ARTICLE IN PRESS

Computational Statistics and Data Analysis ▮ (▮▮▮▮) ▮▮▮–▮▮▮

Contents lists available at ScienceDirect

Computational Statistics and Data Analysis

journal homepage: www.elsevier.com/locate/csda

Noise peeling methods to improve boosting algorithms

Waldyn Martinez [a,*], J. Brian Gray [b]

[a] Department of Information Systems and Analytics, Miami University, Oxford, OH, USA

[b] Department of Information Systems, Statistics and Management Science, University of Alabama, Tuscaloosa, AL, USA

ARTICLE INFO

Article history:
Received 29 April 2014
Received in revised form 26 June 2015
Accepted 27 June 2015
Available online xxxx

Keywords:
AdaBoost
Ensembles
Generalization error
Noise detection
Noise filtering
Outliers

ABSTRACT

Boosting refers to a family of methods that combine sequences of individual classifiers into highly accurate ensemble models through weighted voting. AdaBoost, short for "Adaptive Boosting", is the most well-known boosting algorithm. AdaBoost has many strengths. Among them, there is sufficient empirical evidence pointing to its performance being generally superior to that of individual classifiers. In addition, even when combining a large number of weak learners, AdaBoost can be very robust to overfitting usually with lower generalization error than other competing ensemble methodologies, such as bagging and random forests. However, AdaBoost, as most hard margin classifiers, tends to be sensitive to outliers and noisy data, since it assigns observations that have been misclassified a higher weight in subsequent iterations. It has recently been proven that for any booster with a potential convex loss function, and any nonzero random classification noise rate, there is a data set, which can be efficiently learnable by the booster if there is no noise, but cannot be learned with accuracy better than 1/2 with random classification noise present. Several techniques to identify and potentially delete (peel) noisy samples in binary classification are proposed in order to improve the performance of AdaBoost. It is found that peeling methods generally perform better than AdaBoost and other noise resistant boosters, especially when high levels of noise are present in the data.

1. Introduction

Boosting algorithms combine sets of weak classifiers (learners) $h_t(\boldsymbol{x})$, $t = 1, 2, \ldots, T$, which are typically decision trees, into ensemble predictions in the form of weighted voting. The development of boosting algorithms has its roots in the Probably Approximately Correct (PAC) learning theory (Valiant, 1984). The PAC learning theory states that a concept class is strongly learnable if a learning algorithm is able to output a hypothesis with arbitrary accuracy in polynomial time, when sufficient training samples are readily available. A weakly learnable concept class drops the "arbitrary accuracy" to performing only slightly better than random. Schapire (1990) provided the first instance of an algorithm able to "boost" a weak classifier to a strong PAC learner. Boosting refers to a family of methods able to improve the accuracy of the weak classifiers by obtaining useful information from all the different learners combined. AdaBoost (Freund and Schapire, 1997), the most well-known boosting algorithm, possesses many desirable properties, including the strong PAC learning property. Generally, AdaBoost outperforms not only individual classifiers, but also most competing ensemble methodologies in many aspects. For example, under certain conditions, the test set error rates for subsequent iterations of AdaBoost continue

* Correspondence to: Department of Information Systems and Analytics, FSB 2020, Miami University, Oxford, OH 45056, USA. Tel.: +1 513 529 2154; fax: +1 513 529 9689.

E-mail addresses: martinwg@miamioh.edu (W. Martinez), bgray@cba.ua.edu (J.B. Gray).

http://dx.doi.org/10.1016/j.csda.2015.06.010
0167-9473/

to decrease even when the training error has hit zero (Schapire et al., 1998; Opitz and Maclin, 1999). This property is remarkable, since it is expected that a more complex classifier would overfit the data easily. Additionally, the flexibility of using any weak classifier (with the ability to be trained with weighted data) has made AdaBoost a popular method in data mining and machine learning. On the other hand, AdaBoost tends to be sensitive to outliers and noise. Grove and Schuurmans (1998), Mason et al. (2000) and Dietterich (2000) have provided evidence that AdaBoost does overfit and the generalization error deteriorates rapidly when the data contain noise. Long and Servedio (2010) proved that for any booster with a potential convex loss function, and any nonzero random classification noise rate, there is a data set, which can be efficiently learnable by the booster if there is no noise, but cannot be learned with accuracy better than 1/2 with random classification noise present. Boosters with potential convex loss functions include the most common boosting algorithms to date.

Many methods that automatically handle noisy data and outliers have been proposed to alleviate the limitations of AdaBoost. These methods fall into the category of accommodating noisy observations by reducing the influence they have on the fit. Algorithms such as BrownBoost (Freund, 2001), GentleBoost (Friedman et al., 2000), LogitBoost (Friedman et al., 2000), MadaBoost (Domingo and Watanabe, 2000), LP_{Reg}-AdaBoost (Rätsch et al., 2001), ν-LP and ν-ARC (Rätsch et al., 1999) mostly attempt to accommodate noise by allowing unusual observations to fall on the wrong side of the prediction in subsequent iterations of the algorithm. Most of these methods rely on convex loss functions, and are therefore susceptible to the shortcomings presented in Long and Servedio (2010), that is, the methods cannot guarantee an accuracy better than 1/2 with random classification noise present. In addition, these algorithms do not generally possess the strong PAC learning property as the original AdaBoost does. On the other hand, using a detection/deletion (also called noise filtering or noise peeling) approach by pre-processing the data might be preferable under high noise circumstances (Gamberger et al., 2000). In this approach outliers are identified and deleted, and AdaBoost is refit to the remaining data, hopefully reducing the noise to levels where AdaBoost performs best.

In the next section, we give some preliminaries on the random classification noise framework as well as AdaBoost and ensembles. In Section 3, we study the performance of boosting methods under noise. In Section 4, we propose several techniques to identify and potentially delete or reduce noise in the training sample, in order to improve the performance of AdaBoost. Finally in Section 5, we compare the proposed techniques with the most common noise resistant boosting algorithms under different noise settings and scenarios.

2. Preliminaries

We assume we are given a training sample $S = \{(\boldsymbol{x_1}, y_1), \ldots, (\boldsymbol{x_n}, y_n)\}$ of data pairs that are independently and identically distributed (i.i.d.) according to an unknown distribution P_{XY} with joint density $p(\boldsymbol{x}, y)$, where $Y \in \{-1, +1\}$ is a binary response variable and $\boldsymbol{x} \in \mathbb{R}^{\boldsymbol{p}}$ are the predictors. The training examples are also assumed to be initially generated by a sampling oracle $EX()$ without noise. The general goal of learning is to estimate a function $H : \mathbb{X} \to Y$ such that H will correctly classify unseen examples $(\boldsymbol{x}, y)$. The function is selected such that the generalization error $R[H]$ (also called the expected risk of the function) is minimized:

$$R[H] = E_{\boldsymbol{x},y} g(y, H(\boldsymbol{x})) = \int g(y, H(\boldsymbol{x}))\, dp(\boldsymbol{x}, y), \tag{1}$$

where $g(y, H(\boldsymbol{x}))$ is a suitable loss function. For binary classification the loss function $I(y_i \neq H(\boldsymbol{x}))$ is typically used, where $I(y_i \neq H(\boldsymbol{x})) = 1$ if $(y_i \neq H(\boldsymbol{x}))$, 0 otherwise. The generalization error cannot be minimized directly because the underlying distribution P_{XY} is unknown. The minimum theoretical value of $R[H]$ is often referred to as Bayes' minimum risk (Breiman, 2000). We refer to $P[.]$ as probabilities with respect to P_{XY} and $\hat{P}[.]$ as the probability with respect to the empirical distribution over the sample S.

For boosting methods, we assume a set of T classifiers $h_t(\boldsymbol{x})$, $t = 1, 2, \ldots, T$, is created from the space of classifiers $\mathcal{H}$. The classifiers are usually called base learners, individual learners or individual classifiers, and they are generated from the training data by a base-learning algorithm $\boldsymbol{\mathcal{B}}$. Each classifier takes a $p \times 1$ input vector $\boldsymbol{x}$ and produces a prediction $h_t(\boldsymbol{x}) \in \{-1, +1\}$ for a binary response variable Y. The combined classifier of the prediction is given by:

$$H(\boldsymbol{x}) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})\right), \tag{2}$$

where $sign : \mathbb{R} \to \{-1, +1\}$, such that $sign(x) = -1$ when $x < 0$, $+1$ otherwise; α_t is the weight associated with the tth weak classifier, $0 \leq \alpha_t \leq 1$ and $\sum_{t=1}^{T} \alpha_t = 1$. The AdaBoost algorithm is arguably the best-known boosting method. AdaBoost is described in Algorithm 1. Unlike earlier boosting methods (Schapire, 1990; Freund, 1995), AdaBoost adjusts adaptively to the errors of the weak hypotheses (hence the name Adaptive Boosting). For example, the step 3(e) in Algorithm 1 ensures more weight is given to instances that are misclassified by the base learning algorithm in previous rounds. The task of AdaBoost is to create a set of weak learners and determine their associated weights $\{\alpha_1, \ldots, \alpha_T\}$ based on a training sample of data S, to produce a combined prediction with small generalization error $R[H]$. Many researchers have pointed out that the margins of the observations may hold the answer as to why the AdaBoost algorithm and most ensemble

Please cite this article in press as: Martinez, W., Gray, J.B., Noise peeling methods to improve boosting algorithms. Computational Statistics and Data Analysis (2015), http://dx.doi.org/10.1016/j.csda.2015.06.010

methods outperform individual classifiers (Schapire et al., 1998; Grove and Schuurmans, 1998; Rosset et al., 2004; Reyzin and Schapire, 2006). The ith margin of an observation is given by:

$$m_i = y_i \sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x}). \tag{3}$$

The margin of the ith observation is equal to the difference in the weighted proportion of weak classifiers correctly predicting the ith observation and the weighted proportion of weak classifiers incorrectly predicting the ith observation, so that $-1 \leq m_i \leq 1$. A margin value of -1 indicates that all of the weak learner predictions were incorrect, while a margin value of $+1$ indicates all of the weak learners correctly predicted the observation. A margin close to $+1$ of the ith training observation can be viewed as a measure of "confidence" in the correct prediction for the ith training observation.

Algorithm 1. AdaBoost

1. [Input]: $S = \{(\boldsymbol{x}_i,\ y_i),\ i = 1,\ 2,\ \ldots,\ n\}$ for T iterations
2. [Initialize]: $D_i^{(1)} = \frac{1}{n}$
3. [Loop]: Do For $t = 1,\ \ldots,\ T$
   (a) Obtain h_t on the sample set $\{S,\ D^{(t)}\}$
   (b) Calculate Error rate as $\varepsilon_t = \sum_{i=1}^{n} D_i^{(t)} I\ (y_i \neq h_t(\boldsymbol{x}_i))$
   (c) Break if $\varepsilon_t = 0$ *or* $\varepsilon_t \geq \frac{1}{2}$. [Output]: $H\ (\boldsymbol{x}) = sign\left(\sum_{j=1}^{t-1} \alpha_j h_j(\boldsymbol{x})\right)$
   (d) Set $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$
   (e) Update $D_i^{(t+1)} = \frac{D_i^{(t)} exp(-\alpha_t y_i h_t(\boldsymbol{x_i}))}{Z_t}$, where

$Z_t = \sum_{i=1}^{n} D_i^{(t)} exp\ (-\alpha_t y_i h_t\ (\boldsymbol{x_i}))$

4. [Output]: $H\ (\boldsymbol{x}) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})\right)$

It has been shown that AdaBoost is a PAC (strong) learner (Freund and Schapire, 1997). A strong PAC learner is formalized in Definition 1.

Definition 1 (*Valiant, 1984*)**.** Let $\mathcal{F}$ be a class of concepts. For every distribution P_{XY}, all concepts $f \in \mathcal{F}$, and all $\varepsilon \in (0,\ 1/2)$, $\delta \in (0,\ 1/2)$, a strong PAC learner has the property that with probability at least $1-\delta$ the base learning algorithm $\boldsymbol{\mathcal{B}}$ outputs a hypothesis h with $P\ (h(X) \neq f(X)) \leq \varepsilon$. $\boldsymbol{\mathcal{B}}$ must run in polynomial time in $\frac{1}{\varepsilon}$ and $\frac{1}{\delta}$ using only a polynomial (in $1/\varepsilon$ and $1/\delta$) number of examples.

In this article, we assume the training sample S is subject to a noise process before being fed into the learning algorithm $\boldsymbol{\mathcal{B}}$. The learning algorithm $\boldsymbol{\mathcal{B}}$ will then have access to data generated by a modified oracle $EX^{\eta}()$, where $0 \leq \eta < 1/2$ indicates the noise level. In other words, the random classification noise model can be described by (4):

$$EX^{\eta}(P_{XY}) = \begin{cases} (\boldsymbol{x},\ y)\,, & \text{with probability } 1-\eta \text{ generated by } EX\ (P_{XY}) \\ (\boldsymbol{x},\ -y)\,, & \text{with probability } \eta \text{ generated by } EX\ (P_{XY})\,. \end{cases} \tag{4}$$

This classification noise framework was initially described by Angluin and Laird (1988). It is based on the idea that the PAC learning theory assumes no noise. Even for low levels of classification noise, it may no longer be possible to find any rule with the PAC learning properties consistent with all examples. Angluin and Laird (1988) generalized the definition of a PAC learner to include the setting under classification noise. Definition 2 formalizes this notion:

Definition 2 (*Angluin and Laird, 1988*)**.** Let $\mathcal{F}$ be a class of concepts. For every distribution P_{XY}, all concepts $f \in \mathcal{F}$, and all $\varepsilon \in (0,\ 1/2)$, $\delta \in (0,\ 1/2)$, and $\eta \in [0,\ 1/2)$, a strong PAC learner in the presence of noise has the property that with probability at least $1-\delta$ the base learning algorithm $\boldsymbol{\mathcal{B}}$ outputs a hypothesis h with $P\ (h(X) \neq f(X)) \leq \varepsilon$. $\boldsymbol{\mathcal{B}}$ must run in polynomial time in $\frac{1}{\varepsilon}$, $\frac{1}{\delta}$ and $\frac{1}{(1-2\eta)}$ using only a polynomial number of examples in $\frac{1}{\varepsilon}$, $\frac{1}{\delta}$ and $\frac{1}{(1-2\eta)}$.

3. Boosting and classification noise

The non-resistance of boosting methods to classification noise and outliers has been well documented by several researchers (Grove and Schuurmans, 1998; Dietterich, 2000; Servedio, 2003; Kalai and Servedio, 2005). In its early development, AdaBoost was believed to not overfit the data, even with an extremely large number of rounds. Consequently, due to the intrinsic relationship between overfitting and noise, AdaBoost was also thought to be somewhat resistant to noisy observations. Grove and Schuurmans (1998) and Dietterich (2000) were among the first to provide evidence that AdaBoost does overfit in the limit and it does so more markedly when there is noise in the data. Dietterich (2000) showed that the generalization error for AdaBoost deteriorates more rapidly than bagging (short for bootstrap aggregation, see Breiman, 1996) when noise is present. Kalai and Servedio (2005) proposed a bound showing that no boosting algorithm can boost its

Please cite this article in press as: Martinez, W., Gray, J.B., Noise peeling methods to improve boosting algorithms. Computational Statistics and Data Analysis (2015), http://dx.doi.org/10.1016/j.csda.2015.06.010

accuracy beyond the noise rate, meaning that the generalization error $R[H]$ achievable will at best be equal to the noise rate η. This, in turn, suggests that AdaBoost does not conform to the standard strong PAC learning properties under the classification noise framework. Other authors have noted that AdaBoost rarely overfits under low levels of noise (see Rätsch et al., 2001).

Rätsch et al. (2001) characterized algorithms that maximize the smallest margin on the training data set as hard margin algorithms. Because of the influence of noisy observations, an algorithm that does not focus on the smallest margin(s) might perform better under noisy settings. These algorithms have been referred to as soft or smooth margin algorithms (see Rätsch et al., 2001), which is a concept originally used in support vector machines (Cortes and Vapnik, 1995). Rätsch et al. (2001) found that AdaBoost achieves a hard margin distribution, because it concentrates its resources on a few hard-to-learn points with asymptotically the same margin (called support patterns). Consequently, they provided several regularization methods and generalizations of the original AdaBoost algorithm to achieve a soft margin. In particular, they suggested $\mathrm{LP}_{\mathrm{Reg}}$-AdaBoost (Rätsch et al., 2001), ν-LP and ν-ARC (Rätsch et al., 1999). These methods basically accommodate outliers by allowing unusual observations to fall on the wrong side of the prediction in subsequent iterations of AdaBoost. Other methods have been proposed to improve the performance of boosting algorithms with noisy data, which are based on modifying the observation weighting function (see e.g., Friedman et al., 2000). We can see in step 3(e) of Algorithm 1 that the observation weights $\mathrm{D}_{\mathrm{i}}^{(t)}$ are updated as functions of $\exp\left(-\alpha_t y_i h_t(\boldsymbol{x}_i)\right)$. Friedman (2001) showed that in order to minimize the error rate ε_t, the following adaptive formula is utilized in AdaBoost:

$$f_m(\boldsymbol{x}) = \frac{1}{2} ln\left(\frac{P[y=1|\boldsymbol{x}]}{P[y=-1|\boldsymbol{x}]}\right). \tag{5}$$

Using this formula to update the new weights can cause large fluctuations when noise is present; therefore most of the modifications to AdaBoost for adaptation to noisy environments have been based on the use of different weighting schemes to avoid assigning large weights to noisy observations. Among the methods that modify the weights is GentleBoost (Friedman et al., 2000), which uses the updating function:

$$f_m(\boldsymbol{x}) = \frac{1}{2} ln\left(P[y=1|\boldsymbol{x}] - P[y=-1|\boldsymbol{x}]\right). \tag{6}$$

LogitBoost (Friedman et al., 2000) uses maximum likelihood to minimize a logistic loss function, and MadaBoost (Domingo and Watanabe, 2000) uses a filtering framework to alleviate the outlier problems. Gradient boosting (Friedman, 2001) and stochastic gradient boosting (Friedman, 2002) also use modifications of the loss functions to provide robustness and approximation accuracy to the boosting algorithm. Servedio (2003) proposed SmoothBoost, which is an algorithm that also discourages the overweighting of noisy observations. Zhang and Zhang (2008) proposed a local boosting algorithm, which updates the observation weights $\mathrm{D}_{\mathrm{i}}^{(t)}$ in step 3(e) of Algorithm 1 according to a local error over the neighborhood $N(i)$ of the ith training instance. The local error for the local boosting algorithm is found to be more robust to noise. Other researchers (e.g. Lutz et al., 2008 and Freund, 2009) have also introduced robust variations of well-known boosting algorithms. This list of AdaBoost modifications is not exhaustive, but it represents the most common algorithms used to date. For a more in-depth treatment on variants of boosting algorithms, the interested reader is referred to Ferreira and Figueiredo (2012).

Long and Servedio (2010) proved that for any booster with a potential convex loss function, and any nonzero random classification noise rate, there is a data set, which can be efficiently learned by the booster if there is no noise, but cannot be learned with accuracy better than $1/2$ with random classification noise present. Boosters with potential convex loss functions include the most common boosting algorithms to date. Most of the methods that handle outliers and noisy data, including most soft margin algorithms and those with weighting modifications also use convex loss functions, and are susceptible to the shortcomings presented in Long and Servedio (2010). It was noted in Long and Servedio (2010) that algorithms such as Boost-by-Majority (Freund, 1995), BrownBoost (Freund, 2001), RobustBoost (Freund, 2009), and branching-based boosters do tolerate random classification noise fairly well because they do not rely on convex loss functions.

Fig. 1 illustrates a typical behavior of boosters with convex loss functions in the presence of noise. We ran 100 simulations injecting 10% noise to the TwoNorm data set (see Section 4 for descriptions of data sets) and averaged the test set error rates of AdaBoost, LogitBoost and GentleBoost for each round t. We also included RobustBoost in the simulations to compare the previous noise-tolerant methods to algorithms with non-convex loss functions. We can see that as the number of combined trees grows, the test error rates (an estimate of $R[H]$) of AdaBoost, LogitBoost, GentleBoost and RobustBoost tend to level off at levels higher than that of the noise injected, in this case 10%. This reinforces the idea presented in Kalai and Servedio (2005) stating that no boosting algorithm can achieve an error rate below the noise rate η. We also see that AdaBoost and all of the noise-resistant methods, including RobustBoost have similar error rates. In addition to that, the error curves show some slight deterioration after approximately 50 rounds, which we believe is indicative of noise overfitting. We should mention that although we have shown the error rates up to 300 weak learners in Fig. 1, the trends continue to be similar for larger numbers of trees.

4. Proposed methods

We propose methods that are based on peeling the classification noise instead of accommodating it. Peeling methods basically identify potential outliers, remove them and refit the learning algorithm without the noise. Outlier and noise

Please cite this article in press as: Martinez, W., Gray, J.B., Noise peeling methods to improve boosting algorithms. Computational Statistics and Data Analysis (2015), http://dx.doi.org/10.1016/j.csda.2015.06.010

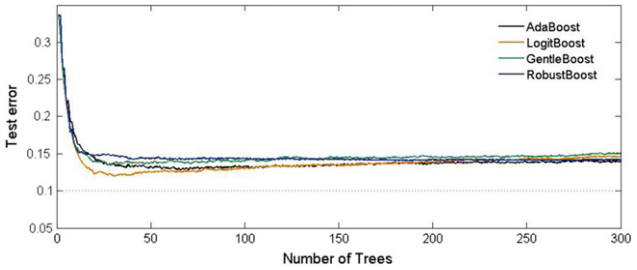

Fig. 1. Test error performance of AdaBoost, LogitBoost, GentleBoost, and RobustBoost for the TwoNorm data set with 10% noise level.

removal methods have been around for a long time. The term "peeling" has been used in the context of density depth. Specifically the idea of convex hull peeling (see e.g. Tukey, 1975; Barnett, 1976; Eddy, 1982) refers to identifying the points that fall in the vertices (extreme points) of a multivariate distribution and layer the points according to the distance from the centroid. Although the methods presented here do not exactly remove all the points on the vertices, we use the term noise peeling to refer to methods based on the idea of identifying extreme points and labeling all or part of them as possible noise. By peeling all or part of the noisy observations, we would potentially mitigate the original limitations of AdaBoost's performance in the presence of noise, while also avoiding the potential drawbacks of using noise-resistant boosters based on convex loss functions. To analyze the accuracy of the proposed methods, we have used clean data sets (see Section 4 for description) and injected the noise level η by flipping the response variable to the wrong response class according to the η rate and further assess the performance under the η noise level. To assess the accuracy of the peeling approaches proposed, we use the following data sets:

Glass Identification Data Set (Matlab Neural Network Toolbox): The Glass data set consists of $n = 214$ observations, and $p = 10$ predictor variables and $Y \in \{-1, +1\}$. The data set is found in the UCI Repository (Bache and Lichman, 2013). A decision tree was fit to the data, and the predictions were used as the response variable to create a "clean" data set.

Crab Data Set (Campbell and Mahon, 1974): The main goal here is to identify the sex $Y \in \{Male, Female\}$ of a crab from its physical measurements. Six characteristics ($p = 6$) of a crab are considered: species (blue, orange), frontal lobe size (mm), rear width (mm), carapace length (mm), carapace width (mm) and body depth (mm), along with $n = 200$ observations. A decision tree fit was also used as the response variable.

Breast Cancer (BC) Data Set (Mangasarian and Wolberg, 1990): The BC data set consists of $n = 699$ observations, and $p = 10$ predictor variables and $Y \in \{-1, +1\}$. The data set is found in the UCI Repository (Bache and Lichman, 2013). A decision tree fit was also used as the response variable.

Ionosphere Data Set (Sigillito et al., 1989): The Ionosphere data set consists of $n = 351$ observations, and $p = 34$ predictor variables and $Y \in \{-1, +1\}$. The data set is found in the UCI Repository (Bache and Lichman, 2013). A decision tree fit was also used as the response variable.

Sonar Data Set (Gorman and Sejnowski, 1988): The Sonar data set consists of $n = 208$ observations, and $p = 60$ predictor variables and $Y \in \{-1, +1\}$. The data set is found in the UCI Repository (Bache and Lichman, 2013).

TwoNorm Data Set (Breiman, 1996): TwoNorm is a synthetic data set consisting of $n = 300$ observations, $p = 20$ predictor variables and $Y \in \{-1, +1\}$. The theoretical expected test error rate (Bayes' minimum) is 2.3%.

ThreeNorm Data Set (Breiman, 1996): ThreeNorm is a synthetic data set consisting of $n = 300$ observations, $p = 20$ predictor variables and $Y \in \{-1, +1\}$.

4.1. Margin Peeling (MP)

The Margin Peeling (MP) approach is based on the idea that a small margin is indicative of poor predictability of the particular observation. This is because, as we can see in Algorithm 1, the term $\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})$ is a weighted average of the predictions of the individual classifiers composing the ensemble. As $0 \leq \alpha_t \leq 1$, a small margin, say close to -1, indicates the observation was misclassified in almost all rounds of boosting, which in turn makes it a candidate to be noise. We begin by obtaining an AdaBoost ensemble and peeling the observation(s) with margin(s) less than an arbitrary value m_L. An intuitive choice can be $m_L = 0$. We should note that m_L can also be chosen with cross-validation. After detecting the candidate observation(s), we remove them to refit AdaBoost to the remaining data. The approach is illustrated in Algorithm 2.

Algorithm 2. AdaBoost after MP	
FIT:	Algorithm 1 to the data
OBTAIN:	$m_i = y_i \sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})$ for each data point
PEEL:	$m_i < m_L$
REFIT:	Algorithm 1

We can see in Fig. 2 the performance of AdaBoost after Margin Peeling for the Glass data set with 5% noise and $m_L = 0$ using 100 simulations with a 60/40 training and validation scheme. We continue to use $m_T = 0$ in all further simulations.

Please cite this article in press as: Martinez, W., Gray, J.B., Noise peeling methods to improve boosting algorithms. Computational Statistics and Data Analysis (2015), http://dx.doi.org/10.1016/j.csda.2015.06.010

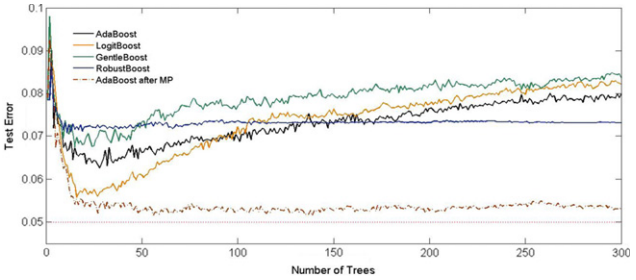

Fig. 2. Test error performance of AdaBoost after MP compared to AdaBoost, LogitBoost, GentleBoost, and RobustBoost for the Glass data set with 5% noise level.

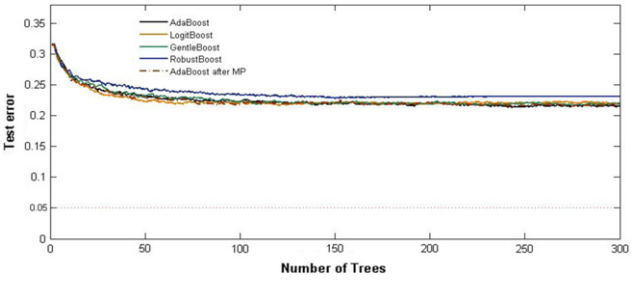

Fig. 3. Test error performance of AdaBoost after MP compared to AdaBoost, LogitBoost, GentleBoost, and RobustBoost for the Sonar data set with 5% noise level.

In this example, the AdaBoost after MP algorithm detected on average 80% of the noisy observations and achieved a false positive rate of 1.36%. Evident in Fig. 2 is that as the number of trees grows, the test error for the MP algorithm levels off at a lower rate than all the other methods. The RobustBoost method and the AdaBoost after MP algorithm do not show a deteriorating test error, which might be indicative of overfitting the noisy instances.

Fig. 3 illustrates the performance of AdaBoost after MP for the Sonar data set with 5% noise and $m_L = 0$ using 100 simulations. These simulations do not show an evident advantage of using the AdaBoost after MP algorithm as compared to the original AdaBoost and the other noise resistant boosters, however the proposed method does compare favorably to the best performing competitors. The poor performance is most likely the result of the AdaBoost after MP Algorithm only detecting on average 6.6% of the noisy observations. Using simulations under different settings we will further investigate the circumstances under which the proposed methods work the best.

4.2. Weighted Misclassification Peeling (WMP)

The Weighted Misclassification Peeling (WMP) approach uses the information on how often a particular observation has been misclassified by each of the weak learners (M_{it}) of the ensemble as an indication of the difficulty in correctly classifying the observation. A difficult-to-classify instance, in turn, suggests a noisy observation. However, because the design of boosting gives more weight to misclassified observations, some weak learners will be biased toward assigning more weight to difficult cases. To mitigate this issue, the percentage of observations correctly classified within the AdaBoost ensemble (r_i) is used as a measure of strength of each weak learner, and finally a weighted rate of misclassification of an observation i is computed as:

$$w_i = \sum_t \frac{r_t}{\sum_t r_t} M_{it}, \tag{7}$$

where $M_{it} = I(y_i \neq h_t(\mathbf{x_i}))$ and $r_t = \sum_i I(y_i = h_t(\mathbf{x_i}))/n$. High values of w_i indicate high level of misclassification, with the advantage that the weak learners of most accuracy in classification receive larger weights. Since the distribution of the w_i is unknown, we could set an arbitrary value of w_i, say w_L chosen through cross validation or intuitively, so that we can consider a point w_i an outlier if $w_i > w_L$. For the simulations presented here, we will use $w_L = 0.5$. The method is presented in Algorithm 3.

Algorithm 3. AdaBoost after WMP	
FIT:	Algorithm 1 to the data
CALCULATE:	w_i for each data point
PEEL:	$w_i > w_L$
REFIT:	Algorithm 1

Please cite this article in press as: Martinez, W., Gray, J.B., Noise peeling methods to improve boosting algorithms. Computational Statistics and Data Analysis (2015), http://dx.doi.org/10.1016/j.csda.2015.06.010

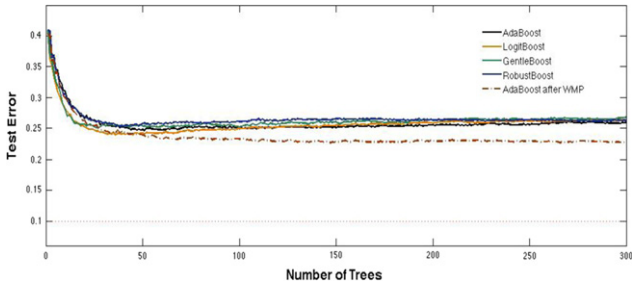

Fig. 4. Test error performance of AdaBoost after WMP compared to AdaBoost, LogitBoost, GentleBoost, and RobustBoost for the ThreeNorm data set with 10% noise level.

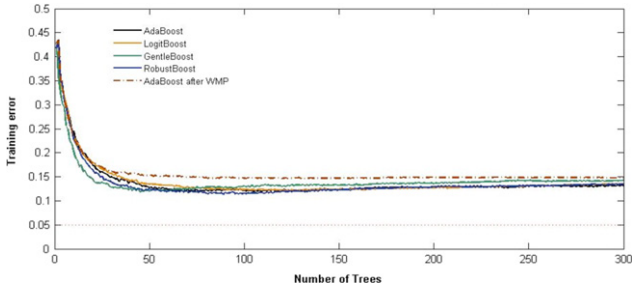

Fig. 5. Test error performance of AdaBoost after WMP compared to AdaBoost, LogitBoost, GentleBoost, and RobustBoost for the Crab data set with 5% noise level.

Fig. 4 illustrates the performance of AdaBoost after WMP for the ThreeNorm data set with 10% noise and $w_L = 0.5$ using 100 simulations. In this example, the AdaBoost after WMP algorithm detected approximately 43.33% of the noisy observations and achieved a false positive rate of 1.76%. This method also performs better than the original AdaBoost or the noise-resistant boosters analyzed here. Note that the performance of the proposed methods could be further improved if a traditional noise-resistant booster is fit after the peeling method, but we have chosen not to do that in order to have a fair comparison.

In Fig. 5, the performance of the WMP algorithm is illustrated with a negative performance for the Crab data set (5% noise and $w_L = 0.5$ using 100 simulations). In this example, the AdaBoost after WMP algorithm detected approximately 42% of the noisy observations and achieved a false positive rate of approximately 3%. For this level of noise, the AdaBoost after WMP algorithm performs slightly worse than the rest of the competitor methods.

4.3. Data Weight Peeling (DWP)

The Data Weight Peeling (DWP) approach is similar to the margin peeling approach, but it is based on the idea that a large data weight $\mathrm{D}_{\mathrm{i}}^{(t)}$ in step 3(e) of Algorithm 1 is also indicative of poor predictability of the particular observation. Boosting assigns large weights to misclassified observations. For every boosting round t we have a distribution $\mathrm{D}_{\mathrm{i}}^{(t)}$. In order to identify the potential noise, we average the weights $\mathrm{D}_{\mathrm{i}}^{(t)}$ assigned to each observation, obtain equation

$$\bar{D}_{\mathrm{i}} = \sum_{t=1}^{T} \mathrm{D}_{\mathrm{i}}^{(t)}/T, \tag{8}$$

and flag a potential outlier by selecting those observations with $\bar{D}_{\mathrm{i}} > \bar{D} + t_{1-\gamma} s_{\bar{D}}$, where

$$\bar{D}_{\mathrm{i}} = \sum_{i=1}^{N} \sum_{t=1}^{T} \mathrm{D}_{\mathrm{i}}^{(t)}/TN, \tag{9}$$

$t_{1-\gamma}$ is the upper $1-\alpha$ quantile of the t-distribution, and

$$s_{\bar{D}} = \frac{1}{\sqrt{T}} \sqrt{\sum_{i=1}^{N} \sum_{t=1}^{T} \left(\mathrm{D}_{\mathrm{i}}^{(t)} - \bar{D}\right) / (TN-1)}, \tag{10}$$

is the standard error of the $\mathrm{D}_{\mathrm{i}}^{(t)}$. The approach is presented in Algorithm 4.

Please cite this article in press as: Martinez, W., Gray, J.B., Noise peeling methods to improve boosting algorithms. Computational Statistics and Data Analysis (2015), http://dx.doi.org/10.1016/j.csda.2015.06.010

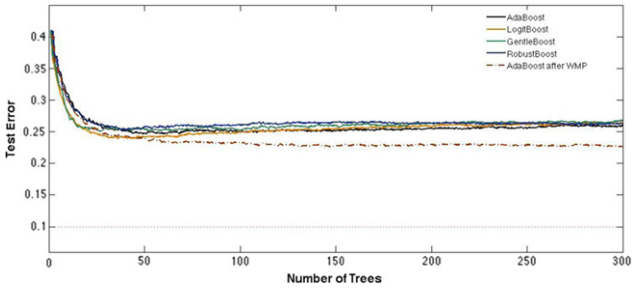

Fig. 6. Test error performance of AdaBoost after DWP compared to AdaBoost, LogitBoost, GentleBoost, and RobustBoost for the Crab data set with 20% noise level.

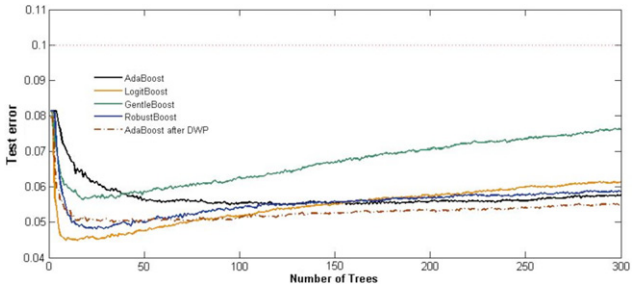

Fig. 7. Test error performance of AdaBoost after DWP compared to AdaBoost, LogitBoost, GentleBoost, and RobustBoost for the Breast Cancer data set with 10% noise level.

Algorithm 4. AdaBoost after Data Weight Peeling	
FIT:	Algorithm 1 to the data
OBTAIN	$D_n^{(t+1)} = \frac{D_n^{(t)} exp(-\alpha_t y_n h_t(\mathbf{x_n}))}{Z_t}$ for all rounds t
CALCULATE:	$\bar{D}_t$, $\bar{D}$, $s_{\bar{D}}$
PEEL:	Observations with $\bar{D}_t > \bar{D} + t_{1-\gamma} s_{\bar{D}}$ for low values of γ.
REFIT:	Algorithm 1

Fig. 6 illustrates the performance of AdaBoost after DWP using 100 simulations with $\gamma = 0.02$ for the Crab data set with 20% noise. In this example the AdaBoost after DWP algorithm detected approximately 22.21% of the noisy observations and achieved a false positive rate of 0.38%. Evident in Fig. 6 is that as the number of trees grows, the test error for AdaBoost after DWP algorithm stabilizes at a rate slightly lower than those of other noise resistant boosters, however for this particular data set the detection of the DWP algorithm was not very high, which in turn possibly led to overfitting the noisy instances. In Fig. 7 the performance of the WMP algorithm is compared to the other noise resistant boosters. This simulation shows that although the WMP algorithm outperforms the competitor methods in the long run, it does not achieve the lowest possible test error rate obtained by the LogitBoost Algorithm. For this particular data set and noise setting, all the methods studied obtain test error rates lower than the noise level injected η, contradicting the results in Kalai and Servedio (2005) in finite number of rounds T.

4.4. Linear Program Peeling (LPP)

The approach here is to use a linear program (LP) to maximize the second smallest margin while allowing the smallest one to be unconstrained. We do this by maximizing a slack variable m such that, we can allow 1 observation margin to be less than m, and the remaining $n - 1$ observations to be equal to or larger than m. The observation with an arbitrary margin less than m is the outlier candidate. After obtaining the LP solution, we have two options. The first one uses the final model found via Algorithm 5 shown below, with the corresponding α_t's obtained through the LP solution. The second option could entail using the regular AdaBoost after the candidate outlier(s) are removed. This method requires as an input an estimate of the noise level in the data set. Estimating the level of noise is challenging in any data set. A method that could be used to obtain the noise level estimate is the support vector data description (SVDD) (Tax and Duin, 1999, 2004) algorithm. SVDD is a non-parametric unsupervised application of support vector machines to find the boundary with smallest volume around the data that can also be used to detect noise (Park et al., 2007). We have used the noise level injected as our estimated noise level ζ to avoid clouding the assessment of the presented method. Consequently we understand that the performance of the LPP method would be a best-case scenario. The LPP method is shown in Algorithm 5.

Please cite this article in press as: Martinez, W., Gray, J.B., Noise peeling methods to improve boosting algorithms. Computational Statistics and Data Analysis (2015), http://dx.doi.org/10.1016/j.csda.2015.06.010

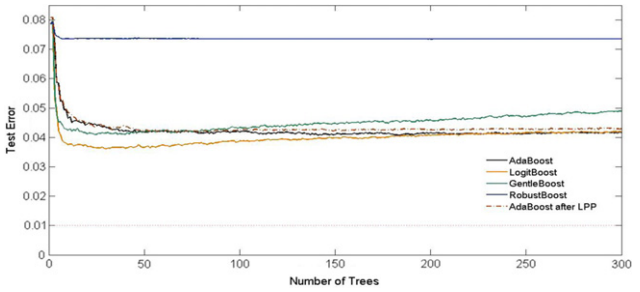

Fig. 8. Test error performance of AdaBoost after LPP compared to AdaBoost, LogitBoost, GentleBoost, and RobustBoost for the Breast Cancer data set with 1% noise level.

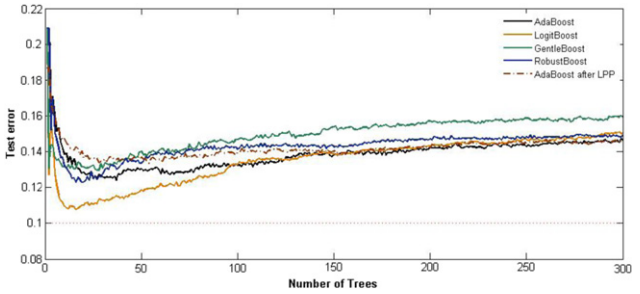

Fig. 9. Test error performance of AdaBoost after LPP compared to AdaBoost, LogitBoost, GentleBoost, and RobustBoost for the Ionosphere data set with 10% noise level.

Algorithm 5. LP Peeling	
FIT:	Algorithm 1 to the data
INPUT:	Estimate of noise level ζ
RUN LP:	max m
	s.t. $y_i \sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x}) < m$ for 1 observation
	$y_i \sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x}) \geq m$ for $n-1$ observations
	$\sum_{t=1}^{T} \alpha_t = 1\, \alpha_t \geq 0,\ t = 1, 2, \ldots, T\ -1 \leq m \leq 1$
REPEAT:	LP for $\lceil \zeta n \rceil$ times.
PEEL:	Observations with $y_i \sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x}) < m$ in each LP round
REFIT:	Algorithm 1

Fig. 8 illustrates the performance of AdaBoost after LPP Peeling using 100 simulations on the BC data set with 1% noise. In this example the AdaBoost after LPP algorithm detected approximately 17.4% of the noisy observations, which is a low detection rate compared to the other methods presented here. The AdaBoost after LPP method achieved a false positive rate of approximately 0.99%. Fig. 8 shows that as the number of trees grows, the test error for the AdaBoost after LPP algorithm stabilizes at a very similar rate to that of noise resistant boosters, but better than the RobustBoost algorithm. This performance is very typical for low noise rates, which might suggest that noise resistant algorithms should be used under higher noise settings. Although the performance of AdaBoost after LPP is slightly worse than the other methods presented here, the method is still worth investigating further using different data sets and under different levels of noise.

In Fig. 9 we can see that the performance of the AdaBoost after LPP algorithm is favorably compared to the other noise resistant boosters as more trees are ensembled, but the LogitBoost algorithm outperforms the proposed method in achieving the lowest possible test error rate.

4.5. Majority Vote Peeling (MVP)

The Majority Vote Peeling (MVP) approach takes into account the fact that an observation might be an unusual point, if most of the weak learners misclassify it. The MVP algorithm therefore flags a noisy observation only if the observation has been misclassified by the majority of the weak learners of the ensemble. In other words, we let $\mathbf{H} \in \{-1, +1\}^{n \times T}$ with elements $h_{it} = \pm 1$ defined as the prediction of the tth weak learner for the ith observation in the training data. Furthermore, we define the matrix $\boldsymbol{HoY} \in \{-1, +1\}^{n \times T}$ with elements $h_{it} \circ y_i$, defined as the element-wise (Hadamard

Please cite this article in press as: Martinez, W., Gray, J.B., Noise peeling methods to improve boosting algorithms. Computational Statistics and Data Analysis (2015), http://dx.doi.org/10.1016/j.csda.2015.06.010

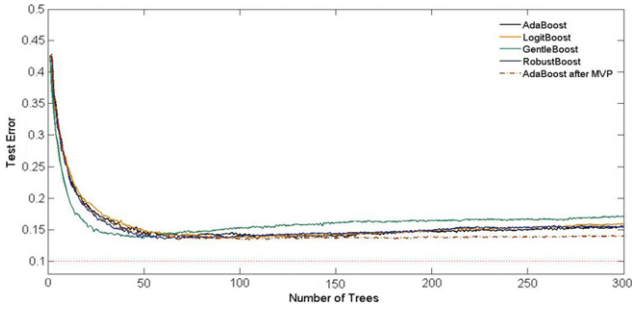

Fig. 10. Test error performance of AdaBoost after MVP compared to AdaBoost, LogitBoost, GentleBoost, and RobustBoost for the Crab data set with 10% noise level.

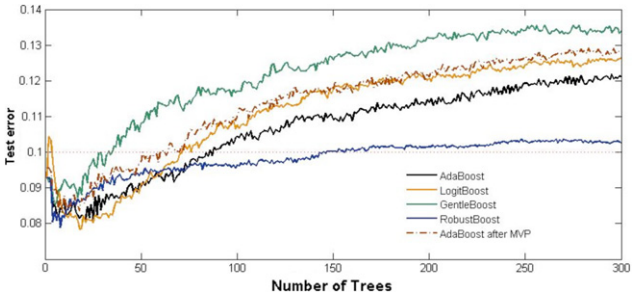

Fig. 11. Test error performance of AdaBoost after MVP compared to AdaBoost, LogitBoost, GentleBoost, and RobustBoost for the Glass data set with 10% noise level.

product) multiplication of h_{it} by y_i. The majority vote of an observation is then defined as $MV_i = sign\left(\sum_t h_{it} \circ y_i\right)$. This is a simple method that does not require setting any parameters.

Algorithm 6. AdaBoost after MVP	
FIT:	Algorithm 1 to the data
CALCULATE:	$MV_i = sign\left(\sum_t h_{it} \circ y_i\right)$
PEEL:	Observations where $MV_i \neq y_i$
REFIT:	Algorithm 1

Figs. 10 and 11 illustrate the performance of the AdaBoost after MVP method with the Crab and Glass data sets under 10% noise respectively. The MVP method detected 63.58% of the noise with a false positive rate of 3.84% for the Crab data set, while detecting on average 20% of the noisy observations for the Glass data set, with a false positive rate of approximately 3.95%. This peeling method performs well in comparison to the other noise resistant methods for the Crab data set, however it performs poorly compared to the other noise-resistant boosters for the Glass data set. This suggests that the performance of the peeling methods might depend not only on the settings and noise level, but also on the structure of the particular data set.

5. Simulations

We observed some initial results of the methods presented in the previous section. To further assess how the methods would perform under different data sets, noise levels and general conditions, we perform several simulations on some synthetic and real data sets. In order to generate noise levels, we need to be able to inject noise to otherwise clean data sets. Therefore, for the real data sets (Glass, Crab, Breast Cancer, Sonar and Ionosphere) data sets, we fit a decision tree to the data, and use the predictions as the response variable Y. We do this in order to eliminate any inherent noisy observations already included in the data set. Afterwards, we inject the desired level of noise by using the random classification noise model described in (4). To avoid random variation in the noise level injected, we have set fixed noise levels η of 1%, 5%, 10% and 20% by flipping the response variable to the wrong output according to η. The noise injection is applied with the same strength to both classes, that is, the chances of flipping the response variable from -1 to $+1$ are the same as the chances of flipping from $+1$ to -1. We also use two synthetic data sets TwoNorm and ThreeNorm (described in Section 4). The data sets were chosen with the purpose of providing different scenarios to test the proposed methods by varying the complexity of the data sets in terms of the number of predictors, as well as the sample sizes.

In the simulations, we want to understand how the different methods proposed perform under different levels of noise. We also compare the proposed methods with the original AdaBoost along with the LogitBoost, GentleBoost and RobustBoost algorithms. All simulations have been performed 100 times with $T = 300$, and the false positive rates of the proposed

Please cite this article in press as: Martinez, W., Gray, J.B., Noise peeling methods to improve boosting algorithms. Computational Statistics and Data Analysis (2015), http://dx.doi.org/10.1016/j.csda.2015.06.010

Table 1
Method comparison for the glass data set.

	0% noise	1% noise	5% noise	10% noise	20% noise
Test set error rate					
AdaBoost	3.07%	4.71%	7.43%	11.02%	19.48%
LogitBoost	2.62%	4.57%	7.53%	11.58%	20.92%
GentleBoost	2.95%	4.86%	7.78%	11.82%	22.35%
RobustBoost	6.76%	6.88%	6.91%	9.51%	20.11%
AdaBoost after MP	3.07%	3.67%	4.97%	6.25%	11.04%
AdaBoost after WMP	3.84%	4.77%	6.55%	7.72%	12.06%
AdaBoost after DWP	5.55%	4.53%	4.67%	7.53%	15.39%
AdaBoost after LPP	3.07%	5.05%	8.50%	11.98%	18.28%
AdaBoost after MVP	3.10%	4.86%	8.08%	11.92%	19.64%
Fraction of noise detected					
AdaBoost after MP	N/A	68.00%	80.00%	79.92%	75.77%
AdaBoost after WMP	N/A	50.00%	61.71%	64.23%	59.23%
AdaBoost after DWP	N/A	85.50%	74.57%	55.85%	26.65%
AdaBoost after LPP	N/A	6.50%	13.29%	24.23%	42.85%
AdaBoost after MVP	N/A	2.00%	9.71%	20.69%	29.69%
False positive rate					
AdaBoost after MP	0.00%	0.34%	1.36%	2.28%	2.98%
AdaBoost after WMP	1.43%	1.13%	2.77%	3.30%	3.09%
AdaBoost after DWP	3.88%	1.63%	0.88%	0.59%	0.26%
AdaBoost after LPP	N/A	1.46%	4.74%	7.70%	11.61%
AdaBoost after MVP	0.13%	0.21%	2.09%	4.20%	6.20%

Table 2
Method comparison for the Crab data set.

	0% noise	1% noise	5% noise	10% noise	20% noise
Test set error rate					
AdaBoost	9.11%	9.57%	12.48%	15.50%	21.75%
LogitBoost	9.29%	9.70%	12.17%	16.05%	23.27%
GentleBoost	9.29%	9.95%	13.29%	17.36%	25.53%
RobustBoost	8.96%	9.63%	12.35%	16.01%	23.84%
AdaBoost after MP	12.15%	12.14%	14.53%	15.68%	21.58%
AdaBoost after WMP	15.86%	10.75%	13.32%	15.74%	21.73%
AdaBoost after DWP	11.33%	10.32%	11.87%	13.46%	20.66%
AdaBoost after LPP	9.11%	10.18%	13.36%	15.92%	20.80%
AdaBoost after MVP	9.46%	10.61%	12.53%	14.98%	19.87%
Fraction of noise detected					
AdaBoost after MP	N/A	61.00%	58.83%	62.83%	56.00%
AdaBoost after WMP	N/A	44.00%	41.67%	47.17%	40.42%
AdaBoost after DWP	N/A	69.00%	59.17%	42.50%	22.21%
AdaBoost after LPP	N/A	16.00%	32.83%	44.17%	51.38%
AdaBoost after MVP	N/A	61.00%	58.83%	63.58%	54.42%
False positive rate					
AdaBoost after MP	3.21%	3.22%	4.02%	4.52%	6.63%
AdaBoost after WMP	8.65%	2.17%	2.97%	3.71%	4.68%
AdaBoost after DWP	3.17%	2.27%	1.48%	0.62%	0.38%
AdaBoost after LPP	N/A	1.40%	3.36%	5.58%	9.73%
AdaBoost after MVP	0.38%	2.59%	3.27%	3.84%	4.82%

methods that involve peeling and refitting AdaBoost have been presented here. With the fraction of noise detected λ and the number of noisy instances κ, the false negative rate can be calculated as $\kappa\ (1-\lambda)\ /n$, so we have chosen not to include it in the simulations. Each simulation partitions the data into 60/40 training and validation data sets with the noise being injected to the training data set. Table 1 illustrates the results for the Glass data set. We can see that even for low levels of noise the best performing methods are the AdaBoost after MP, WMP and DWP methods. The peeling approaches appear to be preferable to the traditional noise boosters for this particular data set. We have also included a 0% noise level η to level the performance of the proposed methods under noise-free data sets.

Table 2 illustrates the results for the Crab data set. We can also see that the peeling methods perform well under moderate to high levels of noise, however, as expected the proposed methods do not perform well under 0% noise or very low levels of noise, such as 1%.

Please cite this article in press as: Martinez, W., Gray, J.B., Noise peeling methods to improve boosting algorithms. Computational Statistics and Data Analysis (2015), http://dx.doi.org/10.1016/j.csda.2015.06.010

Table 3
Method comparison for the Breast Cancer data set.

	0% noise	1% noise	5% noise	10% noise	20% noise
Test set error rate					
AdaBoost	3.57%	4.24%	4.93%	6.03%	8.89%
LogitBoost	3.65%	4.21%	5.28%	6.23%	8.93%
GentleBoost	3.75%	4.99%	6.23%	7.71%	11.11%
RobustBoost	6.01%	7.57%	6.54%	6.03%	8.90%
AdaBoost after MP	3.47%	3.67%	4.28%	4.54%	5.99%
AdaBoost after WMP	3.55%	3.82%	4.15%	4.55%	6.27%
AdaBoost after DWP	3.46%	3.68%	4.29%	5.48%	8.90%
AdaBoost after LPP	3.57%	4.20%	5.07%	6.11%	7.40%
AdaBoost after MVP	3.62%	5.16%	8.40%	12.95%	18.33%
Fraction of noise detected					
AdaBoost after MP	N/A	92.80%	93.43%	92.71%	90.51%
AdaBoost after WMP	N/A	89.20%	89.95%	90.02%	86.30%
AdaBoost after DWP	N/A	89.40%	79.71%	68.43%	40.88%
AdaBoost after LPP	N/A	17.40%	36.81%	56.02%	72.07%
AdaBoost after MVP	N/A	25.40%	36.81%	40.48%	39.32%
False positive rate					
AdaBoost after MP	2.33%	2.95%	3.63%	3.85%	4.08%
AdaBoost after WMP	3.39%	3.19%	3.68%	3.60%	3.88%
AdaBoost after DWP	3.18%	1.40%	0.57%	0.36%	0.18%
AdaBoost after LPP	N/A	0.99%	3.17%	4.41%	5.60%
AdaBoost after MVP	0.82%	4.24%	8.32%	11.29%	11.88%

Table 4
Method comparison for the ionosphere data set.

	0% noise	1% noise	5% noise	10% noise	20% noise
Test set error rate					
AdaBoost	7.98%	9.25%	11.75%	14.60%	22.39%
LogitBoost	8.08%	9.22%	11.73%	14.18%	22.70%
GentleBoost	8.21%	9.38%	12.19%	15.57%	24.59%
RobustBoost	8.96%	9.93%	11.28%	15.05%	23.51%
AdaBoost after MP	7.97%	8.90%	9.46%	10.32%	14.86%
AdaBoost after WMP	8.54%	9.05%	10.89%	12.57%	18.81%
AdaBoost after DWP	8.28%	9.09%	10.12%	12.48%	20.63%
AdaBoost after LPP	7.98%	9.24%	11.85%	14.12%	19.51%
AdaBoost after MVP	7.82%	9.40%	11.44%	14.65%	22.45%
Fraction of noise detected					
AdaBoost after MP	N/A	66.67%	69.09%	69.09%	62.63%
AdaBoost after WMP	N/A	49.00%	47.91%	45.64%	33.95%
AdaBoost after DWP	N/A	63.33%	51.64%	36.55%	18.70%
AdaBoost after LPP	N/A	5.00%	16.00%	29.50%	44.02%
AdaBoost after MVP	N/A	2.33%	3.09%	7.00%	10.60%
False positive rate					
AdaBoost after MP	2.96%	3.12%	3.54%	4.17%	4.99%
AdaBoost after WMP	7.13%	3.44%	3.99%	4.08%	4.19%
AdaBoost after DWP	4.45%	3.59%	1.97%	1.04%	0.67%
AdaBoost after LPP	N/A	1.35%	4.38%	7.35%	11.41%
AdaBoost after MVP	0.27%	0.22%	0.53%	1.22%	1.92%

Table 3 illustrates the results for the Breast Cancer data set. We can also see that most of the peeling methods perform well under all levels of noise. Nevertheless, the AdaBoost after MVP method shows a high false positive rate and an inferior performance compared to the other methods proposed and noise-resistant boosters.

Table 4 illustrates the results for the Ionosphere data set. The best performing methods are the AdaBoost after MP, WMP, DWP and LPP under most levels of noise. The AdaBoost after LPP has a high false positive rate, but compares favorably to most noise-resistant boosters under high levels of noise.

Table 5 illustrates the results for the Sonar data set. In this data set there is not a marked difference in performance of the peeling methods compared to the other methods, except for high levels of noise. This might be the result of the complexity of the data set.

Table 6 illustrates the results for the TwoNorm data set. The AdaBoost after MP, WMP, DWP peeling methods perform the best under all levels of noise.

Please cite this article in press as: Martinez, W., Gray, J.B., Noise peeling methods to improve boosting algorithms. Computational Statistics and Data Analysis (2015), http://dx.doi.org/10.1016/j.csda.2015.06.010

Table 5
Method comparison for the sonar data set.

	0% noise	1% noise	5% noise	10% noise	20% noise
Test set error rate					
AdaBoost	18.94%	20.59%	23.51%	26.15%	31.97%
LogitBoost	19.73%	21.42%	23.96%	26.59%	32.92%
GentleBoost	19.73%	20.81%	23.73%	26.46%	32.64%
RobustBoost	21.36%	22.46%	23.83%	26.14%	33.06%
AdaBoost after MP	18.97%	20.65%	23.28%	25.64%	31.04%
AdaBoost after WMP	20.02%	20.60%	22.21%	24.60%	29.63%
AdaBoost after DWP	20.07%	20.67%	22.74%	24.68%	29.93%
AdaBoost after LPP	18.94%	20.96%	23.76%	25.14%	30.73%
AdaBoost after MVP	18.96%	20.76%	23.59%	26.01%	31.89%
Fraction of noise detected					
AdaBoost after MP	N/A	3.50%	6.57%	8.62%	9.52%
AdaBoost after WMP	N/A	22.50%	25.86%	24.23%	21.36%
AdaBoost after DWP	N/A	34.50%	27.57%	20.69%	10.28%
AdaBoost after LPP	N/A	4.00%	13.14%	20.54%	31.80%
AdaBoost after MVP	N/A	0.00%	0.24%	0.31%	0.24%
False positive rate					
AdaBoost after MP	0.00%	0.07%	0.24%	0.62%	1.50%
AdaBoost after WMP	2.23%	2.51%	2.34%	3.41%	4.09%
AdaBoost after DWP	3.21%	2.66%	1.78%	1.26%	0.91%
AdaBoost after LPP	N/A	1.54%	4.86%	8.26%	13.64%
AdaBoost after MVP	0.00%	0.14%	0.10%	0.08%	0.07%

Table 6
Method comparison for the TwoNorm data set.

	0% noise	1% noise	5% noise	10% noise	20% noise
Test set error rate					
AdaBoost	6.63%	7.43%	10.79%	14.42%	21.39%
LogitBoost	6.91%	7.50%	11.14%	14.96%	21.26%
GentleBoost	6.85%	7.41%	11.34%	15.51%	23.05%
RobustBoost	15.04%	14.43%	11.64%	14.24%	22.48%
AdaBoost after MP	6.63%	7.40%	9.03%	9.73%	15.58%
AdaBoost after WMP	6.58%	7.11%	8.35%	10.09%	14.43%
AdaBoost after DWP	7.12%	6.79%	7.46%	10.36%	18.04%
AdaBoost after LPP	6.63%	7.53%	10.44%	12.99%	18.16%
AdaBoost after MVP	6.62%	7.44%	10.77%	14.04%	20.40%
Fraction of noise detected					
AdaBoost after MP	N/A	9.50%	33.78%	46.78%	47.89%
AdaBoost after WMP	N/A	27.50%	44.89%	42.44%	50.36%
AdaBoost after DWP	N/A	68.00%	58.67%	28.22%	21.81%
AdaBoost after LPP	N/A	5.00%	16.67%	2.61%	46.31%
AdaBoost after MVP	N/A	0.50%	0.78%	1.54%	8.39%
False positive rate					
AdaBoost after MP	0.00%	0.02%	0.49%	1.54%	3.36%
AdaBoost after WMP	0.59%	0.29%	1.19%	2.42%	4.03%
AdaBoost after DWP	2.77%	2.09%	1.12%	0.70%	0.49%
AdaBoost after LPP	N/A	1.06%	4.17%	7.18%	10.74%
AdaBoost after MVP	0.00%	0.02%	0.02%	0.16%	0.86%

Table 7 illustrates the results for the ThreeNorm data set. As in the Sonar data set, there is no clear-cut winning method under low levels of noise, but the peeling methods perform better under moderate to high levels of noise compared to AdaBoost and the other noise-resistant boosters.

The performance of the proposed methods suggests that in most cases noise peeling improves the results of AdaBoost and outperforms the noise-resistant boosters. This is more markedly so in high-noise settings. Under low noise, e.g. 1%, the original AdaBoost algorithm has been shown to perform well (Dietterich, 2000), in most cases outperforming the peeling methods and other noise resistant boosters. The advantages of noise peeling might outweigh the computational burdens especially in high-noise settings, as in some cases the test error rate is almost halved. We have omitted measuring computational costs, as this is beyond the scope of this research. Nevertheless, not all of the methods proposed perform similarly. The AdaBoost after MP method appears to perform the best under data sets with a lower complexity structure (informally defining a low complex data set as a data set with relatively low ratio of variables to number of observations), such as the

Please cite this article in press as: Martinez, W., Gray, J.B., Noise peeling methods to improve boosting algorithms. Computational Statistics and Data Analysis (2015), http://dx.doi.org/10.1016/j.csda.2015.06.010

Table 7
Method comparison for the ThreeNorm data set.

	0% noise	1% noise	5% noise	10% noise	20% noise
Test set error rate					
AdaBoost	20.22%	21.30%	23.12%	25.97%	31.05%
LogitBoost	20.25%	21.31%	23.64%	26.71%	31.69%
GentleBoost	20.30%	21.48%	24.04%	26.63%	32.35%
RobustBoost	20.82%	21.63%	23.98%	26.64%	31.65%
AdaBoost after MP	20.21%	20.47%	21.96%	23.32%	27.71%
AdaBoost after WMP	20.34%	20.87%	21.72%	23.59%	28.16%
AdaBoost after DWP	19.62%	19.99%	21.76%	24.24%	29.57%
AdaBoost after LPP	20.22%	21.45%	23.30%	25.94%	29.08%
AdaBoost after MVP	19.96%	20.70%	22.77%	24.51%	30.34%
Fraction of noise detected					
AdaBoost after MP	N/A	38.00%	40.78%	42.78%	37.25%
AdaBoost after WMP	N/A	41.50%	43.33%	43.33%	36.69%
AdaBoost after DWP	N/A	36.00%	30.00%	23.33%	12.81%
AdaBoost after LPP	N/A	0.50%	12.89%	23.67%	36.42%
AdaBoost after MVP	N/A	13.50%	9.89%	10.89%	10.22%
False positive rate					
AdaBoost after MP	5.09%	5.06%	6.02%	6.94%	7.89%
AdaBoost after WMP	7.29%	6.63%	7.19%	7.59%	7.95%
AdaBoost after DWP	3.68%	3.36%	2.46%	1.76%	1.38%
AdaBoost after LPP	N/A	1.11%	4.36%	7.63%	12.72%
AdaBoost after MVP	0.37%	0.34%	0.83%	1.24%	1.94%

Glass, Breast Cancer, Ionosphere data sets, while doing relatively well on more complex data sets such as ThreeNorm and Sonar. The AdaBoost after DWP and WMP methods are also strong performers in most scenarios, but are particularly strong in highly complex data sets such as the ThreeNorm and Sonar. The AdaBoost after MVP and LPP methods appear not to offer a decided advantage over the original AdaBoost or other noise-resistant boosters in most scenarios investigated. Additionally, the need to obtain a reliable estimated noise level ζ for the LPP algorithm discourages its use unless a clear knowledge of the estimated noise is obtained in advance. In a generic sense, the AdaBoost after MP method offers the most robust performance across all data sets and noise levels investigated. However, for high complexity problems e.g. large p, few n, the AdaBoost after DWP and WMP could offer even better results. It should also be noted that deleting instances from the training data as suggested by the proposed noise peeling methods may lead to class imbalance issues, especially on a rare event classification scenario and this, in turn, could hinder the benefits of using the proposed noise peeling methods. Further research also needs to address the performance of the peeling methods in cases where $p > n$ or $p \gg n$, as is often the case in gene expression data. The proposed methods could be used in a multi-class setting (the reader is referred to Freund and Schapire, 1997 for a description of multi-class versions of AdaBooost and the respective definitions of margins, data weights and other necessary information), but their performance under those circumstances also needs to be investigated.

6. Conclusions

The intolerance of boosting methods to classification noise and outliers has led researchers to create various methods to accommodate outliers, and make boosting less susceptible to overfitting them. Long and Servedio (2010) proved that for any booster with a potential convex loss function, and any nonzero random classification noise rate, there is a data set that cannot be learned with accuracy better than 1/2 with the random classification noise present. Most of the current methods that accommodate outliers use convex loss functions. We took a different approach. Instead of accommodating the outliers, we developed methods that identified them and further peeled (deleted) them before refitting the boosting algorithm. This approach would not totally solve the problem posed by Long and Servedio (2010), since not all of the outliers are identified before refitting AdaBoost, but in our simulations we confirm that AdaBoost performs well in low noise settings (Dietterich, 2000), therefore the peeling approach represents a step forward, improving in most data sets and noise settings the performance of AdaBoost and other existing methods. It is worth noting that the results show that peeling methods perform generally better than AdaBoost and the noise-resistant boosters, especially when high levels of noise are present in the data. We should also note that the performance of the proposed methods could be further improved if a traditional noise-resistant method is fit after the peeling method under high levels of noise. With low levels of noise, most of the peeling methods still perform favorably when compared to most current methods.

Acknowledgments

The authors would like to thank the associate editor and two anonymous referees for providing comments that helped substantially improve this article.

Please cite this article in press as: Martinez, W., Gray, J.B., Noise peeling methods to improve boosting algorithms. Computational Statistics and Data Analysis (2015), http://dx.doi.org/10.1016/j.csda.2015.06.010

References

Angluin, D., Laird, P., 1988. Learning from noisy examples. Mach. Learn. 2, 343–370.
Bache, K., Lichman, M., 2013. UCI Machine Learning Repository. University of California, School of Information and Computer Science. [http://archive.ics.uci.edu/ml].
Barnett, V., 1976. The ordering of multivariate data. J. R. Stat. Soc. 139, 318–354.
Breiman, L., 1996. Bagging predictors. Mach. Learn. 26, 123–140.
Breiman, L., 2000. Some Infinity Theory for Predictor Ensembles, Technical Report 577. Statistics Department, University of California, Available at www.stat.berkeley.edu.
Campbell, N.A., Mahon, R.J., 1974. A multivariate study of two species of rock crab of genus leptograpsus. Aust. J. Zool. 22, 417–425.
Cortes, C., Vapnik, V., 1995. Support vector networks. Mach. Learn. 20, 273–297.
Dietterich, T.G., 2000. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. Mach. Learn. 40 (2), 139–158.
Domingo, C., Watanabe, O., 2000. MadaBoost: a modification of AdaBoost, In: 13th Annual Conference on Computational Learning Theory, pp. 180–189.
Eddy, W., 1982. Convex hull peeling. In: COMPSTAT 1982 5th Symposium, pp. 42–47.
Ferreira, A.J., Figueiredo, M.A., 2012. Boosting algorithms: A review of methods, theory, and applications. Ensemble Mach. Learn. 35–85.
Freund, Y., 1995. Boosting a weak learning algorithm by majority. Inform. Comput. 121 (2), 256–285.
Freund, Y., 2001. An adaptive version of the boost by majority algorithm. Mach. Learn. 43 (3), 293–318.
Freund, Y., Schapire, R.E., 1997. A decision–theoretic generalization of on-line learning and an application to boosting. J. Comput. System Sci. 55, 119–139.
Freund, Y., 2009. A more robust boosting algorithm, ArXiv Preprint. arXiv:0905.2138.
Friedman, J., 2001. Greedy function approximation. A gradient boosting machine. Ann. Statist. 29 (5), 1189–1232.
Friedman, J., 2002. Stochastic gradient boosting. Comput. Statist. Data Anal. 38 (4), 367–378.
Friedman, J., Hastie, T., Tibshirani, R., 2000. Additive logistic regression: A statistical view of boosting. Ann. Statist. 28 (2), 337–407.
Gamberger, D., Lavrac, N., Dzeroski, S., 2000. Noise detection and elimination in data preprocessing: Experiments in medical domains. Appl. Artif. Intell. 14 (2).
Gorman, R.P., Sejnowski, T.J., 1988. Analysis of hidden units in a layered network trained to classify sonar targets. Neural Netw. 1, 75–89.
Grove, A.J., Schuurmans, D., 1998. Boosting in the limit: Maximizing the margin of learned ensembles. In: Proceedings of the Fifteenth National Conference on Artificial Intelligence.
Kalai, A., Servedio, R., 2005. Boosting in the presence of noise. J. Comput. System Sci. 71 (3), 266–290.
Long, P., Servedio, R., 2010. Random classification noise defeats all convex potential boosters. Mach. Learn. 78 (3), 608–615.
Lutz, R., Kalisch, M., Buhlmann, P., 2008. Robustified L_2 boosting. Comput. Statist. Data Anal. 52 (7), 3331–3341.
Mangasarian, O.L., Wolberg, W.H., 1990. Cancer diagnosis via Linear programming. SIAM News 23 (5), 1–18.
Mason, L., Bartlett, P., Baxter, J., 2000. Improved generalization through explicit optimization of margins. Mach. Learn. 38, 243–255.
Opitz, D., Maclin, R., 1999. Popular ensemble methods: An empirical study. J. Artificial Intelligence Res. 11, 169–198.
Park, J., Kang, D., Kim, J., Kwok, J., Tsang, I., 2007. SVDD-based pattern denoising. Neural Comput. 19, 1919–1938.
Rätsch, G., Onoda, T., Müller, K.R., 2001. Soft margins for AdaBoost. Mach. Learn. 42 (3), 287–320.
Rätsch, G., Schölkopf, B., Smola, A., Müller, K.R., Onoda, T., Mika, S., 1999. v-Arc:: Ensemble learning in the presence of outliers. Adv. Neural Inf. Process. Syst. 12.
Reyzin, L., Schapire, R.E., 2006. How boosting the margin can also boost classifier complexity. In: Proceedings of the 23rd International Conference on Machine Learning.
Rosset, S., Zhu, J., Hastie, T., 2004. Boosting as a regularized path to a maximum margin classifier. J. Mach. Learn. Res. 941–973.
Schapire, R.E., 1990. The strength of weak learnability. Mach. Learn. 5, 197–227.
Schapire, R.E., Freund, Y., Bartlett, P., Lee, W.S., 1998. Boosting the margin: a new explanation for the effectiveness of voting methods. Ann. Statist. 26, 1651–1686.
Servedio, R., 2003. Smooth boosting and learning with malicious noise. J. Mach. Learn. Res. 4, 633–648.
Sigillito, V.G., Wing, S.P., Hutton, L.V., Baker, K.B., 1989. Classification of radar returns from the ionosphere using neural networks. Johns Hopkins APL Tech. Dig. 10, 262–266.
Tax, D.M., Duin, R.P., 1999. Support vector domain description. Pattern Recognit. Lett. 20, 1191–1199.
Tax, D.M., Duin, R.P., 2004. Support vector data description. Mach. Learn. 54, 45–66.
Tukey, J., 1975. Mathematics and the picturing of data. In: Proc. International Congress of Mathematicians, vol. 2, pp. 523–531.
Valiant, L.G., 1984. A theory of the learnable. Commun. ACM 27 (11), 1134–1142.
Zhang, C., Zhang, J., 2008. A local boosting algorithm for solving classification problems. Comput. Statist. Data Anal. 52, 1928–1941.

Please cite this article in press as: Martinez, W., Gray, J.B., Noise peeling methods to improve boosting algorithms. Computational Statistics and Data Analysis (2015), http://dx.doi.org/10.1016/j.csda.2015.06.010