Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 EECS Building
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 06-013

Building Multiclass Classifiers for Remote Homology Detection and Fold Recognition

Huzefa Rangwala and George Karypis

April 05, 2006

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
05 APR 2006		**00-00-2006 to 00-00-2006**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Building Multiclass Classifiers for Remote Homology Detection and Fold Recognition	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Department of Computer Science and Engineering,University of Minnesota,200 Union Street SE,Minneapolis,MN,55455-0159**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**14**	

Building Multiclass Classifiers for Remote Homology Detection and Fold Recognition

Huzefa Rangwala and George Karypis

Department of Computer Science & Engineering, University of Minnesota,Minneapolis, MN 55455

rangwala@cs.umn.edu, karypis@cs.umn.edu

Abstract

Motivation *Protein remote homology prediction and fold recognition are central problems in computational biology. Supervised learning algorithms based on support vector machines are currently one of the most effective methods for solving these problem. These methods are primarily used to solve binary classification problems and they have not been extensively used to solve the more general multiclass remote homology prediction and fold recognition problems.*

Methods *We developed a number of methods for building SVM-based multiclass classification schemes in the context of the SCOP protein classification. These methods includes schemes that directly build an SVM-based multiclass model, schemes that employ a second level learning approach to combine the predictions generated by a set of binary SVM-based classifiers, and schemes that build and combine binary classifiers for various levels of the SCOP hierarchy beyond those defining the target classes.*

Results *We performed a comprehensive study analyzing the different approaches using four different datasets. Our results show that most of the proposed multiclass SVM-based classification approaches are quite effective in solving the remote homology prediction and fold recognition problems and that the schemes that use predictions from binary models constructed for ancestral categories within the SCOP hierarchy tend to qualitatively improve the prediction results.*

Website: *http://bioinfo.cs.umn.edu/supplements/mc-fold/*

Keywords: *fold recognition, remote homology, multiclass, hierarchical, structured learning, support vector machines.*

1 Introduction

Breakthroughs in large-scale sequencing have led to a surge in the available protein sequence information that has far outstripped our ability to experimentally characterize their functions. As a result, researchers are increasingly relying on computational techniques to classify proteins into functional and structural families based solely on their primary amino acid sequences. While satisfactory methods exist to detect homologs with high levels of similarity, accurately detecting homologs at low levels of sequence similarity (remote homology detection) still remains a challenging problem.

Over the years several methods have been developed to address the problems of remote homology prediction and fold recognition. These includes methods based on pairwise sequence comparisons [30, 3, 28, 36], on generative models [21, 4], and on discriminative classifiers [18, 25, 23, 24, 15, 16, 35, 22, 31].

Recent advances in string kernels that have been specifically designed for protein sequences and capture their evolutionary relationships [22, 31] have resulted in the development of support vector machines-based (SVMs) [41] discriminative classifiers that show superior performance when compared to the other methods [31].

These SVM-based approaches were designed to solve *one-versus-rest* binary classification problems and to this date, they are primarily evaluated with respect to how well each binary classifier can identify the proteins that belong to its own class (e.g., superfamily or fold). However, from a biologist's perspective, the problem that he or she is facing (and will like to solve) is that of identifying the most likely superfamily or fold (or a short list of candidates) that a particular protein belongs to. This is essentially a multiclass classification problem, in which given a set of K classes, we will like to assign a protein sequence to one of them.

Even though highly accurate SVM-based binary classifiers can go a long way in addressing some of the biologist's requirements, it is still unknown how to best combine the predictions of a set of SVM-based binary classifiers to solve the multiclass classification problem and assign a protein sequence to a particular superfamily or fold. Moreover, it is not clear, if schemes that combine binary classifiers are inherently better suited for solving the remote homology prediction and fold recognition problems over schemes that directly build an SVM-based multiclass classification model.

This problem was recently recognized by Ie *et al*. [17] and developed schemes for combining the outputs of a set of binary SVM-based classifiers for primarily solving the remote homology prediction problem. Specifically borrowing ideas from error-correcting output codes [10, 2, 8], they developed schemes that use a separate learning step to learn how to best scale the outputs of the binary classifiers such that when combined with a scheme that assigns a protein to the class whose corresponding scaled binary SVM prediction is the highest, it achieves the best multiclass prediction performance. In

addition, for remote homology prediction in the context of the SCOP [27] hierarchical classification scheme, they also studied the extent to which the use of such hierarchical information can further improve the performance of remote homology prediction. Their experiments showed that these approaches lead to better results than the traditional schemes that use either the maximum functional output [32] or those based on fitting a sigmoid function [37].

In this paper, motivated by the positive results of Ie *et al*'s. work, we further study the problem of building SVM-based multiclass classification models for remote homology prediction and fold recognition in the context of the SCOP protein classification scheme. We present a comprehensive study of different approaches for building such classifiers including (i) schemes that directly build an SVM-based multiclass model, (ii) schemes that employ a second level learner to combine the predictions generated by a set of binary SVM-based classifiers, and (iii) schemes that build and combine binary classifiers for various levels of the SCOP hierarchy. In addition, we present and study three different approaches for combining the outputs of the binary classifiers that lead to hypothesis spaces of different complexity and expressive power.

These schemes are thoroughly evaluated for both remote homology prediction and fold recognition using four different datasets derived from Astral [5]. Our experimental results show that most of the proposed multiclass SVM-based classification approaches are quite effective in solving the remote homology prediction and fold recognition problems. Among them, schemes employing a two-level learning framework are in general superior to those based on the direct SVM-based multiclass classifiers, even though the performance achieved by the later schemes is quite respectable. Our results also show that the multiclass classifiers that use predictions from binary models constructed for ancestral categories within the SCOP hierarchy tend to qualitatively improve the prediction results.

2 Methods

2.1 K-way Classification Problem

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

2.2 Direct SVM-based K-way Classifier Solution

One way of solving the K-way classification problem using support vector machines is to use one of the many multiclass formulations for SVMs that were developed over the years [11, 12, 42, 1, 9]. These algorithms extend the notions of separating hyperplanes and margins and learn a model that directly separates the different classes.

In this study we evaluate the effectiveness of one of these formulations that was developed by Crammer and Singer [9], which leads to reasonably efficient optimization problems.

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

$$y^* = \underset{i=1}{\overset{K}{\operatorname{argmax}}} \ \{ \langle W_i, x \rangle \}, \tag{1}$$

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

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

W itself is learned from the training data following a maximum margin with soft constraints formulation that gives rise to the following optimization problem [9]:

$$\begin{array}{ll} \min & \frac{1}{2}\beta W^2 + \sum_{i=1}^{m} \xi_i, \\ \text{subject to:} & \forall i, z \ \ \langle W_{y_i}, x_i \rangle + \delta_{y_i,z} - \langle W_z, x_i \rangle \geq 1 - \xi_i \end{array} \tag{2}$$

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

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

2.3 Merging K One-vs-Rest Binary Classifiers

An alternate way of solving the K-way classification problem in the context of SVM is to first build a set of K one-versus-rest binary classification models $\{f_1, f_2, \ldots, f_K\}$, use all of them to predict an instance x, and then based on the predictions of these base classifiers $\{f_1(x), f_2(x), \ldots, f_K(x)\}$ assign x to one of the K classes [10, 2, 37].

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

$$y^* = \underset{i=1}{\overset{K}{\operatorname{argmax}}} \ \{ f_i(x) \}. \tag{3}$$

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

2.3.2 Cascaded SVM-Learning Approaches A promising approach that has been explored in combining the

outputs of K binary classification models is to formulate it as a cascaded learning problem in which a second level model is trained on the outputs of the binary classifiers to correctly solve the multiclass classification problem [17, 10, 2].

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

$$y^* = \operatorname*{argmax}_{i=1}^{K} \{ w_i f_i(x) \}, \quad (4)$$

where w_i is a factor used to scale the functional output of the i^{th} classifier, and the set of K w_i scaling factors make up the model that is being learned during the second level training phase [17]. We will refer to this scheme as the *scaling* scheme (S).

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

$$y^* = \operatorname*{argmax}_{i=1}^{K} \{ w_i f_i(x) + s_i \}. \quad (5)$$

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

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

$$y^* = \operatorname*{argmax}_{i=1}^{K} \{ \langle W_i, f(x) \rangle \}, \quad (6)$$

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

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

2.4 Use of Hierarchical Information

One of the key characteristics of remote homology prediction and fold recognition is that the target classes are naturally organized in a hierarchical fashion. This hierarchical organization is evident in the tree-structured organization of the various known protein structures that is produced by the widely used protein structure classification schemes of SCOP [27], CATH [29] and FSSP [14].

In our study we use the SCOP classification database to define the remote homology prediction and fold recognition problems. SCOP organizes the proteins into four primary levels (class, fold, superfamily, and family) based on structure and sequence similarity. Within the SCOP classification, the problem of remote homology prediction corresponds to that of predicting the superfamily of a particular protein under the constraint that the protein is not similar to any of its descendant families, whereas the problem of fold recognition corresponds to that of predicting the fold (i.e., second level of hierarchy) under the constraint that the protein is not similar to any of its descendant superfamilies.[1]

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

The approach investigated in this study is primarily motivated by the different schemes presented in Section 2.3.2 to combine the functional outputs of multiple one-versus-rest binary classifiers. A general way of doing this is to learn a binary one-versus-rest model for each or a subset of the nodes of the hierarchical classification scheme, and then combine these models using an approach similar to the CS-scheme described in Section 2.2.

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

$$y^* = \operatorname*{argmax}_{i=1}^{K} \{ \langle W_i, f(x) \rangle \}, \quad (7)$$

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

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

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

[1] These two constraints are important because if they are violated, then we are actually solving either the family or remote homology prediction problems, respectively

and moving upwards towards the root of the classification hierarchy uniquely identifies a set of nodes corresponding to higher level classes containing the target class. For example, if the target class is a superfamily, this path will identify the superfamily itself, its corresponding fold, and its corresponding class in the SCOP hierarchy.

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

$$y^* = \underset{i=1}{\overset{K_f}{\operatorname{argmax}}}\{w_i f_i(x) + w_{\bigwedge_i^f} f_{\bigwedge_i^f}(x) + w_{\bigwedge_i^c} f_{\bigwedge_i^c}(x)\}, \quad (8)$$

where $w_i, w_{\bigwedge_i^f}$ and $w_{\bigwedge_i^c}$ are scaling factors learned during training of the second level model.

Note that the underlying model in Equation 8 is essentially an extension of the scaling model of Equation 4 as it linearly combines the predictions of the binary classifiers of the ancestor nodes.

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

The generic problem of classifying within the context of a hierarchical classification system has recently been studied by the machine learning community and a number of alternative approaches have been developed [40, 38, 34].

2.5 Structured Output Spaces

The various models introduced in Sections 2.3.2 and 2.4 can be expressed using a unified framework that was recently introduced for learning in structured output spaces [40, 6, 7, 39].

This framework [40] learns a discriminant function $F : \mathcal{X} \times \mathcal{Y} \rightarrow \mathcal{R}$ over input/output pairs from which it derives predictions by maximizing F over the response variable for a specific given input x. Hence, the general form of the hypothesis h is

$$h(x;\theta) = \underset{y \in \mathcal{Y}}{\operatorname{argmax}} \left\{ F(x,y;\theta) \right\}, \quad (9)$$

where θ denotes a parameter vector. Function F is a θ-parameterized family of functions that is designed such that $F(x,y;\theta)$ achieves the maximum value for the correct output y. Among the various choices for F, if we focus on those that are linear in a combined feature representation of inputs and outputs, $\psi(x,y)$, then Equation 9 can be rewritten as [40]:

$$h(x;\theta) = \underset{y \in \mathcal{Y}}{\operatorname{argmax}} \left\{ \langle \theta, \Psi(x,y) \rangle \right\}. \quad (10)$$

The specific form of Ψ depends on the nature of the problem and it is this flexibility that allows us to represent the hypothesis spaces introduced in Sections 2.3.2 and 2.4 in terms of Equation 10.

For example, consider the simple scaling scheme for the problem of fold recognition (Equation 4). The input space consists of the $f(x)$ vectors of the binary predictions and the output space $\mathcal{Y}$ consists of the set of K_f folds (labeled from $1 \ldots K_f$). Given an example x belonging to fold i (i.e., $y = i$), the function $\Psi(x,y)$ maps the (x,y) pair onto a K_f-size vector whose ith entry (i.e., the entry corresponding to x's fold) is set to $f_i(x)$ and the remaining entries are set to zero. Then, from Equation 10 we have that

$$h(x;\theta) = \underset{i=1}{\overset{K_f}{\operatorname{argmax}}} \left\{ \langle \theta, \Psi(x,i) \rangle \right\} = \underset{i=1}{\overset{K_f}{\operatorname{argmax}}} \left\{ \theta_i f_i(x) \right\}, \quad (11)$$

which is similar to Equation 4 with θ representing the scaling vector w.

Similarly, for the scale & shift approach (Equation 5), the $\Psi(x,y)$ function maps the (x,y) pair onto a feature space of size $2K_f$, where the first K_f dimensions are used to encode the scaling factors and the second K_f dimensions are used to encode the shift factors. Specifically, given an example x belonging to fold i, $\Psi(x,y)$ maps (x,y) onto the vector whose ith entry is $f_i(x)$, it's $(2i)^{th}$ entry is one, and the remaining entries are set to zero. Then, from Equation 10 we have that

$$\begin{aligned} h(x;\theta) &= \underset{i=1}{\overset{K_f}{\operatorname{argmax}}} \left\{ \langle \theta, \Psi(x,i) \rangle \right\} \\ &= \underset{i=1}{\overset{K_f}{\operatorname{argmax}}} \left\{ \theta_i f_i(x) + \theta_{2i} \right\}, \end{aligned} \quad (12)$$

which is equivalent to Equation 5, with the first half of θ corresponding the scale vector w, and the second half corresponding to the shift vector s.

Finally, in the case of the Cramer-Singer approach, the $\Psi(x,y)$ function maps (x,y) onto a feature space of size $K_f \times K_f$. Specifically, given a sequence x belonging to fold i, $\Psi(x,y)$ maps (x,y) onto the vector whose K_f entries starting at $(i-1)K_f$ are set to $f(x)$ (i.e., the fold prediction outputs) and the remaining $(K_f - 1)K_f$ entries are set to zero. Then, by rewriting Equation 10 in terms of the above combined input-output representation, we get

$$\begin{aligned} h(x;\theta) &= \underset{i=1}{\overset{K_f}{\operatorname{argmax}}} \left\{ \langle \theta, \Psi(x,i) \rangle \right\} \\ &= \underset{i=1}{\overset{K_f}{\operatorname{argmax}}} \left\{ \sum_{j=1}^{K_f} \theta_{(i-1)K_f + j} f_j(x) \right\}. \end{aligned} \quad (13)$$

This is equivalent to Equation 6, as θ can be viewed as the matrix W with K_f rows and K_f columns.

Algorithm 1 Learning Weight Vectors with the ranking perceptron algorithm

Input: m: Number of Training Samples.
(x, y): Training Samples.
β: User constant to control separation constraints.
α: Learning rate.
Output: θ: Weight Vector.

1: $\theta \leftarrow 0$
2: **while** STOPPING CRITERION = false **do**
3: **for** $i = 1$ to m **do**
4: $y_i^* = \text{argmax}_{y \in \mathcal{Y}} \langle \theta, \Psi(x_i, y) \rangle$
5: **if** $y_i^* = y_i$ **then**
6: $y_i^* = \text{argmax}_{y \in \mathcal{Y}/y_i} \langle \theta, \Psi(x_i, y) \rangle$
7: **end if**
8: **if** $\langle \theta, \Psi(x_i, y_i) \rangle - \langle \theta, \Psi(x_i, y_i^*) \rangle \leq \beta \|\theta\|_2$ **then**
9: $\theta \leftarrow \theta + \alpha \Psi(x_i, y_i)$
10: $\theta \leftarrow \theta - \alpha \Psi(x_i, y_i^*)$
11: **end if**
12: **end for**
13: **end while**
14: Return θ

2.5.1 Ranking Perceptron.

One way of learning θ in Equation 10, is to use the recently developed extension to Rosenblatt's linear perceptron classifier [33], called *ranking perceptron* [6]. This is an online learning algorithm that iteratively updates θ for each training example that is misclassified according to Equation 10. For each misclassified example x_i, θ is updated by adding to it a multiple of $(\Psi(x_i, y_i) - \Psi(x_i, y_i^*))$, where y_i^* is given from Equation 10 (i.e., the erroneously predicted class for x_i). This online learning framework is identical to that used in standard perceptron learning and is known to converge when the examples are linearly separable. However this convergence property does not hold when the examples are not linearly separable.

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

$$\forall i \;\; \langle \theta, \Psi(x_i, y_i) \rangle - \langle \theta, \Psi(x_i, y_i^*) \rangle \geq \beta \|\theta\|_2, \tag{14}$$

where y_i is x_i's true class and $y_i^* = \text{argmax}_{y \in \mathcal{Y}/y_i} \{\langle \theta, \Psi(x_i, y) \rangle\}$. The idea behind these constraints is to force the algorithm to learn a model in which the correct predictions are well-separated from the highest scoring incorrect predictions (i.e., those corresponding to y_i^*). The degree of acceptable separation, which corresponds to the required margin, is given by $\beta \|\theta\|_2$, where β is a user-specified constant. Note, the margin is expressed in terms of θ's length to ensure that the separation constraints are invariant to simple scaling transformations.

Algorithm 1 shows our extended ranking perceptron algorithm that uses the constraints of Equation 14 to guide its online learning. The key steps in this algorithm are lines 8–10 that update θ based on the satisfaction/violation of the constraints for each one of the m training instances. Since the ranking perceptron algorithm is not guaranteed to converge when the examples are not linearly separable, Algorithm 1 incorporates an explicit *stopping criterion* that after each iteration it computes the training error-rate of θ, and terminates when θ's error rate has not improved in 100 consecutive iterations. The algorithm returns the θ that achieved the lowest training error rate over all iterations.

2.5.2 SVM-Struct.

Recently, an efficient way of learning the vector θ of Equation 10 has been formulated as a convex optimization problem [40]. In this approach θ is learned subject to the following m nonlinear constraints

$$\forall i : \max_{y \in \mathcal{Y}/y_i} \{\langle \theta, \Psi(x_i, y) \rangle\} < \langle \theta, \Psi(x_i, y_i) \rangle. \tag{15}$$

Note, that these constraints are similar in nature to those used in the ranking perceptron algorithm (Equation 14).

The SVM-Struct [40] algorithm, is an efficient way of solving the above optimization problem in which the m nonlinear inequalities are replaced by $|\mathcal{Y}| - 1$ linear inequalities resulting in a total of $m(|\mathcal{Y}| - 1)$ linear constraints and θ is learned using the maximum-margin principle leading to the following hard-margin problem [40]:

$$\begin{aligned} \min_{\theta} \quad & \tfrac{1}{2} \|\theta\|_2^2 \\ \text{subject to} \quad & \langle \theta, \; \Psi(x_i, y_i) - \Psi(x_i, y) \rangle \geq 1 \\ & \forall i, \forall y \in \{\mathcal{Y}/y_i\}. \end{aligned} \tag{16}$$

This hard-margin problem can be converted to a soft-margin equivalent to allow errors in the training set. This is done by introducing a slack variable, ξ, for every nonlinear constraint of Equation 15. The soft-margin problem is expressed as [40]:

$$\begin{aligned} \min_{\theta, \xi} \quad & \tfrac{1}{2} \|\theta\|_2^2 + \tfrac{C}{n} \textstyle\sum_{i=1}^{n} \xi_i, \\ \text{subject to} \quad & \langle \theta, \; \Psi(x_i, y_i) - \Psi(x_i, y) \rangle \geq 1 - \xi_i \\ & \forall i, \xi_i \geq 0, \forall i, \forall y \in \{\mathcal{Y}/y_i\}. \end{aligned} \tag{17}$$

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

2.6 Loss Functions

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

However, in cases where the class sizes vary significantly across the different folds, such a zero-one loss function may not be the most appropriate as it may lead to models where

Table 1: Dataset Statistics.

Statistic	DS1	DS2	DS3	DS4
ASTRAL filtering	90%	40%	25%	40%
Number of Sequences	2115	1119	1294	1651
Number of Folds	25	25	25	27
Number of Superfamilies	47	37	137	158
Avg. Pairwise Similarity	12.8%	11.5%	11.6%	11.4
Avg. Max. Similarity	63.5%	33.9%	32.2%	34.3
Avg. Pairwise Similarity (within folds)	25.6%	17.9%	16.7%	17.4
Avg. Pairwise Similarity (outside folds)	10.4%	11.03%	11.2%	11.0

The percent similarity between two sequences is computed by aligning the pair of sequences using SW-GSM with a gap opening of 5.0 and gap extension of 1.0. "Avg. Pairwise Similarity" is the average of all the pairwise percent identities, "Avg. Max. Similarity" is the average of the maximum pairwise percent identity for each sequence i.e, it measures the similarity to its most similar sequence. The "Avg. Pairwise Similarity (within folds)" and "Avg. Pairwise Similarity (outside folds)" is the average of the average pairwise percent sequence similarity within the same fold and outside the fold for a given sequence.

the rare class instances are often mispredicted. For this reason, an alternate loss function is used, in which penalty for a misclassification is inversely proportional to the class size. This implies that the misclassification of examples belonging to smaller classes weigh higher in terms of the loss. This loss function is referred to as the balanced loss [17]. For the ranking perceptron algorithm (Algorithm 1) the update rules (statements 7 and 8) need to be scaled by the loss function. In case of the SVM-Struct formulation, the balanced loss can be optimized by reweighting the definition of separation which can be done indirectly by rescaling the slack variables ξ_i in the constraint inequalities (Equation 17).

While using the hierarchical information in the cascaded learning approaches (Section 2.4) we experimented with a *weighted* loss function where a larger penalty was assigned when the predicted label did not share the same ancestor compared to the case when the predicted and true class labels shared the same ancestors. This variation did not result in an improvement compared to the zero-one and balanced loss. Hence, we do not report results of using such hierarchical loss functions here.

3 Materials

3.1 Dataset Description

We evaluated the performance of the various schemes on four datasets. The first dataset, referred to as DS1, was created by Ie *et al*. [17] to evaluate the performance of the multiclass classification algorithms that they developed[2], whereas the other three datasets, referred to as DS2, DS3, and DS4, were created for this study[3]. The DS1 dataset was derived from SCOP 1.65, whereas DS2–DS4 were derived from SCOP 1.67. Table 1 summarizes the characteristics of these datasets and presents various sequence similarity statistics.

DS1 and DS2 are designed to evaluate the performance of remote homology prediction and were derived by taking only the domains with less than 95% and 40% pairwise sequence identity according to Astral [5], respectively. This set of domains was further reduced by keeping only the domains belonging to folds that (i) contained at least three superfamilies and (ii) one of these superfamilies contained multiple families. For DS1, the resulting dataset contained 2115 domains organized in 25 folds and 47 superfamilies, whereas for DS2, the resulting dataset contained 1119 domains organized in 25 folds and 37 superfamilies.

DS3 and DS4 were designed to evaluate the performance of fold recognition and were derived by taking only the domains with less than 25% and 40% pairwise sequence identity, respectively. This set of domains was further reduced by keeping only the domains belonging to folds that (i) contained at least three superfamilies and (ii) at least three of these superfamilies contained more than three domains. For DS3, the resulting dataset contained 1294 domains organized in 25 folds and 137 superfamilies, whereas for DS4, the resulting dataset contained 1651 domains organized in 27 folds and 158 superfamilies.

3.2 Binary Classifiers

The various one-versus-rest binary classifiers were constructed using SVMs. These classifiers used the recently developed [31] Smith-Waterman based profile kernel function (SW-PSSM), that has been shown to achieve the best reported results for remote homology prediction and fold recognition.

The SW-PSSM kernel computes a local alignment between two protein sequences, in which the similarity between two sequence positions is determined using a PICASSO like scoring function [13, 26], and a position independent affine gap modeling scheme. We use the optimized parameters for the affine gap model (i.e gap-opening (*go*) and gap-extension (*ge*) costs), and zero-shift (*zs*) for our base classifiers.

For our performance studies, we use the optimal parameter settings of $go = 3.0, ge = 0.75$ and $zs = 1.5$ four our kernel function to build our binary base classifiers using the widely used SVM^{light} [20] program.

3.3 Direct K-way Classifier

The direct K-way classification models were built using the publicly available implementation of the algorithm described in Section 2.2 from the authors [9].

To ensure that the schemes are compared fairly, we use the same SW-PSSM kernel function used by the binary SVM classifiers (Section 3.2). We tested the direct K-way classifiers using linear kernel functions as well, but the performance of the SW-PSSM kernels were substantially better.

3.4 Performance Assessment Measures

We assessed the performance of final classification using zero-one error rates (ZE), where every misclassification was penalized by one. We also evaluated our results using a balanced error rate (BE), which took into account the varying class size distributions. This class-sensitive error rate had a

[2]DS1 is available at http://www1.cs.columbia.edu/compbio/code-learning/

[3]DS2, DS3, and DS4 are available at http://bioinfo.cs.umn.edu/supplements/mc-fold/

lower penalty for misclassifying a test instance belonging to a larger class. In particular the error on each mistake is inversely proportional to the true class size.

3.5 Training Methodology

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

For DS1 and DS2 (DS3 and DS4), the test set is constructed by selecting from each superfamily (fold) all the sequences that are part of one family (superfamily). Thus during training, the dataset does not contain any sequences that are homologous (remote homologous) to the sequences in the test set and thus allows us to evaluate/assess remote homology prediction (fold recognition) performance.

This is a standard protocol for evaluating remote homology detection and fold recognition and has been used in a number of earlier studies [31, 35, 22, 19].

The cascaded models are trained as follows. We split the training data into 10 cross-validation sets, where we learn the binary models from the partitioned dataset and perform classification on the held out set to get prediction outputs. These prediction outputs serve as training samples for the second level learning using the ranking perceptron or the structured SVM algorithm. At the final stage, we compute the prediction for our untouched dataset evaluating the accuracies using zero-error and class size sensitive balanced error rates.

3.6 Model Selection

The performance of SVM depends on the parameter that controls the trade-off between the margin and the misclassification cost ("C" parameter in SVM-Struct), whereas the performance of ranking perceptron depends on the parameter β in Algorithm 1.

We perform a model selection or parameter selection step. To perform this exercise fairly, we split our test set into two equal halves of similar distributions, namely sets A and B. Using set A, we vary the controlling parameters and select the best performing model for set A. We use this selected model and compute the accuracy for set B. We repeat the above steps by switching the roles of A and B. The final accuracy results are the average of the two runs. While using the SVM-Struct program we let C take values from the set $\{0.0001, 0.001, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1.0, 2.0, 4.0, 8.0, 10.0, 16.0, 32.0, 64.0, 128.0\}$. While using the perceptron algorithm we let the margin β take values in the set $\{0.0001, 0.005, 0.001, 0.05, 0.01, 0.02, 0.5, 0.1, 1.0, 2.0, 5.0, 10.0\}$.

4 Results

4.1 Zero-One and Balanced Error Performance

The performance of various schemes in terms of zero-one and balanced error is summarized in Tables 2 and 3 for remote homology prediction and fold recognition, respectively. The schemes that are included in these tables are the following: (i) the MaxClassifier (Section 2.3.1), (ii) the direct K-way classifier (Section 2.2), (iii) the two-level learning approaches based on either the superfamily- or fold-level binary classifiers (Section 2.3.2), and (iv) the two-level learning approaches that also incorporate hierarchical information (Section 2.4).

For the direct K-way and two-level learning approaches these tables show the results obtained by optimizing both zero-one loss (ZL) and balanced loss (BL). Note that since the MaxClassifier relies solely on the outputs of the individual one-vs-rest binary classifiers, it does not explicitly optimize any particular loss function.

For all two-level learning approaches (with and without hierarchical information) these tables show the results obtained by using the scaling (S), scale & shift (SS), and Crammer-Singer (CS) schemes to construct the second-level classifiers.

4.1.1 Performance of Direct K-way Classifier. Comparing the direct K-way classifiers against the MaxClassifier approach we see that, in general, the error rates achieved by the direct approach are smaller for both the remote homology prediction and fold recognition problems. In many cases these improvements are substantial. For example, The BL-optimized direct K-way classifier achieves a 10.9% zero-one error rate for DS2 compared to a corresponding error rate of 21.0% achieved by MaxClassifier. The only exception is the DS3 dataset for which the MaxClassifier achieves slightly better results in terms of ZL than the direct classifier. In addition, unlike the common belief that learning SVM-based direct multiclass classifiers is computationally very expensive, we found that the Crammer-Singer formulation that we used to require time that is comparable to that required for building the various binary classifiers used by the MaxClassifier approach.

4.1.2 Non-Hierarchical Two-Level Learning Approaches. Analyzing the performance of the various two-level classifiers that do not use hierarchical information we see that the scaling (S) and scale & shift (SS) schemes achieve better error rates than those achieved by the Crammer-Singer (CS) scheme. The only exception is the DS3 dataset for which the ZL-based CS scheme achieves the best results.

Since the hypothesis space of the CS scheme is a superset of the hypothesis spaces of the S and SS schemes, we found this result to be surprising at first. However, in analyzing the characteristics of the models that were learned we noticed that the reason for this performance difference is the fact that the CS scheme tended to overfit the data. This was evident by the fact that the CS scheme had lower error rates on the training set than either the S or SS schemes (results not reported here). Since CS's linear model has more parameters than the other two schemes, due to the fact that the size of the training set for all three of them is the same and rather limited, such overfitting can easily occur. We believe that the CS scheme can potentially outperform the other two schemes for problems in which the training set is larger, and this is something

Table 2: Percentage Error for the remote homology detection problem.

	DS1		DS2	
	ZE	BE	ZE	BE
Simple Combination of Binary Outputs				
MaxClassifier	14.7	30.0	21.0	29.7
Direct K-way Classifiers				
ZL	13.5	24.8	20.5	26.5
BL	11.5	23.1	10.9	13.0
Two-Level Approaches				
Without Hierarchy Information				
Ranking Perceptron				
ZL, S	10.6	18.0	11.7	16.5
ZL, SS	13.2	24.5	10.9	13.4
ZL, CS	17.0	34.3	14.2	19.4
BL, S	9.3	16.1	10.9	13.9
BL, SS	10.1	19.5	12.1	15.8
BL, CS	14.7	28.9	17.6	24.1
SVM-Struct				
ZL, S	10.7	18.1	13.4	17.3
ZL, SS	12.4	23.7	13.4	17.3
ZL, CS	12.7	25.2	15.5	19.8
BL, S	9.0	15.9	11.8	15.7
BL, SS	10.7	19.9	12.1	15.1
BL, CS	11.6	19.4	13.0	16.3
With Hierarchy Information				
With Fold-level Nodes				
SVM-Struct				
ZL, S	10.4	18.7	14.7	20.0
ZL, SS	12.4	23.7	14.7	21.4
ZL, CS	13.8	25.0	14.7	19.6
BL, S	11.2	19.6	14.7	21.4
BL, SS	10.1	19.3	12.1	16.9
BL, CS	14.7	26.0	13.0	18.2
With Fold-level and Class-level Nodes				
SVM-Struct				
ZL, S	10.9	19.1	12.6	17.7
ZL, SS	11.2	20.9	13.4	17.8
ZL, CS	14.1	27.6	12.6	17.1
BL, S	11.2	20.2	13.0	18.8
BL, SS	13.5	24.7	12.1	16.8
BL, CS	14.7	26.1	13.0	17.5

ZE and BE denote the zero-one error and balanced error percent rates respectively. ZL and BL are the zero-one and balanced loss functions respectively. S, SS and CS denote the scaling, scale & shift and Crammer-Singer schemes respectively.

Table 3: Percentage Error for the fold recognition problem.

	DS3		DS4	
	ZE	BE	ZE	BE
Simple Combination of Binary Outputs				
MaxClassifier	42.0	60.3	44.4	64.6
Direct K-way Classifiers				
ZL	42.8	59.4	43.0	62.7
BL	38.4	52.3	40.4	56.9
Two-Level Approaches				
Without Hierarchy Information				
Ranking Perceptron				
ZL, S	39.9	52.9	32.2	50.6
ZL, SS	38.4	51.3	27.3	44.8
ZL, CS	34.8	48.9	37.7	56.6
BL, S	39.5	48.7	32.5	48.0
BL, SS	38.8	51.0	29.0	43.0
BL, CS	37.7	49.6	36.0	49.6
SVM-Struct				
ZL, S	41.3	55.2	33.7	50.0
ZL, SS	41.0	54.3	29.0	46.2
ZL, CS	36.6	49.4	32.5	49.6
BL, S	39.9	52.7	30.8	46.6
BL, SS	39.9	52.5	28.1	42.8
BL, CS	41.3	50.5	31.1	43.3
With Hierarchy Information				
With Class-level Nodes				
SVM-Struct				
ZL, S	39.9	52.2	31.9	50.2
ZL, SS	38.4	52.9	29.3	44.6
ZL, CS	39.2	51.8	32.8	52.9
BL, S	39.2	52.4	29.9	45.0
BL, SS	38.1	51.6	29.0	41.7
BL, CS	41.7	50.9	29.9	41.7
With Superfamily-level Nodes				
SVM-Struct				
ZL, S	39.5	53.9	31.3	48.8
ZL, SS	39.9	53.4	31.3	48.4
ZL, CS	37.7	52.1	33.4	51.0
BL, S	40.2	52.6	30.5	44.5
BL, SS	40.6	52.7	29.3	42.8
BL, CS	38.8	48.8	31.0	44.9
With Superfamily-level and Class-level Nodes				
SVM-Struct				
ZL, S	39.2	52.2	27.3	41.0
ZL, SS	39.9	53.9	28.4	44.1
ZL, CS	38.8	54.7	31.3	48.0
BL, S	41.0	50.9	33.7	44.6
BL, SS	39.5	51.5	29.3	42.3
BL, CS	40.2	51.9	30.2	42.4

ZE and BE denote the zero-one error and balanced error percent rates respectively. ZL and BL are the zero-one and balanced loss functions respectively. S, SS and CS denote the scaling, scale & shift and Crammer-Singer schemes respectively.

that we are currently investigating. Note that these observations regarding these three approaches hold for the two-level approaches that use hierarchical information as well.

Comparing the performance of the S and SS schemes against that of the direct K-way classifier we see that the two-level schemes are somewhat worse for DS2 and DS3 and considerably better for DS1 and DS4. In addition, they are consistently and substantially better than the MaxClassifer approach across all four datasets.

4.1.3 Hierarchical Two-Level Learning Approaches. Tables 2 and 3 contains results that show the performance that is achieved by incorporating different types of hierarchical information in the two-level learning framework. For the remote homology prediction problem they present results that combine information from the ancestor nodes (fold and fold+class), whereas for the fold recognition problem they present results that combine information from ancestor nodes (class), descendant nodes (superfamily), and their combination (superfamily+class).

Analyzing the results obtained for the remote homology prediction problems we see that the use of hierarchical information does not improve the error rates. In fact, the two-level schemes that do not use hierarchical information achieve consistently smaller error rates than the ones that do. However, the situation is different for the fold recognition problems in which the use of hierarchical information leads to some improvements for DS4, especially in terms of balanced error.

In terms of which hierarchical information is more beneficial, by looking at the various results we can see that adding information from ancestor nodes is in general better than adding information from descendant nodes, and combining both types of information can sometimes lead to good classification performance. In fact, the best performance (ZE of 27.3% and BE of 41.0%) was achieved by such a combined scheme.

4.1.4 Alternative Performance Assessment Methods. Given a sequence x, the various classification functions that are learned during the second-level learning (Equations 4–8) also return a ranking of the K classes. This ranking provides key information as to what the classifier believes are the most likely classes of x. In the case of the zero-one and balanced error only the first class in this ranked order is considered. If it happens to be correct, then there is no error, whereas if the highest rank class is incorrect, then x is considered to be mispredicted. However, from a practical standpoint, certain mispredictions are worse than others. For example, if x's true class is the second ranked prediction, then this is better than if it was the last ranked prediction.

To better understand the multiclass models produced by incorporating hierarchy information we analyzed the classification errors of the two-level approach that does not use hierarchy information and those produced by the approach that does in terms of their position within the computed ranking. Due to space constraints, we limited our analysis to the DS3 and DS4 datasets and the hierarchy-aware scheme that utilizes class-level information.

For each misclassified sequence x we computed two quantities. The first, referred to as IN, is the number of folds that are part of the same SCOP class with x that were ranked higher than x's true fold. The second, referred to as OUT, is the number of folds that are part of a different SCOP class from x that were ranked higher than x's true fold. The sum of the IN and OUT values for each one of the mispredicted sequences for the various schemes are shown in Table 4. These results show that the schemes that utilize hierarchy information have consistently smaller IN and OUT values and in many cases, these differences are quite substantial. The reduction in terms of the OUT values is general higher, indicating that by incorporating SCOP class information, the classifiers were able to eliminate many of the incorrect rankings that put a fold that belongs to a different SCOP class as a better prediction than a fold within the same SCOP class. The reduction in terms of the IN values indicate that the classifiers utilizing hierarchy information were able to move the correct fold higher up in the ranking. Both of these characteristics are desirable, indicating that the use of hierarchy information does lead to better classifiers, even though they may not reduce the zero-one or the balanced error.

Table 4: Ancestor Level Errors for the fold recognition problem.

		DS3		DS4	
	METHOD	IN	OUT	IN	OUT
Without Hierarchy Information					
	ZL, S	91	411	130	506
	ZL, SS	96	450	136	499
	ZL, CS	91	377	134	490
	BL, S	118	487	137	536
	BL, SS	109	462	136	525
	BL, CS	100	438	137	536
With Class-level Nodes					
	ZL, S	96	371	103	438
	ZL, SS	94	375	102	442
	ZL, CS	90	371	131	472
	BL, S	93	402	118	454
	BL, SS	94	389	118	454
	BL, CS	104	400	118	454

IN and OUT are assessment statistics (See text for details). ZL and BL are the zero-one and balanced loss functions respectively. S, SS and CS denote the scaling, scale & shift and Crammer-Singer schemes respectively.

4.1.5 SVM-Struct versus Ranking Perceptron. For the two-level approaches that do not use hierarchical information, Tables 2 and 3 show the error-rates achieved by both the ranking perceptron and the SVM-struct algorithms. From these results we can see that for the S and SS schemes, the performance achieved by the ranking perceptron are comparable to and in some cases slightly better than those achieved by the SVM-struct algorithm. However, in the case of the CS scheme, SVM-struct is superior to the perceptron and achieves substantially smaller error rates.

This relative performance of the perceptron algorithm is

both surprising as well as expected. The surprising aspect is that it is able to keep up with the considerably more sophisticated, mathematically rigorous, and computationally expensive optimizers used in SVM-struct, which tend to converge to a local minimum solution that is close the global minimum. However, this behavior, especially when the results of the CS scheme are taken into account, was expected because the hypothesis spaces of the S and SS schemes are rather small (the number of variables in the S and SS models are K and $2K$, respectively) and as such the optimization problem is relatively easy. However, in the case of the CS scheme which is parameterized by K^2 variables, the optimization problem becomes harder, and SVM-struct's optimization framework is capable of finding a better solution.

Due to this observation we did not pursue the ranking perceptron algorithm any further when we considered two-level models that incorporate hierarchy information.

4.1.6 Zero-One versus Balanced Loss.

Comparing the two different loss functions we see that for almost all schemes, balanced loss leads to smaller zero-one and balanced error rates. Even though this result was expected for the balanced error, for which balanced loss was specifically designed for, its advantage in terms of zero-one error was surprising. Determining the reason for this behavior is currently under investigation.

4.2 Comparison with Earlier Results

As discussed in the introduction, our research in this paper was motivated by the recent work of Ie *et. al.* [17] in which they looked at the same problem of solving the K-way classification problem in the context of remote homology and fold recognition and presented a two-level learning approach based on the scaling scheme (S) with and without hierarchical information. Table 5 shows the results that were reported in their paper for the DS1 dataset for the remote homology prediction problem. All the methods are similar in nature with the corresponding schemes presented in Table 2.

The key differences between the methods shown in Table 5 and our corresponding methods are that (i) the one-vs-rest binary classifiers were obtained using the profile kernel [22] whereas our schemes used the SW-PSSM kernel, and (ii) our results have been optimized by performing a model selection step (Section 3.6). Comparing the performance of the MaxClassifier scheme in Tables 2 and 5 we can see that our approach achieves substantially smaller error rates. This is a direct consequence of the fact that the SW-PSSM kernel leads to better binary classifiers than the profile kernel, which is in agreement with the results presented in [31]. Also, the performance of our corresponding two-level learners is better than those shown in Table 5. We believe that this is due to the improved binary classifiers as well as model selection.

Note that Ie *et. al.* [17] also presented results in which they used the DS1 dataset for fold recognition as well. However, in their experiments they used the same set of families that were kept aside to evaluate the remote homology prediction performance for assessing the performance of fold recognition. However, as was discussed in Section 3.1, this method does not provide a representative fold recognition performance, since the test sequences are remotely homologous to the folds that we want to predict. For this reason, we did not use DS1 and its corresponding family-based test set in our experiments and we cannot compare our results with the fold-recognition results presented in [17].

Table 5: Comparative results for the remote homology detection problem on dataset DS1

	ZE	BE
Simple Combination of Binary Outputs		
MaxClassifier	20.7	38.0
Two-Level Approaches		
Without Hierarchy Information		
Ranking Perceptron		
ZL, S	21.9	34.4
BL, S	21.8	36.7
SVM-Struct		
ZL, S	21.8	36.7
BL, S	20.7	37.6
With Hierarchy Information		
With Fold-level Nodes		
Ranking Perceptron		
ZL, S	23.0	37.6
BL, S	20.6	34.9
SVM-Struct		
ZL, S	24.8	37.4
BL, S	20.4	37.5

ZE and BE denote the zero-one error and balanced error percent rates respectively. ZL and BL are the zero-one and balanced loss functions respectively. S denotes the scaling scheme results. Note these results were previously published in [17].

5 Discussion and Conclusions

The work described in this paper was designed to answer three fundamental questions. First, whether or not SVM-based approaches that directly learn multiclass classification models can effectively and computationally efficiently solve the problems of remote homology prediction and fold recognition. Second, whether or not the recently developed highly accurate binary SVM-based one-vs-rest classifiers for remote homology prediction and fold recognition can be utilized to build an equally effective multiclass prediction scheme. Third, whether or not the incorporation of binary SVM-based prediction models for coarser and/or finer levels of a typical protein structure hierarchical classification scheme can be used to improve the multiclass classification performance.

The comprehensive experimental evaluation of a number of previously developed methods or novel methods introduced in the course of this work using four different datasets derived from the SCOP protein structure classification scheme showed that, to a large extent, the answer to all three of these questions to be yes. The schemes developed in this work show that SVM-based approaches are a viable tool for developing highly effective classifiers and can be used in

production environments under operational requirements that better serve the needs of the biologists.

Acknowledgment

This work was supported by NSF EIA-9986042, ACI-0133464, IIS-0431135, NIH RLM008713A, the Army High Performance Computing Research Center contract number DAAD19-01-2-0014, and by the Digital Technology Center at the University of Minnesota.

References

[1] F. Aiolli and A. Sperduti. Multiclass classification with multi-prototype support vector machines. *Journal of Machine Learning Research*, 6:817–850, 2005.

[2] E. L. Allwein, R. E. Schapire, and Y. Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. In *Proceedings of the 2000 International Conference on Machine Learning*, pages 9–16, 2000.

[3] Stephen Altschul, Warren Gish, Webb Miller, Eugene Myers, and David Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410, 1990.

[4] P. Baldi, Y. Chauvin, T. Hunkapiller, and M. McClure. Hidden markov models of biological primary sequence information. *PNAS*, 91:1053–1063, 1994.

[5] S. E. Brenner, P. Koehl, and M. Levitt. The astral compendium for sequence and structure analysis. *Nucleic Acids Research*, 28:254–256, 2000.

[6] M. Collins and N. Duffy. New ranking algorithms for parsing and tagging: Kernels over discrete structures and the voted perceptron. *Proceedings of the 40th Annual Meeting of the Association of Computational Linguistics*, pages 263–270, 2002.

[7] Michael Collins. Parameter estimation for statistical parsing models: Theory and practice of distribution-free methods. In *New Developments in Parsing Technology*, pages 1–38. Kluwer, 2001.

[8] K. Crammer and Y. Singer. On the learnability and design of output codes for multiclass problems. *Computational Learning Theory*, pages 35–46, 2000.

[9] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2:265–292, 2001.

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

[11] Y. Guermeur. A simple unifying theory of multi-class support vector machines. Technical Report RR-4669, INRIA, 2002.

[12] Y. Guermeur, A. Elisseeff, and D. Zelus. A comparative study of multi-class support vector machines in the unifying framework of large margin classifiers. *Applied Stochastic Models in Business and Industry*, 21:199–214, 2005.

[13] A. Heger and L. Holm. Picasso:generating a covering set of protein family profiles. *Bioinformatics*, 17(3):272–279, 2001.

[14] L. Holm and C. Sander. The fssp database: fold classification based on structure-structure alignment of proteins. *Nucleic Acids Research*, 24(1):206–209, 1996.

[15] Y. Hou, W. Hsu, M. L. Lee, and C. Bystroff. Efficient remote homology detection using local structure. *Bioinformatics*, 19(17):2294–2301, 2003.

[16] Y. Hou, W. Hsu, M. L. Lee, and C. Bystroff. Remote homolog detection using local sequence-structure correlations. *Proteins:Structure,Function and Bioinformatics*, 57:518–530, 2004.

[17] E. Ie, J. Weston, W. S. Noble, and C. Leslie. Multi-class protein fold recognition using adaptive codes. In *Proceedings of the 2005 International Conference on Machine Learning*, 2005.

[18] T. Jaakkola, M. Diekhans, and D. Haussler. A discriminative framework for detecting remote protein homologies. *Journal of Computational Biology*, 7(1):95–114, 2000.

[19] Tommi Jaakkola, Mark Diekhans, and David Hassler. A dscriminative framework for detecting remote protein homologies. *Journal of Computational Biology*, 7(1/2):95–114, 2000.

[20] T. Joachims. *Advances in Kernel Methods: Support Vector Learning*, chapter Making large-Scale SVM Learning Practical. MIT-Press, 1999.

[21] A. Krogh, M. Brown, I. Mian, K. Sjolander, and D. Haussler. Hidden markov models in computational biology: Applications to protein modeling. *Journal of Molecular Biology*, 235:1501–1531, 1994.

[22] R. Kuang, E. Ie, K. Wang, K. Wang, M. Siddiqi, Y. Freund, and C. Leslie. Profile-based string kernels for remote homology detection and motif extraction. *Computational Systems Bioinformatics*, pages 152–160, 2004.

[23] C. Leslie, E. Eskin, and W. S. Noble. The spectrum kernel: A string kernel for svm protein classification. *Proceedings of the Pacific Symposium on Biocomputing*, pages 564–575, 2002.

[24] C. Leslie, E. Eskin, W. S. Noble, and J. Weston. Mismatch string kernels for svm protein classification. *Advances in Neural Information Processing Systems*, 20(4):467–476, 2003.

[25] L. Liao and W. S. Noble. Combining pairwise sequence similarity and support vector machines for detecting remote protein evolutionary and structural relationships. *Proc. of the Intl. Conf. on Research in Computational Molecular Biology*, pages 225–232, 2002.

[26] D. Mittelman, R. Sadreyev, and N. Grishin. Probabilistic scoring measures for profile-profile comparison yield more accurate short seed alignments. *Bioinformatics*, 19(12):1531–1539, 2003.

[27] A. G. Murzin, S. E. Brenner, T. Hubbard, and C. Chothia. Scop: a structural classification of proteins database for the investigation of sequences and structures. *Journal of Molecular Biology*, 247:536–540, 1995.

[28] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48:443–453, 1970.

[29] C. A. Orengo, A. D. Mitchie, S. Jones, D. T. Jones, M. B. Swindells, and J. M. Thorton. Cath- a hierarchic classification of protein domain structures. *Structure*, 5(8):1093–1108, 1997.

[30] W. Pearson. Rapid and sensitive sequence comparisons with fastp and fasta. *Methods in Enzymology*, 183:63–98, 1990.

[31] H. Rangwala and G. Karypis. Profile based direct kernels for remote homology detection and fold recognition. *Bioinformatics*, 21(23):4239–4247, 2005.

[32] R. Rifkin and A. Klautau. In defense of one-vs-all classification. *Journal of Machine Learning Research*, 5:101–141, 2004.

[33] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–407, 1958.

[34] J. Rousu, C. Saunders, S. Szedmak, and J. S. Taylor. Learning hierarchial multi-category text classification methods. In *Proceedings of the* 2^{nd} *International Conference on Machine Learning*, 2005.

[35] H. Saigo, J. P. Vert, N. Ueda, and T. Akutsu. Protein homology detection using string alignment kernels. *Bioinformatics*, 20(11):1682–1689, 2004.

[36] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

[37] A. J. Smola, P. Bartlett, B. Scholkopf, and D. Shuurmans, editors. *Probabilistic outputs for support vector machines and comparison of regularized likelihood methods*, chapter 5, pages 61–74. Advances in Large Margin Classifiers. MIT Press, 2000.

[38] A. Sun and E. Lim. Hierarchial text classification and evaluation. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, pages 521–528, 2001.

[39] B. Taskar, D. Klein, M. Collins, D. Koller, and C. Manning. Max-margin parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2004. Winner of the Best Paper Award.

[40] I. Tsochantaridis, T. Homann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the 2004 International Conference on Machine Learning*, 2004.

[41] V. Vapnik. *Statistical Learning Theory*. John Wiley, New York, 1998.

[42] J. Weston and C. Watkins. Mulit-class support vector machines. Technical Report CSD-TR-89-04, Department of Computer Science, Royal Holloway, University of London, 1998.