Multiobjective Neural Network Ensembles based on Regularized Negative Correlation Learning

Huanhuan Chen, *Member, IEEE,* and Xin Yao, *Fellow, IEEE*

Abstract—Negative Correlation Learning (NCL) [1], [2] is a neural network ensemble learning algorithm which introduces a correlation penalty term to the cost function of each individual network so that each neural network minimizes its mean-square-error (MSE) together with the correlation. This paper describes NCL in detail and observes that the NCL corresponds to training the entire ensemble as a single learning machine that only minimizes the MSE without regularization. This insight explains that NCL is prone to overfitting the noise in the training set. The paper analyzes this problem and proposes the multiobjective regularized negative correlation learning (MRNCL) algorithm which incorporates an additional regularization term for the ensemble and uses the evolutionary multiobjective algorithm to design ensembles. In MRNCL, we define the crossover and mutation operators and adopt nondominated sorting algorithm with fitness sharing and rank-based fitness assignment. The experiments on synthetic data as well as real-world data sets demonstrate that MRNCL achieves better performance than NCL, especially when the noise level is non-trivial in the data set. In the experimental discussion, we give three reasons why our algorithm outperforms others.

Index Terms—Multiobjective algorithm, Multiobjective Learning, Neural Network Ensembles, Neural Networks, Negative Correlation Learning, Regularization.

1 Introduction

Ensemble of multiple learning machines, i.e. a group of learners that work together as a committee, has attracted a lot of research interests in the machine learning community since it is considered as a good approach to improve the generalization ability [3]. Most ensemble learning algorithms train the individual neural network independently or sequentially, so the advantages of interaction and cooperation among the individual networks are not exploited. However, Liu and Yao [1], [2] have shown that the cooperation with ensemble members is useful for obtaining better ensembles. This new approach opens a new research area where the design and training of the different networks can be interdependent.

Negative Correlation Learning (NCL) [1], [2] emphasizes the interaction and cooperation among individual neural networks in the ensemble and has performed well on a number of empirical applications, including regression problems [4] and classification problems [5]. NCL introduces a correlation penalty term to the cost function of each individual network so that each neural network minimizes its MSE together with the correlation with the ensemble.

According to the definition of NCL, it seems that the correlation term in the cost function acts as the regularization term. However, we observe that the training of NCL with the penalty coefficient λ setting to 1 corresponds to treating the entire ensemble as a single estimator and considering only the empirical training error without regularization. In this case, NCL only reduces the empirical MSE of the ensemble, and it pays less attention to regularizing the complexity of the ensemble, which leads NCL to be prone to overfitting the noise in the training set. Similarly, setting a zero or small positive λ corresponds to independently training these estimators without regularization and in this case, NCL is prone to overfitting as well.

NCL can use the penalty coefficient λ to explicitly alter the emphasis on the MSE and correlation portion of the ensemble and thus alleviate the overfitting problem to some extent. However, NCL could not totally overcome the overfitting problem by tuning this parameter without regularization, especially when dealing with data with non-trivial noise, which will be implicitly evidenced by the empirical work on multi-objective implementation of NCL in this paper. The regularization term is especially beneficial to NCL since large weights are usually connected with near linear dependence among groups of units in the network, negative correlation learning would seem to potentiate the appearance of large weights in the ensemble.

Another problem with NCL is that the parameter λ, which controls the trade-off between empirical error and correlation, needs to be tuned. Although this parameter is crucial to the performance of NCL, there is no formulated approach to select the parameter. Optimization of the parameter usually involves cross validation, whose computation is extremely expensive.

In order to address these problems, this paper proposes a multiobjective regularized negative correlation

• *The authors are with The Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA), School of Computer Science, University of Birmingham, Birmingham B15 2TT, United Kingdom (email: {H.Chen, X.Yao}@cs.bham.ac.uk).*

Manuscript received May 19, 2008; revised March 19, 2009 and July 5, 2009; accepted September 22, 2009.

Digital Object Indentifier 10.1109/TKDE.2010.26 1041-4347/10/$26.00 © 2010 IEEE

learning (MRNCL) algorithm. MRNCL incorporates an additional regularization term for the ensemble, which can be decomposed into different parts for each network. By incorporating an additional regularization term, the training of an individual neural network in MRNCL involves minimization of the three terms: empirical training error term, correlation penalty term and the regularization term.

However, how to balance the tradeoff among the three terms is crucial for the generalization performance of ensemble. Poor generalization occurs if the tradeoff is unbalanced. The usual approach is to assign coefficient parameters to these terms and choose the appropriate coefficients based on tedious trial-and-error processes.

The idea of the paper is the introduction of an evolutionary multiobjective algorithm to search the best tradeoff among the three terms: the empirical error, correlation and regularization. Evolutionary multiobjective algorithms are well suited to search the optimal tradeoff among different objectives by parallelizing the search using a population of networks and biasing toward the Pareto front and, at the same time maintaining population diversity to obtain as many candidate solutions as possible. These properties are especially important in ensemble design.

Since the regularization term is considered as one objective in MRNCL, the networks with appropriate regularization are preferable in MRNCL. Thus the obtained ensemble is regularized and is more robust to noise in the training set.

MRNCL algorithm not only addresses the issues concerned with NCL, but also provides the following advantages: (1) Being a multiobjective algorithm, the approach is able to produce a diverse ensemble. Some individuals are good at minimizing the training error; some pay more attention to cooperation and the others manage to control the complexity. (2) The parameters of individual networks can be effectively obtained in the evolutionary multiobjective algorithm. (3) Due to the regularization term, the obtained ensemble is regularized and is more robust to noise in the training set. (4) There is no need to weigh the different objectives by optimizing the coefficient parameters.

The key contributions of this paper include a) we point out that NCL is prone to overfitting and verify this claim using theoretical and extensive empirical work; b) we propose to add an additional regularization term to control the complexity of the ensemble; c) we implement the algorithm using evolutionary multi-objective algorithm and d) we carry out extensive experimental studies to evaluate and compare MRNCL with some existing ones.

The rest of this paper is organized as follows. After the background description in Section 2, the proposed algorithm is introduced in Section 3. Experimental results and discussions are presented in Section 4. Finally, Section 5 concludes the paper.

2 Background

Neural network ensembles [3] are a learning paradigm where a collection of neural networks is trained for the same task. There have been many ensemble methods studied in the literature, such as Bagging [6], Boosting [7], ensemble of features [8] and so on. Most ensemble learning algorithms train the individual neural network independently or sequentially.

Negative correlation learning [1] [2] is a successful neural network ensemble learning algorithm. It is different from previous works such as bagging or boosting, since NCL emphasizes interaction and cooperation among the individual learners in the ensemble by using an unsupervised penalty term in the error function to produce *biased* individuals whose errors tend to be negatively correlated.

In 2000, Abbass [9] firstly proposed a memetic multiobjective evolutionary approach to evolve artificial neural networks. Two objectives are considered in this algorithm. One is to minimize the error and the other is to minimize the number of hidden units, which can be thought as a kind of regularization measure. This method firstly proposed the idea to consider both regularization and accuracy in the multiobjective algorithm and to combine the individuals in the pareto front for final predictions.

In 2001, McKay et al. [10] presented an alternative anti-correlation measure, root-quartic negative correlation learning (RTQRT–NCL) and used the anti-correlation in training neural network ensembles. The empirical results showed significant improvements for both artificial neural networks (ANN) and genetic programming (GP) learning machines. They also derived a theoretical explanation of the improved performance of RTQRT–NCL in larger ensembles. Later, Abbass [11] employed a multiobjective evolutionary algorithm and a gradient-based local search method to train neural networks and simultaneously optimize their architecture. A neural network ensemble can be generated by combining the networks in the final generation.

In 2004, Jin et al. [12] used a multi-objective evolutionary algorithm to optimize the accuracy and regularization of neural networks. As a natural by-product of the multi-objective evolutionary approach to neural network learning, neural network ensembles can be easily constructed using the obtained networks with different levels of model complexity.

Islam et al. [13] took a constructive approach to building the ensemble, starting from a small group of networks with a minimal architecture. The networks are all partially trained using NCL. The approach can automatically determine weights, network topologies and ensemble membership. In the following work, Brown et al. [14] formalized NCL, providing a statistical interpretation of its success. Furthermore, for estimators that are linear combinations of other functions, they derive an upper bound on the penalty coefficient, based on properties of the Hessian matrix.

Diverse and accurate ensemble learning algorithm [15], [16] is an approach that combines evolving neural network and multiobjective algorithm. In this paper, adaptive Gaussian variance is employed for generating the offspring and memetic pareto artificial neural network algorithm [9] is used for evolving neural networks. Finally, diverse and accurate classifiers can be achieved through these procedures. Oliveira et al. [17] use multiobjective evolutionary algorithms to generate an ensemble to solve the handwritten recognition problem. This algorithm produces a set of classifiers with a small number of features and a low error rate by evolving these classifiers with different *randomly* chosen features. The combination weights of ensemble are obtained by a multiobjective algorithm with two different objectives: diversity and accuracy.

Cooperative coevolution of artificial neural network ensembles [18] combines the coevolution of different subpopulations of diverse networks and the evolution of the combination weights of these networks. In this algorithm, the cooperation with the rest of the population is defined as one objective, and each network is evaluated in the evolutionary process using a multiobjective evolutionary method. Thus, the algorithm encourages the collaboration among individuals and improves the combination schemes for the ensemble.

Chen et al. [19] propose to incorporate bootstrap of data, random feature subspaces [8] and evolutionary algorithms with NCL to automatically design accurate and diverse ensembles. The idea promotes the diversity within the ensemble and simultaneously emphasizes the accuracy and cooperation in the ensemble. Dam et al. [20] apply the NCL algorithm to train the neural network ensemble in learning classifier systems, where NCL is shown to improve the generalization of the ensemble.

In [21], Chen and Yao propose the regularized negative correlation learning (RNCL) algorithm with λ set to 1 and make use of Bayesian inference to infer the explicit regularization parameters. In this paper, we formulate the regularized negative correlation learning as a multi-objective evolutionary learning problem. A multi-objective evolutionary algorithm is used to search effectively the best trade-off among these objectives without searching for the combination parameters to weigh these objectives. Compared with RNCL by gradient descent with Bayesian inference in [21], MRNCL often achieves a little better performance by considering an additional weighting coefficient λ of the correlation term. The potential advantages of the multiobjective approach include: It enables us to observe the interaction and trade-off among different objectives; and it enables us to add or remove an objective easily without changing the overall algorithm. However, the benefits come with the price, more computational time to train MRNCL.

3 Multiobjective Regularized Negative Correlation Learning

This section analyzes NCL and its potential risk of overfitting. In order to address the problem, a multiobjective regularized NCL algorithm is proposed.

3.1 Negative Correlation Learning

NCL introduces a correlation penalty term to the error function of each individual network in the ensemble so that all the networks can be trained interactively on the same training data set.

Given the training set $\{\mathbf{x}_n, y_n\}_{n=1}^N$, NCL combines M neural networks $f_i(\mathbf{x})$ to constitute the ensemble.

$$f_{ens}(\mathbf{x}_n) = \frac{1}{M}\sum_{i=1}^{M} f_i(\mathbf{x}_n).$$

In training network f_i, the cost function e_i for network i is defined by

$$e_i = \sum_{n=1}^{N}(f_i(\mathbf{x}_n) - y_n)^2 + \lambda p_i, \tag{1}$$

where λ is a weighting parameter on the penalty term p_i:

$$\begin{aligned} p_i &= \sum_{n=1}^{N}\left\{(f_i(\mathbf{x}_n) - f_{ens}(\mathbf{x}_n))\sum_{j\neq i}(f_j(\mathbf{x}_n) - f_{ens}(\mathbf{x}_n))\right\} \\ &= -\sum_{n=1}^{N}(f_i(\mathbf{x}_n) - f_{ens}(\mathbf{x}_n))^2 . \end{aligned} \tag{2}$$

The first term on the right-hand side of (1) is the empirical training error of network i. The second term p_i is a correlation penalty function. The purpose of minimizing p_i is to negatively correlate each network's error with the error for the rest of the ensemble. The λ parameter controls a trade-off between the training error term and the penalty term. With $\lambda = 0$, we would have an ensemble with each network trained independently. If λ is increased, more and more emphasis would be placed on minimizing the penalty.

Based on the individual error function, Equation (1), the error function for the ensemble can be obtained by averaging these individual network errors e_i. If $\lambda = 1$, the average error E of all the individual networks e_i is obtained as follows:

$$\begin{aligned} & E \\ = \; & \frac{1}{M}\sum_{i=1}^{M} e_i \\ = \; & \frac{1}{M}\sum_{n=1}^{N}\sum_{i=1}^{M}\left\{(f_i(\mathbf{x}_n) - y_n)^2 - (f_i(\mathbf{x}_n) - f_{ens}(\mathbf{x}_n))^2\right\} \\ = \; & \sum_{n=1}^{N}(f_{ens}(\mathbf{x}_n) - y_n)^2. \end{aligned} \tag{3}$$

As explained in [1], minimizing each e_i individually also minimizes E. According to Equation (3), the error function of NCL is equivalent to training a single estimator $f_{ens}(\mathbf{x}_n)$ instead of training each individual network separately. It is also observed that NCL only minimizes the empirical training MSE $\sum_{n=1}^{N}(f_{ens}(\mathbf{x}_n) - y_n)^2$ but does not regularize the complexity of the ensemble. As discussed in Section 1, only minimizing MSE leads to overfitting. In Section 4, we will present the empirical evidence showing that NCL is prone to overfitting[1].

In order to improve the generalization ability of NCL, the next section presents a new multiobjective regularized NCL algorithm.

3.2 Multiobjective Regularized Negative Correlation Learning (MRNCL)

Following the traditional strategy to avoid overfitting, a regularization term is incorporated into the ensemble error function:

$$\begin{aligned} E_{ens} &= \frac{1}{M}\sum_{n=1}^{N}\sum_{i=1}^{M}(f_i(\mathbf{x}_n) - y_n)^2 - \\ &\lambda\frac{1}{M}\sum_{n=1}^{N}\sum_{i=1}^{M}\left(f_i(\mathbf{x}_n) - f_{ens}(\mathbf{x}_n)\right)^2 + \sum_{i=1}^{M}\alpha_i\mathbf{w}_i^T\mathbf{w}_i, \end{aligned} \tag{4}$$

where $\mathbf{w}_i = (w_{i,1}, \cdots, w_{i,n_i})^T$ is the weight vector of neural network i and n_i is the total number of weights in network i.

This regularization term $\sum_{i=1}^{M}\alpha_i\mathbf{w}_i^T\mathbf{w}_i$ is the weight decay [22] term for the entire ensemble. In order to train each neural network with its regularization, we have to decompose the regularization term into M parts, each part for a network. The error function for network i can be obtained as follows:

$$\begin{aligned} e_i &= \frac{1}{M}\sum_{n=1}^{N}(f_i(\mathbf{x}_n) - y_n)^2 - \\ &\frac{\lambda}{M}\sum_{n=1}^{N}(f_i(\mathbf{x}_n) - f_{ens}(\mathbf{x}_n))^2 + \alpha_i\mathbf{w}_i^T\mathbf{w}_i. \end{aligned} \tag{5}$$

Comparing this error function with the cost function of NCL, Equation (1), MRNCL imposes a regularization term on every individual neural network and MRNCL needs to optimize both the correlation coefficient λ and the regularization parameters α_i.

According to Equation (5), the training of an individual neural network in MRNCL involves minimization of three terms: empirical training error term, correlation penalty term and the regularization term. The generalization of ensemble depends on the tradeoff among the three terms and how to balance the tradeoff among the three terms for different problems becomes an important problem. This paper makes use of an evolutionary multiobjective algorithm to balance the tradeoff.

The formulation of MRNCL is not heuristic but based on the Bayesian statistical model. According to Appendix A, MRNCL is an application of the Bayesian framework in an ensemble system. The squared weight decay term, i.e. the regularization term, corresponds to the prior of the weight vector in the ensemble. This is the reason why we only include the squared weight decay term as the regularization term in the multiobjective algorithm. This intrinsic Bayesian characteristic of MRNCL potentially facilitates the incorporation of Bayesian methods in evolutionary multiobjective algorithms to improve the performance of MRNCL.

According to Equation (5), MRNCL defines the following three objectives.

- Objective of Performance $\sum_{n=1}^{N}(f_i(\mathbf{x}_n) - y_n)^2$
  This objective measures the empirical mean square error based on the training set.
- Objective of Correlation $-\sum_{n=1}^{N}(f_i(\mathbf{x}_n) - f_{ens}(\mathbf{x}_n))^2$
  This correlation term measures the amount of variability among the ensemble members and this term can also be treated as the diversity measure [23]. From both theoretical and experimental results it has been shown that, if the individual networks in an ensemble are unbiased, the most effective combination of them occurs when the errors of the individual networks are negatively correlated. This objective encourages individual networks to negatively correlate their errors and thus helps to generate a diverse ensemble.
- Objective of Regularization $\mathbf{w}_i^T\mathbf{w}_i = \sum_j w_j^2$
  Based on the regularization theory [24], the weight decay term [22] is employed to punish large weights. The weight decay term causes the weights to converge to smaller absolute values than they otherwise would. The regularization term helps the generalization ability of a neural network because large weights can hurt generalization in two different ways: a) excessively large weights leading to hidden nodes can cause the output function to be too rough, possibly with near discontinuities. Excessively large weights leading to output nodes can cause wild outputs far beyond the range of the data if the output activation function is not bounded to the same range as the data. b) Large weights can cause excessive variance of the output [25]. The regularization term is beneficial to NCL since large weights are usually connected with near linear dependence among groups of nodes in the network, and NCL would seem to potentiate the appearance of large weights in the ensemble.

3.3 Component Networks and Evolutionary Operators

The component network in the ensemble is a radial basis function (RBF) network. The output of RBF network is

[1] We also notice that NCL performs well in the previous studies and we suppose that those data sets used by NCL in the previous studies do not have large noise.

computed as a linear combination of K basis functions

$$f(\mathbf{x}) = \sum_{k=1}^{K} w_k \phi_k(\mathbf{x}) = \Phi^T \mathbf{w},$$

where $\mathbf{w} = (w_1, \cdots, w_K)^T$ denotes the weight vector in the output layer and $\Phi = (\phi_1, \cdots, \phi_k)$ is the vector of basis functions. The Gaussian basis functions ϕ_k are defined as

$$\phi_k(\mathbf{x}) = \exp(\frac{\| \mathbf{x} - \mu_k \|^2}{2\sigma_k^2}),$$

where μ_k and σ_k denote means and widths of the Gaussian, respectively. The training of RBF network is separated into two steps. In the first step, the means μ_k are initialized with randomly selected data points from the training set and the variances σ_k are determined as the Euclidean distance between μ_k and the closest $\mu_i (i \neq k, i \in \{1, \cdots, K\})$. Then in the second step we perform gradient descent in the regularized error function (weight decay)

$$\min e = \frac{1}{2} \sum_{n=1}^{N} (y_n - f(\mathbf{x}_n))^2 + \alpha \sum_{k=1}^{K} w_k^2. \tag{6}$$

In order to fine-tune the centers and widths, we simultaneously adjust the output weights, the RBF centers and variances. Taking the derivative of Equation (6) with respect to RBF means μ_k and variances σ_k^2 we obtain

$$\frac{\partial e}{\partial \mu_k} = \sum_{n=1}^{N} (f(\mathbf{x}_n) - y_n) \frac{\partial f(\mathbf{x}_n)}{\partial \mu_k}, \tag{7}$$

with $\frac{\partial f(\mathbf{x}_n)}{\partial \mu_k} = w_k \frac{\mathbf{x}_n - \mu_k}{\sigma_k^2} \phi_k(\mathbf{x}_n)$ and

$$\frac{\partial e}{\partial \sigma_k} = \sum_{n=1}^{N} (f(\mathbf{x}_n) - y_n) \frac{\partial f(\mathbf{x}_n)}{\partial \sigma_k}, \tag{8}$$

with $\frac{\partial f(\mathbf{x}_n)}{\partial \sigma_k} = w_k \frac{\|\mathbf{x} - \mu_k\|^2}{\sigma_k^3} \phi_k(\mathbf{x}_n)$. These two derivatives are employed in the minimization of Equation (6) by scaled conjugate gradient descent, where we always compute the optimal output weights in every evaluation of the error function. The optimal output weights $\mathbf{w}$ can be computed in closed form by

$$\mathbf{w} = (\Phi^T \Phi + \alpha I)^{-1} \Phi^T \mathbf{y}, \tag{9}$$

where $\mathbf{y} = (y_1, \cdots, y_n)^T$ denotes the output vector, and I is an identity matrix.

We use RBF networks as the base learners because of the following advantages. 1) If the centers and widths of the basis functions have been chosen, the optimal output weights $\mathbf{w}$ can be efficiently computed in closed form, which means the performance mostly depends on the selection of basis functions. 2) It is reasonable to define crossover and mutation operators in structural-evolving RBF networks by tuning these basis functions.

Based on the above reasons, the crossover and mutation operators for RBF networks are described as follows.

- Crossover Operator
  Since the performance of a RBF network mostly depends on the basis functions, i.e. the centers and the widths, the crossover operator is defined to exchange the basis functions of two RBF networks. Many crossover techniques exist in the literatures, such as one-point crossover, two-point crossover and "cut and splice" crossover [26]. In a RBF network ensemble, as different networks may have different numbers of basis functions, the "cut and splice" approach has been adopted by randomly choosing separate crossover points for two RBF networks and swap their basis functions beyond those points.
- Mutation Operator
  This paper defines two structural mutation operators for RBF networks.
  1) Deleting one basis function. Randomly select one basis function and delete it.
  2) Adding one basis function. The center of the new basis function is determined by a randomly selected data point from the training set. Then, the width of the basis function is chosen as the minimal distance from other centers in this RBF network.

As the crossover and mutation operations may not generate the optimal combination of basis functions, in order to fine-tune the center, width and the weight vector, we simultaneously adjust the output weights, the RBF centers and widths based on Equations (7), (8) and (9). This procedure is also called parametric mutation [18], which modifies the parameters of the network without modifying its topology. This parametric mutation is performed for a few iterations (in our experiments, only one scaled-conjugate-gradient update is employed).

3.4 Multiobjective Evolution of Ensemble and Rank-based Fitness Assignment

In this paper, we will consider a population of individuals who have three objectives and a multiobjective algorithm is employed to select a set of best classifiers with respect to the three objectives. There are a lot of multiobjective algorithms available and the selection of the most suitable algorithm is not a trivial task [27].

In this paper, nondominated sorting with fitness sharing [28] and rank-based fitness assignment are employed. The idea underlying nondominated sorting is the use of a ranking selection method to emphasize current nondominated individuals and a niching method to maintain diversity in the population. Nondominated sorting is based on layers of Pareto front, which ranks the individuals in the population by fronts that lead to fast convergence to Pareto front in the final population and the diversity is maintained by a niching method in the population.

The nondominated sorting algorithm consists of two stages: One is to obtain the nondominated fronts of different layers and every individual of these fronts is assigned an equal dummy fitness. The algorithm used for obtaining the nondominated set of solutions compares the individuals pairwise and marks as dominated all the individuals that are dominated by at least one member of the population. The second is that the members of every front share their fitness [29] with the constraints that none of the members of a front gets a higher fitness than any of the members of the previous front.

Since the dummy fitness assigned by nondominated sorting is raw, sometimes the range of the raw fitness is too large, leading to the situation that some networks reproduce too rapidly, taking over the population too quickly, and preventing the evolutionary algorithm from searching other areas of the solution space. This paper employs rank-based fitness assignment to reassign the fitness to the networks because rank-based fitness assignment behaves in a more robust manner than proportional fitness assignment. In rank-based fitness assignment, the population is sorted according to the raw fitness values. The fitness assigned to each individual depends only on its position in the individuals' ranking and not on the actual raw fitness value.

Assume the best individual in a population ranks the first. The probability of selecting individual i can be calculated as follows [30]:

$$p_i = \frac{1}{M}(\eta_{\max} - (\eta_{\max} - \eta_{\min})\frac{i-1}{M-1})$$

where M is the population size, $\eta_{\max}$ and $\eta_{\min}$ are two parameters.

$$\eta_{\max} \geq \eta_{\min} \geq 0,$$

$$\eta_{\max} + \eta_{\min} = 2.$$

In order to encourage diversity in the population, our algorithm uses the recommended values [30], $\eta_{\max} = 1.1$ and $\eta_{\min} = 0.9$, to have an appropriate selection pressure. Since we compare the children with the parents before being admitted into the population, a large selective pressure will lead some individuals to reproduce too rapidly and thus limits the search ability of the evolutionary algorithm. It is the reason why we use $\eta_{\max} = 1.1$ and $\eta_{\min} = 0.9$ in our paper.

3.5 Algorithm Description

The details about Multiobjective Regularized Negative Correlation Learning (MRNCL) are summarized in Figure 1. Note that in the crossover and mutation operations, the comparison of the child network with the parent network is conducted as follows.

1) Evaluate the three objective values of the child network.
2) Include the child network into the population, then apply non-dominant sorting with fitness sharing algorithm to obtain the raw fitness values[2] of the child network and the parent network.
3) Compare the raw fitness values and keep the better one.

To determine the time to stop evolution, we selected three threshold values ($t_1 = t_2 = t_3 = 10^{-3}$ in this paper) and compare the thresholds with the differences between the old minimal objective values with the new minimal objective values. If all the differences are lower than the thresholds, the algorithm will be terminated. Otherwise, continue. The maximal number of generations is 200.

4 Experimental Studies

In this section we present the experimental results of MRNCL and MNCL, which employs a multiobjective algorithm (two objectives: training error and correlation term) to train negative correlation ensemble. We use MNCL instead of gradient-based NCL because MRNCL uses multiobjective algorithm and it is fair and natural to employ the multiobjective algorithm to train NCL.

In order to compare our algorithm with previous work on multi-objective ensemble learning, we have obtained the source code from Dr. Yaochu Jin and used the same parameters as their algorithm in [12]. This algorithm evolves multi-layer perception (MLP) using two objectives (training error and regularization, i.e. number of connections in MLP) and we name the algorithm as multiobjective neural network (MoNN) in this paper.

In this section, firstly, we present experimental results of MRNCL and other algorithms on four synthetic classification problems in order to understand the behavior of these algorithms. We also design two experiments with different noise levels to study the characteristics of MRNCL, MNCL and MoNN on noisy data. Secondly, we carry out extensive experiments on 16 benchmark classification data sets to compare the performance of MRNCL, MNCL and other classifiers.

4.1 Experimental Setup

In our experiments, radial basis function (RBF) networks are used as the individual classifiers. The number of hidden nodes is randomly selected but restricted in the range of 5 to 15. The parameters in the evolutionary algorithm are set to: the population size M (100), the number of crossover in one generation 20, the number of mutation in one generation 10, the number of generations (200), the parameter of fitness sharing σ_{share} (0.2). These parameters are chosen after some preliminary experiments. They are not meant to be optimal.

In the experiments, we restrict the minimal hidden nodes of RBF networks as 3 in MRNCL and MNCL to discourage improperly simple networks.

4.2 Synthetic Data Sets

As the first experiment, we demonstrate the results of MRNCL on four synthetic data sets in two dimensions in order to illustrate graphically the decision boundaries.

[2] The raw fitness values depend on their ranked layers (fronts) in the population. If they are in the same layer (front), e.g. they are both non-dominant solutions, the one in less crowded area will receive greater fitness according to the fitness sharing algorithm.

```
1. Generate an initial RBF network population: Generate an initial population of $M$ RBF Networks, the number of hidden nodes for each network, $n_i(i = 1, ..., M)$ is specified randomly restricted by the maximal number of hidden nodes. The centers $\mu_{i,k}$ are initialized with randomly selected data points from the training set and the width $\sigma_{i,k}$ are determined as the Euclidian distance between $\mu_{i,k}$ and the closest $\mu_{i,j}(j \neq k, j \in \{1, \cdots, n_i\})$.

2. Train the initial RBF network population and recode the three objective values for each network.

3. Apply nondominated sorting with rank-based fitness assignment algorithm to obtain the rank-based fitness.

4. For 1 to maximal generation

    • Perform a desired number of crossover operations.
      Choose parents based on roulette wheel selection algorithm and perform crossover. Then perform a few number of updates for weight, center and width. Compare the children with parents and keep the better ones.

    • Perform a desired number of mutation operations.
      Choose parents based on roulette wheel selection algorithm and perform mutation. Then perform a few number of updates for weight, center and width. Compare the children with parents and keep the better ones.

    • Apply nondominated sorting algorithm and obtain the rank-based fitness for the new population.

5. Combine all the classifiers in the population to form the ensemble. (The members of ensemble have equal weights)
```

Fig. 1. Multiobjective Regularized Negative Correlation Learning Algorithm

These four data sets are as follows (1) *synth* is generated from mixtures of two Gaussians by [31]. (2) *Overlap* comes from two Gaussian distributions with equal covariance, and is expected to be separated by a linear plane. (3) *Bumpy* comes from two equal Gaussians but being rotated by 90 degrees. Quadratic boundaries are required. (4) *Relevance* is a case where only one dimension of the data is relevant to separating the data.

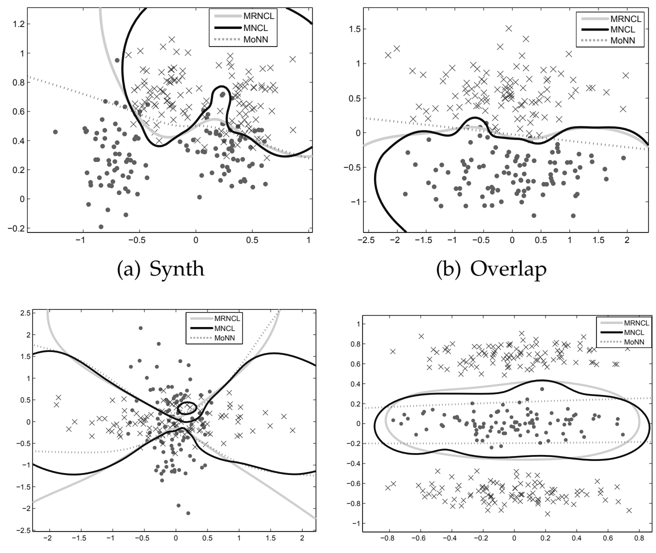
In Figure 2 we present a comparison of MRNCL, MNCL and MoNN. We can observe a similar performance of MRNCL and MNCL in the case of *Relevance*. Since the data set is noise-free, MRNCL and MNCL successfully separate the two classes. In this data set, MoNN generates two linear lines with *unnecessary* training error.

The reason is that MoNN can reduce the regularization by deleting connections and nodes of MLP while it could not always reduce the MSE due to the intrinsic complexity of the data set. In the end, MoNN tends to select the networks with small regularization and thus over-regularizes the ensemble in some cases.

The situation is similar in the case of *Overlap*. Since it is difficult for RBF networks, which are used as component learners in MRNCL and MNCL, to obtain linear decision boundaries [32], MRNCL produces near-linear boundary, while the boundary of MNCL is a little twisty. MoNN generates a linear line according to the expectation to separate the data set.

We observe that MRNCL gives more accurate results in other cases. In the cases of *Synth* and *Bumpy*, MRNCL produces smooth boundary and disregards the outliers in the training points. In the case of *Synth*, MoNN tries to use a *near-linear* boundary to separate the *non-linear* data set consisted of four Gaussians. The generated model is over-regularized and thus degrades the performance. In the case of *Bumpy*, although the decision boundary of MoNN is smooth, it does not generate an appropriate boundary according to expectation. (The optimal boundary is a quadratic one.) Since the noise level is large because of these overlapping points in the case of *Bumpy*, MNCL does not generalize and produces the twisty boundary. In the case of *Synth*, MNCL concentrates on several outliers and generates a corner in the boundary.

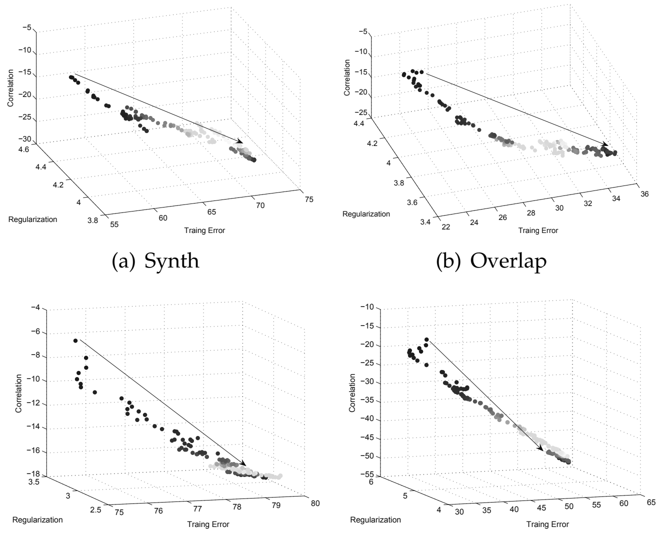
Figure 3 illustrates the mean values of these three objectives in different generations. The arrow points from the first generation to the final generation. According to these figures, MRNCL algorithm tries to minimize the three objectives. However, based on the analysis in Section 3, the empirical training error is negatively correlated with the correlation term. Instead of minimizing the three objectives simultaneously, MRNCL seeks to



(a) Synth (b) Overlap

(c) Bumpy (d) Relevance

Fig. 2. Comparison of MRNCL, MNCL and MoNN on four synthetic classification data sets. Two classes are shown as crosses and dots. The separating lines were obtained by projecting test data over a grid.



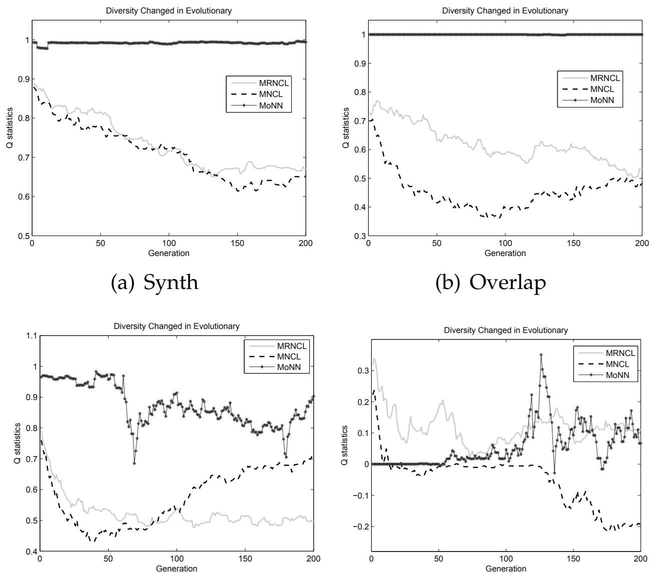
(a) Synth (b) Overlap

(c) Bumpy (d) Relevance

Fig. 3. Illustration of the mean value of these three objectives in different generations. The arrow points from the beginning (Generation = 1) to end (Generation = 100). The gray scale indicates generations.

find a good balance between the two objectives, training error and the correlation term, and MRNCL always minimizes the third objective, the regularization term, in the evolutionary algorithm.

In order to illustrate the effect of negative correlation term in these algorithms, we employ a popular diversity measure, Q statistics[3] [33], and measure the diversity



(a) Synth (b) Overlap

(c) Bumpy (d) Relevance

Fig. 4. Illustration of diversity (measured by a commonly-use diversity measure Q statistics) during the evolution.

in each generation for these three algorithms. The results are presented in Figure 4. In this figure, MRNCL and MNCL encourage diversity[4] in the evolution while MoNN does not pay much attention to increasing diversity in the evolution. In Figure 4(d), since *Relevance* is a noise-free data set, most networks concentrate on the training error and MRNCL does not need more diversity to classify this data set. This indicates that MRNCL can choose the best tradeoff among these objectives for different problems. Based on the formulation of MRNCL and the observations, the main function of the negative correlation term is to encourage diversity in the ensemble.

The 3D view of the last population is illustrated in Figure 5. The negative correlation between the empirical error term and the correlation term has been confirmed by these figures. The final population distributes a good tradeoff between these three objectives for all the data sets. According to this figure, we also notice that almost 80%-90% of the solutions in the last generation are non-dominated solutions. In Section 4.5, we will present the performance of ensembles by using only the non-dominated solutions in MRNCL.

4.3 Experimental Results on Noisy Data

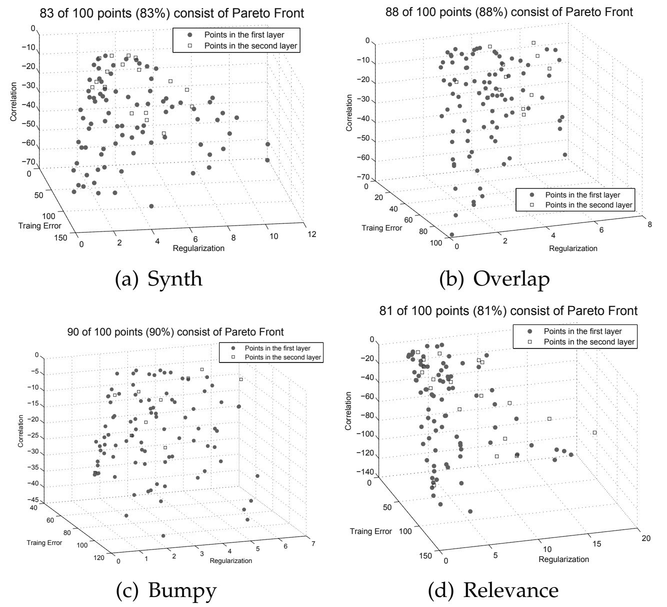
In order to explore the behavior of MRNCL and other multi-objective learning methods with different noise levels, we conduct two additional experiments. In the experiments, we select two data sets: synth and banana[5]
.

To change the noise level, we randomly select different percentages of data points and reverse their labels. We

[3] The definition and calculation of Q statistics have been described in Appendix B.

[4] Small Q statistics indicates large diversity.

[5] http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm



(a) Synth (b) Overlap

(c) Bumpy (d) Relevance

Fig. 5. 3D view of the last population with three objectives: training error, regularization and correlation for four synthetic classification data sets.

run 100 times and report the average results in Figure 6. Figure 6(a) and Figure 6(b) visualize the decision boundaries of MRNCL, MNCL and MoNN with 20% noise.

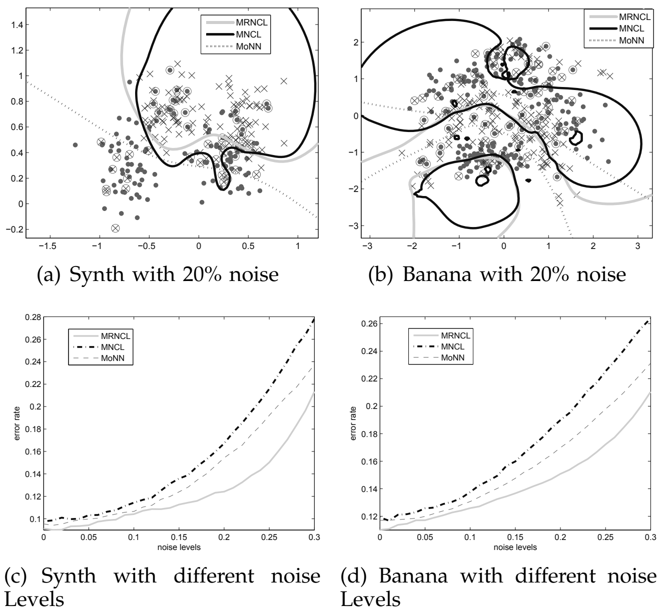
Though the noise level is high, MRNCL produces smooth boundaries. MNCL tries to minimize the training error and it does not generalize well. MoNN generates an over-smooth (inappropriate) decision boundary disregarding the data distribution for the synth data set. Both boundaries are biased from the optimal boundary.

We also plot the curves, Figures 6(c) and 6(d), of classification error vs. noise level for these two data sets. In these two figures, MRNCL is a little better in the beginning, but as the noise level increases, MRNCL significantly outperforms MNCL and MoNN.

In both data sets, MoNN exhibits similar curves in Figures 6(c) and 6(d). In the beginning, the added noise is very small. MoNN over-regularized the ensemble and the obtained performance is worse than MRNCL. When the noise levels are increased (less than 0.1∼0.15), MoNN achieves a similar performance to MRNCL since the large regularization in MoNN helps. However, with the increase of noise levels, MoNN could not be comparable to MRNCL due to the large regularization and small diversity in the obtained ensembles. The results of MRNCL are promising on these classification problems. After the analysis with synthetic data sets, the next section presents the results of the real-world benchmark problems.

4.4 Benchmark Results

In order to evaluate the performance of MRNCL, we compare MRNCL, MNCL and other algorithms on 16 benchmark problems. These data sets used in this paper have been summarized in Table 1.



(a) Synth with 20% noise (b) Banana with 20% noise

(c) Synth with different noise Levels (d) Banana with different noise Levels

Fig. 6. Comparison of MRNCL, MNCL and MoNN on two classification data sets. Two classes are shown as crosses and dots. The separating lines were obtained by projecting test data over a grid. In Figure 6(a) and 6(b), these decision boundaries are MRNCL (gray thick), MNCL (black medium) and MoNN (dotted), respectively. The randomly-selected noise points are marked with a circle. Figure 6(c) and 6(d) show classification error of MRNCL, MNCL and MoNN vs. noise levels on synth and banana data sets. The results are based on 100 runs.

TABLE 1
Summary of Classification Data Sets.

Data Sets	Training Points	Test Points	Input Dimensions
Banana	400	4900	2
Cancer	200	77	9
Diabetics	468	300	8
Solar	666	400	9
German	700	300	20
Heart	170	100	13
Image	1300	1010	18
Ringnorm	400	7000	20
Splice	1000	2175	60
Thyroid	140	75	5
Titanic	150	2051	3
Twonorm	400	7000	20
Waveform	400	4600	21
Magic04	19020	10-fold-cv	10
Satellite	6435	10-fold-cv	36
Spam	4601	10-fold-cv	57

The first 13 data sets have been preprocessed and organized by Rätsch et al.[6] These data sets include one synthetic set (banana) and 12 data sets come from the UCI [34], DELVE[7] and STATLOG repositories. The main difference between the original and Rätsch's data is that

[6] http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm

[7] http://www.cs.toronto.edu/ delve/data/datasets.html

Rätsch converted every problem into binary classes and randomly partitioned every data set into 100 training and testing folds (Splice and Image have only 20 folds in Rätsch's implementation). In addition, every instance is normalized dimension-wise to have zero mean and unit standard deviation. The last three data sets including magic04, satellite and spam are obtained from UCI machine learning repository [34]. The 10-fold cross-validation is used and the results are based on the 100 runs for each dataset.

For Adaboost and Bagging, we combine 200 based learners. Clearly, this number of ensemble size is somewhat arbitrary and may not be optimal. As the base learner we use RBF nets with adaptive centers as described in Section III.C. The parameters of SVM (σ, C) (C is the regularization constant and σ is the width of the RBF-kernel being used) are optimized on the first five training folds of each data set. On each of training folds, a 10-fold-cross validation procedure with grid search will be performed[8]. Finally, the model parameters are computed as the median of the five estimations.

The performance of MRNCL, MNCL, MoNN, Adaboost, Bagging, SVM and RBF network over 100 runs (20 runs for Splice and Image) is summarized in Table 2. The performance of RBF network, Adaboost and SVM for the first 13 data sets is obtained from Rätsch's implementation[9]. We followed the similar methodology for parameter selection and reported the performance of these algorithms for the last three data sets.

According to Table 2, MRNCL outperforms all the other methods in 10 out of 16 data sets, comes second in 6 cases. In comparison with MNCL, MRNCL wins 14 times out of 16 and of them 9 wins are statistically significant. In the results, MNCL performs well in the cases with little noise: Image, Thyroid and Twonorm, which are all synthetic data with little noise (see the lower error rates). The observation validates that MNCL achieves better results when noise is small.

Adaboost with 200 learners seems to overfit the noise and it does not achieve comparable performance to other methods. SVM with cross validation search obtains a good performance in these algorithms as it ranks 1st place on 4 out of 16 data sets.

Based on the empirical results, we notice that MoNN is prone to generate "simple" neural networks with small regularization. The reason is that there are only two objectives, regularization and mean square error (MSE), in MoNN and the regularization term can be reduced to an arbitrary small value while MSE could hardly do it due to the intrinsic complexity of the data set, especially when the data lies in a high-dimensional space.

MoNN uses an elitist non-dominated sorting genetic algorithms (NSGA-II) algorithm [35], which will archive non-dominated solutions in this evolution. In this case, the final population is consisted of many individuals with very small regularization but large MSE, leading the pareto front to be biased to include more individuals with smaller regularization but large error. This is the reason of performance degradation in both synthetic and benchmark experiments for MoNN.

Our algorithm makes use of an additional objective, negative correlation term, to encourage diversity in the population. This objective encourages these networks to behave differently in the population, and thus alleviates the above problem in MoNN. In the experiments, we further restrict the minimal hidden nodes in RBF networks as 3 in MRNCL and MNCL to further discourage improperly simple networks.

Note that in [12], a regression problem, three-dimensional Ackley function, is employed to validate the algorithm ability. The Ackley function is a continuous, multimodal function obtained by modulating an exponential function with a cosine wave of moderate amplitude. For this kind of data sets with lots of local maxima and minima, "simple" networks are beneficial to address the overfitting problems.

4.5 Ensemble Size and Non-dominated Solutions

In the previous section, we have reported the performance of MRNCL using all the networks in the population. It is suggested in [36], [37] that it might be better to use a subset of available neural networks than to use all. For this purpose, we will use the non-dominated solutions to construct a neural network ensemble.

Figure 5 shows the non-dominated solutions in the population and more than 80% of the solutions are non-dominated. We also report the performance and the ensemble size of MRNCL using the entire population and the non-dominated solutions only on the 16 benchmark data sets in Table 3, respectively.

According to this table, the performance of the ensemble using non-dominated solutions is a little better than that using all solutions in the population. The ensemble size is reduced to almost 80%-90% of the population size by adopting the non-dominated solutions. Based on these results, adopting the non-dominated solutions instead of the entire population can improve the ensemble performance and reduce its size. It also gives a potential direction to improve our work by selecting a subset of the non-dominated solutions to constitute a smaller ensemble.

As we observed in Figure 5, although some individuals in the population have large regularization and some have small regularization, most individuals will have appropriate regularization according to different problems. For example, since the data are linear-separable in the relevance data set (Figure 5(d)), the ensemble does not need large regularization. In this case, most of the networks have small regularization. Therefore, an ensemble of all networks would have an appropriate regularization.

[8] The ranges of cross validation search for SVM are $C \in \{1, 10, \cdots, 100\}$ and $\theta \in \{0.1, 0.3, \cdots, 10\}$ (The data has been normalized to unit standard deviation) in both synthetic data sets and benchmark data sets.

[9] http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm

TABLE 2
Comparison among 7 methods on 16 benchmark Data Sets: MRNCL, MNCL, MoNN, Adaboost, Bagging, support vector machine and single RBF classifier. The generalization error in % (standard deviation) on 16 data sets (best method in bold face) has been reported. A win-loss-tie summarization based on mean value and t test (significance level $\alpha = 0.05$) is attached at the bottom of the table. The performance is based on 100 runs (20 runs for Splice and Image). MRNCL gives the best overall performance.

	MRNCL	MNCL	MoNN	Adaboost	Bagging	SVM	RBF
Banana	**10.7±0.7**	11.2±0.7	11.0±0.7	12.3±0.7	11.2±0.7	11.5±0.7	10.8±0.6
Cancer	26.4±4.6	28.2±4.8	27.4±4.9	30.4±4.7	27.3±4.6	**26.0±4.7**	27.6±4.7
Diabetics	**23.2±1.7**	25.3±1.9	24.0±1.7	26.5±2.3	24.2±1.8	23.5±1.7	24.3±1.9
German	24.2±2.1	26.1±2.3	24.2±2.2	27.5±2.5	24.9±2.5	**23.6±2.1**	24.7±2.4
Heart	**15.6±3.0**	16.2±3.1	16.7±3.1	20.3±3.4	17.2±3.4	16.0±3.3	17.6±3.3
Image	**2.6±0.7**	**2.6±0.7**	2.7±0.7	2.7±0.7	3.0±0.6	3.0±0.6	3.3±0.6
Ringnorm	**1.6±0.2**	1.9±0.2	1.7±0.2	1.9±0.3	**1.6±0.2**	1.7±0.1	1.7±0.2
Solar	33.1±1.7	33.4±1.5	33.1±1.8	35.7±1.8	34.1±1.9	**32.4±1.8**	34.4±2.0
Splice	**9.9±0.6**	10.2±0.5	10.1±0.8	10.1±0.5	10.0±0.5	10.9±0.7	10.0±0.8
Thyroid	4.4±2.1	**4.3±2.1**	**4.3±2.3**	4.4±2.2	4.4±2.1	4.8±2.2	4.5±2.1
Titanic	22.3±1.1	**22.2±1.3**	23.6±1.6	22.6±1.2	22.8±1.2	22.4±1.0	23.3±1.3
Twonorm	**2.3±0.1**	2.4±0.1	2.4±0.1	3.0±0.3	2.8±0.2	3.0±0.2	2.9±0.3
Waveform	10.4±0.6	10.6±0.7	10.8±0.8	10.8±0.6	10.2±0.5	**9.9±0.4**	10.7±1.1
Magic04	**12.0±0.6**	15.2±0.8	12.8±0.6	13.1±0.8	14.0±0.6	13.1±0.6	15.6±0.8
Satellite	**9.8±1.1**	12.4±1.3	11.5±1.2	10.9±1.3	11.8±1.1	10.6±1.0	12.8±1.3
Spam	**5.9±1.1**	7.4±1.3	7.0±1.2	6.3±1.1	7.2±1.1	7.0±1.0	8.0±1.2
W-L-T	-	1-14-1	1-15-0	0-15-1	0-14-2	4-12-0	0-16-0
Significant	-	0-9-7	0-6-10	0-10-6	0-5-11	1-6-9	0-8-8

TABLE 3
Comparison of the performance and the ensemble size of MRNCL using the entire population and the non-dominated solutions on 16 benchmark Data Sets.

	Population	Size	Pareto	Size
Banana	10.7±0.7	100	10.6±0.7	82.3±4.7
Cancer	26.4±4.6	100	26.6±4.3	79.6±6.1
Diabetics	23.2±1.7	100	23.0±1.8	87.6±5.2
German	24.2±2.1	100	24.0±1.9	90.1±3.6
Heart	15.6±3.0	100	15.8±3.1	80.3±5.8
Image	2.6±0.7	100	2.5±0.6	84.9±4.7
Ringnorm	1.6±0.2	100	1.7±0.2	84.7±5.1
Solar	33.1±1.7	100	32.8±1.6	79.6±7.9
Splice	9.9±0.6	100	9.6±0.7	81.5±5.3
Thyroid	4.4±2.1	100	4.3±2.2	75.3±6.2
Titanic	22.3±1.1	100	22.1±1.2	78.4±8.4
Twonorm	2.3±0.1	100	2.4±0.1	86.7±3.2
Waveform	10.4±0.6	100	10.4±0.7	81.6±3.9
Magic04	12.0±0.6	100	12.1±0.7	73.6±9.2
Satellite	9.8±1.1	100	9.5±1.3	80.1±4.9
Spam	5.9±1.1	100	6.0±1.0	86.1±5.8

TABLE 4
The mean rank of MRNCL, MNCL, MoNN and SVM based on 16 data sets.

Algorithm	MRNCL	MNCL	MoNN	SVM
Mean Rank	1.5313	3.0938	2.6875	2.6875

4.6 Statistical Comparisons over Multiple Data Sets

Statistical tests on multiple data sets for multiple algorithms are preferred for comparing different algorithms over multiple data sets [38]. In this section, we will conduct statistical tests over multiple data sets by using the Friedman test [39] with the corresponding post-hoc tests.

The Friedman test is a non-parametric equivalence of the repeated-measures analysis of variance (ANOVA) under the null hypothesis that all the algorithms are equivalent and so their ranks should be equal [39], [40]. This paper uses an improved Friedman test proposed by Iman and Davenport [40].

The Friedman test [39] is carried out to test *whether all the algorithms are equivalent*. If the test result rejects the null hypothesis, i.e. these algorithms are equivalent, we can proceed to a post-hoc test. The power of the post-hoc test is much greater when all classifiers are compared with a control classifier and not among themselves. We do not need to make pairwise comparisons when we in fact only test whether a newly proposed method is better than the existing ones.

Based on this point, we would like to choose MRNCL as the control classifier to be compared with. Since the baseline classification algorithms are not comparable to other algorithms, this section will analyze only three algorithms: MNCL, MoNN and SVM against the control algorithm MRNCL.

The Bonferroni-Dunn test [41] is used as post-hoc tests when all classifiers are compared to the control classifier. The performance of pairwise classifiers is significantly different if the corresponding average ranks[10] differ by at least the critical difference

$$CD = q_\alpha \sqrt{\frac{j(j+1)}{6T}}, \tag{10}$$

where j is the number of algorithms, T is the number of data sets and critical values q_α can be found in [38]. For

[10] We rank these algorithms based on the metric on each data set and record the ranking of each algorithm as 1, 2 and so on. Average ranks are assigned in case of ties. The average rank of one algorithm is obtained by averaging over all of data sets. Please refer to Table 4 for the mean rank of these algorithms.

example, when $j = 4$, $q_{0.05} = 2.394$, where the subscript 0.05 is the significance level.

Table 4 lists the mean rank of these algorithms using different training algorithms. Table 5 gives the Friedman test results. Since we employ the same threshold 0.05 for these ensemble training algorithms, the critical difference $CD = 1.0927$, where $j = 4$ and $T = 16$, is the same for these algorithms. Several observations can be made from our results.

Firstly, the null hypothesis that all the algorithms are equivalent is rejected for each algorithm in Table 4. Secondly, the differences between MRNCL and other algorithms including MNCL, MoNN and SVM are greater than the critical difference, so the differences are significant, which means the MRNCL is significantly better than these algorithms in this current experimental setting.

There are at least three reasons why the performance of our algorithm is better than the performance of others.

1) Effective parameters of RBF ensemble, obtained by the evolutionary algorithm, improve the performance of the ensemble. The performance of RBF networks mostly depends on the number of basis functions and the selection of centers and the widths in these basis functions. In RBF network ensemble, better performance is achieved when these individuals cooperate with each other. How to select these parameters is crucial for the ensemble. In most of the existing ensemble algorithms, we have to tune these parameters manually, suffering from the tedious trial-and-error process in practice. However, our algorithm can determine these parameters automatically according to different problems given that you specify some parameters for the evolutionary algorithm. We do not observe great sensitivity to GA parameters, such as the population size, crossover rate and generation number, within their commonly accepted ranges.
2) The multiobjective algorithm promotes the accuracy, diversity and regularization in the ensemble. The accuracy and diversity are considered as two important factors in ensemble algorithms. Our analysis reveals that besides these two factors, regularization of ensemble is another important part for ensemble performance. The regularization term controls the complexity of ensemble and improves the performance of ensemble against noise. The existing ensemble algorithms either focus on accuracy, e.g. Adaboost, and/or diversity, e.g. Bagging and NCL. In order to take all these terms into consideration, our strategy adopts a multiobjective algorithm to generate the accurate, diverse and regularized ensemble.
3) Our algorithm uses a multiobjective algorithm to construct an ensemble to balance the tradeoff for different problems. There is no need to weigh objectives by selecting the coefficients.

4.7 Computational Complexity and Running Time

Based on the algorithm in Figure 1, the major running time of MRNCL is consumed by the training of RBF networks. In the initialization step, we need to train each component RBF network, totally M, in the population. In each generation, indicated by G, we need to train $2C+u$ RBF networks, where C is the number of crossover in one generation and u is the number of mutation in one generation. In total, we need to train $M + (2C + u)G$ RBF networks in MRNCL. To train each RBF network after performing crossover and mutation, we only need to perform a few scaled-conjugate-gradient updates (in our experiments, only one scaled-conjugate-gradient update is employed) to simultaneously adjust the output weights and the RBF centers and widths. This can be performed quickly.

In MRNCL and MNCL, we perform three scaled-conjugate-gradient (SCG) updates on each RBF network in the initialization step. Since only one SCG update is employed to simultaneously adjust the output weights and the RBF centers and widths after crossover and mutation, the total number of SCG is $3M + (2C + u)G = 10,300$ given that these parameters are set with the size of population $M = 100$, the number of crossover in one generation $C = 20$, the number of mutation in one generation $u = 10$ and the number of generation $G = 200$ in this algorithm. The number of training epoches is similar for different data sets.

In MoNN [12], the RProp+ algorithm [42] is employed to train neural networks. The population size is 100 the maximal generation is 200. In each generation, MoNN generates a new population of 100 offspring. With this parameter setting, in total, it will call $100 \times 200 = 20,000$ RProp+ algorithm. Note that RProp+ is implemented in C++ and the training is faster than SCG.

Table 6 shows the average running time of MRNCL, MNCL and MoNN over 100 runs. The running time is the "training and evaluation" time (in seconds) of these algorithm including MRNCL, MNCL and MoNN in one run, which includes the execute time of the algorithm in Figure 1 (for MRNCL) and the evaluation time to calculate the error statistic on test set. The running time in Table 6 is averaged over 100 runs. Note that the running time is not CPU time since there are two CPUs in our computational platform.

The computational environment is windows XP with Intel Core 2 Duo 1.66G CPU and 2G RAM. The algorithms are implemented in Matlab and C language, where C language is used for MATLAB MEX files to implement the RBF network training algorithm and non-dominated sorting algorithm. From Table 6, MRNCL and MNCL consume similar computational time and MoNN takes less time because MoNN is implemented by C++ and our algorithm is programmed by MATLAB. In fact, if MRNCL is implemented by C++, it will cost less time than MoNN since the RBF training is faster than MLP.

TABLE 5
Friedman tests with the corresponding post-hoc tests, Bonferroni-Dunn, to compare estimators and classifiers for multiple data sets. The significance level is 0.05, and $q_{0.05} = 2.394$.

Algorithms	Friedman test	$\text{CD}_{0.05}$	MNCL	MoNN	SVM
Test Results	0.000	1.0927	1.5625	1.1562	1.1562

TABLE 6
Running Time of MRNCL, MNCL and MoNN using different data sets in seconds. Results are averaged over 100 runs.

Time(s)	Banana	Cancer	Diabetics	Solar	German	Heart	Image	Ringnorm
MRNCL	81.3	29.4	68.6	83.0	121.5	26.7	236.2	183.7
MNCL	80.6	28.6	62.7	84.6	126.1	26.3	220.0	192.6
MoNN	38.7	13.1	24.7	37.2	48.6	12.9	92.5	87.9
Time(s)	Splice	Thyroid	Titanic	Twonorm	Waveform	Magic04	Satellite	Spam
MRNCL	288.7	19.4	45.6	193.8	145.7	2316.7	1674.6	1217.4
MNCL	283.6	19.1	46.9	186.8	132.1	2289.7	1617.2	1206.8
MoNN	114.6	5.4	14.7	76.4	53.1	672.8	432.6	386.2

5 CONCLUSIONS

This paper analyzes NCL and points out that NCL is prone to overfitting the noise because NCL does not regularize its complexity. We have proposed a new multiobjective regularized NCL (MRNCL), which incorporates an additional regularization term for NCL. This paper adopts a multiobjective algorithm and treats the training MSE, regularization and correlation as three separate objectives.

In MRNCL, the crossover and mutation operators are defined to vary the structure of RBF networks. The nondominated sorting algorithm with fitness sharing and rank-based fitness assignment are employed to promote diversity in MRNCL.

Several experiments have been carried out to evaluate MRNCL. The experiments on four synthetic classification problems demonstrate the behavior of MRNCL, MNCL and MoNN. These results showed clearly about the advantages and effectiveness of MRNCL due to its regularization term for noisy data. The higher the noise level, the better MRNCL's performance is in comparison with MNCL. The experiments also show that MoNN tends to over-regularize the ensemble and thus degrade the performance. The experiments on two additional classification problems with different noise levels demonstrate further that MRNCL achieves better performance than MNCL, especially when the noise is nontrivial in data sets.

Then, we carry out extensive experiments on 16 benchmark classification data sets to compare the performance of MRNCL, MNCL and other state-of-the-art algorithms. MRNCL performs quite favorably on these data sets. The three major reasons why the performance of our algorithm was so good are given in this paper. 1) Effective parameters of the RBF ensemble, obtained by the evolutionary algorithm, improve the performance of ensembles. 2) The multiobjective algorithm promotes the accuracy, diversity and regularization in the ensemble. 3) The best tradeoff of the three objectives can be achieved by constructing an ensemble generated by multiobjective algorithm.

In [37], Chen et al. demonstrated that the performance of the ensemble can be improved by selecting a small subset of ensemble members using a probabilistic ensemble pruning method. It is one of our future work to incorporate the ensemble selection/pruning algorithms into the multiobjective ensemble learning algorithms to generate more compact ensembles.

Other future work for this study includes a more in-depth study of different evolutionary operators and other fitness ranking methods used in the multi-objective evolutionary algorithms.

ACKNOWLEDGMENT

This work is partially supported by a Dorothy Hodgkin Postgraduate Scholarship to the first author and an EPSRC grant (GR/T10671/01) to the second author. The authors also thank Dr. Yaochu Jin for providing the source code on regularizing neural networks using multi-objective evolutionary algorithms [12] for comparison with our algorithm.

APPENDIX A
BAYESIAN INTERPRETATION OF MRNCL

This subsection describes the probabilistic interpretation of MRNCL and the function of the regularization term.

Given the training set $D = \{\mathbf{x}_n, y_n\}_{n=1}^{N}$, we follow the standard probabilistic formulation and assume that the targets are sampled from the model with additive noise:

$$y_n = f_{ens}(\mathbf{x}_n) + e_n = \frac{1}{M}\sum_{i=1}^{M} f_i(\mathbf{x}_n) + e_n,$$

where e_n is independent sample from some noise process which is further assumed to be mean-zero Gaussian with variance β^{-1}.

According to the Bayesian theorem, given the hyperparameters $\mu = (\mu_1, \cdots, \mu_M)$[11] and β. We obtain the weigh parameters $\mathbf{w} = (\mathbf{w}_1^T, \cdots, \mathbf{w}_M^T)^T$ by maximizing the posterior $P(\mathbf{w} \mid D)$.

$$P(\mathbf{w} \mid D) = \frac{P(D \mid \mathbf{w},\beta)P(\mathbf{w} \mid \mu)}{P(D \mid \mu, \beta)}, \tag{11}$$

where the probability $P(D \mid \mu, \beta)$ is a normalization factor which is independent of $\mathbf{w}$.

The weight vector of each network $\mathbf{w}_i$ is assumed to have a Gaussian distribution with zero mean and variance μ_i^{-1}. The prior of the weight vector $\mathbf{w}$ is obtained as follows.

$$P(\mathbf{w} \mid \mu) = \prod\nolimits_{i=1}^{M} \left(\frac{\mu_i}{2\pi}\right)^{n_i/2} \exp\left(-\frac{1}{2}\mu_i \mathbf{w}_i^T \mathbf{w}_i\right), \tag{12}$$

where n_i is the total number of weights in network i.

Since noise e_n follows a Gaussian distribution with zero mean and variance β^{-1}, the likelihood $P(D \mid \mathbf{w},\beta)$ can be written as

$$P(D \mid \mathbf{w},\beta) = \prod\nolimits_{n=1}^{N} \left(\frac{\beta}{2\pi}\right)^{1/2} \exp\left(-\frac{\beta}{2}e_n^2\right). \tag{13}$$

We omit all constants and normalization factor, and apply Bayesian rules:

$$P(\mathbf{w} \mid D) \propto \exp\left(-\frac{\beta}{2}\sum_{n=1}^{N} e_n^2\right) \cdot \exp\left(-\sum_{i=1}^{M} \frac{\mu_i}{2}\mathbf{w}_i^T\mathbf{w}_i\right). \tag{14}$$

Taking the negative logarithm, the maximum of the posteriori model parameters $\mathbf{w}$ is obtained as the solution to the following optimization problem:

$$\begin{aligned} & \min J_1(\mathbf{w}) \\ = \; & \beta \sum_{n=1}^{N} e_n^2 + \sum_{i=1}^{M} \mu_i \mathbf{w}_i^T \mathbf{w}_i \qquad (15) \\ = \; & \sum_{n=1}^{N} e_n^2 + \sum_{i=1}^{M} \alpha_i \mathbf{w}_i^T \mathbf{w}_i, \\ = \; & \frac{1}{M}\sum_{n=1}^{N}\sum_{i=1}^{M}(f_i(\mathbf{x}_n) - y_n)^2 - \\ & \frac{1}{M}\sum_{n=1}^{N}\sum_{i=1}^{M}\left(f_i(\mathbf{x}_n) - f_{ens}(\mathbf{x}_n)\right)^2 + \sum_{i=1}^{M}\alpha_i \mathbf{w}_i^T\mathbf{w}_i, \end{aligned}$$

where $\alpha_i = \mu_i/\beta$. We substitute μ_i and β with one parameter α_i because the minimization of J_1 only depends on the ratio $\alpha_i = \mu_i/\beta$.

Comparing Equation (15) with (4), MRNCL is equivalent to maximization of the posterior under Bayesian framework when $\lambda = 1$. The likelihood $P(D \mid \mathbf{w},\beta)$ corresponds to the empirical training error terms and the prior over weight vector $P(\mathbf{w} \mid \mu)$ corresponds to the regularization term. The regularization term penalizes large weights, causing the weights to converge to smaller absolute values than they otherwise would.

TABLE 7
A 2×2 table of the relationship between a pair of classifiers f_i and f_j. N^{ab} is the number of data points for which f_i and f_j are correct/wrong when $a = 1/0$ and $b = 1/0$.

	f_j correct(1)	f_j wrong(0)
f_i correct(1)	N^{11}	N^{10}
f_i wrong(0)	N^{01}	N^{00}

Based on the above analysis, MRNCL is an application of Bayesian framework in ensemble system.

Appendix B
Q statistics

Yule's Q statistics [33] computes the "coefficient of association" for two classifiers, f_i and f_j, is

$$Q_{i,j} = \frac{N^{11}N^{00} - N^{01}N^{10}}{N^{11}N^{00} + N^{01}N^{10}}.$$

Q statistics varies between -1 and 1. For statistically independent classifiers, the expectation of $Q_{i,j}$ is 0. Classifiers that tend to classify the same objects correctly will have positive values of Q, and those which commit errors on different objects will produce negative Q value. For an ensemble of M classifiers, the average Q statistics over all pairs of classifiers is,

$$Q = \frac{2}{M(M-1)} \sum_{i=1}^{M-1} \sum_{j=i+1}^{M} Q_{i,j}. \tag{16}$$

References

[1] Y. Liu and X. Yao, "Ensemble learning via negative correlation," *Neural Networks*, vol. 12, no. 10, pp. 1399–1404, 1999.

[2] ——, "Simultaneous training of negatively correlated neural networks in an ensemble," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 29, no. 6, pp. 716–725, 1999.

[3] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 10, pp. 993–1001, 1990.

[4] X. Yao, M. Fischer, and G. Brown, "Neural network ensembles and their application to traffic flow prediction in telecommunications networks," in *Proceedings of International Joint Conference on Neural Networks*, 2001, pp. 693–698.

[5] Y. Liu, X. Yao, and T. Higuchi, "Evolutionary ensembles with negative correlation learning," *IEEE Transaction on Evolutionary Computation*, vol. 4, no. 4, pp. 380–387, 2000.

[6] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[7] R. E. Schapire, "A brief introduction to boosting," in *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, 1999, pp. 1401–1406.

[8] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 832–844, 1998.

[9] H. A. Abbass, "A memetic pareto evolutionary approach to artificial neural networks," in *Proceedings of the fourteenth Australian Joint Conference on Artificial Intelligence*, vol. 2256, 2000, pp. 1–12.

[11] μ_i, $i = 1, 2, \cdots M$, is the inverse variance of the Gaussian distribution of weights for network i.

[10] R. McKay and H. A. Abbass, "Anti-correlation: A diversity promoting mechanisms in ensemble learning," *The Australian Journal of Intelligent Information Processing Systems*, vol. 3, no. 4, pp. 139–149, 2001.

[11] H. A. Abbass, "Speeding up backpropagation using multiobjective evolutionary algorithms," *Neural Computation*, vol. 15, no. 11, pp. 2705–2726, 2003.

[12] Y. Jin, T. Okabe, and B. Sendhoff, "Neural network regularization and ensembling using multi-objective evolutionary algorithms," in *Proceedings of IEEE Congress on Evolutionary Computation (CEC'04)*, 2004, pp. 1–8.

[13] M. M. Islam, X. Yao, and K. Murase, "A constructive algorithm for training cooperative neural network ensembles," *IEEE Transaction on Neural Networks*, vol. 14, no. 4, pp. 820–834, 2003.

[14] G. Brown, J. Wyatt, and P. Tino, "Managing diversity in regression ensembles," *Journal of Machine Learning Research*, vol. 6, pp. 1621–1650, 2005.

[15] A. Chandra and X. Yao, "Evolving hybrid ensembles of learning machines for better generalisation," *Neurocomputing*, vol. 69, no. 7-9, pp. 686–700, 2006.

[16] ——, "Ensemble learning using multi-objective evolutionary algorithms," *Journal of Mathematical Modelling and Algorithms*, vol. 5, no. 4, pp. 417–445, 2006.

[17] L. S. Oliveira, M. Morita, R. Sabourin, and F. Bortolozzi, "Multi-objective genetic algorithms to create ensemble of classifiers," in *Proceedings of the Third International Conference on Evolutionary Multi-Criterion Optimization*, vol. 87, 2005, pp. 592–606.

[18] N. García, C. Hervás, and D. Ortiz, "Cooperative coevolution of artificial neural network ensembles for pattern classification," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 3, pp. 271–302, 2005.

[19] H. Chen and X. Yao, "Evolutionary random neural ensemble based on negative correlation learning," in *Proceedings of IEEE Congress on Evolutionary Computation (CEC'07)*, 2007, pp. 1468–1474.

[20] H. H. Dam, H. A. Abbass, C. Lokan, and X. Yao, "Neural-based learning classifier systems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 1, pp. 26–39, 2008.

[21] H. Chen and X. Yao, "Regularized negative correlation learning for neural network ensembles," *IEEE Transactions on Neural Networks*, 2009, accepted.

[22] A. Krogh and J. A. Hertz, "A simple weight decay can improve generalization," in *Advances in Neural Information Processing Systems*, vol. 4, 1992, pp. 950–957.

[23] A. Krogh and J. Vedelsby, "Neural network ensembles, cross validation, and active learning," in *Advances in Neural Information Processing Systems 7*, Denver, Colorado, USA, 1995, pp. 231–238.

[24] V. N. Vapnik, *The Nature of Statistical Learning Theory*. New York: Springer-Verlag, 1995.

[25] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural Computation*, vol. 4, no. 1, pp. 1–58, 1992.

[26] D. Goldberg, K. Deb, H. Kargupta, and G. Harik, "Rapid, accurate optimization of difficult problems using fast messy genetic algorithms," in *Proceedings of the Fifth International Conference on Genetic Algorithms*, 1993, pp. 56–64.

[27] E. Zitzler, M. Laumanns, L. Thiele, C. M. Fonseca, and V. G. d. Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 117–132, 2003.

[28] N. Srinivas and K. Deb, "Multiobjective function optimization using nondominated sorting genetic algorithms," *Evolutionary Computation*, vol. 2, no. 3, pp. 221–248, 1995.

[29] P. Darwen and X. Yao, "Every niching method has its niche: fitness sharing and implicit sharing compared," in *Proceedings of Parallel Problem Solving from Nature (PPSN) IV*, vol. 1141, Berlin, Germany, 1996, pp. 398–407.

[30] J. Baker, "Adaptive selection methods for genetic algorithms," in *Proceedings of International Conference on Genetic Algorithms and Their Applications*, 1985, pp. 100–111.

[31] B. D. Ripley, *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

[32] R. B. Gramacy and H. K. H. Lee, "Gaussian processes and limiting linear models," Department of Applied Mathematics and Statistics, University of California, Santa Cruz, Technical Report ams2005-01, 2005.

[33] U. Yule, "On the association of attributes in statistics," *Philosophical Transactions of the Royal Society of London. Series A*, vol. 194, pp. 257–319, 1900.

[34] A. Asuncion and D. Newman, "UCI machine learning repository," 2007. [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html

[35] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii," in *Parallel Problem Solving from Nature*, vol. VI, 2000, pp. 849–858.

[36] X. Yao and Y. Liu, "Making use of population information in evolutionary artificial neural networks," *IEEE Transactions on Systems, Man and Cybernetics, Part B: Cybernetics*, vol. 28, pp. 417–425, 1998.

[37] H. Chen, P. Tiňo, and X. Yao, "Predictive ensemble pruning by expectation propagation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 7, pp. 999–1013, 2009.

[38] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine learning research*, vol. 7, pp. 1–30, 2006.

[39] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *Journal of the American Statistical Association*, vol. 32, pp. 675–701, 1937.

[40] R. L. Iman and J. M. Davenport, "Approximations of the critical region of the friedman statistic," *Communications in Statistics*, pp. 571–595, 1980.

[41] O. J. Dunn, "Multiple comparisons among means," *Journal of the American Statistical Association*, vol. 56, pp. 52–64, 1961.

[42] C. Igel and M. Hüsken, "Improving the rprop learning algorithm," in *Proceedings of the 2nd ICSC International Symposium on Neural Computation*, 2000, pp. 115–121.

Huanhuan Chen received the B.Sc. degree from the University of Science and Technology of China, Hefei, China, in 2004, and Ph.D. degree, sponsored by Dorothy Hodgkin Postgraduate Award (DHPA), in computer science at the University of Birmingham, Birmingham, UK, in 2008. His PhD thesis "Diversity and Regularization in Neural Network Ensembles " has received the 2009 CPHC/British Computer Society (BCS) Distinguished Dissertations Award as the runner up.

He is a Research Fellow with the Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA) in School of Computer Science, University of Birmingham. His research interests include statistical machine learning, data fusion, data mining and evolutionary computation.

Dr. Chen is the recipient of the BCS Distinguished Dissertations Award as the runner up (2009), the Value in People (VIP) award from The Wellcome Trust (2009), Dorothy Hodgkin Postgraduate Award (DHPA) from EPSRC (2004) and the Student Travel Grant for the 2006 Congress on Evolutionary Computation (CEC06).

Xin Yao (M'91-SM'96-F'03)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 (ANU), Canberra, in 1990, and continued his work on simulated annealing and evolutionary algorithms. He joined the Knowledge-Based Systems Group, CSIRO (Commonwealth Scientific and Industrial Research Organisation) 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, University College, University of New South Wales (UNSW), Australian Defence Force Academy (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, U.K., as a Professor (Chair) of Computer Science on the April Fool's Day in 1999. Currently, he is the Director of the Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA) and a Changjiang (Visiting) Chair Professor (Cheung Kong Scholar) at the University of Science and Technology of China, Hefei. He was the Editor-in-Chief of the IEEE Transactions on Evolutionary Computation (2003-08), an associate editor or editorial board member of twelve 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 300 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.