TMLAI TRANSACTIONS ON MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE
SOCIETY FOR SCIENCE AND EDUCATION UNITED KINGDOM
VOLUME 2, ISSUE 1
ISSN: 2054 - 7390

Neural Networks Trained by Randomized Algorithms

Qin Qin[1], Qing-Guo Wang[2], Shuzhi Sam Ge[3] and Chao Yu[4]

Department of Electrical and Computer Engineering, National University of Singapore, Singapore, 117576.

g0800434@nus.edu.sg[1], elewqg@nus.edu.sg[2], samge@nus.edu.sg[3], yuchao@nus.edu.sg[4]

ABSTRACT

In this paper, a new model framework is proposed where a group of neural networks are trained with randomized algorithms. By incorporating randomization of random forests into a training algorithm of a neural network, the repeated running of such a revised training algorithm yields multiple independent neural networks. This group of multiple models jointly may outperform individual models. Simulation studies are conducted on various examples including practical ones such as stock markets and show that the proposed model group overall performs better than single neural network and a random forest. When there is significant noise in the data set, the performance of the former drops relatively less than the latter. In particular, the former produces much lower deviation of the performance and higher mean performance compared with the latter. Therefore, the proposed method has strong ability to classify the noisy data and perform robustly.

Keywords: Neural networks, Random forests, Classification, Complex systems, modeling.

1. INTRODUCTION

Neural network [1, 2] is well known in the area of machine learning and used in a wide range of applications for modeling complex systems. It was used as a basic tool to model the activities of the human brain, and learning mechanisms were designed to acquire knowledge. The process of learning for a neural network consists of adjusting the weights of its nodes to minimize a chosen cost functional [3, 4]. Since the nature of the neural network of biology is nonlinear, the neural network is a technique that can be used to handle nonlinear problems. In applications, the neural network is often utilized to recognize the patterns hidden in the input data and output data by exploring the complicated relationships between them. In [5], Sneak Circuit data were modeled with a neural network on back propagation algorithm with the prediction accuracy rate of up to 60%, which is high in the field of Sneak Circuit prediction. The neural network was combined with Bayesian algorithm in [6] to predict the skewed data. A

DOI: 10.14738/tmlai.21.39
Publication Date:10th February 2014
URL: http://dx.doi.org/10.14738/tmlai.21.39

novel neural network was proposed in [7] with multiple prior knowledge, where its structure consists of three layers with hybrid feed forward modes and it is trained by sequential quadratic programming with experimental results in the industrial processes. In [8], the optimal selection was made from sigmoid, radial basis and polynomial functions. An ensemble algorithm was presented in [9] for neural networks for the prediction of wind power, which shows that the prediction accuracy is enhanced compared with the one produced by single neural network. The k-mean algorithm was incorporated into neural networks to form an ensemble framework in dealing with the classification problem on digital mammograms [10], which shows improved accuracy rate compared with existing framework. The random neural network proposed by Gelenbe [11] has an interconnected network of neurons which exchanges spiking signals and involves complicated feedback loops.

A decision tree [12, 13] is a very different model from a neural network. Its model structure mimics a human decision making process, but not a human brain. Its learning process is also different and carried out in stages by separating a data set into branch-like segments with splitting rules. In a tree, the original node contains the entire data. The terminal nodes are called the leaves, and used for predictions. A random forest [14, 15] is a group of multiple decision trees. These trees in a forest are obtained independently through randomized splitting rules. Each tree grows with a random vector which is sampled independently from a distribution. The sampling for all the trees in the random forest is performed under the same distribution. When the size of the random forest increases, the generalization error of the random forest can converge to a limit, which makes the random forest predictions robust to noise. It is shown [16] that a random forest is able to prevent over-fitting. Random forests have found applications in stock markets [17], where both technical indicators and financial news are used in the model and a higher return rate is obtained than the buy-and-hold strategy. In [18], the direction of the Indian stock market index was predicted using random forest, neural network and support vector machine. Random forests also have many applications in the field of classification problem. Yi et al. [19] use the random forest to classify the stellar spectral data and the experiment results show the random forest can have better classification efficiency and produce lower root mean square errors compared with neural network. In [20], random forest is used to classify the high-dimensional data that reflects patient response to drugs, a comparison experiment is conducted, and the results show that random forest has a stronger power in classifying such high-dimension data. It is shown [21] that better accuracy rates can be obtained by random forest in locating the desired region in segmenting an image.

It is noted that when a neural network model is employed as a single model, it may lack robustness, while a random forest usually adopts the majority rule from its various decision trees, which may enhance robustness but have poor predictions from individual trees, causing performance limitations for the entire forest. It would be desirable to combine the strengths of both techniques, that is, a group of models with the individual models similar to a neural

network with good prediction expectation and its group similar to a random forest with prediction robustness, or reduction of variance of prediction errors.

In the view of the above observations, this paper aims to develop a group of neural networks by incorporating randomization during their learning process so as to outperform a neural network and a random forest. Our idea of randomization was motivated by the essence of random forests. However, when attempting to apply the idea of random forests to neural networks, one realizes that two models' structures and their learning techniques are extremely different: the former is actually locally trained in the sense that the training process works on a smaller subset of the data at each stage of learning using a subset of inputs (actually a small subset of the entire inputs), whereas the latter is always globally trained with the full data set at each stage of learning using the complete set of inputs. The extension of random forests to neural networks is not reported in the literature, to our best knowledge, and deserves a careful technical development, which is presented in this paper. It turns out that our new technique can enhance performance compared with either of random forest and neural network when the data is difficult to learn. This should be valuable for modeling of noisy data. For ease of exposition, we consider classification problems in this paper, since the regression problem can be treated similarly with no technical differences or difficulties. It should be also pointed out that the neural network group devised in this paper is different from the standard neural network ensembles in the literature, where either some averaging is used to produce an ensemble from individual neural networks or individual neural networks act on different subsets of the data. Our model works like a random forest, where trees are replaced by neural networks.

The rest of this chapter is organized as follows. The neural network and the random forest are reviewed briefly in Sections 2 and 3, respectively. The proposed method is developed in Section 4. An illustrative example, popular examples and practical examples are presented in Sections 5, 6 and 7, respectively. The paper is concluded in Section 8.

2. NEURAL NETWORKS

A simple neural network is shown in Figure 1. The neural network modeling deals with the following:

1) The function that computes the output from the input based on the weights.

2) The training process that updates the weights between the neurons of different layers.

The function $f(x)$ of a neuron network can be a combination of other functions $h_i(x)$, which can also be the combinations of others functions. The nonlinear weighted sum is a widely adopted form that is utilized in neural network, and it can be described by:

$$f(x) = Q(\sum_i w_i h(x)), \tag{1}$$

where Q is the function that is predefined, for example, the hyperbolic tangent.

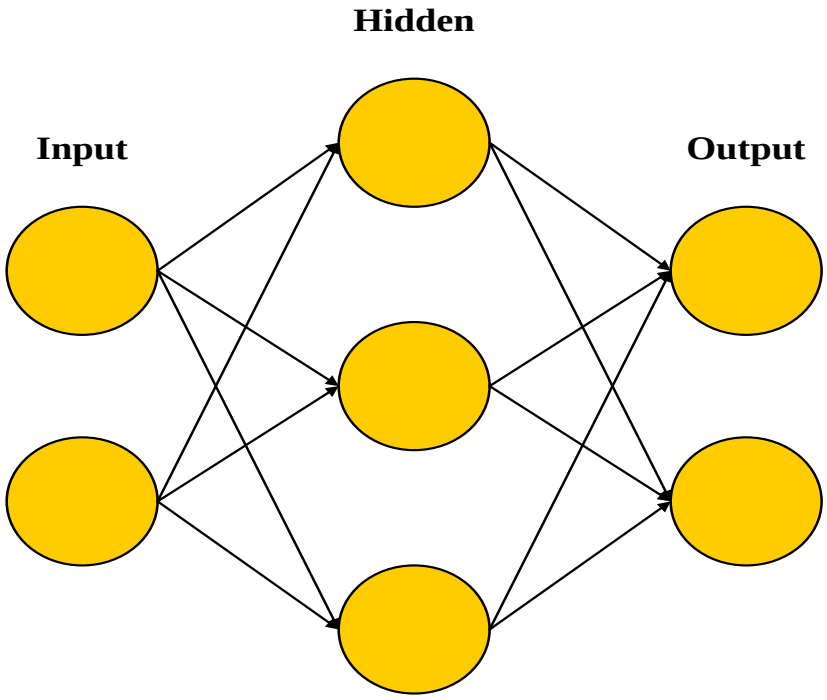

Figure 1 Structure of a simple neural network.

For training, one defines a cost function $J: F \to R$, and then find the optimal f^* such that $J(f^*) \le J(f)$, $\forall f \in F$. For example, we have the data of (x, y), where x represents the input and y the output. We define the cost function:

$$J = E[(f(x) - y)^2]. \tag{2}$$

The task is to find optimal f^* that minimizes the cost function:

$$E[(f^*(x) - y)^2] \le E[(f(x) - y)^2], \quad \forall f \in F. \tag{3}$$

3. RANDOM FORESTS

Random forest is an ensemble of many decision trees. The decision tree is a branch-like graph for the process of decision making. A simple decision tree is depicted in Figure 2. The decision tree splits the data set into subsets according to the splitting criterions. This splitting process is repeated on each subset recursively.

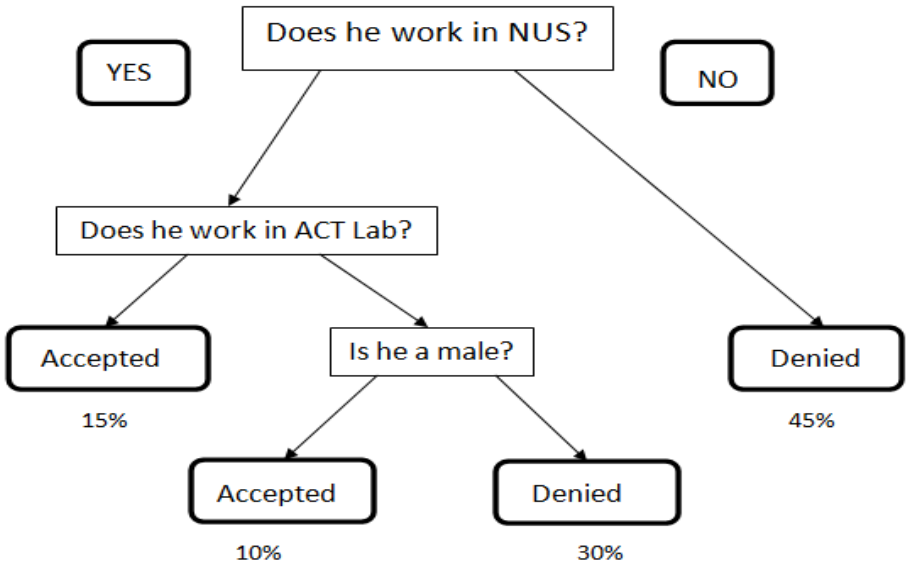

Figure 2 Structure of a simple decision tree.

For a tree, the output is obtained from the input along the way from the root node to the corresponding leaf node.

Let the number of input variables be Q. A random forest is formed by growing each decision tree as follows:

1) Randomly select q variables from Q variables and conduct the splitting at each node;

2) Let the tree grows fully with no pruning;

until the number of trees reaches the desired N. With a random forest, for a test data point, one obtains from each decision tree, its classification label which is called “vote”, and decide the final classification label with the highest votes.

4. THE PROPOSED MODEL

We view a neural network like a decision tree and try to design some randomized algorithm to find a set of neural networks independently. Note that a neural network is trained at each stage of iterative learning with the full data set (this discussion has nothing to do with cross-validation or bootstrap et al, which uses a subset of the data) based on the full set of inputs, and its structure (the layers and neurons) is fixed before and during the training. This is very different from a tree for which its branches and nodes are produced gradually during the training. Due to the above essential difference, it does not look feasible to mimic construction of a random forest by training a neural network with different combinations of inputs, and/or from different subsets of data at different stages. Further, if one chooses different structures (layers and neurons) for different neural networks, there seems no good rational to make such a choice (even for single neural network). As a result, we suppose that N neural networks to be trained have the same structure. Then, the question is how to generate multiple neural

networks independently, givens the same structure for all N models and the same set of date and inputs at each training stage? Note that when the structure is fixed, a particular neural network is determined by its weights. Different weights give different neural networks. This led us to devising the changes to the training algorithms so as to randomize certain aspects of a chosen training algorithm to find *N* neural networks independently. Then, each neural network acts like a decision tree and the resulting neural network group looks like a random forest. It is expected that a single neural network may perform better than a single decision tree. Hopefully, with the same degree of randomization and independence of the individual models as the trees, the resulting neural network group performs better than a random forest.

In the view of the above observations, we look at some training algorithm to introduce randomization. The back propagation algorithm of neural networks is widely used and thus taken for our study and illustration. The other algorithms can be chosen as well and the similar development as below can be done easily. Suppose that the training set is given as $\{(x_1, y_1), (x_2, y_2), ...(x_n, y_n)\}$, where x_i , $i = 1,2,...,n,$ are the input vectors, and y_i , $i = 1,2,...,n,$ are the corresponding outputs. Let the predicted outputs from a neural network is $\hat{y}_i$, $i = 1,2,...,n$. The backpropagation algorithm seeks to minimize the following error function:

$$E = \frac{1}{2}\sum_{i=1}^{n}\left\|\hat{y}_i - y_i\right\|^2 . \tag{4}$$

One chooses the initial weights, w_i^0 , $i = 1,2,...,m$, and update the weights as follows:

$$w_i^{k+1} = w_i^k - \alpha\frac{\partial E}{\partial w_i}, \ i = 1,2,...,m, \tag{5}$$

where α is a learning constant. This updating carries on recursively till $\nabla E = 0$ holds approximately.

Note that the above back propagation algorithm updates all the weights together at each iteration. We now introduce its randomization: randomly choose *p* weights from the total *m* weights to update with (5) while the remaining weights are kept unchanged, at each iteration, and carry on till $\nabla E = 0$ holds approximately. As a result, a neural network has been obtained. Next, choose a new set of initial weights randomly and repeat the above iterations to find the second neural network, and so on till N neural networks are all obtained.

The proposed method is summarized as follow:

1) Set up *N* identical neural networks architectures.

2) For each neural network: set the initial weights randomly; and adopt the back propagation algorithm with the following change: at each iteration, choose randomly *p* weights only to implement the updates while keeping the remaining (*m*-*p*) weights unchanged.

3) Assign the output to the label which has the most votes from *N* trained neural networks.

Breinman [15] recommends that for a random forest, the number of trees in a forest should be a large number, and the number of randomly selected features at each splitting should be equal to the root square of the number of all features. In this paper, we follow Breinman's above recommendations with some adaptation. In particular, suggest that the number of neural networks in the proposed framework be 50, and the number of randomly selected weights be the root square of the number of all weights by comparing the input number in random forest to the number of weights in our framework, that is, $N = 50$ and $p = \sqrt{m}$.

5. ILLUSTRATIVE EXAMPLE

From this section onwards, we conduct comparative simulation studies of three modeling methods: a single neural network (SNN), a random forest (RF) and the proposed group of multiple neural networks with randomized algorithms (MNN). Throughout these sections, we set $N = 50$ and $p = \sqrt{m}$. This implies that a RF consists of 50 decision trees and an MNN group consists of 50 individual neural networks. We make ten random runs for each of RF, SNN and MNN, from which, the best and worst performance as well as the average performance are evaluated on the test data, respectively, for comparison.

For a start, this section presents an illustrative example. We randomly select 4000 data points in the square formed by the four points: [-1 -1], [-1 1], [1 1] and [1 -1]. Choose this simple function:

$$y = x^2 - 0.25, \tag{6}$$

to divide the square into two areas. It happens that 2297 data points are above the function curve and we assign their labels as 1, while 1703 data points are below the function curve and are assigned with the label of 0. All the 4000 data points are used as the training data for the experiment. The test data are generated by additional randomly-drawn 400 data points in the same square. Once again, the resulting 211 data points above the function curve are labeled as 1, while others as 0. This is of course a separable case. Later we introduce noise in the data to make the data worse and worse, and thus classification more and more difficult.

Test results are presented in Table 1, where P1 means predicted 1, P0 predicted 0, T1 is true 1, and T0 true 0. To find the accuracy rate from the table, take the average case of SNN for consideration. Since the number of data points with true 1 and predicted 1 is 210 while the number of data points with true 0 and predicted 0 is 186.1, the total number of correct predictions is (210+186.1), which is divided by the total number of test points, 400, to give the classification accuracy rate of 99.03%. In the same manner, the average accuracy rates for RF and MNN are obtained as 98.83% and 98.65%, respectively. The best accuracy rates for them are 99.75%, 99.25% and 99.00%, respectively, whereas the worst accuracy rates are 98.00%,

98.75% and 98.00%, respectively. It follows that the performance of MNN is not better than that of RF and SNN when the data set is easy to classify.

Table 1 Classification for illustrative data (0% noise level).

	SNN		RF		MNN		
	P1	P0	P1	P0	P1	P0	
Average	210	1	210.1	0.9	209.3	1.7	T1
	2.9	186.1	3.8	185.2	3.7	185.3	T0
Best	211	0	211	0	210	1	T1
	1	188	3	186	3	186	T0
Worst	208	3	210	1	209	2	T1
	5	184	4	185	6	183	T0

In order to check the performance on the noisy data, we randomly select 10% of data points from in both training and testing data sets of the first group with label of 1 (above the function curve) and change their labels to 0, and also select 10% of data points in both training and testing data sets from the second group (below the function curve) with label of 0 and change their labels to 1. Then, we re-run the simulation on the changed data. The results are shown in Table 2. The average accuracy rates for SNN, RF and MNN become 99.00%, 95.86% and 98.70%, respectively. The best accuracy rates for them are 99.75%, 96.50% and 98.75%, respectively, whereas the worst accuracy rates are 98.00%, 95.25% and 98.50%, respectively. Thus, the performance of MNN has improved a lot relatively to other two methods and is better than RF now when the data set is added with 10% noise.

Table 2 Classification for illustrative data (10% noise level)

	SNN		RF		MNN		
	P1	P0	P1	P0	P1	P0	
Average	210.1	0.9	201.9	9.1	209	2	T1
	3.1	185.9	7.4	181.6	3.2	185.8	T0
Best	210	1	204	7	209	2	T1
	0	189	7	182	3	186	T0
Worst	206	5	201	10	209	2	T1
	3	186	9	180	4	185	T0

To see the performance on the data set with higher noise levels, we conduct simulation for the noise levels of 20%, 30%, 40% and 50%, respectively. The classification accuracies at all the noise levels are summarized in Table 3. It is seen from Table 3 that

when the data set has no noise or small noise (10% noise level), three methods perform similarly and all achieve the high accuracy rates.

when the data set has significant noises (20%, 30%, 40% and 50% noise levels), SNN and MNN perform much better than RF. Note that MNN produces a much stable performance (much lower performance deviation between the best and worst). In particular, the worst

performance of MNN is much better than SNN, and the average performance of MNN is better than SNN in most cases as well.

The results indicate that the proposed method has more power than SNN and RF in classifying the noisy data set.

Table 3 Classification accuracy relative to noise level (illustrative data).

Noise level	Method	Average	Best	Worst
No noise	SNN	99.03%	99.75%	98.00%.
	RF	98.83%	99.25%	98.75%
	MNN	98.65%	99.00%	98.00%
10% noise	SNN	99.00%	99.75%	98.00%
	RF	95.86%	96.50%	95.25%
	MNN	98.70%	98.75%	98.50%
20% noise	SNN	98.30%	99.50%	97.00%
	RF	89.78%	90.75%	88.50%
	MNN	98.25%	98.50%	98.00%
30% noise	SNN	96.45%	98.25%	92.50%
	RF	82.05%	84.25%	80.50%
	MNN	96.95%	97.50%	96.25%
40% noise	SNN	90.78%	93.00%	85.75%
	RF	66.48%	68.50%	63.75%
	MNN	92.73%	93.50%	92.00%
50% noise	SNN	54.30%	66.25%	47.75%
	RF	50.08%	53.00%	47.25%
	MNN	61.03%	64.75%	56.50%

6. POPULAR EXAMPLES

6.1 Fisher's Iris data

This data is included in the Matlab software, where its description is available from Matlab helps and cited as follows: "Fisher's iris data consists of measurements on the sepal length, sepal width, petal length, and petal width of 150 iris specimens. There are 50 specimens from each of three species." In this data set, three species are "setosa", "versicolor" and "virginica". Each species consist of 50 specimens. We take 120 data points (40 data points of "setosa", 40 data points of "versicolor" and 40 data points of "virginica" to maintain class ratios same as those in the raw data) as our training set and the remaining 30 data points (10 data points of "setosa", 10 data points of "versicolor" and 10 data points of "virginica") as the testing set.

The simulation results are shown in Table 4, where P2 means predicted "setosa", P1 means predicted "versicolor", P0 means predicted "virginica"; T2 means true "setosa", T1 means true "versicolor", T0 means true "virginica". From Table 4, we can find the average accuracy rates for SNN, RF and MNN as 95.00%, 91.67% and 96.00%, respectively. The best accuracy rates for these three methods are 100.00%, 93.33% and 96.67%, respectively. The worst accuracy rates for the three methods are 90.00%, 90.00% and 93.33%, respectively. It follows that averagely MNN produces the best results when the data set has no noise. Although

the best accuracy rate of SNN is 100%, MNN gives the highest accuracy rate on worst performance. RF gives poor accuracy rates.

Table 4 Classification for fisher's iris data (0% noise level)

	SNN			RF			MNN			
	P2	P1	P0	P2	P1	P0	P2	P1	P0	
Average	10	0	0	10	0	0	10	0	0	T2
	0	10	0	0	10	0	0	10	0	T1
	0	1.5	8.5	0	2.5	7.5	0	1.2	8.8	T0
Best	10	0	0	10	0	0	10	0	0	T2
	0	10	0	0	10	0	0	10	0	T1
	0	0	10	0	2	8	0	1	9	T0
Worst	10	0	0	10	0	0	10	0	0	T2
	0	10	0	0	10	0	0	10	0	T1
	0	3	7	0	3	7	0	2	8	T0

In order to check the performance on noisy data, we randomly select 30% data points that is labeled as "setosa" and change their labels to "versicolor", then we randomly select 30% data points that is labeled as "versicolor" and change their labels to "virginica", finally we randomly select 30% data points that is labeled as "virginica" and change their labels to "setosa". The resulting simulation is exhibited in Table 5. From Table 5, we can determine the average accuracy rates for SNN, RF and MNN as 76.67%, 60.00% and 81.00%, respectively. The best accuracy rates are 86.67%, 73.33% and 83.33%, respectively. The worst accuracy rates are 56.67%, 56.67% and 76.67%, respectively. From the accuracy results, we find that averagely MNN yields the best results when the data set is added with 30% noise. RF still gives poor accuracy rates which decrease obviously. Although the best accuracy rate of SNN is 86.67%, MNN gives a much higher accuracy rate on worst performance than SNN whose accuracy rate is only 56.67%.

Table 5 Classification for fisher's iris data (30% noise level).

	SNN			RF			MNN			
	P2	P1	P0	P2	P1	P0	P2	P1	P0	
Average	7.8	2.2	0	5.6	4.4	0	8	2	0	T2
	1.8	7.6	0.6	0	8.1	1.9	1.8	8.2	0	T1
	0.9	1.5	7.6	2.5	3.2	4.3	0	1.9	8.1	T0
Best	8	2	0	7	3	0	9	1	0	T2
	0	10	0	2	7	1	2	8	0	T1
	0	2	8	0	2	8	0	2	8	T0
Worst	3	7	0	5	5	0	8	2	0	T2
	3	6	1	0	7	3	3	7	0	T1
	0	2	8	0	5	5	0	2	8	T0

In summary, we put the accuracy rates of two cases together in Table 6 for easy comparison. From Table 6, we can see that MNN averagely produces the best results both with

and without added noise. RF gives the poorest accuracy rates. The performance of RF decreases apparently when the data set is added with noise.

In the case of 30% noise added, although SNN can achieve 86.67% classification accuracy rate in its best results, its worst accuracy rate is very low (56.67%). On the contrary, MNN achieves 83.33% classification accuracy rate in its best results, while its worst accuracy rate remains high (76.67%). The results demonstrate that MNN has more power in classifying noisy data than SNN and RF.

Table 6 Classification accuracy relative to noise level (fisher's iris data)

Noise level	Method	Average	Best	Worst
No noise	SNN	95.00%	100.00%	90.00%
	RF	91.67%	93.33%	90.00%
	MNN	96.00%	96.67%	93.33%
30% noise	SNN	76.67%	86.67%	56.67%
	RF	60.00%	73.33%	56.67%
	MNN	81.00%	83.33%	76.67%

6.2 Crabs classification data

This data is included in the Matlab software, where its description is available from Matlab helps and cited as follows: "In this demo we attempt to build a classifier that can identify the sex of a crab from its physical measurements. Six physical characteristics of a crab are considered: species, frontal lip, rear width, length, width and depth. The problem on hand is to identify the sex of a crab given the observed values for each of these 6 physical characteristics." In this data set, we randomly select 160 data points as the training set and take the remaining 40 data points as the testing set.

The simulation results are shown in Table 7, where P1 means predicted "Male", P0 means predicted "Female"; T1 means true "Male", T0 means true "Female". From Table 7, we obtain the average accuracy rates for SNN, RF and MNN as 88.00%, 80.50% and 87.75%, respectively. The best accuracy rates are 95.00%, 85.00% and 90.00%, respectively. The worst accuracy rates are 77.50%, 77.50% and 85.00%, respectively. Thus, RF gives the poorest accuracy rates. SNN and MNN are similar in terms of average performance. The former is better with best performance while the latter is better with worst performance when these two methods are compared with each other.

Table 7 Classification for crabs classification data (0% noise level).

	SNN		RF		MNN		
	P1	P0	P1	P0	P1	P0	
Average	19.8	2.2	16.3	5.7	19	3	T1
	2.6	15.4	2.1	15.9	1.9	16.1	T0
Best	22	0	18	4	20	2	T1
	2	16	2	16	2	16	T0
Worst	19	3	16	6	18	4	T1
	6	12	3	15	2	16	T0

In order to check the performance on noisy data, we randomly select 30% data points that is labeled as "Male" and change their labels to "Female", also we randomly select 30% data points that is labeled as "Female" and change their labels to "Male". The resulting simulation is exhibited in Table 8. From Table 8, we calculate the average accuracy rates for SNN, RF and MNN as 67.50%, 56.25% and 80.00%, respectively. The best accuracy rates are 82.50%, 62.50% and 82.50%, respectively. The worst accuracy rates are 47.50%, 52.50% and 75.00%, respectively. From the accuracy results, we find that averagely the method of MNN gives the best results when the data set is added with 30% noise. RF still produces poorest accuracy rates. Although the best accuracy rate of SNN is the same with that of MNN, SNN gives a much lower accuracy rate on worst performance than MNN.

Table 8 Classification for crabs classification data (30% noise level).

	SNN		RF		MNN		
	P1	P0	P1	P0	P1	P0	
Average	14.5	7.5	12.7	9.3	18.7	3.3	T1
	5.5	12.5	8.2	9.8	4.7	13.3	T0
Best	19	3	15	7	19	3	T1
	4	14	8	10	4	14	T0
Worst	4	18	11	11	18	4	T1
	3	15	8	10	6	12	T0

In summary, we put the accuracy rates of two cases together in Table 9 for easy comparison. From the table, we can see that RF produces the worst results both with and without added noise. The performance of MNN is almost the same with the performance of SNN when the data set has no noise. When the 30% noise is added, the accuracy rates of SNN and RF decrease sharply, while the performance of MNN is still good (80.00%). What is more important, the worst accuracy rate of MNN remains high in the case of 30% noise added, which results in a low deviation of the performance.

The results show that MNN has strong power in classifying noisy data set. It also demonstrates that the proposed method is more robust than the other two methods.

Table 9 Classification accuracy relative to noise level (crabs classification data)

Noise level	Method	Average	Best	Worst
No noise	SNN	88.00%	95.00%	77.50%
	RF	80.50%	85.00%	77.50%
	MNN	87.75%	90.00%	85.00%
30% noise	SNN	67.50%	82.50%	47.50%
	RF	56.25%	62.50%	52.50%
	MNN	80.00%	82.50%	75.00%

7. PRACTICAL EXAMPLES

7.1 Stock data

The stock data is always difficult to classify. In this experiment, we use the data from China stock market. To be concrete, we screen all the stocks over ten years (01.01.2001-31.12.2009) to pick up the cases for classification, using the following rule: today return rate (based on close prices) is above or equal to the band about its center line of the last p-day moving average of return rates, for which for illustration p is set as 20 and the band at 1.8 times the standard deviation of the return rates.

For each of these screened cases, we form its input vector x_i with:

- today's return rate
- yesterday's return rate
- the day before yesterday's return rate
- today's volume/(average volume of last 3 days)
- yesterday's volume/(average volume of last 3 days)
- the day before yesterday's volume/(average volume of last 3 days)

After input vectors are obtained, we design a trading rule in order to see its outcomes and define the output label y_i. We trade each screened case as follows:

Buy: today's close price;

Exit: 1.044 times the entry price in any of the following 3 days, or close price of the third day;

Stop: none.

From each round trade, there is an exit price. We then assign output label y_i to each case as follows:

y=1, if selling price >= 1.01 times buying price

y=0; if selling price < 1.01 times buying price

In the end, we have compiled a pair $[x_i \; y_i]$ for each case. Totally, there are 40,120 cases or data points. We randomly select 1600 points for classification experiment, of which 1200 points are assigned as the training data and the remaining 400 as the testing data.

With such a stock data set, the simulation results are shown in Tables 10 and 11. They indicate that;

- The performance of MNN is the best in all the methods,
- The accuracy rates of MNN are slightly better than the accuracy rates of SNN,
- RF gives the poorest performance.

Since the raw data are already very difficult to model, addition of artificial noise is not required. This experiment implies that our proposed method can work in such a real and very difficult application.

Table 10 Classification for stock data

	SNN		RF		MNN		
	P1	P0	P1	P0	P1	P0	
Average	10.7	141.3	24.7	127.3	8.1	143.9	T1
	14.4	233.6	42.2	205.8	9.6	238.4	T0
Best	13	139	25	127	10	142	T1
	12	236	36	212	9	239	T0
Worst	15	137	21	131	9	143	T1
	23	225	46	202	14	234	T0

Table 11 Accuracy rates of classification for stock data

Method	Average	Best	Worst
SNN	61.08%	62.25%	60.00%.
RF	57.63%	59.25%	55.75%
MNN	61.63%	62.25%	60.75%

7.2 MAGIC gamma telescope data 2004

We download the data from the website of Machine Learning Repository at http://archive.ics.uci.edu/ml/index.html. For completeness, we cite the data description from the website as follows:

"The data are MC generated (see below) to simulate registration of high energy gamma particles in a ground-based atmospheric Cherenkov gamma telescope using the imaging technique. Cherenkov gamma telescope observes high energy gamma rays, taking advantage of the radiation emitted by charged particles produced inside the electromagnetic showers initiated by the gammas, and developing in the atmosphere. This Cherenkov radiation (of visible to UV wavelengths) leaks through the atmosphere and gets recorded in the detector, allowing reconstruction of the shower parameters. The available information consists of pulses left by the incoming Cherenkov photons on the photomultiplier tubes, arranged in a plane, the camera. Depending on the energy of the primary gamma, a total of few hundreds to some 10000 Cherenkov photons get collected, in patterns (called the shower image), allowing to discriminate statistically those caused by primary gammas (signal) from the images of hadronic showers initiated by cosmic rays in the upper atmosphere (background).

Typically, the image of a shower after some pre-processing is an elongated cluster. Its long axis is oriented towards the camera center if the shower axis is parallel to the telescope's optical axis, i.e. if the telescope axis is directed towards a point source. A principal component analysis is performed in the camera plane, which results in a correlation axis and defines an ellipse. If the depositions were distributed as a bivariate Gaussian, this would be an equidensity ellipse. The

characteristic parameters of this ellipse (often called Hillas parameters) are among the image parameters that can be used for discrimination. The energy depositions are typically asymmetric along the major axis, and this asymmetry can also be used in discrimination. There are, in addition, further discriminating characteristics, like the extent of the cluster in the image plane, or the total sum of depositions."

In this data set, we randomly select 1400 data points as a training set and other 400 data points as a testing set. The simulation results are shown in Tables 12 and 13.

Table 12 Classification for MAGIC gamma telescope data 2004.

	SNN		RF		MNN		
	P1	P0	P1	P0	P1	P0	
Average	80.2	119.8	102.6	97.4	77.8	122.2	T1
	20.4	179.6	40.5	159.5	2.9	197.1	T0
Best	79	121	112	88	87	113	T1
	2	198	40	160	3	197	T0
Worst	35	165	101	99	71	129	T1
	2	198	47	153	3	197	T0

Table 13 Accuracy rates of classification for MAGIC gamma telescope data 2004.

Method	Average	Best	Worst
SNN	64.95%	69.25%	58.25%
RF	65.53%	68.00%	63.50%
MNN	68.73%	71.00%	67.00%

From the tables, we can see

- The performance of MNN is the best in all the methods,
- SNN gives the poorest performance.

8. CONCLUSIONS

In this paper, we have proposed a new modeling tool – a group of multiple neural networks trained with randomized algorithms – for modeling complex data. It incorporates the randomization idea of random forests into a training algorithm of a neural network and the repeated running of such a revised training algorithm yields multiple independent neural networks. Such a group of models jointly may outperform individual models. Its effectiveness is demonstrated with a variety of examples including practical ones such as stock markets. It works particularly well for the data difficult to learn or model with the exiting methods.

Our simulation shows that the proposed method overall performs better than single neural networks and a random forest. When there is significant noise in the data set, the performance of the former drops relatively less fast than the latter. In particular, the former produces much lower deviation of the performance and high mean performance compared with the latter. Therefore, the proposed method has strong ability to classify the noisy data and perform robustly.

REFERENCES

[1]. Trevor Hastie, Robert Tibshirani and Jerome Friedman, The Elements of Statistical Learning, Springer New York, 2011.

[2]. Ben Krose, Patrick van der Smagt, An introduction to Neural Networks, 1996.

[3]. Hadzibeganovic, Tarik & Cannas, Sergio A., A Tsallis' statistics based neural network model for novel word learning, Physica A: Statistical Mechanics and its Applications 388 (5): 732–746. DOI:10.1016/j.physa.2008.10.042, 2009.

[4]. Van den Bergh, F. Engelbrecht, AP., Cooperative Learning in Neural Networks using Particle Swarm Optimizers, CIRG, 2000.

[5]. Zou Liping, Zou Tao, The application of neural network for Sneak Circuit Analysis on the aircraft electrical system, Prognostics and System Health Management Conference (PHM-Shenzhen), Page(s): 1-5, 2011.

[6]. Arsene. C.T.C., Lisboa. P.J., Bayesian Neural Network with and without compensation for competing risks, Neural Networks (IJCNN), The International Joint Conference, Page(s): 1-8, 2012.

[7]. Lou Haichuan, Su Hongye, Xie Lei, Gu Yong, Rong Gang, Multiple-prior-knowledge neural network for industrial processes, Automation and Logistics (ICAL), IEEE International Conference, Page(s): 385-390, 2010.

[8]. Kondo. T., Ueno. J., Nonlinear system identification by feedback GMDH-type neural network with architecture self-selecting function, Intelligent Control (ISIC), IEEE International Symposium, Page(s): 1521-1526, September 2010.

[9]. Shuang Han, Yongqian Liu, Jie Yan, Neural Network Ensemble Method Study for Wind Power Prediction, Power and Energy Engineering Conference (APPEEC), Asia-Pacific, Page(s): 1-4, March 2011.

[10]. McLeod. P., Verma. B., Clustered ensemble neural network for breast mass classification in digital mammography, Neural Networks (IJCNN), The International Joint Conference, Page(s): 1-6, June 2012.

[11]. E. Gelenbe, Random neural networks with negative and positive signals and product form solution, Neural Computation, vol. 1, no. 4, pp. 502–511, 1989.

[12]. Safavian S.R. and Landgrebe D., A survey of decision tree classifier methodology, Systems, Man and Cybernetics, IEEE Transactions on, volume: 21, pages: 660-674, 1991.

[13]. Anthony,J.M, Robert, N.F,Yang, L.,Nathaniel, A.W and Steven, D.B., An introduction to decision tree modeling, Journal of Chemometrics, 18: 275–285, 2004.

[14]. Yu. L. Pavlov., Random Forests, Utrecht, VSP, 2000.

[15]. Breiman, L., Random Forests. Machine Learning Journal 45, 532, 2001.

[16]. V. Svetnik, A. Liaw, C. Tong, J. C. Culberson, R. P. Sheridan and B. P. Feuston, Random forest: a classification and regression tool for compound classification and QSAR modeling, Journal of chemical information and computer sciences, volume: 43, pages: 1947--1958, 2003.

[17]. Maragoudakis, M and Serpanos, D., towards stock market data mining using enriched random forests from textual resources and technical indicators. AIAI, IFIP AICT 339, pp. 278–286, 2010.

[18]. Manish Kumar, Thenmozhi M., Forecasting Stock Index Movement: A Comparison of Support Vector Machines and Random Forest, Indian Institute of Capital Markets 9th Capital Markets Conference, January 2006.

[19]. Zhenping Yi, Jingchang Pan, Application of random forest to stellar spectral classification, Image and Signal Processing (CISP), 3rd International Congress, Volume: 7, Page(s): 3129-3232, 2010.

[20]. Dittman. D., Khoshgoftaar. T.M., Wald. R., Napolitano. A., Random forest: A reliable tool for patient response prediction, Bioinformatics and Biomedicine Workshops (BIBMW), IEEE International Conference, Page(s): 289-296, 2011.

[21]. Moschidis. E., Graham. J., Automatic differential segmentation of the prostate in 3-D MRI using Random Forest classification and graph-cuts optimization, Biomedical Imaging (ISBI), 9th IEEE International Symposium, Page(s): 1727-1730, 2012.

[22]. Breiman, L., Random Forests. Machine Learning Journal 45, 532, 2001.