Neurocomputing 73 (2009) 260–273

Contents lists available at ScienceDirect

Neurocomputing

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

Progressive interactive training: A sequential neural network ensemble learning method

M.A.H. Akhand [a], Md. Monirul Islam [b], K. Murase [b,*]

[a] Department of Computer Science and Engineering, Khulna University of Engineering and Technology (KUET), Khulna 9203, Bangladesh
[b] Department of Human and Artificial Intelligence Systems, Graduate School of Engineering, University of Fukui, 3-9-1 Bunkyo, Fukui 910-8507, Japan

ARTICLE INFO

Article history:
Received 14 February 2008
Received in revised form
2 September 2009
Accepted 5 September 2009
Communicated by G. Thimm
Available online 25 September 2009

Keywords:
Neural network ensemble
Indirect communication
Negative correlation learning
Bagging and boosting

ABSTRACT

This paper introduces a progressive interactive training scheme (PITS) for neural network (NN) ensembles. The scheme trains NNs in an ensemble one by one in a sequential fashion where the outputs of all previously trained NNs are stored and updated in a common location, called information center (IC). The communication among NNs is maintained indirectly through IC, reducing interaction among NNs. In this study, PITS is formulated as a derivative of simultaneous interactive training, negative correlation learning. The effectiveness of PITS is evaluated on a suite of 20 benchmark classification problems. The experimental results show that the proposed training scheme can improve the performance of ensembles. Furthermore, the PITS is incorporated with two very popular ensemble training methods, bagging and boosting. It is found that the performance of bagging and boosting algorithms can be improved by incorporating PITS with their training processes.

1. Introduction

Neural network (NN) ensembles, a combination of several NNs, are widely used to improve the generalization performance of NNs. Component NNs in an ensemble are trained for the same or different tasks, and their outputs are combined in a collaborative or competitive fashion to produce the output of the ensemble. No improvement can be obtained when NNs produce similar outputs because the failure of one NN cannot be compensated by the others. Both theoretical and empirical studies have revealed that improved generalization performance can be obtained when NNs maintain proper diversity in producing their outputs [1–3].

Considerable work has been done to determine the effective ways for constructing diverse NNs so that the benefit of combining several NNs can be achieved. There are many ways, such as using different training sets, architectures and learning methods, one can adopt to construct diverse NNs. It is argued that training NNs using different data is likely to maintain more diversity than other approaches [3–5]. This is because it is the training data on which a network is trained that determines the function it approximates. The most popular algorithms that explicitly or implicitly use different training data for different NNs in an ensemble are the bagging [4], boosting [5], random subspace method [25] and negative correlation learning (NCL) [11].

Both bagging and boosting algorithms explicitly manipulate the original training data to create a separate training set for each NN in an ensemble. Bagging creates the separate training set by forming bootstrap replicas of the original training data, while boosting creates it by the same method but with adaptation [6–8]. An NN is trained independently and sequentially by bagging and boosting, respectively, without any training time interaction with other NNs. Since training sets created by bagging and boosting contain some common information (i.e., training examples), NNs produce by the two algorithms are not necessarily negatively correlated owing to the absence of training time interaction among them [11,20].

Like bagging and boosting, NCL [10–12] does not create separate training sets explicitly for NNs in an ensemble. The NCL rather uses a correlation penalty term in the error function of the NNs by which networks can maintain training time interaction. The training method used in NCL is simultaneous where all NNs in the ensemble are trained on the same original training data at the same time. Since NCL provides training time interaction among NNs, it can produce negatively correlated NNs for the ensemble. The main problem with simultaneous training is that NNs in the ensemble may engage in competition [9]. This is because all NNs are trained on the same training data. Furthermore, in NCL, the number of NNs in the ensemble needs to be predefined and the cost of training time interaction is high.

A new scheme, called DECORATE algorithm [13], recently has been proposed that sequentially trains a relatively large number of NNs to select several NNs for constructing an ensemble. The

* Corresponding author. Tel.: +81 776 27 8774; fax: +81 776 27 8420.
E-mail address: murase@synaspe.his.fukui-u.ac.jp (K. Murase).

0925-2312/$ - see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2009.09.001

algorithm uses a separate training set, which is the union of the original training data and randomly created artificial data, for training each NN in the ensemble. The aim of using artificial data is to create diverse NNs for the ensemble. However, the problem with DECORATE is that the diversity among NNs is solely dependent on artificial data. Since the algorithm does not facilitate training time interaction, NNs produced by it are not necessarily negatively correlated.

In the present study, we introduce a progressive interactive training scheme (PITS) that sequentially trains NNs in an ensemble. The PITS uses an information center (IC) for storing the output of already trained NNs. An NN gets information about the task accomplished by the previously trained NN(s) through IC. The idea of using indirect communication is conceived from the artificial ant colony system (ACS) which mimics communication among biological ants via pheromone [14,15]. An individual ant decides its travelling path based on existing pheromone on the trail and also it deposits pheromone on its travelling path. The selection of a travelling path based on the pheromone trail is found more efficient with respect to direct communication with other ants [14,15].

The rest of this paper is organized as follows. Section 2 describes PITS in detail and gives the motivations behind the use of indirect communication. Section 3 explains implementation of PITS into the bagging and boosting. Section 4 first presents the experimental results of PITS along with back-propagation (BP) and NCL; and then compares performance of PITS with those of AdaBoost (a popular variant of boosting) and DECORATE. This section also contains experimental analyses between NCL and PITS; and evaluates performance of bagging and AdaBoost inducing PITS in their training processes. Finally, Section 5 concludes this paper with some remarks and suggestions for future directions.

2. Progressive interactive training scheme (PITS) for ensemble

Since we want to develop PITS from NCL [11], this section first describes NCL, so as to make the paper self contained, and then explains the formulation of PITS. The NCL algorithm, which is widely used for training NNs in ensembles, is an extension of the BP algorithm [19]. The error, $e_i(n)$, of a network i for the n-th training pattern in BP is

$$e_i(n) = \tfrac{1}{2}(f_i(n) - d(n))^2, \tag{1}$$

where $f_i(n)$ and $d(n)$ are the actual and desired outputs for the n-th training pattern, respectively. The problem with this error function is that an NN in the ensemble cannot communicate with other NNs during training. Thus the NNs may produce positively correlated output, when an algorithm trains the NNs on the same training data. It is known that such positive correlations among NNs are not suitable for the performance of ensembles [11,12].

The NCL algorithm, therefore, introduces a penalty term in the error function to establish training time interaction among NNs in the ensemble. According to [3,10], the error of the i-th NN in the ensemble for the n-th training pattern is

$$\begin{aligned} e_i(n) &= \frac{1}{2}(f_i(n) - d(n))^2 + \lambda(f_i(n) - f(n))\sum_{j\neq i}(f_j(n) - f(n)), \\ &= \frac{1}{2}(f_i(n) - d(n))^2 - \lambda(f_i(n) - f(n))^2, \end{aligned} \tag{2}$$

where $f(n)$ is the actual output of the ensemble for the n-th training pattern, and λ is a scaling factor that controls the penalty term. The ensemble output is generally obtained by averaging the outputs of all its component NNs. Thus, for the n-th training pattern, the output of an ensemble consisting of M networks is

$$f(n) = \frac{1}{M}\sum_{i=1}^{M} f_i(n). \tag{3}$$

Similar to the BP algorithm [19], the NCL algorithm [11] also requires the partial derivative of the error function to modify the connection weights of NNs. According to [3], the partial derivative of $e_i(n)$ is

$$\frac{\partial e_i(n)}{\partial f_i(n)} = f_i(n) - d(n) - 2\lambda(f_i(n) - f(n))\left(1 - \frac{1}{M}\right). \tag{4}$$

It is clear from Eq. (4) that NCL needs to know the ensemble output (i.e., $f(n)$) for updating the weight of each NN. This means an NN needs to communicate with all other NNs in the ensemble for updating its weight. This kind of direct interaction among all NNs in the ensemble is time consuming, and NNs may engage in competition during training [9]. In addition, the number of NNs to construct an ensemble needs to be predefined in NCL.

To reduce training time interaction and competition among NNs, PITS employs an indirect communication scheme for training NNs in an ensemble. The indirect communication scheme is found in many living organisms (e.g., ants). In PITS, NNs in the ensemble are trained one by one in a progressive manner, where each NN is concerned with a specific task that has not been solved by any previously trained NN. The training process of PITS starts with a single NN. This network is trained for a certain number of training cycles, and its output is stored in IC after the completion of training. The proposed PITS then trains the second NN with the aim of reducing the remaining ensemble error. The second NN interacts with IC during training to know which parts of the training data were solved by the first NN. After completing the training process of the second NN, PITS updates IC by combining the outputs of the second NN with those of the first NN. This process will continue until the completion of the training of all NNs in the ensemble or the problem has been solved. Since PITS trains NNs in the ensemble one after one, the remaining ensemble error that NNs try to minimize during their training is different. This will definitely reduce the competition among NNs. Furthermore, each NN can get information from all the previously trained NNs only by communicating with the IC. This means the NN can get the information of all previously trained NNs using a single fetch operation.

To formulate PITS from NCL, Eq. (4) can written in the following way when M is large [9]

$$\frac{\partial e_i(n)}{\partial f_i(n)} = f_i(n) - d(n) - 2\lambda(f_i(n) - f(n)).$$

Using the value of $f(n)$ from Eq. (3), the partial derivative can be written as

$$\begin{aligned} \frac{\partial e_i(n)}{\partial f_i(n)} &= f_i(n) - d(n) - 2\lambda\left(f_i(n) - \frac{1}{M}\sum_{j}^{M} f_j(n)\right) \\ &= f_i(n) - d(n) - 2\lambda\left(\frac{M-1}{M}f_i(n) - \frac{1}{M}\sum_{j\neq i} f_j(n)\right). \end{aligned} \tag{5}$$

In Eq. (5), $\sum_{j\neq i} f_j(n)$ is the summed of outputs of all NNs except the i-th NN in the ensemble for the n-th training pattern. The proposed PITS stores this combined output in the IC. Let

$$\sum_{j\neq i} f_j(n) = \sum_{j=1}^{i-1} f_j(n) = f_{IC}(n).$$

PITS updates the IC on a pattern by pattern basis for each NN in the ensemble. The following formulation is used to update the IC,