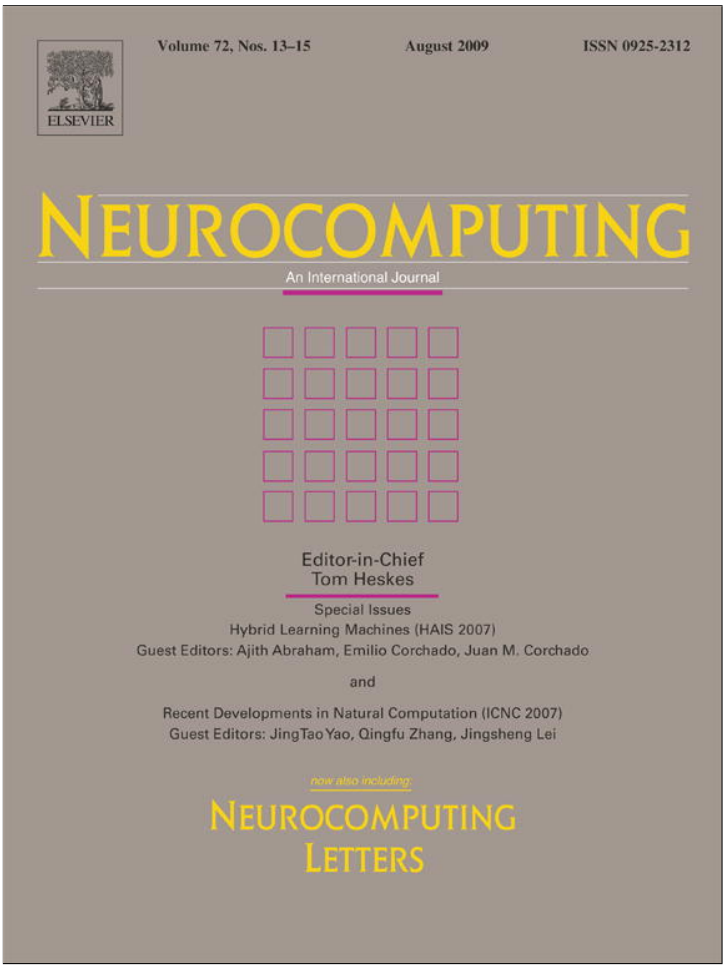

Neurocomputing 72 (2009) 2796–2805

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Selective negative correlation learning approach to incremental learning ☆

Ke Tang [a,*], Minlong Lin [a], Fernanda L. Minku [b], Xin Yao [a,b]

[a] Nature Inspired Computation and Applications Laboratory (NICAL), Department of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, China

[b] The Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA), School of Computer Science, the University of Birmingham, Edgbaston, Birmingham B15 2TT, UK

ARTICLE INFO

Available online 16 April 2009

Keywords:
Negative correlation learning
Neural network ensemble
Incremental learning
Selective ensemble

ABSTRACT

Negative correlation learning (NCL) is a successful approach to constructing neural network ensembles. In batch learning mode, NCL outperforms many other ensemble learning approaches. Recently, NCL has also shown to be a potentially powerful approach to incremental learning, while the advantages of NCL have not yet been fully exploited. In this paper, we propose a selective NCL (SNCL) algorithm for incremental learning. Concretely, every time a new training data set is presented, the previously trained neural network ensemble is cloned. Then the cloned ensemble is trained on the new data set. After that, the new ensemble is combined with the previous ensemble and a selection process is applied to prune the whole ensemble to a fixed size. This paper is an extended version of our preliminary paper on SNCL. Compared to the previous work, this paper presents a deeper investigation into SNCL, considering different objective functions for the selection process and comparing SNCL to other NCL-based incremental learning algorithms on two more real world bioinformatics data sets. Experimental results demonstrate the advantage of SNCL. Further, comparisons between SNCL and other existing incremental learning algorithms, such Learn++ and ARTMAP, are also presented.

1. Introduction

Many machine learning methods involve only a single training session (i.e., batch learning), in which all the available data are presented and learned concurrently. However, in real-world applications, the data are usually not available at one time, but chunk by chunk. In other words, a number of training data sets are presented one after another. Therefore, it is desirable for a learning algorithm to be capable of acquiring knowledge from new data. Such a learning process is referred to as incremental learning [8]. In general, it is suggested that an incremental learning algorithm should enable the learner to use any new training data to further improve its performance. This could involve accommodating new classes of data that are introduced with the new data [22,28]. Further, an incremental learning algorithm is also expected to preserve previously learned knowledge without access to the previous data, i.e., it should not suffer from catastrophic forgetting [22,28].

In the literature, there are several incremental learning algorithms. For example, Carpenter et al. proposed the adaptive resonance theory modules map (ARTMAP) [3] and fuzzy ARTMAP [4] in 1991 and 1992, respectively. These algorithms work based on the generation of new clusters in response to new data that is sufficiently different from previous ones. Kasabov also proposed a one-pass incremental learning algorithm, namely evolving fuzzy neural network (EFuNN) [11]. Another important incremental learning approach, the Learn++ [22], is based on the adaptive boosting (AdaBoost) algorithm [27]. Furthermore, two recently proposed methods, the evolved incremental learning for neural networks (EILNN) [28] and the self-organizing neural grove (SONG) [9] have also been shown to be effective for incremental learning.

Originally proposed for batch learning tasks, negative correlation learning (NCL) is a successful approach to designing neural network (NN) ensembles [16,17]. In the literature, NCL has shown a number of empirical successes on various applications, including classification problems [16,17], regression problems [31], and time-series prediction [14]. It has consistently demonstrated very competitive results with other ensemble learning techniques such as mixtures of experts, bagging, and boosting [2,15]. Motivated by those successes, Minku et al. recently suggested that NCL might also be a potentially powerful approach to incremental learning [20]. The reason is, when using NCL to train an NN ensemble, the individual NNs of the ensemble are highly different among one another and at the same time have high accuracy. So, when new training data are presented, the diversity among the individuals makes them adapt to new data in different ways. Such difference

☆ A preliminary version of this paper appeared as [13].

* Corresponding author.
E-mail addresses: ketang@ustc.edu.cn (K. Tang), sunnyboy@mail.ustc.edu.cn (M. Lin), F.L.Minku@cs.bham.ac.uk (F.L. Minku), x.yao@cs.bham.ac.uk (X. Yao).

0925-2312/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2008.09.022

may help (at least one) NN better adapt to the new data, and thereby enhance the overall performance of the ensemble.

Although Minku et al. presented a comprehensive analysis on NCL's potential for incremental learning, less effort has been made to design specific incremental learning algorithms based on NCL. In [20], two approaches based on NCL are introduced: fixed size NCL (FSNCL) and growing NCL (GNCL). In FSNCL, every time when a new data set is presented, the whole ensemble is used to learn the new data set. Its generalization will mostly lie on the last presented data set. In other words, the previous training affects little when applying the ensemble to testing data. Therefore, FSNCL may severely suffer from catastrophic forgetting, and the generalization performance may improve little or even decrease during the incremental learning process. On the other hand, in GNCL, every time when a new data set is presented, a new NN will be created and incorporated into the ensemble. Only this NN is trained using NCL to make it different from those previously trained NNs, while the existing NNs are kept unchanged. By this means, the generalization performance will improve observably and there will be less catastrophic forgetting. However, since only one NN is incrementally inserted in the ensemble, the GNCL does not take advantage of using multiple NNs to learn new data, and it may have low generalization performance in comparison with FSNCL. Furthermore, as more and more data sets are presented, the size of the ensemble will become too large and eventually be computationally intractable.

In this paper, we propose a selective NCL (SNCL) algorithm for incrementally training NN ensembles. The general idea of SNCL is straightforward. We fix the size of the ensemble to a predefined number. Every time a new data set is presented, the previous ensemble is cloned and trained using NCL. The new trained ensemble is then combined with the previous ensemble to form a larger ensemble. Then, a selection algorithm is applied to prune the combined ensemble to the predefined size.

The selection algorithm is the main difference between SNCL and the algorithms GNCL and FSNCL, proposed in [20]. The previous two approaches either preserve all NNs trained with previous data (GNCL), or discard all of them (FSNCL), and both preserve all the NNs trained with the new data set. Such strategies are more or less blind because whether an existing NN should be preserved or not may change over time and needs to be re-assessed from time to time. Furthermore, the newest NNs do not necessarily have better performance than previous NNs existing in the ensemble. So, they should not always replace the previous NNs. The selection process of SNCL puts both groups of NNs together and assesses them unbiasedly—decision is made solely based on the performance of the ensemble.

In our preliminary work, the general idea of SNCL has been shown to outperform FSNCL and GNCL in three aspects [13]. Firstly, SNCL does not discard the previously trained NNs directly, as the FSNCL does. So it will suffer less from catastrophic forgetting. At the same time, its generalization performance will no be worse than FSNCL, as we will demonstrate experimentally. Secondly, it outperforms the GNCL on accuracy since multiple NNs are introduced to learn the new data. Finally, its size does not increase with the new coming data, which has not been addressed in [20]. In this paper, we further investigate SNCL in the following ways:

- After combining the newly trained ensemble with the previous ensemble, an objective function needs to be designed for the selection algorithm to prune the combined ensemble. Unlike the preliminary work, we consider two objective functions here, which are motivated by different considerations.
- To compare SNCL with FSNCL and GNCL, additional experimental study is carried out on two additional data sets from the bioinformatics domain. The results further demonstrate the efficacy of SNCL.
- In addition to FSNCL and GNCL, SNCL is further compared to other existing incremental learning methods, such as Learn++, SONG, ARTMAP, and EILNN.

The rest of this paper is organized as follows. In Section 2, we briefly describe NCL. In Section 3, details of the SNCL algorithm are presented. Section 4 presents the experimental study and a comparison between SNCL and other incremental learning algorithms. Finally, we conclude this paper and discuss future works in Section 5.

2. Negative correlation learning

This section briefly describes the basic ideas of NCL. Proposed by Liu and Yao [16,17], NCL is a learning technique for training NN ensembles. Suppose that we have a training set denoted by

$$D = \{(\mathbf{x}_1, d_1), \ldots, (\mathbf{x}_N, d_N)\}$$

where $\mathbf{x} \in \mathbf{R}^p$, is the p-dimensional training pattern, d is the target output, and N is number of training patterns. NCL is defined for an NN ensemble of the form:

$$F(n) = \frac{1}{M}\sum_{i=1}^{M} F_i(n) \tag{1}$$

where M is the number of the individual NNs in the ensemble, $F_i(n)$ is the output of the ith NN on the nth training pattern, and $F(n)$ is the output of the ensemble on the nth training pattern.

NCL employs the standard back-propagation (BP) algorithm to train the individual NNs in parallel. The key to the success of NCL is the usage of a different error function. It is well known that for an ensemble to generalize well, the individual NNs of it should be both accurate and diverse. To encourage these, NCL uses the sum of the mean squared error (MSE) and a penalty term as the error function during the learning procedure. When the nth training pattern is presented, the ith NN is trained to minimize the error function:

$$E_i(n) = \tfrac{1}{2}(F_i(n) - d(n))^2 + \lambda p_i(n) \tag{2}$$

where λ is a positive parameter controlling the trade-off between the MSE (accuracy) and the penalty term (diversity):

$$p_i(n) = (F_i(n) - F(n))\sum_{j\neq i}(F_j(n) - F(n)) \tag{3}$$

It can be seen from Eq. (3) that the penalty term explicitly encourages the ith NN to be *negatively correlated* with the rest NNs in the ensemble. By this means, diversity among the individual NNs is achieved. It can be seen that with $\lambda = 0$ we would have an ensemble exactly equivalent to training a set of NNs independently of one another. When λ is increased, more and more emphasis is placed on seeking the negative correlation.

More recently, efforts have been made to carry out negative correlation learning in different ways. For example, Chan and Kasabov proposed the negative correlation learning via correlation-corrected data (NCCD), which reduces network communication bandwidth of NCL so that NCL can be effectively implemented on parallel computers [5]. As stated in the literature, many learning problem requires taking into account different properties of the final model, and it might be more appropriate to handle them separately rather than combine them together as a single objective [1,10]. Hence, Chandra [6] suggests to conduct NCL in a multi-objective optimization manner, and thus the potential difficulty of determining the optimal value for λ is avoided. Both [1,6] mainly concern about how to conduct NCL more effectively

in the context of batch learning, but have not yet been extended to the incremental learning domain. Besides, previous studies have shown that setting λ to 1 as a default works well for various problems. Therefore, we restrict our study on incremental learning to the original NCL algorithm [16,17] in this paper.

3. Incremental learning algorithm based on selective negative correlation learning

3.1. The general approach

Although NCL has shown very successful performance in the literature, most of it is achieved on batch learning problems, i.e., the complete training data set is available at once. The first attempt to extend NCL to incremental learning problems is made only in a very recent work [20]. As we discussed in the Introduction section, the method for adapting NCL to incremental learning task has not been fully exploited. In this section, we propose an incremental learning algorithm (SNCL) based on a selective algorithm.

Fig. 1 shows the architecture of SNCL. SNCL can be viewed as having two components.

The first component is straightforward. When a new set of training data is available, the previously trained ensemble is cloned, and we further train this cloned ensemble with the new training data, using the standard NCL. After that, the newly trained ensemble is combined with the old ensemble to form a "combined" ensemble. As we know, failing to preserve previously learned knowledge is likely to cause catastrophic forgetting in incremental learning. Since the previous data is no longer accessible once the corresponding training phase has been finished, we may only access the learned knowledge via previous trained NN/ensemble. In the first component of SNCL, the previously trained NNs not only influence training the new ensemble, but are also explicitly preserved. Therefore, we expect the first component of SNCL to contribute mainly to improve the generalization performance (via standard NCL) and to alleviate catastrophic forgetting.

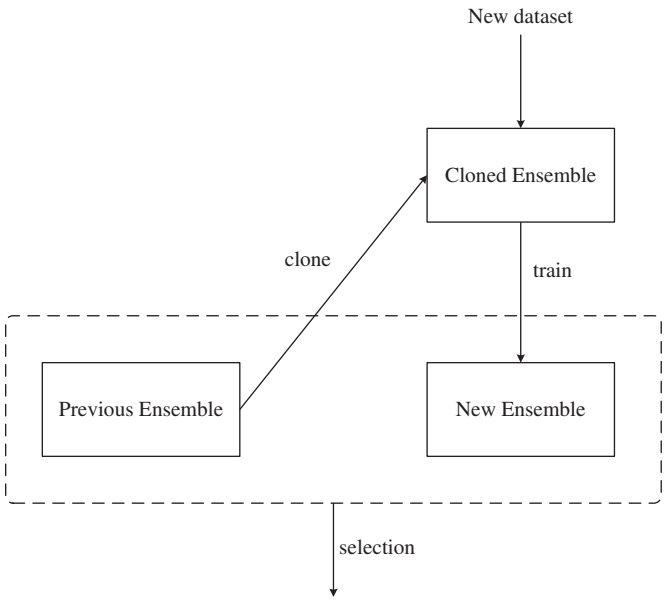

Fig. 1. Architecture of SNCL.

The second component of SNCL is motivated by the idea that not all the individual NNs are beneficial to the ensemble's generalization performance. Such concept is widely known in the batch learning literature. In [32], Yao and Liu showed that constructing an ensemble with only a part of the individual NNs may even lead to better generalization performance. This idea was then theoretically verified in [33]. In [33], Zhou et al. also suggested using a canonical genetic algorithm to select the useful individual NNs, namely, genetic algorithm based selective ensemble (GASEN). However, GASEN only considers batch learning tasks, and we are unaware of any work integrating similar paradigm into NCL. Alternatively, Liu et al. [18] employed a clustering algorithm (k-means) to select representative NNs from a population of NNs, which are trained by evolutionary algorithm and NCL. However, it should be noted that a good individual NN of an ensemble is not necessarily a representative one. Furthermore, the success of a clustering algorithm heavily relies on an appropriate measure of the similarity between individual NNs, which may not be a trivial work. Therefore, we adapt the GA-based approach to SNCL (i.e., use a GA-based selection algorithm to prune those useless individuals).

Compared to GASEN, SNCL adopts different objective functions for the GA and an additional repair operator, which will be described in details soon. It is worth noting that the selection component is more important for incremental learning than for batch learning. For an incremental learning task, when more and more training data sets are available, the first component of SNCL alone will eventually lead to an ensemble with computationally prohibitive size, and many individual NNs of such an ensemble might be redundant for achieving good generalization performance. With a selection component, we manage to always control the size of the ensemble at an acceptable level, while the generalization does not deteriorate.

To summarize, the whole process of SNCL can be illustrated as following:

(1) Let the training data sets be $\mathrm{Tr}\,D_1, \mathrm{Tr}\,D_2, \ldots, \mathrm{Tr}\,D_s$. Initialize an ensemble ens_1 with M NNs.
(2) Train ens_1 with $\mathrm{Tr}\,D_1$ using NCL.
(3) For i from 2 to s, repeat the following steps:
- The previous ensemble ens_{i-1} is cloned as ens_{cl}.
- Train ens_{cl} with $\mathrm{Tr}\,D_i$ using NCL.
- Combine ens_{i-1} and ens_{cl}.
- Apply the selection process to the combined ensemble to get ens_i with M NNs.

(4) Output the final ensemble ens_s.

3.2. The selection algorithm

This section describes the selection algorithm we used to get the M NNs for the final ensemble, given the combined ensemble of size $2M$. In general, selection of the NN individuals can be formulated as the following constrained optimization problem:

$$\begin{cases} \min\limits_{\mathbf{w}} & J(\mathbf{w}), \quad \mathbf{w} = [w_1, w_2, ..., w_{2M}] \\ \text{s.t.} & w_i \in \{0,1\}, \quad \forall i \\ & \sum\limits_i w_i = M \end{cases} \tag{4}$$

where $J(\mathbf{w})$ is a predefined objective function, and it should be carefully designed so that the final ensemble has good generalization performance. $\mathbf{w}$ is a $2M$-dimensional binary vector, $w_i = 1$ means that the ith NN is selected. As summarized in Fig. 2, a GA-based algorithm is employed to solve the above optimization problem.

(1) Initialize the algorithm with the predefined ensemble size M, the size of population *pop size*, the crossover probability p_c, the mutation probability p_m, and the fitness function $J(\mathbf{w})$.
(2) Randomly generate the initial population of $\mathbf{w}$.
(3) Repeat the following steps for a predefined number of generations:
   a. Evaluate fitness of the individuals of current population (here, an individual refers to a candidate $2M$-dimensional binary vector, not an individual of the ensemble).
   b. Use roulette selection to choose parent individuals, and produce off-spring via 1-point crossover (with probability p_c) and mutation (with probability p_m).
   c. Repair the population by greedy strategy:
   If $\sum_j w_j > M$, change one w_j from 1 to 0 which will lead to the smallest $J(\mathbf{w})$, and repeat this process until $\sum_j w_j = M$.
   If $\sum_j w_j < M$, change one w_j from 0 to 1 which will lead to the smallest $J(\mathbf{w})$, and repeat this process until $\sum_j w_j = M$.
   d. Replace the parent individuals with their off-springs.
(4) Output $\mathbf{w}^{opt}$ as the individual with the optimal fitness value. Those NNs corresponding to $\mathbf{w}_j^{opt} = 1$ are used to construct the final ensemble.

Fig. 2. steps of the selection algorithm.

From Fig. 2, we can observe that the GA-based algorithm is a hybridization of a canonical GA [7,19] and a repair operator. Though the canonical GA itself is capable of conducting a global search in the solution space, it may generate some infeasible solutions that do not satisfy the constraint $\sum_i w_i = M$. Hence, the repair operator is required to fix those solutions. Assume that we have an ensemble with more than M NNs (i.e., $\sum_i w_i > M$). At each iteration, the NN whose removal will lead to the smallest $J(\mathbf{w})$ is pruned. In case $\sum_i w_i < M$, NNs are sequentially included into the ensemble following a similar procedure.

The objective function $J(\mathbf{w})$ is the other key component of the selection algorithm. It must be carefully designed to guarantee good generalization performance. In this work, we consider two types of objective functions. The first one is the objective function of NCL itself, and the second one is based on the ensemble's classification error only.

3.2.1. The error function of NCL

One of the main reasons for NCL's success is the use of Eq. (2) as the novel error function. Hence, it is reasonable to expect Eq. (2) to work well for the selection procedure, too. Concretely, Eq. (2) calculates the error on a single training pattern, while we sum it up over all training patterns, and get

$$J(\mathbf{w}) = \sum_{i=1}^{2M}\sum_{n=1}^{N} w_i E_i(n) \tag{5}$$

According to Eqs. (2) and (3), we get

$$J(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{2M}\sum_{n=1}^{N} w_i (F_i(n) - d(n))^2 - \lambda \sum_{i=1}^{2M}\sum_{n=1}^{N} w_i (F(n) - F_i(n))^2 \tag{6}$$

Let an $N \times 2M$ matrix $\mathbf{A}$ denote the outputs of all NNs on all patterns:

$$\mathbf{A} = \begin{pmatrix} (F_1(1) - d(1))^2 & (F_2(1) - d(1))^2 & \dots & (F_{2M}(1) - d(1))^2 \\ (F_1(2) - d(2))^2 & (F_2(2) - d(2))^2 & \dots & (F_{2M}(2) - d(2))^2 \\ \vdots & \vdots & \ddots & \vdots \\ (F_1(N) - d(N))^2 & (F_2(N) - d(N))^2 & \dots & (F_{2M}(N) - d(N))^2 \end{pmatrix} \tag{7}$$

then

$$\frac{1}{2}\sum_{i=1}^{2M}\sum_{n=1}^{N} w_i (F_i(n) - d(n))^2 = \frac{1}{2}\mathbf{e}^T\mathbf{A}\mathbf{w} \tag{8}$$

where $\mathbf{e}$ is a N-dimensional unit vector.

We denote

$$\vec{F}(n) = (F_1(n)\ F_2(n)\ \dots\ F_{2M}(n)) \tag{9}$$

and

$$\mathbf{B} = \begin{pmatrix} (\frac{1}{M}\vec{F}(1)\mathbf{w} - F_1(1))^2 & (\frac{1}{M}\vec{F}(1)\mathbf{w} - F_2(1))^2 & \dots & (\frac{1}{M}\vec{F}(1)\mathbf{w} - F_{2M}(1))^2 \\ (\frac{1}{M}\vec{F}(2)\mathbf{w} - F_1(2))^2 & (\frac{1}{M}\vec{F}(2)\mathbf{w} - F_2(2))^2 & \dots & (\frac{1}{M}\vec{F}(2)\mathbf{w} - F_{2M}(2))^2 \\ \vdots & \vdots & \ddots & \vdots \\ (\frac{1}{M}\vec{F}(N)\mathbf{w} - F_1(N))^2 & (\frac{1}{M}\vec{F}(N)\mathbf{w} - F_2(N))^2 & \dots & (\frac{1}{M}\vec{F}(N)\mathbf{w} - F_{2M}(N))^2 \end{pmatrix} \tag{10}$$

then

$$\lambda \sum_{i\in L}\sum_{n=1}^{N} (F(n) - F_i(n))^2 = \lambda \mathbf{e}^T\mathbf{B}\mathbf{w} \tag{11}$$

Hence,

$$J(\mathbf{w}) = \tfrac{1}{2}\mathbf{e}^T\mathbf{A}\mathbf{w} - \lambda\mathbf{e}^T\mathbf{B}\mathbf{w} \tag{12}$$

Given the above formulation, the algorithm presented in Fig. 2 is used to seek the optimum $\mathbf{w}$.

3.2.2. Using the training error as the objective function

Eq. (11) involves several matrix computations and thereby will be time consuming for large scale applications. This is partially due to the calculation of the second term, which explicitly requires the individuals to be diverse. Though diversity among individual NNs is an important reason for the success of NN ensemble, many previous works also showed that it might not be always indispensable [12,30]. Therefore, it will be interesting to ask whether the selection process can be carried out only based on the classification accuracy, without calculating the correlations between individual NNs. By this means, we aim at achieving good generalization performance with less computational cost. Concretely, we can directly evaluate a $\mathbf{w}$'s fitness using the corresponding ensemble's training accuracy. If multiple ensembles have the same training accuracy, their mean squared errors on the CORRECTLY classified training patterns, denoted by ECP, are calculated. The ECP can be viewed as the "confidence" of an ensemble on its classification results. The smaller the MSE, the more confident the ensemble is. Hence, the one with smaller ECP is preferred.

The above scheme can be easily substituted the selection algorithm with little effort, as described below:

In Steps 3a and 3b, we set the fitness of the individuals as their corresponding training accuracy (i.e., one minus training error), and conduct roulette selection.

In Step 3c, conduct the repair process based on training accuracy. If more than one w_i's result in the same largest accuracy, take the one corresponding to the smallest ECP.

In Step 4, choose the $\mathbf{w}$ with the highest training accuracy. In case multiple $\mathbf{w}$'s take the largest accuracy, the one corresponding to the smallest ECP is chosen.

It should be noted that the above scheme is totally based on the CURRENT training data. Hence, it might bias more to the newly trained NNs, and lead to catastrophic forgetting. As to be shown by experiments, this is what we should pay for the alleviation of computational cost.

4. Experiments

4.1. Experimental setup

To evaluate the proposed method, we experimentally compare SNCL to FSNCL and GNCL. The difference between the two objective functions was also studied. The experiments were carried out on five data sets. Three of them are the main benchmark classification data sets used by Minku et al. in [20], namely the Letter, Vehicle, and Optical Digits data sets from the UCI machine learning repository [21]. The other two, namely the odorant binding proteins (OBP) data set [25] and the structurally conserved residues (SCR) data set [26] were investigated in our previous study. The OBP data set was generated based on a number of odorant binding proteins are obtained from GenDiS [23] and Pfam [29] databases, the task is to differentiate the odorant binding proteins from those non-odorant binding domains. The SCR data set was generated on the basis of protein sequences obtained from MegaMotifBase database [24]. The task is to identify structurally conserved residues on the sequences. Table 1 presents a summary of the data sets. Every data set was averagely divided into several subsets, one for testing and the others for step-by-step incremental learning. After every incremental step, the ensemble is applied to the testing data and testing accuracy is calculated. All NNs were encoded using a 1-of-m output representation for m classes. The output with the highest activation designated the class. We set M to be 10 for both FSNCL and SNCL, and for GNCL, one NN is added for each new training subset. To make fair comparison, all the compared algorithms share common learning rate (0.1) and λ (1) on all data sets. Table 2 presents the other parameters for NCL. As for the GA used in SNCL, we set $pop\_size = 50$, $pc = 0.2$, and $pm = 0.01$, and the number of generations is 50.

Table 1
Data sets.

Data sets	Inputs	Classes	Training patterns	Incremental step	Testing patterns
Letter	16	26	18,000	9	2000
Vehicle	18	4	630	3	216
Optical Digits	64	10	1200	6	4420
OBP	10	2	2031	5	1016
SCR	10	2	6042	6	11,639

Table 2
Parameters for NCL.

Data sets	Hidden nodes	Epochs
Letter	20	250
Vehicle	10	1000
Optical Digits	10	100
OBP	20	500
SCR	5	200

4.2. Experimental results

The experimental results on the five data sets are shown in Tables 3–5, where SNCL-NCL denotes that the error function of NCL is used in the selection algorithm, and SNCL-TE denotes that the training error is used for selection. All the results are the average of 30 executions. Specifically, Table 3 presents the average testing accuracies obtained using different algorithms, standard deviations are also provided. Table 4 presents the average decrement (degradation) of training accuracies on all training data sets. The degradation on training data sets is a criterion to judge whether an algorithm suffers from catastrophic forgetting. Finally, training time of the algorithms is presented in Table 5. Next, we compare different algorithms from the perspective of generalization performance, degradation, and computational cost, respectively. In all the tables, bold letters were used to identify the best approaches on each data set, when there was statistically significant difference.

4.2.1. Generalization performance

ANOVA tests were used to verify whether there are statistically significant differences in testing accuracy (i.e., generalization performance) using different incremental learning algorithms. The results of a multiple comparison test based on the ANOVA

Table 5
Training time (in seconds) of four NCL-based incremental learning algorithms.

	Letter	Vehicle	Optical Digits	OBP	SCR
SNCL-NCL	18,937	274	396	608	604
SNCL-TE	6867	240	192	534	332
FSNCL	5520	210	91	485	183
GNCL	**1116**	**29**	**17**	**80**	**33**

Table 3
Testing accuracy of four NCL-based incremental learning algorithm.

	Letter	Vehicle	Optical Digits	OBP	SCR
SNCL-NCL	0.8712 ± 0.0060	0.8162 ± 0.0116	$\mathbf{0.9337 \pm 0.0027}$	$\mathbf{0.9360 \pm 0.0053}$	0.9213 ± 0.0022
SNCL-TE	0.8780 ± 0.0059	0.8228 ± 0.0155	$\mathbf{0.9321 \pm 0.0072}$	$\mathbf{0.9336 \pm 0.0049}$	0.9201 ± 0.0024
FSNCL	$\mathbf{0.8845 \pm 0.0043}$	0.8144 ± 0.0114	0.9273 ± 0.0025	0.9282 ± 0.0048	0.9209 ± 0.0013
GNCL	0.7412 ± 0.0074	0.8213 ± 0.0131	0.9252 ± 0.0037	$\mathbf{0.9347 \pm 0.0025}$	0.9206 ± 0.0012

Table 4
Degradation of four NCL-based incremental learning algorithm.

	Letter	Vehicle	Optical Digits	OBP	SCR
SNCL-NCL	$\mathbf{0.0068 \pm 0.0037}$	0.0784 ± 0.0087	0.0296 ± 0.0034	0.0163 ± 0.0048	0.0106 ± 0.0021
SNCL-TE	0.0156 ± 0.0032	0.0734 ± 0.0099	0.0306 ± 0.0065	0.0216 ± 0.0049	0.0155 ± 0.0016
FSNCL	0.0115 ± 0.0020	0.0856 ± 0.0074	0.0361 ± 0.0020	0.0198 ± 0.0073	0.0128 ± 0.0015
GNCL	0.0122 ± 0.0055	$\mathbf{0.0254 \pm 0.0073}$	$\mathbf{0.0173 \pm 0.0060}$	$\mathbf{0.0025 \pm 0.0031}$	$\mathbf{0.0053 \pm 0.0010}$

tests are shown in Fig. 3. We can observe that the two SNCL approaches both outperform GNCL on the Letter and Optical Digits data sets, while no significant differences can be observed on the remaining three data sets. Both SNCL-NCL and SNCL-TE outperforms FSNCL on two data sets (optical digits and OBP), but is inferior to FSNCL on the Letter data set. Comparing the two SNCL algorithms, SNCL-NCL is inferior to SNCL-TE on the Letter data set, and they are comparable on the other four data sets.

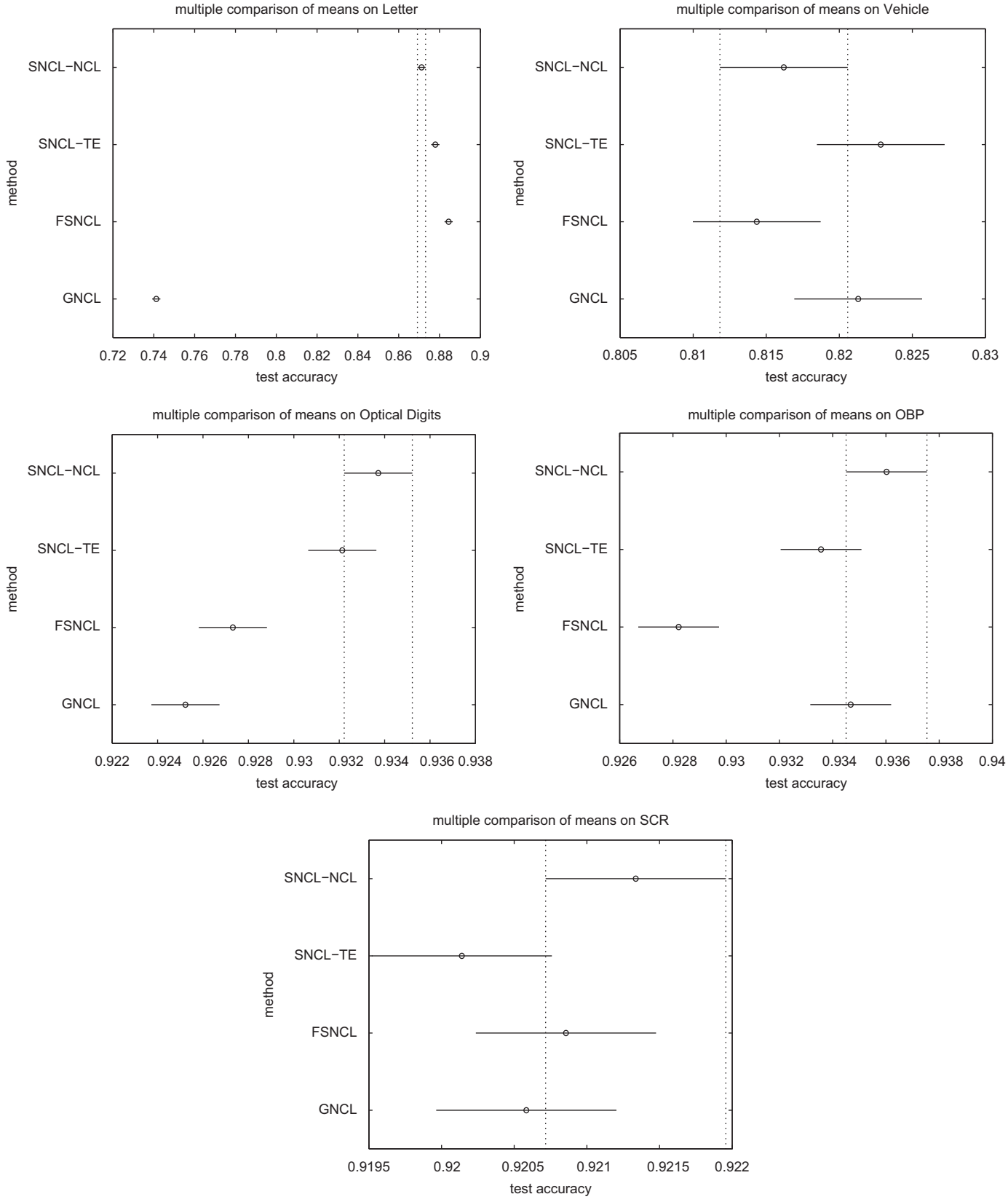

Fig. 3. Multiple comparison tests of the testing accuracy of four NCL-based incremental learning algorithms.

Therefore, we can conclude that the SNCL generally outperform both FSNCL and GNCL in terms of generalization performance, while the usage of different objective functions does not lead to significant difference.

4.2.2. Degradation

The average degradations of each algorithm on the five data sets are presented in Table 4. The larger the degradation, the more likely that an algorithm suffers from catastrophic forgetting.

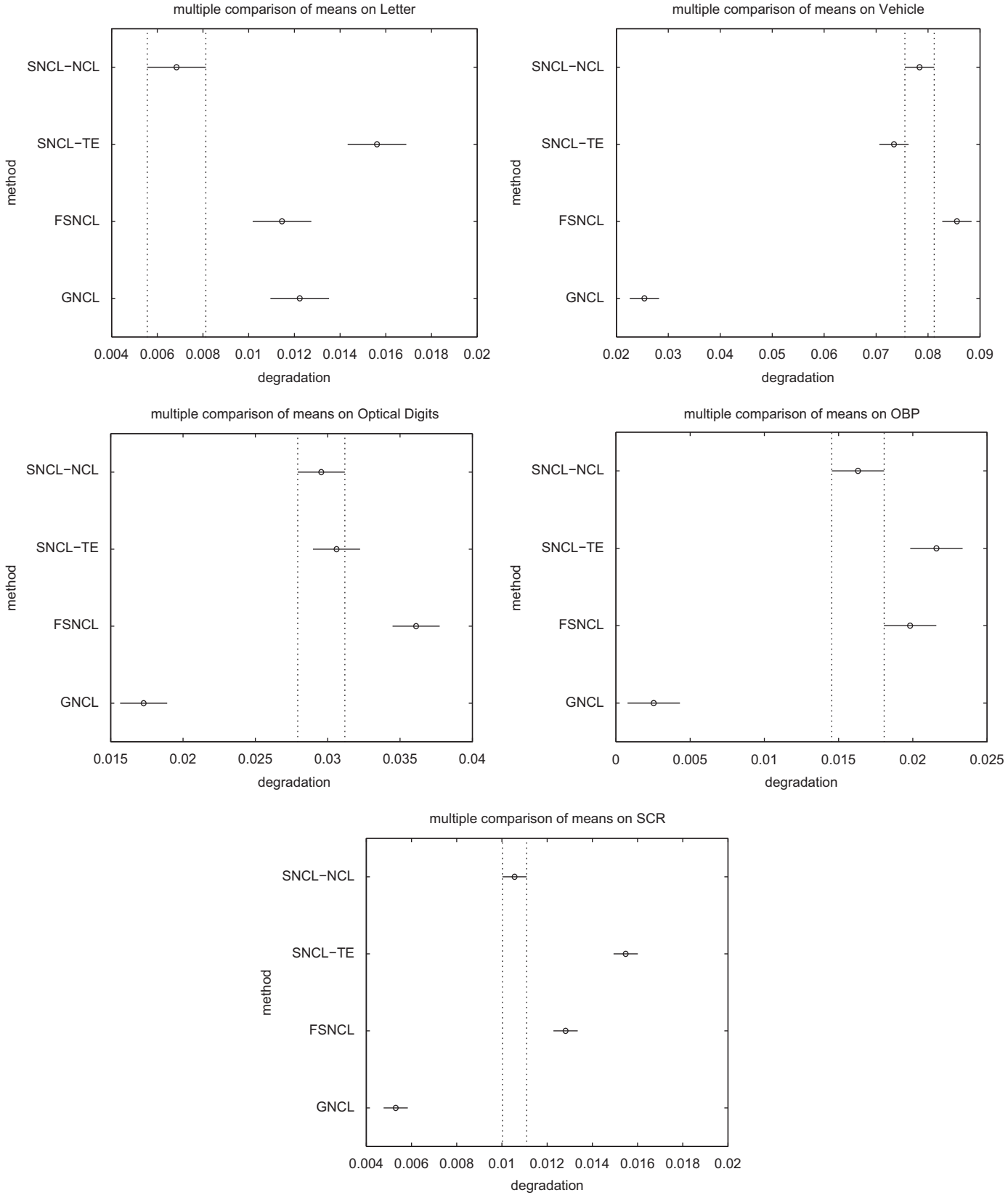

Fig. 4. Multiple comparison tests of the degradation of four NCL-based incremental learning algorithms.

ANOVA tests were also used to identify statistically significant differences. The results are shown in Fig. 4. It can be observed that both SNCL-NCL and SNCL-TE always suffered more catastrophic forgetting than GNCL in all cases with the exception of the Letter data set. In comparison to FSNCL, SNCL-NCL is significantly better on all the five data sets. SNCL-TE won on the Vehicle and Optical Digits data sets, while lose on the Letter and SCR data sets. Further, as we expected, SNCL-TE suffered more catastrophic forgetting than SNCL-NCL on three data sets, while no statistically significant differences were observed on the other two data sets.

4.2.3. Computational time

The training time of all the approaches is presented in Table 5. Since the GNCL trains a single network at each incremental step, it is definitely faster than the other methods. The main difference between FSNCL and SNCL is that SNCL incorporates the selection process. From the table, we can observe that such a selection process is quite time consuming, especially for SNCL-NCL. For example, the selection process even took much longer time than the training process itself. In comparison, the selection process of SNCL-TE is much less costly. Regarding the testing time, FSNCL, SNCL-NCL and SNCL-TE take the exactly same time, because they consist of same number of individual NNs. On the other hand, the testing time of GNCL will increase when new incremental training subsets have been presented, and eventually be more costly than the other approaches.

4.3. Comparison to other approaches

Since several existing approaches have shown good performance on incremental learning. It is interesting to further compare SNCL to them. The comparison between SNCL and Learn++ algorithms on Vehicle and Optical Digits data sets is presented in Table 6. The results of Learn++ were directly taken from the original publication [22,20]. From Table 6, it can be observed that SNCL achieved lower testing accuracy than Learn++ on the Vehicle data set, while performed better on the Optical Digits data set. In comparison, SNCL generally suffers more from catastrophic forgetting, which seems to suggest that Learn++ is better than SNCL on this aspect. However, having a closer look at the training accuracy (i.e., the average accuracy of the final ensemble on all previous training data subsets), we may find that SNCL still maintains significant higher training accuracy than Learn++. Therefore, from the perspective of preserving high accuracy on previous training data, SNCL can be said to outperform Learn++.

Table 6
Comparison between two SNCL approaches and Learn++.

	Vehicle	Optical Digits
Testing accuracy		
SNCL-NCL	0.8162	**0.9337**
SNCL-TE	0.8228	0.9321
Learn++	**0.8300**	0.9270
Training accuracy		
SNCL-NCL	0.8645	**0.9651**
SNCL-TE	**0.8821**	**0.9651**
Learn++	0.8267	0.9400
Degradation		
SNCL-NCL	0.0784	0.0296
SNCL-TE	0.0734	0.0306
Learn++	**0.0733**	**0.0033**

In [20], it was shown that SONG algorithm is comparable to FSNCL and GNCL in terms of generalization performance. Since SNCL shows better generalization performance than the previous NCL-based approaches, we conservatively conclude that SNCL is also comparable to SONG. In addition to Learn++ and SONG, there are also some other incremental learning approaches that are worthy of mention, such as the ARTMAP [3,4], EFuNN [11] and EILNN [28]. Unfortunately, experimental results of the former two approaches are not available in the literature, and EILNN was only verified on the Optical Digits data set with a different experimental protocol (3823 patterns were used for training and validation, and 1797 patterns for testing). Hence, we only compare SNCL to those methods qualitatively here. Both ARTMAP and EFuNN create new clusters if the new data is sufficiently different from the previous data. For this reason, both of them highly rely on precise measurement of the difference between patterns. This is usually a non-trivial task and is sensitive to noises and the order of presentation of the data. On the other hand, SNCL trains new NNs with the whole new training data set, and thereby does not suffer from such problem. EILNN makes use of evolving neural network to handle incremental learning tasks. An evolutionary algorithm is used to adapt the NNs' parameters, such as the learning rates, initial weight distributions and error tolerance. The main problem with EILNN is that the evolution process requires the presentation of all training data sets, though each NN is trained with one data set at a time (i.e., incrementally). Hence, EILNN does require access to previous training data, while SNCL does not.

5. Conclusions and future work

In this paper, a selective negative correlation learning approach, which is specifically designed for incremental learning tasks, is proposed. Every time when a new data set is presented, the previously trained ensemble is cloned, and trained using NCL. After training, the new ensemble is combined with the previously trained ensemble, and a GA-based selection algorithm is utilized to prune the ensemble to a predefined size. Though the general idea of SNCL has been briefly proposed earlier in [13], this paper investigates the SNCL in more details. Specifically, two types of objective functions were investigated within the proposed SNCL framework: NCL error function and the training error. The corresponding algorithms are named SNCL-NCL and SNCL-TE and were experimentally compared to two existing NCL-based incremental learning approaches, namely FSNCL and GNCL, on three UCI data sets and two real-world bioinformatics data sets. Comparisons between SNCL and other well known incremental learning approaches are also presented.

Experimental results show that SNCL is capable of achieving better generalization performance than the other NCL-based approaches and the Learn++ algorithm. Besides, SNCL suffers less from catastrophic forgetting than FSNCL. Finally, SNCL always controls the size of ensemble at an acceptable level, while the generalization does not deteriorate. Such a property is very important for applications with huge amount of training data. When comparing the two SNCL algorithms, we found that SNCL-TE suffers more from catastrophic forgetting, while SNCL-NCL is much more time consuming.

The main drawback of SNCL is that it involves much longer training time than FSNCL and GNCL. The reason is that SNCL adopts a GA-based method in the selection phase, which is very time consuming. Hence, a potential direction for future research is to consider efficient selection algorithm for SNCL. Furthermore, we have not evaluated SNCL from the aspect of accommodating new classes that may be introduced with new data, neither do we

design specific schemes for that. We will also investigate this issue in our future work.

Acknowledgements

This work is partially supported by two National Natural Science Foundation of China Grants (nos. 60802036 and U0835002) and the Fund for Foreign Scholars in University Research and Teaching Programs (Grant no. B07033).

References

[1] H.A. Abbass, An evolutionary artificial neural networks approach for breast cancer diagnosis, Artificial Intelligence in Medicine 25 (2002) 265–281.
[2] G. Brown, Diversity in neural network ensembles, Ph.D. Thesis, School of Computer Science, University of Birmingham, 2004.
[3] G.A. Carpenter, S. Grossberg, J.H. Reynolds, ARTMAP: supervised real-time learning and classification of nonstationary data by a self organizing neural network, Neural Networks 4 (1991) 565–588.
[4] G.A. Carpenter, S. Grossberg, N. Markuzon, J.H. Reynolds, Fuzzy ARTMAP: a neural network architecture for incremental supervised learning of analog multidimensional maps, IEEE Transactions on Neural Networks 3 (1992) 698–713.
[5] Z.S.H. Chan, N. Kasabov, A preliminary study on negative correlation learning via correlation-corrected data (NCCD), Neural Processing Letters 21 (2005) 207–214.
[6] A. Chandra, X. Yao, Ensemble learning using multi-objective evolutionary algorithms, Journal of Mathematical Modelling and Algorithms 5 (2006) 417–445.
[7] K. De Jong, Evolutionary Computation: A Unified Approach, MIT Press, Cambridge, 2006.
[8] C. Giraud-Carrier, A note on the utility of incremental learning, AI Communications 13 (2000) 215–223.
[9] H. Inoue, H. Narihisa, Self-organizing neural grove and its applications, in: Proceedings of the 2005 International Joint Conference on Neural Networks (IJCNN'05), Montreal, Canada, 2005, pp. 1205–1210.
[10] Y. Jin, B. Sendhoff, Pareto-based multi-objective machine learning: an overview and case studies, IEEE Transactions on Systems, Man, and Cybernetics, Part C—Applications and Reviews 38 (2008) 397–415.
[11] N. Kasabov, Evolving fuzzy neural networks for supervised/unsupervised online knowledge-based learning, IEEE Transactions on Systems, Man, and Cybernetics, Part B—Cybernetics 31 (2001) 902–918.
[12] L. Kuncheva, C. Whitaker, Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy, Machine Learning 51 (2003) 181–207.
[13] M. Lin, K. Tang, X. Yao, Selective negative correlation learning algorithm for incremental learning, in: Proceedings of the 2008 International Joint Conference on Neural Networks (IJCNN'08), Hongkong, China, 2008, pp. 2526–2531.
[14] Y. Liu, Negative correlation learning and evolutionary neural network ensembles, Ph.D. Thesis, University College, The University of New South Wales, Australian Defence Force Academy, 1998.
[15] Y. Liu, X. Yao, Negatively correlated neural networks can produce best ensembles, Australian Journal of Intelligent Information Processing Systems 4 (1997) 176–185.
[16] Y. Liu, X. Yao, Ensemble learning via negative correlation, Neural Networks 12 (1999) 1399–1404.
[17] Y. Liu, X. Yao, Simultaneous training of negatively correlated neural networks in an Ensemble, IEEE Transactions on Systems, Man, and Cybernetics, Part B—Cybernetics 29 (1999) 716–725.
[18] Y. Liu, X. Yao, T. Higuchi, Evolutionary ensembles with negative correlation learning, IEEE Transactions on Evolutionary Computation 4 (2000) 380–387.
[19] Z. Michalewicz, Genetic Algorithms + Data Structure = Evolution Programs, Springer, Berlin, 1996.
[20] F.L. Minku, H. Inoue, X. Yao, Negative correlation learning in incremental learning, Natural Computing Journal (Special Issue on Nature-inspired Learning and Adaptive Systems), (2008), in press, doi:10.1007/s11047-007-9063-7.
[21] D. Newman, S. Hettich, C. Blake, C. Merz, UCI repository of machine learning databases, 1998. URL ⟨http://www.ics.uci.edu/~mlearn/MLRepository.html⟩.
[22] R. Polikar, L. Udpa, S.S. Udpa, V. Honavar, Learn++: an incremental learning algorithm for supervised neural networks, IEEE Transactions on Systems, Man, and Cybernetics Part C—Applications and Reviews 31 (2001) 497–508.
[23] G. Pugalenthi, A. Bhaduri, R. Sowdhamini, GenDiS: genomic distribution of protein structural domain superfamilies, Nucleic Acids Research 33 (2005) D252–D255.
[24] G. Pugalenthi, P.N. Suganthan, R. Sowdhamini, S. Chakrabarti, MegaMotifBase: a database of structural motifs in protein families and superfamilies, Nucleic Acid Research 36 (2008) D218–D221.
[25] G. Pugalenthi, K. Tang, P.N. Suganthan, G. Archunan, R. Sowdhamini, A machine learning approach for the identification of odorant binding proteins from sequence-derived properties, BMC-Bioinformatics 8 (2007) 351.
[26] G. Pugalenthi, Ke Tang, P.N. Suganthan, S. Chakrabarti, Identification of structurally conserved residues of proteins in absence of structural homologs using neural network ensemble, Bioinformatics 25 (2) (2009) 204–210.
[27] R.E. Schapire, The boosting approach to machine learning: an overview, in: D.D. Denison, M.H. Hansen, C. Holmes, B. Mallick, B. Yu (Eds.), Nonlinear Estimation and Classification, Springer, Berlin, 2003.
[28] T. Seipone, J.A. Bullinaria, Evolving improved incremental learning schemes for neural network systems, in: Proceedings of the 2005 IEEE Congress on Evolutionary Computation (CEC'05), Edinburgh, UK, vol. 3, 2005, pp. 2002–2009.
[29] E.L. Sonnhammer, S.R. Eddy, R. Durbin, Pfam: a comprehensive database of protein domain families based on seed alignments, Proteins 28 (1997) 405–420.
[30] E.K. Tang, P.N. Suganthan, X. Yao, An analysis of diversity measures, Machine Learning 65 (2006) 247–271.
[31] X. Yao, M. Fischer, G. Brown, Neural network ensembles and their application to traffic flow prediction in telecommunications networks, in: Proceedings of the 2001 International Joint Conference on Neural Networks (IJCNN'01), Washington, DC, vol. 1, 2001, pp. 693–698.
[32] X. Yao, Y. Liu, Making use of population information in evolutionary artificial neural networks, IEEE Transactions on Systems, Man and Cybernetics, Part B—Cybernetics 28 (1998) 417–425.
[33] Z. Zhou, J. Wu, W. Tang, Ensembling neural networks: many could be better than all, Artificial Intelligence 137 (2002) 239–263.

Ke Tang received his B.Eng. degree from the Department of Control Science and Engineering of Huazhong University of Science and Technology, Wuhan, PR China in 2002, and his Ph.D. degree from the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, in 2007. Currently, he is an associate professor at the Department of Computer Science and Technology, University of Science and Technology of China (USTC), Hefei, Anhui. He is also the Chair of IEEE Task For Large Scale Global Optimization. His major research interests include machine learning, pattern analysis, evolutionary computation, data mining, and real-world applications.

Minlong Lin received the B.Sc. degree from the Department of Computer Science and Technology, University of Science and Technology of China (USTC), Hefei, Anhui, China, in 2007. He is currently working towards his Ph.D. degree at the Department of Computer Science and Technology, USTC. His research interests include neural network, machine learning, and pattern recognition.

Fernanda Li Minku received the B.Sc. degree in Computer Science from the Federal University of Paraná Curitiba, Brazil, in 2003, and the M.Sc. degree in Computer Science from the Federal University of Pernambuco, Recife, Brazil, in 2006. She is currently working towards the Ph.D. degree in Computer Science at the University of Birmingham, UK. Her research interests are neural network ensembles, on-line learning, evolutionary computation and co-evolution.

Xin Yao received the B.Sc. degree from the University of Science and Technology of China (USTC), Hefei, Anhui, in 1982, the M.Sc. degree from the North China Institute of Computing Technology, Beijing, in 1985, and the Ph.D. degree from USTC in 1990. He was an associate lecturer and lecturer from 1985 to 1990 at USTC, while working towards his Ph.D. on simulated annealing and evolutionary algorithms. He took up a postdoctoral fellowship in the Computer Sciences Laboratory, Australian National University, Canberra, in 1990, and continued his work on simulated annealing and evolutionary algorithms. He joined the Knowledge-Based Systems Group, CSIRO Division of Building,

Construction and Engineering, Melbourne, in 1991, working primarily on an industrial project on automatic inspection of sewage pipes. He returned to Canberra in 1992 to take up a lectureship in the School of Computer Science, UNSW@ADFA, where he was later promoted to a senior lecturer and associate professor. Attracted by the English weather, he moved to the University of Birmingham, UK, as a professor of computer science in 1999. Currently, he is the director of the Centre of Excellence for Research in Computational Intelligence and Applications and a Changjiang chair (visiting) professor (Cheung Kong Scholar) at the University of Science and Technology of China, Hefei. He is the editor-in-chief of the IEEE Transactions on Evolutionary Computation, an associate editor or editorial board member of several other journals, and the editor of the World Scientific Book Series on Advances in Natural Computation. He has given more than 50 invited keynote and plenary speeches at conferences and workshops worldwide. His major research interests include evolutionary artificial neural networks, automatic modularization of machine learning systems, evolutionary optimization, constraint handling techniques, computational time complexity of evolutionary algorithms, coevolution, iterated prisoner's dilemma, data mining, and real-world applications. He has more than 250 refereed publications. He was awarded the President's Award for Outstanding Thesis by the Chinese Academy of Sciences for his Ph.D. work on simulated annealing and evolutionary algorithms in 1989. He won the 2001 IEEE Donald G. Fink Prize Paper Award for his work on evolutionary artificial neural networks.