Automated Protein Classification Using Consensus Decision

Tolga Can*   Orhan Çamoğlu*   Ambuj K. Singh   Yuan-Fang Wang

Department of Computer Science
University of California, Santa Barbara, CA 93106, USA
Tel:+1-805-893-4321   Fax: +1-805-893-8553
{orhan,tcan,ambuj,yfwang}@cs.ucsb.edu

Abstract

We propose a novel technique for automatically generating the SCOP classification of a protein structure with high accuracy. High accuracy is achieved by combining the decisions of multiple methods using the consensus of a committee (or an ensemble) classifier. Our technique is rooted in machine learning that shows that by judicially employing component classifiers, an ensemble classifier can be constructed to outperform its components. We use two sequence- and three structure-comparison tools as component classifiers. Given a protein structure, using the joint hypothesis we first determine if the protein belongs to an existing category (family, superfamily, fold) in the SCOP hierarchy. For the proteins that are predicted as members of the existing categories, we compute their family-, superfamily-, and fold-level classifications using the consensus classifier. We show that we can significantly improve the classification accuracy compared to those of the individual component classifiers. In particular, we achieve error rates that are 3 to 12 times less than the individual classifiers' error rates at the family level, 1.5 to 4.5 times less at the superfamily level, and 1.1 to 2.4 times less at the fold level.

Keywords: protein classification, SCOP

1 Introduction

A global picture of the protein universe is necessary to gain a better understanding of how proteins function. Therefore, during the past several years numerous classification schemes have been developed for defining the relationships—in terms of sequence, structure, and function—of proteins. Protein classification schemes employ different heuristics and similarity metrics, and can be fully [7] or semi-automated [13] or manual [12] schemes. Of these classification schemes, SCOP [12] is created mainly by manual inspection. This is perhaps the reason that it is accepted by many researchers as the most accurate classification scheme (or the ground truth).

With the exponential growth in the number of newly-discovered protein structures, the view of the protein universe is constantly changing. An updated view will provide a better understanding of the functions and relationships among proteins. As time is of essence, one would like to get the most up-to-date view as quickly as possible. SCOP, being built by labor-intensive visual inspection, is updated only once every 6 months. On the other hand, automated classification schemes have the advantage that the view of the protein universe can be updated frequently to include newly-discovered protein structures in a timely manner. Hence, there have been efforts to infer the SCOP classification of a protein structure by using the results of an automated method [4]. However, the validity of such an approach is bound by the accuracy of the method employed. *Our research is hence geared toward providing timely and accurate SCOP classification results in an automated manner.*

Our approach is rooted in consensus building in

---

*These authors contributed equally to this work.

machine learning, which shows that an ensemble classifier with a better performance can be constructed as a committee of component classifiers. A similar idea has been applied to the fold recognition problem by Lundström et al. [9]. By using a neural-network-based consensus predictor they were able to improve the prediction accuracy. However, their goal in fold recognition was to predict the *structure* of a protein sequence rather than its SCOP classification. Furthermore, using only the sequence information is insufficient for ensuring good SCOP classification results, particularly for remote homologs.

The problem of automated generation of SCOP classification has been shown to be a difficult one. In a recent study, Portugaly and Linial estimated the probability of a protein sequence to have a new fold [14]. Of the protein sequences that they assigned a 90% probability to have a new fold, only half of them actually had new folds. In another work, Lindahl and Elofsson compared several sequence-comparison methods for classification of protein sequences at the family, superfamily, and fold levels [8]. They observed that the best-attained accuracy dropped from 75% to 29% from the family to the superfamily level and down further to 15% for the fold level. After analyzing the results of several methods, the authors concluded that a combination of methods may improve the performance. However, they reported such an improvement was not possible as they achieved only limited success.

Nonetheless, our goal is different: prediction of SCOP classification of a newly-discovered protein structure. Both sequence and structure information are available for this task, and we show in this paper that an ensemble classifier can outperform individual components if (1) a correct information-aggregation framework is used and (2) the right component classifiers are employed. In particular, we show that it can be beneficial in using a consensus decision framework, which is grounded in machine learning, with component classifiers that address both sequence and structure information for predicting accurate SCOP classifications for proteins with known structures. We employ a *decision tree* approach to combine

classification decisions made by two sequence-based, and three structure-based classifiers. Given a recently-solved protein structure, we are able to to predict its SCOP classification with high accuracy, using a consensus decision made by these five classifiers. *What is significant is that the boosted accuracy numbers are close to the theoretically maximum performance achievable using such a consensus building framework.*

The remainder of this paper is organized as follows: In Section 2 we define the problem and describe how a sequence/structure similarity search method can be used as a classifier. In Section 3 we review the sequence- and structure-based classifiers we have used in our framework, and show how they do not agree on some cases. In Section 3.2 we analyze the classification performance of individual methods. We show that it is possible to develop a better classifier with higher accuracy if one can find an intelligent way to combine the decisions of individual methods. In Section 4 we propose a method for building a consensus classifier based on the idea of decision tree learning from machine learning. In Section 5 we evaluate the performance of our algorithm using different versions of the SCOP database. Finally we conclude with a brief discussion.

2 Problem Definition

The main questions to be answered when classifying a novel protein structure are:

    i. Does this protein belong to an existing category (family/superfamily/fold) in SCOP hierarchy, or does it need a new category to be defined?

    ii. If this protein belongs to an existing category, what is its classification (label)?

The SCOP database uses a four-level taxonomy: class, fold, superfamily, and family. The top level, *class*, is defined by the number of secondary structures in the proteins and their general layout. Since the class label can be assigned automatically and the assignment does not present a significant challenge, we do not consider it in our classifications. Note that, SCOP assignments are

actually in the *domain* level, however, in our experiments we have used single domain proteins, therefore in the rest of the paper the word *protein* is used instead of *domain*.

At the family level, proteins are assigned to the same SCOP family if they have a high sequence identity ($\geq 30\%$), or they perform similar functions but have a relatively lower sequence similarity ($\geq 15\%$). So the main similarity measure at the family level is sequence similarity. Thus, we would expect that sequence-comparison tools are more appropriate than structure-comparison tools as component classifiers for automated SCOP classification at the family level. At the superfamily level, though, distant similarities are considered. Proteins in the same superfamily probably have evolutionary relationships that are inferred by structural similarities rather than sequence similarities. Hence, we would expect structure-comparison tools to be more successful than sequence-comparison tools at this level. The fold level is the most blurry level of all. At this level, proteins are grouped together not because they show significant similarity, but because they share similarity in the arrangement of their secondary structures. Remote structure similarities are the major similarity metrics at the fold level. Clearly, we would expect structure-comparison tools to outperform sequence-comparison tools at this level.

These three levels of classification are not independent of each other, i.e., it is a hierarchical scheme where each protein is classified into a family which in turn belongs to a superfamily that is a subclassification of the fold category. Assigning a superfamily to a protein which possesses more than a 30% sequence identity to some of the proteins in the database is a trivial task, since we can accurately assign a family to that protein (based on sequence similarity). Superfamily and fold assignments are inherent in the assigned family. In order to extract true performances at different levels, we have to make a complete separation of different levels in the classification, as in [8]. For example, at the superfamily level only the proteins without family-level relationships are queried, and the family/superfamily-level relationships are ignored when evaluating fold-level performance.

For a given query protein structure we proceed to find its SOCP classification in a bottom-up manner. The ensemble classifier first tries to assign a family label to the query protein. If it is successful, the classification process is complete. Otherwise, it is concluded that the new protein needs a new family category to be defined in SCOP, and its superfamily category is sought instead. If a superfamily cannot be assigned, the ensemble classifier proceeds to the final fold level. The protein is predicted to have a new fold, if the ensemble classifier is not able to assign a category in any of the three levels.

2.1 Using a comparison tool as a component classifier

Each component classifier is trained to answer the first question, i.e., if a query protein belongs to an existing category. We train the classifier using the procedure outlined below: For each query protein p in the query set QS, we find the closest protein, i.e., the nearest neighbor, in the database of proteins with a known classification, based on the similarity criteria defined by the comparison tool, T, that the classifier uses. The similarity score assigned by the comparison tool provides a measure of distance between the query protein and the whole database, e.g., a very low score indicates that the query protein does not possess significant similarity to any of the database proteins.

We sort the query proteins using these similarity scores. The goal of the classifier is to partition the query set into two in such a way that one partition contains all the query proteins that belong to existing categories, and the other partition contains all the proteins that need a new family/superfamily/fold label. The partitioning is achieved through a score cutoff. However, a perfect partitioning is usually not possible due to the blurry boundaries of categories. For each tool, we determine the best score cutoff as the one that *maximizes* the number of correct predictions. If the score of the alignment for the query protein is greater than the cutoff, then T is construed to predict that p belongs to an existing category; other-

wise, p should belong to a new category.

For the second problem of label assignment we use the 1NN (first nearest neighbor) classification method. In this method, a comparison tool finds the nearest neighbor to the query protein (most similar protein) in a database, and assigns the query protein the same label (family, superfamily, or fold) as that of its nearest neighbor.

3 Classifiers Used in Our Ensemble Scheme

We use two sequence classifiers. The first one is a model-based sequence comparison method, HMMER [3], for comparing a protein sequence against models of the SUPERFAMILY database [5]. This database is a hidden Markov model (HMM) library representing all proteins of known structures. It is manually curated to classify proteins at the SCOP *superfamily* level. HMMER assigns a similarity score to the sequence according to its match with a model. In the rest of the paper, we will refer to this method as HMM.

The second sequence classifier we use is PSI-Blast [1]. PSI-Blast is an improved version of Blast that works in iterations. In the first iteration, Blast is run and a new scoring schema is created based on the set of close neighbors. In the ensuing iterations this new scoring scheme is used and updated as new close neighbors are found.

We use three structure classifiers. The first structure classifier, CE [16], is a pairwise structural alignment tool. It uses inter-atomic distances of C_α atoms to find small (8 residues), geometrically-similar fragments from the pair of protein chains. Then, CE combines these fragments to form longer matches.

We also choose Dali [6], a structure-similarity comparison tool, as one of our classifiers. Dali computes the distance matrices of two proteins and then finds the alignment by a Monte Carlo algorithm using these two matrices. Dali is also used to create FSSP [7], an automated protein classification database.

The final classifier we have chosen is Vast [10]. It is designed for identifying remote homologies. It uses secondary structure elements to
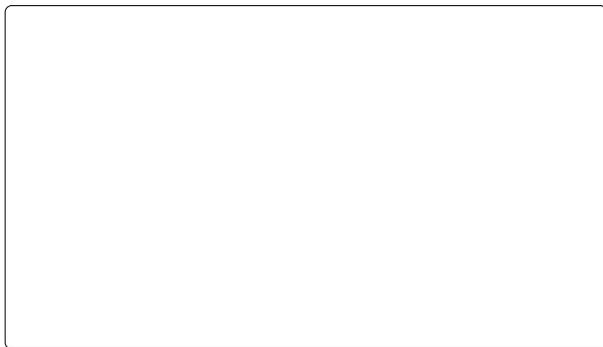

Figure 1. Comparison of HMM and Vast scores. Each point represents a pair of proteins. The coordinates of the points are the scores assigned by the tools to the pairs.

locate initial matches. These properties distinguish it from other structure-comparison tools, since Vast prefers small biologically meaningful matches over optimal global alignment.

3.1 Relationship between the classifiers

Each sequence- and structure-comparison tool described in the previous section assigns a score for a pair of proteins, that indicates the statistical significance of the similarity between them. In particular, we have used the z-scores reported by CE and DALI, p-values reported by VAST, and e-values ($-log$(e-value)) reported by HMM and PSI-Blast as similarity scores. Below, we explore the correlation between the scores of different methods. Figure 1 shows the correlation between the scores of HMM and Vast. Each data point in the 2D plot represents a pair of proteins, and the point coordinates are the comparison scores from HMM and Vast, respectively. A perfect correlation would consist of entries on a straight line. In this case, we see quite a bit of disagreement. This is expected since we are comparing the scores of a sequence-alignment method with those of a structure-alignment method. However, Figures 2 and 3 show that there is considerable disagreement between two structure- or two sequence-alignment methods as well. In a similar study, Shindyalov and Bourne [17] compared CE and Dali scores and showed that there were many proteins that were found similar by CE and dissimilar by Dali, and vice versa. These results show that an intelligent mechanism is needed to resolve the
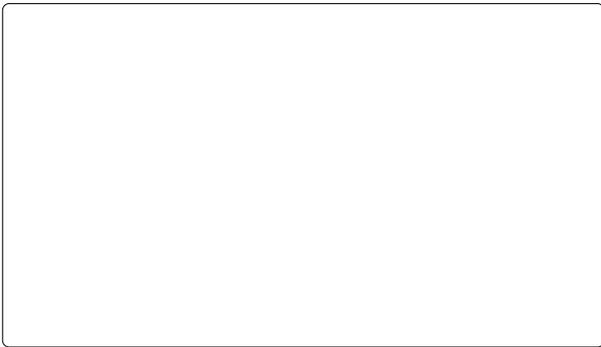
4

Figure 2. Comparison of HMM and PSI-Blast scores.

Figure 3. Comparison of Vast and Dali scores.

inconsistency of these methods. Our solution to this dilemma is to reconcile the discrepancies between the classification tools by combining them using a consensus decision framework.

3.2 Classification performance of individual classifiers

We performed a number of experiments to understand the individual performance (accuracy) of the tools when they are used as component classifiers. In these experiments we assumed classifications of all the proteins in SCOP v1.59 ($DS159$) are known. We then classify the new proteins introduced in SCOP v1.61 ($QS161$) into families, superfamilies, and folds by using structure- and sequence-comparison tools. As described in Section 2, to get the true performances of classifiers at different levels, we need to separate the query set based on levels of similarity. At the family level all new proteins introduced, ($QS161$), are queried. At the superfamily level, only proteins, ($QS161_{newFam}$), that do not have family-level similarities are queried. And at the fold level, only proteins ($QS161_{newSF}$) that do not have family/superfamily-level similarities are queried.

3.2.1 Recognition of members of existing categories

The performance results for each tool are shown in Figure 4. At the family level, we can clearly see that the sequence tools outperform the structure tools by achieving 94.5% and 92.6% accuracy. The success rate of the structure tools is only 89%.

We then consider if it is possible to manage higher success rates by combining the tools. The sixth column of the figure shows the aggregate accuracy of the tools. For 83% of the query proteins, all five tools make correct decisions (as to whether the query protein belongs to a category), for 4.1% of the queries 4 tools, for 1.9% of the queries 3 tools, for 6.9% of the queries 2 tools, and for 2.7% of the queries only one tool makes the correct decision. An interesting point here is that for 98.2% of the proteins, at least one tool is successful. So, it is theoretically possible to classify up to 98.2% of these proteins correctly by combining the results of the individual tools.

At the superfamily level, the structure tools match or better the performance of sequence tools, as expected. However, the overall performance of the tools drops significantly from the family level. This is expected since classification at the family level is the easiest. HMM has one of the best performances, and this is no surprise considering that HMM was manually tuned for superfamily classifications. Among the structure tools, Vast has the best performance with a 78.6% success rate. PSI-Blast performs very poorly with a success rate of only 66.1%. Only 44.7% of the queries can be classified correctly by all five tools. For 4% of the queries, none of them is successful.

Structure tools clearly outperform sequence tools at the fold level. Vast has the best performance with a 85% success rate. PSI-Blast again has the worst performance with a 60.7% success rate. For only 30.9% of the queries, all five tools are successful. An interesting aspect is that all the query proteins are classified correctly by at least one tool. This again raises the possibility of achieving better accuracy through a combination of the results.
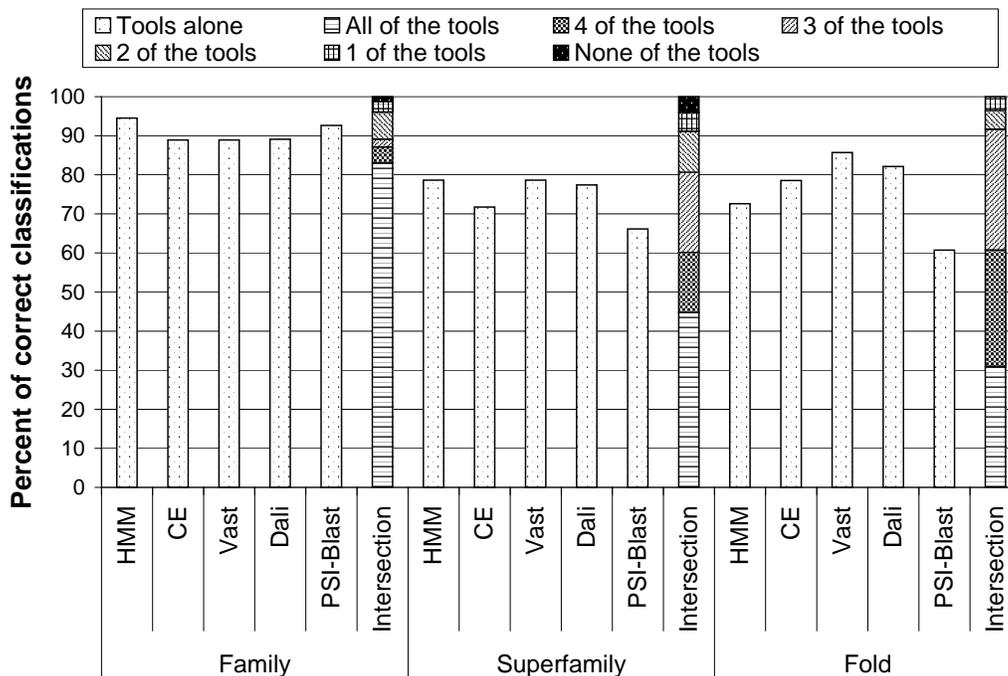
5

Figure 4. Performance of individual classifiers on recognizing members of existing families, superfamilies and folds for the new proteins between SCOP v.159 and SCOP v.161.

3.2.2 Classification assignment

Once a tool has marked a new protein as a member of an existing category, the classification of query protein is complete, i.e., the query protein is assigned to the same category as its nearest neighbor. The next question is to judge the accuracy of this assignment, i.e., whether the assigned category is the correct one. The accuracy results for the different tools are shown in Figure 5. The results are reported for the list of query proteins that are known to be members of existing families, superfamilies, and folds at each level respectively.

The accuracies of the tools are high at the family level. All except Dali have success rates above 90%. HMM has the best performance with 94.8% accuracy and is followed by Blast with 92.3% accuracy. For 76.5% of the queries, all five tools are able to assign the correct family label. For only 2.1% of the queries, none of them is successful.

At the superfamily level, the structure tools perform better and achieve 80.4-81.7% success rates. The poor performance of sequence tools is surprising since sequence similarities still matter at this level. HMM, especially, is expected to perform well, since it is manually tuned for this level. For 6.1% of the queries, none of tools is able to assign the correct superfamily label. This means one can potentially achieve a theoretically maximum 93.9% success rate by combining the results of these five tools. This is quite high compared to the performance of the best tool, 81.7%.

At the fold level, all tools seem to perform poorly. Note that the number of test proteins decreases significantly at this level (37 proteins), because the proteins with family/superfamily relationships are discarded. Therefore, wrong classification of even a single protein has a strong effect (3%) on the accuracy results. At the fold level, PSI-Blast is not able to make even one correct fold assignment whereas Vast assigns correct folds to 54% of the queries. For 35.1% of the queries, none of the tools is able to assign the correct fold label. This high failure rate suggests noise from similar folds, and is investigated in Section 5.3.
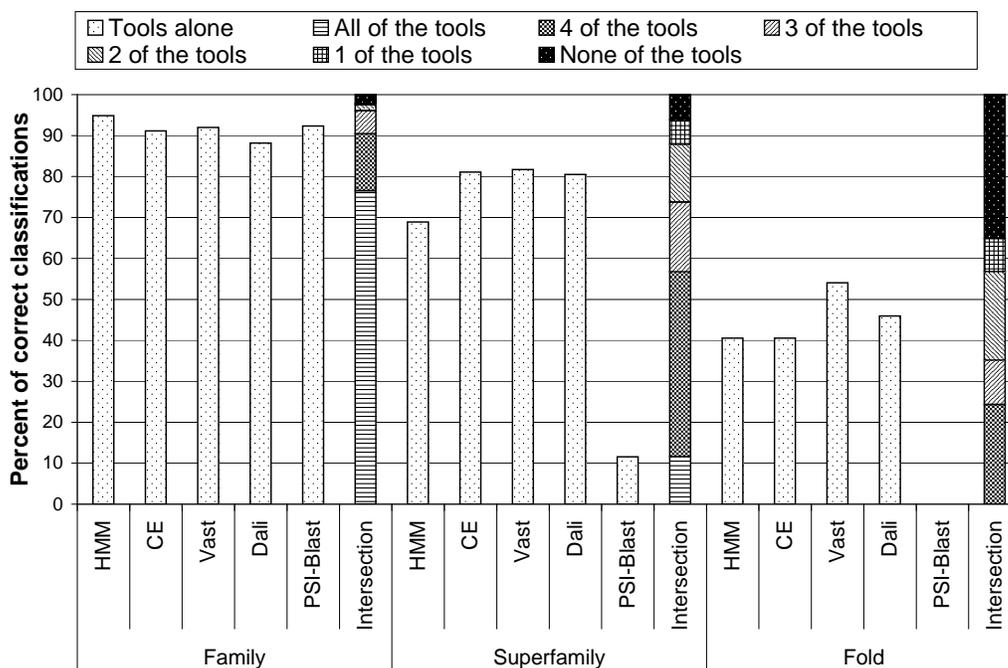
6

Figure 5. Performance of individual classifiers on assigning families, superfamilies and folds to the new proteins between SCOP v.159 and SCOP v.161.

4 Automated Classification Using Ensemble Classifier

As evident from our initial experiments, an ensemble classifier can potentially obtain higher classification accuracy than any single component classifier. There are many studies in the area of machine learning and pattern recognition that address the intelligent design of ensemble classifiers [2, 15, 11]. These include both competitive models (e.g., bagging and arcing) and collaborative models (e.g., boosting). Our method employs a hierarchical decision tree to answer the question whether the query protein belongs to an existing category. If the answer is yes, we then use Bayesian decision rules to assign the protein an appropriate label.

Since our goal is to classify protein structures at three levels of hierarchy in SCOP, a hierarchical classification algorithm should be more appropriate. An overview of the general algorithm is given in Figure 6. Here, parameter p is the protein whose classification is sought and R is the set of classification rules found during training. The first check is to see if there is a protein in the database that shares the same family as p. If so,

---

Input: p is the query protein for classification, R is set of rules.
Algorithm FIND-CLASSIF(p, R)

If HAS-CLASSIF$(p, R, family)$
      return ASSIGN-CLASSIF$(p, R, family)$
else HAS-CLASSIF$(p, R, superfamily)$
      return ASSIGN-CLASSIF$(p, R, superfamily)$
else HAS-CLASSIF$(p, R, fold)$
      return ASSIGN-CLASSIF$(p, R, fold)$
else return *New-Fold*

Figure 6. Overview of classification algorithm.

then the consensus family label is assigned to p. If p is not a member of an existing family, then p is checked to see if it shares a superfamily with one of the proteins in the database. Depending on the answer, either a consensus superfamily label is assigned to p, or fold-level similarity is verified. If p cannot be assigned to a fold, then p is considered a member of a new fold.

4.1 Normalization of similarity scores

In order to combine the results from different tools, we need to find a way to normalize their results into a consistent scale. We achieve this through *binning*. A bin is a *tool-neutral* accuracy extent, e.g., 90%-100%, 80%-100%, instead of a

tool-specific similarity score. The bins are manually crafted to obtain the best spatial resolution and every classification tool produces its own set of bins. The procedure used is as follows: We use $DS159$ as the database and $QS161$ as the query set (See Table 2). For each protein in $QS161$, we record the top similarity score in $DS159$ (nearest neighbor) using a particular comparison tool (or a component classifier). Performing this operation for all the proteins in $QS161$ results in a histogram indexed by the similarity score used by the tool. We then sort the scores and scan, from high score (or similarity) to low score (or similarity), to decide on the bin boundaries. The bins partition the score space of each tool into buckets, each containing roughly the same number of proteins. The kth such bin corresponding to tool i is called E_{ik} (E stands for *Existing*).

We also construct a dual set of bins for each tool from the lower end of the score space, based on the tool's accuracy of predicting that the protein is new (i.e., does not belong to an existing category). The kth such bin corresponding to tool i is called N_{ik} (N stands for *New*). Note that the above bin construction is repeated at each of the three levels: family, superfamily, and fold.

Given a query protein, the score computed by a classification tool can be used to map it to a pair of E and N bins, say (e, n). The classification tool assigns a certain confidence level to this pair of bins based on the training set. This process is repeated for each tool, resulting in a vector of confidence levels. Next, we discuss how the vector of values is combined together through decision tree rules and weighted Bayesian rules.

4.2 Recognition of new categories using decision trees

The rule set R used in Figure 6 affects the performance of the classification algorithm. Intuitively speaking, if rules in R can assume very general and complicated forms, then one can expect an increased likelihood of finding a rule that generates good classification results. However, complicated rules with many tunable parameters result in large pools of candidate rules, which increase training time and size of the training sam-

ples needed. This general trade-off, often referred to as the bias-variance trade-off [2], is well understood in the machine learning community. It states that flexible, general rules provide better (small bias), but less predictable (large variance), classification results. The danger of over-fitting is also much higher with flexible rules.

In our application, we have five component classifiers whose results can be combined to form rules in R. The decision of each component classifier can be considered as an attribute of the query protein. Using the decision tree approach, the final classification decision about the protein is made by consulting a set of attributes in a hierarchical manner.

However, even with only five component classifiers, the number of different decision trees one can construct is huge. There are several parameters that affect the decision tree creation process. The first one is the branching factor at each node of the tree. That is, the decision consulted at each node may *split* the training data into variable numbers of subsets.

Fortunately, for the problem of SCOP classification by combining sequence/structure comparison tools, a biologically sound decision can be made about the branching factor at each node. That is, at each level by consulting the confidence levels, we can split the training data into *three*: one partition being query proteins that belong to existing categories (because of strong evidence of similarity), another partition being query proteins that are new (because of strong evidence of dissimilarity), and the other partition being query proteins that are in the twilight zone. This observation limits the search space to decision trees of the structure seen in Figure 7, where C_i is the combined confidence level of a number of tools at level i, Th_is are the confidence thresholds used to split the training data, and L_1 and L_2 are the labels for new family/superfamily/fold, and existing family/superfamily/fold categories respectively.

We further restrict the tree structure to be at most three levels because of the small number of component classifiers used and the desire to expedite the training process. Even with this fixed decision-tree structure, there are two important
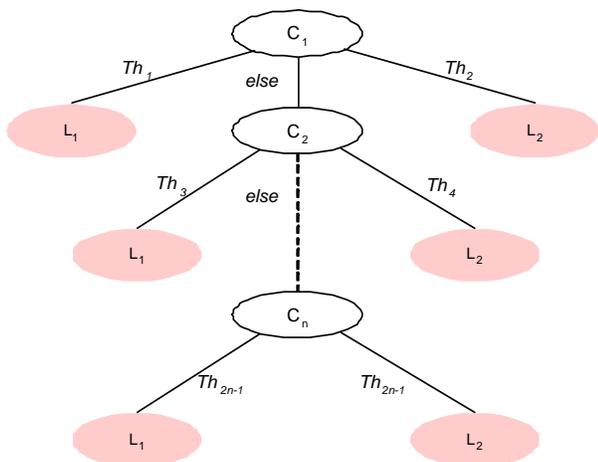
Figure 7. The general structure of the decision trees suitable for protein classification purposes.

sets of parameters that should be determined, C_is and Th_is. These parameters are the most crucial components of the decision tree since the classification accuracies vary remarkably for different choices.

C_is can assume many different forms, and some popular ones are the sum, product, max, and min rules, which compute the sum, product, max, and min of the component scores, respectively. The search space is exceedingly huge without some restriction on the form that C_is can assume. In our experiment, we consider only the sum rules, i.e., the decision is based on the weighted sum of the confidence levels of component classifiers. To further simplify the analysis, each C_i uses the confidence levels of up to three components and always weighs the component scores equally.

We generated all possible trees with C_is composed of confidence levels of at most three tools. There are $15,600$ such trees. The accuracy of each tree is determined by examining the proteins at the leaf nodes. The best accuracy is achieved if all the proteins that are labeled as L_1 are actually new proteins, and all the proteins labeled as L_2 actually belong to existing categories. The erroneously labeled proteins at leaf nodes decrease the accuracy. In other words, to achieve high accuracies we want the leaf nodes to be as pure as possible, containing only proteins sharing existing labels or needing new labels, both not both. In

the machine learning community, this is referred to as minimizing the impurity. Various impurity functions can be used [2]. Most commonly used impurity measure, *entropy impurity* is defined as:

$$i(N) = - \sum_j P(w_j) \log_2 P(w_j), \qquad (1)$$

where $P(w_j)$ is the fraction of patterns at node N that are in category w_j.[1]

To search for the best values for Th_i for each tree, we can consider a global optimization by minimizing the overall impurity (summation of the impurities at each leaf node). However, this is a six dimensional optimization problem (to determine the six thresholds in Figure 7) for each of the $15,600$ trees, which is not feasible. Instead, we can use a greedy approach that does not guarantee optimality but provides efficiency.

In our approach, the best thresholds at a level are determined by examining only the leaf nodes at that level. However, we need to be careful in this strategy, because trying to minimize the impurities usually produce trees with bad accuracy. The reason is that at the expense of achieving low impurity at a certain level, only decisions for a small portion of the training data are made, and a large portion is passed onto the next level. This causes most of the decisions to be made at the bottom of the tree where most of erroneous labeling occurs. To overcome this problem, we allow a certain amount of impurity at each level for the benefit of reducing the number of decisions to be made at the next level.

Using this greedy approach, we could generate very accurate ensemble classifiers that perform much better on the training data than the component classifiers used. However, when we tested these decision trees on the test data set, their performance dropped drastically. The main reason, we suspect, is that the automated threshold finding procedure tends to over-fit the training data. We believe that this again can be attributed

---

[1]In our case, there are two categories: one corresponds to those proteins that can be assigned an existing label and the other corresponds to the proteins that are in need of a new label.
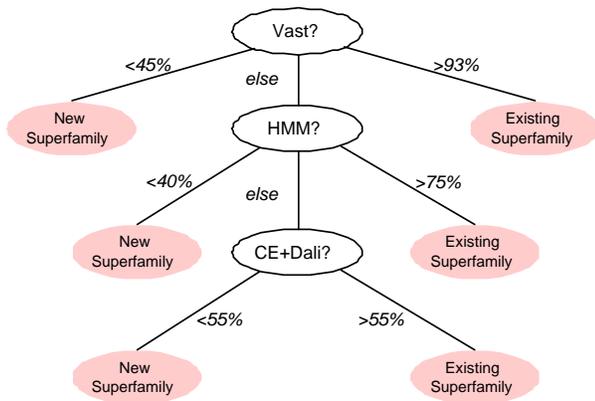
Figure 8. The Decision Tree for recognition of proteins that belong to existing superfamilies.

to the bias-variance tradeoff. I.e., a decision tree with many tunable parameters can perform well on training data, but not on validation data.

As a solution to the over-fitting problem, we tried to construct decision trees manually based on lessons learned from Section 3 on the strength and applicability of the component classifiers. First, the C_is are determined by referring to Figure 4. The performance of each tool is evaluated and the number of proteins for which tools make the right decisions whereas all other tools fail is considered. The tools with the highest performance are chosen to perform at the first level. After the proteins for which the first level tools are able to make decisions on are removed from the query list, the performances of the remaining tools are reconsidered. The single tool or a combination of tools, that perform the rest for the remaining query proteins, is selected to perform on the second level. For the third and the final level, the tool combination that can best classify the proteins, that cannot be classified at levels one and two, is chosen.

After the tool combinations that perform at each level are decided, the confidence thresholds for decisions, Th_i, are chosen. The most important benefit of manual decision tree construction surfaces in choosing these thresholds. First of all, the thresholds should be placed between protein clusters that are statistically significant, meaning that there should be a significant numerical difference between the probabilities above

and below the threshold. Thus, thresholds that are placed between groups of scores are preferred over the thresholds that will give lower error percentages. As a result, over-fitting of the training data is avoided. The second important aspect is the knowledge of the performance of the tools that will run at each level. The tools on the higher levels are allowed to make more errors in their decisions, because the tools on the lower levels perform worse than the ones on the higher levels. So, each statistically significant threshold is tested, and the performances of the tools on the lower level are evaluated. If their performances are not satisfactory another threshold for the higher level that reduces the number proteins which will be passed to the lower level is tested. Thus, a look-ahead procedure is used in decision of the Th_is. Figure 8 shows the manually constructed decision tree for recognition of proteins that belong to an existing superfamily.

The manually constructed decision trees vary with the taxonomy level. When deciding if a protein is from an existing family, we give priority to the sequence tools. So, we first assign a family label based on HMM and PSI-Blast. For our training set, these two tools together are able to assign 90% of the queries confidently. For the remaining proteins, we use the structure tools. An interesting point here is that if sequence tools are unable to find a significant match but structure tools find a significant match, then the protein likely belongs to a new family.

At the superfamily level, Vast and HMM perform the best, as seen from Figure 4. So, they have the top priorities. We first use Vast, and then use HMM on the proteins that Vast cannot predict with confidence. Finally, on the twilight zone of these two tools, we apply Dali and CE. At the fold level, structure tools perform better. So, we use the structure tools in the order of Vast, CE, and Dali. The complete set of decision tree rules is shown in Table 1.

4.3 Classification assignment for members of existing categories

After deciding that a protein is a member of one of the existing categories, we assign its clas-

	Level 1	**Range**	**Level 2**	**Range**	**Level 3**	**Threshold**
Family	HMM+Blast	(60%:95%)	CE+Vast+Dali	(0%:70%) (85%:99%)	HMM+Blast	85%
Superfamily	Vast	(45%:93%)	HMM	(40%:75%)	CE+Dali	55%
Fold	Vast	(50%:85%)	CE	(80%:90%)	Dali	60%

Table 1. Heuristic decision tree rules for recognition of members of existing categories. At each level, a combination of tools is run and the probability of being a member of an existing category is assigned to each protein. The proteins that have probabilities higher than the indicated range are assigned to the predicted category, the ones within the range are passed to the next step, and those below the range are deemed new. For the last level, only a single threshold exists.

sification. The assignment is done by using a weighted Bayesian rule, in which the weights for the components classifiers are determined according to their training accuracies. Each tool assigns the query protein to a category with a certain confidence. These categories are compared and the query protein is assigned to the category that gets the highest probability. We consider the reliability of each tool to weigh its contribution to the consensus decision. Figure 5 provides us this information. When assigning family labels, sequence tools are more reliable than structure tools, and when assigning superfamilies and folds, structure tools are more important than sequence tools.

5 Experimental Evaluation

To validate that consensus decision indeed improves classification performance, we apply the standard validation technique in pattern recognition [2]. Two data sets are used: a training set and a test set. In each case, we need a set of database proteins (proteins of known classification) and query proteins (proteins to be assigned an existing fold/superfamily/family label or a new label). We choose three versions of the SCOP database [12] in our experiments, thus also reflecting the expansion of the protein structure universe. SCOP version 1.59 ($DS159$, May 2002) and version 1.61 (December 2002) are used to generate the training set. The training query set is generated by extracting the protein chains from version 1.61 that were introduced after version 1.59, $QS161$. After training our method using $DS159$ and $QS161$, we apply the same strategy to SCOP versions 1.61 and 1.63 (May 2003) to evaluate the performance of our algorithm. Table 2 shows the number of pro-
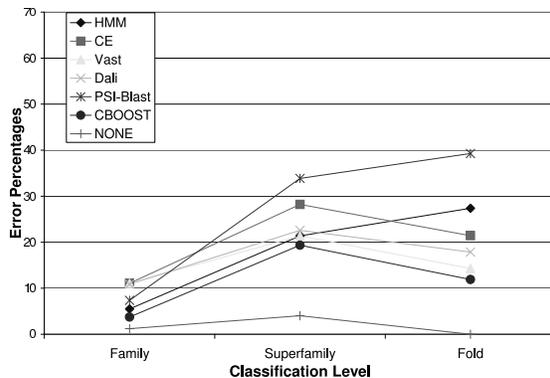

Figure 9. Performance of individual classifiers compared to the ensemble on recognizing members of existing families, superfamilies and folds for the new proteins between SCOP v.159 and SCOP v.161.

tein chains used for training and validation. As seen in the table, the distributions of introduced superfamilies and folds changed dramatically between versions 1.61 and 1.63. This change poses a challenge to our ensemble classifier, since the generated classification rules depend on the training set. However, as the results of our experiments show below, our algorithm performs well in this real-life setting and the difference between the classification accuracy of training and classification accuracy of evaluation tests is in agreement with the difference between training error and generalization error observed in other pattern recognition contexts [2].

5.1 Training Procedure

The ensemble classifier is trained in a similar hierarchical manner described in Section 3. The first case is to recognize if a new protein has an existing classification. We train the ensemble classifier, i.e., produce the decision tree rules, to
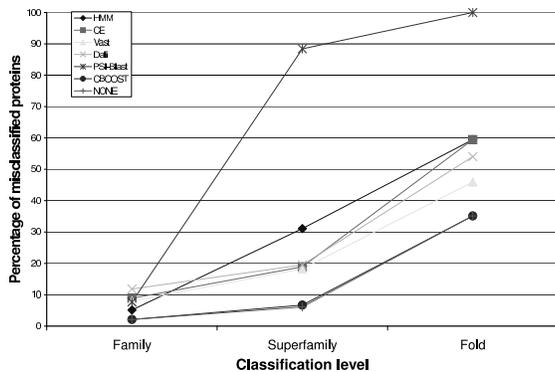
Figure 10. Performance of individual classifiers compared to the ensemble classifier on assigning families, superfamilies and folds to the new proteins between SCOP v.159 and SCOP v.161

	Training	Evaluation
Database	$DS159$ (20449)	$DS161$ (22724)
Query	$QS161$ (2241)	$QS163$ (2825)
$newFam$	248	618
$newSF$	84	424
$newFold$	47	339

Table 2. Database and query data sets and their sizes
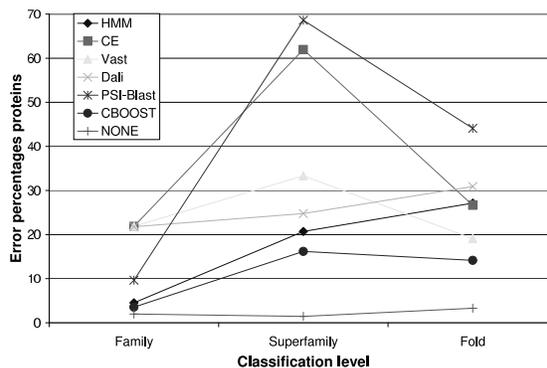

Figure 11. Performance of individual classifiers compared to the ensemble on recognizing members of existing families, superfamilies and folds for the new proteins between SCOP v.161 and SCOP v.163.

perform best on the database $DS159$, proteins in SCOP 1.59, with the queries introduced in 1.61, $QS161$. Figure 9 depicts the comparison of the ensemble with the five component classifiers. At the family level, the best performance among the tools is achieved by HMM with a 5.5% error rate. the ensemble is able to reduce this error rate to 3.7%. Figure 4 shows that for 1.2 % of the queries none of the tools find the correct classification. So, one cannot reduce this error rate below 1.2%, which can be accepted as the theoretical limit. The ensemble manages to outperform HMM 1.5 times, PSI-Blast 2 times, and the structure tools more than 3 times. The ensemble is successful for 96.3% of the proteins.

At the superfamily level the ensemble classifier is not as close to the theoretical limit as it was for the family level. Yet, it still improves the performance of the individual tools. Performance improvements are between 1.4 to 2.9 times. the ensemble is successful for 86.1% of the queries. In Figure 4, the overlap between the correctly-identified proteins by the tools is minimum at the fold level. Figure 9 shows that the ensemble classifier outperforms individual tools from 1.1 to 1.4 times by being successful for 89% of the queries.

The second part of training is the assignment of an existing class label to the new protein. Figure 9 shows that the ensemble classifier outperforms all the component tools. At the family level, none of the tools is successful for 2.1% of the queries,

as can be seen in Figure 5. The ensemble classifier performs at this theoretical limit (97.9% of the proteins). The improvement at the family level is between 2.4 to 5.6 times. At the superfamily level, the ensemble classifier performs almost at the theoretical limit. It has an error rate of 6.7% whereas the theoretical limit is 6.1%. As a result, the performance of the tools has been improved 2.7 to 13 times. At the fold level, the ensemble has an error rate of 35% which is the same as the theoretical limit. Performance of the individual tools has been improved 1.3 to 2.8 times, and 65% of the proteins are assigned to the correct folds.

5.2 Validation Procedure

When we test our algorithm by using queries from $QS163$ on the database $DS161$, we see that the ensemble improves the performance as in Figure 9. None of the tools is able to perform better than ensemble on all levels. Although HMM performs close to the ensemble classifier at the family level, it fails to compete at the superfamily and the fold levels. The ensemble outperforms HMM by 1.3 times at the family and superfamily lev-
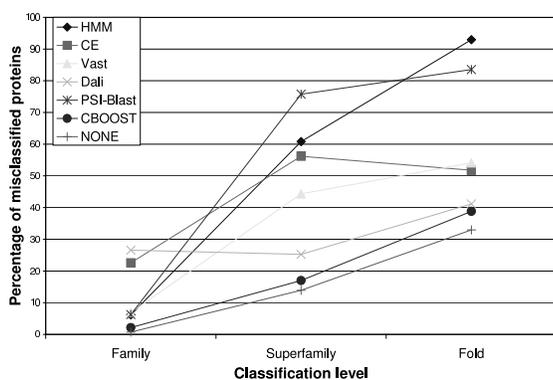
12

Figure 12. Performance of individual classifiers compared to the ensemble on assigning families, superfamilies and folds to the new proteins between SCOP v.161 and SCOP v.163

els. At the fold level the ensemble outperforms the best tool, Vast, by 1.4 times. The ensemble also manages high accuracies at all levels; 96.5% for family, 83.8% for superfamily, and 86% for fold levels.

After training the ensemble on the $DS159$ and $QS161$, we test it with the trained parameters on $DS161$ and $QS163$. By comparing Figure 10 and Figure 12, we can see the effect of improvement. We trained the ensemble classifier to perform as close to the theoretical limit. When we tested it, it performs slightly worse than the theoretical limit, but significantly better than the individual tools. The improvement at the family level is 3 to 12 times, at the superfamily level 1.5 to 4.5 times, and at the fold level 1.1 to 2.4 times. It assigns 97.9% of proteins to correct families, and 83% of the proteins to the correct superfamilies, and 61.2% of the queries to the correct folds.

5.3 Error analysis

We have analyzed the query proteins where the ensemble fails to find the correct classifications. Most of these cases are due to factors that depend on human judgement rather than on computational results using sequence and structural similarity. Below, we present a few of such cases.

The protein structure 1kuu-A, an $\alpha+\beta$ class protein, is predicted by the ensemble to be a member of an existing superfamily *N-terminal nucleophile aminohydrolases*. It is very similar to the members of its predicted superfamily, according

to both sequence and structure classifiers. But, it is actually a member of *Hypothetical protein MTH1020* superfamily. The only difference between these two superfamilies is that the latter lacks the N-terminal nucleophile. Clearly SCOP authors decided to create a new superfamily based on information that cannot be inferred by automated classifiers. The false hits between these two superfamilies contribute to significant portion of the mistakes of the ensemble classifier at the superfamily assignments.

For proteins in *5-bladed beta-propeller* fold, classifiers get false hits from other structurally similar folds. CE and Dali assign proteins from this fold to *6-bladed beta-propeller* fold, and Vast assigns them to *4-bladed beta-propeller* fold. Since these folds have a similar layout, their members are prone to get short but high scoring alignments across classes. This is an example of the Russian Doll effect [13].

6 Conclusion

The most trusted protein classification databases are the manually-curated ones. However, as new protein structures are continuously being discovered, there is a need to automatically update protein classification databases *timely* and *accurately* to account for the new structures. In this paper, we propose a novel method that uses the consensus decision principle, a machine-learning technique, to obtain higher quality classifications of manually curated databases from automated techniques. Our technique achieves 98% success for family assignments, 87% success for superfamily assignments, and 61% success for fold assignments.

References

[1] S. F. Altschul and E. V. Koonin. Iterated profile searches with PSI-BLAST – a tool for discovery in protein databases. *Trends Biochem Sci.*, 23(11):444–7, 1998.

[2] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience, second edition, 2001.

[3] S. R. Eddy. Profile hidden markov models. *Bioinformatics*, 14:755–763, 1998.

[4] G. Getz, M. Vendruscolo, D. Sachs, and E. Domany. Automated assignment of SCOP and CATH protein structure classifications from FSSP scores. *Proteins*,

46:405–415, 2002.

[5] J. Gough. The SUPERFAMILY database in structural genomics. *Acta Cryst.*, D58:1897–1900, 2002.

[6] L. Holm and C. Sander. Protein structure comparison by alignment of distance matrices. *Journal of Molecular Biology*, 233:123–138, 1993.

[7] L. Holm and C. Sander. Mapping protein universe. *Science*, pages 273:595–602, 1996.

[8] E. Lindahl and A. Eloffson. Identification of related proteins on family, superfamily and fold level. *J. Mol. Biol*, 295:613–625, 2000.

[9] J. Lundstrom, L. Rychlewski, J. Bujnicki, and A. Elofsson. Pcons: A neural-network-based consensus predictor that improves fold recognition. *Prot. Sci*, 10:2354–2362, 2001.

[10] T. Madej, J.-F. Gibrat, and S.H. Bryant. Threading a database of protein cores. *Proteins*, 23:356–369, 1995.

[11] R. Meir and G. Ratsch. *Advanced Lectures on Machine Learning*, chapter An introduction to boosting and leveraging. Springer Verlag, 2003.

[12] M. A. 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.

[13] C. A. Orengo, A. D. Michie, 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.

[14] E. Portugaly and M. Linial. Estimating the probability for a protein to have a new fold: A statistical computational model. *Proc. Natl. Acad. Sci.*, 97(10):5161–5166, May 2000.

[15] R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.

[16] H.N. Shindyalov and P.E. Bourne. Protein structure alignment by incremental combinatorial extension (CE) of the optimal path. *Protein Engineering*, 11(9):739–747, 1998.

[17] I. N. Shindyalov and P. E. Bourne. An alternative view of the protein fold space. *Proteins: Structure, Function, and Genetics*, 38:247–260, 2000.