PROTEIN HOMOLOGY DETECTION WITH SPARSE MODELS

BY PAI-HSI HUANG

A dissertation submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Computer Science

Written under the direction of

Professor Vladimir Pavlovic

and approved by

_____

_____

_____

_____

New Brunswick, New Jersey

October, 2008

ABSTRACT OF THE DISSERTATION

Protein Homology Detection with Sparse Models

by Pai-Hsi Huang

Dissertation Director: Professor Vladimir Pavlovic

Establishing structural or functional relationship between sequences, for instance to infer the structural class of an unannotated protein, is a key task in biological analysis. Protein sequences undergo complex transformations such as mutation, insertion and deletion during the evolutionary process and typically share low sequence similarity on the superfamily level, making the task for remote homology detection based on primary sequence only very challenging.

Based on previous studies stating that knowledge based on only a subset of critical positions and the preferred symbols on such positions are sufficient for remote homology detection, we present a series of works, each enforcing different notion of sparsity, to recover such critical positions. We first start with a generative model and present the sparse profile hidden Markov models. Such generative approach recovers some critical patterns and motivates the need for discriminative learning. In our second study, we present a discriminative approach to recover such critical positions and the preferred symbols. In our third study, we address the issue of very few positive training examples, accompanied by a large number of negative training examples, which is typical in many remote homology detection task. Such issue motivates the need for semi-supervised learning. However, though containing abundant useful and critical information, large uncurated sequence databases also contain a lot of noise, which may compromise the

quality of the classifiers. As a result, we present a systematic and biologically motivated framework for semi-supervised learning with large uncurated sequence databases. Combined with a very fast string kernel, our method not only realizes rapid and accurate remote homology detection and show state-of-the-art performance, but also recovers some critical patterns conserved in superfamilies.

Acknowledgements

I would like to thank my advisor, Professor Vladimir Pavlovic, for his guidance, support and encouragement throughout my Ph.D. studies. I came to him without proper mathematical and analytical skills. Under Professor Pavlovic's supervision, I gradually developed the necessary skills and have become more confident. One of the many great things I learned from him is from his saying: *Pai-Hsi, show me the math*, when early in my Ph.D. career, I used to tell him that I *strongly feel* that some mathematical equation can be established. Ever since then, when meeting with him, I always prepare mathematical derivation for his inspection, which in turn hones my analytical skills.

Second, I want to thank my committee member Professor Ali Shokoufandeh. Ali graduated from the Computer Science Department in Rutgers and upon graduation, he started his academic career teaching in Drexel University in Philadelphia, where I completed my undergraduate study. In my senior year, I took his class, Design and Analysis of Data Structures and Algorithms and during the semester he encouraged me to come to Rutgers. Without his encouragement, I cannot imagine having such academic achievement.

Third, I want to thank my committee member Professor Casimir Kulikowski. Professor Kulikowski also served as my committee member in my qualifying exam and his advice significantly influenced my research direction. Based on his advice, I was also able to benefit from my internship experiences.

Fourth, I want to thank the rest of my committee member, Professor Dimitris Metaxas, for his time and interest in this dissertation.

I also want to thank my collaborators. Professor Alexander Kister's previous works motivated the first two major studies in this thesis. I also learned a lot from another collaborator, Pavel Kuksa. Working with Pavel, I learned so many valuable lessons.

For example, *there is always a better and faster way* and *there is still room for improvement*. His insisting in producing high-quality work directly benefited the third part of my thesis: we were able to develop a framework that strongly outperform previously established state-of-the-art methods. Without his help, it will be very difficult to do it all by myself.

Finally, I want to thank two very good friends of mine. I met them when I just came to the United States, alone: Christopher Rajashekar and David Czapla. Pursuing a Ph.D. degree not only takes a lot of dedication but also requires a lot of support, especially for someone whose family is thousands of miles and 12 time zones away. Many a times when I hit a bottleneck and could not see my future clearly, I experienced fear. Their strong support helped me overcome the emotional hardships so that I could continue. They are my second family.

Dedication

To my parents: Shang-Chang Huang and Hsiu-Chin Lee. Without their continuous support and love, I cannot achieve anything.

To my family and friends in Taiwan: your love and support travel 12 time zones and make everything possible.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

One of the main problems of the post-genomic era is the ability to classify a genomic or amino acid sequence into its proper protein family, and thereby to predict, to some degree of approximation, its structure and function. However, the gap between the decoded sequences and classified proteins is quickly widening with the advent of large-scale sequencing techniques. Currently there are more than 61 million DNA sequences in GenBank [6] and approximately $349,480$ annotated and 5.3 million unannotated sequences in UNIPROT [3], thus making development of computational aids for sequence annotation based on *primary sequence only* a critical and timely task. In this work, we focus on the problem of predicting protein remote homology (superfamily) based on the primary sequence information since it is inexpensive.

In the context of Structural Classification Of Proteins (SCOP) [47], a manually curated protein data set derived from PDB [7], sequences are grouped in a tree hierarchy containing classes, folds, superfamilies and families, from the root to the leaves of the tree. The leaf level represents the family; proteins in the same family have clear evolutionary relationship. Proteins in the same superfamily, one level above, are likely to share a common evolutionary origin and therefore perform similar functions. Proteins in the same fold, one level further above, share a common three-dimensional pattern and have the same major secondary structures in the same arrangement and with the same topological connections. Remote homology detection means classification of protein sequences on the superfamily level. The difficulty of the task arises from low primary sequence identities among proteins in the same superfamily. We depict the SCOP assignment in Figure 1.1.

Initial approaches to computationally-aided homology detection involved methods

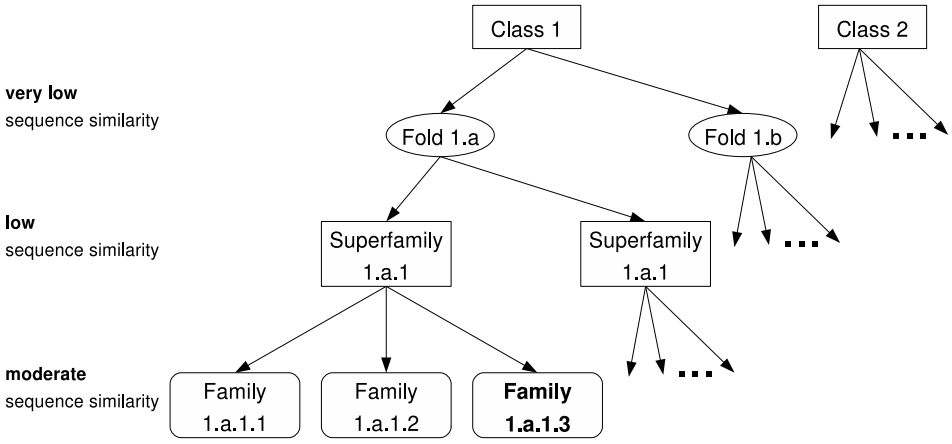


Figure 1.1: Structural Classification of Protein (SCOP) [47]: proteins in the same family have clear evolutionary relationship and moderate sequence conservation; proteins in the same superfamily are likely to share a common evolutionary origin and are typically low in sequence conservation.

such as BLAST [62] and FASTA [63], both of which rely on aligning the query sequence to a database of known sequences (pairwise alignment). However, the weakness of the pairwise approach is its lack use of data: alignment is performed on the query sequence to *each* of the sequences in the database *one at a time* without leveraging knowledge from multiple sequences exhibiting similar features. To overcome the shortcomings of the pairwise comparison framework, later methods based on *profiles* [24] and *profile hidden Markov models* [19] collect aggregate statistics from a group of sequences known to belong to the (super)family of interest. The statistics are compiled to construct one or *a library of models* [23]. Upon query time, we align an unknown sequence to all models derived from multiple alignments to detect a significant *hit*.

Profile hidden Markov models have demonstrated great success in protein homology detection. The linear structure of a profile HMM offers direct interpretation to the underlying process that generates the sequences. However, as *generative* models, profile HMMs are estimated from sequences known to belong to the same functional group (positive sequences) and do not attempt to capture the differences between members (positive sequences) and non-members (negative sequences). Also, various studies have shown that profile HMMs are unable to detect members with low sequence identity, which typically occurs in *remote* homology detection problems (on the superfamily

level). On the one hand, there is no doubt that the methods described above have been very successful, but on the other hand, they all become less reliable when more distant, less homologous proteins are considered.

To overcome the deficiencies of generative models, Jaakkola *et al.* proposed *SVM-Fisher* in [30]. The idea is to combine a generative model (profile HMM) with a discriminative model (support vector machines, SVM) and perform homology detection in two stages. In the first stage, the generative model, *trained with positive sequences only*, extracts fixed-length features from all sequences (*positive and negative*). In the second stage, given the features, the discriminative model constructs the decision boundary between the two classes. Also, in [45] Liao *et al.* estimate the similarity between two sequences induced from a *generative* model. The authors then use the estimated similarity values to construct a *kernel matrix* and perform remote homology detection with kernel-based learning techniques (SVM). Finally, in [35] Kuang *et al.* represent each sequence as a *generative* model and define the similarity between two sequences as the size of intersection of the local neighborhood induced by the generative models.

The class of *direct string kernels*, on the other hand, bypasses the need of a *generative* model and directly estimate the decision boundaries of the *discriminative* models. Similar to the profile kernel, the *spectrum kernel* [42] and the *mismatch kernel* [43] determine the similarity between two sequences as a function of the size of the intersection of the local neighborhood induced by the *observed* substrings within the sequences. The two methods operate *directly* on the strings without estimating any generative model. The same group of authors also proposed similar direct string kernels based on related principles such as the *gapped kernel*, the *substitution kernel* and the *wildcard kernel* in [41]. However, among all the proposed kernels, the mismatch kernel remains the state-of-the-art. Finally, in [46], Lingner *et al.* proposed a class of kernels based on the *distance distribution* among substrings in the sequences. The distance-based direct string kernels outperform the mismatch kernel.

We summarize the crucial differences between the generative and discriminative models in the following. First, discriminative learning involves *participation of negative examples* whereas in generative learning *only positive examples* participate in the

estimation procedure. Consequently, in discriminative learning we use more samples (positive and negative) to estimate a classifier. A discriminative classifier usually has fewer parameters to estimate when compared to its generative counterpart. Second, discriminative learning focuses on identifying *the difference between two groups of examples* whereas generative learning focuses on modeling the *commonly shared characteristics within a group*. Since there is no guarantee that no negative examples can posses such a characteristic, generative models are more susceptible to falsely detect a hit than the discriminative models. The advantage of the discriminative models for the protein remote homology detection task largely comes from the use of negative examples since labeled examples are scarce. We will further discuss generative and discriminative learning in Section 2.1.

In this study, we present a series of methods for the remote homology detection task. First in Chapter 3 we start from a set of *sparse generative* models that focus on a small set of positions and residues in protein sequences, based on studies [34, 55, 33] showing that a small number of *key residues* in a sequence (some 10-20% of the sequence) are characteristic and specific for a given superfamily. In the study, we describe an automated procedure to uncover such critical positions and encode the captured information as a set of probabilistic patterns in *sparse* profile HMMs. We show that a four-fold compression in model complexity can be achieved while maintaining the performance. Our further investigation reveals that while some remotely homologous (positive) sequences *skip* some less conserved critical positions, another group of unrelated (negative) sequences partially conform to the imposed probabilistic pattern, which motivate the use of *discriminative models* for the protein remote homology detection task, since the *generative models* focus on capturing the commonly shared characteristics among a group and do not guarantee that such characteristic will be absent in other unrelated groups.

Next, in Chapter 4 we present a *hybrid* framework as in [30] and propose a set of biologically motivated features extracted from the *generative* models. We also propose to use a *sparse discriminative* classifier to select the *key features*. We test the hypothesis again on a larger benchmark data set used in various studies and show that the combination of the features and sparse classifiers demonstrate comparable performance

with the state-of-the-art methods. The combination of the biologically motivated features and the sparse discriminative classifier also offers *simple* and *biologically intuitive* interpretation for each superfamily. We are able to recover the *critical positions*, the positions that are conserved throughout evolution.

In Chapter 5, we attempt to take advantage of the presence of hierarchy of class exhibited in protein sequences. We employ the *joint training and feature sharing* learning paradigm by experimenting on two sets of features. First a generative model extracts the sufficient statistics (model-based) as features and second a set of features directly derived from the observed sequences (string-based). Under such learning setting, our experiments show that while use of sufficient statistics demonstrate good performance with very few features, relying on different feature extractors for different superfamilies hinders joint training and feature sharing. On the other hand, though joint training and feature sharing is possible using features that are directly derived from the observed strings, the high dimensionality and complex correlation structure among the string features severely degrade the performance. As a result, features that have fixed dimensionality and are mutually orthogonal might have the most potential of benefiting from the joint training and feature sharing setting.

Finally, in Chapter 6 we present a systematic and biologically motivated framework for leveraging unlabeled data using large uncurated sequence databases. First, we propose to use a previously established kernel, the Sparse Spatial Sample Kernels (SSSK). This class of biologically motivated kernels model mutation, insertion and deletion effectively and induce low-dimensional feature space; moreover, the computational complexity of kernel evaluation based on feature matching is independent of the size of the alphabet set and such key characteristics opens the door for rapid large-scale semi-supervised learning. Second, we propose a biologically meaningful way of extracting relevant information from the unlabeled database for semi-supervised learning. Third we propose a method to remove the bias caused by overly represented or duplicated sequences which are commonly seen in uncurated sequence databases. Our experimental results show that the combination of these approaches yields state-of-the-art performance that are significantly better than previously published methods and also exhibit

order-of-magnitude differences in experimental running time.

We organize our work in the following way. In Chapter 2, we present an overview of the computational tools that we use for the study. We keep the discussion at a general level and leave the specific details regarding how the methods are used for later chapters. In the subsequent four chapters, we demonstrate in detail the three pieces of major works mentioned in the previous paragraph chronologically. Finally, we give a conclusion of our study and discuss our future works.

Chapter 2

Background and Related Works

2.1 Generative and Discriminative Learning

In a classification problem, one aims to learn or approximate an unknown *target* function $f : X \rightarrow Y$, or $P(Y|X)$, where X is a random sample drawn from the *sample space* and Y the label or class of X. When using generative classifiers we search for the parameters $\theta^* = \text{argmax}_{\theta \in \Theta} P_\Theta(X, Y)$ by maximizing the *joint probability*, where Θ denotes the parameter space. We then obtain $P_\theta(Y = y|X = x)$, the probability of an example $X = x$ belonging to class $Y = y$ using the *Bayes Rule*:

$$P_\theta(Y = y|X = x) \quad = \quad \frac{P_\theta(X = x|Y = y)P_\theta(Y = y)}{\sum_{y' \in Y} P_\theta(X = x|Y = y')P_\theta(Y = y')}. \tag{2.1}$$

With discriminative classifiers on the other hand we search for $\theta^* = \text{argmax}_\theta P_\theta(Y|X)$ by directly maximizing the *conditional probability*. Once we obtain the optimal parameters with respect to the model, we estimate $P(Y = y|X = x)$ directly with θ^*. For both classes of classifiers, the final prediction usually is made using the following decision rule:

$$y^* \quad = \quad \underset{y \in Y}{\text{argmax}}\, P_{\theta^*}(Y = y|X). \tag{2.2}$$

Estimating the parameters of a generative model typically involves estimating $P(Y = y)$ and $P(X = x|Y = y), \forall x \in X, y \in Y$. The second term requires knowledge or assumption of the sample space under each label, y. However, when such assumption is inconsistent with the truth, the injected bias will degrade the performance of the estimated classifiers. Discriminative classifiers on the other hand make no such assumption and estimate $P(Y = y|X = x)$ directly. As a consequence, discriminative models are considered more robust. Comparisons between these two types of classifiers have been

the interest of various studies [50, 54, 57]. In the following, we motivate such comparison using two related classifiers: the *Naive Bayes* classifier as a *generative* model, and the *logistic regression* classifier as a *discriminative* model, as was done by Ng *et al.* in [50] and Mitchell in [49].

The generative classifiers operate under the assumption that given the class label, the distribution of the sample is known. Let $X = [X^{(1)},\ X^{(2)},\ \cdots,\ X^{(d)}]$, a *random vector* with d random variables. Then for each class $y \in Y$, in addition to $\theta_y = P(Y = y)$ we also need to estimate:

$$P(X^{(1)} = x^{(1)}, X^{(2)} = x^{(2)}, \cdots, X^{(d)} = x^{(d)} | Y = y). \tag{2.3}$$

If all features are *discrete*, without knowing the dependencies among the features, the number of parameters we need to estimate in Equation 2.3 is *exponential* in d, leading to the problem known as the *curse of dimensionality*. The Naive Bayes classifiers further assume that *all features are mutually independent given the class*. Consequently, given the class, the joint data likelihood reduces to a product:

$$\begin{aligned} P(X^{(1)} = x^{(1)}, X^{(2)} = x^{(2)}, \cdots, X^{(d)} = x^{(d)} | Y = y) &= \prod_{i=1}^{d} P(X^{(i)} = x^{(i)} | Y = y) \\ &= \prod_{i=1}^{d} \theta_{yi}. \end{aligned} \tag{2.4}$$

Equation 2.4 indicates that the number of parameters we need to estimate now is *linear* in d. We show a Naive Bayes classifier in Figure 2.1(a). In the figure, the arrows point from the label Y to the sample space X, indicating assumption or knowledge of the sample distribution under the label. Further, absence of arcs among all random variables suggests that all random variables are *mutually independent* given the label (conditional independence). We also show a logistic regression classifier, the discriminative counterpart of Naive Bayes classifiers in Figure 2.1(b). Note that in the figure, the arrows point in the reversed direction, indicating absence of assumption regarding the sample distribution.

Like the Naive Bayes classifiers, the logistic regression classifiers also assume *mutual independence* among all random variables. Denote $|Y|$ as the number of classes, the parameters need to be estimated are $\theta_{yj}, 1 \leq y \leq |Y| - 1, 0 \leq j \leq d$, where $j = 0$

(a) naive Bayes classifier (generative)

(b) logistic regression classifier (discriminative)

Figure 2.1: Left panel: A Naive Bayes classifier; The arrows point from the label Y to the sample space X, indicating assumption or knowledge of the sample distribution given the label. Further, absence of arcs between all pairs of random variables indicates that all random variables are mutually independent given the class (conditional independence). Right panel: A Logistic Regression classifier, the discriminative counterpart of a Naive Bayes classifier. Note that the arrows point in the reversed direction, indicating absence of assumption regarding the sample distribution.

corresponds to the constant term[1]. Let $\theta_y = \{\theta_{yj}\}_{j=0}^{d}$; the parametric form of the posterior probability of logistic regression is:

$$P(Y = y|X = x) = \begin{cases} \frac{\exp(\theta_y^T x)}{1+\sum_{y'=1}^{|Y|-1}\exp(\theta_{y'}^T x)} & \text{if } y < |Y|, \\ \frac{1}{1+\sum_{y'=1}^{|Y|-1}\exp(\theta_{y'}^T x)} & \text{if } y = |Y|. \end{cases}$$

When the classification problem is *binary*, *i.e.* $|Y| = 2$, we have

$$P(Y = 1|X = x) = \phi(\theta_1, x) \quad = \quad \frac{\exp(\theta_1^T x)}{1 + \exp(\theta_1^T x)}, \tag{2.5}$$

where $\phi(.)$ is the cumulative distribution function (CDF) of a *logistic* distribution.

To establish the connection between the Naive Bayes and logistic regression classifiers, consider a *binary-class* classification problem with only binary inputs. Under the Naive Bayes classifier, we estimate $P(Y = 1|X = x)$ using the Bayes rule. With some simple algebraic manipulation, as discussed in [49], we obtain the following:

$$\begin{aligned} P(Y = 1|X = x) &= \frac{P(X = x|Y = 1)P(Y = 1)}{\sum_{y\in\{0,1\}} P(X = x|Y = y)P(Y = y)} \\ &= \frac{\theta_1 \prod_{i=1}^{d} \theta_{1i}^{x_i}(1-\theta_{1i})^{1-x_i}}{\theta_1 \prod_{i=1}^{d}(\theta_{1i})^{x_i}(1-\theta_{1i})^{1-x_i} + \theta_0 \prod_{i=1}^{d}(\theta_{0i})^{x_i}(1-\theta_{0i})^{1-x_i}} \\ &= \frac{1}{1 + \exp((\ln\frac{\theta_0}{\theta_1} + \sum_{i=1}^{d}\ln\frac{1-\theta_{0i}}{1-\theta_{1i}}) + \sum_{i=1}^{d} x_i(\ln\frac{\theta_{0i}(1-\theta_{1i})}{\theta_{1i}(1-\theta_{0i})}))} \\ &= \frac{1}{1 + \exp([1\ x^T]\tilde{\theta})}. \end{aligned} \tag{2.6}$$

[1]We need to augment the random vector X with $X' = [1\ X^T]^T$

Note that Equation 2.6 is exactly the parametric form of a logistic regression classifier, suggesting that *we can train a generative model discriminatively*: instead of maximizing the *joint likelihood* in Equation 2.4 and obtaining $\{\theta_y\}_{y\in Y}$ and $\{\theta_{yi}\}_{i=1}^{d}$, by maximizing the *conditional likelihood* of a Naive Bayes classifier directly with respect to $\tilde{\theta}$, we obtain a discriminative logistic classifier. The logistic classifier is also more *economical* in the sense that we only need to estimate $(|Y|-1)(d+1)$ parameters, while for Naive Bayes classifiers, we need to estimate $|Y|(d+1)$ parameters. Equation 2.6 also suggests that when the variables are discrete, both Naive Bayes and logistic regression classifiers define *linear* decision boundaries in the input space. Such statement is in general not true when the variables are continuous. Mitchell showed in [49] that the decision boundary is still linear when the variables are normally distributed and share a common covariance matrix Σ; however, when the variables do not share a covariance matrix, the decision boundary is *quadratic* in the input space.

To estimate the parameters of a Naive Bayes classifier, we apply the *maximum likelihood* or *maximum a posterior* principle and obtain the estimates in closed-form. In contrast, estimating the parameters of a logistic regression classifier is more computationally intensive and relies on iterative procedures. Several methods have been proposed, for example the *Iteratively Reweighted Least Squares* (IRLS) or the cyclic coordinate ascent algorithms. Since the objective function is convex, under mild conditions (full rank outer product matrix) the estimate is guaranteed to be *globally* optimal.

In [50] Ng *et al.* discussed the accuracy and convergence properties of the Naive Bayes and logistic regression classifiers. The authors concluded that *regardless of the truth or falsity of the assumption over the sample distribution under the labels*; Naive Bayes classifiers converge to its (lower) asymptotic accuracy at a higher speed. However, with more training examples, the logistic regression classifiers finally reach its (higher) asymptotic accuracy.

2.2 Hidden Markov Models

Hidden Markov models (HMMs) are probabilistic graphical models frequently employed in signal processing, speech recognition, and sequence classification tasks. An HMM has a set of states, $S = \{s_1, s_2, \cdots, s_n\}$ and each state, when visited, emits an observation, chosen from a finite discrete [2] alphabet set Σ based on a probability distribution specified by the *emission matrix* E; after a state s_i is visited, the next state s_j to be visited is selected based on another probabilistic distribution specified by the *transition matrix*, A; finally, π is a probability distribution over all states, specifying the probability of the random process starting at each state. We summarize the mathematical properties of Σ, A, E, and π in the following:

- Σ: a finite and discrete alphabet set with $|\Sigma| = m$.
- $\pi = [\pi_1\ \pi_2\ \cdots\ \pi_n] \in r^n$, where $r \in [0, 1]$ with $\sum_{i=1}^{n} \pi_i = 1$.
- $A \in r^{n \times n}$, where $r \in [0, 1]$: Row i corresponds to state s_i. Element A_{ij} specifies $P(S_{t+1} = s_j | S_t = s_i)$ with $\sum_{j=1}^{n} A_{ij} = 1, \forall i$.
- $E \in r^{n \times m}$, where $r \in [0, 1]$: Row i corresponds to state s_i. Element E_{il} specifies $P(X_t = \sigma_l | S_t = s_i)$ with $\sum_{l=1}^{m} E_{il} = 1, \forall i$.

The process terminates at a pre-specified time or when the end state, if any, is visited. We show a very simple HMM in Figure 2.2 in the state space.

HMMs are used when we only observe the output of a stochastic process and need to obtain information on the sequence of states that emit the sequential observations. We typically use an HMM to answer the following questions:

1. Given an observed string $X = x_1\ x_2\ \cdots\ x_{T_X}$, and a model $H(\pi, A, E, \Sigma)$, what is the most probable sequence of states, $y_1\ y_2\ \cdots\ y_{T_X}$, that generated X?
2. Given an observed string $X = x_1\ x_2\ \cdots\ x_{T_X}$, and an HMM model $H(\pi, A, E, \Sigma)$, what is $P(X|H)$, the probability that model H generated X?

[2] HMMs with continuous or mixed emission will not be discussed in this study.

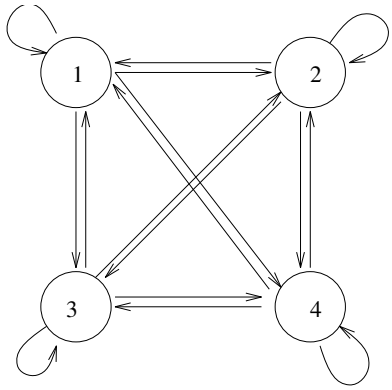


Figure 2.2: A Hidden Markov Model

3. Can we use our observation X to adjust H such that $P(X|H)$ is optimal? (Optimality in this context means maximizing $P(X|H)$ by adjusting the parameters in H.)

As *generative* models, HMMs only require *positive* examples assumed to be generated from the same source for training; an HMM attempts to capture common characteristics shared by all training examples. Once the model H is adjusted using the training examples with the *forward-backward smoothing* procedure (question 3), given an unknown *observed* sequence X^{new} we can obtain Y^*, the most likely *hidden* sequence of states that generated X^{new} with the *Viterbi* algorithm (question 1), or we can sum over all possible generating paths with *forward* or *backward* inference procedure (question 2) to obtain $P(X^{new}|H)$, the probability that H generated X^{new}. We note that the values $P(Y^*, X^{new}|H)$ and $P(X^{new}|H)$ directly depend on the length of the sequence $T_{X^{new}}$: the longer the sequence, the smaller the probability. As a result, when the final goal is to output a class membership (as oppose to labeling each observation with a state membership), a decision rule cannot be devised using such quantities alone. Common practices involve normalizing the negative log-likelihood, also called the *score* of a sequence with the sequence length, resulting in the following decision rule:

$$c(X) = sign(-\frac{1}{T_{X^{new}}} \log(P(X^{new}|H)) \quad > \quad \delta), \tag{2.7}$$

where δ is a pre-specified positive threshold. If the score is greater than δ, then we determine that the model H generated X^{new}. Alternatively, in some other applications there

might be $K > 1$ competing models with the prior probabilities $P(H^{(1)})$,$P(H^{(2)})$,$\cdots$,$P(H^{(K)})$; under such situation we assign the membership of X^{new} with the following decision rule:

$$k^* \quad = \quad \underset{k}{\text{argmax}} \frac{P(X^{new}|H^{(k)})P(H^{(k)})}{\sum_{k'=1}^{K} P(X^{new}|H^{(k')})P(H^{(k')})}. \tag{2.8}$$

Note this equation is consistent with Equation 2.2.

Finally, for a general HMM with n states, making inference on a sequence with length T takes $O(n^2T)$ time. More details of inference algorithms can be found in [53].

2.3 Logistic Regression Models

We discuss the logistic regression classifiers introduced in Section 2.1 in further details by assuming a *binary-class* setting (*i.e.* $Y \in \{-1, +1\}$). Generalization to a *multi-class* setting is straightforward and can be found in [26]. Given the training examples, $D = \{(x_i, y_i)\}_{i=1}^{n}$, the logistic regression model defines the probability of example x_i belonging to class $+1$ as:

$$P(y_i = +1|x_i, \theta) = \pi_i \quad = \quad \psi(\theta^T x_i) \tag{2.9}$$

with the *link function* $\psi(\cdot)$. When we use the cumulative distribution function (CDF) of a logistic distribution in place of $\psi(\cdot)$, we obtain a *logistic* regression model. Alternatively, when we use the CDF of the standard normal distribution, we obtain a *probit* [20, 21] regression model. Choice of different link functions usually does not affect the result in any significant way. In this study, we focus our discussion on the logistic model; To estimate the model, we search for θ^*,the parameter vector that maximizes the following joint likelihood function of the observed data, D:

$$P(D|\theta) \quad = \quad \prod_{i=1}^{n} \frac{1}{1 + \exp(-y_i x_i^T \theta)}. \tag{2.10}$$

Though no closed-form solution exists, any iterative gradient ascend method can be employed to find the globally optimal solution of the convex likelihood function.

Use of the logistic model provides a simple and intuitive description of data. If the assumption $P(y = +1|x, \theta) = \psi(\theta^T x)$ holds, then the contribution of each predictor variable $x^{(j)}, 1 \leq j \leq d$, is reflected in the corresponding model parameter $\theta^{(j)}$. For a

binary predictor, the corresponding parameter being large in absolute value suggests preference of *presence* (when positive) or *absence* (when negative) of feature $x^{(j)}$. For a continuous predictor, the interpretation is similar; however, caution must be exercised as the predictors may not have similar scales.

The parameter θ also offers a probabilistic interpretation. Define the *odds* of an event with probability p of occurring as $\frac{p}{1-p}$; then the odds of an example x belonging to class $+1$ is:

$$odds(y = +1|x, \theta) \quad = \quad \frac{\pi}{1-\pi} = \exp(\theta^T x), \tag{2.11}$$

where π is defined in Equation 2.9. Define a new example x', such that $x'^{(i)} = x^{(i)}, \forall 1 \leq i \leq d$, except $x'^{(j)} = x^{(j)} + 1$, meaning presence of the j^{th} feature (when feature is binary) or we increase the j^{th} predictor of x by one unit (when feature is continuous). The estimated odds of x' having label $+1$, is:

$$odds(y' = +1|x', \theta) \quad = \quad \exp(\theta^T x + \theta^{(j)}) = \frac{\pi}{1-\pi}\exp(\theta^{(j)}). \tag{2.12}$$

Equation 2.12 indicates that the odds now are multiplied by $\exp(\theta^{(j)})$ when we increase $x^{(j)}$ by one unit (when $x^{(j)}$ is continuous) or when the feature represented by $x^{(j)}$ is present (binary).

Finally, the *Bayesian learning paradigm* naturally incorporates our prior belief upon the structure of the models. If we expect the model to be *sparse* (most of the parameter values are zero), we place a *prior distribution* centered at the origin on the parameters. Two popular sparsity-enforcing priors are *Gaussian* and *Laplacian*. We will discuss the effects of these two priors on logistic regression models in Section 4.2.4.

2.4 Kernel-Based Learning

We motivate this section with the *least squares* (linear) regression problem. Given a set of input and output pairs $\{(x_i, y_i)\}_{i=1}^n$ with $Y \in R$, the goal is to estimate a function $f(x, \beta) = x^T\beta$ such that the sum of squared differences between the target outputs and the predicted outputs is minimum:

$$\hat{\beta} \quad = \quad \underset{\beta}{\operatorname{argmin}}(y - X\beta)^T(y - X\beta), \tag{2.13}$$

where $X = [x_1\ x_2\ \cdots\ x_n]^T$, the n-by-$(d+1)$ design matrix (with the intercept term), $y = [y_1\ y_2\ \cdots\ y_n]^T$ and all x_i's are augmented to accommodate the intercept term. If the inverse of the square matrix X^TX exists, we have a closed-form solution for the least squares problem:

$$\hat{\beta} \quad = \quad (X^TX)^{-1}X^Ty, \tag{2.14}$$

and the optimal hypothesis (parameter) $\hat{\beta}$ is unique. However, when the data set is small ($n < d$), the matrix X^TX is *singular* and its inverse does not exist. A general remedy for singularity is to add a constant λ to the diagonal of the matrix, resulting in the *ridge regression* problem. Consequently, with proper choice of λ the solution is:

$$\hat{\beta} \quad = \quad (X^TX + \lambda I)^{-1}X^Ty, \tag{2.15}$$

where I denotes a $(d+1)$-by-$(d+1)$ *identity matrix*.

We typically want a hypothesis that is consistent with the data and have low complexity, or equivalently, a hypothesis with *generalization* power for future unseen data to avoid *overfitting*. One popular and well-established method of controlling the complexity is to minimize the *norm* (length) of β and the sum of squares simultaneously. Minimizing the *2-norm* of β leads to the following objective function:

$$\hat{\beta} \quad = \quad \underset{\beta}{\operatorname{argmin}} \frac{1}{2}\lambda\beta^T\beta + (y - X\beta)^T(y - X\beta), \tag{2.16}$$

where $\lambda \in R$ controls how the regularization term $\beta^T\beta$ influences the final hypothesis. Note that if $\lambda = 0$, we have the original least squares problem in Equation 2.13. To obtain $\hat{\beta}$, we set the derivative of Equation 2.16 to 0. With some algebraic manipulations, as shown in the following, we have

$$\frac{\partial}{\partial\beta}[\frac{1}{2}\lambda\beta^T\beta + (y - X\beta)^T(y - X\beta)] \quad \stackrel{set}{=} \quad 0 \tag{2.17}$$

$$-X^Ty + (X^TX)\beta + \lambda\beta \quad = \quad 0 \tag{2.18}$$

$$\hat{\beta} \quad = \quad (X^TX + \lambda I)^{-1}X^Ty. \tag{2.19}$$

First, note that Equations 2.19 and 2.15 are identical and this establishes the connection between ridge regression and regularization. Moreover, adding a constant to the

diagonal of the matrix $X^T X$ is equivalent to placing a set of mutually independent zero-mean normal distributions as *prior distributions* on the hypothesis under the *Bayesian learning paradigm*[3]. Finally, deriving the solution in an alternative way reveals that β is a *linear combination of inputs*:

$$\frac{\partial}{\partial \beta}\frac{1}{2}\lambda\beta^T\beta + \sum_{i=1}^{n}\left(y_i - x_i^T\beta\right)^2 \stackrel{set}{=} 0 \tag{2.20}$$

$$\implies \hat{\beta} = \sum_{i=1}^{n}\left(\frac{y_i - x_i^T\beta}{\lambda}\right)x_i = X^T\alpha. \tag{2.21}$$

Expressing β as a linear combination of the input opens the door for *kernel-based learning*. Replacing β with $X^T\alpha$ in the objective function, we obtain:

$$\lambda\alpha^T XX^T\alpha + y^T y - 2y^T XX^T\alpha + \alpha^T XX^T XX^T\alpha \tag{2.22}$$

$$= \lambda\alpha^T K\alpha + y^T y - 2y^T K\alpha + \alpha^T KK\alpha. \tag{2.23}$$

The matrix $K = XX^T$ is an n-by-n *kernel matrix*[4], with each element $K(i,j)$ representing the inner product (similarity) between two examples x_i and x_j. The significance of the kernel matrix is that *training and testing examples need not have explicit representations*: all is required is the inner product of any two points in the feature space. Such property is crucial when the dimensionality of the feature space is high or when *mapping* the examples to a very high-dimensional space is necessary; Let $\phi(.) : x \mapsto \phi(x)$ be such a mapping function, with $\phi(x) \in R^{d'}$ and d' large. Explicit mapping of $\phi(x)$ and performing inner product directly in the kernel space might incur prohibitive computational cost; however, if we can perform *implicit* evaluation of $\phi(x)^T\phi(y)$ efficiently, we can apply kernel-based learning techniques to solve the problems at hand.

Solving the problems described in Equations 2.23 and 2.16 will yield identical solution. The crucial difference is that, in Equation 2.16 the hypothesis β is d-dimensional whereas in Equation 2.23, the hypothesis α is n-dimensional. In some problems where obtaining training points is costly and/or $d \gg n$, estimating β is impractical even with

[3]Minimizing 1-norm corresponds to placing a set of mutually independent *Laplacian* (double exponential) distributions as prior distributions on the hypothesis.

[4]Some literature refer to $X^T X$ as the *Gram* matrix.

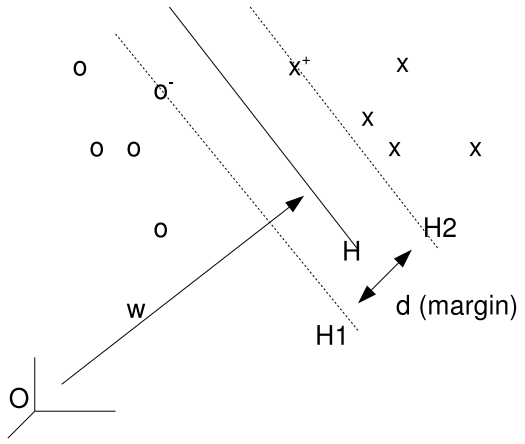


Figure 2.3: An example of two classes ('o' represents negative class) that are linearly separable. A support vector machines aims to maximize the margin between the two classes.

heavy regularization since inverting the outer-product matrix requires $O(d^3)$ time. Kernel learning offers a practical alternative: each α_i in α corresponds to an example x_i and obtaining α typically requires $O(n^3)$ time. Once we have α, we can obtain β using $\beta = \sum_{i=1}^{n} \alpha_i \phi(x_i)$, if explicit representation of $\phi(x)$ is available. The program described in Equation 2.23 is called the *dual* form of the *primal* program described in 2.16. Note that in the last term of the objective function in Equation 2.23, α appears as a *quadratic* term, suggesting that to solve α we need to solve a *quadratic programming* problem.

The analysis indicates that we can think of kernel-based learning as learning in the kernel-induced space with regularization. Once the kernel matrix is available, we can apply various data analysis techniques such as kernel nearest neighbor analysis, kernel Fisher discriminant analysis [48], kernel principal component analysis [59], and kernel logistic regression [70]*etc.* In the next section, we introduce support vector machines (SVM), a class of *discriminative* classifier that learns in the kernel space.

2.5 Support Vector Machines

Given a set of *linearly separable* training data, $\{(x_i, y_i)\}_{i=1}^{n}$, $X \in R^d, Y \in \{-1, +1\}$, a support vector machine (SVM) [67, 15, 12] aims to produce a hypothesis that is consistent with the data and maximizes the *margin*. The example is depicted in figure 2.3 with the he symbol 'o' representing the negative class and the symbol 'x' the positive

class. The hyperplane that maximizes the margin is H, characterized by its *normal* vector, w; the dotted hyperplanes $H1$ and $H2$ are both parallel to H with training examples lying on them. The margin is defined as the vertical distance between the parallel hyperplanes $H1$ and $H2$. Expressed mathematically, given a set of *linearly separable* data, an SVM solves the following problem:

$$\begin{aligned} \text{maximize :} \quad & d \\ \text{subject to :} \quad & x_i^T w + b \geq 1, \ \forall i : y_i = +1 \\ & x_i^T w + b \leq -1, \ \forall i : y_i = -1. \end{aligned} \tag{2.24}$$

For any point x^+ lying on $H2$ the following must be true:

$$(x^+)^T w + b \ = \ 1. \tag{2.25}$$

Likewise, for any point o^- lying on $H1$, we have:

$$(o^-)^T w + b \ = \ -1. \tag{2.26}$$

Equations 2.25 and 2.26 together imply:

$$\frac{(x^+ - o^-)^T}{w^T w} w \ = \ \frac{2}{w^T w} w. \tag{2.27}$$

Note that the term on the left-hand side is the projection of the vector $(x^+ - o^-)$ on w and the length of the projection is the margin, d:

$$\sqrt{(\frac{2}{w^T w})^2 w^T w} \ = \ \frac{2}{\|w\|_2} = d. \tag{2.28}$$

Equation 2.28 implies that *to maximize the margin d, we minimize the 2-norm of* w, leading to the following equivalent program:

$$\begin{aligned} \text{minimize :} \quad & \frac{1}{2}\|w\|_2^2 \\ \text{subject to :} \quad & x_i^T w + b \geq 1, \ \forall i : y_i = +1 \\ & x_i^T w + b \leq -1, \ \forall i : y_i = -1. \end{aligned} \tag{2.29}$$

The program in 2.29 seeks a hypothesis w, with minimum 2-norm, that is consistent with the data. Such formulation has been discussed in Section 2.4 as a form of *regularization*.

The term $\frac{1}{2}$ was added for mathematical convenience. The Lagrangian of the program is:

$$L_P(w, b, \alpha) = \frac{1}{2} w^T w - \sum_{i=1}^{n} \alpha_i [y_i (x_i^T w + b) - 1], \tag{2.30}$$

where all α_i's are the *Lagrange multipliers*. Setting the derivatives of the Lagrangian with respect to w and b to zero, we obtain:

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i = X^T \alpha \tag{2.31}$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0. \tag{2.32}$$

Equation 2.31 implies that the final solution w is a *linear combination of the input*. Substituting Equations 2.31 and 2.32 back into Equation 2.30, we obtain the dual form:

$$L_D(w, b, \alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \alpha^T K \alpha. \tag{2.33}$$

Therefore, maximizing the *primal* program in Equation 2.30 is equivalent to minimizing the *dual* program in Equation 2.33. The kernel matrix $K = XX^T$ in Equation 2.33 implies that SVM is capable of maximizing the margin and estimating a linear classifier in the (high-dimensional) kernel-induced space. Together with the kernel-based learning techniques discussed in Section 2.4, we can achieve such learning efficiently given that we can compute the kernel matrix efficiently.

The KKT complementarity conditions state that the optimal solution, (α^*, w^*, b^*) must satisfy the followings:

$$\alpha_i^* [y_i (x_i^T w^* + b^*) - 1] = 0, \text{ and} \tag{2.34}$$

$$\alpha_i^* \geq 0, \tag{2.35}$$

for all $1 \leq i \leq n$. Moreover, in Equation 2.34 either $\alpha_i^* = 0$ or $y_i (x_i^T w^* + b^*) - 1 = 0$ is true but not both. Therefore, we know that $\forall i : y_i (x_i^T w^* + b^*) - 1 \neq 0$, α_i^* must vanish and $\forall i : y_i (x_i^T w^* + b^*) - 1 = 0$, $\alpha_i^* > 0$. Consequently, the points with weight $\alpha_i^* > 0$ lie on hyperplanes $H1$ or $H2$ in figure 2.3 and are called the *support vectors*.

The support vectors jointly define the decision boundary H. The Lagrange multipliers α serve as dual variables and also give intuition about the importance of each training point. The examples with $\alpha_i^* = 0$ are not considered important and do not affect the decision boundary in any way; therefore, if removed, the decision boundary H stays intact whereas removing any examples with $\alpha_i^* > 0$ re-defines the decision boundary.

In many applications, expecting linear separability among the training examples in the kernel-induced space may not be practical since noise might be present in the data. Violation of the linear separability constraint results in no feasible region in the primal space and consequently, an unbounded objective function. To remedy such situation, we need to introduce the *slack variables* ξ. Each slack variable represents the vertical distance between the corresponding example and H, provided that the examples falls on the wrong side of H. With ξ, we have a new program:

$$\begin{aligned} \text{minimize :} \quad & \frac{1}{2}\|w\|_2^2 + C\|\xi\| \\ \text{subject to :} \quad & x_i^T w + b \geq 1 - \xi_i, \ \forall i : y_i = +1 \\ & x_i^T w + b \leq -1 + \xi_i, \ \forall i : y_i = -1, \\ & \xi_i \geq 0, \ \forall 1 \leq i \leq n \end{aligned} \tag{2.36}$$

where C is a parameter controlling the tolerance for misclassification. Similar analysis used earlier in this section can be applied to show that the dual form of program in 2.36 induces a kernel matrix and explicit representation of the data is not necessary. The support vector machines constructed from the linearly separable and non-linearly separable cases are also called the *hard margin* and *soft margin* support vector machines, respectively. When the classes are linearly separable in the kernel space, use of soft margin SVM usually results in smoother decision boundaries. Once we obtain α^*, the decision rule for classification given an example x is:

$$\hat{y} \quad = \quad sign(\sum_{i:\alpha^*>0} \alpha_i^* y_i K(x, x_i)). \tag{2.37}$$

Equation 2.37 only relies on the kernel value and again does not require explicit representation of any example. Additionally, only the support vectors participate in the decision, suggesting that SVM is a model sparse in the number of support vectors, not

to be confused with a model sparse in the number of non-zero parameters. Finally, one may use any standard quadratic programming package to obtain an SVM. For more details, please refer to [67, 12, 15].

2.6 Feature Sharing and Joint Training in Multi-class classification

Many frameworks have been proposed in the past decade to solve the multi-class classification problem ($|Y| > 2$). First, under the *one-vs-all*[5] framework, we independently train $|Y|$ *binary classifiers*, $f(Y = y|X),\ \forall y \in Y$, with positive examples from class y and negative examples from all other classes $y' \neq y$ and assign class membership to a new example x^{new} with Equation 2.2. Second, under the *all-vs-all*[6] framework, we estimate $\binom{|Y|}{2}$ *binary* classifiers for all pairs of classes $(y, y'),\ y, y' \in Y, y \neq y'$ and assign membership to a new example by *voting*. Third, In [68, 67], the authors proposed to solve a single optimization problem which simultaneously finds $|Y|$ classifiers, in contrast to the one-vs-all framework, where $|Y|$ optimization problems are solved *independently*. Finally, under the *error-correcting code* [17] framework, we train L [7] *binary* classifiers independently and combine the results of the binary classifiers to perform the classification task. Note that both one-vs-all and all-vs-all frameworks are special cases of the error-correcting code framework. Rifken *et al.* presents a more detailed review and comprehensive comparison of these approaches in [56].

In some applications, the classes form a hierarchy. For example, in an object classification problem, we expect the classes *television* and *computer monitor* to share many features and form a sub-cluster under a larger class, say, electronics. Likewise, proteins naturally form a hierarchy due to their structural and functional similarities. The classifiers trained under the *one-vs-all*, *all-vs-all* and *one-single-machine* frameworks mentioned in the previous paragraph do not exploit the presence of the hierarchy and the similarities among subsets of classes, which might result in sub-optimal performance of classifiers. In [65, 64], Torralba *et al.* proposed the idea of *joint training and feature*

[5] Also known as one-vs-rest.

[6] Also known as one-vs-one.

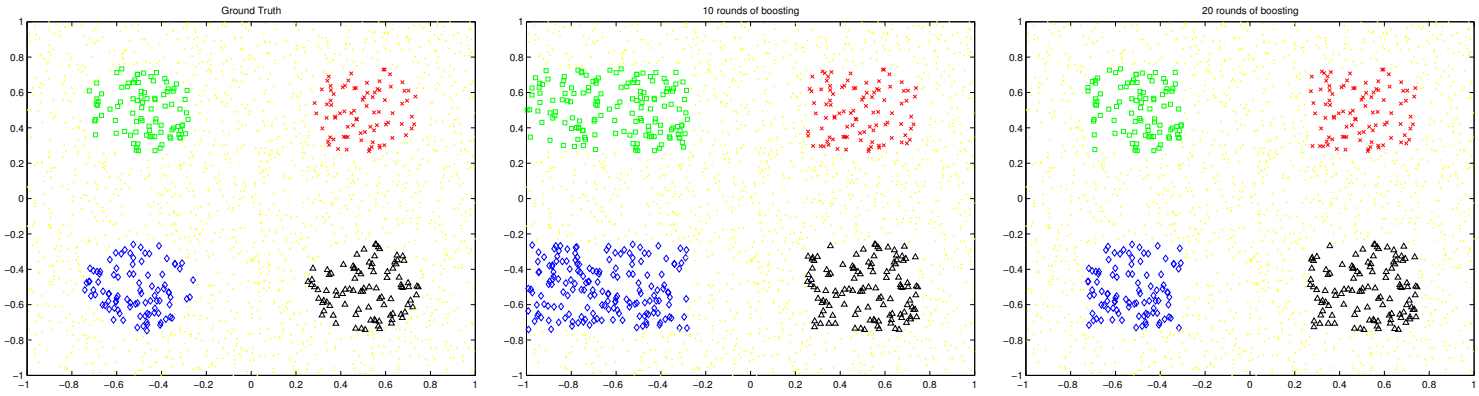
[7] L need not equal to $|Y|$.

sharing. The proposed method, closely related to the *error-coding* framework, attempts to exploit the presence of such existing hierarchy. The central idea is to form *mega-classes* in which all participating classes jointly share a feature. These mega-classes are then combined with *boosting* [58] to estimate a multi-class classifier capable of discriminating objects from $|Y| + 1$ class, where the last class is treated as *background* or *everything else*. The authors stated that the *joint training* and *feature sharing* framework needs fewer features to achieve a desired level of performance; additionally, more efficient use of training data also results in faster classification time. They support such statements with the observation of presence of feature sharing among several different but similar classes. In applications in which training examples are scarce and the classes form sub-clusters, making efficient use of data is very important. The joint training and feature sharing framework combines a series of weak classifiers that discriminate examples from a subset of classes against all other examples based on the shared feature. Such practice results in increase of positive examples (within the formed mega-classes).

The authors proposed to use *tree stumps* as the weak classifier required in the boosting framework; the tree stump in the m^{th} boosting round has the following form:

$$h_m(v, y) = \begin{cases} a\delta(x^f > \theta) + b & \text{if } y \in S(n), \\ k^y & \text{Otherwise,} \end{cases} \tag{2.38}$$

where f indexes the feature, n indexes all possible mega-classes, $\delta(.)$ the indicator function, $y \in Y$ and a, b, θ as well as k^y are the parameters to be estimated. The authors noted that the set of all possible mega-classes, S, in equation 2.38, is the power set of Y and therefore the proposed method incurs exponential complexity for training. As a remedy, they proposed using a *greedy* scheme to search for a combination of classes and a single feature in each boosting round: each iteration starts with the empty set being the mega-class and as the search proceeds, new classes are added *one at a time* to the existing mega-class. The greedy version of the proposed method now only requires $O(|Y|^2)$ time for training. After M boosting iterations we obtain a collection of tree stumps $H = \{h_m(v, y)\}_{m=1}^{M}$ as the estimated model. Given a new example x^{new} we associate with it $|Y|$ scores using H and assign the membership with the decision rule in Equation 2.2.



(a) (b) (c)

Figure 2.4: A toy example with four classes: red (crosses), green (squares), blue (diamonds) and black (triangles) with the yellow (dots) class as the background. The features are the vertical and horizontal coordinates. Each pair of neighboring classes shares a common feature. For example, the cross and triangle classes share the horizontal feature and therefore form a mega-class based on this feature; such feature must be within a range for an example to be considered a member of the mega-class. Sub-figure (a) shows the ground truth. Sub-figure (b) shows the decision boundary after 10 boosting rounds. Sub-figure (c) shows the decision boundary after 20 boosting rounds. The employed weak classifier is the *tree stumps*.

To visualize how the joint training and feature sharing framework exploits common features, we designed a toy example. In figure 2.4(a), we showed 2000 examples belonging to 5 classes: red (crosses), green (squares), blue (diamonds) and black (triangles) with yellow (dots) as the *background* class. The features are the horizontal and vertical coordinates of the examples. Each pair of neighboring classes shares a common feature: for example, the cross and triangle classes share the horizontal feature; such feature needs to be within a positive range for an example to be considered a member of the two classes. Using the *tree stump* in equation 2.38 as the weak classifier, we performed 10 and 20 boosting rounds and show the resulting decision boundaries in figures 2.4(b) and 2.4(c), respectively. The decision boundaries made by 20 boosting rounds are already very close to the ground truth. The ROC score for all (four) classes are greater than 0.99 after 20 rounds of boosting.

2.7 Semi-supervised Learning Paradigm

The performance of the supervised methods depends greatly on the availability and quality of the labeled data. However, in some applications, labeled data are costly and laborious to obtain. For example, in an image retrieval problem, obtaining labeled data requires human annotation and in protein sequence analysis, annotating the function and structure of a protein sequence requires time-consuming *in vitro* experiments. In the presence of limited number of labeled training sequences, the performance of the classifiers estimated under the supervised setting might be sub-optimal. Enlarging the size of the training set by leveraging the unlabeled data under the semi-supervised learning paradigm will very likely improve the accuracy of the classifiers under correct settings.

Recent advances in computational methods have relied heavily on the use of unlabeled data. For example, in [14], Cohen *et al.* leveraged unlabeled data to perform Bayesian network structure learning for the task of facial expression recognition; Nigam *et al.* leveraged unlabeled data in [51], combined with a Naive Bayes classifier, to estimate the parameters of the classifier for text classification tasks; finally, in the context of protein homology detection, Gough *et al.* in [23], Weston *et al.* in [69], and Kuang *et al.* in [35] leveraged unlabeled protein sequences to enhance the accuracy of the classifiers for superfamily detection; all studies have reported significant improvement of performance once semi-supervised learning setting is adopted.

Traditional semi-supervised learning setting involves use of probabilistic classifiers under the *generative* setting. The common goal of such studies is to build more accurate generative classifiers that summarize the data. In [1], Altschul *et al.* take advantage of unlabeled data by *iteratively* recruit sequences in the unlabeled database to refine the parameters in a profile [24] for protein sequence analysis. Building a profile for a set of sequences, known to exhibit certain qualities, for example, belonging in the same functional group, can be formulated as a *generative* learning problem. Initially, the authors build a generative model using the labeled sequences. Next, they use the estimated models to scan through the unlabeled data set to identify a set of candidates

which, in the next iteration, are used as *labeled* examples to refine the models. We can imagine such practice as re-estimating the parameters of a probability distribution, with possibly larger coverage in the feature space after each round of recruiting process. Under such setting, the candidates absorbed in each iteration by the generative methods are used in the next iteration to improve the generative classifiers. Gough *et al.* in [23] adopted the same framework in subsequent studies for similar tasks.

Alternatively, in other studies, the candidates recruited by the generative classifiers are used to refine the decision boundaries of the *discriminative* models. In [69], Weston *et al.* used *generative* models, such as BLAST [62] and PSI-BLAST, to identify candidates and used them to form a neighborhood kernel. The authors also report significant improvements in performance under such setting. Other alternative approaches such as *transductive learning* [61] also exist. However, we do not study the effects of these approaches in our study.

Finally, the authors in [71, 14] showed that if the models used to identify neighbors deviate from the true underlying data generating process, use of unlabeled data has detrimental effect on the quality of the classifiers. Also, we will show in Chapter 6 that in some situations, pre-processing the recruited candidates is necessary and if such practice is not carried out, the performance of the classifiers will be compromised. We should always use an recruit unlabeled examples cautiously.

Chapter 3

Sparse Generative Models for Protein Homology Detection

Some of the most powerful early approaches for protein sequence classification are based on statistical models known as the hidden Markov models [23, 18, 4, 28]. However various studies show that as the percentage of the sequence identities of related proteins goes below 30%, the chance of their relationship being detected by these methods becomes increasingly small. On the one hand, there is no doubt that the hidden Markov models have been very successful for protein classification, but on the other hand, they all become less reliable when more distant, less homologous proteins are considered.

To considerably improve reliability of protein sequence classification, there are two possible directions. First, in [24] Gribskov *et al.* construct a *position-specific profile* from a group of sequences. The profiles exhibit linear structure, which guarantees linear inference time. Furthermore, each *local* position posses different degree of preference for mutation, insertion and deletion, in contrast to traditional sequence-sequence comparison techniques in which a *global* scoring matrix for substitution and a set of global parameters for opening, extending and closing a gap are utilized. The authors reported promising results for discriminating members and nonmembers of *Globin* and *Immunoglobulins* sequences. Second, in [13], Casbon *et al.* reported that matching profiles constructed using structure-based sequence alignments by incorporating structural information to the sequence alignment also show some promise for the remote homology detection task. The advantages of these two approaches are clear: first, sequence comparison based on structural information creates reliable foundation for the alignment, and second, discovering profiles in the group of proteins can reveal features that may allow alignment of distantly related proteins.

Based on structural alignments, Kister *et al.* in [34, 33] suggest that a *small* set of *key residues (positions)* residing on some secondary structure are sufficient to discriminate members and non-members of a superfamily. Further, for two neighboring key positions sitting on a common secondary structure, the number of interposing residues between them is highly conserved, whereas for two neighboring key positions sitting on neighboring secondary structures, the number of interposing residues between them may not be conserved. In the study, each secondary structure corresponds to a *word* described by a collection of *patterns* (regular expressions). The patterns within a secondary structure (word) consist of a few *critical positions* that are manually picked by domain experts. After all patterns in all words are determined, the authors suggest to perform detection using a *filtering* approach. To detect members in a database of sequences, first all sequences matching any patterns in the first word are extracted. The matching sequences form another (smaller) sequence database. In the next iteration, all sequences in the newly constructed database *matching* any pattern in the second word are extracted, resulting in another (possibly smaller) database. The procedure iterates and terminates when all words are exhausted. All sequences in the final sequence database posses all required patterns and the authors grant them membership to the group of interest. The authors reported that the proposed approach identified proteins from the *Cadherin* superfamily with low false positive rate.

Though the method proposed by Kister *et al.* successfully identified proteins from a group of sequences with low false positive rate, the procedure is not automated and requires knowledge of higher order structural information. The presence of structural information may improve the quality of the alignment; however, such information is costly and laborious to obtain. The goal of the study presented in this chapter is to develop an automated procedure that identifies these *key positions* without knowledge of secondary or higher order structure from the protein sequences.

Based on the studies performed by Kister *et al.* , we further hypothesize that besides the key positions, the additional knowledge of *the distances between each neighboring pair of key positions* allows one to classify an unknown protein into an appropriate group of sequences and *no structural information is required*. In this work, the key

positions serve as a basis of development of computer algorithms for protein sequence classification based on *sparse profile hidden Markov models*. A direct result is a class of models with lower complexity and therefore fewer parameters to estimate compared with its dense counter part: the traditional profile hidden Markov models.

3.1 Background

We first introduce the profile hidden Markov models and their biological interpretations. We then propose a new class of models, by *generalizing* the profile HMMs, to test our hypothesis.

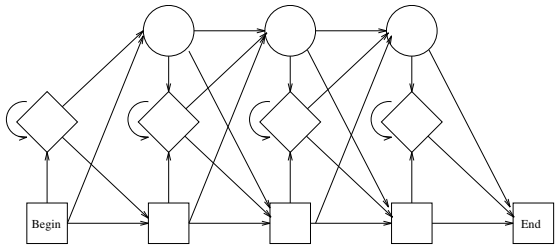
3.1.1 Profile hidden Markov models

Profile hidden Markov models (PHMMs) are one of the most commonly used statistical models for protein sequence analysis [19]. A profile HMM that describes a group of functionally and/or structurally related proteins is *position specific* and consists of three types of states: *match*, *insertion*, and *deletion*. The match states model conserved positions within the group and are parametrized by a probability distribution. Such distribution specifies the chance of observing an amino acid residue at each corresponding position. Each match state also accompanies an insertion and a deletion state; insertion states model the random insertion process in evolution and are parametrized by emission distributions that are very close to the *background distribution* of amino acids; deletion states model the random deletion process in evolution and are *mute*; in other words, visitation of such states does not induce observation of a symbol. We show a profile HMM in Figure 3.1(b). In the figure, the match, insertion and deletion states are represented using squares, diamonds and circles, respectively.

To estimate a profile HMM that captures the commonly shared characteristics within a group of sequences, we need a high-quality *multiple alignment* of the sequences. We show an example of multiple alignment in Figure 3.1(a). In the figure, each row corresponds to a sequence and each column marked by an asterisk induces a *match state* in the estimated profile HMM. A high-quality multiple alignment from a group of

```
HBA_HUMAN    ...VGA--HAGEY...
HBB_HUMAN    ...V----NVDEV...
MYG_PHYCA    ...VEA--DVAGH...
GLB3_CHITP   ...VKG------D...
GLB5_PETMA   ...VYS--TYETS...
LGB2_LUPLU   ...FNA--NIPKH...
GLB1_GLYDI   ...IAGADNGAGV...
                  ***  *****
```

(a) multiple alignment



(b) PHMM

Figure 3.1: Left panel: a multiple alignment; the positions marked by an asterisk correspond to a match state in the estimated PHMM. Right panel: a profile hidden Markov model

related biosequences reveals important evolutionary relationship among the sequences: for example, in Figure 3.1(a) most of the sequences in the first column emit a symbol 'V' while the last two sequences have symbols 'F' and 'I', suggesting a *mutation* or a *substitution* event at this position; next, a deletion event occurs in the second sequence in the second column (indicated by '-'); finally, the last sequence emits two extra symbols in the columns not corresponding to a match state, suggesting an *insertion* event. In addition to the evolutionary relationship among protein sequences, the multiple alignment also provides a general guideline to the structure for the estimated profile HMM. For example, the multiple alignment in Figure 3.1(a) suggests that the estimated profile HMM will have 8 columns (8 match states). Moreover, based on the multiple alignment, we can also obtain an initial estimates for all the parameters from the alignment under the *maximum likelihood* framework. The initial estimates sometimes play a crucial role for parameter searching, since we cannot guarantee that only one global optimum exists.

The estimated profile HMM collects a rich set of statistics from the input multiple alignment. As a result, we can infer the degree of sequence conservation at each position. Its characteristic linear structure enables making inference in *linear* time. For a profile HMM with m match states, making inference on a sequence X with length T_X takes $O(mT_X)$ time, compared to $O(n^2T_X)$ time for a general-structure HMM with n states.

Finally, the position-specific modeling technique, although effective, often results in

models that are too specific, a problem known as *overfitting*. To avoid such problem, we usually prefer estimation methods based on the *maximum a posteriori* principle over those based on the *maximum likelihood* principle.

3.1.2 Sparse Profile Hidden Markov Models

To reflect our hypothesis that only a small set of key residues is required to construct a descriptive model for a protein superfamily, we propose a class of *sparse profile hidden Markov models*, based on generalizing the emission duration of a traditional profile HMM: in addition to the emission and transition probabilities, each *emitting* state in a sparse profile HMM carries information about the state's duration distribution, the number of symbols we expect to observe upon visitation of such state. In a traditional *dense* profile HMM, each state emits only one symbol and multiple emissions are modeled by *self-looping* transitions as shown by the insertion (diamond) states in Figure 3.1(b). However, use of self-looping transitions implicitly assumes that the number of emitted symbols follows a *geometric* distribution: denote D as a random variable for the number of residues emitted by an insertion state where the self-looping transition probability is p, then $P(D = d|p) = p^{d-1}(1-p)$. In a duration-explicit HMM [1], we remove such assumption by explicitly specifying a probability distribution for the duration. A direct consequence is that the model obtains more descriptive power, with the expense of needing to estimate the corresponding parameters. The class of duration-explicit HMMs have shown great success as a tool for ab-initio identification of genes in DNA sequences [11].

We define a sparse profile HMM as $SPHMM = \{S, A, B, D, \Sigma, \Pi\}$, where S, A, B, Σ, Π are defined in Section 2.2 and $D = \{d_{s_i}\}_{i=1}^{n}$ denotes the duration distribution over all states and n the total number of states in the sparse profile HMM. We partition the states in a sparse profile HMM into two subsets: *critical* and *non-critical*. The critical states (analogous to the *match* states in a profile HMM) emit patterns of residues that are characteristic to a superfamily, which we also refer to as the *key residues*. The

[1] Also known as *segmental* HMMS.

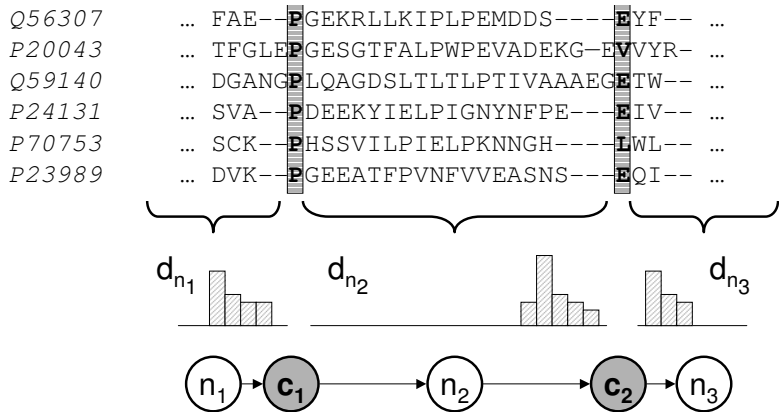


Figure 3.2: Sparse profile HMM. Shown is a multiple alignment of a set of protein sequences with a corresponding sparse profile states n_1, c_1, n_2, c_2. Residues corresponding to critical states c_i are shaded. Also depicted are examples of duration distributions d_{n_i}. Note the small number of critical positions as well as the sparseness of the distance distribution.

non-critical states (analogous to the insertion states in a profile HMM) emit multiple symbols and the number of emitted symbols follows the specified probability distributions for duration. Note that in a traditional profile HMM, *every* match state (position) is considered as a critical position using our terminology. We assign one non-critical state between each pair of neighboring critical states. This sparse model is illustrated in 3.2 where the critical (shaded) states are denoted by the letter 'c' and the non-critical (non-shaded) states are denoted by the letter 'n'. While it is not necessary in general, we assume that the critical states have a fixed duration of one (to relax such condition one may use a Markov chain to model multiple emissions). To express disinterest in the residue composition in the non-critical states, we assign background emission probabilities [2] to the non-critical states. The distribution over the duration assigned to one non-critical state specifies the degree of conservation in distance (number of residues) between the two neighboring critical positions. As a result, together with the residue composition specified in the critical states, the durations specified in the non-critical states form a set of *probabilistic patterns* that are characteristic to a superfamily. Finally, it is also possible for a sequence to *skip* a critical states (*i.e.* traversing between neighboring non-critical states $n_i \rightarrow n_{i+1}$ directly). We model such deletion event with

[2] Another possibility is the *uniform* distribution.

deletion states in the sparse profile HMMs (not shown in Figure 3.2).

The number of critical states is usually small and we only need on the order of 25% critical states of what a traditional, dense profile needs, as we will show in later section. A direct consequence results in lower model complexity and therefore fewer parameters to estimate for *sparse* profile HMMs.

3.2 Methodologies

In this section, we first describe the functional groups for the homology detection task. Next, we discuss the complexity of making inference in a sparse profile HMM. Finally, we outline our automated learning procedure, inspired by the ones employed by PSI-BLAST [1] and *Superfamily* [23], to iteratively recruit unlabeled sequences for improved model sensitivity.

3.2.1 Objects of the investigation

In this work we analyze three protein superfamilies, *Beta-Galactosidase*, *Cadherin*, and *Fibronectin*, all under the *Immunoglobulin-like beta-sandwich* fold. Spatial structures of sandwich proteins are composed of a class of secondary structure called β-strands, which form two main β-sheets packing face to face. The determination of H-bonds between the main chain atoms allows us to determine the arrangements of the strands and identify those strands that make up the two main sandwich sheets. Analysis of the arrangements of strands in all known sandwich-like protein with known structures revealed the definite rule that is valid for almost all sandwich proteins [33]. This rule describes the formation of a certain fundamental supersecondary sandwich substructure called *interlock*, consisting of two pairs of strands. Our goal is to use an automated procedure to identify the critical positions on these β-strands and discriminate between members and non-members of these superfamilies.

3.2.2 Making inference in a sparse profile HMM

Given a protein sequence $X = x_1 x_2 \cdots x_{T_X}$ and a sparse profile HMM we are concerned about two tasks: (1) computing the probability that sparse profile HMM generates X and (2) obtaining the optimal alignment of X with respect to the model. The two closely related tasks are critical for computational classification of protein sequences and learning of protein family/superfamily models.

We employ the generalized forward message passing for duration-explicit HMMs [52] for computing the probability that the sparse profile HMM generates X; on the other hand, obtaining the alignment of X to the sparse model means computing the most likely (Viterbi) path that generates the sequence. Let $Y = y_1 y_2 \cdots y_{T_X}$ denote such path, we are interested in finding $(Y^*, D^*) = \text{argmax}_{Y,D}\, P(Y, D|X, SPHMM)$, where D^* denotes the duration for each state induced by the optimal path. We perform the search with the generalized Viterbi algorithm for duration-explicit HMMs.

In a general-structure duration-explicit HMM, computational complexity for both algorithms is $O(m^2 D_{max} T_X)$, where m denotes the number of states in the model and D_{max} the maximal distance allowed. For sparse profile HMMs, with the *linear* model structure accompanying m_c *critical* states, we can further reduce the computational complexity to $O(m_c D_{max} T_X)$. In contrast, making inference with a traditional profile with m_p states incurs $O(m_p T_X)$ complexity and this may raise a concern about extra computational burden of making inference using our models. However, the computational time of a sparse profile HMM and a traditional profile HMM is similar since $m_c D_{max} \approx m_p$.

3.2.3 The automated learning procedure

Our central hypothesis states that a small number of key residues with the distance between each neighboring pair allows for reliable classification of protein sequences. To construct a set of sparse profile HMM that represent each superfamily, we implement an automated learning procedure. The procedure consists of two main steps: (1) construction of traditional profile HMMs for each superfamily of interest and (2) construction

of sparse profile HMMs based on traditional profiles.

We represent each superfamily under investigation with a set of *seeds* in the ASTRAL compendium [9] (in other words, a superfamily might be represented by more than one model under this setting) and employ an automated procedure inspired by the ones employed in *Superfamily* [23] and PSI-BLAST [1] for construction of sparse profile HMMs. In our experiments, we perform *semi-supervised learning* by leveraging unannotated sequences in the *non-redundant* (NR) sequence database, containing nearly one million sequences. We depict the procedure in Figure 3.3. For each *seed*, we first use BLAST [62] to define two data sets: the set of close homologues and the set of candidate remote homologues of the seed, both extracted from the NR database. The set of close homologues consists of protein sequences that are very similar, on the sequence level, to the query sequence (seed) and the probability of having such similarity by chance (the p-value) is extremely low. The set of candidate remote homologues, on the other hand, consists of (1) diverged sequences that may not be so similar to the query sequence but are evolutionarily related and (2) unrelated sequences (the negative sequences). During the iterative recruiting process, we hope to filter out the evolutionarily related sequences such that incorporation of these sequences into the training set might enlarge the coverage of the generative model and hence, enable us to detect more remote homologues. Once the iterative procedure terminates with a set of homologous sequences, we use a commonly employed program, CLUSTALW [32], to produce a multiple alignment, from which we estimate a profile HMM and use it to scan the candidate set. We remove the sequences with high scores (low e-values) from the candidate set and append it to the homologue set. In the next iteration, we use the newly constructed homologue set to refine the profile HMM and repeat this procedure four times and use the homologue set obtained in the last iteration to construct our final profile HMM.

Once the iterative procedure terminates, we estimate the conservation of each position in the constructed profile HMM and extract the highly conserved positions. We estimate the degree of conservation using the *entropy* of the emission probabilities. For each position, if the entropy of the distribution is lower than a pre-selected threshold δ, we consider it conserved (critical); otherwise, we consider it diverged (non-critical).

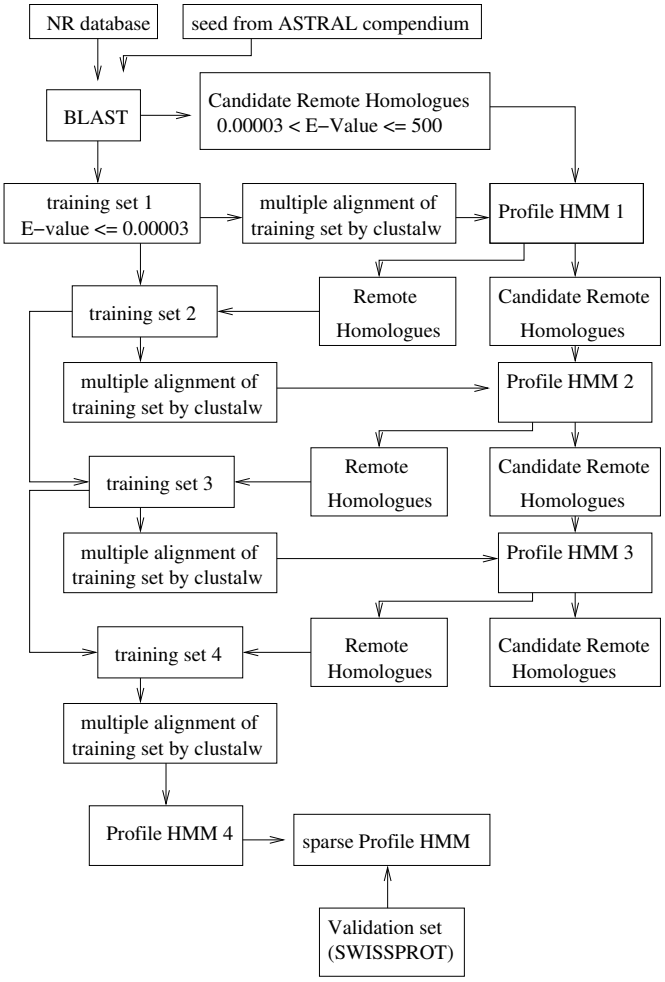


Figure 3.3: The automated learning procedure.

Next, we estimate the duration distribution of the non-critical states from the multiple alignment (illustrated in 3.2) by *smoothing* the empirical distribution.

Our ultimate goal is the selection of a *minimal* set of key positions (states) while maintaining a high level of protein classification performance using the sparse models. To understand how many of these highly conserved positions are necessary to obtain reasonable performance, we perform a model validation step. We construct several models with different numbers of critical positions and evaluate the classification performance of the models with Swiss-Prot [8]. We note that not every sequence in Swiss-Prot has superfamily annotation; therefore we use the sequences detected by *Superfamily* [23] as a benchmark. Our empirical studies, outlined in the next section, show that the performance curves exhibit a plateau for a range of critical states. An optimal order of the sparse profile HMM is selected as the minimal one after which the classification performance begins to exhibit serious degradation.

For each sequence X, we first assign it a score based on the *log likelihood ratio*:

$$score(X, H^{SPHMM}) \quad = \quad \log \frac{P(X|H^{SPHMM})P(H^{SPHMM})}{P(X|H^{NULL})P(H^{NULL})}, \tag{3.1}$$

where H^{NULL} denotes the *null* model and we set the prior probabilities of the models to the *uniform distribution*. The *null* hypothesis is simply a model with a single state, with self-looping transition, emitting observations based on background frequencies. The normalized score of each sequence reflects how well the sequence aligns to the sparse profile HMM. Altschul *et al.* showed in [2] that the scores obtained by performing pairwise alignment follow an extreme value distribution when the alignments are ungapped and when the alignments are gapped the scores only *roughly* follows an extreme value distribution. We adopt such reasoning and fit an *extreme value distribution* to the scores of all sequences in the database. We estimate the parameters of the distribution using the procedure for *type-I censored data* [40] and select an appropriate e-value threshold for the discrimination task. The e-value of the score s of a sequence is database dependent and denotes the expected number of sequences with scores higher than or equal to s by chance: the smaller the e-value, the more significant the *hit*. The final decision rule is to assign superfamily membership to a sequence if its e-value is

below the threshold.

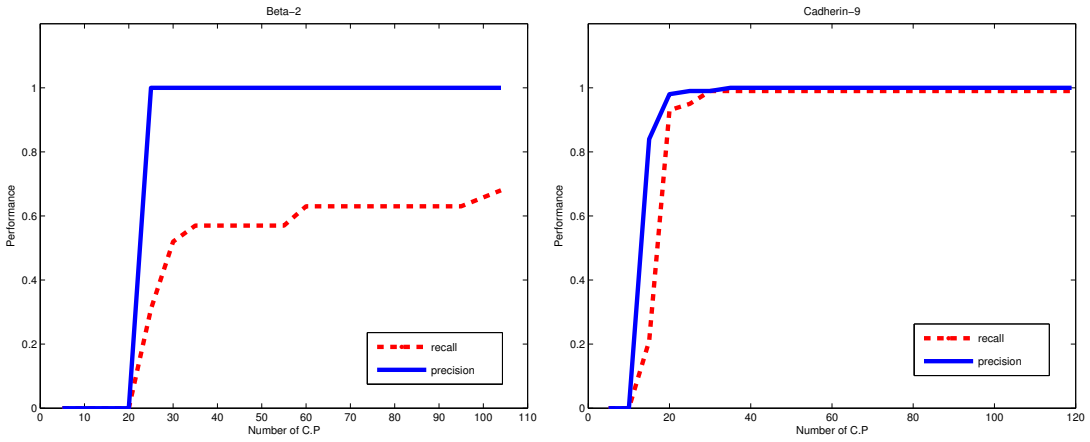
3.3 Results and Discussion

To test our hypothesis about the sufficiency of sparse models for protein sequence classification, we perform two sets of experiments. In the first set of experiments, we compare the performance of sparse profile HMMs as a function of the complexity of the model (number of key residues). In the second set of experiments, we show the results of protein classification using the estimated sparse profile models on a large set of proteins in UNIPROT [3].

3.3.1 Selection of optimal order for sparse profile hidden Markov models

We validate sparse profile HMMs and select the optimal number of key positions by evaluating performance of the models on the Swiss-Prot protein database, using the automated procedure outlined in Section 3.2.3. The performance is summarized using *recall* and *precision* scores. Our preliminary results show that only about 25% of the sequence positions and distance between each neighboring positions are sufficient to reliably describe protein superfamilies, as we will show in the following.

To illustrate the study results, we consider models built on two seeds from two superfamilies, *Beta-Galactosidase* and *Cadherin*. The results of the experiments are summarized in Table 3.1 and Figure 3.4. We define *recall* as the ratio of the number of true positives detected by the models to the total number of true positives in the sequence database and *precision* as the ratio of number of true positives detected by the models to the total hits reported. A high recall rate suggests high sensitivity of the model to the positive sequences while a high precision rate suggests high accuracy of the model among the selected positive sequences. Using the same seeds, *Superfamily* detects 19 and 153 hits in Swiss-Prot while our models detected 13 and 152 hits. We observe that both experiments show serious degradation in performance when the number of critical positions fall below 25% of the length of the original profile HMM (around 25-30



(a) (b)

Figure 3.4: Performance (recall and precision) as a function of the number of key positions for (a) beta Galactosidase and (b) Cadherin models constructed from two ASTRAL seeds.

positions). For Cadherin, the model demonstrates high precision and recall rate with as few as 25 key positions; for Beta-Galactosidase, the performance degradation occurs at around 30 key positions. We note that the dense profile HMM for Beta-Galactosidase also has a 68% recall rate (the full model in the last row of Table 3.1), benchmarked using the hits reported by *Superfamily*. We note that in this study, we estimate the profile HMMs using very simple and primitive techniques whereas the authors of *Superfamily* estimate their profile HMMs using many sophisticated optimization techniques, such as *position-based sequence weighting* [27] and *estimating sequence weights to achieve target saving* [3], and mixture of Dirichlet priors [10, 60]. As a result, our profile HMMs focus on sequences that exhibit higher similarity to the training sequences, whereas the ones in *Superfamily* exhibit higher generalization power with various optimization techniques. The reasonable but not optimal recall rate of the sparse profile HMMs is directly carried over from the dense model and therefore is not a direct consequence of collapsing states from a traditional profile HMM, as we will show in later sections. Our experimental results suggest that we are able to maintain the performance of the models after significantly reducing the complexity of the model.

In Figure 3.5 we show the distribution of the log likelihood ratio scores of Swiss-Prot

[3]ISMB99 tutorial on using HMMs, available at http://www.cse.ucsc.edu/research/compbio/sam.html

Table 3.1: Number of critical positions vs performance for Beta-Galactosidase (left) and Cadherin (right)

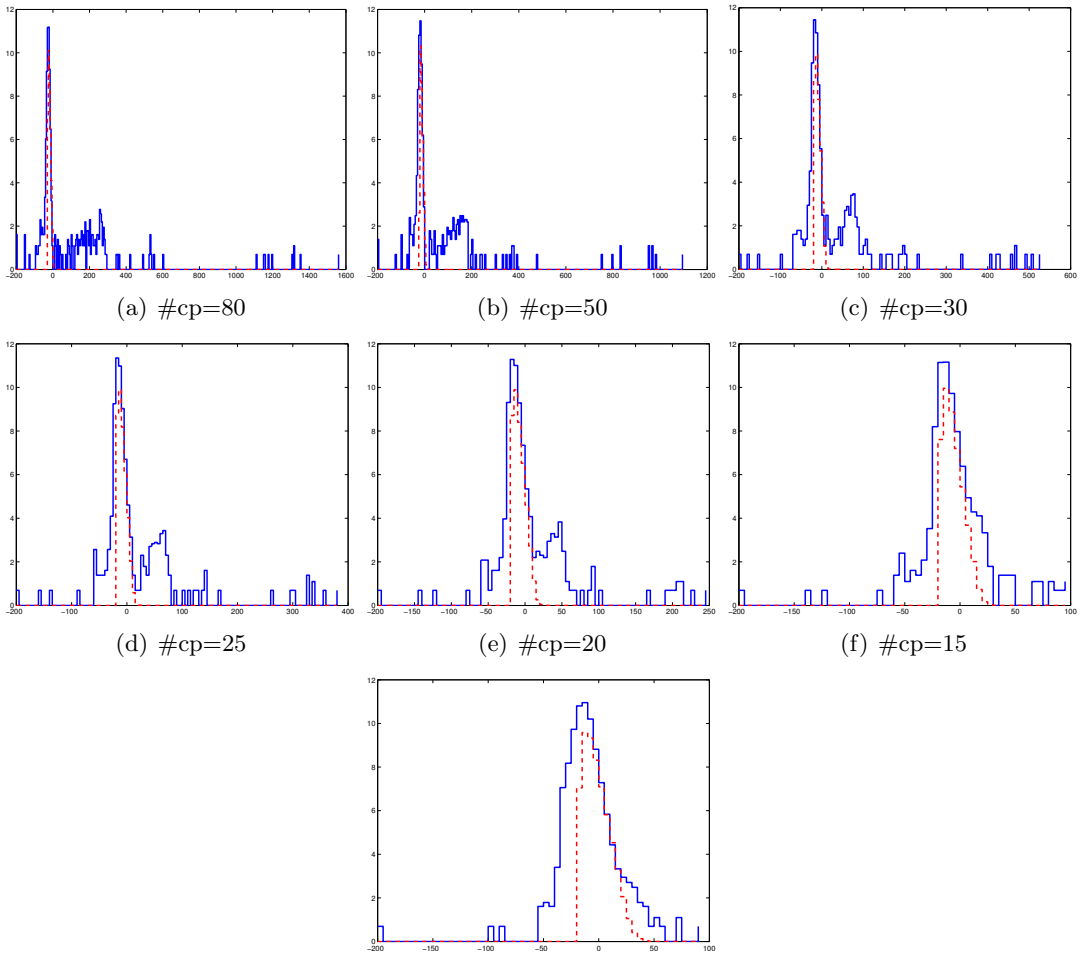
number of critical positions	number of hits reported	recall	precision
5	12	0	0
10	0	0	0
15	0	0	0
20	0	0	0
25	6	0.32	1
30	10	0.53	1
35	11	0.58	1
40	11	0.58	1
45	11	0.58	1
50	11	0.58	1
55	11	0.58	1
60	12	0.63	1
65	12	0.63	1
70	12	0.63	1
75	12	0.63	1
80	12	0.63	1
85	12	0.63	1
90	12	0.63	1
95	12	0.63	1
104 (full)	13	0.68	1

Beta-Galactosidase

number of critical positions	number of hits reported	recall	precision
5	35	0.00	0.00
10	16	0.21	0.00
15	38	0.93	0.84
20	146	0.95	0.98
25	146	0.99	0.99
30	152	0.99	0.99
35	152	0.99	1.00
40	152	0.99	1.00
45	152	0.99	1.00
50	152	0.99	1
55	152	0.99	1
60	152	0.99	1.00
65	152	0.99	1.00
70	152	0.99	1.00
75	152	0.99	1.00
80	152	0.99	1.00
85	152	0.99	1.00
90	152	0.99	1.00
95	152	0.99	1.00
119 (full)	152	0.99	1.00

Cadherin

sequences as a function of key positions of the Cadherin models (horizontal axis). We observe that clear separation of members (ticks on the right) and non-members (ticks on the left) of the Cadherin superfamily becomes more pronounced as the number of critical positions increases. Hence, as the model becomes more sparse, the margins in scores between the members and non-members become smaller. An important question is how sensitive the models are to unseen positive sequences in larger databases such as UNIPROT [3].



(a) #cp=80 (b) #cp=50 (c) #cp=30

(d) #cp=25 (e) #cp=20 (f) #cp=15

(g) #cp=10

Figure 3.5: Empirical distribution of log likelihood ratio scores of sequences in Swiss-Prot as a function of number of key positions in the Cadherin model (solid line). We also show the fitted extreme value distribution in dashed lines. As the number of critical positions increases, we observe better separation (larger margin) between the members and non-members of the superfamily.

Table 3.2: Performance measure of the models against UNIPROT

superfamily	number of critical positions	number of hits reported	number of hits reported by *Superfamily*	number of critical positions superfamily has	recall rate
Beta-Galactosidase	25	217	321	107.33	67.60%
Fibronectin	25	858	1858	99.89	69.48%
Cadherin	25	757	876	107.29	86.47%

3.3.2 Classification performance on UNIPROT

We scan through the UNIPROT protein sequence database with the collection of estimated sparse profile HMMs, each corresponding to a seed sequence in the ASTRAL compendium. Again, we benchmark our results based on the reported hits from *Superfamily* since not every sequence in UNIPROT has superfamily annotation. We summarize the results in Table 3.2. For the models in *Superfamily* we report the average length of the profile HMMs. We achieve four-fold compression (from about 100 to 25 key residues) with slightly lower recall rates.

3.3.3 Discussion

To understand why the estimated models have low recall rate, we perform a more detailed investigation. The results of the investigation reveal that first, there are further diverged families that do not *fully* conform to the probabilistic patterns specified by the profile HMMs. Such divergence results in poor alignment of the *false negative* sequences with respect to the dense profile. From the alignment, we observe that the false negative sequences *skipped* some critical positions, suggesting that as the sequences in the superfamily diverge, conservation on some positions will be more relaxed. A subset of critical positions appears to be more important than the others. The second important observation is that some sequences that are non-members of the group partially conform to the probabilistic patterns imposed by the sparse profile HMMs (*false positives*). Though these sequences still have relatively low e-values and are not assigned membership to the superfamily, the inflated scores bias the estimation of parameters of the EVD and thus the e-values of each positive sequence. In fact, in the learning community, it is well-known that *generative* models focus on capturing the common

characteristics among the members within a class. However, there is no guarantee that no non-members of the class exhibit such characteristics. To make such a distinction we need to use *discriminative* models and we will discuss the effect of such models in the following chapters.

3.4 Conclusion and future work

In this study, we outlined and implemented an automated procedure that estimates a class of sparse profile HMMs and recovers the *critical positions* in a superfamily. We show that the sparse profile HMMs may be used to model several superfamilies of sandwich-like [33] proteins. The sparsity of the models is based on the hypothesis that a *small* subset of *key positions* in the sequences and the distance between each pair of neighboring positions are sufficient for discriminating members and non-members of a superfamily. In the first part of our experiment, we show that our models achieve a four-fold compression compared to the traditional dense profile HMMs and maintain reasonable performance. Our investigation reveals first, the models are still too specific and further diverged families within the superfamily are not captured, thus motivating the need for using more sophisticated optimization tools for generalizing the profile HMMs. In the second part of our experiments, we observe that as the sequences in a superfamily diverge, conservation at some positions become more relaxed and some undetected members do not fully conform to the probabilistic pattern imposed by the sparse profile HMMs. We also observe that some non-members of the superfamily partially conform to the probabilistic patterns imposed by the sparse profile HMMs. We conclude that use of *generative models* to capture the common characteristics of a group of related sequences does not guarantee that no non-members of the group posses such characteristics. Such conclusion motivates employing *discriminative models* to perform protein remote homology detection. We believe that further carefully crafted dense models and use of *negative sequences* during the estimation process we will improve the performance of the sparse models. In future studies, we propose to incorporate *Bayesian learning paradigm* into our estimation procedure and make use of the Dirichlet mixtures proposed by Brown *et al.* in [10, 60] to further generalize the model and improve the

sensitivity. We also propose to employ *discriminative* learning approaches to make distinction between the members and non-members of the superfamily.

3.5 Acknowledgments

This work was carried out by jointly working with Professor Alexander Kister.

Chapter 4

Sparse Discriminative Models for Protein Homology Detection

In the previous chapter, we focus our study on protein classification using *generative* models and perform experiments on three superfamilies under the *Immunoglobulin-like beta-sandwich fold*. In this chapter, we focus our study on protein classification using *discriminative* models under the *supervised* learning setting with a more diverse benchmark data set. We will also illustrate the differences between the generative and discriminative approaches in this study.

Early approaches to computationally-aided homology detection, such as BLAST [62] and FASTA [63] rely on aligning the query sequence to a database of known sequences (pairwise alignment). However, the weakness of the pairwise approach is its lack use of data: alignment is performed on the query sequence to each sequence in the database *one at a time*. Later methods such as profiles [24] and profile hidden Markov models (profile HMMs) [19] collect aggregate statistics from a set of sequences known to belong to the same functionally and/or structurally related group. Upon query time, we align an unknown sequence to all models to determine if there is a significant *hit*. Profile HMMs have demonstrated great success in protein homology detection. The linear structure of the profile HMM offers direct interpretation to the underlying process that generates the sequences: the *match* states represents positions in the family that are *conserved* throughout the evolutionary process. However, such generative approaches only make use of positive training examples, resulting in the classifier focusing on capturing the commonly shared characteristics of the examples within the group. While such practices may be effective, there is no guarantee that no negative examples posses such characteristics, which renders the generative models susceptible to false

positives. The discriminative approaches attempt to capture the distinction between different classes by considering both positive and negative examples simultaneously to estimate the decision boundary.

In [30] Jaakkola *et al.* proposed *SVMFisher*, a class of *discriminative* classifiers, for protein remote homology detection. The idea is to combine a generative model (profile HMM) and a discriminative model (SVM) and perform homology detection in two stages. In the first stage, the generative model, *trained using positive examples only*, extracts features from all sequences (*positive and negative*). In the second stage, with the fixed-length features, the discriminative model constructs the decision boundary between the two classes.

The class of *string kernels*, on the other hand, bypasses the first stage and directly model the decision boundary using SVMs. The *spectrum kernel* [42], and the *mismatch kernel* [43] define different notions of *neighborhood* for observed contiguous substrings within the sequences and determine the similarity between the two sequences as a function of the size of the intersection of such neighborhood.

Previous studies show that both SVMFisher and the class of string kernels are more effective than the generative models. Despite their great success these two approaches are not readily interpretable, or when an interpretation of the models is available, it may not be biologically intuitive. For example, the model should be able to explain how sequences in the same superfamily evolve over time. Are there certain positions that are *critical* to a superfamily? If so, what kind of physical/chemical properties should such position possess? Although the profile HMMs attempt to offer such explanations but as generative models they do not make use of negative examples and lack the discriminative interpretability.

The central idea of this work is to develop an interpretable method for protein remote homology detection. Our approach is motivated by the results presented in Kister *et al.* in [33, 55, 34] that postulate the existence of a *small* subset of positions and residues in protein sequences may be sufficient to discriminate among different protein classes. We aim to recover these *critical positions* and the corresponding preferred residues with a new set of features embedded in a class of discriminative models. The combination

of the features and the classifier may offer a *simple* and *intuitive* interpretation to the underlying biological mechanism that generates the biosequences.

4.1 Related works

Denote X as a protein sequence. Jaakkola *et al.* proposed to use the gradient of the log-likelihood of the sequence X with respect to the model parameters as features in [30, 31]:

$$\begin{aligned} f_{\tilde{x},\tilde{s}} &= \frac{\partial}{\partial \theta_{\tilde{x},\tilde{s}}} \log P(X|\Theta) \\ &= \frac{\xi(\tilde{x},\tilde{s})}{\theta_{\tilde{x}|\tilde{s}}} - \xi(\tilde{s}), \end{aligned} \tag{4.1}$$

where $\tilde{\text{x}} \in \Sigma$, the alphabet set, $\tilde{\text{s}} \in S$, the emitting states in the model, Θ represents the set of parameters of the model, $\theta_{\tilde{x}|\tilde{s}}$ represents the emission probability of symbol $\tilde{\text{x}}$ at state $\tilde{\text{s}}$, and $\xi(\tilde{x},\tilde{s})$ as well as $\xi(\tilde{s})$ are the sufficient statistics, obtained using the forward-backward algorithm in [52]:

$$\begin{aligned} \xi(\tilde{x},\tilde{s}) &= \sum_{t=1}^{T_X} P(S_t = \tilde{s}, X_t = \tilde{x}|X,\Theta) \\ &= \sum_{t=1}^{T_X} P(S_t = \tilde{s}|X,\Theta) I(X_t = \tilde{x}), \end{aligned} \tag{4.2}$$

where T_X is the length of X, S_t is the state that is traversed at time t, X_t is the t^{th} symbol of X, $1 \leq t \leq T_X$, and $I(.)$ denotes the indicator function. The authors referred to the extracted *fixed-length* features as the *Fisher scores* and use the features to estimate an SVM for superfamily classification. Feature dimensionality can be further reduced with use of *9-component Dirichlet mixture prior* proposed by Brown *et al.* in [10, 60]. The SVMFisher approach has received some criticism due to the need to perform an inference procedure with quadratic complexity. Although the criticism does address a valid concern for a general HMM, in the case of a profile HMM, such issue does not exist: the linear structure enables one to make inference in linear time, as suggested in the previous chapters.

The methods based on *string kernels*, on the other hand, bypass the need of a generative model as a feature extractor. Given a sequence X, the *spectrum-k* kernel [42]

first *implicitly* maps it to a d-dimensional vector, with $d = |\Sigma|^k$:

$$\Phi_k(X) \quad = \quad \left(\sum_{\alpha \in X} I(\alpha = \gamma) \right)_{\gamma \in \Sigma^k} . \tag{4.3}$$

Next, the similarity between X and Y is defined as:

$$K(X, Y) \quad = \quad \Phi_k(X)^T \Phi_k(Y), \tag{4.4}$$

where in Equation 4.3 α denotes all *k-mers* in X and γ denotes a member in the set of all *k-mers* induced by Σ, the alphabet set. A *k-mer* is a contiguous substring of length k. The $mismatch(k, m)$ kernel ([43]) relaxes exact string matching by allowing up to m mismatches between α and γ:

$$\Phi^{mismatch(k,m)}(X) \quad = \quad \left(\sum_{\alpha \in X} I(\alpha \in N(\gamma, m)) \right)_{\gamma \in \Sigma^k} , \tag{4.5}$$

where $N(\alpha, m)$ denotes the set *k-mer neighborhood* induced by α up to m mismatches. The complexity for evaluating the N-by-N spectrum-k kernel is $O(N^2 max(u, T))$ where T denotes length of the longest sequence and u the number of *unique* k-mer extracted from all sequences; for the mismatch(k,m) kernel, evaluating the kernel value for sequences X and Y incurs $O(k^{m+1}|\Sigma|^m(T_X + T_Y))$ computational time. Explicit inclusion of the amino acid substitution process allows the mismatch kernel to outperform the spectrum kernel [43].

4.2 Proposed features and methods

Our computational approach to remote homology detection involves two steps: feature extraction with dimensionality reduction followed by joint classification and feature selection in the constructed feature space. A crucial aspect of this approach lies in the ability to impose the sparsity constraint, leading to significant reduction in the number of utilized features and the interpretability of the final model. We show the proposed *hybrid* procedure in Figure 4.1.

4.2.1 Feature extraction and dimensionality reduction

We use the sufficient statistics of the sequences with respect to the profile HMM as features. This choice of features may allow immediate biological interpretation of the

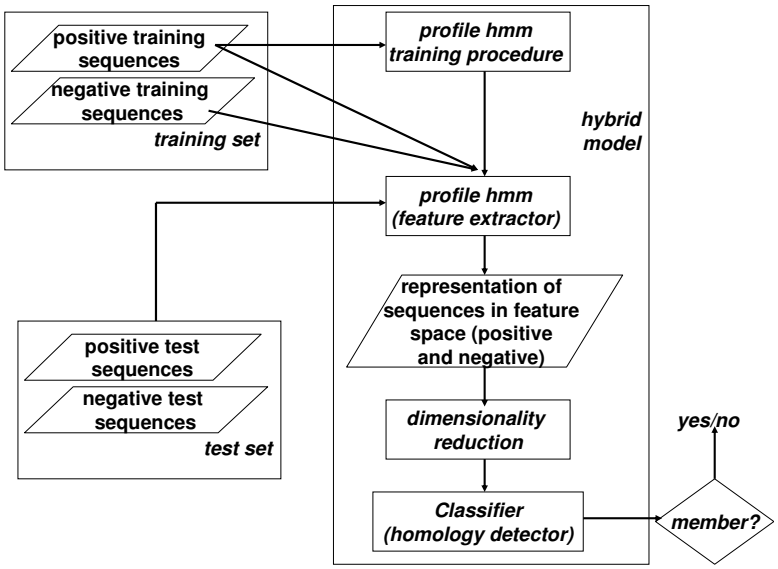


Figure 4.1: A schematic depiction of our hybrid model.

constructed model. In particular, we use the sufficient statistics that are associated with the symbols of the *match* states. We focus only on the match states because the structure of a profile HMM indicates that these states represent the positions that are conserved throughout evolution. We obtain these features using Equation 4.2 with

$$P(S_t = \tilde{s}|X, \Theta) \quad = \quad \frac{\alpha_{\tilde{s}}(t)\beta_{\tilde{s}}(t)}{P(X|\Theta)} \tag{4.6}$$

where $\alpha_{\tilde{s}}(t)$ and $\beta_{\tilde{s}}(t)$ are the forward and backward probabilities defined in [52]. In this setting, each example is represented by a vector of length $d = m|\Sigma|$ where m is the number of match states in the profile HMM and $|\Sigma| = 20$. To reduce dimensionality we partition all 20 amino acids into the following four groups, according to their chemical and physical properties:

- Group 1 – Non polar, hydrophobic: {F, M, W, I, V, L, A, P}.
- Group 2 – Negatively charged, polar, hydrophilic: {D, E}.
- Group 3 – No charge, polar, hydrophilic: {C, N, Q, T, Y, S, G}.
- Group 4 – Positively charged, polar, hydrophilic: {H, K, R}.

As a result, we represent each example by the following:

$$f_{g,\tilde{s}} = \sum_{\tilde{x} \in \Sigma} \xi(\tilde{x}, \tilde{s}) I(\tilde{x} \in \text{Group } g), \tag{4.7}$$

where $g \in \{1, 2, 3, 4\}$ represents each group of amino acid. The partition reduces the dimensionality, d, from $20m$ to $4m$, compared to $9m$ in [31, 30]. Our experiments in Section 4.3 confirm the effectiveness of this representation [1].

4.2.2 Classification and feature selection via logistic regression

Let f_i be the features extracted from the i^{th} sequence, X_i and $y_i \in \{1, -1\}$ be the response variable, where $y_i = 1$ denotes membership of the superfamily. The logistic regression model defines the probability of sequence X_i belonging to the superfamily of interest as in Equation 2.9:

$$P(y_i = +1|f_i, \theta) = \pi_i \quad = \quad \psi(\theta^T f_i) \tag{4.8}$$

where θ is the parameter of the model and the *link function* $\psi(\cdot)$ is the cumulative distribution function (CDF) of a logistic distribution. To estimate the parameters, we use the *maximum likelihood* principle by maximizing the *joint likelihood* of the observed data D:

$$P(D|\theta) \quad = \quad \prod_{i=1}^{n} \frac{1}{1 + \exp(-y_i f_i^T \theta)}. \tag{4.9}$$

As suggested in Chapter 2 logistic regression model and SVM are both *discriminative* classifiers and often yield similar results.

4.2.3 Interpretation of the logistic model with the proposed features

Use of the logistic model provides a simple and intuitive description of data. If the assumption $P(y = 1|f, \theta) = \psi(\theta^T f)$ holds, then the contribution of each predictor variable $f^{(j)}$, $1 \leq j \leq d$, is reflected in the corresponding model parameter $\theta^{(j)}$. A coefficient with large absolute value implies that the corresponding position has a strong preference for a type of amino acids: the position prefers a specific group of amino acids to be present when the coefficient is large and positive and prefers a specific group of amino acids to be absent when the coefficient is large and negative.

---

[1] We have also performed dimensionality reduction using the 9-component Dirichlet mixtures and do not notice any significant difference in performance

Moreover, θ also offers a probabilistic interpretation. Define the *odds* of an event with probability p of occurring as $\frac{p}{1-p}$; given the estimated parameter $\hat{\theta}$, and a feature vector f_i representing sequence X_i in the *feature space*, the estimated *odds* of sequence X_i belonging to the superfamily is:

$$odds(X_i \in \text{supFam}) = \frac{\hat{P}(X_i \in \text{supFam})}{1 - \hat{P}(X_i \in \text{supFam})} = \exp(\hat{\theta}^T f_i) \tag{4.10}$$

Define a new sequence $X_{i'}$ such that $f_{i'}^{(k)} = f_i^{(k)}, \forall 1 \leq k \leq d$ except $f_{i'}^{(j)} = f_i^{(j)} + 1$ for one specific $1 \leq j \leq d$, meaning we increase the j^{th} covariate of example i by one unit. In this case, the estimated odds of the new sequence $X_{i'}$ is:

$$\begin{aligned} odds(X_{i'} \in \text{supFam}) &= \exp(\hat{\theta}^T f_i + \hat{\theta}_j) \\ &= odds(X_i \in \text{supFam})\exp(\hat{\theta}_j) \end{aligned} \tag{4.11}$$

Equation 4.11 indicates that the odds are multiplied by $\exp(\hat{\theta}_j)$ when we increase the j^{th} covariate of example i by one unit. For example, suppose that at position $\tilde{s}$, the corresponding parameter $\hat{\theta}^{\tilde{x}|\tilde{s}}$ for symbol $\tilde{x}$ is $0.1615 = \log(1.175)$. Then the odds of a sequence X belonging to the superfamily increases by 17.5 percent if in X the symbol $\tilde{x}$ aligns to the model at position $\tilde{s}$.

One may argue that the preference or absence of a specific type of amino acids at a position in a group of sequences is already reflected in the profile HMM and using a logistic model to recover the desired information is redundant. However, this need not be the case: one position in a specific superfamily may prefer a certain type of amino acids which is also preferred by another group of sequences. In this case the corresponding coefficient in the logistic model we proposed will be *insignificant* (close to 0) and this is one of the crucial differences between a generative model (profile HMMs) and a discriminative model (logistic regression model) as we mentioned in earlier chapters. A coefficient corresponding to a certain type of amino acids at one position will be considered significant if those amino acids are present in the superfamily of interest (the positive examples) and are absent in all other superfamilies (the negative examples) or vice versa.

4.2.4 Use of sparsity-enforcing Regularizers

When the provided positive examples and negative examples are *separable*, then the objective function in Equation 4.9 is *unbounded* and infinitely many solutions exist. As a result, performing regularization on the parameter θ is necessary. Imposing such regularization on θ can be interpreted as placing a *prior distribution* on θ under the *Bayesian learning paradigm*.

Our belief that the model may be *sparse* leads us to choose the prior distribution $\theta \sim N(0, A)$, with some covariance matrix A. In our study, we set A to be some *diagonal* matrix and the induced objective function becomes the *posterior distribution* of θ:

$$J(\theta)_{Gaussian} \quad \propto \quad e^{-\frac{1}{2}\|\theta\|_2^2} \prod_{i=1}^{n} \frac{1}{1+\exp(-y_i x_i^T \theta)}. \tag{4.12}$$

Such an assignment of covariance matrix states that all θ's are *mutually independent*. The independence assumption is clearly violated, since the features we use are sufficient statistics. However, it is impractical to assume a general covariance structure for θ, as we will need to either specify or estimate $\binom{d}{2}$ parameters in advance. On the other hand, Gaussian priors often do not set the coefficients corresponding to the irrelevant features to 0, because the shape of the distribution is too mild around the origin. Therefore, we also use Laplacian priors which *promote* and *enforce* sparsity. In such setting, we assume $\theta_i \sim N(0, \tau_i)$ for $1 \leq i \leq d$. Furthermore, we place a *hyper prior* γ on every τ_i:

$$p(\tau_i|\gamma) \quad = \quad \frac{\gamma}{2} e^{-\frac{\gamma \tau_i}{2}}$$

Integrating out every τ_i, we have

$$p(\theta_i|\gamma) \quad = \quad \frac{\sqrt{\gamma}}{2} e^{-\sqrt{\gamma}|\theta_i|},$$

resulting in the following objective function:

$$J(\theta)_{Laplacian} \quad \propto \quad e^{-\|\theta\|_1} \prod_{i=1}^{n} \frac{1}{1+\exp(-y_i x_i^T \theta)}. \tag{4.13}$$

The hierarchical model shows that each θ_i now follows a Laplace distribution. The Laplacian priors produce *sparser* models than Gaussian priors. We plot the density functions of standard Gaussian (solid line) and a standard Laplacian (dashed line) distributions in Figure 4.2.

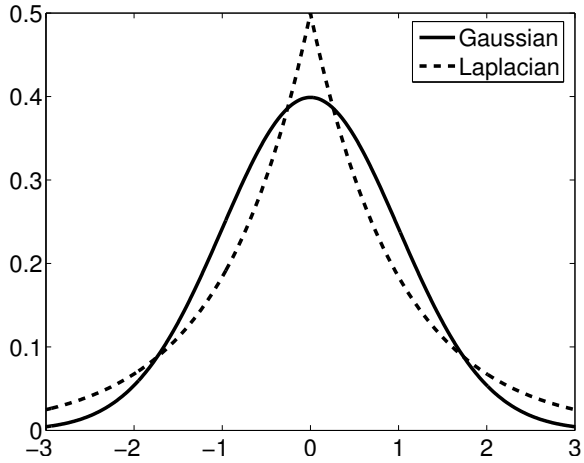


Figure 4.2: The density functions of a standard Gaussian (solid line) and a standard Laplacian (broken line) distributions.

4.2.5 A similar setting with SVM

Given the feature vectors, we may also build the decision boundary using an SVM. In the case of a linear kernel, like the logistic model, the SVM also builds a linear decision boundary to discriminate between the two classes. However, the results produced by an SVM is interpretable *only* when a linear (or possible *polynomial*) kernel is employed. While the objective functions in an SVM and a logistic regression settings are different, with proper regularization on the logistic regression model the results are often similar.

4.3 Experiments and results

We use the data set published in [35] to perform our experiments. The data set contains 54 families from SCOP 1.59 [47] with 7329 SCOP domains. No sequence shares more than 95% identity with other sequence in this data set. Variants of this data set have been used a s gold standard for protein remote homology detection in previous studies. Sequences in the SCOP database are domains extracted from proteins in the Protein Data Bank [7], which is a centralized repository for proteins with known three dimensional structures. Sequences in SCOP are placed in a tree-like hierarchy. Proteins in the same family clearly share a common revolutionary origin and proteins in the same superfamily typically have low sequence identity but are very likely to be evolutionarily

related. Remote homology detection means classification on the superfamily level. The tree hierarchy of the SCOP database is shown in Figure 1.1.

Jaakkola *et al.* proposed in [31, 30] the following setup for the experiments. Suppose a superfamily S^i with k families is under fold F^j. Pick the sequences in the $k-1$ families as the *positive training* sequences and use the sequences in the left-out as the *positive test* sequences. Negative training and testing sequences come from two different folds F^l and $F^m, l \neq m, l \neq j, m \neq j$, to avoid giving the classifier unnecessary advantage. All sequences in fold F^j but not in superfamily S^i are not used since their relationship with the *target* superfamily S^i is uncertain. In subsequent studies, Liao *et al.* [45], Leslie *et al.* [42, 43] and Kuang *et al.* [35] used different versions of the database as more sequences are deposited into the SCOP database.

We evaluate all methods using the *Receiver Operating Characteristic* (ROC) and ROC-50 scores [25]. The ROC-50 score is the (normalized) area under the ROC curve computed for up to 50 false positives. With small number of positive testing sequences and large number of negative test sequences the ROC-50 score is more indicative of the prediction accuracy of a homology detection method.

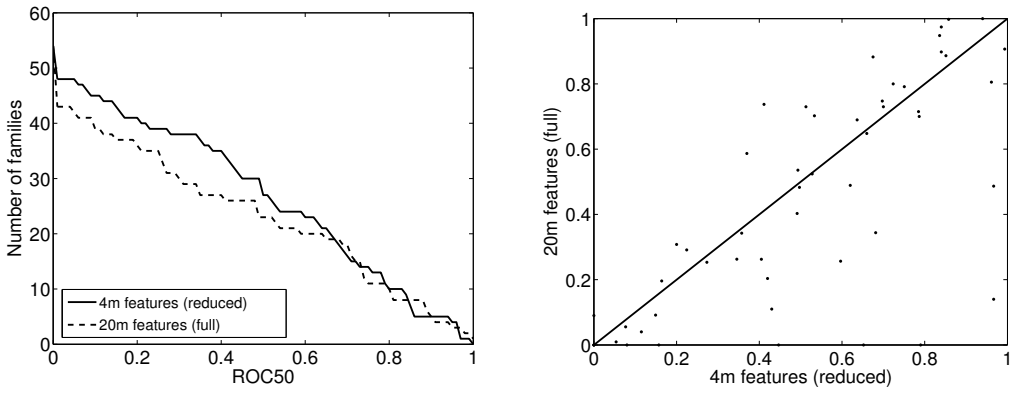
We obtain all profile HMMs for our *hybrid* procedure in the following way: first, we locate the profile most suitable for the experiment and download the (hand-curated) multiple alignment from PFam [5]; next, we estimate an initial profile HMM from the multiple alignment; finally we refine the profile HMM using the labeled positive training sequences in the data set. We use an algorithm similar to the Expectation-Maximization (EM) [16] algorithm to refine the profile HMM with 9-component mixture of Dirichlet priors [10, 60][2]. To avoid over-representation of sequences, we also incorporate a *position-based* sequence weighting scheme [27]. Once a profile HMM for the superfamily of interest is estimated, we use it to extract *fixed-length features*, the sufficient statistics with respect to the emission probabilities of the *match* states and we use the extracted features to train the *discriminative* classifier, the logistic regression model.

[2]With mixture of Dirichlet priors, the Maximization step can no longer be performed using closed-form solutions. As a result, to speed up estimation, instead of obtaining the *posterior mode*, we obtain the *posterior mean*. Typically the likelihood of the observed data increases up to three digits of precision after the decimal point. Then the algorithm starts bumping around some mode.

For logistic models, we perform our experiments on Normal and Laplace priors using *Bayesian Regression Software* (BBR) [22]. Precision γ in the Laplace models are set to the value suggested in the paper. Experiments using linear kernel SVM make use of an existing machine-learning package called *Spider* (available at *http://www.kyb.tuebingen.mpg.de/bs/people/spider*).

In Figure 4.3, we compare the performance of the full and reduced feature sets. The classifier used is the logistic classifier with Normal prior. The two sets of features perform similarly with SVM (linear kernel) and therefore are not reported. The dimensionality of the full feature set is $|\Sigma|m = 20m$ where m denotes the number of *match* states whereas the dimensionality of the reduced features is $4m$. Figure 4.3(a) shows the number of families (vertical axis) achieving a corresponding ROC-50 score (horizontal axis) for the two sets of features. We observe that the performance of the two sets of features are comparable, although in the area of low ROC-50 scores, the set of reduced features perform better, implying higher prediction accuracy. Figure 4.3(b) shows the pairwise scatter-plot of the ROC-50 scores for these two sets of features. A point falling under the diagonal line in the figure represents a case in which the reduced feature set achieves better performance. Out of 54 experiments, 28 and 21 of them fall under and above the diagonal, respectively. The p-value of the sign test is 0.39, indicating no strong evidence to support the claim that dimensionality reduction degrades the performance. In all subsequent reports, all logistic models use the reduced feature set.

In Figure 4.4 we compare the performance of different models. Figure 4.4(a) and 4.4(b) indicate that with ROC-50 score greater than 0.4, both logistic models (Normal and Laplacian priors) dominate the mismatch-(5,1) kernel. Furthermore, the performance of both logistic models are comparable in the area of high ROC-50 score (> 0.8); but in the area of low ROC-50 score, the logistic model with Laplacian prior shows slightly higher prediction accuracy. Finally, SVMFisher performs well in the are of high ROC-50 score; however, the performance starts to degrade when ROC-50 score falls under 0.8. In Figure 4.4(c), points falling above the diagonal corresponds to an experiment in which the logistic model with Normal prior performs better than the mismatch(5,1)



(a) (b)

Figure 4.3: Comparison of performance of the full and reduced feature sets. The classifier used here is the logistic classifier with Normal prior. Panel (a) shows the number of families whose ROC-50 scores are better than a given threshold for the sets of full and reduced features. Panel (b) depicts the pairwise scatter-plot of ROC-50 scores for the two classifiers utilizing these two sets of features.

kernel. Out of 54 experiments, 30 and 22 points fall above and below the diagonal, respectively, resulting in a p-value of 0.33, indicating no strong evidence to conclude which one of the two methods performs better. Likewise, in Figure 4.4(d), 25 and 25 fall above and under the diagonal line, suggesting that the performance of the logistic model with a Laplacian prior is comparable to that of a Normal prior.

We summarize the mean ROC and ROC-50 scores of different methods for a quick reference and comparison in Table 4.1.

Table 4.1: Mean ROC and ROC-50 scores for different homology detection methods

	mean ROC score	mean ROC-50 score
Logistic-Normal(4m)	.883256	.491564
Logistic-Laplace(4m)	.847313	.438070
Logistic-Normal(20m)	.813895	.426925
Logistic-Laplace(20m)	.864500	.474170
mismatch(5,1)	.874890	.416650
SVM-Fisher	.756618	.319048

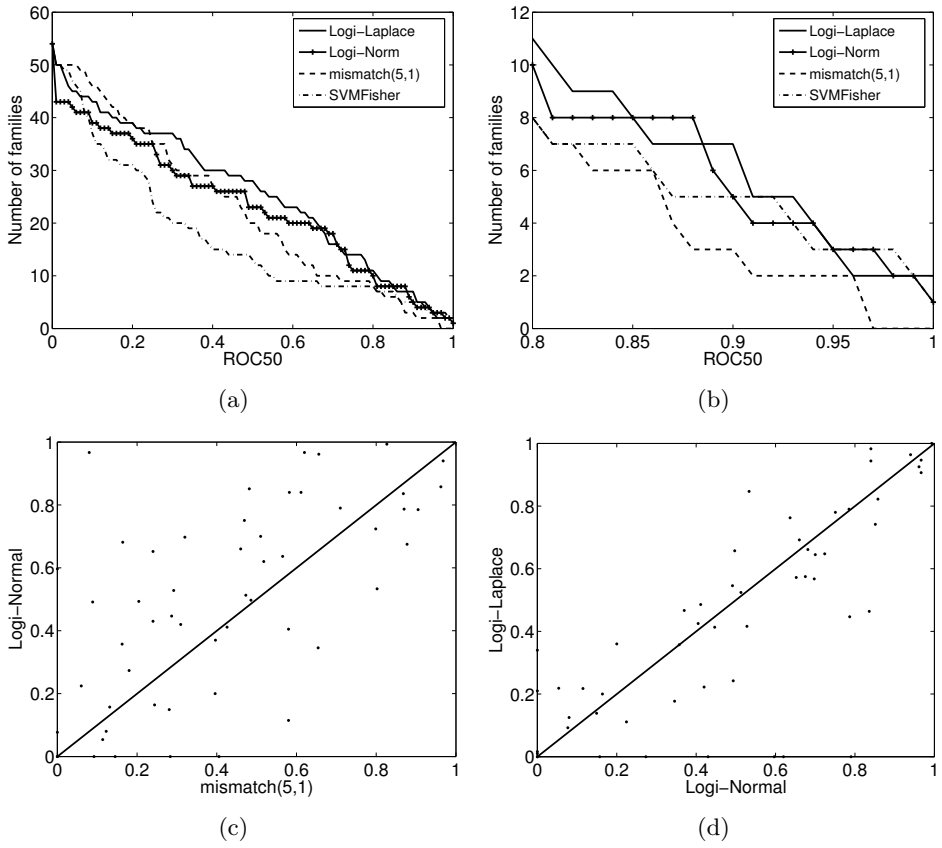


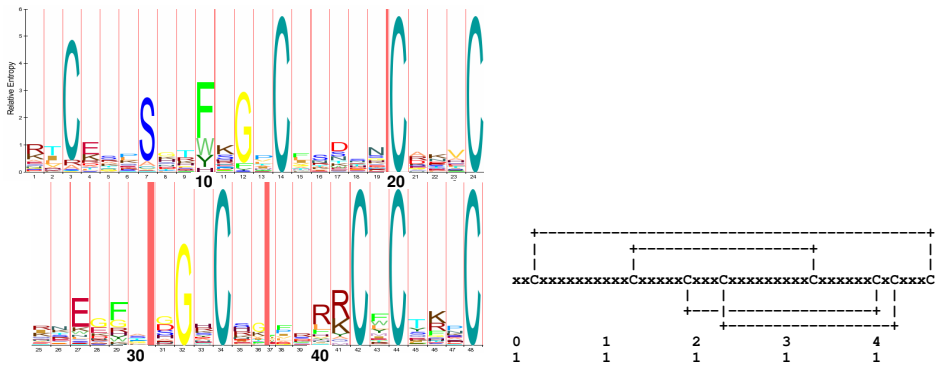
Figure 4.4: Comparison of performance of mismatch(5,1) kernel, SVM-Fisher, and logistic model with Normal and Laplacian priors. Panel (a) shows the number of families whose ROC-50 scores are better than a given threshold. Panel (b) shows the detail plot of the high ROC-50 score region of (a). Panel (c) shows the pairwise scatter-plot of ROC-50 scores for the logistic model with Normal prior and the mismatch(5,1) kernel. Panel (d) shows the pairwise scatter-plot of ROC-50 scores for the logistic models with Normal and scatter-plot of ROC-50 scores for the logistic models with Normal and Laplace priors.

4.3.1 The sparse models

Enforcing sparsity in the number of parameters can be viewed as a *feature selection* process. The logistic model with Laplacian priors discards the irrelevant features by setting the corresponding parameters to 0. Among 54 experiments, there are on average 480 features to select from. The Laplacian prior selects only about 43 features per experiment, resulting in more than 90% reduction in the final number of selected features. At the same time, the performance of the model with the reduced feature set remains indistinguishable from that of the model with a full feature set.

The set of features selected by the sparse model can offer interesting insights into the biological significance of the discovered *critical positions*. For example, our experimental results indicate that the performance of this class of classifiers is good and consistent on the *Scorpion toxin-like* superfamily. In one particular family, *Plant defensins*, out of 188 features, the logistic model with Laplacian prior selects 19 features, scattered on approximately 12 positions. The ROC-50 score of the classifier on this superfamily is 1. Upon further investigation, we extract these *critical positions* along with their preferred symbols: {(18[18], **E**), (20[20], **C**), (23[23], **H**), (24[24], **C**), (29[29], **G**), (34[32], **G**), (35[33], **K/Y**), (36[34], **C**), (37[35], **D/Y**), (38[36], **G/N**), (41[42], **C**), (43[44], **C**)}, where in each pair, the leading number corresponds to the position in our profile HMM, the number in the bracket corresponds to the position in the HMM-logo in Figure 4.5(a), and the letter the preferred symbol at that position. The positions slightly disagree because we use a different heuristic to determine whether a column in a multiple alignment corresponds to a *match* state or an *insertion* state. We also show the schematic representation of the family suggested by the *PROSITE database* [29] in Figure 4.5(b); in this figure each symbol 'C' represents a conserved cysteine involved in a disulphide bond. Hence, our classifier captures some of the conserved cysteine residues: (positions 20,24,34,41, and 43), with the preferred symbols that are *critical* for *discerning* plant defensins from other similar proteins. Other conserved cysteine residues are not selected as critical features. Upon detailed examination, it became clear that while conserved in Plant defensins, these positions also appeared to be conserved in other similar families with the symbol and were, therefore, not deemed discriminative. A set of 9 different residues seemed to play a more critical classification role.

Next, we extract the *positive* coefficients from the estimated θ for the logistic model and plot the weights in Figure 4.6(a). The figure indicates that there are three *critical region* for this family: positions $18-29$, $33-39$, and $40-44$. We obtain the primary and secondary structure in schematic diagrams of 8 sequences belonging to this family from the *PDBSum database* [39]. We observe some common secondary structure among these 8 sequences: a *β-strand* occurs in the neighborhood of positions $3-7$, and a *helix* occurs in in the neighborhood of positions $18-28$, followed by two more *β-strands*



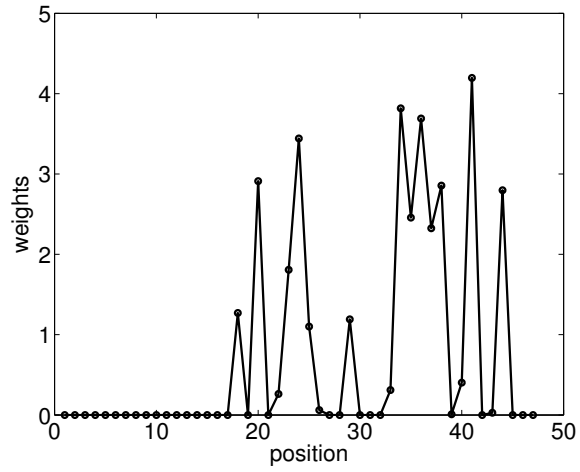
(a) (b)

Figure 4.5: Panel (a): The HMM-logo of the *plant defensins* family (under the *scorpion toxin-like* superfamily. We obtain this logo from PFam. Panel (b): The schematic representation of the *plant defensins* family suggested by PFam and PROSITE.

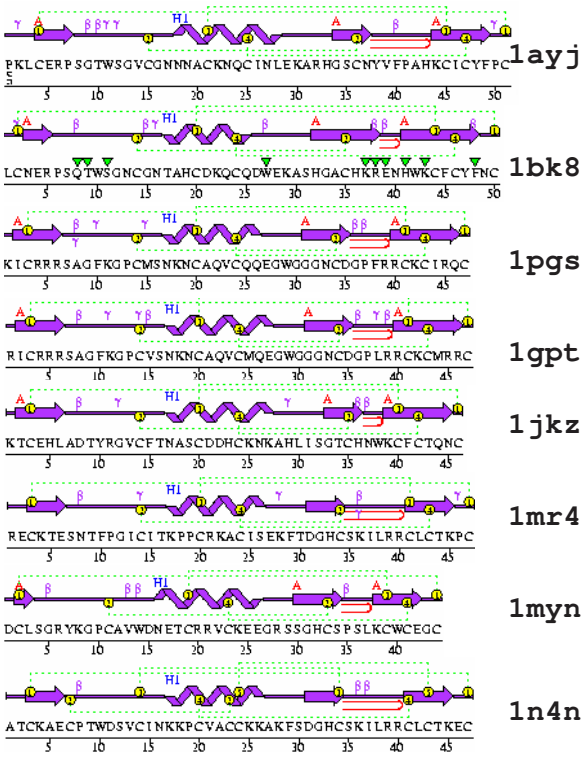
occurring in the neighborhood of positions $32-38$ and $41-48$, respectively. There seems to be some correlation between these *conserved* secondary structures and the *critical regions* indicated by our logistic model with Laplacian prior. We are currently performing further analysis to understand the connection between them.

We obtain similar results for another family under the same superfamily: the *short-chain scorpion* family. The ROC-50 score of the classifier for this family is 0.91. We show the HMM-logo for this family in Figure 4.7(a). Out of 116 features in this family, the sparse classifiers selects 14 of them: { (1, **C**), (4, **N**), (7, **C**), (11,**C**), (15, **G**), (17, **A**), (18, **S**), (19, **G/S**), (20, **G**), (21, **Y**), (22, **C**), (24, **G**), (27, **C**), (29, **C**)}. Our sparse model captures all conserved cysteine residues in this family, indicating that unlike in the family *plant defensins*, these conserved cysteine residues are *unique* to this family. On the other hand, the conserved lysine (K) residue at positions 12, 21, and 26 in Figure 4.7(a) are deemed *insignificant* by our sparse model because sequences not belonging to this family also have such residue aligned to these positions.

Finally, we note that the dense models with Normal priors also achieve classification performance similar to the sparse model. However, the weights learned by the dense models did not allow any immediate interpretation of importance nor selection of a small set of critical discriminative features.

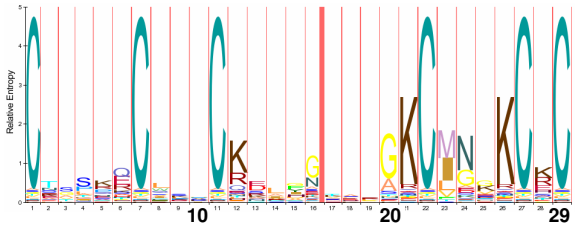


(a)

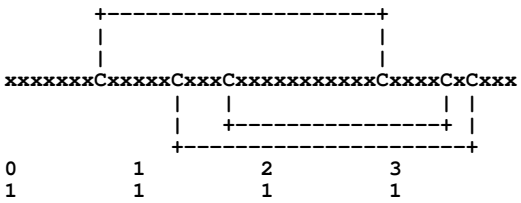


(b)

Figure 4.6: Panel (a): The sum of the positive coefficients in each position for the *Plant defensins* family. Panel (b): The primary and secondary structure, in schematic diagrams, of eight sequences belonging to the this family. We obtain the diagrams from PDBsum.



(a)



(b)

Figure 4.7: Panel (a): The HMM-logo of the *short-chain scorpion toxins* family. This logo is obtained from PFam. Panel (b): the schematic representation of this family suggested by PFam and PROSITE.

4.4 Relationship to kernel methods

Tseuda *et al.* proposed the *marginalized kernels* in [66] for biological sequence analysis. Let $x \in \mathcal{X}$ be a set of *observable* variables and $h \in \mathcal{H}$ be a set of *unobservable* (*hidden*) variables. The authors define $K_z(z, z')$ as the *joint kernel*, where $z = (x, h)$. Then in the *marginalized kernel* setting, the similarity between two examples x and x' is defined as:

$$K(x, x') = \sum_{h \in \mathcal{H}} \sum_{h' \in \mathcal{H}} p(h|x) p(h'|x') K_z(z, z'), \tag{4.14}$$

where we sum over all hidden variables. Further, given two sequences X and Y, define the *count kernel* as:

$$K(X, Y) = \sum_{\sigma \in \Sigma} c_\sigma(X) c_\sigma(Y), \tag{4.15}$$

where $c_\sigma(X)$ denotes the number of times the symbol σ occurs in sequence X. Finally, let $\mathcal{H}$ be the set of states in an HMM, Tseuda *et al.* showed that the corresponding *marginalized count kernel* is:

$$\begin{aligned} K(X, Y) &= \sum_{h, h'} p(h|X) p(h'|Y) K_z(Z_X, Z_Y) \\ &= \sum_{\tilde{x}, \tilde{s}} \xi(\tilde{x}, \tilde{s}|X) \xi(\tilde{x}, \tilde{s}|Y) \end{aligned} \tag{4.16}$$

for all $\tilde{x} \in \Sigma$ and $\tilde{s} \in S$, where S represents the set of states in the HMM. Equation 4.16 indicates that, under the kernel setting, the kernel induced by our feature set is a *marginalized count kernel*. We also note that Equation 4.15 implies that the *spectrum-k* kernel is a k-th order count kernel, in which σ spans through all possible k-mers induced by Σ. Finally Tsuda *et al.* also showed that the *SVMFisher* also corresponds to a special case of marginalized count kernel.

4.4.1 Comparison with the spectrum-k kernel

In Equation 4.16, each hidden variable is associated with a posterior probability obtained from the forward-backward algorithm. We call these the *soft labels*. However, one may also use *hard labels* in such setting: determine the labels using the Viterbi path, the most probably path that generated the observed sequence. Use of the Viterbi path results in the following similarity measure between two sequences X and Y:

$$\begin{aligned} K_V(X,Y) &= \sum_{t,t'} \sum_{\tilde{x}} [I(X_t = Y_{t'} = \tilde{x})] \cdot \\ &\quad [\sum_{\tilde{s}} I(V(t|X) = V(t'|Y) = \tilde{s})], \end{aligned} \tag{4.17}$$

where $V(t|X)$ denotes the state that symbol X_t aligns to in the Viterbi sequence; on the other hand, the similarity between X and Y defined by the *spectrum-k* kernel with $k = 1$ is:

$$K_{S(1)}(X,Y) = \sum_{t,t'} \sum_{\tilde{x}} I(X_t = Y_{t'} = \tilde{x}). \tag{4.18}$$

As a result, the kernel induced by our feature set also has a close relationship with the spectrum-k kernel. The extra term in the end of Equation 4.17 is imposed by our feature extractor, in this case, the profile HMM. The impact of this term can be seen in the following example. Consider a new sequence Y', obtained by randomly permuting Y; then it is very likely that $K_V(X,Y') \neq K_V(X,Y)$ since Y' will align differently with the feature extractor; on the other hand, it is clear that $K_{S(1)}(X,Y') = K_{S(1)}(X,Y)$. We believe that this is one of the reasons that in the string kernel setting, k must be a moderately large number, for example 3 or 5. In contrast, our equivalent kernel is able

to exploit the existence of latent match states in computing a tractable and empirically effective similarity score.

Furthermore, upon close examination of Equations 4.17 and 4.18, the kernel induced by our features using the Viterbi path *is a spectrum-1 kernel* with an *augmented alphabet set*. While the *spectrum-k* kernel uses the 20 amino acids as the alphabet set, the kernel induced by our features augments the alphabet set Σ to $\Sigma' = \Sigma \times Z_m^+$, where Z_m^+ is the set of all positive integers up to m, the number of *match* states in the profile HMM.

For the mismatch(k,m_k) kernel, computing each element in the kernel matrix requires $O(k^{m_k+1}|\Sigma|^{m_k}(T_X + T_Y))$ time. Denote m as the number of *match* states in the profile HMM; using our *hybrid* model with a linear kernel, computation of each element in the kernel matrix requires $O(m)$ time, since only the inner product of the sufficient statistics of two sequences needs to be computed. The complexity of the forward-backward procedure, required to obtain the sufficient statistics of a sequence X with length T_X is $O(m(T_X + |\Sigma|))$; the complexity is *linear* instead of quadratic in m due to the *linear* structure of the profile HMMs. Finally, the complexity of constructing the feature extractor (profile HMM) is $O(mnT)$ where n is the number of *labeled positive* examples and T the length of the longest sequence in the training set. Among 54 experiments, the total number of positive sequences is 1398; each family on average has 26 positive training sequences. Each profile HMM on average has 123 *match* states; the average positive sequence length is 147 residues per sequence. Each profile on average takes 12 E-M like iterations to train and takes 105 seconds on a $2.80GHz$(x2) machine with 1024MB of RAM; as for logistic models, given the features it takes BBR on average 1 minute to estimate a model and 3 seconds to predict classification results per experiment.

More recent methods based on *profile kernels* [35] have shown significant promise. Unlike our setting, profile kernels leverage the benefits of unlabeled data. Also each sequence is represented by a profile, resulting in increased computational complexity of the classification approach. As a result, the method is fundamentally different from those compared in this paper and we do not include the comparison in the current study.

4.5 Conclusion

In this study we introduce a method for learning *sparse* feature models for the remote homology detection problem. To extract the features, we use a profile HMM that represents the superfamily of interest. These features are the sufficient statistics of the query sequence with respect to the designed profile HMM. As such, the features offer insight to the underlying evolutionary process such as the degree of conservation of each position in the superfamily.

Using interpretable logistic classifiers with Laplace priors, the learned models exhibit more than 90% reduction in the number of selected features. These results indicate that it may be possible to discover very *sparse* models for certain protein superfamilies, which might confirm the hypothesis suggested in [33, 55, 34] that a small subset of positions and residues in protein sequences may be sufficient to discriminate among different protein classes. We show that the *sparse* model select some *critical positions* that are consistent with current reports. However, at present the full set of selected positions may not fully agree with the proposed hypotheses. Further analysis is needed to study the correspondences between the computation and hypothesized models.

In our future work we will further investigate and consider biological interpretation of the resulting *sparse* models. In addition, we will expand our framework to utilize additional sets of physically motivated features as well as the unlabeled data, leveraging the benefits of large training sets.

Chapter 5

Joint Training and Feature Sharing for Protein Remote Homology Detection

In the brief introduction to the *joint training and feature sharing* framework presented in Section 2.6, we noted that in the context of the Structural Classification of Proteins (SCOP) proteins form a tree-like hierarchy according to their structural similarity, as shown in Figure 1.1. Naturally, we expect sharing of features among sub-classes. In this chapter, we test this hypothesis on two different types of features under the joint training and feature sharing framework:

- The set of sufficient statistics induced by the profile HMMs.
- The mismatch(5,1) features proposed in [43].

We again use the SCOP 1.59 data set for performance evaluation. The data set contains 54 experiments with 23 superfamilies.

5.1 Complexity of the joint training framework

We provide the *greedy* algorithm discussed in Chapter 2:

Algorithm for joint training and feature sharing

Input: Set of examples $\{(x_i, y_i)\}_{i=1}^{N}$ and set of labels Y

1. Initialize the weights $w_i^y = 1$ and set $H(x_i, y) = 0, \forall 1 \leq i \leq N, y \in Y$.

2. repeat for $m = 1, 2, \cdots, M$

   a: $C(0) \leftarrow \{\}, C' \leftarrow Y, n \leftarrow 1$

   b: while $C' \neq \{\}$

      i):for each $c' \in C'$

      $S' \leftarrow C(n-1) \cup \{c'\}$

      Fit shared stump for every feature f:

$$h_m(x, y) = \begin{cases} a\delta(x^f > \theta) + b & \text{if } y \in S' \\ k^y & \text{otherwise} \end{cases}$$

      Evaluate error for every feature f:

$$J_{wse}(n) = \sum_{y \in Y} \sum_{i=1}^{n} w_i^y (z_i^y - h_m(x_i, y))^2$$

      ii):Find the best combination $((c')^*, f^*)$ that maximizes the objective function J.

      iii):$C(n) \leftarrow C(n-1) \cup \{(c')^*\}, C' \leftarrow C' \backslash \{(c')^*\}, n \leftarrow n+1$

   c: Find the best *candidate* set $C(i^*)$ where $i^* = \text{argmin}_i J_{wse}(i)$ and the feature f

   d: $\forall 1 \leq i \leq N, y \in Y$, update:

$$H(x_i^f, y) \leftarrow H(x_i^f, y) + h_m(x_i^f, y)$$

$$w_i^y \leftarrow w_i^y e^{-z_i^y h_m(x_i^f, y)}$$

Output: The final boosted joint classifier: $H = \{h_m(f, C^*, a, b, \theta)\}_{m=1}^{M}$

$z_i^y = 1$, if the label of example x_i is y, otherwise, $z_i^y = -1$

The algorithm is quoted from [65]

With M boosting rounds, N training examples, the complexity of the greedy algorithm is $O(NMd|\Theta||Y| + dM|Y|^3|\Theta|)$ where d is the dimensionality of the feature space and Θ is the set of all possible values for the parameter θ; for continuous variables, $|\Theta| \in O(N)$. The complexity of the algorithm is *cubic* in the size of Y and with

continuous variables, *quadratic* in the size of the training set.

5.2 Experimental Results

We first present the results obtained using sufficient statistics as features and then we present the results with mismatch(5,1) features.

5.2.1 The sufficient statistics

The joint training framework requires fixed dimensionality over all experiments. However, the profile HMMs for each experiments have variable length which indicates that the dimensionality of the features are different across experiments. Padding the short feature sets with 0 to accommodate longer feature sets is a possible solution but we need to select the positions for padding carefully or it will not have any biological meaning. To obtain the positions for padding, we need to perform a multiple alignment on all (54) available profile HMMs and possibly with the help of phylogenetic trees. Such requirement will result in a search problem with a very large search space.

Consequently, under this learning setting we resort to performing all 54 detection problems *separately* and *independently* as *binary* class problems and we cannot achieve sharing of features with this feature set. In each detection problem, we set the negative training sequences as the *background* sequences. We show the ROC-50 plots in Figure 5.1. In Figure 5.1(a) we compare the boosted tree stump with $10, 30, 50, 70, 90$ boosting iterations (solid, color lines) with the mismatch(5,1) kernel (black dashed line) and the kernel induced by the whole set of sufficient statistics (black line with '+' sign). Figure 5.1(b) provides a more detailed view in the high ROC-50 area. We observe from the plot that while boosted tree stumps show some substantial degradation from the kernel induced by the whole set of sufficient statistics, such performance is accomplished with as few as 10 features. Moreover, the boosted tree stumps demonstrate comparable performance with the mismatch(5,1) kernel, which corresponds to $3,200,000$ features. We also provide the mean ROC and ROC-50 scores for the competing classifiers in Table 5.1.

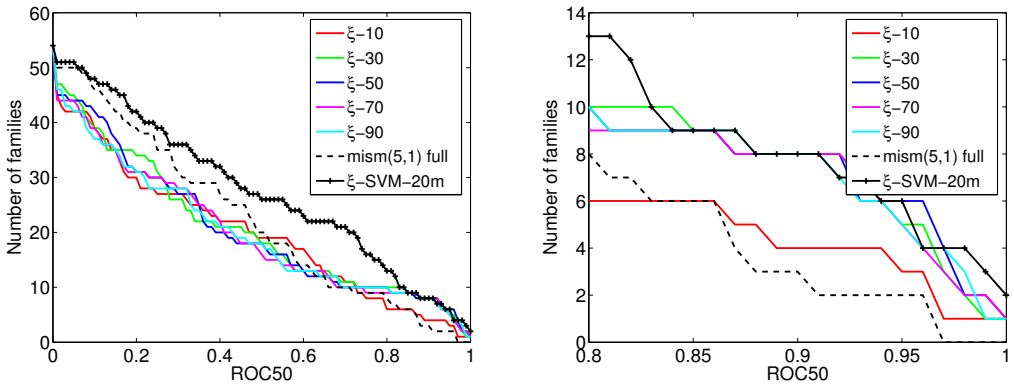


Figure 5.1: ROC50 curve for the sufficient statistics features and full mismatch(5,1) features. The horizontal axis represents a given ROC50 score and the vertical axis denotes the number of experiments, out of 54, achieved higher than or equal to the specified ROC50 score. For sufficient statistics, the number of selected features is denoted by the number of rounds. The ROC50 curve of mismatch(5,1) is achieved using the **full** set of features, which corresponds to $3,200,000$ features . The sub-figure in the right panel is a detailed view of the curves in the high ROC50 area.

Table 5.1: Mean ROC and ROC-50 scores for different number of selected features using sufficient statistics as features.

	mean ROC score	mean ROC-50 score
sufficient statistics (10 features)	.637628	.367255
sufficient statistics (30 features)	.748540	.380518
sufficient statistics (50 features)	.750328	.376488
sufficient statistics (70 features)	.750428	.370480
sufficient statistics (90 features)	.756543	.368080
mismatch(5,1) (3.2M features)	.874890	.416650

5.2.2 The mismatch(5,1) features

For the mismatch(k,m) kernel, the dimensionality of the feature space is the set of all possible k-mers induced by the alphabet set. Moreover, the corresponding feature extractor is constant across all experiments whereas for the sufficient statistics, each experiment has a different feature extractor. As a result, the kernel matrix is constant across all experiments and joint training and feature sharing with the feature set induced by the mismatch(5,1) kernel is possible. However, as suggested in Section 5.1, the complexity of joint training and feature sharing framework is *cubic* in the number of

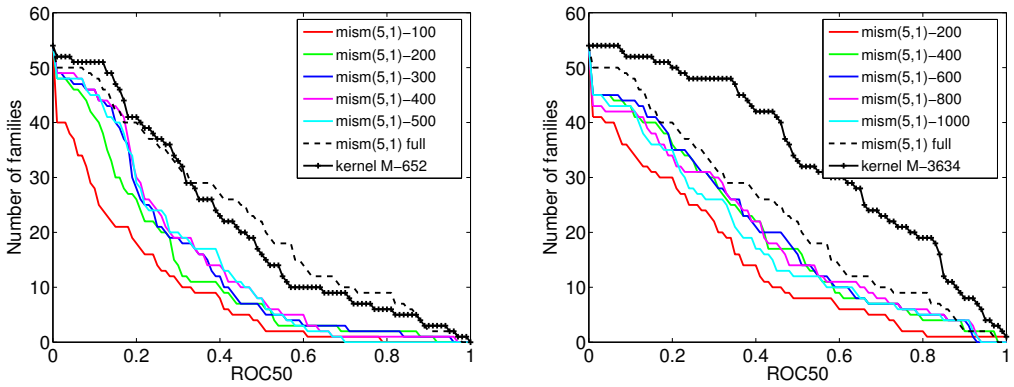
classes, which is 23 on this particular data set and *linear* in the dimensionality of the feature space, which is exponential in k (for proteins, $|\Sigma| = 20$ and when k=5, $|\Sigma|^k = 3,200,000$). Therefore *pre-selection* of features is necessary.

We construct *explicit* representations for all sequences in the SCOP 1.59 data set. Such construction is possible only when we employ the *sparse* representation. The data matrix is *extremely sparse*. With 7,329 sequences, only 117,366,759 entries have non-zero counts, which only constitute 0.5% of the data matrix. The data set induces 3,189,331 unique mismatch(5,1) features out of 3.2 million of possible features. On average, each feature is shared by 37 sequences, which indicates very little overlap in features. To pre-select features, we use the χ^2 statistics, a well-known statistic often used to determine independence between variables in *contingency tables*. The larger the χ^2 score is, the less likely that the two variables of interest are independent of each other. In the experiment, the two variables we pick are the features and the class label.

First we pre-select 652 features and present the ROC-50 plot in Figure 5.2(a). In the figure, we compare the boosted tree stump with 100, 200, 300, 400 and 500 mismatch(5,1) features (solid color lines), the kernel induced by the *full* mismatch(5,1) features (black dashed line) and the kernel induced by the *pre-selected* 652 features with high χ^2 scores. First, we observe that the kernel induced by the pre-selected features performs on par with the kernel induced by the full feature set, suggesting that the quality of the pre-selected features is not compromised. However, the performance of the boosted tree stumps is very poor. Upon detailed investigation, we observe that the positive training and test sequences do not share many features. In addition, even sharing of features among *all positive training* sequences is scarce.

To further understand the problem, we perform an *unfair test*. In this unfair experiment, we continue using the χ^2 statistics to pre-select the features but *with the knowledge of the positive testing sequences*. We show the result of the experiment in Figure 5.2(b). In the figure we compare the boosted tree stumps with 200, 400, 600, 800 and 1,000 features (solid color lines), the kernel induced by the full mismatch(5,1) features (black dashed line) and the kernel induced by the pre-selected mismatch(5,1) features with the knowledge of positive testing sequences (black line with '+' sign).

First, we observe that the kernel induced by the pre-selected features demonstrate significant improvement over the kernel induced by the full feature set, which is expected. This suggests again that the quality of the selected features is not compromised. However, even with such optimal feature set, the boosted tree stumps still demonstrate poor performance. Though larger rounds of boosting is possible and should be carried out, but with complexity cubic in $|Y|$ the computational time required becomes unreasonable.



(a) (b)

Figure 5.2: ROC50 curve for the mismatch(5,1) features. In the left panel, we show the performance of boosted tree stumps with $100, 200, 300, 400$ and 500 iterations (solid color lines) over 652 pre-selected features using the χ^2 scores. We also compare with the kernel induced by the full mismatch(5,1) features (black dashed line) and the kernel induced by the pre-selected features (black line with '+' sign). In the right panel, we pre-select 3634 features using χ^2 score *with knowledge of the positive test sequences*. We compare the boosted tree stumps with $200, 400, 600, 800$ iterations (solid color lines), the kernel induced by the full mismatch(5,1) feature set (black dashed line) and the kernel induced by the pre-selected features (black line with '+').

With further investigation, we observe that the joint training feature sharing algorithm exhibit some interesting behavior. First, we observe that even among positive training sequences, sharing of features is scarce: in the m^{th} boosting round, the algorithm picks feature f^m, shared by a few positive training sequences ($X(m)$); the estimated tree stump makes mistakes on the positive training sequences ($\bar{X}(m)$) without feature f^m, leading to increase of weight for these examples. In the next iteration, the algorithm focuses on examples $\bar{X}(m)$ and picks a feature $f^{m+1} \neq f^m$ shared by $\bar{X}(m)$. If in examples $X(m)$ feature f^{m+1} is absent the $(m+1)^{th}$ tree stump make

mistakes on $X(m)$, causing the weights of these examples to increase, which is typically the case according to our observation. Consequently, the algorithm picks feature f^m in the $(m+2)^{th}$ boosting round. The learning algorithm ends up bouncing back and forth on these features.

To resolve the observed problem, we further enforce the constraint that once a feature is selected by the tree stump, the same feature becomes unavailable and can never be selected again. Under such constraint, we observe that the mismatch(5,1) features exhibit complex correlation structure. Denote $N(\alpha, m)$ as the the *m-neighbor* of a k-mer feature α (all possible k-mers with up to m mismatch). If any k-mer α has a high χ^2 score, then any $\gamma \in N(\alpha, m)$ is also very likely to have a high χ^2 score, since under such feature space, observing α induces observation for all $\gamma \in N(\alpha, m)$. Consequently, all k-mers in $N(\alpha, m)$ and α will be pre-selected and these highly correlated mismatch(5,1) features form clusters made up of the m-neighbors in the the pre-selected feature set with the choice of $m = 1$ in this experiment. This indirectly breaks the constraint since once a feature is picked by a tree stump, although the same feature is not eligible for picking, but its 1-neighbor will very likely to be picked by the tree stump in subsequent boosting iterations; with such dependency structure, the algorithm exhibit the same behavior discussed in the previous paragraph.

Finally we also note that pre-selection of features should be performed very carefully when using high-dimensional features with very little overlap such as the mismatch(5,1) features. We observe that some examples end up with a *null* representation: they posses *none* of the pre-selected features. Numerical problems may occur in such cases when using classifiers that rely on the rank of the data matrix.

5.2.3 Discussion

In the experiment with sufficient statistics, we show that with fewer than 100 features the boosted tree stumps achieve performance comparable to that of the mismatch(5,1) kernel on the full feature set. We provide the following explanation. Let m_p denote the number of *match* states in the profile. Using sufficient statistics as features, we obtain $|\Sigma|m_p$ features ($|\Sigma|$ features for each position). The correlation structure is

only present *within* each position. Between each pair of positions, the correlation has been *integrated out* by the forward and backward inference procedure. We believe that the poor performance of boosted tree stumps on the mismatch(5,1) feature set is not directly related to the discriminative power of the classifier. We strongly believe the problem actually comes from the scarce feature overlap among examples in the same class and the complex correlation structure among the features.

5.3 Conclusion

In this study, we apply the joint training and feature sharing framework to two different type of features. First we apply the algorithm on the sufficient statistics feature set and show that with fewer than 100 boosting iterations, the performance of the boosted tree stumps is comparable to that of the kernel induced by the full set of mismatch(5,1) features. A major shortcoming of the sufficient statistics is the underlying feature extractor is not constant over different superfamilies and therefore we cannot take full advantage of feature sharing framework.

We also apply the algorithm on the mismatch(5,1) feature set. The algorithm requires explicit feature representation and the complexity is linear in the dimensionality of the features and cubic in the number of classes, thus incurring the need to perform feature pre-selection. However, the pre-selection process picks clusters of features, which makes the algorithm ineffective. In addition, the complex correlation structure among features also makes combination of this feature set and the boosted tree stumps undesirable. As we have shown using the toy example designed in Section 2.6 and using the sufficient statistics to perform protein remote homology detection, the boosted tree stumps do have discriminative power. The key is to choose a *sensible* and preferably *orthogonal* feature set with sufficient feature overlap among examples in the same class. For future works, we propose to search for such feature set for protein remote homology detection. We also propose to group mismatch(5,1) features first, pick a representative feature from the group with high scores and apply the joint training and feature sharing algorithm on the pre-selected representatives.

Chapter 6

Fast and Accurate Semi-supervised Protein Homology Detection with Large Uncurated Sequence Databases

Remote homology detection problem is typically characterized by very few *positive training* sequences accompanied by a large number of negative training examples. Experimentally labeling the sequences is costly, leading to the need to leverage *unlabeled* data to refine the decision boundary. The profile kernel [35] and the mismatch neighborhood kernel [69] both use large unlabeled data sets and show significant improvements over the sequence classifiers trained under the supervised setting. We believe the major contributions for their great success come from first, leveraging unlabeled data and second, the use of *mutational neighborhood* to model the amino acid substitution process. However, kernel evaluation based on the induced mutational neighborhood incurs exponential complexity in the size of the alphabet set hence hindering the use of such powerful tools.

Another missing component in previous studies for large-scale semi-supervised protein homology detection is a systematic and biologically motivated approach for leveraging the unlabeled data set. In this study, we address both issues. First, we employ a class of previously established kernels, the Sparse Spatial Sample Kernels (SSSK) [38]. This class of biologically motivated kernels model mutation, insertion and deletion effectively and induce low-dimensional feature space; moreover the complexity of kernel evaluation based on feature matching is independent of the size of the alphabet set and such key characteristics opens the door for rapid large-scale semi-supervised learning. Second, we propose a biologically meaningful way of extracting relevant information from the unlabeled database for semi-supervised learning. Third, we also propose a

method to remove the bias caused by overly represented or duplicated unlabeled sequences, which are common in large uncurated sequence databases. Our experimental results show that the combination of these approaches yields state-of-the-art performance that are significantly better than previously published methods and also exhibit order-of-magnitude differences in experimental running time.

6.1 Background

In this section, we briefly review previously published state-of-the-art methods for protein homology detection. We denote the alphabet set as Σ in the whole study. Given a sequence X the *spectrum-k* kernel [42] and the *mismatch(k,m)* kernel [43] induce the following $|\Sigma|^k$-dimensional representation for the sequence:

$$\Phi(X) = \left(\sum_{\alpha \in X} I(\alpha, \gamma) \right)_{\gamma \in \Sigma^k}, \tag{6.1}$$

where under the spectrum-k kernel, $I(\alpha, \gamma) = 1$ if $\alpha = \gamma$ and under the mismatch(k,m) kernel, $I(\alpha, \gamma) = 1$ if $\alpha \in N(\gamma, m)$ and $N(\gamma, m)$ denotes the set of k-mer *mutational neighborhood* induced by the k-mer γ for up to m mismatches or substitutions.

Both spectrum-k and mismatch(k,m) kernel directly extract string features based on the observed sequence. Under the mismatch representation, all substitutions are treated as equally likely, which may not be deemed practical due to the physical and chemical properties of amino acids. The profile kernel [35] takes such constraints into consideration: given a sequence X and its corresponding profile [24] P_X, Kuang *et al.* [35, 36] define the $|\Sigma|^k$-dimensional profile(k,σ) representation of X as:

$$\Phi^{profile(k,\sigma)}(X) = \left(\sum_{i=1 \cdots (T_{P_X} - k + 1)} I(P_X(i, \gamma) < \sigma) \right)_{\gamma \in \Sigma^k}, \tag{6.2}$$

where σ is a pre-defined threshold, T_{P_X} denotes the length of the profile and $P_X(i, \gamma)$ the cost of *locally* aligning the k-mer γ to the k-length segment starting at the i^{th} position of P_X. Explicit inclusion of the amino acid substitution process allows both the mismatch and profile kernels to significantly outperform the spectrum kernel and demonstrate state-of-the-art performance under both supervised and semi-supervised

settings [69, 35]. However, such method of modeling substitution process induces a k-mer mutational neighborhood that is exponential in the size of the alphabet set during the matching step for kernel evaluation; for the mismatch(k,m) kernel, the size of the induced k-mer neighborhood is $k^m|\Sigma|^m$ and for the profile(k,σ) kernel, the size of the neighborhood is bounded below by $k^m|\Sigma|^m$, above by $|\Sigma|^k$, and is dependent on the threshold parameter σ. Increasing m or σ to incorporate more mismatches will incur higher complexity for computing the kernel matrix hence hindering the use of such powerful tools.

Finally, to construct the sequence profiles required for computation of the profile kernel, we need to leverage the unlabeled sequences to avoid overfitting of the profile. For the mismatch string kernel, Weston *et al.* propose to use the *sequence neighborhood kernel* to leverage the unlabeled sequences in [69].

6.1.1 The sequence neighborhood kernel

The sequence neighborhood kernels take advantage of the unlabeled data using the process of neighborhood induced regularization. Let $\Phi^{orig}(X)$ be the original representation of sequence X. Also, let $N(X)$ denote the *sequence neighborhood* of X [1]. Weston *et al.* proposed in [69] to re-represent X using:

$$\Phi^{new}(X) = \frac{1}{|N(X)|} \sum_{X' \in N(X)} \Phi^{orig}(X'). \tag{6.3}$$

Under the new representation, the kernel value between the two sequences X and Y becomes:

$$K^{nbhd}(X,Y) = \sum_{X' \in N(X), Y' \in N(Y)} \frac{K(X',Y')}{|N(X)||N(Y)|}. \tag{6.4}$$

Note that under such settings, all *training* and *testing* sequences will assume a new representation, whereas in a traditional semi-supervised setting, unlabeled data are used during the *training phase only* (the framework we adopted in Chapter 3). The authors choose the mismatch representation for the sequences and show that the discriminative

[1] We will discuss how to define $N(X)$ in later sections.

power of the classifiers improve significantly once information regarding the neighborhood of each sequence is available. However, the exponential size of the incurred k-mer mutational neighborhood makes large-scale semi-supervised learning under the mismatch representation very computationally demanding and cannot be performed using only moderate computational resources.

6.2 Proposed methods

In this section, we first discuss the *sparse spatial sample kernels* (SSSK) for protein homology detection. Such kernels effectively model the insertion, deletion and substitution processes and the complexity of the string matching step for kernel evaluation is independent of the size of the alphabet set. The kernels show very promising results under the supervised setting and also under the semi-supervised setting with a small unlabeled sequence data set [38]. Next, we discuss a systematic and biologically motivated way to extract only relevant information from the unlabeled database. Finally we also discuss how to remove the bias caused by duplicated or overly represented sequences which are commonly found in large uncurated sequence databases. The combination of the proposed methods enables fast and accurate semi-supervised learning for protein homology detection.

6.2.1 The sparse spatial sample kernel

The class of *sparse spatial sample kernels*, proposed by Kuksa *et al.* [38] have the following form:

$$K^{(t,k,d)}(X,Y) = \sum_{\substack{(a_1,d_1,\ldots,d_{t-1},a_t) \\ a_i \in \Sigma^k, 0 \leq d_i < d}} C(a_1,d_1,\cdots,a_{t-1},d_{t-1},a_t|X) \cdot C(a_1,d_1,\cdots,a_{t-1},d_{t-1},a_t|Y), \tag{6.5}$$

where $C(a_1,d_1,\cdots,a_{t-1},d_{t-1},a_t|X)$ denotes the number of times we observe substring $a_1 \overset{d_1}{\leftrightarrow} a_2, \overset{d_2}{\leftrightarrow}, \cdots, \overset{d_{t-1}}{\longleftrightarrow} a_t$ (a_1 separated by d_1 characters from a_2, a_2 separated by d_2 characters from a_3, etc.) in the sequence X. This is illustrated in Figure 6.1. The kernel implements the idea of sampling the sequences at different resolutions and comparing the resulting spectra; similar sequences will have similar spectrum at one or

more resolutions. This takes into account possible mutations as well as insertions and deletions. Each sample consists of t spatially-constrained probes of size k, each of which lie less than d positions away from its neighboring probes. The parameter k controls the individual probe size, d controls the locality of the sample and t controls the cardinality of the sampling neighborhood. In this work, we use short samples of size 1 (*i.e.* $k = 1$) and set t to 2 (*i.e.* features are pairs of monomers) or 3 (*i.e.* features are triples of monomers). The spatial sample kernels, unlike the family of spectrum kernels [42, 43], not only take into account the feature counts, but also include spatial configuration information, *i.e.* how the features are positioned in the sequences. The spatial information can be critical in establishing similarity of sequences under complex transformations such as the evolutionary processes in protein sequences. The addition of the spatial information experimentally demonstrates very good performance, even with very short sequence features (*i.e.* $k = 1$), as we will show in section 6.4.

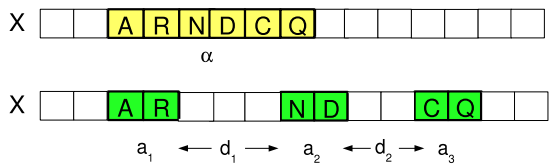


Figure 6.1: Contiguous k-mer feature α of a traditional spectrum/mismatch kernel (top) contrasted with the sparse spatial samples of the proposed kernel (bottom).

The use of short features can also lead to significantly lower computational complexity of the kernel evaluations. The dimensionality of the features induced by the spatial sample kernels is $|\Sigma|^t d^{t-1}$ for the choice of $k = 1$. As a result, for triple(1,3) ($k = 1$, $t = 3$, $d = 3$) and double-(1,5) ($k = 1$, $t = 2$, $d = 5$) feature sets, the dimensionalities are $72,000$ and $2,000$, respectively, compared to $3,200,000$ for the spectrum(k) [42], mismatch(k,m) [43] and profile(k,σ) [35] kernels with the common choice of $k = 5$. In Figure 6.2 we show the differences between the spatial (double(1,5)) and the spectrum (mismatch(5,1)) features on two slightly diverged sequences, S and S'. In the mismatch features, each symbol 'X' represent an arbitrary symbol in the alphabet set, Σ. As a result, each feature basis corresponds to $|\Sigma|$ features. Such way of modeling substitution induces a k-mer mutational neighborhood in $O(k^m|\Sigma|^m)$ size. In contrast, the

spatial features sample the sequences at different resolutions and therefore performing string matching does not require neighborhood expansion; matching on a position with substitution is achieved by extending the current spectrum. Such way of modeling substitution opens the door for a matching algorithm with low complexity *i.e.* independent of the size of the alphabet, which in turns opens the door for fast large-scale semi-supervised learning, as we will see in Section 6.4. In the figure, we represent all common features between the original and the mutated strings with bold fonts and red (light) color.

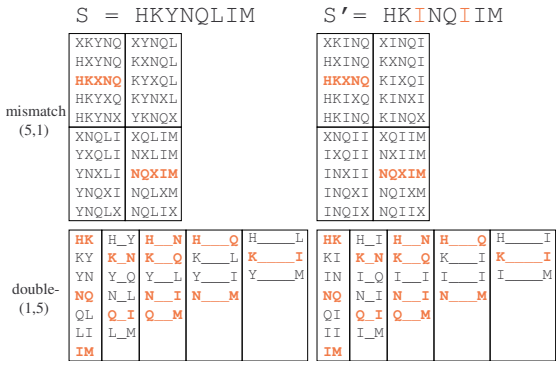


Figure 6.2: Differences in handling substitutions by the mismatch and spatial features. We represent all common features between the original and the mutated strings, S and S', with bold fonts and red (light) color. Each symbol 'X' under the mismatch representation represent an arbitrary symbol in the alphabet set Σ. As a result, each feature basis corresponds to $|\Sigma|$ features.

To compute the kernel values under the supervised setting, we first extract the features and sort the extracted features in linear time using counting sort. Finally we count the number of distinct features and for each observed feature, we update the kernel matrix. For N sequences with the longest length n and u distinct features, computing the N-by-N kernel matrix takes linear $O(dnN + min(u, dn)N^2)$ time.

Under the semi-supervised setting, on the one hand, direct use of equation 6.4 for computation of the refined kernel values between sequences X and Y requires $|N(X)| \times |N(Y)|$ kernel evaluations (*i.e.* quadratic running time in the size of the sequence neighborhood); on the other hand, use of Equation 6.3 requires explicit representation of the sequences which can be problematic when the dimensionality of the feature space is high. As a result, performing such *smoothing* operation over the mismatch(5,1)

representation is computationally intensive for both methods due to first, the exponential length of the induced k-mer mutational neighborhood and second, the quadratic running time induced by equation 6.4.

Equation 6.3 lends a useful insight into the complexity of the smoothing operation. For any explicit representation $\Phi(X)$, its smoothed version can be computed in time linear in the size of the neighborhood $N(X)$, therefore the smoothed kernel can also be evaluated in time linear in the neighborhood size. As mentioned before, the smoothed representation under the mismatch features cannot be efficiently computed because of the exponential size of the induced k-mer neighborhood; however, for the double and triple feature sets the smoothed representations can be computed explicitly, if desired. In our experiments, we do not compute the explicit representation and instead use implicit computations over induced representations: for each neighborhood set $N(X)$, we first sort the features and then obtain counts for distinct features to evaluate the kernel. The low-dimensional feature space and efficient feature matching induced by the kernels ensure low complexity for kernel evaluation. Kuksa *et al.* provides a more detailed description of algorithm for spatial kernel evaluation under both supervised and semi-supervised settings in [37].

6.2.2 Extracting relevant information from the unlabeled sequence database

Remote homology detection problem is typically characterized by *few positive sequences* accompanied by a large number of *negative examples*. Experimentally labeling the sequences is costly, leading to the need to leverage *unlabeled data* to refine the decision boundary. In [69], Weston *et al.* leverage the unlabeled sequences to construct a *sequence neighborhood kernel* under the mismatch representation to refine the decision boundary. However, in most sequence databases, we have multi-domain protein sequences in abundance and thus, such multi-domain sequences can be similar to several unrelated single-domain sequence, as noted in [69]. Direct use of such long sequences may falsely establish similarities among unrelated sequences. Under semi-supervised learning setting, our goal is to recruit *neighbors*, or *homologues* of training and testing

sequences and use these intermediate neighbors to establish similarity between the remotely homologous proteins, which bear little to no similarity on the primary sequence level. As a result, the quality of the intermediate neighboring sequences is crucial for inferring labels of remote homologues. Sequences that are too long will contribute excessive features, while sequences that are too short often have missing features and hence induce very sparse representation, which in turn bias the averaged neighborhood representation. As a result, the performance of the classifiers will be compromised with direct use of these sequences. Weston *et al.* in [69] proposed to only capture neighboring sequences with maximal length of 250 as a remedy. However, such practice may not offer a direct and meaningful biological interpretation and may discard valuable information. In this study, we propose to extract only *statistically significant sequence regions*, reported by PSI-BLAST, from the unlabeled neighboring sequences. We summarize all competing methods in below:

- *unfiltered*: all neighboring sequences are recruited. This is to show how much excessive or missing features in neighboring sequences that are too long or too short compromise the performance of the classifiers.

- *extracting the most significant region*: for each recruited unlabeled neighboring sequence, we extract only the most *statistically significant sequence region* reported by PSI-BLAST; such sub-sequence is more likely to be biologically relevant to the query sequence.

- *filter out sequences that are too long or too short*: for each query sequence X, we remove any neighboring sequences Y if $T_Y > 2T_X$ or $T_Y < \frac{T_X}{2}$, where T_X is the length of sequence X. This method will alleviate the effect of the excessive and missing features induced by the unfiltered method.

- *maximal length of 250*: this is the method proposed by Weston *et al.* in their study.

To recruit neighbors of a sequence X, we query the unlabeled database using PSI-BLAST [1] with two iterations. We recruit all sequences with e-values less than or

equal to 0.05 as the neighboring sequences of X. To obtain only relevant information from a neighboring sequence, we extract from the unlabeled neighboring sequence the most significant region (lowest e-value) reported by PSI-BLAST. We illustrate the procedure in Figure 6.3. In the figure, given the query sequence, PSI-BLAST reports sequences (hits) containing substrings that exhibit statistically significant similarity with the query sequence. For each reported hit with e-value less than or equal to 0.05, we extract the most significant region and recruit the extracted sub-sequence to the neighboring set of the query sequence.

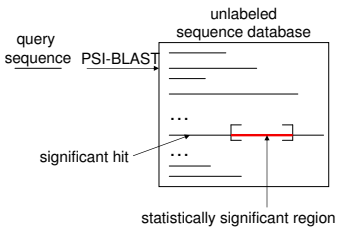


Figure 6.3: Extracting only statistically significant regions (red/light color, bold line) from the significant hit reported by PSI-BLAST

6.2.3 Clustering the neighboring sets

The smoothing operation in Equation 6.3 is susceptible to overly represented neighbors in the unlabeled data set since if we we append many replicated copies of a neighbor to the set, the computed average will be biased towards such sequence. In large uncurated sequence databases, duplicated and overly represented sequences are common. For example, some sequences in Swiss-Prot have the so-called *secondary accession numbers*. Such sequences can be easily identified and removed. However, there are two other types of duplication that are harder to find: sequences that are nearly identical and sequences that contain substrings that have high sequence similarity and are significant hits to the query sequence. Existence of such examples will bias the estimate of the averaged representation, hence compromising the performance of the classifiers. Pre-processing the data is necessary to remove such bias. In this study we propose to cluster the neighboring sets as a remedy. Conducting clustering analysis typically incurs quadratic complexity in the number of sequences to be clustered. As a result, though clustering the

union of all neighbor sets is more desirable, to minimize the experimental running time we propose to cluster each reported neighbor set *one at a time*; for example, the union of all neighbor sets (e-value less than or equal to 0.05) induced by the NR unlabeled database is $129,646$, while the average size of the neighbor sets is 115 (reported in later sections). Clustering each reported neighbor set individually will lead to tremendous saving in experimental running time.

We use the program *CDHit* [44] for clustering analysis. The program employs a heuristic (incremental clustering algorithm) to avoid all-by-all comparisons. First, the sequences are sorted in decreasing length with the longest one representing the clustering center. Next, each remaining sequences is compared to each existing clustering center and will be assigned to the first cluster in which the similarity between the cluster representative and the query sequence exceeds a pre-defined threshold. If no such cluster exists, the sequence will form a new cluster. In this study we perform clustering at 70% sequence identity level.

6.3 Results on Previously Published Methods

For completeness of this study, we present results on previously published methods in this section and we will present the experimental results of our proposed method in the next section. We present experimental results for protein remote homology detection under the semi-supervised setting on the SCOP 1.59 [47] data set, published in [69]. The data set contains 54 target families with $7,329$ isolated domains. Only $2,862$ domains out of $7,329$ are labeled, leading to $4,467$ unlabeled sequences and allowing to perform experiments in both supervised (labeled sequences only) and semi-supervised (labeled and unlabeled sequences) settings. Different instances of this data set have been used as a gold standard for protein remote homology detection in various studies.

We evaluate all methods using the *Receiver Operating Characteristic* (ROC) and ROC50 [25] scores. The ROC50 score is the (normalized) area under the ROC curve computed for up to 50 false positives. With a small number of positive testing sequences and a large number of negative testing sequences, the ROC50 score is typically more

indicative of the prediction accuracy of a homology detection method than the ROC score.

In all experiments, all kernel values $K(X, Y)$ are normalized using

$$K'(X, Y) = \frac{K(X, Y)}{\sqrt{K(X, X)K(Y, Y)}} \tag{6.6}$$

to remove the dependency between the kernel value and the sequence length. We use the sequence neighborhood kernel in Equation 6.4, as in [69], under the spatial sample representation. To perform our experiments, we use an existing SVM implementation from a standard machine learning package SPIDER[2] with default parameters.

6.3.1 The SCOP data set

In the supervised setting, only the labeled sequences participate in the experiments. In the semi-supervised experiments, we also use the unlabeled data set consisting of the remaining unlabeled sequences (4467 sequences).

Supervised learning

We compare the performance of our proposed methods with previously published state-of-the-art methods under the supervised setting. We also include the performance of generative models (profile HMMs) and the sufficient statistics features that we previously proposed to facilitate a more complete study. We summarize the performance in Table 6.1 and the method ξ-SVM-20m corresponds to use of sufficient statistics as features without dimensionality reduction and with SVM as the classifier. We also note that the feature extractor for the sufficient statistics make use of the profile HMMs, which are also a competing method in the comparison and are estimated using positive training sequences only. From the table we observe that the sufficient statistics and triple-(1,3) are comparable and have better performance over all other methods. Interestingly, contrary to common findings in previous studies, the profile HMMs demonstrate reasonably good performance under the supervised setting and also outperform

[2]http://www.kyb.tuebingen.mpg.de/bs/people/spider

the mismatch(5,1) kernel. We believe that such discrepancy comes from the fact that we used many optimization tools to generalize these profiles (high-quality hand-curated multiple alignment from PFAM [5] to estimate the initial profiles, position-based sequence weighting [27], and 9-component mixture of Dirichlet priors [10, 60]) while in other studies, only very simple and primitive training was performed. We also show the ROC-50 plot in Figure 6.4(a). In the plot, the horizontal axis corresponds to the ROC-50 scores and the vertical axis denotes the number of experiments, out of 54, with an equivalent or higher ROC-50 score. For clarity, we do not display the plot for every method. We observe that both model-based methods (profile HMMs and sufficient statistics) show better performance in the area of high ROC scores while in the area of low ROC scores, the methods based on directly extracting features from the observed strings (mismatch, double and triple) show better performance. We provide the following explanation: for a test sequence to obtain high scores in a model-based method, a good *global* alignment is required. In the case of profile HMM, if a remote homologue aligns well *globally* with the model, the score of the sequence will be high; on the other hand, using sufficient statistics as features and computing the similarity between two sequences with their inner product implies *global* sequence comparison between the two examples. As the sequences in the superfamily diverge, it becomes harder to capture similarity by globally aligning two sequences. As a result, while both model-based methods (profile HMM and sufficient statistics) show good discrimination power in the area of high ROC-50 scores, as the superfamilies diverge, the discrimination power starts to degrade in the area of low ROC-50 scores. We have also observe that among the three string kernels, both triple(1,3) and double(1,5) kernels dominate the mismatch(5,1) kernel.

Semi-supervised learning

We compare the performance on the same data set in Table 6.2 and in Figure 6.4(b) under the semi-supervised setting using unlabeled sequences extracted from the SCOP 1.59 data set. All methods except profile HMMs use the kernel smoothing method in Equation 6.4. For profile HMMs, it is not clear how to smooth over the *test sequences*

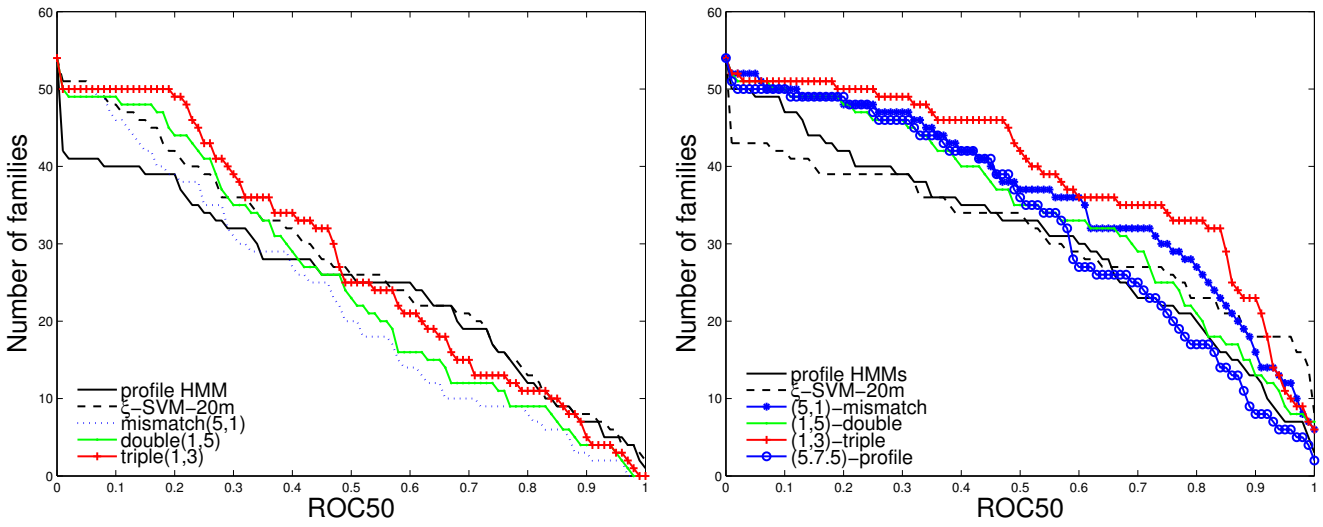
Table 6.1: Comparison of the performance on the SCOP 1.59 data set under the supervised setting.

Method	ROC	ROC50	# dim.
(5, 1)-mismatch	0.8749	0.4167	3200000
SVM-pairwise†	0.8930	0.4340	$N^{\ddagger}$
(1,5) double	0.8901	0.4629	2000
(1,3) triple	0.9148	0.5118	72000
Profile HMMs	0.8511	0.4597	-
ξ-SVM-20m	0.8864	0.5096	variable

†: directly quoted from [45]

‡: number of training sequences

while smoothing over the *training sequences* can be performed in a straightforward way by just appending the neighboring sequences into the training set. Furthermore, the negative training sequences do not participate in the estimation process. To build each profile, we run only 2 E-M iterations to ensure reasonable computational time. We show the ROC-50 plots in Figure 6.4 and summarize the performance in Table 6.2. From the plot we observe the model-based methods (profile HMMs and sufficient statistics) demonstrate inferior performance in the area of low ROC-50 scores. Note that for the profile kernel, the initial representation of each sequence is a profile (model-based) but the final representation in the k-mer space is induced by performing *local* alignment between two profiles *implicitly*. Comparison of profile kernel and the pure profile HMMs method as well as the kernel induced by sufficient statistics provides an interesting contrast. All three models rely on some underlying generative model, where in profile kernels the similarity value is induced by performing local alignment of substrings; in contrast, when using sufficient statistics as features, the features are obtained by aligning two sequences under some probabilistic framework *globally*. Profile kernel demonstrates inferior performance in the are of high ROC50 scores when compared with the other two model-based methods. However, in the area of high ROC50 scores, profile kernel shows better performance, indicating that inferring *remote* homologues of diverged superfamilies by *locally* aligning the sequences might be more effective than performing a *global* alignment.



(a) SCOP data set. Supervised setting.

(b) SCOP data set. Semi-supervised setting

Figure 6.4: Left panel: Comparison of the performance (ROC50) in the supervised setting. Right panel: Comparison of the performance (ROC50) in a semi-supervised setting using SCOP 1.59 as the unlabeled data set. Spatial triple kernel outperforms both profile and mismatch neighborhood kernels.

Table 6.2: Comparison of the performance under the semi-supervised setting with the unlabeled sequences extracted from SCOP1.59

Method	ROC	ROC50
(5, 1)-mismatch neighborhood	0.9093	0.6745
(5,7.5)-profile	0.9190	0.6069
(1,5)-double neighborhood	0.9282	0.6383
(1,3)-triple neighborhood	0.9382	0.7262
Profile HMMs	0.8985	0.5732
ξ-SVM-20m	0.8826	0.5804

6.4 Experimental Results on Our Proposed Methods

We present experimental results obtained by our proposed methods in this section on the spatial features and we no longer show the results for sufficient statistics and HMMs due to the discussed weakness in the previous chapter. Previously, we show that the class of spatial sample kernels achieve the state-of-the-art performance under the supervised setting and semi-supervised setting, where in the semi-supervised setting, the unlabeled data set comes from the SCOP 1.59 [47] sequence database itself. Note that sequences in the SCOP database are represented in single domains and therefore, use of such unlabeled data set does not raise any concern over extracting relevant

		double(1,5)			triple(1,3)		
	#neighbors	ROC	ROC50	p-value	ROC	ROC50	p-value
PDB							
unfiltered	14/5/311	.9333	.7324	.3498	.9393	.7444	3.46e-04
region	14/5/311	.9533	.7352	-	.9666	.8074	-
no tails	11/3/286	.9255	.6926	.0197	.9433	.7456	3.50e-03
by length	11/2/300	.9254	.6848	6.02e-02	.9418	.7127	4.53e-05
Swiss-Prot							
unfiltered	56/28/385	.9145	.6360	6.55e-04	.9245	.6908	2.46e-04
region	56/28/385	.9593	.7635	-	.9752	.8556	-
no tails	27/4/385	.9160	.6318	2.12e-04	.9361	.6938	1.55e-06
by length	21/3/385	.9070	.5652	2.03e-05	.9300	.6514	7.33e-07
NR							
unfiltered	115/86/490	.9319	.6758	1.40e-03	.9419	.7328	1.07e-05
region	115/86/490	.9715	.7932	-	.9824	.8861	-
no tails	55/13/399	.9463	.6775	4.40e-03	.9575	.7438	9.47e-06
by length	38/10/426	.9275	.6656	7.32e-04	.9513	.7401	2.66e-06

*p-value: signed-rank test on ROC50 scores against region
#neighbors: mean/median/max

Table 6.3: The overall prediction performance of all compared methods over various unlabeled data sets.

information from a multi-domain sequence. In this study, we use three larger unlabeled sequence databases, some of which contains abundant multi-domain protein sequences as well as duplicated or overly represented sequences. The three databases are PDB [7][3] (116,697 sequences), Swiss-Prot [8][4] (101,602 sequences), and the *non-redundant* (NR) sequence database (534,936 sequences). To adhere to the true semi-supervised setting, *all sequences in the unlabeled data sets that are identical to any test sequences are removed*.

We use the same evaluation methods (ROC and ROC50) outlined in the previous chapter, the same benchmark data set (SCOP 1.59), and the same standard machine learning package, SPIDER, to estimate SVMs with default parameters.

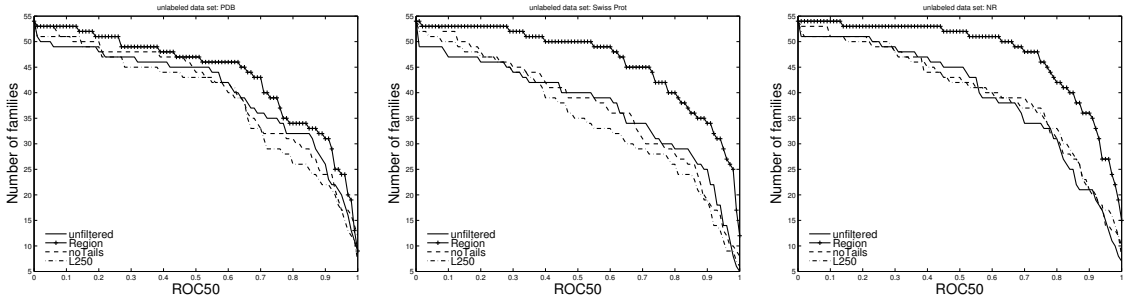
6.4.1 Experimental results without clustering

In Table 6.3, we show the performance in ROC and ROC50 scores for the four competing methods on the double(1,5) and triple(1,3) feature sets using 3 different unlabeled sequence data sets. We denote the method of filtering out sequences that exhibit a two-fold difference in length with the query sequence as *no tails* and the method of filtering out sequences whose length is greater than 250 as *by length*. In all but one case, extracting only relevant regions from the unlabeled sequence leads to significant improvement in the ROC and ROC50 scores compared to the unfiltered method. In the second column, we note the number of recruited neighbors (mean, median, and max). We also calculate the p-values of each competing method **against** the *region* method using Wilcoxon signed-rank test. In all cases except one, we observe statistically significant improvement in classification performance. Extracting significant regions for neighborhood smoothing improves the ROC and ROC50 scores on average by 0.0373 and 0.1048, respectively, when compared to the *unfiltered* method. We show the ROC50 plots of the four competing methods using the triple(1,3) feature set in Figure 6.5. In the figures, the horizontal axis corresponds to an ROC50 score and the vertical axis denotes the number of experiments, out of 54, with the corresponding or higher ROC50 score. In all cases, we observe that the ROC50 curves for region extraction show strong dominance over all other competing methods. Based on the table and figures, we also observe that filtering out neighboring sequences based on the length degrades the performance of the classifiers on the PDB (Figure 6.5(a)) and Swiss-Prot (Figure 6.5(b)) unlabeled sequence databases while in the case of using the NR data set (Figure 6.5(a)), the classifier shows slight improvement. Although filtering out sequences based on the length removes the unnecessary and noisy features from irrelevant regions within the sequences, at the same time, longer unlabeled sequences that carry critical information for inferring the class labels of the test sequences are also discarded. In a larger unlabeled data set (NR), such problem is alleviated since larger

[3] as of Dec. 2007

[4] We use the same version as the one used in [69] for comparative analysis of performance

databases are more likely to contain short sequences carrying such critical information.



(a) PDB (b) Swiss-Prot (c) NR

Figure 6.5: The ROC50 plots of four competing methods using the triple-(1,3) feature set with PDB, Swiss-Prot and NR databases as unlabeled data sets, respectively. The ROC50 curves of the method that only extracts relevant regions from the neighboring sequences consistently show strong dominance over all competing methods.

6.4.2 Experimental results with clustering

In Table 6.4, we present the performance in ROC and ROC50 scores for the four competing methods on the double(1,5) and triple(1,3) feature sets using 3 different unlabeled data sets. All smoothed representations are induced by the reduced neighbor sets. In contrast to Table 6.3, extracting relevant regions from neighboring sequences and performing clustering on the neighbor sets significantly improve performance on **all** unlabeled data sets. With clustering, extracting regions improves the ROC and ROC50 scores on average by 0.0245 and 0.0994, respectively, when compared to the *unfiltered* method. We again observe performance degradation when filtering out neighboring sequences based on their lengths. In the second column, we show the number of neighbors (mean, median, and maximum) after clustering. In most cases, we observe a 2-3 fold reduction in the number of neighbors contrasting to the neighborhood size reported in Table 6.3. We note that the reduction in the neighborhood size is critical for faster training and classification.

Finally we show the experimental running time in Table 6.5 under various settings, performed on a 3.6GHz CPU, based on the $2,862$ labeled sequences in the SCOP 1.59 data set. The average reduced running time for kernel evaluation is 1.40 seconds for the double(1,5) kernel and 38 seconds for the triple(1,3) kernel. The average reduction

		double(1,5)			triple(1,3)		
	#neighbors	ROC	ROC50	p-value	ROC	ROC50	p-value
PDB							
unfiltered	11/4/116	.9369	.7142	6.74e-02	.9439	.7585	4.70e-03
region	11/4/120	.9599	.7466	-	.9717	.8240	-
no tails	9/3/102	.9291	.6902	4.8e-03	.9490	.7545	2.30e-03
by length	7/2/104	.9229	.6589	1.10e-03	.9490	.7211	2.66e-05
Swiss-Prot							
unfiltered	30/17/223	.9526	.6397	3.76e-04	.9464	.7474	1.50e-03
region	27/15/210	.9582	.7701	-	.9732	.8605	-
no tails	15/3/192	.9214	.6446	1.95e-04	.9395	.7160	2.30e-06
by length	10/2/107	.9100	.5841	1.21e-05	.9348	.6817	7.33e-07
NR							
unfiltered	77/55/344	.9403	.6874	5.62e-04	.9556	.7566	2.20e-05
region	67/47/339	.9734	.8048	-	.9861	.8944	-
no tails	37/10/310	.9452	.6815	2.90e-04	.9602	.7486	2.06e-07
by length	24/8/263	.9313	.6686	1.00e-03	.9528	.7595	2.56e-07

*p-value: signed-rank test on ROC50 scores against region
#neighbors: mean/median/max

Table 6.4: The overall prediction performance of all compared methods over various unlabeled data sets **with clustering** the neighbor sets. All neighbor sets are clustered on a 70% sequence identity level and representatives of each cluster are chosen to form a reduced neighbor set.

in running time for kernel evaluation is 10.32% for the double(1,5) kernel and 10.66% for the triple(1,3) kernel. Clustering takes very little CPU time; for example, clustering the neighbor sets induced by the NR sequence database on all $2,862$ labeled sequences takes 126.24 in total on the region-based method. We want to note the large fold change in running time by adding one spatial sample ($t = 3$ for triple in contrast to $t = 2$ in double). Increasing the number of spatial samples by 1 implies multiplying the complexity for string matching by d, the maximum number of distance allowed between two samples. After clustering, the reduction in experimental running time will be significant for other tasks that require more spatial samples (increasing t), larger distance between each spatial samples (increasing d), larger sequence database (increasing N) or longer sequences (increasing n) since the complexity for feature matching exhibit multiplicative dependency on these parameters; performing feature matching incurs $O(d^{t-1}HNn)$ and $O(k^{m+1}|\Sigma|^m HnN)$ complexity for spatial and mismatch(k,m) kernels, respectively, where H denotes the size of the sequence neighborhood. Performing

	double(1,5)		triple(1,3)	
	without clustering	with clustering	without clustering	with clustering
PDB				
unfiltered	10.70	10.19	170.45	161.01
region	10.22	9.98	99.57	95.05
no tails	10.17	9.94	103.39	104.49
by length	9.97	9.85	73.85	73.36
Swiss-Prot				
unfiltered	16.14	12.84	802.27	719.62
region	12.74	11.17	289.03	240.15
no tails	11.61	10.52	186.06	160.84
by length	10.64	10.01	107.62	94.28
NR				
unfiltered	23.26	18.60	1451.52	1345.89
region	15.69	13.54	630.95	531.07
no tails	13.69	12.32	383.80	348.52
by length	11.66	10.74	245.23	213.02

Table 6.5: The experimental running time (seconds) for constructing the (2862-by-2862) kernel matrices on each unlabeled data set under different settings. The experiments are performed on a 3.6GHz CPU.

clustering reduces the neighborhood size by two fold on average, which in turn implies less computational resources for storage: under the discriminative kernel learning setting, we need to save the support vectors along with their corresponding neighbor sets. The savings in experimental time for kernel evaluation will be even more pronounced if the previously described parameters are increased simultaneously. For a detailed analysis of computational complexity, please refer to [37].

6.4.3 Comparison with other state-of-the-art methods

We compare the performance of our proposed method with previously published state-of-the-art methods over various unlabeled sequence databases and present the overall prediction performance of all compared methods in Table 6.6. For spatial kernels, all reported scores are based on extracting the most significant region and performing clustering on the neighbor sets. We perform all experiments on a $3.6GHz$ machine with 2GB of memory. Computation of the mismatch neighborhood kernels is computationally demanding and typically cannot be accomplished on a single machine for anything but

PDB	ROC	ROC50
double-(1,5) neighborhood	.9599	.7466
triple-(1,3) neighborhood	**.9717**	**.8240**
profile(5,7.5)	.9511	.7205
Swiss-Prot		
double-(1,5) neighborhood	.9582	.7701
triple-(1,3) neighborhood	**.9732**	**.8605**
profile(5,7.5)	.9709	.7914
mismatch nbhd†	.955	.810
NR		
double-(1,5) neighborhood	.9720	.8076
triple-(1,3) neighborhood	**.9861**	**.8944**
profile(5,7.5)-2 iterations	.9734	.8151
profile(5,7.5)-5 iterations‡	.984	.874
profile(5,7.5)-5 iter. with secondary structure‡	.989	.883

†:directly quoted from [69] ;‡:directly quoted from [36]

Table 6.6: The overall prediction performance of all compared methods over various unlabeled data sets. For spatial kernels, all reported scores are based on extracting the most significant region and performing clustering on the neighbor sets.

relatively small unlabeled data sets. Therefore, the results for the mismatch neighborhood kernel can only be shown using the previously published summary statistics [69] on Swiss-prot, a moderately populated sequence database. For each unlabeled data set, we highlight the best ROC and ROC50 scores; on all unlabeled data sets, the triple(1,3) neighborhood kernel achieves the best performance. Furthermore, we achieve such performance by only 2 PSI-BLAST iterations. For example, the triple(1,3) neighborhood kernel with 2 PSI-BLAST iterations outperforms the profile(5,7.5) kernel with 5 PSI-BLAST iterations. Moreover, the triple(1,3) neighborhood kernel with 2 PSI-BLAST iterations on the PDB unlabeled data set already outperforms the profile(5,7.5) kernel with 2 PSI-BLAST iterations on the NR unlabeled data set. We also note that the performance of our kernels is achieved using primary sequence information only. However, as shown in the table, the triple(1,3) kernel still outperforms the profile(5,7.5) kernel with added secondary structure information. Such higher order information (*e.g.* secondary structure), if available and desirable, can be easily included in the feature set. In this study, we do not pursue such direction.

We also show the statistical significance of the observed differences between pairs of

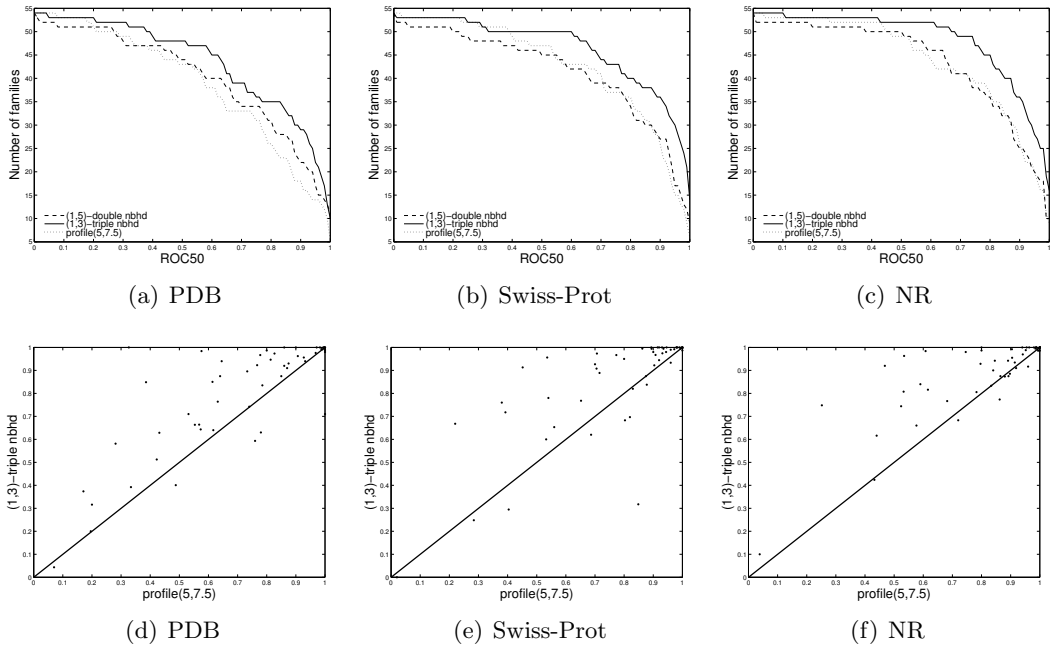
methods on various unlabeled data sets in Table 6.7. All the entries in the table are the p-values of the Wilcoxon signed-rank test using the ROC50 scores. For each unlabeled data set, we highlight the method that has the best overall performance. The triple(1,3) kernel consistently outperform all other kernels, with high statistical significance.

SCOP 1.59				
	mismatch	profile	double	triple
mismatch	-	2.245e-03	1.804e-02	3.570e-06
profile	2.245e-03	-	2.874e-01	9.615e-09
double	1.804e-02	2.874e-01	-	6.712e-06
triple	3.570e-06	9.615e-09	6.712e-06	-
PDB				
	double	triple	profile	
double	-	1.017e-01	4.762e-02	
triple	1.017e-01	-	7.666e-06	
profile	4.762e-02	7.666e-06	-	
Swiss-Prot				
	double	triple	profile	
double	-	9.242e-05	4.992e-01	
triple	9.242e-05	-	2.419e-04	
profile	4.992e-01	2.419e-04	-	
NR				
	double	triple	profile	
double	-	8.782e-06	9.762e-01	
triple	8.782e-06	-	7.017e-06	
profile	9.762e-01	7.017e-06	-	

Table 6.7: Statistical significance (p-values of the Wilcoxon signed-rank test) of the observed differences between pairs of methods (ROC50 scores) on unlabeled data sets. Triple denotes the triple-(1,3) neighborhood kernel, double denotes the double-(1,5) neighborhood kernel, mismatch denotes the mismatch(5,1) neighborhood kernel, and profile denotes the profile(5,7.5) kernel.

Next, in the upper panel of Figure 6.6, we show the ROC50 plots of the double(1,5) neighborhood, triple(1,3) neighborhood and profile(5,7.5) kernels using PDB (first column), Swiss-Prot (second column) and NR (third column) sequence databases as the unlabeled data sets. The ROC50 curves of the triple(1,3) neighborhood kernel on all unlabeled data sets consistently show strong dominance over those of other two kernels. Furthermore, the performance of the double(1,5) neighborhood kernel is on par with that of the profile(5,7.5) kernel. In the lower panel, we show the scatter plots of the ROC50 scores of the triple(1,3) kernel and the profile(5,7.5) kernel. Any point falling

above the diagonal line in the figures indicates better performance of the triple(1,3) kernel over the profile(5,7.5) kernel. As can be seen from these plots, the triple kernel outperforms the profile kernel on all three data sets (43/37/34 wins and 4/5/10 ties, out of 54 experiments, on PDB, Swiss-Prot, and NR data sets, respectively).



(a) PDB (b) Swiss-Prot (c) NR

(d) PDB (e) Swiss-Prot (f) NR

Figure 6.6: In the upper panel, we show the ROC50 plots of three different features using PDB, Swiss-Prot and NR databases as unlabeled data sets, respectively. In the lower panel, we show the scatter-plot of ROC50 scores of the triple-(1,3) kernel (vertical) and the profile(5,7.5) kernel (horizontal). Any point above the diagonal line in the figures (d),(e),(f) indicates better performance for the triple-(1,3) kernel.

Finally, in Table 6.8, we show the experimental running time for constructing the kernel matrix, based on all available sequences in the SCOP 1.59 data set. The size of the kernel matrix is 7329-by-7329. For the semi-supervised setting (neighborhood kernels), we report average running time on the data sets used (*i.e.* PDB, Swiss-Prot, and non-redundant (NR) sequence databases). As mentioned in previous sections, both mismatch and profile kernels require higher complexity to perform feature matching due to the exponential size of the mutational neighborhood, which in turns depend on the size of the alphabet set, whereas the complexity of performing feature matching for the spatial features is independent of the alphabet set size. This complexity difference leads to order-of-magnitude improvements in the running times of the spatial sample kernels

over the mismatch and profile kernels. The difference is even more pronounced when kernel smoothing is used under a semi-supervised setting. The neighborhood mismatch kernel becomes substantially more expensive to compute for large unlabeled data sets as indicated in [36, 69] by the authors.

Method	Running time (s)
supervised methods	
Triple(1,3) kernel	112
Double(1,5) kernel	54
Mismatch(5,1) kernel	948
semi-supervised methods	
Triple(1,3) neighborhood kernel	327
Double(1,5) neighborhood kernel	67
Mismatch(5,1) neighborhood kernel	-
Profile(5,7.5) kernel	10 hours†

†: the running time reported in [36]

Table 6.8: Experimental running time of all methods based on all sequences in the SCOP 1.59 data set. The size of the kernel is 7329-by-7329. For triple and double kernels, under the semi-supervised setting, the reported running time are based on extracting relevant regions and performing clustering on neighboring sets.

6.5 Discussion

We first illustrate the benefit of extracting only statistically significant regions from the neighboring sequences from a machine learning perspective and then we discuss the biological motivation of the spatial feature sets. The spatial features allow alphabet-free matching and model substitution, insertion and deletion effectively. The combination of both methods leads to fast and accurate semi-supervised protein remote homology detection. In the end, we perform complexity comparison and show better sensitivity of the spatial kernel with other state-of-the-art methods.

6.5.1 Motivation for extracting relevant regions

To illustrate the benefit of extracting only statistically significant regions from an unlabeled sequence, consider the example in Figure 6.7. In the figure, colors indicate membership: yellow (shaded) corresponds to the positive class and green (pattern) corresponds to the negative class. Sequences that demonstrate statistically significant

similarity are more likely to be evolutionarily related and therefore to belong to the same superfamily. The goal is to infer membership of the test (unshaded) sequences via the unlabeled sequence (in the middle). In the figure, arcs indicate (possibly weak) similarity induced by shared features, denoted by the black boxes, and absence of arcs indicates no similarity. As can be seen from the figure, the positive training and test sequences share no features and therefore have no similarity; however, the unlabeled sequence shares some features with both sequences in the reported region, which are very likely to be biologically relevant to both positive training and test sequences and therefore establishes the similarity between them. On the other hand, if the whole unlabeled sequence is recruited as a neighbor without discarding irrelevant regions, the similarity between the positive training and negative testing sequences will be incorrectly established, hence compromising the performance of the classifiers.

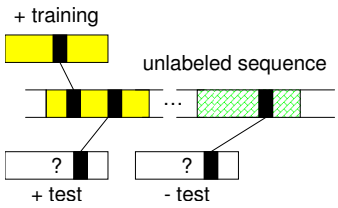


Figure 6.7: The importance of only extracting relevant region from neighboring sequences (in the middle): in the figure, the colors indicate the membership: yellow (shaded) indicates membership of the positive class and green (pattern) indicates membership of the negative class. The goal is to infer the label of the test (unshaded) sequences via the intermediate neighboring sequences. The arcs in the figure indicate (possibly weak) similarity and absence of arcs indicates no similarity. The black boxes in the sequence correspond to the shared features.

6.5.2 Biological Motivation of the spatial feature sets

Compared to mismatch/profile kernels, the feature sets induced by our kernels cover segments of variable length (*e.g.* 2-6 and 3-7 residues in the case of the double(1,5) kernel and the triple(1,3) kernels, respectively), whereas the mismatch and profile kernels cover segments of fixed length (*e.g.* 5 or 6 residues long) as illustrated in Figure 6.1. Sampling at different resolutions also allows to capture similarity in the presence of more complex substitution, insertion and deletion processes, while sampling at a fixed resolution, the approach used in mismatch and spectrum kernels, limits the sensitivity

in the case of multiple insertions/deletions or substitutions. We illustrate the benefit of multi-resolution sampling in Figure 6.8. In the figure, we show six slightly diverged sequences with the presence of both mutation and insertion. We also show the double(1,5) and mismatch(5,1) kernel matrices. We observe that the spatial kernel still captures substantial amount of similarities whereas the mismatch kernel, which performs fixed-resolution sampling captures little similarities among the related sequences. Both images are shown on the same scales. Increasing the parameter m (number of mismatches allowed) to accommodate multiple substitutions, in the case of mismatch kernels, leads to an exponential growth in the size of the k-mer mutational neighborhood, and results in high computational complexity. On the other hand, increasing the threshold σ in the profile kernel also incurs an exponential growth in the size of mutational neighborhood since in a highly diverged region the profile may be flat.

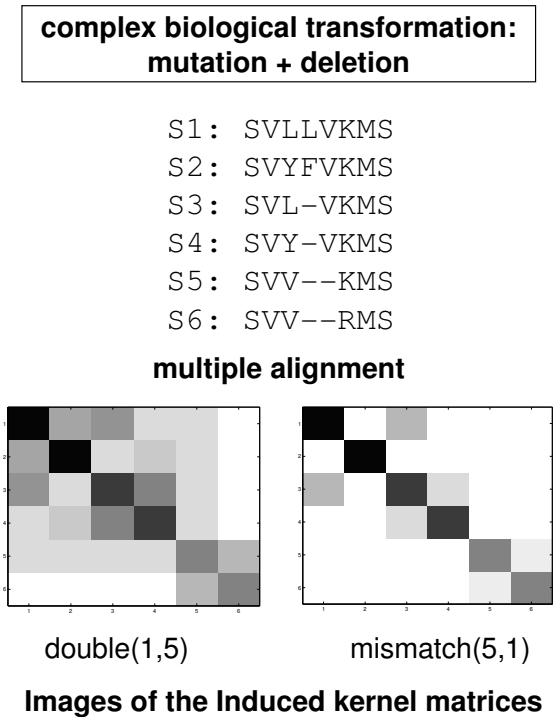


Figure 6.8: The benefit of multi-resolution sampling: in the presence of both mutations and insertions, the spatial kernel still captures substantial amount of similarities in such moderately conserved region; on the other hand, the mismatch kernel, which performs fixed-resolution sampling captures little similarity among related sequences.

6.5.3 Complexity comparison

Both mismatch and profile kernels have higher complexity compared to the spatial sample kernels due to the exponential k-mer neighborhood size and high dimensionality of the feature space. The cardinalities of the k-mer neighborhood induced by the mismatch and profile features are $O(k^m|\Sigma|^m)$ and $O(M_\sigma)$, with $k^m|\Sigma|^m \leq M_\sigma \leq |\Sigma|^k$, where $k \leq 5$ and $|\Sigma| = 20$, compared to a much smaller feature space size of $d^{t-1}|\Sigma|^t$ for the sample kernels, where t is 2 or 3 and d is 3 or 5, respectively. This complexity difference leads to order-of-magnitude improvements in the running times of the sample kernels over the mismatch and profile kernels. To compute an N-by$N$ matrix, the running time for the triple(1,d) kernel is $O(d^2nM + d^2|\Sigma|^3N^2)$, for double(1,$d$) it is $O(dnN + d|\Sigma|^2N^2)$, for mismatch(k,m) it is $O(k^{m+1}|\Sigma|^m nN + |\Sigma|^k N^2)$ and profile(k,σ) it is $O(kM_\sigma nN + |\Sigma|^k N^2)$.

Finally, in previous studies [69, 36], to achieve good accuracy, the number of PSI-BLAST iterations needs to be at least 5, while our performance is achieved with only 2 iterations. Each additional PSI-BLAST iteration requires substantial amount of computational resources.

6.5.4 Kernel-induced data manifolds

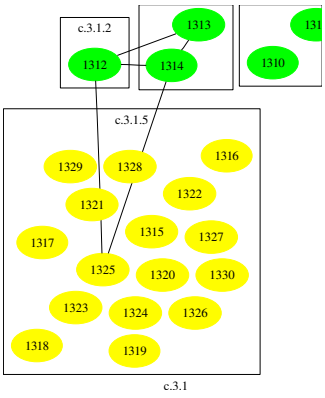
To further illustrate the sensitivities of different kernels, we compare the data manifolds induced by the spatial (triple(1,3)) and spectrum (mismatch(5,1) and profile(5,7.5)) kernels in both supervised and semi-supervised settings in Figure 6.9[5]. The figures show the data manifold for the *FAD/NAD(P)-binding domain*. The green (dark) nodes represent the training sequences and the yellow (light) nodes represent the testing sequences. Each cluster (box) represents a family within the superfamily. We normalize the kernel as discussed in Section 6.4 to remove the dependencies between the kernel values and sequence length. We draw an edge between two sequences X and Y if $K(X, Y) > \delta$, where δ is chosen so that the number of the falsely detected sequences (among negative training and negative testing sequences) is controlled at 10% of the total number of

[5]for visualization, we use the *fdp* package in *Graphviz*: graphviz.org

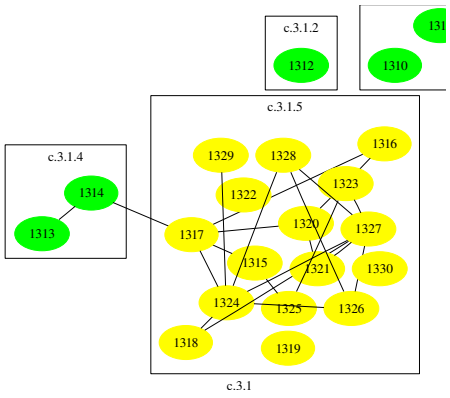
negative sequences. In the upper panel, we show the triple(1,3) (Figure 6.9(a)) and mismatch(5,1) (Figure 6.9(b)) kernel-induced manifolds under the supervised setting. We observe that in the manifold induced by the sparse spatial sample kernel the held-out family has already been connected to two other training families whereas in the manifold induced by the mismatch kernel, only one such connection exists. In the lower panel, we show the manifolds induced by the triple(1,3) (Figure 6.9(c)) and profile(5,7.5) (Figure 6.9(d)) kernels under the semi-supervised setting by leveraging the unlabeled sequences in the non-redundant data set. We note that the two manifolds are similar while the computational cost for constructing the chosen spatial sample kernel (triple) is much lower than that of the chosen spectrum (profile) kernel. Experimental computational time for evaluating the triple(1-3) neighborhood kernel on the non-redundant set takes 112 seconds and evaluating the profile(5,7.5) kernel takes around 10 hours. We further note that Table 6.6 indicates that the triple(1-3) neighborhood kernel outperforms the profile(5,7.5) kernel.

6.6 Conclusion

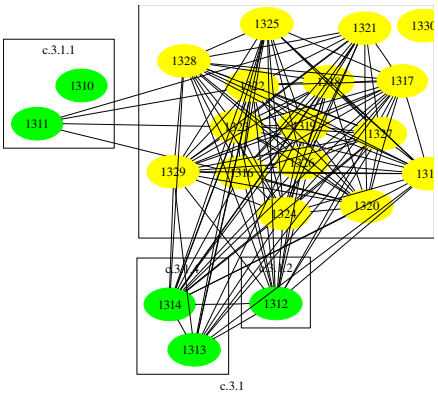
In this study, we propose a systematic and biologically motivated approach for extracting relevant information from unlabeled sequence database under the semi-supervised learning setting. We also propose to perform clustering on each neighbor sets to remove the bias caused by duplicated or overly represented neighboring sequences which are commonly found in large uncurated sequence databases. Combing these approaches with the sparse spatial sample kernels we achieve fast and accurate semi-supervised protein homology detection on three large unlabeled sequence databases. The spatial kernels induce low-dimensional feature space, effectively model mutation, insertion, and deletion with multi-resolution sampling and incur low computational complexity for kernel evaluation; its running time on string matching is independent of the size of the alphabet set, making rapid kernel evaluation possible on large sequence databases. The resulting classifiers based on our proposed methods significantly outperform previously published state-of-the-art methods in performance accuracy and exhibit order-of-magnitude differences in experimental running time.



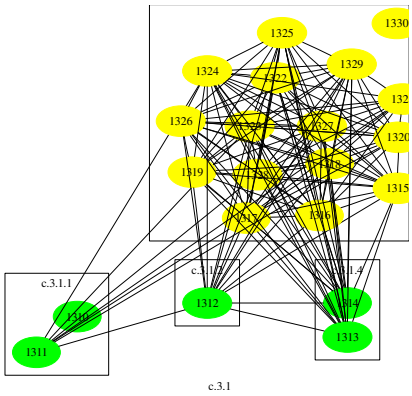
(a) triple(1,3), supervised



(b) mismatch(5,1), supervised



(c) triple(1,3), semi-supervised on nr



(d) profile(5,7.5), semi-supervised on nr

Figure 6.9: Kernel-induced data manifold for the *FAD/NAD(P)-binding domain* superfamily (*C*.3.1) with 4 families under the supervised and semi-supervised settings for spatial (triple(1,3)) and spectrum-like (mismatch(1,5) and profile(5,7.5)) kernels on the SCOP 1.59 data set. The green (darker) and yellow (lighter) nodes are the training and testing sequences, respectively. The numbers in the nodes index the sequences.

6.7 Acknowledgments

This work was carried out by working jointly with Pavel Kuksa.

Chapter 7

Conclusion

In this study, we present three major works for protein remote homology detection. We start from a *sparse generative* model with the hypothesis that a *small set* of *key positions* with the corresponding preferred residues and the distances between each neighboring pair of key positions are sufficient to discriminate between members and non-members of a superfamily. Our automated learning procedure captures the critical positions and the distances and estimates a collection of *sparse profile HMMs*, which encode the captured information into a set of probabilistic patterns. We show that a four-fold compression of model complexity can be achieved while maintaining performance. We also observe that as the sequences in the superfamily diverge, conservation in some critical positions becomes relaxed. Remotely homologous (positive) sequences *do not fully* conform to the estimated patterns and *skip* some of these critical positions; on the other hand, some unrelated (negative) sequences also conform partially to the imposed probabilistic patterns. The observations motivate the need to employ *discriminative models* for the protein remote homology task since *generative models* focus on capturing the commonly shared characteristics within the group and cannot guarantee that such characteristics will be absent in other unrelated groups.

Next, we construct *discriminative* models that rely on both *positive* and *negative* sequences for estimation, which results in richer sets of sequences for the training process. In the study we present a *hybrid* model. First, we estimate a generative model (profile HMMs) based on the *positive sequences only*. In the second stage, we use the generative model as feature extractors to construct a set of *biologically meaningful* features (sufficient statistics with respect to the generative model) from the provided sequences for *discriminative learning*. Combining the features with a class of *sparsity enforcing*

priors under the *Bayesian learning paradigm*, we obtain a class of *simple* and *intuitive* models that provide meaningful biological interpretation. The features and *sparse* classifiers recover the *key positions* with the corresponding preferred residues. We test the prediction accuracy of the models on a widely used benchmark protein sequence data set and show that under the *supervised* setting, not only do our *hybrid* models achieve performance comparable to that of the state-of-the-art methods, but they also offer *simple*, *intuitive*, and *biologically meaningful* interpretation.

Estimating the similarity between two sequences using sufficient statistics as features under the framework in our second study implies matching two sequences *globally*. As a result, while the *hybrid* models demonstrate better performance in the area of high ROC-50 scores, which corresponds to *moderately conserved* superfamilies, the models only achieve moderate discriminative power with diverged superfamilies in the area of low ROC50 scores, motivating the need to use an approach based on local alignments of sequences. Also, a missing component in previous studies for large-scale semi-supervised protein homology detection is a systematic and biologically motivated approach for leveraging the unlabeled data set. In our final study, we address both issues. First, we employ a class of previously established kernels, the Sparse Spatial Sample Kernels (SSSK). This class of biologically motivated kernels model mutation, insertion and deletion effectively and induce low-dimensional feature space; moreover, the computational complexity of kernel evaluation based on *local* feature matching is independent of the size of the alphabet set and such key characteristics opens the door for rapid large-scale semi-supervised learning. Second, we propose a biologically meaningful way of extracting relevant information from the unlabeled database for semi-supervised learning. Third, we also propose a method to remove the bias caused by overly represented or duplicated unlabeled sequences which are commonly seen in uncurated sequence databases. Our experimental results show that the combination of these approaches yields state-of-the-art performance that are significantly better than previously published methods and also exhibit order-of-magnitude differences in experimental running time.

We also explore the possibility of taking advantage of the feature sharing and joint

training framework. Our preliminary experiments of using sufficient statistics as features indicates that decent performance can be achieved with very few features though such feature set does not permit feature sharing and results in a regular binary classification problem. On the other hand, use of the mismatch(5,1) feature set results in various computational difficulties such as exponential dimensionality and complex correlation structures among the features.

For future works, we propose to apply the algorithm on other tasks such as protein remote fold recognition and motif elucidation. We also propose to find sensible feature sets that permits the feature sharing and joint training framework for the possibility of achieving better biological interpretation. The works presented in this thesis can also be readily applied to other data with sequential nature, for example, document classification.

References

[1] S. Altschul et al. Gapped Blast and PSI-Blast: A new generation of protein database search programs. *NAR*, 25:3389–3402, 1997.

[2] Stephen F. Altschul and Warren Gish. Local alignment statistics. *Methods in Enzymology*, 266:460–480, 1996.

[3] Amos Bairoch, Rolf Apweiler, Cathy H. Wu, Winona C. Barker, Brigitte Boeckmann, Serenella Ferro, Elisabeth Gasteiger, Hongzhan Huang, Rodrigo Lopez, Michele Magrane, Maria J. Martin, Darren A. Natale, Claire O'Donovan, Nicole Redaschi, and Lai-Su L. Yeh. The Universal Protein Resource (UniProt). *Nucl. Acids Res.*, 33(suppl-1):D154–159, 2005.

[4] P. Baldi, Y. Chauvin, Y. Hunkapliier, and M. McClure. Hidden markov models of biological primary sequence information. In *Proceedings of the National Academy of Sciences*, volume 91, pages 1059–1063, 1994.

[5] Alex Bateman, Lachlan Coin, Richard Durbin, Robert D. Finn, Volker Hollich, Sam Griffiths-Jones, Ajay Khanna, Mhairi Marshall, Simon Moxon, Erik L. L. Sonnhammer, David J. Studholme, Corin Yeats, and Sean R. Eddy. The Pfam protein families database. *Nucleic Acids Research*, 32(Database-Issue):138–141, 2004.

[6] Dennis A. Benson, Ilene Karsch-Mizrachi, David J. Lipman, James Ostell, and David L. Wheeler. Genbank. *Nucl. Acids Res.*, 33(suppl-1):D34–38, 2005.

[7] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. The Protein Data Bank. *Nucleic Acids Research*, 28:235–242, 2000.

[8] B. Boeckmann, A. Bairoch, R. Apweiler, M.C. Blatter, A. Estreicher, E. Gasteiger, M.J. Martin, K. Michoud, C. O'Donovan, I. Phan, S. Pilbout, and M. Schneider. The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003. *Nucleic Acids Res*, 31:365–370, 2003.

[9] Steven E. Brenner, Patrice Koehl, and Michael Levitt. The ASTRAL compendium for protein structure and sequence analysis. *Nucl. Acids Res.*, 28(1):254–256, 2000.

[10] Michael Brown, Richard Hughey, Anders Krogh, I. Saira Mian, Kimmen Sjölander, and David Haussler. Using dirichlet mixture priors to derive hidden markov models for protein families. In *Proceedings of the 1st International Conference on Intelligent Systems for Molecular Biology*, pages 47–55. AAAI Press, 1993.

[11] C. Burge and S. Karlin. Prediction of complete gene structures in human genomic DNA. *J. Mol. Biol.*, 268:78–94, 1997.

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

[13] James Casbon and Mansoor A. S. Saqi. S4: structure-based sequence alignments of SCOP superfamilies. *Nucleic Acids Research*, 33:D219–D222, 2005.

[14] Ira Cohen, Fabio G. Cozman, Nicu Sebe, Marcelo C. Cirelo, and Thomas S. Huang. Semi-supervised learning of classifiers: Theory, algorithms for bayesian network classifiers and application to human-computer interaction.

[15] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, March 2000.

[16] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Jorunal of the Royal Statistical Society. Series B*, 1977(1):1–38, 1977.

[17] Thomas G. Dietterich and Ghulum Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

[18] Richard Durbin, Sean R. Eddy, Anders Krogh, and Graeme Mitchison. *Biological sequence analysis: Probabilistic models of proteins and nucleic acids*. Cambridge University Press, 1998.

[19] SR Eddy. Profile hidden Markov models. *Bioinformatics*, 14(9):755–763, 1998.

[20] Mário A. T. Figueiredo. Adaptive sparseness for supervised learning. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(9):1150–1159, 2003.

[21] Mário A. T. Figueiredo and Anil K. Jain. Bayesian Learning of Sparse Classifiers. In *CVPR (1)*, pages 35–41, 2001.

[22] Genkin, Alexander, Lewis, D. David, Madigan, and David. Large-scale bayesian logistic regression for text categorization, August 2007.

[23] Julian Gough, Kevin Karplus, Richard Hughey, and Cyrus Chothia. Assignment of Homology to Genome Sequences using a Library of Hidden Markov Models that Represent all Proteins of Known Structure. *J. Mol. Biol.*, 313(4):903–919, 2001.

[24] M. Gribskov, A.D. McLachlan, and D. Eisenberg. Profile analysis: detection of distantly related proteins. *Proceedings of the National Academy of Sciences*, 84:4355–4358, 1987.

[25] M. Gribskov and N. Robinson. Use of receiver operating characteristic (ROC) analysis to evaluate sequence matching, 1996.

[26] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2001.

[27] Steven Henikoff and Jorja G. Henikoff. Position-based sequence weights. *J Mol Biol.*, 243(4):574–8, 11 1994.

[28] T J Hubbard and J Park. Fold recognition and ab initio structure predictions using hidden markov models and beta-strand pair potentials. *Proteins*, 23:398–402, 1995.

[29] Nicolas Hulo, Amos Bairoch, Virginie Bulliard, Lorenzo Cerutti, Edouard De Castro, Petra S. Langendijk-Genevaux, Marco Pagni, and Christian J. A. Sigrist. The PROSITE database. *Nucl. Acids Res.*, 34:D227–230, 2006.

[30] Tommi Jaakkola, Mark Diekhans, and David Haussler. Using the Fisher kernel method to detect remote protein homologies. In *Proceedings of the Seventh International Conference on Intelligent Systems for Molecular Biology*, pages 149–158. AAAI Press, 1999.

[31] Tommi Jaakkola, Mark Diekhans, and David Haussler. A discriminative framework for detecting remote protein homologies. In *Journal of Computational Biology*, volume 7, pages 95–114, 2000.

[32] D.G. Higgins J.D. Thompson and T.J. Gibson. CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position specific gap penalties and weight matrix choice. . *Nucleic Acids Research.*, 22:4673–4680, 1994.

[33] Alexander E. Kister, Alexei V. Finkelstein, and Israel M. Gelfand. Common features in structures and sequences of sandwich-like proteins. *PNAS*, 99(22):14137–14141, 2002.

[34] Alexander E. Kister, Michael A Roytberg, Cyrus Chothia, Jurii M. Vasiliev, and Israel M. Gelfand. The sequence determinants of cadherin molecules. *Protein Sci*, 10(9):1801–1810, 2001.

[35] Rui Kuang, Eugene Ie, Ke Wang, Kai Wang, Mahira Siddiqi, Yoav Freund, and Christina Leslie. Profile-based string kernels for remote homology detection and motif extraction. In *CSB '04: Proceedings of the 2004 IEEE Computational Systems Bioinformatics Conference (CSB'04)*, pages 152–160, August 2004. http://www.cs.columbia.edu/compbio/profile-kernel.

[36] Rui Kuang, Eugene Ie, Ke Wang, Kai Wang, Mahira Siddiqi, Yoav Freund, and Christina Leslie. Profile-based string kernels for remote homology detection and motif extraction. *J Bioinform Comput Biol*, 3(3):527–550, June 2005.

[37] Pavel Kuksa, Pai-Hsi Huang, and Vladimir Pavlovic. Kernel methods and algorithms for general sequence analysis. Technical Report RU-DCS-TR630, Department of Computer Sciences, Rutgers University, 2008.

[38] Pavel Kuksa, Pai-Hsi Huang, and Vladimir Pavlovic. Spatially-constrained sample kernel for sequence classification. In *The Learning Workshop (SNOWBIRD)*, 2008.

[39] Roman A. Laskowski, Victor V. Chistyakov, and Janet M. Thornton. PDBsum more: new summaries and analyses of the known 3D structures of proteins and nucleic acids. *Nucl. Acids Res.*, 33:D266–268, 2005.

[40] Jerald F. Lawless. *Statistical Models And Methods For Lifetime Data*. Wiley, 1982.

[41] Christina Leslie and Rui Kuang. Fast string kernels using inexact matching for protein sequences. *J. Mach. Learn. Res.*, 5:1435–1455, 2004.

[42] Christina S. Leslie, Eleazar Eskin, and William Stafford Noble. The spectrum kernel: A string kernel for svm protein classification. In *Pacific Symposium on Biocomputing*, pages 566–575, 2002.

[43] Christina S. Leslie, Eleazar Eskin, Jason Weston, and William Stafford Noble. Mismatch string kernels for svm protein classification. In *NIPS*, pages 1417–1424, 2002.

[44] Weizhong Li and Adam Godzik. Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, 22(13):1658–1659, 2006.

[45] Li Liao and William Stafford Noble. Combining pairwise sequence similarity and support vector machines for remote protein homology detection. In *RECOMB*, pages 225–232, 2002.

[46] Thomas Lingner and Peter Meinicke. Remote homology detection based on oligomer distances. *Bioinformatics*, 22(18):2224–2231, 2006.

[47] L. Lo Conte, B. Ailey, T.J. Hubbard, S.E. Brenner, A.G. Murzin, and C. Chothia. SCOP: a structural classification of proteins database. *Nucleic Acids Res.*, 28:257–259, 2000.

[48] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K. Muller. Fisher discriminant analysis with kernels, 1999.

[49] Tom M. Mitchell. *Generative and Discriminative Classifiers: Naive Bayes and Logistic Regression*, chapter 1, pages 1–17. 2005.

[50] A. Ng and M. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes, 2002.

[51] Kamal Nigam, Andrew K. McCallum, Sebastian Thrun, and Tom M. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, 39(2/3):103–134, 2000.

[52] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proc. the IEEE*, 77(2):257–286, February 1989.

[53] L. R. Rabiner. A tutorial on hidden Markov models and selected apllications in speech recognition. In A. Waibel and K.-F. Lee, editors, *Readings in Speech Recognition*, pages 267–296. Kaufmann, San Mateo, CA, 1990.

[54] R. Raina, Y. Shen, A. Ng, and A. McCallum. Classification with hybrid generative /discriminative models, 2003.

[55] B. Reva, A. Kister, S. Topiol, and I. Gelfand. Determining the roles of different chain fragments in recognition of immunoglobulin fold. *Protein Eng.*, 15(1):13–19, 2002.

[56] Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *J. Mach. Learn. Res.*, 5:101–141, 2004.

[57] Teemu Roos, Hannes Wettig, Peter Grünwald, Petri Myllymäki, and Henry Tirri. On discriminative bayesian network classifiers and logistic regression. *Mach. Learn.*, 59(3):267–296, 2005.

[58] R. Schapire. The boosting approach to machine learning: An overview, 2001.

[59] Bernhard Schölkopf, Alexander J. Smola, and Klaus-Robert Müller. Kernel principal component analysis. *Advances in kernel methods: support vector learning*, pages 327–352, 1999.

[60] Kimmen Sjolander, Kevin Karplus, Michael Brown, Richard Hughey, Anders Krogh, I. S Mian, and David Haussler. Dirichlet mixtures: A method for improving detection of weak but significant protein sequence homology. Technical report, Santa Cruz, CA, USA, 1996.

[61] Martin Szummer and Tommi Jaakkola. Partially labeled classification with markov random walks. In *Advances in Neural Information Processing Systems*, volume 14, 2001.

[62] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, and D.J. Lipman. Basic Local Alignment Search Tool. *Journal of Molecular Biology*, pages 403–410, 1990.

[63] W.R. Pearson and D.J. Lipman. Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences of the United States of America*, 85:2444–2448, 1988.

[64] Antonio Torralba, Kevin P. Murphy, and William T. Freeman. Sharing visual features for multiclass and multiview object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(5):854–869, 2007.

[65] Antonio B. Torralba, Kevin P. Murphy, and William T. Freeman. Sharing features: Efficient boosting procedures for multiclass object detection. In *CVPR (2)*, pages 762–769, 2004.

[66] Koji Tsuda, Taishin Kin, and Kiyoshi Asai. Marginalized kernels for biological sequences. *Bioinformatics*, 18(suppl_1):S268–275, 2002.

[67] Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, September 1998.

[68] J. Weston and C. Watkins. Support vector machines for multiclass pattern recognition. In *Proceedings of the Seventh European Symposium On Artificial Neural Networks*, 4 1999.

[69] Jason Weston, Christina Leslie, Eugene Ie, Dengyong Zhou, Andre Elisseeff, and William Stafford Noble. Semi-supervised protein classification using cluster kernels. *Bioinformatics*, 21(15):3241–3247, 2005.

[70] J. Zhu and T. Hastie. Kernel logistic regression and the import vector machine, 2001.

[71] Xiaojin Zhu. Semi-supervised learning literature survey. Technical Report 1530, Computer Sciences, University of Wisconsin-Madison, 2005. http://www.cs.wisc.edu/∼jerryzhu/pub/ssl_survey.pdf.

Vita

Pai-Hsi Huang

EDUCATION

October 2008 Ph.D. in Computer Science, Rutgers University, U.S.A.

May 2005 M.S. in Statistics, Rutgers University, U.S.A.

January 2004 M.S. in Computer Science, Rutgers University, U.S.A.

June 2001 B.S. in Computer Science, Drexel University, U.S.A.

EXPERIENCE

Jun.2007—Aug.2007 Biomarker Project Assistant, Oncology BioMarker Group, Novartis Pharmaceutical, East Hanover, NJ

Sep.2005—Jun.2008 Teaching Assistant, Department of Computer Science, Rutgers University, New Brunswick, NJ

Jun.2005—Aug.2005 Graduate Associate, Rosetta Inpharmatics (Merck), Seattle, WA

Sep.2004—May.2005 Research Assistant, Department of Computer Science, Rutgers University, New Brunswick, NJ

Sep.2001—May.2004 Teaching Assistant, Department of Computer Science, Rutgers University, New Brunswick, NJ

PUBLICATION

Large-Scale Region-based Neighborhood Method for Semi-supervised Protein Sequence Classification. Pavel Kuksa, Pai-Hsi Huang and Vladimir Pavlovic. (submitted for review)

Scalable High-Performance Sequence Classification. Pavel Kuksa, Pai-Hsi Huang and Vladimir Pavlovic. (submitted for review)

Scalable Algorithms for String Kernels with Inexact Matching. Pavel Kuksa, Pai-Hsi Huang and Vladimir Pavlovic. (submitted for review)

Fast Protein Homology and Fold Detection with Sparse Spatial Sample Kernels. Pavel Kuksa, Pai-Hsi Huang and Vladimir Pavlovic. Nineteenth International Conference on Pattern Recognition (ICPR) 2008, Tampa, Florida, U.S.A.

A Fast, Large-Scale Learning Method for Protein Sequence Classification. Pavel Kuksa, Pai-Hsi Huang and Vladimir Pavlovic. Eighth International Workshop on Data Mining in Bioinformatics (BIOKDD) 2008, Las Vegas, NV, U.S.A.

Fast and Accurate Multi-Class Protein Fold Recognition with Spatial Sample Kernels. Pavel Kuksa, Pai-Hsi Huang and Vladimir Pavlovic. Seventh Annual International Conference on Computational Systems Bioinformatics (CSB) 2008, Stanford, CA, U.S.A.

Spatially-constrained Sample Kernel for Sequence Classification. Pavel Kuksa, Pai-Hsi Huang, and Vladimir Pavlovic. Snowbird Learning Workshop, 2008, Snowbird, UT, U.S.A.

Protein Homology Detection with Biologically Inspired Features and Interpretable Statistical Models. Pai-Hsi Huang and Vladimir Pavlovic. International Journal of Data Mining in Bioinformatics. Accepted for submission.

Sparse Logistic Classifiers for Interpretable Protein Homology Detection. Pai-Hsi Huang and Vladimir Pavlovic. IEEE International Conference on Data Mining (ICDM) 2006, Hong Kong, under International Workshop on Data Mining in Bioinformatics.

Inexpensive d-Dimensional Matching. Ljubomir Perkovic, Eric Schmutz and Bae-Shi Huang. Random Structures and Algorithms, Vol 20, No. 1, 2002, 50-58.