Fast Neural Network Ensemble Learning via Negative-Correlation Data Correction

Zeke S. H. Chan and Nik Kasabov

***Abstract*—This letter proposes a new negative correlation (NC) learning method that is both easy to implement and has the advantages that: 1) it requires much lesser communication overhead than the standard NC method and 2) it is applicable to ensembles of heterogenous networks.**

***Index Terms*—Distributed computing, ensemble learning, negative correlation (NC) learning.**

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 heterogenous or obtained from a third-party source, e.g., multilayer perceptrons (MLPs) and radial basis functions (RBFs) to provide a more diversified output. 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 heterogenous 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*).

Manuscript received May 13, 2004; revised March 3, 2005. This work was supported by the Knowledge Engineering and Discover Research Institute (KEDRI) Postdoctoral Fellow Research Fund.

The authors are with the Knowledge Engineering and Discover Research Institute (KEDRI), Auckland University of Technology, Auckland, New Zealand (e-mail: shchan@aut.ac.nz; nkasabov@aut.ac.nz).

Digital Object Identifier 10.1109/TNN.2005.852859

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: 1) inclusion of a correlation penalty function in the error function of each network and 2) periodic communication between component networks. Let $\mathbf{T} = \{\mathbf{x}, \mathbf{d}\} = \{(x(1), d(1)), (x(2), d(2)), \ldots, (x(N), d(N))\}$ 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 ith network outputs $\mathbf{F}_i$, $\forall i = [1, 2, \ldots, M]$. Consequently, the error $\mathbf{E}$ is also the average of the ith network errors $\mathbf{E}_i = \|\mathbf{d} - \mathbf{F}_i\|^2$. $\mathbf{F}$ *and* $\mathbf{E}$ are given by

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

where n denote the index of the training sample. We define a correlation penalty $\mathbf{P}_i$ that measures the error correlation between the ith 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 ith 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

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

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

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

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 ith network through standard backpropagation, the partial derivative of the ensemble error $\mathbf{E}$ with respect to $\mathbf{F}_i$ is obtained using (1)–(3)

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

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. NCCD

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 ith network in an ensemble of

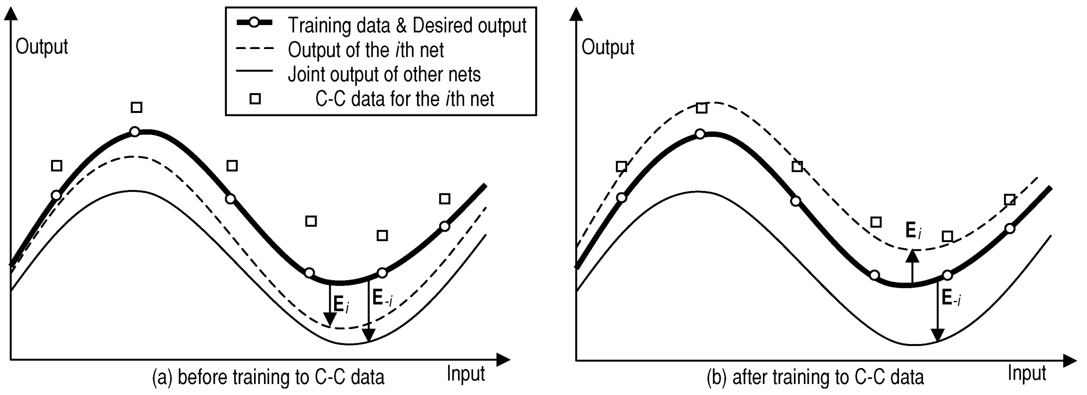
Training data & Desired output
Output of the *i*th net
Joint output of other nets
C-C data for the *i*th net
Output
Input
$\mathbf{E}_i$
$\mathbf{E}_{-i}$
(a) before training to C-C data
(b) after training to C-C data

Fig. 1. Illustration of the principle of C-C data. a) Output errors of the ith network and the rest of networks, denoted by $\mathbf{E}_i$ and $\mathbf{E}_{-i}$, respectively, are positively correlated. b) After training to the C-C data, $\mathbf{E}_i$ and $\mathbf{E}_{-i}$ become negatively correlated.

M network in Fig. 1. Let $\mathbf{E}_i$ denote the prediction error of the ith network and $\mathbf{E}_{-i} = \sum_{j\neq i}\mathbf{E}_j$ denote the joint prediction error of all but the ith 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 $\mathbf{T}_i^{(c)} = \{\mathbf{x}, \mathbf{c}_i\}$, for the ith 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 ith network on the C-C data $\mathbf{T}_i^{(c)}$, $\mathbf{E}_i$ 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

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

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 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 emphasizes 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 center is used to centralize all information flow and its tasks are to: 1) generate the C-C data for each network, 2) send them out, and 3) 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 C-C 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

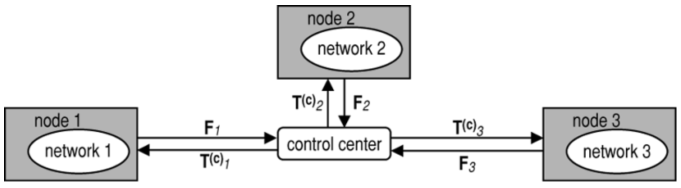

Fig. 2. Example of distributed computing environment suitable for implementing NC learning using C-C data on an ensemble of three component networks.

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 $g_{\text{update}}$
        epochs and then obtain network output.
Step 2) Wait upon receipt of the $i$th network output $\mathbf{F}_i$ at the control
        center:
        a) update the ensemble output $\mathbf{F}$;
        b) create the C-C target data $\mathbf{c}_i$ using (5);
        c) send C-C data $\mathbf{T}_i^{(c)} = \{\mathbf{x}, \mathbf{c}_i\}$ to the $i$th network and
           train it for $g_{\text{update}}$ epochs;
        d) send network output $\mathbf{F}_i$ to control center.
Step 3) Stop if each network has trained for a total of $g_{\text{tot}}$ 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 118th–617th time point and the 618th–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 $M = 20$ MLP networks. Each network contains one hidden layer of six neurons. The hidden and output activation functions are

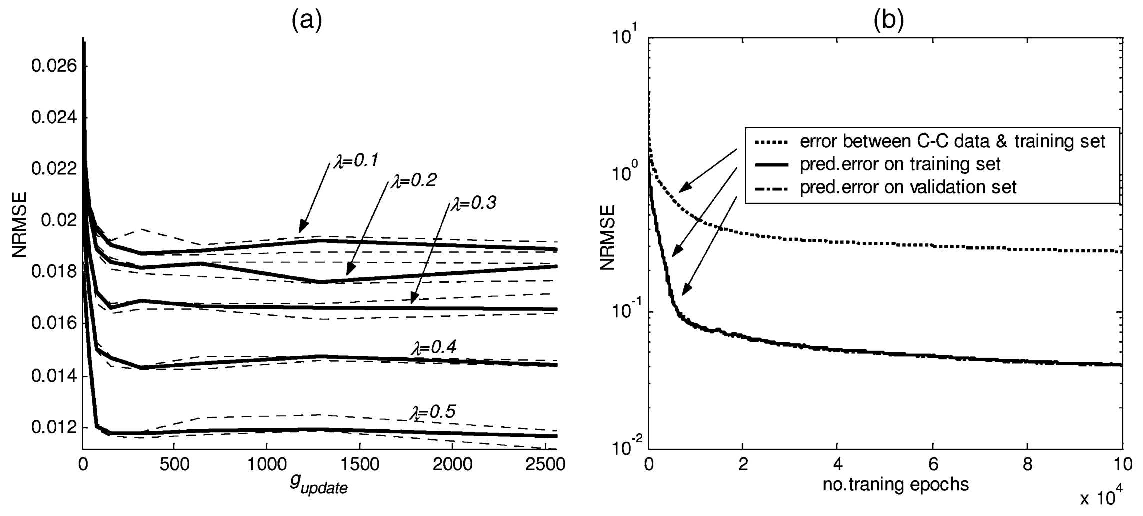

Fig. 3. Error plots showing the convergence of NCCD on the Mackey–Glass data. (a) Plot of test set errors versus update interval gupdate. (b) Plot of C-C data errors, training set errors, and validation set errors.

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 $g_{\text{update}} = [10, 20, 40, \ldots, 2, 560]$ and $\lambda = [0.1, 0.2, \ldots, 0.5]$ (beyond 0.5 the system diverges[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 1) the error between its C-C data and the original training data (denoted the C-C data error), 2) its prediction error on the training set (the training error), and 3) 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.

Performance comparison between independent training, CELS and NCCD is shown in Table I. 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 c.f. 0.0100), yet it requires communication overhead of only $n_{\text{ex}} = (M \times g_{\text{tot}}/g_{\text{update}}) = (20 \times 10\,000/100) = 2000$ 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 2500.

TABLE I
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**

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 NCCDs capability to 1) ensemble heterogenous networks that may use nonbackpropagation type learning and to 2) improve the generalization performance of the ensembles through NC learning. We create ensembles of 1) four MLPs, 2) two MLPs plus two RBFs, 3) four RBFs, and 4) four 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 parameters[2] $C =$

[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 [3, eq. (12)], which is corrected in a later publication [6, eq. (10)]. The correction leads to a maximum value of roughly 0.5, which corresponds to the divergent point we discover here.

[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 tradeoff between the flatness of the prediction function and the amount up to which deviations larger than ε are tolerated.

TABLE II
PERFORMANCE COMPARISON BETWEEN INDEPENDENT TRAINING, NCCD (ON THE ENSEMBLES OF 1) FOUR MLPS, 2) FOUR RBFS, AND 3) TWO RBFS PLUS TWO MLPS AND 4) FOUR 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**	/	/

10^{-4} and $\varepsilon = 0.1$, 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 II. 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 four MLPs, NCCD scores slightly higher error (12.2% c.f. 12.0%), yet requiring 130 time lesser network communications $((M \times g_{\mathrm{tot}}/g_{\mathrm{update}}) = (4 \times 25/1) = 100$ c.f. $(N \times g_{\mathrm{tot}}) = (518 \times 25) = 12\,950)$. NCCD is clearly the more cost-effective method.

V. CONCLUSION

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

ACKNOWLEDGMENT

The authors would like to thank 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," *Oper. Res.*, 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 Trans. Syst., Man, Cybern. B, Cybern.*, vol. 29, no. 6, pp. 716–725, Dec. 1999.
[4] B. E. Rosen, "Ensemble learning using decorrelated neural networks," *Connection Sci.*, vol. 8, pp. 373–384, 1996.
[5] A. Smola and B. Scholkopf, "A tutorial on support vector regression," *Statist. Comput.*, vol. 14, pp. 199–222, 2004.
[6] M. Islam, X. Yao, and K. Murase, "A constructive algorithm for training cooperative neural network ensembles," *IEEE Trans. Neural Netw.*, vol. 14, no. 4, pp. 820–834, Jul. 2003.

Exponential ε-Regulation for Multi-Input Nonlinear Systems Using Neural Networks

Shaosheng Zhou, James Lam, Gang Feng, and Daniel W. C. Ho

***Abstract*—This paper considers the problem of robust exponential ε-regulation for a class of multi-input nonlinear systems with uncertainties. The uncertainties appear not only in the feedback channel but also in the control channel. Under some mild assumptions, an adaptive neural network control scheme is developed such that all the signals of the closed-loop system are semiglobally uniformly ultimately bounded and, under the control scheme with initial data starting in some compact set, the states of the closed-loop system is guaranteed to exponentially converge to an arbitrarily specified ϵ-neighborhood about the origin. The important contributions of the present work are that a new exponential uniformly ultimately bounded performance is proposed and that the design parameters and initial condition set can be determined easily. The development generalizes and improves earlier results for the single-input case.**

***Index Terms*—Adaptive control, multi-input system, neural network, nonlinear system, uncertain system, uniform ultimate boundedness.**

I. INTRODUCTION

The study of uncertain nonlinear systems has been one of the most active research topics in recent years (see, for example, [3]–[5], [7]–[9] and references therein). Neural network techniques have been found to be particularly useful for controlling nonlinear systems with uncertainties [5], [8]. Neural network approximators can be used to parameterize an unknown nonlinear function over a compact set to any degree of accuracy [2]. Representative work can be found in, to just name a few, [5], [6], [8]–[11], and [13]. By using neural network, Polycarpou and Mears [8] developed control laws that guarantee semiglobal uniform ultimate boundedness. Zhang *et al.* [11], [12] developed control laws to solve the control problem with uncertainty in control channel. All the plants considered in [8] and [10]–[12] are single-input nonlinear systems with uncertainties. Recently, Kwan and Lewis [5] developed control laws to control a more general class of multi-input plants without uncertainty in the control channel. It is noted that the developments in [5], [8], and [10]–[12] all guarantee uniform ultimate boundedness of the closed-loop signals. That is, all the closed-loop signals converge not to a point but to a ball-type residual set (asymptotic bounding). As the magnitude of the uncertain nonlinearities increases, the size of the residual set may also increase. Therefore, these control schemes may not be applicable if the uncertain nonlinearities dominate the system dynamics to the extent that the achievable residual set is beyond the application range.

Motivated by [5], [8], and [10]–[12], this paper develops a dynamic feedback control scheme based on neural networks for a new class of multi-input plants with dominating uncertain nonlinearity in the feedback channel and time-invariant uncertainty in the control channel. The

Manuscript received December 31, 2002; revised April 4, 2005. This work was supported in part by RGC under Grant 7103/01P and the National Natural Science Foundation of P. R. China under Grant 60574080.

S. Zhou is with the Institute of Automation, Qufu Normal University, Qufu 273165, Shandong, China (e-mail: zss@myself.com).
J. Lam is with the Department of Mechanical Engineering, University of Hong Kong, Hong Kong, China (e-mail: james.lam@hku.hk).
G. Feng is with the Department of Manufacturing Engineering and Engineering Management, City University of Hong Kong, Hong Kong, China.
D. W. C. Ho is with the Department of Mathematics, City University of Hong Kong, Hong Kong, China.

Digital Object Identifier 10.1109/TNN.2005.853335