A Deep Forest Improvement by Using Weighted Schemes

Lev Utkin, Andrei Konstantinov, Anna Meldo,
Mikhail Ryabinin, Viacheslav Chukanov
Peter the Great St.Petersburg Polytechnic University (SPbPU)
St.Petersburg, Russia
lev.utkin@gmail.com, andrue.konst@gmail.com, anna.meldo@yandex.ru
mihail-ryabinin@yandex.ru, kauter1989@gmail.com

***Abstract*—A modification of the confidence screening mechanism based on adaptive weighing of every training instance at each cascade level of the Deep Forest is proposed. The modification aims to increase the classification accuracy. It is carried out by assigning weights to training instances at each forest cascade level in accordance with their classification accuracy. Larger values of accuracy produce smaller weights. Two strategies for using the weights are considered. The first one when the weights are regarded as probabilities of choosing the corresponding instances in building decision trees. According to the second strategy, the weights are used in splitting rules. The modification increases the classification accuracy and may reduce the training time for many real datasets. Numerical experiments illustrate good performance of the proposed modification in comparison with the original Deep Forest proposed by Zhou and Feng.**

I. Introduction

One of the very popular approaches to classification is the ensemble methodology. A basic idea of the classifier ensemble learning is to construct multiple classifiers from the original data and then to aggregate their predictions when classifying unknown samples [1]. One of the interesting ensemble-based classifier based on random forests (RFs) [2] and the stacking algorithm [3] was proposed by Zhou and Feng [4] and is called the Deep Forest (DF) or gcForest. Its structure consists of layers similarly to a multi-layer neural network structure, but each layer in gcForest contains many RFs instead of neurons. gsForest can be regarded as an multi-layer ensemble of decision tree ensembles. As pointed out by Zhou and Feng [4], gcForest is much easier to train and can perfectly work when there are only small-scale training data in contrast to deep neural networks which require great effort in hyperparameter tuning and large-scale training data. It is explained by Zhou and Feng [4] that the main motivation for developing the DF is to build a deep model which requires a small amount of training data due to a small number of training parameters. A lot of numerical experiments provided by Zhou and Feng [4] illustrated that gcForest outperforms many well-known methods or comparable with existing methods.

Many modifications of the DF have been developed last time [5], [6], [7], [8], [9], [10], [11]

One of the crucial shortcomings of the DF is that it passes all training and testing instances through all levels of the cascade, leading to significant increase of time complexity. In order to overcome this difficulty, another improvement of the original DF was proposed by Pang et al. [12], which significantly reduces the training and testing times of forests at each level. According to the improvement, training examples with high confidence (the maximum value of the estimated class vector) directly pass to the final stage rather than passing through all the levels. Pang et al. [12] introduced a confidence screening mechanism in the general framework of the DF, which categorizes instances at every level of the cascade into two subsets: one is easy to predict; and the other is hard. The improvement opens a door for developing new models improving the DF.

Therefore, following the ideas of the DF improvement, we propose a new modification of the confidence screening mechanism based on adaptive weighing of every training instance at each cascade level depending on its mean class vector at the previous level. It is called the Adaptive Weighted Deep Forest (AWDF). Two ways are considered for applying weights. The first one is when the weighted instances are randomly chosen for training trees in accordance with their weights. This leads to reducing the set of "active" instances at every level of the forest cascade. The second way is to use weights in implementing a splitting rule for training the decision trees. The numerical experiments have shown that AWDF provides outperformed results.

We will consider the standard classification problem which can be formally written as follows. Given n training instances $S = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_n, y_n)\}$, in which $\mathbf{x}_i \in \mathbb{R}^m$ represents a feature vector involving m features and $y_i \in \{1, ..., C\}$ represents the class of the associated instances, the task of classification is to construct an accurate classifier $c : \mathbb{R}^m \to \{1, ..., C\}$ that maximizes the probability that $c(\mathbf{x}_i) = y_i$ for $i = 1, ..., n$.

The paper is organized as follows. A short introduction to the gcForest architecture proposed by Zhou and Feng [4] is given in Section 2. Section 3 provides a description of the confidence screening mechanism proposed by Pang et al. [12]. AWDF algorithm is considered in Section 4. Numerical experiments with real data illustrating cases when the proposed AWDF outperforms gcForest are given in Section 5. Concluding remarks are provided in Section 6.

II. Deep Forests: a short introduction

One of the important peculiarities of gcForest [4] is its cascade structure. Every cascade is represented as an ensemble of RFs, i.e., the DF is an ensemble of the decision tree

ISSN 2305-7254

ensembles. The cascade structure is a main part of a total gcForest structure. It implements the idea of representation learning by means of the layer-by-layer processing of raw features. Each level of cascade structure receives feature information processed by its preceding level, and outputs its processing result to the next level. The architecture of the cascade proposed by Zhou and Feng [4] is shown in Fig. 1. It can be seen from the figure that each level of the cascade consists of several (four in the picture) different 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. The gcForest representational learning ability is enhanced by applying the second part of gcForest called as the so-called multi-grained scanning. The multi-grained scanning structure uses sliding windows to scan the raw features. Its output is a set of feature vectors produced by sliding windows of multiple sizes. We mainly pay attention to the first part of gcForest because our modification relates to the cascade structure.

Given an instance, each forest produces an estimate of a class probability distribution by counting the percentage of different classes of examples at the leaf node where the concerned instance falls into, and then averaging across all trees in the same forest as it is schematically shown in Fig. 2. The class distributions form a class vector or a RF class probability distribution, which is then concatenated with the original input vector to be input to the next level of cascade. The usage of the class vector as a result of the RF classification is very similar to the idea underlying the stacking algorithm [3] which trains the first-level learners using the original training dataset. Then the stacking algorithm generates a new dataset to train 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 contrast to the standard stacking algorithm, gcForest simultaneously uses the original input vector and the class vectors (meta-learners) at the next level of cascade by means of their concatenation. Different ways can be proposed for the feature vector representation at every level of the cascade. One of the ways is to add new class vectors to the vector from the previous level. In this case, the feature vector is enlarged and enlarged after every cascade level. Another way is to concatenate only class vectors produced at a level with the original vector. In this case, the feature vector does not change after every cascade level. We will use the second scheme.

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. Zhou and Feng [4] propose to use different forests (random forests, complete-random tree forests) at every level in order to provide the diversity which is an important requirement for the RF construction.

III. The confidence screening mechanism

According to [12], the main idea underlying the confidence screening mechanism is that an instance is pushed to the next level of the cascade only if it is determined to require a higher

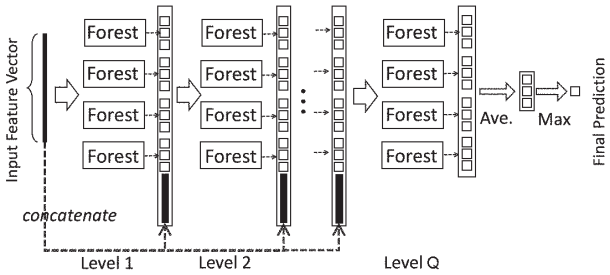

Fig. 1. The architecture of the cascade forest [4]

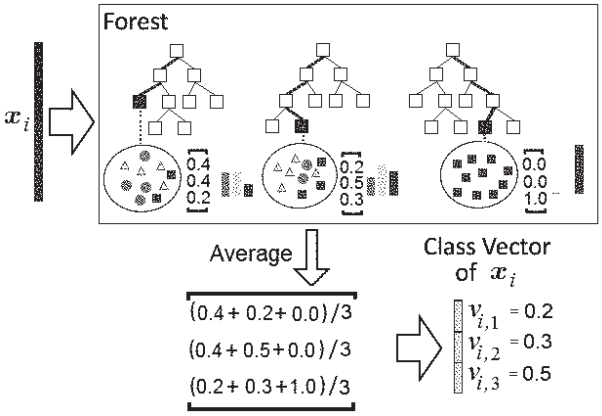

Fig. 2. The RF class probability distribution computation

level of learning; otherwise, it is predicted using the model at the current level.

A decision tree in every forest produces an estimate of the class probability distribution $\mathbf{p} = (p_1, ..., p_C)$ by counting the percentage of different classes of training instances at the leaf node where the concerned instance falls into. Then the class probabilities $\mathbf{v}_i = (v_{i,1}, ..., v_{i,C})$ of $\mathbf{x}_i$ for every RF are computed by averaging all class probability distributions p across all trees as it is shown in Fig. 2, where we partly modify a picture from [4] in order to illustrate how elements of the class vector are derived as a simple sum.

Suppose that every RFs consists of T decision trees, every cascade level contains M RFs, and the number of cascade levels is Q. Then a current level of the cascade produces M class vectors $\mathbf{v}_{i,1}, ..., \mathbf{v}_{i,M}$ which are then concatenated with the original vector $\mathbf{x}_i$ to be input to the next level of the cascade, i.e., the training set for the next level is defined as

$$S^* = \{((\mathbf{x}_i, \mathbf{v}_{i,1}, ..., \mathbf{v}_{i,M}), y_i), i = 1, ..., n\}.$$

At level q, if the prediction confidence of one instance is larger than threshold η_q, then its final prediction is produced at the current level, otherwise it needs to go through the next level (and potentially all levels in the cascade). One of the ways to define the prediction confidence of instance $((\mathbf{x}_i, \mathbf{v}_{i,1}, ..., \mathbf{v}_{i,M}), y_i)$ is to find the mean vector of class probabilities, namely,

$$\mathbf{v}_i = \frac{1}{M} \sum_{k=1}^{M} \mathbf{v}_{i,k}.$$

Let us introduce the indicator I defined as

$$I = \begin{cases} 1, & \max(v_{i,1}, ..., v_{i,C}) \geq \eta_q, \\ 0, & \text{otherwise.} \end{cases}$$

The choice of the threshold η_q is considered by Pang et al. [12] in detail, where η_q at level q is determined automatically based on the cross-validated error rate of all the training instances.

If the indicator I is 0, then the feature vector $\mathbf{x}_i$ has to go through the next level. If $I = 1$, the final prediction is produced at the current level during testing such that

$$y_i = \arg_{c=1,...,C} \max v_{i,c}.$$

During training, we do not need to go through the next level if $I = 1$ and

$$\arg_{c=1,...,C} \max v_{i,c} = y_i,$$

otherwise the instance has to go through the next level $q + 1$.

IV. The adaptive weighted deep forest classifier

In the adaptive weighted deep forest classifier, a special weighted scheme is proposed. Let us return to the definition of the class probabilities in the DF. A decision tree in every RF produces an estimate of the class probability distribution $\mathbf{p} = (p_1, ..., p_C)$ by counting the percentage of different classes of training instances at the leaf node where the concerned instance falls into. Let us consider a training instance with the feature vector $\mathbf{x}_i$ and class label y_i. Then we compute the class probabilities $\mathbf{v}_i = (v_{i,1}, ..., v_{i,C})$ of the instance $\mathbf{x}_i$ for every RF at a level of the forest cascade by averaging all class probability distributions $\mathbf{p}$ across all trees as it is shown in Fig. 2.

Let us introduce the vector $\mathbf{o}_i = (0, ..., 0, 1_{y_i}, 0, ..., 0)$, where the index of the unit element is y_i. It is obvious that the training instance $\mathbf{x}_i$ is perfectly classified by every RF if the corresponding vector of class probabilities $\mathbf{v}_i$ totally coincides with the vector $\mathbf{o}_i$. In accordance with the confidence screening mechanism, the instance $\mathbf{x}_i$ in this case should not go through the next level of the forest cascade. One of the ways to stop the instance moving through the next level is to assign to it the weight 0. Suppose now that the the training instance $\mathbf{x}_i$ is misclassified, and its vector of class probabilities $\mathbf{v}_i$ is far from the vector $\mathbf{o}_i$. Then we have to try to build the RF at the next level such that the training instance $\mathbf{x}_i$ will be not misclassified. This implies that this instance has to participate in the tree building. This can be done by assigning to it a weight closed to 1. Hence, we can conclude that the distance (or its function) between vectors $\mathbf{v}_i$ and $\mathbf{o}_i$ defines whether the instance will be used at the next level or not. Moreover, we can introduce weights which take into account the difference of the distances.

So, a weight is assigned to every instance $\mathbf{x}_i$ at a current forest cascade level in accordance with its mean class vector $\mathbf{v}_i$ at the previous level. Then the weight w_i is determined as a function f of a distance between the mean vector of class probabilities $\mathbf{v}_i$ and the vector $\mathbf{o}_i$, denoted as $d(\mathbf{v}_i, \mathbf{o}_i)$. By having the mean class vector $\mathbf{v}_i$ for instance $\mathbf{x}_i$ at the current level q, we can write the weight of instance $\mathbf{x}_i$ as the following function:

$$w_i = f(d(\mathbf{v}_i, \mathbf{o}_i)).$$

The weight w_i is used for training RFs at the next level $q+1$. It is obvious that the function f increases with $d(\mathbf{v}_i, \mathbf{o}_i)$. In particular, if instance $\mathbf{x}_i$ is correctly classified at level q such that the distance $d(\mathbf{v}_i, \mathbf{o}_i)$ is 0, then the weight w_i has to be 0. In this case, the instance is not used at the next level. In other words, due to the small weight, the instance $\mathbf{x}_i$ will have lesser chance to appear in the trees of the next level compared to other instances. If the distance is 1 ($\mathbf{x}_i$ is totally incorrectly classified), then the weight w_i has to be also 1 or to have some maximal value. Simple examples of the function f are $w_i = (d(\mathbf{v}_i, \mathbf{o}_i))^2$ or $w_i = d(\mathbf{v}_i, \mathbf{o}_i)$. Moreover, the distance can be also differently taken. One of the most popular distances is Euclidean one, i.e., $d(\mathbf{v}_i, \mathbf{o}_i) = ||\mathbf{v}_i - \mathbf{o}_i||_2$.

Another way for determining the weights is to consider the following function:

$$w_i = 1 - f(\mathbf{v}_i \cdot \mathbf{o}_i^{\mathrm{T}}).$$

In this case, we analyze only a probability of the class y_i, i.e., v_{i,y_i}. If this probability of an instance is close to 1 (correct classification), then the corresponding weight of the instance is close to 0. If the probability of the instance is close to 0 (incorrect classification), then the corresponding weight is close to 1. A simplest case is $w_i = 1 - \mathbf{v}_i \cdot \mathbf{o}_i^{\mathrm{T}}$. This definition of weights almost coincides with the rule for decision about going the instance through the next level in the confidence screening mechanism (see the previous section).

It should be noted that the normalized weights $w_1, ..., w_n$ define a probability distribution on instances in the training data, which can be used for building decision trees of a RF.

We define two strategies for using the weights. In accordance with the first strategy, we randomly draw instances from the training set with replacement in accordance with this probability distribution. If the weight of the i-th instance is very close to 0, it does not take part in building decision trees. In other words, instances with a high prediction confidence are predicted using the model at the current level.

In addition to the above weighted procedure, we can also introduce a threshold η_q to compare it with the value $1 - w_i$. If the $1 - w_i \geq \eta_q$, then the corresponding instance is predicted by using the model at current level. In sum, the number of instances for training are reduced at every level simplifying the whole training process. However, if the training set is imbalanced, then the number of randomly drawn instances of a class may be very small to be used for training.

The proposed approach is very close to the AdaBoost algorithm [13], where instances from a training set are drawn for classification from an iteratively updated sample distribution defined on elements of the training set. Every level of the DF can be viewed as an iteration in AdaBoost. The sample distribution ensures that instances misclassified by the previous classifier (at the previous iteration) are more likely to be included in the training data of the next classifier. In each iteration, the weights of all misclassified instances are increased while the weights of correctly classified examples are decreased.

According to the second strategy, we apply the procedure of direct use of the weights during learning in a splitting rule, which is implemented in many versions of the decision tree

TABLE I. A BRIEF INTRODUCTION ABOUT DATA SETS

Data set	Abbreviation	m	n	C
Adult Income	Adult	14	48842	2
Car	Car	6	1728	4
Diabetic Retinopathy	Diabet	20	1151	2
EEG Eye State	EEG	15	14980	2
Haberman's Breast Cancer Survival	Haberman	3	306	2
Ionosphere	Ion	34	351	2
Seeds	Seeds	7	210	3
Seismic Mining	Seismic	19	2584	2
Teaching Assistant Evaluation	TAE	5	151	3
Tic-Tac-Toe Endgame	TTTE	9	958	2

algorithms, for example, in C4.5 and CART. In particular, the weights and the entropy measure are combined in the splitting rule. The weights are again viewed as probabilities of instances and used in definition of the entropy measure. This strategy does not simplify the whole deep forest training process because all instances are used for training the trees. However, the problem of a lack of instances of a certain class at some level is avoided in this case.

The confidence screening mechanism can be considered as a special case of the AWDF classifier. Let us assign the weights 0 or 1 to an instance at level q if its prediction confidence is larger or smaller than threshold η_q, respectively. In this case, the instance with weight 0 does not go through the next level, and RFs of the next level are built without this instance. This implies that the confidence screening mechanism can be regarded as a special case of the proposed AWDF classifier.

V. NUMERICAL EXPERIMENTS

In order to illustrate the AWDF classifier, we investigate the model for datasets from UCI Machine Learning Repository [14]. Table I is a brief introduction about these data sets, while more detailed information can be found from, respectively, the data resources. Table I shows the number of features m for the corresponding dataset, the number of training instances n and the number of classes C.

AWDF uses a software in Python which implements gcForest and is available at http://lamda.nju.edu.cn/code_gcForest.ashx. The AWDF classifier has the same cascade structure as the standard gcForest described in [4] (two completely-random tree forests and two random forests at every level, completely-random trees are generated by randomly selecting a feature for split at each node of the tree). The forest cascade is used for numerical experiments without the Multi-Grained Scanning part of gcForest. 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 50 repetitions, where in each run, we randomly select $n_{\text{tr}} = 4n/5$ training data and $n_{\text{test}} = n/5$ testing data. We apply the second strategy to training RFs by using the weighted instances because we are interesting in increasing the accuracy of the AWDF classifier, but not training or testing time. However, we also provide results for the first strategy.

First of all, our aim is to compare AWDF with gcForest and to consider different cases of the weight function f. We denote $w_i = 1-(\mathbf{v}_i \cdot \mathbf{o}_i^{\mathrm{T}})^2$ as $1-w^2$; $w_i = 1-(\mathbf{v}_i \cdot \mathbf{o}_i^{\mathrm{T}})^{1/2}$ as $1-w^{1/2}$; $w_i = 1 - (\mathbf{v}_i \cdot \mathbf{o}_i^{\mathrm{T}})$ as $1 - w$; $w_i = ||\mathbf{v}_i - \mathbf{o}_i||_2$ (Euclidean distance) as L_2. Numerical results of comparison of gcForest (gcF) and four cases of the weight definition are shown in Table II, where the first column contains abbreviations of the tested data sets, the second column is the accuracy measure by using gcForest, other columns correspond to the accuracy measures of AWDF by different functions of weights. It should be noted that the largest values of the accuracy measures obtained by different numbers of trees are shown in Table II. It can be seen from Table II that at least one of the cases of the proposed AWDF classifier outperforms gcForest for most considered data sets. The best performance on each dataset is shown in bold.

TABLE II. ACCURACY MEASURES GCFOREST AND FOUR CASES OF THE WEIGHT FUNCTION FOR THE SECOND STRATEGY OF USING WEIGHTS

Data set	gcF	$1-w^2$	$1-w^{1/2}$	L_2	$1-w$
Adult	86.12	86.20	86.26	86.11	**86.30**
Car	98.28	98.42	**98.74**	98.45	98.51
Diabet	69.05	69.06	**69.69**	69.16	69.13
EEG	95.75	95.82	**96.16**	95.74	95.94
Haberman	74.15	73.90	**74.77**	74.09	73.65
Ion	94.32	94.32	94.37	94.26	**94.64**
Seeds	93.17	**93.62**	**93.62**	93.08	92.72
Seismic	93.50	93.63	93.53	**93.67**	93.61
TAE	52.76	53.76	55.14	55.26	**55.76**
TTTE	99.03	99.09	**99.11**	**99.11**	98.99

We can also conclude from Table II that there is no the best choice of the weight function for all datasets. Though, we can also see that the function $w_i = 1 - (\mathbf{v}_i \cdot \mathbf{o}_i^{\mathrm{T}})^{1/2}$ provides the largest number of the best results. This function makes weights to be close to 0 if an instance is correctly classified. At the same time, the "bad" instances have weights close to 1, and they introduce a large impact in computing the splitting rule. In contrast to this function, function $1-(\mathbf{v}_i \cdot \mathbf{o}_i^{\mathrm{T}})^2$ shows worse results. This is due to the fact that the resulting weights are close to 0.5. As a result, the difference between weights of correctly and incorrectly classified instances is smaller, and the separation effect is reduced.

In order to investigate how the accuracy measures depend on the number of decision trees in the RF, we depict the corresponding dependencies in Figs. 3-6 for some datasets. We compare gcForest and the AWDF classifier by four different weight functions. We again can see from 3-6 that AWDF classifier outperforms gcForest for all datasets. Table II is composed from largest values of accuracy measures given in Figs. 3-6. It is difficult to determine which type of the function f provides better results for all datasets. However, we again can see that the function $w_i = 1 - (\mathbf{v}_i \cdot \mathbf{o}_i^{\mathrm{T}})^{1/2}$ leads to the largest accuracy measures for many datasets. We also should select the function $w_i = 1 - (\mathbf{v}_i \cdot \mathbf{o}_i^{\mathrm{T}})$, which also provides outperforming results. For many dataset, the Euclidean distance, i.e., $w_i = ||\mathbf{v}_i - \mathbf{o}_i||_2$ gives the smallest accuracy measures.

Table III shows accuracy measures in accordance with the first strategy of using weights when we randomly draw instances from the training set with replacement in accordance with the probability distribution corresponding to the weights. One can see from Table III that AWDF is comparable with gcForest, but it yields the best accuracy on smaller number of datasets tested. This can be explained by the fact that we significantly reduce the training time by removing a large part of instances from the training process after the first cascade

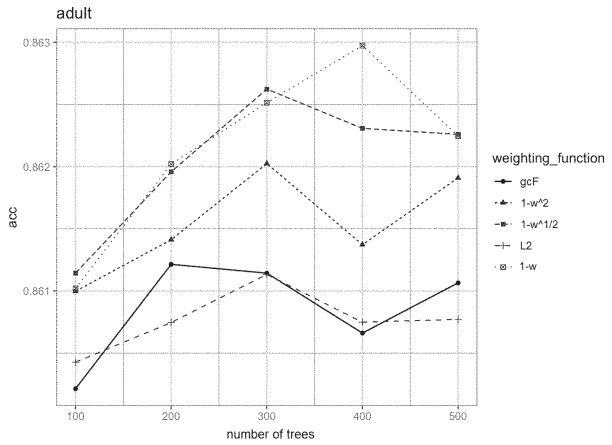

Fig. 3. Accuracy measures as a function of the number of trees for the Adult dataset

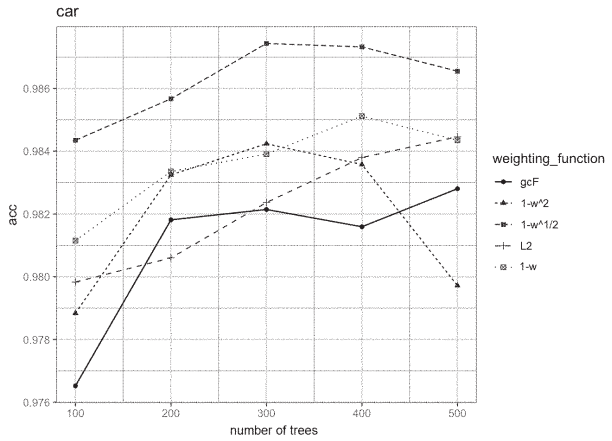

Fig. 4. Accuracy measures as a function of the number of trees for the Car dataset

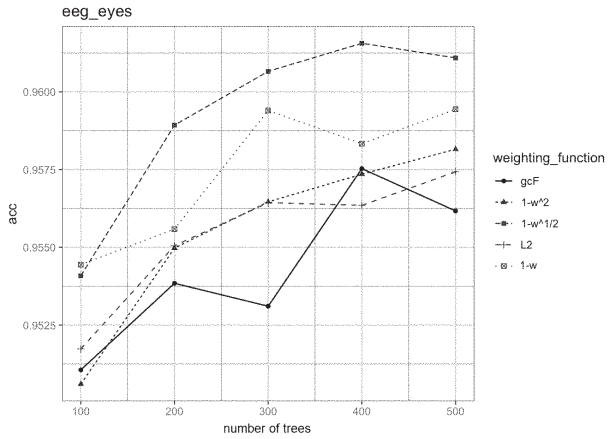

Fig. 5. Accuracy measures as a function of the number of trees for the EEG dataset

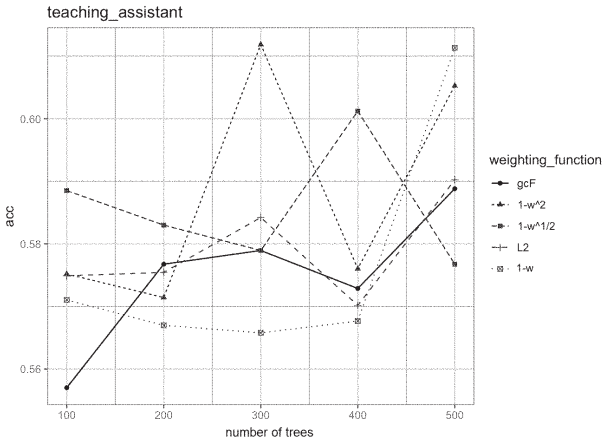

Fig. 6. Accuracy measures as a function of the number of trees for the TAE dataset

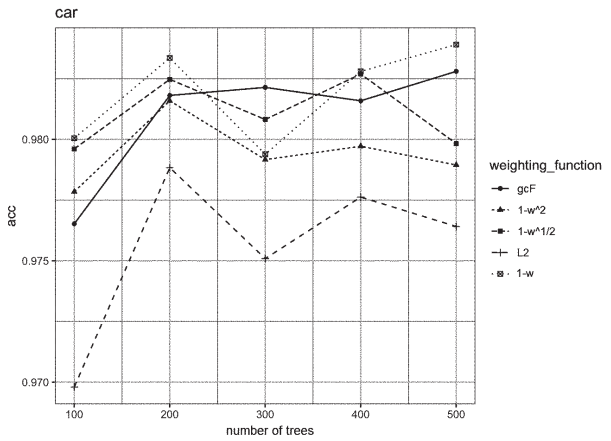

Fig. 7. Accuracy measures as a function of the number of trees for the Car dataset for the first strategy of using weights

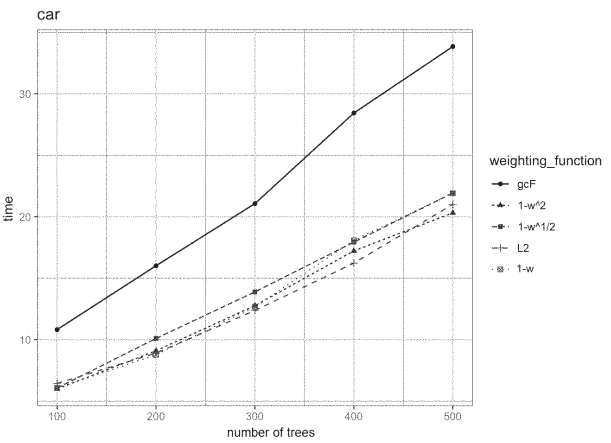

Fig. 8. The training time of classifiers as a function of the number of trees for the Car dataset for the second strategy of using weights and threshold $\eta_q = 0.95$

TABLE III. ACCURACY MEASURES GCFOREST AND FOUR CASES OF THE WEIGHT FUNCTION FOR THE FIRST STRATEGY OF USING WEIGHTS

Data set	gcF	$1-w^2$	$1-w^{1/2}$	L_2	$1-w$
Adult	**86.12**	85.98	85.72	85.95	85.84
Car	98.28	98.16	98.27	97.88	**98.39**
Diabet	69.05	68.57	**69.62**	68.25	68.38
EEG	**95.75**	94.92	95.29	94.70	95.13
Haberman	**74.15**	72.17	73.22	73.35	72.97
Ion	**94.32**	93.94	94.10	94.21	93.78
Seeds	**93.17**	92.90	92.09	92.45	91.88
Seismic	93.50	**93.79**	**93.79**	93.60	93.47
TAE	52.76	51.88	52.38	53.63	**53.65**
TTTE	**99.03**	98.33	98.81	98.41	98.63

TABLE IV. ACCURACY MEASURES GCFOREST AND FOUR CASES OF THE WEIGHT FUNCTION FOR THE SECOND STRATEGY OF USING WEIGHTS AND THRESHOLD $\eta_q = 0.95$

Data set	gcF	$1-w^2$	$1-w^{1/2}$	L_2	$1-w$
Adult	86.12	86.09	86.10	86.04	**86.13**
Car	**98.28**	96.46	96.83	96.16	96.54
Diabet	69.05	69.71	69.71	**69.92**	69.84
EEG	**95.75**	92.54	92.44	92.45	92.47
Haberman	74.15	74.52	73.47	73.10	**73.59**
Ion	94.32	**95.18**	94.26	94.91	93.51
Seeds	93.17	93.26	92.90	**93.71**	93.17
Seismic	93.50	93.37	93.68	**93.73**	93.54
TAE	52.76	53.26	**54.39**	53.88	53.38
TTTE	99.03	97.42	**97.62**	97.20	97.22

level. Perhaps, a more fine tuning of the AWDF classifier (choice of an appropriate function of weights) may improve the classification results.

Examples of the dependencies of the accuracy measures on the number of decision trees in every RF for the first strategy of using weights are shown in Fig. 7. We clearly see from Fig. 7 that AWDF does not outperform gcForest. It follows from the above results that the second strategy of using weights provides better accuracies in comparison with the first strategy.

Let us consider how the introduction of threshold η_q for AWDF impacts on the accuracy measures. Table IV shows accuracy measures in accordance with the second strategy of using weights by $\eta_q = 0.95$. One can see from Table IV that the AWDF classifier yields the best accuracy in most cases. At the same time, we have to note that the training time by using the threshold is significantly reduced. Fig. 8 shows examples of the training time as a function of the number of trees. One can see that the training time is reduced for AWDF in comparison with gcForest.

VI. CONCLUSION

The proposed modification of the confidence screening mechanism based on adaptive weighing of every training instance at each cascade level of DF has demonstrated good performance in comparison with gcForest by means of numerical experiments on several datasets. Its implementation is very simple and is similar to the well-known AdaBoost model in a sense that it updates weights of training instances at each level of the forest cascade.

The idea underlying AWDF is very simple and its implementation does not require a large additional time because weights are computed in a simple way without solving optimization problems.

Similarly to gcForest, the main advantage of the AWDF classifier is that it opens a door for developing many new adaptive weight models which could take into account different rules for updating and assigning the weights to instances at different levels of the cascade. Moreover, the weights can be assigned in accordance with the problem solved, for example, for improving the DF transfer learning algorithms, for improving the distance metric learning algorithms, etc. In other words, the weights can control the DF models. The development of the corresponding algorithms is a problem for further research.

It should be noted that the proposed approach for adaptive weighing of every training instances at each cascade level can be simply extended on cases when classifiers different from RFs are used at every level because there are weighted versions of the most classifiers. The choice of optimal structures is also a problem for further research.

ACKNOWLEDGEMENT

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

REFERENCES

[1] Z.-H. Zhou, *Ensemble Methods: Foundations and Algorithms*. Boca Raton: CRC Press, 2012.

[2] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[3] D. Wolpert, "Stacked generalization," *Neural networks*, vol. 5, no. 2, pp. 241–259, 1992.

[4] Z.-H. Zhou and J. Feng, "Deep forest: Towards an alternative to deep neural networks," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI'17)*. Melbourne, Australia: AAAI Press, 2017, pp. 3553–3559.

[5] Y. Guo, S. Liu, Z. Li, and X. Shang, "BCDForest: a boosting cascade deep forest model towards the classification of cancer subtypes based on gene expression data," *BMC Bioinformatics*, vol. 19(Suppl 5):118, pp. 1–13, 2018.

[6] K. Miller, C. Hettinger, J. Humpherys, T. Jarvis, and D. Kartchner, "Forward thinking: Building deep random forests," 20 May 2017, arXiv:1705.07366.

[7] L. Utkin and M. Ryabinin, "A deep forest for transductive transfer learning by using a consensus measure," in *Artificial Intelligence and Natural Language. AINL 2017*, ser. Communications in Computer and Information Science, A. Filchenkov, L. Pivovarova, and J. Zizka, Eds. Cham: Springer, 2018, vol. 789, pp. 194–208.

[8] L. Utkin and M. Ryabinin, "A Siamese deep forest," *Knowledge-Based Systems*, vol. 139, pp. 13–22, 2018.

[9] L. Utkin and M. Ryabinin, "Discriminative metric learning with deep forest," May 2017, arXiv:1705.09620v1.

[10] H. Wen, J. Zhang, Q. Lin, K. Yang, T. Jin, F. Lv, X. Pan, P. Huang, and Z.-J. Zha, "Multi-level deep cascade trees for conversion rate prediction," May 2018, arXiv:1805.09484.

[11] T. Wu, Y. Zhao, L. Liu, H. Li, W. Xu, and C. Chen, "A novel hierarchical regression approach for human facial age estimation based on deep forest," in *2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC)*. Zhuhai: IEEE, 2018, pp. 1–6.

[12] M. Pang, K. Ting, P. Zhao, and Z.-H. Zhou, "Improving deep forest by confidence screening," in *Proceedings of the 18th IEEE International Conference on Data Mining (ICDM'18)*, Singapore, 2018, pp. 1–6.

[13] Y. Freund and R. Schapire, "A decision theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[14] D. Dheeru and E. Karra Taniskidou, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml