# The Ensemble Bridge Algorithm: A New Modeling Tool for Drug Discovery Problems

Mark Culp*

Department of Statistics, West Virginia University, Morgantown, West Virginia 26506

Kjell Johnson*

Pfizer Global Research and Development, Groton, Connecticut 06340

George Michailidis

Department of Statistics, University of Michigan, Ann Arbor, Michigan 48109-1107

Received September 9, 2009

Ensemble algorithms have been historically categorized into two separate paradigms, boosting and random forests, which differ significantly in the way each ensemble is constructed. Boosting algorithms represent one extreme, where an iterative greedy optimization strategy, weak learners (e.g., small classification trees), and stage weights are employed to target difficult-to-classify regions in the training space. On the other extreme, random forests rely on randomly selected features and complex learners (learners that exhibit low bias, e.g., large regression trees) to classify well over the entire training data. Because the approach is not targeting the next learner for inclusion, it tends to provide a natural robustness to noisy labels. In this work, we introduce the ensemble bridge algorithm, which is capable of transitioning between boosting and random forests using a regularization parameter $\nu \in [0,1]$. Because the ensemble bridge algorithm is a compromise between the greedy nature of boosting and the randomness present in random forests, it yields robust performance in the presence of a noisy response and superior performance in the presence of a clean response. Often, drug discovery data (e.g., computational chemistry data) have varying levels of noise. Hence, this method enables a practitioner to employ a single method to evaluate ensemble performance. The method's robustness is verified across a variety of data sets where the algorithm repeatedly yields better performance than either boosting or random forests alone. Finally, we provide diagnostic tools for the new algorithm, including a measure of variable importance and an observational clustering tool.

## INTRODUCTION

Classification algorithms have had a long history in statistics and machine learning. Over the years, various parametric and nonparametric approaches have been proposed, for example, linear discriminant analysis, nearest neighbor classifiers, perceptrons, and classification trees, just to name a few.[1] Historically, these traditional techniques are used to fit a single model to a set of data. The resulting model is used to predict future observations and to interpret relationships between features (variables) and the response.

In the 1990s, ensemble techniques like boosting and bagging (and later random forests) began to appear and demonstrated superior performance to traditional classification techniques. Since their introduction, boosting and random forests have become mainstream modeling tools with performance often exceeding the performance of traditional methods.[2] The rise in popularity of these methods can be seen in the increase in research and publication about them. Clearly, boosting and random forests ensemble methods have proved quite successful in practice. For example, applications of these methods have ranged from compound classification,[2,3] QSAR modeling,[4] virtual screening,[5,6] and chemical safety evaluation.[7] Before we join these two methods, however, we provide a brief overview of each one and their distinct characteristics.

**Table 1.** AdaBoost Algorithm

| | |
|---|---|
| (1) | initialize weights, $w_{1,i} = 1/n$ |
| (2) | **for** $m \in \{1,\cdots,M\}$ **do** |
| | (a) fit $y = f_m(x)$ as the base weighted classifier |
| | (b) compute $err_m = \sum_{i=1}^{N} w_{m,i} I\{y_i \neq f_m(x_i)\}$ and $\alpha_m = \log((1 - err_m)/(err_m))$ |
| | (c) updates weights as $w_{m+1,i} = w_{m,i} \exp\{\alpha_m I\{y_i \neq f_m(x_i)\}\}$ scaled to sum to one $\forall i \in \{1,\cdots,N\}$ |
| (3) | **end for** |
| (4) | output the final classifier given by $F(x) = sign(\sum_{m=1}^{M} \alpha_m f_m(x))$ |

In the early 1990s, boosting algorithms appeared,[8−11] motivated by probably approximately correct (PAC) learning theory,[12] in which a number of weak classifiers (a classifier that predicts marginally better than random) are combined (or boosted) to produce an ensemble classifier with a superior generalized misclassification error rate. Freund and Schapire's[10] AdaBoost algorithm (Table 1) provided a practical implementation of Kerns and Valiant's[13] concept of boosting a weak learner into a strong learner. In short, boosting ensemble methods iteratively generate base learners, using observation weights at each iteration to target misclassified observations from the previous iteration's base learner. The

* Corresponding author e-mail: mculp@stat.wvu.edu (M.C.); kjell.johnson@pfizer.com (K.J.).

10.1021/ci9003392 
Published on Web 02/01/2010

performance of the ensemble at each iteration is described by a stage weight. Stage weights near zero imply that the current learner is an inadequate model for the entire data, while weights close to 1 imply that the learner is a very good model of the data. Thus, the stage weights make boosting robust to overfitting. The final classifier is the sign ($\pm 1$) of the linear combination of the stage weights and corresponding base learners. Because boosting uses past information to guide the construction of the current learner, the next learner is chosen deterministically on the basis of the previous learners. This deterministic nature can be quite useful for targeting difficult to classify observations; however, a noisy response can have a detrimental effect and worsen the predictive performance of the ensemble.[14,15]

Despite this well-known drawback, boosting has been successfully employed in many classification problems. This success has attracted a lot of attention from both the statistics and the machine learning communities. For example, Friedman et al.[16] connected the AdaBoost algorithm to statistical concepts of loss functions, additive modeling, and logistic regression and showed that boosting can be interpreted as a stagewise additive model. This fundamental understanding of boosting led to several generalizations including the Real AdaBoost, Gentle AdaBoost, and LogitBoost algorithms. In addition to these modifications, several authors revealed that boosting is robust to overfitting.[16−18] That is, as the model complexity grows (i.e., the number of training iterations increases), the test set error does not increase. This characteristic is extremely important for any learning method and ensures that the learner will be generalizable to other data sets stemming from a feature space similar to the training set.

In the mid-1990s, another class of ensemble methods was independently developed. Bootstrap aggregation (bagging), developed by Breiman,[19] used multiple, independent versions of the training data to generate an ensemble. In short, at each iteration of bagging, a bootstrap sample of the training set is selected and, in turn, is used to train the learner. After a prespecified number of iterations, the process is stopped. All learners in the ensemble are used to predict each observation, and the predictions are averaged to produce a final value. Contrary to intuition, randomness via sampling with replacement of the training set yields an ensemble that significantly improved on misclassification rates.

Upon unveiling this technique, several authors extended Breiman's concept of adding randomness into the learning process. Because classification trees were a popular learner for bagging,[14] Dietterich introduced the idea of random split selection, where trees are built using a random subset of the top $k$ predictors at each split in the tree. Another approach was to build entire trees on the basis of random subsets of descriptors.[20,21] These generalizations to the original bagging algorithm were incorporated into a unified algorithm called random forests.[15] The random forests algorithm is presented in Table 2.

Breiman[15] showed that the linear combination of many independent learners reduces the variance of the overall ensemble relative to any individual learner in the ensemble. Random forests achieve this variance reduction by selecting strong, complex learners that exhibit low bias. This ensemble of many independent, strong learners yields an improvement in misclassification error rates. Because each learner is selected independently of all previous learners, random forests is robust to a noisy response.[15] At the same time, the independence of learners often underfits data when the response is not noisy, which is most likely due to the lack of stage weights.

**Table 2.** Random Forests Algorithm with Bootstrapping

| | |
|---|---|
| (1) | **for** $m \in \{1,\cdots,M\}$ **do** |
| | (a) select a bootstrap sample, $x_{\text{boot}}$ |
| | (b) fit $y = f_m(x_{\text{boot}})$ |
| (2) | **end for** |
| (3) | output the final classifier given by $F(x) = sign(\sum_{m=1}^{M} f_m(x))$ |

While boosting and random forests methods have been extensively researched and used in applied work, each method has historically been considered as a separate modeling paradigm under the ensemble framework. In addition to working under separate learning paradigms, each of these methods have fundamentally opposite operating characteristics: random forests provides a lower testing error in the presence of a noisy response, while boosting provides lower testing error in the presence of a clean response. If these methods can be joined, it seems plausible that the resulting hybrid algorithm may capture the benefits of both, while producing a more predictive model.

To connect boosting and random forests and to harness the advantages of both methods, we present an algorithm called Ensemble Bridge (EBridge), which acts as a bridge between random forests and boosting. EBridge is characterized by a regularization parameter $\nu \in [0,1]$, whose extreme values recover random forests ($\nu = 0$) and boosting ($\nu = 1$). This parameter allows EBridge to balance between the random feature strategy of random forests and the weighting strategy of boosting. Moreover, we will show that the EBridge algorithm retains the beneficial characteristics of both random forests and boosting. Finally, we have empirically found that EBridge is competitive with and in many cases significantly outperforms both boosting and random forests.

The remainder of the Article is organized as follows: Section 2 introduces the EBridge algorithm; Section 3 illustrates the algorithm's performance on a number of proprietary and publicly available chemistry data sets; and Section 4 provides discussion and concluding remarks.

## BRIDGING BOOSTING AND RANDOM FORESTS

To connect boosting and random forests, the two fundamentally opposite operating characteristics of each algorithm must be joined. First, the algorithm must employ a learner that spans the weak (boosting) to strong (random forests) range. In addition, the algorithm must be able to control the degree of correlation among learners across iterations: boosting learns using the past history of the ensemble, while each iteration in a random forest is independent. Although boosting and random forests operate under opposite conditions, it is possible to select mechanisms that connect the methods.

First, we must identify a learner that is flexible enough to be both weak and strong. Conveniently, as mentioned above, classification trees are the most popular learner for both boosting and random forests.[19,22] Moreover, a classification tree can produce both weak and strong learners: as its size

increases, the learning strength of the tree increases. Boosting, therefore, favors trees with small depth (size), while random forests favors trees with large depth. Hence, a method that connects boosting and random forests using classification trees must control tree depth.

Because boosting performs optimally with small trees, and random forests performs optimally with large trees, we introduce a tuning parameter, $\nu \in [0,1]$, to control tree depth. Specifically, let $a$ be the desired tree depth for boosting (typically small) and $b$ be the desired tree depth for bagging (typically large). The depth of the tree is then determined by $d(\nu) = \lfloor a\nu \rfloor + \lfloor b(1 - \nu) \rfloor$. Therefore, this adjustment achieves the desired result where larger trees are used for small $\nu$ (random forests) and smaller trees are used for large $\nu$ (boosting).

In addition to controlling tree depth, the algorithm must construct a process that can control the degree of learning from past iterations of the algorithm. Again, conveniently, trees can control the degree of learning through observation weights. Recall that boosting uses past information to guide the development of the current learner, while random forests does not. To capture past information, boosting updates the weights of each observation at each iteration. Poorly predicted observations get increased weight in the next iteration, while correctly predicted observations get less. Alternatively, random forests use randomly selected features at each iteration to ensure that each tree in the ensemble is independent. This implies that all observations receive the same weight at each iteration. The same tuning parameter, $\nu$, used above for determining tree depth can also be used to control the degree of randomness. For a given iteration, this parameter will control the extent to which randomly selected features are used in the algorithm. As $\nu$ increases to 1, there is a smaller degree of randomly selected features and a larger degree of correlation among trees. On the other hand, as $\nu$ decreases to 0, there is a larger degree of randomly selected features and a smaller degree of correlation among trees.

To control the degree of correlation across iterations, we introduce the controlled bootstrap, which uses $\nu$ to control the degree of randomness in the selection of observations for tree building. This mechanism combines bootstrapping and random permutation in two separate phases. First, $n - \lfloor \nu n \rfloor$ cases are sampled with replacement using the boosting observations weights as the probabilities, forming a bootstrapped sample U. Second, a new permuted sample B of size $\lfloor \nu n \rfloor$ is taken from the remaining cases without replacement. The final sample $X_{U \cup B}$ of size $n$ is constructed to train a tree classifier at each iteration. This sampling strategy controls the effect of randomness and achieves random forests when $\nu = 0$ and boosting when $\nu = 1$.

To control feature selection, we define $k(\nu) = \lfloor \theta(1 - \nu) \rfloor + \lfloor p\nu \rfloor$, where $p$ is the total number of measured features, and $\theta = \sqrt{p}$ for classification problems and $\theta = p/3$ for regression problems. This control would then be used for either random split selection or random feature selection depending on the context.

## THE ENSEMBLE BRIDGE ALGORITHM

The classification problem for ensemble classifiers is formulated as follows: a training data set comprising $N$ observations $\{y_i, x_i\}$ is available, where $y_i \in \{-1,1\}$ and $x_i$ is a $p$-dimensional vector of categorical and/or numerical measurements. The objective is to define a committee of classification trees where weighted (sometimes uniform) majority voting is adopted to determine the class for a new observation, for which only the feature vector (i.e., $x_{N+1}$) is observed.

**Table 3.** Ensemble Bridge Algorithm

(1) select the value of the tuning (regularization) parameter $\nu \in [0,1]$
(2) initialize weights, $w_{1,i} = 1/n$
(3) **for** $m \in \{1,\cdots,M\}$ **do**
  (a) fit $y = f_m(x;\nu)$ as the base weighted classifier using the nongreedy bootstrap, $k(\nu)$, and $d(\nu)$
  (b) compute $err_m = \sum_{i=1}^{N} w_{m,i} I\{y_i \neq f_m(x_i;\nu)\}$ and $\alpha_m = \log((1 - err_m)/(err_m))$
  (c) let $\alpha_m(\nu) = (1 - \nu) + \nu\alpha_m$
  (d) updates weights as $w_{m+1,i} = w_{m,i} \exp\{\nu\alpha_m I\{y_i \neq f_m(x_i;\nu)\}\}$ scaled to sum to one $\forall i \in \{1,\cdots,N\}$
(4) **end for**
(5) output the final classifier given by $F_v(x) = sign(\sum_{m=1}^{M} \alpha_m(\nu) f_m(x;\nu))$

The Ensemble Bridge Algorithm is presented in Table 3. If one were to fit the EBridge Algorithm without the randomly selected features and without step 5, then the algorithm would be identified as AdaBoost subject to regularization with shrinkage parameter $\nu$.[23] Friedman notes that as regularization strengthens ($\nu$ decreases), the number of iterations must increase to improve predictive performance. However, before we further discuss this relationship, strategies for choosing $\nu$, and extensions to the EBridge algorithm, we provide empirical evidence that assesses the performance of the proposed procedure.

## PERFORMANCE OF THE EBRIDGE ALGORITHM

To assess the performance of the EBridge algorithm and to evaluate the effect of the learning rate $\nu$ on the testing error, we have collected a range of external publicly available and internal proprietary data sets. The external data consist of eight sets evaluated by Bruce et al.[24] and one analyzed by Kuhn.[25] Specifically, the data sets from Bruce et al.[24] are angiotensin-converting enzyme (ACE) inhibitors, acetylcholinesterase (AchE) inhibitors, benzodiazepine receptor (BZR) ligands, cyclooxygenase-2 (COX2) inhibitors, dihydrofolate reductase (DHFR) inhibitors, glycogen phosphorylase b (GPB) inhibitors, thermolysin (THER) inhibitors, and thrombin (THR) inhibitors. Similar to Bruce et al.,[24] we classify compounds using the median activity as a threshold between inactive and active compounds and use the same structural descriptors. While we have selected several data sets where performance has been evaluated, our purpose is not to detract from this excellent work, but to illustrate the EBridge method and to identify data scenarios where one may consider using this method. The final external data set was analyzed by Kuhn[25] and was originally presented by Kazius et al.[26] This data set (Kazius) consists of 4335 compounds of which 2400 were identified as mutagenic based on an Ames test. For these compounds, 1579 dragonX[27] descriptors were generated. We have also applied the EBridge algorithm to three internal, proprietary data sets. The first set, Pfizer1, consists of 5631 compounds, which were submitted to an in-house solubility screen. Based on this screen, 3493 compounds were classified as soluble and

**Table 4.** Average Accuracy of the EBridge Algorithm for Different Values of the Learning Rate $\nu$ with the Ensemble Size Set to 50[a]

| | method | | | | | | |
|---|---|---|---|---|---|---|---|
| data set | single tree | $\nu = 0$ | $\nu = 0.01$ | $\nu = 0.05$ | $\nu = 0.10$ | $\nu = 0.50$ | $\nu = 1$ |
| ACE | 0.76 | 0.80 | 0.80 | **0.81** | 0.81 | 0.80 | 0.79 |
| AchE | 0.64 | 0.67 | 0.67 | 0.67 | 0.68 | 0.68 | **0.69** |
| BZR | 0.73 | 0.73 | 0.74 | **0.74** | 0.74 | 0.71 | 0.71 |
| COX2 | 0.68 | 0.68 | 0.68 | **0.70** | 0.69 | 0.67 | 0.68 |
| DHFR | 0.82 | 0.84 | 0.84 | **0.85** | 0.85 | 0.84 | 0.84 |
| GPB | 0.62 | 0.75 | 0.74 | 0.74 | 0.75 | 0.76 | **0.76** |
| THER | 0.61 | 0.71 | 0.72 | 0.72 | **0.73** | 0.73 | 0.72 |
| THR | 0.61 | 0.65 | 0.66 | **0.66** | 0.66 | 0.66 | 0.65 |
| Kazius | 0.76 | 0.81 | 0.81 | 0.82 | **0.82** | 0.81 | 0.79 |
| Pfizer1 | 0.76 | 0.82 | 0.81 | **0.82** | 0.81 | 0.81 | 0.80 |
| Pfizer2 | 0.85 | 0.90 | 0.90 | 0.91 | **0.91** | 0.91 | 0.91 |
| Pfizer3 | 0.68 | 0.76 | 0.76 | 0.78 | **0.78** | 0.78 | 0.77 |

[a] Notice that for all but two data sets, the lowest test error lies between random forests ($\nu = 0$) and boosting ($\nu = 1$). Further, small values of $\nu$ (0.05 or 0.10) perform optimally for 10 of 12 data sets.

**Table 5.** Average Kappa of the EBridge Algorithm for Different Values of the Learning Rate $\nu$ with the Ensemble Size Set to 50[a]

| | method | | | | | | |
|---|---|---|---|---|---|---|---|
| data set | single tree | $\nu = 0$ | $\nu = 0.01$ | $\nu = 0.05$ | $\nu = 0.10$ | $\nu = 0.50$ | $\nu = 1$ |
| ACE | 0.52 | 0.60 | 0.59 | **0.62** | 0.61 | 0.60 | 0.57 |
| AchE | 0.28 | 0.34 | 0.35 | 0.35 | 0.36 | 0.36 | **0.38** |
| BZR | 0.47 | 0.46 | 0.47 | **0.48** | 0.47 | 0.43 | 0.42 |
| COX2 | 0.37 | 0.36 | 0.36 | **0.39** | 0.38 | 0.35 | 0.37 |
| DHFR | 0.65 | 0.67 | 0.69 | **0.70** | 0.70 | 0.68 | 0.67 |
| GPB | 0.25 | 0.50 | 0.48 | 0.49 | 0.50 | 0.51 | **0.52** |
| THER | 0.24 | 0.42 | 0.44 | 0.44 | **0.46** | 0.45 | 0.44 |
| THR | 0.23 | 0.32 | 0.32 | **0.34** | 0.34 | 0.32 | 0.32 |
| Kazius | 0.51 | 0.62 | 0.61 | 0.62 | **0.63** | 0.61 | 0.58 |
| Pfizer1 | 0.22 | 0.30 | 0.30 | 0.28 | 0.29 | 0.31 | **0.31** |
| Pfizer2 | 0.68 | 0.79 | 0.79 | 0.81 | **0.81** | 0.81 | 0.80 |
| Pfizer3 | 0.21 | 0.46 | 0.45 | 0.50 | 0.52 | **0.52** | 0.49 |

[a] Notice that for all but three data sets, the highest kappa value lies between random forests ($\nu = 0$) and boosting ($\nu = 1$). Further, small values of $\nu$ (0.05 or 0.10) perform optimally for 8 of 12 data sets.

the rest as insoluble. In addition, 72 continuous structural descriptors were generated for each compound. Pfizer2 consists of 438 compounds, which were classified as being associated with vasculitis ($n = 92$) or not associated with vasculitis ($n = 346$). One hundred sixty-six (166) structural binary fingerprint descriptors were generated for each of the compounds in this set. Finally, Pfizer3 is a subset of 325 dihydrofolate reductase inhibitors that were analyzed in Sutherland et al.[28] The response is pIC50 for rat liver enzyme, which has been categorized into low ($n = 122$) and high ($n = 203$). While these data are publicly available, we generated two hundred twenty-eight (228) proprietary, continuous structural descriptors to be used in the model.

To gauge the performance of EBridge, we consider two measures applied to a withheld testing set. The first measure is accuracy on the testing set, while the second measure is kappa.[29] The kappa statistic adjusts for the unbalanced classes often present in these data sets, by comparing observed agreement to expected agreement if the classes were balanced (refer to Culp et al.[30] for an in-depth discussion of this measure). For the experiment, each analysis was replicated 50 times, the tree depth was chosen as discussed above, and 60% of the observations were used for training. The nongreedy bootstrap was used for the EBridge algorithm as the random mechanism throughout this Article. This corresponds to a random forests with bootstrapping for $\nu = 0$, and to AdaBoost for $\nu = 1.0$. For this analysis, we have empirically noticed that $a = 4$ and $b = 8$ performs sufficiently for tree depth as defined above. In this study, all modeling and analysis were performed in R, version 2.10.0.[31]

The results in Table 4 show testing error rates averaged over 50 independently sampled testing data sets for various choices of $\nu$. From these results, we observe that EBridge with $\nu = 0.05$ often yielded the best overall performance and was quite competitive in all cases. The kappa results in Table 5 provide the same general result, showing that $\nu = 0.05$ was dominant in some data sets and was competitive in others. These strong empirical results indicate that the EBridge algorithm is a competitive performer, which typically outperforms either boosting or random forest alone.

## ROBUSTNESS TO NOISE

In general, the performance of boosting is inferior to random forests when the response contains noise.[14,15] Hence, in the presence of a noisy response, we would expect the EBridge algorithm to favor smaller values of $\nu$. To investigate the effect of noise on the selection of $\nu$, we randomly selected 5% (and 10%) of the observations from each data set and permuted the response. Similar to the earlier analyses (Table 4), 50 randomly selected training and testing sets were generated for each data set, where 60% of the data was used for testing, $a = 2$, $b = 7$, and $M = 100$. For this experiment, we are not interested in assessing accuracy directly, because

**Table 6.** Average Percent Increase in Misclassification Error for the EBridge Algorithm for Varying Values of $\nu$[a]

| data set | $\nu = 0.01$ | $\nu = 0.05$ | $\nu = 0.10$ | $\nu = 0.50$ |
|---|---|---|---|---|
| ACE | **7.9** | 17.4 | 10.4 | 15.1 |
| AchE | 2.9 | **2.6** | 4.8 | 4.9 |
| BZR | **5.6** | 7.4 | 11.2 | 10.7 |
| COX2 | **3.9** | 7.0 | 4.7 | 2.5 |
| DHFR | **6.9** | 8.6 | 9.8 | 16.3 |
| GPB | 16.3 | **15.8** | 21.8 | 23.3 |
| THER | 14.7 | 17.6 | 14.8 | **11.0** |
| THR | **5.5** | 7.2 | 9.3 | 8.7 |
| Kazius | **2.1** | 2.9 | 4.8 | 7.3 |
| Pfizer1 | **2.6** | 2.7 | 6.9 | 8.2 |
| Pfizer2 | **4.1** | 5.8 | 6.1 | 8.7 |
| Pfizer3 | **0.8** | 1.5 | 2.2 | 5.4 |

[a] In this case, 5% of the original data was randomly selected, and the response was permuted. EBridge favors smaller values of $\nu$ in the presence of a noisy response.

**Table 7.** Average Percent Increase in Misclassification Error for the EBridge Algorithm for Varying Values of $\nu$[a]

| data set | $\nu = 0.01$ | $\nu = 0.05$ | $\nu = 0.10$ | $\nu = 0.50$ |
|---|---|---|---|---|
| ACE | **13.0** | 18.7 | 16.9 | 21.2 |
| AchE | 15.3 | **11.4** | 15.5 | 12.4 |
| BZR | 9.7 | **8.6** | 12.7 | 14.3 |
| COX2 | **2.5** | 8.2 | 4.4 | 7.7 |
| DHFR | **18.6** | 21.6 | 24.9 | 36.7 |
| GPB | 29.9 | **25.9** | 29.4 | 35.3 |
| THER | **19.9** | 21.3 | 23.1 | 21.7 |
| THR | 18.2 | 21.8 | 17.4 | **15.4** |
| Kazius | **5.2** | 5.6 | 7.8 | 13.8 |
| Pfizer1 | 10.8 | **10.7** | 15.9 | 20.2 |
| Pfizer2 | **7.1** | 8.3 | 10.2 | 14.8 |
| Pfizer3 | **3.2** | 4.4 | 4.7 | 12.6 |

[a] In this case, 10% of the original data was randomly selected, and the response was permuted. EBridge favors smaller values of $\nu$ in the presence of a noisy response.

accuracy will of course be worse than the case where no noise is added to the data. Instead, we want to estimate the percent increase in misclassification relative to the original data and determine the optimal values of $\nu$.

Tables 6 and 7 illustrate the percent increase in misclassification for varying values of $\nu$. For the 5% noise experiment, the optimal value of $\nu$ is smaller for all data sets (Table 6) than the optimal values of $\nu$ from the original, unperturbed data (Table 4). For the 10% noise experiment, the optimal value of $\nu$ is less than or equal to the optimal value of $\nu$ from the original, unperturbed data (Table 4) for 11 or 12 data sets (Table 7).

The overall performance and robustness to noise results confirm that the EBridge algorithm performs quite competitively for varying $\nu$. The value of $\nu = 0.05$ tended to perform quite well in both experiments, but other values also performed well, thus making it difficult to select this value in practice. In section 4.1, we further explore the choice of tree depth size, the impact of $M$, and discuss a strategy for choosing $\nu$ for the proposed EBridge algorithm.

## A STRATEGY FOR ESTIMATING $\nu$

Through several empirical examples, we have demonstrated that small values of $\nu$ have improved performance relative to boosting or random forests. However, this choice of $\nu$ is not optimal for all data sets. To select a near optimal $\nu$, one must also account for tree depth size. If the approximate ensemble size and competitive tree depth size are known for boosting and random forests for a specific data set, then we propose the method in Table 8 for selecting a near optimal choice of $\nu$ and corresponding tree depth size.

**Table 8.** Strategy for Choosing $\nu$

(1) fix $M$ large enough so that the training error for boosting is stabilized
(2) let $a$ be an optimal depth size for boosting (typically 2 or 3)
(3) let $b$ be an optimal depth size for bagging (typically 7−10)
(4) choose $\nu$ on the basis of testing error analysis (vary $\nu \in (0,1)$; as a rule of thumb always try $\nu = 0.1$), while setting the corresponding tree depth size to $d_{a,b}(\nu) = \lfloor a\nu \rfloor + \lfloor b(1 - \nu) \rfloor$

The results in Tables 4−7 empirically confirm that this strategy works with a tree depth of 2 and 7 for boosting and random forests, respectively. The value of $\nu = 0.05$ is quite competitive for these data, which would be considered if one followed the strategy in Table 8.

## VARIABLE IMPORTANCE

Variable importance measures have been constructed for both boosting and random forests. In fact, variable importance can be calculated for any method that employs an ensemble,[16] like EBridge. To summarize the variable importance calculation, each variable is first ranked by its contribution in determining the shape of each classification tree in the ensemble. Next, a score is obtained for each tree in the committee, and each variable's contribution is determined from the score averaged over all of the trees in the ensemble. Hence, the variable importance ranking reflects a variable's association to the underlying response: the higher is the variable importance score, the stronger is the relationship between the variable and the response.

Figure 1 illustrates the variable importance rankings of EBridge for the vasculitis data. Clearly, the importance scheme places a relative ordering across variables, with the top 10 descriptors illustrated in Figure 1B. For proprietary reasons, we cannot unveil the substructures represented in Figure 1B. However, we can say that this substructure ranking appears to be quite relevant for vasculitis. Specifically, for several compounds known to be associated with vasculitis (salicylic acid, thiouracil, warfarin, coumarin, and hydrochlorothiazide), each contains at least two substructures listed in the top 10.

## CLUSTERING

Breiman[15] proposed an interesting idea for using random forests to cluster the data. Essentially, a similarity measure on the observations is constructed from the aggregated predictions in the individual learners. Compounds that fall in the same terminal node have a similarity, $s(i,j)$, equal to the probability class estimate of that node. The $n \times n$ matrix, $S_t \equiv [s(i,j)]_{ij}$, represents this similarity measure for all pairwise observations on tree $t$. Furthermore, the similarity for the ensemble is $S = \sum_t S_t$. To perform clustering, we construct a dissimilarity matrix $D$ from $S$. Generalized principal component analysis is then applied to $D$, and the first two components $Z_1$ and $Z_2$ are computed. Plotting the data on these principal axes yields a two-dimensional

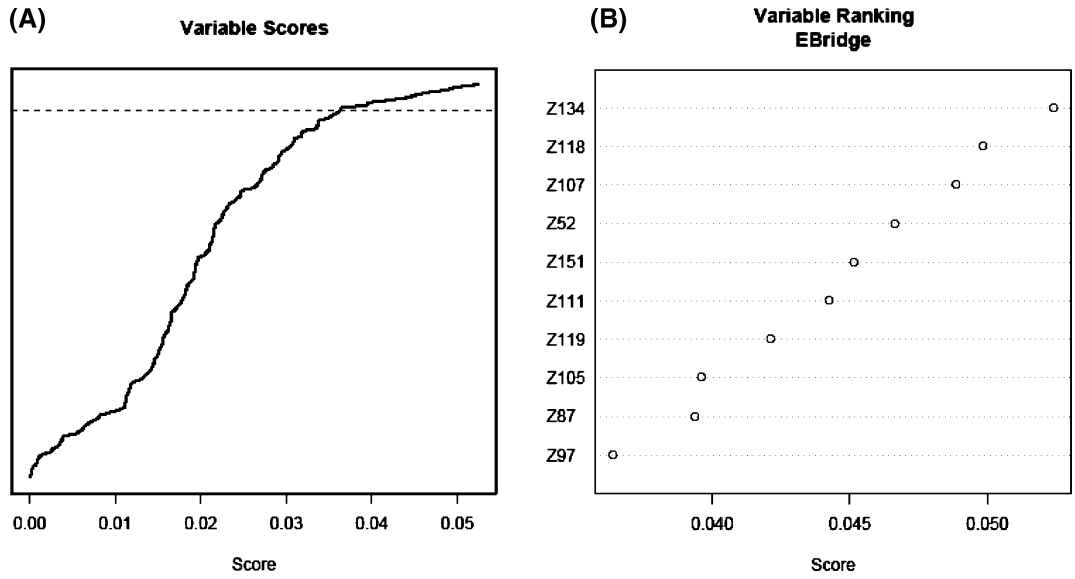


**Figure 1.** (A) EBridge importance ranking for variables in vasculitis data. (B) EBridge importance ranking for top 10 variables in vasculitis data.

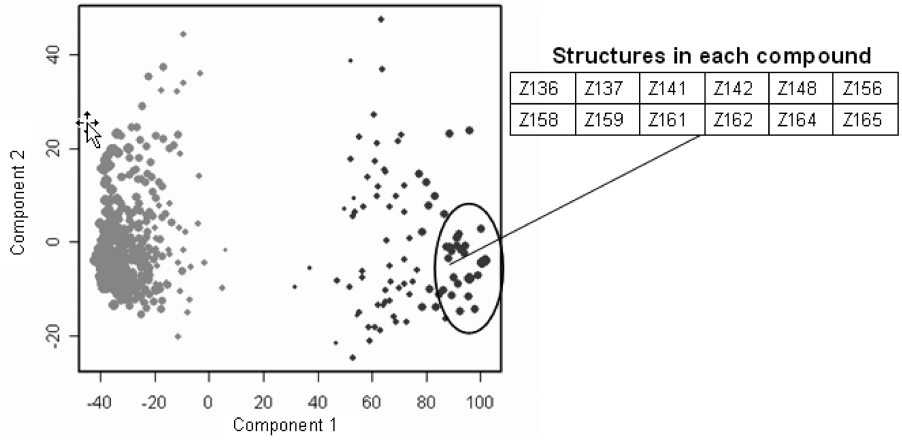


**Figure 2.** Clustering results based on the EBridge algorithm.

clustering of the data, which can be used to obtain a visual assessment of the clustering. This is often referred to as random forest clustering.[15]

The usage of random forest clustering is not restricted to the random forest ensemble and can be applied to any ensemble method where the similarity measure is generalized as $S = \sum_t \alpha_t S_t$, where $\alpha_t$ is the stage weight. We next motivate clustering with EBridge. Recall that random forests do not target difficult to classify observations in the data, which as a corollary tends to lead to training error greater than zero. As a result, the clustering produced by the random forest tends to yield inseparable classes, which render the corresponding random forest clustering difficult to interpret. On the other hand, boosting tends to aggressively target observations in the data, which typically yield a zero misclassification error rate. This results in a visual display of separable classes. However, it is known for boosting that $F(x) \rightarrow \pm\infty$ as the iterations grow.[32] This tends to yield clusters that are highly polarized between the classes with a high degree of separation; that is, class one and class two are separable and far apart, which is not overly interesting. The ensemble bridge algorithm provides the compromise between boosting and random forests, which makes it well suited for clustering.

The results for the first two components are presented in Figure 2 with light gray representing the compounds that are classified as not having vasculitis as a side-effect and the dark gray observations representing compounds that are classified as having vasculitis as a side-effect. The size of the point represents its probability, where larger points have a higher probability. This result provides a clustering of the groups, and the proximity between points represents similarity. The classes are separable for this example, which is important because any subset within a cluster will either represent the side-effect or the no-side-effect group. In this analysis, we first obtained the subset of observations in the vasculitis cluster that had the highest confidence (points in the ellipse in Figure 2). Using these points, we then determined the chemical structures that must be in each one of the compounds in that group. Specifically, every compound that predicted vasculitis with high confidence (i.e., in the ellipse) had 12 common substructures. Cross listing these structures with all of the compounds reveals that 43% of the vasculitis compounds have all of these substructures, while only 16% of the nonvasculitis compounds have all of these substructures. This result would be useful for future analysis of drugs because one would want to screen a current candidate drug with all of these substructures for vasculitis.

As a final comment, it is noteworthy that none of the identified structures resulting from the cluster analysis appear in the variable importance result for EBridge in Figure 1B. This is not surprising for two reasons: (i) the clustering result requires each of these structures to be present in a compound and is not assessing importance of an individual structure relative to the other structures in the data; and (ii) the result only accounts for 93 out of the 438 compounds and hence is not representative of the entire data. In other words, the variable importance in Figure 1B is used to globally assess individual variable importance, while the clustering result can be used to assess local structural relationships. Analyses of both types of results are essential for interpreting a compound structural relationship with respect to the outcome of interest.

## EXTENSION TO GRADIENT EBRIDGE

Friedman et al.[16] showed that the AdaBoost can be presented as a solution to a forward stagewise additive model that optimizes the exponential loss criteria. This stagewise process can be approximated by following a gradient[22] that is defined by a loss function, $L(y,F)$. Examples of commonly used loss functions include $L(y,F) = (y - F)^2$ (squared error) and $L(y,F) = e^{-yF}$ (exponential). Let $F_{m-1}(x) = \sum_{\ell=1}^{m-1} \alpha_\ell f_\ell(x)$ be the ensemble at iteration $m$, and $L(y,F)$ be the desired loss function. Gradient boosting seeks to find the next tree, $f_m(x)$, and stageweight, $\alpha_m$, such that that the loss of adding $\alpha_m f_m(x)$ into the existing ensemble is minimized with respect to $L(y,F)$.

The EBridge algorithm can also be extended using the gradient approach. To begin, we first generate a sequence of weights that are compromised between giving each observation equal weight (random forests) and optimized in the direction of steepest descent for the loss function (gradient boosting), that is,

$$w_i = \frac{1}{N}(1 - \nu) + \nu \left| \frac{\partial L(y, F)}{\partial f} \right|_{F=F_{m-1}}$$

A new tree, $f_m(x)$, is then built using these weights, while the EBridge parameter $\nu$ continues to govern tree depth, $d(\nu)$, and the controlled bootstrap. Once the tree is generated, $\alpha_m$ is chosen to minimize the loss of adding the new term to the ensemble: $\alpha_m = \arg\min_\alpha \sum_i w_i L(y_i, F_{m-1}(x_i) + \alpha f_m(x_i))$. The ensemble is then updated using $\nu$ to regularize as $F_m(x) = F_{m-1}(x) + \tau_m(\nu) f_m(x)$ with $\tau_m(\nu) = (1/M)(1 - \nu) + 2\nu\alpha_m$. The parameter $\tau_m(\nu)$ is chosen such that when $\nu = 0$, $\tau_m(0) = 1/M$, which is necessary to produce a random forest. When $\nu = 1$, $\tau_m(1) = 2\alpha_m$, which results in gradient boosting. The factor of 2 is necessary to get an exact equivalency in the boosting case using exponential loss. In other words, optimization with exponential loss yields the equivalent EBridge algorithm presented above.

As a final comment, this result also establishes a direct connection between EBridge and stochastic gradient boosting.[33] To make this connection, the bootstrap sample in the randomization step for selecting $f_m(x)$ would be replaced with subsampling.

## CONCLUSIONS AND DISCUSSION

A vast majority of the current and historical work surrounding the popular ensemble methods of boosting and random forests is based on the assumption that these methods are mutually exclusive. Thus, much historical work has encompassed head-to-head comparisons of these methods in an ongoing empirical battle of data sets. Depending on the setting, either boosting or random forests can be shown to outperform the other. Generally, boosting provides a superior test set error rate when the data set response is not noisy. On the other hand, random forests provide a superior test set error rate when the data set response contains noise. In this work, we have proposed Ebridge, a flexible and efficient algorithm that naturally bridges the ensemble techniques of boosting and random forests. This algorithm is akin to shrinkage boosting and is regulated by the learning rate parameter $\nu$ that controls the weight distribution at the resampling stage, the tree depth, and the stage weights. Across a wide range of empirical data and descriptor sets presented in this work, the test misclassification performance for EBridge is very competitive, and in most cases is superior to boosting, shrinkage boosting, and random forests. In addition to superior performance, the algorithm has been shown to be naturally robust to moderate amounts of noise, a common problem of many real-world data sets. Furthermore, this tuning parameter can be used to gauge the degree of noise in the data, with smaller values of the parameter being favored with noisier data. While the algorithm's performance is often superior, its interpretation is on-par with that of both boosting and random forests. Specifically, variables can be ranked by their overall contribution to the hybrid ensemble, and samples can be clustered by their internal proximity within the ensemble.

From the practitioner's point-of-view, EBridge is attractive not only because it often outperforms boosting and random forests, but also because it provides a single go-to ensemble method. Hence, users need only to use EBridge, rather than comparing the results of boosting and random forests separately. Moreover, the method can be easily implemented, for example in R, where boosting and random forests routines already exist.

**Supporting Information Available:** Code for EBridge implementation. This material is available free of charge via the Internet at http://pubs.acs.org.

## REFERENCES AND NOTES

(1) Ripley, D. R. *Pattern Recognition and Neural Networks*; Cambridge University Press: Cambridge, 1996.
(2) Svetnik, S.; Wang, T.; Tong, C.; Liaw, A.; Sheridan, R. P.; Song, Q. Boosting: An Ensemble Learning Tool for Compound Classification and QSAR Modeling. *J. Chem. Inf. Model.* **2005**, *45*, 786–799.
(3) Deconinck, E.; Zhang, M. H.; Coomans, D.; Vander Heyden, Y. Classification Tree Models for the Prediction of Blood-Brain Barrier Passage of Drugs. *J. Chem. Inf. Model.* **2006**, *46*, 1410–1419.
(4) Palmer, D. S.; O'Boyle, N. M.; Glen, R. C.; Mitchell, J. B. O. Random Forest Models to Predict Aqueous Solubility. *J. Chem. Inf. Model.* **2007**, *47*, 150–158.
(5) Plewczynski, D.; Spieser, S. A. H.; Koch, U. Assessing Different Classification Methods for Virtual Screening. *J. Chem. Inf. Model.* **2006**, *46*, 1098–1106.
(6) Ehrman, T. M.; Barlow, D. J.; Hylands, P. J. Virtual Screening of Chinese Herbs with Random Forest. *J. Chem. Inf. Model.* **2007**, *47*, 264–278.
(7) Zhang, Q. Y.; Aires-de-Sousa, J. Random Forest Prediction of Mutagenicity from Empirical Physiochemical Descriptors. *J. Chem. Inf. Model.* **2007**, *47*, 1–8.
(8) Schapire, R. E. The Strength of Weak Learnability. *Mach. Learn.* **1990**, *5*, 197–227.

(9) Freund, Y. Boosting a Weak Learning Algorithm by Majority. *Inf. Comput.* **1995**, *121*, 256–285.
(10) Freund, Y.; Schapire, R. E. Adaptive Game Playing Using Multiplicative Weights. *Games Econ. Behav.* **1999**, *29*, 79–103.
(11) Freund, Y.; Schapire, R. E. Experiments with a New Boosting Algorithm. In *Mach. Learn., Proceedings of the Thirteenth International Conference, Bari, Italy*; Saitta, L., Ed.; Morgan Kaufmann: San Francisco, CA, 1996.
(12) Valiant, L. G. A Theory of the Learnable. *Commun. ACM* **1984**, *27*, 1134–1142.
(13) Kearns, M.; Valiant, L. G. Crytographic Limitations on Learning Boolean Formulae and Finite Automata. In *Annual ACM Symposium on Theory of Computing, Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing, Seattle, Washington*; Johnson, D. S., Ed.; ACM Press: New York, 1989.
(14) Dietterich, T. An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization. *Mach. Learn.* **2000**, *40*, 139–158.
(15) Breiman, L. Random Forests. *Mach. Learn.* **2001**, *45*, 5–32.
(16) Friedman, J. H.; Hastie, T.; Tibshirani, R. Additive Logistic Regression: A Statistical View of Boosting. *Ann. Stat.* **2000**, *38*, 337–374.
(17) Schapire, R. E.; Freund, Y.; Bartlett, P.; Lee, W. S. Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods. *Ann. Stat.* **1998**, *26*, 1651–1686.
(18) Freund, Y.; Mansour, Y.; Schapire, R. E. Generalization Bounds for Averaged Classifiers. *Ann. Stat.* **2004**, *32*, 1698–1722.
(19) Breiman, L. Bagging Predictors. *Mach. Learn.* **1996**, *26*, 123–140.
(20) Ho, T. K. The Random Subspace Method for Constructing Decision Forests. *IEEE Trans. Pattern Anal. Mach. Intell.* **1998**, *13*, 340–354.
(21) Amit, Y.; Geman, D. Shape Quantization and Recognition with Randomized Trees. *Neural Comput.* **1997**, *9*, 1545–1588.
(22) Hastie, T.; Tibshirani, R.; Friedman, J. *Elements of Statistical Learning*, 2nd ed.; Springer-Verlag: New York, 2009.
(23) Friedman, J. Greedy Function Approximation: A Gradient Boosting Machine. *Ann. Stat.* **2001**, *29*, 1189–1232.
(24) Bruce, C. L.; Melville, J. L.; Pickett, S. D.; Hirst, J. D. Contemporary QSAR Classifiers Compared. *J. Chem. Inf. Model.* **2007**, *47*, 219–227.
(25) Kuhn, M. Building Predictive Models in R Using the caret Package. *J. Stat. Soft [Online]* 2008; Vol. 28, Issue 5; http://www.jstatsoft.org/v28/i05 (accessed December 14, 2009).
(26) Kazius, J.; McGuire, R.; Bursi, R. Derivation and Validation of Toxicophores for Mutagenizity Prediction. *J. Med. Chem.* **2005**, *48*, 213–320.
(27) Talete, S. R. L. DRAGON for Windows and Linux; URL http://www.talete.mi.it/.
(28) Sutherland, J. J.; O'Brien, L. A.; Weaver, D. F. A Comparison of Methods for Modeling Quantitative Structure-Activity Relationships. *J. Med. Chem.* **2004**, *47*, 5541–5554.
(29) Cohen, J. A Coefficient of Agreement for Nominal Data. *Educ. Psychol. Meas.* **1960**, *20*, 37–46.
(30) Culp, M.; Johnson, K.; Michailidis, G. ada: An R Package for Stochastic Boosting. *J. Stat. Soft.* **2006**, *17*, 1–27.
(31) *R: A Language and Environment for Statistical Computing*, version 2.10.0; R Foundation for Statistical Computing: Vienna, Austria, 2009.
(32) Mease, D.; Wyner, A. Evidence Contrary to the Statistical View of Boosting. *J. Mach. Learn. Res.* **2008**, *9*, 131–156.
(33) Frieman, J. Stochastic Gradient Boosting. *Comp. Stat. Data Anal.* **2002**, *38*, 367–378.

CI9003392