Predicting a Biological Response of Molecules from Their Chemical Properties Using Diverse and Optimized Ensembles of Stochastic Gradient Boosting Machine

Tarek Abdunabi
University of Waterloo
200 University Avenue West
Waterloo, Ontario, Canada N2L 3G1
Email: tabdunab@uwaterloo.ca

Otman Basir
University of Waterloo
200 University Avenue West
Waterloo, Ontario, Canada N2L 3G1
Email: obasir@uwaterloo.ca

***Abstract*—The development of a new drug largely depends on trial and error. It typically involves synthesizing thousands of compounds that finally becomes a drug. This process is extremely expensive and slow. Therefore, the ability to accurately predict the biological activity of molecules, and understand the rationale behind those predictions would be of great value to the pharmaceutical industry. Gradient Boosting Machines (GBMs) are powerful ensemble learning techniques that have been successfully applied to several low-dimensional applications. Despite their high accuracy, GBMs suffer from major drawbacks such as high memory-consumption. In this paper, using real, high-dimensional (i.e. 1776 predictors) molecules dataset, we demonstrate that by using different feature selection/reduction techniques, the computations costs for building and tuning GBMs can be substantially reduced at a slight drop in prediction accuracy. In addition, by fusing the decisions made by the ensembles using two fusion techniques, namely a majority vote and an optimized feedforward neural network, we obtain a better prediction accuracy than the individual accuracy of all ensembles.**

I. Introduction

THE activity of predicting the biological response of molecules from their chemical properties is classified as Quantitative Structure-Activity Relationship (QSAR) [1]. QSAR methods require the identification of at least one lead molecule that elicit the activity. Then, the QSAR method correlates the computed properties from molecules' structure with their activities. The resulting correlation is then used to predict the activity of hypothetical molecules. This correlation may also help in understanding the structural features that contribute to activities. Based on the set of features of molecules that are correlated with activity, and methods used to identify the correlation, QSAR methods are broadly classified into *traditional* methods, and *3D* methods [2].

QSAR *traditional* methods use combination of three types of features. The first type is bulk molecular properties, such as computed/measured molar refractivity, molar volume, and computed/measured octanol/water partition coefficient. The second type is topological and geometrical features, such as the lengths of the principal axes, aspect ratios, number of aromatic bonds, and connectivity indices. The third type only applies in cases where the molecules consist of substitutions on a common parent structure. Traditional methods predictive accuracy is high in some systems, and poor in others.

QSAR *3D* methods use as features direct measurements of the three-dimensional shapes of molecules and three-dimensional distribution of charges in and about molecules. 3D methods often produce better predictions than traditional methods [2].

The development of a new drug largely depends on trial and error. It typically involves synthesizing thousands of compounds that finally becomes a drug. This process requires, on average, thousands of dollars, and a considerable amount of time, ranging from few days to few weeks. As a consequence, drug discovery is extremely expensive and slow [2]. Therefore, the ability to accurately predict the biological activity of molecules, and understand the rationale behind those predictions would be of great value to the pharmaceutical industry.

When building a predictive model for domain-specific area, such as drug discovery, one approach is build a model from theory and then adjust its parameters based on the observed data. However, in most real-life applications such models are not available. Moreover, initial knowledge about the relationships between the input variables and outcomes is not available. These limitations lead to an alternative approach using non-parametric machine learning techniques to build a model directly from the data. The common approach in data-driven modeling is to build a single strong predictive model. However, recently, the focus has shifted towards building an ensemble of models for the same task. The idea is to combine a large number of relatively week simple models to construct a stronger and more reliable ensemble prediction [3].

Random forests [4] is one of the most prominent examples of ensemble learning, which relies on simple averaging of models in the ensemble. Another prominent example is the family of boosting methods. In boosting methods new models

978-1-4799-5666-1/14/$31.00 ©2014 IEEE

are added sequentially to the ensemble. At each iteration, a new base-leaner model is trained with respect to the error of the ensemble learned in the previous iterations [3].

A gradient-descent based formulation of boosting methods was derived by [5], [6], [7]. The models based on this formulation were called Gradient Boosting Machines (GBMs). In GBMs, the learning algorithm sequentially fits new models to provide more accurate prediction of the response variable. The goal is to maximally correlate the new base-learners with the negative gradient of the loss function associated with the learned ensemble in the previous iterations. GBMs have been successfully applied to several low-dimensional applications such as mapping Electromyography (EMG) and Electroencephalography (EEG) sensor readings to human movement tracking and activity recognition. Despite their high accuracy, GBMs suffer from major drawbacks such as high memory-consumption. In addition, given the fact that the learning algorithm is essentially sequential, it has problems with parallelization by design [3].

In this paper, using real, high-dimensional molecules dataset, we demonstrate that by using different feature selection/reduction techniques, the computations costs for building and tuning GBMs can be substantially reduced at a slight drop in prediction accuracy. In addition, we show that by fusing the decisions made by the ensembles built using feature selection/reduction techniques we obtain better prediction accuracy than all ensembles including the ensemble built using all predictors.

The reminder of this paper is organized as follows: Section II briefly introduces stochastic gradient boosting. Section III explains data preparation, and ensembles building and tuning using several feature selection/reduction techniques. The performance of the built ensembles is evaluated in Section IV. Section V compares the computations time required to build and tune each ensemble. Section VI presents two fusion techniques, namely majority vote and stacking an optimized feedforward neural network, used to fuse ensembles' decisions. Finally, Section VII provides the conclusion.

II. Stochastic Gradient Boosting

Given a system consisting of a set of random input variables $x = \{x_1, ..., x_n\}$, a random output/response y, and a training sample $\{y_i, Xi\}_1^N$ of known (y, X) values. The goal of the function estimation problem is to find a function $F^*(X)$ that maps x to y, such that the expected value of a specific loss function $\psi(y, F(X))$ is minimized. Gradient boosting algorithm 1 approximates $F^*(X)$ by an additive expansion of the form $F(X) = \sum_{m=0}^{M} \beta_m h(X; a_m)$, where the functions $h(X; a_m)$ (base-learner) are simple function of X with parameters $a = \{a_1, a_2, ...\}$. At each iteration, the algorithm fits the base-learner (line 3) and calculate the model update for the current iteration (line 4), and expansion parameters (line 5). Then, the current approximation $F_m(X)$ is then updated (line 6) [8].

Algorithm 1 Gradient Tree Boost [8]

1: Initialize $F_0(X) = argmin_\gamma \sum_{i=1}^{N} \psi(y_i, \gamma)$
2: **for** $m = 1$ to M **do**
3: $\quad \tilde{y}_{im} = -[\frac{\partial \psi(y_i, F(X_i))}{\partial F(X_i)}]_{F(X)=F_{m-1}(X)}$, $i = 1, N$
4: $\quad \{R_{lm}\}_1^L = L-$terminal node tree$(\{\tilde{y}_{im}, X_i\}_1^N)$
5: $\quad \gamma_{lm} = argmin_\gamma \sum_{X_i \in R_{lm}} \psi(y_i, F_{m-1}(X) + \gamma)$
6: $\quad F_m(X) = F_{m-1}(X) + \upsilon \cdot \gamma_{lm} 1(X \in R_{lm})$
7: **end for**

Motivated by the hybrid bagging boosting procedure proposed in [9], Friedman [8] made a minor modification to gradient boosting algorithm 1 to incorporate randomness and produce the Stochastic Gradient boosting algorithm 2. At each iteration, a subsample of the training data is randomly drawn (without replacement) (line 3) and used, instead of the full training data, to fit the base-learner (line 4) and calculate the model update for the current iteration (line 5). The incorporation of randomness into the procedure substantially improves the accuracy and execution speed of gradient boosting.

Algorithm 2 Stochastic Gradient Tree Boost [8]

1: Initialize $F_0(X) = argmin_\gamma \sum_{i=1}^{N} \psi(y_i, \gamma)$
2: **for** $m = 1$ to M **do**
3: $\quad \{\pi(i)\}_1^N = rand\_perm\{i\}_1^N$
4: $\quad \tilde{y}_{\pi(i)m} = -[\frac{\partial \psi(y_{\pi(i)}, F(X_{\pi(i)}))}{\partial F(X_{\pi(i)})}]_{F(X)=F_{m-1}(X)}$, $i = 1, \tilde{N}$
5: $\quad \{R_{lm}\}_1^L = L-$terminal node tree$(\{\tilde{y}_{\pi(i)m}, X_{\pi(i)}\}_1^{\tilde{N}})$
6: $\quad \gamma_{lm} = argmin_\gamma \sum_{X_{\pi(i)} \in R_{lm}} \psi(y_{\pi(i)}, F_{m-1}(X_{\pi(i)}) + \gamma)$
7: $\quad F_m(X) = F_{m-1}(X) + \upsilon \cdot \gamma_{lm} 1(X \in R_{lm})$
8: **end for**

III. Building and Tuning The Ensembles

Using the *R* package *gbm* [10], one implementation of the Stochastic Tree-Based GBM described in section II, we need to decide on five main hyperparameters to build our ensembles. These are: the loss function (distribution), the number of iterations (n.trees), the depth of each tree (interaction.depth), the shrinkage (or learning rate) parameter, and the subsampling rate (bag.fraction).

For classification problems, the *bernoulli* distribution, and *bag.fraction* = 0.5 are recommended for the loss function and subsampling rate respectively [11]. Therefore, we need to tune the remaining hyperparameters: n.trees, interaction.depth, and shrinkage to obtain the best performance. To reduce the number of the tunning hyperparameters, we decided to choose a small shrinkage (learning rate) of 0.001. For each ensemble built, the remaining hyperparameters are tunned using repeated (3 times) 10-fold cross validation grid-search.

A. Computational Resources

The Amazon Elastic Compute Cloud (EC2) web service is used to accelerate the data-intensive computations involved in building and tuning the ensembles. The utilized compute-optimized instance (*c3.8xlarge*) has the highest performing

processors (Intel Xeon E5-2680 v2) currently available in EC2, with 32 CPUs and 60 GiB (1 Gibibyte = 1 x 2^{30} bytes) physical memory. The used *c3.8xlarge* instance runs Ubuntu Server 13.10 (64-bit) operating system.

All models are built using the *R* statistical computing programming language (version 3.0.1) [12]. Several additional *R* packages are used. Some of the main packages include: *caret* [13], *gbm* [10], *pROC* [14], *nnet* [15], and *doMC* [16] (for parallel computing).

For reproducible results, the *R* code and data, used in building and tuning the ensembles, are provided as a supplementary material, and will be made available on the first author's *Github* webpage.

B. Data Preparation

The data is obtained from the Kaggle.com's competition: "Predicting a Biological Response" [17] held between March 16, 2012 and June 15, 2012. The objective of the competition was to build a predictive model to optimally relate molecular information to an actual biological response.

The competition's sponsor (Boehringer Ingelheim Ltd.) provided the data in the comma separated values (CSV) format. Each row in the data set represents a molecule. The first column contains experimental data describing an actual biological response; the molecule was seen to elicit this response (1 or Active), or not (0 or Inactive). The remaining columns represent molecular descriptors (D1 through D1776). These are calculated properties that can capture some of the characteristics of the molecule - for example size, shape, or elemental constitution. The descriptor matrix has been normalized. Only the training data (with 3751 observations, and 1776 variables/predictors) is used in building and tuning our ensembles.

As shown in Figure 1, the distribution of the two classes (Active, Inactive) in the provided dataset is quite balanced.

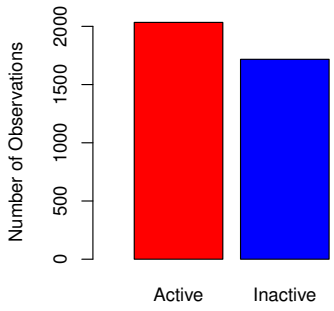

Fig. 1. Classes distribution in the complete dataset

To account for the outcome (classes) distribution when splitting the available dataset to "training" data (70%, to be used in building and tuning the ensembles), and "test" data (30%, to be used in evaluating the performance of the ensembles), we use stratified random sampling within the classes. In this way, the classes distributions will be quite balanced in both datasets as shown in Figure 2.

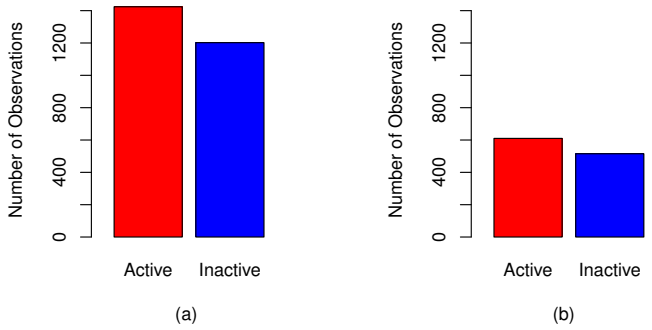

Fig. 2. Classes distribution: (a) training data, (b) testing data

C. Building the GBM Ensemble Using All Predictors

The first GBM ensemble is built using all 1776 predictors in our dataset. Figure 3 shows GBM classification accuracy against the tuning hyperparameters using all predictors. The optimized ensemble is shown in Table I.

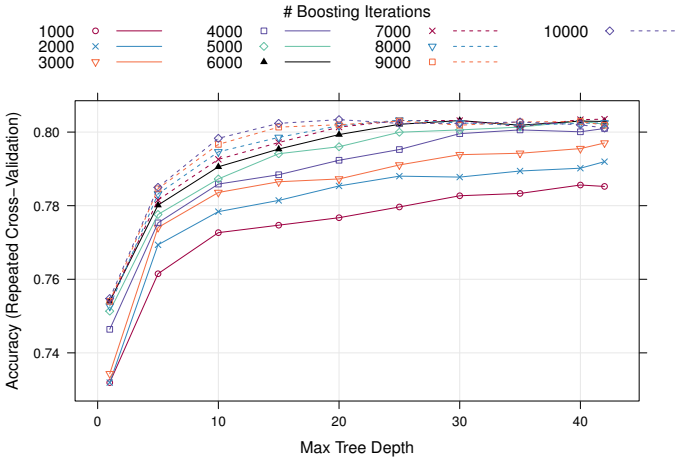

Fig. 3. Stochastic GBM classification accuracy versus the tuning hyperparameters using all predictors (1776)

TABLE I
OPTIMIZED ENSEMBLE USING ALL (1776) PREDICTORS

No. of trees	Interaction depth	CV accuracy	Accuracy SD
7000	42	80.4%	3.2%

D. Building the Ensemble Using PCA

Principal component analysis (PCA) is one of the most widely used methods in modern data analysis. It is a simple, non-parametric method able to reduce a complex high-dimensional dataset to a lower dimension. PCA identifies relationships by generating new variables/components which are linear combinations of variables that have common variation [18].

For our dataset, PCA generates 1776 components (equals to the number of predictors). Figure 4 shows the variance explained by the first 10 PCA components. Around 70% of

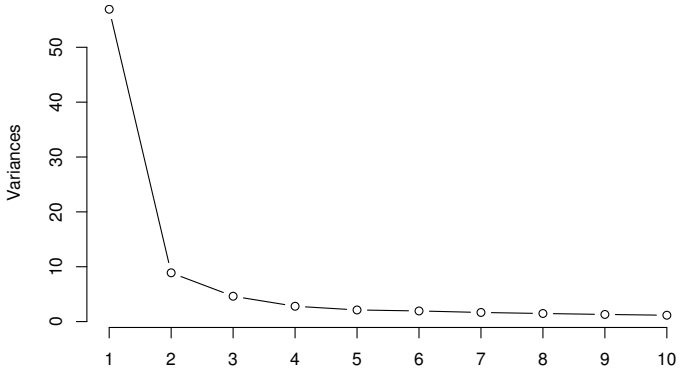

Fig. 4. Variance explained by first 10 PCA components

the variance is explianed by first two components. However, plotting the first component againts the second component, as shown in Figure 5, reveals the complexity of the dataset, suggesting that more components are needed in order to separate the classes.

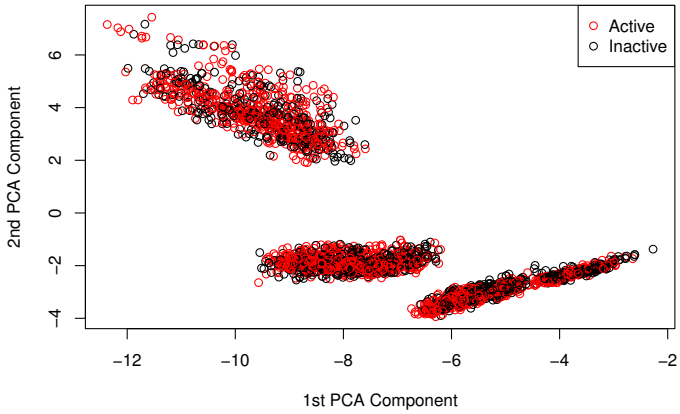

Fig. 5. First vs second PCA components

Several methods have been proposed to determine the number of PCA components to retain for further analysis and predictive modeling [19]. We use two popular methods to build two GBM ensembles, namely proportion of total variance explained, and Kaiser-Guttman rule.

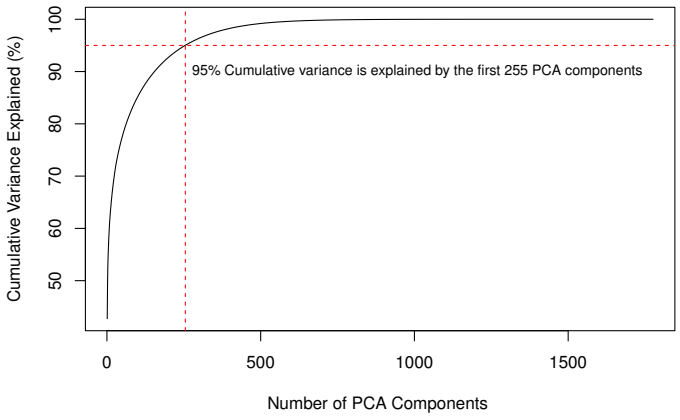

Fig. 6. Cumulative variance explained by PCA components

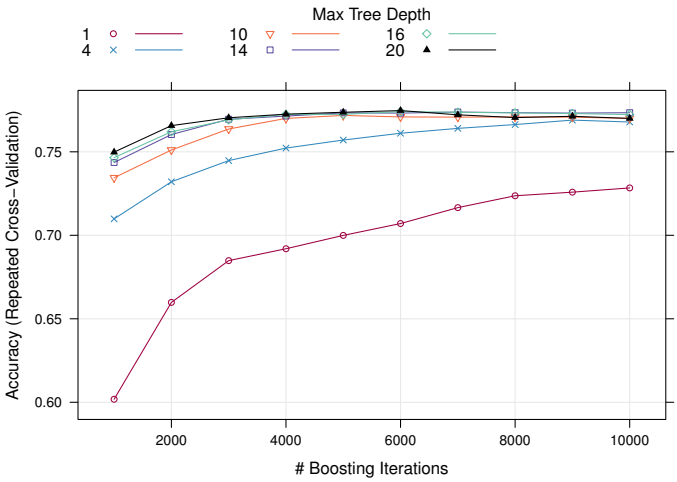

Fig. 7. Stochastic GBM classification accuracy versus the tuning hyperparameters using the first 255 PCA components (95% cumulative variance)

In the proportion of total variance explained method, enough components are selected to explain at least x% of the total variance. As shown in Figure 6, the first 255 PCA components are required to explain 95% of the total variance. Figure 7 shows GBM classification accuracy against the tuning hyperparameters using the first 255 PCA components. The optimized ensemble is shown in Table II.

TABLE II
OPTIMIZED ENSEMBLE USING THE FIRST 255 PCA COMPONENTS

No. of trees	Interaction depth	CV accuracy	Accuracy SD
6000	20	77.5%	3.27%

In Kaiser-Guttman rule [20], PCA components with eigenvalues larger than 1.0 are selected. Using this rule, the first

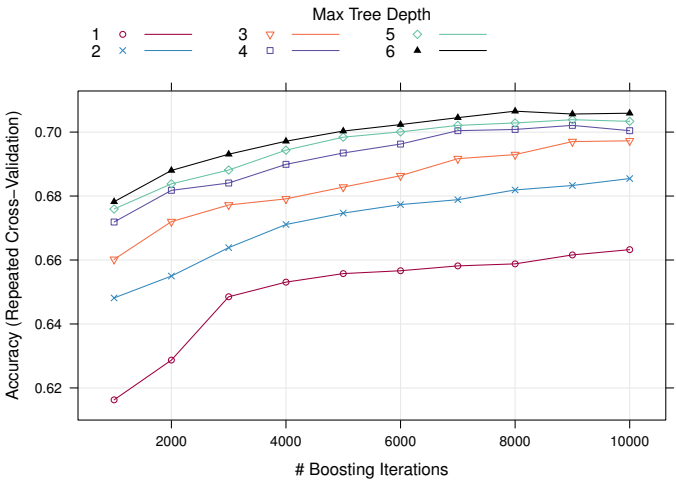

Fig. 8. Stochastic GBM classification accuracy versus the tuning hyperparameters using the first 11 PCA components (Kaiser-Guttman rule)

11 PCA components of our dataset are selected to build the GBM ensemble. Figure 8 shows the tuning of the GBM hyperparameters using the first 11 PCA components. The optimized model is shown in Table III.

TABLE III
OPTIMIZED ENSEMBLE USING THE FIRST 11 PCA COMPONENTS

No. of trees	Interaction depth	CV accuracy	Accuracy SD
8000	6	70.7%	2.89%

E. Building the Ensemble Using Predictors Area Under ROC Curve

Receiver Operating Characteristic (ROC) Curves [21], [22], [23] are general methods used to determine an effective threshold such that values above the threshold are indicative of a specific event. With categorical outcomes and numeric predictors, the area under the ROC curve can be used to quantify predictor relevance. If the predictor could perfectly separate the classes, there would be a cutoff for the predictor that would achieve a sensitivity and specificity of 1 and the area under the curve would be 1. A completely irrelevant predictor would have an area under the curve of approximately 0.5 [24].

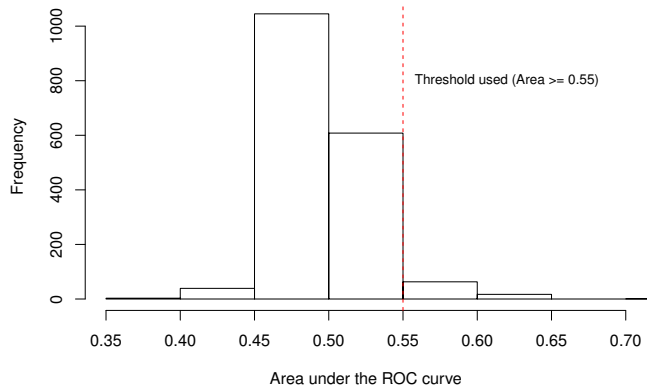

Fig. 9. Frequency of predictors vs their area under ROC curve

Figure 9 shows the frequency of predictors versus their area under ROC. Using the area greater than or equal to 0.55 as our threshold, only 81 predictors are selected to build the GBM ensemble. Figure 10 shows the tuning of the GBM hyperparameters using the selected 81 predictors. The optimized model is shown in Table IV.

TABLE IV
OPTIMIZED ENSEMBLE USING 81 PREDICTORS (ROC AREA >= 0.55)

No. of trees	Interaction depth	CV accuracy	Accuracy SD
10000	11	78.5%	2.96%

F. Building the Ensemble Using the Relief Algorithm

The Relief algorithm [25] is a generic method originally developed for classification problems with two classes. It

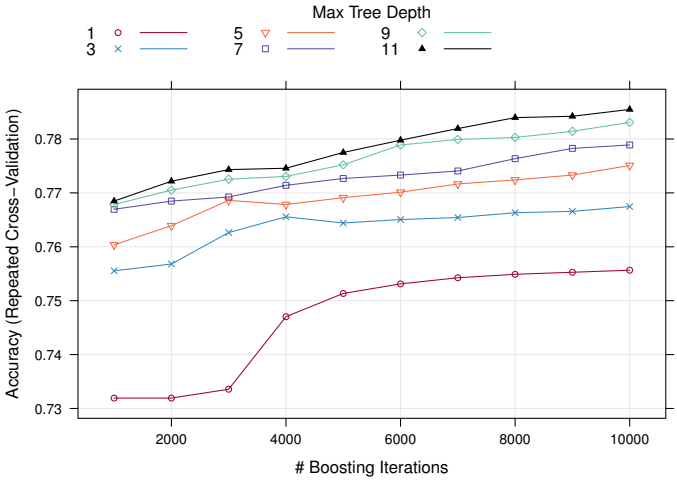

Fig. 10. Stochastic GBM classification accuracy versus the tuning hyperparameters using 81 predictors (with ROC area >= 0.55)

attempts to estimate the quality of predictors according to how well their values distinguish between instances that are near to each other. For a randomly selected training instant *Ri*, the Relief algorithm finds its two nearest neighbors: one from the same class, called nearest hit *H*, and the other from the different class, called nearest miss *M*. It updates the quality estimation *W[P]* for all predictors *P* depending on their values for *Ri*, *M*, and *H*. If instances *Ri* and *H* have different values of the predictor *P* then the predictor *P* separates two instances with the same class which is not desirable so the quality estimation *W[P]* is decreased. On the other hand, if instances *Ri* and *M* have different values of the predictor *P* then the predictor *P* separates two instances with different class values which is desirable so the quality estimation *W[P]* is increased. The whole process is repeated for *m* times, where *m* is a user-defined parameter [26].

The ReliefF algorithm [27] is an improved version of the Relief algorithm that can be used for classification problems with more than two classes. It uses more than a single nearest neighbor, and can handle missing predictor values. The RReliefF algorithm [28] is another extension to handle regression problems [24].

In contrast to the majority of heuristic methods for estimating the quality of predictors, which assume the conditional independence of the predictors, relief algorithms can estimate the quality of the predictors with strong dependencies between themselves [26].

Figure 11 shows the frequency of predictors versus their Relief score. Kira and Rendell [25] suggest that a threshold can be much smaller than $\frac{1}{\sqrt{\alpha m}}$, where α is the desired false-positive rate, and m is the number of randomly selected training instances used to calculate relief scores. For our dataset, with 5% false positive rate, and $m = 2626$ the suggested threshold is ≈ 0.087. However, this value seems to be inappropriate for our predictors' scores. Therefore, we decided to use a score greater than or equal to 0.04 as our threshold resulting in selecting 86 predictors.

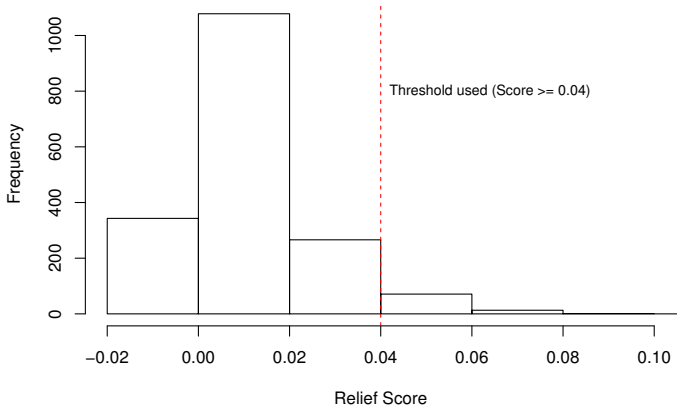

Fig. 11. Frequency of predictors vs their Relief score

Figure 12 shows the tuning of the GBM hyperparameters using the selected 86 predictors. The optimized model is shown in Table V.

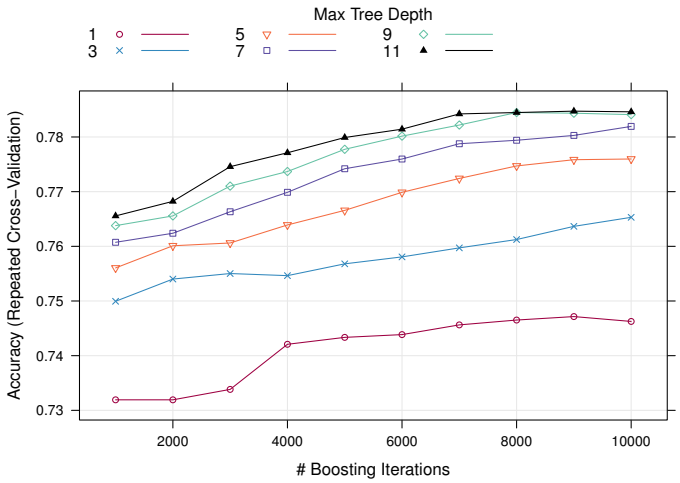

Fig. 12. Stochastic GBM classification accuracy versus the tuning hyperparameters using 86 predictors (with Relief score >= 0.04)

TABLE V
OPTIMIZED ENSEMBLE USING 86 PREDICTORS (RELIEF SCORE >= 0.04)

No. of trees	Interaction depth	CV accuracy	Accuracy SD
9000	11	78.5%	2.59%

G. Building the Ensemble Using GBM predictor Importance

Many machine learning models have built-in approaches for measuring the aggregate effect of the predictors on the model [24]. In boosted trees, for each tree, the prediction accuracy on the out-of-bag portion of the data is recorded. Then the same is done after permuting each predictor value. The difference between the two accuracies are then averaged over all trees, and normalized by the standard error. The importances are summed over each boosting iteration [29].

Figure 13 shows the top 20 predictors based on GBM ensemble built with all predictors. The measure of importance

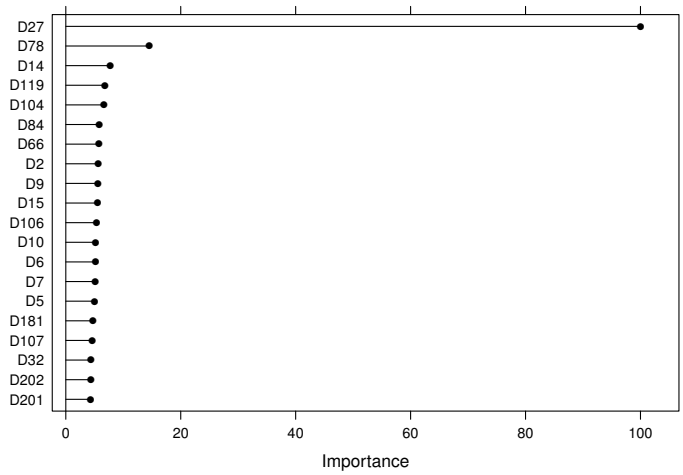

Fig. 13. Top 20 predictors based on model-based predictor importance

is scaled to have a maximum value of 100. By choosing a measure of importance greater than 1 as our cutoff, 132 predictors are selected to build the GBM ensemble.

Figure 14 shows the tuning of the GBM hyperparameters using the selected 132 predictors. The optimized model is shown in Table VI.

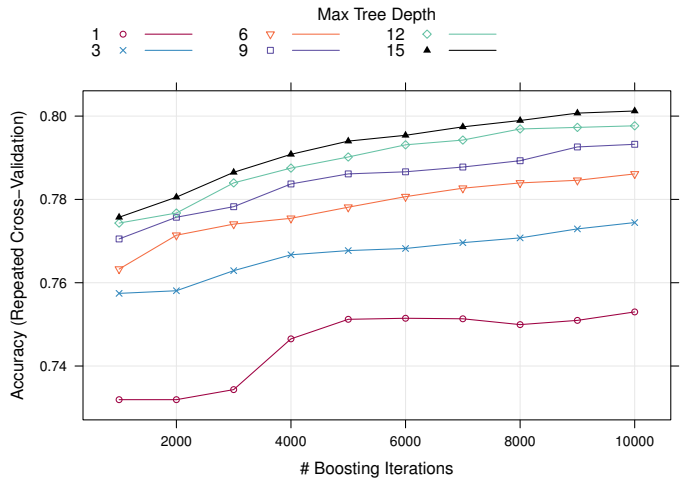

Fig. 14. Stochastic GBM classification accuracy versus the tuning hyperparameters using 132 predictors (with Model-based importance > 1)

TABLE VI
OPTIMIZED ENSEMBLE USING 132 PREDICTORS (GBM IMPORTANCE > 1)

No. of trees	Interaction depth	CV accuracy	Accuracy SD
10000	15	80.1%	2.9%

IV. PERFORMANCE EVALUATION

The ensembles' in-sample and out-of-sample performance are shown in Table VII and Table VIII respectively. These results show that ensembles built using features selection/reduction have comparable accuracy to the ensemble built using all predictors.

TABLE VII
ENSEMBLES' IN-SAMPLE PERFORMANCE

Metric	Ensemble 1 (all predictors)	Ensemble 2 (PCA 95%)	Ensemble 3 (PCA Kaiser)	Ensemble 4 (ROC area)	Ensemble 5 (Relief score)	Ensemble 6 (GBM importance)
Accuracy	99.62%	98.13%	79.93%	91.32%	90.40%	96.53%
95% Confidence interval	(99.30, 99.82)%	(97.54, 98.62)%	(78.35, 81.45)%	(90.17, 92.37)%	(89.21, 91.50)%	(95.76, 97.20)%
Sensitivity	99.72%	98.81%	82.94%	93.19%	92.49%	97.33%
Specificity	99.50%	97.34%	76.37%	89.10%	87.94%	95.59%
Kappa	99.23%	96.24%	59.47%	82.47%	80.62%	93.01%
No Information Rate (NIR)	54.23%	54.23%	54.23%	54.23%	54.23%	54.23%
P-Value [Accuracy > NIR]	$< 2 \times 10^{-16}$	$< 2 \times 10^{-16}$	$< 2 \times 10^{-16}$	$< 2 \times 10^{-16}$	$< 2 \times 10^{-16}$	$< 2 \times 10^{-16}$
Positive predictive value	99.58%	97.78%	80.61%	91.02%	90.08%	96.32%
Negative predictive value	99.67%	98.57%	79.07%	91.70%	90.81%	96.80%
Prevalence	54.23%	54.23%	54.23%	54.23%	54.23%	54.23%
Detection rate	54.07%	53.58%	44.97%	50.53%	50.15%	52.78%
Detection prevalence	54.30%	54.80%	55.79%	55.52%	55.67%	54.80%
Balanced accuracy	99.61%	98.07%	79.65%	91.14%	90.21%	96.46%

TABLE VIII
ENSEMBLES' OUT-OF-SAMPLE PERFORMANCE

Metric	Ensemble 1 (all predictors)	Ensemble 2 (PCA 95%)	Ensemble 3 (PCA Kaiser)	Ensemble 4 (ROC area)	Ensemble 5 (Relief score)	Ensemble 6 (GBM importance)
Accuracy	79.82%	78.40%	71.56%	79.11%	78.84%	78.67%
95% Confidence interval	(77.36, 82.23)%	(75.88, 80.77)%	(68.82, 74.18)%	(76.62, 81.45)%	(76.34, 81.20)%	(76.15, 81.03)%
Sensitivity	82.62%	82.46%	76.56%	81.80%	82.13%	81.64%
Specificity	76.50%	73.59%	65.63%	75.92%	74.95%	75.15%
Kappa	59.26%	56.30%	42.41%	57.84%	57.26%	56.92%
No Information Rate (NIR)	54.22%	54.22%	54.22%	54.22%	54.22%	54.22%
P-Value [Accuracy > NIR]	$< 2 \times 10^{-16}$	$< 2 \times 10^{-16}$	$< 2 \times 10^{-16}$	$< 2 \times 10^{-16}$	$< 2 \times 10^{-16}$	$< 2 \times 10^{-16}$
Positive predicted value	80.64%	78.72%	72.52%	80.10%	79.52%	79.55%
Negative predicted value	78.80%	77.98%	70.27%	77.89%	74.95	77.56%
Prevalence	54.22%	54.22%	54.22%	54.22%	54.22%	54.22%
Detection rate	44.80%	44.71%	41.51%	44.36%	44.53%	44.27%
Detection prevalence	55.56%	56.80%	57.24%	55.38%	56.00%	55.64%
Balanced accuracy	79.56%	78.03%	71.09%	78.86%	78.54%	78.39%

V. COMPUTATIONS TIME

Computations cost is an important factor in model building especially for high-dimensional and big data applications. The time required to build and tune our ensembles using Amazon cloud computing (32 cores) is: ensemble 1 ≈ 43.2 hours, ensemble 2 ≈ 2.9 hours, ensemble 3 ≈ 3.58 minutes, ensemble 4 ≈ 31.32 minutes, ensemble 5 ≈ 32.06 minutes, and ensemble 6 ≈ 1.1 hours. It is evident that the use of features selection/reduction methods significantly reduced the time required to build and tune our GBMs ensembles.

VI. FUSING ENSEMBLES' DECISIONS

Feature selection/reduction methods, used to build the ensembles 2 to 6, use different approaches to quantify the strength of the relationship between the predictors and the outcome. Although, there is some overlap between the selected predictors, the built ensembles are quite diverse as shown in Section III. In this section, we fuse the decisions made by ensembles 2 to 6 using two approaches, namely a majority vote and an optimized feedforward neural network. For unbiased results, we use stratified random sampling to split the test dataset to training and testing datasets, as shown in Figure 15, to build and test a feedforward neural network [30].

A. Majority Vote

The biological response (*Active*/*Inactive*) of molecules is predicted (for the test dataset) by the five ensembles (ensemble 2 to 6). Then, a final prediction is made using a majority vote (i.e. the molecule is *Active* if at least three ensembles predicted *Active*, otherwise, the molecule is *Inactive*). The prediction accuracy of ensembles 1 to 6 are 79.53%, 78.04%, 72.7%, 79.82%, 79.23%, and 79.53% respectively. The prediction accuracy using the majority vote is 79.53%. The fusion accuracy is better than ensembles 2 to 5, and equals to ensembles 1 and 6.

B. Optimized feedforward Neural Network

The predictions of ensembles 2 to 6 on the training dataset are used to build and tune a feedforward neural network.

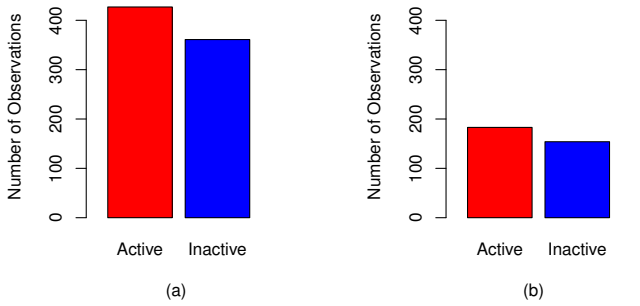

Fig. 15. Class distribution in training data (a) and testing data (b) used to train and test the feedforward neural network

As shown in Figure 16, the optimized feedforward Neural network has 25 hidden units, and weight decay of 1. The fusion model is applied to the test dataset, and compared to the predictions of ensembles. The optimized feedforward neural network accuracy is 80.12% with 95% confidance interval of (75.45%, 84.25%), which outperforms all ensemables, and majority vote fusion.

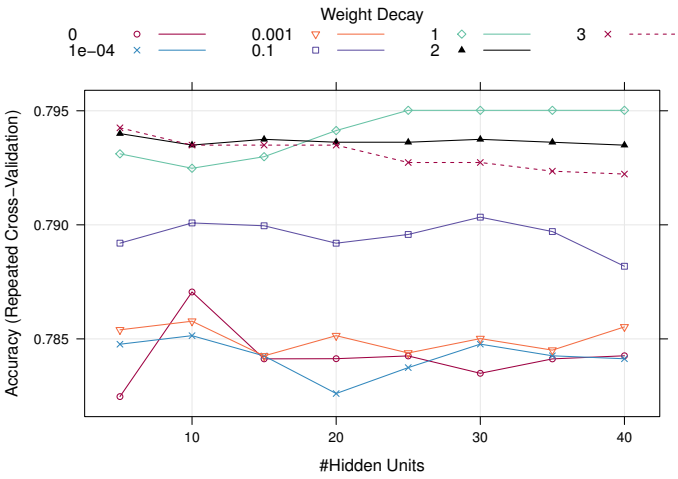

Fig. 16. Tuning the feedforward neural network

VII. Conclusion

The ability to accurately predict the biological activity of molecules, and understand the rationale behind those predictions are of great value to the pharmaceutical industry. Using different feature selection/reduction techniques, diverse and optimized Tree-based Gradient Boosting Machines were built for real, high-dimensional molecules dataset. The results show that by using these techniques, the computations time for building an optimized GBMs using all predictors is significantly reduced at a slight drop in prediction accuracy. A better prediction accuracy is obtained by fusing ensembles decisions using a majority vote and an optimized feedforward neural network. For future work, more feature selection, models, and fusion techniques will be investigated.

References

[1] D. J. Abraham, *Burger's medicinal chemistry and drug discovery*. Wiley Online Library, 2003.

[2] A. N. Jain, K. Koile, and D. Chapman, "Compass: predicting biological activities from molecular surface properties. performance comparisons on a steroid benchmark," *Journal of Medicinal Chemistry*, vol. 37, no. 15, pp. 2315–2327, 1994.

[3] A. Natekin and A. Knoll, "Gradient boosting machines, a tutorial," *Frontiers in neurorobotics*, vol. 7, 2013.

[4] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[5] Y. Freund and R. E. Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting," in *Computational learning theory*. Springer, 1995, pp. 23–37.

[6] J. Friedman, T. Hastie, R. Tibshirani *et al.*, "Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors)," *The annals of statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[7] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of Statistics*, pp. 1189–1232, 2001.

[8] ——, "Stochastic gradient boosting," *Computational Statistics & Data Analysis*, vol. 38, no. 4, pp. 367–378, 2002.

[9] L. Breiman, "Using adaptive bagging to debias regressions," Technical Report 547, Statistics Dept. UCB, Tech. Rep., 1999.

[10] G. R. with contributions from others, *gbm: Generalized Boosted Regression Models*, 2013, r package version 2.1. [Online]. Available: http://CRAN.R-project.org/package=gbm

[11] G. Ridgeway, "Generalized boosted models: A guide to the gbm package," *Update*, vol. 1, no. 1, 2007.

[12] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2013. [Online]. Available: http://www.R-project.org/

[13] M. K. C. from Jed Wing, S. Weston, A. Williams, C. Keefer, A. Engelhardt, T. Cooper, Z. Mayer, and the R Core Team, *caret: Classification and Regression Training*, 2014, r package version 6.0-24. [Online]. Available: http://CRAN.R-project.org/package=caret

[14] X. Robin, N. Turck, A. Hainard, N. Tiberti, F. Lisacek, J.-C. Sanchez, and M. Müller, "proc: an open-source package for r and s+ to analyze and compare roc curves," *BMC Bioinformatics*, vol. 12, p. 77, 2011.

[15] W. N. Venables and B. D. Ripley, *Modern Applied Statistics with S*, 4th ed. New York: Springer, 2002, iSBN 0-387-95457-0. [Online]. Available: http://www.stats.ox.ac.uk/pub/MASS4

[16] R. Analytics, *doMC: Foreach parallel adaptor for the multicore package*, 2014, r package version 1.3.3. [Online]. Available: http://CRAN.R-project.org/package=doMC

[17] Kaggle.com. (2012) Competition: Predicting a biological response. [Online]. Available: http://www.kaggle.com/c/bioresponse

[18] J. Shlens, "A tutorial on principal component analysis," *arXiv preprint arXiv:1404.1100*, 2014.

[19] P. R. Peres-Neto, D. A. Jackson, and K. M. Somers, "How many principal components? stopping rules for determining the number of non-trivial axes revisited," *Computational Statistics & Data Analysis*, vol. 49, no. 4, pp. 974–997, 2005.

[20] L. Guttman, "Some necessary conditions for common-factor analysis," *Psychometrika*, vol. 19, no. 2, pp. 149–161, 1954.

[21] D. G. Altman and J. M. Bland, "Diagnostic tests 3: receiver operating characteristic plots." *BMJ: British Medical Journal*, vol. 309, no. 6948, p. 188, 1994.

[22] C. D. Brown and H. T. Davis, "Receiver operating characteristics curves and related decision measures: A tutorial," *Chemometrics and Intelligent Laboratory Systems*, vol. 80, no. 1, pp. 24–38, 2006.

[23] T. Fawcett, "An introduction to roc analysis," *Pattern recognition letters*, vol. 27, no. 8, pp. 861–874, 2006.

[24] M. Kuhn and K. Johnson, *Applied predictive modeling*. Springer, 2013.

[25] K. Kira and L. A. Rendell, "The feature selection problem: Traditional methods and a new algorithm," in *AAAI*, 1992, pp. 129–134.

[26] M. Robnik-Šikonja and I. Kononenko, "Theoretical and empirical analysis of relieff and rrelieff," *Machine learning*, vol. 53, no. 1-2, pp. 23–69, 2003.

[27] I. Kononenko, "Estimating attributes: analysis and extensions of relief," in *Machine Learning: ECML-94*. Springer, 1994, pp. 171–182.

[28] M. Robnik-Šikonja and I. Kononenko, "An adaptation of relief for attribute estimation in regression," in *Machine Learning: Proceedings of the Fourteenth International Conference (ICMLŠ97)*, 1997, pp. 296–304.

[29] M. Kuhn, "Variable importance using the caret package," 2012.

[30] G. Bebis and M. Georgiopoulos, "Feed-forward neural networks," *Potentials, IEEE*, vol. 13, no. 4, pp. 27–31, 1994.