A Multi-Level Approach to SCOP Fold Recognition

Keith Marsolo
Srinivasan Parthasarathy
The Ohio State University
Department of Computer Science and Engineering
Contact: srini@cse.ohio-state.edu

Chris Ding
Computational Research Division
Lawrence Berkeley National Laboratory
Berkeley, CA, USA

***Abstract*— The classification of proteins based on their structure can play an important role in the deduction or discovery of protein function. However, the relatively low number of solved protein structures and the unknown relationship between structure and sequence requires an alternative method of representation for classification to be effective. Furthermore, the large number of potential folds causes problems for many classification strategies, increasing the likelihood that the classifier will reach a local optima while trying to distinguish between all of the possible structural categories. Here we present a hierarchical strategy for structural classification that first partitions proteins based on their SCOP class before attempting to assign a protein fold. Using a well-known dataset derived from the 27 most-populated SCOP folds and several sequence-based descriptor properties as input features, we test a number of classification methods, including Naïve Bayes and Boosted C4.5. Our strategy achieves an average fold recognition of 74%, which is significantly higher than the 56-60% previously reported in the literature, indicating the effectiveness of a multi-level approach.**
Keywords: Protein Fold Recognition, Protein Structure, Multi-Class Classification, Ensemble Learning.

I. Introduction

Proteins are an important functional unit in countless cellular and biological processes. Rather than play a singular role, many proteins participate as a member of a group, or complex. High-yield genomic and proteomic experiments have led to the identification of large numbers of protein complexes. While the role of an individual protein within a given complex has been determined for a few molecules, for the vast majority, that role remains a mystery. It is believed, however, that a protein's function is strongly influenced by its structure. Therefore, it is reasonable to believe that proteins with a similar structure might have a similar function. As such, grouping molecules based on their structure will likely help predict a protein's function.

One way of determining similarity is through alignment. A number of different alignment methods exist, but they generally fall into two different categories: sequence-based or structure-based. While each are effective in certain cases, there are drawbacks to both approaches. Sequence-based alignment will fail when two proteins are structurally-similar but share little in the way of sequence homology. On the other hand, structure-based methods rely on data derived from a solved structure. Unfortunately, the number of proteins whose structure has been solved is much smaller than the number of proteins that have been sequenced. As of April 2005, the number of solved proteins in the Protein Data Bank[1] (PDB) stands at just over 30,000, while there are more than 2.1 million sequenced proteins in the PIR-NREF database[2]. Thus, to be truly effective, any method of comparing proteins should not be limited only to molecules with a solved structure. The exact nature of the relationship between a protein's sequence and its structure remains one of the open challenges in computational biology and until a viable solution is obtained, it is not possible to derive a structure directly from sequence. Therefore, a different approach is needed to determine similarity.

A potential solution is to use sequence-based properties to classify proteins whose structure is known. Using these proteins to construct and validate a classifier, one can use the resulting model on unclassified proteins to assign a structure-based label. Such a strategy has been employed by a number of researchers [4,5,13,14]. Given a set of proteins from the Structural Classification of Proteins (SCOP) database, each classifier attempts to correctly predict a protein's *fold*. Using the terminology of the SCOP database, two proteins that belong to the same fold share a common three-dimensional pattern with the same major secondary structure elements (SSEs) in the same arrangement with the same topological connections [10]. In the SCOP hierarchy, folds are grouped into different *classes*, where a class is defined by the topographical arrangement of the secondary structures of its member proteins.

Most of the recent work on this problem uses machine-learning techniques such as Support Vector Machines (SVMs) or Neural Networks (NNs). These methods are highly-tunable and have been shown to provide excellent performance in certain applications. A drawback to both of these techniques is that they are most effective when dealing with a binary, or two-class decision problem. In order to handle a multi-class (or in this case, multi-fold) dataset, a variation of one-vs-others (OvO) or all-vs-all (AvA) classification is often chosen.

With an one-vs-others approach, a classifier is constructed to decide between two classes: the class in question (the "true" class) and the rest (the "others"). Given k classes, k different classifiers are constructed and an input protein is assigned the label of whatever classifier returns a yes vote. In the case of a multiple yes votes, a number of different tie-breaking solutions

[1] http://www.rcsb.org/
[2] http://pir.georgetown.edu/pirwww/search/pirnref.shtml

have been proposed. For example, Ding and Dubchak describe a "unique" one-vs-others (uOvO) strategy that uses a series of 2-way classifiers to decide amongst those classes with a yes vote [5]. With OvO classification, the number of objects in the true class is often very small compared to the number of others. In a dataset with k different classes and an equal number of objects per class, this results in a classifier that tries to distinguish just $1/k$ of the objects from the rest. Given an unequal number of objects per class, this number can be even lower. As a result, an individual classifier can have high accuracy even if it misclassifies everything in the true class.

Using an all-vs-all strategy and a dataset with k classes, a classifier is constructed for every possible pair of classes, resulting in $k(k-1)/2$ different classifiers. Given an input object, it is tested with every classifier, and the class returning the largest number of "yes" classifications is assigned to the object. The drawback here is that AvA requires the construction of a large number of classifiers, while at the same time using a smaller number of datapoints in the construction of those classifiers, which can lead to over-fitting. As a result, care must be taken in order for either of these approaches to be effective.

Ding and Dubchak were one of the earliest groups to report on the problem of fold recognition, comparing the classification accuracy of a neural network and a support vector machine (testing the effectiveness of both OvO and AvA classification) on a set of proteins taken from the 27 most-populated SCOP folds, with less than 25% sequence identity between every protein in the set [5]. They were able to recognize the correct fold with an accuracy of approximately 56% using a number of sequence-based properties as feature vectors for their classifier. Using a slightly smaller dataset and many of the same input features, Tan et al. developed a rule-based classifier that combines the best classifiers from the OvO and AvA methods to generate a single classifier for each of the possible folds. Their best results improve fold recognition to roughly 60% [14].

Classifying the same dataset and input features, but this time employing a Bayesian Network-based approach, Chinnasamy et al. improve on the average fold recognition results reported by Ding and Dubchak, but fail to increase accuracy above 60% [4]. Finally, again starting with the same dataset and input features, Shi et al. employ an evolutionary algorithm to select the most relevant features and classify using a SVM, but their average accuracy remains around 56% for the 27-fold problem [13].

As an alternative to the above methods, in this paper we propose a tiered, or multi-level, approach to classification. Just as proteins in SCOP are grouped in a hierarchical fashion, it is possible to first partition a dataset based on certain high-level similarities. For each partition, one can then create a more specific, fine-grained classifier. To our knowledge, it is the first time such an approach has been applied to this particular problem. Here we report our method and results using a two-level classification strategy.

In the first stage, we classify proteins based on their SCOP class. For each class of proteins, we then classify by fold. Our approach is flexible, allowing for any classification strategy to be used at any stage. We test our method with a Naïve Bayes classifier and several decision tree-based meta-learning strategies. Using a dataset and feature vectors similar to those reported elsewhere [4,5,13,14], we achieve an improvement in accuracy of 15-20% compared with previously published results, predicting the correct fold with an accuracy up to 74%. We only need to create a single classifier for the Class level and then one for each Fold, for a total of five classifiers. As opposed to previous methods, our model is fast, scalable, and can easily be recreated if there is a large change in the dataset or input features.

II. Multi-Level Classification

Databases like SCOP group proteins in a hierarchical, tree-like fashion based on shared structural characteristics. The top of the hierarchy is referred to as the Class level. The classes are divided into Folds, which are further divided into *Superfamilies*. Each Superfamily is itself composed of several individual *Families*. Since proteins at each level are grouped based on shared characteristics, we would expect a protein to have a higher structural similarity to those proteins that are within the same class (intra-class) than those that are without (inter-class). The same would hold true at the fold level and on down through the hierarchy. For this reason, we believe a classification scheme should take advantage of such a natural hierarchy. Most of the existing work on this problem have taken a "flat" view toward classification, focusing only on the fold level, totally ignoring any class-level information.

The only reason it is currently possible to classify proteins at the fold level with a binary classifier is that there are few folds with the requisite number of members needed to effectively train a classifier. The current version of the SCOP database (May 2005) lists 887 possible folds. As the number of solved protein structures increases, so will the number of folds that can be classified. When this occurs, the use of binary classification strategies will become less and less practical. When presented with 887 potential folds, an OvO strategy will require the training of 887 2-way classifiers. An AvA method, on the other hand, will require the construction of a total of 392,941 different classifiers. The use of cross-validation will increase that number even more. With *n*-fold cross validation, $1/n$ of the dataset is set aside as a testing set (a *fold*) and a classifier is trained on the remaining members. This process is repeated n times with a different $1/n$ of the dataset set aside in each iteration (an object can only be a member of one testing fold). Therefore, when using n-fold cross-validation and an OvO classification strategy on k possible classes, a total of kn classifiers must be trained, while AvA increases that number to $nk(k-1)/2$. Finally, if one wishes to classify proteins further down the hierarchy, where the number of potential choices is even larger, the problem quickly becomes intractable.

As stated previously, classification techniques that can effectively handle multi-class data, such as Bayesian or decision

tree-based methods, have a difficult time distinguishing between a large number of possible classes. One way to try and reduce the potential confusion is to pre-process or pre-partition the data before classification occurs. By using a *multi-tiered* or multi-level classification strategy, one can cut down on the number of potential outcomes, which is useful when faced with noisy, real-world data that is not clearly separable. The multi-tier strategy we present here is meant to be general. It can be applied to any domain where the data falls into a natural hierarchy (or one where such a hierarchy can be readily deduced). In addition, any classification strategy can be used at the different levels of the hierarchy. If a certain method is found to be more effective at one stage, it can be used there and replaced with something else at the others.

A. Algorithm

We provide the implementation details of our multi-level strategy in Section III-B. The general idea, however, is to first classify proteins at the class level, grouping them based on global, high-level features. We refer to this stage as *Classification by SCOP Class.* Once this partitioning is complete, we subdivide further, classifying each protein by fold (denoted *Class-specific Fold Classification*). The intent of this step is to improve accuracy by using a increasingly fine-grained classification model, separating the data based on more local, fold-specific attributes than a typical decision tree that is forced to distinguish between all possible classes (which we call *All-Folds Classification*). With this approach, the classifier tends to employ two different sets of features in classification. As we show below, at the Class level, the focus is on those elements that can partition the dataset on global, high-level features. At the Fold level, more emphasis is placed on the features that allow a distinction based on more local, specific attributes.

B. Validation

The intuition behind our classification strategy is based on several well-known and empirically-validated assumptions [3,9], which are listed below:

- Multi-stage classifiers may be more accurate than single-stage classifiers.
- Smaller decision trees have higher accuracy than larger trees.
- Locally optimizing information tends to produce small, shallow, accurate trees.

Of particular importance are the last two assumptions. By dividing the overall task of global fold recognition into the smaller sub-tasks of recognizing class and class-specific folds, we hope to produce smaller, more accurate decision trees that avoid the problem of getting stuck in a local minima. Large decision trees often suffer from *noise*, *fragmentation* and *subtree replication* [3,11]. Noise can cause irrelevant features to be used as selection criteria, which in turn can lead to overfitting. A similar effect can arise due to tree fragmentation, where there are a large number of leaf nodes that only represent a few objects. Finally, a large decision tree

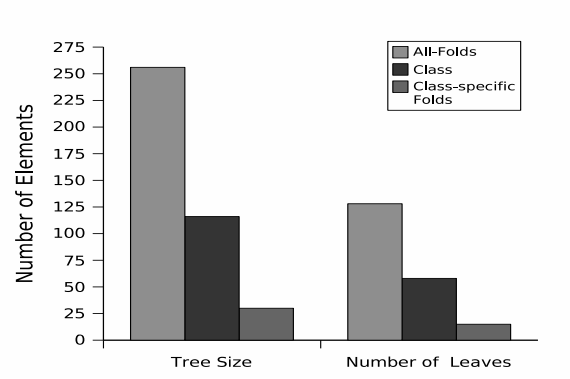

Fig. 1. Average decision tree size and average number of leaves for each of the classification variations tested (Class, All-Folds, Class-specific Folds). Results reflect a boosted C4.5 decision tree.

can suffer from subtree replication, where a certain piece of the tree is repeated throughout the overall structure, which can also cause fragmentation. We now provide results that motivate the development of our multi-tier strategy and empirically validate our approach.

In Figure 1 we provide the average decision tree size and average number of leaf nodes for the different single-stage classification strategies mentioned above: Class, Class-specific Fold and All-Folds. These values were obtained when running the boosted C4.5 experiments discussed in Section III-B on one of the single property datasets (described in Section III-A). While the actual numbers may differ depending on the property chosen, the overall trends and differences between classification strategies remain.

Figure 1 clearly shows the drawback to using a single-stage classifier on a large multi-class decision problem. The size of the average tree for All-Folds classification is greater than 250 nodes, and it contains over 125 leaves. With a dataset of approximately 650 proteins (the size of our set), fragmentation becomes a real concern. In contrast, in Figure 1 one can see that using Class or Class-specific Fold classification results in a significantly smaller decision tree and far fewer leaf nodes. The Class tree is less than half the size of the All-Folds tree, and the Class-specific Fold tree is roughly an eighth the size of the original.

The problem of subtree replication is illustrated in Figure 2. Using one of the single-property datasets as an example, we list the top five decision tree attributes in terms of average number of occurrences per tree (results again from the boosted C4.5 experiments). Shown are the top five attributes for the All-Folds method and the corresponding counts of those attributes using the other methods. The top attribute appears an average of 17 times in each tree when classifying using All-Folds. That same attribute occurs less than six times when classifying by SCOP Class and fewer than three times in a Class-specific-Fold tree. In general, the All-Folds attributes appear 2-3 times more often than in the Class trees and 4-6 times more often than the Class-specific Fold trees. It should

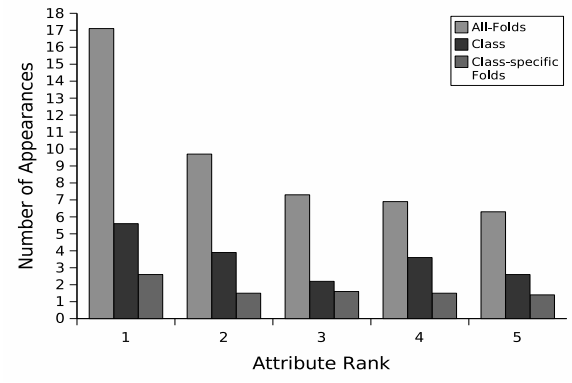

Fig. 2. Average number of appearances per tree for each of the top five attributes used as splitting criteria in the All-Folds method on one of the single-property datasets.

TABLE I

RANK OF THE TOP 5 ATTRIBUTES OF A *Class* DECISION TREE IN THE *Class-specific Fold* AND *All-Folds* METHODS (USING THE SAME DATASET AS FIGURE 2). A "–" INDICATES A RANK BELOW 5.

Class	Class-Specific Fold	All-Folds
1	1	1
2	–	2
3	–	4
4	–	5
5	–	–

be noted that the top attributes in the All-Folds trees are not necessarily the top attributes in the trees of the other methods (and vice versa).

In Table I, using the same single-property dataset as in Figure 2, we list the top 5 attributes (in terms of average number of occurrences per tree) in the Class method. We also list the corresonding rank of those attributes in the Class-specific-Fold and All-Folds methods. A "–" indicates that the attribute is not among the top 5 attributes. As one can see in Table I, there is a high degree of overlap between the Class and All-Folds methods and almost no overlap between the Class and Class-specific Fold trees. Similar results are seen with the other datasets. This gives credence to our belief that using a multi-level classification strategy allows a classifier to focus on different features at different levels: global features at the top level, local features at the lower level.

In light of these results, we feel that a multi-stage strategy like the one proposed here will eliminate many of the issues that arise from a large multi-class decision problem. Furthermore, by breaking the task of classification into a number of stages, we can more finely-tune our overall classifier, taking advantage of specific strengths that an individual classifier may have at a particular level, using a different method for each one.

TABLE II

SEQUENCE-BASED PROPERTIES USED AS FEATURE VECTORS FOR CLASSIFICATION.

Symbol	Property	Dim.
c	Amino Acid Composition	20
h	Hydrophobicity	21
p	Polarity	21
s	Predicted Secondary Structure	21
v	van der Waals Volume	21
z	Polarizability	21

III. METHODS

Here we provide details on our dataset and give a description of our classification experiments.

A. Dataset

The dataset used in these experiments is based on the dataset first described in the work by Ding and Dubchak [5]. There, a training set was taken from the 27 most populated SCOP folds of the PDB_Select set, in which no two proteins share more than 35% sequence identity for aligned subsequences longer than 80 residues. This training set contained 311 proteins. Ding and Dubchak used an independent test set derived from the PDB_40D set, which consists of all SCOP sequences having less than 40% sequence identity with each other. Using the same 27 SCOP folds, 385 proteins were selected, and any PDB_40D protein having more than 35% sequence identity with the proteins in the training set was excluded. When combined together, the training and test sets yield a total of 696 proteins.

Since the publication of the work by Ding and Dubchak [5], the protein identifiers used in the SCOP database have changed. We were looking to use this dataset for a different set of experiments and needed to match the original identifiers to those currently used by SCOP. Using the latest labels from SCOP (version 1.67), a Perl script was written to automate the matching process, but we were unable to find a match for every protein. Rather than attempt to manually match the proteins that remained, we simply removed them from consideration, leaving a total of 653 proteins in our dataset.

We represent each protein in our dataset using the same sequence properties as those listed in previous experiments [4,5,13,14]. The feature vectors characterize the following properties for each protein (the symbol for each descriptor given in parentheses): amino acid composition (c), hydrophobicity (h), polarity (p), predicted secondary structure (s), van der Waals volume (v), and polarizability (z). For a more detailed discussion on the derivation of these properties, the reader is referred to the work by Dubchak et al. [6] or Ding and Dubchak [5]. The feature vector for the amino acid composition consists of 20 dimensions. All of the rest have 21. These properties are presented in Table II.

We test each descriptor individually and also combine them to create longer feature vectors, concatenating the feature

vector of one descriptor onto the end of another. The combined datasets are referenced by their combined symbols (cs, csh, etc.). These combined vectors ranged in size from 41 dimensions (cs) to 125 (cshpvz).

B. Experiments

In order to show the effectiveness of our classification strategy, we need to run a number of different tests in order to establish a series of baseline results. As such, we conduct the following classification experiments:

1) **Classification by SCOP class.** In this test, each SCOP fold is replaced by its corresponding SCOP class label (α, β, $\alpha + \beta$, α/β). Table III lists the identification number of the SCOP folds contained in our dataset and their corresponding class. The purpose of this test is to establish that it is possible to improve accuracy by classifying proteins at an "upper," or more global level. This mode of classification is analogous to the "Class" method of the previous section.
2) **Classification by SCOP fold.** This experiment is an attempt to replicate the work reported elsewhere in the literature [4,5,13,14]. Here, we create a classifier to distinguish among the 27 different folds in our dataset. Rather than employ a multi-classifier variation of OvO or AvA, we use a single classifier for this task, deciding between all possible classes at once. The results from this experiment are the same as the All-Folds results discussed in the previous section.
3) **Classification by SCOP fold after partitioning by class.** We repeat the previous experiment, but rather than classify the entire dataset at once, we manually partition by class and then distinguish between folds. We hope that classification accuracy will improve after removing those proteins that fall into different areas of the structural hierarchy. This experiment is equivalent to Class-specific Fold classification.
4) **Multi-Level Classification.** The final experiment is a test of our multi-tiered classification strategy. First, we classify each protein based on its SCOP class, using 10-fold cross-validation. Then, for each SCOP class, we take all of the proteins that were correctly classified and construct a classifier that distinguishes between the folds present in that class (see Table III for the relationship between class and fold).

   For each SCOP class, we take a number of random samples (twenty in our tests) of the data, dividing it into 70%/30% training/testing splits. While a protein can appear in more than one sample, we do not allow them to appear more than once within a given sample (i.e. they are randomly selected without replacement). Finally, in an attempt to ensure that every protein is adequately represented, we limit the number of samples in which a protein can be present to half the total number of samples.

   To compute the accuracy of our multi-stage method, we take the number of proteins that were misclassified at the SCOP class level and add to that the average number of misclassifications at the fold level. To compute the average, we sum the number of misclassifications for each random sample and divide by the total number of samples. By using the ideas behind Experiments 1 and 3, we hope to show a substantial improvement in classification accuracy over the results of Experiment 2.

TABLE III
SCOP CLASSES AND THEIR CORRESPONDING FOLDS

Class	Folds	Total
α	1, 2, 4, 7, 9, 11	6
β	20, 23, 26, 30, 31, 32, 33, 35, 39	9
α/β	46, 47, 48, 51, 54, 57, 59, 62, 69	9
$\alpha + \beta$	72, 87, 110	3

C. Experimental Setup

All of our experiments were conducted on a PC with a 2.8 GHz Pentium 4 CPU and 1.5 GB RAM, running Debian Linux with a custom 2.6.9 kernel. The classification process was done using the WEKA data mining toolkit, version 3.4[3] and Sun Java 1.4.2. Classification accuracy is given as the True Positive Rate (TPR), or the number of correct classifications divided by the total. We test each experiment using a number of different classifiers, including Naïve Bayes, Random Forests, C4.5 with Bootstrap Aggregation (bagging) and C4.5 with Adaptive Boosting (boosting) [1,2,7,8,12]. We left all of the classifiers on their default setting, except for the C4.5 algorithm, which we changed to use binary splits. All of the proteins were combined into one large dataset and classified using 10-fold cross-validation.

IV. RESULTS

We now present the results of our classification experiments. We found that the classification accuracy of Random Forests and Bagged C4.5 were very close to that of Boosted C4.5, though when there was a difference, Boosted C4.5 tended to provide the highest accuracy. Thus, in order to improve clarity, we omit the results from Random Forests and Bagged C4.5.

Classification by SCOP class: In Figure 3, we provide the results of our experiments in trying to recognize the correct SCOP class. As we show next, the accuracy when classifying by SCOP class is higher than the accuracy when classifying by fold. This is due to the fact that we are dealing with 4 classes, as opposed to 27 folds. For most of the single parameter datasets, the accuracy hovers around 55% when using the Naïve Bayes classifier and approximately 60% when using a Boosted C4.5 decision tree. The *c* and *s* datasets perform much better than the others, with accuracy above 70% for both classifiers. Performance can be improved further by combining the datasets into larger feature vectors. Figure 3, shows that combining the feature vectors increases accuracy to 80% with the Naïve Bayes classifier and to 84% with Boosted C4.5. This

[3]http://www.cs.waikato.ac.nz/~ml/weka/

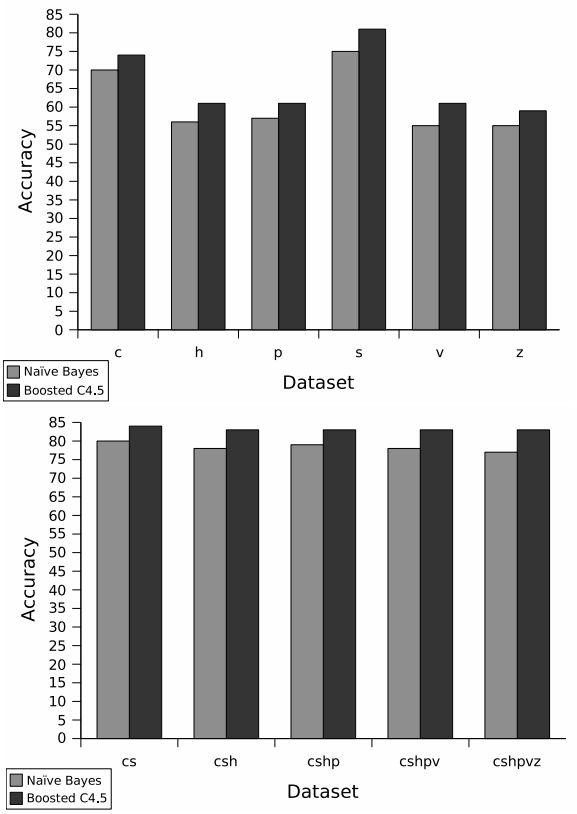

Fig. 3. Accuracy when classifying by SCOP class.

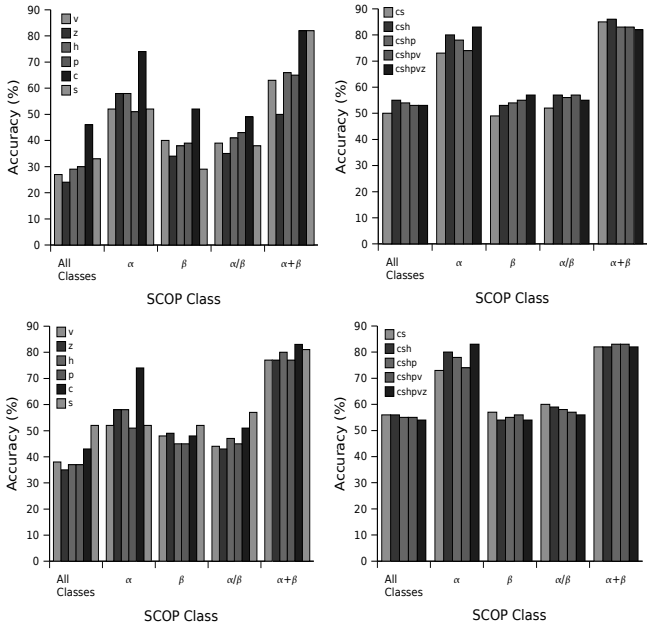

Fig. 5. Accuracy for SCOP Fold recognition by Class. The top figure represents results obtained using a Naïve Bayes classifier. The bottom figure contains those returned by a Boosted C4.5 decision tree. Also provided are the results for single-stage fold recognition using the entire dataset.

is 3 to 5% better than the best single parameter dataset for the Boosted and Bayes classifiers, respectively.

Classification by SCOP fold: In Ding and Dubchak [5], the highest accuracy for the 27-fold problem is obtained using an all-vs-all support vector machine. This requires the construction of 351 different classifiers. The best classification accuracy with the AvA SVM was around 56% using the *csh* and *cshp* datasets. The results in Figure 4 show that we are able to achieve the same classification accuracy with a single classifier.

Our accuracy on the single parameter datasets are much higher than the results of Ding and Dubchak [5]. There is only one case where the Naïve Bayes classifier does not outperform the SVM and the Boosted C4.5 classifier always outperforms the SVM by at least 10%. These results are also higher than those reported elsewhere [13].

Classification by SCOP fold after partitioning by class: We provide the accuracy of trying to recognize the correct SCOP fold when the dataset has already been partitioned by class. In theory, this should be an easier problem, as there are fewer folds to choose from and the dataset for each class is smaller than the original. As one can see in Figure 5, the accuracy for most of the class-specific datasets is higher than that of the multi-class (listed under the *All Classes* column). There are a few instances where this does not hold, most notably for the Naïve Bayes classifier and the β class. Overall, however, classification accuracy is substantially improved. Since it is unlikely that the user would ever be presented with a dataset that is already partially classified, these results may not seem to be very useful. However, they are presented here to give some intuition as to why a multi-level classification strategy might be more effective than a single-stage strategy when dealing with a large number of possible folds.

Multi-Level Classification: We present the results of our multi-level classification strategy in Figure 6. As one can see, these results are much higher than those of the single-stage solution (given in Figure 4). For the combined datasets, the multi-stage results are a full 12%-18% higher than the corresponding single-stage value. In addition, *our best Multi-Stage results are around 14% higher than any result we have seen in the literature on this problem.*

In Table IV, we provide an analysis of the True Positive Rate by fold for our single-tier and multi-tier methods on the *cs* dataset, as well as the results of the unique One-vs-Others Support Vector Machine as reported in Ding and Dubchak [5]. As one can see, the average results for all of our methods are much higher than those reported previously, with a 12% difference between our Multi-Level Boosted C4.5 classifier and the uOvO SVM. There are certain folds where the SVM outperforms both Bayes and C4.5, but in general it has a lower TPR, especially when compared to the multi-level solutions.

V. Discussion

In this work, we describe our multi-stage classification strategy and present the results achieved when classifying a well-known protein dataset. Using our method, we can

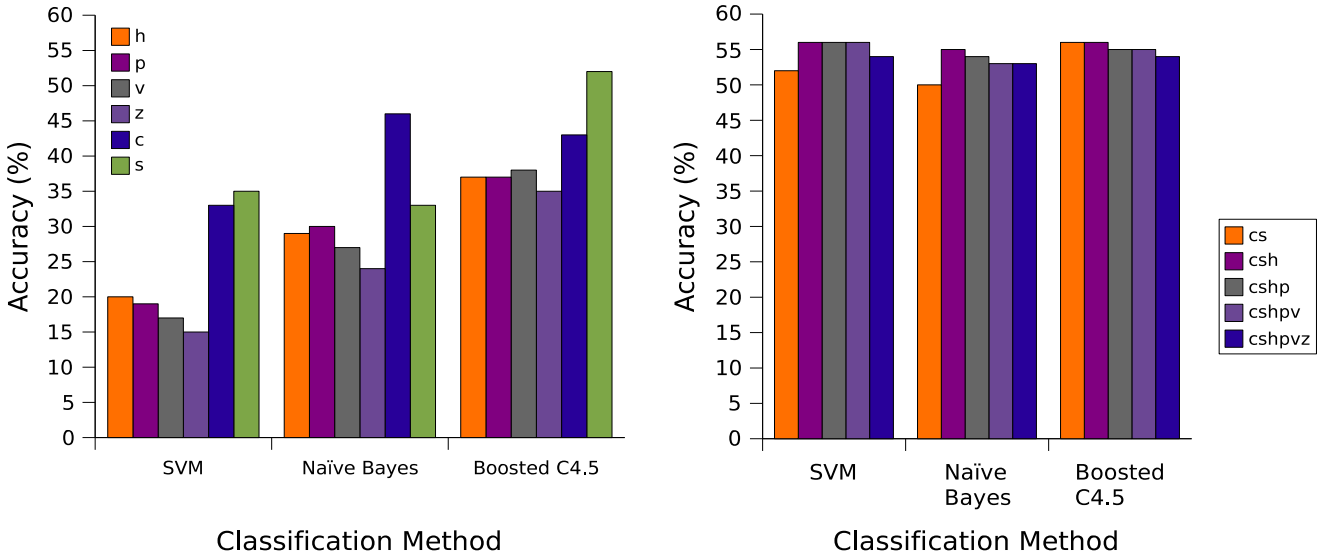

Fig. 4. Accuracy for SCOP Fold recognition when using a single-stage Naïve Bayes or Boosted C4.5 classifier. The SVM results are taken from Ding and Dubchak [5]. The single parameter SVM datasets reflect the cross-validation results provided there. The SVM results given for the combined datasets were taken from the AvA experiments.

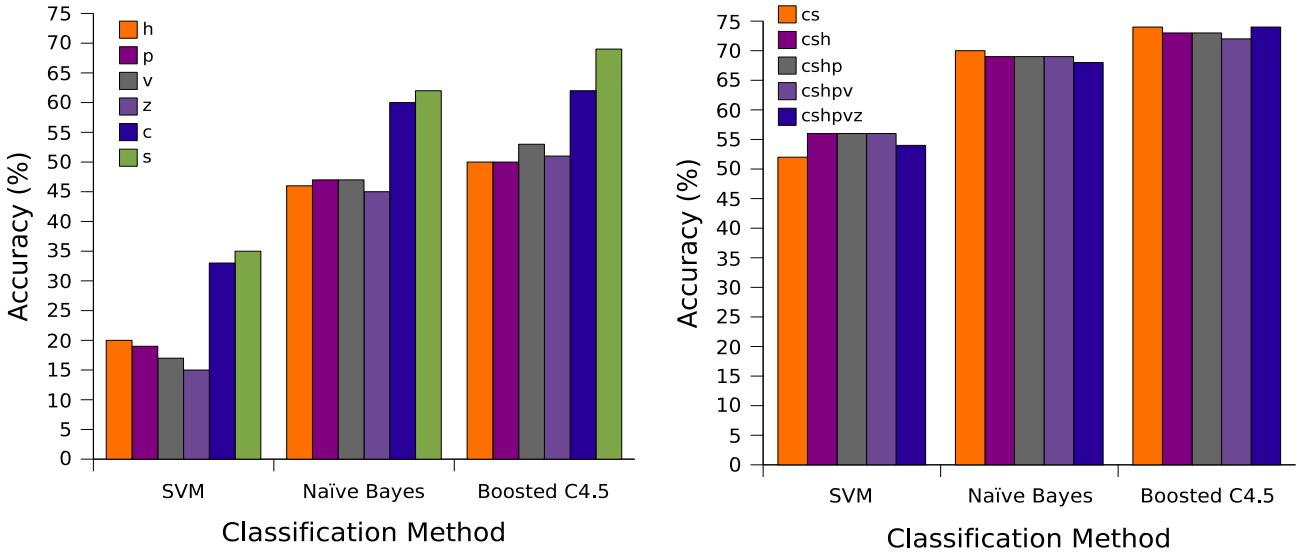

Fig. 6. Accuracy for SCOP Fold recognition when using Naïve Bayes or Boosted C4.5 classifier with the Multi-Level classification strategy. The SVM results are taken from Ding and Dubchak [5]. The single parameter SVM datasets reflect the cross-validation results provided there. The SVM results given for the combined datasets were taken from the AvA experiments.

improve classification accuracy by approximately 15% over values seen in the literature.

Trying to predict the fold of a protein is extremely challenging. There are a large number of possible folds, and many folds contain a small number of members. Most existing solutions to this problem rely on some type of all-vs-all or one-vs-others classification strategy. As stated previously, both of these strategies have their drawbacks. Our proposed method leverages the fact that folds that fall within the same SCOP class share similar structural properties. Therefore, *we can use high-level, global similarities to partition the data at the class level*. This partitioning simplifies the classification problem, both in the number of proteins that must be classified and in the number of potential folds. It is true that the proteins within each partition will be more structurally similar, which would seem to present a more difficult classification challenge. However, in a given partition, *we can use local, more discriminating features to distinguish between proteins*

TABLE IV

TRUE POSITIVE RATE (TPR) BY SCOP FOLD FOR UNIQUE ONE-VS-OTHERS SUPPORT VECTOR MACHINE (UOVO SVM) (TAKEN FROM [5]), SINGLE-STAGE NAÏVE BAYES AND BOOSTED C4.5 CLASSIFIERS AND MULTI-LEVEL NAÏVE BAYES AND BOOSTED C4.5 CLASSIFIERS. ALSO GIVEN IS AVERAGE TPR (ACCURACY).

Fold	uOvO SVM	Naïve Bayes	Boosted C4.5	Multi-Level Bayes	Multi-Level Boost
1	0.83	0.74	0.74	0.76	0.66
3	0.67	0.94	0.69	0.89	0.78
4	0.47	0.77	0.63	0.97	0.82
7	0.63	0.47	0.60	0.41	0.51
9	1	0.87	0.57	0.74	0.71
11	0.56	0.53	0.47	0.42	0.43
20	0.60	0.43	0.64	0.71	0.74
23	0.17	0.58	0.32	0.40	0.28
26	0.54	0.59	0.59	0.55	0.70
30	0.33	0.54	0.39	0.32	0.30
31	0.50	0.50	0.53	0	0.66
32	0.32	0.23	0.27	0.33	0.30
33	0.50	0.25	0.25	0.28	0.32
35	0.25	0.31	0.15	0.34	0.27
39	0.50	0.63	0.69	0.58	0.72
46	0.65	0.67	0.75	0.67	0.75
47	0.54	0.35	0.48	0.59	0.51
48	0.35	0.48	0.39	0.34	0.27
51	0.47	0.51	0.39	0.63	0.49
54	0.36	0.46	0.41	0.40	0.45
57	0.25	0.57	0.36	0.36	0.49
59	0.29	0.52	0.48	0.55	0.47
62	0.71	0.50	0.25	0.44	0.38
69	0.25	0.50	0.50	0.45	0.39
72	0.25	0.23	0.23	0	0.54
87	0.30	0.24	0.24	0.14	0.49
110	0.83	1	1	1	0.97
avg	0.51	0.55	0.56	0.69	0.73

and assign the correct fold. As additional protein structures are crystallized and labeled, this strategy can also be used to help classify unknown proteins at the Superfamily or Family level.

Our strategy can be used with any type of classifier, it need not be Naïve Bayes or Boosted C4.5. It is also possible to use different classifiers for each tier. If new data is acquired, it is relatively simple to regenerate the classification model. One can also use this strategy in a semi-supervised environment, using unlabeled data to either tune the classifier or to make predictions about the possible fold/function of an unknown protein. Also, if desired, one can create a classifier that is biased toward a particular class or fold. In this manner, one can more reliably identify proteins of that fold, with a possible detriment to the identification accuracy of the other folds.

REFERENCES

[1] L. Breiman. Bagging predictors. *Machine Learning*, 26(2):123–140, 1996.

[2] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[3] L. A. Breslow and D. W. Aha. Simplifying decision trees: A survey. Technical Report AOC-96-014, NCARAI, 1996.

[4] A. Chinnasamy, W. K. Sung, and A. Mittal. Protein structure and fold prediction using tree-augmented naive bayesian classifier. In *Proc. PSB 2004*, Stanford, CA, 2004. World Scientific Press.

[5] C. H. Q. Ding and I. Dubchak. Multi-class protein fold recognition using support vector machines and neural networks. *Bioinformatics*, 17(4):349–358, April 2001.

[6] I. Dubchak, I. Muchnik, S. Holbrook, and S-H Kim. Prediction of protein folding class using global description of amino acid sequence. *Proc. Natl. Acad. Sci. USA*, 92:8700–8704, September 1995.

[7] Y. Freund and R.E. Schapire. Experiments with a new boosting algorithm. In *13th Intl Conf on Machine Learning*, pages 148–146, 1996.

[8] George H. John and Pat Langley. Estimating continuous distributions in (bayesian) classifiers. In *11th Conf on Uncertainty in AI*, pages 338–345, 1995.

[9] Sreerama K. Murthy. Automatic construction of decision trees from data: A multi-disciplinary survey. *Data Mining and Knowledge Discovery*, 2(4):345–389, 1998.

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

[11] G. Pagallo and D. Haussler. Boolean feature discovery in empirical learning. *Machine Learning*, pages 71–99, 1990.

[12] J.R. Quinlan. *C4.5 : programs for machine learning*. Morgan Kaufmann Publishers, San Mateo, Calif., 1993.

[13] S. Y. M. Shi, P. N. Suganthan, and K. Deb. Multi-class protein fold recognition using multi-objective evolutionary algorithms. In *Proc. IEEE CIBCB*. IEEE, 2004.

[14] A. C. Tan, D. Gilbert, and Y. Deville. Multi-class protein fold classification using a new ensemble machine learning approach. *Genome Informatics*, 14:206–217, 2003.