ECAI 2008
M. Ghallab et al. (Eds.)
IOS Press, 2008

doi:10.3233/978-1-58603-891-5-122

MTForest: Ensemble Decision Trees based on Multi-Task Learning

Qing Wang and **Liang Zhang** and **Mingmin Chi** and **Jiankui Guo** [1]

Abstract. Many ensemble methods, such as Bagging, Boosting, Random Forest, etc, have been proposed and widely used in real world applications. Some of them are better than others on noise-free data while some of them are better than others on noisy data. But in reality, ensemble methods that can consistently gain good performance in situations with or without noise are more desirable. In this paper, we propose a new method namely MTForest, to ensemble decision tree learning algorihms by enumerating each input attribute as extra task to introduce different additional inductive bias to generate diverse yet accurate component decision tree learning algorithms in the ensemble. The experimental results show that in situations without classification noise, MTForest is comparable to Boosting and Random Forest and significantly better than Bagging, while in situations with classification noise, MTForest is significantly better than Boosting and Random Forest and is slightly better than Bagging. So MTForest is a good choice for ensemble decision tree learning algorithms in situations with or without noise. We conduct the experiments on the basis of 36 widely used UCI data sets that cover a wide range of domains and data characteristics and run all the algorithms within the Weka platform.

1 Introduction

Decision-tree is one of the most successful and widely used learning algorithms, due to its various attractive features: simplicity, comprehensibility, no parameters, and being able to handle mixed-type data. The most widely used decision tree learning algorithm is C4.5 [1] which recently had been ranked 1st in the "top10 algorithms in data mining" [16].

Ensemble methods train a collection of learners and then combine their predictions to make final decision. Since the generalization ability of an ensemble could be significantly better than that of a single learner, so studying the methods for constructing good ensembles has become one of the most active research areas in supervised learning [8]. And a lot of ensemble methods to improve the generalization ability of decision tree learning algorithms have been proposed and widely used in real word applications.

Typically, an ensemble is built in two steps, that is, generating multiple component learners and then combining their predictions. According to the styles of training the component learners, current ensemble learning algorithms can be roughly categorized into two classes, that is, algorithms where component learners must be trained sequentially and algorithms where component learners could be trained in parallel [9]. The representative of the first category is Boosting [4], which sequentially generates a series of component learners and iteratively increases the weights on the instances most recently be misclassified by the former component learner. The representative of the second category is Bagging [2] which independently generates many samples from the original training set via bootstrap sampling and then trains the component learners from each of these samples. Other representatives of this category include Random Forest [3], Randomized C4.5 [5], Random Subspace [6], etc.

Many ensemble methods for decision trees have been proposed and widely used in real world applications. Some of them are better than others on noise-free data while some of them are better than others on noisy data, such as Boosting and Random Forest are better than Bagging in situations without noise, while Bagging is more robust to noise and is better than Boosting and Random Forest in situations with noise [7]. But in reality, due to time and cost reason, ensemble methods that can consistently gain good performance in situations with or without noise is more desirable.

In this paper, we propose a new way to ensemble decision tree based on multi-task learning which generates diverse but accurate component decision tree learners in the ensemble through using different input attribute as extra task to introduce different inductive bias to the decision tree learning process. The resulting forest can achieve better performance on both noise-free and noisy data and have the following desirable characteristics:

1. Its accuracy is as good as Random Forest and Boosting and can achieve significantly improvement over Bagging in situations without noise.
2. Its accuracy is slightly better than Bagging and significantly better than Random Forest and Boosting in situations with an amount of noise.
3. It is simple and easy to parallelize.

The rest of this paper is organized as follows. In Section 2, we introduce the related works for ensemble decision tree learning algorithms. In Section 3, we introduce our ensemble method for decision tree learning. In Section 4, we describe the experimental setup and results in detail. Finally, we make a conclusion and outline our main directions for further research.

2 Related works

Bagging [2] is one of the older, simpler, and better known methods for creating an ensemble of classifiers which independently generates many samples from the original training set via bootstrap sampling and then trains a component learner from each of these samples. The Bagging algorithm has achieved great success in building ensembles of decision trees, neural networks and other unstable learning algorithms. Boosting [4] sequentially generate component classifiers by

[1] Department of Computer and Information Technology, Fudan University, Shanghai, China. Email:{wangqing,lzhang,mmchi,gjk}@fudan.edu.cn

iteratively increases the weights on the instances most recently be misclassified and have gain great success on both stable and unstable learning algorithms. Both Bagging and Boosting are methods that generate a diverse ensemble of classifiers by manipulating the training data.

Ho's random subspace technique [6] selects random subsets of the available features to be used in training the individual decision trees in an ensemble. Ho's approach randomly selects one half of the available features for each decision tree and creates ensembles of size 100. Ho summarized the results as follows: The subspace method is better in some cases, about the same or worse in other cases when compared to the other two forest building techniques Bagging and Boosting [6]. One other conclusion was that the subspace method is best when the data set has a large number of features and samples, and that it is not good when the data set has very few features coupled with a very small number of samples or a large number of classes [6].

Dietterich introduced an approach termed randomized C4.5 [5] to ensemble C4.5 learning algorithm. In this approach, at each node in the decision tree, the 20 best splits are determined and one of them is randomly selected for use at that node other than select the best split. For continuous attributes, it is possible that multiple tests from the same attribute will be in the top 20. Through experiments with 33 data sets from the UCI repository, it was found that randomized C4.5 can gain substantial improvement over bagging but is not comparable to Boosting on noise-free data, while randomized C4.5 is more robust than Boosting on noisy data.

Breiman's Random Forest [3] technique incorporates elements of random subspaces and bagging and is specific to using decision trees as the base classifier. At each node in the tree, a subset of the available features is randomly selected and the best split available within this subset is selected for split. Also, bagging is used to create the training set of data items for each individual tree. The number of features randomly chosen (from n total) at each node is a parameter of this approach. Through experiments with 16 data sets from the UCI repository and 4 synthetic data sets, it was found that Random Forest is comparable to Boosting and sometimes better on noise-free data and is more robust than Boosting on noisy data.

Empirical study [5, 7] on these ensemble methods for decision tree learning have shown that Boosting and Random Forest are the best ensemble methods for decision tree in situation without noise, while Bagging is best ensemble methods in situations with noise. So in this paper, we use Bagging, Boosting and Random Forest as benchmark ensemble methods to compare with our method.

3 Ensemble decision trees based on multi-task learning

Multi-Task Learning (MTL) [11] trains multiple tasks simultaneously while using a shared representation and has been the focus of much interest in the machine learning community over the last decade. It has been empirically [11, 15] as well as theoretically [13, 15] shown to often significantly improve performance relative to learning each task independently. When the training signals are for tasks other than the main task, from the point of view of the main task, the other tasks are serving as a bias [11]. This multi-task bias causes the learner to prefer hypotheses that explain more than one task, i.e. it must be biased to prefer hypotheses that have utility across multiple tasks.

Because in multi-task learning extra task is serve as additional inductive bias, we can use different extra task to bias each component learner in the ensemble to generate different component learners [11, 14]. The multi-task learning theory guarantees that the component learner will be with high accuracy if the extra task is related to the main task and the component learner will be with high diversity if the each extra task represents different bias. But in most learning environments, we are only given the training data which is composed of a vector of input attributes $\{A_1, A_2, ..., A_n\}$ and the class variable *C* and we do not have any other extra related tasks information. In [12], it has shown that some of the attributes that attribute selection process discards can beneficially be used as extra outputs for inductive bias transfer. So in our method, we treat each input attribute as extra task to bias each component decision tree in the ensemble. It obvious that we can generate a good ensemble in which component learners could be with high accuracy as well as high diversity given that each attribute (task) highly correlated with the class attribute and not highly correlated with each other. So it does better if we using a feature selection step to choose these attributes subset as extra tasks, but to make the algorithm simple and easy to implement, in this paper, we simply use each attribute in the input as an extra task to bias each component decision tree learning algorithm in the ensemble.

3.1 The MTForest algorithm

Our ensemble method is described in Algorithm 1. In MTForest, we generate different component decision trees by use different input attribute as extra task together with the main classification task. We call this two-task decision tree algorithm below. The two-task decision tree learning process is similar to standard C4.5 decision tree learning algorithm except that the Information Gain and Gain Ratio criteria of each split S_i is calculated by combine the main classification task and the extra task, showing below.

$$\text{MTIG}(S_i) = \text{MainTaskIG}(S_i) + weight * \text{ExtraTaskIG}(S_i) \quad (1)$$

$$\text{MTGR}(S_i) = \text{MainTaskGR}(S_i) + weight * \text{ExtraTaskGR}(S_i) \quad (2)$$

The parameter *weight* is served as an trade-off parameter between the classification accuracy and the diversity of each component two-task decision trees. In our experiments, we set the value of *weight* as 2. To further enhance the diversity among the component decision trees in the ensemble, we grow each two-task decision tree to maximum size and do not prune and incorporate our algorithm with Bagging, i.e. constructing each two-task decision tree on a new training set using bootstrap sampling from original training set.

Our ensemble method enumerates each attribute in the input as an extra task to bias each component decision tree learning algorithm in the ensemble, so its ensemble size is equal to the number of attribute in the input which is different from most of other ensemble methods such as Bagging, Boosting, Random Forest that need to specify the ensemble size. Also the building process of each two-task decision tree learning algorithm do not depend on each other, so MTForest can easily be parallelized.

For numeric attributes, in our implementation, we process in the following way. When a numeric attribute be choose as extra task, we first discretize this attribute by *k*-bin discretization where *k* =10, then in selecting the splitting attribute, this numeric attributes (task) are treated the same as non-numeric class attributes.

4 Experimental methodology and results

In this section, we describe the experimental methodology, the data sets, and the obtained results.

Algorithm 1 The MTForest Algorithm

Input: Training instances set D, k, $weight$
Output: A collection of two-task decision tree classifiers S
S={};
for each attribute A_i in the input
If A_i is numeric
discretize attribute A_i by k-bin discretization;
D_i = bootstrap sampling from D with the same size;
Using A_i as extra task together with the main classification task C to create an **unpruned** two-task decision tree $T_i$using Eq.1 and Eq.2 on the training instances set D_i;
S=$S \bigcup T_i$;
return S

4.1 Methodology

We conduct experiments under the framework of *Weka*[18]. For the purpose of our study, we use the 36 well-recognized data sets from the UCI repositories [17] which represent a wide range of domains and data characteristics. There is a brief description of these data sets in Table 1. We adopted the following three steps to preprocess data sets.

1. First, missing values in each data set are filled in using the unsupervised filter *ReplaceMissingValues* in *Weka*;
2. Second, numeric attributes are discretized using the unsupervised filter *Discretize* in *Weka*;
3. It is well known that, if the number of values of an attribute is almost equal to the number of instances in a data set, this attribute does not provide any information to the class. So,we use the unsupervised filter *Remove* in *Weka* to delete attribute does not provide any information to the class. Two occurred within the 36 data sets, namely *Hospital Number* in data set *colic.ORIG* and *Animal* in data set *zoo*.

Table 1. Description of the data sets used in the experiments.

Datasets	Size	Attribute	Classes	Missing	Numeric
anneal	898	39	6	Y	Y
anneal.ORIG	898	39	6	Y	Y
audiology	226	70	24	Y	N
autos	205	26	7	Y	Y
balance-scale	625	5	3	N	Y
breast-cancer	286	10	2	Y	N
breast-w	699	10	2	Y	N
car	1728	7	4	N	N
colic	368	23	2	Y	Y
colic.ORIG	368	28	2	Y	Y
credit-a	690	16	2	Y	Y
credit-g	1000	21	2	N	Y
diabetes	768	9	2	N	Y
glass	214	10	7	N	Y
heart-c	303	14	5	Y	Y
heart-h	294	14	5	Y	Y
heart-statlog	270	14	2	N	Y
hepatitis	155	20	2	Y	Y
hypothyroid	3772	30	4	Y	Y
ionosphere	351	35	2	N	Y
iris	150	5	3	N	Y
kr-vs-kp	3196	37	2	N	N
labor	57	17	2	Y	Y
letter	20000	17	26	N	Y
lymph	148	19	4	N	Y
mushroom	8124	23	2	Y	N
primary-tumor	339	18	21	Y	N
segment	2310	20	7	N	Y
sick	3772	30	2	Y	Y
sonar	208	61	2	N	Y
soybean	683	36	19	Y	N
tic-tac-toe	958	10	2	N	N
vehicle	846	19	4	N	Y
vote	435	17	2	Y	N
yeast	1484	10	10	N	Y
zoo	101	18	7	N	Y

In our experiments, we compare MTForest to Bagging C4.5, Boosting C4.5 and Random Forest in terms of classification accuracy in both noise-free and noisy situations. We use the implementation of C4.5 (*weka.classifiers.trees.J48*), Random Forest *(weka.classifiers.trees.RandomForest)*, Bagging *(weka.classifiers.meta.Bagging)* and Boosting *(weka.classifiers.meta.AdaBoostM1)* in Weka, and implement our algorithms under the framework of *Weka*. For Bagging and Boosting, we set the ensemble size as 50; while for Random Forest we set the ensemble size as 100. We done this for two reasons, first because it is large enough to ensure convergence of the ensemble effect with most of our data sets, second it is the same ensemble sizes used in [3]. To Random Forest, an important parameter is the number of features randomly selected at each node; in our experiments we use the default value because it can achieve best results in most case [7]. For our methods, the ensemble size is the number of input attributes which is often far smaller than 50 except on two data set(*audiology, sonar*) and we set the value of parameter *weight* as 2.

In all experiments, the classification accuracy of each algorithm on a data set was obtained via 10 runs of ten-fold cross validation. Runs with the various algorithms were carried out on the same training sets and evaluated on the same test sets. To compare two ensemble algorithms across all domains, we employ the statistics used in [5], namely the win/draw/loss record. The win/draw/loss record presents three values, the number of data sets for which algorithm *A* obtained better, equal, or worse performance than algorithm *B* with respect to classification accuracy. We report the statistically significant win/draw/loss record; where a win or loss is only counted if the difference in values is determined to be significant at the 95% level by a paired *t*-test.

4.2 Results

Table 2 shows the comparison results of two-tailed *t*-test with a 95% confidence level between each pair of algorithms, in which each entry *w/t/l* means that the algorithm at the corresponding row wins in *w* data sets, ties in *t* data sets, and loses in *l* data sets, compared to the algorithm at the corresponding column. Table 4 shows the detailed experimental results of the mean classification accuracy and standard deviation of each algorithm on each data set, and the average values are summarized at the bottom of the table. From Table 2 and 4 we can see that MTForest can achieve substantial improvement over C4.5 on most data set (13 wins and 2 losses) which suggest that MTForest is potentially a good ensemble technique for decision tree. MTForest can also gain significantly improvement over Bagging (7 wins and 2 losses) and is comparable to two state-of-the-art ensemble technique for decision trees, Boosting (8 wins and 8 losses) and RandomForest (3 wins and 4 losses). An interesting phenomenon on *iris* data set is that, MTForest is the only ensemble method which can gain improvement over the C4.5 while other three ensemble methods used to compare all decrease the accuracy over C4.5.

An important issue of an ensemble method is the question of how well it performs in situations when there is a large amount of classification noise, i.e., training examples with incorrect class labels. Since

Table 2. Summary of experimental results on noise-free data with two-tailed t-test with 95% confidence level. Each cell contains the number of wins, ties and losses between the algorithm in that row and the algorithm in that column.

w/t/l	C4.5	Bagging	Boosting	Random Forest
Bagging	11/24/1			
Boosting	16/17/3	12/17/7		
Random Forest	15/19/2	9/22/5	6/25/5	
MTForest	13/21/2	7/27/2	8/20/8	3/29/4

Table 3. Summary of experimental results on noisy data with two-tailed t-test with 95% confidence level. Each cell contains the number of wins, ties and losses between the algorithm in that row and the algorithm in that column.

w/t/l	C4.5	Bagging	Boosting	Random Forest
Bagging	15/21/0			
Boosting	10/14/12	3/13/20		
Random Forest	13/17/6	6/19/11	15/21/0	
MTForest	18/14/4	5/27/4	20/15/1	9/23/4

some noise in the outputs is often present, robustness with respect to noise is a desirable property. Following Breiman [3], the following experiment was done which changed about one in twenty class labels (i.e., injecting 5% noise). For each data set in the experiment, we randomly split off 10% of the data set as a test set, and runs are made on the remaining training set. The noisy version of the training set is gotten by changing, at random, 5% of the class labels into an alternate class label chosen uniformly from the other labels. We repeat this process 100 times to compute the classification accuracy of each algorithm in this noisy situation.

Table 3 shows the comparison results of two-tailed *t*-test with a 95% confidence level between each pair of algorithms in this noisy situations, in which each entry *w/t/l* has the same meanings as in Table 2. Table 5 shows the detailed experimental results of the mean classification accuracy and standard deviation of each algorithm on each data set in this noisy situation, and the average values are summarized at the bottom of the table. From Table 3 and 5 we can see that MTForest can achieve substantial improvement over C4.5 on most data set (18 wins and 4 losses) which suggest that MTForest is potentially a good ensemble technique for decision tree in this noisy situations. And MTForest can significantly outperform Boosting (20 wins and 1 losses) and Random Forest (9 wins and 1 losses) and is slightly better than Bagging in this noisy situation (5 wins and 4 losses). From Table 5, we can also see that MTForest has the best average value (83.30) over all the data sets used.

5 Conclusion and Future Work

We address the problem of ensemble decision trees learning algorithms that can consistently gain good performance in situations with or without noise. Previous study has shown that Bagging can always improve the classification performance of decision tree learning algorithms on both noise-free and noisy data, but its performance on noise-free data is not comparable to Boosting and Random Forest. In this paper, we propose a new ensemble method for decision tree learning algorithms by enumerating each input attribute as extra task together with the main classification task to generate different component decision trees in the ensemble. The experimental results show that the performance of our algorithm can comparable to Boosting and Random Forest on noise-free data and as good as Bagging on noisy data.

Dietterich [8] indicated that roughly there are four ensemble schemes, that is, perturbing the training set, perturbing the input attributes, perturbing the output representation, and injecting randomness to the learning algorithm. The success of MTForest suggests that we can also inject different additional inductive bias to the learning algorithm to create an ensemble of classifiers.

It will be interesting to explore whether or not we can using selective ensemble technique [10] to select a subset of the two-task decision trees created to improve the performance; exploiting task relatedness to assign different weight to each component decision tree classifiers in the ensemble; extending multi-task ensemble technique to ensemble stable classifier (such as Naive Bayes, KNN) where bagging can not work well. These have been left to be investigated in the future.

ACKNOWLEDGEMENTS

This research is partially supported by the National Key Basic Research Program (973) of China under grant No.2005CB321905. We thank the anonymous reviewers for their great helpful comments.

REFERENCES

[1] J.Quinlan, C4.5:Programs for Machine Learning, Morgan Kaufmann, (1993).
[2] L.Breiman, Bagging Predictors, *Machine Learning*, 24, pp.123-140, (1996).
[3] L. Breiman, Random Forests, *Machine Learning*, 45, pp.5-32, (2001).
[4] R.E. Schapire, A Brief Introduction to Boosting, In *Proc.16th International Joint Conference on Artificial Intelligence*, pp.1401-06, (1996).
[5] T.G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization. *Machine Learning*, 40, 139-157, (2000).
[6] T.Ho, The Random Subspace Method for Constructing Decision Forests, *IEEE Trans. Pattern Analysis and Machine Intelligence*, 20, 832-844, (1998).
[7] R.E. Banfield, L.O. Hall, K. W. Bowyer, and W.P. Kegelmeyer, A Comparison of Decision Tree Ensemble Creation Techniques. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29, 173-180, (2007).
[8] T.G. Dietterich. Ensemble learning. In: The Handbook of Brain Theory and Neural Net-works, 2nd edition, M.A. Arbib, Ed. Cambridge, MA: MIT Press, (2002).
[9] Z.H. Zhou and Y. Yu, Adapt Bagging to Nearest Neighbor Classifiers, *Journal of Computer Science and Technology*, 20, 48-54, (2005).
[10] Z.H. Zhou, J.X. Wu, W. Tang. Ensembling neural networks: Many could be better than all. *Artificial Intelligence*, 137, pp.239-263 (2002).
[11] R.Caruana,Multi-Task Learning, *Machine Learning*, 28, pp.41-75, (1997).
[12] R.Caruana, Virginia R. Benefitting from the Variables that Variable Selection Discards. *Journal of Machine Learning Research*, 3, pp.1245-64, (2003).
[13] J. Baxter, A model for inductive bias learning, *Journal of Artificial Intelligence Research*, 12, pp.149-198, (2000).
[14] Qiang Ye and P.W. Munro. Improving a Neural Network Classifier Ensemble with Multi-task Learning, *In Proc International Joint Conference on Neural Networks*, (2006).
[15] R.K. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal Machine Learning Research*, 6, pp.1817-1853, (2005)
[16] Xindong Wu, Vipin Kumar, Ross, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey Mclachlan, Angus Ng, Bing Liu, Philip Yu, Zhi-Hua Zhou, Michael Steinbach, David Hand, Dan Steinberg, Top 10 algorithms in data mining, *Knowledge and Information Systems*, 14, pp. 1-37, (2008).
[17] Blake. C., Merz. C. J. UCI repository of machine learning databases. In Department of ICS, University of California, Irvine.http://www.ics.uci.edu/ mlearn/MLRepository.html.
[18] Witten, I. H., Frank, Data Mining:Practical Machine Learning Tools and Technology with Java Implementation, Morgan Kaufmann, (2000).

Table 4. The detailed experimental results of the classification accuracy and standard deviation on data without additionally introduced noise.

Datasets	C4.5	Bagging	Boosting	Random Forest	MTForest
anneal	98.65±0.97	98.68±0.92	99.55±0.68	99.35±0.79	99.23±0.80
anneal.ORIG	90.36±2.51	91.86±2.48	92.32±2.16	91.67±2.37	92.65±2.41
audiology	77.22±7.69	80.97±7.50	84.82±7.13	79.97±6.85	82.13±6.98
autos	81.54±8.32	85.47±6.81	87.69±7.55	84.97±7.68	83.11±8.03
balance-scale	64.14±4.16	75.09±4.94	71.89±4.32	78.47±3.85	79.85±3.92
breast-cancer	75.26±5.04	73.76±5.85	66.04±8.21	70.07±7.36	69.49±6.96
breast-w	94.01±3.28	95.44±2.71	96.70±2.08	96.34±2.44	95.67±2.49
car	92.22±2.01	93.59±1.80	96.72±1.50	94.70±1.66	92.37±1.95
colic	84.31±6.02	84.83±5.81	82.56±5.85	84.37±5.47	85.65±5.36
colic.ORIG	71.76±5.63	68.08±3.53	71.67±5.45	72.58±5.00	68.09±3.43
credit-a	85.06±4.12	85.71±3.91	82.99±4.15	85.01±3.81	85.83±4.00
credit-g	72.61±3.49	74.04±4.03	73.64±3.15	75.53±3.21	74.03±2.86
diabetes	73.89±4.70	73.92±4.53	72.07±4.67	73.15±3.96	72.57±4.69
glass	58.14±8.48	59.90±9.33	56.51±9.50	60.55±8.94	61.56±8.98
heart-c	79.14±6.44	79.98±6.66	78.15±7.29	78.78±7.12	79.48±6.94
heart-h	80.10±7.11	80.97±6.92	79.34±7.14	80.10±6.03	79.24±6.90
heart-statlog	79.78±7.71	79.74±6.89	78.26±7.34	79.25±6.45	79.59±7.01
hepatitis	81.12±8.42	81.83±7.64	83.53±8.77	82.14±6.51	82.39±8.34
hypothyroid	93.24±0.44	93.26±0.44	92.26±0.94	92.58±0.78	93.14±0.65
ionosphere	87.47±5.17	89.40±4.69	92.22±4.53	90.86±4.69	91.45±4.40
iris	95.99±4.64	95.67±5.05	94.53±6.24	95.27±5.04	96.27±4.28
kr-vs-kp	99.44±0.37	99.46±0.37	99.60±0.31	99.27±0.44	99.46±0.36
labor	84.97±14.24	84.99±14.06	86.30±14.78	89.76±12.12	89.47±12.53
letter	81.31±0.78	84.24±0.83	89.89±0.78	89.78±0.68	90.81±0.61
lymph	78.21±9.74	79.70±9.61	83.26±8.85	83.04±9.49	80.31±9.60
mushroom	100±0.00	100±0.00	100±0.00	100±0.00	100±0.00
primary-tumor	41.01±6.59	45.19±6.16	42.63±6.61	41.29±6.05	43.66±6.77
segment	93.42±1.67	94.03±1.47	95.24±1.37	96.07±1.23	95.32±1.27
sick	98.16±0.68	98.25±0.68	98.15±0.73	98.22±0.65	98.28±0.74
sonar	71.09±8.40	74.08±8.95	79.21±9.02	78.58±9.06	78.59±9.14
soybean	92.63±2.72	94.05±2.61	94.04±2.61	93.80±2.72	93.86±2.82
tic-tac-toe	85.57±3.21	94.73±2.04	98.92±1.03	97.05±1.76	95.88±1.89
vehicle	70.74±3.62	72.10±3.82	72.55±4.02	72.63±3.56	73.05±3.76
vote	96.27±2.79	96.27±2.67	94.80±3.05	96.18±2.85	96.25±2.58
yeast	52.56±3.44	54.35±3.89	52.93±3.74	52.43±3.53	53.25±3.52
zoo	92.61±7.33	93.20±7.37	97.34±5.75	94.65±6.03	94.48±6.64
mean	82.06±4.77	83.52±4.64	83.84±4.76	84.12±4.45	84.07±4.54

Table 5. The detailed experimental results of the classification accuracy and standard deviation on data with 5% randomly introduced noise.

Data Set	C4.5	Bagging	Boosting	Random Forest	MTForest
anneal	98.61±0.98	98.70±0.92	95.71±2.25	98.01±1.32	98.51±1.18
anneal.ORIG	90.28±2.74	91.48±2.69	90.27±2.73	90.06±2.52	91.54±2.58
audiology	76.61±8.13	80.37±7.39	80.63±8.41	78.03±7.66	81.21±7.73
autos	76.01±10.12	83.42±8.32	81.36±8.55	82.30±8.57	80.35±8.54
balance-scale	65.95±4.85	74.35±5.50	68.83±4.87	77.16±4.31	78.92±4.16
breast-cancer	74.08±5.64	73.23±6.42	65.52±8.05	68.13±7.64	68.44±7.48
breast-w	92.86±3.49	94.79±2.94	93.81±3.09	95.72±2.53	94.99±2.68
car	91.51±1.95	92.83±1.86	93.69±1.82	94.03±1.62	92.85±2.00
colic	84.15±5.89	84.62±6.00	79.18±7.23	83.58±5.98	84.65±5.65
colic.ORIG	66.23±1.40	66.31±1.23	66.28±1.23	72.12±4.72	67.90±3.31
credit-a	84.85±4.14	85.77±4.14	79.97±4.18	84.00±4.52	84.51±4.14
credit-g	72.16±3.41	73.79±3.96	72.18±3.46	74.81±3.49	73.64±2.97
diabetes	74.01±4.87	74.37±4.66	70.62±5.13	72.64±4.77	71.74±4.31
glass	55.60±8.89	59.29±8.81	56.50±8.64	59.63±8.61	60.08±8.84
heart-c	78.18±6.65	80.07±6.28	77.20±6.97	78.78±7.06	78.10±6.65
heart-h	80.23±7.69	80.53±7.19	78.01±6.90	79.87±6.20	78.46±7.03
heart-statlog	77.59±7.13	79.22±7.00	76.67±7.23	78.37±7.00	77.70±7.50
hepatitis	81.24±9.50	81.77±8.04	82.02±8.57	82.00±7.34	82.60±8.08
hypothyroid	93.23±0.44	93.25±0.44	91.95±0.85	91.69±0.90	92.91±0.67
ionosphere	87.61±4.92	89.52±4.51	90.71±5.04	90.89±4.51	91.88±4.20
iris	95.27±4.77	95.33±5.23	91.93±6.31	93.40±6.80	94.87±5.09
kr-vs-kp	99.25±0.46	99.30±0.40	94.78±1.32	98.28±0.69	99.21±0.45
labor	83.80±14.62	85.60±13.26	89.27±12.88	88.89±12.77	89.30±12.37
letter	80.56±0.83	83.74±0.87	85.77±0.89	87.49±0.76	89.81±0.62
lymph	77.84±9.62	78.79±8.82	81.76±9.55	82.82±9.26	79.72±10.26
mushroom	99.99±0.01	99.99±0.01	96.31±0.66	99.95±0.07	99.87±0.12
primary-tumor	41.89±6.88	44.14±5.81	41.39±6.81	40.97±6.33	42.71±6.13
segment	92.92±1.71	93.90±1.56	92.65±1.80	94.36±1.65	94.83±1.46
sick	98.04±0.78	98.11±0.77	96.75±0.98	97.22±0.77	97.99±0.77
sonar	71.32±9.79	74.09±9.30	75.16±9.05	77.31±8.75	77.09±9.01
soybean	92.19±2.97	94.04±2.76	89.88±3.26	92.24±2.67	93.45±3.05
tic-tac-toe	83.81±3.55	92.79±2.47	95.53±2.10	94.80±2.23	93.20±2.57
vehicle	68.56±4.41	71.59±3.88	70.54±3.81	71.49±3.76	72.32±3.32
vote	95.60±3.10	96.11±2.81	93.18±3.80	95.15±3.18	96.15±2.86
yeast	51.72±3.45	53.39±3.85	51.87±3.78	51.18±3.42	52.44±3.60
zoo	92.39±6.71	93.30±6.97	81.85±16.47	93.16±6.69	95.17±6.03
mean	81.28±4.90	83.11±4.67	81.10±5.24	83.07±4.75	83.30±4.65