Disponible en www.sciencedirect.com

Journal of Applied Research and Technology

Journal of Applied Research and Technology 13 (2015) 188-196

www.jart.ccadet.unam.mx

Original

A constructive algorithm for unsupervised learning with incremental neural network

Jenq-Haur Wang, Hsin-Yang Wang, Yen-Lin Chen, Chuan-Ming Liu*

Department of Computer Science and Information Engineering, National Taipei University of Technology, Taipei, Taiwan

Received 19 April 2014; accepted 18 August 2014

Abstract

Artificial neural network (ANN) has wide applications such as data processing and classification. However, comparing with other classification methods, ANN needs enormous memory space and training time to build the model. This makes ANN infeasible in practical applications. In this paper, we try to integrate the ideas of human learning mechanism with the existing models of ANN. We propose an incremental neural network construction framework for unsupervised learning. In this framework, a neural network is incrementally constructed by the corresponding subnets with individual instances. First, a subnet maps the relation between inputs and outputs for an observed instance. Then, when combining multiple subnets, the neural network keeps the corresponding abilities to generate the same outputs with the same inputs. This makes the learning process unsupervised and inherent in this framework.

In our experiment, Reuters-21578 was used as the dataset to show the effectiveness of the proposed method on text classification. The experimental results showed that our method can effectively classify texts with the best F1-measure of 92.5%. It also showed the learning algorithm can enhance the accuracy effectively and efficiently. This framework also validates scalability in terms of the network size, in which the training and testing times both showed a constant trend. This also validates the feasibility of the method for practical uses.

Keywords: Artificial neural network; Unsupervised learning; Text classification

1. Introduction

Artificial neural network (ANN) has been applied to many tasks including pattern recognition, optimization, clustering, and classification. However, comparing with other classification methods, ANN needs enormous memory space and training time to build the model. This makes ANN infeasible in practical applications. In this paper, we try to integrate the ideas of human neural system and learning mechanism with the existing models of ANN. We propose an incremental neural network construction framework that is inspired by human learning process. In this framework, a neural network is incrementally constructed by basic components called *subnets* corresponding to the observed instances. It's able to adjust its structure and weights automatically based on training instances by an unsupervised learning algorithm inherent in this framework. First, for each new training instance, a subnet is constructed by mapping the corresponding inputs and outputs into nodes and their connections with edges in a graph. Since the subnet is able to generate the same outputs with the same inputs once the edges have been built, it is considered to learn the ability to identify the corresponding pattern of input features. Then, to obtain a neural network with abilities to identify multiple patterns of input features, we combine the corresponding subnets each weighted with a construction coefficient to reflect the relative importance. For instances whose input and output nodes already exist, the edge weight in the corresponding instance is adjusted by a learning coefficient. Then, given test instances, the constructed neural network is likely to generate the corresponding outputs given the same inputs.

To evaluate the incremental neural network construction framework in real applications, Reuters-21578 was used as the dataset in our experiments to show the effectiveness of the proposed algorithm on text classification. The experimental result showed that our method can effectively classify texts with the best F1-measure of 92.5%. It also showed the learning algorithm can enhance the accuracy effectively and efficiently. As the amount of data increases, the training and testing time showed a constant trend. This validates the better scalability of our proposed approach for practical uses.

*Corresponding author.
E-mail address: cmliu@csie.ntut.edu.tw (C.M. Liu).

2. Related works

Artificial neural networks (ANNs) have been widely studied and adopted in many tasks, for example, pattern recognition (Bishop, 1996), text classification (Apte et al., 1994), optimization (Cochocki & Unbehauan, 1993), and data clustering (Herrero et al., 2001), to name a few. Recently, applications of ANNs even extend to workflow scheduling in critical infrastructure systems (Vukmirović et al., 2012) and intelligent control (Rivera-Mejía et al., 2012). Since the widespread use of ANN, various types of neural networks have been proposed. The most popular and frequently used type of artificial neural networks is multilayer perceptron (Rosenblatt, 1962) with back propagation (Rumelhart et al., 1986). The algorithm determines the network weights by gradient descent procedure, but the network structure is found by trial and error. Thus, the training and test processes are usually time-consuming. This impedes the practical applications of ANNs. Also, the network structure is one major issue in neural networks. It's not easy to automatically determine the best number of nodes and the number of layers in an ANN. For example, Shen and Hasegawa (2006) proposed an incremental network called Self-Organizing Incremental Neural Network (SOINN) for classification and topology learning. They adopted a two-layer structure which allows new nodes to be inserted. Aran and Alpaydin (2003) proposed a constructive algorithm with Multiple Operators using Statistical Test (MOST) for determining the network structure. Operators such as adding and removing one or more hidden units to/from a layer and adding a new layer are used, and 5×2 cross validation is used to find near optimal results. This makes a high time complexity.

Text classification is well studied in the research field of information retrieval. The state-of-the-art in text classification usually applies machine learning techniques such as Support Vector Machines (SVM). However, supervised learning is usually needed that requires the labelled data for training process. As the amount of data grows, the efficiency in training and testing will make it impractical for real applications. Applying neural networks for text classification is also quite common. For example, Lee et al. (2000) proposed to analyze terms and meaningful textual features to improve the accuracy of back-propagation neural networks. Liu and Zhang (2001) proposed to change the objective function to improve the efficiency of feed-forward neural networks. Zhang and Zhou (2006) proposed a model called Back Propagation for Multi-Label Learning (BP-MLL), which improved the training time for back propagation neural networks. Ghiassi et al. (2012) proposed a Dynamic Architecture for Artificial Neural Network (DAN2) to provide better scalability for feed-forward neural networks.

In our proposed approach, we try to modify the existing model of artificial neural networks by integrating the idea of human learning process. In this framework, an incremental construction process is used to add nodes and links corresponding to the observed instances. Unsupervised learning mechanism is inherent with the learning coefficient to assign relative weights in each observation. This helps improve the efficiency without affecting the effectiveness.

3. The proposed method

This paper proposes an incremental neural network (INN) construction algorithm that consists of three parts: subnet construction, subnet combination, and unsupervised learning. Each part is described in the following subsections in details.

3.1. Subnet construction

The basic element in an incremental neural network consists of input and output nodes and edges connecting the nodes with weights. Given an observed instance with feature set I and response set O, Subnet Construction module first creates the corresponding nodes in I and O if they do not already exist. Then, the edges $E = I \times O = \{(i,o) \mid i \in I \text{ and } o \in O\}$ are added to reflect the structural relations in the observed instance. The resulting subnet is constructed as $S = \{I, O, E\}$.

For example, given an observed instance with two features A and B ($I = \{A, B\}$) and a response X ($O = \{X\}$), we denote this relation between input and output as follows: $(A, B) \rightarrow X$. The simplest subnet to capture this "ability" is a simple network structure with two input nodes and one output node as illustrated in Figure 1.

Next time the neural network encounters input features A and B together, it will react and trigger the output X as it already learns the pattern based on the previous learning "experience" that is reflected by the network structure in the corresponding subnet.

This step is simple, but very critical to the INN construction framework. First, it allows addition of new nodes and edges in the given network structure. Second, the learning mechanism is inherent in the framework since the corresponding "ability" to recognize the pattern associated with the input features, and to respond the learned output is directly reflected given the observed instance. That makes learning process fast and simple. This is inspired by the Hebbian theory (Hebb, 1949) of human neural systems, which is often summarized as "neurons that fire together wire together".

Note that since there's no difference between the importance of new edges, each new edge is given the same initial edge weight. When different subnets are combined in the next step, the relative weights between subnets are more critical to determine their relative importance.

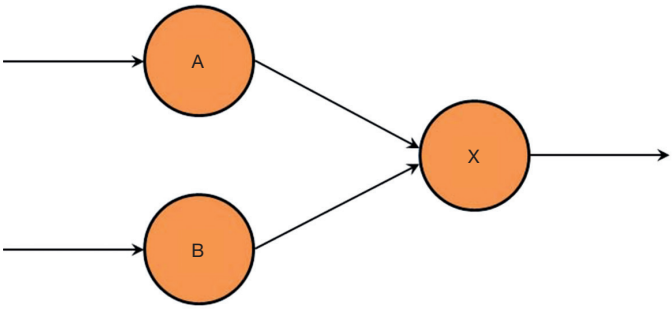

Fig. 1. A sample subnet with two input nodes and one output node.

3.2. Subnet combination

When there are more than one observed instances, the corresponding subnets can be constructed as in the previous section. To make the network capable of recognizing patterns learned, the next step is to combine the subnets. The combination of subnets $S_i = \{I_i, O_i, E_i\}$ can be done as follows. First, the input and output nodes of the combined neural network are simply the union of the corresponding input nodes I_i and output nodes O_i in each component subnet S_i. That is, $I = Union(I_i)$ and $O = Union(O_i)$. Then, the edges of the combined neural network are added if they do not already exist. Each subnet in combination is assigned a constant called *construction coefficient* α_i, which represents the relative degree of influence for this subnet S_i. All the edge weights in each subnet are modified by α_i, that is, $E = Union(E_i)$ where edge weight $w_{jk} = w_{jk} * \alpha_i$. An example of combination with three simple subnets is illustrated in Figure 2.

For instances whose edges already exist, the corresponding edge weight will be added. That makes each observed instance some influence on the final network structure. This is also consistent with human learning process where every single observation leaves traces in human brains. Note that the construction coefficients α_i reflect the relative importance of each component subnet S_i in combination. It depends on the relative weight of influence for abilities of recognizing different patterns.

3.3. Unsupervised learning

When the system is given instances one by one, the incremental neural network construction can proceed until the system is equipped with enough abilities to identify patterns of input features. Then, the system can be used to solve the corresponding problems which consist of these individual abilities of recognition. Although it's similar to the training and testing phases in a supervised learning process, this learning can be done in an unsupervised way without actually differentiating the two phases. It is similar to human learning process. Memorizing new things is a kind of training process, but the re-writing of neural networks in human brains can also be invoked each time when we perceive known objects or recall existing memories. The differences in the contexts in which we perceive existing things might affect our memory, and thus modifying the corresponding perceptions of the same objects.

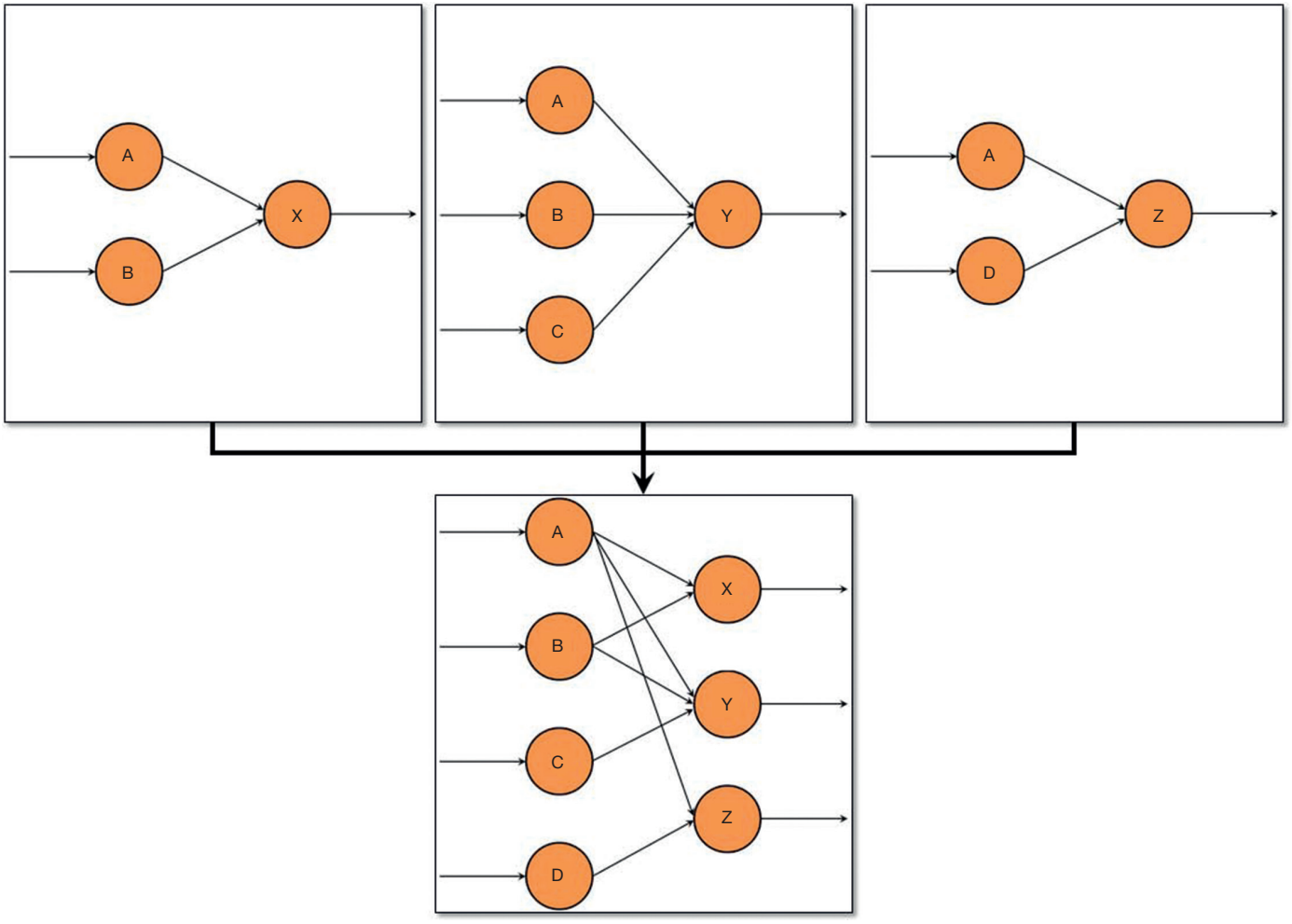

Fig. 2. An example of subnet combination with three subnets.

The process of unsupervised learning is inherent in our proposed framework. Depending on different objectives, we can incrementally construct and enrich the neural network as long as new instances are given. When there's a match with the edges in existing subnets, the corresponding connection between input and output is reinforced. In the proposed approach, this is done by updating the corresponding edge weight w_{jk} by adding a *learning coefficient* β. This reflects the learning speed and also the flexibility of the proposed approach. It will also affect the precision of certain learned cases if the learning coefficient is too large. Thus, the appropriate values of learning coefficients also affect the effectiveness of the proposed system. In this study, we will set learning coefficients the same values as construction coefficients. This makes training and testing processes similar impacts on the network structure. Besides learning coefficients, there's no extra learning mechanism needed as in conventional models of artificial neural networks. This makes learning mechanism inherent in our proposed framework.

In our design, the weight w_{jk} of an edge e_{jk} is influenced by two major factors: construction coefficient α_i and learning coefficient β. The edge weight is multiplied by construction coefficient when the subnet was combined with other subnets; while the edge weight is added by learning coefficient when the same pattern was perceived. Thus, construction coefficient reflects the relative degree of influence of newly perceived patterns, while learning coefficient reflects the learning speed of the unsupervised learning process.

There are several benefits of the proposed approach. Although the network structure is incrementally modified with every construction and combination operation of subnets, the learning process is very fast. This is due to the simple operations in our design. The network structure has a linear growth which makes the construction time remain relatively constant to the network size. The major characteristic of the proposed approach is the scalability. Since the network structure is only incrementally modified, there's no need to retrain the whole network from scratch when new abilities of recognition are needed. Finally, the framework is tolerant to noises in observation. Since each instance might have some impact on the final network structure, a few noises in the training data will not greatly affect the overall learning result as long as there are more correct instances than noises.

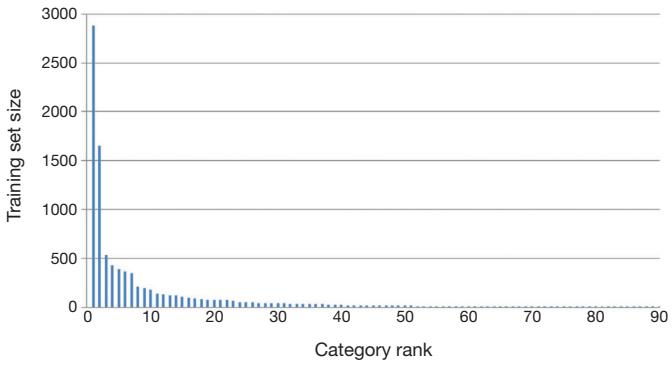

Fig. 3. The distribution of training set size in Reuters-21578.

4. Experiments

To evaluate the feasibility of INN construction and unsupervised learning, we adopted text classification as our main focus in experiments. We utilized the well-known Reuters-21578 collection as our dataset. The ModApte was used to divide documents into training and test sets. After removing categories without training and test documents in Reuters-21578 data set, there are 90 categories in total. But the distribution of documents in classes is very unbalanced: the largest class contains 2877 documents, while 33% of the classes contain less than 100 documents. The detailed distribution of Reuters-21578 data set is shown in Figure 3.

Given such unbalanced distribution of class sizes, we utilized 90 classes as well as the largest 10 classes in our experiments.

The system architecture for incremental neural network construction for text classification consists of five major modules: feature identification, feature node construction, relation analysis, similarity estimation, and unsupervised learning. The architecture is depicted in Figure 4.

4.1. Modules for incremental neural network construction in text classification

In this subsection, we illustrate the process of INN construction for text classification using a simple example.

First, in feature identification, we match the text features with the existing nodes in the network structure. If there's a match, the text features will be converted into feature node number. If there's no match, a new feature node number will be generated in training cases. Non-existing text features will not be considered in testing cases.

Second, in feature node construction, the corresponding new nodes are created and the edges are added into the existing neural network. For example, suppose the current neural network consists of three input nodes {A, B, C} and two output nodes {X, Y} as shown in Figure 5.

Now, given a new training instance (B, C, D) → X, the results are shown as follows.

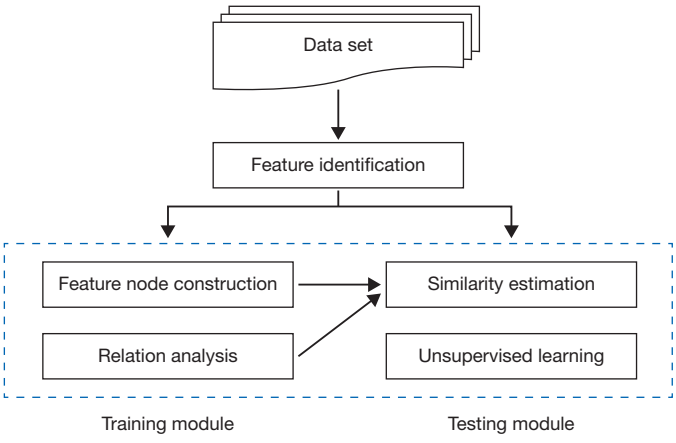

Fig. 4. The system architecture of incremental neural network for text classification.

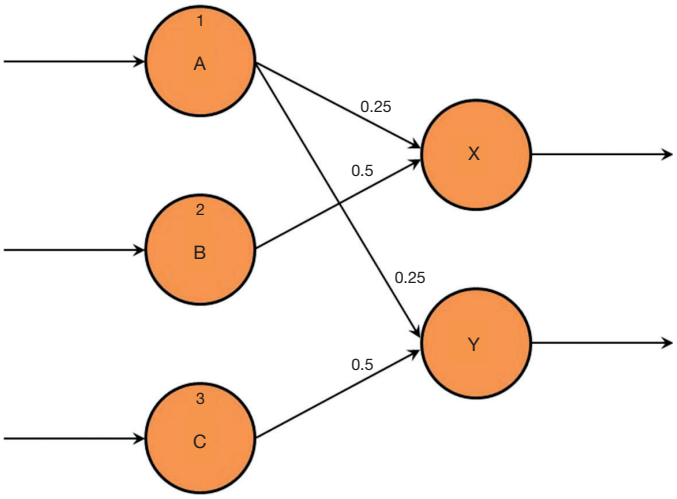

Fig. 5. An example of incremental neural network with three input nodes {A, B, C} and two output nodes {X, Y}.

As shown in Figure 6, a new node D is constructed that corresponds to the training instance (B, C, D) → X. In the case of a test instance (A, C, D), the network structure will not be modified since it's not for training.

The third module of relation analysis is to generate the corresponding mapping between inputs and outputs. New relations will be generated according to the corresponding instance. If the relation already exists, it will be merged into existing relations. For example, suppose the construction coefficient is 0.25, with the training instance (B, C, D) → X, the original network structure in Figure 5 will be modified as follows.

As shown in Figure 7, the weight of edge (B, X) is merged since it already exists when the training instance was observed.

Next, similarity estimation is needed to derive the most likely category for the test instance. For example, in the above-mentioned case as shown in Figure 5, given the test instance (A, C, D), we can derive the similarity between input instance (A, C, D) and class X as 0.25, and the similarity between input instance (A, C, D) and class Y as 0.75. Thus, the output of this test case will be class Y.

Finally, unsupervised learning can be automatically done when we add the corresponding weights by learning coefficients β. For example, suppose a learning coefficient of 0.25 for the test case (A, C, D), the network structure in Figure 5 after unsupervised learning is modified as follows.

As shown in Figure 8, the feature node D was not added since the test case is not for training. But the effects of learning coefficient β can be observed since perceiving existing things will reinforce existing connections and thus re-write the corresponding weights.

4.2. Effectiveness for text classification

To evaluate the performance of our proposed approach for text classification, we used the popular Weka platform and compared the performance with multilayer perceptron with back propagation (ANN), Naïve Bayes (NB), SVM, and kNN. After some trial and error, we obtained the best performance on ANN when there's one hidden layer with 64 nodes. For kNN, we obtained the best performance when $k = 3$. To test the effects of unsupervised learning, we divided our method into two types: INN-UL and INN for incremental neural network with and

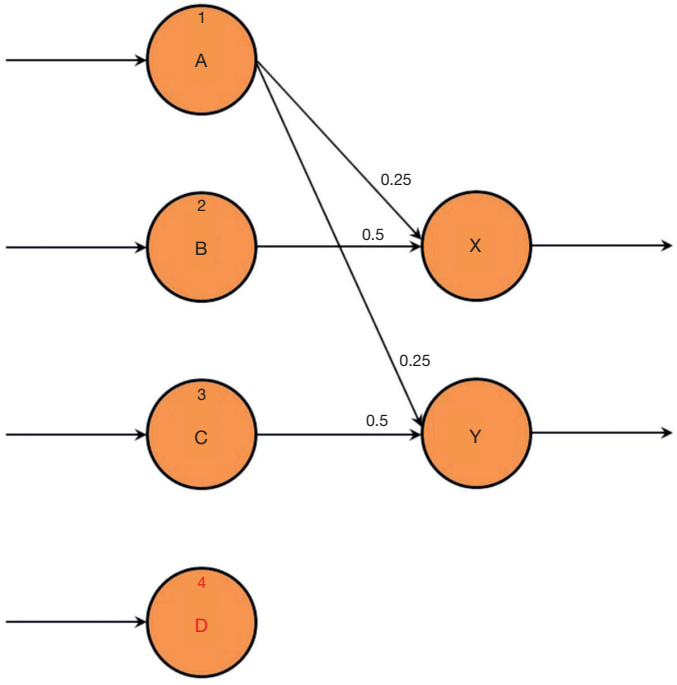

Fig. 6. The incremental neural network after feature node construction when observing a new training instance (B, C, D) → X.

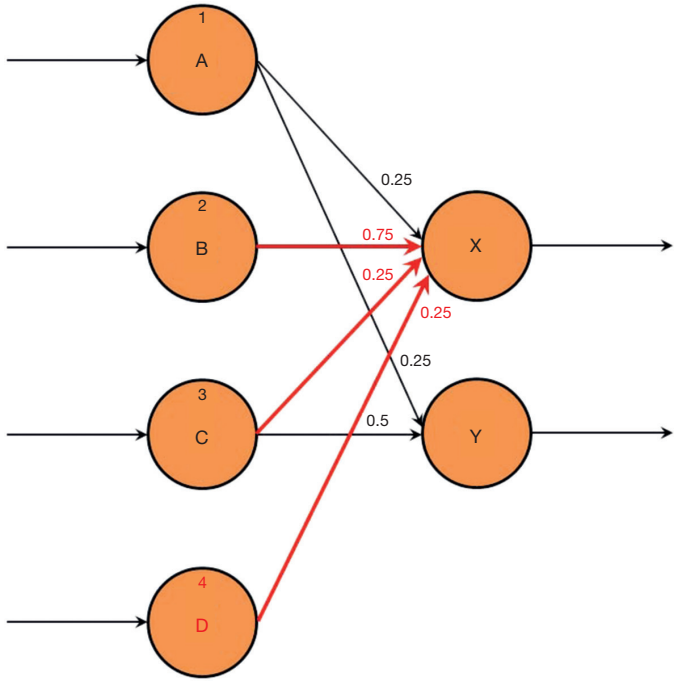

Fig. 7. The incremental neural network after relation analysis when observing a new training instance (B, C, D) → X.

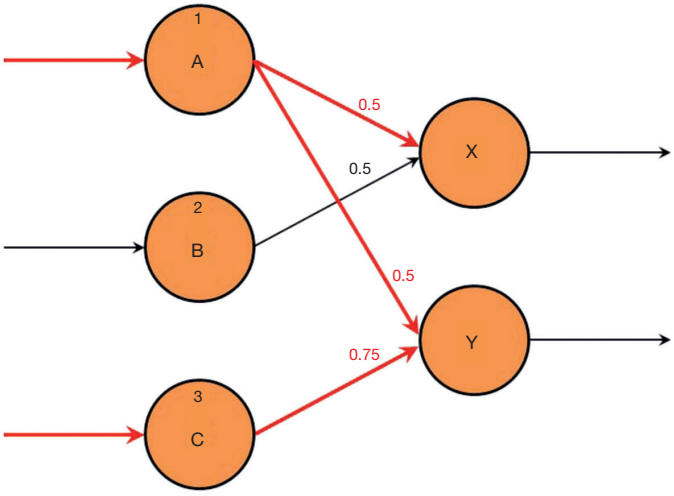

Fig. 8. The incremental neural network after unsupervised learning for the test case of (A, C, D).

without unsupervised learning. The evaluation metrics for effectiveness are: micro-averaging and macro-averaging precision, recall, and $F1$-measure. They are explained as follows.

The precision of a text classifier on class c_i is defined as the fraction of documents assigned to class c_i that are actually from the class, that is:

$$P(c_i) = \frac{n_{ii}}{\sum_j n_{ji}} \tag{1}$$

where n_{ij} denotes the number of documents in class c_i that are assigned class c_j.

On the other hand, the recall of a text classifier on class c_i is the fraction of documents in class c_i that are really assigned to the correct class, that is:

$$R(c_i) = \frac{n_{ii}}{\sum_j n_{ij}} \tag{2}$$

$F1$ measure is defined as the harmonic mean of precision and recall, that is:

$$F1(c_i) = \frac{2 * P(c_i) * R(c_i)}{P(c_i) + R(c_i)} \tag{3}$$

Since there are more than one class, to combine multiple performance metrics into one, we utilized two different ways: micro-averaging, and macro-averaging. Macro-averaging precision is defined as a simple arithmetic average of the individual precision of each class c_i, that is:

$$P_{macroavg} = \frac{\sum_i P(c_i)}{n} \tag{4}$$

where n is the number of classes. Macro-averaging recall and $F1$ measure can be similarly defined. On the other hand, micro-averaging precision is defined as the precision of all classification decisions from all classes, that is:

$$P_{microavg} = \frac{\sum_i TP_i}{\sum_i TP_i + \sum_i FN_i} \tag{5}$$

where TP_i is the true positive rate for class c_i, and FN_i is the false negative rate for class c_i. Micro-averaging recall and $F1$ measure can also be similarly defined.

Regarding efficiency, we evaluated the training and testing times at different training set sizes for different methods.

Using these evaluation metrics, we evaluated the performance of our proposed approach. First, we checked the effects of text classification for the 10 largest classes among the 90 classes in ModApte of Reuters-21578 collection using different training set sizes. For the 10 largest classes, the classification accuracy is shown in Table 1.

As shown in Table 1, the micro-averaging and macro-averaging $F1$ measures for SVM are the best among different methods. However, the proposed INN and INN-UL can achieve comparable performances to SVM, and better performances than ANN, NB, and kNN.

Next, to check the effects of the training set size, the micro-averaging and macro-averaging $F1$ for the 10 largest classes at different training set sizes are shown in Figure 9.

As shown in Figure 9, the classification performances of INN and INN-UL for the 10 largest classes are among the top 3 at different training set sizes. The performances for INN, INN-UL, and SVM are comparable. This validates the effectiveness of our proposed approach.

Similarly, for the 90 classes, the classification accuracy is shown in Table 2, and the micro-averaging and macro-averaging $F1$ for different training set sizes are shown in Figure 10.

As shown in Figure 10, although the general performance for 90 classes is inferior to those for the 10 largest classes, similar results can be obtained that validates the comparable performance of INN and INN-UL to SVM in the case of unbalanced distribution for 90 classes.

4.3. Efficiency for text classification

To evaluate the efficiency of the proposed approach for text classification, we utilized the same dataset as the previous section and compared the training and testing times with existing methods. The training time for the 10 largest classes is shown in Figure 11.

Table 1
The accuracy for the 10 largest classes.

	INN	INN-UL	kNN	SVM	NB	ANN
Micro-$F1$	0.912	0.925	0.705	**0.932**	0.788	0.913
Macro-$F1$	0.803	0.823	0.576	**0.834**	0.664	0.784

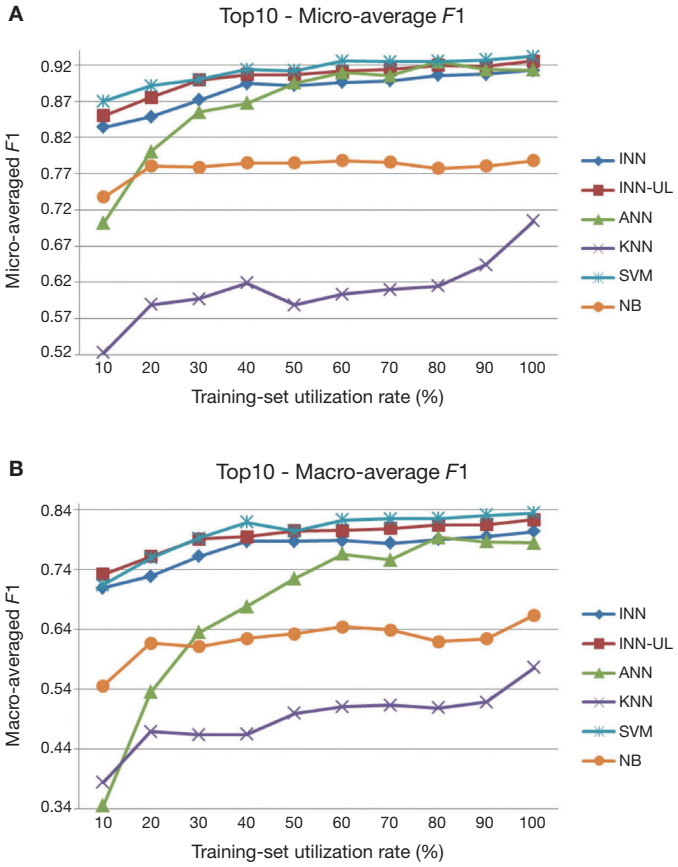

Fig. 9. The (A) micro-averaging $F1$-measure (B) and macro-averaging $F1$-measure for the 10 largest classes in Reuters-21578 at different training set sizes.

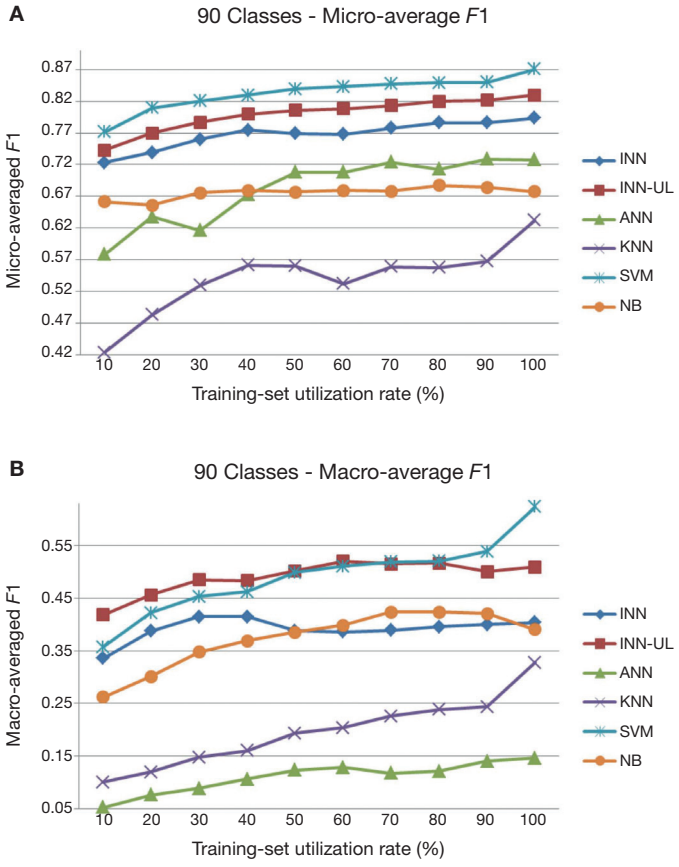

Fig. 10. The (A) micro-averaging $F1$-measure (B) and macro-averaging $F1$-measure for the 90 classes in Reuters-21578 at different training set sizes.

As shown in Figure 11A, the training time for conventional ANN and NB are much larger than the other methods. Thus, we scaled the chart for the remaining methods as in Figure 10B. Although kNN takes the least time for training, INN and INN-UL remain relatively constant to the training set size. SVM takes more time to train when the data size gets larger. This shows the efficiency of our proposed approach in training time.

Next, we compared the testing time for the 10 largest classes at different training set sizes in Figure 12.

As shown in Figure 12A, the testing time for conventional ANN and NB are still much larger than the other methods. Thus, we scaled the chart for the remaining methods in Figure 12B. kNN takes longer time for testing, while INN and INN-UL perform the best in which the testing time almost remain constant for different training set sizes. This shows the scalability of our proposed method.

Then, in the case of 90 classes, similar results for training time can be obtained as shown in Figure 13.

Table 2
The accuracy for the 90 classes.

	INN	INN-UL	kNN	SVM	NB	ANN
Micro-$F1$	0.793	0.830	0.632	**0.871**	0.677	0.727
Macro-$F1$	0.404	0.510	0.328	**0.624**	0.390	0.146

Finally, the testing times for the 90 classes are shown in Figure 14.

As shown in Figure 14B, although the testing time for INN, INN-UL, and SVM are comparable, INN and INN-UL show a relatively constant testing time at different training set sizes. The testing time for SVM at larger training set size showed a slight increasing trend. The efficiency of the proposed approach in testing time can be validated at different training set sizes.

5. Discussions

According to the experimental results in the previous section, we have the following observations:

1. Although SVM still achieves the best performance for text classification in terms of micro-averaging and macro-averaging $F1$ measures, our proposed methods INN and INN-UL have comparable performances at different training set sizes. INN and INN-UL consistently performs better than ANN, NB, and kNN.
2. The efficiency of the proposed approach in training and testing times are consistently the best among all methods at different training set sizes. Both the training and testing times showed a relatively constant complexity. This validates the

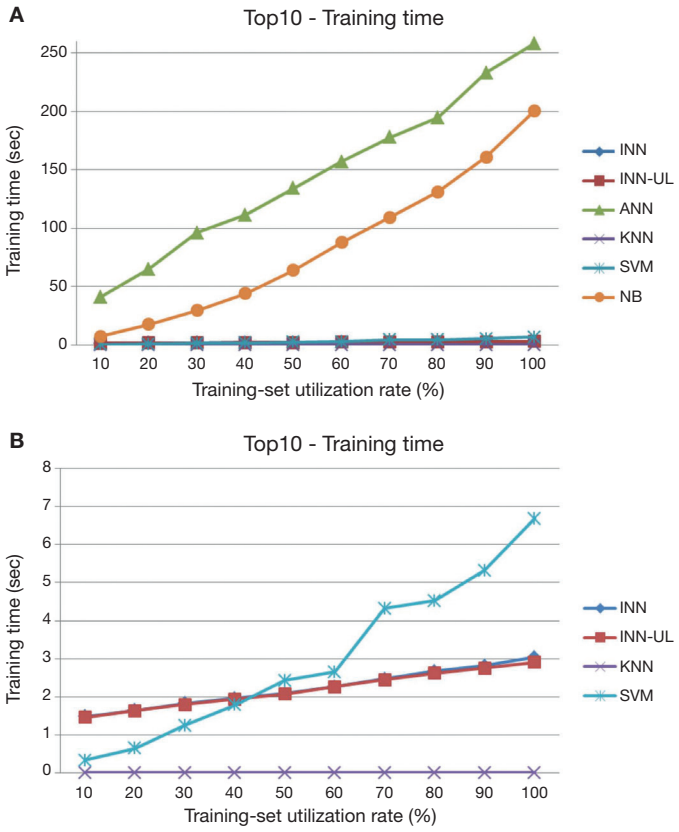

Fig. 11. The training time for the 10 largest classes at different training set sizes.

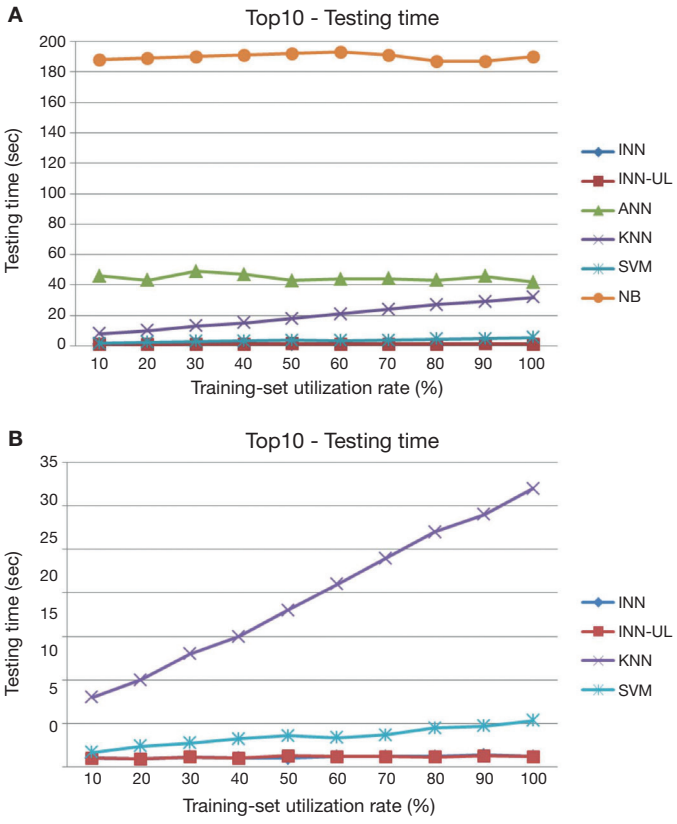

Fig. 12. The testing time for the 10 largest classes at different training set sizes.

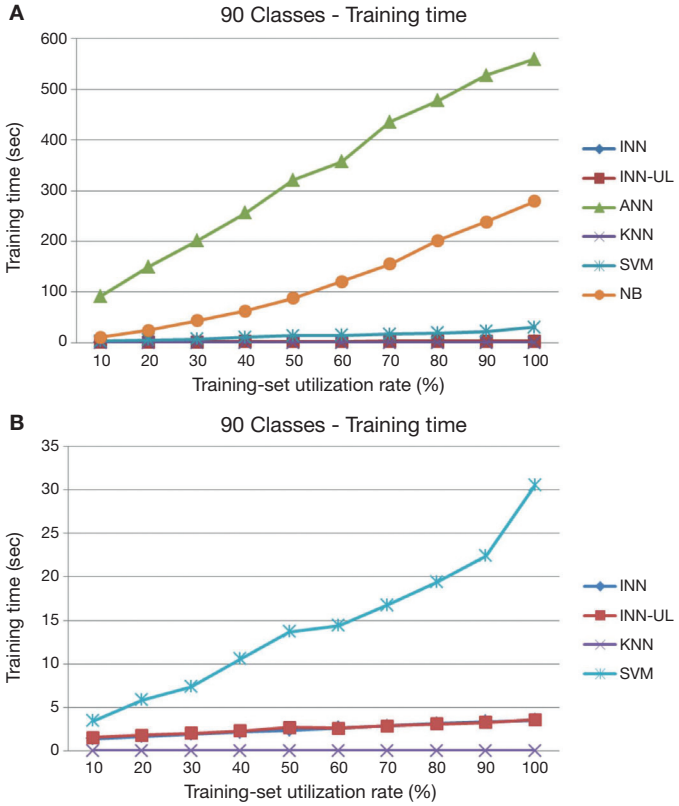

Fig. 13. The training time for the 90 classes at different training set sizes.

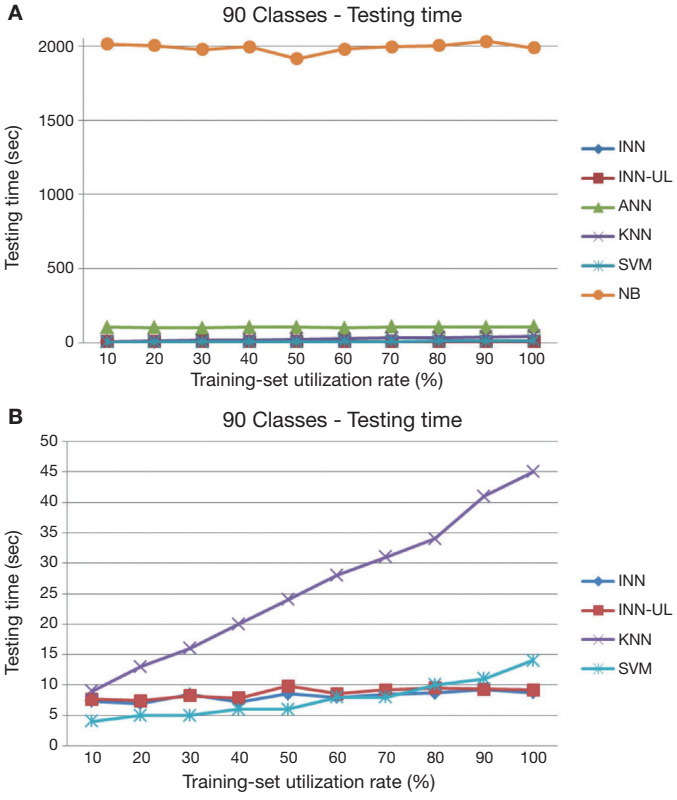

Fig. 14. The testing time for the 90 classes at different training set sizes.

high scalability of our proposed approach in text classification. It helps improve the feasibility in real applications.

3. Comparing to most existing approaches of ANN which assumes a two-layer network structure with one hidden layer, only one single-layer network structure is assumed in the proposed approach. Also, the operations for network construction and combination in the proposed approach only involve the addition of nodes and edges. In the case of text classification, there is limited impact in the absence of hidden layers and node removal operations. Thus, the feasibility of the algorithm in other application domains has to be investigated in the future.

6. Conclusions

ANNs have wide applications, but existing models might not be feasible in practical uses. In this paper, we have proposed a new model that incrementally constructs neural networks. Also, in our experiment in text classification, our method can effectively classify texts with the best *F*1-measure of 92.5%. It also showed the learning algorithm can enhance the accuracy effectively and efficiently. This framework validates scalability in terms of the network size, in which the training time and test time both showed a constant trend. Further investigation is needed to verify the effects in other applications.

Acknowledgements

The authors would like to thank the support from National Science Council, Taiwan, under the grant number NSC102-2219-E-027-005.

References

Apte, C., Damerau, F., & Weiss, S.M. (1994). Automated learning of decision rules for text categorization. *ACM T. Inform. Syst., 12,* 233-251.

Aran, O., & Alpaydin, E. (2003). An incremental neural network construction algorithm for training multilayer perceptrons. In Kaynak, O., Alpaydin, E., Oja, E., & Xu, L. (Eds.), *Artificial Neural Networks and Neural Information Processing.* Istanbul, Turkey: ICANN/ICONIP 2003.

Bishop, C.M. (1996). *Neural Networks for Pattern Recognition* (1st ed.). New York: Oxford University Press.

Cochocki, A., & Unbehauan, R. (1993). *Neural networks for optimization and signal processing* (1st ed.). New York: Wiley.

Ghiassi, M., Olschimke, M., Moon, B., & Arnaudo, P. (2012). Automated text classification using a dynamic artificial neural network model. *Expert Syst. Appl., 39,* 10967-10976.

Hebb, D.O. (1949). *The Organization of Behavior.* New York: Wiley & Sons.

Herrero, J., Valencia, A., & Dopazo, J. (2001). A hierarchical unsupervised growing neural network for clustering gene expression patterns. *Bioinformatics, 17,* 126-136.

Lee, H.M., Chen, C.M., & Hwang, C.W. (2000). A neural network document classifier with linguistic feature selection. In Loganantharaj, R., Palm, G., & Ali, M. (Eds.), *Intelligent problem solving. methodologies and approaches* (pp. 555-560). New York: Springer.

Liu, Z.Q., & Zhang, Y. (2001). A competitive neural network approach to web-page categorization. *Int. J. Uncertain. Fuzz., 9,* 731-741.

Rivera-Mejía, J., Léon-Rubio, A.G., & Arzabala-Contreras, E. (2012). PID based on a single artificial neural network algorithm for intelligent sensors. *Journal of Applied Research and Technology, 10,* 262-282.

Rosenblatt, F. (1962). *Principles of neurodynamics: perceptrons and the theory of brain mechanisms.* Washington: Spartan Books.

Rumelhart, D.E., Hinton, G.E., & McClelland, J.L. (1986). Learning representations by back-propagating errors. *Nature, 323,* 533-536.

Shen, F., & Hasegawa, O. (2006). An incremental network for on-line unsupervised classification and topology learning. *Neural Networks, 19,* 90-106.

Vukmirović, S., Erdeljan, A., Imre, L., & Čapko, D. (2012). Optimal workflow scheduling in critical infrastructure systems with neural networks. *Journal of Applied Research and Technology, 10,* 114-121.

Zhang, M.L., & Zhou, Z.H. (2006). Multi-label neural networks with applications to functional genomics and text categorization. *IEEE T. Knowl. Data En., 18,* 1338-1351.