Robust boosting classification models with local sets of probability distributions

Lev V. Utkin, Yulia A. Zhuk[a,b]

[a] *Department of Control, Automation and System Analysis, St.Petersburg State Forest Technical University, Russia, lev.utkin@gmail.com*
[b] *Department of Information Technology and Systems, St.Petersburg State Forest Technical University, Russia, zhuk_yua@mail.ru*

Abstract

Robust classification models based on the ensemble methodology are proposed in the paper. The main feature of the models is that the precise vector of weights assigned for examples in the training set at each iteration of boosting is replaced by a local convex set of weight vectors. The minimax strategy is used for building weak classifiers at each iteration. The local sets of weights are constructed by means of imprecise statistical models. The proposed models are called RILBoost (Robust Imprecise Local Boost). Numerical experiments with real data show that the proposed models outperform the standard AdaBoost algorithm for several well-known data sets.

Key words: machine learning, classification, boosting, imprecise model, robust

1. Introduction

The classification problem can be formally written as follows. Given n training data (examples, instances, patterns) $S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_n, y_n)\}$, in which $\mathbf{x}_i \in \mathbb{R}^m$ represents a feature vector involving m features and

$y_i \in \{0, 1, ..., k-1\}$ indices the class of the associated examples, the task of classification is to construct an accurate classifier $c : \mathbb{R}^m \to \{0, 1, ..., k-1\}$ that maximizes the probability that $c(\mathbf{x}) = y_i$ for $i = 1, ..., n$. Generally $\mathbf{x}_i$ may belong to an arbitrary set $\mathcal{X}$, but we consider the special case for simplicity $\mathcal{X} = \mathbb{R}^m$. Moreover, we assume that $k = 2$ and $y_i \in \{-1, 1\}$. The main problem of classification is to find a real valued function $f(\mathbf{x}, \mathbf{w}, b)$ having parameters $\mathbf{w}$ and b such that $\mathbf{w} = (w_1, ..., w_m) \in \mathbb{R}^m$ and $b \in \mathbb{R}$, for example, $f(\mathbf{x}, \mathbf{w}, b) = \langle \mathbf{w}, \mathbf{x} \rangle + b$. Here $\langle \mathbf{w}, \mathbf{x} \rangle$ denotes the dot product of two vectors $\mathbf{w}$ and $\mathbf{x}$. The sign of the function determines the class label prediction or $c(\mathbf{x})$.

A classification problem is usually characterized by an unknown probability distribution $p(\mathbf{x}, y)$ on $\mathbb{R}^n \times \{-1, +1\}$ defined by the training set or examples $\mathbf{x}_i$ and their corresponding class labels y_i. Many classification models accept the uniform distribution $p(\mathbf{x}, y)$ which means that every example in the training set has the probability $1/n$. In particular, the empirical risk functional [1] and the well known support vector machine (SVM) method [2, 3, 4] exploit this assumption. We will denote the probabilities of examples $h = (h_1, ..., h_n)$.

At the same time, many weighted classification models violate this assumption by assigning unequal weights to examples in the training set in accordance with some available additional information. These weights can be regarded as a probability distribution. Of course, the equal weights as well as the arbitrary weights can be viewed as a measure of importance of every example. Many authors adhere to this interpretation of weights. It should be noted however that both the interpretations do not formally contradict

one another.

One of the very popular approaches to classification is the ensemble methodology. The basic idea of classifier ensemble learning is to construct multiple classifiers from the original data and then aggregate their predictions when classifying unknown samples. It is carried out by means of weighing several weak or base classifiers and by combining them in order to obtain a classifier that outperforms every one of them. The improvement in performance arising from ensemble combinations is usually the result of a reduction in variance of the classification error [5]. This occurs because the usual effect of ensemble averaging is to reduce the variance of a set of classifiers.

There are many books and review works devoted to the ensemble methodology and the corresponding algorithms due to the popularity of the approach. One of the first books studying how to combine several classifiers together in order to achieve improved recognition performance was written by Kuncheva [6]. The book explains and analyzes different approaches to ensemble methodology comparatively. A comprehensive and exhaustive recent review of the ensemble-based methods and approaches can be found in the work of Rokach [7]. This review provides a detailed analysis of the ensemble-based methodology and various combination rules for combining weak classifiers. Another very interesting and detailed review is given by Ferreira and Figueiredo [8]. The authors of the review cover a wide range of boosting algorithms recently available. This is the only review where the authors attempt to briefly consider and compare a huge number of modifications of boosting algorithms. Re and Valentini [9] proposed a qualitative comparison and analysis of the ensemble methods and their applicability to

astronomy and astrophysics.

The most well known ensemble-based technique is boosting. Boosting is a method for improving the performance of weak classifiers which are combined into a single composite strong classifier in order to achieve a higher accuracy. One of the most popular boosting methods is AdaBoost (Adaptive Boosting) proposed by Freund and Schapire [10]. AdaBoost is a general purpose boosting algorithm that can be used in conjunction with many other learning algorithms to improve their performance via an iterative process.

As pointed out by Rokach [7], AdaBoost improves the performance accuracy for two main reasons:

1. It generates a final classifier whose misclassification rate can be reduced by combining many classifiers whose misclassification rate may be high.

2. It produces a combined classifier whose variance is significantly lower than the variances produced by the weak base learners.

AdaBoost has many advantages. One of them is that the algorithm is adaptive, i.e., it is able to take advantage of weak hypotheses that are more accurate than it was assumed a priori. Adaptation is carried out by changing the weights of all misclassified and correctly classified examples in accordance with some rule. In fact, the uniform probability distribution (equal wights assigned to all examples) is replaced by another probability distribution in each iteration of the algorithm in order to improve the performance accuracy in the next iteration. The weights of examples in boosting methods can be regarded as a discrete probability distribution (probability mass function) over a training set. The algorithm searches for a suitable probability distribution starting from the uniform distribution. On the one hand, the

adaptation may lead to overfitting in the high-noise regime by assigning too much weights onto a few hard-to-learn examples. This is one of the bottlenecks of AdaBoost and similar ensemble-based classification algorithms.

Many modifications of AdaBoost have been proposed to overcome the problem of overfitting. A very popular technique to prevent overfitting is to stop boosting algorithms after a small number of iterations. There are a lot of papers illustrating the successful early stopping rules, for example, [11, 12, 13]. At the same time, Mease and Wyner [11] show by means of simple and visual examples that the early stopping rule may lead to incorrect classification results.

Another interesting approach to overcome overfitting is to restrict the set of possible weights of examples in training data or their probability distributions according to a certain rule. The restriction can be carried out by means of various ways, for example, by introducing an upper bound for the weights, by generating only smooth distributions, etc. (see [14, 15, 16]). There exist other approaches to the problem of overfitting, see, for example, [17, 18, 19, 20, 21]. They can be viewed as a very small part of many algorithms developed in order to avoid overfitting for noisy data.

A quite different algorithm is proposed in this paper in order to overcome some difficulties of the AdaBoost including the problem of avoiding overfitting. The main idea of the algorithm is to assign to examples not a probability distribution, but a set of probability distributions in each iteration. If we imagine the simplex of all possible weights (probability distributions), then a point within the simplex which is used in the standard AdaBoost in each iteration as a weight vector is transformed to a set of distributions, i.e.,

we get some polyhedron around the point. So, every iteration has a separate polyhedron constructed in accordance with some rule, namely, by using the so-called imprecise statistical models [22], in particular, an extension of the well-known ε-contaminated (robust) model [23].

Let us consider by means of a simple example how the weights are changed after every iteration of the AdaBoost algorithm. Suppose we have a training set consisting of three examples. This implies that we have probability distribution $h = (h_1, h_2, h_3)$ in each iteration of the boosting such that h belongs to the unit simplex. Starting from the point $(1/3, 1/3, 1/3)$, the probability distribution moves within the unit simplex. This is schematically shown in Fig. 1. Since the probabilities tend to concentrate on "hard" examples [10], we could suppose that one of the possible points to which the probability distribution moves is an extreme (corner) point of the simplex depicted in Fig. 1. Indeed, we assign the weight 1 to a "hard" example and weight 0 to other examples. The idea of the proposed model is to replace distribution points by small simplexes which are schematically depicted in Fig. 2. We will return to the pictures when we consider the formal statement of the classification problem.

It is supposed that every distribution within the polyhedron or the simplex can be used as a weight vector for classification. However, we select a single distribution according to the minimax strategy. In order to solve the classification problem under the set of probability distributions we select the "worst" distribution providing the largest value of the expected risk. It corresponds to the minimax (pessimistic) strategy in decision making and can be interpreted as an insurance against the worst case [24]. The minimax

strategy makes the classification problem to be robust.

It should be noted that the proposed algorithm may be efficient when there are available only small training samples. It was pointed out by Hertz et al. [25] that classification on the basis of small training samples is an important problem. The authors of [25] indicated that the successful generalization from a very small number of training data often requires the use of additional available information. As a result, they proposed a boosting algorithm which can learn from very small samples. The authors used 10% of the well known real data sets (Wine, Ionosphere, Breast Cancer) as training samples, i.e., they used 18, 35, 70 training examples, respectively (see Table 1).

The problem of small sample data may arise in the context of object recognition where it is often difficult and expensive to acquire large sets of training examples. Interesting and demonstrative real-world examples in defense of possibility to classify with small samples can be found in [26]. One of the examples is that a young child learns many categories per day, and this does not require a large set of training images for each category. Trying to cope with this problem by means of exploiting additional information, Fei-Fei et al. [26] provided very interesting results. The authors demonstrated that, contrary to intuition, useful aspects of a new object category may be learned from a few training example by using specific prior information.

Areas where often only small samples can be available are genomics and proteomics. It is difficult to find, for instance, DNA micro-arrays with a large number training patterns. This number is usually very small (a few dozen). As noted by Guyin et al. [27], training techniques that use regularization

avoid the problem of overfitting when a decision function constructed on the basis of small training samples will perform poorly on test data.

We use the technique which allows us to relax the condition of strict assigning the weights to examples (which may be biased by small training examples) in each iteration of the boosting algorithm in order to avoid the problem of overfitting.

The idea to use local sets of probability distributions in the form of imprecise statistical models to develop every classifier gives the corresponding name of the whole boosting algorithm: RILBoost (Robust Imprecise Local Boost).

There are several problems which have to be solved by using the above idea for classification. The first problem is how to construct the minimax classifier at each iteration of the boosting algorithm which deals with a set of probability distributions. The second problem is how to update the probability distribution in every iteration.

The first problem is studied in Section 2 in detail. For solving the second problem, two algorithms are proposed. They are considered in Sections 3 and 4. Numerical examples illustrating the proposed algorithms are given in Section 5. In Section 6, concluding remarks are made.

2. Preliminary results: a robust classification model and sets of probability distributions

In this section, we consider one of the robust classification models based on use of sets of probability distributions. This is not a boosting model, but it is a constructive element which has to be described in order to implement

the proposed robust boosting method.

2.1. Two types of robust classification models

Robust models have been exploited in classification problems due to the opportunity to avoid some strong assumptions underlying the standard classification models. There are several different definitions of robustness in literature. An exhaustive laconic review of robust models in machine learning was given in [28]. As pointed out in [28], the use of robust optimization in classification is not new. There are a lot of published results providing various robust classification and regression models (see, for instance, [29, 30, 31, 32, 33, 34, 35, 36]) in which box-type uncertainty sets are considered.

One of the interpretations of robustness stems from perturbations of the training data. In some cases, the training data and the testing data are sampled from different processes, they may be corrupted, they may be obtained by means of unsatisfactory equipment, etc. In order to take into account these factors, many popular robust classification models are based on the assumption that inputs are subject to an additive noise, i.e., $\mathbf{x}_i^* = \mathbf{x}_i + \triangle\mathbf{x}_i$, where noise $\triangle\mathbf{x}_i$ is governed by a certain distribution. The simplest way for dealing with the noise is to consider a simple bounded uncertainty model $\|\triangle\mathbf{x}_i\| \leq \delta_i$ with uniform priors. According to this model, the data is uncertain, specifically, for every i, the i-th "true" data point is only known to belong to the interior of an Euclidian ball of radius δ_i centered at the "nominal" data point $\mathbf{x}_i$. It is often assumed that each point can move around within a box. This model has a very clear intuitive geometric interpretation [37]. The maximally robust classifier in this case is the one that maximizes

the radius of balls, i.e., it corresponds to the largest radius such that the corresponding balls around each data point are still perfectly separated. The choice of the maximally robust classifier is explained by the pessimistic strategy in decision making. Applying the above ideas to the SVM with the hinge loss function provides the following optimization problem (see, for instance, a similar problem for binary classification problem [38]):

$$\min_{\mathbf{w},\xi,b,\triangle\mathbf{x}_i} \left(\frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^{n} \xi_i \right),$$

subject to

$$y_i \left(\langle \mathbf{w}, \phi(\mathbf{x}_i + \triangle\mathbf{x}_i) \rangle + b \right) \geq 1 - \xi_i, \ \xi_i \geq 0,$$

$$\|\triangle\mathbf{x}_i\| \leq \delta_i, \ i = 1, ..., n.$$

Here ϕ is a feature map $\mathcal{X} \to H$ such that the data points are mapped into an alternative higher-dimensional feature space H. In other words, this is a map into an inner product space H such that the inner product in the image of ϕ can be computed by evaluating some simple kernel $K(\mathbf{x}, \mathbf{y}) = (\phi(\mathbf{x}), \phi(\mathbf{y}))$, such as the Gaussian kernel

$$K(\mathbf{x}, \mathbf{y}) = \exp\left(- \|\mathbf{x} - \mathbf{y}\|^2 / \gamma^2 \right).$$

γ is the kernel parameter determining the geometrical structure of the mapped samples in the kernel space. It is pointed out in [39] that the problem of selecting a proper parameter γ is very important in classification. When a very small γ is used ($\gamma \to 0$), then $K(\mathbf{x}, \mathbf{y}) \to 0$ for all $\mathbf{x} \neq \mathbf{y}$ and all mapped samples tend to be orthogonal to each other, despite their class labels. In this case, both between-class and within-class variations are very large. On the other hand, when a very large γ is chosen ($\gamma^2 \to \infty$), then $K(\mathbf{x}, \mathbf{y}) \to 1$

and all mapped samples converge to a single point. This obviously is not desired in a classification task. Therefore, a too large or too small γ will not result in more separable samples in H.

Another interpretation of robustness stems from the literature on robust statistics [23], which studies how an estimator or an algorithm behaves under a small perturbation of the statistics model. There are many reasons for the perturbation of the statistical model. One of the main reasons is a small amount of learning data. Therefore, another class of robust models is based on relaxing strong assumptions about a probability distribution of data points (see, for instance, [32]). We propose a model which can be partially regarded as a special case of these models and it is based on using the framework of *ε-contaminated (robust) models* [23]. They are constructed by eliciting a Bayesian prior distribution $p = (p_1, ..., p_n)$ as an estimate of the true prior distribution. The imprecise ε-contaminated model is a class $\mathcal{M}(p, \varepsilon)$ of probability distributions $h = (h_1, ..., h_n)$ which for fixed $\varepsilon \in (0, 1)$ and p is the set $\mathcal{M}(p, \varepsilon) = \{h_i = (1 - \varepsilon)p_i + \varepsilon q_i\}$, where q_i is arbitrary and $q_1 + ... + q_n = 1$. In other words, there is an unknown precise "true" probability distribution in $\mathcal{M}(p, \varepsilon)$, but we do not know it and only know that it belongs to the set $\mathcal{M}(p, \varepsilon)$. This model is also called the linear-vacuous mixture [22]. The rate ε reflect the amount of uncertainty in p [40]. In other words, we take an arbitrary probability distribution $q = (q_1, ..., q_n)$ from the unit simplex denoted by $S(1, n)$. Of course, the assumption that q is restricted by the unit simplex $S(1, n)$ is one of possible types of ε-contaminated models. Generally, there are a lot of different assumptions which produce specific robust models.

If we have assumed in some robust models [37] that each point from the training set can move around within an Euclidean ball or a box, then the proposed robust model assumes that the probability h_i of each point (but not a data point itself) can move around within a unit simplex under some restrictions. This is the main idea for constructing the robust *local* classification models below.

Most methods for solving classification problems are based on minimizing the expected risk [1], and they differ mainly by loss functions $l(\mathbf{w}, b,\phi(\mathbf{x}))$ which depends on the classification parameters $\mathbf{w}, b$ and the vector $\mathbf{x}$ in the feature space H. The expected risk can be written as follows:

$$R(\mathbf{w},b) = \int_{\mathbb{R}^m} l(\mathbf{w}, b,\phi(\mathbf{x}))\mathrm{d}F(\mathbf{x}).$$

The standard SVM technique is to assume that F is empirical (non-parametric) probability distribution whose use leads to the empirical expected risk [1]

$$R_{\mathrm{emp}}(\mathbf{w},b) = \frac{1}{n}\sum_{i=1}^{n} l(\mathbf{w}, b, \phi(\mathbf{x}_i)). \tag{1}$$

The assumption of the empirical probability distribution means that every point $\mathbf{x}_i$ has the probability $p_i = 1/n$.

Now we have to consider the case when F belongs to a set of probability distributions. In particular, we have to study how to get the expected risk $R(\mathbf{w},b)$ when we have the set $\mathcal{M}(p, \varepsilon)$ instead of a single probability distribution. One of the possible ways for dealing with the expected risk by the set $\mathcal{M}(p, \varepsilon)$ is to use the minimax (pessimistic) strategy. According to the minimax strategy, we select a probability distribution from the set $\mathcal{M}(p, \varepsilon)$ such that the expected risk $R(\mathbf{w},b)$ achieves its maximum for every fixed $\mathbf{w}$. It

should be noted that the "optimal" probability distributions may be different for different values of parameters $\mathbf{w}$. The minimax strategy can be explained in a simple way. We do not know a precise probability distribution and every distribution from $\mathcal{M}(p, \varepsilon)$ can be selected. Therefore, we should take the "worst" distribution providing the largest value of the expected risk. The minimax criterion can be interpreted as an insurance against the worst case because it aims at minimizing the expected loss in the least favorable case ([24]). This criterion of decision making can be regarded as the well-known Γ-minimax[1] ([40, 41, 42]).

Another strategy which can be used for dealing with the set $\mathcal{M}(p, \varepsilon)$ is the minimin strategy. It can be regarded as a direct opposite to the minimax strategy. According to the minimin strategy, the expected risk $R(\mathbf{w},b)$ is minimized over all probability distributions from the set $\mathcal{M}(p, \varepsilon)$ as well as over all values of parameters $\mathbf{w}$, ρ. The strategy can be called optimistic because it selects the "best" probability distribution from the set $\mathcal{M}(p, \varepsilon)$. However, it can not be called robust and, therefore, it is not studied here.

2.2. The set of probabilities and a set of weighted SVMs

Let $h = (h_1, ..., h_n)$ be a probability distribution which belongs to the set $\mathcal{M}(p, \varepsilon)$. The maximum value of the expected risk $R(\mathbf{w},b)$ is

$$\overline{R}(\mathbf{w},b) = \max_{h \in \mathcal{M}(p,\varepsilon)} R(\mathbf{w},b).$$

[1] This criterion is often given in terms of utilities. Therefore, it is usually called Γ-maximin.

The minimax expected risk with respect to the minimax strategy is now of the form:

$$\overline{R}(\mathbf{w}_{\text{opt}},b_{\text{opt}}) = \min_{\mathbf{w},b} \overline{R}(\mathbf{w},b) = \min_{\mathbf{w},b} \max_{h\in\mathcal{M}(p,\varepsilon)} R(\mathbf{w},b).$$

The upper bound for the expected risk can be found as a solution to the following programming problem:

$$\overline{R}(\mathbf{w},b) = \max_{h} \sum_{i=1}^{n} h_i \cdot l(\mathbf{w},b,\phi(\mathbf{x}_i))$$

subject to $h \in \mathcal{M}(p,\varepsilon)$.

The obtained optimization problem is linear with optimization variables $h_1,...,h_n$, but the objective function depends on $\mathbf{w}$. Therefore, it can not be directly solved by well-known methods. In order to overcome this difficulty, note, however, that all points h belong to the convex set $\mathcal{M}$ in a finite dimensional space. According to some general results from linear programming theory, an optimal solution to the above problem is achieved at *extreme points* of the set $\mathcal{M}$ denoted as $\mathcal{E}(\mathcal{M})$, and the number of its extreme points is n. This implies that there holds

$$\overline{R}(\mathbf{w},b) = \max_{k=1,...,n} \sum_{i=1}^{n} h_i^{(k)} \cdot l(\mathbf{w},b,\phi(\mathbf{x}_i)),\ h_i^{(k)} \in \mathcal{E}(\mathcal{M}). \tag{2}$$

By fixing parameters $\mathbf{w}$ and b, we can represent the problem as a set of optimization problems of the form:

$$\overline{R}_k(\mathbf{w},b) = \sum_{i=1}^{n} h_i^{(k)} \cdot l(\mathbf{w},b,\phi(\mathbf{x}_i)),\ h_i^{(k)} \in \mathcal{E}(\mathcal{M}). \tag{3}$$

The optimal values of $\mathbf{w}$ and b correspond to the *largest* value of the objective function $\overline{R}_k(\mathbf{w},b)$, $k = 1,...,n$.

The next task is to minimize the upper expected risk $\overline{R}_k(\mathbf{w},b)$ over the parameters $\mathbf{w}$ and b. This task will be solved in the framework of the SVM. Moreover, it can be solved by means of algorithms available for a weighted SVM method (see, for example, [43]).

First, we write the primal optimization problem taking into account the hinge loss function $l(\mathbf{w},b,\phi(\mathbf{x})) = \max\{0, 1 - y(\langle \mathbf{w}, \phi(\mathbf{x})\rangle + b)\}$. The problem is

$$\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n} h_i^{(k)}\xi_i \to \min_{\mathbf{w},\xi,b}$$

subject to

$$y_i(\langle \mathbf{w},\phi(\mathbf{x}_i)\rangle + b) \geq 1 - \xi_i,\ \xi_i \geq 0,\ i = 1,...,n.$$

Here C is a user-specified positive parameter, which controls the trade-off between classification violation and margin maximization.

Second, by using the well-known methods, we write the dual form (Lagrangian) as

$$L_k(\varphi) = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\varphi_i\varphi_j y_i y_j K(\mathbf{x}_i,\mathbf{x}_j) + \sum_{i=1}^{n}\varphi_i \to \max_{\varphi}$$

subject to

$$0 \leq \varphi_i \leq C \cdot h_i^{(k)},\ i = 1,...,n,\ \sum_{i=1}^{n}\varphi_i y_i = 0.$$

Here $\varphi = (\varphi_1,...,\varphi_n)$ are Lagrange multipliers. The function $f(\mathbf{x},\mathbf{w},b)$ can be rewritten in terms of Lagrange multipliers as

$$f(\mathbf{x},\mathbf{w},b) = \sum_{i=1}^{n}\varphi_i K(\mathbf{x}_i,\mathbf{x}) + b.$$

The weighted SVM differs from the standard SVM by the upper bounds of Lagrange multipliers φ_i in the dual problem. The *largest* value of the

expected risk $\overline{R}_k$ corresponds to the *smallest* value of the Lagrangian. This implies that the optimal values of φ_i, $i = 1, ..., n$, correspond to the *smallest* value of objective function L_k, $k = 1, ..., n$.

2.3. Extreme points of the imprecise contamination model

The above quadratic optimization problems use the extreme points of the set $\mathcal{M}(p, \varepsilon)$. Therefore, our next task is to find these extreme points. It is a very simple task when the set $\mathcal{M}(p, \varepsilon)$ is produced by the introduced imprecise ε-contaminated model. Note that the probability distribution q is restricted by the unit simplex $S(1, n)$. This implies that its extreme points are of the form:

$$(1, 0, ..., 0),\ (0, 1, ..., 0), ...,\ (0, 0, ..., 1).$$

Hence, we can write the k-th extreme point of the set $\mathcal{M}(p, \varepsilon)$ as follows:

$$((1 - \varepsilon)p_1, ..., (1 - \varepsilon)p_k + \varepsilon, ..., (1 - \varepsilon)p_n).$$

One can see that the k-th extreme point consists of $n-1$ elements $(1-\varepsilon)p_i$, $i = 1, ..., n$, $i \neq k$, and the k-th element $(1-\varepsilon)p_k+\varepsilon$. Then the constraints for Lagrange multipliers φ_i in the k-th dual optimization problem $L_k(\varphi)$ become of the form:

$$0 \leq \varphi_i \leq C \cdot (1 - \varepsilon)p_i,\ i = 1, ..., n,\ i \neq k,$$
$$0 \leq \varphi_k \leq C \cdot (1 - \varepsilon)p_k + C \cdot \varepsilon.$$

The same optimization problems for one-class classification by various imprecise models have been proposed by Utkin [44], Utkin and Zhuk [45].

3. A general idea of a class of robust boosting-like classifiers

A typical boosting algorithm can be represented as Algorithm 1. It consists of the following steps which distinguish the boosting classifiers from other ones. First, the classification is carried out by means of a number (say, T) of classifiers c_t, $t = 1, ..., T$. Second, weights of examples in the training set for every classifier are updated in accordance with certain rules taking into account the accuracy of the classifier. Third, various classifiers are combined in order to form a final classifier.

Algorithm 1 A general boosting algorithm

Require: T (number of iterations), (x, y) (training set)

Ensure: Classifiers c_t, errors α_t, $t = 1, ..., T$

Assign an equal weight for examples x

repeat

Build classifier c_t with minimize the weighted error over the examples x

Update the weights h_t of examples x

until $t > T$

The final classifier M^* is formed by a weighted vote of the individual c_t according to its accuracy on the weighted training set

The AdaBoost method (see Algorithm 2) is one of the ensemble-based methods. It can be regarded as a special case of Algorithm 1 with the justified second and third steps. Initially, identical weights $h = (1/n, ..., 1/n)$ are assigned to all examples. In each iteration the weights of all misclassified examples are increased while the weights of correctly classified examples are decreased (see Algorithm 2). As a consequence, the weak classifier is forced

to focus on the difficult examples of the training set by performing additional iterations and creating more classifiers. Furthermore, a weight α_t is assigned to every individual classifier. This weight measures the overall accuracy of the classifier and is a function of the total weight $e(t)$ of the correctly classified examples calculated with respect to the distribution h_t. The weight α_t measures also the importance that is assigned to the classifier c_t. Thus, higher weights are given to more accurate classifiers. These weights are used for the classification of new examples. The distribution h_t is updated using the rule shown in Algorithm 2. The effect of this rule is to increase the weight of examples misclassified by h_t, and to decrease the weight of correctly classified examples. Thus, the weight tends to concentrate on "hard" examples. The final classifier c is a weighted majority vote of the T weak classifiers.

We have mentioned that one of the main reasons for poor classification results of AdaBoost in the high-noise regime is that the algorithm assigns too much weights onto a few hard-to-learn examples. This leads to overfitting.

Let us consider by means of a simple example how the weights are changed after every iteration of the AdaBoost algorithm. Suppose we have a training set consisting of three examples. This implies that we have probability distribution $h = (h_1, h_2, h_3)$ such that h belongs to the unit simplex. Starting from the point $(1/3, 1/3, 1/3)$, the probability distribution moves within the unit simplex. This is schematically shown in Fig. 1. We repeat here the description of Fig. 1 given in the introductory section. Since the probabilities tend to concentrate on "hard" examples [10], we could suppose that one of the possible points to which the probability distribution moves is an extreme point depicted in Fig. 1. Indeed, we assign the weight 1 to a "hard" example

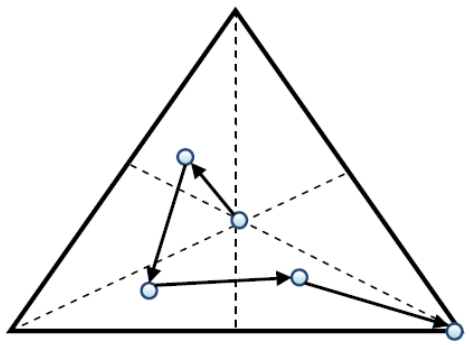

Figure 1: The unit simplex of probabilities of examples in each iterations for the AdaBoost

and weight 0 to other examples.

A main general idea of the proposed algorithms is that we replace the precise probability distribution $h^{(t)}$ or precise weight vector at each iteration by a set of probability distributions denoted $\mathcal{P}(p, \theta)$ defined by a prior probability distribution p and a model parameter θ which determines the set $\mathcal{P}$ and may have different meanings for different imprecise probability models, for instance, it may measure the probability (like the confidence probability) that the distribution p is incorrect or it may be the size of possible errors in p. Some imprecise models which can be represented by p and θ are outlined by Utkin in [44]. Except for the linear-vacuous mixture [22] which has been studied in our work, these are the imprecise pari-mutuel model, the imprecise constant odds-ratio model, bounded vacuous and bounded ε-contaminated robust models, Kolmogorov-Smirnov bounds. It is important to note that the set $\mathcal{P}(p, \theta)$ is assumed to be convex. This implies that it is totally defined by its extreme points. Let $\mathcal{E}(\mathcal{P}(p, \theta))$ be a set of Q extreme points of $\mathcal{P}(p, \theta)$. The set $\mathcal{P}(p, \theta)$ is schematically depicted in Fig. 2 in the form of small tri-

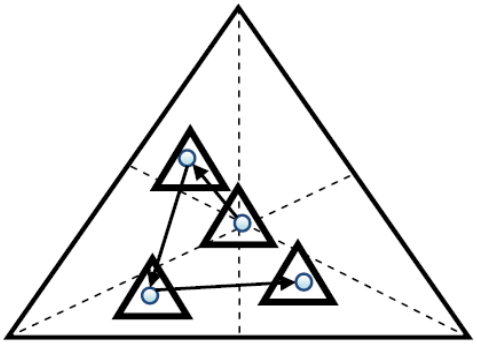

Figure 2: The unit simplex of probabilities and sets of probabilities in the first proposed algorithm

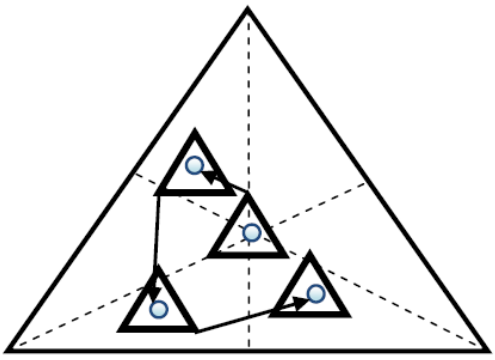

Figure 3: The unit simplex of probabilities and sets of probabilities in the second proposed algorithm

Algorithm 2 The AdaBoost algorithm

Require: T (number of iterations), S (training set)

Ensure: $c_t, \alpha_t,\ t = 1, ..., T$

$t \leftarrow 1$; $h_1(i) \leftarrow 1/n$; $i = 1, ..., n$.

repeat

Build classifier c_t using distribution h_t

$e(t) \leftarrow \sum_{i:c_t(x_i) \neq y_i} h_t(i)$

if $e(t) > 0.5$ **then**

$T \leftarrow t - 1$

exit Loop.

end if

$\alpha_t \leftarrow \frac{1}{2} \ln \left(\frac{1-e(t)}{e(t)} \right)$

$h_{t+1}(i) \leftarrow h_t(i) \cdot \exp\left(-\alpha_t y_t c_t(x_i)\right)$

Normalize h_{t+1} to be a proper distribution.

$t++$

until $t > T$

$c(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{T} \alpha_t c_t(\mathbf{x}) \right)$

angles which are drawn around the points p. Arrows schematically show updating of these probabilities. At every iteration, we solve Q classification problems corresponding to Q extreme points of the set $\mathcal{P}(p, \theta)$. According to some decision rule (for instance, the minimax strategy), we build the optimal classifier and determine the optimal probability distribution or weights of examples. The probabilities or weights of examples are modified by using the function G of both the optimal probability distribution $q^{(t)}(k_0)$ and the "central" probability distribution $p^{(t)}$ of the small simplex.

Algorithm 3 A general boosting algorithm

Require: T (number of iterations), (x, y) (training set), θ (imprecise model parameter)

Ensure: Classifiers c_t, errors α_t, $t = 1, ..., T$

Assign an equal weight for examples x $(p^{(1)} = (1/n, ..., 1/n))$

repeat

Compute Q extreme points $q^{(t)}(k) \in \mathcal{E}(\mathcal{P}(p^{(t)}, \theta))$ of the set $\mathcal{P}(p^{(t)}, \theta)$, $k = 1, ..., Q$

Build classifiers $c_t(k)$ for every probability distribution $q^{(t)}(k)$, $k = 1, ..., Q$

Select a classifier $c_t(k_0)$ maximizing the expected loss and select the corresponding extreme point $q^{(t)}(k_0)$

Update the distribution $p^{(t)}$ as follows: $p^{(t+1)} \leftarrow G(p^{(t)}, q^{(t)}(k_0))$

until $t > T$

The final classifier M^* is formed by a weighted vote of the individual c_t according to its accuracy on the weighted training set

Algorithm 3 outlines a generalized scheme of the robust boosting-like

approach using the imprecise probability model shortly described above. We should select two main distinguished steps. The first one is to compute the extreme points and to use them for classifying. In other words, we build Q classifiers such that their weights are totally defined by extreme points of the set $\mathcal{P}(p^{(t)}, \theta)$. This is a common step for all approaches. The second important step depends on the function G. The function $G(p^{(t)}, q^{(t)}(k_0))$ defines a strategy of the updating procedure and of the whole algorithm of the RILBoost method. Generally, we can write the following linear combination of the distributions $p^{(t)}$ and $q^{(t)}(k_0)$, namely, we could take the following probability distribution for updating: $\alpha \cdot p^{(t)} + (1-\alpha)q^{(t)}(k_0)$. Here $\alpha \in [0, 1]$. In particular, by applying the standard AdaBoost method and its standard updating rule, we get

$$p_i^{(t+1)} = \left(\alpha \cdot p^{(t)} + (1-\alpha)q^{(t)}(k_0)\right) \exp\left(-\alpha_t y_t c_t(x_i)\right). \tag{4}$$

By taking $\alpha = 1$ (see Fig. 2) and $\alpha = 0$ (see Fig. 3), we obtain the RILBoost I and the RILBoost II algorithms, respectively, which will be studied below.

If the training time of the standard AdaBoost for a data set having n examples is τ units of time, then the training time of the proposed algorithms is $n\tau$. The test time of RILBoost I or RILBoost II coincides with the same time of the standard AdaBoost. One can see that the training time is considerably increased with the number of training data. However, the algorithms become to be efficient by the small number of examples. Therefore, this is a condition for their applying.

It should be noted that the well-known boosting algorithms AdaBoost.M1 and AdaBoost.M2 [7] of the multi-class classification can be simply imple-

mented and modified in the framework of Algorithm 3 under condition that the weak classifiers are the multi-class SVM [46] because in this case extreme points can be used for solving the optimization problems. The obtained modifications of AdaBoost.M1 and AdaBoost.M2 differ from the proposed RILBoost algorithms only by updating rules and weak classifiers c_t.

4. Two special robust classification algorithms

4.1. Algorithm I

Let us return to Algorithm 3. We consider only two cases of the linear function G. The first case is $\alpha = 1$ (see Fig. 2) and we get the RILBoost I algorithm. Its main idea is that we replace the precise probability distribution $p_i^{(t)}$ at each iteration by the set of probability distributions $\mathcal{P}(p, \theta) = \mathcal{M}(p^{(t)}, \varepsilon)$ produced by the imprecise contamination model with the predefined robust rate ε. At that, the set $\mathcal{M}(p^{(t)}, \varepsilon)$ is convex and every its element is defined by its extreme points. By using the extreme points $\mathcal{E}(\mathcal{M}(p^{(t)}, \varepsilon))$ of the set $\mathcal{M}(p^{(t)}, \varepsilon)$, we search for optimal values of Lagrange multipliers φ in accordance with the weighted SVM considered above. The error rate is computed by using the probability distribution $p_i^{(t)}$ as $e(t) = \sum_{i:c_t(x_i)\neq y_i} p_i^{(t)}$. The updated probability distribution is determined by using the rule $p_i^{(t+1)} = p_i^{(t)} \cdot \exp\left(-\alpha_t y_t c_t(x_i)\right)$. The main peculiarity of the algorithm is that it uses only "centers" $p^{(t)}$ of sets $\mathcal{M}(p^{(t)}, \varepsilon)$ for updating. Moreover, the error rate $e(t)$ is computed through this distribution $p^{(t)}$ despite the fact that the actual probability distribution leading to the vector of errors is one of the extreme points $\mathcal{E}(\mathcal{M}(p^{(t)}, \varepsilon))$, but not $p^{(t)}$. The above is the main differences between the RILBoost I algorithm and the standard Ad-

aBoost. Other steps of the RILBoost I algorithm and the AdaBoost coincide. Algorithm 4 outlines the main steps of the RILBoost I.

Algorithm 4 The RILBoost I algorithm

Require: T (number of iterations), S (training set), ε (robust rate)

Ensure: $c_t, \alpha_t,\ t = 1, ..., T$

$t \leftarrow 1$; $p_i^{(t)} \leftarrow 1/n$; $i = 1, ..., n$.

repeat

Compute extreme points $\mathcal{E}(\mathcal{M}(p^{(t)}, \varepsilon))$ of the set $\mathcal{M}(p^{(t)}, \varepsilon)$

Build classifier c_t using the set $\mathcal{E}(\mathcal{M}(p^{(t)}, \varepsilon))$

$e(t) \leftarrow \sum_{i:c_t(x_i)\neq y_i} p_i^{(t)}$

if $e(t) > 0.5$ **then**

$T \leftarrow t - 1$

exit Loop.

end if

$\alpha_t \leftarrow \frac{1}{2} \ln\left(\frac{1-e(t)}{e(t)}\right)$

$p_i^{(t+1)} \leftarrow p_i^{(t)} \cdot \exp\left(-\alpha_t y_t c_t(x_i)\right)$

Normalize $p^{(t+1)}$ to be a proper distribution.

$t++$

until $t > T$

$c(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{T} \alpha_t c_t(\mathbf{x})\right)$

4.2. Algorithm II

The second special case is $\alpha = 0$ (see Fig. 3), we obtain the RILBoost II algorithm. Almost all steps of the RILBoost I and II algorithms coincide.

However, the main idea of the second algorithm is that the error rate is computed by using the probability distribution $\pi^{(t)}$ from $\mathcal{E}(\mathcal{M}(p^{(t)}, \varepsilon))$, i.e., there holds $e(t) = \sum_{i:c_t(x_i)\neq y_i} \pi_i^{(t)}$. Moreover, the updated probability distribution is determined by using the rule $p_i^{(t+1)} = \pi_i^{(t)} \cdot \exp\left(-\alpha_t y_t c_t(x_i)\right)$. The main peculiarity of the algorithm is that it uses "optimal" extreme points $\pi^{(t)}$ of sets $\mathcal{M}(p^{(t)}, \varepsilon)$, $t = 1, ..., T$, for building the classifier (see Algorithm 5).

Algorithm 5 The RILBoost II algorithm

Require: T (number of iterations), S (training set), ε (robust rate)

Ensure: $c_t, \alpha_t,\ t = 1, ..., T$

$t \leftarrow 1$; $p_i^{(t)} \leftarrow 1/n$; $i = 1, ..., n$.

repeat

Compute extreme points $\mathcal{E}(\mathcal{M}(p^{(t)}, \varepsilon))$ of the set $\mathcal{M}(p^{(t)}, \varepsilon)$

Build classifier c_t using the set $\mathcal{E}(\mathcal{M}(p^{(t)}, \varepsilon))$

Determine optimal probability distribution $\pi^{(t)}$ from $\mathcal{E}(\mathcal{M}(p^{(t)}, \varepsilon))$

$e(t) \leftarrow \sum_{i:c_t(x_i)\neq y_i} \pi_i^{(t)}$

if $e(t) > 0.5$ **then**

$T \leftarrow t - 1$

exit Loop.

end if

$\alpha_t \leftarrow \frac{1}{2} \ln\left(\frac{1-e(t)}{e(t)}\right)$

$p_i^{(t+1)} \leftarrow \pi_i^{(t)} \cdot \exp\left(-\alpha_t y_t c_t(x_i)\right)$

Normalize $p^{(t+1)}$ to be a proper distribution.

$t++$

until $t > T$

$c(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{T} \alpha_t c_t(\mathbf{x})\right)$

Let us analyze robustness of the proposed models with respect to possible changes and incorrectness of weights in every iteration. Every single classifier is robust from this point of view because changes of weights within the small simplex $\mathcal{M}(p_t, \varepsilon)$ do not impact on the selecting one of the extreme points of $\mathcal{M}(p_t, \varepsilon)$ in the t-th iteration. One can see that the error rate $e(t)$ in every iteration in RILBoost II also does not change with changes of weights within $\mathcal{M}(p_t, \varepsilon)$ because it is determined through the extreme points of $\mathcal{M}(p_t, \varepsilon)$. Therefore, coefficients α_t are also robust. As a result, we get the robust boosting-like algorithm RILBoost II. The algorithm RILBoost I is only partially robust because coefficients α_t may change with changes of the weights within $\mathcal{M}(p_t, \varepsilon)$.

In order to prove that the proposed approach avoids the problem of overfitting, we have to show that the proposed algorithm chooses the extreme points which reduce the large weights or probabilities assigned to hard-to-learn examples. Let $p_r = (p_1(r), ..., p_n(r))$ be the vector of weights of examples before the r-th iteration. Without loss of generality, we assume that the first example is misclassified at this iteration. The contrary case, i.e., the correctly classified example, is not studied because it is one of the outcomes of the next iteration and is considered below. Since we have the misclassified first example, then the boosting updating algorithm has to increase the weight $p_1(r)$ of the first example in order to obviate the classification error. According to the structure of set $\mathcal{E}_r$ of extreme points, we can increase the weight only of one example. This follows from the extreme points $(0, ..., 1_k, ..., 0)$ of the set of probability distributions q in the imprecise ε-contaminated model or the linear-vacuous mixture. After increasing $p_1(r)$,

we have a new weight $h_1(r)$ and two alternatives. The first alternative is that the first example is again misclassified. Hence, the upper bound in (2) is $\overline{R}(w,b) = h_1 \cdot l(w,b,\phi(x_1))$ because the loss function $l(w,b,\phi(x_i))$ is non-zero only for $i = 1$ and h_1 is produced by $q^{(1)} = (1,0,...,,0)$. The index r is omitted here for short. Now the boosting algorithm again increases the weight h_1 in order to try to obviate the classification error. The second alternative is that the first example becomes to be correctly classified, i.e., $l(w,b,\phi(x_1)) = 0$. This implies that the risk measure $R(w,b)$ achieves its largest value $\overline{R}(w,b)$ at another extreme point which is different from the point produced by $q^{(1)} = (1,0,...,,0)$. Hence, the probability $p_1(r)$ is decreased and we avoid the problem of overfitting. The cases of two and larger misclassified errors can be considered in the same way.

5. Numerical experiments

We illustrate the method proposed in this paper via several examples, all computations have been performed using the statistical software R [47]. We investigate the performance of the proposed method and compare it with the standard AdaBoost by considering the accuracy measure (ACC), which is the proportion of correctly classified cases on a sample of data, i.e., ACC is an estimate of a classifier's probability of a correct response. This measure is often used to quantify the predictive performance of classification methods and it is an important statistical measures of the performance of a binary classification test. It can formally be written as $ACC = N_T/N$. Here N_T is the number of test data for which the predicted class for an example coincides with its true class, and N is the total number of test data.

We will denote the accuracy measure for the proposed RILBoost algorithms as $ACC_{\mathrm{S(I)}}$, $ACC_{\mathrm{S(II)}}$ and for the standard AdaBoost as ACC_{st}.

The proposed method has been evaluated and investigated by the following publicly available data sets: Haberman's Survival, Pima Indian Diabetes, Mammographic masses, Parkinsons, Breast Tissue, Indian Liver Patient, Lung cancer, Breast Cancer Wisconsin Original (BCWO), Seeds, Ionosphere, Ecoli, Vertebral Column 2C, Yeast, Blood transfusion, Steel Plates Faults, Planning Relax, Fertility Diagnosis, Wine, Madelon, QSAR biodegradation, Seismic bumps [48], Notterman Carcinoma gene expression data (Notterman CGED) [49]. All data sets except for the Notterman Carcinoma data are from the UCI Machine Learning Repository [48]. The Notterman CGED resource is http://genomics-pubs.princeton.edu/oncology. A brief introduction about these data sets are given in Table 1, while more detailed information can be found from, respectively, the data resources. Table 1 shows the number of features m for the corresponding set and numbers of examples in negative n_0 and positive n_1 classes, respectively

It should be noted that the first two classes (carcinoma, fibro-adenoma) in the Breast Tissue data set are united and regarded as the negative class. Two classes (disk hernia, spondylolisthesis) in the Vertebral Column 2C data set are united and regarded as the negative class. The class CYT (cytosolic or cytoskeletal) in Yeast data set is regarded as negative. Other classes are united as the positive class. The first class in the Seeds data set is regarded as negative. Rosa and Canadian varieties of wheat in the Seeds data set are united. The second and the third classes of Lung cancer data set are united and regarded as the positive class. The largest class "cytoplasm"

(143 examples) in the Ecoli data set is regarded as the negative classes, other classes are united. The second and the third classes in the Wine data set are united.

The number of instances for training will be denoted as n. Moreover, we take $n/2$ examples from every class for training. These are randomly selected from the classes. The remaining instances in the dataset are used for validation. It should be noted that the accuracy measures are computed as average values by means of the random selection of n training examples from the data sets many times.

In all these examples a sets of weighted SVMs as a classifier has been used.

The goal of the RILBoost algorithm evaluation is to investigate the performance of the proposed algorithms by different values of the parameter ε in the linear-vacuous mixture model and to compare the proposed algorithms with the standard AdaBoost. Therefore, experiments are performed to evaluate the effectiveness of the proposed method under different values of ε and different numbers of training examples. It should be noted that the standard AdaBoost takes place when the parameter ε is 0. Therefore, all starting points of curves characterizing the classification accuracy correspond to the standard AdaBoost method accuracy.

First, we study the accuracy measure of the RILBoost algorithms as a function of ε by using $T = 10$ iterations and by different sizes of the training set. Fig. 4 illustrates how the accuracy measures depend on ε by training $n = 20, 40$ randomly selected examples from the Haberman's Survival, Pima Indian Diabetes, Breast Tissue, Mammographic masses data sets. The left

Table 1: A brief introduction about data sets

	m	n_0	n_1
Haberman's Survival	3	225	81
Pima Indian Diabetes	8	268	500
Mammographic masses	5	516	445
Parkinsons	22	147	48
Breast Tissue	9	36	70
Indian Liver Patient	10	416	167
Lung cancer	56	9	23
BCWO	9	458	241
Notterman CGED	7457	18	18
Seeds	7	70	140
Ionosphere	34	225	126
Ecoli	7	143	193
Vertebral Column 2C	6	210	100
Yeast	8	463	1021
Blood transfusion	4	570	178
Steel Plates Faults	27	1783	158
Planning Relax	12	130	52
Fertility Diagnosis	9	12	88
Wine	13	59	119
Madelon	500	1000	1000
QSAR biodegradation	41	356	699
Seismic bumps	15	2414	170

pictures of Fig. 4 illustrate the accuracy of the RILBoost I algorithm. One can see from this picture that the robust procedure outperforms the standard AdaBoost mainly for $n = 20$. This shows that the algorithm provides the best results when the training set is rather small. Moreover, one can see that there are some values of the robust parameter ε for which the accuracy measure is larger than for the standard AdaBoost. For example, the values for the Haberman's Survival data set are in the interval from 0.15 till 0.35. One can see from the figures corresponding to the Pima Indian Diabetes data set that the AdaBoost methods outperforms both the RILBoost algorithms except for one case of the RILBoost II algorithm for $n = 20$ when $\varepsilon = 0.45$. The right pictures of Fig. 4 illustrate the accuracy of the RILBoost II algorithm. One again can see from the picture that the robust procedure mainly outperforms the standard AdaBoost for $n = 20$. It is interesting to note that only RILBoost II algorithm provides better results by $n = 40$ for the Breast Tissue data set (see the corresponding right picture of Fig. 4). For the Mammographic Masses data set, only the case $n = 20$ provides larger values of $ACC_{\mathrm{S(I)}}$ and $ACC_{\mathrm{S(II)}}$ in comparison with ACC_{st}.

Similar results for other data sets can be found in Figs. 5-9.

In order to compare RILBoost I and RILBoost II algorithms we also compute the largest difference between accuracies of the RILBoost I, II algorithms and the standard AdaBoost method, i.e., we compute two differences $\Delta_{\mathrm{I}} = \max_{\varepsilon}(ACC_{\mathrm{S(I)}} - ACC_{\mathrm{st}})$ and $\Delta_{\mathrm{II}} = \max_{\varepsilon}(ACC_{\mathrm{S(II)}} - ACC_{\mathrm{st}})$ for various data sets and for different values of n. The positive value of Δ_{I} (Δ_{II}) indicates that the proposed algorithm outperforms the standard AdaBoost at least for one value of ε. Negative value says that the standard AdaBoost is

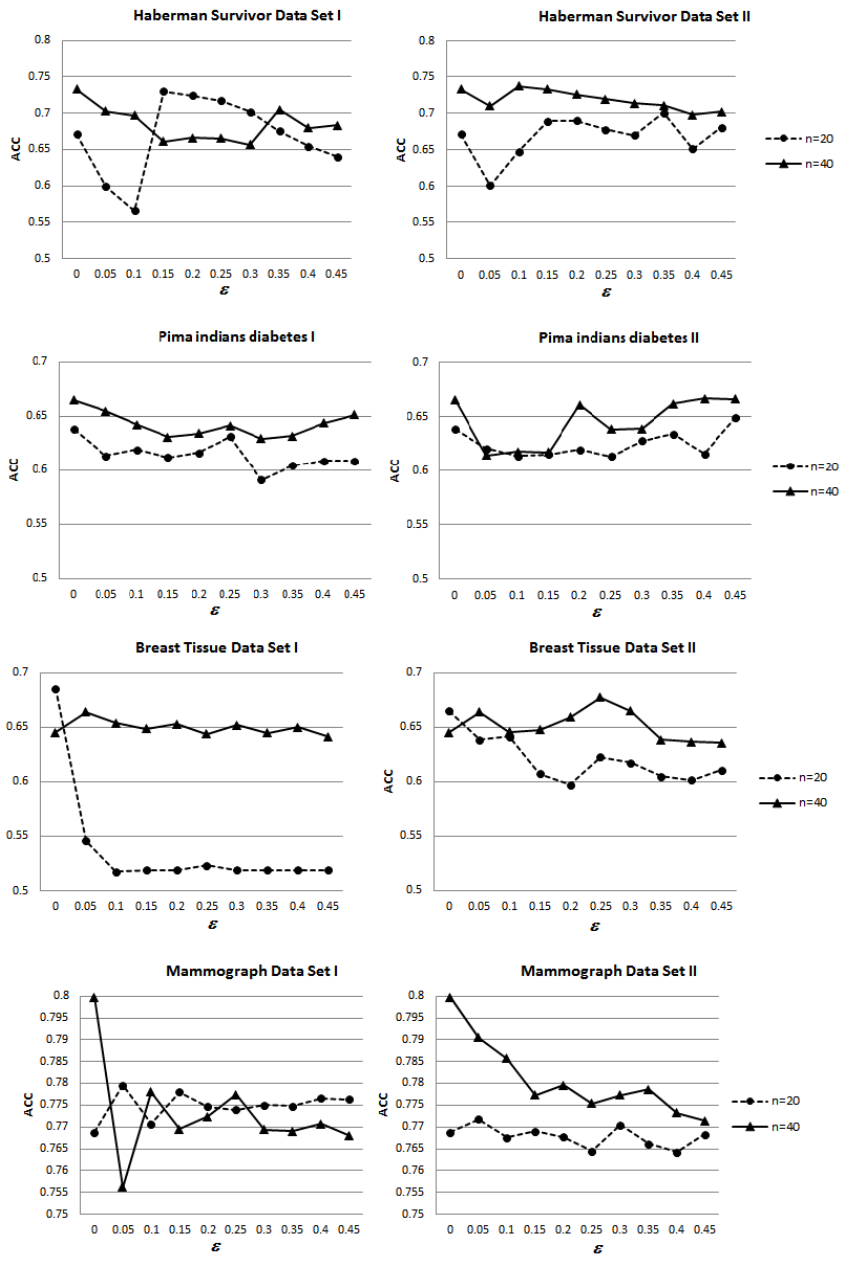

Figure 4: Accuracy by different ε and n for Haberman's Survival, Pima Indian Diabetes, Breast Tissue, Mammographic masses data sets

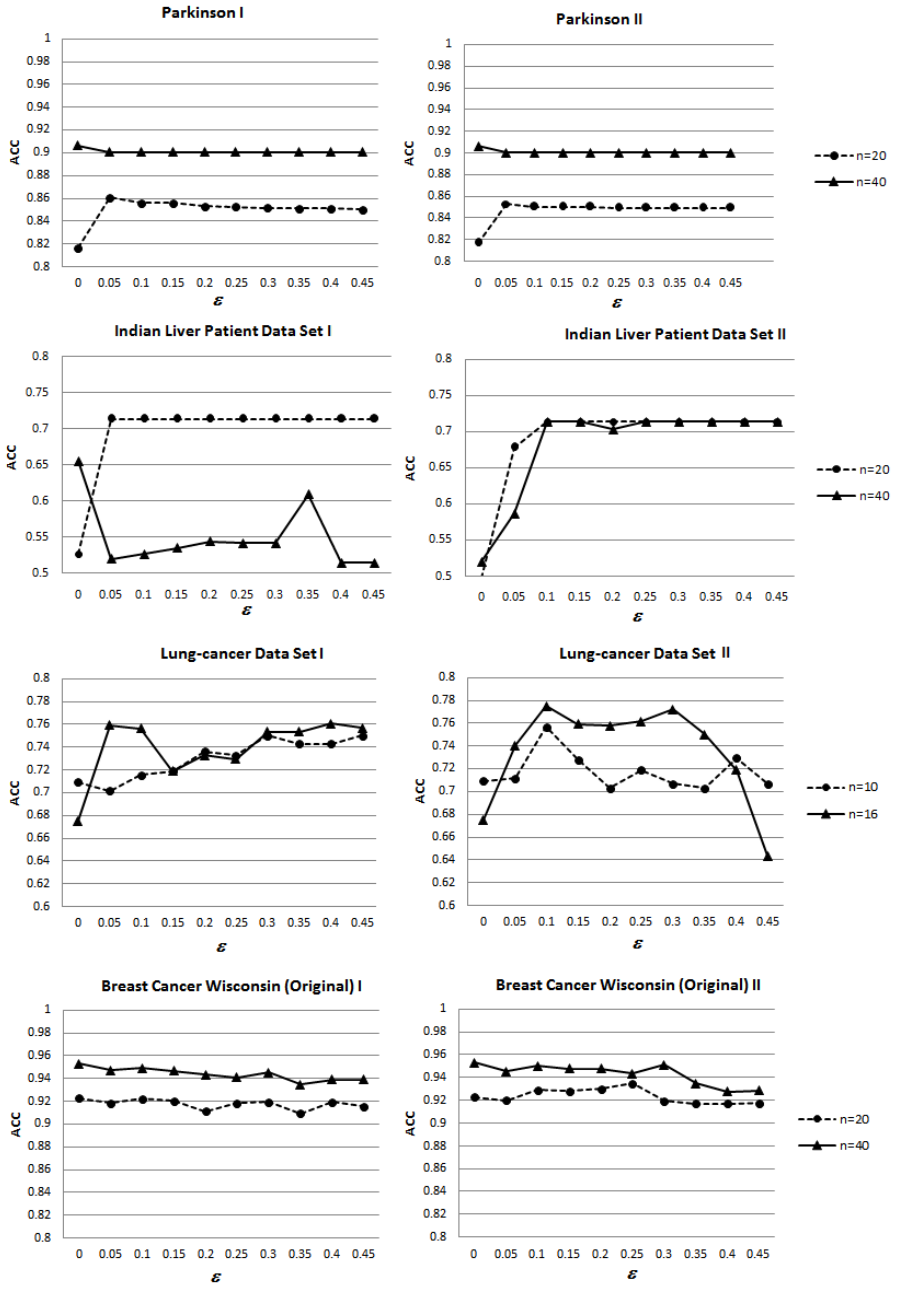

Figure 5: Accuracy by different ε and n for Parkinsons, Indian Liver Patient, Lung cancer, BCWO data sets

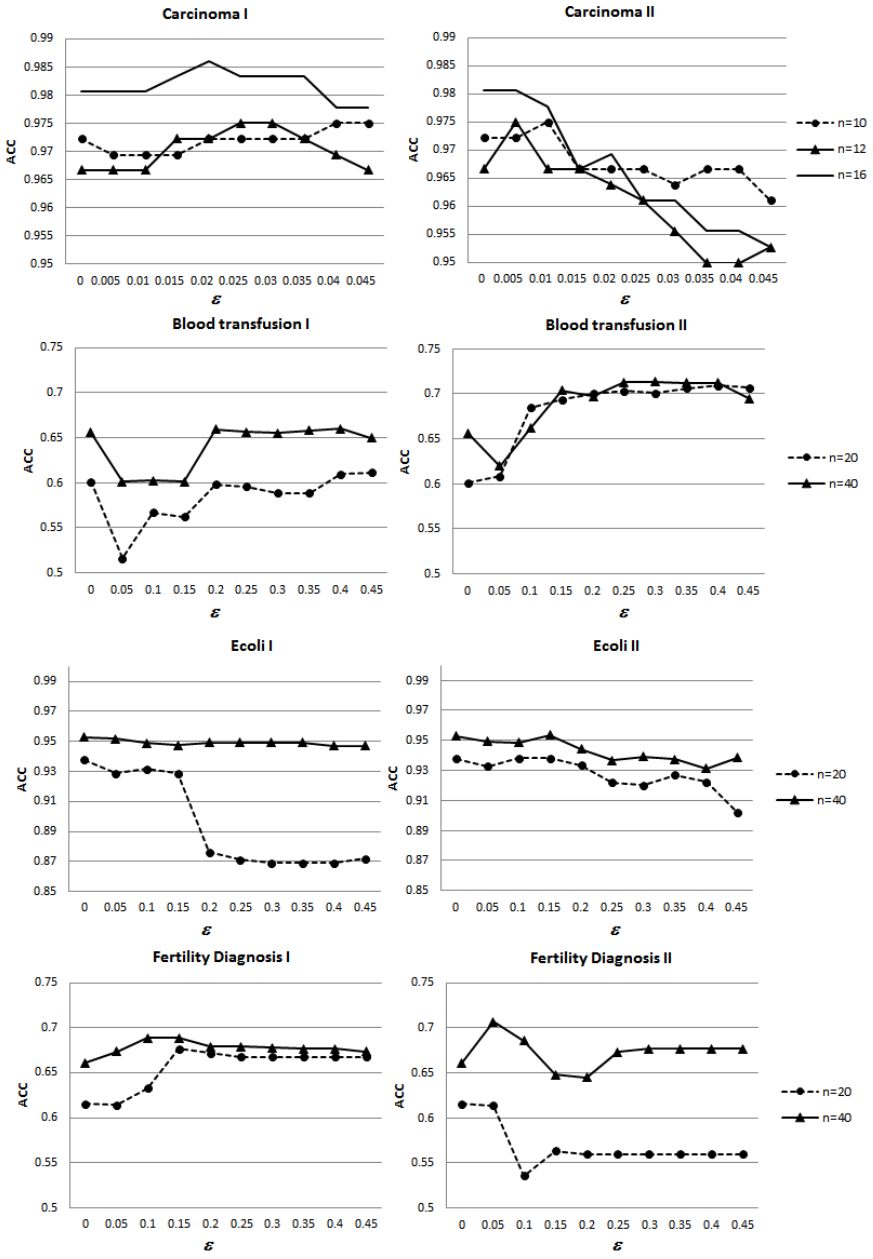

Figure 6: Accuracy by different ε and n for Notterman CGED, Blood transfusion, Ecoli, Fertility Diagnosis data sets

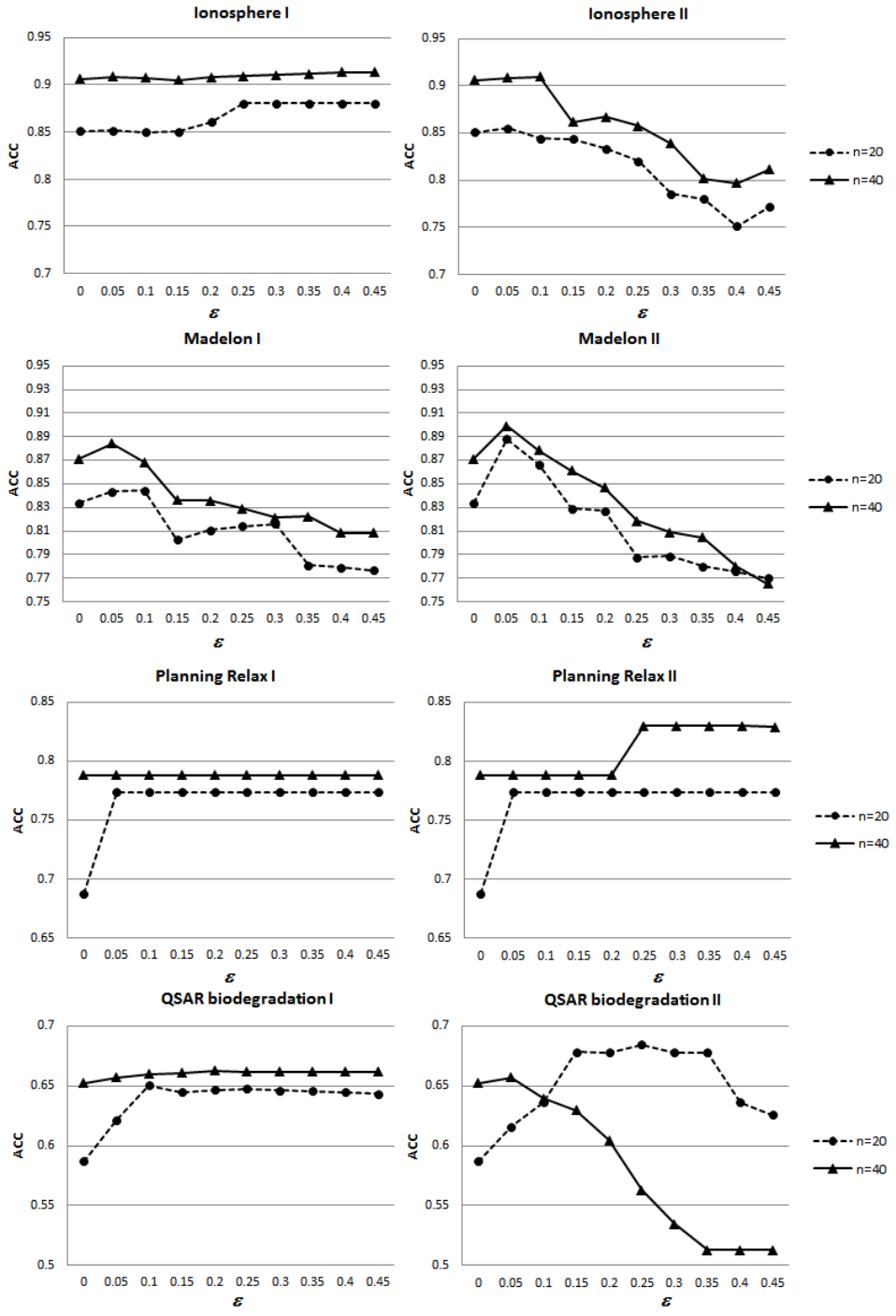

Figure 7: Accuracy by different ε and n for Ionosphere, Madelon, Planning Relax, QSAR biodegradation data sets

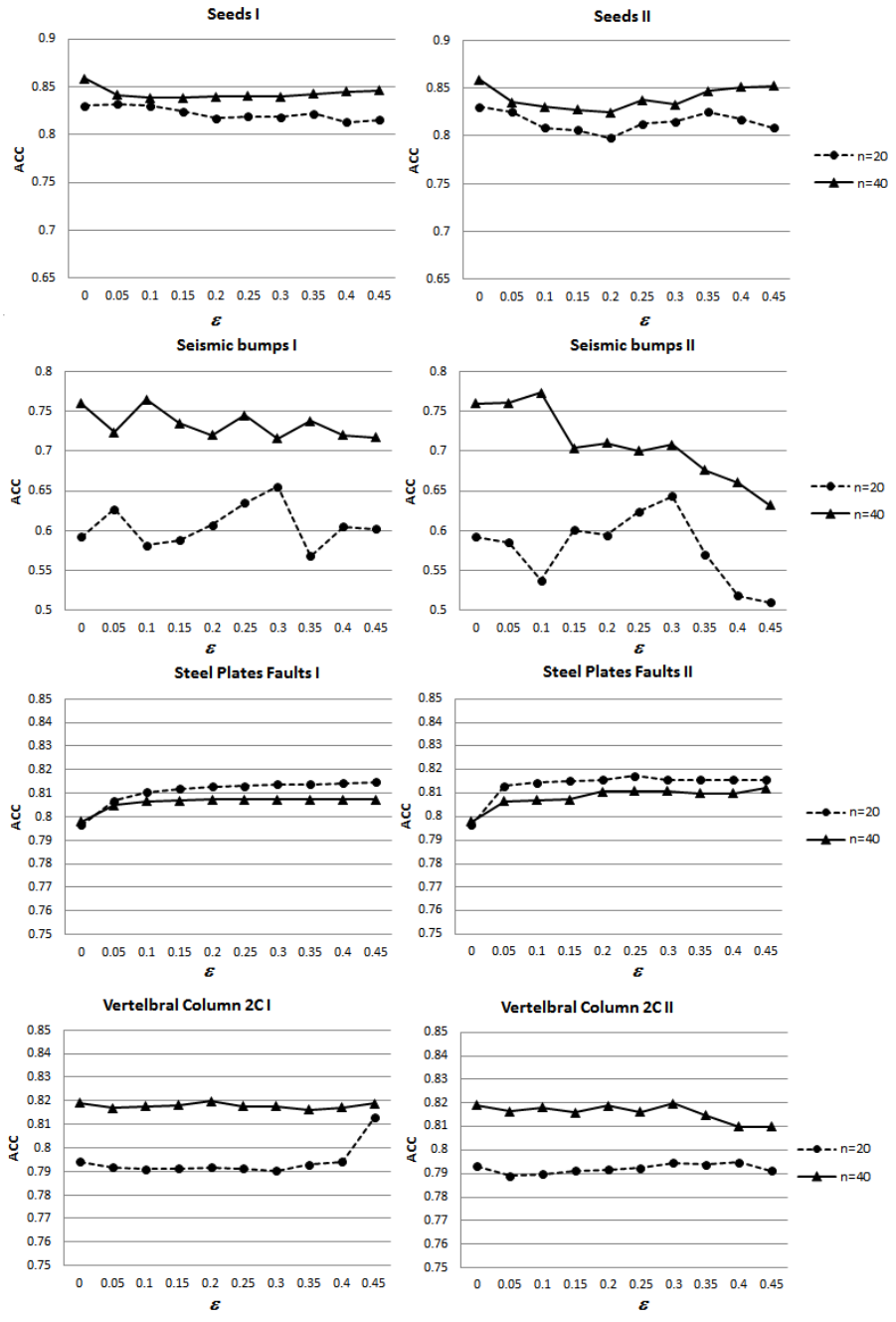

Figure 8: Accuracy by different ε and n for Seeds, Seismic bumps, Steel Plates Faults, Vertebral Column 2C data sets

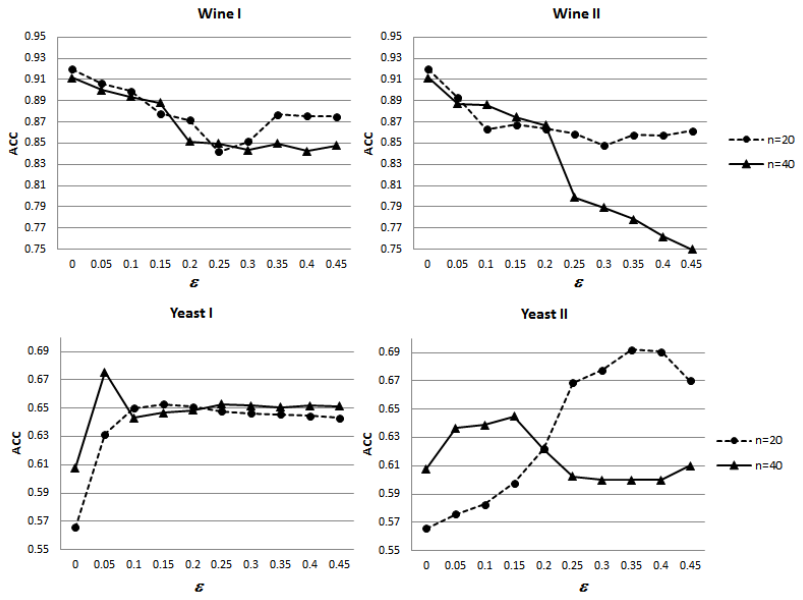

Figure 9: Accuracy by different ε and n for Wine and Yeast data sets

better in comparison with the proposed algorithms. The computation results are given in Tables 2-4. One can see from the tables that the proposed algorithms outperform the AdaBoost for small numbers of examples in training sets almost for all data sets analyzed. Moreover, it is clearly seen from Tables 2-4 that the RILBoost II algorithm outperforms the RILBoost I algorithm also for most data sets.

In order to formally show the outperforming the proposed algorithms, we apply the paired t-test which has been proposed and described by Demsar [50] for testing whether the average difference in the performance of two classifiers is significantly different from zero. Since we use the differences Δ_{I} and Δ_{II}, then we compare them with 0 which represents the standard AdaBoost. If we denote performance score of a classifier Δ_i on the i-th out

Table 2: The largest differences between accuracies of RILBoost I, RILBoost II and AdaBoost by different n

	n	Δ_{I}	Δ_{II}
Haberman's	20	0.059	0.029
Survival	40	−0.029	0.004
Pima Indian	20	−0.007	0.011
Diabetes	40	−0.01	0.002
Breast	20	0.011	0.030
Tissue	40	−0.022	−0.009
Mammographic	20	0.011	0.003
masses	40	−0.022	−0.009
Parkinsons	20	0.044	0.036
	40	−0.006	−0.006
Indian Liver	20	0.187	0.216
Patient	40	−0.046	0.194
Lung	10	0.406	0.048
cancer	16	0.085	0.100
BCWO	20	−0.0004	0.012
	40	−0.004	−0.002

Table 3: The largest differences between accuracies of RILBoost I, RILBoost II and AdaBoost by differenn n

	n	Δ_{I}	Δ_{II}
Seismic bumps	20	0.0632	0.0508
	40	0.0049	0.0135
QSAR	20	0.063	0.0972
biodegradation	40	0.0107	0.0048
Madelon	20	0.0107	0.0545
	40	0.013	0.0276
Seeds	20	0.002	−0.0052
	40	−0.0124	−0.0062
Ecoli	20	−0.0064	0.0002
	40	−0.0012	0.0005
Ionosphere	20	0.0293	0.004
	40	0.0071	0.0037
Vertebral	20	0.0187	0.0016
Column 2C	40	0.0008	0.0007

Table 4: The largest differences between accuracies of RILBoost I, RILBoost II and AdaBoost by different n

	n	Δ_{I}	Δ_{II}
Yeast	20	0.0872	0.1262
	40	0.0678	0.0371
Blood	20	0.0099	0.1077
transfusion	40	0.0037	0.0571
Steel Plates	20	0.0181	0.0209
Faults	40	0.0094	0.0141
Planning	20	0.0857	0.0857
Relax	40	0	0.0418
Fertility	20	0.0611	−0.0016
Diagnosis	40	0.028	0.046
Wine	20	−0.0135	−0.027
	40	−0.018	−0.0258
Notterman	10	0.003	0.003
CGED	12	0.008	0.008
	16	0.006	0.000

Table 5: The t-test by different n for the first and the second algorithms

n	Δ_{I}		Δ_{II}	
	t	95%-CI	t	95%-CI
20	2.670	$[0.012, 0.092]$	3.452	$[0.016, 0.066]$
40	0.486	$[-0.010, 0.016]$	2.194	$[0.001, 0.043]$

of M data sets, then the t statistics is computed as $\overline{\Delta}\sqrt{n}/s$. Here $\overline{\Delta}$ is the mean value of scores, i.e., $\overline{\Delta} = \sum_{i=1}^{M} \Delta_i/M$, s is the standard deviation of scores. The t statistics is distributed according to the Student distribution with $M-1$ degrees of freedom. The results of computing the t statistics and the corresponding 95 percent confidence intervals (95%-CI) on the basis of Tables 2-4 are given in Table 5.

It can be seen from Table 5 that the second algorithm provides the best results in comparison with the first algorithm and the standard AdaBoost. Moreover, both the algorithms outperform the standard AdaBoost when we have a small training set (see the case $n = 20$). At the same time, it follows from the case $n = 40$ that the advantage of the proposed algorithms can be under a question when the training set increases. This implies that the algorithms can be useful when only small training sets are available.

Another question is the choice of a strategy for realizing the weak classifiers in the boosting algorithms. Table 6 illustrates the use of the minimin strategy and its comparison with the minimax strategy for several data sets (Ionosphere, Pima Indian Diabetes, Steel Plates Faults, Seeds, Blood Transfusion, Ecoli, QSAR biodegradation). It can be clearly seen from the table that the minimax strategy outperforms the minimin one in most cases. In

Table 6: Comparison of the minimax and minimin strategies

		minimax		minimin	
	n	Δ_{I}	Δ_{II}	Δ_{I}	Δ_{II}
Ionosphere	20	0.0293	0.004	−0.0057	−0.0428
	40	0.0071	0.0037	0.0003	−0.0109
Pima Indian	20	−0.007	0.011	−0.0144	−0.0256
Diabetes	40	−0.01	0.002	−0.0142	−0.0142
Steel Plates	20	0.0181	0.0209	0.033	0.0226
Faults	40	0.0094	0.0141	−0.0226	−0.0226
Seeds	20	0.002	−0.0052	−0.0024	−0.0048
	40	−0.0124	−0.0062	−0.0124	−0.0062
Blood	20	0.0099	0.1077	0.0425	0.0725
Transfusion	40	0.0037	0.0571	0.0486	0.0313
Ecoli	20	−0.0064	0.0002	−0.0024	0.0012
	40	−0.0012	0.0005	−0.0012	−0.0015
QSAR	20	0.063	0.0972	0.0846	0.0935
biodegradation	40	0.0107	0.0048	0.0107	0.0048

order to formally show the fact that the minimax strategy outperforms the minimin strategy, we again apply the paired t-test [50]. The corresponding results are shown in Table 7. One can see from the table that the t-test demonstrates the clear outperforming of the minimax strategy for the second algorithm.

Let us study how the parameter α in (4) impacts on the classification performance. As an example, Fig. 10 illustrates how accuracy depends on differ-

Table 7: The t-test for comparison the minimin and minimax strategies

n	Δ_{I}		Δ_{II}	
	t	95%-CI	t	95%-CI
20	0.447	$[-0.017, 0.024]$	-2.102	$[-0.037, 0.003]$
40	0.032	$[-0.021, 0.021]$	-2.553	$[-0.0277, -0.001]$

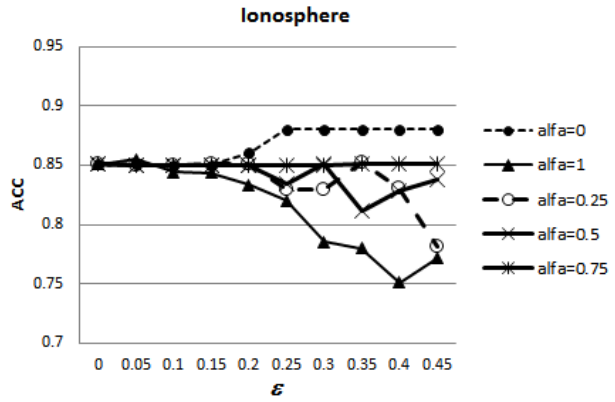

Figure 10: Accuracy by different ε and α for Ionosphere data set

ent values of the parameter α ($\alpha = 0, 0.25, 0.5, 0.75, 1$) by $n = 20$ and by using the second algorithm. It should be noted that the differences Δ_{II} by different α are 0.0293, 0.0011, 0.004, 0.004, 0.004. This example shows the monotone dependence of Δ_{II} from the parameter α. However, another example studied the same dependence for the Seismic bumps data set gives the following quite different dependence: $\Delta_{\mathrm{II}} = (0.0508, 0.1149, 0.0618, 0.0465, 0.0632)$. It can be seen from the above results that there is an optimal value of α which provides the largest value of Δ_{II}. Here we can see that this value of α is 0.25. Unfortunately, the optimal parameter α can be obtained only by considering all possible values in a predefined grid.

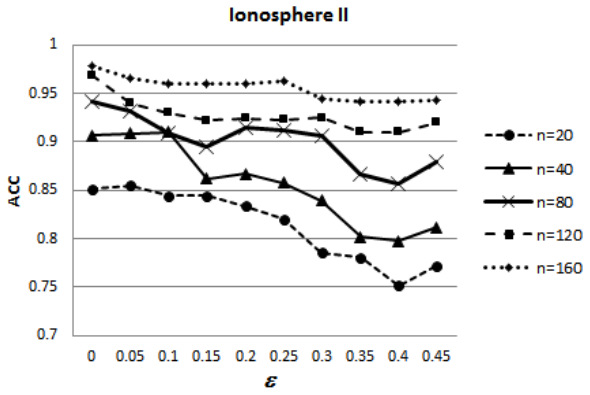

Figure 11: Accuracy by different ε and n for Ionosphere data set

Let us study now how the amount of training data impacts on the classification performance of the proposed algorithms. It is shown in Fig. 11 the accuracy measures as functions of ε by different numbers n ($n = 20, 40, 80, 120, 160$) of training data randomly taken from the Ionosphere data set. It can be seen from Fig. 11 that the impact of ε on the accuracy measure is reduced with increase of n. This fact is supported by the following values of differences $\Delta_{\mathrm{II}}(n)$: 0.004, 0.0037, -0.01, -0.012, -0.029, which are reduced with increase of n.

The impact of the increased amount ($n = 100$) of training data on the classification accuracy is shown in Fig. 12 and in Table 8 where the classification performance of the RILBoost II is considered for Yeast, QSAR biodegradation, Ecoli, Blood transfusion data sets. It can be seen from the figure that the behavior of the proposed method by $n = 100$ does not essentially differ from the same behavior by $n = 20$ and 40. Moreover, Table 8 shows that the difference between the proposed method and the AdaBoost is mainly reduced with increase of n.

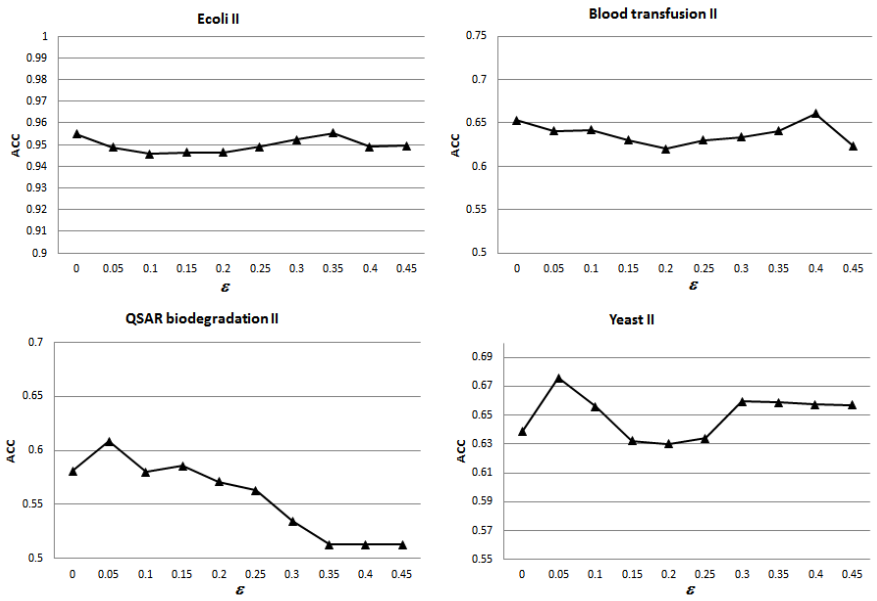

Figure 12: Accuracy by $n = 100$ for some data sets

Table 8: The largest differences between accuracies of RILBoost II and AdaBoost by different n

	Δ_{II}		
n	20	40	100
Yeast	0.1262	0.0371	0.0371
QSAR biodegradation	0.0972	0.0048	0.0275
Ecoli	0.0002	0.0005	0.0005
Blood transfusion	0.1077	0.0571	0.0078

Now we consider the performance of the proposed models with synthetic data having two features x_1 and x_2. The training set consisting of two subsets is generated in accordance with the normal probability distributions such that $N_1 = 100$ examples (the first class) are generated with mean values $\mathbf{m}_1 = (2, 2)$ and $N_2 = 100$ examples (the second class) have mean values $\mathbf{m}_2 = (3, 3)$. The standard deviation is $s = 3$ for both subsets and both features. We randomly select $n = 20$ and $n = 40$ examples from the generated 200 examples for getting two training sets. The remaining $N_1 + N_2 - n$ examples are used for testing. As a result, we obtain extremely noisy training sets because examples from the both classes are very mixed due to the large standard deviation. The number of iterations is 10. By taking the RBF kernel parameter 2 and the RILBoost II algorithm, we get the results shown in Fig.13. It can be seen from the picture that the proposed model provides larger values of accuracy in comparison with the standard AdaBoost when the number of examples in the training set is rather small. This example clearly shows when the proposed algorithms should be used. They are efficient in comparison with the standard AdaBoost when the number of training data is small and when examples of different classes are very mixed.

6. Conclusion

One can not to assert that the proposed models outperform the standard AdaBoost in all cases. It can be seen from the experiments that the algorithms provide higher classification accuracy measures for some publicly available data sets. At the same time, there are data sets which show that the standard AdaBoost method may be better than the RILBoost methods.

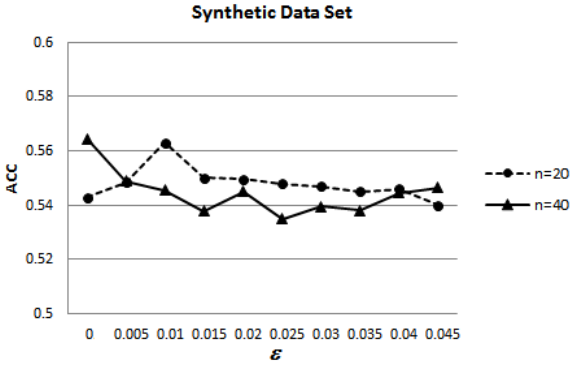

Figure 13: Accuracy by different n for synthetic data

In spite of the above fact, the proposed approach is interesting and can be successfully applied in classification.

We have studied only two algorithms corresponding to the extreme cases of the function G when $\alpha = 1$ and $\alpha = 0$. It is interesting to investigate other cases of α and to study how values of this parameter impact on the accuracy of classification. This is a direction for future research.

Every "optimal" minimax classifier in each iteration of the proposed algorithms is determined by computing the expected loss or the Lagrangian for all extreme points of the set of probability distributions. Then the obtained "optimal" extreme point is used to determine vectors of classification errors and the error rate $e(t)$. However, the "optimal" classifier can be selected on the basis of the error rate computed for every extreme point. In other words, we compute $e(t)$ for every extreme point of the set of probability distributions. The largest value of $e(t)$ corresponds to the "optimal" classifier in each iteration. The study of this approach is another direction for further work.

It should be noted that we have studied only one class of imprecise sta-

tistical models. However, it is interesting to investigate how the statistical models [44] different from the imprecise ε-contaminated model such that the imprecise pari-mutuel model, the imprecise constant odds-ratio model, bounded vacuous and bounded ε-contaminated robust models, Kolmogorov-Smirnov bounds can be incorporated into the boosting model.

The RILBoost methods have been applied to modifying only the AdaBoost method. However, one can see from the proposed approach that it can simply be extended on many boosting-like models whose detailed review is given, for example, by Ferreira and Figueiredo [8]. This can be carried out by specifying Algorithm 1.

Acknowledgement

We would like to express our appreciation to the anonymous referees whose very valuable comments have improved the paper. The reported study was partially supported by the Ministry of Education and Science of Russian Federation and by RFBR, research project No. 14-01-00165-a.

References

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

[2] V. Cherkassky, F. Mulier, Learning from Data: Concepts, Theory, and Methods, Wiley-IEEE Press, UK, 2007.

[3] B. Scholkopf, A. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, The MIT Press, Cambridge, Massachusetts, 2002.

[4] A. Webb, Statistical Pattern Recognition, 2nd Edition, Wiley, 2002.

[5] R. Dara, M. Kamel, N. Wanas, Data dependency in multiple classifier systems, Pattern Recognition 42 (7) (2009) 1260 – 1273.

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

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

[8] A. Ferreira, M. 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.

[9] M. Re, G. Valentini, Ensemble methods: a review, in: Data Mining and Machine Learning for Astronomical Applications, Data Mining and Knowledge Discovery Series, Chapman & Hall, 2012, Ch. 26, pp. 563–594.

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

[11] D. Mease, A. Wyner, Evidence contrary to the statistical view of boosting, Journal of Machine Learning Research 9 (2008) 131–156.

[12] G. Ridgeway, GBM 1.5 package manual (2005).

[13] T. Zhang, B. Yu, Boosting with early stopping: Convergence and consistency, Annals of Statistics 33 (4) (2005) 1538–1579.

[14] C. Domingo, O. Watanabe, MadaBoost: A modification of AdaBoost, in: Proceedings of the Thirteenth Annual Conference on Computational Learning Theory, Morgan Kaufmann, San Francisco, 2000, pp. 180–189.

[15] M. Nakamura, H. Nomiya, K. Uehara, Improvement of boosting algorithm by modifying the weighting rule, Annals of Mathematics and Artificial Intelligence 41 (2004) 95–109.

[16] R. Servedio, Smooth boosting and learning with malicious noise, Journal of Machine Learning Research 4 (9) (2003) 633–648.

[17] T. Bylander, L. Tate, Using validation sets to avoid overfitting in AdaBoost, in: Proceedings of the 19th International FLAIRS Conference, 2006, pp. 544–549.

[18] R. Jin, Y. Liu, L. Si, J. Carbonell, A. Hauptmann, A new boosting algorithm using input-dependent regularizer, in: Proceedings of Twentieth International Conference on Machine Learning (ICML 03), AAAI Press, Washington DC, 2003, pp. 1–9.

[19] A. Lorbert, D. M. Blei, R. E. Schapire, P. J. Ramadge, A Bayesian boosting model, ArXiv e-printsarXiv:1209.1996.

[20] Y. Lu, Q. Tian, T. Huang, Interactive boosting for image classification, in: Proceedings of the 7th International Conference on Multiple Classifier Systems, MCS'07, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 180–189.

[21] M. Warmuth, K. Glocer, G. Ratsch, Boosting algorithms for maximizing the soft margin pages, in: Advances in Neural Information Processing Systems NIPS, MIT Press, 2007, pp. 1–8.

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

[23] P. Huber, Robust Statistics, Wiley, New York, 1981.

[24] C. Robert, The Bayesian Choice, Springer, New York, 1994.

[25] T. Hertz, A.B. Hillel, D. Weinshall, Learning a kernel function for classification with small training samples, in: Proceedings of the 23rd international conference on Machine learning, ACM, 2006, pp. 401–408.

[26] L. Fei-Fei, R. Fergus, P. Perona, One-shot learning of object categories, IEEE Transactions on Pattern Analysis and Machine Intelligence, 28 (4), 2006, 594–611.

[27] I. Guyon, J. Weston, S. Barnhill, V. Vapnik, Gene selection for cancer classification using support vector machines, Machine Learning, 46, 2002, 389–422.

[28] H. Xu, C. Caramanis, S. Mannor, Robustness and regularization of support vector machines, The Journal of Machine Learning Research 10 (7) (2009) 1485–1510.

[29] C. Bouveyron, S. Girard, Robust supervised classification with mixture models: Learning from data with uncertain labels, Pattern Recognition 42 (11) (2009) 2649 – 2658.

[30] A. Cerioli, M. Riani, A. Atkinson, Robust classification with categorical variables, in: A. Rizzi, M. Vichi (Eds.), Compstat 2006 - Proceedings in Computational Statistics, Physica-Verlag HD, 2006, pp. 507–519.

[31] L. Ghaoui, G. Lanckriet, G. Natsoulis, Robust classification with interval data, Tech. Rep. Report No. UCB/CSD-03-1279, University of California, Berkeley, California 94720 (2003).

[32] G. Lanckriet, L. Ghaoui, C. Bhattacharyya, M. Jordan, A robust minimax approach to classification, Journal of Machine Learning Research 3 (2002) 555–582.

[33] G. Lanckriet, L. Ghaoui, M. Jordan, Robust novelty detection with single-class mpm, in: S. Becker, S. Thrun, K. Obermayer (Eds.), Advances in Neural Information Processing Systems, Vol. 15, MIT Press, Cambridge, MA, 2003, pp. 905–912.

[34] F. Provost, T. Fawcett, Robust classification for imprecise environments, Machine Learning 42 (3) (2001) 203–231.

[35] T. Trafalis, R. Gilbert, Robust support vector machines for classification and computational issues, Optimization Methods and Software 22 (1) (2007) 187–198.

[36] L. Xu, K. Crammer, D. Schuurmans, Robust support vector machine training via convex outlier ablation, in: Proceedings of the 21st National Conference on Artificial Intelligence (AAAI-06), Vol. 21, AAAI Press; MIT Press, Boston, Massachusetts, 2006, pp. 536–542.

[37] A. Ben-Tal, L. Ghaoui, A. Nemirovski, Robust optimization, Princeton University Press, Princeton, New Jersey, 2009.

[38] J. Bi, T. Zhang, Support vector classification with input data uncertainty, in: L. Saul, Y. Weiss, L. Bottou (Eds.), Advances in Neural Information Processing Systems, Vol. 17, MIT Press, Cambridge, MA, 2004, pp. 161–168.

[39] J. Wang, H. Lu, K. Plataniotis, J. Lu, Gaussian kernel optimization for pattern classification, Pattern Recognition 42 (7) (2009) 1237 – 1247.

[40] J. Berger, Statistical Decision Theory and Bayesian Analysis, Springer-Verlag, New York, 1985.

[41] I. Gilboa, D. Schmeidler, Maxmin expected utility with non-unique prior, Journal of Mathematical Economics 18 (2) (1989) 141–153.

[42] M. Troffaes, Decision making under uncertainty using imprecise probabilities, International Journal of Approximate Reasoning 45 (1) (2007) 17–29.

[43] X. Yang, Q. Song, Y. Wang, A weighted support vector machine for data classification, International Journal of Pattern Recognition and Artificial Intelligence 21 (5) (2007) 961–976.

[44] L. Utkin, A framework for imprecise robust one-class classification models, International Journal of Machine Learning and Cybernetics. To appear. doi:10.1007/s13042-012-0140-6.

[45] L. Utkin, Y. Zhuk, Robust novelty detection in the framework of a contamination neighbourhood, International Journal of Intelligent Information and Database Systems 7 (3) (2013) 205–224.

[46] J. Weston, C. Watkins, Multi-class support vector machines, Technical Report CSD-TR-98-04, Royal Holloway, University of London, Department of Computer Science (May 1998).

[47] R Development Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria (2005).
URL `http://www.R-project.org`

[48] A. Frank, A. Asuncion, UCI machine learning repository (2010).
URL `http://archive.ics.uci.edu/ml`

[49] D. Notterman, U. Alon, A. Sierk, A. Levine, Transcriptional gene expression profiles of colorectal adenoma, adenocarcinoma, and normal tissue examined by oligonucleotide arrays, Cancer Research 61 (2001) 3124–3130.

[50] J. Demsar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine Learning Research 7 (2006) 1–30.