International Journal of Pattern Recognition and Artificial Intelligence

AN IMPRECISE BOOSTING-LIKE APPROACH TO CLASSIFICATION

LEV V. UTKIN

Department of Control, Automation and System Analysis
Saint-Peterburg State Forest Technical University
Saint-Petersburg, 194021, Russia
lev.utkin@gmail.com
http://<www.levvu.narod.ru>

A new approach for ensemble construction based on restricting a set of weights of examples in training data to avoid overfitting is proposed in the paper. The algorithm called EPIBoost (Extreme Points Imprecise Boost) applies imprecise statistical models to restrict the set of weights. The updating of the weights within the restricted set is replaced by updating the weights in the linear combination of extreme points within the unit simplex. The approach allows us to construct various algorithms by applying different imprecise statistical models for producing the restricted set. It is shown by various numerical experiments with real data sets that the EPIBoost algorithm may outperform the standard AdaBoost by some parameters of imprecise statistical models.

Keywords: Machine learning; classification; boosting; AdaBoost; imprecise model; extreme points; training data.

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 $c(\mathbf{x}_i) = y_i$ 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 $y_i \in \{-1, 1\}$.

A classification problem is usually characterized by an unknown probability distribution $p(\mathbf{x}, y)$ on $\mathbb{R}^m \times \{-1, +1\}$ 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 [27] and the well known support vector machine (SVM) method [4,25,31] exploit the assumption of the uniform distribution.

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 and will be denoted below as $h = (h_1, ..., h_n)$. 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 [6]. 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 widely studied how to combine several classifiers together in order to achieve improved recognition performance was written by Kuncheva [13]. 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 [23]. 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 [21] 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. 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 1). 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 1. 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.

As pointed out by Rokach [23], 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.

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

The main advantage of AdaBoost over other boosting techniques is that it 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 a very important feature of the AdaBoost and other ensemble-based algorithms. It 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. The updated probability distribution can be arbitrary, i.e. the set of possible probability distributions is not restricted. On the one hand, this arbitrariness can be viewed as a way for searching for an optimal solution, and this is a positive feature of the AdaBoost algorithm. On the other hand, this arbitrariness is one of the shortcomings of this approach. It has been reported by many authors that the main reason for poor classification results of AdaBoost in the high-noise regime is that the algorithm produces a skewed data distribution, by assigning too large weights onto a few hard-to-learn examples. This leads to overfitting. Moreover, the algorithm does not take into account the possible additional information about the training set. For instance, the number of examples in the training set and confidence probabilities might give a very useful information which restricts the set of probability distributions.

Many modifications of AdaBoost have been proposed to overcome the problem of overfitting. In order to prevent overfitting, one popular regularization technique is to stop boosting algorithms after a very small number of iterations [16]. The statistics community has put a lot of emphasis on early stopping as evidenced by the large number of papers on this topic. For example, the paper by Zhang and Yu [33] tells readers that "boosting forever can overfit the data" and that "therefore in order to achieve consistency, it is necessary to stop the boosting procedure early." Standard implementations of boosting such as the popular gbm package for R by Ridgeway [22] implement data-derived early stopping rules. The reasoning behind early stopping is that after enough iterations have occurred any additional iterations will lead to overfitting and consequently larger misclassification error. Mease and Wyner [16] illustrate by means of simple and visual examples that the early stopping rule may lead to incorrect classification results.

There exist other approaches to the problem of overfitting. For example, the SoftBoost algorithm [30] does not concentrate too much on outliers and hard to classify examples by optimizing the soft margin. The WeightBoost algorithm [12] tries to cope with overfitting for noisy data by introducing an input-dependent regularization factor to the combination weight. Jin et al. [12] derive a learning procedure for WeightBoost, which is guaranteed to minimize training errors. Another algorithm for coping with overfitting called I.Boosting was proposed by Lu et al. [15], where weights of examples and their features are assigned in accordance with some rule. Another interesting model called VIBoost (boosting stemming from variational inference) and vulnerable to noise was proposed by Lorbert et al. [14]. It is based on embedding binary classification in a Bayesian model in the framework of boosting methods. The authors of the work [3] propose the FitValidationSet algorithm. According to the algorithm, half of the training set is removed to form a validation set. The sequence of base classifiers, produced by AdaBoost from the training set, is applied to the validation set, creating a modified set of weights. The training and

validation sets are switched, and a second pass is performed. The final classifier votes using both sets of weights.

In order to avoid the problem of overfitting, several modifications of boosting algorithms which restrict the set of weights of examples in training data or their probability distributions according to a certain rule were proposed. In fact, every modification is a choice of one or another rule restricting the weights. Servedio [26] proposed the SmoothBoost algorithm which generates only smooth distributions which do not assign too much weight to any single example. Nakamura et al. [17] proposed the algorithm called NadaBoost which adjusts the weighting scheme by adding a threshold for the maximum allowable weight. In other words, the authors [17] introduce thresholds to judge whether weights are too high or not. Another method called MadaBoost [7] provides an upper bound on example weights based on initial probability. The idea of restricting the weights of examples underlying in the above modifications will be used in the proposed algorithm.

It should be noted that this is only a 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 avoid the arbitrariness of weights assigned to examples at each iteration by restricting a set of probability distributions in accordance with some predefined rule. 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 (left picture). 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 (left picture). Indeed, we assign the weight 1 to a "hard" example and weight 0 to other examples.

Let us restrict the possible area where the probability distribution can move. This area is depicted in Fig. 1 (right picture) in the form of a hexagon. One can see that the probabilities again tend to concentrate on "hard" examples, but we do not get now the unit and zero probabilities of examples. If the restricted area is constructed in accordance with some justified rule, then we can develop new algorithms avoiding overfitting.

The next task is how to define a "justified rule" for constructing the set of probability distributions. One of the ways for solving the task is to apply the imprecise statistical models [28]. The following imprecise probability models are proposed to be used in the paper: the linear-vacuous mixture [28] which can be regarded as an extension of the well-known ε-contaminated (robust) model [11]; the pari-mutuel model, the constant odds-ratio model; bounded vacuous and bounded ε-contaminated robust models. Every above imprecise model has a parameter with a clear statistical

explanation which defines the size of the probability set such that we can construct different sets from the smallest one containing a single probability distribution till the unit simplex. It is interesting to note that the smallest set leads to the standard classifier with certain weights of examples. The largest set (the unit simplex) corresponds to the standard AdaBoost algorithm. It is necessary to point out that the imprecise probability models have been used in classification problems. For example, a classification algorithm based on the likelihood-based learning approach using sets of probability distributions was proposed by Antonucci et al. [1]. Corani and Zaffalon [5], Zaffalon [32] used imprecise probabilities for building reliable classifiers from small or incomplete data sets. However, the above authors exploit the imprecise probabilities in order to replace a single probability distribution of examples by a set of distributions. In contrast to this idea, we reduce the largest set of weights of examples to a smaller set produced by imprecise statistical models.

The use of imprecise probability models instead of the approaches applied to boosting algorithms like NadaBoost and MadaBoost is caused by four main reasons. First, the imprecise statistical models are well justified from the statistic point of view and their parameters have a strong interpretation. For example, the upper bounded contaminated robust model has a very close connection with well-known Kolmogorov-Smirnov bounds and its parameter can be expressed through confidence probabilities and numbers of observations or examples. Second, the imprecise models produce convex sets of probability distributions. The property of convexity is very important for implementing the proposed boosting algorithms. Third, every imprecise model is characterized by a parameter which determines the size of the produced set of probability distributions. Therefore, when we do not have prior information about the parameter, we can change it and to choose the best boosting model corresponding to one of the parameter values. Fourth, the imprecise statistical models have simple procedures for Bayesian updating. Therefore, by applying the information about classification errors at every iteration of boosting, we can update the sets of probability distributions in order to take into account this information.

The next task is how to modify weights or the probability distributions within the restricted set. The second idea underlying the proposed approach allows us to solve this task. This idea is to replace the weights of examples by weights of extreme points of the restricted set of probabilities. Suppose that the restricted set of probability distributions is convex. In this case, the probability set is totally defined by its extreme points. This implies that every probability distribution from the set can be obtained as the linear combination of the extreme points with certain weights λ. It should be noted that λ is not a vector of weights of examples, but it is a vector of weights of extreme points. This implies that updating of the probability distribution can be replaced by updating of weights λ in accordance with some rule which will be proposed below. In other words, we change the weights of extreme points, but not the weights of examples as it is carried out in AdaBoost. Of course, this does not mean that the weights of examples are not changed. They are changed

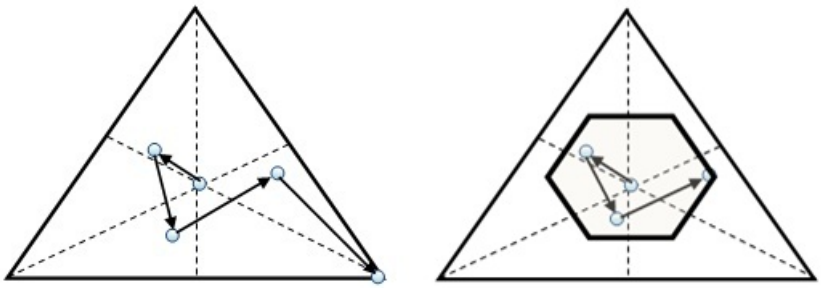

Fig. 1. The unit simplexes of probabilities and possible weights of examples in each iterations

indirectly through updating weights of extreme points. The important fact here is that weights λ do not have restrictions and can be changed within the unit simplex. This idea has another virtue. A number of extreme points is always larger than the number of examples in the training set or it is equal to the number of examples. Therefore, we get a more flexible way for the adaptation.

The ideas of using the extreme points and the imprecise statistical models give the corresponding name of the algorithm EPIBoost (Extreme Points Imprecise Boost).

The set of all probability distributions is a special (extreme) case corresponding to the largest set. This implies that the standard AdaBoost algorithm can be regarded as a special case of the proposed algorithm.

2. Extreme Points Imprecise Boost (EPIBoost)

By accepting the probabilistic interpretation of weights, we assume that there exists a set $\mathcal{P}$ of probability distributions defined by some information about the training set or by some other reasons. There are various kinds of the information which could be exploited. By using the imprecise statistical models, we can use the information about confidence probabilities for the set $\mathcal{P}$, about the number of examples in the training set. Another kind of the additional information arises when we have imbalanced data sets. Class imbalance occurs when there are significantly fewer training instances of one class compared to another class. In this case, the weights of examples from the small class increase at a higher rate compared to those of the prevalent class. In many applied problems, there is information about the most important examples or about typical examples for a given class. This prior information can also change the set $\mathcal{P}$.

At the same time, we do not need to have some additional information for determining the restricted set $\mathcal{P}$. We can apply the imprecise statistical models by changing their parameters. In fact, this way will be mainly considered below. Two extreme special cases have to be mentioned here. When we do not have the

additional information or do not want to exploit imprecise statistical models, the set $\mathcal{P}$ is the largest one and it coincides with the unit simplex having n vertices. Another extreme special case is a precise probability distribution. In this case, the set $\mathcal{P}$ is reduced to a point (an element).

We assume that the set $\mathcal{P}$ is convex, i.e., it is generated by finitely many linear constraints. This implies that it is totally defined by its extreme points or vertices denoted $\mathcal{E}(\mathcal{P})$. Suppose that we have a set of r extreme points $q_k = (q_1^{(k)}, ..., q_n^{(k)})$ of the set $\mathcal{P}$, i.e., $q_k \in \mathcal{E}(\mathcal{P})$, $k = 1, ..., r$. Then every probability distribution $h = (h_1, ..., h_n)$ from $\mathcal{P}$ can be represented as the linear combination of the extreme points

$$h = \sum_{k=1}^{r} \lambda_k \cdot q_k. \tag{1}$$

Here $\lambda = (\lambda_1, ..., \lambda_r)$ is a vector of weights such that $\lambda_1 + ... + \lambda_r = 1$.

The main idea of the proposed adaptation procedure is to change the weights λ instead of the probabilities h for decreasing the misclassification measure. It does not mean that the probability distribution h is not changed. It is changed inside the set $\mathcal{P}$ due to changes of the weights λ. However, the weights do not have additional restrictions and can be changed in the unit simplex having r vertices. In other words, we cannot change h in the unit simplex, but we can change the weights λ within the unit simplex without restrictions. The change of λ has to be directed to minimize the misclassification measure.

So, the first step is the random selection of the initial vector λ from the unit simplex or assigning the value $1/r$ to every weight. This procedure can be carried out by means of generating the random numbers in accordance with the Dirichlet distribution [24]. Then the corresponding probability distribution h belonging to $\mathcal{P}$ can be computed by using (1). This distribution is applied to the weighted classification procedure in order to get the classifier c_t.

The next step is to define and to compute the error measure of the classifier c_t. Let us introduce the k-th extreme point misclassification rate denoted by ε_k as follows:

$$\varepsilon_k = \sum_{i: c_t(x_i) \neq y_i} q_i^{(k)}.$$

According to the above measure, we assume that the examples have the probability distribution $q^{(k)}$ which coincides with the k-th extreme point. If we return to a standard classification procedure, then ε_k is the error ε under condition that it is measured with respect to the distribution $q^{(k)}$ on which the weak learner was trained. The above measure also means the contribution of the k-th extreme point into the classification errors. Indeed, if the i-th example is misclassified, then the probability $q_i^{(k)}$ is the contribution of the k-th extreme point into the probability of the i-th example h_i because $\sum_{k=1}^{r} \lambda_k q_i^{(k)} = h_i$. When there are s misclassified

examples with numbers $i_1, ..., i_s$, the misclassification rate ε_k is the contribution of the k-th extreme point into the probability of all errors $H_- = h_{i_1} + ... + h_{i_s}$.

Let us introduce the correct classification rate denoted by ζ_k as follows:

$$\zeta_k = \sum_{i:c_t(x_i)=y_i} q_i^{(k)} = 1 - \sum_{i:c_t(x_i)\neq y_i} q_i^{(k)} = 1 - \varepsilon_k.$$

This is the contribution of the k-th extreme point into the correct classification. Denote the probability of all correctly classified examples as $H_+ = 1- H_-$. It is obvious that there hold

$$H_- = \sum_{k=1}^{r} \lambda_k \varepsilon_k, \; H_+ = \sum_{k=1}^{r} \lambda_k \zeta_k.$$

It is interesting to note that the probability H_- is none other than the error $e(t)$ which is measured with respect to the distribution h on which the weak learner was trained, i.e., $H_- = e(t)$. The error $e(t)$ is similar to those defined by Freund and Schapire [10] in the standard AdaBoost.

We have to update the distribution h for the next iteration by decreasing probabilities of correctly classified examples H_+ and by increasing probabilities of misclassified examples H_-. This updating rule follows from a similar rule justified by Freund and Schapire [10]. According to this rule, the probabilities tend to concentrate on "hard" examples.

The updating of H_+ and H_- is carried out by changing the weights λ. However, one can see that, increasing (decreasing) the weights λ_k, we increase (decrease) simultaneously both H_+ and H_-, respectively. However, it is important here that the updated distribution h is normalized. In other words, we actually update $H_+/(H_+ + H_-)$ and $H_-/(H_+ + H_-)$, but not H_+ and H_- after the normalization. At this stage, updated H_+ and H_- do not satisfy the condition $H_+ + H_- = 1$. It is easily to see that $H_-/(H_+ + H_-)$ increases and $H_+/(H_+ + H_-)$ decreases with increase of λ_k when $\varepsilon_k > \zeta_k$ and vice versa. According to the above, the following rule for updating λ is proposed. If $\varepsilon_k > \zeta_k$, then λ_k is increased. If $\varepsilon_k \leq \zeta_k$, then λ_k is decreased. It should be noted that the condition $\varepsilon_k > \zeta_k$ is equivalent to the condition $\varepsilon_k > 0.5$ or $1 - 2\varepsilon_k > 0$. Modifying the similar rule for the standard AdaBoost, we can write

$$\lambda_k \leftarrow \lambda_k \cdot \exp\left(-\alpha_t \cdot (\zeta_k - \varepsilon_k)\right) = \lambda_k \cdot \exp\left(-\alpha_t \cdot (1 - 2\varepsilon_k)\right),$$

where

$$\alpha_t = \frac{1}{2} \ln\left(\frac{1 - e(t)}{e(t)}\right) = \frac{1}{2} \ln\left(\frac{1 - \sum_{k=1}^{r} \lambda_k \varepsilon_k}{\sum_{k=1}^{r} \lambda_k \varepsilon_k}\right).$$

Freund and Schapire [10] defined α_t as a measure of the importance that is assigned to the classifier c_t. Finally, we can write the algorithm of the EPIBoost which is represented as Algorithm 2.

Algorithm 2 The EPIBoost algorithm

Require: T (number of iterations), S (training set), $q_1, ..., q_r$ (extreme points of $\mathcal{P}$)

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

$t \leftarrow 1$

$\lambda_k(1) \leftarrow 1/r$; $k = 1, ..., r$, or it can be randomly selected from the unit simplex.

repeat

Compute the vector $h \leftarrow \sum_{k=1}^{r} \lambda_k(t) \cdot q_k$

Train classifier c_t using distribution h

Compute error rate $\varepsilon_k \leftarrow \sum_{i:c_t(x_i) \neq y_i} q_i^{(k)}$

Compute error $e(t) \leftarrow \sum_{k=1}^{r} \lambda_k(t)\varepsilon_k$

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

$T \leftarrow t - 1$

exit Loop.

end if

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

$\lambda_k(t+1) \leftarrow \lambda_k(t) \cdot \exp\left(-\alpha_t \cdot (1 - 2\varepsilon_k)\right)$

Normalize $\lambda(t+1)$ to be a proper distribution.

$t++$

until $t > T$

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

Various rules for updating λ different from the given above can be proposed. For example, another rule is

$$\lambda_k \leftarrow \begin{cases} \lambda_k + 1 - \frac{1-\varepsilon_k}{\varepsilon_k}(1 - \lambda_k), & \varepsilon_k > \zeta_k, \\ \lambda_k + \frac{\varepsilon_k}{1-\varepsilon_k}\lambda_k, & \varepsilon_k \leq \zeta_k. \end{cases}$$

This rule satisfies the aforementioned requirements for weights λ. The choice of the "best" rule depends on a specific classification problem and is a direction for further research.

Note that the obtained probability distribution h again belongs to the set $\mathcal{P}$ because it is the linear combination of the corresponding extreme points of $\mathcal{P}$.

Let us consider now a special case where the set $\mathcal{P}$ is the largest one (the unit simplex). The number of extreme points is $r = n$ and they are of the form:

$$q_k = (0, ..., 1_k, ..., 0),\ k = 1, ..., n.$$

Hence, we get $h = (\lambda_1, \lambda_2, ..., \lambda_n)$. The misclassification rate is $\varepsilon_k = 1$ if the k-th example is misclassified, otherwise $\varepsilon_k = 0$. It can be seen that the misclassification rate is the indicator of the misclassification. The error is

$$H_- = e(t) = \sum_{i:c_t(x_i) \neq y_i} \lambda_k.$$

Hence, there holds

$$\lambda_k \leftarrow \begin{cases} \lambda_k \cdot \exp(\alpha_t), & \text{the } k\text{-th example is misclassified,} \\ \lambda_k \cdot \exp(-\alpha_t), & \text{otherwise.} \end{cases}$$

We have obtained the standard AdaBoost algorithm. This implies the standard AdaBoost is a special case of the proposed algorithm when we do not restrict the set of probability distributions or there is complete ignorance about probabilities of examples.

Conclusion 1. The standard AdaBoost algorithm is based on complete ignorance about probability distributions of examples in the training data.

Let us point out several positive features of the EPIBoost algorithm.

First, the number of extreme points is always larger than the number of examples, i.e., $r \geq n$. This important condition gives us a more sensitive approach because the larger number of adaptation parameters (λ_k) is adjustable.

Second, when we have a few examples, it is difficult to determine the precise error rate ε_k for every example in accordance with the AdaBoost-based algorithms by using testing data. This is caused by a small number of misclassified examples. As a result, the weights are changed very quickly as it is shown in Fig. 1. The algorithm becomes unstable and sensitive to noise or to small changes of examples. By using the EPIBoost algorithm with extreme points, every extreme point is "responsible" for every misclassified example. The error rate ε_k for every extreme point is not zero if there is at least one misclassified example. This is a very important feature of the proposed approach.

Third, arbitrary constraints for the set $\mathcal{P}$ of probability distributions of examples can be introduced as an additional prior information about training set. Moreover, the set $\mathcal{P}$ can be changed in every iteration in accordance with a certain rule, for instance, by means of the Bayesian updating. This gives us a way for developing special modifications of EPIBoost taking into account peculiarities of training data, for instance, the well known imbalance problem.

Forth, when the set $\mathcal{P}$ is parametric, we can look for its optimal parameter or parameters providing the best classification accuracy. On the other hand, the parameters of the set $\mathcal{P}$ can be viewed as a shortcoming of the EPIBoost algorithm because we need to correctly choose them.

3. Imprecise statistical models

We will consider a small part of the well known imprecise statistical models and determine sets of extreme points of the probability sets produced by the models.

3.1. *The linear-vacuous mixture model*

The imprecise ε-contaminated model or linear-vacuous mixture [28] is a class $\mathcal{P}(\varepsilon)$ of probabilities such that for fixed $\varepsilon \in (0, 1)$ and p_i there holds $\mathcal{P}(\varepsilon) = \{(1-\varepsilon)p_i + \varepsilon\pi_i\}$,

where π_i is arbitrary and $\pi_1 + ... + \pi_n = 1$. The rate ε reflects the amount of uncertainty in π [2]. The models can be regarded as an extension of the well known ε-contaminated (robust) model [11] .

We take the probabilities p_i as $p = (1/n, ..., 1/n)$. Hence, the linear-vacuous mixture is defined by probabilities $(1-\varepsilon)n^{-1} + \varepsilon\pi_i$. The k-th extreme point of the set $\mathcal{P}(\varepsilon)$ consists of n elements such that all elements are equal to $(1-\varepsilon)n^{-1}$ and the k-th element is $(1-\varepsilon)n^{-1} + \varepsilon$ for $k = 1, ..., n$.

3.2. *Pari-mutuel models*

Another model is Walley's imprecise pari-mutuel model [28] for which the set of probability distributions is defined as follows:

$$\mathcal{P}_P(\varepsilon) = \{p_i \le (1+\varepsilon)\pi_i\}.$$

The above class is such that the probabilities of events do not exceed a constant multiple of p_i. The upper probability of the i-th point is $(1+\varepsilon)p_i$. The lower probability is

$$\max\{(1+\varepsilon)p_i - \varepsilon, 0\}.$$

The difference between the upper and lower probabilities is always ε. It is constant over all events for which p_i is not too close to 0 or 1.

Let us consider the extreme points of the set $\mathcal{P}_P(\varepsilon)$ under condition that the empirical distribution is $p = (1/n, ..., 1/n)$. The set $\mathcal{P}_P(\varepsilon)$ under this condition has n extreme points. The extreme points consist of $n-1$ elements $(1+\varepsilon)n^{-1}$ and one element $(1+\varepsilon)n^{-1} - \varepsilon$.

Some real application of the pari-mutuel model were studied by Pelessoni et al. [19].

3.3. *Constant odds-ratio models*

Another type of neighborhood models is the so-called constant odds-ratio (π, ε) model. The set of probability distribution for the model is defined as

$$\mathcal{P}_C(\varepsilon) = \{\pi_i/\pi_j \ge (1-\varepsilon)p_i/p_j\}.$$

Here i and j are arbitrary numbers of points from 1 to n.

Walley [28] indicates that the constant odds-ratio model is convenient for statistical applications because its form is not changed by conditioning or statistical updating.

Let us consider the extreme points of the set $\mathcal{P}_C(\varepsilon)$ under condition $p = (1/n, ..., 1/n)$. The set $\mathcal{P}_C(\varepsilon)$ under this condition has $2n$ extreme points. The first n points are

$$\left(\frac{1-\varepsilon}{n(1-\varepsilon)+\varepsilon}, ..., \frac{1}{n(1-\varepsilon)+\varepsilon}, ..., \frac{1-\varepsilon}{n(1-\varepsilon)+\varepsilon}\right),$$

where the element $(n(1-\varepsilon)+\varepsilon)^{-1}$ in the k-th extreme point is located on the k-th position.

The second n points are

$$\left(\frac{1}{n-\varepsilon},...,\frac{1-\varepsilon}{n-\varepsilon},...,\frac{1}{n-\varepsilon}\right),$$

where the element $(1-\varepsilon)/(n-\varepsilon)$ in the k-th extreme point is located on the k-th position.

3.4. *Upper bounded contaminated robust models*

We consider a simple generalization of the vacuous model by introducing upper bounds for the probabilities $\pi_1,...,\pi_n$, namely, we state that $\pi_i \leq \theta$, $i=1,...,n$, where θ is a number between $1/n$ and 1. We denote the reduced set of probability distributions as $\mathcal{P}_U(\theta)$. The lower possible bound for θ is $1/n$ because the set $\mathcal{P}_U(\theta)$ becomes empty when $\theta < 1/n$.

So, the set $\mathcal{P}_U(\theta)$ is produced by the following system of $2n$ inequalities $0 \leq \pi_i \leq \theta$, $i=1,...,n$, and one equality $\pi_1+...+\pi_n=1$. It is not difficult to find all extreme points of the set $\mathcal{P}_U(\theta)$, which depend on the value θ. Let k be an integer from 1 to $n-1$ such that the following condition is fulfilled:

$$\frac{1}{n-k+1} < \theta \leq \frac{1}{n-k}.$$

Then the set $\mathcal{P}_U(\theta)$ has $k\cdot\binom{n}{k}$ extreme points such that every extreme point consists of exactly $n-k$ elements θ, $k-1$ elements 0 and one element $1-(n-k)\theta$.

For example, if $n=3$ and $\theta=0.4$, then $k=1$, and we have three extreme points of the form $(\theta,\theta,1-2\theta)$, i.e.,

$$(0.4,0.4,0.2),\ (0.4,0.2,0.4),\ (0.2,0.4,0.4).$$

Note that points of the form $(\theta,1-\theta,0)$ do not belong to the set $\mathcal{P}_U(\theta)$ in this case because $1-\theta \geq \theta$ and the condition $h_i \leq \theta$ is violated.

3.5. *Lower bounded vacuous models*

Another generalization of the vacuous model is introducing lower bounds for the probabilities $\pi_1,...,\pi_n$, namely, we state that $\pi_i \geq \theta$, $i=1,...,n$, where θ is a number between 0 and $1/n$. We denote the reduced set of probability distributions as $\mathcal{P}_L(\theta)$. The upper possible bound for θ is $1/n$ because the set $\mathcal{P}_L(\theta)$ becomes empty when $\theta \geq 1/n$. We can state that the set $\mathcal{P}_L(\theta)$ is produced by the following system of $2n$ inequalities and one equality: $\theta \leq \pi_i \leq 1$, $i=1,...,n$, $\pi_1+...+\pi_n=1$. The set $\mathcal{P}_L(\theta)$ has n extreme points such that every extreme point consists of exactly $n-1$ elements θ and one element $1-(n-1)\theta$.

4. Numerical experiments

We illustrate the method proposed in this paper via several examples, all computations have been performed using the statistical software R [20]. 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. This measure is often used to quantify the predictive performance of classification methods. So, ACC is an estimate of a classifier's probability of a correct response, and it is an important statistical measures of the performance of a binary classification test. It can formally be written as

$$ACC = \frac{N_T}{N}.$$

where 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 EPIBoost algorithm as ACC_{imp} 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 Data Set [9], Pima Indian Diabetes [9], Breast Tissue Data Set, Notterman Carcinoma gene expression data [18]. All data sets except for the Notterman Carcinoma data are from the UCI Machine Learning Repository [9]. The following is a brief introduction about these datasets, while more detailed information can be found from, respectively, the data resources.

The "Haberman's Survival Data Set" from the UCI Machine Learning Repository contains cases from a study on the survival of patients who had undergone surgery for breast cancer. The number of features (attributes) is three: age of patient at time of operation (x_1), patient's year of operation (x_2), and the number of positive axillary nodes detected (x_3). The classes are defined by the survival status of every patient ($y = -1$: the patient survived 5 years or longer; $y = 1$: the patient died within 5 year). The total number of examples is 306, of which the number of examples with $y = -1$ is 225.

The "Pima Indian Diabetes" data set, also from the UCI Machine Learning Repository [9]. The Pima data set has eight features ($m = 8$), with 768 training instances (examples) of which 500 are labeled as positive ($y = 1$). Every example is characterized by numerical-valued features.

The "Breast Tissue Data Set" from the UCI Machine Learning Repository [9] contains 106 instances are characterized by 9 numerical-valued feature. Every instance corresponds to one of the six classes of freshly excised tissue studied using electrical impedance measurements: carcinoma, fibro-adenoma, mastopathy, glandular, connective, adipose. The first two classes (carcinoma, fibro-adenoma) are contain 36 instances and are labeled as negative ($y = -1$).

The Notterman Carcinoma Data [18] is a gene expression data which contains 36 samples collected from patients. Among them, 18 normal tissues (labelled as positive) and 18 carcinomas (labeled as negative). A total of 7457 genes

were selected for classifying. The corresponding data resource is http://genomics-pubs.princeton.edu/oncology. For an extensive description of the data set, see [18].

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. The linear-vacuous mixture model is used for the experiments because this model is the most simple and is explainable in a simple way as a contaminated model.

In all these examples the weighted SVM as a classifier has been used. The linear kernel is exploited for the Breast Tissue Data Set. The RBF kernels with parameters 22 and 16 and 0.005 are used for the Pima Indian Diabetes data set, Haberman's Survival Data Set and the Notterman Carcinoma Data, respectively.

The first goal of the EPIBoost method evaluation is to investigate the stability of the proposed algorithm. One of the problems solving by the imprecise method is to reduce the large variations of weights in each iteration. Therefore, the first experiment is intended to compare the performances of the proposed EPIBoost and the standard AdaBoost. Fig. 2 shows how the accuracy measures for the Breast Tissue Data Set are changed in each iteration. We take $T = 10$ iterations and the corresponding accuracy measures are independently computed in each iteration. 6 curves are depicted in Fig. 2 such that every curve is determined for a value of rate ε in the linear-vacuous mixture model. Curve 1 corresponds to $\varepsilon = 1$ (the standard AdaBoost), curve 6 corresponds to $\varepsilon = 0$ (precise probability distributions of examples or the weighted SVM), curves 2, 3, 4, 5 correspond to values 0.8, 0.6, 0.4, 0.2 of ε, respectively. One can see that curve 6 is constant because there is only one probability distribution and the weights of examples cannot be changed. This is an "extreme" case which results the very small accuracy. Another "extreme" case is when $\varepsilon = 1$. In this case, we have the standard AdaBoost. It can be seen from the Fig. 2 that curve 1 is quite unstable. This implies that there are large variations of weights in each iteration. On the one hand, the standard AdaBoost algorithm takes many iterations to get a reasonable accuracy. On the other hand, a large number of iterations may lead to a very complex composite classifier, which is significantly less accurate than a single classifier, i.e., it may lead to overfitting again. This shortcoming of the standard AdaBoost has been pointed out by many authors (see, for example, [23]). One possible way to avoid overfitting is to keep the number of iterations as small as possible. However, the unstable "behavior" is an obstacle for constructing a strong classifier. At the same time, we can see that curves 2, 3, 4, 5 "behave" much better in comparison with curve 1. This is caused by restrictions introduced for the weights by means of the linear-vacuous mixture model. This implies that the approach underlying the EPIBoost algorithm may be a way to avoid overfitting.

The similar picture is shown in Fig. 3 where the accuracy measures under the same conditions are given for the "Pima Indian Diabetes" data set. However, it is difficult to say here whether the case $\varepsilon = 1$ provides worse results or not. One can

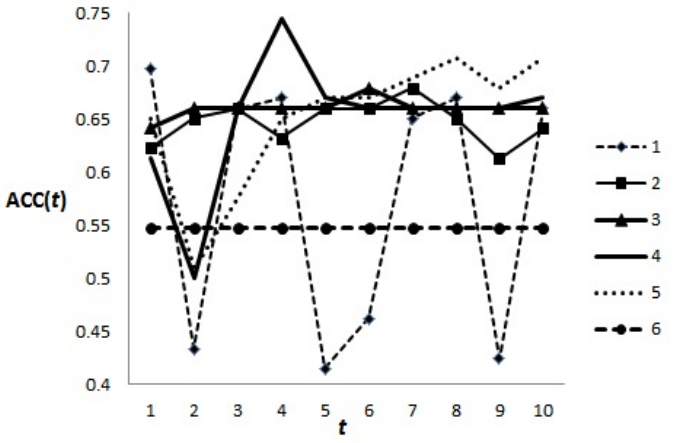

Fig. 2. Accuracy of each iteration for the Breast Tissue Data Set by different ε

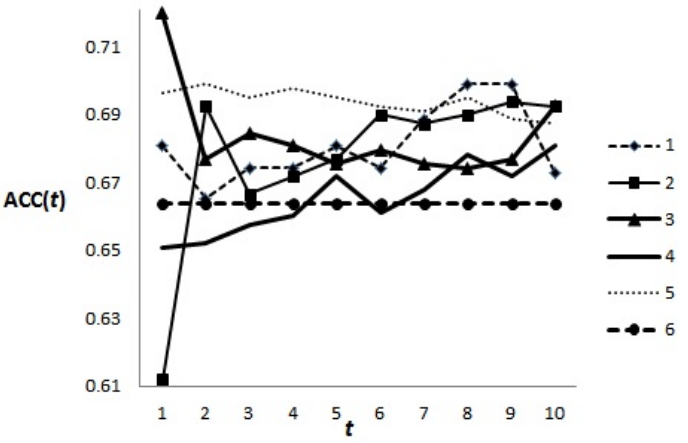

Fig. 3. Accuracy of each iteration for the Pima Indian Diabetes data set by different ε

conclude that EPIBoost algorithm for different ε as well as the standard AdaBoost "behave" similarly.

The second goal of the EPIBoost method evaluation is to investigate the performance of the proposed algorithm by different values of the parameter ε in the linear-vacuous mixture model. Therefore, the second experiment is performed to evaluate the effectiveness of the proposed method under different values of ε and different numbers of training examples. First, we study the accuracy measure of the EPIBoost algorithm as a function of ε by using $T = 10$ iterations and by different sizes of the training set. Fig. 4 illustrates how the accuracy depends on ε by training

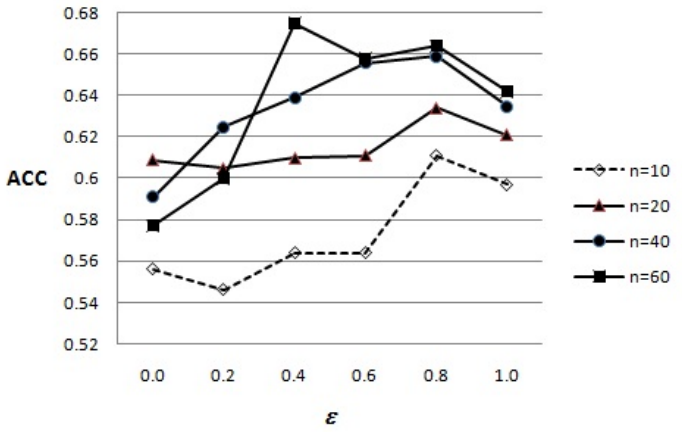

Fig. 4. Accuracy by different ε and n for the Breast Tissue Data Set

$n = 10, 20, 40, 60$ randomly selected examples from the Breast Tissue Data Set. It is interesting to note that the largest values are achieved at $\varepsilon = 0.8$ except for the case $n = 60$. We can also observe that the standard AdaBoost ($\varepsilon = 1$) provides worse accuracy values than the EPIBoost algorithm. It is also very important to note that the proposed method gives low accuracy values by $\varepsilon = 0$ for all n. This again confirms that the adaptive methods like AdaBoost provide better results in comparison with "precise" methods.

The same dependencies for the Pima Indian Diabetes data set are depicted in Fig. 5. One can see from the figure that the accuracy for this data set weakly depends on the parameter ε for relatively large training sets ($n = 40$ and 60), but it strongly depends on ε for the very small training set ($n = 10$). Moreover, the standard AdaBoost has strongly smaller values of the accuracy than the imprecise method by small training sets ($n = 10$ and 20). The above implies that, comparing to the standard AdaBoost method, the proposed algorithm is more applicable for the cases having the limited number of data samples.

The same dependencies for Haberman's Survival Data Set are shown in Fig. 6.

Figs. 4-6 show in particular that the EPIBoost algorithm can provide a reasonable performance even in cases of very small amounts of data (see curves by $\varepsilon = 0.8$), which was one of the main reasons to consider the imprecise models.

Fig. 7 illustrates how the accuracy of the proposed algorithm depends on ε for Haberman's Survival Data Set by different numbers of iterations $T = 10, 20, 30$ and by $n = 40$. It can be seen from Fig. 7 that the large number of iterations may reduce the accuracy due to overfitting (see the curve for $T = 30$) for the large values of ε, i.e., for the standard AdaBoost. However, this fact does not take place for $\varepsilon = 0.2$ and 0.4. The case $\varepsilon = 0$ does not depend on the number of iterations.

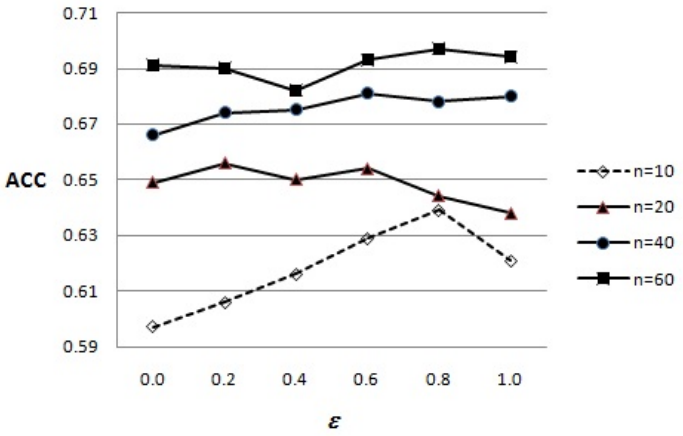

Fig. 5. Accuracy by different ε and n for the Pima Indian Diabetes data set

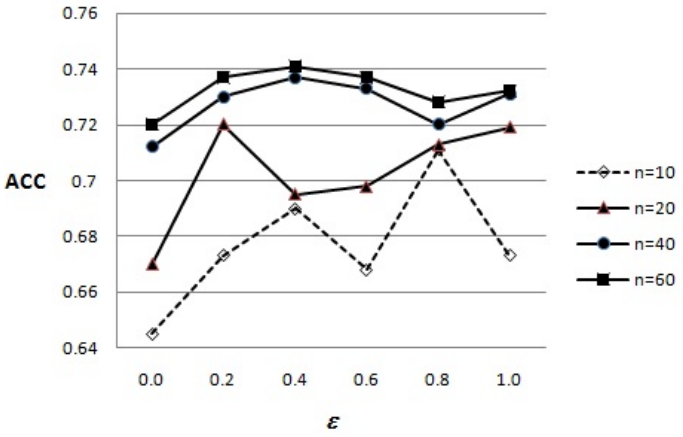

Fig. 6. Accuracy by different ε and n for Haberman's Survival Data Set

Fig. 8 illustrates the performance of the proposed method for the Notterman Carcinoma gene expression data by taking $T = 20$ iterations. It can be seen from Fig. 8 that the accuracy strongly depends on the parameter ε by small training sets. However, its impact is very weak when there are available many training data (see the case of $n = 32$).

The same dependencies for $T = 10$ are depicted in Fig. 9.

An important question is how the choice of initial vector of weights may impact on the accuracy measure. In order to study the question we compare the accuracy measures for Haberman's Survival Data Set by different values of ε. The number of

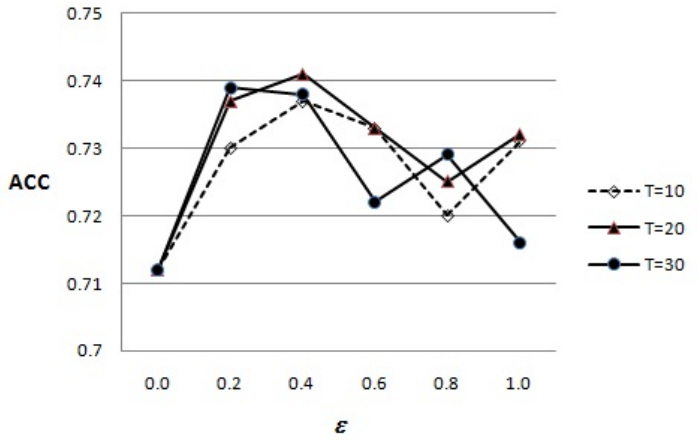

Fig. 7. Accuracy by different ε and T for Haberman's Survival Data Set

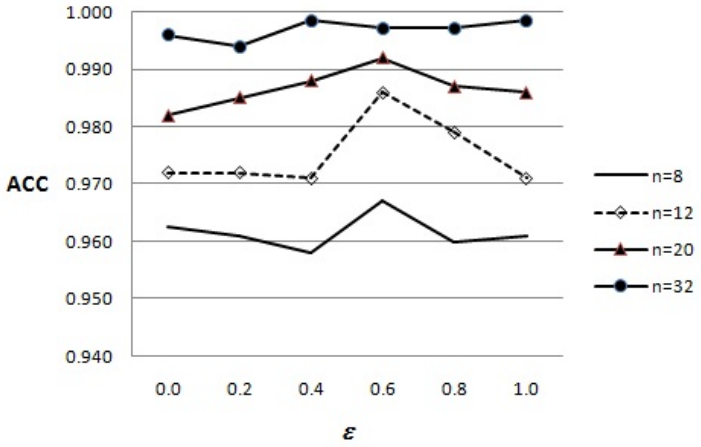

Fig. 8. Accuracy by different ε and n for the Notterman Carcinoma Data by $T = 20$

iterations is 20, the number of examples is 40. Fig. 10 illustrates the performance of the proposed method by different values of ε for the identical initial weights of examples (curve 1), random weights uniformly selected from the unit simplex (curve 2) and random weights uniformly selected from the set $\mathcal{P}(\varepsilon)$ (curve 3). One can see that there is no a considerable difference between ways for selecting the initial vector of weights. However, we see that the third curve differs from the others in the area of large values of ε when the set $\mathcal{P}(\varepsilon)$ is rather large. The same relationship can be observed for different parameters n and T.

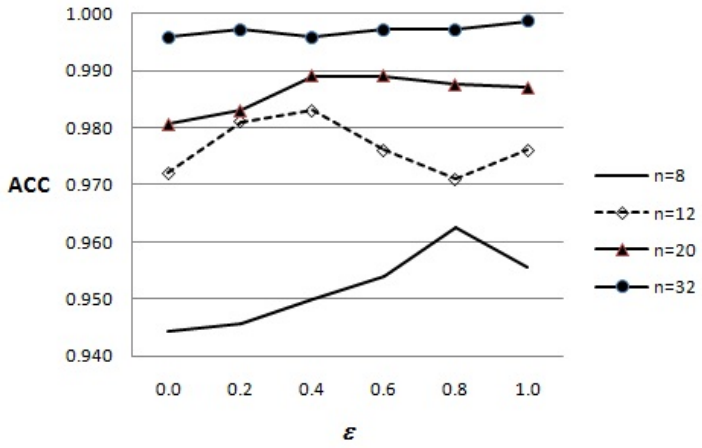

Fig. 9. Accuracy by different ε and n for the Notterman Carcinoma Data by $T = 10$

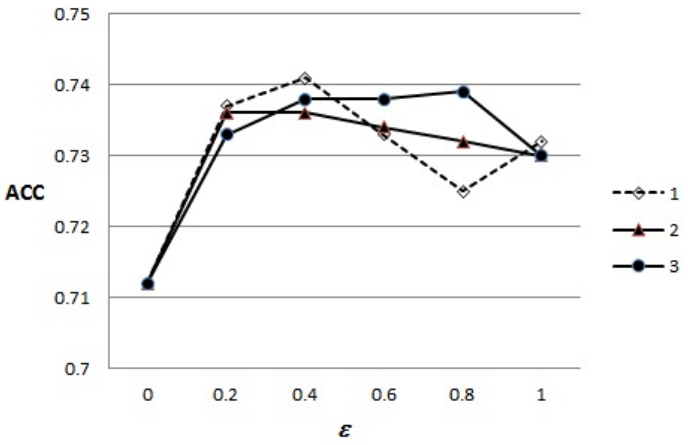

Fig. 10. Accuracy by different ε and by different initial vectors of weights for Haberman's Survival Data Set

5. Conclusion

In this paper we have presented a new approach for ensemble construction based on restricting a set of weights of examples in training data by applying the imprecise statistical models. The goal of the EPIBoost algorithm is to avoid the problem of overfitting and to take into account some information about training data, for instance, their number. One of the ideas underlying the algorithm is updating the probability distribution of examples within the restricted set. It is carried out by replacing the probability distributions of examples by weights in the linear com-

bination of extreme points of the restricted set, which, in turn, can move within the unit simplex. The proposed approach is rather general because it allows us to construct various algorithms by applying different imprecise statistical models or different rules for producing the restricted set.

The EPIBoost is simple from the computation point of view. In fact, it differs from the standard AdaBoost in two steps: computing the probability distributions h and computing the error rate ε_t. This does not mean that other steps coincide with the standard AdaBoost algorithm. They are just similar from the complexity point of view.

It should be noted that the EPIBoost is close to the boosting algorithms which restrict sets of possible weights, for instance, NadaBoost [17] and MadaBoost [7]. On the one hand, these algorithms can be regarded as special cases of the EPIBoost. On the other hand, in contrast to the NadaBoost or MadaBoost algorithms, the EPIBoost does not directly restricts weights of examples. It works with r "virtual" examples (extreme points) having unrestricted weights λ which can be changed inside the unit simplex, but it does not work with "real" examples having weights h. As a result, the updating rule is applied to the weights λ, but not to the weights h. This allows us to use various imprecise statistical models by constructing a set of the boosting algorithms.

In order to numerically evaluate the EPIBoost algorithm, we have applied the linear-vacuous mixture or imprecise ε-contaminated model and the SVM as the weak classifier. The choice of the imprecise ε-contaminated model is caused by the fact that this statistical model is the most simple and explainable in a simple way. Moreover, values of its parameter ε produce a variety of probability sets from the largest one (the unit simplex) by $\varepsilon = 1$ till a single distribution by $\varepsilon = 0$. Therefore, we can compare the standard AdaBoost, the standard classifier (SVM) and the EPIBoost algorithm by changing the parameter ε.

The comparison was made using some sets of real data, including Haberman's Survival Data Set, Pima Indian Diabetes, Breast Tissue Data Set, Notterman Carcinoma gene expression data. The experimental results showed that there are certain values of the parameter ε providing the better accuracy. In many cases, these values differ from 1. The above says that the EPIBoost algorithm may outperform the standard AdaBoost if to make a suitable choice of ε.

Additional experiments have been made to test the stability of the proposed algorithm at each iteration. It has been shown again that the EPIBoost algorithm may be more stable in comparison with the standard AdaBoost by some values of the parameter ε.

We believe that this work opens ways for future work which is mainly directed towards three goals. Firstly, we are working on developing a set of models dealing with imbalanced data. Secondly, we are working on models which allow us to change the restricted set at each iteration of the boosting algorithm. One of the ideas in this direction is to apply Walley's imprecise Dirichlet model [29]. Thirdly, we say that the reduce of the unit simplex of weights h to the set $\mathcal{P}$ can improve the classification

algorithm. One the other hand, we use the precise probability distribution h for training of the weak classifier whose choice might be wrong in the case of the small training set. In order to overcome this difficulty, we are working on models that use "small" sets of h and minimax weak classifiers.

Finally, it is necessary to note that the choice of the best imprecise statistical model in specific classification problems is still an open question. One of the ways is to apply all models and to compare the obtained accuracy measures in order to select the most accurate classifier. A search of a more optimal way is also a problem for future research.

Acknowledgement

I would like to express my appreciation to the anonymous referees whose very valuable comments have improved the paper.

References

1. A. Antonucci, M. Cattaneo and G. Corani. Likelihood-based robust classification with Bayesian networks. In *Advances in Computational Intelligence*, pages 491–500. Springer, Berlin Heidelberg, 2012.
2. J. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer-Verlag, New York, 1985.
3. T. Bylander and L. Tate. Using validation sets to avoid overfitting in AdaBoost. In *Proceedings of the 19th International FLAIRS Conference*, pages 544–549, 2006.
4. V. Cherkassky and F. Mulier. *Learning from Data: Concepts, Theory, and Methods*. Wiley-IEEE Press, UK, 2007.
5. G. Corani and M. Zaffalon. Learning reliable classifiers from small or incomplete data sets: The naive credal classifier 2. *Journal of Machine Learning Research*, 9:581–621, 2008.
6. R. Dara, M. Kamel, and N. Wanas. Data dependency in multiple classifier systems. *Pattern Recognition*, 42(7):1260 – 1273, 2009.
7. C. Domingo and O. Watanabe. MadaBoost: A modification of AdaBoost. In *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, pages 180–189, San Francisco, 2000. Morgan Kaufmann.
8. A. Ferreira and M. Figueiredo. Boosting algorithms: A review of methods, theory, and applications. In *Ensemble Machine Learning: Methods and Applications*, pages 35–85. Springer, New York, 2012.
9. A. Frank and A. Asuncion. UCI machine learning repository, 2010.
10. Y. Freund and R. 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.
11. P. Huber. *Robust Statistics*. Wiley, New York, 1981.
12. R. Jin, Y. Liu, L. Si, J. Carbonell, and A. Hauptmann. A new boosting algorithm using input-dependent regularizer. In *Proceedings of Twentieth International Conference on Machine Learning (ICML 03)*, pages 1–9, Washington DC, 2003. AAAI Press.
13. L. Kuncheva. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, New Jersey, 2004.
14. A. Lorbert, D. M. Blei, R. E. Schapire, and P. J. Ramadge. A bayesian boosting model. *ArXiv e-prints*, 2012.

15. Y. Lu, Q. Tian, and T. Huang. Interactive boosting for image classification. In *Proceedings of the 7th International Conference on Multiple Classifier Systems, MCS'07*, pages 180–189, Berlin, Heidelberg, 2007. Springer-Verlag.
16. D. Mease and A. Wyner. Evidence contrary to the statistical view of boosting. *Journal of Machine Learning Research*, 9:131–156, 2008.
17. M. Nakamura, H. Nomiya, and K. Uehara. Improvement of boosting algorithm by modifying the weighting rule. *Annals of Mathematics and Artificial Intelligence*, 41:95–109, 2004.
18. D. Notterman, U. Alon, A. Sierk, and A. Levine. Transcriptional gene expression profiles of colorectal adenoma, adenocarcinoma, and normal tissue examined by oligonucleotide arrays. *Cancer Research*, 61:3124–3130, April 2001.
19. R. Pelessoni, P. Vicig and M. Zaffalon. Inference and risk measurement with the pari-mutuel model. *International Journal of Approximate Reasoning*, 51(9):1145–1158, 2010.
20. R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2005.
21. M. Re and G. Valentini. Ensemble methods: a review. In *Data Mining and Machine Learning for Astronomical Applications*, Data Mining and Knowledge Discovery Series, chapter 26, pages 563–594. Chapman & Hall, 2012.
22. G. Ridgeway. *GBM 1.5 package manual*, 2005.
23. L. Rokach. Ensemble-based classifiers. *Artificial Intelligence Review*, 33(1-2):1–39, 2010.
24. D. K. R.Y. Rubinstein. *Simulation and the Monte Carlo method, 2nd Edition*. Wiley, New Jersey, 2008.
25. B. Scholkopf and A. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. The MIT Press, Cambridge, Massachusetts, 2002.
26. R. Servedio. Smooth boosting and learning with malicious noise. *Journal of Machine Learning Research*, 4(9):633–648, 2003.
27. V. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.
28. P. Walley. *Statistical Reasoning with Imprecise Probabilities*. Chapman and Hall, London, 1991.
29. P. Walley. Inferences from multinomial data: Learning about a bag of marbles. *Journal of the Royal Statistical Society, Series B*, 58:3–57, 1996. with discussion.
30. M. Warmuth, K. Glocer, and G. Ratsch. Boosting algorithms for maximizing the soft margin pages. In *Advances in Neural Information Processing Systems NIPS*, pages 1–8. MIT Press, 2007.
31. A. Webb. *Statistical Pattern Recognition, 2nd Edition*. Wiley, 2002.
32. M. Zaffalon. Credible classification for environmental problems. *Environmental Modelling & Software*, 20:1003–1012, 2005.
33. T. Zhang and B. Yu. Boosting with early stopping: Convergence and consistency. *Annals of Statistics*, 33(4):1538–1579, 2005.