Concise Papers

NeC4.5: Neural Ensemble Based C4.5

Zhi-Hua Zhou, *Member*, *IEEE*, and Yuan Jiang

Abstract—Decision tree is with good comprehensibility while neural network ensemble is with strong generalization ability. In this paper, these merits are integrated into a novel decision tree algorithm NeC4.5. This algorithm trains a neural network ensemble at first. Then, the trained ensemble is employed to generate a new training set through replacing the desired class labels of the original training examples with those output from the trained ensemble. Some extra training examples are also generated from the trained ensemble and added to the new training set. Finally, a C4.5 decision tree is grown from the new training set. Since its learning results are decision trees, the comprehensibility of NeC4.5 is better than that of neural network ensemble. Moreover, experiments show that the generalization ability of NeC4.5 decision trees can be better than that of C4.5 decision trees.

Index Terms—Machine learning, decision tree, neural networks, ensemble learning, neural network ensemble, generalization, comprehensibility.

1 Introduction

Generalization ability is important for learning algorithms because the main purpose of learning is to accurately predict unseen data. Many kinds of algorithms with strong generalization ability have been developed, among which an impressive one is neural network ensemble [4], [10]. Through training many neural networks and then combining their predictions, neural network ensemble has behaved so remarkably well that it has become a hot topic and been successfully applied to many real domains.

Comprehensibility, i.e., the transparency of learned knowledge and the ability to give explanation for reasoning process, is also important for learning algorithms especially when they are to be used in reliable applications. Generally speaking, decision trees are with good comprehensibility because the learned knowledge is explicitly represented in trees, while neural networks are with poor comprehensibility because the learned knowledge is implicitly encoded in a lot of connections [5]. It is obvious that since a neural network ensemble comprises many neural networks, its behavior is more difficult to be understood than that of a single neural network. In other words, the comprehensibility of neural network ensemble is even worse than that of neural network.

Therefore, an interesting issue rises. That is, whether some learning algorithms that exerts both the good comprehensibility of decision tree and the strong generalization ability of neural network ensemble can be developed. In this paper, such an issue is investigated and a novel decision tree algorithm NeC4.5, i.e., Neural ensemble based C4.5, is proposed. NeC4.5 could be viewed as a variant of C4.5 decision tree [6] where a neural network ensemble is used to preprocess the training data. Since the learning results of NeC4.5 are trees instead of ensembles of neural networks, the comprehensibility of NeC4.5 is better than that of neural network ensemble. Moreover, experiments show that the generalization ability of NeC4.5 decision trees can be better than that of C4.5 decision trees.

The rest of this paper is organized as follows: Section 2 presents the NeC4.5 algorithm and explores the reason why it can work. Section 3 reports on experiments. Section 4 summarizes the main contribution of this paper and discusses several issues related to the proposed algorithm.

2 NeC4.5

The notation conventions used in this paper are summarized in Table 1.

Suppose there is a training set $S = \{(x_1, y_1), (x_2, y_2), \cdots, (x_l, y_l)\}$. A neural network ensemble can be trained from S. Here, bagging [2] is employed to train the ensemble, which utilizes bootstrap sampling [3] to generate multiple training sets from the original training set and then trains a neural network from each generated training set. Note that other kinds of ensemble learning algorithms can also be used here.

For each feature vector x_i $(i = 1, 2, \cdots, l)$, if it is fed to the trained neural network ensemble N^*, then a class label y'_i will be output from the ensemble. Through replacing y_i by y'_i, a new example (x_i, y'_i) is obtained. Such a process can be repeated so that a new training set $S' = \{(x_1, y'_1), (x_2, y'_2), \cdots, (x_l, y'_l)\}$ is generated, where all the feature vectors appear in S also appear in S'.

S' can be greatly enlarged by including extra training data generated by the neural network ensemble. This is done by randomly generating some feature vectors and then feeding them to the trained ensemble. For each randomly generated feature vector x'_j $(j = 1, 2, \cdots, m)$, if it is fed to N^*, then a class label y'_j will be output from the ensemble. By combining x'_j and y'_j, an example (x'_j, y'_j) is obtained. The amount of the extra training data can be controlled by the *extra data ratio*, which is computed through dividing the number of extra training examples by the size of the original training set, i.e., $\mu = m/l$.

Note that the scheme of using a trained system to generate examples has been employed in some information fusion paradigms [7], where the examples are generated by different sensors and then utilized to build a fuser. From the view of ensemble learning, the different sensors can be regarded as component learners while the fuser is the combiner. Therefore, in these paradigms, the examples are generated and used inside the ensemble, that is, generated by the component learners and used by the combiner. But, in NeC4.5, the examples are generated by the ensemble and used outside of the ensemble, that is, used to train a learner which is not a part of the ensemble. The pseudocode of NeC4.5 is shown in Table 2.

In order to explore the reason why NeC4.5 works, suppose the target to be learned is a function $F : X \rightarrow Y$. Note that such a function expresses a distribution in the feature space determined by X and Y. Let F_N denote the function implemented by a neural network ensemble trained on a given training set. Then, the probability for F_N to approach F is as (1):

$$\mathrm{P}_{F_N} = \mathrm{P}_{F=F_N} = 1 - \mathrm{P}_{F \neq F_N} = 1 - err_N. \tag{1}$$

Let F_T denote the function implemented by a decision tree trained on a given training set. Then, the probability for F_T to approach F is as (2):

$$\mathrm{P}_{F_T} = \mathrm{P}_{F=F_T} = 1 - \mathrm{P}_{F \neq F_T} = 1 - err_T. \tag{2}$$

Err_T can be broken into three parts. The first part is an error term caused by the limited learning ability of the decision tree. That is,

• *The authors are with the National Laboratory for Novel Software Technology, Nanjing University, Mailbox 419, Hankou Road 22, Nanjing 210093, China. E-mail: zhouzh@nju.edu.cn, jy@ai.nju.edu.cn.*

Manuscript received 12 Apr. 2003; revised 14 Aug. 2003; accepted 12 Nov. 2003.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0032-0403.

1041-4347/04/$20.00 © 2004 IEEE Published by the IEEE Computer Society

TABLE 1
Notation Conventions Used in This Paper

i, j	counter of feature vectors
X, Y	the input space and the set of labels
x, y	a feature vector and its desired label
x, y'	a feature vector and the label output from neural network ensemble
l, m	number of original training examples and extra training examples
μ	*extra data ratio*
F	a function $F : X \rightarrow Y$
err	error rate
$(\cdot)_T$	$(\cdot)$ of a decision tree
$(\cdot)_N$	$(\cdot)$ of a neural network ensemble
$(\cdot)_T^*$	$(\cdot)$ of a decision tree grown from a training set generated by neural network ensemble
$(\cdot)_T^{(c)}$	$(\cdot)$ of a decision tree grown from a training set which contains no noise and captures the whole target distribution
$(\cdot)_T^{(n)}$	$(\cdot)$ of a decision tree grown from a training set which captures the whole target distribution but contains noise
$(\cdot)_T^{(s)}$	$(\cdot)$ of a decision tree grown from a training set which contains no noise but does not capture the whole target distribution

even the function $F_T^{(c)}$ implemented by a decision tree grown from a training set which contains no noise and captures the whole target distribution may still make some error in prediction. Such kind of error is denoted by $err_T^{(c)}$. Note that $err_T^{(c)}$ may be extremely small. The probability for $F_T^{(c)}$ to approach the target F is as (3):

$$\mathrm{P}_{F_T^{(c)}} = \mathrm{P}_{F=F_T^{(c)}} = 1 - \mathrm{P}_{F\neq F_T^{(c)}} = 1 - err_T^{(c)} \tag{3}$$

The second part is an error term caused by the noise contained in the training set, which is denoted by $err_T^{(n)}$. The last part is an error term caused by the fact that a finite sample, such as a training set which does not contains all possible feature vectors, cannot fully capture the target distribution, which is denoted by $err_T^{(s)}$. Therefore, the error of a decision tree can be decomposed as (4):

$$err_T = err_T^{(c)} + err_T^{(n)} + err_T^{(s)}. \tag{4}$$

Now, suppose that the given training set has fully captured the target distribution, that is, all the possible feature vectors have appeared in the training set. In this case, $err_T^{(s)}$ is zero, and err_T is dominated by $err_T^{(n)}$. Let F_T^* denote the function implemented by a decision tree grown from the training set processed by a neural network ensemble in the way as NeC4.5 does. Note that in this case, no extra training data is generated since all possible feature vectors have already appeared in the original training set. From the view of F_N, this training set contains no noise because all the examples are generated from the same distribution. However, this distribution is not really the target distribution F, and the probability for which to approach F is P_{F_N}. Therefore, the probability for F_T^* to approach F is as (5)[1]:

$$\mathrm{P}_{F_T^*} = \mathrm{P}_{F=F_T^*} = \mathrm{P}_{F_N}\mathrm{P}_{F_T^{(c)}}. \tag{5}$$

Considering (1) and (3), (5) can be transformed to (6):

$$\mathrm{P}_{F_T^*} = (1 - err_N)\left(1 - err_T^{(c)}\right). \tag{6}$$

Comparing (6) with (2), it can be derived that $\mathrm{P}_{F_T^*}$ is greater than P_{F_T}, i.e., F_T^* approaches F better than F_T does, if (7) holds:

1. Note that the rightest term should appear as a conditional one because $F_T^{(c)}$ is computed from the output of F_N, but since the use of $F_T^{(c)}$ implicitly expresses the statistical dependence, here the term is simplified. The authors wish to thank the anonymous reviewer who indicated this issue.

TABLE 2
The NeC4.5 Algorithm

```
Input: training set S = {(x_1, y_1), ..., (x_l, y_l)}, extra data ratio μ,
    neural learner N, trials of bootstrap sampling T
Output: decision tree DT
Process:
    N* = Bagging(S, N, T)      /* train a neural network ensemble N*
                                  from S via Bagging */
    S' = ∅
    for i = 1 to l {           /* process the original training set
                                  with the trained ensemble*/
        y'_i = N*(x_i : (x_i, y_i) ∈ S)
        S' = S' ∪ {(x_i, y'_i)}
    }
    for j = 1 to μ × l {       /* generate extra training data from the
                                  trained ensemble */
        x'_j = Random( )       /* generate a random feature vector */
        y'_j = N*(x'_j)
        S' = S' ∪ {(x'_j, y'_j)}
    }
    DT = C4.5(S')              /* grow a C4.5 decision tree from the
                                  new training set */
```

$$err_N < \frac{err_T - err_T^{(c)}}{1 - err_T^{(c)}}. \tag{7}$$

Since $err_T^{(n)}$ dominates err_T and $err_T^{(c)}$ may be extremely small, it is obvious that (7) can be satisfied so far as err_N is much smaller than err_T. This indicates that using a neural network ensemble to process the original training set in the way as NeC4.5 does can benefit the construction of the decision tree, even when no extra training data is generated, given that the ensemble is significantly more accurate than the decision tree directly grown from the original training set, and the original training set contains much noise.

Then, suppose the original training set contains no noise but has not fully captured the target distribution. In this case, $err_T^{(n)}$ is zero, and err_T is dominated by $err_T^{(s)}$. For simplifying the discussion, assume the training set generated by the neural network ensemble contains all the possible feature vectors. Then, from the view of F_N, this training set captures the whole distribution. However, this distribution is not really the target distribution F, and the probability for which to approach F is P_{F_N}. Therefore, the probability for F_T^* to approach F can be expressed as (5) again. With a similar derivation, (7) is obtained. This indicates that using a neural network ensemble to generate more training examples in the way as NeC4.5 does can benefit the construction of the decision tree, given that the ensemble is significantly more accurate than the decision tree directly grown from the original training set, and the original training set has not fully captured the target distribution.

Overall, above analysis shows that utilizing a neural network ensemble in the way as NeC4.5 does can be beneficial to the construction of a decision tree. This is because the original training set may contain much noise, and may not fully capture the target distribution.

3 EXPERIMENTS

NeC4.5 and C4.5 are compared on 20 data sets from the UCI Machine Learning Repository [1]. Five runs of 10-fold cross validation are performed on each data set, and the average result is reported.

For each data set, the predictive error rates and the sizes of the trees, i.e., the number of tree nodes, are recorded. Here, the parameter μ of NeC4.5 is set to 100 and 0 percent, respectively. Note that in the latter case, no extra training data is generated by the neural network ensemble. The predictive error rates of the

TABLE 3
Comparing NeC4.5 with C4.5 (The Numbers Following "±" are the Standard Deviations)

Data set	C4.5		NeC4.5 (μ = 100%)		NeC4.5 (μ = 0%)		Neural
	Error	Size	Error	Size	Error	Size	Ensemble
australian	.155±.047	25.6± 6.0	.159±.048	31.6± 9.1	.160±.042	19.4± 9.9	.131±.038
balance	.342±.069	33.4± 2.3	.224±.044	168.0±28.7	.328±.056	29.4± 1.3	.158±.024
breast	.272±.072	10.4± 8.0	.252±.054	25.2± 8.5	.270±.045	12.8± 3.9	.233±.063
cleveland	.489±.069	47.3± 5.0	.433±.062	70.0±17.8	.416±.042	30.3± 4.5	.270±.017
credit	.138±.029	17.2± 1.8	.119±.049	17.0± 5.7	.127±.024	10.9± 4.0	.113±.032
diabetes	.249±.047	24.6± 5.4	.246±.032	64.5± 6.4	.242±.044	23.8±10.0	.214±.045
heart	.211±.091	28.3± 3.9	.194±.070	44.2± 9.8	.207±.088	22.2± 4.2	.181±.073
ionosphere	.109±.034	13.5± 1.4	.112±.045	29.3± 5.5	.096±.024	13.3± 2.3	.089±.050
iris	.080±.061	4.5± 0.7	.040±.047	16.0± 4.5	.067±.063	4.5± 0.5	.022±.031
liver	.319±.050	26.7± 6.0	.322±.056	39.8± 7.1	.295±.042	24.6± 5.0	.269±.087
page	.030±.007	41.9± 3.8	.030±.006	486.4±108.0	.030±.006	27.7± 1.9	.022±.003
sonar	.254±.102	14.1± 1.5	.244±.087	32.0± 4.1	.205±.083	15.3± 1.1	.188±.051
thyroid	.074±.062	8.0± 1.4	.060±.069	21.6± 5.0	.070±.032	7.1± 1.9	.035±.024
vehicle	.292±.033	71.2± 7.7	.264±.047	194.4±11.3	.275±.042	72.8± 7.7	.196±.022
voting	.053±.030	9.0± 4.1	.050±.037	13.0± 8.5	.053±.030	9.0± 4.1	.037±.033
waveform21	.235±.017	291.9±17.0	.209±.021	760.9±13.3	.213±.012	235.1±10.5	.170±.009
waveform40	.252±.019	313.8±13.6	.218±.025	747.1±19.2	.228±.021	240.7± 9.9	.174±.014
wine	.057±.060	6.0± 1.6	.036±.043	26.1± 2.3	.051±.055	6.0± 1.6	.023±.020
wdbc	.063±.026	11.0± 2.1	.060±.032	50.7± 8.4	.063±.032	11.6± 1.5	.038±.021
wpbc	.248±.099	14.0± 3.5	.238±.047	16.9± 9.2	.243±.051	7.4± 3.4	.232±.052

neural network ensembles employed by NeC4.5 are also recorded. Each ensemble comprises five BP networks [8] with one hidden layer containing 10 hidden units. During the training process, the generalization error of each network is estimated in each epoch on a validation set. If the error does not change in five consecutive epochs, the training of the network is terminated in order to avoid overfitting. The validation set used by a neural network is bootstrap sampled [3] from its training set. The experimental results are tabulated in Table 3.

Table 3 shows that the generalization ability of NeC4.5 with μ = 100 percent is better than that of C4.5. In detail, pairwise two-tailed t-tests indicate that there are 10 data sets (*balance*, *breast*, *cleveland*, *credit*, *heart*, *iris*, *vehicle*, *waveform21*, *waveform40*, and *wine*) where NeC4.5 with μ = 100 percent is significantly more accurate than C4.5, while there is no significant difference on the remaining 10 data sets. Table 3 also shows that the learning results of NeC4.5 with μ = 100 percent are more complex than that of C4.5 except on *credit*. However, it is evident that the comprehensibility of NeC4.5 with μ = 100 percent is far better than that of neural network ensemble, because the learned knowledge of NeC4.5 is explicitly represented in the trees while that of neural network ensemble is implicitly encoded in the connections of the networks and the voting relationship among different networks.

On the other hand, Table 3 shows that the learning results of NeC4.5 with μ = 0 percent are more simple than or at least comparable to that of C4.5. This observation reveals that the increased complexity of NeC4.5 decision trees with μ = 100 percent is owed to the use of the extra training data generated from the trained ensemble. Moreover, Table 3 shows that the generalization ability of NeC4.5 with μ = 0 percent is still better than that of C4.5. In detail, pairwise two-tailed t-tests indicate that there are seven data sets (*cleveland*, *diabetes*, *ionosphere*, *liver*, *sonar*, *waveform21*, and *waveform40*) where NeC4.5 with μ = 0 percent is significantly more accurate than C4.5, while there is no significant difference on the remaining 13 data sets. This observation supports the claim that employing a neural network ensemble to process the original training set is beneficial even when no extra training data is generated.

However, since NeC4.5 with μ = 100 percent improves the generalization ability of C4.5 on 10 data sets while NeC4.5 with μ = 0 percent improves on only seven data sets, it is evident that extra training data generated from the trained neural network ensemble is helpful for decision tree induction. Moreover, Table 3 shows that there are only three data sets (*cleveland*, *waveform21*, and *waveform40*) where both NeC4.5 with μ = 0 percent and that with μ = 100 percent improve the generalization ability. For the remaining data sets, NeC4.5 with μ = 0 percent could improve the generalization ability does not necessarily means that NeC4.5 with μ = 100 percent could do so, and vice versa. This observation implies that the improvement on the generalization ability caused by extra training data is not stable.

Then, an interesting issue arises. That is, whether some appropriate values of μ exist, which enables the generalization ability of NeC4.5 be better than that of C4.5 on any data sets. To explore this issue, further experiments are performed on data sets (*australian*, *page*, *thyroid*, *voting*, *wdbc*, and *wpbc*) where neither NeC4.5 with μ = 100 percent nor NeC4.5 with μ = 0 percent is significantly more accurate than C4.5. The results are depicted in Figs. 1 and 2. Note that for better display, the predictive error rates of NeC4.5 have been normalized according to that of C4.5. In other words, the results shown in these figures are the error ratios of NeC4.5 against C4.5.

Figs. 1 and 2 reveal that for any data set, the generalization ability of NeC4.5 could be better than that of C4.5, given that μ is set to an appropriate value. However, such a value is not a

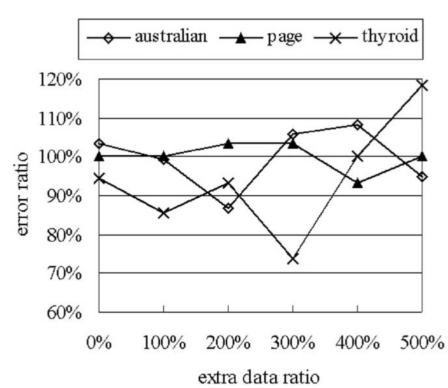

Fig. 1. NeC4.5 on *australian*, *page*, and *thyroid*.

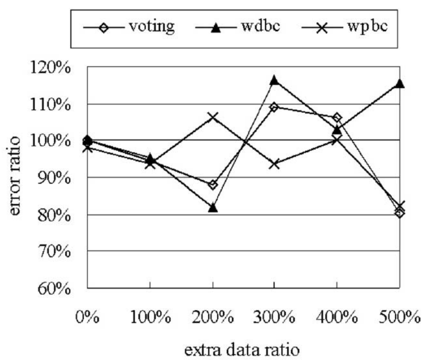

Fig. 2. NeC4.5 on *voting*, *wdbc*, and *wpbc*.

constant. The best values shown in the figures are 200 percent on *australian* and *wdbc*, 300 percent on *thyroid*, 400 percent on *page*, and 500 percent on *voting* and *wpbc*, respectively.

Table 3 has revealed that NeC4.5 with μ = 100 percent is relatively safe because it never significantly deteriorates the generalization ability of C4.5. But, when the value of μ becomes bigger, NeC4.5 becomes not so safe since there are cases where it significantly deteriorates the generalization ability of C4.5, as shown in Figs. 1 and 2. The reason might be that when too many extra training data are generated, the chances of overfitting is enlarged. However, this is only a conjecture that should be justified by rigorous theoretical analysis and far more experiments.

4 Conclusion and Discussion

In this paper, a variant of C4.5 decision tree algorithm named NeC4.5 is proposed, which utilizes neural network ensemble to preprocess the training data for decision tree induction. Such an algorithm can work well because the original training set may contain much noise and may not capture the whole target distribution. Since its learning results can be more accurate than that of C4.5 while the reasoning process remains explicitly explainable, NeC4.5 provides a good choice for tasks where both the generalization ability and the comprehensibility are concerned.

Since decision trees and neural networks are alternative paradigms for classification, much research has addressed the issue of developing hybrid learning algorithms that combine them together. A review on this topic can be found in [9]. Different from previous hybrid learning algorithms, neither does NeC4.5 use decision tree to help determine the topology of neural networks, nor does it use neural network ensemble to refine the splits or even embed as splits into the decision trees. Therefore, NeC4.5 exhibits a new way to hybrid learning.

A deficiency of NeC4.5 is that the cost of building a decision tree is burdened by the training of a neural network ensemble. Roughly speaking, the training time cost of NeC4.5 can be broken into three parts, i.e., $O = O_1 + O_2 + O_3$, where O_1 is used to train the neural network ensemble, O_2 is used to generate data from the ensemble, and O_3 is used to build the decision tree. O_3 is slightly larger than the time cost for training a decision tree from the original training set. O_2 is not neglectable but the dominating part is O_1 because training multiple neural networks requires much time costs. However, obtaining a stronger decision tree may be worthy of the extra time cost in many applications. Moreover, it is noteworthy that the training process is usually offline, while the predictive process of an NeC4.5 decision tree is almost as efficient as that of a C4.5 decision tree.

The experiments reported in this paper show that given an appropriate value of μ, the generalization ability of NeC4.5 can be better than that of C4.5. However, how to determine the appropriate value of μ remains an open problem. Moreover, designing appropriate mechanisms to control the generation of extra training data is also an important issue to be investigated in the future.

Acknowledgments

The comments and suggestions from the anonymous reviewers greatly improved this paper. This work was supported by the National Outstanding Youth Foundation of China under the Grant No. 60325207.

References

[1] C. Blake, E. Keogh, and C.J. Merz, "UCI Repository of Machine Learning Databases" http://www.ics.uci.edu/~mlearn/MLRepository.html, Dept. of Information and Computer Science, Univ. of California, Irvine, 1998.
[2] L. Breiman, "Bagging Predictors," *Machine Learning,* vol. 24, no. 2, pp. 123-140, 1996.
[3] B. Efron and R. Tibshirani, *An Introduction to the Bootstrap.* New York: Chapman & Hall, 1993.
[4] L.K. Hansen and P. Salamon, "Neural Network Ensembles," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 12, no. 10, pp. 993-1001, Oct. 1990.
[5] *Working Notes of the IJCAI-95 Workshop on Machine Learning and Comprehensibility,* Y. Kodratoff and C. Nédellec, eds., 1995.
[6] J.R. Quinlan, *C4.5: Programs for Machine Learning.* San Mateo, Calif: Morgan Kaufmann, 1993.
[7] N.S.V. Rao, "On Fusers that Perform Better than Best Sensor," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 23, no. 8, pp. 904-909, Aug. 2001.
[8] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, "Learning Internal Representations by Error Propagation," *Parallel Distributed Processing: Explorations in the Microstructure of Cognition,* vol. 1, pp. 318-362, 1986.
[9] Z.-H. Zhou and Z.-Q. Chen, "Hybrid Decision Tree," *Knowledge-Based Systems,* vol. 15, no. 8, pp. 515-528, 2002.
[10] Z.-H. Zhou, J. Wu, and W. Tang, "Ensembling Neural Networks: Many Could be Better than All," *Artificial Intelligence,* vol. 137, nos. 1-2, pp. 239-263, 2002.