Annals of Biomedical Engineering, Vol. 35, No. 6, June 2007 (© 2007) pp. 1043–1052
DOI: 10.1007/s10439-007-9312-z

Learning to Translate Sequence and Structure to Function: Identifying DNA Binding and Membrane Binding Proteins

ROBERT E. LANGLOIS, MATTHEW B. CARSON, NITIN BHARDWAJ, and HUI LU

Department of Bioengineering, University of Illinois at Chicago, 218 SEO MC 063, 851 S. Morgan, Chicago, IL 60607-7052, USA

(Received 18 September 2006; accepted 2 April 2007; published online 13 April 2007)

Abstract—A protein's function depends in a large part on interactions with other molecules. With an increasing number of protein structures becoming available every year, a corresponding structural annotation approach identifying such interactions grows more expedient. At the same time, machine learning has gained popularity in bioinformatics providing robust annotation of genes and proteins without sequence homology. Here we have developed a general machine learning protocol to identify proteins that bind DNA and membrane. In general, there is no theory or even rule of thumb to pick the best machine learning algorithm. Thus, a systematic comparison of several classification algorithms known to perform well is investigated. Indeed, the boosted tree classifier is found to give the best performance, achieving 93% and 88% accuracy to discriminate non-homologous proteins that bind membrane and DNA, respectively, significantly outperforming all previously published works. We also attempted to address the importance of the attributes in function prediction and the relationships between relevant attributes. A graphical model based on boosted trees is applied to study the important features in discriminating DNA-binding proteins. In summary, the current protocol identified physical features important in DNA and membrane binding, rather than annotating function through sequence similarity.

Keywords—AdaBoost, Support vector machines, C4.5, Feature interactions.

INTRODUCTION

The interaction between proteins and biological macromolecules comprises a pivotal role in almost every cellular process, including gene regulation and signal transduction. Such macromolecules include but are not limited to proteins, metabolites, nucleic acids, lipids, and carbohydrates. The ability to identify if a protein binds one or more of these macromolecules would elucidate any number of steps in cellular activities of interest. Additionally, the potential of high-throughput structural genomics[11] to produce a great number of protein structures lacking functional annotation motivates a corresponding functional annotation approach. Further, this approach should not lean on homology, should be general enough to identify every function, and should be fast. Machine learning becomes a natural choice given these requirements. Indeed, it has become quite popular in a number of bioinformatics applications, including fold recognition,[29] subcelluar localization,[31] and genomics.[38] However, no learning algorithm clearly dominates the rest.[33] It follows that a number of learning algorithms must be tested in order to get maximum performance. In this work, the focus is to compare the ability of a number of learning algorithms to identify both protein–DNA and protein–membrane interactions as seen in Figs. 1 and 2, respectively.

There exist experimental and computational techniques to annotate both DNA-binding and membrane-binding proteins. Specifically, two conventional experimental approaches to identify DNA-binding proteins include gel mobility[12] and filter binding assays.[34] While these approaches determine whether a protein binds DNA, they do not reveal where the DNA binds. Similarly, a high-throughput technique, chromatin immunoprecipitation on a microarray (ChIP-chip),[9] incorporates microarray technology and allows researchers to create a genome wide map of protein–DNA interactions. More elaborate approaches (often used in conjunction) include genetic analysis[18] and X-ray crystallography.[15] These techniques provide high quality explicit binding data. In addition, several experimental techniques exist to identify membrane-binding proteins. Surface plasmon resonance (SPR) analysis[22] provides an effective means to identify proteins that bind membrane. In

Address correspondence to Hui Lu, Department of Bioengineering, University of Illinois at Chicago, 218 SEO MC 063, 851 S. Morgan, Chicago, IL 60607-7052, USA. Electronic mail: huilu@uic.edu

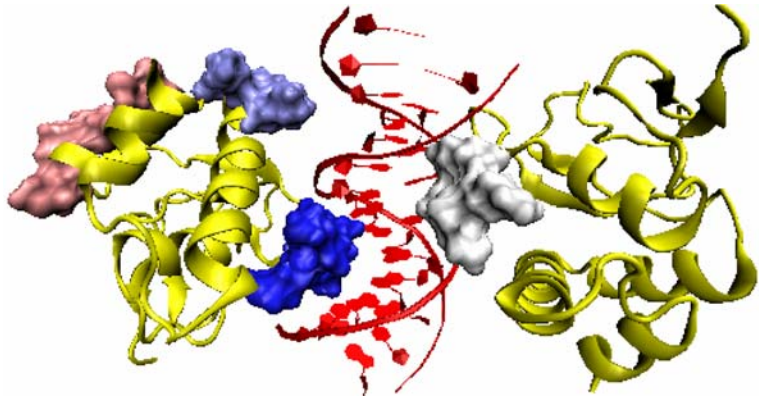

FIGURE 1. An example DNA-binding protein with the relative orientation of binding. This figure depicts the SMAD MH1 domain (1MHD) that mediates TGF-beta signaling from the cell membrane to the nucleus. The protein is shown in a yellow cartoon representation. The first four largest cationic patches have also been mapped on the surface and are colored according to their order on a blue–white–red scale: largest patch in blue, second largest in light blue and third and fourth largest in white and light red, respectively.

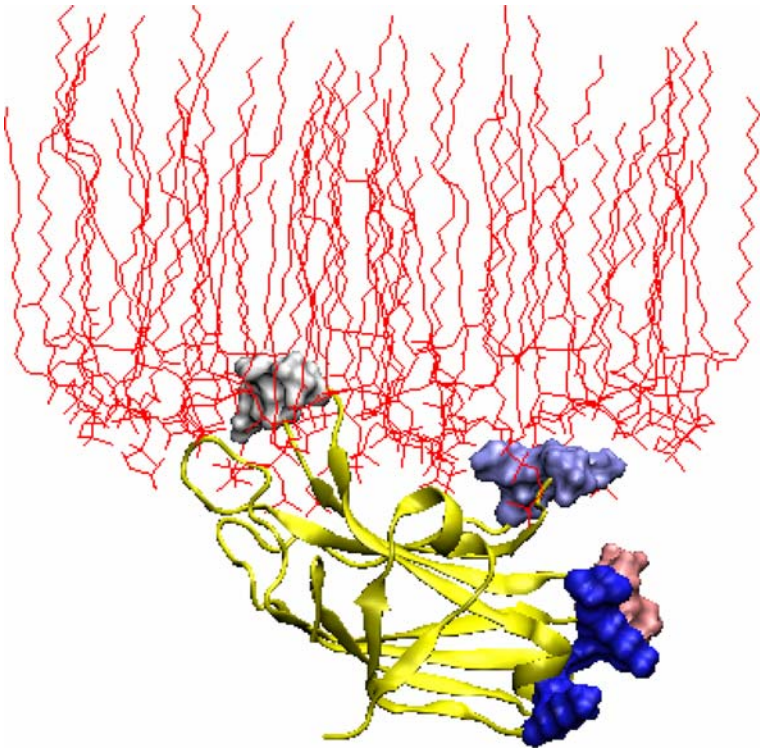

FIGURE 2. An example membrane-binding protein with the relative orientation of binding. This figure shows the C2 domain of PKC-α (1DSY) that is involved in Ca^{2+} dependent membrane signaling. The protein is shown in a yellow cartoon representation. The first four largest cationic patches have also been mapped on the surface and are colored according to their order on a blue–white–red scale: largest patch in blue, second largest in light blue and third and fourth largest in white and light red, respectively. The membrane without hydrogens is shown in red.

addition, other techniques such as fluorescence resonance energy transfer (FRET) analysis[46] have also uncovered important characteristics of protein–membrane binding such as affinity toward specific lipids. Nevertheless, such experimental methods prove costly in both time and money.

In silico efforts have also been used to identify DNA-binding proteins, the binding sites on such proteins, and the location in the genome that these proteins bind.[26,42,43] In binding site prediction, a support vector machines (SVM) classifier was employed using evolutionary and structural features to predict the binding site of specific structural motifs with 78% accuracy.[28] Likewise, a Naïve Bayes classifier and amino acid sequence have been utilized to achieve an accuracy of 78% with homology information.[47] In our

own effort, we achieved 70% accuracy using SVM on a non-homologous, larger test set without the help of evolutionary information.[4,5]

The following studies use a combination of sequence and structure-based features in conjunction with a variety of classifiers to predict if a protein binds DNA. Descriptors such as structural motifs and electrostatic potential have been used to achieve 78% accuracy over a set of three specific DNA-binding structural motifs.[40] Likewise, a hidden Markov model using structural information has been employed to identify helix-turn-helix DNA-binding motifs achieving about 71% accuracy.[35] A neural network combined with composition, sequence and structure was used to identify general DNA-binding proteins achieving 79% accuracy;[1,2] subsequently, these results were improved to 83% accuracy by adding charge, dipole moment, and quadrupole moment.[3]

Likewise, we have published work investigating the discrimination of DNA-binding,[6] RNA-binding,[6] and membrane-binding proteins using a combination of sequence and structural features.[7] Specifically, in the DNA-binding study, we achieved 86% accuracy using SVM outperforming all previously published data. The improvement in accuracy was achieved through a new definition of positive electrostatic patch and the addition of surface amino acid composition. The main drawback of that work was the "black box" nature of SVM, which makes it difficult to evaluate the importance and correlation of the sequence and structure-based features.

Membrane-binding proteins represent another important class playing a significant role in many biological processes including cell signaling and membrane trafficking.[44] The list of proteins known to bind membrane has grown exponentially in the recent past[14,23] and is expected to grow in parallel to our interest in these proteins. In spite of the growing interest in these proteins, few attempts have been made to identify such proteins *in silico*. Specifically, we have built the only machine learning protocol for automatic identification of membrane-binding proteins and have achieved a balanced success with an accuracy greater than 93%.[7]

In this work, we evaluate several state-of-the-art classifiers in an attempt to improve the accuracy of discriminating DNA- and membrane-binding proteins while determining the best classifier suited to protein binding prediction. Specifically, we compare a class of tree-based algorithms (boosted decision trees,[19] boosted decision stumps,[19] and the C4.5[37] decision tree) to SVM, which provides a baseline connection to our previous work. Moreover, we use a graphical model built using a variation of the Adaptive Boosting (AdaBoost)[19] algorithm to analyze the interactions between relevant characteristics that help determine function. The contribution of this graphical model is that it not only provides knowledge of the important physical properties in binding DNA and membrane, but also serves as a guide to future feature design.

METHODS

Dataset

We constructed several datasets; each dataset contains examples from one of two classes referred to as the positive class and the negative class. The positive class consists of examples that bind to our target molecule (DNA, RNA, membrane). The negative class consists of examples known not to bind to the target of the positive class. The first dataset comprises 75 DNA-binding proteins and 214 proteins that do not bind DNA. The second dataset comprises 37 RNA-binding proteins and the previously mentioned 75 DNA-binding proteins. These are subsets of our original datasets[6] culled using the PISCES server[45] such that no two proteins have more than 20% identity and each structure has a resolution better than 3 Å. The third dataset also comes from prior work[7] and comprises a positive set of 40 membrane-binding proteins that have no more than 40% sequence identity amongst any pair and a negative set of 230 proteins with a structural resolution better than 3 Å and less than 35% identity between each pair. We select a slightly higher threshold of 40% to remove redundant structures in the positive set as there is a smaller number of solved structures related to membrane binding. The negative set in the DNA-binding dataset is a subset of the negative proteins found in the membrane-binding dataset; this negative set was first constructed in Stawiski *et al.*[41] An analysis of the distance between the proteins in the negative and positive sets for DNA-binding and membrane-binding is found in Bhardwaj *et al.*,[6] and [7] respectively. The function of the proteins in the negative set cover a wide range from chaperoning protein folding to removing hydrogen. The datasets used in this work are available at http://proteomics.bioengr.uic.edu/pro-dna and http://proteomics.bioengr.uic.edu/pro-mem.

Feature Representation

A protein can be represented either as a sequence of labels or a set of atom types and coordinates. This representation is not favorable for machine learning because most supervised learning algorithms require the comparison of aligned features of the same length. In order to solve this problem, a protein structure is reduced to a fixed set of features encoding the

characteristics of a protein, which might be important to its function. Here, to identify DNA-binding proteins, the protein structure is translated to a set of 42 numerical features encoding both sequence and structure. The sequence-based features include the amino acid composition (20 features), and the net charge calculated using the CHARMM[8] force field (1 feature). Likewise, the structure-based features comprise surface amino acid composition derived from DSSP[25] (again 20 features) and the size of the largest positively charged patch[6,7] (1 feature). To identify membrane-binding proteins, we choose similar set of features including the net charge and the two kinds of composition. However, we use a slightly different cationic patch definition on the surface of membrane-binding proteins[7] (see Figs. 1 and 2). In addition, we also add the cumulative patch sizes of the first two, three and four largest patches as features to describe membrane-binding proteins.[7]

Classification Methods

A classifier is a supervised machine learning algorithm that attempts to generate a function (set of rules or model) from a set of training examples that best generalizes the model to unseen examples. Each example consists of an input pair, a feature vector and class label. Given an unseen feature vector ($\mathbf{x}_i$), the classifier attempts to identify the correct label (y). The following classifiers comprise a popular subset of available classifiers each of which has been implemented in our open source machine learning workbench MALIBU.[30]

Decision Trees

A decision tree[36] constructs from the training data a tree model where every internal node represents a decision and every leaf a classification. The learning process starts by finding a split on a single attribute that best classifies the training data; then the dataset is recursively split into two parts repeating these steps on each subset. There are a number of loss (or impurity) functions that are used to find the best split or the split with the minimum loss (or error). Specifically, the C4.5[37] decision tree algorithm developed by Quinlan uses a loss function known as the information gain, which is motivated by information theory. The decision tree has several advantages. Firstly, it is fast to train and evaluate. Secondly, the model (or function) learned during the training process is usually compact and easy to interpret. Finally, a decision tree does not require much data preprocessing, natively handling most attributes types. Note that most machine learning algorithms have tunable parameters. In this work, the results reported using the C4.5 decision tree algorithm use the default values empirically found to work well on a number of datasets.

AdaBoost

The AdaBoost algorithm originally proposed by Freund and Schapire[19] iteratively constructs an ensemble of weak learning algorithms over a varying distribution of the dataset. Specifically, a weak learning algorithm is trained over some distribution of the dataset starting with the uniform distribution. After each training cycle, the distribution of the dataset is altered such that incorrectly predicted examples are given a higher weight and correctly predicted examples a lower weight. The AdaBoost algorithm has been shown to minimize an exponential loss function.[19]

The AdaBoost algorithm possesses several advantages. Firstly, it is relatively simple to implement and works with many off-the-shelf classifiers. Secondly, AdaBoost achieves competitive (if not better) results when compared to other state-of-the-art classifiers.[33] Thirdly, AdaBoost does not require special knowledge or a significant amount of tuning when compared to SVM and neural networks.

The current implementation of our confidence-rated AdaBoost[39] classifier uses both C4.5 and our own implementation of the ID3[36] tree learning algorithm using entropy to find the best split.[20] From here on, we will refer to AdaBoost on C4.5 as AdaC4.5 and AdaBoost on our custom ID3 implementation as AdaTree. One final variant entails AdaBoost using one-level decision trees, often referred to as a stump, as the weak learning algorithm. Here, we will refer to this as AdaStump. The boosted tree algorithms are run for 800 iterations (more than three times the number of examples in the largest dataset). The decision trees used as weak learners are grown to produce an error of no less than 10%. Note that while we would like to grow the trees to the maximum depth, the AdaBoost algorithm requires a *weak* learner; this is left intentionally vague. However, we know trees have a large variance depending on the distribution of the dataset; for this reason we constrain the trees to produce an error of 10% to satisfy the requirements of the AdaBoost algorithm yet still build a complex classifier. One last point, the C4.5 algorithm does not take weights directly (like our custom implementation) so we use weighted sampling with replacement to change the training set distribution accordingly.

Support Vector Machines

The SVM[16] classifier uses the "kernel trick" to perform linear classification on non-linear problems. The linear classification is accomplished by finding the hyperplane that maximizes the distance between the

closest points, the maximum margin hyperplane. It is equivalent to solving the quadratic optimization problem:

$$\begin{aligned} \min_{w,b,\xi_i} \frac{1}{2} w \cdot w + C \sum_i \xi_i \quad & \text{subject to} \\ y_i(\phi(x_i) \cdot w + b) \geq 1 - \xi_i, \quad & i = 1, \ldots, m \\ \xi_i \geq 0, \quad & i = 1, \ldots, m \end{aligned} \tag{1}$$

The above problem summarizes the soft-margin SVM where C is the cost parameter that helps tolerate noise within the data and $\Phi(x_i)$ is some non-linear mapping. Applying the Lagrange transformation gives the dual:

$$L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \alpha_i \alpha_j y_i y_j \phi(x_i) \phi(x_j) \tag{2}$$

The ''kernel trick'' refers to the substitution of a kernel function $\mathbf{K}(x_i,x_j)$ for $\Phi(x_i)\Phi(x_j)$ providing an efficient approach to solve the quadratic programming problem without explicit use of the non-linear transform.[10]

In this work we use the LIBSVM[13] implementation of SVM with the full range of available kernels. This implementation of SVM has a number of tunable parameters. The kernel's gamma parameter, γ, selects one among a family of Gaussian or sigmoid functions. The soft-margin SVM's cost parameter, C, trades noise for error. Finally, the polynomial kernel has the degree, d. Note that in order to use the charge and patch size features, they must first be normalized; here we used min–max normalization (the composition features are already normalized).

Classifier Evaluation

The goal of the classifier is to find the function or model that best generalizes the training data. In order to determine how well a classifier generalizes the training data, it is necessary to evaluate the model learned over the training set on a held out test set. In order to provide a robust benchmark, each classifier is evaluated over several validation techniques and metrics.

Cross-Validation

In n-fold cross-validation (n-CV), the dataset is partitioned into n subsets. The classifier is trained n times leaving one subset out on each round of training. The omitted subset is used for testing to calculate the metric of interest and every value in the dataset contributes to the average of this metric. The cross-validation technique is demonstrably superior on smaller datasets[21] to the more common hold-out technique (where just one portion of the dataset is held out for testing). Leave-one-out cross-validation (LOO) refers to cross-validation when n equals the number of examples in the dataset. It has several known deficiencies;[27] however, we use it here to compare with previous work. Moreover, these deficiencies are migrated by the use of other validation schemes.

In the following results, we use 2-, 5-fold, and LOO cross-validation. The 2-fold cross-validation is more likely a pessimistic estimate of the following results and could vary wildly depending on the splits. For this reason, every metric reported using 2-fold cross-validation is averaged over 100 runs of randomly select splits. For a similar reason, the metrics reported for 5-fold cross-validation are averaged over 40 runs. Note that averaging leave-one-out will not have any effect on the reported metrics; so, the results reported correspond to a single run.

Metrics

The performance of the classifiers is measured using four metrics. Specifically, the following threshold metrics include accuracy, sensitivity, and specificity.

Accuracy (Acc.), Eq. (3), is the ratio of correct predictions to the total number of predictions.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

Sensitivity (Sen.), Eq. (4), also known as recall or true positive rate, TPR, is defined as the probability that a prediction is predicted positive given the example is positive. It is approximated by the fraction of true positives predicted as positive.

$$\text{Sensitivity} = \frac{TP}{TP + FN} \tag{4}$$

Specificity (Spe.), Eq. (5), is the probability that a prediction is predicted negative given the example is negative; it is approximated by the fraction of true negatives predicted as negative.

$$\text{Specificity} = \frac{TN}{TN + FP} \tag{5}$$

These metrics are referred to as threshold metrics because they depend on the threshold used in classification. In other words, a classifier generally produces a real valued prediction; the prediction is assigned to the positive or negative class by determining whether it is greater or less than some threshold (usually 0 or 0.5). Thus, if the threshold is changed the above metrics also change. In contrast, the fourth metric, area under the receiving operating characteristic curve (AUC), is an order metric. That is, it measures the ordering of the predictions relative to the true values of the examples. The receiving operating characteristic curve (ROC) is

generated by sweeping a threshold from the most negative confidence-rated prediction to the most positive, calculating the true positive rate (sensitivity, *y*-axis) and false positive rate (1-specificity, *x*-axis). This metric is analogous to sorting every prediction by its confidence then swapping examples until they are segregated by their true class label. In fact, the AUC has an attractive property; it is insensitive to changes in the class distribution.[17]

RESULTS

DNA-Binding Protein Classification

The first problem of interest concerns the ability to discover proteins that bind DNA given a structure. Here, we compare several learning algorithms (Table 1) over varying sizes of the training set. The learning algorithms comprise four tree-based algorithms and SVM. Specifically, the C4.5 decision tree algorithm forms the weak learning algorithm for the AdaBoost procedure; its results demonstrate the effectiveness of boosting. Next, the boosted C4.5 algorithm (AdaC4.5) serves as a baseline to compare our custom decision tree implementation, which forms the weak learners in AdaTree and AdaStump. Finally, the odd man out, SVM, provides a connection to our previous study;[6] however, in this study we choose to maximize the accuracy rather than find a more balanced prediction.

The results in Table 1 demonstrate that our method is effective in discriminating DNA-binding proteins. That is, given a large random set of proteins (with the same distribution as our dataset) the best classifier, AdaTree, should correctly assign on average about 88 of 100 proteins to the appropriate category. Likewise, given a protein that binds DNA, this classifier will assign 66 of 100 correctly to that category. Finally, given a protein that does not bind DNA, about 96 of 100 will be correctly assigned to this category. Indeed, this is an unbalanced result originating from both an unbalanced dataset and a set of classifiers that minimize the overall error. In other words, each of these metrics depends on the distribution of the dataset. The area under the ROC curve (AUC) furnishes a metric independent of the dataset distribution. It also gives some indication of the tradeoff between sensitivity and specificity when varying the threshold. Specifically, the AdaC4.5 learning algorithm achieves almost a 90% AUC; that is, about 90% of the predictions are ordered correctly. This ordering is important both for achieving good results on other distributions of the dataset and allowing the learning algorithm to produce a meaningful confidence in its prediction.

TABLE 1. Comparing classification and evaluation methods over the protein–DNA dataset.

		AdaTree	AdaC4.5	AdaStump	SVM	C4.5
2-CV	ACC	86.5	86.3	83.6	84.5	79.4
	SEN	61.4	61.5	60.5	57.5	59.8
	SPE	95.3	95.0	91.7	94.0	86.3
	AUC	88.0	88.8	81.6	84.4	61.3
5-CV	ACC	87.2	86.6	84.1	85.7	79.4
	SEN	63.8	61.4	61.2	59.1	59.9
	SPE	95.4	95.4	92.2	95.0	86.3
	AUC	88.4	89.6	82.7	85.9	54.1
LOO	ACC	88.5	86.5	85.1	86.3	80.0
	SEN	66.7	61.3	62.7	62.7	65.3
	SPE	96.3	95.3	93.0	93.9	85.0
	AUC	88.7	89.8	84.6	86.3	74.0

Likewise, there are several more important trends in Table 1. For example, one interesting result stems from the comparison of sensitivities for each classifier. That is, none of the observed sensitivities vary much from the C4.5 algorithm. In fact, in each superior learning algorithm (to C4.5), the increase in accuracy corresponds to a proportional increase in specificity. However, the better learning algorithms also have a larger AUC. Indeed, a larger AUC indicates that trading sensitivity for specificity will most likely have less effect on the overall accuracy over a larger range. Another interesting result in Table 1 originates from the relative independence of each classifier over each metric for different sizes of the training set. That is, for the first four algorithms, accuracy and sensitivity show the greatest change with training set size, yet this change is limited to only a few percent. If 2-fold cross-validation is a pessimistic estimate and leave-one-out an optimistic estimate, then the results of 5-fold cross-validation can be considered reliable and probably will not change much on a larger dataset. Note that only the AUC for the C4.5 algorithm improves dramatically with the increase in training examples. Finally, it is interesting to note that the results of the slowest algorithm (SVM) and the second fastest (AdaStump) match relatively well, i.e. on this dataset the speed ratio between AdaStump and SVM was on average about 1:25, respectively.

Membrane-Binding Protein Classification

The second problem of interest concerns the ability to discover proteins that bind membrane given a structure. In this study, we employ the same set of classifiers (Table 2) as the previous study with the same parameters and validation techniques. One noticeable difference between Tables 1 and 2 stands the relatively better accuracy in discriminating membrane-binding proteins. However, one might argue that this increased

TABLE 2. Comparing classification and evaluation methods over the protein–membrane dataset.

		AdaTree	AdaC4.5	AdaStump	SVM	C4.5
2-CV	ACC	91.2	91.3	90.5	86.7	87.5
	SEN	48.1	50.9	56.6	57.4	53.3
	SPE	98.6	98.2	96.3	91.7	93.4
	AUC	91.1	92.8	80.2	83.4	73.6
5-CV	ACC	93.1	92.1	91.3	91.6	88.5
	SEN	63.3	56.1	60.7	67.5	55.6
	SPE	98.3	98.3	96.6	95.7	94.2
	AUC	93.6	94.2	83.1	90.4	74.2
LOO	ACC	93.4	92.3	91.6	86. 8	88.6
	SEN	67.5	57.5	65.0	65.0	55.0
	SPE	97.9	98.7	96.1	90.6	94.4
	AUC	93.4	93.6	84.6	88.1	65.5

accuracy results from an even larger class skew highlighted by the larger imbalance between sensitivity and specificity. Nevertheless, the AUC is also higher and remains robust to such changes in class distribution.[17] Also, the decision tree consistently performs better achieving 88% accuracy.

Looking at the accuracy metric alone, the boosted decision stumps perform very well over this dataset even outperforming SVM. However, the AUC captures the true performance of these learning algorithms showing SVM is much better than boosted stumps while boosted trees outperform all the algorithms. In fact, these results are consistent with another large-scale benchmarking experiment[33] in which it was observed empirically that when AdaBoost on Trees does well, it performs much better than any other learning algorithm (personal communication with Rich Caruana). However, it has the potential to perform quite badly on datasets with significant class noise (mislabeled data).[32]

RNA from DNA-Binding Discrimination

Knowing that RNA- and DNA-binding proteins share many similar characteristics, we next investigate how well our current descriptor can discriminate these two protein classes. Table 3 compares the ability of the AdaTree algorithm to discriminate DNA- and RNA-binding proteins over various training set sizes. Specifically, the results in terms of accuracy and AUC are not very encouraging compared to our previous experiments. This is probably a defect in our feature representation. The rest of the metrics on Table 3 are less discouraging though. That is, we can say with 45% probability that a given RNA-binding protein will be predicted as RNA-binding. However, we can say with 82% probability that a given DNA-binding protein will be predicted correctly as DNA-binding.

TABLE 3. Comparing the ability of classification methods to discriminate DNA from RNA-binding proteins over different evaluation methods.

		2-CV	5-CV	LOO
AdaTree	ACC	65.6	69.0	70.5
	SEN	38.0	42.1	45.9
	SPE	79.3	82.2	82.7
	AUC	63.9	69.4	73.6

The features that best discriminate RNA- and DNA-binding proteins based on an AdaStump model (data not shown) correspond to the content of arginine, histidine, tryptophan, and tyrosine. From biophysical point of view, arginine and histidine make favorable hydrogen bond contacts with double stranded DNA.[24] Likewise, histidine makes favorable hydrogen bond contacts with single stranded DNA[24] and tryptophan has favorable van der Waals interactions.[24] Finally, tyrosine makes favorable hydrogen bond contacts with the sugar groups on RNA.[24] Thus, the features captured in AdaStump model can be explained from biophysical interactions.

Using the model built to discriminate DNA-binding proteins from proteins that do not bind DNA (excluding RNA-binding proteins), 14 RNA-binding are predicted as DNA-binding and 23 as non-binding. This is a slight improvement over previous work[6], which predicted 16 and 21, respectively. However, the final accuracy value of 70% is still much worse than the 91% accuracy[6] reported previously. This could be the result of AdaBoost overfitting on the nosier dataset.

Interactions Depicted in the Boosted Stump Model

Figures 3 and 4 illustrate the models learned by boosting one-level decision trees (decision stumps) over the protein–DNA and protein–membrane datasets, respectively. The root of each decision stump contains a decision: if true the left leaf is used otherwise the right leaf is used. Every leaf gives a confidence in its prediction and serves as a weight for the AdaBoost algorithm. The final decision is reached by summing over all the chosen leaves: if the total is greater than zero the protein is predicted as binding DNA (Fig. 3) or membrane (Fig. 4) otherwise not. The models in Figs. 3 and 4 were stopped after 13 iterations to give a more interpretable model. The LOO accuracy of the model stopped at 13 iterations reaches 83% (compared to 85% for the full model) over the protein–DNA dataset and 90% (91.6%) over the protein–membrane dataset. Thus, these models contain a majority of the information used in the final models of the AdaStump algorithm.

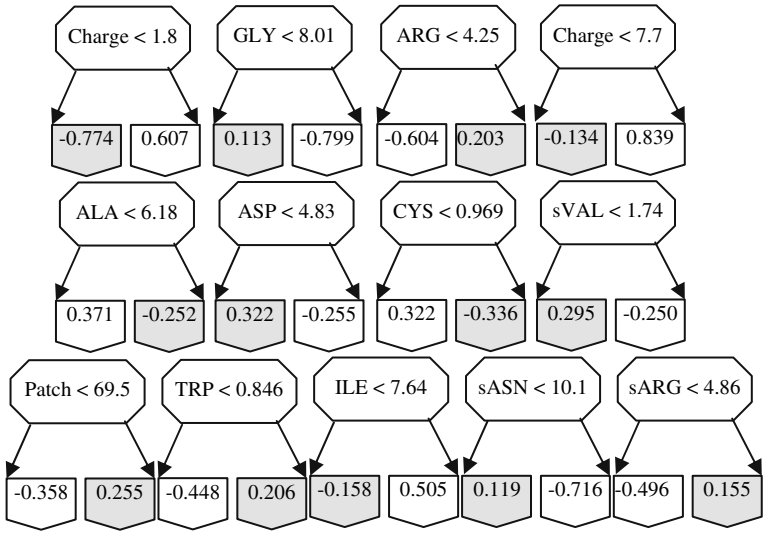

FIGURE 3. A graphical representation of the AdaStump model built on the protein–DNA dataset. At the root of each node, a single decision is made testing whether the feature in question is less than a learned threshold; if so, the value on the left leaf is used, otherwise the right leaf is used. The final decision is made by summing up these values; if the final value is greater than zero, the protein is predicted to bind DNA otherwise it is predicted not to bind DNA. The *s* in *s*ASN stands for surface amino acid composition of asparagine.

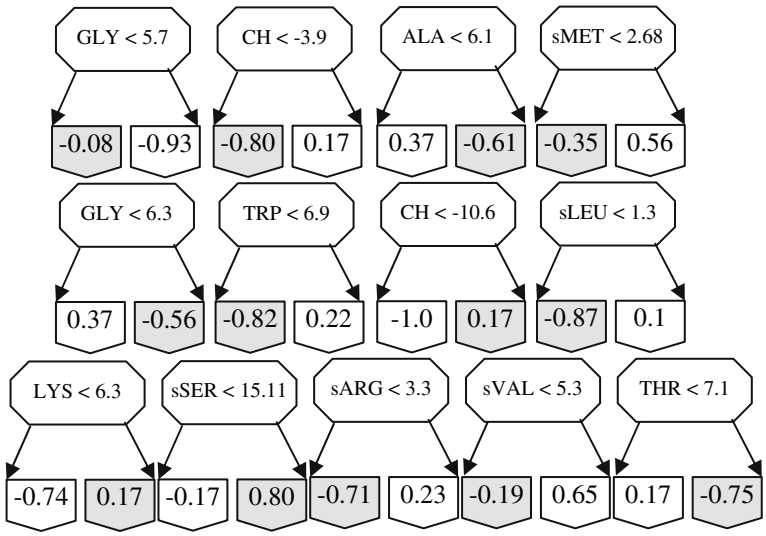

FIGURE 4. A graphical representation of the AdaStump model built on the protein–membrane dataset. At the root of each node, a single decision is made testing whether the feature in question is less than a learned threshold; if so, the value on the left leaf is used, otherwise the right leaf is used. The final decision is made by summing up these values; if the final value is greater than zero, the protein is predicted to bind membrane otherwise it is predicted not to bind membrane. The *s* in *s*ASN stands for surface amino acid composition of asparagine.

One observation that can be made over both models is the predominance of sequence-based features over the structure-based. That is, in Fig. 4, none of the surface patches play a significant role in discriminating membrane-binding proteins. Further, the surface patch is not used in the DNA-binding model until much later and it is not as confident as some other features. Another observation entails the duplication of features. In both models charge appears twice. This serves to illustrate one method the AdaBoost algorithm employs to achieve good generalization; specifically, it widens the l_2 margin[19] by refining and expanding the rules learned.

Several interesting rules can be extracted from Fig. 4 (membrane-binding). For example, if a protein has more than 5.3% surface valine or more than 15% surface serine, it is more likely to bind membrane. Both of these are smalls, neutral amino acids and may be

important to binding. Since the protein does not immerse itself in the membrane, we do not expect a significant number of surface hydrophobic residues and in the model we do not find any. Likewise, in both models we see a number of exclusionary rules. Such rules do not directly give us any information about the DNA- or membrane-binding proteins but do perform a useful function by weeding out proteins that are far away in some characteristic. Specifically, both models exclude proteins with a larger proportion of glycine, i.e., proteins that are overly flexible in some way.

DISCUSSION

This current work improves on previous work focused on discriminating DNA- and membrane-binding proteins in a number of ways. First, we have demonstrated that the boosted decision tree algorithm outperforms the other classifiers, achieving 93% and 88% accuracy for membrane-binding and DNA-binding, respectively. This study also provided a rigorous benchmark comparing each classifier over a set of important metrics and varying the training set size. Second, we were able to take advantage of the nonlinear nature and simplicity of the boosted model to graphically illustrate which features are actually important in the learned models. Specifically, we have found that proteins with larger proportions of valine and serine on the surface are more likely to bind membrane. Likewise, we found that sequence-based features dominate the AdaStump model. This seems to motivate a corresponding sequence-based approach except that here we are dealing with known structural domains. In order to develop a truly sequence-based approach, we would have to find a way to deal with larger sequences that contain an unknown number of domains, only one of which may contain the function of interest.

Among the classifiers, the boosted decision trees performed the best on nearly every metric and for each training set size. We also discussed how the area under the ROC serves as a better metric for unbalanced datasets in that it is unaffected by the underlying class distribution. This is important for future work since there is most likely a larger skew between DNA-binding (or membrane-binding) proteins and other proteins. Also, a larger AUC indicates that the learning algorithm will produce better confidence values and make more robust predictions because it measures the relative ordering of predictions.

Finally, we tackled the issue of how RNA-binding proteins are handled by our classifier. Without including them in the training, we found a majority would be predicted as not DNA-binding. Furthermore, we showed that a classifier trained on DNA- vs. RNA-binding could correctly predict a given DNA-binding protein with 80% probability.

ACKNOWLEDGMENTS

This work is partially supported by NIH grant P01 AI060915 to H. L. (PI. Mike Johnson). R.E.L. is supported by NIH training grant T32 HL 07692: Cellular Signaling in Cardiovascular System (PI, John Solaro). N.B. gratefully acknowledges the kind support from FMC Technologies Inc., Fellowship.

REFERENCES

[1]Ahmad, S., M. M. Gromiha, and A. Sarai. Prediction of DNA binding in proteins from composition, sequence and structure. *Genome Inf*. 13:308–309, 2002.

[2]Ahmad, S., M. M. Gromiha, and A. Sarai. Analysis and prediction of DNA-binding proteins and their binding residues based on composition, sequence and structural information. *Bioinformatics* 20:477–486, 2004.

[3]Ahmad, S., and A. Sarai. Moment-based prediction of DNA-binding proteins. *J. Mol. Biol*. 341:65–71, 2004.

[4]Bhardwaj, N., and H. Lu. Residue-level prediction of DNA-binding sites and its application on DNA-binding protein predictions. *FEBS Lett*. 581:1058–1066, 2007.

[5]Bhardwaj, N., R. Langlois, G. Zhao, and H. Lu. Structure based prediction of binding residues on DNA-binding proteins. In: *Proceedings of 27th Annual Conference of the IEEE Engineering in Medicine and Biology Society*, Shanghai, China.

[6]Bhardwaj, N., R. E. Langlois, G. Zhao, and H. Lu. Kernel-based machine learning protocol for predicting DNA-binding proteins. *Nucleic Acids Res*. 33:6486–6493, 2005.

[7]Bhardwaj, N., R. V. Stahelin, R. E. Langlois, W. Cho, and H. Lu. Structural bioinformatics prediction of membrane-binding proteins. *J. Mol. Biol*. 359:486–495, 2006.

[8]Brooks, B., R. E. Bruccoleri, B. Olafson, D. States, S. Swaminathan, and M. Karplus. Charmm: a program for macromolecular energy, minimization, and dynamics calculations. *J. Comput. Chem*. 4:187–217, 1983.

[9]Buck, M. J., and J. D. Lieb. Chip-chip: considerations for the design, analysis, and application of genome-wide chromatin immunoprecipitation experiments. *Genomics* 83:349–360, 2004.

[10]Burges, C. J. C. A tutorial on support vector machines. *Data Min. Knowl. Disc*. 2:121–167, 1998.

[11]Burley, S. K., S. C. Almo, J. B. Bonanno, M. Capel, M. R. Chance, T. Gaasterland, D. Lin, A. Sali, F. W. Studier, and S. Swaminathan. Structural genomics: beyond the human genome project. *Nat. Genet*. 23:151–157, 1999.

[12]Cajone, F., M. Salina, and A. Benelli-Zazzera. 4-Hydroxynonenal induces a DNA-binding protein similar to the heat-shock factor. *Biochem. J*. 262:977–979, 1989.

[13]Chang, C.-C., and C.-J. Lin. Libsvm: a library for support vector machines, 2003. Available from: http://www.csie.ntu.edu.tw/∼cjlin/papers/libsvm.pdf.

[14]Cho, W., and R. V. Stahelin. Membrane–protein interactions in cell signaling and membrane trafficking. *Annu. Rev. Biophys. Biomol. Struct*. 34:119–151, 2005.
[15]Chou, C.-C., T.-W. Lin, C.-Y. Chen, and A. H. J. Wang. Crystal structure of the hyperthermophilic archaeal DNA-binding protein sso10b2 at a resolution of 1.85 angstroms. *J. Bacteriol*. 185:4066–4073, 2003.
[16]Cortes, C., and V. Vapnik. Support-vector networks. *Mach. Learn*. 20:273–297, 1995.
[17]Fawcett, T. Roc graphs: notes and practical considerations for data mining researchers, 2003. Available from: http://www.hpl.hp.com/techreports/2003/HPL-2003-4.pdf.
[18]Freeman, K., M. Gwadz, and D. Shore. Molecular and genetic analysis of the toxic effect of rap1 overexpression in yeast. *Genetics* 141:1253–1262, 1995.
[19]Freund, Y., and R. E. Schapire. Experiments with a new boosting algorithm. In: *Proceedings of 13th Annual International Conference on Machine Learning*, Bari, Italy, 1996.
[20]Friedman, J., T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting, 1998. Available from: http://www-stat.stanford.edu/∼jhf/ftp/boost.ps.
[21]Goutte, C. Note on free lunches and cross-validation. *Neural. Comp*. 9:1211–1215, 1997.
[22]Henriette Mozsolits, M.-I. A. Surface plasmon resonance spectroscopy: an emerging tool for the study of peptide–membrane interactions. *Peptide. Sci*. 66:3–18, 2002.
[23]Hurley, J. H., and T. Meyer. Subcellular targeting by membrane lipids. *Curr. Opin. Cell. Biol* 13:146–52, 2001.
[24]Jones, S., D. T. A. Daley, N. M. Luscombe, H. M. Berman, and J. M. Thornton. Protein–RNA interactions: a structural analysis. *Nucleic Acids Res*. 29:943–954, 2001.
[25]Kabsch, W., and C. Sander. Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers* 22:2577–2637, 1983.
[26]Kaplan, T., N. Friedman, and H. Margalit. Ab initio prediction of transcription factor targets using structural knowledge. *PLoS. Comp. Biol*. 1:e1, 2005.
[27]Kearns, M., and D. Ron. Algorithmic stability and sanity-check bounds for leave-one-out cross-validation. *Neural. Comp*. 11:1427–1453, 1999.
[28]Kuznetsov, I. B., Z. Gou, R. Li, and S. Hwang. Using evolutionary and structural information to predict DNA-binding sites on DNA-binding proteins. *Proteins Struct. Funct. Bioinform*. 64:19–27, 2006.
[29]Langlois, R. E., A. Diec, O. Perisic, Y. Dai, and H. Lu. Improved protein fold assignment using support vector machines. *Int. J. Bioinform. Res. Appl*. 1:319–334, 2006.
[30]Langlois, R. E., and H. Lu. User manual of malibu: machine learning workbench for bioinformatics applications, 2007. Available from: http://proteomics.bioengr.uic.edu/malibu/.
[31]Lei, Z., and Y. Dai. A novel approach for prediction of protein subcellular localization from sequence using fourier analysis and support vector machines. In: *Proceedings of 4th ACM SIGKDD Workshop on Data Mining in Bioinformatics*, Seattle, 2004.
[32]McDonald, R. A., D. J. Hand, and I. A. Eckley. An empirical comparison of three boosting algorithms on real data sets with artificial class noise. In: Lecture Notes in Computer Science, edited by T. Windeatt and F. Roli. Berlin: Springer-Verlag, 2003, pp. 35–44.
[33]Niculescu-Mizil, A., and R. Caruana. An empirical comparison of supervised learning algorithms In: *Proceedings of 23rd Annual International Conference on Machine Learning*, Pittsburgh, Pennsylvania, 2006.
[34]Oehler, S., R. Alex, and A. Barker. Is nitrocellulose filter binding really a universal assay for protein–DNA interactions? *Anal. Biochem*. 268:330–336, 1998.
[35]Pellegrini-Calace, M., and J. M. Thornton. Detecting DNA-binding helix-turn-helix structural motifs using sequence and structure information. *Nucleic. Acids Res*. 33:2129–2140, 2005.
[36]Quinlan, J. R. Induction of decision trees. *Mach. Learn*. 1:81–106, 1986.
[37]Quinlan, J. R. Improved use of continuous attributes in c4.5. *J. Artif. Intell. Res*. 4:77–90, 1996.
[38]Saetrom, P., R. Sneve, K. I. Kristiansen, O. Snove Jr, T. Grunfeld, T. Rognes, and E. Seeberg. Predicting non-coding rna genes in escherichia coli with boosted genetic programming. *Nucleic. Acids. Res*. 33:3263–3270, 2005.
[39]Schapire, R. E., and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Mach. Learn*. 37:297–336, 1999.
[40]Shanahan, H. P., M. A. Garcia, S. Jones, and J. M. Thornton. Identifying DNA-binding proteins using structural motifs and the electrostatic potential. *Nucleic Acids Res*. 32:4732–4741, 2004.
[41]Stawiski, E. W., L. M. Gregoret, and Y. Mandel-Gutfreund. Annotating nucleic acid-binding function based on protein structure. *J. Mol. Biol*. 326:1065–1079, 2003.
[42]Stormo, G. DNA binding sites: representation and discovery. *Bioinformatics* 16:16–23, 2000.
[43]Stormo, G. D. Computer methods for analyzing sequence recognition of nucleic acids. *Annu. Rev. Biophys. Biophys. Chem*. 17:241–263, 1988.
[44]Teruel, M. N., and T. Meyer. Translocation and reversible localization of signaling proteins: a dynamic future for signal transduction. *Cell* 103:181–184, 2000.
[45]Wang, G., and R. L. Dunbrack Jr. Pisces: a protein sequence culling server. *Bioinformatics* 19:1589–1591, 2003.
[46]Wu, P. G., and L. Brand. Resonance energy transfer: methods and applications. *Anal. Biochem*. 218:1–13, 1994.
[47]Yan, C., M. Terribilini, F. Wu, R. L. Jernigan, D. Dobbs, and V. Honavar. Predicting DNA-binding sites of proteins from amino acid sequence. *BMC Bioinform*. 7:262–272, 2006.