad et al. [1].

\(^2\)http://predictioncenter.org
5.2.4 Transmembrane Helix Prediction

Proteins which span the cell membrane have proven to be quite difficult to crystallize in most cases and are generally too large for NMR studies. Computational methods to elucidate transmembrane protein structure are a quick means to obtain approximate topology. Many of these proteins are composed of an inter-cellular, extra-cellular, and membrane portions where the membrane portion contains primarily hydrophobic residues in helices. Accurately predicting these helix segments allows them to be excluded from function studies as they are usually not involved in the activity of the protein.

MEMSAT [93] in its most recent incarnation uses profile inputs to a neural network to predict whether residues in a transmembrane protein are part of a transmembrane helix, interior or exterior loop, or interior or exterior helix caps. Kernytsky and Rost have benchmarked a number of methods and maintain a server to compare the performance of new methods which we employ in our evaluation [108].

5.2.5 Local Structure Alphabets

The notion of local, recurring substructure in proteins has existed for many years primarily in the form of the secondary structure classifications. With the advent of fragment assembly methods for tertiary structure prediction [187], there has been increased interest in methods for predicting the backbone conformation of a fixed length section of protein. This extended local structure, usually a superset of traditional 3-state secondary structure, can be a significant first step towards full tertiary structure.

Many local structure alphabets have been generated by careful manual analysis of structures such as the alphabet of DSSP [97] while others have been derived through purely computational means. One such example are the Protein Blocks of de Brevern et al. [41] which were constructed through the use of self-organizing maps, a clustering technique. The method uses residue dihedral angles during clustering and attempts to account for order dependence between local structure elements which should improve predictability. Karchin et al. used neural nets to predict local structure for a variety of alphabets [99]. They found Protein Blocks to be the best choice according to their ‘bits saved per position,’ a measure of how much prediction improvement there is for the alphabet over simply predicting the most frequent character.

5.2.6 Relative Solvent Accessibility Prediction

Solvent accessibility determines the degree to which a residue in a protein structure can interact with a solvent molecule. This is important, as it can ascertain the local shape of protein based on whether the residue is buried/exposed. The residue-wise notion of solvent accessibility is defined by DSSP [97] by determining the
accessible surface area relative to the maximum possible surface area obtainable for the specific amino acid residue.

Predicting solvent accessibility can be formulated as a classification problem by defining buried or exposed classes by thresholding on the relative solvent accessibility value (normally 16% or 25%), and can also be a regression or density estimation problem of attempting to determine the percentage value using sequence information only. There are many methods available for solvent accessibility prediction that deploy a wide range of learning methods including neural networks [172], bi-recurrent neural networks [152], information theory statistics [137] and support vector machines [139] using the set of standard sequence derived features.

### 5.2.7 Residue-wise Contact Order Prediction

Pairs of residues are considered to be in contact if their $C_{\beta}$ atoms are within a threshold radius, generally 12 Å. Residue-wise contact order [111] is an average of the distance separation between contacting residues within a sphere of set threshold. It defines the extent to which a residue makes long-range contacts in native protein structure, and can be used to set up constraints in the overarching three-dimensional structure prediction problem, and explain protein-folding rates [151].

A support vector regression method [192] has used a combination of local sequence-derived information in the form of PSI-BLAST profiles [7] and predicted secondary structure information [92], and global information based on amino acid composition and molecular weight for good quality estimates of the residue-wise contact order value. Amongst other techniques, critical random networks [112] use PSI-BLAST profiles as a global descriptor for this estimation problem.

### 5.3 Methods and Algorithms

We approach the protein residue annotation problem by utilizing local sequence information around each residue in a supervised machine learning framework. We use support vector machines (SVM) [87, 207] in both classification and regression formulations to address the problem of annotating residues with discrete labels and continuous values respectively.

The task of assigning a label to the residue $x$ from one of the $K$ possible annotation labels is a typical multi-class classification problem. The general strategy is to build $K$ one-versus-rest binary SVM classification models that assign a residue to be in a particular class or not.

We use the error insensitive support vector regression $\epsilon$-SVR [207, 189] for learning a function $f(x)$ for estimation in case of determining a quantity, as in the case of solvent accessibility prediction problem.
In this work we focus on several key aspects related to the formulation and solution of the classification and regression problems. In particular we explore different types of sequence information associated with the residues, develop efficient ways to encode this information to form fixed length feature vectors, and design sensitive kernel functions to capture the similarity between residues in the feature spaces.

### 5.3.1 Sequence-based Information

PROSAT can use any general user-supplied features. In our empirical evaluation for a given protein \( X \) of length \( n \) we encode the sequence information using PSI-BLAST position specific scoring matrices, predicted secondary structure, and position independent scoring matrices like BLOSUM62. These feature matrices are referred to as \( \mathcal{P} \), \( \mathcal{S} \), and \( \mathcal{B} \), respectively.

For a sequence of length \( n \), PSI-BLAST generates a position-specific scoring matrix \( \mathcal{P} \) of dimensions \( n \times 20 \), where the 20 columns of the matrix correspond to the twenty amino acids (described in Section 2.4.2 on page 22).

We use YASSPP [104] to predict secondary structure and generate a position-specific secondary structure matrices. For a length \( n \) sequence, the result is \( \mathcal{S} \), a \( n \times 3 \) feature matrix. Predicted secondary structure is an example of a local structure alphabet and plays a critical role in protein structure prediction. YASSPP [104] has an identical framework to PROSAT and is one of the best performing secondary structure prediction methods with a reported \( Q_3 \) accuracy of 80%.

BLOSUM62 provides a less computationally expensive feature of protein sequences may be obtained from a position independent scoring matrix. This feature matrix is used to improve the classification accuracy in cases where a sequence does not have a sufficiently large number of homologous sequences. To make effective use of PROSAT’s capabilities we create a \( n \times 20 \) feature matrix, \( \mathcal{B} \) where each row of the matrix is a copy of the BLOSUM62 row corresponding to the amino acid at that position in the sequence.

### 5.3.2 Kernel Functions and Coding Schemes

For a pair of sequences \( X \) and \( Y \), let a specific set of derived features for the sequences be matrices \( F \) and \( G \), respectively. To simplify notation we use \( F_i \) to indicate the \( i \)th row of matrix \( F \), which corresponds to the features associated with the \( i \)th residue of \( X \). A kernel function computes a similarity between two objects and selection of an appropriate kernel function for a problem is key to the effectiveness of support vector machine learning. We consider several individual kernels of interest and then proceed to describe combinations of kernels used in this study.
Our first contribution is a two-parameter linear window-kernel, denoted by $\mathcal{W}_{w,f}$ which computes the similarity between two $w$mers, $w$mer($x_i$) and $w$mer($y_j$) according to their features $w$mer($F_i$) and $w$mer($G_j$), respectively. The kernel function is defined as

$$
\mathcal{W}_{w,f}(x_i, y_j) = \sum_{k=-f}^{f} (F_{i+k}, G_{j+k}) + \langle \sum_{k=f+1}^{w} F_{i+k}, \sum_{k=f+1}^{w} G_{j+k} \rangle + \langle \sum_{k=-w}^{-f-1} F_{i+k}, \sum_{k=-w}^{-f-1} G_{i+k} \rangle. 
$$

(5.1)

The parameter $w$ governs the size of the $w$mer considered in computing the kernel while $f$ offers control over the fine-grained versus coarse-grained sections of the window. Rows within $\pm f$ contribute an individual dot product to the total similarity while rows outside this range are first summed and then their dot product is taken. In all cases $f \leq w$ and as $f$ approaches $w$, the window kernel becomes simply a sum of the dot products, the most fine-grained similarity measure considered.

The rationale behind this kernel design is that some problems may require only approximate information for sequence neighbors which are far away from the central residue while nearby sequence neighbors are more important. Specifying $f \ll w$ merges these distant neighbors into only a coarse contribution to the overall similarity, as it only accounts for compositional information and not the specific positions where these features occur. The window kernel is defined as a dot-product, which makes it equivalent to linear kernel with a feature encoding scheme that takes into account the two variable parameters, $w$ and $f$. Hence, we can embed the dot-product based $\mathcal{W}$ within other complex kernel functions.

Another individual kernel we use extensively is the second order exponential kernel, $K_{nsoe}$, developed in our earlier works for secondary structure and local structure information prediction [104, 162], and also described in Section 4.4.3 on page 77. Given any base kernel function $\mathcal{K}$, we define $K^2$ as

$$
K^2(x, y) = \mathcal{K}(x, y) + (\mathcal{K}(x, y))^2.
$$

(5.2)

which is a second-order kernel in that it computes pairwise interactions between the elements $x$ and $y$. We then define $K_{nsoe}$ as

$$
K_{nsoe}(x, y) = \exp \left( 1 + \frac{K^2(x, y)}{\sqrt{K^2(x, x)K^2(y, y)}} \right)
$$

(5.3)

which normalizes $K^2$ and embeds it into an exponential space.

We also use the standard radial basis kernel function ($rbf$), defined for some parameter $\gamma$ by $K_{rbf}(x, y) = \exp(-\gamma ||x - y||^2)$. In our studies we notice that the classification and regression performance generally improves using unit length normalized vectors. By setting the $\gamma$ parameter and using normalized unit length
vectors it can be shown that the standard $r_{bf}$ kernel is equivalent (up to a scaling factor) to a first order exponential kernel which is obtained by replacing $\mathcal{K}^2(x, y)$ with only the first-order term as $\mathcal{K}(x, y)$ in Equation 5.2, and plugging this modified $\mathcal{K}^2(x, y)$ in the normalization framework of Equation 5.3.

We investigate the performance of $\mathcal{K}^{nsoe}$ and $\mathcal{K}^{r_{bf}}$ kernels with $\mathcal{W}_{w, f}$ as the base kernel. From here forward, we denote the $nsoe$ to be the kernel $\mathcal{K}^{nsoe}$ using the $\mathcal{W}_{w, f}$ as the base, $r_{bf}$ to be the kernel $\mathcal{K}^{r_{bf}}$ using the normalized form with $\mathcal{W}_{w, f}$ as the base, and $lin$ to be the base linear kernel $\mathcal{W}_{w, f}$.

We also investigate the use of fusion kernels which are generated via a linear combination of other kernels. In our case, we use a fusion of second-order exponential kernels on different features of a protein sequence. Considering two sequences with features $F_l$ and $G_l$ for $l = 1, \ldots, k$, our fusion kernel is defined

$$
\mathcal{K}^{fusion}(x_i, y_j) = \sum_{l=1}^{k} \omega_l \mathcal{K}^{nsoe}(F_l^i, G_l^j)
$$

(5.4)

where the weights $\omega_l$ are supplied by the user. In most cases, these weights are equal but they may be altered according to domain-specific information.

### Table 5.1. Problem-specific Datasets.

<table>
<thead>
<tr>
<th>Problem</th>
<th>Source</th>
<th>#C</th>
<th>#Seq</th>
<th>#Res</th>
<th>#CV</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disorder Prediction</td>
<td>DisPro [29]</td>
<td>2</td>
<td>723</td>
<td>215612</td>
<td>10</td>
<td>30</td>
</tr>
<tr>
<td>Protein-DNA Site</td>
<td>DISIS [140]</td>
<td>2</td>
<td>693</td>
<td>127240</td>
<td>3</td>
<td>20</td>
</tr>
<tr>
<td>Residue-wise Contact</td>
<td>SVM [192]</td>
<td>$\infty$</td>
<td>680</td>
<td>120421</td>
<td>15</td>
<td>40</td>
</tr>
<tr>
<td>Solvent Accessibility</td>
<td>RS126 [173]</td>
<td>$\infty$</td>
<td>126</td>
<td>23356</td>
<td>7</td>
<td>25</td>
</tr>
<tr>
<td>Solvent Accessibility</td>
<td>RS126 [173]</td>
<td>2</td>
<td>126</td>
<td>23356</td>
<td>7</td>
<td>25</td>
</tr>
<tr>
<td>Local Structure</td>
<td>Pronet [142]</td>
<td>16</td>
<td>1600</td>
<td>286238</td>
<td>3</td>
<td>40</td>
</tr>
<tr>
<td>Transmembrane Helix</td>
<td>Phobius [98]</td>
<td>4</td>
<td>247</td>
<td>95025</td>
<td>3</td>
<td>80*</td>
</tr>
<tr>
<td>Transmembrane Helix</td>
<td>Static Test [108]</td>
<td>2</td>
<td>2247</td>
<td>238084</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

#C, #Seq, #Res, #CV, and % denote the number of classes, sequences, residues, number of cross validation folds, and the maximum pairwise sequence identity between the sequences, respectively. $\infty$ represents the regression problem. *Even though the percent identity between sequence-pairs is high, the dataset is used only for training and an independent evaluation is performed on the static benchmark [108].

### 5.4 Experimental Results

#### 5.4.1 Datasets

Our empirical evaluations are performed for different sequence annotation problems on previously defined datasets. Table 5.1 presents information regarding the source and key features of different datasets used in
our cross validation and comparative studies.

The general protocol we used for evaluating the different parameters, and features, as well as comparing to previously established studies remained fairly consistent across the different problems. In particular we used a $n$-fold cross validation methodology, where $1/n$th of the database in consideration was used for testing and the remaining dataset was used for training, with the experiment being repeated $n$ times. The number of cross validation was set based on the method that had used the same dataset previously for comparative purposes. This allows us ease of comparison to state-of-the-art methods.

The datasets were used in previous studies, and we ensured that the pairwise sequence identities between the different subsets of data in the different cross validation folds were less than 40%. Even though, the average sequence identity for the transmembrane-helix dataset was 80%, the dataset was only used to train a model for a blind independent evaluation on the static benchmark [108].

### 5.4.2 Evaluation Methodology

We measure the quality of the methods using the standard receiver operating characteristic (ROC) scores. We also compute other standard statistics, including precision as $TP/(TP+FP)$, and recall as $TP/(TP+FN)$. We also evaluate the accuracy of $K$-way multiclass classification by as $Q_K = (\sum_{i=1}^{K} T_{Pi})/(Total\ Residues)$. Here, TP, FP, TN, FN denote the standard true positives, false positives, true negatives, and false negatives, respectively. We also compute the $F_1$ score given as $2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$.

The ROC score serves as a good quality measure in case of unbalanced classes, where measuring the accuracy or $Q_K$, especially in case of binary classification model may be skewed by predicting a particular class with larger number of instances. In such cases, it is essential to observe the precision and recall values, which penalize the classifiers for under-prediction as well as over-prediction. The $F_1$ score is a weighted average of precision and recall lying between 0 and 1, and also is a good measure for different classification problems.

The regression performance is assessed by computing the standard Pearson correlation coefficient ($CC$) between the predicted and observed true values for every protein in the datasets. We also compute the root mean square error $rmse$ between the predicted and observed values for every proteins. The results reported are averaged across the different proteins and cross validation steps. For the $rmse$ metric, a lower score implies a better quality prediction.

For the best performing models, we also report the $errsig$ rate as the significant difference margin for $Q_K$ and $CC$ scores (to distinguish between two methods). $errsig$ is defined as the standard deviation divided by the square root of the number of proteins ($\sigma/\sqrt{N}$), and can help us assess how significant the differences
5.4.3 ProSAT Prediction Performance

For all the problems, we perform a comprehensive set of experiments encompassing a range of parameters, to determine the kernel type, features, and \( \mathcal{W} \) parameters (i.e., \( w \) and \( f \)).

Disorder Prediction Performance

Table 5.2 shows the binary classification performance measured using the ROC and \( F_1 \) scores achieved on the disorder dataset after a ten fold cross validation experiment.

Comparing the ROC performance of the \( P^{nsoe} \), \( P^{rbf} \), and \( P^{lin} \) models across different values of \( w \) and \( f \) used for parametrization of the base kernel (\( \mathcal{W} \)), we observe that the \( nsoe \) kernel shows superior performance to the \( lin \) kernel and slightly better performance compared to the normalized \( rbf \) kernel used in this study.

Comparing the characteristics of the different features keeping the kernel fixed to \( nsoe \), we can notice that use of \( P \) gives better classification performance compared to the \( S \) and \( B \) features. However, integrating features i.e., use of fusion kernels with \( P B \), \( P S \), and \( P S B \) tends to improve the disorder prediction over the kernels that use only one set of features, with the best results achieved by a combination of all three features.

An interesting trend can be observed for the \( B^{nsoe} \) and \( S^{nsoe} \) results. As we increase the \( w \) parameter, keeping the \( f \) parameter fixed to a low value of one or three, the percentage increase in the ROC value for the \( B \) features is higher, which suggests that the \( B \) features are more suited to be adopted in a coarse setting.

The best performing fusion kernel shows comparable performance to DisPro [29] that uses a bi-recurrent neural network to encapsulate profile, secondary structure and relative solvent accessibility information. DisPro [29] achieves a ROC score of 0.878 for the dataset.

Contact Order Performance

In Table 5.3 we present the regression performance for estimating the residue wise contact order. These results are evaluated by computing the correlation coefficient and rmse values averaged across the different proteins in the dataset.

Analyzing the effect of the \( w \) and \( f \) parameters for estimating the residue-wise contact order values, we observe that a model trained with \( f < w \) generally shows better \( CC \) and \( rmse \) values. The best models as measured by the \( CC \) scores after 15-fold cross-validation are highlighted in Table 5.3. A model with
Table 5.2. Classification Performance on the Disorder Dataset.

<table>
<thead>
<tr>
<th>w</th>
<th>f = 1</th>
<th>f = 3</th>
<th>f = 5</th>
<th>f = 7</th>
<th>f = 9</th>
<th>f = 11</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ROC F1</td>
<td>ROC F1</td>
<td>ROC F1</td>
<td>ROC F1</td>
<td>ROC F1</td>
<td>ROC F1</td>
</tr>
<tr>
<td>$\rho_{\text{lin}}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.775 0.312</td>
<td>0.800 0.350</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>0.815 0.366</td>
<td>0.817 0.380</td>
<td>0.816 0.384</td>
<td>0.816 0.384</td>
<td>0.824 0.404</td>
<td>0.823 0.403</td>
</tr>
<tr>
<td>11</td>
<td>0.821 0.378</td>
<td>0.826 0.391</td>
<td>0.828 0.396</td>
<td>0.826 0.400</td>
<td>0.824 0.404</td>
<td>0.823 0.403</td>
</tr>
<tr>
<td>13</td>
<td>0.823 0.384</td>
<td>0.829 0.398</td>
<td>0.832* 0.405</td>
<td>0.830 0.404</td>
<td>0.828 0.407</td>
<td>0.826 0.409</td>
</tr>
<tr>
<td>$\rho_{rbf}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.811 0.370</td>
<td>0.811 0.369</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>0.845 0.442</td>
<td>0.849 0.450</td>
<td>0.848 0.445</td>
<td>0.845 0.442</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>11</td>
<td>0.848 0.464</td>
<td>0.855 0.478</td>
<td>0.858 0.482</td>
<td>0.858 0.480</td>
<td>0.855 0.470</td>
<td>0.853 0.468</td>
</tr>
<tr>
<td>13</td>
<td>0.848 0.473</td>
<td>0.855 0.484</td>
<td>0.859 0.490</td>
<td>0.861* 0.492</td>
<td>0.860 0.487</td>
<td>0.857 0.478</td>
</tr>
<tr>
<td>$\rho_{nsoe}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.815 0.377</td>
<td>0.816 0.379</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>0.847 0.446</td>
<td>0.852 0.461</td>
<td>0.852 0.454</td>
<td>0.851 0.454</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>11</td>
<td>0.848 0.469</td>
<td>0.856 0.482</td>
<td>0.860 0.491</td>
<td>0.862 0.491</td>
<td>0.861 0.485</td>
<td>0.862 0.485</td>
</tr>
<tr>
<td>13</td>
<td>0.847 0.473</td>
<td>0.856 0.485</td>
<td>0.861 0.491</td>
<td>0.864 0.495</td>
<td>0.865* 0.494</td>
<td>0.864 0.492</td>
</tr>
<tr>
<td>$\mathcal{B}_{\text{nsoe}}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.753 0.314</td>
<td>0.752 0.312</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>0.810 0.427</td>
<td>0.815 0.434</td>
<td>0.816 0.435</td>
<td>0.815 0.429</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>11</td>
<td>0.820 0.459</td>
<td>0.827 0.465</td>
<td>0.831 0.468</td>
<td>0.832 0.472</td>
<td>0.832 0.472</td>
<td>0.831 0.471</td>
</tr>
<tr>
<td>13</td>
<td>0.821 0.465</td>
<td>0.829 0.469</td>
<td>0.833 0.473</td>
<td>0.835 0.473</td>
<td>0.836 0.476</td>
<td>0.836 0.476</td>
</tr>
<tr>
<td>$\mathcal{S}_{\text{nsoe}}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.786 0.332</td>
<td>0.791 0.342</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>0.806 0.387</td>
<td>0.812 0.397</td>
<td>0.814 0.395</td>
<td>0.814 0.389</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>11</td>
<td>0.811 0.417</td>
<td>0.818 0.424</td>
<td>0.821 0.426</td>
<td>0.823 0.427</td>
<td>0.822 0.420</td>
<td>0.821 0.415</td>
</tr>
<tr>
<td>13</td>
<td>0.812 0.434</td>
<td>0.821 0.438</td>
<td>0.825 0.436</td>
<td>0.827* 0.437</td>
<td>0.827 0.434</td>
<td>0.826 0.428</td>
</tr>
<tr>
<td>$\rho_{\mathcal{B}_{\text{nsoe}}}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.825 0.399</td>
<td>0.824 0.395</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>0.862 0.487</td>
<td>0.865 0.491</td>
<td>0.865 0.487</td>
<td>0.863 0.481</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>11</td>
<td>0.864 0.502</td>
<td>0.869 0.509</td>
<td>0.872 0.513</td>
<td>0.873 0.514</td>
<td>0.873 0.513</td>
<td>0.873 0.510</td>
</tr>
<tr>
<td>13</td>
<td>0.863 0.509</td>
<td>0.869 0.514</td>
<td>0.873 0.517</td>
<td>0.875 0.518</td>
<td>0.876 0.518</td>
<td>0.876 0.519</td>
</tr>
<tr>
<td>$\rho_{\mathcal{S}_{\text{nsoe}}}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.836 0.418</td>
<td>0.838 0.423</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>0.860 0.472</td>
<td>0.862 0.476</td>
<td>0.860 0.473</td>
<td>0.859 0.468</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>11</td>
<td>0.861 0.490</td>
<td>0.867 0.496</td>
<td>0.868 0.498</td>
<td>0.868 0.495</td>
<td>0.866 0.488</td>
<td>0.865 0.485</td>
</tr>
<tr>
<td>13</td>
<td>0.860 0.497</td>
<td>0.867 0.503</td>
<td>0.870 0.503</td>
<td>0.871* 0.503</td>
<td>0.870 0.498</td>
<td>0.868 0.492</td>
</tr>
<tr>
<td>$\rho_{\mathcal{S}<em>{\mathcal{B}</em>{\text{nsoe}}}}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.842 0.428</td>
<td>0.841 0.428</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>0.869 0.497</td>
<td>0.870 0.499</td>
<td>0.869 0.494</td>
<td>0.867 0.489</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>11</td>
<td>0.871 0.516</td>
<td>0.875 0.518</td>
<td>0.877 0.517</td>
<td>0.877 0.512</td>
<td>0.874 0.508</td>
<td>0.873 0.507</td>
</tr>
<tr>
<td>13</td>
<td>0.869 0.519</td>
<td>0.875 0.522</td>
<td>0.878 0.521</td>
<td>0.879* 0.519</td>
<td>0.879 0.518</td>
<td>0.876 0.514</td>
</tr>
</tbody>
</table>

DISPro [29] reports a ROC score of 0.878. The numbers in bold show the best models for a fixed w parameter, as measured by ROC. $\rho$, $\mathcal{B}$, and $\mathcal{S}$ represent the PSI-BLAST profile, BLOSUM62, and YASSPP scoring matrices, respectively. nsoe, rbf, and lin represent the three different kernels studied using the $W_{w,f}$ as the base kernel. * denotes the best classification results in the sub-tables, and ** denotes the best classification results achieved on this dataset. For the best model we report a $Q_5$ accuracy of 84.60% with an errsig rate of 0.33.
Table 5.3. Residue-wise Contact Order Estimation Performance

<table>
<thead>
<tr>
<th>$w$</th>
<th>$f = 1$</th>
<th>$f = 3$</th>
<th>$f = 5$</th>
<th>$f = 7$</th>
<th>$f = 9$</th>
<th>$f = 11$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CC rmse</td>
<td>CC rmse</td>
<td>CC rmse</td>
<td>CC rmse</td>
<td>CC rmse</td>
<td>CC rmse</td>
</tr>
<tr>
<td>$p_s$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.683 0.720</td>
<td><strong>0.686 0.718</strong></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>0.685 0.714</td>
<td>0.694 0.707</td>
<td>0.702 0.698</td>
<td><strong>0.703 0.697</strong></td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>11</td>
<td>0.683 0.713</td>
<td>0.695 0.703</td>
<td>0.704 0.694</td>
<td><strong>0.705 0.692</strong></td>
<td>0.704 0.691</td>
<td>0.704 0.692</td>
</tr>
<tr>
<td>15</td>
<td>0.680 0.714</td>
<td>0.694 0.703</td>
<td>0.703 0.693</td>
<td><strong>0.704 0.691</strong></td>
<td>0.704 0.690</td>
<td>0.704 0.690</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$p_s$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.703 0.699</td>
<td><strong>0.707 0.696</strong></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>0.709 0.687</td>
<td>0.716 0.680</td>
<td><strong>0.721 0.677</strong></td>
<td>0.720 0.677</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>11</td>
<td>0.707 0.686</td>
<td>0.718 0.676</td>
<td><em><em>0.723</em> 0.671</em>*</td>
<td>0.722 0.671</td>
<td>0.720 0.672</td>
<td>0.718 0.673</td>
</tr>
<tr>
<td>15</td>
<td>0.704 0.686</td>
<td>0.716 0.675</td>
<td><strong>0.723 0.669</strong></td>
<td>0.723 0.669</td>
<td>0.721 0.669</td>
<td>0.720 0.670</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$p_s$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.704 0.696</td>
<td><strong>0.708 0.692</strong></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>0.712 0.683</td>
<td>0.719 0.677</td>
<td><strong>0.723 0.672</strong></td>
<td>.722 0.672</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>11</td>
<td>0.711 0.681</td>
<td>0.720 0.673</td>
<td><strong>0.725 0.667</strong></td>
<td>0.725 0.666</td>
<td>0.724 0.666</td>
<td>0.722 0.667</td>
</tr>
<tr>
<td>15</td>
<td>0.709 0.680</td>
<td>0.719 0.672</td>
<td><strong>0.726</strong></td>
<td>0.665</td>
<td>0.726 0.664</td>
<td>0.725 0.664</td>
</tr>
</tbody>
</table>

$CC$ and rmse denotes the average correlation coefficient and rmse values. The numbers in bold show the best models as measured by $CC$ for a fixed $w$ parameter. $p_s$ and $s$ represent the PSI-BLAST profile and YASSPP scoring matrices, respectively. $nsoe$, $rbf$, and $lin$ represent the three different kernels studied using the $W_w, f$ as the base kernel. * denotes the best regression results in the sub-tables, and ** denotes the best regression results achieved on this dataset. For the best results the errsig rate for the $CC$ values is 0.003. The published results [192] uses the default rbf kernel to give $CC = 0.600$ and $rmse = 0.78$.

The best estimation performance achieved by our $\epsilon$-SVR based learner uses a fusion of the $p_s$ and $s$ feature matrices and improves $CC$ by 21%, and $rmse$ value by 17% over the $\epsilon$-SVR technique of Song and Barrage [192]. Their method uses the standard rbf kernel with similar local sequence-derived amino acid and predicted secondary structure features. The major improvement of our method can be attributed to our fusion-based kernel setting with efficient encoding, and the normalization introduced in Equation 5.3. In our setting, using the default parameters for the $\gamma$, regression tube, and regularization parameters always lead to over-fitting of the data with the original rbf kernel. This trend has been noted in our previous studies [158].

We also tested the methods using only the $p_s$ features, and addition of $B$ features but did not see a significant improvement in the contact order estimation results and hence, do not report these results here.

**Transmembrane Helix Performance**

To predict transmembrane proteins helices with ProSAT, we set up a multi-class classification problem to differentiate between the helical and non-helical regions of transmembrane proteins. In particular, to determine the orientation and topological structure of the helices we used a dataset that annotated the intermediate equivalent $CC$ values but having a lower $f$ value is considered better because of the reduced dimensionality achieved by such models.
5.4.3 PROSAT Prediction Performance

Table 5.4. Classification Performance on the Transmembrane Helix Dataset.

<table>
<thead>
<tr>
<th>w</th>
<th>Method</th>
<th>Q2</th>
<th>RC</th>
<th>PC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>pnsoe</td>
<td>84</td>
<td>81</td>
<td>87</td>
</tr>
<tr>
<td>5</td>
<td>MEMSAT3</td>
<td>83</td>
<td>78</td>
<td>88</td>
</tr>
<tr>
<td>7</td>
<td>TMHMM1</td>
<td>80</td>
<td>68</td>
<td>81</td>
</tr>
<tr>
<td>9</td>
<td>PHDpsihtm08</td>
<td>80</td>
<td>76</td>
<td>83</td>
</tr>
<tr>
<td></td>
<td>HMMTOP2</td>
<td>80</td>
<td>69</td>
<td>89</td>
</tr>
<tr>
<td></td>
<td>PHDhtm08</td>
<td>78</td>
<td>76</td>
<td>82</td>
</tr>
</tbody>
</table>

RC and PC denote recall and precision, respectively. The numbers in bold show the best models for a fixed \( w \) parameter, as measured by \( Q_4 \) accuracy score, and ** denotes the best classification results achieved on this dataset. Results for MEMSAT3 [93] and \( p_{nsoe} \) were obtained by evaluating it on the TMH static benchmark [108] and submitting the results of prediction to the server. We use the \( p_{nsoe} \) kernel with \( w = f = 7 \). All the other results were obtained from the TMH static benchmark evaluation web-site.

We performed a three-fold cross validation study for the four-way multi-class classification problem on a dataset consisting of only transmembrane proteins [98]. Note that our training set makes it difficult to develop models for differentiating between globular and transmembrane proteins, as well as signal-peptide proteins as highlighted by Phobius [98], which uses a combination of hidden Markov models and neural networks for discriminating between the different residues.

Table 5.4 (a) shows the classification performance evaluated using the \( Q_4 \) accuracy and ROC scores for the \( p_{nsoe} \) kernel. Based on the classification performance metrics, we see that the better models have a finer representation of the \( w \)-mers, i.e., where \( w = f \).

To obtain the predictions for sequences in the static benchmark [108] we used the \( p_{nsoe} \) kernel with \( w \) and \( f \) parameters set to 7. We used all the 247 transmembrane proteins available in the Phobius dataset to build a four-way classification model, and annotated the 2247 sequences present in the static benchmark (which provides independent evaluation). For evaluation, we used a mapping from four to two classes rather than building a binary classification model. Residues marked as helices were mapped to helices while all others, including intermediate residues, were mapped to non-helices. One of the best membrane-helix prediction program, MEMSAT3 [93], was also evaluated for comparison purposes as these results were not reported in the static benchmark. MEMSAT3 also annotates residues into multiple classes: outside/inside loop, outside/inside helix cap, and internal helix. Loops were mapped to non-helices while other were mapped to helices.

Table 5.4 (b) shows some of the best performing schemes in comparison to our prediction method, denoted \( p_{nsoe} \), evaluated by an independent server on the static benchmark (We do not have the true predictions available for these sequences). We obtain the best \( Q_2 \) accuracy evaluated on a per-residue basis, and have

---

3 Static Benchmark for testing Transmembrane helix prediction at http://cubic.bioc.columbia.edu/services/tmh_benchmark/index.html
5.4.3 PROSAT Prediction Performance

Table 5.5. Relative Solvent Accessibility Class Prediction and Regression Performance.

<table>
<thead>
<tr>
<th>Cutoff % Method</th>
<th>0% Regression</th>
<th>5% Regression</th>
<th>16% Regression</th>
<th>Regression</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Q2 ROC F1</td>
<td>Q2 ROC F1</td>
<td>Q2 ROC F1</td>
<td>CC rmse</td>
</tr>
<tr>
<td>( P_{rbf} ), ( w, f = 3 )</td>
<td>87.0 0.845 0.486</td>
<td>79.9 0.855 0.664</td>
<td>78.0 0.855 0.755</td>
<td>0.648 0.211</td>
</tr>
<tr>
<td>( P_{rbf} ), ( w, f = 5 )</td>
<td>87.1 0.845 0.491</td>
<td>80.4 0.857 0.670</td>
<td>78.3 0.857 0.758</td>
<td>0.654 0.209</td>
</tr>
<tr>
<td>( P_{rbf} ), ( w, f = 7 )</td>
<td>87.1 0.844 0.491</td>
<td>80.2 0.856 0.668</td>
<td>78.4 0.856 0.758</td>
<td>0.653 0.209</td>
</tr>
<tr>
<td>( P_{rbf} ), ( w, f = 9 )</td>
<td>86.9 0.843 0.487</td>
<td>80.3 0.855 0.667</td>
<td>78.3 0.855 0.756</td>
<td>0.654 0.208</td>
</tr>
<tr>
<td>( P_{rbf} ), ( w, f = 11 )</td>
<td>87.2 0.843 0.486</td>
<td>80.2 0.855 0.666</td>
<td>78.3 0.854 0.756</td>
<td>0.654 0.208</td>
</tr>
<tr>
<td>( P_{nsoe} ), ( w, f = 3 )</td>
<td>87.5 0.845 0.491</td>
<td>80.2 0.857 0.669</td>
<td>78.5 0.858 0.758</td>
<td>0.641 0.211</td>
</tr>
<tr>
<td>( P_{nsoe} ), ( w, f = 5 )</td>
<td>87.6 0.847 0.494</td>
<td>80.8 0.860 0.671</td>
<td>78.7* 0.861 0.762</td>
<td>0.647 0.209</td>
</tr>
<tr>
<td>( P_{nsoe} ), ( w, f = 7 )</td>
<td>87.7* 0.846 0.493</td>
<td>81.0* 0.859 0.670</td>
<td>78.6 0.861 0.760</td>
<td>0.646 0.210</td>
</tr>
<tr>
<td>( P_{nsoe} ), ( w, f = 9 )</td>
<td>87.7 0.846 0.493</td>
<td>80.9 0.859 0.670</td>
<td>78.5 0.860 0.760</td>
<td>0.648 0.209</td>
</tr>
<tr>
<td>( P_{nsoe} ), ( w, f = 11 )</td>
<td>87.7 0.846 0.494</td>
<td>80.9 0.859 0.670</td>
<td>78.5 0.859 0.760</td>
<td>0.650 0.209</td>
</tr>
<tr>
<td>1-stage SVM</td>
<td>86.2 - -</td>
<td>79.8 - -</td>
<td>77.8 - -</td>
<td>- -</td>
</tr>
</tbody>
</table>

The cutoff % is in terms of relative accessible solvent area and determines which residues are exposed (above the cutoff) and buried (at or below the cutoff). The one-stage SVM is that of Kim and Park [109]. \( Q_2 \) measures are reported by these two methods but not ROC or F1 measures. CC and rmse denotes the average correlation coefficient and rmse values. We observed the best classification and regression performance by setting \( w = f \). The numbers in bold show the best models for a \( w, f \) parameter.

Good precision and recall scores as well.

Solvent Accessibility Performance

We approached solvent accessibility as both a labeling problem and a regression problem. For labeling, we chose varying cutoff thresholds to define each residue as either buried if at or below the threshold or exposed if above the threshold. For regression, PROSAT was used to generate continuous valued estimates of each residues relative accessible surface area. We explored \( P_{rbf} \) and also \( P_{nsoe} \) but restricted the study to models where \( w = f \). In each case, 7-fold cross-validation was performed on the full RS126 dataset.

The results for both classification and regression are shown in Table 5.5 along with a leading method for solvent accessibility prediction [109]. The general trend appears to be that prediction performance by \( P_{rbf} \) is slightly exceeded by \( P_{nsoe} \). The best window size appears to be \( w = 7 \) for both our kernels for prediction. Both kernels exceed the performance of the previously published SVM method which uses an \( rbf \) kernel with window length of 15 and profiles with some predicted local structure as inputs. For regression, the \( P_{rbf} \) outperforms \( P_{nsoe} \) and increasing window size improves the performance.
Table 5.6. Classification Performance on the Protein-DNA Interaction Site Prediction.

<table>
<thead>
<tr>
<th>w</th>
<th>f = 1</th>
<th>f = 3</th>
<th>f = 5</th>
<th>f = 7</th>
<th>f = 9</th>
<th>f = 11</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROC F1</td>
<td>ROC F1</td>
<td>ROC F1</td>
<td>ROC F1</td>
<td>ROC F1</td>
<td>ROC F1</td>
<td>ROC F1</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>p</td>
<td>3</td>
<td>0.743</td>
<td>0.452</td>
<td>0.745</td>
<td>0.453</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>0.738</td>
<td>0.446</td>
<td>0.743</td>
<td>0.449</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>0.712</td>
<td>0.421</td>
<td>0.728</td>
<td>0.438</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>0.685</td>
<td>0.398</td>
<td>0.708</td>
<td>0.415</td>
<td>-</td>
</tr>
<tr>
<td>rbf</td>
<td>3</td>
<td>0.735</td>
<td>0.441</td>
<td>0.741</td>
<td>0.456</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>0.744</td>
<td>0.453</td>
<td>0.745</td>
<td>0.455</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>0.731</td>
<td>0.443</td>
<td>0.743</td>
<td>0.457</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>0.708</td>
<td>0.415</td>
<td>0.729</td>
<td>0.440</td>
<td>-</td>
</tr>
<tr>
<td>nsoe</td>
<td>3</td>
<td>0.754</td>
<td>0.463</td>
<td>0.756</td>
<td>0.463</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>0.745</td>
<td>0.458</td>
<td>0.749</td>
<td>0.460</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>0.728</td>
<td>0.440</td>
<td>0.743</td>
<td>0.459</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>0.700</td>
<td>0.411</td>
<td>0.727</td>
<td>0.435</td>
<td>-</td>
</tr>
<tr>
<td>g</td>
<td>3</td>
<td>0.754</td>
<td>0.463</td>
<td>0.756</td>
<td>0.463</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>0.745</td>
<td>0.458</td>
<td>0.749</td>
<td>0.460</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>0.728</td>
<td>0.440</td>
<td>0.743</td>
<td>0.459</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>0.700</td>
<td>0.411</td>
<td>0.727</td>
<td>0.435</td>
<td>-</td>
</tr>
<tr>
<td>g</td>
<td>3</td>
<td>0.749</td>
<td>0.463</td>
<td>0.753</td>
<td>0.465</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>0.749</td>
<td>0.455</td>
<td>0.750</td>
<td>0.455</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>0.734</td>
<td>0.445</td>
<td>0.743</td>
<td>0.452</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>0.711</td>
<td>0.420</td>
<td>0.733</td>
<td>0.445</td>
<td>-</td>
</tr>
</tbody>
</table>

The numbers in bold show the best models for a fixed w parameter, as measured by ROC. p, rbf, and nsoe represent the PSI-BLAST profile and YASSPP scoring matrices, respectively. nsoe, rbf, and lin represent the three different kernels studied using the V w, f as the base kernel. * denotes the best classification results in the sub-tables, and ** denotes the best classification results achieved on this dataset. For the best model we report a Q2 accuracy of 83.0% with an errsig rate of 0.34.

Protein-DNA Interaction Sites Performance

Analyzing the ROC and F1 scores obtained on the protein-DNA interaction site prediction problem in Table 5.6, we observe that for the lin kernels the classification accuracy decreases with increasing w sizes but fixed f parameters. This suggests that for predicting protein-DNA interaction sites, finer order-specific information holds more value compared to the coarser information. This trend was reversed in the case of disorder prediction where coarser information did have some benefit over entirely using the fine information. This is likely due to the inherent nature of these properties.

Further, the lin kernel for a small w = 3 value shows better results than the nsoe and rbf kernel. The linear kernel with the coarse information can extract some of the pairwise information that is extracted by
the rbf and nsoe kernels. The value of $w$ plays an important part in reducing the size of feature vectors and hence, the computational complexity. As such, models with lower $w$ values may be preferred over models with higher $w$ values when the classification accuracy gap is not large.

The best model is obtained by combining the $P$ and $S$ features which gives a raw $Q_2$ accuracy of 83%. The protein-DNA interaction site program DISIS uses a two-level approach to solve this problem [140]. The first level, which uses SVM learning with profile, predicted secondary structure, and predicted solvent accessibility as inputs, gives $Q_2 = 83\%$ to which our performance compares favorably. DISIS goes on to smooth this initial prediction using a rule-based approach which improves accuracy. We have not yet explored this type of multi-level approach.

**Local Structure Alphabet Performance**

We chose to use the Protein Blocks [41] as our target alphabet for local structure prediction. There are sixteen members in this alphabet which significantly increases prediction difficulty over traditional three-letter secondary structure prediction.

We used a dataset consisting of 1600 proteins derived from the SCOP [136] version 1.57 database, classes a to e, and where no two protein domains have more than 75% sequence identity. This dataset was previously used for predicting of profile-profile scoring functions using neural networks [142]. We computed the local structure alphabets, Protein Blocks [41] using the 3D structure for the proteins.

Due to the high computational requirements associated with such a large training set, we evaluated our nsoe kernels on a wide set of parameters for $w$ and $f$, but only on a small subset of the 1600 proteins present in the dataset. From this experiment, we observed that prediction was best when $w = f$ and used this to limit the choice of parameters for larger-scale evaluation. Once these promising models were determined, we carried out a 3-way cross validation experiment using all 1600 protein for each parameter set. Table 5.7 reports the classification accuracy in terms of the $Q_{16}$ accuracy and average of the ROC scores for different members of the Protein Blocks.

From Table 5.7 we can draw the well-established conclusion of this paper that the nsoe kernel performs marginally better than the rbf kernel. The addition of predicted secondary structure information, $S$ features does improve the $Q_{16}$ performance marginally as was expected for local structure prediction. Our $Q_{16}$ results are very encouraging, since they are above 67%, whereas the prediction accuracy for a random predictor would be only 6.25%. Competitive methods for local structure alphabet prediction have reported a $Q_{16}$ accuracy of 40.7% [53]. However, these results cannot be directly compared with our method, as they were obtained on a different train/test dataset. We are in the process of comparing PROSAT’s performance to other
datasets and methods [99, 41, 53].

<table>
<thead>
<tr>
<th>Table 5.7. Classification Performance on the Local Structure Alphabet Dataset.</th>
<th>$w = f = 5$</th>
<th>$w = f = 7$</th>
<th>$w = f = 9$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ROC $Q_{16}$</td>
<td>ROC $Q_{16}$</td>
<td>ROC $Q_{16}$</td>
</tr>
<tr>
<td>$P^{rfj}$</td>
<td>0.82 64.9</td>
<td>0.81 64.7</td>
<td>0.81 64.2</td>
</tr>
<tr>
<td>$P^{nsse}$</td>
<td>0.83 67.3</td>
<td>0.82 67.7</td>
<td>0.82 67.7</td>
</tr>
<tr>
<td>$P^{srj}$</td>
<td>0.84 66.4</td>
<td>0.84 66.9</td>
<td>0.83 67.2</td>
</tr>
<tr>
<td>$P^{snsse}$</td>
<td>0.85 68.0</td>
<td>0.84 68.5</td>
<td>0.83 68.9**</td>
</tr>
</tbody>
</table>

$w = f$ gave the best results on testing on few sample points, and hence due to the expensive nature of this problem, we did not test it on a wide set of parameters. ** denotes the best scoring model based on the $Q_{16}$ scores. For this best model the errsig rate of 0.21.

5.5 MONSTER Web Server

We developed and made available a web server called MONSTER [163]$^4$, Minnesota prOteiN Sequence annotaTion servER. MONSTER provides biologists and practitioners an easy-to-use service for predicting local structural and functional properties of proteins predicted using sequence information only.

Currently, MONSTER provides residue-wise annotation prediction services that include assigning each residue of sequence to its predicted secondary structure, transmembrane-helix region, disorder region, protein-dna site, local structure alphabet, solvent accessible surface area, and contact-order.

The profile-based window encoding kernel schemes that were developed as part of the generalized annotation framework (ProSAT discussed in the above Section) is the predictor that serves users of MONSTER. The accuracy of MONSTER is detailed in Section 5.4.3 on page 106, and this section provides the server implementation details of MONSTER. A detailed technical report about MONSTER is under review but can be obtained from the url (http://www.cs.umn.edu/research/technical_reports.php?page=report&report_id=08-010).

MONSTER is one of the few web servers like SCRATCH [28]$^5$ and PredictProtein [175]$^6$ that provide a whole suite of prediction services for a protein sequence, with an emphasis on prediction of local properties.

$^4$MONSTER: http://bio.dtc.umn.edu/monster
$^5$SCRATCH: http://www.ics.uci.edu/~baldig/scratch
$^6$PredictProtein: http://predictprotein.org
in the structural and functional space.

## 5.5.1 Server Implementation

MONSTER provides a user-friendly intuitive interface to analyze the structural and functional properties of protein sequence. The web server interface for MONSTER was developed using a popular content management system, Drupal\(^7\), and the back-end programs were developed using the ProSAT [164, 160] toolkit, which internally uses the publicly available SVM\(^\text{light}\) program [88] for training the maximum margin classification and regression models. MONSTER provides seven prediction services which include: (i) secondary structure prediction, (ii) transmembrane-helix prediction, (iii) disorder region prediction, (iv) protein-dna binding site prediction, (v) local structure alphabet (protein blocks [41]) prediction, (vi) solvent accessibility surface area estimation, and (vii) residue-wise contact order prediction.

![Figure 5.1. Screenshot of MONSTER's INPUT FORM](image)

The user submits a protein sequence as input, along with the selection of properties that he/she would like to predict. Figure 5.1 is a screen shot of the input form that a MONSTER user is presented with.

MONSTER provides users with the option of registering, or staying anonymous. In case of registered users, MONSTER provides efficient and better job management by providing the status of multiple submissions by the user, and also providing the output of completed jobs at a common location. Unregistered users\(^7\)http://drupal.org
can only view their submissions using the email provided link. For both users the submissions are stored on the MONSTER server for a period of three months.

The final output is presented to the user as a well formatted HTML output page. The link containing the url of this output link is returned to the user via a user-provided email address. The output for the various residue-wise classification problems is provided in three formats: (i) a sequence of predicted discrete labels for each residue, (ii) a plot showing the SVM prediction scores generated for every residue by each of the one-versus-rest classifiers, and (iii) a space-delimited text file (called the profile files) containing the SVM prediction scores generated for every residue by each of the one-versus-rest binary classifiers. In case of the residue-wise estimation or regression problem the output is provided in two simple formats: (i) a plot, and (ii) a text file (profile) showing the estimated value by the \( \varepsilon \)-SVR model. The users have the option of downloading all the profile files for the various prediction services as an archive. Figures 5.2 and 5.3 show the output screenshots for the disorder prediction and secondary structure prediction of an example sequence, respectively.
5.6 Conclusions

In this work we have developed a generic support vector machine based framework for producing predictive models to annotate protein sequences. We have tested our framework, ProSAT, with different sets of features on several annotation problems. We have evaluated multiclass classification and binary classification models for predicting local structure, solvent accessibility, transmembrane helical regions, disorder prediction, and protein-DNA interaction site prediction. We have also tested regression models for residue-wise contact order estimation and solvent accessibility prediction.

Our experimental evaluation showed that, in general, the nsoe kernel achieves better performance than the standard rbf kernels across a wide range of problems and datasets, even though for some problems, these improvements are rather small. In addition, our results showed that for some problems, by incorporating local information at different levels of granularity, we were able to achieve better performance when compared to the traditional fine-grain approach. Overall, ProSAT outperformed state-of-the-art, tuned prediction methods for residue-wise contact order, solvent accessibility, transmembrane helices, and local structure alphabet prediction problems. We also show comparable performance on the protein-DNA interaction and disordered prediction problems.

ProSAT was also used in two separate other studies. Firstly ProSAT’s accurate residue-wise transmembrane helix predictions were used within a framework for identifying transmembrane helix segments as well as orientation of these segments. This method called TOPTMH [3] used ProSAT’s prediction as a base classification, integrating it with hydrophobicity information and other physico-chemical properties and modeling the different components using a hidden markov model. Secondly, another study used ProSAT for predicting ligand-binding site residues. These binding site predictions were then incorporated into improving the homology modeling of binding-site regions, by using a target-template alignment that used predicted binding site information. Results of this study showed a lot of promise and far-reaching applications in understanding the relationship between target and ligand pairs [105].

ProSAT along with MONSTER provides to the practitioners an efficient and easy-to-use tool for a wide variety of annotation problems. The results of some of these predictions can be used to assist in solving the overarching 3D structure prediction problem. In the future, we intend to use this annotation framework to predict various 1D features of a protein and effectively integrate them to provide valuable supplementary information for determining the 3D structure of proteins.
Incremental Window-based Sequence Alignment

6.1 Introduction

The current state-of-the-art sequence alignment algorithms have a well defined optimal dynamic programming based solution, introduced decades ago. These optimal algorithms, Smith-Waterman [188] and Needleman-Wunsch [138] solve the local and global sequence alignment problems respectively. Over the years, alignment methods have advanced with several variations of the optimal alignment method, use of gap modeling techniques [69], heuristics [6,149], and more recently the use of profile [64,51,7] and structure information [94].

In recent years, there has been a considerable research effort in developing kernel-based methods for building discriminatory models for remote homology detection and fold recognition (See Section 3.3.3 on page 29 for a comprehensive literature review). Motivated by these developments and our success with string kernels for remote homology detection and fold recognition (Chapter 3), we use similar ideas to develop a sequence alignment method.

These set of heuristic, window-based alignment methods are presented in this article [159]. Our methods incrementally construct the alignment by using the highest scoring pairs of residues between the two sequences at each step. The residue pair scoring was borrowed from string kernel theory where to score the residue pairs in consideration, we examined short subsequences, referred to a w-mers centered around each of
the two residues. We introduced several heuristics to identify aligned residue pairs using the \textit{w}mers coupled with profile information.

### 6.2 Methods and Algorithms

The overall methodology of our alignment algorithms is to incrementally construct the alignment by using various heuristics to identify the pairs of aligned residues. The key idea shared by these algorithms is that they determine whether or not a pair of residues should be aligned together by examining the (short) subsequences, referred to as \textit{w}mers, that are centered around each of the two residues (Detailed in Section 2.4.1 on page 21).

Given a pair of residues \(x_i\) and \(y_j\) at positions \(i\) and \(j\) of sequence \(X\) and \(Y\) respectively, \textit{w}mers around the residues, \(\text{wmer}_X(i)\) and \(\text{wmer}_Y(j)\) are scored \(\text{wscore}(x_i, y_j)\) given by Equation 2.2 on page 21) using a profile-to-profile scoring scheme, in particular the PICASSO scoring function given by Equation 2.4 on page 23. PICASSO scoring function was used in this study because it was effective to capture PSI-BLAST generated information within the context of remote homology detection and fold recognition (Chapter 3).

#### 6.2.1 Central Alignment Scheme (CA).

This is the simplest alignment algorithm that we developed and computes the alignment by progressively aligning the pairs of residues that have the highest positive \(\text{wscore}\) values subject to the constraint that they do not conflict with the portion of the alignment that has been constructed thus far.

Specifically, given two sequences \(X\) and \(Y\) of length \(n\) and \(m\), respectively and a value for \(w\), it starts by computing the set \(P_w\) of residue-pairs that are candidates for inclusion in the alignment by considering only the pairs that have positive \(\text{wscore}\) values. That is,

\[
P_w = \{(x_i, y_j) \mid \text{wscore}(x_i, y_j) > 0\},
\]

where \(w < i \leq n - w\) and \(w < j \leq m - w\). Then it performs a series of iterations in which it performs the following three steps: First, it extracts from \(P_w\) the residue-pair with the highest \(\text{wscore}\) value \((x_{i^*}, y_{j^*})\): Second, it aligns \(x_{i^*}\) against \(y_{j^*}\): Third, it removes from \(P_w\) all residue-pairs that cannot be part of a valid alignment given that \(x_{i^*}\) and \(y_{j^*}\) have been aligned with each other. This process terminates when \(P_w\) becomes empty. Positions that do not belong to any of the selected residue pairs are left unaligned (i.e., aligned against spaces).

The residue pairs that need to be removed are: (i) \((x_{i^*}, y_l)\) \(\forall l\), (ii) \((x_k, y_{j^*})\) \(\forall k\), (iii) \((x_k, y_l)\) \(\forall (k > i^* \land l < j^*)\), and (iv) \((x_k, y_l)\) \(\forall (k < i^* \land l > j^*)\). The first two conditions remove from \(P_w\) all residue-pairs involving
6.2.2 Subset Alignment Scheme (SA).

A limitation of the central alignment scheme is that it may leave a large number of residues unaligned because (i) it only considers the residue-pairs with positive $w$ scores, and (ii) it will not align the first and last $w$ positions of the two sequences ($P_w$ contains only pairs involving interior residues).

To address this problem we developed the subset alignment scheme (SA), which can be considered an extension to the CA scheme. Specifically, the SA scheme modifies the second and third steps of the CA algorithm as follows. During the second step, in addition to including the $(x_i^*, y_j^*)$ pair in the alignment, it also includes in the alignment all previously unaligned residue-pairs of the form $(x_i^* + k, y_j^* + k)$ for $-w < k < +w$. That is, it can potentially include all residue-pairs involved in $(x_i^*, y_j^*)$’s $w$-mer. Note that due to the incremental nature of the algorithm, step (ii) essentially extends the alignment around the $(x_i^* + k, y_j^* + k)$ residue-pair until it encounters a residue (from either $X$ or $Y$) that has already been aligned. We will refer to this as the alignment extension operation. During the third step the SA algorithm removes from $P_w$ all residue-pairs that are now in conflict with all aligned residue-pairs that were selected in second step.

6.2.3 Central and Subset Alignment Scheme (CSA).

A potential problem with the SA scheme, is that it may align a pair of residues $(x_i^* + k, y_j^* + k)$ with each other, even when $P_w$ contains residue-pairs with higher $w$-score values for either or both of the two residues. This happens, because SA’s alignment extension operation extends the alignment as soon as it extracts the highest scoring residue pair from $P_w$ and there may be some higher-scoring $w$-mers for these positions in $P_w$.

For this reason, we developed a hybrid scheme that combines the CA and SA approaches. Specifically, the new scheme first computes a CA alignment and then augments it by applying the alignment extension approach used by SA to each pair of its aligned residues.

6.2.4 Variable $w$-mer Alignment Scheme

The alignment schemes, CA, SA and CSA were discussed in the context of a fixed length $w$-mer. The potential drawback of this scheme is that if $w$ is set to a relatively large value, it may fail to identify positive scoring subsequences; whereas if it is set too low, it may fail to reward sequence-pairs that have relatively long similar subsequences.
Similar to the variable-length kernels introduced in Section 3.4.1, we extend the alignment algorithms to operate with variable length \( w \)mers. The key difference from the use of fixed length \( w \)mers centered around residue pairs \( x_i \) and \( y_j \) is the fact that we define length \( w^* \) in the range of 1 to \( w \), such that

\[
w^* = \arg\max_K \{K\text{score}(x_i, y_j)\}, \quad \varepsilon \in K = 1 \ldots w
\]

(6.2)

where \( K\text{score} \) is nothing but the \( w\text{score} \) with varying parameter \( w \) defined in Equation 3.3 on page 33.

Our alignment schemes start by computing the set \( P'_w \) of residue pairs that are candidates for inclusion in the alignment by considering only pairs that have positive \( w^*\text{score} \) values. With this change all steps of our alignment algorithms remain same. It is important to point out that the SA scheme using the variable length \( w \)mers will have its alignment extension operation till \( w^* \).

As a notation reference we denote the variable \( w \)mer alignment algorithms by \( CA^v \), \( SA^v \) and \( CSA^v \) to distinguish them from the fixed \( w \)mer alignment algorithms denoted by \( CA^f \), \( SA^f \) and \( CSA^f \) in this study.

### 6.3 Experimental Results

We evaluated the performance of the computed alignments on two recently published benchmark datasets and compared them against the alignments computed by existing state-of-the-art dynamic programming-based profile-to-profile local and global sequence alignment algorithms. Our results show that the new algorithms achieve alignments that are comparable or better to those achieved by existing algorithms. Moreover, our results also showed that these algorithms can be used to provide better information as to which of the aligned positions are more reliable—a critical piece of information for comparative modeling applications.

#### 6.3.1 Evaluation Methodologies and Metrics

We evaluated the performance of the proposed window-based alignment algorithms by considering (i) the quality of the alignment itself and (ii) the extent to which the inherent ordering of the aligned pairs of residues can be used to identify portions of the alignment that are more reliable than others. In order to assess alignment quality we used two widely used methodologies, often referred to as template-based [51] and model-based [52], whereas the reliability was assessed by following a methodology that was recently proposed in the context of comparative modeling [202].

**Template-based Approach.**

The first method for evaluating alignment quality compares the differences between the alignment generated to template alignments [51, 178, 52]. These template alignments are generally derived from various structural
alignment programs and are considered to be the gold standard.

We use three quality scores, namely the developer’s score ($f_D$) [178], the modeler’s score ($f_M$) [178] and the Cline score (CS) [30] to compare the template alignments with the generated alignments. The developer’s score is the number of correctly aligned residue pairs in the generated alignment divided by the length of the template alignment. (The length of an alignment is defined as the number of aligned residue pairs.) The modeler’s score computes the ratio of correctly aligned residue pairs with the length of the generated alignment. The Cline score was developed to address the issues with $f_M$ and $f_D$ by penalizing both under-alignment and over-alignment, and also crediting regions in the generated alignment that may be shifted by a few positions relative to the reference alignment [51, 30]. The steps for computation of the Cline score can be found in the study [30].

Note that the $f_D$ and $f_M$ scores are equivalent to the more traditional measures of recall and precision [55], respectively that are used extensively to measure prediction performance. In the rest of the discussion we will primarily refer to $f_D$ and $f_M$ by the more intuitive names of recall and precision, respectively.

**Model-based Approach.**

An alternative to using a template-based approach is to build a structural model from the alignment and evaluate the similarity between the model and the template structure [52, 145]. Starting from the alignment between a pair of proteins (one protein considered to be the query protein, the second considered to be the target protein whose 3D structure is known), a model protein is created which consists of the carbon alpha, $C_\alpha$ atoms of the query protein. The atomic coordinates of this model protein are the atomic coordinates of the target protein i.e., for every aligned pair of residues, the query protein has its $C_\alpha$ atomic coordinates replaced by the corresponding atomic coordinates of the target protein. The similarity between the two structures (the model protein and target protein) after a structural super-imposition [125], is used as an assessment of sequence alignment quality.

In our study, we computed this similarity using the LGscore [38] that takes into account the common segments between the pair of proteins. LGscore computes the similarity between two protein structures (model and template structure) based on the common segments between them. It is desirable to have long common segments with high structural similarity. The LGscore measure was used to evaluate the structures obtained by threading methods [145] in the CAFASP2 [57] and LiveBench [23] experiments as well as a sequence alignment quality measure [52].

Note that instead of LGscore other structural similarity methods or protein modeling assessment measures can be used for evaluating the quality of the model (e.g. rmsd measure [96], global distance test score
Reliability of Aligned Regions.

In comparative modeling and several other applications, it is essential not only to align residue pairs but also to provide some reliability index or confidence measure associated with the aligned residue pairs. While building protein structure models using comparative modeling strategies it is important to include only those regions where the alignment is considered to be good or reliable [86, 30, 179, 133, 202].

One of the reliability assessment measures calculated a smoothed profile-derived alignment score. The score for each of the aligned residue in the template alignment was computed using a triangular smoothing window of size 5. The reliability was assessed by setting up a threshold value for the smoothed profile-derived score [202]. Our approach for reliability assessment was very similar to this method.

Using the template-based benchmarks we evaluated the reliability of the aligned residue pairs by ranking the aligned pairs in the query alignment. We score the aligned positions using fixed length $w$ scores or variable length $w$ scores. The reliability measure is computed as the recall at different percent levels of incorrectly aligned residue pairs (up to 5%). The notion of a hit is defined as having the same aligned residue pairs in both the query and template alignments. The difference in our reliability scheme was the use of a profile-to-profile scoring functions equally weighted at all positions of the $w$ mer rather than using a smoothing $w$ mer [202].

6.3.2 Dataset Description

For the template-based assessment scheme we used a dataset created to evaluate the various profile-to-profile scoring functions for protein sequence alignment [51]. The dataset consists of 588 reference alignment pairs having high structural similarity but low sequence identity ($\leq 30\%$). This dataset was selected to have a high pairwise structural similarity using the consensus of FSSP [78] and CE [184].

For the model-based evaluation scheme, we used a benchmark created from SCOP 1.39 filtered to only contain domains with less than 50% pairwise sequence identity [52]. This dataset contains of 9983 protein domain pairs, such that 1903 belong to the same families, 3101 share only the same superfamily, and 4979 share only the same fold. Due to the non-symmetrical nature of models built from alignments, each pair of sequences were evaluated twice—leading to a benchmark of 19966 domain pairs.
6.3.3 Assessment of Incremental Window-based Alignments

Table 6.1 on the next page provides an extensive set of results illustrating the performance of the CA, SA, and CSA schemes on the template-based dataset for different values of $w$ and for fixed- and variable-length $w$mers. Note that the column labeled “$CS_{\leq 15\%}$” shows the CS results for the subset of sequence-pairs that have less than 15% sequence identity (i.e., a subset that is inherently harder to align well).

**Central vs Subset vs Combined.**

The results of Table 6.1 on the following page show that with respect to the CS scores, SA tends to perform better than either CA or CSA, whereas CA performs consistently the worst. The only exception is for variable-length $w$mers, in which SA’s performance is comparable to that of CSA. The relative advantage of SA is more evident if we consider the subset of sequence-pairs with less that 15% sequence identity, for which its CS scores are consistently higher than those achieved by the other schemes (SA achieves a score of 0.649 whereas CA and CSA achieves scores of 0.614 and 0.628, respectively).

By looking at the performance of the various schemes in terms of recall, we can see that SA’s higher CS-based performance is due to the fact that it achieves significantly better recall values than the other schemes. This was to be expected, as it was one of the motivation behind the development of SA. Also, the precision-based results show that CA achieves somewhat better precisions than CSA, whereas SA’s precision is comparable or better to that of the other schemes.

**Fixed vs Variable Length Alignments.**

Analyzing the performance of alignment methods that use fixed length $w$mers compared to the methods that use variable length $w$mers, we notice that for the CA and CSA schemes, for the same $w$mer length the recall as well as the precision scores have higher values. Note that the higher recall is expected, because the methods using a variable $w$mer size window will have a higher flexibility in allowing larger number of $w$mers (with a positive score) to be picked for the candidate set $P'_w$. Another key observation is that $SA^f$ performs better in terms of recall than $SA^v$. This is because for the same value of $w$, the $w^*$ value selected by $SA^v$ may be smaller than $w$ (i.e., the value used by $SA^f$). As a result, $SA^f$’s alignment extension operations will involve longer windows, which can produce longer alignments than $SA^v$, and thus higher recall values.
Table 6.1. Accuracy on a Template-based Dataset.

<table>
<thead>
<tr>
<th>Model Type</th>
<th>WMer</th>
<th>FM (Precision)</th>
<th>FD (Recall)</th>
<th>CS</th>
<th>CS ≤ 15%</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>fixed</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>central (CA\textsuperscript{f})</td>
<td>\textit{wm}er = 2</td>
<td>0.805</td>
<td>0.791</td>
<td>0.803</td>
<td>0.600</td>
</tr>
<tr>
<td></td>
<td>\textit{wm}er = 3</td>
<td>0.799</td>
<td>0.776</td>
<td>0.794</td>
<td>0.596</td>
</tr>
<tr>
<td></td>
<td>\textit{wm}er = 4</td>
<td>0.791</td>
<td>0.756</td>
<td>0.782</td>
<td>0.587</td>
</tr>
<tr>
<td></td>
<td>\textit{wm}er = 5</td>
<td>0.776</td>
<td>0.732</td>
<td>0.764</td>
<td>0.572</td>
</tr>
<tr>
<td>subset (SA\textsuperscript{f})</td>
<td>\textit{wm}er = 2</td>
<td>0.802</td>
<td>0.835</td>
<td>0.826</td>
<td>0.626</td>
</tr>
<tr>
<td></td>
<td>\textit{wm}er = 3</td>
<td>0.805</td>
<td>0.842</td>
<td>0.831</td>
<td>0.642</td>
</tr>
<tr>
<td></td>
<td>\textit{wm}er = 4</td>
<td>0.805</td>
<td>0.842</td>
<td>0.832</td>
<td>0.644</td>
</tr>
<tr>
<td></td>
<td>\textit{wm}er = 5</td>
<td>0.802</td>
<td>0.838</td>
<td>0.828</td>
<td>0.649</td>
</tr>
<tr>
<td>combined (CSA\textsuperscript{f})</td>
<td>\textit{wm}er = 2</td>
<td>0.791</td>
<td>0.822</td>
<td>0.816</td>
<td>0.619</td>
</tr>
<tr>
<td></td>
<td>\textit{wm}er = 3</td>
<td>0.785</td>
<td>0.819</td>
<td>0.814</td>
<td>0.623</td>
</tr>
<tr>
<td></td>
<td>\textit{wm}er = 4</td>
<td>0.779</td>
<td>0.811</td>
<td>0.808</td>
<td>0.624</td>
</tr>
<tr>
<td></td>
<td>\textit{wm}er = 5</td>
<td>0.767</td>
<td>0.798</td>
<td>0.798</td>
<td>0.624</td>
</tr>
<tr>
<td><strong>variable</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>central (CA\textsuperscript{v})</td>
<td>\textit{wm}er = 2</td>
<td>0.799</td>
<td>0.804</td>
<td>0.809</td>
<td>0.595</td>
</tr>
<tr>
<td></td>
<td>\textit{wm}er = 3</td>
<td>0.802</td>
<td>0.807</td>
<td>0.812</td>
<td>0.605</td>
</tr>
<tr>
<td></td>
<td>\textit{wm}er = 4</td>
<td>0.805</td>
<td>0.797</td>
<td>0.810</td>
<td>0.611</td>
</tr>
<tr>
<td></td>
<td>\textit{wm}er = 5</td>
<td>0.805</td>
<td>0.797</td>
<td>0.807</td>
<td>0.614</td>
</tr>
<tr>
<td>subset (SA\textsuperscript{v})</td>
<td>\textit{wm}er = 2</td>
<td>0.798</td>
<td>0.827</td>
<td>0.820</td>
<td>0.615</td>
</tr>
<tr>
<td></td>
<td>\textit{wm}er = 3</td>
<td>0.798</td>
<td>0.834</td>
<td>0.825</td>
<td>0.629</td>
</tr>
<tr>
<td></td>
<td>\textit{wm}er = 4</td>
<td>0.798</td>
<td>0.836</td>
<td>0.827</td>
<td>0.634</td>
</tr>
<tr>
<td></td>
<td>\textit{wm}er = 5</td>
<td>0.794</td>
<td>0.832</td>
<td>0.823</td>
<td>0.636</td>
</tr>
<tr>
<td>combined (CSA\textsuperscript{v})</td>
<td>\textit{wm}er = 2</td>
<td>0.795</td>
<td>0.822</td>
<td>0.813</td>
<td>0.600</td>
</tr>
<tr>
<td></td>
<td>\textit{wm}er = 3</td>
<td>0.797</td>
<td>0.827</td>
<td>0.820</td>
<td>0.614</td>
</tr>
<tr>
<td></td>
<td>\textit{wm}er = 4</td>
<td>0.800</td>
<td>0.831</td>
<td>0.824</td>
<td>0.621</td>
</tr>
<tr>
<td></td>
<td>\textit{wm}er = 5</td>
<td>0.800</td>
<td>0.832</td>
<td>0.825</td>
<td>0.628</td>
</tr>
</tbody>
</table>

In the table, \( f_M \) denotes the Modeler’s score, \( f_D \) denotes the Developer’s score, CS denotes the Cline score, and \( CS \leq 15\% \) denotes the Cline score for a subset of sequence pairs sharing less than 15% sequence identity.
6.3.3 Assessment of Incremental Window-based Alignments

Sensitivity of Schemes with respect to varying \( w \)\( \text{mer} \) size

Looking at the performance achieved by the various schemes in Table 6.1 on the preceding page as \( w \) ranges from two to five, we see that in general, SA’s and CSA’s performance does not significantly change (e.g., CS scores stay within a tight range), whereas CA’s performance tends to deteriorate with increasing \( w \). This latter behavior is due to the fact that as we increase the \( w \)\( \text{mer} \) size, fewer \( w \)mers will have a positive score and hence will not be included as part of the set \( P_w \). We see a direct effect of this leading to a decrease in the recall scores. Also increase in the \( w \)\( \text{mer} \) size does lead to a decrease in precision score as well. This is because for a larger \( w \)\( \text{mer} \) window the positive scoring \( w \)mers may not be due to the more “central” positions. Evidence of this can be seen by comparing the behavior of the CA\(^v\) scheme in which both the precision and recall scores stay the same.

Another key observation is that the schemes that utilize variable length \( w \)mers tend to perform better for larger values of \( w \). This is because of the flexibility associated with using a variable length \( w \)mer.

Model-based Performance Assessment

We perform a comprehensive assessment on the model-based benchmark. In Table 6.2 on the following page we report only the best results achieved rather than showing results for varying \( w \)\( \text{mer} \) sizes as done in Table 6.1 on the preceding page.

Firstly, we notice the difference in the LGscore values for the family, superfamily and fold pairs clearly showing the difficulty nature of the three sets of problems, with the fold-pairs being the hardest to model followed by the superfamily and family level pairs.

Similar to the template-based results, the SA scheme has the best LGscore at the family, superfamily and fold levels for both the variable and fixed \( w \)\( \text{mer} \) setting. A surprising fact was that the performance results as measured by the LGscore did not decrease with increasing \( w \)\( \text{mer} \) lengths. In fact, we observed that the use of a higher \( w \)\( \text{mer} \) size of 5 for the fixed length scheme achieved the best results of 1.53 and 4.29 for the fold and superfamily level problems. We also observe slightly better performance for the variable \( w \)\( \text{mer} \) schemes compared to the fixed \( w \)\( \text{mer} \) schemes.

The performance of the CSA\(^v\) alignment method was the lowest for both the family and superfamily level pairs which contrasts the results seen previously on the template-based dataset in Table 6.1 on the facing page.
6.3.4 Comparison with Earlier Results

Table 6.2. Alignment Accuracy Results on a Model-based Dataset.

<table>
<thead>
<tr>
<th>Alignment Scheme</th>
<th>Family</th>
<th>Superfamily</th>
<th>Fold</th>
</tr>
</thead>
<tbody>
<tr>
<td>CA^f (2)</td>
<td>14.86</td>
<td>1.66</td>
<td>0.04</td>
</tr>
<tr>
<td>SA^f (5)</td>
<td>16.44</td>
<td>4.29</td>
<td>1.53</td>
</tr>
<tr>
<td>CSA^f (2)</td>
<td>15.47</td>
<td>2.53</td>
<td>0.203</td>
</tr>
<tr>
<td>CA^v (5)</td>
<td>15.10</td>
<td>2.43</td>
<td>0.12</td>
</tr>
<tr>
<td>SA^v (5)</td>
<td>16.48</td>
<td>4.05</td>
<td>1.05</td>
</tr>
<tr>
<td>CSA^v (5)</td>
<td>14.05</td>
<td>2.32</td>
<td>0.14</td>
</tr>
</tbody>
</table>

The numbers in the parameter indicate the *wmer* length for the various alignment schemes.

6.3.4 Comparison with Earlier Results

Template-based Benchmark.

Table 6.3 on page 128 shows the comparative performance of our window based schemes against some of the best profile-to-profile scoring techniques studied previously [51]. In the table we show results for the schemes pdotp, correlp and coach. pdotp uses dot product to compute the similarity between two profiles, correlp computes the Pearson correlation between the profile columns, whereas coach [50] uses an asymmetrical complex dot product between the HMM profile and a position frequency matrix.

We show results of these schemes as published previously [51] using SAM T99 profiles (The performance of these alignment methods using SAM T99 profiles is 3-4% better than the PSI-BLAST based profiles [51]) Our methods show comparable performance to these alignment methods using SAM T99 templates.

We also compare the results of the window based alignment methods to a local Smith-Waterman [188] alignment algorithm implementation (SW-PSSM) using the same profile-to-profile scoring function as used for the window based alignments (Equation 2.4 on page 23). Within this local alignment framework we use an affine gap model along with a zero-shift parameter [215] to maintain certain necessary requirements of a good optimal alignment. We optimize the gap modeling parameters (gap opening (*go*), gap extension (*ge*)) and the zero shift value (*zs*) to obtain highly optimal alignments for comparative purposes.

We observe in Table 6.3 on page 128 that the incremental window-based alignment schemes perform very competitively when compared to our fully optimized SW-PSSM implementation. Also notice the superiority of our optimized SW-PSSM implementation to the alignment methods using pdotp, correlp and coach as their profile-profile scoring functions. The difference in the SW-PSSM results with the other standard alignment
Figure 6.1. Cline Score Comparison of SW-PSSM scheme against \(SA^f\) scheme for the 588 alignment pairs in the template-based dataset

Techniques may be due to the use of a more sensitive PICASSO based profile-to-profile scoring function. Further, these results verify that we are comparing our novel window based alignment methods to a fully optimized SW-PSSM alignment algorithm.

The performance of the window-based scheme is actually very promising. We select one of the better performing schemes (\(SA^f\)) and compare it to the optimized SW-PSSM algorithm using the CS score. Figure 6.1 shows that the comparative performance of the two methods across the 588 alignment pairs in the dataset.

**Model-based Benchmark.**

Our results in Table 6.4 on page 129 reiterate the closeness in performance of the incremental window based alignment method to the highly optimized SW-PSSM alignment algorithm for the family, superfamily and fold level subsets.

Table 6.4 on page 129 also shows results for the optimized local (local sequence alignment using a global scoring matrix), global (global sequence alignment using a global scoring matrix), PSI (3D-PSSM [107] based global sequence alignment against a profile [63] obtained from PSI-BLAST), SSPSI [52](3D-PSSM based global sequence alignment against a profile obtained from PSI-BLAST using secondary structure information) and structural (alignment using structural super-imposition by lgscore2) alignment methods published previously [52]. The structural alignment sets up a higher reference quality score for the benchmark.
Using sequence alignment techniques we would like to achieve these high levels of accuracy. The results shown in Table 6.4 on the facing page for the various previously published schemes, as well as for our methods are the best achieved after optimization of the various parameters.

### Table 6.3. Comparative Performance with Earlier Results on Template-based Dataset.

<table>
<thead>
<tr>
<th>Alignment Scheme</th>
<th>$f_M$</th>
<th>$f_D$</th>
<th>CS</th>
<th>$CS_{\leq 15%}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA$^f$ (3)</td>
<td>0.805</td>
<td>0.842</td>
<td>0.831</td>
<td>0.642</td>
</tr>
<tr>
<td>SA$^v$ (4)</td>
<td>0.798</td>
<td>0.836</td>
<td>0.827</td>
<td>0.634</td>
</tr>
<tr>
<td>SW-PSSM</td>
<td>0.803</td>
<td>0.852</td>
<td>0.841</td>
<td>0.689</td>
</tr>
<tr>
<td>pdotp (T99)</td>
<td>0.806</td>
<td>0.829</td>
<td>0.832</td>
<td>0.697</td>
</tr>
<tr>
<td>correlp (T99)</td>
<td>0.794</td>
<td>0.835</td>
<td>0.829</td>
<td>0.702</td>
</tr>
<tr>
<td>coach (T99)</td>
<td>0.797</td>
<td>0.830</td>
<td>0.829</td>
<td>0.697</td>
</tr>
</tbody>
</table>

The optimized SW-PSSM results are achieved using $go = 3.0$, $ge = 0.75$, $zs = 1.0$. In the table pdotp, correlp, coach use a dot product, correlation function, and a HMM based profile-profile scoring function. T99 denotes the use of SAM T99 based profiles respectively.

We further analyze the data by annotating a model as being correct based on the LGscore value. As done in the study [52] we use the less strict LGscore cutoff ($10^{-3}$) to define a correct model and a more stringent cutoff ($10^{-5}$) to identify models of higher quality. The percentage of models correct based on these cutoffs are shown in Table 6.5 on page 130. Both the incremental window-based alignment methods, as well as the SW-PSSM alignment method, are able to pick the correct models with similar degrees of accuracy. Our techniques also seem to identify a higher percentage of correct models when compared to the previously studied schemes, especially PSI and SSPSI, both of which also incorporate some profile information. As seen from Table 6.5 on page 130 our methods are able to pick a larger fraction of higher quality models for the family and superfamily levels.

**Reliability Performance.**

Table 6.6 on page 131 shows the reliability performance for the window based alignment schemes in comparison to the optimized SW-PSSM based alignment scheme. These results correspond to the average recall scores obtained for all the alignment pairs at different error rates using the procedure described in Section 6.3.1 on page 122.
Table 6.4. Comparative Performance with Earlier Results on a Model-based Dataset.

<table>
<thead>
<tr>
<th>Alignment Scheme</th>
<th>Family</th>
<th>Superfamily</th>
<th>Fold</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA $^f$ (5)</td>
<td>16.44</td>
<td>4.29</td>
<td>1.53</td>
</tr>
<tr>
<td>SA $^v$ (5)</td>
<td>16.48</td>
<td>4.05</td>
<td>1.05</td>
</tr>
<tr>
<td>SW-PSSM</td>
<td>16.66</td>
<td>4.38</td>
<td>2.02</td>
</tr>
<tr>
<td>local</td>
<td>14.1</td>
<td>2.0</td>
<td>0.7</td>
</tr>
<tr>
<td>global</td>
<td>15.1</td>
<td>2.9</td>
<td>1.4</td>
</tr>
<tr>
<td>PSI</td>
<td>15.8</td>
<td>3.3</td>
<td>1.4</td>
</tr>
<tr>
<td>SSPSI</td>
<td>16.0</td>
<td>4.1</td>
<td>2.6</td>
</tr>
<tr>
<td>structural</td>
<td>19.4</td>
<td>9.1</td>
<td>8.0</td>
</tr>
</tbody>
</table>

The optimized SW-PSSM results are achieved using $go = 3.0$, $ge = 0.75$, $zs = 3.0$. All the results are optimized for their relevant parameters.

Though the SW-PSSM algorithm showed slightly better performance in terms of the overall alignment quality (Table 6.3 on the preceding page and Table 6.4), it is interesting to note the window-based schemes using variable length $w$mers showed far better performance at the lower error rates. In particular before seeing any incorrect predictions in the ranked aligned positions, the alignment methods using variable length $w$mers have a recall around 0.260 compared to the recall of 0.205 for the SW-PSSM algorithm. Note that the recall performance of the CSA scheme is slightly better than the CA scheme and slightly worse compared to the SA alignment scheme. These results can be explained by the fact that the high scoring residue pairs aligned by CA are also aligned by the CSA scheme.

As we allow more errors this gap is reduced which can be explained by the higher recall achieved by the SW-PSSM algorithm (Table 6.3 on the preceding page). Using the variable length scores to rank the aligned residue pairs shows a similar trend to the fixed length scoring scheme.

**Complexity Performance**

The incremental window-based alignments introduced in this study have a similar time complexity to the standard dynamic programming based alignment algorithms like Smith-Waterman [188]. The complexity to align sequence pairs of length $n$ on the average is $O(n^2)$ owing to the computational cost involved with scoring every residue-pair between the two sequences. The use of profile information generated using PSI-BLAST involves an additional run-time complexity of searching a database per query. However the advantage
Table 6.5. Fraction of Correct Models based on the LGscore.

<table>
<thead>
<tr>
<th>LGscore</th>
<th>$&lt; 10^{-3}$</th>
<th>$&lt; 10^{-5}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alignment Scheme</td>
<td>Fm</td>
<td>Sf</td>
</tr>
<tr>
<td>SA$^f$ (3)</td>
<td>74</td>
<td>27</td>
</tr>
<tr>
<td>SA$^o$ (3)</td>
<td>74</td>
<td>28</td>
</tr>
<tr>
<td>SW-PSSM</td>
<td>74</td>
<td>27</td>
</tr>
<tr>
<td>local</td>
<td>66</td>
<td>10</td>
</tr>
<tr>
<td>global</td>
<td>70</td>
<td>12</td>
</tr>
<tr>
<td>PSI</td>
<td>72</td>
<td>18</td>
</tr>
<tr>
<td>SSPSI</td>
<td>73</td>
<td>21</td>
</tr>
<tr>
<td>structural</td>
<td>86</td>
<td>60</td>
</tr>
</tbody>
</table>

The optimized SW-PSSM results are achieved using $go = 3.0$, $ge = 0.75$, $zs = 3.0$. All the results are optimized for their relevant parameters. Fm, Sf and Fd denote the family-level, superfamily-level and fold-level performance results respectively.

of using profile information has been well documented, especially from an alignment accuracy standpoint [72, 134].

6.4 Conclusions

We have developed algorithms that identify the aligned pairs of residues using an incremental approach. These algorithms capture the most similar pairs of subsequences as part of the final alignment. The concepts from string-kernel theory (use of ungapped subsequences, scored using profiles) played an integral role in the design of these alignment algorithms.

Our comprehensive experimental study on the template-based and model-based benchmark datasets showed comparable performance to a fully optimized Smith-Waterman profile-based implementation. In terms of the reliability performance of the aligned residue-pairs we notice that the alignment schemes using variable length $w$mers had very promising results. Amongst the window-based schemes we noticed that the subset alignment, SA using both the fixed and variable $w$mers showed the best performance. The sensitivity analysis done by varying the $w$mer size showed the SA schemes to have a robust performance.

The simplicity of our methods and competitive alignment quality as well as aligned region reliability has application of our algorithms in key bioinformatics problems, especially comparative modeling.
### Table 6.6. Reliability Assessment: Recall for the first $k\%$ errors.

<table>
<thead>
<tr>
<th>Method</th>
<th>0%</th>
<th>1%</th>
<th>2%</th>
<th>3%</th>
<th>4%</th>
<th>5%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Fixed Length Scores</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CA$^f$ (3)</td>
<td>0.176</td>
<td>0.281</td>
<td>0.365</td>
<td>0.434</td>
<td>0.494</td>
<td>0.541</td>
</tr>
<tr>
<td>SA$^f$ (3)</td>
<td>0.186</td>
<td>0.297</td>
<td>0.384</td>
<td>0.459</td>
<td>0.519</td>
<td>0.563</td>
</tr>
<tr>
<td>CSA$^f$ (3)</td>
<td>0.180</td>
<td>0.286</td>
<td>0.370</td>
<td>0.438</td>
<td>0.498</td>
<td>0.545</td>
</tr>
<tr>
<td>CA$^v$ (3)</td>
<td>0.254</td>
<td>0.364</td>
<td>0.450</td>
<td>0.515</td>
<td>0.566</td>
<td>0.603</td>
</tr>
<tr>
<td>SA$^v$ (3)</td>
<td>0.260</td>
<td>0.368</td>
<td>0.454</td>
<td>0.521</td>
<td>0.572</td>
<td>0.612</td>
</tr>
<tr>
<td>CSA$^v$ (3)</td>
<td>0.260</td>
<td>0.367</td>
<td>0.454</td>
<td>0.520</td>
<td>0.571</td>
<td>0.610</td>
</tr>
<tr>
<td>SW-PSSM</td>
<td>0.205</td>
<td>0.320</td>
<td>0.405</td>
<td>0.480</td>
<td>0.541</td>
<td>0.586</td>
</tr>
<tr>
<td><strong>Variable Length Scores</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CA$^f$ (3)</td>
<td>0.175</td>
<td>0.281</td>
<td>0.365</td>
<td>0.435</td>
<td>0.495</td>
<td>0.542</td>
</tr>
<tr>
<td>SA$^f$ (3)</td>
<td>0.189</td>
<td>0.300</td>
<td>0.387</td>
<td>0.462</td>
<td>0.521</td>
<td>0.567</td>
</tr>
<tr>
<td>CSA$^f$ (3)</td>
<td>0.180</td>
<td>0.286</td>
<td>0.370</td>
<td>0.439</td>
<td>0.499</td>
<td>0.547</td>
</tr>
<tr>
<td>CA$^v$ (3)</td>
<td>0.253</td>
<td>0.363</td>
<td>0.448</td>
<td>0.514</td>
<td>0.564</td>
<td>0.602</td>
</tr>
<tr>
<td>SA$^v$ (3)</td>
<td>0.258</td>
<td>0.367</td>
<td>0.454</td>
<td>0.520</td>
<td>0.571</td>
<td>0.610</td>
</tr>
<tr>
<td>CSA$^v$ (3)</td>
<td>0.259</td>
<td>0.367</td>
<td>0.453</td>
<td>0.519</td>
<td>0.569</td>
<td>0.607</td>
</tr>
<tr>
<td>SW-PSSM</td>
<td>0.204</td>
<td>0.318</td>
<td>0.405</td>
<td>0.480</td>
<td>0.541</td>
<td>0.586</td>
</tr>
</tbody>
</table>

The optimized SW-PSSM results are achieved using $go = 3.0$, $ge = 0.75$, $zs = 3.0$. The numbers in the parenthesis represent the window width used for the results shown.
Conclusion

In this dissertation we presented several algorithms that have significantly advanced the state-of-the-art computational techniques for the structural and functional characterization of proteins. Specifically, our contributions have led to the development of methods for remote homology detection, fold recognition, sequence alignment, prediction of local structure and function of protein, and a novel pairwise local structure similarity score estimated from sequence. Each of these techniques have shown significant improvements in the performance as seen by the different empirical results. These methods have also proven to be applicable in different steps of a protein structure prediction server using a comparative modeling based methodology.

7.1 Thesis Summary

The focus of this thesis has been in prediction of global as well as local characteristics of proteins. In this section we summarize the contributions and main results of this thesis.

String Kernels for Remote Homology Detection and Fold Recognition

The profile-based direct string kernel developed by us for solving the binary one-versus-rest classification problems within the remote homology and fold recognition contexts provide the following claims: (i) use of profile information better captures the sequence signal in identifying remote homologs and folds, (ii) kernel functions that take into account the similarity between protein sequences directly outperform the feature-based schemes, and (iii) use of time-tested local alignment algorithms [188] if optimized for their gap and shift parameters outperform window-based or subsequence based methods (like the BF-PSSM kernel or Spectrum kernels [123]).

An independent study by Hochreiter et al [75], two years after development of our profile-based kernel
functions established our methods to show the best $ROC$ performance on a common remote homology detection benchmark. Specifically, our kernel function shows 13% and 81% improvement in the average $ROC_{50}$ performance over the next best kernel-based method [116] for the fold recognition benchmark.

Cascaded-Level Learning for Hierarchical Multiclass Classification  We also perform a detailed study to evaluate methods that help in the identification of the most likely fold or superfamily that a protein sequence belongs to. The one-versus-rest classifiers developed by us only answer the question whether a sequence belongs to a particular class or not. We build upon the strong performance of these binary one-versus-rest classification models for solving the $k$-way multiclass classification scheme in a cascaded level setting, learning a set of parameters for each class using large margin principles. Our comprehensive study evaluates three approaches for building these classifiers: (i) schemes that directly build an SVM-based multiclass model, (ii) schemes that employ a second-level learner to combine the predictions generated by a set of binary SVM-based classifiers, and (iii) schemes that build and combine binary classifiers for various levels of the SCOP hierarchy.

Our empirical results in Section 3.7 on page 57 show that direct $k$-way SVM-based formulations and algorithms based on the two-level learning paradigm are quite effective for solving these problems and achieve better results than those obtained by using a set of binary one-vs-rest SVM-based classifiers. Moreover, our results and analysis showed that the two-level schemes that incorporate predictions from binary models constructed for ancestral categories within the SCOP hierarchy tend to not only lead to lower error rates but also reduce the number of errors in which a superfamily is assigned to an entirely different fold and a fold is predicted as being from a different SCOP class.

$f_{RMSD}$: Local Structural Predicted Similarity Score  In this thesis we formulated a new prediction problem i.e., $f_{RMSD}$ prediction problem, similar to learning a profile-to-profile scoring scheme [142]. Predicting the $f_{RMSD}$ scores for residue-pairs using sequence information was motivated by the operational characteristics of structure-to-structure alignment algorithms and necessities of accurate sequence alignments for inferring functional and structural properties between a protein with known function/structure and unknown protein.

Our prediction methodology uses the support vector regression and support vector classification framework for solving the problem. This information is effectively encoded in fixed-length feature vectors. We develop and test novel second-order pairwise exponential kernel functions designed to capture the conserved signals of a pair of local windows centered at each of the residues and use a fusion-kernel-based approach to incorporate the profile and secondary structure-based information.
The experimental results show that there is a high correlation (0.681 – 0.768) between the estimated and actual fragment-level RMSD scores. Moreover, the performance of our algorithms is considerably better than that obtained by state-of-the-art profile-to-profile scoring schemes when used to solve the fragment-level RMSD prediction problems.

The successful prediction of $f_{RMSD}$ scores has several applications to protein structure prediction. In this thesis we review them in Section 4.2.1 on page 73 and perform a case study showing the applicability of $f_{RMSD}$ predictions in improving the sensitivity of sequence alignments. We also show the $f_{RMSD}$ scores effective for assessing the quality of sequence alignments by accurate predictions of $f_{RMSD}$ values for residue-pairs derived from a standard sequence alignment algorithm.

**Sequence Alignment Algorithms** We have developed novel sequence alignment algorithms to improve the accuracy of template-target alignment using two approaches: (i) developing a heuristic alignment algorithm that incrementally selects the most similar scoring $w_{mer}$ or subsequence pairs centered around the residue-pairs, and (ii) learning a sensitive profile-to-profile scoring scheme that emulates the local structure compatibility between pairs of residues using sequence information only, known as the $f_{RMSD}$ scores.

The heuristic incremental alignment approach that uses the highest profile-profile scoring ungapped subsequence, of fixed or variable width at each iteration shows comparable performance in terms of accuracy as well as complexity on a template-based as well as model-based benchmark in comparison to the previous state-of-the-art sequence alignment algorithms (especially optimal dynamic-programming-based approaches). The strength of such an alignment approach is brought forward by an assessment that computes a reliability or confidence score [202] per aligned residue-pair. In such an evaluation the incremental alignment slightly outperforms the standard alignment algorithms, and hence showing potential in comparative modeling applications.

**Normalized Second Order Exponential Kernel Functions** As part of our work for estimating $f_{RMSD}$, and developing the generalized protein structure annotation toolkit, ProSAT we developed a novel kernel function, the normalized second order exponential kernel function ($nsoe$ given by Equations 4.4 and 4.5 in Section 4.4.3 on page 77). This $nsoe$ kernel function captures the pairwise dependencies among the residues used at various positions within each subsequence or $w_{mer}$ (used for encoding of residues) better in comparison to the $rbf$ kernel. Just like the $rbf$ and Gaussian kernels [207], the exponential component of the $nsoe$ kernel captures non-linear relationships within the data as well.

For the $f_{RMSD}$ estimation problem the performance of the $nsoe$ kernel is 3-13% better than the $rbf$ kernel (see Section 4.5.4 on page 82). The results presented for the local structure and function prediction
problems evaluated using PROSAT shows that the \textit{nsoe} kernel generally outperforms the \textit{rbf} kernels on the wide-range of studied problems (see Section 5.4.3 on page 106). These performance improvements reported are statistically significant as evaluated by a student t-test.

**Generalized Framework for Local Structure and Function Prediction**  
Our work develops a generalized protein sequence annotation framework using kernel-based techniques. The generalized framework was motivated by the fact that there are countless problems in computational biology that use kernel-based methods [181], and the method developed for different prediction problems followed a similar protocol. Our developed framework provides novice users to use any input information in the form of feature matrices and predict a discrete class label or estimate a continuous valued annotation for each residue.

PROSAT was evaluated by training multiclass classification and binary classification models for predicting local structure, solvent accessibility, transmembrane helical regions, disorder prediction, and protein-dna interaction site prediction. We have also tested regression models for residue-wise contact order estimation and solvent accessibility prediction. The performance of PROSAT was comparable to custom-tailored solutions for each of these different prediction problems.

We also advocate the use of variable-width windows around sequence positions. Our empirical results show that for the disorder prediction problem, only rough information about distant sequence neighbors may be required for accurate predictions. We explore this issue by examining the performance trade-off between fine-grained near-neighbor and coarse-grained distant-neighbor information.

PROSAT was has already been used in two separate other studies. Firstly PROSAT’s accurate residue-wise transmembrane helix predictions were used within a framework for identifying transmembrane helix segments as well as orientation of these segments. This method called TOPTMH [3] used PROSAT’s prediction as a base classification, integrating it with hydrophobicity information and other physico-chemical properties and modeling the different components using a hidden markov model. Secondly, another study used PROSAT for predicting ligand-binding site residues. These binding site predictions were successfully incorporated into improving the homology modeling of binding-site regions, by using a target-template alignment that used predicted binding site information [105].

PROSAT along with MONSTER provides to the practitioners an efficient and easy-to-use tool for a wide variety of annotation problems. The results of some of these predictions can be used to assist in solving the overarching 3D structure prediction problem and assist in inferring the function of protein.

## 7.2 Future Research Directions
**Integrating Heterogeneous Data**  In our study for developing predictive models for protein structure prediction model, we integrate different sequence derived information (e.g., profile information, predicted secondary structure information, amino acid substitution matrix information) by using a linear weighted combination of different kernels associated with each information type. A weighted combination of kernel matrices leads to an admissible or valid kernel as well [207]. The weights for the different kernels are determined using a grid search, and are shown to be effective.

Recently, there have been several attempts to combine the information from heterogeneous data sources by learning weights across the different kernels using semi-definite programming [119, 143], second order cone programming [9], semi-infinite linear programming [193, 165], sparsity exploiting semi-definite programming approaches [204], or a Bayesian framework [39]. Such approaches fall under the class of problems called multiple kernel learning that use a convex programming based method for learning the optimal weights across the different kernels for training predictive models. A limitation of such approaches is that these methods have limited scalability, and do not work for large training sets [118].

**Protein Sequence Classification**  The success of developed profile kernels [155] and hierarchical multiclass classification [157] for remote homology detection and fold recognition using sequence information is applicable to other sequence classification tasks as well. Inferring functional characteristic of proteins like protein sub-cellular localization [122, 219, 222, 16], enzyme classification [154], function classification [81, 12] using the Gene Ontology (GO) definitions are interesting and challenging problems where kernel-based methods have been used. The work by Boden et al [18] to predict nucleolar proteins using support vector machines builds upon the profile-based kernels developed by us to train highly accurate predictive models.

Classifying proteins into functional classes using definitions like the GO database also has a hierarchical structure that relates the various classes. Thus our work on using the hierarchical structure prevalent in the SCOP database [157] can be extended for solving the GO classification problem. The challenge is that unlike the problems of superfamily and fold classification, function prediction problems are multi-labeled i.e., a single protein can perform two or more functions. Within this context, a Bayesian framework that models the contextual information within the hierarchical structure of the GO database shows promising results [12].

Integrating the outputs of the one-versus-rest classifiers using the framework we developed would have to be changed to account the multi-label constraint. In such a case, the evaluation metric would not necessarily be error rate but would evaluate the prediction results using metrics that measure the ranking performance. The problem of protein function prediction is very subjective, and the problem being solved depends on the database and methodology used. A comprehensive survey by Pandey et al [146] discusses the span of
problems and methods related to function prediction. In this Section we propose solving a multi-label multi-class classification, and we hypothesize that our window based kernels should perform better in comparison to the local alignment kernels because different parts of the protein may be involved in different functional activities.

**Protein Ligand Modeling and Interaction Studies** In order to perform its function a protein (typically enzyme) must bind to another small molecule called ligand. Two proteins binding to the same molecule may perform the same function, and as such proteins may have multiple binding partners and binding sites [84, 106]. Thus predicting the function of proteins using sequence as well as structure information could be modeled as a function of its binding site.

Also, such target-ligand activity relationships forms the underlying basis for chemogenomics studies [24, 21, 71, 66, 168, 59] that attempt to identify active compounds that bind to a new target within the same gene family. This has applications in the discovery of new therapeutic targets, which assists in the discovery of novel drugs.

The work presented in this thesis has already seen applicability in modeling of ligand binding sites using protein sequence information, only [105]. Using PROSAT’s residue annotation framework a sequence-based prediction model was trained to identify potential binding sites. Further, only this binding site was modeled using comparative modeling. The target-template alignment methodology used in the study incorporated a score based on whether the residue-pairs between the template and target proteins had a matching binding site or not.

Using the information of protein sequence or structure around the binding site, an function classification methodology could be developed. Unlike the previously proposed solution that used the entire sequence for the classification new kernels could be developed that would utilize the properties of binding site residues.
Bibliography


Notations and Definitions
### Table A.1. Key Notations and Symbols

<table>
<thead>
<tr>
<th>Notation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{P}$</td>
<td>Position Specific Scoring Matrix.</td>
</tr>
<tr>
<td>$\mathcal{F}$</td>
<td>Position Specific Frequency Matrix.</td>
</tr>
<tr>
<td>$\mathcal{S}$</td>
<td>Position Specific YASSPP-derived Scoring Matrix.</td>
</tr>
<tr>
<td>GSM</td>
<td>Global Scoring Matrix (BLOSUM62).</td>
</tr>
<tr>
<td>$\mathcal{K}$</td>
<td>Kernel Function</td>
</tr>
<tr>
<td>$w$-mer</td>
<td>$(2w+1)$-length subsequence.</td>
</tr>
<tr>
<td>$w$score</td>
<td>Score of the $w$-mer using a particular scoring function.</td>
</tr>
</tbody>
</table>

**Kernel Functions (Similarity Measures)**

<table>
<thead>
<tr>
<th>Method</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>AF-PSSM</td>
<td>All fixed-length $w$-mers using $\mathcal{P}$.</td>
</tr>
<tr>
<td>BF-PSSM</td>
<td>Best fixed-length $w$-mers using $\mathcal{P}$.</td>
</tr>
<tr>
<td>BV-PSSM</td>
<td>Best variable-length $w$-mers using $\mathcal{P}$.</td>
</tr>
<tr>
<td>SW-PSSM</td>
<td>Smith-Waterman based alignment using $\mathcal{P}$.</td>
</tr>
<tr>
<td>AF-GSM</td>
<td>All fixed-length $w$-mers using GSM.</td>
</tr>
<tr>
<td>BF-GSM</td>
<td>Best fixed-length $w$-mers using GSM.</td>
</tr>
<tr>
<td>BV-GSM</td>
<td>Best variable-length $w$-mers using GSM.</td>
</tr>
<tr>
<td>SW-GSM</td>
<td>Smith-Waterman based alignment using GSM.</td>
</tr>
</tbody>
</table>

**Coding Schemes for multiclass classifier**

<table>
<thead>
<tr>
<th>Method</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S$</td>
<td>Simple Scaling</td>
</tr>
<tr>
<td>SS</td>
<td>Scale &amp; Shift</td>
</tr>
<tr>
<td>CS</td>
<td>Crammer-Singer</td>
</tr>
</tbody>
</table>

**Incremental Window-based Alignment Algorithms**

<table>
<thead>
<tr>
<th>Method</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>CA$^f$</td>
<td>Central Alignment using fixed-length $w$-mers.</td>
</tr>
<tr>
<td>SA$^f$</td>
<td>Subset Alignment using fixed-length $w$-mer.</td>
</tr>
<tr>
<td>CSA$^f$</td>
<td>Central and Subset Alignments Scheme using fixed-length $w$-mers.</td>
</tr>
<tr>
<td>CA$^v$</td>
<td>Central Alignment using variable-length $w$-mers.</td>
</tr>
<tr>
<td>SA$^v$</td>
<td>Subset Alignment using variable-length $w$-mers.</td>
</tr>
<tr>
<td>CSA$^v$</td>
<td>Central and Subset Alignment using variable-length $w$-mers.</td>
</tr>
</tbody>
</table>

**Local Structure Quality Prediction**

<table>
<thead>
<tr>
<th>Method</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{PS}_{pic}$</td>
<td>Picasso Scoring Function.</td>
</tr>
<tr>
<td>dotp</td>
<td>Dot-product.</td>
</tr>
<tr>
<td>pair</td>
<td>Pairwise Coding Scheme.</td>
</tr>
<tr>
<td>conc</td>
<td>Concatenation Coding Scheme.</td>
</tr>
</tbody>
</table>