Boosting methods for Protein Fold Recognition: An Empirical Comparison

Yazhene Krishnaraj and Chandan K. Reddy
Department of Computer Science
Wayne State University, Detroit, MI - 48202.

Abstract

Protein fold recognition is the prediction of protein's tertiary structure (Fold) given the protein's sequence without relying on sequence similarity. Using machine learning techniques for protein fold recognition, most of the state-of-the-art research has focused on more traditional algorithms such as Support Vector Machines (SVM), K-Nearest Neighbor (KNN) and Neural Networks (NN). In this paper, we present an empirical study of two variants of Boosting algorithms - AdaBoost and LogitBoost for the problem of fold recognition. Prediction accuracy is measured on a dataset with proteins from 27 most populated folds from the SCOP database, and is compared with results from other literature using SVM, KNN and NN algorithms on the same dataset. Overall, Boosting methods achieve 60% fold recognition accuracy on an independent test protein dataset which is the highest prediction achieved when compared with the accuracy values obtained with other methods proposed in the literature. Boosting algorithms have the potential to build efficient classification models in a very fast manner.

1. Introduction

Proteins regulate functioning of a cell and also carry out many tasks that are essential to life. There are four distinct aspects of a protein's structure: Primary structure, Secondary structure, Tertiary structure and Quaternary structure. Tertiary Structure (henceforth referenced as Fold) of a protein is the three dimensional structure formed by secondary structure folding back upon itself and it determines the function of the protein.

Experimental and computational methods are two well-known methods used for the prediction of protein structure [8]. Experimental methods such as X-ray crystallography and Nuclear Magnetic Resonance (NMR) spectroscopy are not only expensive but also time-consuming, whereas computational methods are efficient, less expensive and also provide more insights into protein's function and evolutionary origins. Computational methods used to solve this problem can be broadly categorized into four different groups namely [8]: (1) *Homology Modeling* [7], where given a protein sequence, its structure is assigned using sequence similarity. It assumes that two proteins have the same structure if they have high sequence homology. (2) *Threading* [6], which detects the structural similarities by comparison with a library of known structures even for low similarity protein sequences. (3) The *De Novo Approach* [11], where prediction is done only from the sequence alone with physics and chemistry laws. With the use of suitable energy function, protein folding is simulated in atomic detail using methods like molecular dynamics or Monte Carlo simulations. (4) The *machine learning techniques* where algorithms learn from training data to build a classifier which will identify the three-dimensional fold of a protein using classification methods from its sequence information.

Considering the importance of the protein structure in proteomics and the challenges in building more complex classification models to solve the structure, we are surprised to see that the boosting approach is not thoroughly investigated in the literature for protein fold recognition problem. This paper provides a comprehensive study and discusses the application of boosting algorithms to the problem of protein fold recognition. The boosting meta-algorithm is an efficient, simple, and easy to manipulate machine learning technique that can potentially use any weak learner available. In simple terms, boosting algorithms combine weak learning models that are slightly better than random models.

2. Machine Learning Methods

Researchers used different machine learning algorithms for the problem of protein fold recognition. In this section, we will describe the existing methods proposed in the literature, and these include Support Vector Machine, Neural Networks and K-Nearest Neighbor.

2.1. Support Vector Machines

Support Vector Machine (SVM) is a popular classification algorithm based on statistical learning theory developed by Vapnik and his colleagues [12] at Bell Laboratories, and has been improved by others since then. SVM creates a decision boundary by mapping input vectors to a high-dimensional space where maximum-margin linear hyperplane separates different classes in the training data. Ding and Dubchak [2] used One-versus-Others (OvO) method, unique One-versus-Others (uOvO) method and All-versus-All (AvA) method to handle multiple classes. If there are K classes, OvO method partitions K classes into a two-class problem by having proteins from one fold as '*true*' class and the rest of the proteins are grouped into '*others*' class [2]. K two-way classifiers are trained by repeating this procedure for each of the K classes. When a new test protein comes in, all the K classifiers are used to predict the protein fold which thus yields K results. More than one classifier might predict the protein to a positive class because of the '*False Positive*' problem. To overcome this, they introduced uOvO method which is an extension of OvO where the classifiers with positive predictions are considered in the second step. A series of two-way classifiers are trained for each of the pairs and voting is done to predict the final fold. AvA method [2] constructs $K(K\text{-}1)/2$ number of two-way classifiers and voting is done to decide the protein fold.

In the paper by Huang and his colleagues [5], SVM is used as a base classifier for their Hierarchical Learning Architecture (HLA). HLA is a network with two levels. In the first level, protein is classified into one of its four major classes. Second level of the network further classifies the proteins into 27 folds. They adapted multi-class SVM classifier from previous studies [9].

2.2. Neural Networks

Neural Networks (NN) is a widely used algorithm in many applications and is modeled based on biological neural systems. For the classification task, the model is trained in such a way that the input and output nodes are connected with weighted links based on input-output association of the training data. Ding and Dubchak [2] used a three-layer feed-forward NNs in their experiments. In the NN architecture that they used, the number of input nodes were same as the number of feature vectors with one hidden node and two output nodes (one for '*true*' class to indicate one protein fold and the '*Other*' for all the other protein folds). Huang and his colleagues [5] used three different NN models as base classifiers for their HLA. First, Multi-layer Perceptron (MLP), a feed forward NN with three hidden layers was used. Second, Radial Basis Function Network (RBFN), a three layer NN model with one hidden layer was used. Finally, General Regression Neural Networks (GRNN), a four layer NN was used with only one hidden layer.

2.3. K-Nearest Neighbors

K-Nearest Neighbors (KNN) is an instance based learning algorithm where the modeling of the classifier is deferred until the classification of the test data. The training data with n input attributes represented in an n-dimensional space. When a test object needs to be classified, proximity measure (like Euclidean distance) is used to find K nearest neighbors and voting is taken to assign the class. Paper by Okun [10] uses K-Local Hyperplane Distance Nearest Neighbor Algorithm (HKNN) [13]. This algorithm forms a linear local hyperplane for each class in the data set. When a test data comes in, distance between the test point and these local hyperplanes are calculated to decide the class.

3. Boosting Methodology

We will now describe the boosting approach for solving the fold recognition problem. The basic idea of boosting is to repeatedly apply a weak learner to modified versions of the training data, thereby producing a sequence of weak classifiers for a predefined number of iterations. To begin with, all the data points are initialized with uniform weights. After this initialization, each boosting iteration fits a weak learner to the weighted training data. Error is computed and the weight of the correctly classified instances is lowered while the incorrectly classified instances will get higher weights. Note that the performance of the model built in each iteration will influence the model built in next iteration because of changing the weights of the incorrectly classified instances. The final model obtained by boosting algorithm is a linear combination of several weak learning models weighted by their own performance. Here is a brief outline of each boosting iteration: For a predefined number (t) of iterations the algorithm performs the following steps. (1) Apply a weak learner to the weighted training data. (2) Compute the error e of the weak model. (3) If e equals to zero, or e greater or equal to 0.5, terminate the model generation. If not, then for each instance, if instance classified correctly by the model multiply the weight of instance by e/(1-e). (4) Normalize the weights for all instances [14]. Finally (after the t iterations), for each of the t (or less) models, add log(e/(1-e)) to the weight of the class predicted by model.

LogitBoost performs additive logistic regression and generates models that maximizes the probability of a class. In each iteration this algorithm fits a regression model to a weighted training data. For a two class problem, if the weak learner minimizes the squared error, then the probability of the first class is maximized. This can be extended to the

multi-class as well. In general, AdaBoost optimizes the exponential loss where as LogitBoost optimizes the probability. For more details about the LogitBoost algorithm, the readers are referred to [3]. We applied these two popular boosting algorithms to the problem of protein fold recognition.

4. Experimental Results

4.1. Data set

SCOP (Structural Classification of Proteins) database classifies protein of known structure from Protein Data Bank (PDB) based on their evolutionary, functional and structural relationships [4]. Principal levels in the SCOP hierarchy are Family, Superfamily, Fold and Class. The dataset used in our study was taken from the paper by Ding and Dubchak [2]. 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. They derived an independent test set from the PDB 40D set, which consists of all SCOP sequences having less than 40% sequence identity with each other. The training set consists of 311 proteins and the test set consists of 383 proteins. Feature vectors for machine learning methods were extracted from the primary protein sequences [1]. The feature vectors characterize six different properties of a protein: amino acid composition(C), hydrophobicity(H), polarity(P), predicted secondary structure(S), van der Waals volume(V), and polarizability(Z). The feature vector for the amino acid composition consists of 20 dimensions where as all of the rest have 21 dimensions. First, the test was done on 6 parameter sets (C, S, H, P, V, Z) individually. Second, the test was done with new feature vectors ranging in size from 41 dimensions (C+S) to 125 (C+S+H+P+V+Z) by combining all the properties one at a time.

4.2. Tools

The experiments described in this section were performed on a PC with a 1.73 GHz Intel Core Duo CPU and 1 GB RAM, using Windows Vista operating system. Data mining toolkit WEKA (Waikato Environment for Knowledge Analysis) version 3.4.11 is used for classification. WEKA is an open source toolkit and it consists of collection of machine learning algorithms for solving data mining problems [14]. The AdaboostM1 and LogitBoost algorithm are discussed in section 3. For AdaboostM1, default parameters of WEKA were changed to perform 100 iterations with re-sampling. J48 (WEKA's own version of C4.5) decision tree algorithm was used as a base classifier for boosting. For LogitBoost, 100 iterations were done with re-sampling and decision stump was used as the weak learner. For each parameter set considered, 10 fold cross-validation was performed to build the model and an independent test data was evaluated using this model.

4.3. Prediction Accuracy

The standard Q percentage accuracy [1, 2] is used to measure the prediction accuracy of the algorithms. Assume that we have K number of classes with N number of test proteins such that n_1 is number of test proteins observed in class F_1, n_2 proteins in class F_2 and so on. N can be expressed as $N = [n_1 + n_2, .., n_k]$. Let a_1 be the number of proteins that are correctly classified as class F_1, a_2 be the number of proteins that are correctly classified as class F_2, and so on. The total number of proteins that are correctly classified can be given as $A = [a_1 + a_2, .., a_k]$. The class accuracy is given by $Q_i = a_i/n_i$. The overall accuracy is calculated by taking the weighted average of the individual class accuracies.

4.4. Discussion

Our experimental results clearly show that boosting outperforms all the state-of-the-art methods proposed in the literature for the protein fold recognition problem. Table 1 compares the prediction accuracy by various methods on the combination of all six parameter datasets and note that for some of the methods accuracy values for all the parameter sets are not available. LogitBoost used 104 features (C+S+H+P+V+Z) and achieved the maximum accuracy of 60.13% , whereas AdaBoost algorithm used just 62 features (C+S+H) and achieved the maximum accuracy of 58.22% . When we compared the prediction accuracy for the individual folds, boosting methods showed the highest values in most of the them. AdaBoost did well in 10-fold cross-validation of training set as well, with overall accuracy of 53.8% which was at least 8% more compared to other methods. Also, when we analyzed the individual fold accuracies, cross-validation accuracy of boosting was higher for most of the folds. In particular, AdaBoost and LogitBoost predicted all the proteins correctly for some of the folds and for some folds the percentage increase in accuracy was quite significant.

In addition to the improvements in the classification accuracy, the boosting approach provides two other advantages. First, boosting provides a better interpretable model. Only a subsets of features are used in model building. These features can provide an evidence for the biological relationship of those features with respect to the folds considered. These insights can provide vital feedback to SCOP database to generate hierarchies of data primarily based on these features. Second, run-time efficiency of boosting is higher for

Table 1. Comparison of Prediction Accuracy (in percentage) by various classifiers on the combination of all six parameter datasets.

Classifier	C	CS	CSH	CSHP	CSHPV	CSHPVZ
OvO NN	20.50	36.80	40.60	41.10	41.20	41.80
OvO SVM	43.50	43.20	45.20	43.20	44.80	44.90
uOvO SVM	49.40	48.60	51.10	49.40	50.90	49.60
AvA SVM	44.90	52.10	56.00	56.50	55.50	53.90
HKNN	—	—	57.10	57.90	55.80	—
HLA (MLP)	32.70	48.60	47.50	43.20	43.60	44.70
HLA (RBFN)	44.90	53.80	53.30	54.30	55.30	56.40
HLA (GRNN)	—	—	—	—	—	45.20
HLA (SVM)	—	—	—	—	—	53.20
Ada Boost	**51.96**	**57.7**	58.22	57.18	57.18	**57.18**
Logit Boost	46.21	56.4	**58.49**	**58.75**	**60.31**	56.14

both training and testing phases of the algorithm. Specifically, testing of the algorithm is extremely fast because it needs only a subset of features. Testing time for both the algorithms took less than 1 second for this dataset. For the dataset with all the parameters (C+S+H+P+V+Z) AdaBoost took only around 7 minutes and LogitBoost took around 11 minutes to train the model. Since the weights are computed based on the overall classifier obtained after every iteration, LogitBoost took slightly more time for training compared to AdaBoost. However, both the boosting algorithm's runtime is significantly better than SVM, NN or KNN. Considering the fact that training a SVM model is significantly slower, we anticipate that models such as AvA SVM (all vs all) will take time which is orders of magnitude larger than our boosted models.

5. Conclusion

In this paper, we presented an empirical study on the performance and advantages of using the boosting algorithms (AdaBoost and LogitBoost) to solve the problem of protein fold recognition which is a crucial task in proteomics. The classification models built using these methods not only show significant improvements in the accuracy but also performed the classification task in lesser time. The classification accuracy is 60.13% which is higher than existing methods. Boosting algorithms have the potential to build efficient classification models in a very fast manner. Improvements in the accuracy are achieved by efficient boosting models that were built in much lesser time compared to the state-of-the-art models such as Support Vector Machines, Neural Networks or Nearest Neighbor. Considering the fact that this higher accuracy is achieved using basic boosting algorithms, further work on boosting with hierarchical learning architectures or other modifications will be a promising future direction that one might want to pursue.

Acknowledgments

This work is partially funded by the Wayne State University faculty research award.

References

[1] P. Baldi, S. Brunak, Y. Chauvin, C. Anderson, and H. Nielsen. Assessing the accuracy of prediction algorithms for classification:an overview. *Bioinformatics*, 16(5):412–424, 2000.

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

[3] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting. *Annals of Statistics*, 28(2):337–407, 2000.

[4] C. Hadley and D. Jones. A systematic comparison of protein structure classifications: Scop, cath and fssp. *Biological Science*, 7(9):1099–1112, 1999.

[5] C. Huang, C. Lin, and N. Pal. Hierarchical learning architecture with automatic fearture selection for multiclass protein fold classification. *IEEE transactions on NanoBioscience*, 2(4):221–232, 2003.

[6] D. Jones. Genthreader: an efficient and reliable protein fold recognition method for genomic sequences. *Molecular Biology*, 287(4):797–815, 1999.

[7] K. Karplus, C. Barrett, and R. Hughey. Hidden markov models for detecting remote protein homologies. *Bioinformatics*, 14(10):846–856, 1998.

[8] R. Langlois, A. Diec, O. Perisic, Y. Dai, and H. Lu. Improved protein fold assignment using support vector machines. *Int. J. Bioinformatics Research and Applications*, 1(3):319–334, 2006.

[9] C. Lin and C. Hsu. A comparison of methods for multiclass support vector machines. *IEEE Trans. Neural Networks*, 13(2):415–425, 2002.

[10] O. Okun. Protein fold recognition with k-local hyperplane distance nearest neighbor algorithm. *Proceedings of the 2nd European Workshop on Data Mining and Text Mining for Bioinformatics*, pages 47–53, 2004.

[11] B. Rost and C. Sander. Prediction of protein secondary structure at better 70% accuracy. *Journal of Molecular Biology*, 232(2):584–599, 1993.

[12] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer, New York, 1995.

[13] P. Vincent and Y. Bengio. K-local hyperplane and convex distance nearest neighbor algorithms. *Advances in Neural Information Processing Systems*, 14:985–992, 2002.

[14] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques, 2nd Edition*. Morgan Kaufmann, 2005.