

Regularized Negative Correlation Learning for Neural Network Ensembles

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

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

Abstract

Negative Correlation Learning (NCL) [1], [2] is a neural network ensemble algorithm which introduces a correlation penalty term into the cost function of each individual network so that each neural network minimizes its mean-square-error (MSE) together with the correlation of the ensemble. This paper analyzes NCL and finds that the training of NCL corresponds to training the entire ensemble as a single learning machine which only minimizes the MSE error without regularization. This analysis explains that NCL is prone to overfitting the noise in the training set. The paper analyzes this problem and proposes the regularized negative correlation learning (RNCL) algorithm which incorporates an additional regularization term for the whole ensemble. RNCL decomposes the ensemble's training objectives, including MSE and regularization, into a set of sub-objectives, and each sub-objective is implemented by an individual neural network. In this paper, we also provide a Bayesian interpretation for RNCL and provide the automatical regularization parameters optimization algorithm based on Bayesian inference. The RNCL formulation is applicable to any nonlinear regression estimator minimizing the MSE. The experiments on synthetic data as well as real-world data sets demonstrate that RNCL achieves better performance than NCL, especially when the noise level is non-trivial in the data set.

I. 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 because it is considered as a good approach to improve the generalization ability [3]. The term “ensemble” can be used to describe the paradigm that brings together a number of learning machines for the same task. This technique originates from Hansen and Salamon's work [3], which showed that the generalization ability of a neural network can be significantly improved through ensembling a number of neural networks. Because of the simple and effective properties, ensemble research has become a hot topic in machine learning communities and has already been successfully applied to many areas, for example face recognition [4], character recognition [5], image analysis [6], etc.

Negative Correlation Learning (NCL) [1], [2], a neural network ensemble algorithm developed in the evolutionary computation literature, has shown a number of empirical applications, including regression problems [7] and classification problems [8]. NCL introduces a correlation penalty term into the cost function of each individual network so that each neural network minimizes its MSE error together with the correlation of the ensemble.

From the definition of NCL, it seems that the correlation term in the cost function acts as the regularization term. However, after careful analysis, we observe that the training of NCL corresponds to treating the entire ensemble as a single learning machine by considering only the empirical training error. Although NCL can use the penalty coefficient to explicitly alter the emphasis on the individual MSE and correlation portions of the ensemble, setting a zero or lower coefficient corresponds to independently training the estimators and loses the advantages of NCL. In most cases NCL sets the penalty coefficient to be or close to the particular value which corresponds to treating the entire ensemble as a single learning machine.

By treating the entire ensemble as a single learning machine and minimizing the MSE error without regularization, NCL only reduces the empirical MSE error of the ensemble, but pays less attention to regularizing the complexity of the ensemble. As we know, neural network and other machine learning algorithms which only rely on the empirical MSE error are prone to overfitting the noise [9].

Based on the above analysis, NCL, relying on minimizing the MSE, is prone to overfitting the noise in the training set. The paper analyzes this problem and proposes the theoretical and empirical evidences. In order to solve this problem, this paper proposes regularized negative correlation learning (RNCL) algorithm which incorporates an additional regularization term for the ensemble. Then we describe that the regularization term for the ensemble can be decomposed into different parts for each network.

In this paper, we describe how the training algorithm of NCL is equivalent to training a single learning machine and how RNCL controls the complexity by adding a regularization term. A parameter is used to control the tradeoff between MSE and regularization and the parameter is crucial to ensemble's generalization ability. We provide a Bayesian interpretation for RNCL, and propose an automatical algorithm for regularization parameters optimization based on Bayesian inference. The RNCL algorithm is a generic ensemble algorithm, which is applicable to any nonlinear regression estimator minimizing the MSE, for example MLP and radial basis function (RBF) neural network. In this paper we show an example using MLP as the base estimator.

The rest of this paper is organized as follows. After the background description in Section II, the proposed algorithm is described in Section III. Experimental results and discussion are presented in Section IV. Finally, Section V concludes the paper.

II. BACKGROUND

Ensemble of learning machines [3] is a learning paradigm where a collection of estimators/classifiers are trained for the same task. There have been many ensemble methods studied in the literatures. Generally speaking, these ensemble algorithms can be classified into three categories.

In the first kind of algorithms, each base learner is trained with a subset of training samples, drawn uniformly at random from the original training set. The typical algorithms include Bagging [10] and its variants [11].

The second kind of algorithms introduces weights on the training points and pays more attention to those training samples that are misclassified by former classifiers in the training of next classifier. Adaboosting [12] is a successful algorithm in this kind of algorithms.

The third kind of ensemble algorithms, different from the previous work such as Bagging or Adaboosting, emphasizes interaction and cooperation among the individual base learners in the ensemble. It uses a penalty term in the error function to produce biased individual learners whose errors tend to be negatively correlated. Negative correlation learning [1] [2] is a representative ensemble algorithm in this category. This paper will focus on this kind of ensemble learning algorithm.

NCL was firstly proposed by Liu et. al [1], where negative correlation learning and evolutionary learning (EENCL) are combined to automatically design neural network (NN) ensemble. EENCL emphasizes the cooperation and specialization among different individual NNs during the individual NN design. This provides an opportunity for different NNs to interact with each other to solve one single problem.

Islam et al. [13] take a constructive approach to build the ensemble, starting from a small group of networks with minimal architecture. The networks are all partially trained using negative correlation learning. To our knowledge, the approach can automatically determine weights, network topologies and ensemble membership.

Brown et al. [14] formalized this technique, 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.

Chen et al. [15] proposed to incorporate bootstrap of data, random feature subspace [11] and evolutionary algorithm with negative correlation learning 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.

Since this paper applies regularization technique to NCL, we present some relevant background in the following.

Many recent studies have shown that the generalization ability of a neural network depends on a balance between the empirical training error and the complexity of the network. Bad generalization occurs if the tradeoff is unbalanced. Based on the observations, weight decay [9] was proposed to control the complexity of the network. Weight decay adds a penalty term to the error function. The usual penalty is the sum of squared weights times a decay constant. In a linear model, this form of weight decay is equivalent to ridge regression [16]. The weight decay penalty term causes the weights to converge to smaller absolute values than they otherwise would. The regularization term does help the generalization ability of neural network because large weights can hurt generalization in two different ways: a) Excessively large weights leading to hidden units can cause the output function to be too rough, possibly with near discontinuities; Excessively large weights leading to output units 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 [17].

The generalization ability of the network depends crucially on the decay constant, especially with small training sets. One approach to choose the decay constant is to train several networks with different values of the decay constant and estimate the generalization error for each network and then choose the decay constant that minimizes the estimated generalization error.

Fortunately, there is a superior alternative to estimating the decay constant: Bayesian inference. Bayesian inference makes it possible to efficiently estimate decay constants. Compared with the traditional approach, Bayesian approach is attractive in being logically consistent, simple, and flexible. The application of Bayesian inference to single neural network, introduced by MacKay as a statistical approach to avoid overfitting [18] [19], were successful. Then, the Bayesian technique has been successfully applied to model selection for Least Squares Support Vector Machine [20] and sparse Bayesian Learning, i.e., Relevance Vector Machine [21]. Another important feature of Bayesian inference is that error bars [18] can be assigned to network predictions in regression problems and the probability of prediction [22] can be assigned to classification results that avoids making over-confident predictions in regions of sparse data.

III. REGULARIZED NEGATIVE CORRELATION LEARNING

This section presents negative correlation learning and its potential dangerous of overfitting. In order to resolve the problem, regularized negative correlation learning (RNCL) is proposed. This section also presents the parameter optimization procedures of RNCL by Bayesian inference.

A. Negative Correlation Learning

Negative Correlation Learning (NCL) introduces a correlation penalty term into 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 sets $\{\mathbf{x}_n, y_n\}_{n=1}^N$, NCL combines M neural networks $f_i(\mathbf{x})$ to constitute the ensemble.

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

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

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

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

$$p_i = \sum_{n=1}^N \left\{ (f_i(\mathbf{x}_n) - f_{ens}(\mathbf{x}_n)) \sum_{j \neq i} (f_j(\mathbf{x}_n) - f_{ens}(\mathbf{x}_n)) \right\} = - \sum_{n=1}^N (f_i(\mathbf{x}_n) - f_{ens}(\mathbf{x}_n))^2. \quad (3)$$

The first term in the right-hand side of (2) 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 errors 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 training with plain back propagation, exactly equivalent to training a set of networks independently of one another. If λ is increased, more and more emphasis would be placed on minimizing the penalty.

Based on the individual error function, Equation (2), the error function for the ensemble can be obtained by averaging these individual network error e_i . If $\lambda = 1$, the average error E of all the individual network e_i is obtained as follows:

$$E = \frac{1}{M} \sum_{i=1}^M e_i = \frac{1}{M} \sum_{n=1}^N \sum_{i=1}^M \{ (f_i(\mathbf{x}_n) - y_n)^2 - (f_i(\mathbf{x}_n) - f_{ens}(\mathbf{x}_n))^2 \} = \sum_{n=1}^N (f_{ens}(\mathbf{x}_n) - y_n)^2. \quad (4)$$

From Equation (4), 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 error $\sum_{n=1}^N (f_{ens}(\mathbf{x}_n) - y_n)^2$ but does not regularize the complexity of the ensemble. As discussed in Section I, the error function that only minimizes MSE error is prone to overfitting the noise. In Section IV, we also present the empirical evidences that NCL is prone to overfitting.

In order to improve the generalization ability of NCL, in the next section we propose regularized negative correlation learning (RNCL).

B. Regularized Negative Correlation Learning

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

$$E_{ens} = \sum_{n=1}^N (f_{ens}(\mathbf{x}_n) - y_n)^2 + \sum_{i=1}^M \alpha_i \mathbf{w}_i^T \mathbf{w}_i, \quad (5)$$

where $\mathbf{w}_i = [w_{i,1}, \dots, w_{i,n_i}]^T$ is 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 [9] term for the entire ensemble. In order to train each neural network with its regularization, we have to decompose the regularization term to M parts, each part for each network. The error function for network i can be obtained as follows:

$$e_i = \sum_{n=1}^N (f_i(\mathbf{x}_n) - y_n)^2 - \sum_{n=1}^N (f_i(\mathbf{x}_n) - f_{ens}(\mathbf{x}_n))^2 + \alpha_i \mathbf{w}_i^T \mathbf{w}_i. \quad (6)$$

Comparing this error function with the cost function of NCL, Equation (2), RNCL imposes a regularization term into every individual neural network and RNCL optimizes the regularization parameter α_i instead of the correlation parameter λ .

RNCL is implemented by gradient descent method for training neural networks. According to equation (6), the minimization of the error function of the ensemble is achieved by minimizing the error functions of each individual network. RNCL provides a way to decompose the learning task of the ensemble with regularization into a number of subtasks for each individual network. The algorithm can be summarized in Figure 1.

This paper uses scaled conjugate gradient (SCG) [23] algorithm for fast RNCL training.

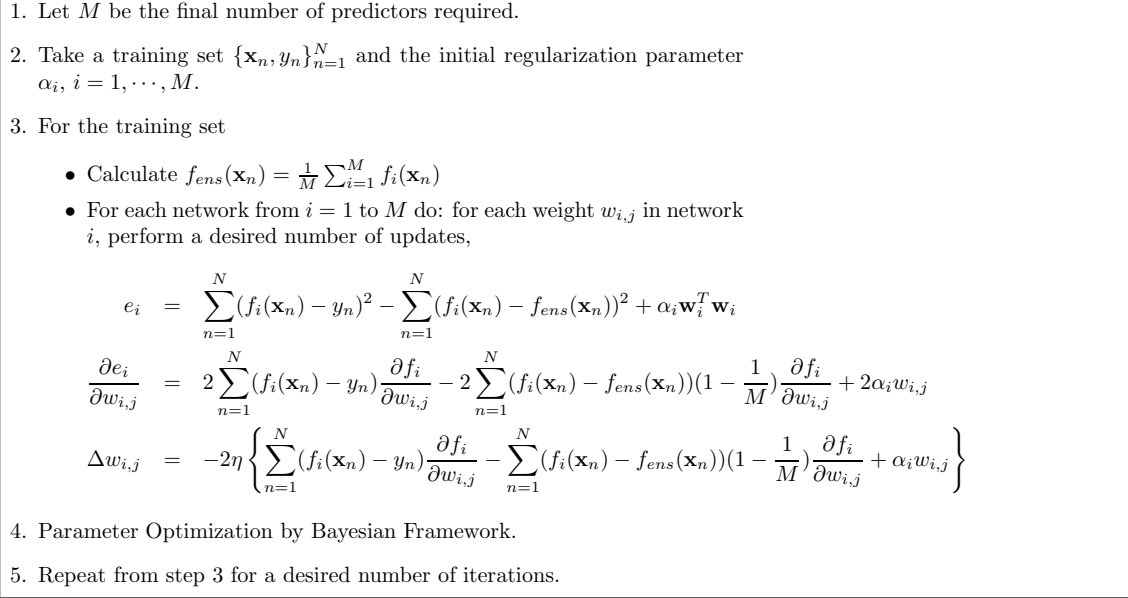


Fig. 1. Regularized Negative Correlation Learning Algorithm

C. Bayesian Interpretation and Regularized Parameter Optimization

This subsection describes the probabilistic interpretation of RNCL, the function of the regularization term and how to optimize these parameters by Bayesian inference. We separate the subsection into two parts: the first part describes the model specification and the probabilistic interpretation of RNCL; The second part describes the procedures to optimize the regularization parameters.

1) *Bayesian Interpretation of RNCL*: 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:

$$y_n = f_{ens}(\mathbf{x}_n) + e_n = \frac{1}{M} \sum_{i=1}^M f_i(\mathbf{x}_n) + e_n, \quad (7)$$

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 $\mu = [\mu_1, \mu_2, \dots, \mu_M]^1$ and β .

¹ $\mu_i, i = 1, 2, \dots, M$, is the inverse variance of the Gaussian distribution of weights for network i .

We obtain the weigh parameters $\mathbf{w} = [\mathbf{w}_1^T, \dots, \mathbf{w}_M^T]^T$ by maximizing the posterior $P(\mathbf{w} | D)$.

$$P(\mathbf{w} | D) = \frac{P(D | \mathbf{w}, \beta) P(\mathbf{w} | \mu)}{P(D | \mu, \beta)}, \quad (8)$$

where the probability $P(D | \mu, \beta)$ 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.

$$P(\mathbf{w} | \mu) = \prod_{i=1}^M \left(\frac{\mu_i}{2\pi} \right)^{n_i/2} \exp \left(-\frac{1}{2} \mu_i \mathbf{w}_i^T \mathbf{w}_i \right), \quad (9)$$

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 $P(D | \mathbf{w}, \beta)$ can be written as

$$P(D | \mathbf{w}, \beta) = \prod_{n=1}^N \left(\frac{\beta}{2\pi} \right)^{1/2} \exp \left(-\frac{\beta}{2} e_n^2 \right). \quad (10)$$

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

$$P(\mathbf{w} | D) \propto \exp \left(-\frac{\beta}{2} \sum_{n=1}^N e_n^2 \right) \cdot \exp \left(-\sum_{i=1}^M \frac{\mu_i}{2} \mathbf{w}_i^T \mathbf{w}_i \right). \quad (11)$$

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

$$\min J_1(\mathbf{w}) = \frac{1}{2} \beta \sum_{n=1}^N e_n^2 + \frac{1}{2} \sum_{i=1}^M \mu_i \mathbf{w}_i^T \mathbf{w}_i. \quad (12)$$

The posterior $P(\mathbf{w} | D)$ can also be written as a Gaussian distribution, please refer to Appendix A for detail.

$$P(\mathbf{w} | D) = \frac{\exp(-J_1(\mathbf{w}))}{\int \exp(-J_1(\mathbf{w})) d\mathbf{w}} = \frac{\exp(-\frac{1}{2}(\mathbf{w} - \mathbf{w}_{MP})^T A(\mathbf{w} - \mathbf{w}_{MP}))}{(2\pi)^{W/2} |A|^{-\frac{1}{2}}}, \quad (13)$$

where A is the hessian matrix of the cost function J_1 , W is the total number of weights in the ensemble and the subscript MP indicates the most probable values.

The error function J_1 is made up of two terms. The first, $\frac{1}{2} \beta \sum_{n=1}^N e_n^2$, is the sum of the empirical training errors. The second, $\frac{1}{2} \sum_{i=1}^M \mu_i \mathbf{w}_i^T \mathbf{w}_i$, is the regularization term, measuring the amount of square of weights.

Comparing Equation (12) with (5), RNCL is equivalent to maximization of the posterior under Bayesian framework. The likelihood $P(D | \mathbf{w}, \beta)$ constitutes the empirical training errors terms and the prior over weight vector $P(\mathbf{w} | \mu)$ contributes to the regularization term. The

regularization term penalizes large weights, causing the weights to converge to smaller absolute values than they otherwise would.

Based on the above analysis, RNCL is an application of Bayesian framework in ensemble system. Instead of simultaneously optimizing the weigh vectors of ensemble, RNCL manages to train the entire ensemble by decomposing the job into a set of subtasks, which significantly reduces computational complexity under Bayesian framework.

Take an RBF neural network ensemble with linear outputs as an example. If we treat the ensemble as a single estimator, the training of the entire ensemble involves inversion of a matrix, whose computational complexity is $O(W^3) \sim O(M^3 n_i^3)$, where $W = \sum_{i=1}^M n_i$ (n_i is the number of weights in network i and M is the size of ensemble) is the total number of weights in ensemble. By decomposing the operation into a set of sub-operations, the computational complexity is reduced to $O(M n_i^3)$. Since M , the size of ensemble, is often set to be equal or greater than 25, the reduction of computational complexity is non-trivial.

Although there are two types of parameters: μ_i and β , the minimization of J_1 only depends on the ratio $\alpha_i = \mu_i/\beta$. These ratios, controlling the tradeoff between the empirical training errors and the regularization term, are crucial to the performance of ensemble. The next section will present a Bayesian approach to automatically optimize these parameters.

2) *Inference of Regularization Parameters:* In order to find the optimal parameters μ and β , we need to maximize the posterior of $P(\mu, \beta | D)$.

According to Bayesian rule, the posteriors of μ and β are obtained by

$$P(\mu, \beta | D) = \frac{P(D | \mu, \beta) \cdot P(\mu, \beta)}{P(D)} \propto P(D | \mu, \beta), \quad (14)$$

where a flat prior is assumed on the hyperparameters μ and β . According to Equation (8),

$$P(D | \mu, \beta) = \frac{P(D | \mathbf{w}, \beta) P(\mathbf{w} | \mu)}{P(\mathbf{w} | D)} \propto \sqrt{\frac{\prod_{i=1}^M \mu_i^{n_i} \beta^N}{\det A}} \exp(-J_1(\mathbf{w})). \quad (15)$$

In order to maximize the probability $P(D | \mu, \beta)$, negative logarithm is obtained:

$$J_2 = \frac{1}{2} \sum_{i=1}^M \mu_i \mathbf{w}_{i,MP}^T \mathbf{w}_{i,MP} + \frac{1}{2} \beta \sum_{n=1}^N e_{n,MP}^2 - \frac{1}{2} \sum_{i=1}^M n_i \log \mu_i - \frac{1}{2} N \log \beta + \frac{1}{2} \log \det A, \quad (16)$$

where the subscript MP indicates the most probable values.

Setting the gradient to zero and we can get the optimal $\alpha_i = \mu_i/\beta$, and please refer to Appendix B for detail.

$$\alpha_i^{new} = \frac{\sum_{n=1}^N e_{n,MP}^2 \left(n_i - \sum_{j \in n_i} \frac{\alpha_i}{\lambda_j + \alpha_i} \right)}{\mathbf{w}_{i,MP}^T \mathbf{w}_{i,MP} \left(N - \sum_{j=1}^W \frac{\lambda_j}{\lambda_j + \hat{\alpha}_j} \right)}, \quad (17)$$

where $\hat{\alpha}_j = [\alpha_1^{(1)}, \dots, \alpha_1^{(n_1)}, \dots, \alpha_M^{(1)}, \dots, \alpha_M^{(n_M)}]^T$ and the superscript indicates the repetition number of α_i . $j \in n_i$ indicates the range $[\sum_{t=1}^{i-1} n_t + 1, \dots, \sum_{t=1}^i n_t]$, and λ_i is the eigenvalue of the Hessian matrix $\nabla \nabla \left(\frac{1}{2} \sum_{n=1}^N e_n^2 \right)$.

When the eigenvalue decomposition λ_j is calculated, the update rule of α_i^{new} involves only vector products that can be evaluated very quickly. In order to reduce the computational complexity for large data sets, one can choose to calculate only the largest eigenvalues using the expectation maximization approach [24].

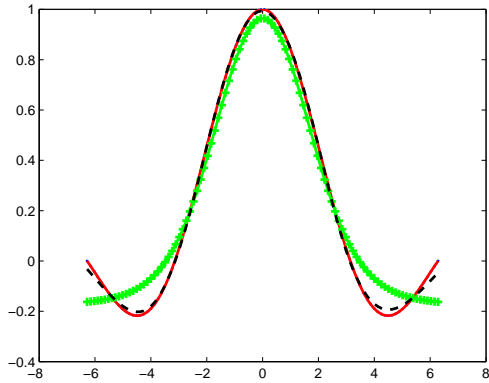
The learning algorithm thus proceeds by repeated application of (17) (step 4 in Figure 1), concurrent with training RNCL, equivalent to updating of the posterior statistics and from (11), until some suitable convergence criteria have been satisfied.

IV. EXPERIMENTAL ANALYSIS

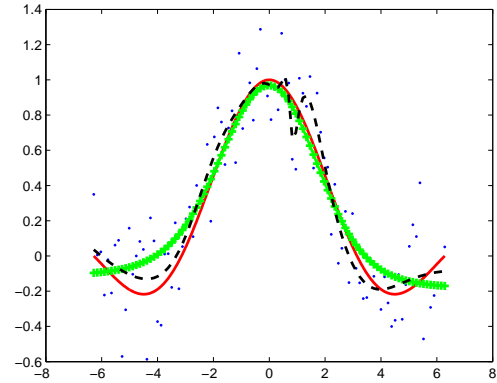
In this section we will present the experimental results of RNCL. Firstly, we present experimental results of RNCL on two synthetic regression problems and four synthetic classification problems in order to understand the behavior of the algorithm. We also design four experiments (two regression and two classification) with different noise levels to study the characteristics of RNCL and NCL with noise data. Secondly, we carry out extensive experiments on 8 benchmark regression data sets and 13 benchmark classification data sets to compare the performance of RNCL, NCL and Bagging.

A. Experimental Setup

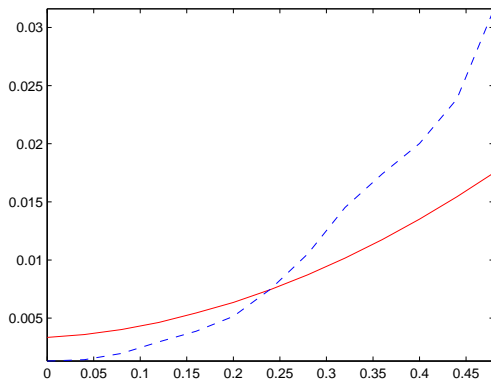
In this experiments, three-layer feed-forward multilayer perceptions (MLPs) are used as the base learner. The number of hidden nodes is randomly selected but restricted in the range 3 to 15. The initial connection weights for individual NNs are randomly chosen. We employ scaled conjugate gradient (SCG) algorithm to train MLP, NCL and RNCL. Since negative correlation learning algorithm uses MSE and correlation to train ensemble, it is not necessary to employ a



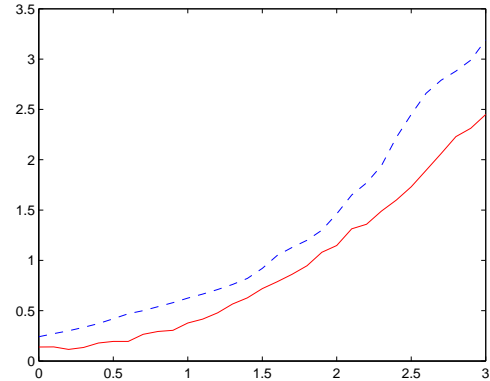
(a) Sinc Noise Free



(b) Sinc with Gaussian noise (mean 0, variance 0.2)



(c) Sinc with Different Noise Levels



(d) Friedman Test with Different Noise Levels

Fig. 2. Comparison of NCL and RNCL on regression data sets: Sinc and Friedman test. In Figure 2(a) and 2(b), the lines in green (wide zigzag), black(dashed) and red(solid) are obtained by RNCL, NCL and the noise-free function, respectively. Figure 2(c) and 2(d) show mean square error (MSE) of RNCL (red solid) and NCL (blue dashed) on Sinc and Friedman test with different noise levels. Results are based on 100 runs.

large ensemble. We use 25 MLPs to constitute the ensemble of NCL and RNCL. For Bagging, we employ 100 MLPs to constitute the ensemble. The input attributes of data sets are scaled to mean zero and unit variance as the preprocessing procedure.

B. Synthetic Data Sets

As the first experiment, we compare RNCL and NCL algorithms on two synthetic regression data sets: Sinc and Friedman test. Figure 2(a) and 2(b) show the output of RNCL and NCL on sinc function with different noise levels. In the noise-free case, NCL perfectly approximates the

actual function, while RNCL does not approximate the function very well near the tail. However, when the noise level increases, NCL, only depending on minimizing empirical error, overfits the noise in the training set while RNCL is more robust to noise than NCL, refer to Figure 2(b).

In order to explore the behaviors of RNCL and NCL with different noise levels, we add mean zero and different variance Gaussian noise to sinc and Friedman test problems. Figures 2(c) and 2(d) show the average results of 100 runs. Since the standard deviations of the targets: sinc and Friedman test, are different, the range of noise levels are different in Figure 2(c) and 2(d).

For sinc data set, when the noise level (variance) is below 0.24, NCL outperforms RNCL. When the noise level is greater than 0.24, MSE of RNCL increases slower than that of NCL when the noise increases. For Friedman test data set, RNCL outperforms NCL all the time and the difference between RNCL and NCL becomes greater when noise level passes 2.5. From these figures, we can observe that RNCL is more robust to noise.

In the following, we demonstrate the application of RNCL on classification problems. Firstly, we apply RNCL and RNCL on four synthetic data in two dimensions in order to illustrate graphically the decision boundary.

These four data sets are (1) *synth* are generated from mixtures of two Gaussians by [25]. (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 by 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 3 we present a comparison of RNCL and NCL. We can observe a similar performance of RNCL and NCL in the case of *Relevance*. Since the data set is noise-free, both RNCL and NCL successfully separate the two classes. The situation is a little similar in the case of *Overlap*, RNCL produces a linear boundary according to the expectation while NCL concentrates on the three overlapping points and generates a linear plane with a *corner*. Although the training error of NCL is smaller, it does not generalize well for this data set.

We observe that RNCL gives more accurate results in the other cases. In the cases of *Synth* and *Bumpy*, RNCL produces smooth boundary and disregards the outliers in the training points. In the case of *Bumpy*, the noise level is great because of these overlapping points. NCL does not generalize and produces the twisty boundary. Although the boundary line of NCL for the case of *Synth* seems smooth, it does separate the decision boundary into two parts and disregards the

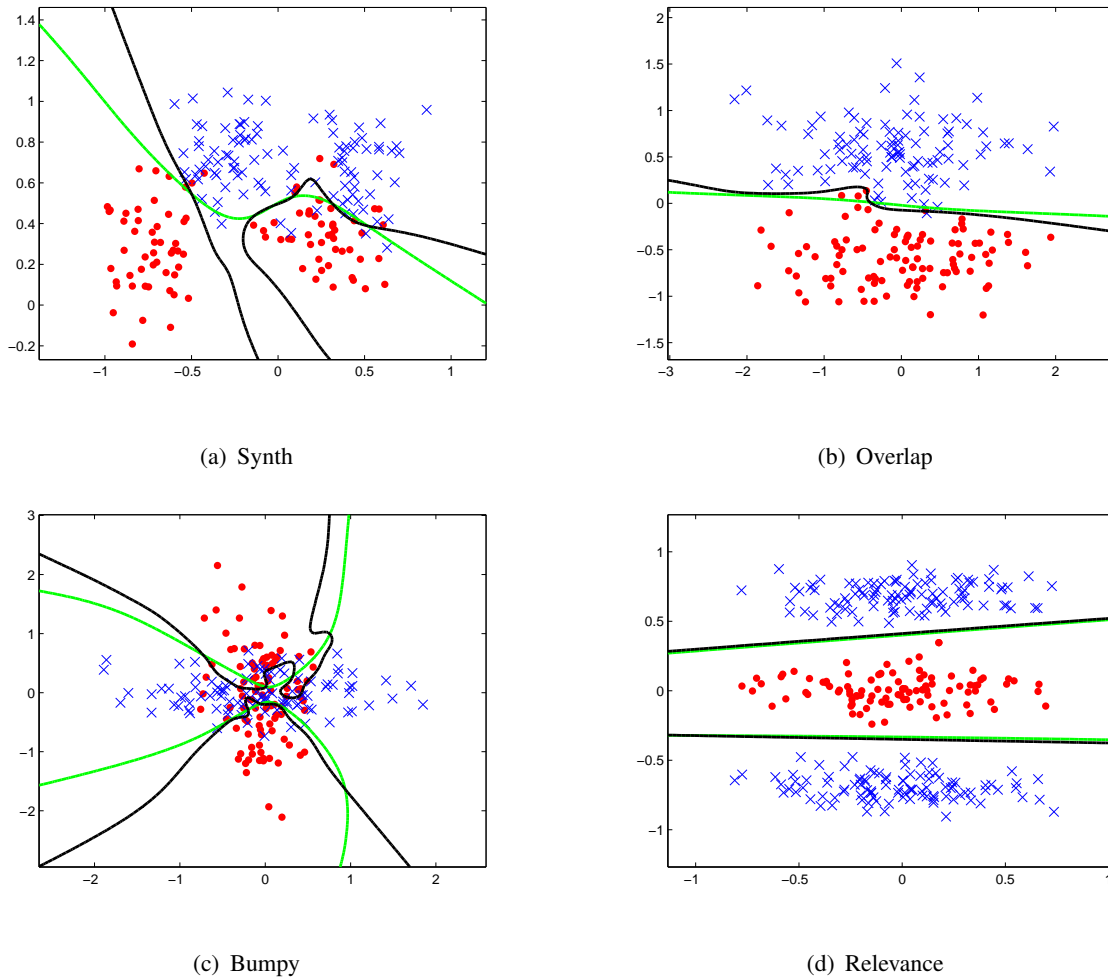


Fig. 3. Comparison of RNCL and NCL on four synthetic classification data sets. Two classes are shown as crosses and dots. The separating lines were obtained by projecting test data over a grid. The lines in green (light) and black (dark) were obtained by RNCL and NCL, respectively.

future points between the two boundaries.

In order to check the behaviors of RNCL and NCL on noise classification problems, we conduct similar noise experiments as the regression problems. In the experiments, we select two data sets: synth and banana ².

To change the noise level, we randomly select different percentages of data points and reverse their labels. We run 100 times and report the average results in Figure 4. Figure 4(a) and Figure 4(b) visualize the decision boundaries of RNCL and NCL with 20% noise points.

²<http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm>

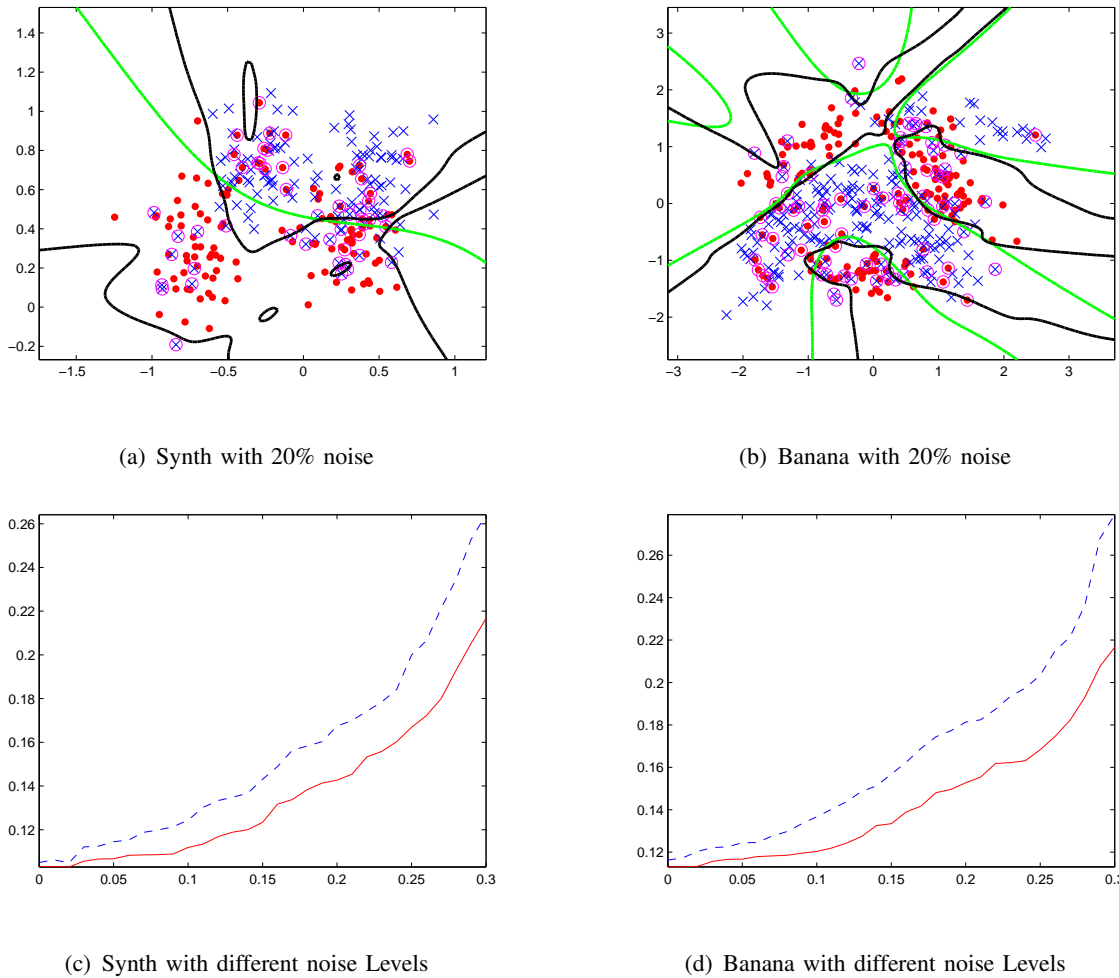


Fig. 4. Comparison of RNCL and NCL on two classification data sets. Two classes are shown as crosses and dots. The separating lines were obtained by projecting test data over a grid. In Figure 4(a) and 4(b), the decision boundary in green(light) and black(dark) are obtained by RNCL and NCL, respectively. The randomly-selected noise points are marked with a circle. Figure 4(c) and 4(d) show classification error of RNCL (red solid) and NCL (blue dashed) vs. noise levels on Synth and banana data sets. The results are based on 100 runs.

Though the noise level is high, RNCL produces smooth boundary. NCL tries to minimize the training error and it does not generalize well. We also plot the curve, Figure 4(c) and 4(d), of classification error vs. noise level for these two data sets. In these two figures, RNCL is a little better in the beginning, but as the noise level increases, RNCL significantly outperforms NCL.

The results of RNCL are promising on these regression and classification problems. Based on the results and analysis, RNCL inherits the advantages of NCL and can achieve a good performance with a small ensemble. The regularization term does work in RNCL and improves

TABLE I
SUMMARY OF REGRESSION DATA SETS

Data Sets	Function	Variable	Training Points	Test Points
Mexican Hat	$y = \text{sinc} x = \frac{\sin x }{ x }$	$x \sim U[-2\pi, 2\pi]$	250	1000
Friedman 1	$y = 10 \sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5$	$x_i \sim U[0, 1]$	250	1000
Gabor	$y = \frac{1}{2} \pi \exp[-2(x_1^2 + x_2^2)] \cos[2\pi(x_1 + x_2)]$	$x_i \sim U[0, 1]$	250	1000
Multi	$y = 0.79 + 1.27x_1x_2 + 1.56x_1x_4 + 3.42x_2x_5 + 2.06x_3x_4x_5$	$x_i \sim U[0, 1]$	250	1000
Plane	$y = 0.6x_1 + 0.3x_2$	$x_i \sim U[0, 1]$	250	1000
Polynomial	$y = 1 + 2x + 3x^2 + 4x^3 + 5x^4$	$x \sim U[0, 1]$	250	1000
Sinc	$y = \frac{\sin x}{x}$	$x \sim U[0, 2\pi]$	250	1000
Boston House	—	-	400	106

its ability against noise, which is especially important in practice since most of the actual data are contaminated by noise. After the analysis with synthetic data sets, the next section presents the results for the real-world benchmark problems.

C. Experimental Results

In order to evaluate the performance of RNCL, we test RNCL, NCL and Bagging on 8 regression benchmark problems and 13 classification benchmark problems. The information on the data sets used for regression is tabulated in Table I. The Mexican hat has been used by Weston et al. [26] in investigating the performance of support vector machines. Friedman 1 has been used by Breiman [10] in testing the performance of Bagging. Gabor, Multi, and Sinc have been used by Hansen [27] in comparing several ensemble approaches. Plane has been used by Ridgeway et al. [28] in exploring the performance of boosted naive Bayesian regressors. The Boston house data set is obtained from UCI machine learning repository [29]. In the 100 runs, we randomly select 400 data points for training set and the rest 106 points are used for testing. The constraints on the variables are also shown in Table I, where $U[x, y]$ means a uniform distribution over the interval determined by x and y . Note that in our experiments additive Gaussian noise, except for Boston House data set, is generated on the output of standard deviation one-third of that of the target $y(x)$.

The classification data set used in this paper has been summarized in Table II. These data sets

TABLE II
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

TABLE III

COMPARISON OF NCL, BAGGING AND RNCL ON 8 REGRESSION DATA SETS, BY % ERROR (STANDARD DEVIATION) AND TEST P VALUE BETWEEN BAGGING VS. RNCL AND NCL VS. RNCL. THE P VALUE WITH A STAR MEANS THE TEST IS SIGNIFICANT. THESE RESULTS ARE AVERAGES OF 100 RUNS.

Data Sets	Bagging	P value	NCL	P value	RNCL
Mexican Hat	0.0064(0.0018)	0.07	0.0069(0.0013)	0.00*	0.0060(0.0017)
Friedman	1.75(0.34)	0.00*	1.05(0.24)	0.00*	0.82(0.21)
Gabor	0.020(0.004)	0.00*	0.015(0.002)	0.00*	0.009(0.004)
Multi	0.038(0.010)	0.05	0.105(0.030)	0.00*	0.035(0.009)
Plane	0.83e-4(0.48e-4)	0.00*	1.64e-4(0.52e-4)	0.11	1.42e-4(0.48e-4)
Polynomial	0.159(0.075)	0.00*	0.128(0.044)	0.00*	0.076(0.023)
Sinc	0.0067(0.0018)	0.07	0.0070(0.0015)	0.00*	0.0066(0.0016)
Boston House	13.41(4.71)	0.02*	12.56(3.62)	0.47	12.16(3.51)

TABLE IV

COMPARISON OF NCL, BAGGING AND RNCL ON 13 BENCHMARK DATA SETS, BY % ERROR (STANDARD DEVIATION) AND T TEST P VALUE BETWEEN BAGGING VS. RNCL AND NCL VS. RNCL. THE P VALUE WITH A STAR MEANS THE TEST IS SIGNIFICANT. THESE RESULTS ARE AVERAGES OF 100 RUNS.

Error	Banana	Cancer	Diabetics	Solar	German	Heart	
Bagging	11.41(0.78)	28.12(4.87)	24.23(1.78)	34.97(1.51)	24.97(2.10)	18.71(3.10)	
Bagging vs. RNCL	0.00*	0.16	0.00*	0.00*	0.10	0.00*	
NCL	11.09(0.68)	28.42(4.61)	24.57(1.96)	35.42(1.79)	25.88(2.19)	18.28(3.68)	
Bagging vs. RNCL	0.01*	0.21	0.03*	0.00*	0.00*	0.00*	
RNCL	10.42(0.65)	26.31(4.77)	23.16(1.62)	33.86(1.71)	24.01(2.23)	16.32(3.11)	
Error	Image	Ringnorm	Splice	Thyroid	Titanic	Twonorm	Waveform
Bagging	3.34(0.69)	1.84(0.31)	11.62(0.62)	4.48(2.36)	23.98(1.37)	3.03(0.30)	11.68(0.62)
Bagging vs. RNCL	0.00*	0.05	0.00*	0.31	0.15	0.00*	0.00*
NCL	2.65(0.46)	1.64(0.24)	11.23(0.72)	4.45(2.37)	22.71(1.32)	2.61(0.26)	12.30(0.76)
Bagging vs. RNCL	0.06	0.01*	0.03*	0.34	0.89	0.13	0.00*
RNCL	2.79(0.68)	1.79(0.19)	10.53(0.64)	4.03(2.11)	22.42(1.03)	2.79(0.21)	9.91(0.48)

have been preprocessed and organized by Rätsch et al.³ These data sets include one synthetic set (banana) along with 12 other real-world data sets coming from the UCI [29], DELVE⁴ and STATLOG repositories. The main difference between the original and Rätsch's data is that 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 the Rätsch's implementation and we generate additional 80 folds by random sampling to make the experiments consistent). In addition, every instance is normalized dimension-wise to have zero mean and unit standard deviation.

Table III reports the performance of these algorithms on the 8 benchmark regression data sets. According to the tables, RNCL performs quite favorably in these data sets. For example, RNCL outperforms the other two methods in 7 out of 8 data sets, in which 6 wins are significant against NCL and 4 wins are significant against Bagging.

The performance of RNCL, NCL and Bagging on classification problems have been tabulated

³<http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm>

⁴<http://www.cs.toronto.edu/~delve/data/datasets.html>

TABLE V

RUNNING TIME OF RNCL AND NCL ON REGRESSION DATA SETS IN SECONDS. RESULTS ARE AVERAGED OVER 100 RUNS.

Data Sets	Mexican Hat	Friedman	Gabor	Multi	Plane	Polynomial	Sinc	House
NCL	6.4	16.1	8.4	21.2	3.3	23.6	6.2	28.6
RNCL	19.6	143.2	30.8	113.2	7.9	62.3	20.3	269.5

TABLE VI

RUNNING TIME OF RNCL AND NCL ON CLASSIFICATION DATA SETS IN SECONDS. RESULTS ARE AVERAGED OVER 100 RUNS.

Error	Banana	Cancer	Diabetics	Solar	German	Heart	
NCL	3.6	8.2	4.6	12.1	21.6	1.6	
RNCL	12.1	29.4	17.6	36.0	168.5	16.7	
Error	Image	Ringnorm	Splice	Thyroid	Titanic	Twonorm	Waveform
NCL	21.3	4.8	43.6	3.1	0.8	4.0	4.2
RNCL	277.2	20.7	426.7	8.4	2.6	34.8	41.7

in Table IV. Based on the tables, RNCL performs well since RNCL outperforms all the other methods in 11 out of 13 data sets, comes second in 2 cases. In the results, NCL outperforms RNCL in the cases: Ringnorm and Twonorm, which are both synthetic data with few noise (see the lower error rate). This observation also validates that NCL achieves better results when noise is small and RNCL is more robust to noise than NCL.

D. Computational Complexity and Running Time

Based on the algorithm in Figure 1, RNCL consists of two main parts: neural network training using negative correlation learning and Bayesian parameter optimization.

In the first part, for each component neural network, totally M neural networks in the ensemble, one need to train the network with a amount of epoches. Since the scaled conjugate gradient algorithm is employed in RNCL, the training can be evaluated quickly.

In the second part, the major running time is consumed in the calculation of hessian matrix and eigen-decomposition of the hessian matrix. This paper employs the fast multiplication by the hessian [30], [31] method to estimate the hessian matrix. Although the eigenvalues have to be calculated only once, their calculation in the eigenvalue problem becomes computationally

TABLE VII

COMPARISON OF RNCL AND NCL WITH EQUAL TIME ON FOUR REGRESSION PROBLEMS AND FOUR CLASSIFICATION PROBLEMS. NCL IS RUN 10 TIMES IN 8 EXPERIMENTS WITH RANDOMLY SELECTED REGULARIZATION PARAMETERS BETWEEN 0 AND 1. THE FIRST ROW REPORTS THE BEST PERFORMANCE OF NCL IN THE 10 RUNS. THE RESULTS ARE THE AVERAGE RESULTS OF 20 RUNS.

Data Sets	Mexican Hat	Friedman	Gabor	House	Banana	Cancer	Diabetics	Solar
(Best) NCL	0.0064	0.93	0.013	12.33	10.86	27.11	23.32	34.16
RNCL	0.0060	0.82	0.009	12.16	10.42	26.31	23.16	33.86

expensive for large data sets. In this case, one can choose to calculate only the largest eigenvalues using an expectation maximization approach [24].

RNCL is an iterative algorithm to update the regularization parameters, in most of time it will converge in less than 8 iterations. Because most of the computation time is consumed in the first part if the hessian matrix is not so large, the computation time of RNCL is almost 5-10 times of NCL. In Table V and VI, we show the average running time of RNCL and NCL over 100 runs. The computational environment is windows XP with Intel Core 2 Duo 1.66G CPU and 2G RAM. These algorithms including RNCL and NCL are programmed in C++.

Based on these Tables, the running time of RNCL is almost 5-10 times of that of NCL. In order to fairly compare RNCL with NCL, the following experiments are carried out: We run NCL 10 times on 8 data sets with randomly selected regularization parameter in the range of $[0, 1]$ and let the total time of NCL equals to the running time or RNCL.

Table VII reports the best performance of NCL in 10 runs on 8 data sets based on the average results of 20 runs. From the table, we observe that even given the same amount of time to NCL, NCL cannot pick the best regularization parameter as Bayesian inference does in RNCL. The Bayesian parameter optimization procedures do improve the performance of NCL.

V. CONCLUSION

This paper analyzes negative correlation learning and points out that NCL is prone to overfitting the noise because NCL does not regularize its complexity. In the following, the paper analyzes this problem and proposes the regularized negative correlation learning (RNCL) which incorporates an additional regularization term for NCL. RNCL decomposes the ensemble's training

objectives, including MSE and regularization, into a set of sub-objectives, and each sub-objective is implemented by each component neural network. RNCL inherits the advantage of NCL. RNCL formulation is applicable to any nonlinear regression estimator minimizing the MSE. In this paper, we also provide the statistical Bayesian interpretation for the RNCL and propose an automatical regularization parameters optimization procedures based on Bayesian inference.

Several experiments have been carried out to evaluate RNCL. The experiments on two synthetic regression problems and four synthetic classification problems demonstrate the behaviors of RNCL and NCL. The following experiments on two regression and two classification problems with different noise levels demonstrate that RNCL achieves better performance than NCL, especially when the noise is non-trivial in data sets. Secondly, we carry out extensive experiments on 8 benchmark regression and 13 benchmark classification data sets to compare the performance of RNCL, NCL and Bagging. RNCL has shown an excellent performance on these data sets.

This paper analyzes the computational complexity of RNCL and carries out experiments to demonstrate that even NCL is given the same time as RNCL, NCL could not achieve the same performance as RNCL. Based on the analysis, RNCL has the advantages of NCL and improves the performance of NCL. The noise-robustness characteristic of RNCL is especially important when the training data are contaminated with noise.

ACKNOWLEDGMENT

This work is partially supported by a Dorothy Hodgkin Postgraduate Scholarship to the first author.

APPENDIX

A. Further Details of Gaussian Posterior

Considering the normalization term, the posterior of weigh vector \mathbf{w} is described as

$$P(\mathbf{w} | D) = \frac{\exp(-J_1(\mathbf{w}))}{\int \exp(-J_1(\mathbf{w}))d\mathbf{w}}. \quad (18)$$

In order to obtain the result, the Taylor expansion of $J_1(\mathbf{w})$ is employed at point \mathbf{w}_{MP} .

$$J_1(\mathbf{w}) = J_1(\mathbf{w}_{MP}) + \frac{1}{2}(\mathbf{w} - \mathbf{w}_{MP})^T A(\mathbf{w} - \mathbf{w}_{MP}), \quad (19)$$

where \mathbf{w}_{MP} is the most probable weight vector, A is the hessian matrix of $J_1(\mathbf{w})$.

$$A = \nabla\nabla J_1 = \nabla\nabla \left(\sum_{i=1}^M \frac{\mu_i}{2} \mathbf{w}_i^T \mathbf{w}_i + \frac{\beta}{2} \sum_{n=1}^N e_n^2 \right) = \text{diag}(\Lambda) + \beta \nabla\nabla \left(\frac{1}{2} \sum_{n=1}^N e_n^2 \right), \quad (20)$$

where $\Lambda = [\mu_1^{(1)}, \dots, \mu_1^{(n_1)}, \mu_2^{(1)}, \dots, \mu_2^{(n_2)}, \dots, \mu_M^{(1)}, \dots, \mu_M^{(n_M)}]^T$ and the superscript indicates the repetition number of μ_i .

The integral can be computed as below:

$$\begin{aligned} \int \exp(-J_1(\mathbf{w})) d\mathbf{w} &= \int \exp(-J_1(\mathbf{w}_{MP}) - \frac{1}{2}(\mathbf{w} - \mathbf{w}_{MP})^T A(\mathbf{w} - \mathbf{w}_{MP})) d\mathbf{w} \\ &= \exp(-J_1(\mathbf{w}_{MP})) \cdot (2\pi)^{W/2} \det A^{-\frac{1}{2}}. \end{aligned} \quad (21)$$

Based on these equations, the exact posterior of \mathbf{w} is obtained as follows

$$P(\mathbf{w} | D) = \frac{\exp(-J_1(\mathbf{w}))}{\int \exp(-J_1(\mathbf{w})) d\mathbf{w}} = \frac{\exp(-\frac{1}{2}(\mathbf{w} - \mathbf{w}_{MP})^T A(\mathbf{w} - \mathbf{w}_{MP}))}{(2\pi)^{W/2} \det A^{-\frac{1}{2}}}. \quad (22)$$

B. Details of Parameter Updates

Since the update rule for $\alpha_i = \mu_i/\beta$ is can be obtained from the derivation of J_2 .

$$J_2 = \frac{1}{2} \sum_{i=1}^M \mu_i \mathbf{w}_{i,MP}^T \mathbf{w}_{i,MP} + \frac{1}{2} \beta \sum_{n=1}^N e_{n,MP}^2 - \frac{1}{2} \sum_{i=1}^M n_i \log \mu_i - \frac{1}{2} N \log \beta + \frac{1}{2} \log \det A. \quad (23)$$

In order to apply the partial derivation to J_2 , we need to apply partial derivation to $\log \det A$.

Since $\det A = \prod_{j=1}^W (\beta \lambda_j + \Lambda_j)$, where λ_j is the eigenvalue of the Hessian matrix $\nabla\nabla \left(\frac{1}{2} \sum_{n=1}^N e_n^2 \right)$ and W is the number of weights in ensemble.

$$\begin{aligned} \frac{\partial}{\partial \mu_i} \log \det A &= \frac{\partial}{\partial \mu_i} \log \left(\prod_{j=1}^W (\beta \lambda_j + \Lambda_j) \right) = \sum_{j \in n_i} \frac{1}{\beta \lambda_j + \mu_i}, \\ \frac{\partial}{\partial \beta} \log \det A &= \frac{\partial}{\partial \beta} \log \left(\prod_{j=1}^W (\beta \lambda_j + \Lambda_j) \right) = \sum_j \frac{\lambda_j}{\beta \lambda_j + \Lambda_j}, \end{aligned} \quad (24)$$

where $j \in n_i$ indicates the range $[\sum_{t=1}^{i-1} n_t + 1, \dots, \sum_{t=1}^i n_t]$.

The gradient of $\log P(D | \mu, \beta)$ toward μ_i and β are:

$$\frac{\partial J_2}{\partial \mu_i} = \frac{1}{2} \mathbf{w}_{i,MP}^T \mathbf{w}_{i,MP} - \frac{1}{2} \frac{n_i}{\mu_i} + \frac{1}{2} \sum_{j \in n_i} \frac{1}{\beta \lambda_j + \mu_i}, \quad (25)$$

$$\frac{\partial J_2}{\partial \beta} = \frac{1}{2} \sum_{n=1}^N e_{n,MP}^2 - \frac{N}{2\beta} + \frac{1}{2} \sum_j \frac{\lambda_j}{\beta \lambda_j + \Lambda_j}, \quad (26)$$

Setting the gradient to zero and the optimal μ_i and β can be obtained:

$$\mu_i^{new} = \frac{1}{\mathbf{w}_{i,MP}^T \mathbf{w}_{i,MP}} \left(n_i - \sum_{j \in n_i} \frac{\mu_i}{\beta \lambda_j + \mu_i} \right), \quad (27)$$

$$\beta^{new} = \frac{1}{\sum_{n=1}^N e_{n,MP}^2} \left(N - \sum_{j=1}^W \frac{\beta \lambda_j}{\beta \lambda_j + \Lambda_j} \right). \quad (28)$$

Combining both equations (27) and (28) and the relation $\alpha_i = \mu_i/\beta$, we obtain the following equation:

$$\beta \left[\sum_{i=1}^M \alpha_i \mathbf{w}_{i,MP}^T \mathbf{w}_{i,MP} + \sum_{n=1}^N e_{n,MP}^2 \right] = N \quad (29)$$

In the following, we reformulate the optimization problem, Equation (23), in μ_i and β into a scalar optimization problem in $\alpha_i = \mu_i/\beta$. Therefore, we first replace that optimization problem by an optimization problem in β and α_i by the relation $\mu_i = \beta \alpha_i$. Since the equation (29) also holds in the scalar optimization, we search for the optimum only along this curve in the $(\alpha_i$ and $\beta)$ space.

By elimination of β from equation (29), the minimization problem from J_2 is obtained in a straightforward way:

$$J_3 = \sum_{j=1}^W \log\left(1 + \frac{\lambda_j}{\hat{\alpha}_j}\right) + N \log \left(\sum_{i=1}^M \alpha_i \mathbf{w}_{i,MP}^T \mathbf{w}_{i,MP} + \sum_{n=1}^N e_{n,MP}^2 \right), \quad (30)$$

where $\hat{\alpha}_j = \Lambda_j/\beta$ and $\Lambda = [\mu_1^{(1)}, \dots, \mu_1^{(n_1)}, \mu_2^{(1)}, \dots, \mu_2^{(n_2)}, \dots, \mu_M^{(1)}, \dots, \mu_M^{(n_M)}]^T$.

Setting $\frac{\partial J_3}{\partial \alpha_i} = 0$, the update rule $\alpha_i^{new} = \mu_i/\beta$ can be obtained as follows:

$$\alpha_i^{new} = \frac{\sum_{n=1}^N e_{n,MP}^2 \left(n_i - \sum_{j \in n_i} \frac{\alpha_i}{\lambda_j + \alpha_i} \right)}{\mathbf{w}_{i,MP}^T \mathbf{w}_{i,MP} \left(N - \sum_{j=1}^W \frac{\lambda_j}{\lambda_j + \hat{\alpha}_j} \right)}, \quad (31)$$

where $j \in n_i$ indicates the range $\left[\sum_{t=1}^{i-1} n_t + 1, \dots, \sum_{t=1}^i n_t \right]$.

REFERENCES

- [1] Y. Liu and X. Yao. Ensemble learning via negative correlation. *Neural Networks*, 12(10):1399–1404, 1999.
- [2] Y. Liu and X. Yao. Simultaneous training of negatively correlated neural networks in an ensemble. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 29(6):716–725, 1999.
- [3] L. K. Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990.

- [4] F. J. Huang, T. Chen, Z. Zhou, and H. Zhang. Pose invariant face recognition. In *Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition 2000*, pages 245–250, Washington, DC, USA, 2000.
- [5] L. K. Hansen, L. Liisberg, and P. Salamon. Ensemble methods for handwritten digit recognition. In *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing*, pages 333–342, Helsingoer, Denmark, 1992.
- [6] K. J. Cherkauer. Human expert level performance on a scientific image analysis task by a system using combined artificial neural networks. In *Proceedings of AAAI-96 Workshop on Integrating Multiple Learned Models for Improving and Scaling Machine Learning Algorithms*, pages 15–21, Menlo Park, CA, USA, 1996.
- [7] 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*, pages 693–698, 2001.
- [8] Y. Liu, X. Yao, and T. Higuchi. Evolutionary ensembles with negative correlation learning. *IEEE Transaction on Evolutionary Computation*, 4(4):380–387, 2000.
- [9] A. Krogh and J. A. Hertz. A simple weight decay can improve generalization. In *Advances in Neural Information Processing Systems*, volume 4, pages 950–957, 1992.
- [10] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
- [11] T. K. Ho. The random subspace method for constructing decision forests. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 20(8):832–844, 1998.
- [12] R. E. Schapire. A brief introduction to boosting. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 1401–1406, 1999.
- [13] M. M. Islam, X. Yao, and K. Murase. A constructive algorithm for training cooperative neural network ensembles. *IEEE Transaction on Neural Networks*, 14(4):820–834, 2003.
- [14] G. Brown, J. Wyatt, and P. Tino. Managing diversity in regression ensembles. *Journal of Machine Learning Research*, 6:1621–1650, 2005.
- [15] H. Chen and X. Yao. Evolutionary random neural ensemble based on negative correlation learning. In *Proceedings of IEEE Congress on Evolutionary Computation (CEC'07)*, pages 1468–1474, 2007.
- [16] A. E. Hoerl and R. W. Kennard. Ridge regression: biased estimation for nonorthogonal problems. *Technometrics*, 42(1):80–86, 2000.
- [17] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, 1992.
- [18] D. J. C. MacKay. Bayesian interpolation. *Neural Computation*, 4(3):415–447, 1992.
- [19] D. J. C. MacKay. The evidence framework applied to classification networks. *Neural Computation*, 4(3):720–736, 1992.
- [20] T. Van Gestel, J. A. K. Suykens, G. Lanckriet, A. Lambrechts, B. De Moor, and J. Vandewalle. Bayesian framework for least-squares support vector machine classifiers, gaussian processes, and kernel fisher discriminant analysis. *Neural Computation*, 14(5):1115–1147, 2002.
- [21] M. E. Tipping. Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1(3):211–244, 2001.
- [22] J.T. Kwok. Moderating the outputs of support vector machine classifiers. *IEEE Transactions on Neural Networks*, 10(5):1018–1031, 1999.
- [23] M. F. Møller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Network*, 6(4):525–533, 1993.

- [24] R. Rosipal and M. Girolami. An expectation-maximization approach to nonlinear component analysis. *Neural Computation*, 13(3):505–510, 2001.
- [25] B. D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.
- [26] J.A.E. Weston, M.O. Stitson, A. Gammerman, V. Vovk, and V. Vapnik. Experiments with support vector machines. Technical Report CSD-TR-96-19, Royal Holloway University of London, London, 1996.
- [27] J.V. Hansen. *Combining predictors: Meta machine learning methods and bias/variance and ambiguity decompositions*. Ph.d. dissertation, Department of Computer Science, University of Aarhus, Denmark, 2000.
- [28] G. Ridgeway, D. Madigan, and T. Richardson. Boosting methodology for regression problems. In *Proceedings of Artificial Intelligence and Statistics*, pages 152–161, 1999.
- [29] A. Asuncion and D.J. Newman. UCI machine learning repository, 2007.
- [30] M. Møller. *Efficient Training of Feed-Forward Neural Networks*. Phd dissertation, University of Aarhus, Denmark, 1993.
- [31] B. A. Pearlmutter. Fast exact multiplication by the hessian. *Neural Computation*, 6(1):147–160, 1994.