The Knowledge Engineering Review, Vol. 29:1, 78–100. © Cambridge University Press, 2013
doi:10.1017/S0269888913000313
First published online 23 August 2013

Bagging and boosting variants for handling classifications problems: a survey

SOTIRIS B. KOTSIANTIS

Department of Mathematics, Educational Software Development Laboratory, University of Patras, Patras 26504, Greece; e-mail: sotos@math.upatras.gr

Abstract

Bagging and boosting are two of the most well-known ensemble learning methods due to their theoretical performance guarantees and strong experimental results. Since bagging and boosting are an effective and open framework, several researchers have proposed their variants, some of which have turned out to have lower classification error than the original versions. This paper tried to summarize these variants and categorize them into groups. We hope that the references cited cover the major theoretical issues, and provide access to the main branches of the literature dealing with such methods, guiding the researcher in interesting research directions.

1 Introduction

Experimental observations confirm that a given learning algorithm outperforms all others for a specific problem or for a exact subset of the input data, but it is abnormal to find a single expert achieving the best results on the overall problem domain (Kuncheva, 2004). As a consequence the multiple learner systems try to exploit the locally different behavior of the base classifiers to improve the accuracy and the reliability of the overall inductive learning system. There are also hopes that if some learner fails, the overall system can recover the error (Valentini & Masuli, 2002).

The aim of ensemble generation is a set of classifiers such that they are at the same time as different to each other as possible while remaining as accurate as possible when viewed individually (Seni & Elder, 2010). Independence (or diversity) is important because ensemble learning can only get better on individual classifiers when their errors are not correlated (Brown *et al*., 2005; Aksela & Laaksonen, 2006). Obviously these two aims (maximum accuracy of the individual predictors and minimum correlation of incorrect predictions) conflict with each other, as two perfect classifiers would be rather alike, and two maximally different classifiers could not at the same time both be very accurate.

Bagging and boosting are two of the most well-known ensemble learning methods thanks to their theoretical performance guarantees and well-built experimental results. In bagging (Breiman, 1996), the training set is randomly sampled T times with replacement, producing T training sets with sizes equal to the original training set. Since the original set is sampled with replacement, a number of training instances are repeated in the new training sets, and a number of are not present at all. The final classifier is formed by aggregating the classifiers produced by the T training sets. Theoretical results confirm that the expected misclassification probability of bagging has the same bias component as a single bootstrap replicate, while the variance component is decreased by the factor of bootstrap replicates (Fumera *et al*., 2005).

Boosting, on the other hand, induces the ensemble of classifiers by adaptively changing the distribution of the training set based on the accuracy of the earlier created classifiers. There are

https://doi.org/10.1017/S0269888913000313 Published online by Cambridge University Press

several boosting variants; AdaBoost (Freund & Schapire, 1996a, 1997) is the most well-known. These methods assign a weight to each example. Initially, all the examples have the same weight. In each iteration a new classifier is constructed using the base learning method. The construction of the base classifier must take into account the weights distribution. Then, the weight of each example is adjusted, depending on the rightness of the prediction of the base classifier for that example. The final classification is produced from a weighted vote of the base classifiers.

The balance between diversity (Kuncheva & Whitaker, 2003) and accuracy has been examined in various papers including (Dietterich, 2000), which among other results reported that bagged trees are regularly much more uniform than boosted trees. But it was also found that increasing levels of noise cause much more diverse bagged trees, and that bagging starts to outperform boosted trees for high noise levels.

Since bagging and boosting are an effective and open framework, several researchers have proposed their variants, some of which have turned out to have lower classification error than the original versions. This paper tried to summarize these variants and categorize them into groups. In the present paper, we have limited our references to recent refereed journals, published books and conferences. In addition, we have added some references regarding the original work that started the particular line of research under discussion. The reader should be cautioned that a single paper cannot be a comprehensive review of all bagging and boosting variants algorithms. Instead, our goal has been to provide a representative sample of existing lines of research. In each of our listed areas, there are many other papers that more comprehensively detail relevant work.

Our next section covers wide-ranging issues of bagging variants. Boosting variants are covered in section 3. Section 4 deals with the locally application of bagging and boosting techniques. In section 5, bagging and boosting variants for data-streams are presented. Section 6 presents the techniques that combine bagging and boosting techniques. Finally, the last section concludes this work.

2 Bagging

The bagging algorithm (bootstrap aggregating) by Breiman (1996) votes classifiers generated by different bootstrap samples. Given the parameter T which is the number of repetitions, T bootstrap samples $S_1, S_2, \ldots, S_T$ are generated. From each sample S_i a classifier C_i is induced by the same learning algorithm and the final classifier C^* is formed by aggregating T classifiers. A final classification of instance x is generated by a uniform voting scheme on $C_1, C_2, \ldots, C_T$, that is, it is assigned to the class predicted most frequently by these base classifiers. The approach is presented briefly in Figure 1.

The heart of bagging's potential is found in the majority voting over results from a substantial number of bootstrap examples. As a first approximation, the majority voting helps to cancel out the impact of random variation. However, it still puts on views at least two open issues: First, there is no clear understanding yet of the conditions under which bagging outperforms the base classifier. Second, only empirical and rough guidelines have been proposed so far to determine a suitable ensemble size for a task.

(input LS learning set; T number of bootstrap samples; LA learning algorithm output C classifier)
begin
 for i =1 to T do
 begin
  S_i:= bootstrap sample from LS; {sample with replacement}
  C_i:= LA (S_i) ; {generate a base classifier}
 end;{endfor}
C* = $argmax_y \sum_{i=1}^{T} C_i(x = y)$ {the most often predicted class}
end

Figure 1 The bagging algorithm.

https://doi.org/10.1017/S0269888913000313 Published online by Cambridge University Press

With regard to the first issue, at present the main explanation of bagging operation is given in terms of its capability to decrease the variance component of the misclassification probability, which was related by Breiman (1996) to the degree of 'instability' of the base learner, informally defined as the tendency of undergoing large changes in its decision function as a result of small changes in the training set. The more unstable a classifier, the higher the variance component of its misclassification probability and, therefore, the higher improvement attained by bagging. Theoretical investigations of why bagging works has also been found in Friedman and Hall (2007), Buhlmann and Yu (2002), Buja (2006). With the influence equalization viewpoint, bagging is interpreted as a perturbation technique aiming at improving the robustness against outliers (Grandvalet, 2004). According to the usual interplay between bias and variance in non-parametric statistics, the hope of bagging is to earn more by reducing the variance than increasing the bias, so that in general, bagging could pay-off in terms of the mean squared error (Buhlmann, 2012).

With regard to the second issue above, it is well known that bagging misclassification rate tends to an asymptotic value as the ensemble size enlarges. Works in the literature focused on determining the ensemble size sufficient to reach the asymptotic misclassification rate, empirically showing that suitable values are between 10 and 20 depending on the particular dataset and base learner (Bauer & Kohavi, 1999). Fumera *et al.* (2008) applied an analytical framework for the analysis of linearly combined classifiers to ensembles generated by bagging. This provides an analytical model of bagging misclassification probability as a function of the ensemble size. This allowed the authors to derive a novel and theoretically rule for choosing bagging ensemble size according to the dataset characteristics.

Wagging (Bauer & Kohavi, 1999) is a well known variant of bagging; bagging uses re-sampling to get the datasets for training, whereas wagging uses re-weighting for each training example, pursuing the effect of bagging in a different manner. One of the first variants of bagging that uses a sampling ratio different from the standard value is Rvotes (Breiman, 1999). Another variant of bagging—Ivotes sequentially generates small datasets using a more sophisticated sampling. According to this sampling each new training data sample should have $\sim 50\%$ instances that were misclassified by the ensemble including previously generated classifiers (Breiman, 1999). Moreover, random forests (Breiman, 2001) use random subsamples of the training data and randomizing the algorithm for learning base-level classifiers (decision trees).

In our point of view bagging ensemble variants can be categorized in: (i) methods that select classifiers generated by bagging, (ii) methods that use alternative bootstrap sampling schemes, (iii) methods that use different subsets of features, (iv) methods that use different voting rule, (v) methods that change labels of examples by adding a controlled amount of noise, (vi) methods that use heterogeneous components, (vii) methods that are implemented in order to be used to stable learners, (viii) methods that apply bagging based on the characteristics of test instances, and (ix) methods that apply bagging in a online environment.

2.1 Selective bagging variants

Important shortcomings of ensemble methods are the loss of speed in classification with respect to the base classifier and the growth in storage needs with increasing numbers of learners. In order to remedy these drawbacks, one can try to retain just those classifiers that are essential to solve the classification task and eliminate those whose contribution is unnecessary. Ensemble pruning reduces the storage needs and speeds up the classification process.

Selective bagging (or trimmed bagging or robust bagging) is composed by three steps: (1) training individual classifiers based on bootstrap sampling; (2) selecting many classifiers instead of all by some optimization algorithm, and (3) combining these classifiers by majority voting. However, selecting these predictors and excluding those 'useless' ones from the ensemble is not an easy task. It is considered as an optimization task. If traditional techniques are used, the computational cost is too extensive to be useful in real world applications. In Banfield *et al.* (2005) the original complete ensemble is pruned (or thinned, using the term proposed by the authors) by

https://doi.org/10.1017/S0269888913000313 Published online by Cambridge University Press

sequential backward selection: classifiers that do not improve the classification performance are progressively removed from the ensemble. Several researchers also proposed methods to select classifiers generated by bagging, with the aim of improving the ensemble accuracy and reducing its size such as Banfield *et al.* (2007) and Martinez-Munoz and Suarez (2007). In Zhou *et al.* (2002), Yi *et al.* (2004) the problem of finding a globally optimal subset of classifiers is solved approximately by means of a genetic algorithm (GA).

Zaman and Hirose (2008) tried to explore the advantages of robust bagging. They carried out experiments on benchmark datasets and recommend from the results that robust bagging performs quite similar compare with the standard bagging when applied to unstable base classifiers such as decision trees, but performs better when applied to more stable base classifiers as Fisher Linear Discriminant Analysis and Nearest Mean Classifier. Similarly, in Croux *et al.* (2007) experiments, trimmed bagging performs comparably to standard bagging when applied to decision trees, but yields better results when applied to more stable base classifiers, like support vector machines.

Martinez-Munoz *et al.* (2007) presented a study of different selective bagging techniques using decision stump as base classifier. A number of different selective ensemble methods were tested. Four of these were greedy strategies based on first reordering the elements of the ensemble in accordance with some rule that takes into account the complementarity of the predictors. Complementarity means that the errors of the different hypothesis are not correlated. Sub-ensembles of increasing size were then built by incorporating the ordered classifiers one by one. A halting criterion stops the aggregation process. The other two approaches are selection techniques that attempt to find optimal sub-ensembles using either GAs or semi-definite programming. Experiments performed on 24 benchmark classification tasks show that the selection of a small subset (about 10–15%) of the original pool of classifiers generated with bagging can significantly increase the accuracy and decrease the complexity of the ensemble.

Recently, Xu *et al.* (2010) proposed a constraint bagging forecasting method for stock price prediction. In this approach, each of predictors is first constructed by bootstrapping using neural networks. Second, constraint conditions are set to select competitive predictors. Finally, with the appropriate predictors, ensemble is applied to forecast stock prices.

There are also clustering-based pruning techniques. These techniques first employ a clustering algorithm so as to discover groups of models that make similar predictions. Afterwards, each cluster is separately pruned in order to increase the overall diversity of the ensemble (Bakker & Heskes, 2003; Fu *et al.*, 2005).

2.2 Alternative sampling bagging variants

Bootstrap samples are random samples of size n selected with replacement from the original n-sized sample, so not all bootstrap samples are uniformly informative, due to the randomness associated to the number of distinct observations in the bootstrap sample. The variability of this number is neither necessary nor desirable, having negative effects on the performance of the bootstrap technique in some applications.

One of the first variants of bagging that uses a sampling ratio different from the standard value is Rvotes (Breiman, 1999). This algorithm is designed to increase the training speed when very large databases are available for learning. In Rvotes the individual ensemble classifiers are built using small bootstrap samples, as small as 0.5% of the original training data. As a consequence, the generalization performance of the proposed algorithm is rather poor in problems where the training data is not rich and redundant in some way. There are other studies where improvements in the generalization performance of bagging are obtained by using sample sizes different from the original training data such as Terabe *et al.* (2001) and Hall and Samworth (2005). Alternative bootstrap resampling scheme is also presented in Jimnez-Gamero *et al.* (2004), which only uses a portion of the set of all possible bootstrap samples. Rafael Pino-Mejias *et al.* (2004) proposed another alternative bootstrap procedure, namely the reduced bootstrap, which only uses a portion of the set of all possible bootstrap samples, as the sampling algorithm for bagging.

https://doi.org/10.1017/S0269888913000313 Published online by Cambridge University Press

A comparable idea is about building an ensemble of classifiers each of which is trained based on a particular weighting over the training examples (a weighting is a set of weights associated with the cases). The task concerns search in a big weighting space. In this view, Wang and Wang (2006) proposed to incorporate a GA. It performs a wide yet efficient search for appropriate weightings (chromosomes).

Pino-Mejias *et al.* (2008) have considered a modified procedure, the reduced bootstrap, where the bootstrap samples must have a certain minimum number of distinct observations. The performance of this modified bagging has been empirically studied over ten datasets for two learning algorithms, decision trees and the multilayer perceptron with good results.

Standard bagging uses resampling with replacement to produce bootstrap samples of equal size as the original set. Without-replacement methods typically use half samples. These choices of sampling sizes are arbitrary and need not be optimal in terms of the accuracy of the ensemble. Martinez-Munoz and Suarez (2010) use the out-of-bag estimates of the accuracy to select a near-optimal value for the sampling ratio. Ensembles of classifiers trained on independent samples whose size is such that the out-of-bag error of the ensemble is as low as possible generally improve the accuracy of standard bagging and can be efficiently built.

Baszczyski *et al.* (2010) argue that the classification accuracy of bagging classifier can be improved by drawing to bootstrap samples instances being more consistent with their assignment to decision classes. Giving more chance to select consistent instances (i.e., instances which certainly belong to a class) and decreasing a chance for selecting border or noisy ones may lead to creating more accurate and diversified base classifiers in the bagging scheme. Baszczyski *et al.* (2010) proposed a variable consistency generalization of the bagging scheme where such sampling is controlled by two types of measures of consistency: rough membership and monotonic measure. The results of experiments show that variable consistency bagging improves accuracy on inconsistent data.

Dagging (Ting & Witten, 1997) is an alternative to bagging that combines classifiers induced on disjoint subsets of the data. It is appropriate when either the data originally comes from disjoint sources, or when data is plentiful, that is when the learning algorithm has reached the plateau on the learning curve.

In the paper of Fushiki (2010), bootstrap methodology is adapted to resolve some problems in small datasets. The bootstrap predictive distribution is obtained by applying the bagging algorithm to the plug-in distribution with the maximum likelihood estimator.

2.3 Bagging variants that use different subsets of features

Well known is random subspace method introduced by Ho (1998), which consists of training several classifiers from datasets constructed with a given proportion u of features picked randomly from the original set of features. There are also other 'non-random' methods, where the correlation between each feature and the class is computed and the base classifier is trained only on the most correlated subset of features. Other favorite class feature selection iterative methods have been considered by Puuronen *et al.* (2001) using contextual merit measures instead of the simple correlation.

Bryll *et al.* (2003) presented attribute bagging (AB), a similar technique for improving the accuracy and stability of classifier ensembles induced using random subsets of features. AB establishes an appropriate attribute subset size and then randomly selects subsets of features, creating projections of the training set on which the ensemble classifiers are built. Bryll *et al.* (2003) also demonstrated that ranking the attributes by their classification accuracy and voting using only the best subsets improves the performance of the ensemble.

Stefanowski (2007) also studied to integrate bootstrap sampling with advanced methods of feature selection, which are based on evaluation of the relationship between single feature, or feature subsets, and the target class. Shirai *et al.* (2008) introduced a method for selecting both samples and features at the same time.

Panov and Dzeroski (2007) proposed to use a combination of concepts used in bagging and random subspaces. Their algorithm randomly selects a subset of the features at the start and use a

https://doi.org/10.1017/S0269888913000313 Published online by Cambridge University Press

deterministic version of the base algorithm. The results of the authors experiments show that this approach has a comparable performance to that of random forests (Breiman, 2001), with the added advantage of being applicable to any base-level algorithm without the need to randomize the base learner. Latinne *et al*. (2000) proposed a similar technique to combine the standard bagging with random selection of feature subsets over each bootstrap sample.

Cai *et al*. (2008a) took into account the diversity of classification margins in feature subspaces for improving the performance of bagging. Cai *et al*. (2008a) first studied the average error rate of bagging, convert the task into an optimization problem for determining the weights for feature subspaces, and then assigned the weights to the subspaces via a randomized technique in classifier construction.

Another model, called vertical bagging decision trees model (VBDTM), is proposed by Zhang *et al*. (2010). The VBDTM model gets an aggregation of learners by means of the combination of predictive attributes. In the VBDTM model, all train instances and just parts of features based on rough sets, take part in learning of every classifier.

2.4 Bagging variants that use different voting rule

According to Skurichina and Duin (2000) experiments, simple majority vote is a not good choice of the combining rule for bagging. The weighted majority vote rule is often better choice.

Derbeko *et al*. (2002) provide facts that a variance optimized bagging consistently improves on bagging. In this method, the authors search for a linear combination of the base learners such that the weights are optimized to decrease variance. Minimum variance combinations are computed using quadratic programming. This optimization technique is borrowed from mathematical finance where it is called Markowitz Mean-Variance Portfolio Optimization (Markowitz, 1959).

Nanculef *et al*. (2007) proposed a modification of bagging to explicitly trade off diversity and individual accuracy. The process separates the bootstrap replicates produced of the algorithm in two subsets: one consisting of the instances misclassified by the ensemble at the previous iteration and the other consisting of the instances correctly identified. A high individual accuracy of a new classifier on the first subset increases diversity. On the other hand, a high accuracy on the second subset decreases diversity. Nanculef *et al*. (2007) trade off between both components using a parameter $\lambda \in [0, 1]$ that changes the cost of a misclassification on the second subset.

2.5 Bagging by smearing

Regularly the attribute values of the examples are not modified in the ensemble generation process. One exception is called 'output smearing' (Breiman, 2000) which varies the class labels by adding a controlled quantity of noise. This lead Frank and Pfahringer (2006) to the idea of combining smearing with bagging, by smearing the subsamples involved in the bagging process.

Su *et al*. (2008) proposed a random missing value corruption method that first generate multiple incomplete datasets from the initial dataset by randomly injecting missing values with a small missing ratio, and then apply a bagging predictor trained on each of the incomplete dataset to give classifications. The final decision of the class is the result of voting of the classifications.

2.6 Bagging variants for handling imbalanced datasets

The imbalanced class distribution is described as having many more instances of some class than others. Classification of data with imbalanced class distribution has posed a significant drawback of the performance achievable by most classification algorithms, which assume a relatively balanced class distribution and equal misclassification costs.

It must be mentioned that weak classifiers can create problems in bagging, especially when the distribution of the class is very unbalanced. Suppose the marginal distribution of the response is

https://doi.org/10.1017/S0269888913000313 Published online by Cambridge University Press

unbalanced so that it is very difficult for a model using the predictors to perform better than the marginal distribution. Under those circumstances the rare class will likely be misclassified most of the time because votes will be typically being won by the class that is far more common.

Many approaches have been developed using bagging ensembles to deal with class imbalance problems (Li, 2007; Galar *et al*., 2012).

A way to overcome the class imbalance problem in each bag is to take into account the classes when instances are randomly drawn from the original dataset. Instead of performing a random sampling of the dataset, an oversampling process of the minority class (OverBagging) can be carried out before training each learner (Wang & Yao, 2009).

On the contrary to OverBagging, UnderBagging procedure uses undersampling instead of oversampling. The undersampling process is usually only applied to the majority class; however, a resampling with replacement of the minority class can also be applied so as to produce *a priori* more diverse ensembles (Tao *et al*., 2006). Roughly balanced bagging (Hido *et al*., 2009) is rather similar to UnderBagging, but it does not bootstrap a totally balanced bag.

2.7 Bagging with heterogeneous components

Hothorn and Lausen (2005) proposed to add the outcome of classifiers (linear discriminant variables, predicted conditional class probabilities or predicted classes) to the set of original features for bagging classification trees. The trees implicitly select the most informative predictors, and, in a sense, bundle the additional classifiers.

Double bagging is a parallel ensemble method, where an additional classifier model is trained on the out-of-bag samples and then the posteriori class probabilities of this additional classifier are added with the inbag samples to train another classifier (Hothorn & Lausen, 2003). So the performance of the double bagging depends on two factors: (1) the classes of the dataset are linearly separable so that the additional predictors are informative (the additional classifier is able to discriminate the classes), (2) the size of the out-of-bag samples as to construct an additional classifier model. The sub-sampled version of double bagging depends on two hyper parameters, subsample ratio and an additional classifier. Faisal *et al*. (2010) proposed an embedded cross-validation-based selection technique to select these parameters automatically.

Coelho and Nascimento (2010) presented an evolutionary approach for optimally designing Bagging models composed of heterogeneous components. Different learning algorithms are usually associated with different search/representation biases, thereby increase ensemble diversity. Since the automatic configuration of the best heterogeneous bagging model for a given classification problem turns is a combinatorial optimization problem, a customized GA has been adopted for this purpose (Coelho & Nascimento, 2010).

Kotsiantis and Kanellopoulos (2010) proposed a technique that uses different subsets of the same training dataset with the concurrent use of a voting scheme for combining different learners instead of similar learners. The authors performed a comparison of their method with other well known ensembles using the same base learners and stated that their method had better accuracy in most difficult cases.

2.8 Bagging variants that use stable learners

The main condition for which bagging would improve the performance are that the base learning algorithm should be good on average but unstable with respect to small changes in the training set. A learning method is termed 'unstable' if small alterations in the training-test set split can result in large changes in the resulting classifier. For this reason, some authors implement special bagging variants that can use stable learners.

Valentini and Dietterich (2003), introduced Lobag (low bias bagging), for aggregating support vector machines. This approach combines the low-bias properties of SVMs with the low-variance properties of bagging. Li and Yang (2008) obtained top k features according to their mutual

https://doi.org/10.1017/S0269888913000313 Published online by Cambridge University Press

information estimated on the bootstrap samples, and then built SVM classifiers based on these informative features.

Zhou and Yu (2005a) developed an approach to apply bagging to nearest neighbor classifiers. In order to add instability to the learning process to favor diversity of classifier, they used Minkowsky distance with a different randomly selected value of power for each classifier. In a subsequent work (Zhou & Yu, 2005b) this method was coupled with boostrap sampling and attribute filtering.

Jiang *et al*. (2005) proposed a new variant of bagging named DepenBag. This algorithm obtains bootstrap samples at first. Then, it employs a causal discoverer to induce from each set a dependency model expressed as a Directed Acyclic Graph (DAG). The features without connections to the class in all the DAGs are then removed. Finally, a classifier is trained from each of the resulted samples to constitute the ensemble. The experiments of Jiang *et al*. (2005) show that DepenBag is time consuming but effective for building ensembles of nearest neighbor classifiers.

Sohn and Shin (2007) compared the performances of classifier combination methods (such as bagging and modified random subspace method) to logistic regression in relation to various characteristics of initial dataset. The authors experimental study results indicate the following: when training set is large, performances of logistic regression and bagging are not significantly different. However, when training set size is small, the performance of logistic regression is worse than bagging. When training dataset size is small and correlation is strong, both modified random subspace method and bagging perform better than the other tested methods.

Yang *et al*. (2009) proposed a novel bagging null space locality preserving discriminant analysis (bagNLPDA) method. The bagNLPDA method first projects all the training instances into the range space of a so-called locality preserving total scatter matrix without losing any discriminative information. The projected instances are then randomly sampled using bagging to generate a set of bootstrap samples. Null space discriminant analysis is performed in each sample and the results of them are combined using majority voting.

3 Boosting

According to the resampling mechanism of bagging each weak classifier is independent of another and thus the weak classifiers can be built in parallel. On the contrary, in boosting relies the probability of a specific instance being selected to be part of a particular classifiers training set depends on the performance of the previously built classifiers on that instance. In other words, the probability distribution is updated so that the instances being incorrectly predicted by previous classifiers can be selected more frequently than those being properly predicted. Thus, boosting tries to generate new classifiers that are able to better classify the 'hard' instances for the previous ensemble members.

There are several boosting variants; AdaBoost (Freund & Schapire, 1996a, 1997) is the most well-known. The algorithm is presented briefly in Figure 2. Originally boosting scheme was developed for binary classification problems. Freund and Schapire (1997) extended AdaBoost to a multi-class case, versions of which they called AdaBoost.M1 and AdaBoost.M2. Freund and Schapire (1997) extended AdaBoost.M2 to boosting regression problems and called it AdaBoost.R. It solves regression problems by reducing them to classification ones.

Breiman's (1999b) arc-gv algorithm, was used as empirical tool to study AdaBoost's convergence properties; AdaBoost is difficult to analyze theoretically so such empirical tools are very useful.

In boosting-by-resampling, training sample size is fixed and training instances resampled according to a probability distribution. In boosting-by-reweighting, weights assigned to each instance and used in each iterations to train the base classifier. However, this technique is only useful when the weak learning algorithm can handle weighted instances.

Kuncheva *et al*. (2002) carried out experiments on seven datasets for different sample sizes, different number of sub-classifiers in the ensembles, and two linear base learning algorithms. The results confirmed in a quantitative way the explanation behind the success of Boosting for linear classifiers for increasing training sizes, and the poor performance of bagging in this case.

https://doi.org/10.1017/S0269888913000313 Published online by Cambridge University Press

Given: $(x_1, y_1), \ldots, (x_m, y_m)$; $x_i \in \mathcal{X}$, $y_i \in \{-1, +1\}$
Initialise weights $D_1(i) = 1/m$
for t = 1 to T do
Find $h_t = \arg\min_{h_j} \epsilon_j = \sum_{i=1}^{m} D_t(i)(y_i \neq h_j(x_i))$
if $\epsilon_t \geq 1/2$ then stop
Set $\alpha_t = \frac{1}{2}\log(\frac{1-\epsilon_t}{\epsilon_t})$
Update
$D_{t+1}(i) = \frac{D_t(i)exp(-\alpha_t y_i h_t(x_i))}{Z_t}$, so that: $exp(-\alpha_t y_i h_t(x_i)) \begin{cases} < 1, & y_i = h_t(x_i) \\ > 1, & y_i \neq h_t(x_i) \end{cases}$
Output the final classifier:

$$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$

Figure 2 The AdaBoost algorithm.

Diversity measures point out that boosting succeeds in inducing diversity even for stable classifiers whereas bagging does not.

The best explanation of boosting generalization capabilities so far is margin theory (Schapire *et al.*, 1997). Domingos (2000) showed the relation between margin theory explanation of boosting and bias-variance explanation. By now, statistical views of boosting have existed for a number of years. One such view is due to Friedman *et al.* (2000) who propose that boosting is stagewise additive model fitting.

Gao and Zhou (2010) introduce the notion of approximation stability, and prove that the approximation stability is sufficient for generalization and necessary for learnability of asymptotical empirical risk minimization (AERM). Then, Gao and Zhou (2010) prove that AdaBoost has approximation stability and thus good generalization. Moreover, an exponential bound for AdaBoost is provided. All bounds obtained in that paper do not rely on any space complexity measure, but rather on the way the algorithm searches the space, and therefore can be used still when the VC dimension is infinite.

Schapire and Singer (1999) identified two scenarios where AdaBoost is likely to fail: (i) when there is insufficient training data relative to the 'power' of the base classifiers, and (ii) when the training errors of the base classifiers become outsized quickly. Schapire and Singer (1999) proposed Real AdaBoost, a generalized version of AdaBoost, in which weak classifiers are piece-wise functions whose output is a real value representing the confidence-rated prediction. To construct such weak classifiers, one splits the input space X into non-overlapping blocks $X_1, X_2, \ldots, X_N$ so that the predictions of the weak classifier are the same for all instances falling into the same block. In the case of decision stump classifier, this is equivalent to dividing the real line into intervals. Meanwhile, determining the appropriate number of blocks for weak classifiers learned by Real AdaBoost is a demanding task because small ones may not accurately approximate the real distribution while large ones may cause over-fitting. Le and Satoh (2007) have developed Ent-Boost, a novel boosting scheme for efficiently learning weak classifiers using entropy measures. Class entropy information is used to automatically estimate the optimal number of bins. A scheme, combining error size and proximity to the classification border of the training samples, has been proposed by Gomez-Verdejo *et al.* (2006) for Real AdaBoost ensemble designs.

From the perspectives of additive regression model and exponential loss function, Friedman *et al.* (2000) proposed some new boosting algorithms including 'GentleBoost' and 'LogitBoost'. Following the results of stochastic approximation theory a stochastic approximation boosting algorithm, SABoost, is proposed by Tsaoa and Chang (2007).

Most of the boosting algorithms in practice usually optimize a convex loss function and do not make use of the margin distribution. Shen and Li (2010) designed a new boosting algorithm, Margin-Distribution boosting (MDBoost), which directly maximizes the average margin and minimizes the margin variance at the same time.

https://doi.org/10.1017/S0269888913000313 Published online by Cambridge University Press

In our point of view boosting ensemble variants can be categorized in: (i) methods that try to handle imbalance datasets, (ii) methods that use alternative bootstrap sampling schemes, (iii) methods that use different subsets of features, (iv) methods that use different voting rule, (v) methods that try to handle noisy datasets, (vi) methods that are implemented in order to be used to stable learners, (vii) methods that apply boosting based on the characteristics of test instances, (viii) methods that change labels of examples and (ix) methods that apply boosting in a online environment.

3.1 Boosting variants for handling imbalanced datasets

Since AdaBoost is an accuracy-oriented algorithm, its learning strategy is bias toward the majority class as it contributes more to the overall classification accuracy. Consequently, the identification performance on the small class is not always satisfactory by applying AdaBoost.

AdaCost (Fan *et al.*, 1999) provided a cost-sensitive alternative to AdaBoost. The key difference between AdaCost and AdaBoost is the formula for updating example weights. While AdaBoost treats instances of each class similarly, AdaCost differentiates between examples of the minority and majority classes. Further, it uses a user-specified cost ratio into its 'misclassification cost adjustment function'. In doing so, AdaCost can put emphasis on minority class examples based on input from the user.

SMOTEBoost (Chawla *et al.*, 2003) combines data sampling with boosting. The data sampling aspect of SMOTEBoost uses the synthetic minority oversampling technique (SMOTE) (Chawla *et al.*, 2002). SMOTE is an oversampling technique that creates new minority class examples by finding the k nearest neighbors of each minority example and extrapolating between these examples. SMOTEBoost combines SMOTE with AdaBoost by performing SMOTE before building the weak hypothesis.

In the paper of Sun *et al.* (2007), the AdaBoost algorithm is adapted for advancing the classification of imbalanced data. Three cost-sensitive boosting algorithms were developed by introducing cost items into the learning framework of AdaBoost. For each proposed boosting algorithm, its weight update parameter is deduced taking the cost items into consideration.

RareBoost (Joshi *et al.*, 2001) was also designed specifically to address the problem of class imbalance. Unlike cost-sensitive boosting techniques, RareBoost does not take cost into account when reweighting instances. The weights are updated according to how well an iteration generated hypothesis distinguishes between false positives and true positives as well as false negatives and true negatives.

Seiffert *et al.* (2008) empirically evaluated the differences between boosting implementations using imbalanced training data. Using 10 boosting algorithms, four learners and 15 datasets, Seiffert *et al.* (2008) found that boosting by resampling performs as well as, or significantly better than, boosting by reweighting.

Most cost sensitive boosting (CSB) methods directly modify the original AdaBoost procedure. Unfortunately, the effectiveness of most cost sensitive boosting methods is checked only by experiments. Cai *et al.* (2008b) showed that several typical CSB methods can also be view as gradient descent for minimizing a unified objective function.

3.2 Boosting variants for handling noisy data

Yin and Dong (2011) identified noisy data as having at least mislabeled instances. Noise is likely to occur in many problems. There is increasing evidence to show that boosting is quite susceptible to noise. One of the most convincing experimental studies that find this phenomenon has been reported in (Dietterich, 2000). In his experiments, Dietterich compares the performance of AdaBoost and bagging on some standard learning datasets and investigates the accuracy of these methods in relation to the addition of classification noise to the training data. As expected, the prediction accuracy of both AdaBoost and bagging deteriorates as the noisy level increases. However, the increase in the error is much more significant in AdaBoost.

https://doi.org/10.1017/S0269888913000313 Published online by Cambridge University Press

Some researchers Servedio (2003), Vezhnevets and Barinova (2007) reported that boosting concentrates on few abnormal instance, which leads to overfitting, thus modifications on the reweighting scheme or methods of imposing limit on skewness of the weights distribution were proposed. Kalai and Servedio (2005) also studied boosting in the presence of random classification noise. The authors showed that a modified version of a boosting algorithm can achieve accuracy arbitrarily close to the noise rate.

Recently, Gao *et al.* (2010) proposed an improved boosting algorithm with adaptive filtration. A filtering algorithm is designed first based on Hoeffding inequality to identify mislabeled or atypical instances. By introducing the filtering algorithm, the authors modify the loss function such that influences of mislabeled or atypical instances are penalized. Experiments performed on eight different UC Irvine Machine Learning Repository (UCI) datasets show that this boosting algorithm almost always obtains better classification accuracy than simple AdaBoost.

3.3 Boosting variants that use alternative sampling techniques

Conservative boosting (Kuncheva & Whitaker, 2002) uses a softer equation to update the sampling distribution in which the value of the sampling distribution is only updated for the misclassified patterns. In addition, this method allows the reinitialization of the sampling distribution. Torres-Sospedra *et al.* (2007b) proposed the weighted conservative boosting a new relaxed version of AdaBoost. In this case the sampling distribution is updated with a weighted equation that depends on the dataset and on the number sub-classifiers of the ensemble. This method can be seen as a softer version of AdaBoost that avoid training exclusively on difficult instances.

The work of Li *et al.* (2005) give some insights on what the sample size should be when one knows the true distribution or when a plugin estimate for the global distribution can be approximated. Boosting learning procedure can be divided into two sequential stages which are named reducing fitting error stage and reducing variance stage. Traditional sampling methods such as roulette wheel selection is suitable for the learning procedure of reducing fitting error stage, and based on the characteristics of reducing variance stage, a new sampling method is proposed by Wang *et al.* (2006) named SS method. Based on continuous sampling plan and SS sampling methods, a new two stages based adaptive sampling boosting method named ASSBoosting is proposed by Wang *et al.* (2006).

Hall *et al.* (2007) examined a variant of boosting in large datasets for speeding up support vector machine learning. The idea is to build the classifiers in the ensemble with small subsets of the available training data. Obviously, building each sub-classifier with a small subset of the data will be much faster than using all the data. The authors found that boosted support vector machines can learn accurate models from smaller subsamples.

In Zhang *et al.* (2008), each sub-classifier is also obtained by applying a given weak learner to a subsample which is drawn from the original training set according to the probability distribution maintained over the training set. A parameter is introduced into the reweighted scheme proposed in AdaBoost to update the probabilities assigned to training.

FilterBoost (Bradley & Schapire, 2008) uses non-monotonic adaptive sampling together with a filter to sequentially estimate the edge of each weak hypothesis. When the edge is approximated with high probability the algorithm updates its classifier and continues to select and train the next weak hypothesis.

3.4 Boosting variants that use different subsets of features

Yin *et al.* (2005) introduced a strategy of boosting-based feature combination. Different from the general boosting, at each round of this variant boosting, some weak classifiers are built on different feature sets. And then these classifiers are combined by weighted voting. Redpath and Lebart (2005) identified feature subset by the regularized version of Boosting, that is, $\text{AdaBoost}_{\text{Reg}}$. Their search strategy is the floating feature search. In Xu and Zhang (2006), a

https://doi.org/10.1017/S0269888913000313 Published online by Cambridge University Press

bootstrap sample is generated randomly and then the feature with the highest score on this sample is obtained at each iteration. After that, the weight of instance recognized correctly by this feature is reduced by a factor.

Based on principal component analysis (PCA), Rodriguez *et al.* (2006) proposed a new ensemble classifier generation technique rotation forest. Its main idea is to simultaneously encourage diversity and individual accuracy. Specifically, diversity is promoted by using PCA to do feature extraction for each base classifier and accuracy is sought by keeping all principal components and also using the whole dataset to train each base classifier. Zhang and Zhang (2008b) presented an ensemble classifier generation technique RotBoost, which is constructed by combining rotation forest and AdaBoost. The experiments conducted with 36 real-world datasets available from the UCI repository demonstrate that RotBoost can generate ensemble classifiers with lower prediction error than either Rotation Forest or AdaBoost more often than the reverse.

Garcia-Pedrajas and Ortiz-Boyer (2008) proposed a boosting approach to random subspace method (RSM) to achieve an improved performance and avoid some of the major drawbacks of RSM. The random selection of inputs in RSM, its source of success, can also be a key problem. For several problems some of the selected subspaces may lack the discriminant ability to separate the different classes. These subspaces produce poor classifiers that harm the accuracy of the ensemble. Garcia-Pedrajas and Ortiz-Boyer (2008) search subspaces that optimize the weighted classification accuracy given by the boosting algorithm, and then the new classifier added to the ensemble.

Rodriguez and Maudes (2008b) proposes another variant AdaBoost. It is based on considering, as the base classifiers for boosting, not only the last weak classifier, but a classifier formed by the last r selected weak classifiers (r is a parameter of the method). If the weak classifiers are decision stumps, the combination of r weak classifiers is a decision tree.

Garcia-Pedrajas (2009) presented an approach that is based on using the distribution given by the weighting scheme of boosting to construct a nonlinear supervised projection of the original features, instead of using the weights of the instances to train the next classifier.

In multi-view learning, a co-training procedure for classification problems was developed (Wang & Zhou, 2010). The idea is that better classifiers can be built at the individual view level, rather than constructed on all the available views. Peng *et al.* (2011) proposed a boosting algorithm that builds base classifiers independently from each data type (view) that provides a partial view about an object of interest. Dissimilar from AdaBoost, where each view has its own re-sampling weight, this algorithm uses a single re-sampling distribution for all views at each boosting round. This distribution is resolved by the view whose training error is minimal. This shared sampling mechanism can restrict noise to individual views, in that way reducing sensitivity to noise.

Mumbo is another multiview boosting algorithm: each example of the learning sample is represented by several independent sets of features (Koco & Capponi, 2011). Each one of these representations is called a view. Even though these models are learned on different representations of the same examples, they are by no means equal accuracy. Sub-classifiers learned on some views perform better than those learned on other views, due to the noise in the data and/or views, or the lack of information, etc.

3.5 Boosting variants that use different voting rule

According to Skurichina and Duin (2000), simple majority vote not a first-class choice for the combining rule of boosting. The weighted majority vote rule is frequently a good choice. The average, mean, and product combining rules may also perform well and sometimes better than the weighted majority vote combining rule.

Oza (2003) proposed an averaged version of AdaBoost called averaged boosting (AveBoost). This version applied an averaged equation that depends on the equation used in AdaBoost, on the

https://doi.org/10.1017/S0269888913000313 Published online by Cambridge University Press

distribution values of the previous learner and on the number of learners that have been trained. Torres-Sospedra *et al.* (2007a) proposed the mixed method averaged conservative boosting. In this method the authors apply the conservative equation used in Conserboost along with the averaged process used in AveBoost.

Torres-Sospedra *et al.* (2008) studied the performance of several different combination methods for ensembles previously trained with AdaBoost in order to see which combiner fits better on these ensembles. The results show that the accuracy of the ensembles can be improved by using a stacked combiner (Wolpert, 1994). The training in stacked combiner is divided into two steps. In the first one, the experts are trained. In the second one, the learner is trained with the outputs provided by the experts to produce the final combination rule.

Gambina *et al.* (2009) analyzed boosting process from a game theoretic perspective. The close connections between game theory and boosting were first studied in Freund and Schapire (1996b), where boosting was implemented using the Littlestone and Warmuth's (1994) 'weighted majority' algorithm for playing repeated games. The iterative structure of repeated games suggests backwards-looking adaptation.

3.6 Boosting by smearing

Melville and Mooney (2005) presented a meta-learner diverse ensemble creation by oppositional relabeling of artificial training examples, that uses an existing 'strong' learner (one that provides high accuracy on the training data) to build an effective diverse committee. This is accomplished by adding different randomly constructed examples to the training set when building new committee members. These artificially constructed examples are given category labels that disagree with the current decision of the committee, thereby easily and directly increasing diversity when a new classifier is trained on the augmented data and added to the committee.

Phama and Smeulders (2008) proposed a similar boosting algorithm. The method consists of calling the input learner after randomizing the labels of the dataset and subsequently calling it with a systematic update of the labels. This method with the help of AdaBoost makes intensive use the given base learner by both reweighting and relabeling the original training set. The algorithm requires more time than the AdaBoost algorithm in training, but not in classification.

Leskes and Torenvliet (2008) presented a boosting algorithm where unlabeled examples are used to enforce agreement between several different learning algorithms. Not only do the learning algorithms learn from the given training set but they are supposed to do so while agreeing on the unlabeled examples. In the set of experiments which Leskes and Torenvliet (2008) performed, a reduction of up to 40% was observed in the number of labeled examples necessary in order to achieve desired classification accuracy.

3.7 Boosting of stable learners

Although boosting is a very successful method for improving the performance of any classification algorithm, its application to k-NN methods is problematic. The reweighting approach is not useful, as weighting an instance affects the classification of its neighbors, but not the classification of the instance itself. If one uses resampling, one can sample the instances using the distribution given by the boosting method. However, this approach is not likely to produce an improvement in the testing error.

O'Sullivan *et al.* (2000) applied a feature boosting method called Featureboost to a k-NN classifier. The reported results showed improved performance over AdaBoost on three used datasets. Featureboost is aimed at applying boosting principles to sample features instead of instances.

Instead of applying AdaBoost to a typical classification problem, Amores *et al.* (2006) use it for learning a distance function and the resulting distance is used into k-NN. The proposed method (boosted distance with nearest neighbor) outperforms the k-NN classifier used with several different distances.

https://doi.org/10.1017/S0269888913000313 Published online by Cambridge University Press

Lu *et al*. (2007) incorporated feature re-weighting into boosting and proposed a new feature re-weighting approach for adaptive discriminant analysis. The presented algorithm i.Boosting not only weights the samples, but also weights the feature elements iteratively. Besides, in i.Boosting, relevance user feedback provides boosting with more misclassification information.

Garcia-Pedrajas and Ortiz-Boyer (2009) developed two methods for boosting k-NN classifiers. Both methods are based on changing the view that each classifier of the ensemble has of the instances, modifying the input space in a manner that supports the accurate classification of difficult instances: (a) selecting a subset of the inputs; and (b) transforming the inputs by means of a nonlinear projection.

4 Locally application of bagging and boosting

Both bagging and boosting uses a voting technique which is unable to take into account the heterogeneity of the instance space. When majority of the sub-classifiers give a wrong prediction for a new instance then the vote scheme will give wrong prediction. The problem may consist in small weights of base classifiers that are highly accurate in a restricted region of the instance space because this accuracy is swamped by their inaccuracy outside the restricted area. It may also consist in the usage of sub-classifiers that are accurate in most of the space but still confuse the whole classification committee in some restricted areas of the space. To overcome this problem Tsymbal and Puuronen (2000) suggested a dynamic integration approach that estimates the local accuracy of the base classifiers by analyzing the accuracy in near-by instances. The goal is to use each sub-classifier in that subarea for which it is the most reliable one.

Local decision bagging (Alaiz-Rodriguez, 2008b) is a dynamically weighted ensemble where its members are generated as in the standard bagging approach but lately, dynamically selected and combined in a way determined by the test sample. It differs from the local learning approach in Kotsiantis *et al*. (2005) where decision stumps in the ensemble are induced, for every single test example.

Zhu and Yang (2008) also proposed lazy bagging (LB), which builds bootstrap replicate bags based on the characteristics of test instances. Upon receiving a test instance, LB trims bootstrap bags by taking into consideration nearest neighbors in the training data. The authors hypothesis is that an unlabeled instance's nearest neighbors provide valuable information to generate a classifier with refined decision boundaries. Kotsiantis *et al*. (2006) proposed a technique of boosting localized weak learners; rather than having constant weights attached to each learner (as in standard boosting approaches), the authors allow weights to be functions over the input domain. In order to find out these functions, the authors recognize local regions having similar characteristics and then build local experts on each of these regions describing the association between the data characteristics and the target value.

The main idea of the local boosting algorithm (Zhang & Zhang, 2008a) can be summarized as follows: in the process of generating each base classifier, a local error is calculated for each training instance and a function of this local error is utilized to update the probability that the instance is selected to be part of next classifier's training set. When classifying a new instance, the similarity information between it and each training instance is taken into account.

5 Bagging and boosting variants for data streams

Batch-based algorithms require an amount of memory which is linearly proportional to the size of the dataset. When trained on large datasets and possibly infinite data streams (Gama, 2010), it is needed to have an online algorithm that would converge to the batch algorithm with memory constraints that are independent of the training set size.

Data streams pose several challenges on data mining algorithm design (Kuncheva, 2004b). First, algorithms must use limited resources (time and memory). Second, they must deal with data whose nature or distribution changes over time.

https://doi.org/10.1017/S0269888913000313 Published online by Cambridge University Press

The following constraints apply in a data stream model:

1. Data arrives as a potentially infinite sequence. As a result, it is impossible to store it all; only a small summary can be computed and stored.
2. The speed of arrival of data is fast, so each particular element has to be processed in real time.
3. The distribution generating the items may change over time. The concept drift means that the statistical properties of the class, which the model is trying to predict, change over time in unpredicted ways. This causes problems because the predictions become less accurate as time passes. Consequently, data from the past may turn into irrelevant (or even harmful) for the current prediction.

Lee and Clyde (2004) developed a lossless online bagging algorithm that draws training example weights for each base model according to a Gamma distribution. However, their algorithm is lossless relative to a batch bagging algorithm.

Oza (2005) presented online versions of bagging and boosting that require only one pass through the training data. Online learning algorithms process each training example once 'on arrival' without the need for storage, and maintain a current model that reflects all the training examples seen up to this point. Oza (2005) has also proven that the classifiers returned by bagging and online bagging converge to the same classifier given the same training set as the number of models and training examples tends to infinity under two conditions. The first is that the base learning algorithm return classifiers that converge toward the same classifier as the number of training instance grows. The second is that, given a fixed training set *Tr*, the online and batch base model learning algorithms return the same classifier for any number of copies of *Tr*. Oza (2005) has also proven that for naive bayes base models, the online and batch boosting algorithms converge to the same classifier as the number of models and training examples tend to infinity.

Liu and Yu (2007) integrated the gradient-based feature selection with an online boosting framework. This online boosting algorithm provides an efficient way of updating the discriminative feature set, as well as presents a unified objective for both feature selection and weak classifier updating. Grabner and Bischof (2006) also proposed an on-line version for feature selection.

Raphael Pelossof *et al.* (2009) presented an online boosting algorithm for updating the weights of a boosted classifier, which yields a close approximation to the edges found by AdaBoost algorithm. Like online boosting proposed by Oza (2005), their algorithm sequentially updates the weights of the weak hypotheses. However, unlike Online Boosting that needs the weak hypotheses to be naive bayes classifiers which can change in an online manner, their algorithm only deals with adapting the weights of a set of weak hypotheses.

Recently, two new variants of bagging were proposed: ADWIN bagging and adaptive-size hoeffding tree (ASHT) bagging. ASHT bagging uses trees of different sizes, and ADWIN bagging uses ADWIN as a change detector to decide when to discard underperforming ensemble members (Bifet *et al.*, 2009a). Bifet *et al.* (2009b) improved ADWIN bagging using Hoeffding adaptive trees, trees that can adaptively learn from data streams that change over time. Bifet *et al.* (2010a) propose leveraging bagging. This method combines the simplicity of bagging with adding more randomization to the input and output.

Babenko *et al.* (2009) developed a boosting framework that can be used to derive online boosting algorithms for various cost functions. Within this framework, the authors derive online boosting algorithm for logistic regression.

Leistner *et al.* (2009) evaluated various on-line boosting algorithms in form of a competitive study on standard machine learning problems as well as on common computer vision applications such as tracking and autonomous training of object detectors. Authors' results show that using on-line gradient-boost (Mason *et al.*, 1999) with robust loss functions leads to superior results.

Elwel and Polikar (2011) proposed Learn++.NSE algorithm that learns incrementally, without requiring access to previously seen data. This algorithm trains one new classifier for each batch of data it receives, and combines these classifiers using a dynamically weighted majority voting.

https://doi.org/10.1017/S0269888913000313 Published online by Cambridge University Press

The algorithm determines the voting weights, based on each classifiers time-adjusted accuracy on current and past environments.

Minku and Xin (2012) inspired by that different diversity levels in an ensemble of learning machines are required in order to maintain high generalization on both old and new concepts, proposed a new online ensemble learning approach called diversity for dealing with drifts. Recently, Brzezinski and Stefanowski (2013) propose a new data stream classifier, called the accuracy updated ensemble (AUE2). AUE2 combines accuracy-based weighting mechanisms known from block-based ensembles with the incremental nature of Hoeffding trees.

6 Variants that combine bagging and boosting

MultiBoosting (Webb, 2000) is another method that can be considered as wagging committees formed by AdaBoost. Kotsiantis and Pintelas (2004) built an ensemble using a voting methodology of bagging and boosting ensembles with 10 sub-classifiers in each one. The authors performed a comparison with simple bagging and boosting ensembles with 25 sub-classifiers, as well as other well known combining methods, on standard benchmark datasets and state that the presented technique was the most accurate.

Yasumura *et al.* (2005) proposed another ensemble learning method, integration of boosting and bagging (IBB). This method creates initial classifiers by bagging, and then builds sub-classifiers by boosting using the previously created classifiers. IBB uses a reweighting technique and data adaptation technique. The reweighting technique increases a weight of a sample which is misclassified by both the ensemble classifier and previously created sub-classifier. The data adaptation is realized by controlling the number of iteration in boosting.

Martinez-Munoz and Suarez (2007) used boosting to determine the order in which classifiers are aggregated in a bagging ensemble. Early stopping in the aggregation of the classifiers in the ordered bagging ensemble allows the identification of ensembles that require less memory for storage and classify faster. In all the classification problems investigated by Martinez-Munoz and Suarez (2007) pruned ensembles with 20% of the original classifiers showed statistically significant improvements over bagging. However, in problems where boosting is superior to bagging, these improvements were not sufficient to reach the accuracy of the corresponding boosting ensembles.

In the paper of Zhang *et al.* (2009), boosting is used to determine the order in which base predictors are aggregated into a double-bagging ensemble, and a sub-ensemble is constructed by early stopping the aggregation process based on two heuristic stopping rules. In all the investigated classification problems of Zhang *et al.* (2009), the pruned ensembles give better accuracy than bagging, boosting ensembles in most cases.

7 Conclusion

This paper describes the best-know bagging and boosting variants in relative detail. We should remark that our list of references is not a comprehensive list of papers discussing bagging and boosting variants: our aim was to produce a review of the key ideas, rather than a simple list of all publications which had discussed or made use of those ideas. Despite this, we hope that the references cited cover the major theoretical issues, and provide access to the main branches of the literature dealing with such methods, guiding the researcher in interesting research directions. After a better understanding of each method, the possibility of integrating two or more algorithms together to solve a problem could be investigated.

In many classification tasks, bagging improves the generalization performance of individual base learners. However, due to need of repeated executions of the algorithm, the computational requirements to estimate generalization error of this algorithm by cross validation, is very expensive. In order to address this problem, Hernandez-Lobato *et al.* (2007) investigated the properties of an efficient estimator based on the Monte Carlo approach (Esposito & Saitta, 2003).

https://doi.org/10.1017/S0269888913000313 Published online by Cambridge University Press

Bagging is regarded as a general technique appropriate for most problems since (i) its performance is either similar or significantly better than a single classifier (but it does not hurt performance as boosting may do) and (ii) it allows a parallelized development of its members, while boosting is a necessarily sequential approach.

However, boosting has greater average effect, leading to substantially larger error reductions than bagging on average (Bauer & Kohavi, 1999). For binary problems, as the boosting algorithms iterate, keeping the error rate of weak learner below 50% is not too difficult. In the multi-case, the error rate still must remain below 50%. For the n-class case, a random classifier will give an error rate of $(n-1)/n$ which can be far from 50%. Therefore, for the multi-class case, we must investigate weak learners that are enough powerful or we should split the multi-class problem in many binaries.

Some weak learners are unable even in principle to represent complex decision boundaries, while overly complex weak learners quickly lead to overfitting. In boosting, this problem appears strongly related to the difficult problem of feature selection. The selection of base learner influences the performance of ensemble methods greatly. Decision trees and neural networks are most often used as base learners. The performance enhancement for the ensemble usually comes with the first 10 classifiers combined, but for boosting decision trees may continue to further improve with larger ensemble sizes.

Shirai *et al.* (2009) compared boosting with bagging in different strengths of learning algorithms. The authors' conclusions are (1) boosting is effective for simpler algorithms while bagging is effective for more complicated algorithms, (2) boosting algorithm using random feature subset selection works almost always regardless of base algorithm, and (3) bagging using random feature subset selection works better than simple bagging.

Khoshgoftaar *et al.* (2011) compared the performance of several boosting and bagging techniques in the context of learning from imbalanced and noisy binary-class data. The experiments proved that the bagging techniques generally outperform boosting, and hence in noisy data environments, bagging is better method for handling class imbalance.

Simple majority vote is a not good choice of the combining rule for bagging and boosting. A weighted voting is a better choice. Bootstrapping is most effective when the training sample size is smaller than the data dimensionality. In this case, one obtains bootstrap replicates with the dissimilar statistical characteristics. In the other hand, in bagging, sub-classifiers become less diverse and thus less effective when the number of training instances increases. Boosting constructs more diverse classifiers and becomes more useful when the training sample size increases.

Reyzin and Schapire (2006) found that Breiman considered minimum margin instead of average or median margin, which suggests that the margin-based explanation still has chance to survive. If this explanation succeeds, a strong connection between AdaBoost and SVM could be found. It is obvious that this topic is well worth studying.

Open problems still remain the relationship between the presented ensemble methods and dataset characteristics as well the applicability of these methods to active learning and semi-supervised learning.

When trained on large datasets and possibly infinite data streams, it is needed to have an online algorithm. Massive online analysis (MOA) is a software environment for implementing algorithms and running experiments for online learning from evolving data streams (Bifet *et al.*, 2010b). MOA includes a collection of online methods such as boosting, bagging, and Hoeffding trees.

Despite the obvious advantages, ensemble methods have at least three weaknesses. The first weakness is increased storage as all component classifiers, instead of a single classifier, need to be stored. The second weakness is increased computation because in order to classify an input query, all component classifiers (instead of a single classifier) must be processed. The last weakness is decreased comprehensibility. With involvement of multiple classifiers it is more difficult to perceive the underlying reasoning process leading to a decision. A first attempt for extracting meaningful rules from ensembles was presented in Wall *et al.* (2003).

https://doi.org/10.1017/S0269888913000313 Published online by Cambridge University Press

Acknowledgement

The author would like to thank the reviewers for their insightful and constructive comments, which contributed greatly to the overall quality and completeness of the paper.

References

Aksela, M. & Laaksonen, J. 2006. Using diversity of errors for selecting members of a committee classifier. *Journal of Pattern Recognition* **39**(4), 608–623.

Alaiz-Rodriguez, R. 2008. Local decision bagging of binary neural classifiers. *Lecture Notes in Artificial Intelligence* **5032**, 1–12.

Amores, J., Sebe, N. & Radeva, P. 2006. Boosting the distance estimation application to the K-nearest neighbor classifier. *Pattern Recognition Letters* **27**, 201–209.

Babenko, B., Yang, M. H. & Belongie, S. 2009. A family of online boosting algorithms. In *IEEE 12th International Conference on Computer Vision Workshops*, September 27–October 4, Kyoto, 1346–1353.

Bakker, B. & Heskes, T. 2003. Clustering ensembles of neural network models. *Neural Networks* **16**(2), 261–269.

Banfield, R. E., Hall, L. O. & Bowyer, K. W. 2007. A comparison of decision tree ensemble creation techniques. *IEEE Transactions Pattern Analysis and Machine Intelligence* **29**, 173–180.

Banfield, R. E., Hall, L. O., Bowyer, K. W. & Kegelmeyer, W. P. 2005. Ensemble diversity measures and their application to thinning. *Information Fusion* **6**(1), 49–62.

Baszczyski, J., Sowiski, R. & Stefanowski, J. 2010. Variable consistency bagging ensembles. *Lecture Notes in Computer Science* **5946**, 40–52.

Bauer, E. & Kohavi, R. 1999. An empirical comparison of voting classification algorithms: bagging, voting, and variants. *Machine Learning* **36**, 105–139.

Bifet, A., Holmes, G. & Pfahringer, B. 2010a. Leveraging bagging for evolving data streams. *Lecture Notes in Artificial Intelligence* **6321**, 135–150.

Bifet, A., Holmes, G., Kirkby, R. & Pfahringer, B. 2010b. MOA: massive online analysis. *Journal of Machine Learning Research* **11**, 1601–1604.

Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R. & Gavalda, R. 2009a. New ensemble methods for evolving data streams. In *15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, 139–148.

Bifet, A., Holmes, G., Pfahringer, B. & Gavalda, R. 2009b. Improving adaptive bagging methods for evolving data streams. *Lecture Notes in Artificial Intelligence* **5828**, 23–37.

Bradley, J. K. & Schapire, R. E. 2008. Filterboost: regression and classication on large datasets. *Advances in Neural Information Processing Systems* **20**, 185–192.

Breiman, L. 1996. Bagging predictors. *Machine Learning* **24**, 123–140.

Breiman, L. 1999. Pasting small votes for classification in large databases and on-line. *Machine Learning* **36**(1–2), 85–103.

Breiman, L. 1999b. Prediction games and arcing algorithms. *Neural Computation* **11**(7), 1493–1517.

Breiman, L. 2000. Randomizing outputs to increase prediction accuracy. *Machine Learning* **40**, 229–242.

Breiman, L. 2001. Random forests. *Machine Learning* **45**(1), 5–32.

Brown, G., Wyatt, J., Harris, R. & Yao, X. 2005. Diversity creation methods: a survey and categorisation. *Information Fusion* **6**(1), 5–20.

Bryll, R., Gutierrez-Osuna, R. & Quek, F. 2003. Attribute bagging: improving accuracy of classifier ensemble by using random feature subsets. *Pattern Recognition* **36**, 1291–1302.

Brzezinski, D. & Stefanowski, J. 2013. Reacting to different types of concept drift: the accuracy updated ensemble algorithm. *IEEE Transactions on Neural Networks and Learning Systems* **24**(7), in press.

Buhlmann, P. 2012. Bagging, boosting and ensemble methods. *Handbook of Computational Statistics – Springer Handbooks of Computational Statistics*, 985–1022.

Buhlmann, P. & Yu, B. 2002. Analyzing bagging. *The Annals of Statistics* **30**(4), 927–961.

Buja, W. S. 2006. Observations on bagging. *Statistica Sinica* **16**, 323–351.

Cai, Q-T., Chun-Yi, P. & Chang-Shui, Z. 2008a. A weighted subspace approach for improving bagging performance. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, March 31–April 4, Las Vegas, 3341–3344.

Cai, Q-T., Chun-Yi, P. & Chang-Shui, Z. 2008b. Cost-sensitive boosting algorithms as gradient descent. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, March 31–April 4, Las Vegas, 2009–2012.

Chawla, N. V., Hall, L. O., Bowyer, K. W. & Kegelmeyer, W. P. 2002. SMOTE: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research* **16**, 321–357.

Chawla, N. V., Lazarevic, A., Hall, L. O. & Bowyer, K. 2003. SMOTE-Boost: improving prediction of the minority class in boosting. *Principles of Knowledge Discovery in Databases, PKDD-2003*, 107–119.

https://doi.org/10.1017/S0269888913000313 Published online by Cambridge University Press

Coelho, A. & Nascimento, D. 2010. On the evolutionary design of heterogeneous bagging models. *Neurocomputing* **73**(16–18), 3319–3322.
Croux, C., Joossens, K. & Lemmens, A. 2007. Trimmed bagging. *Computational Statistics and Data Analysis* **52**, 362–368.
Derbeko, P., Yaniv, R. & Meir, R. 2002. Variance optimized bagging. *Lecture Notes in Artificial Intelligence* **2430**, 60–72.
Dietterich, T. 2000. An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization. *Machine Learning* **40**, 139–157.
Domingos, P. 2000. A unified bias-variance decomposition for zero-one and squared loss. In *Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, 564–569.
Elwel, R. & Polikar, R. 2011. Incremental learning of concept drift in nonstationary environments. *IEEE Transactions on Neural Networks* **22**(10), 1517–1531.
Esposito, R. & Saitta, L. 2003. Monte Carlo theory as an explanation of bagging and boosting. In *18th International Joint Conference on Artificial Intelligence IJCAI'03*, Acapulco, Mexico, 499–504.
Faisal, Z., Uddin, M. M. & Hirose, H. 2010. On selecting additional predictive models in double bagging type ensemble method. *Lecture Notes in Computer Science* **6019**, 199–208.
Fan, W., Stolfo, S. J., Zhang, J. & Chan, P. K. 1999. AdaCost: misclassification cost-sensitive boosting. In *16th International Conference on Machine Learning*, Slovenia, 97–105.
Frank, E. & Pfahringer, B. 2006. Improving on bagging with input smearing. *Lecture Notes in Artificial Intelligence* **3918**, 97–106.
Freund, Y. & Schapire, R. E. 1996a. Experiments with a new boosting algorithm. In *13th International Conference on Machine Learning*, Bari, Italy, 148–156.
Freund, Y. & Schapire, R. E. 1996b. Game theory, on-line prediction and boosting. In *Ninth Annual Conference On Computational Learning Theory-COLT '96*, Desenzano sul Garda, Italy, 325–332.
Freund, Y. & Schapire, R. E. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences* **55**(1), 119–139.
Friedman, J. H. & Hall, P. 2007. On bagging and nonlinear estimation. *Journal of Statistical Planning and Inference* **137**(3), 669–683.
Friedman, J. H., Hastie, T. & Tibshirani, R. 2000. Additive logistic regression: a statistical view of boosting. *Annals of Statistics* **38**, 367–378.
Fu, Q., Hu, S. X. & Zhao, S. Y. 2005. Clustering-based selective neural network ensemble. *Journal of Zhejiang University Science* **6A**(5), 387–392.
Fumera, G., Roli, F. & Serrau, A. 2005. Dynamics of variance reduction in bagging and other techniques based on randomisation. *Lecture Notes in Computer Science* **3541**, 316–325.
Fumera, G., Roli, F. & Serrau, A. 2008. A theoretical analysis of bagging as a linear combination of classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **30**(7), 1293–1299.
Fushiki, T. 2010. Bayesian bootstrap prediction. *Journal of Statistical Planning and Inference* **140**, 65–74.
Galar, M., Fernandez, A., Barrenechea, E., Bustince, H. & Herrera, F. 2012. A review on ensembles for the class imbalance problem: bagging-boosting and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* **42**(4), 463–484.
Gama, J. 2010. *Knowledge Discovery from Data Streams*. CRC Press.
Gambina, A., Szczureka, E., Dutkowski, J., Bakunc, M. & Dadlezc, M. 2009. Classification of peptidemass fingerprint data by novel no-regret boosting method. *Computers in Biology and Medicine* **39**, 460–473.
Gao, W. & Zhou, Z-H. 2010. Approximation Stability and Boosting. *Lecture Notes in Artificial Intelligence* **6331**, 59–73.
Gao, Y., Gao, F. & Guan, X. 2010. Improved boosting algorithm with adaptive filtration. In *8th World Congress on Intelligent Control and Automation*, July 6–9, Jinan, China, 3173–3178.
Garcia-Pedrajas, N. 2009. Supervised projection approach for boosting classifiers. *Pattern Recognition* **42**, 1742–1760.
Garcia-Pedrajas, N. & Ortiz-Boyer, D. 2008. Boosting random subspace method. *Neural Networks* **21**, 1344–1362.
Garcia-Pedrajas, N. & Ortiz-Boyer, D. 2009. Boosting k-nearest neighbor classifier by means of input space projection. *Expert Systems with Applications* **36**, 10570–10582.
Gomez-Verdejo, V., Ortega-Moral, M., Arenas-Garcia, J. & Figueiras-Vidal, A. 2006. Boosting by weighting critical and erroneous samples. *Neurocomputing* **69**, 679–685.
Grabner, H. & Bischof, H. 2006. On-line boosting and vision. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, June 17–22, 260–267.
Grandvalet, Y. 2004. Bagging equalizes inuence. *Machine Learning* **55**, 251–270.
Hall, L., Baneld, R., Bowyer, K. & Kegelmeyer, W. 2007. Boosting lite—handling larger datasets and slower base classifiers. *Lecture Notes in Computer Science* **4472**, 161–170.

https://doi.org/10.1017/S0269888913000313 Published online by Cambridge University Press

Hall, P. & Samworth, R. J. 2005. Properties of bagged nearest neighbour classifiers. *Journal of the Royal Statistical Society Series B* **67**(3), 363–379.
Hernandez-Lobato, D., Martinez-Munoz, G. & Suarez, A. 2007. Out of bootstrap estimation of generalization error curves in bagging ensembles. *Lecture Notes in Computer Science* **4881**, 47–56.
Hido, S., Kashima, H. & Takahashi, Y. 2009. Roughly balanced bagging for imbalanced data. *Statistical Analysis and Data Mining* **2**, 412–426.
Ho, T. K. 1998. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **20**(8), 832–844.
Hothorn, T. & Lausen, B. 2003. Double-bagging: combining classifiers by bootstrap aggregation. *Pattern Recognition* **36**(6), 1303–1309.
Hothorn, T. & Lausen, B. 2005. Bundling classifiers by bagging trees. *Computational Statistics and Data Analysis* **49**, 1068–1078.
Jiang, Y., Jin-Jiang, L., Gang, L., Honghua, D. & Zhi-Hua, Z. 2005. Dependency bagging. *Lecture Notes in Artificial Intelligence* **3641**, 491–500.
Jimnez-Gamero, M. D., Muoz-Garca, J. & Pino-Mejas, R. 2004. Reduced bootstrap for the median. *Statistica Sinica* **14**, 1179–1198.
Joshi, M. V., Kumar, V. & Agarwal, R. C. 2001. Evaluating boosting algorithms to classify rare classes: comparison and improvements. *IEEE International Conference on Data Mining*, 257–264.
Kalai, A. & Servedio, R. 2005. Boosting in the presence of noise. *Journal of Computer and System Sciences* **71**, 266–290.
Khoshgoftaar, T. M., Van Hulse, J. & Napolitano, A. 2011. Comparing boosting and bagging techniques with noisy and imbalanced data. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans* **41**(3), 552–568.
Koco, S. & Capponi, C. 2011. A boosting approach to multiview classification with cooperation. In *European Conference on Machine Learning and Knowledge Discovery in Databases ECML-PKDD'11*, Athens, 209–228.
Kotsiantis, S. & Pintelas, P. 2004. Combining bagging and boosting. *International Journal of Computational Intelligence* **1**(4), 324–333.
Kotsiantis, S. B. & Kanellopoulos, D. 2010. Bagging different instead of similar models for regression and classification problems. *International Journal of Computer Applications in Technology* **37**(1), 20–28.
Kotsiantis, S. B., Kanellopoulos, D. & Pintelas, P. E. 2006. Local boosting of decision stumps for regression and classification problems. *Journal of Computers* **1**(4), 30–37.
Kotsiantis, S. B., Tsekouras, G. E. & Pintelas, P. E. 2005. Local bagging of decision stumps. In *18th International Conference on Innovations in Applied Artificial Intelligence*, Bari, Italy, 406–411.
Kuncheva, L. I., Skurichina, M. & Duin, R. P. W. 2002. An experimental study on diversity for bagging and boosting with linear classifiers. *Information Fusion* **3**, 245–258.
Kuncheva, L. & Whitaker, C. J. 2002. Using diversity with three variants of boosting: aggressive. *Lecture Notes in Computer Science* **2364**, 81–90.
Kuncheva, L. I. & Whitaker, C. J. 2003. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Journal of Machine Learning* **51**, 181–207.
Kuncheva, L. 2004a. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley.
Kuncheva, L. 2004b. Classifier ensembles for changing environments. *Lecture Notes in Computer Science* **3077**, 1–15.
Latinne, P., Debeir, O. & Decaestecker, Ch. 2000. Mixing bagging and multiple feature subsets to improve classification accuracy of decision tree combination. In *10th Belgian-Dutch Conference on Machine Learning*. Tilburg University, 15–22.
Le, D.-D. & Satoh, S. 2007. Ent-boost: boosting using entropy measures for robust object detection. *Pattern Recognition Letters* **28**, 1083–1090.
Lee, H. & Clyde, M. A. 2004. Lossless online bayesian bagging. *Journal of Machine Learning Research* **5**, 143–151.
Leistner, C., Saffari, A., Roth, P. & Bischof, H. 2009. On robustness of on-line boosting—a competitive study. In *IEEE 12th International Conference on Computer Vision Workshops*, Kyoto, Japan, 1362–1369.
Leskes, B. & Torenvliet, L. 2008. The value of agreement a new boosting algorithm. *Journal of Computer and System Sciences* **74**, 557–586.
Li, C. 2007. Classifying imbalanced data using a bagging ensemble variation (BEV). *45th Annual Southeast Regional Conference*, 203–208.
Li, G.-Z. & Yang, J. Y. 2008. Feature selection for ensemble learning and its application. In *Machine Learning in Bioinformatics*, Zhang, Y.-Q. & Rajapakse, J. C. (eds.). Wiley.
Li, W., Gao, X., Zhu, Y., Ramesh, V. & Boult, T. 2005. On the small sample performance of boosted classifiers. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, June 20–25, 574–581.

https://doi.org/10.1017/S0269888913000313 Published online by Cambridge University Press

Littlestone, N. & Warmuth, M. K. 1994. The weighted majority algorithm. *Information and Computation* **108**(2), 212–261.
Liu, X. & Yu, T. 2007. Gradient feature selection for online boosting. In *IEEE 11th International Conference on Computer Vision (ICCV 2007)*, Rio de Janeiro, Brazil, 1–8.
Lu, Y., Tian, Q. & Huang, T. 2007. Interactive boosting for image classification. *Lecture Notes in Computer Science* **4472**, 180–189.
Markowitz, H. 1959. *Portfolio Selection: Efficient Diversification of Investments*. Yale University Press.
Martinez-Munoz, G. & Suarez, A. 2007. Using boosting to prune bagging ensembles. *Pattern Recognition Letters* **28**(1), 156–165.
Martinez-Munoz, G. & Suarez, A. 2010. Out-of-bag estimation of the optimal sample size in bagging. *Pattern Recognition* **43**, 143–152.
Martinez-Munoz, G., Hernandez-Lobato, D. & Suarez, A. 2007. Selection of decision stumps in bagging ensembles. *Lecture Notes in Computer Science* **4668**, 319–328.
Mason, L., Baxter, J., Bartlett, P. & Frean, M. 1999. Functional gradient techniques for combining hypotheses. *Advances in Large Margin Classifiers*. MIT Press, 221–247.
Melville, P. & Mooney, R. 2005. Creating diversity in ensembles using artificial data. *Information Fusion* **6**, 99–111.
Minku, L. L. & Xin, Y. 2012. DDD: a new ensemble approach for dealing with concept drift. *IEEE Transactions on Knowledge and Data Engineering* **24**(4), 619–633.
Nanculef, R., Valle, C., Allende, H. & Moraga, C. 2007. Bagging with asymmetric costs for misclassified and correctly classified examples. *Lecture Notes in Computer Science* **4756**, 694–703.
Oza, N. C. 2003. Boosting with averaged weight vectors. *Lecture Notes in Computer Science* **2709**, 15–24.
Oza, N. 2005. Online bagging and boosting. In *2005 IEEE International Conference on Systems, Man and Cybernetics*, October 10–12, Hawaii, USA, 2340–2345.
O'Sullivan, J., Langford, J., Caruna, R. & Blum, A. 2000. Featureboost: a metalearning algorithm that improves model robustness. *17th International Conference on Machine Learning*, 703–710.
Panov, P. & Dzeroski, S. 2007. Combining bagging and random subspaces to create better ensembles. *Lecture Notes in Computer Science* **4723**, 118–129.
Pelossof, R., Jones, M., Vovsha, I. & Rudin, C. 2009. Online coordinate boosting. In *IEEE 12th International Conference on Computer Vision Workshops*, Kyoto, Japan, 1354–1361.
Peng, J., Barbu, C., Seetharaman, G., Fan, W., Wu, X. & Palaniappan, K. 2011. ShareBoost: boosting for multi-view learning with performance guarantees. *Lecture Notes in Computer Science* **6912**, 597–612.
Phama, T. & Smeulders, A. 2008. Quadratic boosting. *Pattern Recognition* **41**, 331–341.
Pino-Mejias, R., Jimenez-Gamero, M., Cubiles-de-la-Vega, M. & Pascual-Acosta, A. 2008. Reduced bootstrap aggregating of learning algorithms. *Pattern Recognition Letters* **29**, 265–271.
Pino-Mejias, R., Cubiles-de-la-Vega, M., Lpez-Coello, M., Silva-Ramrez, E. & Jimnez-Gamero, M. 2004. Bagging classification models with reduced bootstrap. *Lecture Notes in Computer Science* **3138**, 966–973.
Puuronen, S., Skrypnyk, I. & Tsymbal, A. 2001. Ensemble feature selection based on contextual merit and correlation heuristics. *Lecture Notes in Computer Science* **2151**, 155–168.
Redpath, D. B. & Lebart, K. 2005. Boosting feature selection. *Lecture Notes in Computer Science* **3686**, 305–314.
Reyzin, L. & Schapire, R. E. 2006. How boosting the margin can also boost classifier complexity. In *23rd International Conference on Machine Learning*, Pittsburgh, 753–760.
Rodriguez, J. J., Kuncheva, L. I. & Alonso, C. J. 2006. Rotation forest: a new classier ensemble method. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **28**(10), 1619–1630.
Rodriguez, J. J. & Maudes, J. 2008. Boosting recombined weak classifiers. *Pattern Recognition Letters* **29**, 1049–1059.
Schapire, R., Freund, Y., Bartlett, P. & Lee, W. S. 1997. Boosting the margin: a new explanation for the effectiveness of voting methods. In *Fourteenth International Conference on Machine Learning-ICML'97*, 322–330.
Schapire, R. E. & Singer, Y. 1999. Improved boosting algorithms using confidence-rated predictions. *Machine Learning* **37**, 297–336.
Seiffert, C., Khoshgoftaar, T., Hulse, J. & Napolitano, A. 2008. Resampling or reweighting: a comparison of boosting implementations. In *20th IEEE International Conference on Tools with Artificial Intelligence-ICTAI'08*, Ohio, USA, 445–451.
Seni, G. & Elder, J. 2010. Ensemble methods in data mining: improving accuracy through combining predictions. *Synthesis Lectures on Data Mining and Knowledge Discovery* **2**(1), 1–126.
Servedio, R. A. 2003. Smooth boosting and learning with malicious noise. *The Journal of Machine Learning Research* **4**, 633–648.
Shen, C. & Li, H. 2010. Boosting through optimization of margin distributions. *IEEE Transactions on Neural Networks* **21**(4), 659–667.

https://doi.org/10.1017/S0269888913000313 Published online by Cambridge University Press

Shirai, S., Kudo, M. & Nakamura, A. 2008. Bagging, random subspace method and biding. *Lecture Notes in Computer Science* **5342**, 801–810.

Shirai, S., Kudo, M. & Nakamura, A. 2009. Comparison of bagging and boosting algorithms on sample and feature weighting. *Lecture Notes in Computer Science* **5519**, 22–31.

Skurichina, M. & Duin, R. 2000. The role of combining rules in bagging and boosting. *Lecture Notes in Computer Science* **1876**, 631–640.

Sohn, S. Y. & Shin, H. W. 2007. Experimental study for the comparison of classifier combination methods. *Pattern Recognition* **40**, 33–40.

Stefanowski, J. 2007. Combining answers of sub-classifiers in the bagging-feature ensembles. *Lecture Notes in Artificial Intelligence* **4585**, 574–583.

Su, X., Khoshgoftarr, T. M. & Zhu, X. 2008. VoB predictors: voting on bagging classifications. In *19th International Conference on Pattern Recognition-ICPR'2008*, December 8–11, Florida, USA, 1–4.

Sun, Y., Kamel, M. S., Wong, A. & Wang, Y. 2007. Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition* **40**, 3358–3378.

Tang, W. 2003. Selective ensemble of decision trees. *Lecture Notes in Artificial Intelligence* **2639**, 476–483.

Tao, D., Tang, X., Li, X. & Wu, X. 2006. Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **28**(7), 1088–1099.

Terabe, M., Washio, T. & Motoda, H. 2001. The effect of subsampling rate on subagging performance. *ECML2001/PKDD2001*, 48–55.

Ting, K. & Witten, I. 1997. Stacking bagged and dagged models. In *Fourteenth International Conference on Machine Learning-ICML '97*, Tennessee, USA, 367–375.

Torres-Sospedra, J., Hernandez-Espinosa, C. & Fernandez-Redondo, M. 2007a. Mixing aveboost and conserboost to improve boosting methods. In *International Joint Conference on Neural Networks*, Orlando, Florida, USA, August 12–17, 672–677.

Torres-Sospedra, J., Hernandez-Espinosa, C. & Fernandez-Redondo, M. 2007b. Designing a multilayer feedforward ensemble with the weighted conservative boosting algorithm. In *International Joint Conference on Neural Networks*, Orlando, Florida, USA, August 12–17, 684–689.

Torres-Sospedra, J., Hernandez-Espinosa, C. & Fernandez-Redondo, M. 2008. Researching on combining boosting ensembles. In *International Joint Conference on Neural Networks-IJCNN2008*, Hong Kong, 2290–2295.

Tsaoa, C. & Chang, Y. I. 2007. A stochastic approximation view of boosting. *Computational Statistics and Data Analysis* **52**, 325–334.

Tsymbal, A. & Puuronen, S. 2000. Bagging and boosting with dynamic integration of classifiers. *Lecture Notes in Artificial Intelligence* **1910**, 116–125.

Valentini, G. & Masuli, F. 2002. Ensembles of learning machines. *Lecture Notes in Computer Science* **2486**, 3–19.

Valentini, G. & Dietterich, T. G. 2003. Low bias bagged support vector machines. In *20th International Conference on Machine Learning ICML-2003*, Washington, USA, 752–759.

Vezhnevets, A. & Barinova, O. 2007. Avoiding boosting overfitting by removing confusing shamples. In *ECML2007*, Poland, September, 430–441.

Wall, R., Cunningham, P., Walsh, P. & Byrne, S. 2003. Explaining the output of ensembles in medical decision support on a case by case basis. *Artificial Intelligence in Medicine* **28**(2), 191–206.

Wang, X. & Wang, H. 2006. Classification by evolutionary ensembles. *Pattern Recognition* **39**, 595–607.

Wang, C.-M., Yang, H.-Z., Li, F.-C. & Fu, R.-X. 2006. Two stages based adaptive sampling boosting method. In *Fifth International Conference on Machine Learning and Cybernetics*, Dalian, August 13–16, 2925–2927.

Wang, S. & Yao, X. 2009. Diversity analysis on imbalanced data sets by using ensemble models. *IEEE Symposium on Computational Intelligence and Data Mining*, 324–331.

Wang, W. & Zhou, Z.-H. 2010. A new analysis of co-training. In *27th International Conference on Machine Learning-ICML'10*, Haifa, Israel, 1135–1142.

Webb, G. I. 2000. MultiBoosting: a technique for combining boosting and wagging. *Machine Learning* **40**, 159–196.

Wolpert, D. H. 1994. Stacked generalization. *Neural Networks* **5**(6), 1289–1301.

Xu, W., Meiyun, Z., Mingtao, Z. & He, R. 2010. Constraint bagging for stock price prediction using neural networks. In *International Conference on Modelling, Identification and Control*, Okayama, Japan, July 17–19, 606–610.

Xu, X. & Zhang, A. 2006. Boost feature subset selection: a new gene selection algorithm for microarray data set. In *International Conference on Computational Science*, UK, 670–677.

Yang, L., Gong, W., Gu, X., Li, W. & Liu, Y. 2009. Bagging null space locality preserving discriminant classifiers for face recognition. *Pattern Recognition* **42**, 1853–1858.

https://doi.org/10.1017/S0269888913000313 Published online by Cambridge University Press

Yasumura, Y., Kitani, N. & Uehara, K. 2005. Integration of bagging and boosting with a new reweighting technique. In *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC05)*, Vienna, Austria, 338–343.

Yi, X.-C., Ha, Z. & Liu, C.-P. 2004. Selective bagging based incremental learning. In *Third International Conference on Machine Learning and Cyhemetics*, Shanghai, August 26–29, 2412–2417.

Yin, H. & Dong, H. 2011. The problem of noise in classification: Past, current and future work. In *2011 IEEE 3rd International Conference on Communication Software and Networks (ICCSN)*, May 27–29, 412–416.

Yin, X.-C., Liu, C.-P. & Zhi, H. 2005. Feature combination using boosting. *Pattern Recognition Letters* **26**, 2195–2205.

Zaman, F. & Hirose, H. 2008. A robust bagging method using median as a combination rule. In *IEEE 8th International Conference on Computer and Information Technology Workshops*, Dhaka, Bangladesh, 55–60.

Zhang, C. X. & Zhang, J. S. 2008a. A local boosting algorithm for solving classification problems. *Computational Statistics & Data Analysis* **52**(4), 1928–1941.

Zhang, C. X. & Zhang, J. S. 2008b. RotBoost: a technique for combining Rotation Forest and AdaBoost. *Pattern Recognition Letters* **29**, 1524–1536.

Zhang, C. X., Zhang, J. S. & Zhang, G.-Y. 2008. An efficient modified boosting method for solving classification problems. *Journal of Computational and Applied Mathematics* **214**, 381–392.

Zhang, C. X., Zhang, J. S. & Zhang, G.-Y. 2009. Using boosting to prune double-bagging ensembles. *Computational Statistics and Data Analysis* **53**, 1218–1231.

Zhang, D., Zhou, X., Leung, S. & Zheng, J. 2010. Vertical bagging decision trees model for credit scoring. *Expert Systems with Applications* **37**, 7838–7843.

Zhou, Z.-H., Wu, J. & Tang, W. 2002. Ensembling neural networks: many could be better than all. *Artificial Intelligence* **137**(1–2), 239–263.

Zhou, Z. H. & Yu, Y. 2005a. Adapt bagging to nearest neighbor classiers. *Journal of Computer Science and Technology* **20**(1), 48–54.

Zhou, Z. H. & Yu, Y. 2005b. Ensembling local learners through multimodal perturbation. *IEEE Transactions on Systems, Man, and Cybernetics Part B:Cybernetics* **35**(4), 725–735.

Zhu, X. & Yang, Y. 2008. A lazy bagging approach to classification. *Pattern Recognition* **41**, 2980–2992.

https://doi.org/10.1017/S0269888913000313 Published online by Cambridge University Press