Expert Systems With Applications 141 (2020) 112978

Contents lists available at ScienceDirect

Expert Systems With Applications

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

An imprecise deep forest for classification

Lev V. Utkin*

Department of Telematics, Peter the Great St. Petersburg Polytechnic University, St. Petersburg, Russia

ARTICLE INFO

Article history:
Received 7 July 2018
Revised 1 September 2019
Accepted 25 September 2019
Available online 27 September 2019

Keywords:
Classification
Random forest
Decision tree
Deep forest
Imprecise Dirichlet model
Imprecise probabilities

ABSTRACT

An imprecise deep forest classifier, which can be regarded as a modification of the deep forest proposed by Zhou and Feng, is presented in the paper. In the proposed classifier, the precise class probabilities at leaf nodes of decision trees in the deep forest are replaced with interval-valued probabilities produced by Walley's imprecise Dirichlet model. The obtained sets of probabilities are averaged over all trees in a random forest as the imprecise probability masses. In order to use the stacking algorithm in the deep forest, a number of the class probability distributions are generated from sets of the random forest class probabilities such that decision trees of the next level of the deep forest cascade are trained on the basis of the extended training set which is added by new generated class distributions. Numerical examples illustrate how the incorporated imprecision may lead to outperforming results, especially, when the training set is rather small. The proposed classifier is the first modification of the deep forest, which efficiently deals with small datasets and takes into account a lack of sufficient training data. The ideas underlying the classifier may be a basis for development of a set on new machine learning models for small datasets.

1. Introduction

The ensemble methodology can be regarded as one of the efficient machine learning approaches to classification and regression. A lot of ensemble-based methods developed last decades are simple powerful algorithms that are easy to implement and provide excellent results. These methods are based on constructing the so-called weak or base classifiers from training data and on aggregating their predictions when classifying unknown samples in order to obtain a strong classifier that outperforms every one of them. The formal definitions of weak and strong classifiers were introduced by Schapire (1990). A comprehensive description of ensemble-based methods is presented in Zhou's book (Zhou, 2012).

Most ensemble-based models can be conditionally divided into three groups which differ by aggregation techniques: bagging, stacking and boosting. The first group includes bagging methods (Breiman, 1996) improving accuracy by means of combining multiple classifiers based on bootstrapped samples. Classifiers are usually trained by using multiple different training sets produced from an original training set. One of the most well-known and effective bagging models is the random forest (RF) (Breiman, 2001), which uses a large number of randomly built individual decision trees in order to combine their predictions. RFs reduce the possible correlation between decision trees by selecting different subsamples of the feature space. Another interesting ensemble-based technique is the stacking algorithm (Wolpert, 1992). It combines different base classifiers by means of a meta-learner that takes into account which classifiers are reliable and which are not. One of the stacking models is when outputs of the base classifiers are viewed as features of the training feature vectors for the meta-learner. The third technique is boosting. It also improves the performance of weak classifiers by means of their combining into a single strong classifier by using voting for combining the weak classifiers. The voting mechanisms in bagging and boosting are differently implemented.

A lot of ensemble-based methods have been developed in last decades. Various reviews devoted to foundations and applications of the ensemble-based models can be found in the literature (Fawagreh, Gaber, & Elyan, 2014; Ferreira & Figueiredo, 2012; Jurek, Bi, Wu, & Nugent, 2014; Kuncheva, 2004; Polikar & Ma, 2012; Ren, Zhang, & Suganthan, 2016; Rokach, 2010; Wozniak, Grana, & Corchado, 2014; Yang, Yang, Zhou, & Zomaya, 2010).

A new ensemble-based method called the deep forest or gcForest was proposed in Zhou and Feng (2017). It combines several ensemble-based methods, including RFs and stacking. Its structure is similar to a multi-layer neural network structure, but each layer in gcForest contains many RFs instead of neurons. As pointed out in Zhou and Feng (2017), gcForest is much easier to train in comparison with deep neural networks which require great effort in hyperparameter tuning and large-scale training data. It can also perfectly work when there are only small-scale training data.

* Corresponding author.
E-mail address: utkin_lv@spbstu.ru

https://doi.org/10.1016/j.eswa.2019.112978
0957-4174/

Following the pioneering work (Zhou & Feng, 2017), modifications of the deep forests have been proposed (Guo, Liu, Li, & Shang, 2017; 2018; Miller, Hettinger, Humpherys, Jarvis, & Kartchner, 2017; Utkin, Kovalev, & Meldo, 2019; Utkin & Ryabinin, 2018; Wen et al., 2018; Zhang et al., 2018). All the modifications and applications of the deep forests are based on applying a class probability distribution which is produced by every tree for each input example. The probability distribution is computed by counting the percentage of different classes of examples at the leaf node where the concerned example falls into. The combination of the class distributions forms the RF class vector which is used in a special stacking scheme. In other words, the class probability distributions play a crucial role in the deep forest. The main problem of these distributions is that they are assumed to be precise. However, it is difficult to expect that they can be determined in the precise way, especially, by a small amount of training data. One of the ways to overcome this difficulty is to use the imprecise probabilities or imprecise statistical inference models (Walley, 1991).

One of the first ideas of applying the imprecise probability theory to decision trees was presented in Abellan and Moral (2003) where probabilities of classes at decision tree leaves are estimated by using an imprecise model, and the so-called Credal Decision Tree model is proposed. Following this work, several papers devoted to applications of imprecise probabilities to decision trees and RFs were presented (Abellan, 2013; Abellan, Mantas, & Castellano, 2017; Abellan, Mantas, Castellano, & Moral-Garcia, 2018; Mantas & Abellan, 2014), where the authors developed new splitting criteria taking into account imprecision of training data and the noise data. In particular, the authors consider the application of the Walley's imprecise Dirichlet model (IDM) (Walley, 1996). The IDM can be regarded as a set of all Dirichlet distributions restricted by some set of its parameters which are defined by means of the common Bayesian updating procedure. The main advantage of the IDM in its application to the classification problems is that it produces a convex set of probability distributions, which has nice properties and depends on a number of observations. Another interesting RF called the fuzzy RF is proposed in Bonissone, Cadenas, Garrido, and Diaz-Valladares (2010). Imprecise probabilities have been also used in classification problems in Destercke and Antoine (2013).

In spite of many interesting results obtained by applying the imprecise probabilities to decision trees and RFs, there are no publications studying the propagation of imprecise probabilities through levels of the deep forest cascade and their usage in the stacking algorithm. Therefore, a new algorithm for using the imprecise probabilities, in particular, the IDM, in the deep forest is proposed. It is called the Imprecise Deep Forest (ImprDF). The main ideas underlying the ImprDF are the following:

1. The precise class probabilities at a leaf node of a decision tree in the deep forest are replaced with interval-valued probabilities such that every interval is produced by the IDM and strongly depends on the number of training examples which fall into the leaf node. This replacement allows us to take into account a small amount of training examples. Moreover, if training examples of a certain class are absent at a leaf node, then the probability of the class is not zero in this case. This is also very important to avoid the overfitting.
2. The obtained sets of probabilities produced by the IDM are averaged over all trees in a RF in order to get a set of class probabilities for the RF.
3. In order to use the stacking algorithm in the deep forest, a number of the class probability distributions are generated from the set of the RF class probabilities. Decision trees of the next level of the forest cascade are trained on the basis of the extended training set which is added by new generated class distributions. The increased number of the training examples is compensated by an updated hyperparameter of the IDM.

All parts of the proposed algorithm are simple from the computational point of view. Numerical examples show that the incorporated imprecision may lead to outperforming results, especially, when the training set is rather small or imbalanced.

The paper is organized as follows. A short description of gcForest is given in Section 2. Precise probabilities of classes for trees and imprecise probability masses represented in the form of probability intervals as well as their properties are considered in Section 3. A definition and properties of Walley's IDM are provided in Section 4. The main algorithm of the ImprDF and the application of the IDM in the deep forest are considered in Section 5. A question of generation of the random class probabilities for extending the training set and for using the extended set at the next level of the forest cascade is studied in Section 6. A general algorithm for training the ImprDF is represented in the same section. An algorithm for testing is given in Section 7. Numerical experiments with real data illustrating cases when the proposed ImprDF outperforms gcForest are given in Section 8. Concluding remarks are provided in Section 9.

2. A short introduction to the deep forest

Before considering the ImprDF, we briefly introduce gcForest (Zhou & Feng, 2017). The gcForest algorithm can be divided into two parts. The first part is the so-called Multi-Grained Scanning structure which uses sliding windows to scan the raw features. Its output is a set of feature vectors produced by sliding windows of multiple sizes. The second part of the gcForest is a cascade forest structure where each level of a cascade receives feature information processed by its preceding level, and outputs its processing result to the next level (Zhou & Feng, 2017).

One of the important ideas underlying the cascade forest structure is a class distribution produced by every tree for each input example. The distribution is computed by counting the percentage of different classes of examples at the leaf node where the concerned example falls into. It produces a class vector by means of averaging class distributions across all trees in the same forest. The class vector is then concatenated with the original vector to be input to the next level of the cascade.

The use of the class vector as a result of the RF classification is very similar to the idea underlying the stacking method (Wolpert, 1992). The stacking algorithm trains the first-level learners using the original training data set. Then it generates a new data set for training the second-level learner (meta-learner) such that the outputs of the first-level learners are regarded as input features for the second-level learner while the original labels are still regarded as labels of the new training data. In fact, the class vectors in the gcForest can be viewed as the meta-learners. In contrast to the stacking algorithm, the gcForest simultaneously uses the original vector and the class vectors (meta-learners) at the next cascade level by means of their concatenation. This implies that the feature vector is enlarged and enlarged after every cascade level. The architecture of the cascade (Zhou & Feng, 2017) is shown in Fig. 1. It can be seen from the figure that each level of the cascade consists of two different pairs of RFs which generate 3-dimensional class vectors concatenated each other and with the original input. It should be noted that this structure of forests can be modified in order to improve the gcForest for a certain application. After the last level, we have the feature representation of the input feature vector, which can be classified in order to get the final prediction.

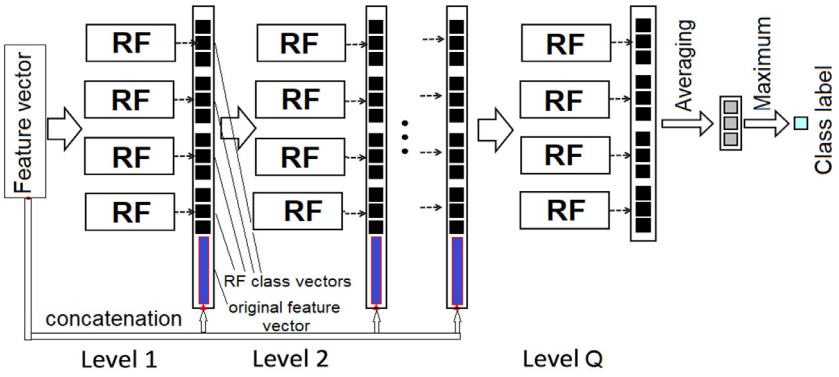

Fig. 1. The architecture of the cascade forest (Zhou & Feng, 2017).

3. Probabilities of classes for trees and imprecise probability masses

The classification problem can be formally written as follows. Given n training data (examples, instances, patterns) $S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$, in which $\mathbf{x}_i \in \mathbb{R}^m$ represents a feature vector involving m features and $y_i \in \{1, \ldots, C\}$ represents the class of the associated examples, the task of classification is to construct an accurate classifier c: $\mathbb{R}^m \to \{1, \ldots, C\}$ that maximizes the probability that $c(\mathbf{x}_i) = y_i$ for $i = 1, \ldots, n$. Generally, $\mathbf{x}_i$ may belong to an arbitrary set $\mathcal{X}$, but we consider a special case for simplicity when $\mathcal{X} = \mathbb{R}^m$.

One of the important peculiarities of decision trees is a probability distribution of classes at each leaf node. This probability distribution is used for computing probabilities of classes for a RF and for making decision about a class label of a testing example.

Formally, each leaf node $l \in CL = \{1, \ldots, L\}$ stores votes for the semantic class labels denoted as $\mathbf{n}_l = (n_{l,1}, \ldots, n_{l,C})$. Here $n_{l,c}$ is the number of feature vectors from the class c which fall into the lth leaf node. This is equivalently to storing a categorical probability distribution over classes $c \in \{1, \ldots, C\}$ in a vector $p_l = (p_{l,1}, \ldots, p_{l,C}) \in [0, 1]^C$. Every vector p_l is normalized such that the probabilities of all classes in a leaf satisfy the condition

$$\sum_{c=1}^{C} p_{l,c} = 1. \tag{1}$$

If the vector $\mathbf{x}$ falls into the lth leaf node in a tree, then the prediction of the decision tree for feature vector $\mathbf{x}$ to be of class c is given by

$$p_{l,c} = \Pr\{c|\mathbf{x}\} = \frac{n_{l,c}}{\sum_{i=1}^{C} n_{l,i}} = \frac{n_{l,c}}{n_l}. \tag{2}$$

Here n_l is the number of all feature vectors which fall into the lth leaf node.

It is obvious that the above estimates of class probabilities cannot be considered precise by a small number of training data. Even we have a lot of training examples, it does not guarantee that a large part of examples fall into a certain leaf node, i.e., n_l is large for all $l \in CL$. This implies that interval-valued or imprecise probabilities $p_{l,c}$ should be taken in place of the precise ones.

3.1. Imprecise probability masses for trees

Let us consider the imprecise probability masses as a simplest framework for formalizing the imprecision of the class probabilities. Detailed descriptions of the imprecise probability masses is provided in de Campos, Huete, and Moral (1994) and Destercke and Antoine (2013). The imprecise probability masses can be represented as a family of intervals $I = \{[a_i, b_i],\ i = 1, \ldots, C\}$ such that $0 \le a_i \le b_i \le 1$ for all $i \in \{1, \ldots, C\}$. For the sake of simplicity, the subscript l, denoting a leaf node number, is dropped when the context is clear. The interval bounds are interpreted as probability bounds over singletons. If a random variable is defined on a finite domain D of C elements, then the intervals from I induce a probability set

$$\mathcal{P} = \{p \in D : a_i \le p_i \le b_i,\ \ i = 1, \ldots, C\}. \tag{3}$$

An important property of the considered intervals is their reachability because a reachable probability interval can be expressed using its tight bounds on probabilities (de Campos et al., 1994). To guarantee that lower and upper bounds are reachable for each element of D by at least one probability in $\mathcal{P}$, the intervals must satisfy the following conditions:

$$\sum_{i \ne j} a_j + b_i \le 1, \quad \sum_{i \ne j} b_j + a_i \ge 1, \ \ \forall i \in \{1, \ldots, C\}. \tag{4}$$

It is shown in de Campos et al. (1994) that the following bounds can be determined for an event A from $\{1, \ldots, C\}$ under condition of the reachable probability interval:

$$a(A) = \max\left(\sum_{i \in A} a_i, 1 - \sum_{i \in A^c} b_i\right), \tag{5}$$

$$b(A) = \min\left(\sum_{i \in A} b_i, 1 - \sum_{i \in A^c} a_i\right). \tag{6}$$

The above bounds $a(A)$ and $b(A)$ are largest lower and smallest upper probability bounds for an event A. If A is a singleton, i.e., $A = \{k\}$, and the lower and upper bounds of all intervals are reachable, then we have tight bounds on probabilities. Moreover, if the lower and upper bounds are not reachable, then they can be transformed to the following reachable bounds:

$$a_k^* = \max\left(a_k, 1 - \sum_{i \ne k} b_i\right), \tag{7}$$

$$b_k^* = \min\left(b_k, 1 - \sum_{i \ne k} a_i\right). \tag{8}$$

3.2. Imprecise probability masses for random forests

If a random forest consists of T decision trees, then the interval probabilities produced by different trees should be combined. If the tth tree has a weight w_t, then the combined lower A_i and upper B_i bounds for the RF class probability intervals are simply obtained as follows:

$$A_i = \sum_{t=1}^{T} w_t a_i^{(t)}, \quad B_i = \sum_{t=1}^{T} w_t b_i^{(t)}, \ \ \forall i \in \{1, \ldots, C\}. \tag{9}$$

The intervals $[A_i,\ B_i]$, $i = 1, \ldots, C$, are reachable if intervals $[a_i^{(t)}, b_i^{(t)}]$, $i = 1, \ldots, C$, are reachable for every $t = 1, \ldots, T$, and the condition $\sum_{t=1}^{T} w_t = 1$ is valid. Indeed, the first condition of the reachability can be simply proved as follows:

$$\sum_{i \neq j} A_j + B_i = \sum_{i \neq j} \sum_{t=1}^{T} w_t a_j^{(t)} + \sum_{t=1}^{T} w_t b_i^{(t)}$$
$$= \sum_{t=1}^{T} w_t \left(\sum_{i \neq j} a_j^{(t)} + b_i^{(t)} \right)$$
$$\leq \sum_{t=1}^{T} w_t \cdot 1 = 1. \quad (10)$$

The second condition can be proved in the same way.

In particular, the property of reachability remains to be valid for intervals $[A_i,\ B_i]$ when all weights are identical, i.e., $w_t = 1/T$ for all $t = 1, \ldots, T$.

It should be noted that there are many rules for combining intervals, but the main ones are the weighted mean, disjunction and conjunction (Destercke & Antoine, 2013). According to the disjunction rule, every RF class probability interval is obtained as a union of the tree class probability intervals. As a result, we may get non-informative RF class probability intervals when the number of trees is rather large. In contrast to the disjunction rule, the conjunction rule is based on intersection of the tree class probability intervals and may produce empty intervals. The weighted mean does not have the above peculiarities. Moreover, it corresponds to the combining rule used in original RFs by precise class probabilities. Therefore, we propose to apply the weighted means to combine the intervals.

4. The imprecise dirichlet model

So far, we have considered the probability bounds without indication how these bounds could be constructed. One of the possible ways for that is to use the IDM.

Let $U = \{u_1, \ldots, u_C\}$ be a set of possible outcomes u_i. Assume the standard multinomial model: n observations are independently chosen from U with an identical probability distribution $\Pr\{u_i\} = p_i$ for $i = 1, \ldots, C$, where each $p_i \geq 0$ and $\sum_{i=1}^{C} p_i = 1$. Denote $\mathbf{p} = (p_1, \ldots, p_C)$. Let n_i denote the number of observations of u_i in n trials, so that $n_i \geq 0$ and $\sum_{i=1}^{C} n_i = n$. Under the above assumptions the random variables $n_1, \ldots, n_C$ have a multinomial distribution.

The Dirichlet $(s,\ t)$ prior distribution for $\boldsymbol{p}$, where $\mathbf{t} = (t_1, \ldots, t_C)$, has the following probability density function (DeGroot, 1970):

$$\pi(\mathbf{p}) = \Gamma(s) \left(\prod_{i=1}^{C} \Gamma(st_i) \right)^{-1} \cdot \prod_{i=1}^{C} p_i^{st_i - 1}. \quad (11)$$

Here the parameter $t_i \in (0,\ 1)$ is the mean of p_i under the Dirichlet prior; the hyperparameter $s > 0$ determines the influence of the prior distribution on posterior probabilities; the vector $\mathbf{t}$ belongs to the interior of the C-dimensional unit simplex denoted by $S(1,\ C)$; $\Gamma(\cdot)$ is the Gamma-function which satisfies $\Gamma(x+1) = x\Gamma(x)$ and $\Gamma(1) = 1$.

It should be noted that the Dirichlet prior is a conjugate density for the multinomial distribution, i.e., the posterior density is also Dirichlet density. This is an important property which defines the computational simplicity of the Dirichlet distribution. Given the data $\mathbf{n} = (n_1, \ldots, n_C)$ the Dirichlet $(s,\ \mathbf{t})$ prior density generates a posterior density function

$$\pi(\mathbf{p}|\mathbf{n}) \propto \prod_{i=1}^{C} p_i^{n_i + st_i - 1}, \quad (12)$$

which is seen to be the probability density function of a Dirichlet $(n+s, \mathbf{t}^*)$ distribution, where $t_i^* = (n_i + st_i)/(n+s)$.

The imprecise Dirichlet model is defined (Walley, 1996) as the set of all Dirichlet $(s,\ \mathbf{t})$ distributions such that $\mathbf{t} \in S(1,\ C)$. For the IDM, the hyperparameter s determines how quickly upper and lower probabilities of events converge as statistical data accumulate. It is defined (Walley, 1996) as a number of observations needed to reduce the imprecision (difference between upper and lower probabilities) to half its initial value. Smaller values of s produce faster convergence and stronger conclusions, whereas large values of s produce more cautious inferences. The detailed discussion concerning the parameter s and the IDM can be found in Bernard (2005) and Walley (1996).

Let n_k denote the observed number of occurrences of u_k in n trials, $n = \sum_{k=1}^{C} n_k$. Then bounds for the predictive probability $P(u_k|\mathbf{n},\ \mathbf{t},\ s)$ under the Dirichlet posterior distribution are (Walley, 1996):

$$\underline{P}(u_k|\mathbf{n}, s) = \frac{n_k}{n+s},\ \overline{P}(u_k|\mathbf{n}, s) = \frac{n_k + s}{n+s}. \quad (13)$$

Before making any observations, $n_k = n = 0$, so that $\underline{P}(u_k|\mathbf{n}, s) = 0$ and $\overline{P}(u_k|\mathbf{n}, s) = 1$ for all $k = 1, \ldots, C$. This is the vacuous probability model. Therefore, by using the IDM, we do not need to choose one specific prior. The IDM can be regarded as a model producing the set $\mathcal{P}_D(s, C)$ of probabilities $\mathbf{p}$ such that

$$\frac{n_k}{n+s} \leq p_k \leq \frac{n_k + s}{n+s},\ k = 1, \ldots, C. \quad (14)$$

A similar imprecise model is the *linear-vacuous mixture* or *imprecise ε-contaminated model* producing the set $\mathcal{P}(\varepsilon, \pi)$ of probabilities $\mathbf{p} = (p_1, \ldots, p_C)$ such that $p_i = (1-\varepsilon)\pi_i + \varepsilon q_i$ for some $\varepsilon \in [0, 1]$ and for all $i \in \{1, \ldots, C\}$, where $\pi = (\pi_1, \ldots, \pi_C)$ is the elicited probability distribution and $(q_1, \ldots, q_C)$ can be any probability distribution in the unit simplex denoted as $S(1,\ C)$ (Walley, 1991). The rate ε reflects how "close" we feel that p must be to π (Berger, 1985). If we assume that $\pi = (n_1/n, \ldots, n_C/n)$, then the set $\mathcal{P}(\varepsilon, \pi)$ coincides with $\mathcal{P}_D(s, n)$, i.e., the imprecise ε-contaminated model provides the same set of probabilities as the IDM. There is a connection between parameters s and ε in this case, which is of the form:

$$\varepsilon = s/(n+s). \quad (15)$$

One of the important properties of the IDM as well as the imprecise ε-contaminated model is that the obtained lower and upper bounds are reachable. This implies that we have the tight bounds for intervals of class probabilities.

5. The IDM and the deep forest

Let us return to the definition of the tree class probability assuming that the ith training example $(\mathbf{x}_i,\ y_i)$ falls into a leaf node with number $l(t)$ of the tth decision tree of a RF. If the vector of stored votes corresponding to this leaf node is $\mathbf{n}_{l(t)} = (n_{l(t),1}, \ldots, n_{l(t),C})$, then we can find bounds for probability $p_{l(t),c}$ defined in (2) in accordance with the IDM. It follows from (14) that the bounds are of the form:

$$a_{l(t),c} = \frac{n_{l(t),c}}{n_{l(t)} + s} \leq p_{l(t),c} \leq \frac{n_{l(t),c} + s}{n_{l(t)} + s} = b_{l(t),c},\ \forall c \in \{1, \ldots, C\}. \quad (16)$$

If we assume that all weights of trees are identical, then, according to (9), bounds for the RF class probabilities are

$$A_c = \frac{1}{T} \sum_{t=1}^{T} a_{l(t),c},\ B_c = \frac{1}{T} \sum_{t=1}^{T} b_{l(t),c},\ \forall c \in \{1, \ldots, C\}. \quad (17)$$

Let us consider a toy numerical example for illustrating the probability bounds obtained by means of the IDM. Suppose that

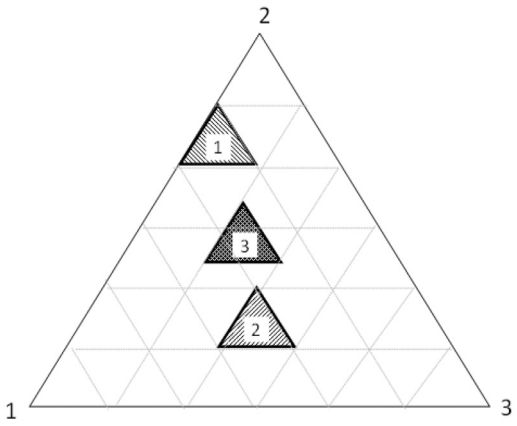

Fig. 2. Illustration of the subsets of class probability distributions for two trees and for the random forest produced by the IDM.

a RF consists of two trees such that 1, 4, 0 examples from three classes, respectively, fall into a leaf node with number $l(1)$ of the first tree, and 2, 1, 2 examples from three classes, respectively, fall into a leaf node with number $l(2)$ of the second tree. If some example $\mathbf{x}$ falls into the same leaf nodes, then we can write bounds for the class probability distributions produced by the first and the second trees. The bounds under condition $s = 1$ for the first tree are

$$[1/6, 2/6];\ \ [4/6, 5/6];\ \ [0/6, 1/6].$$

The bounds for the second tree are

$$[2/6, 3/6];\ \ [1/6, 2/6];\ \ [2/6, 3/6].$$

A very visual and clear representation of imprecise probability distributions is a unit tetrahedron or a unit probability simplex whose edges all have length 1 (see Fig. 2). Every point in the simplex is a representation of a precise probability distribution (p_1, p_2, p_3). Therefore, vertices 1, 2, 3 of the unit probability simplex shown in Fig. 2 correspond to three probability distributions (1,0,0), (0,1,0), (0,0,1), respectively. A subset of probability distributions can be depicted in the unit simplex as a region such that every point of the region corresponds to a probability distribution from the subset. The corresponding subsets of probability distributions produced by the above intervals are illustrated in Fig. 2, where the dashed triangles 1 and 2 correspond to the intervals for the first and the second trees, respectively. A subset of the class probability distributions (the triangle 3) for the RF is obtained by averaging the tree probability distributions (triangles 1 and 2). It is important to note that the set of the RF class probability distributions has the same form as the sets of the tree class probability distributions. This property allows us to simplify the additional feature vectors generation in implementation of the stacking algorithm.

A RF consisting of two decision trees with the imprecise output is illustrated in Fig. 3. It is supposed that $C = 3$. In accordance with the number of examples of every class that fall into certain leaf nodes and the IDM with $s = 1$, bounds for the class probability distributions of the example $\mathbf{x}_i$ are computed. The bounds are taken from the toy numerical example presented above. Examples of the 1-st, 2-nd and 3-rd classes are depicted as small circles, triangles and squares, respectively. It can be seen from Fig. 3 how subsets of the unit simplex of probabilities are produced by the corresponding probability intervals. The RF imprecise output is also depicted as the subset of the unit simplex. It is obtained by averaging the same subsets of two decision trees.

Suppose that the bounds A_c and B_c produce a set of probability distributions $\mathcal{P}_{\text{RF}}(s)$. According to the deep forest architecture, each forest for an example produces an estimate of a class distribution or a class vector, which is then concatenated with the original feature vector to be input to the next level of cascade. The RF class distributions in gcForest are regarded as augmented features which, according to the stacking algorithm (Wolpert, 1992), may significantly improve the classification accuracy of the whole deep forest. Since we use the IDM, then the output of a RF is not the class vector, but a set of probability distributions produced by the bounds (17). We cannot use the set of distributions as augmented features. Therefore, our aim is to develop a way for using the set and for training RF at the next level of the deep forest cascade. The idea underlying the proposed way is to replace a set of distributions $\mathcal{P}_{\text{RF}}(s)$ by a finite set of K random probability distributions drawn from the subset $\mathcal{P}_{\text{RF}}(s)$ of the unit simplex $S(1, C)$. It can be done by the uniform generation of points from $\mathcal{P}_{\text{RF}}(s)$. As a result, every training feature vector is extended by $M \cdot C$ augmented features, where M is a number of RFs at every level of the forest cascade. Every training example is transformed to K new examples with $m + M \cdot C$ features. It should be noted that the generated set of K augmented features of dimension $M \cdot C$ consists of precise probabilities. We denote this set as $\mathcal{F}_q$, where q is the number of the level.

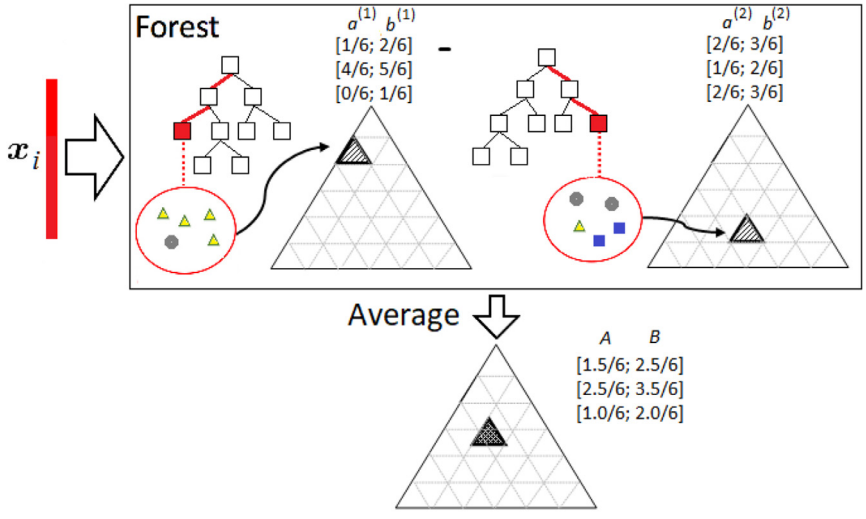

Fig. 3. Illustration of the tree class distributions and the forest class distributions.

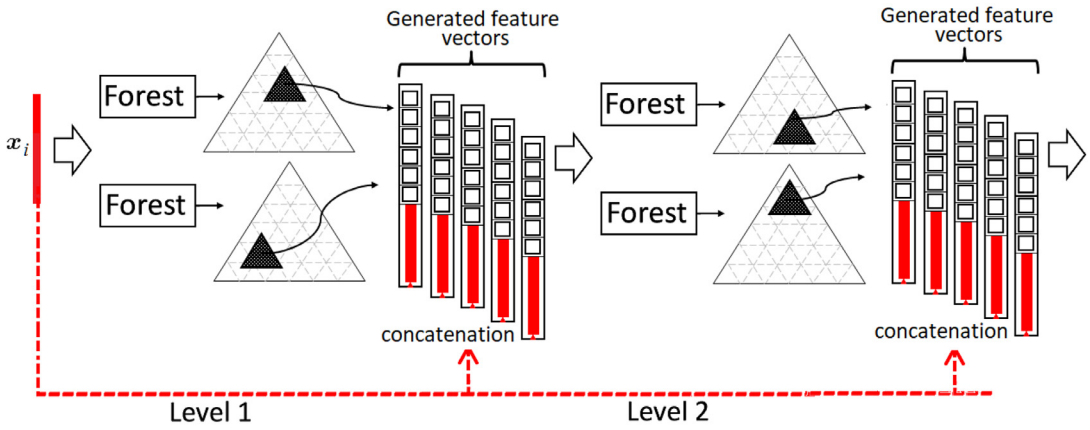

Fig. 4. Two levels of the imprecise deep forest.

The next question is how to use the extended feature vectors at the next cascade level. A problem is that new examples have probabilities different from $1/T$. In spite of the uniformly drawn K probability distributions from the subset $\mathcal{P}_{\rm RF}(s)$, we only know that the sum of K probabilities is $1/T$, but we do not know probabilities of every example. In order to compensate this uncertainty, it is proposed to increase the value of the hyperparameter s such that this parameter for the next level is defined as $s \cdot K$. In sum, the next level has bounds for the class probabilities of the form:

$$A_c = \frac{1}{TK}\sum_{t=1}^{T\cdot K}\frac{n_{l(t),c}}{n_{l(t)}+sK}, \;\; B_c = \frac{1}{TK}\sum_{t=1}^{T\cdot K}\frac{n_{l(t),c}+sK}{n_{l(t)}+sK}, \;\; \forall c \in \{1,\ldots,C\}. \tag{18}$$

Here $n_{l(t)}$ and $n_{l(t),c}$ are defined from analysis of $T \cdot K$ training examples.

We can explain the above in the following way. Let us return to the imprecise ε-contaminated model which has the connection with the IDM. In contrast to the hyperparameter s of the IDM, the parameter ε has a simple and clear explanation. If we suppose that the parameter ε is not changed, then the modified parameter s in accordance with (15) is determined as

$$s = \frac{\varepsilon \cdot n}{1-\varepsilon}. \tag{19}$$

Hence the parameter s should be increased when we replace n with $K \cdot n$ by the same ε.

On the one hand, we increase the number of training data by means of generating new feature vectors from the subsets of probability distributions. On the other hand, we preserve the imprecision of the estimated class probability distributions by increasing the hyperparameter s in the IDM.

Another question is how the increased number of feature vectors impacts on the forest training process. It is important to note that all feature vectors obtained from a training example $(\mathbf{x}_i, y_i)$ have the identical part $\mathbf{x}_i$ and the same class label y_i. Therefore, we have the difference only in the augmented features. Suppose that we have a single class probability distribution or a single concatenated vector of the class probability distributions which are assumed to be precise. How can we believe to these probability distributions when there are only a small number of training examples which fall into a considered leaf node? Moreover, every probability distribution from the subset is under a question. If all new feature vectors are from the same class, then a decision tree "averages" these vectors. Therefore, an average decision may be better than the decision making on the basis of a single questionable example.

Fig. 4 illustrates the idea of generated feature vectors at every level of the forest cascade. Suppose that two RF are used at every level and $C = 3$. The output of every forest is a subset of the unit simplex of probabilities shown in Fig. 4, Random class probability distributions drawn from the first subset form the first part of new feature vectors for the next level. This part consists of C probabilities ($C = 3$ in our example). The second part is formed in the same way by using the imprecise output of the second RF. These feature vectors form the set $\mathcal{F}_1$. After concatenation of the obtained random class distributions and the original vector $\mathbf{x}_i$, we get a set of new feature vectors which are used as inputs for training the RF at the next (second) level.

6. Generation of random class probabilities

The next question is how to generate random class probabilities from the set $\mathcal{P}_{\rm RF}(s)$ produced by bounds A_c and B_c, $c = 1,\ldots,C$, from (17). Due to the reachability property of the bounds, the procedure of generating random class distributions is very simple. Since the set $\mathcal{P}_{\rm RF}(s)$ is a finite-dimensional compact convex set, i.e., it is generated by finitely many linear constraints. This implies that the set $\mathcal{P}_{\rm RF}(s)$ is totally defined by its extreme points or vertices denoted as $\mathcal{E}(\mathcal{P}_{\rm RF})$, i.e., every point from the set can be represented as a linear combination of the extreme points from $\mathcal{E}(\mathcal{P}_{\rm RF})$. It is simply to show that the set $\mathcal{P}_{\rm RF}(s)$ has C extreme points $q_k = (q_1^{(k)},\ldots,q_C^{(k)})$ such that the kth extreme point, $k = 1,\ldots,C$, is of the form:

$$q_k^{(k)} = B_k, \;\; q_i^{(k)} = A_i, \;\; i = 1,\ldots,C, \;\; i \neq k, \tag{20}$$

where A_i and B_i are defined from (17).

The proof of the above is similar to the proof of Proposition 1 in Utkin (2015). Constraints producing the set $\mathcal{P}_{\rm RF}(s)$ have C variables $p_{l(t),1},\ldots,p_{l(t),C}$. This implies that every extreme point is produced by C equalities, including the equality $p_{l(t),1}+\ldots+p_{l(t),C} = 1$. Due to the property (10) of the considered bounds, we get C extreme points (20).

Given C extreme points, every class distribution $p = (p_1,\ldots,p_C)$ from $\mathcal{P}_{\rm RF}(s)$ can be represented as the linear combination of the extreme points

$$p_k = \sum_{i=1}^{C}\lambda_i \cdot q_i^{(k)}, \;\; k = 1,\ldots,C. \tag{21}$$

Here $\lambda = (\lambda_1,\ldots,\lambda_C)$ is a vector of weights such that $\lambda_1+\ldots+\lambda_C = 1$ and $\lambda_k \geq 0$, $k = 1,\ldots,C$.

The main idea of the random point generation is to generate points from the unit simplex of weights λ. This procedure can be

carried out by means of generating the random numbers in accordance with the Dirichlet distribution (Rubinstein & Kroese, 2008). Then the corresponding probability distribution p belonging to $\mathcal{P}_{\text{RF}}(s)$ can be computed by using (21).

An algorithm for training the ImprDF is represented as Algorithm 1. For the sake of simplicity, it is assumed that numbers of trees in every RF and numbers of forests at every level of the forest cascade are the same, respectively.

Algorithm 1 A general training algorithm for the ImprDF.

Require: Training set $S = \{(\mathbf{x}_i, y_i), \quad i = 1, \ldots, n\}, \mathbf{x}_i \in \mathbb{R}^m$, $y_i \in \{1, \ldots, C\}$; number of levels Q; number of forests at every level M; number of trees in every forest T; hyperparameter of the IDM s; number of augmented feature vectors K

Ensure: Trained ImprDF

1: **for** $q = 1$, $q \le Q$ **do**
2: Initialize $S^* \leftarrow \emptyset$
3: **for** $k = 1$, $k \le M$ **do**
4: **for** $t = 1$, $t \le T$ **do**
5: Train the tth tree from the kth RF at the qth level on the basis of S
6: For every $(\mathbf{x}_i, y_i) \in S$, compute bounds for the tree class probability distribution by using (16)
7: **end for**
8: For every $(\mathbf{x}_i, y_i) \in S$, compute bounds for the RF class probability distribution by using (17)
9: **for** $j = 1$, $j \le K$ **do**
10: For every $(\mathbf{x}_i, y_i) \in S$, generate M random vectors λ from the unit simplex $S(1, C)$
11: For every $(\mathbf{x}_i, y_i) \in S$, compute M class distributions $p^{(1)}, \ldots, p^{(M)}$ by using (21) and (22)
12: Concatenate $p \leftarrow (p^{(1)}, \ldots, p^{(M)})$
13: Construct a new feature vector $\mathbf{x}_i^{(j)} \leftarrow (\mathbf{x}_i, p)$
14: $S^* \leftarrow S^* \cup (\mathbf{x}_i^{(j)}, y_i)$
15: **end for**
16: **end for**
17: Update the training set $S \leftarrow S^*$ and the hyperparameters $\leftarrow s \cdot K$
18: **end for**

7. Testing

The prediction or testing phase is carried out in the same way as the training except for the last level, where we make decision about a class of a testing example, and the procedure of averaging the probability intervals for every decision tree. Moreover, we use RFs which have been trained in the training phase. For a testing feature vector $\mathbf{x}$ at the first level of the forest cascade, we get bounds for the forest class probabilities on the basis of (16) and (17) using the trained decision trees (Steps 6 and 8 in Algorithm 1). Then we randomly generate augmented feature vectors (Steps 9–15 in Algorithm 1) and find probabilities intervals for all randomly generated feature vectors.

The next step is to average the class probability intervals obtained for all vectors for every tree. It is important to note that the obtained mean bounds are reachable due to (10). This implies that the intervals have the same properties as the intervals obtained for a single tree. The above procedures are repeated for all levels of the forest cascade. The main difference with the training phase is in the last level of the forest cascade. We propose the following strategy for decision making about the class of $\mathbf{x}$. It selects the class label which corresponds to the largest middle value of the mean intervals, where the mean bounds for probabilities of every class are obtained over all forest outputs at the last level. Suppose we get the bounds for class probabilities $A_c(j)$ and $B_c(j)$, where $j \in \{1, \ldots, M\}$ is the index of the RF. Then the mean bounds are computed as follows:

$$A_c = \frac{1}{M}\sum_{j=1}^{M} A_c(j), \;\; B_c = \frac{1}{M}\sum_{j=1}^{M} B_c(j), \;\; \forall c \in \{1, \ldots, C\}. \tag{22}$$

Then the class of $\mathbf{x}$ is defined as

$$c_{\text{opt}} = \arg\max_{c \in \{1, \ldots, C\}} \frac{A_c + B_c}{2}. \tag{23}$$

It should also be noted that we do need to take into account the different imprecision of the class probability interval because widths of the probability intervals for different classes are identical. Indeed, it follows from (14) that there holds

$$b_k - a_k = \frac{n_k + s}{n + s} - \frac{n_k}{n + s} = \frac{s}{n + s}. \tag{24}$$

An algorithm for testing the ImprDF is represented as Algorithm 2.

Algorithm 2 A general testing algorithm for the ImprDF.

Require: Feature vector $\mathbf{x} \in \mathbb{R}^m$; number of levels Q; number of forests at every level M; number of trees in every forest T

Ensure: Class label y

1: Initialize the set of augmented feature vectors (random probability distributions) $\mathcal{F}_1 = \emptyset$
2: **for** $q = 1$, $q \le Q - 1$ **do**
3: **for** $k = 1$, $k \le M$ **do**
4: **for** $t = 1$, $t \le T$ **do**
5: Compute mean bounds for the tree class probability distribution by using (16) $(\mathbf{x}, p)$ over all $p \in \mathcal{F}_q$.
6: **end for**
7: Perform Steps 8–15 of Algorithm 1, but the set $\mathcal{F}_{q+1}$ is formed instead of S^*
8: **end for**
9: **end for**
10: Compute the mean bounds A_c and B_c in accordance with (22)
11: Define y in accordance with (23) or (25)

Only a single strategy for decision making about the class of the testing example $\mathbf{x}$ has been considered, which is represented by (22). Another important strategy can be regarded as the robust decision making. The strategy selects a class label which corresponds to the largest lower bound of the mean interval. According to the strategy, the class of $\mathbf{x}$ is defined as

$$c_{\text{opt}} = \arg\max_{c \in \{1, \ldots, C\}} A_c. \tag{25}$$

We can use a more general cautious strategy when the class of $\mathbf{x}$ is defined as

$$c_{\text{opt}} = \arg\max_{c \in \{1, \ldots, C\}} \eta A_c + (1 - \eta) B_c. \tag{26}$$

Here η can be viewed as a tuning parameter of the classifier. The above strategies are directions for further research.

8. Numerical experiments

In order to illustrate the ImprDF, we investigate the model for many data sets from UCI Machine Learning Repository (Lichman, 2013). Table 1 is a brief introduction about these data sets, while more detailed information can be found from, respectively, the data resources. Table 1 shows the number of features m for the corresponding data set, the number of examples n and the number of classes C.

Table 1
A brief introduction about data sets.

Data set	Abbreviation	m	n	C
Mammographic masses	MM	5	961	2
Haberman's Survival	HS	3	306	2
Seeds	Seeds	7	210	3
Ionosphere	Ion	34	351	2
Ecoli	Ecoli	8	336	8
Yeast	Yeast	8	1484	8
Parkinson	Park	23	351	2
Glass Identification	Glass	10	214	7
Indian Liver Patient Dataset	ILPD	10	583	2
Car Evaluation	Car	6	1728	4
Waveform Database Generator	Wave	40	5000	3
Soybean (Small)	Soyb	35	47	4
Wholesale Customer Region	WCR	8	440	3
Diabetic Retinopathy	Diab	20	1151	2
Mice Protein Expression	Mice	82	1080	8
Teaching Assistant Evaluation	TAE	5	151	3

The ImprDF has the same cascade structure as the standard gcForest (Zhou & Feng, 2017). The difference is that each level of the cascade structure consists of two RFs. The number of cascade levels is taken 4. The ImprDF uses a software in R implementing the gcForest, which is available at https://github.com/Laurae2/Laurae. The forest cascade is used for numerical experiments without the Multi-Grained Scanning part. Accuracy measure A used in numerical experiments is the proportion of correctly classified cases on a sample of data. To evaluate the average accuracy, we perform a cross-validation with 100 repetitions, where in each run, we randomly select $n_{\text{tr}} = 2n/3$ training data and $n_{\text{test}} = n/3$ testing data. We take the number of generated vectors K equal to $8 \cdot C$.

First, in order to study how the value of the hyperparameter s of the IDM and the number of trees T in every RF impact on the classification accuracy, we study the ImprDF as well as gcForest by different numbers of s and T, namely, we take $s = 0, 1, 2, \ldots, 8$ and $T = 50,\ 100,\ 150,\ 200$, in all forests at all levels. The first data set is MM. Results of numerical experiments for the MM data set are shown in Table 2. It contains the ImprDF accuracy measures as functions of the number of trees T and the hyperparameter s. It should be noted that the case $s = 0$ corresponds to the standard deep forest, i.e., to gcForest.

The corresponding dependencies are depicted in Fig. 5. It is clearly seen from Fig. 5 that the accuracy measure depends on the parameter s. Curves with circular, triangular, cross, diamond markers correspond to the values of $T = 50,\ 100,\ 150,\ 200$, respectively. The same markers will be used for representing numerical results with other data sets. One can see that cases $T = 50$ and $T = 100$ show that the accuracy measure increases with some values of s, and it takes largest values by $s = 4$ ($T = 50$) and $s = 5$ ($T = 100$). It

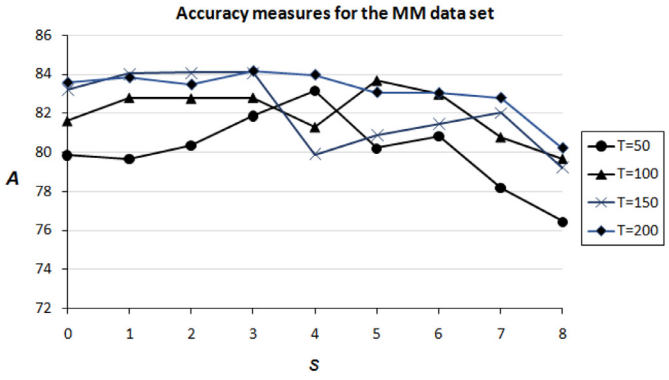

Fig. 5. Accuracy measures for the MM data set as functions of the number of trees T and the IDM parameter s.

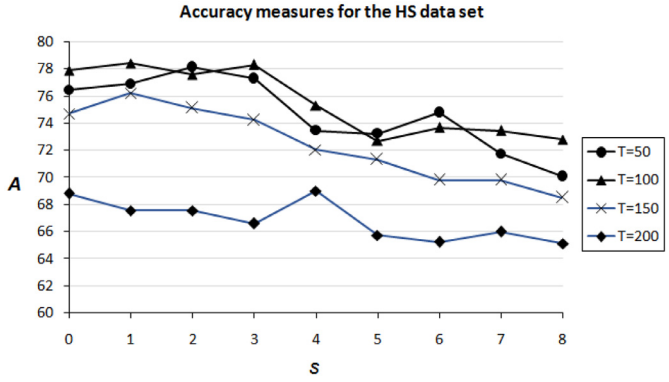

Fig. 6. Accuracy measures for the HS data set as functions of the number of trees T and the IDM parameter s.

is interesting to note that the similar dependence can be observed by $T = 150$ and $T = 200$, but it is not so visible. One can also see from Fig. 5 that the accuracy growth takes place by small values of s. Moreover, the accuracy measure is reduced by large values of s. This implies that the imprecision incorporating may positively impact on the accuracy by certain values of the IDM hyperparameter.

Results of numerical experiments for the HS data set are shown in Table 3 and depicted in Fig. 6. One can see from Fig. 6 that the behavior of the classifier is quite similar. We again have some "optimal" values of s, which provide the largest values of the accuracy measures. It is interesting to note that the ImprDF provides very bad results for $T = 200$ and even for $T = 150$. This is due to the fact that the data set HS is rather small. There are 306 examples in the data set such that 200 examples are for

Table 2
The ImprDF accuracy for the MM data set by different T and s in every forest.

T/s	0	1	2	3	4	5	6	7	8
50	79.85	79.65	80.34	81.86	**83.14**	80.20	80.83	78.17	76.45
100	81.60	82.80	82.76	82.81	81.28	**83.69**	82.98	80.76	79.65
150	83.21	84.02	**84.08**	84.06	79.90	80.86	81.45	82.01	79.20
200	83.55	83.81	83.45	**84.14**	83.95	83.05	83.02	82.77	80.20

Table 3
The ImprDF accuracy for the HS data set by different T and s in every forest.

T/s	0	1	2	3	4	5	6	7	8
50	76.45	76.90	**78.14**	77.32	73.46	73.24	74.8	71.76	70.12
100	77.9	**78.42**	77.60	78.3	75.30	72.70	73.67	73.45	72.80
150	74.72	**76.24**	75.12	74.24	72.04	71.31	69.82	69.82	68.50
200	68.81	67.56	67.56	66.6	**69.02**	65.76	65.23	66.02	65.15

Table 4
The ImprDF accuracy for the Wave data set by different T and s in every forest.

T/s	0	1	2	3	4	5	6	7	8
50	**83.40**	83.80	83.39	83.2	82.10	82.70	84.5	82.46	83.28
100	86.12	**87.25**	85.12	86.43	85.12	85.80	84.12	83.34	82.84
150	87.78	88.17	87.82	**88.54**	84.39	87.45	88.34	86.23	85.70
200	**89.32**	89.30	88.34	89.28	87.72	87.12	86.52	86.78	86.12

Table 5
The ImprDF accuracy for the Glass data set by different T and s in every forest.

T/s	0	1	2	3	4	5	6	7	8
50	84.62	88.35	89.94	91.28	**91.42**	90.14	84.34	84.50	78.32
100	85.03	86.44	84.95	**88.70**	85.18	80.68	76.30	74.69	71.34

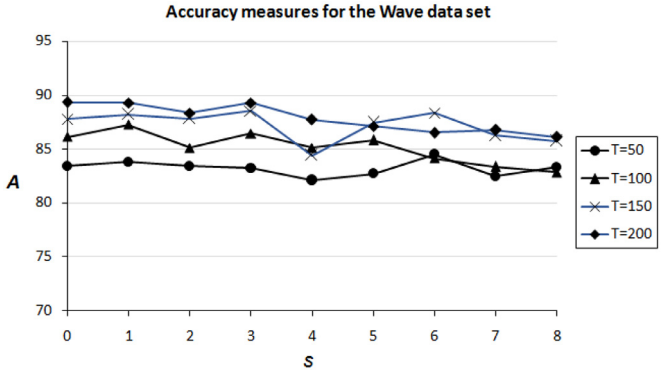

Fig. 7. Accuracy measures for the Wave data set as functions of the number of trees T and the IDM parameter s.

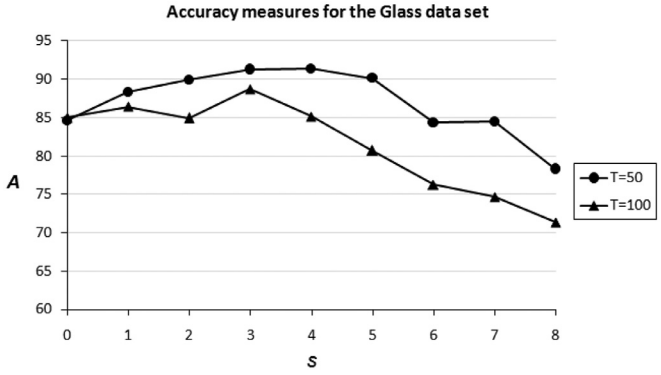

Fig. 8. Accuracy measures for the Glass data set as functions of the number of trees T and the IDM parameter s.

training. This implies that a lot of trees repeat each other in this case.

The next Wave data set is characterized by the large number of examples ($n = 5000$). Results of numerical experiments for the Wave data set are shown in Table 4 and depicted in Fig. 7. It can be seen from the numerical results that the value s does not practically impact on the accuracy measures. One can observe that the best result is obtained for $s = 0$ and $T = 200$. This is due to the large amount of training data. Moreover, such the behavior of the accuracy measures comes to agreement with the IDM, in particular, with (14). Indeed, the imprecision of class probabilities is very small when the number of examples is very large.

In contrast to the Wave data set, the Glass data set is very small and the number of class is large ($n = 214$ and $C = 7$). This implies that we have large intervals of class probabilities. Results of numerical experiments for the Glass data set are shown in Table 5 and depicted in Fig. 8. It can be seen from the numerical results that the imprecise model by $s = 3$ and $s = 4$ allows us to get the largest accuracy measure. Moreover, the difference between the accuracy by $s = 4$ and $s = 0$ is the largest among the data sets considered above. The results concerning the Glass data set imply that the application of the IDM to the deep forest may be the most effective when the amount of training data is very small.

Now we compare the best results for data sets from Table 6 obtained by means of the ImprDF (A_{ImprDF}) with the best results which have been found in the literature (A_{best}), in particular, in Adnan and Islam (2016), Costa, Farias, Bedregal, Santiago, and Canuto (2018), Katuwal, Suganthan, and Zhang (2018), Sagir and Sathasivam (2017), Wang et al. (2016), Zhou and Feng (2017), and with gcForest without the Multi-Grained Scanning part (A_{gcF}). The corresponding numerical results are shown in Table 6. It can be seen from Table 6 that the proposed ImprDF is comparable with the best known results. At the same time, the ImprDF outperforms

Table 6
Comparison of the ImprDF with the best known results and with gcForest.

Data set	A_{best}	A_{gcF}	A_{ImprDF}
MM	84.12	83.55	84.14
HS	78.43	77.90	78.42
Seeds	92.90	92.85	92.94
Ion	92.50	92.42	92.51
Ecoli	91.20	91.09	91.18
Yeast	63.45	63.45	63.26
Park	91.63	91.62	91.62
Glass	91.44	85.03	91.42
ILPD	74.85	74.85	74.86
Car	92.18	91.67	91.98
Wave	89.39	89.32	89.32
Soyb	89.54	84.80	87.24
WCR	75.80	75.80	75.80
Diab	72.30	72.25	72.26
Mice	96.80	95.72	96.78
TAE	55.60	55.60	56.10

gcForest for many data sets which are characterized by small numbers of training examples (MM, HS, Seeds, Glass, Soyb, TAE) or by a large number of classes (Ecoli,Yeast, Mice).

It is also interesting to consider the ImprDF by different numbers of training examples. In order to carry out this study, we take a reduced Wave data set with randomly chosen examples from the original Wave data set. This data set is used for numerical experiments because it has a lot of examples such that a part of the examples can be removed. Fig. 9 illustrates how the accuracy measures of the ImprDF and gcForest obtained for the Wave data set by $s = 3$ (two solid lines) and $s = 5$ (one dashed line) depend on the number of training examples n. Curves with circular and triangular markers correspond to the ImprDF and gcForest, respectively. It can be seen from Fig. 9 that the ImprDF outperforms gcForest

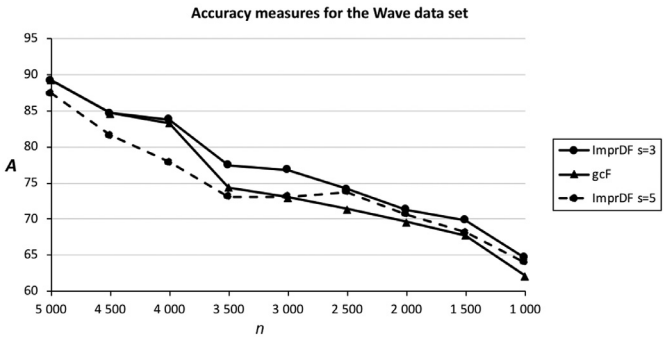

Fig. 9. Accuracy measures for the Wave data set as functions of the number of training examples n by the IDM parameters $s = 3$ and $s = 5$.

by $n \leq 4000$ when $s = 3$. The same numerical results are shown by $s = 5$. In this case, the ImprDF outperforms gcForest by $n \leq 2500$. The above results imply that the hyperparameter s of the IDM should be increased by decreasing the number of training examples.

Optimal values of the hyperparameter s of the IDM can be determined only by means of the cross-validation procedure. However, some rules can be elicited from the given numerical experiments and from the meaning of the hyperparameter. If the dataset is small, then values of s should be taken as large as possible. It is clearly seen from results obtained for the Glass dataset with $n = 214$ (Table 4 and Fig. 8) that the optimal values of s are large, for example, $s = 3$, $s = 4$. Moreover, it increases with reducing the number of trees. This observation comes to an agreement with the meaning of s when large values of s produce more cautious inferences. If the dataset is large, then values of s should be taken as small as possible, for example, $s = 0$ or $s = 1$. One can see from results obtained for the Wave dataset with $n = 5000$ (Table 4 and Fig. 7) that an optimal value of s is even 0 for some cases because we do not need to produce cautious inferences. Generally, it is difficult to predict an optimal value of s because it depends on many factors and parameters, including, a number of trees in every RF, a number of training examples in a dataset, a portion of examples for training every tree, a depth of trees, etc.

Summarizing the results of various numerical experiments, we can write the following conclusions:

1. The proposed ImprDF provides outperforming results when the amount of training data is small or the number of classes is large.
2. There is an optimal value of the hyperparameter s of the IDM, which provides the largest value of the accuracy measure. Unfortunately, this optimal value cannot be determined a priori. It depends on a certain dataset and other parameters, for example, the number of trees in every RF, the number of levels in the deep forest, etc.
3. The large number of trees may worsen the results when a dataset analyzed is rather small because trees repeat each other in this case.
4. The ImprDF and gcForest provide similar results when they are trained on large datasets.

9. Conclusion

An imprecise deep forest classifier has been presented in the paper. The main idea underlying this model is to take into account a small number of stored votes corresponding to leaf nodes. It has been shown that this can be done by using an imprecise model, in particular, the IDM. It follows from the numerical experiments that incorporating the imprecision into the deep forest may lead to outperforming results. This implies that we can positively answer the question whether the imprecision can improve a classifier or not.

The following limitations of the proposed ImprDF have to be mentioned. The deep forest is not a computationally simple algorithm itself. Therefore, the use of sets of the class probability distributions requires to generate a large number of additional feature vectors at every level of the deep forest cascade in order to adequately represent these imprecise sets of distributions and leads to additional computational time which has to be taken into account. In order to efficiently generate the additional feature vectors, we have to apply specific imprecise statistical inference models which produce convex sets of probability distributions with vertices known in the explicit form. This peculiarity restricts the use of arbitrary imprecise models for constructing the classifier. According to the algorithm implementing the proposed classifier, the additional feature vectors are generated with identical original feature vectors. If the original feature vectors have a large dimensionality, then the impact of different augmented features may be reduced. One of the ways to cope with this difficulty is to increase the number of RFs at every level of the forest cascade. However, this way again leads to the additional computational time. The use of imprecise statistical models for taking into account the available imprecision of the class probability distributions leads to additional tuning parameters, for example, the hyperparameter s in the IDM. This peculiarity significantly complicates the proposed classifier.

It should be noted that only a single imprecise model has been applied to the deep forest, namely, the IDM[1]. However, there are several interesting imprecise models which may lead to better results, for example, the pari-mutuel model (Walley, 1991), the constant odds-ratio model (Walley, 1991), Kolmogorov–Smirnov bounds (Johnson & Leone, 1964; Wasserman, 2006), the Nonparametric Predictive Inference (NPI) model (Coolen, 2011). The use of these models can be viewed as a direction for further research.

In addition to the research direction considered above, we also indicate the following important future research directions. First, the ImprDF considered in the paper solves the classification problem. However, the ideas underlying the classifier can be extended on the machine learning models solving regression problems (regression RFs), survival analysis problems (random survival forests (Ishwaran, Kogalur, Blackstone, & Lauer, 2008)), if these models are included into the deep forest as elements. Second, we propagate the imprecise probabilities through levels of the deep forest cascade, but it is interesting to study a case when the "worst" probability distribution is selected from a set of distributions in order to implement a robust strategy. Third, it has been mentioned that the number of generated feature vectors at every level may be very large, and this peculiarity of the ImprDF significantly complicates the classification model. One of the ideas to reduce the set of feature vectors is to apply a very interesting approach called a confidence screening mechanism (Pang, Ting, Zhao, & Zhou, 2018), which significantly reduces the training and testing times of forests at each level. According to the improvement, training instances with high confidence (the maximum value of the estimated class vector) directly pass to the final stage rather than passing through all the levels. On the one hand, this approach may significantly reduce the computational complexity of the ImprDF. On the other hand, it is interesting to modify the confidence screening mechanism itself in order to deal with sets of the class probability distributions. Finally, in order to improve the RFs and to solve various machine learning problems different from the classification one, weights of trees can be used. The weights are regarded as train-

[1] It is supposed that the imprecise ε-contaminated model provides the same sets of class probability distributions.

ing parameters computed in accordance with some loss function which depends on a problem solved. Examples of the RF models using weights of trees can be found in papers (Utkin et al., 2019; Utkin & Ryabinin, 2018). It is interesting to improve the ImprDF by using the similar approach.

By carrying out the numerical experiments, we have not studied large-scale data sets, like the well-known MNIST (Le Cun, Bottou, Bengio, & Haffner, 1998), CIFAR-10 (Krizhevsky & Hinton, 2009), etc. The main reason is that we aimed to show that incorporating the imprecision may improve a classifier in principle. We have shown that by many numerical examples. The cascade architecture of the deep forest added by the stacking algorithm has helped to get outperforming results because the imprecision was used as new features indicating our belief to the obtained class probability distributions.

Declaration of Competing Interest

The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Credit authorship contribution statement

Lev V. Utkin: Conceptualization, Methodology, Software, Validation, Formal analysis, Data curation, Writing - original draft, Writing - review & editing.

Acknowledgement

This work is supported by the Russian Science Foundation under grant 18-11-00078.

The author would like to thank Frank Coolen and Thomas Augustin for the idea to consider the imprecise class probabilities in deep forests instead of precise ones during a seminar at the Department of Statistics of LMU Munich. The author would also like to express his appreciation to the anonymous referees whose valuable comments have improved the paper.

References

Abellan, J. (2013). Ensembles of decision trees based on imprecise probabilities and uncertainty measures. *Information Fusion, 4*, 423–430.
Abellan, J., Mantas, C., & Castellano, J. (2017). A random forest approach using imprecise probabilities. *Knowledge-Based Systems, 134*, 72–84.
Abellan, J., Mantas, C., Castellano, J. G., & Moral-Garcia, S. (2018). Increasing diversity in random forest learning algorithm via imprecise probabilities. *Expert Systems With Applications, 97*, 228–243.
Abellan, J., & Moral, S. (2003). Building classification trees using the building classification trees using the total uncertainty criterion. *International Journal of Intelligent Systems, 18*(12), 1215–1225.
Adnan, M., & Islam, M. (2016). Forest CERN: A new decision forest building technique. In *Pacific-Asia conference on knowledge discovery and data mining* (pp. 304–315). Springer: Cham.
Berger, J. (1985). *Statistical decision theory and Bayesian analysis*. New York: Springer-Verlag.
Bernard, J. M. (2005). An introduction to the imprecise Dirichlet model for multinomial data. *International Journal of Approximate Reasoning, 39*(2–5), 123–150.
Bonissone, P., Cadenas, J., Garrido, M., & Diaz-Valladares, R. (2010). A fuzzy random forest. *International Journal of Approximate Reasoning, 51*, 729–747.
Breiman, L. (1996). Bagging predictors. *Machine Learning, 24*(2), 123–140.
Breiman, L. (2001). Random forests. *Machine learning, 45*(1), 5–32.
de Campos, L., Huete, J., & Moral, S. (1994). Probability intervals: A tool for uncertain reasoning. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 2*, 167–196.
Coolen, F. P. A. (2011). Nonparametric predictive inference. In *International encyclopedia of statistical science* (pp. 968–970). Berlin, Heidelberg: Springer.
Costa, V., Farias, A., Bedregal, B., Santiago, R., & Canuto, A. (2018). Data from combining multiple algorithms in classifier ensembles using generalized mixture functions, arXiv:1806.01540.
DeGroot, M. (1970). *Optimal statistical decisions*. New York: McGraw-Hill.
Destercke, S., & Antoine, V. (2013). Combining imprecise probability masses with maximal coherent subsets: Application to ensemble classification. In *Synergies of soft computing and statistics for intelligent data analysis* (pp. 27–35). Berlin, Heidelberg: Springer.
Fawagreh, K., Gaber, M., & Elyan, E. (2014). Random forests: From early developments to recent advancements. *Systems Science & Control Engineering, 2*(1), 602–609.
Ferreira, A., & Figueiredo, M. (2012). Boosting algorithms: A review of methods, theory, and applications. In C. Zhang, & Y. Ma (Eds.), *Ensemble machine learning: Methods and applications* (pp. 35–85). New York: Springer.
Guo, Y., Liu, S., Li, Z., & Shang, X. (2017). Towards the classification of cancer subtypes by using cascade deep forest model in gene expression data. In *2017 IEEE international conference on bioinformatics and biomedicine (BIBM)* (pp. 1664–1669).
Guo, Y., Liu, S., Li, Z., & Shang, X. (2018). BCDForest: A boosting cascade deep forest model towards the classification of cancer subtypes based on gene expression data. *BMC Bioinformatics, 118*, 1–13.
Ishwaran, H., Kogalur, U. B., Blackstone, E. H., & Lauer, M. S. (2008). Random survival forests. *Annals of Applied Statistics, 2*, 841–860.
Johnson, N., & Leone, F. (1964). *Statistics and experimental design in engineering and the physical sciences*. New York: Wiley. 1
Jurek, A., Bi, Y., Wu, S., & Nugent, C. (2014). A survey of commonly used ensemble-based classification techniques. *The Knowledge Engineering Review, 29*(5), 551–581.
Katuwal, R., Suganthan, P., & Zhang, L. (2018). An ensemble of decision trees with random vector functional link networks for multi-class classification. *Applied Soft Computing, 70*, 1146–1153.
Krizhevsky, A., & Hinton, G. (2009). *Learning multiple layers of features from tiny images. Technical report*. Computer Science Department: University of Toronto.
Kuncheva, L. (2004). *Combining pattern classifiers: Methods and algorithms*. New Jersey: Wiley-Interscience.
Le Cun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*(11), 2278–2324.
Lichman, M. (2013). *UCI machine learning repository*.
Mantas, C., & Abellan, J. (2014). Analysis and extension of decision trees based on imprecise probabilities: Application on noisy data. *Expert Systems with Applications, 41*(5), 2514–2525.
Miller, K., Hettinger, C., Humpherys, J., Jarvis, T., & Kartchner, D. (2017). Forward thinking: Building deep random forests, arXiv:1705.07366.
Pang, M., Ting, K. M., Zhao, P., & Zhou, Z. H. (2018). Improving deep forest by confidence screening. In *Proceedings of the 18th IEEE international conference on data mining (ICDM'18)* (pp. 1–6). Singapore.
Polikar, R., & Ma, Y. (2012). Ensemble learning. In C. Zhang (Ed.), *Ensemble machine learning: Methods and applications* (pp. 1–34). New York: Springer.
Ren, Y., Zhang, L., & Suganthan, P. N. (2016). Ensemble classification and regression-recent developments, applications and future directions. *IEEE Computational Intelligence Magazine, 11*(1), 41–53.
Rokach, L. (2010). Ensemble-based classifiers. *Artificial Intelligence Review, 33*(1–2), 1–39.
Rubinstein, R. Y., & Kroese, D. K. (2008). *Simulation and the Monte Carlo method* (2nd ed.). New Jersey: Wiley.
Sagir, A., & Sathasivam, S. (2017). Intelligence system based classification approach for medical disease diagnosis. *AIP Conference Proceedings, 1870*. 040047–1–040047–5 AIP Publishing
Schapire, R. (1990). The strength of weak learnability. *Machine Learning, 5*(2), 197–227.
Utkin, L. (2015). The imprecise Dirichlet model as a basis for a new boosting classification algorithm. *Neurocomputing, 151*(3), 1374–1383.
Utkin, L., Kovalev, M., & Meldo, A. (2019). A deep forest classifier with weights of class probability distribution subsets. *Knowledge-Based Systems, 173*, 15–27.
Utkin, L., & Ryabinin, M. (2018). A Siamese deep forest. *Knowledge-Based Systems, 139*, 13–22.
Walley, P. (1991). *Statistical reasoning with imprecise probabilities*. London: Chapman and Hall.
Walley, P. (1996). Inferences from multinomial data: Learning about a bag of marbles. *Journal of the Royal Statistical Society, Series B, 58*, 3–57. With discussion
Wang, Y., Li, Y., Pu, W., Wen, K., Shugart, Y., Xiong, M., & Jin, L. (2016). Random bits forest: A strong classifier/regressor for big data. *Scientific Reports, 6*(30086), 1–7.
Wasserman, L. (2006). *All of nonparametric statistics*. New York: Springer.
Wen, H., Zhang, J., Lin, Q., Yang, K., Jin, T., Lv, F., Pan, X., Huang, P., & Zha, Z.-J. (2018). Multi-level deep cascade trees for conversion rate prediction, arXiv:1805.09484.
Wolpert, D. (1992). Stacked generalization. *Neural Networks, 5*(2), 241–259.
Wozniak, M., Grana, M., & Corchado, E. (2014). A survey of multiple classifier systems as hybrid systems. *Information Fusion*, 3–17.
Yang, P., Yang, E., Zhou, B., & Zomaya, A. (2010). A review of ensemble methods in bioinformatics. *Current Bioinformatics, 5*(4), 296–308.
Zhang, Y.-L., Zhou, J., Zheng, W., Feng, J., Li, L., Liu, Z., Li, M., Zhang, Z., Chen, C., Li, X., & Zhou, Z.-H. (2018). Distributed deep forest and its application to automatic detection of cash-out fraud, arXiv:1805.04234v2.
Zhou, Z. H. (2012). *Ensemble methods: Foundations and algorithms*. Boca Raton: CRC Press.
Zhou, Z.-H., & Feng, J. (2017). Deep forest: Towards an alternative to deep neural networks. arXiv:1702.08835v2.