Information Sciences 317 (2015) 315–328

Contents lists available at ScienceDirect

Information Sciences

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

# Improving over-fitting in ensemble regression by imprecise probabilities

Lev V. Utkin [a], Andrea Wiencierz [b,*]

[a] Department of Control, Automation, and System Analysis, Saint Petersburg State Forest Technical University, Institutsky pereulok 5, Saint-Petersburg 194021, Russia
[b] Department of Mathematics, University of York, York YO10 5DD, United Kingdom

ARTICLE INFO

*Article history:*
Received 7 November 2014
Received in revised form 24 March 2015
Accepted 17 April 2015
Available online 23 April 2015

*Keywords:*
Regression
AdaBoost algorithm
Over-fitting
Linear-vacuous mixture model
Kolmogorov–Smirnov bounds

ABSTRACT

In this paper, generalized versions of two ensemble methods for regression based on variants of the original AdaBoost algorithm are proposed. The generalization of these regression methods consists in restricting the unit simplex for the weights of the instances to a smaller set of weighting probabilities. Various imprecise statistical models can be used to obtain a restricted set of weighting probabilities, whose sizes each depend on a single parameter. For particular choices of this parameter, the proposed algorithms reduce to standard AdaBoost-based regression algorithms or to standard regression. The main advantage of the proposed algorithms compared to the basic AdaBoost-based regression methods is that they have less tendency to over-fitting, because the weights of the hard instances are restricted. Several simulations and applications furthermore indicate a better performance of the proposed regression methods in comparison with the corresponding standard regression methods.



## 1. Introduction

Regression analysis is one of the main problems in applied statistics. Roughly speaking, the aim is to estimate a function $f : \mathcal{X} \to \mathbb{R}$, where $\mathcal{X} \subset \mathbb{R}^m$ with $m \in \mathbb{N}$, from a finite set of noisy samples $(x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{X} \times \mathcal{Y}$ with $\mathcal{Y} \subset \mathbb{R}$ and $n \in \mathbb{N}$. A large number of regression methods were developed in the last decades, many of which are based on the minimization of a risk functional defined by a certain loss function and by the probability distribution of the data [11,20,24]. In practice, the estimated function is obtained by minimizing the so-called empirical risk (possibly regularized) defined as the sum of the loss values for the given data points divided by $n$, which can be interpreted as the risk functional associated with the empirical distribution of the data. In this paper, we focus on this kind of regression methods within the discussed algorithms, because it is very easy to incorporate individual weights for the instances, which is a core element of the algorithms we generalize. The empirical distribution can be represented as the point $\hat{p} = (n^{-1}, \ldots, n^{-1})$ in the unit simplex with $n$ vertices denoted by $S(1, n)$, which represents the set of all discrete

* Corresponding author. Tel.: +44 1904 32 3667.
*E-mail addresses:* lev.utkin@mail.ru (L.V. Utkin), andrea.wiencierz@york.ac.uk (A. Wiencierz).

http://dx.doi.org/10.1016/j.ins.2015.04.037
0020-0255/

probability measures on the $n$ observations. Hence, weighted estimates can simply be interpreted as minimizers of the risk functional associated with another discrete probability distribution $p = (p_1, \ldots, p_n)$ of the data, different from the empirical distribution $\hat{p}$. A popular technique in regression estimation is the ensemble methodology. The popularity of ensemble methods for regression stems from success of boosting methods for classification, in particular, of the well-known AdaBoost (Adaptive Boosting) algorithm proposed by Freund and Schapire [6]. AdaBoost is a general purpose boosting algorithm that can be used in conjunction with many different learning algorithms to improve their performance. The basic scheme of the AdaBoost algorithm for classification is the following: Initially, a standard classifier is estimated, assigning identical weights to all examples, then, in each of a previously fixed number of iterations, the weights of all misclassified examples are increased, while the weights of correctly classified examples are decreased, before again computing a classifier accounting for the unequal weights of the instances. In this way, with each step, the classifier focuses more and more on the difficult examples of the training data set, thereby improving the classification accuracy. The final result obtained by AdaBoost is a weighted majority vote of the classifiers of each iteration, which has a better prediction performance than each of the individual classifiers alone.

Several detailed reviews of different kinds of boosting methods were published in the last decade [2,5,14–16]. One of the first boosting algorithms for regression is the so-called AdaBoost.R2 proposed in Drucker [4], where real-valued residuals replace the 0–1 misclassification errors in the evaluation of the estimates. However, the base regression estimates are evaluated by the weighted average of the absolute values of the residuals scaled to $[0, 1]$, which is a similar error measure to the misclassification rate. Up to the recent years, many more boosting methods for regression were developed, a recent survey is provided in Mendes-Moreira et al. [15]. In contrast to most of the ensemble-based algorithms using the weighted average of base regression estimates as their final regression functions, Kegl [13] analyzed the choice of the weighted median and proposed the corresponding algorithm called MedBoost. Another interesting boosting scheme for regression problems is proposed in Solomatine and Shrestha [19], where a threshold value for the residuals is introduced to transform the real-valued errors back to the 0–1 errors, which directly fit into the original AdaBoost scheme. This adaptation of the AdaBoost algorithm is called AdaBoost.RT and its properties were investigated in Shrestha and Solomatine [17].

A common feature of these boosting algorithms is that they iteratively search for a discrete probability distribution of the training data such that the regression error is minimized. If the algorithm searches too long or concentrates too much on a few hard-to-learn examples, the problem of over-fitting can occur. There are different approaches to deal with this problem. One possibility is to explicitly use the maximum number of iterations as a regularization parameter and select it optimally, for example, by cross-validation. Another way to regularize ensemble-based boosting algorithms can be derived following the idea of shrinkage proposed for gradient boosting by Friedman [8]. Shrinkage reduces the learning rate of the algorithm, which means that it shrinks the difference of the regression estimates between two subsequent steps. In the AdaBoost-based regression methods considered here, this idea can be transferred to shrinking the difference of the weights of each instance between two steps or to limiting the maximum size of the weights, thus preventing the algorithm from focusing too much on certain instances. In this paper, we follow this idea but we propose to use imprecise statistical models like the linear-vacuous mixture model or Kolmogorov–Smirnov bounds to restrict the set of weighting probabilities. To modify the boosting algorithms accordingly, we replace the adaption of the instances' weighting probabilities with the updating of weights in the convex linear combination of the extreme points of the restricted set. Thus, we here present a general tool for modifying available boosting algorithms and for constructing a number of new ensemble-based methods less prone to the problem of over-fitting.

In the following two sections, we propose the corresponding modifications of two popular boosting algorithms: AdaBoost.R2 introduced in Drucker [4] and AdaBoost.RT proposed in Solomatine and Shrestha [19]. Section 4 reviews suitable imprecise probability models to obtain the restricted set of weighting probabilities. Finally, we assess the performance of the modified algorithms by means of synthetic and real data.

## 2. AdaBoost.R2 and its modification

At first, we modify the AdaBoost.R2 algorithm proposed in Drucker [4]. The scheme of this boosting algorithm for regression is presented as Algorithm 1. Given a training data set $Z = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ and a regression method which is suitable for weighted estimation, the algorithm requires a maximum number of iterations $T \in \mathbb{N}$ to be chosen a priori. Then, the iteration index $t$ is set to one and the weighting probabilities $p_i^{(1)}$ are set to $n^{-1}$ for all $i \in \{1, \ldots, n\}$. (Alternatively, the vector $p^{(1)}$ could be randomly selected from the unit simplex $S(1, n)$.) In each iteration step $t \in \{1, \ldots, T\}$, a regression function $\hat{f}^{(t)}$ is estimated using the weights $p^{(t)}$. In contrast to AdaBoost for classification, where the estimated classifiers are evaluated by their average misclassification error, the regression estimates are evaluated on the basis of the absolute residuals $|y_i - \hat{f}^{(t)}(x_i)|$ with $i \in \{1, \ldots, n\}$. Yet, to obtain an overall error measure similar to the misclassification rate, the absolute residuals are divided by the maximum value $D^{(t)}$ such that the weighted sum $\epsilon^{(t)}$ of the normalized residuals $\hat{e}_1^{(t)}, \ldots, \hat{e}_n^{(t)}$ lies in the interval $[0, 1]$. If $\epsilon^{(t)} > 0.5$, we exit the loop and use only the first $t - 1$ regression estimates to determine the final result. In the context

**Algorithm 1**. AdaBoost.R2.

**Require:** Maximum number of iterations $T$ and training data set $Z$.
**Ensure:** $\alpha^{(t)}$ and $\hat{f}^{(t)}$ for all $t \in \{1,\dots,T\}$;
  set $t \leftarrow 1$ and $p^{(t)} \leftarrow (n^{-1},\dots,n^{-1})$;
  **repeat**
    estimate $\hat{f}^{(t)}$ using weighting probabilities $p^{(t)}$;
    compute $D^{(t)} \leftarrow \max_{j\in\{1,\dots,n\}} |y_j - \hat{f}^{(t)}(x_j)|$;
    compute normalized errors for all $i \in \{1,\dots,n\}$:
    $\hat{e}_i^{(t)} \leftarrow \frac{|y_i - \hat{f}^{(t)}(x_i)|}{D^{(t)}}$;
    calculate the overall error of $\hat{f}^{(t)}$:
    $\epsilon^{(t)} \leftarrow \sum_{i=1}^{n} p_i^{(t)} \hat{e}_i^{(t)}$;
    **if** $\epsilon^{(t)} > 0.5$ **then**
      $T \leftarrow t-1$;
    **end if**
    compute contribution of $\hat{f}^{(t)}$ to the final result:
    $\alpha^{(t)} \leftarrow \ln\left(\frac{1-\epsilon^{(t)}}{\epsilon^{(t)}}\right)$;
    adapt weights for all $i \in \{1,\dots,n\}$:
    $p_i^{(t+1)} \leftarrow p_i^{(t)} \exp\left(-\alpha^{(t)}\left(1-\hat{e}_i^{(t)}\right)\right)$;
    normalize $p^{(t+1)}$ to be a proper probability measure;
    $t++$
  **until** $t > T$
  normalize $\alpha^{(1)},\dots,\alpha^{(T)}$ such that $\sum_{t=1}^{T}\alpha^{(t)} = 1$;
  compute regression function $\hat{f} \leftarrow \sum_{t=1}^{T}\alpha^{(t)}\hat{f}^{(t)}$.

of classification this is a sensible stopping criterion, because it means that classifiers with an error rate higher than 50% may not contribute to the combined result. However, in the regression context the usefulness of this stopping criterion is less clear. Here, it corresponds to stopping the iterations when the situation arises, where the average normalized residual is larger than 50% of the maximum absolute residual. If $\epsilon^{(t)} \leqslant 0.5$, the overall error is used to determine the contribution $\alpha^{(t)}$ of the estimated function $\hat{f}^{(t)}$ in the combined result $\hat{f}$. Furthermore, the weighting probabilities of the instances are adapted by the formula:

$$p_i^{(t+1)} = p_i^{(t)} \exp\left(-\alpha^{(t)}\left(1-\hat{e}_i^{(t)}\right)\right)$$

for all $i \in \{1,\dots,n\}$. Thus, the weights of examples with relatively large residuals are increased, while the others are decreased. As the last step within each iteration, we normalize $(p_1^{(t+1)},\dots,p_n^{(t+1)})$ to obtain a proper weighting distribution where $\sum_{i=1}^{n} p_i^{(t+1)} = 1$. Finally, when the loop is ended, the $\alpha^{(1)},\dots,\alpha^{(T)}$ are adjusted such that $\sum_{t=1}^{T}\alpha^{(t)} = 1$ and the combined result $\hat{f} = \sum_{t=1}^{T}\alpha^{(t)}\hat{f}^{(t)}$ is determined.

According to the adaption rule, the distribution of weighting probabilities $p$ can be an arbitrary point in $S(1,n)$ including its vertices. Indeed, as already shown for the basic AdaBoost algorithm in Freund and Schapire [6], the weighting probabilities of the examples tend to concentrate on instances which have large residuals compared with the other data points and may be outliers. Hence, the regression function will be estimated by taking mainly these hard-to-learn examples into account. The obtained estimated function will perform well on these extreme data points but may perform rather poor on other examples. This property is called over-fitting, because the resulting overall fit is too much influenced by a few data points, while the actual functional relation is better reflected by the neglected examples. Therefore, also the out-of-sample prediction performance of such a regression estimate may be very bad.

Let us now consider a set $\mathcal{P}$ of probability distributions, which is a subset of the unit simplex, i.e., $\mathcal{P} \subset S(1,n)$. We assume that $\mathcal{P}$ is convex, more precisely, that it is produced by finitely many linear constraints. This implies that it is totally defined by its $r$ extreme points $q^{(k)} = \left(q_1^{(k)},\dots,q_n^{(k)}\right)$ for all $k \in \{1,\dots,r\}$ with $r \in \mathbb{N}$. Thus, every probability distribution $p \in \mathcal{P}$ can be represented as

$$p = \sum_{k=1}^{r} \lambda_k q^{(k)},$$

where $\lambda = (\lambda_1,\dots,\lambda_r)$ is a vector of weights such that $\sum_{k=1}^{r}\lambda_k = 1$.

The core idea of the modification of AdaBoost.R2 we propose here is to adapt the weights in $\lambda$ instead of updating directly $p$. This does not mean that the weighting distribution $p$ is not updated in the iterations, but it changes only within the set $\mathcal{P}$ and through adaption of $\lambda$. For the weights $\lambda_1,\dots,\lambda_r$ there are no additional restrictions, they move freely in the unit simplex having $r$ vertices denoted by $S(1,r)$. Thus, in the scheme of the modified algorithm presented as Algorithm 2, we replace $p^{(t)}$ with $\sum_{k=1}^{r}\lambda_k^{(t)}q^{(k)}$. Instead of initializing $p^{(1)}$ with the empirical distribution, we set $\lambda^{(1)} = (r^{-1},\dots,r^{-1})$. (Alternatively, the

vector $\lambda^{(1)}$ could be randomly selected from $S(1,r)$.) When we substitute $p^{(t)}$ in the formula of the overall error measure of the $t$th regression estimate, we obtain the following representation:

$$\epsilon^{(t)} = \sum_{i=1}^{n} \hat{e}_i^{(t)} p_i^{(t)} = \sum_{i=1}^{n} \hat{e}_i^{(t)} \sum_{k=1}^{r} \lambda_k^{(t)} q_i^{(k)} = \sum_{k=1}^{r} \lambda_k^{(t)} \hat{\varepsilon}_k^{(t)},$$

where $\hat{\varepsilon}_k^{(t)} = \sum_{i=1}^{n} \hat{e}_i^{(t)} q_i^{(k)}$ can be interpreted as the contribution of the $k$th extreme point to the average normalized residual. It corresponds to the mean value of the normalized residuals with respect to the discrete distribution $q^{(k)} \in \mathcal{P}$. Moreover, the above representation unveils a nice characteristic of the proposed modification of the algorithm. In fact, it implies that the $n$ examples are transformed to $r \geqslant n$ virtual data points (i.e., the extreme points) with associated residuals $\hat{\varepsilon}_k^{(t)}$ and weights $\lambda_k^{(t)}$ for all $k \in \{1,\ldots,r\}$.

**Algorithm 2.** Imprecise AdaBoost.R2.

---

**Require:** Maximum number of iterations $T$, training data set $Z$ and extreme points $q^{(1)},\ldots,q^{(r)}$ of $\mathcal{P}$.
**Ensure:** $\alpha^{(t)}$ and $\hat{f}^{(t)}$ for all $t \in \{1,\ldots,T\}$; set $t \leftarrow 1$ and $\lambda^{(1)} \leftarrow (r^{-1},\ldots,r^{-1})$;
**repeat**
compute the vector of weighting probabilities:
$p^{(t)} \leftarrow \sum_{k=1}^{r} \lambda_k^{(t)} q^{(k)}$;
estimate $\hat{f}^{(t)}$ using weighting probabilities $p^{(t)}$;
compute $D^{(t)} \leftarrow \max_{j\in\{1,\ldots,n\}} \left| y_j - \hat{f}^{(t)}(x_j) \right|$;
compute normalized errors for all $i \in \{1,\ldots,n\}$:
$\hat{e}_i^{(t)} \leftarrow \frac{|y_i - \hat{f}^{(t)}(x_i)|}{D^{(t)}}$;
compute error portions for all $k \in \{1,\ldots,r\}$:
$\hat{\varepsilon}_k^{(t)} \leftarrow \sum_{i=1}^{n} \hat{e}_i^{(t)} q_i^{(k)}$;
calculate the overall error of $\hat{f}^{(t)}$:
$\epsilon^{(t)} \leftarrow \sum_{k=1}^{r} \lambda_k^{(t)} \hat{\varepsilon}_k^{(t)}$;
**if** $\epsilon^{(t)} > 0.5$ **then**
$T \leftarrow t - 1$;
**end if**
compute contribution of $\hat{f}^{(t)}$ to the final result:
$\alpha^{(t)} \leftarrow \ln\left(\frac{1-\epsilon^{(t)}}{\epsilon^{(t)}}\right)$;
adapt weights for all $k \in \{1,\ldots,r\}$:
$\lambda_k^{(t+1)} \leftarrow \lambda_k^{(t)} \exp\left(-\alpha^{(t)}\left(1 - \hat{\varepsilon}_k^{(t)}\right)\right)$;
normalize $\lambda^{(t+1)}$ such that $\sum_{k=1}^{r} \lambda_k^{(t+1)} = 1$;
$t{+}{+}$
**until** $t > T$
normalize $\alpha^{(1)},\ldots,\alpha^{(T)}$ such that $\sum_{t=1}^{T} \alpha^{(t)} = 1$;
compute regression function $\hat{f} \leftarrow \sum_{t=1}^{T} \alpha^{(t)} \hat{f}^{(t)}$.

---

From this interpretation, it is straightforward to derive the updating rule to obtain the weights $\lambda_1^{(t+1)},\ldots,\lambda_r^{(t+1)}$. In the same way as the weighting probabilities of the data are adapted in Algorithm 1, we increase the weights of those extreme points with large errors $\hat{\varepsilon}_k^{(t)}$ and vice versa. Hence, we simply adapt the updating rule given in AdaBoost.R2 and update the weights of the extreme points by

$$\lambda_k^{(t+1)} = \lambda_k^{(t)} \exp\left(-\alpha^{(t)}\left(1 - \hat{\varepsilon}_k^{(t)}\right)\right)$$

for all $k \in \{1,\ldots,r\}$. The $\lambda_1^{(t+1)},\ldots,\lambda_r^{(t+1)}$ are also normalized to fulfill the condition $\sum_{k=1}^{r} \lambda_k^{(t+1)} = 1$. Note that the obtained weighting probability distribution $p^{(t+1)}$ again belongs to the set $\mathcal{P}$ because it is a convex linear combination of the corresponding extreme points.

Let us now consider the special case where we do not have additional information, and thus, $\mathcal{P} = S(1,n)$. In this case, there are $r = n$ extreme points corresponding to the vertices of the unit simplex, e.g., for $k = 1$ we have $q^{(1)} = (1,0,\ldots,0)$. Then, $p^{(t)} = \left(\lambda_1^{(t)},\ldots,\lambda_n^{(t)}\right)$ and the $k$th extreme point mean error $\hat{\varepsilon}_k^{(t)} = \hat{e}_k^{(t)}$ for all $t \in \{1,\ldots,T\}$. Hence, we get the following updated weights for all $k \in \{1,\ldots,n\}$:

$$\lambda_k^{(t+1)} = p_k^{(t)} \exp\left(-\alpha^{(t)}\left(1-\hat{e}_k^{(t)}\right)\right) = p_k^{(t+1)},$$

which coincide with those obtained in AdaBoost.R2. This implies that the proposed algorithm is a generalization of the standard AdaBoost.R2 and covers it as the special case where the set of weighting probabilities is not restricted.

The proposed Imprecise AdaBoost.R2 algorithm has several positive features in comparison with the standard AdaBoost.R2. As the number of extreme points of $\mathcal{P}$ is always larger than or equal to the number of examples, the modified algorithm can have a larger number of parameters to adjust. In this case, the weighting probabilities can be adapted in finer steps within the set $\mathcal{P}$. Furthermore, when we have only a few examples, the overall errors $\epsilon^{(t)}$ of the $\hat{f}^{(t)}$ with $t \in \{1,\ldots,T\}$ can only be determined with much uncertainty due to the high variance of the estimates. As a result, the weights may change very quickly and the algorithm may become unstable. The proposed modification of the AdaBoost.R2 algorithm is less affected by this problem if $\mathcal{P}$ is a proper subset of $S(1,n)$, because in this case the weights cannot be too large and hence neither the differences between the weighting probabilities of an instance in two subsequent iteration steps. Finally, any set of discrete probabilities defined by linear constraints can be used in the algorithm. This allows to introduce any prior information of this kind about the training data. In Section 4, we discuss a selection of imprecise statistical models to derive $\mathcal{P}$, but in principle it can be any convex subset of $S(1,n)$. Moreover, it is possible to further generalize the proposed Imprecise AdaBoost.R2 algorithm and allow the set $\mathcal{P}$ to be changed in every iteration step according to some rule, for instance, by means of generalized Bayesian updating.

## 3. Threshold AdaBoost algorithm and its modification

In this section, we consider the AdaBoost.RT algorithm introduced in Solomatine and Shrestha [19]. This algorithm is based on the idea that the training examples can be classified into two classes by comparing the accuracy of the predicted values with a predefined relative error threshold. Then, the evaluation of the regression estimates $\hat{f}^{(t)}$ within the iterated loop of the algorithm can be done on the basis of the average misclassification error like in the basic AdaBoost algorithm for binary classification. Algorithm 3 outlines the scheme of the AdaBoost.RT algorithm.

**Algorithm 3.** AdaBoost.RT

**Require:** Maximum number of iterations $T$, training data set $Z$, threshold $\tau$ and power coefficient $l$.
**Ensure:** $\alpha^{(t)}, \beta^{(t)}$ and $\hat{f}^{(t)}$ for all $t \in \{1,\ldots,T\}$;
  set $t \leftarrow 1$ and $p^{(t)} \leftarrow (n^{-1},\ldots,n^{-1})$;
  **repeat**
    estimate $\hat{f}^{(t)}$ using weighting probabilities $p^{(t)}$;
    compute relative errors for all $i \in \{1,\ldots,n\}$:
    $\hat{e}_i^{(t)} \leftarrow \left|\frac{y_i - \hat{f}^{(t)}(x_i)}{y_i}\right|$;
    calculate the overall error rate of $\hat{f}^{(t)}$:
    $\epsilon^{(t)} \leftarrow \sum_{\{i:\hat{e}_i^{(t)} > \tau\}} p_i^{(t)}$;
    compute $\beta^{(t)} \leftarrow \left(\epsilon^{(t)}\right)^l$;
    compute contribution of $\hat{f}^{(t)}$ to the final result:
    $\alpha^{(t)} \leftarrow -\ln\left(\beta^{(t)}\right)$;
    adapt weights for all $i \in \{1,\ldots,n\}$ by:
    **if** $\hat{e}_i^{(t)} \leqslant \tau$ **then**
      $p_i^{(t+1)} \leftarrow p_i^{(t)} \beta^{(t)}$;
    **else**
      $p_i^{(t+1)} \leftarrow p_i^{(t)}$;
    **end if**
    normalize $p^{(t+1)}$ to be a proper probability measure;
    $t++$
  **until** $t > T$
  normalize $\alpha^{(1)},\ldots,\alpha^{(T)}$ such that $\sum_{t=1}^{T} \alpha^{(t)} = 1$;
  compute regression function $\hat{f} \leftarrow \sum_{t=1}^{T} \alpha^{(t)} \hat{f}^{(t)}$.

In contrast to the normalized absolute residuals of AdaBoost.R2, here the regression errors $\hat{e}_1^{(t)},\dots,\hat{e}_n^{(t)}$ are given by the absolute values of the relative residuals, for each $t\in\{1,\dots,T\}$. These residuals are compared to a threshold value $\tau\in\mathbb{R}_{\geqslant 0}$. The corresponding examples are considered as misclassified if their residual exceeds $\tau$ and as correctly classified otherwise. Thus, as in AdaBoost for classification, each estimated function $\hat{f}^{(t)}$ is evaluated by its overall misclassification rate $\epsilon^{(t)}=\sum_{\{i:\hat{e}_i^{(t)}>\tau\}}p_i^{(t)}$. Furthermore, the weights are updated according to a rule depending on $\tau$. The weights associated with examples with small relative residuals are decreased, while those of the examples considered as misclassified remain constant. By normalizing $p_1^{(t+1)},\dots,p_n^{(t+1)}$ to obtain a probability distribution, the weighting probabilities of the misclassified examples are, in fact, increased.

An important feature of the algorithm is that it does not stop when the overall error rate $\epsilon^{(t)}$ is greater than 0.5. In AdaBoost.RT it is not necessary to explicitly state a stopping criterion, because the computation scheme for the weights $\alpha^{(t)}$ of the regression estimates in the combined result implies that poor estimates are almost neglected and vice versa. That is, if $\epsilon^{(t)}$ is high, so is $\beta^{(t)}=(\epsilon^{(t)})^l$ for some $l\in\mathbb{N}$, and thus, $\alpha^{(t)}=-\ln(\beta^{(t)})$ will be very small compared to better estimates of other iterations. In Shrestha and Solomatine [17] it is also argued that even if $\epsilon^{(t)}>0.5$ for some of the estimates in the ensemble, the final output of the ensemble-based algorithm is better than that of a single regression estimate. That is why AdaBoost.RT does not have a stopping rule like the AdaBoost.R2 algorithm, although it would fit the framework of this algorithm very well, as the evaluation is based on a pseudo misclassification error rate.

**Algorithm 4.** Imprecise AdaBoost.RT

---

**Require:** Maximum number of iterations $T$, training data set $Z$, threshold $\tau$, power coefficient $l$ and extreme points $q^{(1)},\dots,q^{(r)}$ of $\mathcal{P}$.

**Ensure:** $\alpha^{(t)},\beta^{(t)}$ and $\hat{f}^{(t)}$ for all $t\in\{1,\dots,T\}$;

set $t\leftarrow 1$ and $\lambda^{(1)}\leftarrow(r^{-1},\dots,r^{-1})$;

**repeat**

compute the vector of weighting probabilities:

$p^{(t)}\leftarrow\sum_{k=1}^{r}\lambda_k^{(t)}q^{(k)}$;

estimate $\hat{f}^{(t)}$ using weighting probabilities $p^{(t)}$;

compute relative errors for all $i\in\{1,\dots,n\}$:

$\hat{e}_i^{(t)}\leftarrow\left|\frac{y_i-\hat{f}^{(t)}(x_i)}{y_i}\right|$;

compute error portions for all $k\in\{1,\dots,r\}$:

$\hat{\varepsilon}_k^{(t)}\leftarrow\sum_{i=1}^{n}\hat{e}_i^{(t)}q_i^{(k)}$;

calculate the overall error rate of $\hat{f}^{(t)}$:

$\epsilon^{(t)}\leftarrow\sum_{\{k:\hat{\varepsilon}_k^{(t)}>\tau\}}\lambda_k^{(t)}$;

compute $\beta^{(t)}\leftarrow\left(\epsilon^{(t)}\right)^l$;

compute contribution of $\hat{f}^{(t)}$ to the final result:

$\alpha^{(t)}\leftarrow-\ln\left(\beta^{(t)}\right)$;

adapt weights for all $k\in\{1,\dots,r\}$ by:

**if** $\hat{\varepsilon}_k^{(t)}\leqslant\tau$ **then**

$\lambda_k^{(t+1)}\leftarrow\lambda_k^{(t)}\beta^{(t)}$;

**else**

$\lambda_k^{(t+1)}\leftarrow\lambda_k^{(t)}$;

**end if**

normalize $\lambda^{(t+1)}$ such that $\sum_{k=1}^{r}\lambda_k^{(t+1)}=1$;

$t++$

**until** $t>T$

normalize $\alpha^{(1)},\dots,\alpha^{(T)}$ such that $\sum_{t=1}^{T}\alpha^{(t)}=1$;

compute regression function $\hat{f}\leftarrow\sum_{t=1}^{T}\alpha^{(t)}\hat{f}^{(t)}$.

---

In spite of the virtues of AdaBoost.RT, it has the shortcoming that the threshold must be selected a priori, because the performance of the algorithm is sensitive to $\tau$. If $\tau$ is too low, then it is generally very difficult to get a sufficient number of correctly predicted examples. Furthermore, the standard AdaBoost.RT algorithm has the same tendency to over-fitting as the AdaBoost.R2 algorithm due to the too large set of weighting probabilities. In order to overcome this disadvantage,

we propose a modified version of AdaBoost.RT where the set of weighting probabilities is restricted to the convex set $\mathcal{P}$ with extreme points $q^{(k)} = \left(q_1^{(k)}, \ldots, q_n^{(k)}\right)$ with $k \in \{1, \ldots, r\}$. The scheme of the modified AdaBoost.RT is presented as Algorithm 4. Again, we can interpret the proposed modification as replacing the $n$ training data with $r$ virtual examples (i.e., the extreme points of $\mathcal{P}$) with residuals $\hat{\varepsilon}_k^{(t)}$ and weights $\lambda_k$ for all $k \in \{1, \ldots, r\}$. Then, the overall error rates $\epsilon^{(t)}$ are obtained as $\sum_{\{k:\hat{\varepsilon}_k^{(t)} > \tau\}} \lambda_k^{(t)}$.

Let us again consider the special case without additional information about the weighting probabilities, and thus, $\mathcal{P} = S(1, n)$ with $r = n$ vertices. Then, for all $t \in \{1, \ldots, T\}$ we obtain $p^{(t)} = \left(\lambda_1^{(t)}, \ldots, \lambda_n^{(t)}\right)$ and $\lambda_k^{(t+1)} = p_k^{(t+1)}$ for all $k \in \{1, \ldots, n\}$, while the $k$th extreme point mean error is given by $\hat{\varepsilon}_k^{(t)} = \hat{e}_k^{(t)}$ and $\epsilon^{(t)} = \sum_{k:\hat{\varepsilon}_k^{(t)} > \tau} p_k^{(t)}$. Hence, also the standard AdaBoost.RT algorithm is a special case of its here proposed modification.

## 4. Imprecise statistical models

In this section, we briefly review a selection of imprecise statistical models that can be used to determine the set of weighting probabilities $\mathcal{P} \subset S(1, n)$. In particular, we consider three different imprecise neighborhood models around the empirical distribution $\hat{p}$ of the training data and one statistical approach derived from the Kolmogorov–Smirnov test. Every imprecise neighborhood model is characterized by a parameter $v$, which in some case can be interpreted as the (subjective) probability that the elicited probability distribution $p$ is incorrect or as the size of possible errors in $p$. The size of the Kolmogorov–Smirnov-based set of weighting probabilities is determined by the coverage level $\gamma$.

### 4.1. The linear-vacuous mixture model

The linear-vacuous mixture or imprecise $v$-contaminated models produce the set $\mathcal{P}(v, p)$ of probabilities $\pi = (\pi_1, \ldots, \pi_n)$ such that $\pi_i = (1 - v)p_i + vb_i$ for some $v \in [0, 1]$ and for all $i \in \{1, \ldots, n\}$, where $p = (p_1, \ldots, p_n)$ is the elicited probability distribution and $(b_1, \ldots, b_n)$ can be any probability distribution in $S(1, n)$ [25, Sections 2.9.2 and 3.3.5]. The set $\mathcal{P}(v, p)$ is a convex subset of the unit simplex; it coincides with $S(1, n)$ when $v = 1$, while $\mathcal{P}(v, p) = \{p\}$ if $v = 0$. For $p = \hat{p} = (n^{-1}, \ldots, n^{-1})$, the set $\mathcal{P}(v, \hat{p})$ has $r = n$ extreme points $q_k \in S(1, n)$ with $k \in \{1, \ldots, n\}$, which are all of the same form: the $k$th element is given by $(1 - v)n^{-1} + v$ and the other $n - 1$ elements are equal to $(1 - v)n^{-1}$. For example, the extreme point $q_2$ is given by

$$q_2 = \left(\frac{1 - v}{n}, \frac{1 - v}{n} + v, \ldots, \frac{1 - v}{n}\right).$$

### 4.2. The pari-mutuel model

Another imprecise neighborhood model is the imprecise pari-mutuel model [25, Sections 2.9.3 and 3.3.5], for which the set of probability distributions is defined as

$$\mathcal{P}_P(v, p) = \{\pi \in S(1, n) : \pi_i \leqslant (1 + v)p_i \ \forall\, i \in \{1, \ldots, n\}\},$$

where $v \in [0, +\infty)$ and $p = (p_1, \ldots, p_n)$ is the elicited distribution. The set $\mathcal{P}_P(v, p)$ consists of all probability distributions $\pi$ such that the weighting probabilities of the $n$ atomic events do not exceed a constant multiple of the probabilities given by the distribution $p$. Lower and upper probabilities of the $i$th point are given by $\max\{0, (1 + v)p_i - v\}$ and $\min\{(1 + v)p_i, 1\}$, respectively. The difference between the upper and lower probabilities is $v$, as long as $p_i$ is far enough from 0 or 1 and $v$ is not too large.

When we consider the empirical distribution $\hat{p} = (n^{-1}, \ldots, n^{-1})$ as the elicited distribution, the extreme points of the set $\mathcal{P}_P(v, \hat{p})$ depend on the chosen parameter $v$ as expressed in the following proposition.

**Proposition 1.** *Let $z_1, \ldots, z_n$ with $n \in \mathbb{N}$ be a set of univariate data and let $\mathcal{P}_P(v, \hat{p})$ be the set of probabilities according to a pari-mutuel neighborhood model around the empirical distribution $\hat{p} = (n^{-1}, \ldots, n^{-1})$ of the data for some $v \in [0, +\infty)$.*

(1) If $v \leqslant (n - 1)^{-1}$, then the set $\mathcal{P}_P(v, \hat{p})$ has $r = n$ extreme points $q_k \in S(1, n)$ with $k \in \{1, \ldots, n\}$, which are of the following form: the $k$th element is given by $(1 + v)n^{-1} - v$ and the other $n - 1$ elements are equal to $(1 + v)n^{-1}$.

(2) If $(n - 1)^{-1} < v < (n - 1)$, then the set $\mathcal{P}_P(v, \hat{p})$ has $r = s\binom{n}{s}$ extreme points, where $s \in \mathbb{N}$ and it is defined by the inequality

$$\frac{1}{n - s + 1} \leqslant \frac{1 + v}{n} \leqslant \frac{1}{n - s}.$$

The extreme points have the same form: $n-s$ elements have value $(1+v)n^{-1}$, there is one element given by $1-(n-s)(1+v)n^{-1}$, and the other $s-1$ elements are equal to zero.

(3) If $v \geqslant (n-1)$, then $\mathcal{P}_P(v,\hat{p}) = S(1,n)$.

The third part of this proposition is obvious, because in this case the upper probabilities of all points are equal to one. The proofs of parts (1) and (2) can be found in Utkin [22, Propositions 1 and 3]. In fact, the set $\mathcal{P}_P(v,\hat{p})$ can also be obtained from $\mathcal{P}(v',\hat{p})$ by reflection about the point $p$, when $v'$ and $v$ are appropriately chosen.

### 4.3. The constant odds-ratio model

The third imprecise neighborhood model we consider is the so-called constant odds-ratio $(\pi, v)$ model [25, Sections 2.9.4 and 3.3.5]. For this model, the set of probability distributions defined as the neighborhood of a given distribution $p$ is

$$\mathcal{P}_C(v,p) = \left\{ \pi \in S(1,n) : \frac{\pi_i}{\pi_j} \geqslant (1-v)\frac{p_i}{p_j} \ \forall\ i,j \in \{1,\ldots,n\} \right\}.$$

Here, for all $i \in \{1,\ldots,n\}$, the probabilities $\pi_i > 0$ because otherwise $\mathcal{P}_C(v,p)$ is not well-defined. This implies that $v \in [0,1)$. Under the constant odds-ratio model, the lower and upper weighting probabilities of the $i$th atomic event are given by

$$\underline{\pi}_i = \frac{(1-v)p_i}{1-vp_i} \quad \text{and} \quad \overline{\pi}_i = \frac{p_i}{1-v(1-p_i)}.$$

According to Walley [25, Section 2.9.4.], this model is particularly convenient for statistical applications because its shape is not changed by conditioning or statistical updating.

When we take the empirical distribution of the training data $\hat{p} = (n^{-1},\ldots,n^{-1})$ as the probability distribution of interest, the corresponding set $\mathcal{P}_C(v,\hat{p})$ has $2n$ extreme points $q_k \in S(1,n)$ with $k \in \{1,\ldots,2n\}$. As shown in Utkin [22, Proposition 4], the first $n$ points are such that the element $(n(1-v)+v)^{-1}$ is located at the $k$th position, while the other $n-1$ elements have value $(1-v)(n(1-v)+v)^{-1}$, e.g.,

$$q_1 = \left( \frac{1}{n(1-v)+v}, \frac{1-v}{n(1-v)+v}, \ldots, \frac{1-v}{n(1-v)+v} \right).$$

The other $n$ extreme points have at the $k$th position the element $(1-v)(n-v)^{-1}$ and $(n-v)^{-1}$ at the remaining positions, e.g., for $k = 2n-1$ it is

$$q_{2n-1} = \left( \frac{1}{n-v}, \ldots, \frac{1-v}{n-v}, \frac{1}{n-v} \right).$$

### 4.4. Kolmogorov–Smirnov bounds

A statistical approach to constructing bounds for the set of weighting probabilities can be derived from confidence bounds for the probability distribution of the data. Such confidence bounds can be obtained by inverting the so-called Kolmogorov–Smirnov test.

Let $F$ denote the cumulative distribution function associated with the unknown probability measure $P$ of some univariate data $z_1,\ldots,z_n \in \mathcal{Z} \subset \mathbb{R}$, with $n \in \mathbb{N}$, and let $\widehat{F}_n$ be the associated empirical cumulative distribution function. The Kolmogorov–Smirnov test is a nonparametric test for the null hypothesis that $z_1,\ldots,z_n$ were generated by some particular distribution $F_0$. As test statistic the supremum vertical distance of $\widehat{F}_n$ and $F_0$ is considered. It can be shown that the distribution of this test statistic under the null hypothesis is independent of $F_0$. The quantiles $k_{n,1-\gamma}$ of the test distribution are available in tables for a certain range of sample sizes and some different test levels $\gamma \in (0,1)$. For large $n$, the quantiles can be approximated by a simple formula. The null hypothesis is rejected at level $\gamma \geqslant P_{F_0}(||\widehat{F}_n - F_0||_\infty > k_{n,1-\gamma})$ if the observed supremum distance given by

$$\max_{1,\ldots,n} \max\left\{ \frac{i}{n} - F_0(z_{(i)}), F_0(z_{(i)}) - \frac{i-1}{n} \right\},$$

where $z_{(1)} \leqslant z_{(2)} \leqslant \ldots \leqslant z_{(n)}$ are the ordered observations, is larger than $k_{n,1-\gamma}$. By considering all distribution functions such that the test does not reject the null hypothesis at the chosen $\gamma$, we obtain a $1-\gamma$ confidence band for $F$ which is of the form

$$\mathcal{C}_{1-\gamma} = \left\{ F' : \underline{F}_n(z) \leqslant F'(z) \leqslant \overline{F}_n(z)\ \forall z \right\},$$

where for all $z \in \mathcal{Z}$

$$\underline{F}_n(z) = \max\left\{ \widehat{F}_n(z) - k_{n,1-\gamma}, 0 \right\} \text{ and}$$
$$\overline{F}_n(z) = \min\left\{ \widehat{F}_n(z) + k_{n,1-\gamma}, 1 \right\}.$$

As the quantiles of the exact test distribution are not available for all sample sizes and all test levels, several modifications of the above confidence band were suggested, replacing $k_{n,1-\gamma}$ by an upper approximation $d_{n,1-\gamma}$, e.g., the upper bounds provided by the so-called Dvoretzky–Kiefer–Wolfowitz inequality, resulting in more conservative but easy-to-compute confidence bands for the distribution function $F$. Therefore, we use $d_{n,1-\gamma}$ as the general notation in the following, but refer to the limiting functions $\underline{F}_n(z)$ and $\overline{F}_n(z)$ of all confidence bands of the above type (with $d_{n,1-\gamma}$ instead of $k_{n,1-\gamma}$) as Kolmogorov–Smirnov bounds. See Wassermann [26, Section 2] or Johnson and Leone [12, Section 8.9.3] for more details.

It was shown in Utkin and Coolen [23, Section 5] that it is possible to derive a set of probability mass functions $\mathcal{P}_K(\gamma)$ from a confidence band of the type $\mathcal{C}_{1-\gamma}$. Moreover, they showed that the set $\mathcal{P}_K(\gamma)$ is a convex subset of $S(1,n)$ with $s\binom{n}{s}$ extreme points, where $s \in \mathbb{N}$ is determined from the condition

$$nd_{n,1-\gamma} < s \leqslant 1 + nd_{n,1-\gamma}.$$

Every extreme point has $s-1$ elements of size 0, one element taking the value $(s - nd_{n,1-\gamma})(n(1-d_{n,1-\gamma}))^{-1}$, and $n-s$ elements of size $(n(1-d_{n,1-\gamma}))^{-1}$.

## 5. Numerical experiments

To study how well the proposed algorithms may solve practical problems, we conduct several numerical experiments. Thereby, we use weighted Support Vector Regression (SVR [18,20]) with absolute loss function and Gaussian kernel as regression estimator within the algorithms and we determine the set of weighting probabilities by means of the linear-vacuous mixture model. The corresponding set of weights has the same shape as the unit simplex and is also centered around the discrete uniform distribution, its extent being determined by the parameter $v \in [0,1]$. The closer $v$ is to one, the larger the set of weighting probabilities, allowing more extreme weights for individual observations. By contrast, for $v = 0$ only the empirical distribution is contained in the set of weighting probabilities. In this case, the generalized algorithms reduce to standard SVR, while for $v = 1$, they equal their associated standard AdaBoost methods. We make different simulations to study the performance of the proposed regression methods for different choices of $v \in [0,1]$, before we assess their practical performance on the basis of five real data sets from the UCI Machine Learning Repository [1]. As we argued before, the main goal of our proposed generalizations is to make the corresponding AdaBoost algorithms less prone to the problem of over-fitting. Therefore, we expect a better performance of the regression methods for $v < 1$.

From each of the (synthetic or real) data sets we randomly select two distinct subsets: a training data set of $n$ examples to learn the model, and a test data set of $n_{test}$ instances to evaluate the performance of the algorithms. For the synthetic data, the performance is assessed by four different measures. On the one hand, we consider commonly used criteria based on the squared error loss, more precisely, the square roots of the Mean Square Prediction Error (RMSPE) and of the Mean Square Residual (RMSR), which are defined by

$$RMSPE = \sqrt{\frac{\sum_{i=1}^{n_{test}} \left(f(x_i) - \hat{f}(x_i)\right)^2}{n_{test}}} \quad \text{and}$$

$$RMSR = \sqrt{\frac{\sum_{i=1}^{n_{test}} \left(y_i - \hat{f}(x_i)\right)^2}{n_{test}}},$$

respectively, where $f$ denotes the true function, $\hat{f}$ is the function estimated by one of the proposed algorithms, and $\hat{f}(x_i)$ is the predicted value of $y_i$ for each $i \in \{1,\ldots,n_{test}\}$. Both measures are computed on the basis of the test data set in each simulation run. On the other hand, it seems reasonable to evaluate the goodness of the algorithms also by criteria based on the absolute loss, since within the algorithms the errors are evaluated by their (scaled) absolute values and moreover the configuration of SVR chosen here minimizes the absolute loss. Hence, in each simulation run, we additionally compute the Mean Absolute Prediction Error (MAPE) and the Mean Absolute Residual (MAR)

$$MAPE = \frac{\sum_{i=1}^{n_{test}} \left|f(x_i) - \hat{f}(x_i)\right|}{n_{test}} \quad \text{and}$$

$$MAR = \frac{\sum_{i=1}^{n_{test}} \left|y_i - \hat{f}(x_i)\right|}{n_{test}}.$$

As usual, the criteria values are finally averaged over the simulation runs. The smaller the values of the average error measures are, the better the corresponding regression method. Regarding the numerical examples analyzing real data, the RMSPE and the MAPE cannot be computed, because the true function $f$ is unknown. Hence, in this case, we use only the overall RMSR and MAR obtained from repeatedly drawing training and test data sets.

### 5.1. Synthetic data

A major aim of analyzing synthetic data is to investigate how the parameter $v$ of the linear-vacuous mixture model introduced in the previous section influences the performance of the regression methods based on the modified boosting algorithms. Therefore, we conduct the simulations for five different choices of $v$, namely $v \in \{0, 0.25, 0.5, 0.75, 1\}$. Recall that, when $v = 1$ then $\mathcal{P} = S(1, n)$, and thus, we have the standard basic boosting algorithm on the one extreme of the $v$ range, whereas $\mathcal{P} = \{\hat{p}\}$ for $v = 0$, which reduces the modified boosting algorithms to the standard SVR with identical weights of examples. Moreover, we want to obtain some evidence of whether the residual-based measures are in line with the prediction error measures.

In our simulations, we consider two different kinds of data sets. The first is generated according to the following setup. In each of 40 runs, we generate 200 examples $(x_j, y_j) \in \mathbb{R}^2$ for all $j \in \{1, \ldots, 200\}$ by

$$x_j = 0.02(j-1) - 2 \quad \text{and}$$
$$y_j = \exp\left(-x_j^2\right) + 0.5\eta_j,$$

where $\eta_j$ is a random number drawn from the uniform distribution on the interval $[-1, 1]$. Similar data sets were formerly used [10]. From these 200 data points, we randomly draw a training data set and a distinct test data set. The number of training examples is $n = 10$ and the number of testing examples $n_{test} = 60$. We then apply one of the proposed regression methods to the training data and obtain an estimate $\hat{f}$ of the function $f$. Here, we set the number of iterations in the boosting algorithms to $T = 20$ and consider $v \in \{0, 0.25, 0.5, 0.75, 1\}$. Finally, we compute the performance measures.

As a variant of this synthetic data set, we consider also an asymmetric noise. In contrast to the above case, we here generate the random errors according to the following rule: for all $j \in \{1, \ldots, 200\}$, we draw a random number $a_j \in [0, 1]$, then $\eta_j$ is drawn from $[-1, 1]$ if $a_j < 0.7$ and from the interval $[0, 4]$ if $a_j \geqslant 0.7$. All random numbers are generated according to uniform distributions on the corresponding intervals.

Table 1 shows the performance measures obtained by the modified AdaBoost.R2 algorithm for different values of the parameter $v$. The results indicate that the proposed regression method achieves the best results when the linear-vacuous model with $v = 0.5$ is used to restrict the set of weighting probabilities. Table 2 shows the results for the data set with

**Table 1**
Performance of the modification of AdaBoost.R2 for different values of $v$.

| $v$ | *RMSR* | *RMSPE* | *MAR* | *MAPE* |
|---|---|---|---|---|
| 0.0 | 0.456 | 0.347 | 0.379 | 0.298 |
| 0.25 | 0.415 | 0.296 | 0.346 | 0.251 |
| 0.5 | 0.414 | 0.295 | 0.346 | 0.249 |
| 0.75 | 0.413 | 0.296 | 0.345 | 0.249 |
| 1 | 0.416 | 0.298 | 0.347 | 0.251 |

**Table 2**
Performance of the modification of AdaBoost.R2 for different values of $v$ adding the asymmetric noise.

| $v$ | *RMSR* | *RMSPE* | *MAR* | *MAPE* |
|---|---|---|---|---|
| 0.0 | 0.829 | 0.457 | 0.640 | 0.391 |
| 0.25 | 0.703 | 0.397 | 0.541 | 0.344 |
| 0.5 | 0.713 | 0.399 | 0.548 | 0.345 |
| 0.75 | 0.722 | 0.396 | 0.552 | 0.339 |
| 1 | 0.742 | 0.410 | 0.569 | 0.348 |

**Table 3**
Performance of the modification of AdaBoost.RT for different values of $v$.

| $v$ | *RMSR* | *RMSPE* | *MAR* | *MAPE* |
|---|---|---|---|---|
| 0.0 | 0.456 | 0.347 | 0.379 | 0.298 |
| 0.25 | 0.415 | 0.301 | 0.347 | 0.249 |
| 0.5 | 0.415 | 0.307 | 0.349 | 0.249 |
| 0.75 | 0.424 | 0.317 | 0.356 | 0.258 |
| 1 | 0.437 | 0.337 | 0.367 | 0.273 |

**Table 4**
Performance of the modification of AdaBoost.RT for different values of $v$ adding the asymmetric noise.

| $v$ | *RMSR* | *RMSPE* | *MAR* | *MAPE* |
|---|---|---|---|---|
| 0.0 | 0.829 | 0.457 | 0.640 | 0.391 |
| 0.25 | 0.710 | 0.345 | 0.534 | 0.297 |
| 0.5 | 0.723 | 0.336 | 0.539 | 0.288 |
| 0.75 | 0.723 | 0.356 | 0.542 | 0.307 |
| 1 | 0.727 | 0.377 | 0.551 | 0.329 |

asymmetric errors. Here, according to all four performance measures, the best results are achieved for $v = 0.25$. The additional asymmetric noise produces some outliers which the standard AdaBoost.R2 tends to fit too well, because these points are assigned high weights. As the proposed modification of the algorithm restricts these weights, the problem of over-fitting is avoided.

Let us now analyze the simulation results for the modification of the AdaBoost.RT algorithm. The performance criteria in the case of the symmetric errors are shown in Table 3 and the situation with the additional asymmetric noise is presented in Table 4. For these analyses, we set the threshold to $\tau = 0.5$ and set $l = 2$ (see Algorithm 4). For the modified AdaBoost.RT, all four criteria indicate that the value of $v$ implying the best performance of the algorithm in both error scenarios is $v = 0.25$. Furthermore, we observe that the modification of AdaBoost.RT slightly outperforms the modification of AdaBoost.R2 in the second data settings, while this is reversed for the first.

Hence, in the analyses of the first type of synthetic data, the proposed modifications of the AdaBoost-based algorithms perform better than the original ones, corresponding to $v = 1$, and better than the standard SVR, corresponding to $v = 0$.

In addition to the above analyses, we will consider another synthetic data set, which is the well-known data set Friedman#1 [7]. In each of 40 simulation runs we generate 200 examples of 10 independent variables, which are uniformly distributed in the interval $[0, 1]$. Only 5 of these 10 variables are selected randomly and then used to produce the values of the output variable for all $j \in \{1, \ldots, 200\}$ in the following way:

$$y_j = 10 \sin(\pi x_{j,1} x_{j,2}) + 20(x_{j,3} - 0.5)^2 + 10x_{j,4} + 5x_{j,5} + \eta_j,$$

where $\eta_j$ is a random variable drawn from a standard normal distribution. Here, we used 20 training examples and 40 test examples.

The performance measures obtained for the modification of AdaBoost.R2 are shown in Table 5 and the results for the modification of AdaBoost.RT with $\tau = 0.1$ and $l = 2$ are given in Table 6. Like for the second error scenario of the previously discussed synthetic data, we find that the modification of the AdaBoost.RT algorithm attains slightly lower average errors than the regression method based on the modification of AdaBoost.R2. Here, the value $v = 0.5$ implies the best performance of both generalized algorithms. As before, all four criteria provide the same evidence, as expected with the average absolute errors being generally lower than the corresponding square error measures due to the chosen configuration of SVR.

For all the simulations, we set the maximum number $T$ of iterations of the generalized boosting algorithms to 20. Of course, the performance of the regression methods could depend on this parameter because the more iterations there are the more often the algorithm can adapt the weights. To assess the sensitivity of our findings to our specific choice of $T$,

**Table 5**
Performance of the modification of AdaBoost.R2 for the Friedman#1 data for different values of $v$.

| $v$ | *RMSR* | *RMSPE* | *MAR* | *MAPE* |
|---|---|---|---|---|
| 0.0 | 3.12 | 2.80 | 2.57 | 2.22 |
| 0.25 | 3.08 | 2.74 | 2.53 | 2.13 |
| 0.5 | 3.06 | 2.72 | 2.52 | 2.12 |
| 0.75 | 3.07 | 2.72 | 2.52 | 2.13 |
| 1 | 3.09 | 2.75 | 2.54 | 2.16 |

**Table 6**
Performance of the modification of AdaBoost.RT for the Friedman#1 data for different values of $v$.

| $v$ | *RMSR* | *RMSPE* | *MAR* | *MAPE* |
|---|---|---|---|---|
| 0.0 | 3.12 | 2.80 | 2.57 | 2.22 |
| 0.25 | 3.08 | 2.75 | 2.53 | 2.19 |
| 0.5 | 2.96 | 2.63 | 2.41 | 2.03 |
| 0.75 | 2.98 | 2.63 | 2.44 | 2.05 |
| 1 | 3.07 | 2.71 | 2.47 | 2.11 |

**Table 7**
RMSR for the Friedman#1 data for different $T$.

| $T$ | 20 | 40 | 80 | 160 |
|---|---|---|---|---|
| Standard AdaBoost.R2 | 3.09 | 3.09 | 3.09 | 3.09 |
| Generalized AdaBoost.R2 with $v = 0.5$ | 3.06 | 3.06 | 3.06 | 3.06 |
| Standard AdaBoost.RT | 3.07 | 3.12 | 3.14 | 3.14 |
| Generalized AdaBoost.RT with $v = 0.5$ | 2.96 | 2.95 | 2.94 | 2.94 |

**Table 8**
RMSPE for the Friedman#1 data for different $T$.

| $T$ | 20 | 40 | 80 | 160 |
|---|---|---|---|---|
| Standard AdaBoost.R2 | 2.75 | 2.75 | 2.75 | 2.75 |
| Generalized AdaBoost.R2 with $v = 0.5$ | 2.72 | 2.72 | 2.72 | 2.72 |
| Standard AdaBoost.RT | 2.71 | 2.75 | 2.77 | 2.77 |
| Generalized AdaBoost.RT with $v = 0.5$ | 2.63 | 2.62 | 2.62 | 2.62 |

**Table 9**
MAR for the Friedman#1 data for different $T$.

| $T$ | 20 | 40 | 80 | 160 |
|---|---|---|---|---|
| Standard AdaBoost.R2 | 2.54 | 2.54 | 2.54 | 2.54 |
| Generalized AdaBoost.R2 with $v = 0.5$ | 2.52 | 2.52 | 2.52 | 2.52 |
| Standard AdaBoost.RT | 2.47 | 2.51 | 2.54 | 2.54 |
| Generalized AdaBoost.RT with $v = 0.5$ | 2.41 | 2.41 | 2.41 | 2.41 |

**Table 10**
MAPE for the Friedman#1 data for different $T$.

| $T$ | 20 | 40 | 80 | 160 |
|---|---|---|---|---|
| Standard AdaBoost.R2 | 2.16 | 2.16 | 2.16 | 2.16 |
| Generalized AdaBoost.R2 with $v = 0.5$ | 2.12 | 2.12 | 2.12 | 2.12 |
| Standard AdaBoost.RT | 2.11 | 2.15 | 2.19 | 2.19 |
| Generalized AdaBoost.RT with $v = 0.5$ | 2.03 | 2.02 | 2.01 | 2.01 |

we consider additional simulations for the Friedman#1 data set with different maximum numbers of iterations, namely, $T \in \{20, 40, 80, 160\}$. Thereby, we consider the generalized algorithms for $v = 0.5$ and compare them with their standard AdaBoost counterparts. The results are displayed in Tables 7–10.

We observe that the performance of the generalized algorithms is stable over all the choices of $T$. Moreover, they outperform their standard versions in all cases. For the standard AdaBoost.RT algorithm, it can be observed that the performance even tends to deteriorate as $T$ gets larger. This could be due to the fact that, in contrast to the standard AdaBoost.R2 algorithm, this algorithm does not have a stopping rule.

### 5.2. Real data

In addition to the simulations, we evaluate the performance of the proposed regression methods on the basis of five publicly available data sets from the UCI Machine Learning Repository [1]: Slump Test [28], Concrete [27], Wine Quality [3], Energy Efficiency [21], and Yacht Hydrodynamics [9]. The Slump Test data set contains 103 data points. There are seven input variables characterizing the slump flow of concrete and three output variables in the data set. Here, we use only the third output variable: 28-day compressive strength. In the Concrete data set there are 1030 data points characterizing the concrete compressive strength as a highly nonlinear function of age and ingredients, including cement, blast furnace slag, fly ash, water, etc. There are eight input variables and one output variable, namely the concrete compressive strength. The Wine Quality data set contains 1599 examples of red wine and 4898 examples of white wine, for which eleven physicochemical characteristics such as alcohol content or residual sugar are recorded. The output variable is the sensory quality of the wine measured on a discrete scale between 0 (very bad) and 10 (excellent). Here, we focus on the subset concerning the red wine. In the Energy Efficiency data set, there are 768 data points corresponding to building shapes characterized by eight real-valued features, which are considered to predict the heating and the cooling load, respectively, of the buildings.

**Table 11**
RMSR for the UCI data sets by using AdaBoost.R2 with different $v$.

| $v$ | 0 | 0.25 | 0.5 | 0.75 | 1 |
|---|---|---|---|---|---|
| Slump test | 9.08 | 8.79 | 8.75 | 8.85 | 9.08 |
| Concrete data | 27.0 | 16.7 | 16.6 | 16.8 | 17.0 |
| Wine quality | 0.878 | 0.850 | 0.837 | 0.838 | 0.845 |
| Energy efficiency | 10.40 | 9.90 | 9.88 | 10.10 | 10.10 |
| Yacht hydrodynamics | 16.8 | 16.2 | 15.9 | 16.3 | 16.4 |

**Table 12**
MAR for the UCI data sets by using AdaBoost.R2 with different $v$.

| $v$ | 0 | 0.25 | 0.5 | 0.75 | 1 |
|---|---|---|---|---|---|
| Slump test | 6.37 | 6.38 | 6.39 | 6.37 | 6.37 |
| Concrete data | 13.5 | 13.5 | 13.5 | 13.5 | 13.5 |
| Wine quality | 0.696 | 0.691 | 0.688 | 0.686 | 0.695 |
| Energy efficiency | 8.63 | 8.43 | 8.43 | 8.56 | 8.60 |
| Yacht hydrodynamics | 9.60 | 9.50 | 9.60 | 9.60 | 9.52 |

**Table 13**
RMSR for the UCI data sets by using AdaBoost.RT with different $v$.

| $v$ | $\tau$ | 0 | 0.25 | 0.5 | 0.75 | 1 |
|---|---|---|---|---|---|---|
| Slump test | 0.08 | 9.14 | 8.74 | 8.51 | 8.48 | 8.75 |
| Concrete data | 0.3 | 32.1 | 16.8 | 16.9 | 17.1 | 17.1 |
| Wine quality | 0.05 | 0.831 | 0.829 | 0.826 | 0.829 | 0.829 |
| Energy efficiency | 0.1 | 10.60 | 10.00 | 9.81 | 9.69 | 9.73 |
| Yacht hydrodynamics | 0.025 | 16.8 | 16.7 | 16.5 | 16.2 | 16.3 |

**Table 14**
MAR for the UCI data sets by using AdaBoost.RT with different $v$.

| $v$ | $\tau$ | 0 | 0.25 | 0.5 | 0.75 | 1 |
|---|---|---|---|---|---|---|
| Slump test | 0.08 | 6.37 | 6.34 | 6.46 | 6.76 | 7.36 |
| Concrete data | 0.3 | 13.5 | 13.4 | 13.5 | 13.5 | 13.6 |
| Wine quality | 0.05 | 0.688 | 0.688 | 0.688 | 0.688 | 0.688 |
| Energy efficiency | 0.1 | 8.76 | 8.38 | 8.30 | 8.26 | 8.46 |
| Yacht hydrodynamics | 0.025 | 9.78 | 9.34 | 9.38 | 9.48 | 9.64 |

Here, we consider only the heating load as dependent variable. Finally, the Yacht Hydrodynamics data set contains 308 data points used to predict the hydrodynamic performance of sailing yachts from six features measuring their dimensions and velocity.

We analyze all data sets with the proposed generalized algorithms, again for different choices of $v \in \{0, 0.25, 0.5, 0.75, 1\}$ and with $T = 20$. To evaluate the average residual error measures, RMSR and MAR, we perform a cross-validation with 100 repetitions, where in each run, we randomly select $n = 40$ training data and $n_{test} = 40$ test data. Since the results shown in the previous subsection indicate that all four performance measures point into the same direction, we have no concerns about relying only on the residual error measures here.

The results of the computations for the modified AdaBoost.R2 method are given in Tables 11 and 12, while the performance criteria for the generalized AdaBoost.RT algorithm with $l = 2$ are displayed in Tables 13 and 14.

The obtained figures confirm the results of the simulations and indicate a superior fit of the proposed regression methods for $v \in \{0.25, 0.5, 0.75\}$, the evidence provided by the RMSR being the most clear. Thus, if the mixture probability $v$ is neither too small nor too big, both modified algorithms perform better than their basic counterparts (corresponding to $v = 1$) and standard SVR (corresponding to $v = 0$). As the results of the numerical examples indicate that the value of $v$ does indeed affect the performance of the proposed algorithms, in a practical setting, the choice of this parameter should be made very carefully, e.g., on the basis of a cross-validation scheme.

## 6. Conclusion

We proposed generalizations of two ensemble-based boosting algorithms for regression where the unit simplex for the weights of the instances is restricted to a smaller set of weighting probabilities. This smaller set is obtained by imprecise

statistical models like, e.g. the linear-vacuous mixture model. The modified algorithms are more flexible and tend less to over-fitting. Numerical experiments indicate that both modified algorithms perform better than their standard counterparts, if the restricted set of probabilities is neither too small nor too big. Moreover, we find that among the extreme cases, the standard AdaBoost-based algorithms are always at least as good as standard SVR and often much better.

## Acknowledgements

This work was partly conducted while the second author was visiting the Saint Petersburg State Forest Technical University on a fellowship from the German Academic Exchange Service (DAAD).

## References

[1] K. Bache, M. Lichman, UCI Machine Learning Repository, 2013
[2] P. Bühlmann, T. Hothorn, Boosting algorithms: regularization, prediction and model fitting, Stat. Sci. 22 (4) (2007) 477–505.
[3] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, J. Reis, Modeling wine preferences by data mining from physicochemical properties, Decis. Support Syst. 47 (4) (2009) 547–553.
[4] H. Drucker, Improving regressors using boosting techniques, in: Proc. of the 14th Intenational Conferences on Machine Learning, Morgan Kaufmann, San Francisco, CA, USA, 1997, pp. 107–115.
[5] A.J. Ferreira, M.A.T. Figueiredo, Boosting algorithms: a review of methods, theory, and applications, in: C. Zhang, Y. Ma (Eds.), Ensemble Machine Learning: Methods and Applications, Springer, New York, 2012, pp. 35–85.
[6] Y. Freund, R.E. Schapire, A decision theoretic generalization of on-line learning and an application to boosting, J. Comput. Syst. Sci. 55 (1) (1997) 119–139.
[7] J.H. Friedman, Multivariate adaptive regression splines, Ann. Stat. 19 (1) (1991) 1–82.
[8] J.H. Friedman, Greedy function approximation: a gradient boosting machine, Ann. Stat. 29 (5) (2001) 1189–1232.
[9] J. Gerritsma, R. Onnink, A. Versluis, Geometry, resistance and stability of the delft systematic yacht hull series, Int. Shipbuild. Prog. 28 (1981) 276–297.
[10] P.-Y. Hao, Interval regression analysis using support vector networks, Fuzzy Sets Syst. 60 (2009) 2466–2485.
[11] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning: Data Mining, Inference and Prediction, Springer, New York, 2001.
[12] N.L. Johnson, F. Leone, Statistics and Experimental Design in Engineering and the Physical Sciences, vol. 1, Wiley, New York, 1964.
[13] B. Kegl, Robust regression by boosting the median, in: Learning Theory and Kernel Machines, Lecture Notes in Computer Science, vol. 2777, Springer, Berlin Heidelberg, 2003, pp. 258–272.
[14] L.I. Kuncheva, Combining Pattern Classifiers: Methods and Algorithms, Wiley-Interscience, New Jersey, 2004.
[15] J. Mendes-Moreira, C. Soares, A.M. Jorge, J.F. de Sousa, Ensemble approaches for regression: a survey, ACM Comput. Surv. 45 (1) (2012) 1–40.
[16] L. Rokach, Ensemble-based classifiers, Artif. Intell. Rev. 33 (1–2) (2010) 1–39.
[17] D.L. Shrestha, D.P. Solomatine, Experiments with AdaBoost.RT, an improved boosting scheme for regression, Neural Comput. 18 (7) (2006) 1678–1710.
[18] A.J. Smola, B. Schölkopf, A tutorial on support vector regression, Stat. Comput. 14 (2004) 199–222.
[19] D.P. Solomatine, D.L. Shrestha, AdaBoost.RT: a boosting algorithm for regression problems, in: Proc. of the International Joint Conference on Neural Networks, Budapest, Hungary, 2004, pp. 1163–1168.
[20] I. Steinwart, A. Christmann, Support Vector Machines, Springer, 2008.
[21] A. Tsanas, A. Xifara, Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools, Energy Build. 49 (2012) 560–567.
[22] L.V. Utkin, A framework for imprecise robust one-class classification models, Int. J. Mach. Learn. Cybernet. 5 (3) (2012) 379–393.
[23] L.V. Utkin, F.P.A. Coolen, On reliability growth models using Kolmogorov–Smirnov bounds, Int. J. Performability Eng. 7 (1) (2011) 5–19.
[24] V. Vapnik, Statistical Learning Theory, Wiley, New York, 1998.
[25] P. Walley, Statistical Reasoning with Imprecise Probabilities, Chapman and Hall, London, 1991.
[26] L. Wasserman, All of Nonparametric Statistics, Springer, New York, 2006.
[27] I.-C. Yeh, Modeling of strength of high performance concrete using artificial neural networks, Cem. Concr. Compos. 28 (12) (1998) 1797–1808.
[28] I.-C. Yeh, Modeling slump flow of concrete using second-order regressions and artificial neural networks, Cem. Concr. Compos. 29 (6) (2007) 474–480.