

Fast Neural Network Ensemble Learning via Negative-Correlation Data Correction

Zeke S. H. Chan and Nik Kasabov, *Senior Member, IEEE*

Abstract— This letter proposes a new Negative Correlation (NC) learning method that is both easy to implement and has the advantages that (i) it requires much lesser communication overhead than the standard NC method; and (ii) it is applicable to ensembles of heterogeneous networks.

Index Terms—Negative Correlation Learning, Ensemble Learning, Distributed Computing.

I. INTRODUCTION

THERE are three aspects desirable for practical neural network ensemble learning: first, an efficient parallel computing scheme that can cope with the computationally intensive task of training multiple neural networks; second, the capability to integrate different network models that may be heterogeneous or obtained from a third-party source, e.g. MLPs and RBFs to provide a more diversified output; and third, a cooperative learning method that promotes interaction between networks.

For implementing cooperative learning, Negative Correlation (NC) method [1-3] has been shown to improve ensemble generalization performance both theoretically and practically. Standard NC learning requires modifying the error functions of component networks to incorporate a penalty function that promote negatively correlated prediction errors, which in effect causes the networks to diversify in their outputs and each network to specialize in a particular aspect of the data. However, the methods developed so far can only assemble networks that use backpropagation training, and can demand prohibitively high communication bandwidth (between component networks) that hinders parallel speedup. In addition, every component network must be reprogrammed to include the penalty function in the training objective, which raises difficulties in using third party codes. These drawbacks significantly limit the practical application of NC method for ensemble learning.

To overcome these limitations, we propose a new NC based

method called *Negative Correlation Learning via Correlation-Corrected Data* (NCCD). It differs from all previous NC methods that instead of modifying every component network's error function to incorporate error correlation information, NCCD modifies the training data and creates sets of *correlation-corrected data* (C-C data) as new training data, which induce NC learning when the component networks are trained on them. Thus without the requirement of recoding each component network, NC learning becomes very simple to implement and can be used for assembling heterogeneous networks. Another major advantage is that NCCD significantly reduces network communication bandwidth, making parallel speedup more effective (due to the higher computation-to-communication ratio or *granularity*).

II. OVERVIEW OF NEGATIVE CORRELATION LEARNING

Ensemble learning involves two stages: training the networks and combining their outputs. Here we focus on the application of NC learning to the former, and assume simple averaging for combining network outputs. NC learning requires i) inclusion of a correlation penalty function in the error function of each network and ii) periodic communication between component networks. Let $\mathbf{T}=\{\mathbf{x},\mathbf{d}\}=\{(x(1),d(1)),(x(2),d(2)),\dots,(x(N),d(N))\}$ represents the training data where N is the number of patterns and $\{\mathbf{x},\mathbf{d}\}$ are the input and output (target) vector respectively. We form an ensemble of M networks whose joint output \mathbf{F} is the average of the i th network outputs \mathbf{F}_i , $\forall i=[1,2,\dots,M]$. Consequently, the error \mathbf{E} is also the average of the i th network errors $\mathbf{E}_i=\|\mathbf{d}-\mathbf{F}_i\|^2$. \mathbf{F} and \mathbf{E} are given by

$$\mathbf{F}(n) = \frac{1}{M} \sum_{i=1}^M \mathbf{F}_i(n) \quad \mathbf{E}(n) = \frac{1}{M} \sum_{i=1}^M \mathbf{E}_i(n) \quad (1)$$

where n denote the index of the training sample. We define a correlation penalty \mathbf{P}_i that measures the error correlation between the i th network and the rest of the ensemble as follows. Recall that the goal of generalization is to learn the generating function of the output and not the target data itself, otherwise "overfitting" of data may occur. We use $\mathbf{F}(n)$ to approximate the generating function such that $(\mathbf{F}_i(n)-\mathbf{F}(n))$ approximates the error of the i th network and $\sum_{\forall j \neq i} (\mathbf{F}_j(n)-\mathbf{F}(n))$ the joint error of the rest of the ensemble from the generating function. The error correlation \mathbf{P}_i is then obtained as their product

Manuscript received May 13, 2004. This research is supported by the KEDRI postdoctoral fellow research fund.

Zeke S. H. Chan is with the Knowledge Engineering and Discover Research Institute (KEDRI), Auckland University of Technology, AUT Tech Park, 581-585 Great South Rd., Ronald Trotter House, Penrose, Auckland, New Zealand ((649) 917-9546; e-mail: shchan@aut.ac.nz).

Nik Kasabov is currently the director of KEDRI (e-mail: nkasabov@aut.ac.nz).

$$\mathbf{P}_i(n) = (\mathbf{F}_i(n) - \mathbf{F}(n)) \sum_{\forall j \neq i} (\mathbf{F}_j(n) - \mathbf{F}(n)) \quad (2)$$

For NC learning, the new error function \mathbf{E}_i is a weighted sum of the original error function and the penalty function \mathbf{P}_i , given by

$$\mathbf{E}_i(n) = \frac{1}{2} \|\mathbf{F}_i(n) - \mathbf{d}(n)\|^2 + \lambda \mathbf{P}_i(n) \quad (3)$$

where $0 \leq \lambda \leq 1$ is the *hyperparameter* (a term used to describe a similar instance in network regularization) that adjusts the strength of the correlation penalty. For learning the weights of the i th network through standard backpropagation, the partial derivative of the ensemble error \mathbf{E} with respect to \mathbf{F}_i is obtained using (1), (2) and (3)

$$\frac{\partial \mathbf{E}(n)}{\partial \mathbf{F}_i(n)} = \frac{1}{M} [(\mathbf{F}_i(n) - \mathbf{d}(n)) - 2\lambda(\mathbf{F}_i(n) - \mathbf{F}(n))] \quad (4)$$

For network training, the derivative in (4) requires periodic updating of the ensemble output $\mathbf{F}(n)$. In Liu and Yao's CELS [3], which is the best NC learning algorithm reported to date, the updating is on a pattern-by-pattern basis, making the communication bandwidth very high. To measure the communication overhead, we define n_{ex} to be the total communication cost for sending each network's prediction for every training sample to update $\mathbf{F}(n)$ during the entire training period. Given g_{tot} denote the total number of training epochs, CELS requires a communication overhead of $n_{\text{ex}} = g_{\text{tot}} \times N$.

III. NEGATIVE CORRELATION LEARNING VIA CORRELATION-CORRECTED DATA (NCCD)

Correlation-Corrected (C-C) data are transformed training data that induce NC learning when the networks are trained on them. Each network is assigned its own set of C-C data that is updated periodically. The principle of NCCD is illustrated in the example of training the i th network in an ensemble of M network in Fig. 1. Let \mathbf{E}_i denote the prediction error of the i th network and $\mathbf{E}_{-i} = \sum_{j \neq i} \mathbf{E}_j$ denote the joint prediction error of all but the i th networks. Initially, the output errors are positively correlated (both \mathbf{E}_i and \mathbf{E}_{-i} have the same directional vector, Fig. 1(a)), causing a large joint error $|\sum_i \mathbf{E}_j| > |\mathbf{E}_i| + |\mathbf{E}_{-i}|$. This is often known as the *co-linearity* problem [4]. NCCD creates a new set of training data called the C-C data, denoted $\mathbf{T}^{(c)}_i = \{\mathbf{x}, \mathbf{c}_i\}$, for the i th network. It consists of the original training input data \mathbf{x} and the transformed target data \mathbf{c}_i whose error to the original training data is negatively correlated with the joint output (note that \mathbf{c}_i appears above the true output, Fig. 1(b)). Now by training the i th network on the C-C data $\mathbf{T}^{(c)}_i$, \mathbf{E}_i becomes negatively correlated with \mathbf{E}_{-i} . The two errors cancel each other out, yielding a smaller joint error $|\sum_i \mathbf{E}_j| < |\mathbf{E}_i| + |\mathbf{E}_{-i}|$, therefore alleviating the co-linearity problem.

The critical procedure of NCCD is the generation of the transformed target data \mathbf{c}_i , which is derived as the desired network output \mathbf{F}'_i that minimizes the ensemble error \mathbf{E} . It is obtained by setting the derivative of \mathbf{E} (4) to zero

$$\text{At } \frac{\partial \mathbf{E}(n)}{\partial \mathbf{F}_i(n)} = 0, \quad \mathbf{F}_i^*(n) = \mathbf{c}_i(n) = \frac{\mathbf{d}(n) - 2\lambda \mathbf{F}(n)}{1 - 2\lambda} \quad (5)$$

The transformed target \mathbf{c}_i in (5) must be periodically updated with the latest ensemble output \mathbf{F} to promote network interaction. Since \mathbf{c}_i embeds all relevant error-correlation information, it can be used for independent network training for a long period of time without updating and still induce NC learning. In fact, the update interval for \mathbf{c}_i must be long enough to allow the networks to capture the salient features of \mathbf{c}_i and yet short enough to allow network interaction. By empirical experiment, we find that the optimal update interval is problem-dependent and is in the range of 1-1000 epochs (each epoch represents one pass of all training patterns). This is markedly different from CELS that emphasises simultaneous learning of all networks through pattern-by-pattern updating of the partial derivative in (4). The longer updating interval reduces communication bandwidth dramatically, making NCCD effective for speedup using parallel computing.

Fig. 2 shows the distributed computing environment applicable for implementing NCCD. Each component network of the ensemble operates on a different processor node. A control centre is used to centralize all information flow and its tasks are to i) generate the correlation-corrected data for each network, ii) send them out and iii) collect the trained network outputs. Let g_{update} and g_{tot} denote the length of C-C data update period (in epochs) and the total number of training epochs allowable per network, respectively. The updating of the correlation-corrected data \mathbf{c}_i may be implemented synchronously (after all networks have finished training for g_{update} epochs) or asynchronously (whenever a network has finished training for g_{update} epochs). Here we implement the later as both methods perform similarly. The procedures are summarized in the following pseudo codes:

- Step 1:* Initialize M networks with random weights. Partially train each network to the training data $\mathbf{T} = \{\mathbf{x}, \mathbf{d}\}$ for g_{update} epochs and then obtain network output.
- Step 2:* Wait upon receipt of the i th network output \mathbf{F}_i at the control centre:
- a) Update the ensemble output \mathbf{F}
 - b) Create the C-C target data \mathbf{c}_i using (5)
 - c) Send C-C data $\mathbf{T}^{(c)}_i = \{\mathbf{x}, \mathbf{c}_i\}$ to the i th network and train it for g_{update} epochs.
 - d) Send network output \mathbf{F}_i to control centre.
- Step 3:* Stop if each network has trained for a total of g_{tot} epochs; else go to *step 2*.

IV. EXPERIMENTS AND RESULTS

We apply NCCD to both regression and classification problems. The regression problem is on predicting the Mackey-Glass chaotic time-series, generated by

$$\dot{x}(t) = \beta x(t) + \frac{\alpha x(t - \tau)}{1 + x^{10}(t - \tau)}$$

with parameters $\alpha=0.2$, $\beta=0.1$ and $\tau=14$ and initial conditions $x(0)=1.2$, $x(t-\tau)=0$ for $0 \leq t \leq \tau$ and time-step=1. The input variables are $\{x(t), x(t-6), x(t-12), x(t-18)\}$ and the output variable is $x(t+6)$. Both the training set and test set consist of 500 data points taken from the 118-617th time point and the 618-1117th time point respectively. Performance is assessed by the prediction error on the test set measured in normalized root-mean-square (NRMSE) error, which is the root-mean-square error divided by the standard deviation of the series. Our ensemble setup follows that for CELS [3]. It contains $M=20$ multilayer perceptron networks (MLPs). Each network contains one hidden layer of 6 neurons. The hidden and output activation functions are hyperbolic tangent and linear function respectively. The total number of training epochs g_{tot} is set to 10,000.

First of all, we investigate the convergence property of NCCD at different parameter settings. NCCD is evaluated 5 times at each combination of update intervals $g_{\text{update}}=[10, 20, 40, \dots, 2,560]$ and $\lambda=[0.1, 0.2, \dots, 0.5]$ (beyond 0.5 the system diverges: see footnote¹note 1) and the results are plotted in Fig. 3(a). First, we see that performance is very sensitive to λ ; in this case, the higher the value of λ , the lower the prediction errors become, and the optimal value is 0.5. At fixed values of λ , the error curves are generally similar: errors are highest at $g_{\text{update}}=1$, but then decrease rapidly with increasing g_{update} until they stabilize after $g_{\text{update}}>100$. Performances are most consistent (with small error variances) in the range $100 < g_{\text{update}} < 500$. This result agrees with the previous discussion that the length of g_{update} must be balanced to facilitate both network interaction and network learning of the salient features of the C-C data. A large value of optimal g_{update} is of significant advantage in terms of efficiency because it reduces communication bandwidth. We can use empirical methods like cross-validation to determine the optimal values of λ and g_{update} .

Next, we use $g_{\text{update}}=100$ and $\lambda=0.5$, randomly choose a component network and plot (i) the error between its C-C data and the original training data (denoted the C-C data error), (ii) its prediction error on the training set (the training error) and (iii) on the validation set (the validation error) over the training process of 10,000 epochs in Fig. 3(b). Note that the C-C data error decreases unilaterally as the training error and the validation error decreases, showing that lesser correlation-correction on the original target is required to create the C-C data as the network predicts with higher accuracy. We can use this property of the C-C data error for diagnosis of convergence problems and adaptive-control of NCCD parameters such as λ (e.g. lower λ if C-C data error starts rising). It contributes to an important advantage of NCCD and will be further investigated in future works.

¹ NC learning diverges above a certain maximum value of λ . In [3], the maximum value of λ is 1.0, but this value is based on a mistake made in the derivative of E (eq.12, [3]), which is corrected in a later publication (eq.10, [6]). The correction leads to a maximum value of roughly 0.5, which corresponds to the divergent point we discover here.

Performance comparison between independent training, CELS and NCCD is shown in Table 1. Both NC learning methods outperform independent training. NCCD is more cost-effective than CELS in terms of communication overhead: at $g_{\text{update}}=100$ and $\lambda=0.5$, it scores slightly higher error (0.0115 cf 0.0100), yet it requires communication overhead of only $n_{\text{ex}}=(M \times g_{\text{tot}}/g_{\text{update}})=(20 \times 10,000/100)=2,000$ rather than $n_{\text{ex}}=(N \times g_{\text{tot}})=(500 \times 10,000)=5 \times 10^6$, which is lesser by a factor of 2,500.

The classification problem is a real-life problem of Australian credit card approval. The task is to assess credit card applications based on a number of attributes. The data set consists of 690 cases and each case has two output classes and 14 input variables. We use the first 518 cases for training and the remaining 172 cases for testing, following that for CELS [3]. The ensemble output is obtained using the “winner-take-all” (WTA) method – i.e. the output node that has the highest activation among all networks wins. Performance of NCCD is measured in test error averaged over 30 runs.

In this problem, we demonstrate NCCD’s capability to (i) ensemble heterogenous networks that may use non-backpropagation type learning, and to (ii) improve the generalization performance of the ensembles through negative correlation learning. We create ensembles of (i) 4 MLPs, (ii) 2 MLPs plus 2 RBFs, (iii) 4 RBFs and (iv) 4 Support Vector Regression networks (SVRs) [5] and compare their performances with and without NCCD training. For NCCD settings, we use $g_{\text{update}}=1$ and $\lambda=0.25$ based on limited trials. The parameter settings for the networks are as follows. Each MLP has ten hidden nodes and the total number of training epochs is set to 25 (following that for CELS [3]). The RBFs use a two-stage learning method that first positions the basis functions (10 spherical Gaussians) with the Expectation Maximization algorithm, and then adjusts only the connection weights using the least square method in subsequent training. The SVRs use linear ϵ -insensitive cost function with Gaussian kernel and parameters² $C=10^{-4}$ and $\epsilon=0.1$, and they are trained using the quadratic programming method. Note that neither the RBFs nor SVRs use backpropagation-type learning algorithms.

Results are shown in Table 2. Although each ensemble performs differently from each other, NCCD improves the generalization performance of all by 0.2%-1.0%. Comparing with CELS on the ensemble of 4 MLPs, NCCD scores slightly higher error (12.2% c.f. 12.0%), yet requiring 130 time lesser network communications ($(M \times g_{\text{tot}}/g_{\text{update}})=(4 \times 25/1)=100$ c.f. $(N \times g_{\text{tot}})=(518 \times 25)=12,950$). NCCD is clearly the more cost-effective method.

² Here we use the same symbols ϵ and C common for formulating support vector machines or regression networks: ϵ is the maximum prediction deviation allowable from the observations, and C is the hyperparameter that determines the trade-off between the flatness of the prediction function and the amount up to which deviations larger than ϵ are tolerated.

V. CONCLUSIONS

The theory and experiments have demonstrated the proposed NC learning method.

ACKNOWLEDGMENT

This research is supported by the KEDRI postdoctoral fellow research fund. The authors are grateful to the various reviewers for their helpful comments.

REFERENCES

- [1] R. T. Clemen and R. L. Winkler, "Limits for the precision and value of information from dependent sources," *Operation Research*, vol. 33, pp. 427-442, 1985.
- [2] M. Perrone and L. N. Cooper, "When networks disagree: ensemble methods for hybrid neural networks," in *Neural Networks for Speech and Image Processing*, R. J. Mammone, Ed. London, U.K.: Chapman & Hall, 1993.
- [3] Y. Liu and X. Yao, "Simultaneous Training of Negatively Correlated Neural Networks in an Ensemble," *IEEE Transactions on Systems, Man and Cybernetics-Part B: Cybernetics*, vol. 29, pp. 716-725, 1999.
- [4] B. E. Rosen, "Ensemble Learning using Decorrelated Neural Networks," *Connection Science*, vol. 8, pp. 373-384, 1996.
- [5] A. Smola and B. Scholkopf, "A tutorial on support vector regression," *Statistics and Computing*, vol. 14, pp. 199-222, 2004.
- [6] M. Islam, X. Yao, and K. Murase, "A Constructive Algorithm for Training Cooperative Neural Network Ensembles," *IEEE Transactions on Neural Networks*, vol. 14, pp. 820-834, 2003.



Zeke S. H. Chan is currently a senior research fellow at the Knowledge Engineering Discovery and Research Institute (KEDRI) in Auckland, New Zealand. His research interest lies in the area of Bayesian Inference and Computation Intelligence; in particular, Markov Chain Monte Carlo, Evolutionary Computation, Bayesian parameter estimation and prediction for Neural Networks in forecasting, control and Bioinformatics applications.

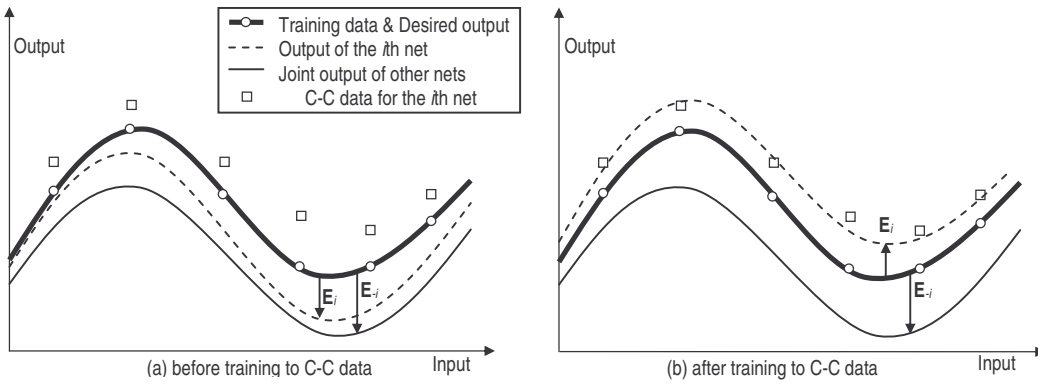


Fig. 1 Illustration of the principle of correlation-corrected (C-C) data. a) The output errors of the i th network and the rest of networks, denoted by E_i and E_{-i} respectively, are positively correlated. b) After training to the C-C data, E_i and E_{-i} become negatively correlated.

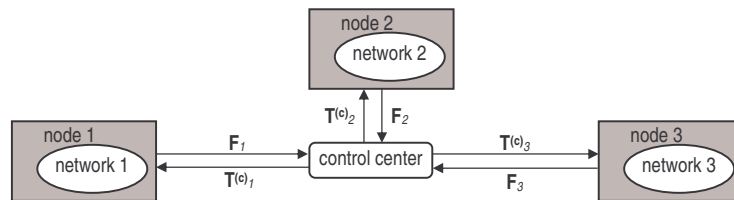


Fig. 2 An example of distributed computing environment suitable for implementing NC learning using correlation-corrected data on an ensemble of three component networks.

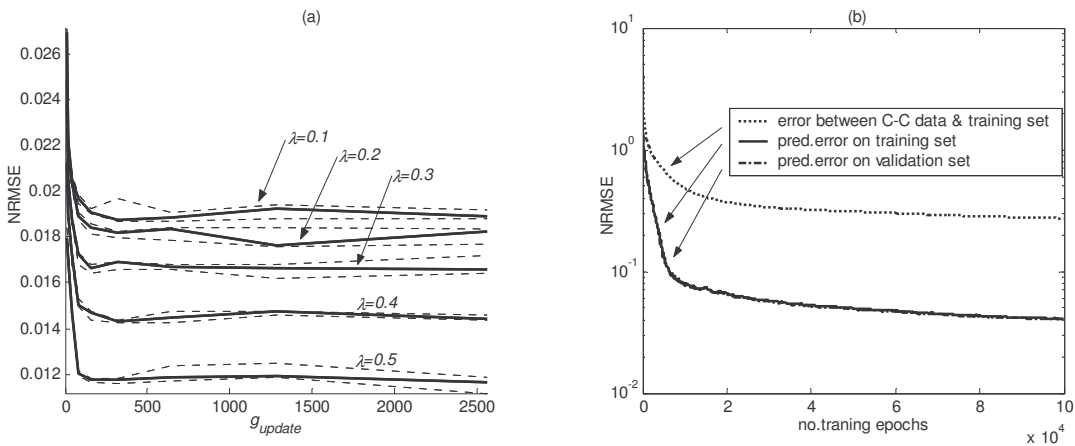


Fig. 3 Error plots showing the convergence of NCCD on the Mackey-Glass data: (a) plot of test set errors versus update interval g_{update}; (b) plot of C-C data errors, training set errors and validation set errors.

Table 1 Comparison of test errors achieved by different methods on the Mackey-Glass data. n_{ex} represents the number of network communications required.

	Test error (NRMSE)	Com.overhead, n_{ex}
Indep.learning	0.02	0
CELS	0.0100	5×10^6
NCCD	0.0115	2000

Table 2 Performance comparison between independent training, NCCD (on the ensembles of (i) 4 MLPs, (ii) 4 RBFs and (iii) 2 RBFs plus 2 MLPs and (iv) 4 SVRs on Australian credit card assessment. n_{ex} represents the amount of required network communications. Note that CELS is only applicable to ensembles of MLPs.

Ensemble	Indep.learning	NCCD		CELS	
	Mean Err.	Mean Err.	n_{ex}	Mean Err.	n_{ex}
4 MLPs	12.4%	12.2%	100	12.0%	12,950
2 RBFs+2 MLPs	12.9%	12.6%	100	/	/
4 RBFs	13.9%	12.9%	100	/	/
4 SVRs	14.0%	13.4%	100	/	/