An imprecise boosting-like approach to regression

Lev V. Utkin
Department of Control, Automation, and System Analysis, Saint Petersburg State Forest Technical University
lev.utkin@mail.ru

Andrea Wiencierz
Department of Statistics, LMU Munich
andrea.wiencierz@stat.uni-muenchen.de

Abstract

This paper is about a generalization of ensemble methods for regression which are based on variants of the basic AdaBoost algorithm. 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. The proposed algorithms cover the standard AdaBoost-based regression algorithms and standard regression as special cases. Various imprecise statistical models can be used to obtain the restricted set of probabilities. One 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. Finally, some simulations and applications also indicate a better performance of the proposed generalized methods.

Keywords. Regression, AdaBoost, algorithm, linear-vacuous mixture model, Kolmogorov–Smirnov bounds.

1 Introduction

Regression modeling is one of the main problems in applied statistics. Roughly speaking, the aim is to estimate a function $f : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X} \subset \mathbb{R}^m$ with $m \in \mathbb{N}$ and $\mathcal{Y} \subset \mathbb{R}$, from a finite set of noisy samples $(x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{X} \times \mathcal{Y}$ for some $n \in \mathbb{N}$. 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 (see, e.g., [9, 18, 21]). In practice, the estimated function is obtained by minimizing the so-called empirical risk (possibly regularized), 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. 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 $S(1, n)$. In this paper, we focus on this kind of regression methods within the proposed algorithms, because it is very easy to incorporate individual weights for the instances, which is a core element of the algorithms we want to generalize. The 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 than the empirical distribution $\hat{p}$.

A very popular approach to regression 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 [5]. 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. Thus, 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. Detailed reviews of boosting methods can be found, e.g., in [1, 3, 12, 14].

One of the first boosting algorithms for regression is the so-called AdaBoost.R2 proposed in [2], 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 have been developed, a recent survey is provided in [13]. In contrast to most of the ensemble-based algorithms using the weighted average of base regression estimates as their final regression functions, [11] analyzed the choice of the weighted median and proposed the corresponding algorithm called MedBoost. The author proved boosting-type convergence of the algorithm and gave clear conditions for the convergence of the robust training error. Another interesting boosting scheme for regression problems is proposed in [17], 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 AdaBoost algorithm for classification. This adaptation of the AdaBoost algorithm is called AdaBoost.RT and its properties were further investigated in [15].

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. The adapted weighting probabilities may be arbitrary points in the unit simplex. This can lead to over-fitting, when too large weights are assigned to a few hard-to-learn examples. There are different approaches to deal with this problem. One way of overcoming the problem of over-fitting in the context of regression is the so-called shrinkage regularization, where the weights of the base regression estimates are reduced, and thus, the learning rate of the boosting algorithm (see, e.g., [7]). Another interesting approach is based on restricting the weights, e.g., by fixing a maximum size of the weights a priori. In this paper, we follow this idea but we propose to use imprecise statistical models like the linear-vacuous mixture model or the 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 which avoid the problem of over-fitting.

In the following two sections, we propose the corresponding modifications of two popular boosting algorithms: AdaBoost.R2 introduced in [2] and AdaBoost.RT proposed in [17]. Section 4 reviews suitable imprecise probability models to obtain the restricted set of weighting probabilities. Finally, we present the results of simulations based on synthetic data and on real data.

2 AdaBoost.R2 and its modification

At first, we modify the AdaBoost.R2 algorithm proposed in [2]. The scheme of this boosting algorithm for regression is presented as Algorithm 1. Given a

Algorithm 1 AdaBoost.R2

```
Require: Maximum number of iterations $T$ and training data set $Z$.
Ensure: $\alpha^{(t)}$ and $\hat{f}^{(t)}$ for all $t \in \{1,\ldots,T\}$;
  set $t \leftarrow 1$ and $p^{(t)} \leftarrow (n^{-1},\ldots,n^{-1})$;
  repeat
    estimate $\hat{f}^{(t)}$ using weighting probabilities $p^{(t)}$;
    compute $D^{(t)} \leftarrow \max_{j\in\{1,\ldots,n\}} |y_j - \hat{f}^{(t)}(x_j)|$;
    compute normalized errors for all $i \in \{1,\ldots,n\}$:
    $\hat{e}_i^{(t)} \leftarrow \frac{|y_i - \hat{f}^{(t)}(x_i)|}{D^{(t)}}$;
    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 $i \in \{1,\ldots,n\}$:
    $p_i^{(t+1)} \leftarrow p_i^{(t)} \exp\left(-\alpha^{(t)}(1-\hat{e}_i^{(t)})\right)$;
    normalize $p^{(t+1)}$ to be a proper distribution;
    $t++$
  until $t > T$
  normalize $\alpha^{(1)},\ldots,\alpha^{(T)}$ such that $\sum_{t=1}^{T} \alpha^{(t)} = 1$;
  compute regression function $\hat{f} \leftarrow \sum_{t=1}^{T} \alpha^{(t)} \hat{f}^{(t)}$.
```

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 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 $\epsilon^{(t)} \leq 0.5$, 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:

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

for all $i \in \{1,\ldots,n\}$. 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 $(p_1^{(t+1)},\ldots,p_n^{(t+1)})$ to obtain a proper weighting distribution where $\sum_{i=1}^n p_i^{(t+1)} = 1$. Finally, when the loop is ended, the $\alpha^{(1)},\ldots,\alpha^{(T)}$ 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 [5], 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 the other examples. This property is called over-fitting, because the out-of-sample prediction performance of such a regression estimate may be very bad, as the actual functional relation is better reflected by the neglected examples.

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, i.e., it is produced by finitely many linear constraints. This implies that it is totally defined by its extreme points $q^{(k)} = (q_1^{(k)},\ldots,q_n^{(k)})$ for all $k \in \{1,\ldots,r\}$ 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 $\lambda = (\lambda_1,\ldots,\lambda_r)$ 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 λ 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 λ. For the weights $\lambda_1,\ldots,\lambda_r$ 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 $\lambda^{(1)} = (r^{-1},\ldots,r^{-1})$. (Alternatively, the vector $\lambda^{(1)}$ could be randomly selected from $S(1,r)$). When

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 $D^{(t)} \leftarrow \max_{j\in\{1,\ldots,n\}} |y_j - \hat{f}^{(t)}(x_j)|$;
    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 $k \in \{1,\ldots,r\}$:
    $\lambda_k^{(t+1)} \leftarrow \lambda_k^{(t)} \exp\left(-\alpha^{(t)}(1-\hat{\varepsilon}_k^{(t)})\right)$;
    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 $\hat{f} \leftarrow \sum_{t=1}^T \alpha^{(t)} \hat{f}^{(t)}$.
```

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 $r \geq n$ virtual data points (i.e., the extreme points) with associated residuals $\hat{\varepsilon}_k^{(t)}$ and weights $\lambda_k^{(t)}$ for all $k \in \{1, \ldots, r\}$.

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

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

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, $p^{(t)} = (\lambda_1^{(t)}, \ldots, \lambda_n^{(t)})$ 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 $k \in \{1, \ldots, n\}$:

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

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 Bayesian updating.

3 Threshold AdaBoost algorithm and its modification

In this section, we consider the AdaBoost.RT algorithm introduced in [15]. 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.

In contrast to the normalized absolute residuals of AdaBoost.R2, here the regression errors $\hat{e}_1^{(t)}, \ldots, \hat{e}_n^{(t)}$ are given by the absolute values of the relative residuals, for each $t \in \{1, \ldots, T\}$. These residuals are compared to a threshold value $\tau \in \mathbb{R}_{\geq 0}$. The corresponding examples are considered as misclassified if their residual exceeds τ 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 τ. The weights associated with examples with small relative residuals are decreased, while those of the examples considered as misclassified remain constant. By normalizing $p_1^{(t+1)}, \ldots, p_n^{(t+1)}$ 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 $\beta^{(t)} = \left(\epsilon^{(t)}\right)^l$ for some $l \in \mathbb{N}$, and thus, $\alpha^{(t)} = -\ln\left(\beta^{(t)}\right)$ will be very small com-

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 $t \in \{1, \dots, T\}$;
  set $t \leftarrow 1$ and $p^{(t)} \leftarrow (n^{-1}, \dots, n^{-1})$;
  repeat
    estimate $\hat{f}^{(t)}$ using weighting probabilities $p^{(t)}$;
    compute relative errors for all $i \in \{1, \dots, n\}$:
     $\hat{e}_i^{(t)} \leftarrow \left| \frac{y_i - \hat{f}^{(t)}(x_i)}{y_i} \right|$;
    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 $i \in \{1, \dots, n\}$ by:
    if $\hat{e}_i^{(t)} \le \tau$ 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 distribution;
    $t{+}{+}$
  until $t > T$
  normalize $\alpha^{(1)}, \dots, \alpha^{(T)}$ such that $\sum_{t=1}^T \alpha^{(t)} = 1$;
  compute regression function $\hat{f} \leftarrow \sum_{t=1}^T \alpha^{(t)} \hat{f}^{(t)}$.
```

pared to better estimates of other iterations. In [15] 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.

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 τ. If τ 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 overfitting 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 $q^{(k)} = (q_1^{(k)}, \dots, q_n^{(k)})$ with $k \in \{1, \dots, r\}$. 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 λ_k for all $k \in \{1, \dots, r\}$. Then, the overall error rates $\epsilon^{(t)}$ are obtained as $\sum_{\{k:\hat{\varepsilon}_k^{(t)} > \tau\}} \lambda_k^{(t)}$.

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 $\hat{\varepsilon}_k^{(t)} \le \tau$ 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 $\hat{f} \leftarrow \sum_{t=1}^T \alpha^{(t)} \hat{f}^{(t)}$.
```

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 $t \in \{1, \dots, T\}$ we obtain $p^{(t)} = (\lambda_1^{(t)}, \dots, \lambda_n^{(t)})$ and $\lambda_k^{(t+1)} = p_k^{(t+1)}$ for all $k \in \{1, \dots, n\}$, 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 proposed modification.

4 Imprecise statistical models

In this section, we briefly review a selection of imprecise statistical models which can be used to determine the set of weighting probabilities $\mathcal{P} \subset S(1, n)$. In particular, we consider two 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 common parameter ν, which in some cases can be interpreted as the (subjective) probability that the elicited probability distribution p is incorrect. Further interpretations of the parameter ν are, for example, as the size of possible errors in p or the amount of information on which the model is based.

4.1 The linear-vacuous mixture model

The linear-vacuous mixture or imprecise ν-contaminated models produce the set $\mathcal{P}(\nu, p)$ of probabilities $\pi = (\pi_1, \ldots, \pi_n)$ such that $\pi_i = (1-\nu)p_i + \nu b_i$ for some $\nu \in [0, 1]$ 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 $S(1, n)$. The set $\mathcal{P}(\nu, p)$ is a convex subset of the unit simplex; it coincides with $S(1, n)$ when $\nu = 1$, while $\mathcal{P}(\nu, p) = \{p\}$ if $\nu = 0$. For $p = \hat{p} = (n^{-1}, \ldots, n^{-1})$, the set $\mathcal{P}(\nu, \hat{p})$ 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 $(1-\nu)n^{-1} + \nu$ and the other $n - 1$ elements are equal to $(1-\nu)n^{-1}$. For example, the extreme point q_2 is given by the vector

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

4.2 The pari-mutuel model

Another imprecise neighborhood model is the imprecise pari-mutuel model [22, Subsection 3.3.5], for which the set of probability distributions is defined as

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

where $\nu \in [0, +\infty)$ and $p = (p_1, \ldots, p_n)$ is the elicited distribution. The set $\mathcal{P}(\nu, p)$ consists of all probability distributions π such the weighting probabilities of the points do not exceed a constant multiple of the probabilities given by the distribution p. The set $\mathcal{P}(\nu, p)$ can also be obtained from $\mathcal{P}_P(\nu, p)$ by reflecting $\mathcal{P}(\nu, p)$ about the point p. Lower and upper probabilities of the i-th point are given by $\max\{0, (1+\nu)p_i - \nu\}$ and $\min\{(1+\nu)p_i, 1\}$, respectively. The difference between the upper and lower probabilities is ν, as long as p_i is far enough from 0 or 1.

When we consider the empirical distribution $\hat{p} = (n^{-1}, \ldots, n^{-1})$ as the elicited distribution, the extreme points of the set $\mathcal{P}_P(\nu, \hat{p})$ depend on the chosen parameter ν 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 $\mathcal{P}_P(\nu, \hat{p})$ 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 $\nu \in [0, +\infty)$.*

(1) If $\nu \leq (n-1)^{-1}$, then the set $\mathcal{P}_P(\nu, \hat{p})$ 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 $(1+\nu)n^{-1} - \nu$ and the other $n-1$ elements are equal to $(1+\nu)n^{-1}$.

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

$$\frac{1}{n-s+1} \leq \frac{1+\nu}{n} \leq \frac{1}{n-s}.$$

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

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

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 [19, Propositions 1 and 3].

4.3 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 $z_1, \ldots, z_n$ with $n \in \mathbb{N}$ and $\hat{F}_n$ their empirical cumulative distribution function. The Kolmogorov–Smirnov test is a nonparametric test for the null hypothesis that $z_1, \ldots, z_n$ have been generated by some particular distribution F_0. As test statistic the supremum vertical distance of $\hat{F}_n$ 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 $\gamma \geq P_{F_0}(||\hat{F}_n - F_0||_\infty > k_{n,1-\gamma})$ 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 $z_{(1)} \leq z_{(2)} \leq \ldots \leq z_{(n)}$, 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 γ, we obtain a $1-\gamma$ confidence band for F which is of the form

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

with

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

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 have been 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 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 [23, Section 2] and [10, Subsection 8.9.3] for more details.

It has been shown in [20] that it is possible to derive a set of probability mass functions $\mathcal{P}_K(\gamma)$ corresponding to the confidence band for the cumulative distribution function of the type $\mathcal{C}_{1-\gamma}$. 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

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

Every extreme point has $s-1$ elements of size 0, one element with 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, see, e.g., [16, 18]) 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. We make different simulations to study the impact of the choice of ν on the performance of the proposed regression methods, before we apply them to analyze two data sets from the UCI Machine Learning Repository [4].

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 two measures: the square roots of the Mean Square Prediction Error (RMSPE) and of the average Residual Sum of Squares (RRSS), which are defined by

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

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. As usual, the RMSPE and RRSS values are finally averaged over the simulation runs. The smaller the values of these average error measures are, the better the corresponding regression method. Regarding the numerical examples analyzing real data the RMSPE cannot be computed, because the true function f is unknown. Hence, in this case, we only compare the overall RRSS obtained from repeatedly drawing training and test data sets. Furthermore, since the main purpose of the numerical examples is to show the application of the methods to simple and illustrative problems, the hyperparameters are not optimized.

5.1 Synthetic data

The aim of analyzing synthetic data is to investigate how the parameter ν 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 ν, namely $\nu \in \{0, 0.25, 0.5, 0.75, 1\}$. Recall that, when $\nu = 1$ then $\mathcal{P} = S(1,n)$, and thus, we have the standard basic boosting algorithm on the one extreme of the ν range, whereas $\mathcal{P} = \{\hat{p}\}$ for $\nu = 0$, which reduces the modified boosting algorithms to the standard SVR with identical weights of examples.

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

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

where η_j is a random number drawn from the uniform distribution on the interval $[-1, 1]$. Similar data

Table 1: Performance of the modification of AdaBoost.R2 by different ν

ν	$RRSS$	$RMSPE$
0.0	0.456	0.344
0.25	0.428	0.311
0.5	0.427	0.31
0.75	0.435	0.32
1	0.443	0.329

Table 2: Performance of the modification of AdaBoost.R2 by different ν adding the asymmetric noise

ν	$RRSS$	$RMSPE$
0.0	0.835	0.437
0.25	0.704	0.373
0.5	0.717	0.378
0.75	0.726	0.374
1	0.749	0.39

sets have been used, e.g., in [8]. 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 $\nu \in \{0, 0.25, 0.5, 0.75, 1\}$. 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 $j \in \{1, \dots, 200\}$, we draw a random number $a_j \in [0, 1]$, then η_j is drawn from $[-1, 1]$ if $a_j < 0.7$ and from the interval $[0, 4]$ if $a_j \geq 0.7$. All random numbers are generated according to uniform distributions on the corresponding intervals.

Table 1 shows the performance measures $RRSS$ and $RMSPE$ for the modified AdaBoost.R2 algorithm by different values of the parameter ν. We observe that the proposed regression method achieves the best results when the linear-vacuous model with $\nu = 0.5$ is used to restrict the set of weighting probabilities. Table 2 shows the results for the data set with asymmetric errors. Here, the best results are achieved for $\nu = 0.25$. The additional asymmetric noise produces some kind of 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 AdaBoost.RT algorithm. By using the same initial data as for the modification of AdaBoost.R2, we get the performance measures for the same scenarios. The case of the symmetric errors is shown in Table 3 and the situation with the additional asymmetric noise is presented in Table 4. For the analyses, we set the threshold to $\tau = 0.5$ and $l = 2$ (see Algorithm 4). Also for the modified AdaBoost.RT, the values of ν implying the best performance of the algorithm in the first and second error scenarios are $\nu = 0.5$ and $\nu = 0.25$, respectively. Furthermore, we observe that the modification of AdaBoost.RT slightly outperforms the modification of AdaBoost.R2 in both data settings.

Table 3: Performance of the modification of AdaBoost.RT by different ν

ν	$RRSS$	$RMSPE$
0.0	0.465	0.364
0.25	0.409	0.295
0.5	0.408	0.288
0.75	0.411	0.293
1	0.424	0.311

Table 4: Performance of the modification of AdaBoost.RT by different ν adding the asymmetric noise

ν	$RRSS$	$RMSPE$
0.0	0.837	0.458
0.25	0.71	0.345
0.5	0.723	0.336
0.75	0.723	0.356
1	0.727	0.377

Hence, in the analyses of the first synthetic data set, the proposed modifications of the AdaBoost-based algorithms perform better than the original ones, corresponding to $\nu = 1$, and better than the standard SVR, corresponding to $\nu = 0$. In addition to the above analyses, we will consider another synthetic data set, which is the well-known data set Friedman#1 [6]. 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 $j \in \{1, \dots, 200\}$ 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 η_j is a random variable drawn from a standard normal distribution. Here, we used 20 training examples and 40 test examples.

The RRSS and RMSPE measures obtained by using the modification of AdaBoost.R2 are shown in

Table 5: Performance of the modification of AdaBoost.R2 for Friedman#1 data by different ν

ν	$RRSS$	$RMSPE$
0.0	3.12	2.8
0.25	3.08	2.74
0.5	3.06	2.72
0.75	3.07	2.723
1	3.09	2.75

Table 6: Performance of the modification of AdaBoost.RT for Friedman#1 data by different ν

ν	$RRSS$	$RMSPE$
0.0	3.12	2.8
0.25	3.08	2.75
0.5	2.96	2.63
0.75	2.98	2.63
1	3.07	2.71

Table 5 and the results for the modification of AdaBoost.RT with $\tau = 0.1$ and $l = 2$ are given in Table 6. Here again, 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. The value $\nu = 0.5$ implies the best performance of both generalized algorithms, but here we observe that they outperform standard boosting only with higher values of ν.

5.2 Real data

In addition to the simulations, we evaluate the performance of the proposed regression methods by analyzing two publicly available data sets from the UCI Machine Learning Repository [4]: Slump Test [25], and Concrete [24]. 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 1 030 data points characterizing the concrete compressive strength as a highly nonlinear function of age and ingredients which include cement, blast furnace slag, fly ash, water, etc. There are eight input variables and one output variable, namely the concrete compressive strength.

We analyze both data sets with the proposed algorithms, again for different choices of $\nu \in \{0, 0.25, 0.5, 0.75, 1\}$ and with $T = 20$. To evaluate the average residual error measure RRSS we perform a cross-validation with 40 repetitions, where in each run, we randomly select $n = 40$ training data and $n_{test} = 40$ test data. The results of the computations are given in Table 7 for the modified AdaBoost.R2 method and in Table 8 those for the generalized AdaBoost.RT algorithm with $l = 2$. The obtained figures confirm the results of the simulations and indicate a superior fit of the proposed regression methods for $\nu \in \{0.25, 0.5, 0.75\}$. Thus, if the mixture probability ν is neither too small nor too big both modified algorithms perform better in terms of RRSS than their basic counterparts, which correspond to $\nu = 1$.

Table 7: RRSS for the UCI data sets by using AdaBoost.R2 with different ν

ν	0	0.25	0.5	0.75	1
Slump Test	9.08	8.79	8.75	8.85	9.08
Concrete	27	16.7	16.6	16.8	17

Table 8: RRSS for the UCI data sets by using AdaBoost.RT with different ν

ν	τ	0	0.25	0.5	0.75	1
Slump Test	0.08	9.14	8.74	8.51	8.48	8.75
Concrete	0.3	32.1	16.8	16.9	17.1	17.1

The results of the numerical examples indicate that the value of ν does indeed affect the performance of the proposed algorithms. Hence, 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 the 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 further indicate that among the extreme cases (recall that for $\nu = 0$ it corresponds to standard SVR and for $\nu = 1$ to the basic boosting algorithm), the standard AdaBoost algorithms are always at least as good as the standard SVR and often much better. Moreover, we found that both modified algorithms perform better than their standard counterparts, if the mixture probability ν is neither too small nor too big. The core idea behind the presented modifications could be adapted to deal with imprecise data, too, as the imprecise data induce a set of compatible probability distributions.

Acknowledgments

This work was 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] P. Bühlmann and T. Hothorn. Boosting Algorithms: Regularization, Prediction and Model Fitting. *Statistical Science*, 22(4):477–505, 2007.

[2] H. Drucker. Improving regressors using boosting techniques. In *Proc. of the 14th Intenational Conferences on Machine Learning*, pages 107–115, San Francisco, CA, USA, 1997. Morgan Kaufmann.

[3] A.J. Ferreira and M.A.T. Figueiredo. Boosting algorithms: A review of methods, theory, and applications. In C. Zhang and Y. Ma, editors, *Ensemble Machine Learning: Methods and Applications*, pages 35–85. Springer, New York, 2012.

[4] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[5] Y. Freund and R.E. Schapire. A decision theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[6] J.H. Friedman. Multivariate adaptive regression splines. *The Annals of Statistics*, 19(1):1–82, 1991.

[7] J.H. Friedman. Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

[8] P.-Y. Hao. Interval regression analysis using support vector networks. *Fuzzy Sets and Systems*, 60:2466–2485, 2009.

[9] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference and Prediction.* Springer, New York, 2001.

[10] N.L. Johnson and F. Leone. *Statistics and experimental design in engineering and the physical sciences*, volume 1. Wiley, New York, 1964.

[11] B. Kegl. Robust regression by boosting the median. In *Learning Theory and Kernel Machines. Lecture Notes in Computer Science*, volume 2777, pages 258–272. Springer, Berlin Heidelberg, 2003.

[12] L.I. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms.* Wiley-Interscience, New Jersey, 2004.

[13] J.M. Moreira, C. Soares, A.M. Jorge, and J.F. de Sousa. Ensemble approaches for regression: A survey. *ACM Computing Surveys*, 45(1):1–40, 2012.

[14] L. Rokach. Ensemble-based classifiers. *Artificial Intelligence Review*, 33(1-2):1–39, 2010.

[15] D.L. Shrestha and D.P. Solomatine. Experiments with AdaBoost.RT, an improved boosting scheme for regression. *Neural Computation*, 18(7):1678–1710, 2006.

[16] A.J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14:199–222, 2004.

[17] D.P. Solomatine and D.L. Shrestha. AdaBoost.RT: a boosting algorithm for regression problems. In *Proc. of the International Joint Conference on Neural Networks*, pages 1163–1168, Budapest, Hungary, 2004.

[18] I. Steinwart and A. Christmann. *Support Vector Machines.* Springer, 2008.

[19] L.V. Utkin. A framework for imprecise robust one-class classification models. *International Journal of Machine Learning and Cybernetics*, 2012. To appear.

[20] L.V. Utkin and F.P.A. Coolen. On reliability growth models using Kolmogorov-Smirnov bounds. *International Journal of Performability Engineering*, 7(1):5–19, 2011.

[21] V. Vapnik. *Statistical Learning Theory.* Wiley, New York, 1998.

[22] P. Walley. *Statistical Reasoning with Imprecise Probabilities.* Chapman and Hall, London, 1991.

[23] L. Wasserman. *All of Nonparametric Statistics.* Springer, New York, 2006.

[24] I-Cheng Yeh. Modeling of strength of high performance concrete using artificial neural networks. *Cement and Concrete Composites*, 28(12):1797–1808, 1998.

[25] I-Cheng Yeh. Modeling slump flow of concrete using second-order regressions and artificial neural networks. *Cement and Concrete Composites*, 29(6):474–480, 2007.