Regularized Negative Correlation Learning for Neural Network Ensembles

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

***Abstract*—Negative correlation learning (NCL) is a neural network ensemble learning algorithm that 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 of the ensemble. This paper analyzes NCL and reveals that the training of NCL (when $\lambda = 1$) corresponds to training the entire ensemble as a single learning machine that only minimizes the MSE without regularization. This analysis explains the reason why NCL is prone to overfitting the noise in the training set. This paper also demonstrates that tuning the correlation parameter λ in NCL by cross validation cannot overcome the overfitting problem. 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 an automatic algorithm to optimize regularization parameters based on Bayesian inference. The RNCL formulation is applicable to any nonlinear estimator minimizing the MSE. The experiments on synthetic as well as real-world data sets demonstrate that RNCL achieves better performance than NCL, especially when the noise level is nontrivial in the data set.**

***Index Terms*—Ensembles, negative correlation learning (NCL), neural network ensembles, neural networks, probabilistic model, regularization.**

I. Introduction

ENSEMBLES of multiple learning machines, i.e., groups of learners that work together as committee, have attracted a lot of research interest in the machine learning community since this method 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 Salamons' 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 the machine learning community 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] is a specific neural network ensemble algorithm and it has been applied in a number of empirical problems, including regression problems [7] and classification problems [8]. NCL 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 of 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 set 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 while it pays less attention to regularizing the complexity of the ensemble, leading NCL to be prone to overfitting the noise in the training set. Similarly, setting λ to a zero or small positive value 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 to explicitly alter the emphasis on the individual MSE and correlation portions 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 nontrivial noise, which will be evidenced by the empirical work in this paper.

The paper analyzes the overfitting problem of NCL and in order to solve this problem, this paper proposes regularized negative correlation learning (RNCL) algorithm which incorporates an additional regularization term into 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 present how the training algorithm of NCL is equivalent to training a single learning machine when $\lambda = 1$ and how RNCL controls the complexity by adding a regularization term. The regularization parameter is used to control the tradeoff between MSE and regularization and this parameter is crucial to ensemble's generalization ability.

We provide a Bayesian interpretation for RNCL, and propose an algorithm for parameter optimization based on Bayesian inference. The RNCL algorithm is a generic ensemble algorithm, and it is applicable to any nonlinear regression estimator minimizing the MSE, for example multilayer perceptrons (MLPs)

Manuscript received April 02, 2008; revised July 05, 2009 and September 17, 2009; accepted September 24, 2009. First published November 17, 2009; current version published December 04, 2009. The work of H. Chen was supported in part by a Dorothy Hodgkin Postgraduate award. The work of X. Yao was supported by EPSRC under Grant GR/T10671/01.

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, U.K. (e-mail: H.Chen@cs.bham.ac.uk; X.Yao@cs.bham.ac.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNN.2009.2034144

and radial basis function (RBF) neural networks. In this paper, we give an example using RBF as the base estimators.

The contributions of this paper are as follows. 1) This paper gives evidence that NCL is prone to overfitting. Although (3), in which the training of NCL can be seen as training a single estimator when $\lambda = 1$, has been noticed before [9], we are the first to claim that NCL, including $\lambda = 1$ and λ selected by cross validation, is prone to overfitting and validate this using extensive empirical experiments. 2) We propose the RNCL algorithm with an additional regularization term and demonstrate how to decompose the training of ensemble into a set of subtasks. 3) A Bayesian interpretation of RNCL and a parameter optimization procedure based on Bayesian inference have been presented to provide the theoretical foundation for RNCL and optimize the regularization parameter in RNCL, respectively. This work extends the application of Bayesian inference to ensemble methods and demonstrates how to decouple the training and optimization of ensemble into a group of subtasks.

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 discussions 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 literature. 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. NCL [1], [2] is a representative ensemble algorithm in this category. This paper will focus on this kind of ensemble learning algorithm.

NCL was first proposed by Liu *et al.* [1], where NCL and evolutionary learning are combined to automatically design neural network (NN) ensembles. This algorithm 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] took a constructive approach to build the ensemble, starting from a small group of networks with minimal architecture. The networks are all partially trained using NCL. To our knowledge, the approach can automatically determine weights, network topologies, and ensemble membership.

Brown *et al.* [9] formalized this technique and provided 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.* [14] proposed to incorporate bootstrap of data, random feature subspace [11], and evolutionary algorithm 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.

In [15], Chen and Yao propose the multiobjective regularized negative correlation learning (MRNCL) algorithm which formulates RNCL algorithm as a multiobjective evolutionary learning problem. Compared with MRNCL in [15], RNCL explicitly optimizes the regularization coefficients using Bayesian inference while MRNCL implicitly optimizes the tradeoff among the three terms using a multiobjective evolutionary algorithm. By considering an additional weighting coefficient λ of the correlation term, MRNCL sometimes achieves slightly better performance than RNCL, where λ is always set to 1. The advantages of RNCL with Bayesian inference include its sound theoretical underpinning and computational efficiency in comparison with MRNCL.

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 the 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 [16] 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 [17]. 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. 1) 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. 2) Large weights can cause excessive variance of the output [18].

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 [19], was successful. Then, the Bayesian technique has been successfully applied to model selection for least squares support vector machine [20], sparse Bayesian learning, i.e., relevance vector machine [21] and probabilistic classification vector machine [22], which is a partial Bayesian algorithm by employing different truncated Gaussian priors for different classes. Another important feature of Bayesian inference is that error bars [19] can be assigned to network predictions in regression problems and the probability of prediction [23] can be assigned to classification results that can avoid making overconfident predictions in regions of sparse data. Bishop [24] has given a comprehensive review and application of Bayesian methods on single learners. In this paper, we extend the Bayesian inference from a single learner to ensemble methods and demonstrate how to decouple the training and optimization of ensemble into a group of subtasks.

III. Regularized Negative Correlation Learning

This section presents NCL and its potential problem of overfitting. In order to address the problem, RNCL is proposed. This section also presents an efficient procedure to optimize regularization parameters by Bayesian inference in RNCL.

A. 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 [1].

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

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

To train 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 errors for the rest of the ensemble. The λ parameter controls a tradeoff between the training error term and the penalty term. With $\lambda = 0$, we would have an ensemble with each network training independently. If λ is increased, more and more emphasis would be placed on minimizing the penalty.

Based on the individual error function (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 network e_i is obtained as follows:

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

From (3), the error function of NCL (when $\lambda = 1$) 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 I, the learner that only minimizes MSE is prone to overfitting the noise.

When λ is set to zero or a small positive value, the training of NCL corresponds to independently training these individual estimators without regularization, and this might lead to overfitting as well. Although NCL can use the penalty coefficient to explicitly alter the emphasis on the individual MSE and correlation portions of the ensemble and thus improve the performance to some extent. However, NCL could not overcome the overfitting problem by only tuning this parameter, especially when dealing with nontrivial noise data. In Section IV, we give the empirical evidence that NCL, including $\lambda = 1$ and λ selected by cross validation, is prone to overfitting.

In order to improve the generalization ability of NCL, in the next section, we propose RNCL.

B. Regularized Negative Correlation Learning

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

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

where $\mathbf{w}_i = [w_{i,1}, \ldots, w_{i,n_i}]^T$ 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 [16] term for the entire ensemble. In order to train each neural network with its regularization, we decompose the regularization term to M parts, each part for a network. The error function for network i can be obtained as follows:

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

Comparing this error function with the cost function of NCL (1), RNCL imposes a regularization term on every individual

1. Let M be the final number of predictors required.
2. Take a training set $\{\mathbf{x}_n, y_n\}_{n=1}^N$ and the initial regularization parameter α_i, $i = 1, \cdots, M$.
3. For the training set
   - Calculate $f_{ens}(\mathbf{x}_n) = \frac{1}{M}\sum_{i=1}^M f_i(\mathbf{x}_n)$
   - For each network from $i = 1$ to M do: for each weight $w_{i,j}$ in network i, perform a desired number of updates,

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

$$\frac{\partial e_i}{\partial w_{i,j}} = \frac{2}{M}\sum_{n=1}^N (f_i(\mathbf{x}_n) - y_n)\frac{\partial f_i(\mathbf{x}_n)}{\partial w_{i,j}} - \frac{2}{M}\sum_{n=1}^N (f_i(\mathbf{x}_n) - f_{ens}(\mathbf{x}_n))(1 - \frac{1}{M})\frac{\partial f_i(\mathbf{x}_n)}{\partial w_{i,j}} + 2\alpha_i w_{i,j}$$

4. Parameter Optimization by Bayesian inference.
5. Repeat from step 3 for a desired number of iterations.

Fig. 1. Regularized negative correlation learning algorithm.

neural network and it optimizes the regularization parameter α_i instead of the correlation parameter λ.

RNCL is implemented by scaled conjugate gradient (SCG) [25] algorithm to fast train neural networks. According to (5), 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 Fig. 1.

C. Bayesian Interpretation and Regularized Parameter Optimization

This section describes the probabilistic interpretation of RNCL, the function of the regularization term, and how to optimize these parameters by Bayesian inference. We separate this section 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:

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

where e_n is an 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 $\boldsymbol{\mu} = [\mu_1, \mu_2, \ldots, \mu_M]$[1] and β, the weigh vector $\mathbf{w} = [\mathbf{w}_1^T, \ldots, \mathbf{w}_M^T]^T$ can be obtained by maximizing the posterior $P(\mathbf{w}|D)$

$$P(\mathbf{w}|D) = \frac{P(D|\mathbf{w},\beta)P(\mathbf{w}|\boldsymbol{\mu})}{P(D|\boldsymbol{\mu},\beta)} \tag{6}$$

where the probability $P(D|\boldsymbol{\mu},\beta)$ 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:

$$P(\mathbf{w}|\boldsymbol{\mu}) = \prod_{i=1}^M \left(\frac{\mu_i}{2\pi}\right)^{n_i/2} \exp\left(-\frac{1}{2}\mu_i \mathbf{w}_i^T \mathbf{w}_i\right) \tag{7}$$

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

The traditional Bayesian methods [19], [24], [26] often use an isotropic Gaussian prior over weights $\mathbf{w}$ where the covariance matrix is an identity matrix multiplied by a parameter, which means these weights in the learner share the same prior. RNCL extends this by imposing different regularization parameters for different networks in the ensemble. The prior of RNCL becomes a block-isotropic Gaussian prior whose covariance matrix is diagonal matrix with M different values. That is, each network has its own different prior.

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

$$P(D|\mathbf{w},\beta) = \prod_{n=1}^N \left(\frac{\beta}{2\pi}\right)^{1/2} \exp\left(-\frac{\beta}{2}e_n^2\right). \tag{8}$$

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

$$P(\mathbf{w}|D) \propto \exp\left(-\frac{\beta}{2}\sum_{n=1}^N e_n^2\right) \cdot \exp\left(-\sum_{i=1}^M \frac{\mu_i}{2}\mathbf{w}_i^T \mathbf{w}_i\right). \tag{9}$$

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

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

$$\min J_1(\mathbf{w}) = \frac{1}{2}\beta\sum_{n=1}^{N} e_n^2 + \frac{1}{2}\sum_{i=1}^{M}\mu_i \mathbf{w}_i^T\mathbf{w}_i. \tag{10}$$

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

$$\begin{aligned} P(\mathbf{w}|D) &= \frac{\exp\left(-J_1(\mathbf{w})\right)}{\int \exp\left(-J_1(\mathbf{w})\right) d\mathbf{w}} \\ &\approx \frac{\exp\left(-\frac{1}{2}(\mathbf{w}-\mathbf{w}_{MP})^T A(\mathbf{w}-\mathbf{w}_{MP})\right)}{(2\pi)^{W/2}|A|^{-\frac{1}{2}}} \end{aligned} \tag{11}$$

where A is the Hessian matrix of the cost function J_1 and the subscript MP indicates the most probable values.

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

Comparing (10) with (4), RNCL is equivalent to maximization of the posterior under Bayesian framework. The likelihood $P(D|\mathbf{w},\beta)$ corresponds to the empirical training error term and the prior over weight vector $P(\mathbf{w}|\boldsymbol{\mu})$ corresponds to the regularization term. The regularization term penalizes large weights, causing the weights to converge to smaller absolute values than they otherwise would.

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

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 $O(Mn_i^3)$. Since M, the size of ensemble, is often set to be equal or greater than 25, the reduction of computational complexity is nontrivial.

Although there are two types of parameters: μ_i and β, the minimization of J_1 only depends on the ratios $\alpha_i = \mu_i/\beta$. 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 most probable values of $\boldsymbol{\mu}$ and β, we need to maximize the posterior of $P(\boldsymbol{\mu},\beta|D)$.

According to Bayesian rule, the posteriors of $\boldsymbol{\mu}$ and β are obtained by

$$P(\boldsymbol{\mu},\beta|D) = \frac{P(D|\boldsymbol{\mu},\beta)P(\boldsymbol{\mu},\beta)}{P(D)} \propto P(D|\boldsymbol{\mu},\beta) \tag{12}$$

where flat priors are assumed on the hyperparameters $\boldsymbol{\mu}$ and β. According to (7), (8), and (11), the marginal likelihood[2] can be obtained in the following way [20]:

$$\begin{aligned} P(D|\boldsymbol{\mu},\beta) &= \frac{P(D|\mathbf{w},\beta)P(\mathbf{w}|\boldsymbol{\mu})}{P(\mathbf{w}|D)} \\ &\approx \frac{(2\pi)^{W/2}|A|^{-\frac{1}{2}}\prod_{i=1}^{M}\left(\frac{\mu_i}{2\pi}\right)^{n_i/2}\left(\frac{\beta}{2\pi}\right)^{N/2}}{\exp\left(-\frac{1}{2}(\mathbf{w}-\mathbf{w}_{MP})^T A(\mathbf{w}-\mathbf{w}_{MP})\right)} \\ &\quad \cdot \exp\left(-J_1(\mathbf{w})\right). \end{aligned} \tag{13}$$

By using the Gaussian approximation $J_1(\mathbf{w}) \approx J_1(\mathbf{w}_{MP}) + (1/2)(\mathbf{w}-\mathbf{w}_{MP})^T A(\mathbf{w}-\mathbf{w}_{MP})$ and the relation $W = \sum n_i$, where W is the total number of weights in the ensemble

$$\begin{aligned} P(D|\boldsymbol{\mu},\beta) &\approx (2\pi)^{W/2}|A|^{-\frac{1}{2}}\prod_{i=1}^{M}\left(\frac{\mu_i}{2\pi}\right)^{n_i/2}\left(\frac{\beta}{2\pi}\right)^{N/2} \\ &\quad \cdot \exp\left(-J_1(\mathbf{w}_{MP})\right) \\ &\approx \left(\frac{1}{2\pi}\right)^{N/2}\sqrt{\frac{\prod_{i=1}^{M}\mu_i^{n_i}\beta^N}{\det A}}\exp\left(-J_1(\mathbf{w}_{MP})\right) \\ &\propto \sqrt{\frac{\prod_{i=1}^{M}\mu_i^{n_i}\beta^N}{\det A}}\exp\left(-J_1(\mathbf{w}_{MP})\right). \end{aligned} \tag{14}$$

In order to maximize the probability $P(D|\boldsymbol{\mu},\beta)$, negative logarithm is applied

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

where the subscript MP indicates the most probable values.

Setting the gradient to zero, we can get the most probable $\alpha_i = \mu_i/\beta$ (refer to Appendix II for detail)

$$\alpha_i^{MP} = \frac{\sum_{n=1}^{N}e_{n,MP}^2}{\mathbf{w}_{i,MP}^T\mathbf{w}_{i,MP}} \frac{\left(n_i - \sum_{j\in n_i}\frac{\alpha_i}{\zeta_j+\alpha_i}\right)}{\left(N - \sum_{j=1}^{W}\frac{\zeta_j}{\zeta_j+\hat{\alpha}_j}\right)} \tag{15}$$

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

The RNCL learning is conducted in an iterative manner. In each iteration, the ensemble is first trained (step 3 in Fig. 1) by SCG algorithm with previous regularization parameters α_i, followed by the estimation of new most probable α_i^{MP} values by (15) (step 4 in Fig. 1), and then we incorporate the new α_i^{MP} in the ensemble. The learning algorithm repeats the process (steps 3 and 4 in Fig. 1), until some suitable convergence criteria have been satisfied.

[2]The Gaussian approximation has been employed in Equation (11) to calculate the integral $\int \exp(-J_1(\mathbf{w}))d\mathbf{w}$. Refer to Appendix I for details.

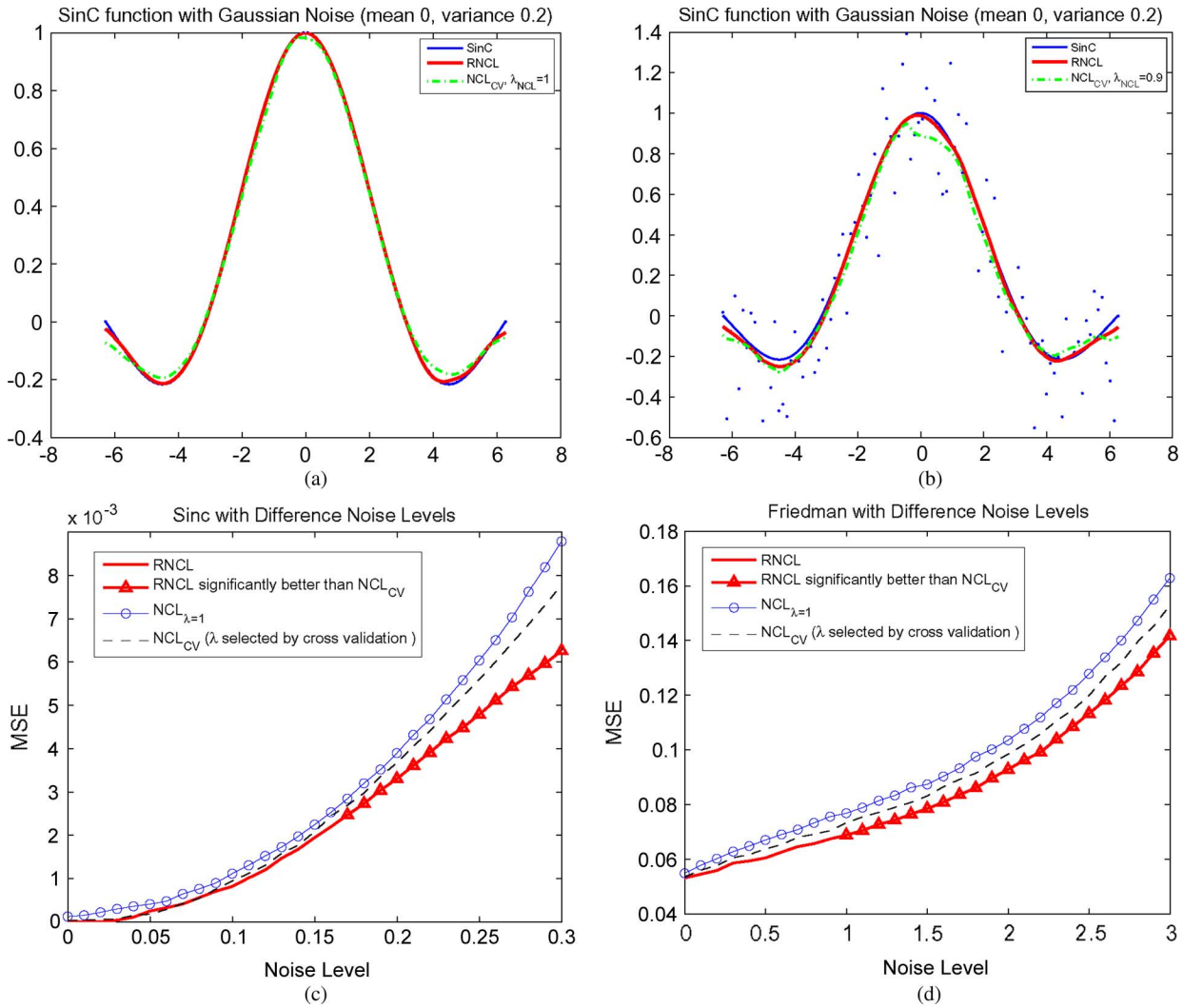

Fig. 2. Comparison of RNCL and NCL on regression data sets: sinc and Friedman test. In (a) and (b), the wide, dashed, and solid lines are obtained by RNCL, $\mathrm{NCL_{CV}}$, and the noise-free function, respectively. In (c) and (d), MSE of RNCL (solid), $\mathrm{NCL}_{\lambda=1}$ (circled), and $\mathrm{NCL_{CV}}$ (dashed) on sinc and Friedman test with different noise levels are shown. A statistical t-test (95% significance level) is conducted to compare RNCL with $\mathrm{NCL_{CV}}$ and the triangles represent those points where RNCL significantly outperforms $\mathrm{NCL_{CV}}$. We do not report λ values in (c) and (d). Since for each noise level [$\{0, 0.01, \ldots, 0.3\}$ in (c) and $\{0, 0.1, \ldots, 3\}$ in (d)], $\mathrm{NCL_{CV}}$ uses cross validation to search a λ value and thus there are 31 λ values for each data set. (a) Sinc free of noise. (b) Sinc with Gaussian noise (mean 0, variance 0.2). (c) Sinc with different noise levels. (d) Friedman with different noise levels.

Similarly to other methods using Bayesian evidence framework [19], [24], [26], the α estimation requires the computation of the Hessian's eigenvalues in each iteration. The resulting computational cost is $O(W^3)$, where W is the total number of weights in the ensemble. When the eigenvalue ζ_j is calculated, the update rule of α_i^{MP} involves only vector products that can be evaluated very quickly. In order to reduce the computational complexity for large ensembles, one can choose to calculate only the largest eigenvalues using the expectation maximization approach [27].

IV. Experimental Analysis

In this section, we will present the experimental results of RNCL. First, 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 problems) with different noise levels to study the characteristics of RNCL and NCL with noise data. Second, we carry out extensive experiments on eight benchmark regression data sets and 13 benchmark classification data sets to compare the performance of RNCL, NCL,[3] and Bagging.[4]

A. Experimental Setup

In the experiments, RBF neural networks (NNs) are used as the base learners. The number of hidden nodes is randomly selected but restricted in the range 3–12. The initial centers,

[3]We have implemented two versions of NCL algorithms. The first $\mathrm{NCL}_{\lambda=1}$ uses $\lambda = 1$ and the second algorithm $\mathrm{NCL_{CV}}$ selects the λ using cross validation within the range $\lambda \in \{0, 0.1, \ldots, 1\}$. We notice that λ could be a little greater than 1 [$\lambda \leq (M/M-1)$, M is the ensemble size] to guarantee the positive definite of Hessian matrix [9]. Since we use $M = 25$ in this paper, the up-bound of λ (1.0417) is close to 1 and we will not use the λ values which are greater than 1.

[4]We have implemented two versions of Bagging algorithms. The first Bagging ensemble consists of 100 RBF networks with regularization, in which the regularization parameters are randomly selected in the range $\{0, 0.1, \ldots, 2\}$, and the second has 100 RBF networks without regularization.

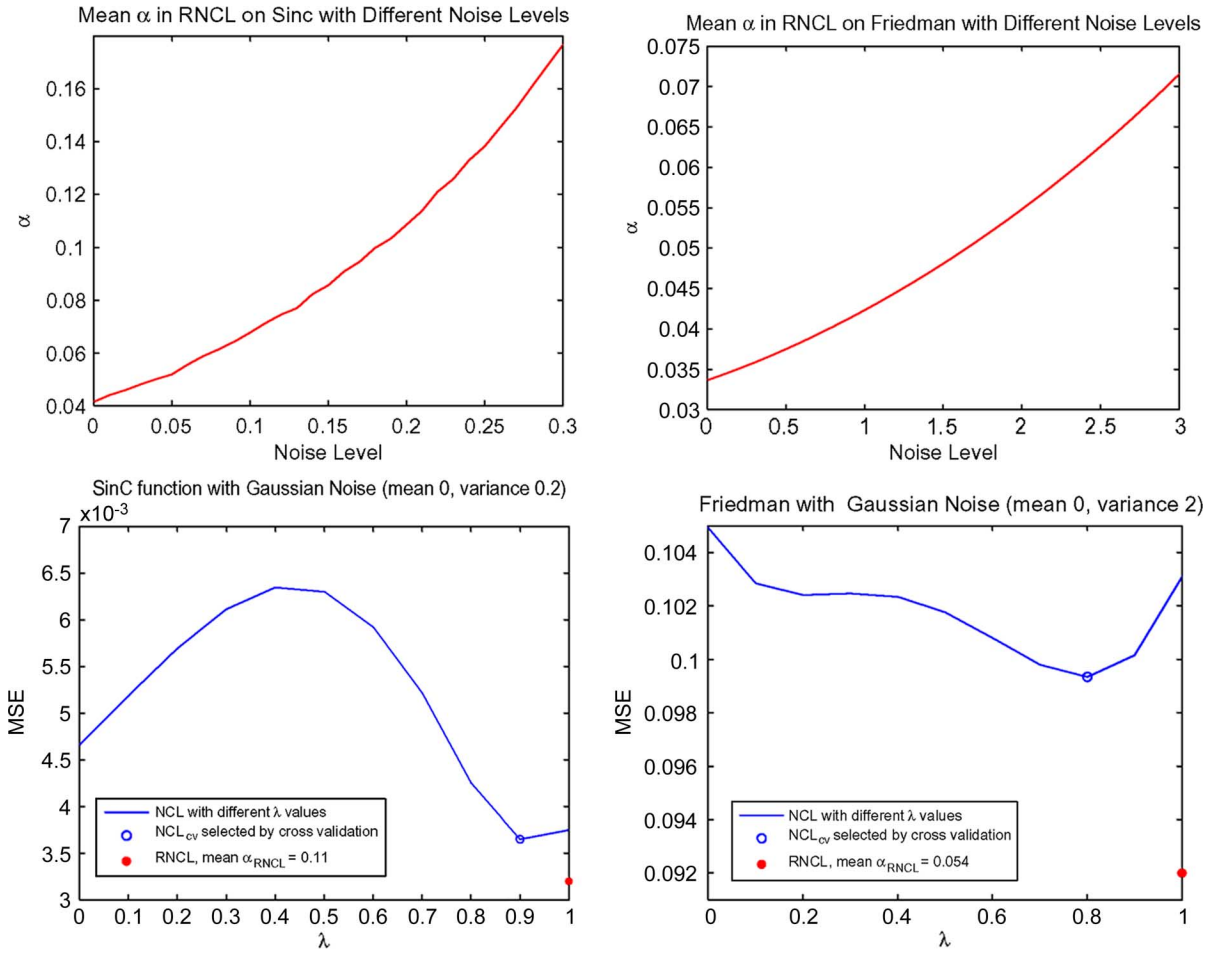

Fig. 3. First row reports the mean α value obtained in RNCL versus different noise levels on sinc and Friedman data. Results are based on 100 runs. The second row shows the selection of λ () and the performance of RNCL, on sinc (0.2 noise level) and Friedman test (2 noise level).

widths for individual NNs, are randomly selected. The details of the specification including the derivations of error with respect to centers and widths of an RBF network are presented in Appendix III.

We employ SCG algorithm to train NCL and RNCL. We use 25 NNs to constitute the ensemble of NCL and RNCL. For Bagging, we employ 100 NNs 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, $\text{NCL}_{\lambda=1}$, and NCL_{CV} on two synthetic regression data sets: sinc and Friedman test. Fig. 2(a) and (b) shows the output of RNCL and NCL_{CV} on sinc function with different noise levels. In the noise-free case, both RNCL and NCL_{CV} perfectly approximate the actual function, though there is a little misfit for NCL_{CV} near the tail. When the noise level increases, NCL_{CV}, though selects the parameter by cross validation, overfits the noise in the training set, while RNCL is more robust to noise than NCL; refer to Fig. 2(b).

In order to explore the behavior of RNCL, NCL_{CV}, and $\text{NCL}_{\lambda=1}$ with different noise levels, we add mean zero and different levels of Gaussian noise to sinc and Friedman test problems. Fig. 2(c) and (d) shows the average results of 100 runs. Since the standard deviations of the targets: sinc and Friedman test, are different, the range of noise levels is different in Fig. 2(c) and (d).

For sinc data set, when the noise level (variance) is small, RNCL and NCL_{CV} perform similarly and their performances are better than $\text{NCL}_{\lambda=1}$. When the noise level becomes greater, MSE of RNCL increases slower than that of NCL_{CV} and $\text{NCL}_{\lambda=1}$ and NCL_{CV} performs a litter better than $\text{NCL}_{\lambda=1}$. For Friedman test data set, RNCL outperforms NCL_{CV} and $\text{NCL}_{\lambda=1}$ all the time and the difference between RNCL and NCL becomes greater when noise level becomes greater.

We also conduct statistical t-test (95% significance level) to compare RNCL and NCL_{CV} and record the significant points as triangles. From both figures, RNCLs significantly outperform NCL_{CV} and $\text{NCL}_{\lambda=1}$ when noise level becomes high.

In Fig. 3, we have illustrated the mean of regularization parameters α obtained in RNCL versus different noise levels and the parameter λ selected by NCL_{CV}.

The first row in Fig. 3 reports the mean α value[5] obtained in RNCL versus different noise levels on sinc and Friedman data. The results are based on 100 runs. When the noise level

[5]Since we optimize α_i for each individual networks in the ensemble, in this figure, we only show the mean α_i value.

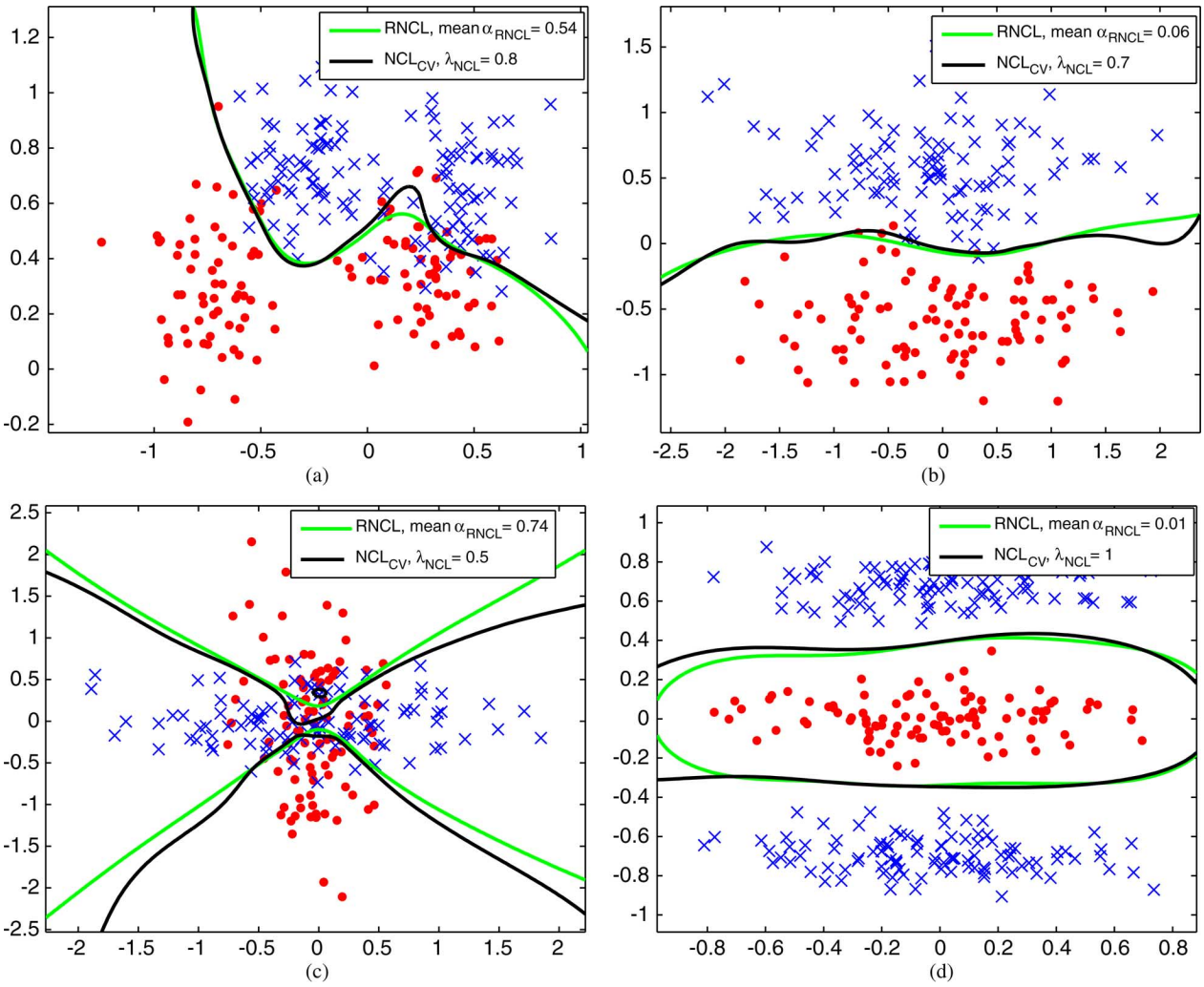

Fig. 4. Comparison of RNCL and $\mathrm{NCL_{CV}}$ on four synthetic classification data sets. Two classes are shown as crosses and dots. The separating lines were obtained by projecting test data over a grid. The wide and thin lines were obtained by RNCL and $\mathrm{NCL_{CV}}$, respectively. (a) Synth. (b) Overlap. (c) Bumpy. (d) Relevance.

increases, the learner will become complex to fit the data, and in this situation, large regularization is preferred to control the complexity in the model. Bayesian parameter selection in RNCL does reflect this tendency when the noise level increases.

The second row in Fig. 3 reports the selected λ parameter in $\mathrm{NCL_{CV}}$ and the performance of RNCL. It is observed that $\mathrm{NCL_{CV}}$ could not beat RNCL even if it uses the optimal correlation parameter λ. Figs. 3 and 2 confirm that $\mathrm{NCL_{CV}}$ could not overcome the overfitting problem by only tuning the λ parameter for regression problems.

In the following, we demonstrate the application of RNCL on classification problems. First, we apply RNCL and $\mathrm{NCL_{CV}}$ on four synthetic data in two dimensions in order to illustrate graphically the decision boundary.

These four data sets are: 1) *synth* is generated from mixtures of two Gaussians by [28]; 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°, quadratic boundaries are required; 4) *relevance* is a case where only one dimension of the data is relevant to separating the data.

In Fig. 4, we present a comparison of RNCL and $\mathrm{NCL_{CV}}$. 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*, and RNCL produces a smoother boundary than $\mathrm{NCL_{CV}}$.

We observe that RNCL gives more accurate results in the other cases. In the cases of *synth*, RNCL disregards the outliers in the training points and produces smooth boundary, while $\mathrm{NCL_{CV}}$ generates a corner in the decision boundary due to several outliers. In the case of *bumpy*, the noise level is great because of these overlapping points. $\mathrm{NCL_{CV}}$ does not generalize very well and produces a little twisty boundary. RNCL generates a quadratic boundary according to the expectation.

In order to check the behavior of RNCL and $\mathrm{NCL_{CV}}$ on noise classification problems, we conduct similar noise experiments as the regression problems. In the experiments, we select two data sets: synth and Gaussian. Gaussian is a synthetic two-class 2-D data set which is sampled from a mixture of four Gaussians. Each class is associated with two of the Gaussians so that the optimal decision boundary is nonlinear.

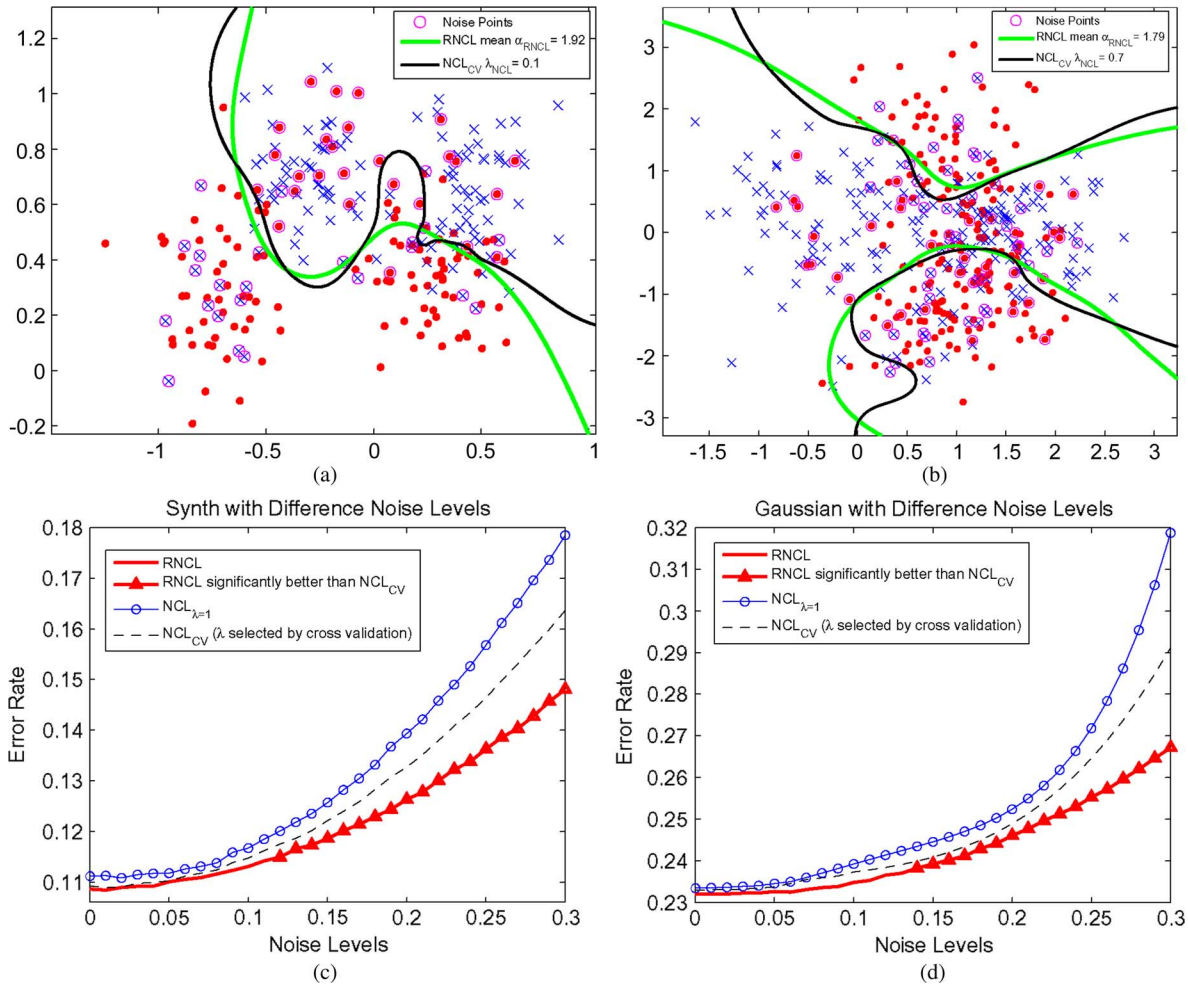

Fig. 5. Comparison of RNCL and $\mathrm{NCL_{CV}}$ on two classification data sets. Two classes are shown as crosses and dots. The separating lines were obtained by projecting test data over a grid. In (a) and (b), the decision boundaries in solid and dark are obtained by RNCL and $\mathrm{NCL_{CV}}$, respectively. The randomly selected noise points are marked with a circle. In (c) and (d), classification error of RNCL (solid), $\mathrm{NCL}_{\lambda=1}$ (circled), and $\mathrm{NCL_{CV}}$ (dashed) versus noise levels on synth and Gaussian data sets are shown. The results are based on 100 runs. A statistical t-test (95% significance level) is conducted to compare RNCL with $\mathrm{NCL_{CV}}$ and the triangles represent those points where RNCL significantly outperforms $\mathrm{NCL_{CV}}$. We do not report λ values in (c) and (d). Since for each noise level $[\{0, 0.01, \ldots, 0.3\}$ in (c) and (d)], $\mathrm{NCL_{CV}}$ uses cross validation to search a λ value and thus there are 31 λ values for each data set. (a) Synth with 20% noise. (b) Gaussian with 20% noise. (c) Synth with difference noise levels. (d) Gaussian with different noise levels.

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 Fig. 5. Fig. 5(a) and (b) visualizes the decision boundaries of RNCL and $\mathrm{NCL_{CV}}$ with 20% noise points.

Although the noise level is high, RNCL produces smooth boundary. $\mathrm{NCL_{CV}}$ does not generalize well. We also plot the curve [Fig. 5(c) and (d)] of classification error versus 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 $\mathrm{NCL_{CV}}$ and $\mathrm{NCL}_{\lambda=1}$.

We also conduct statistical t-test (95% significance level) to compare RNCL and $\mathrm{NCL_{CV}}$ and record the significant points as triangles. From both figures, RNCL significantly outperforms $\mathrm{NCL_{CV}}$ and $\mathrm{NCL}_{\lambda=1}$ when noise level becomes great.

In Fig. 6, we have illustrated the regularization parameter α obtained in RNCL versus different noise levels and the selected λ values by $\mathrm{NCL_{CV}}$.

The first row in Fig. 6 reports the mean α value in by RNCL versus different noise levels on synth and Gaussian data. The results are based on 100 runs. Similarly to regression problems, large regularization parameter α is preferred to control the complexity of the model training with large noise data. Bayesian parameter selection in RNCL does reflect this tendency when the noise level increases.

The second row in Fig. 6 reports the selected λ parameter in $\mathrm{NCL_{CV}}$ and the performance of RNCL. It is observed that $\mathrm{NCL_{CV}}$ could not beat RNCL even if it uses the optimal correlation parameter. Both Figs. 6 and 5 confirm that $\mathrm{NCL_{CV}}$ could not overcome the overfitting problem by only tuning the λ parameter for classification problems.

The results of RNCL are promising on these regression and classification problems. Based on the results and analysis, the regularization term does work in RNCL and improves its ability against noise, which is especially important in practice since most of the actual data are contaminated by noise. Section IV-D will conduct experiments to compare Bayesian inference and cross-validation search for the regularization parameter α in RNCL. After the analysis with synthetic data sets, the next section presents the results for the real-world benchmark problems.

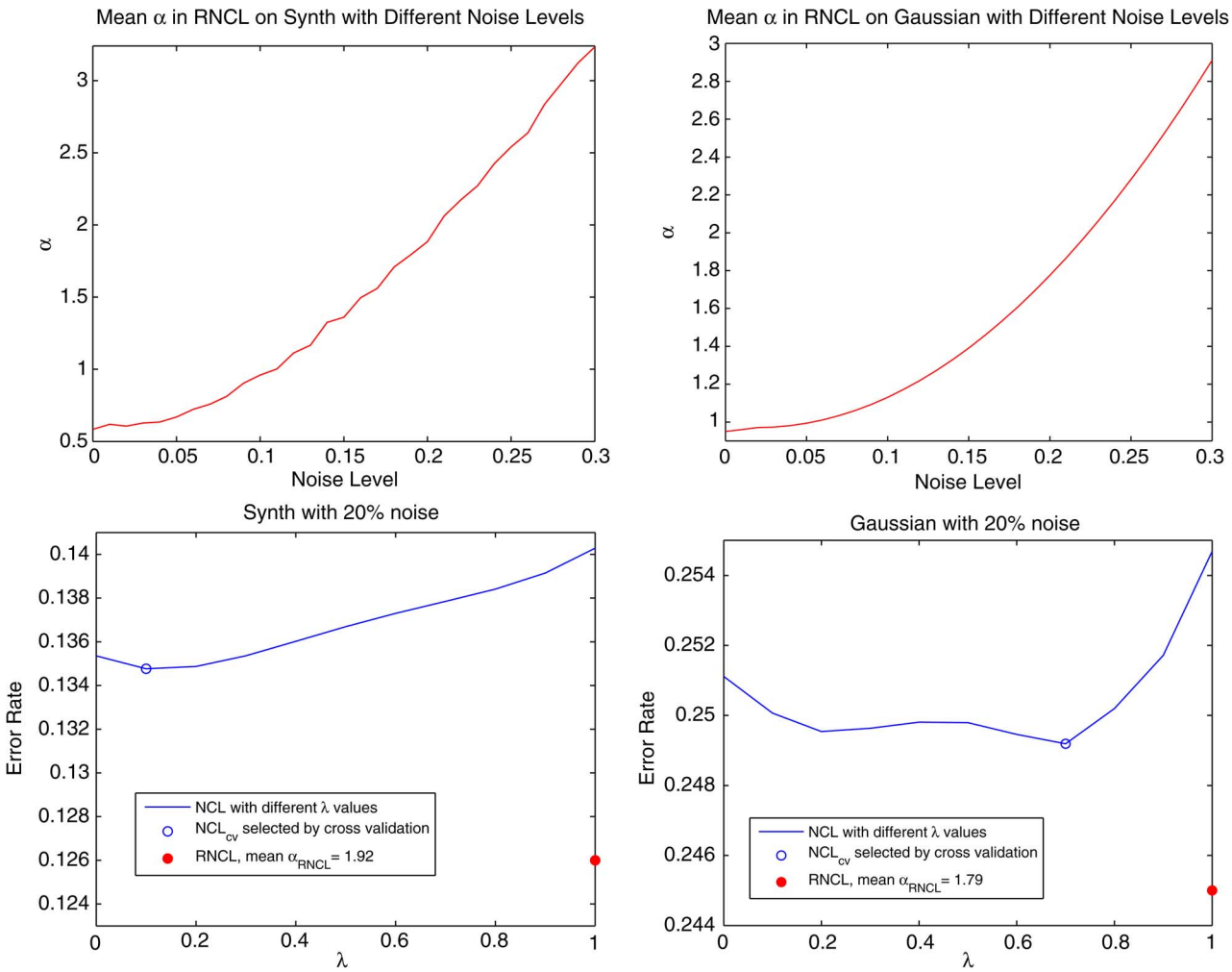

Fig. 6. First row reports the mean α value in RNCL versus different noise levels using synth and Gaussian data. Results are based on 100 runs. The second row shows the selected λ by $\mathrm{NCL}_{\mathrm{CV}}$ and the performance of RNCL (solid dot) on synth (20% noise level) and Gaussian (20% noise level).

C. Benchmark Results

In order to evaluate the performance of RNCL, we test RNCL, $\mathrm{NCL}_{\mathrm{CV}}$, $\mathrm{NCL}_{\lambda=1}$, $\mathrm{Bagging}_{\mathrm{reg}}$ (100 RBF networks with regularization, in which the regularization parameters are randomly selected in the range $\{0, 0.1, \ldots, 2\}$), $\mathrm{Bagging}_{\mathrm{unreg}}$ (100 RBF networks without regularization), and $\mathrm{RBF}_{\mathrm{Bayes}}$[6] on eight regression benchmark problems and 13 classification benchmark problems. The information on the data sets used for regression is presented in Table I. The Mexican hat has been used by Weston *et al.* [29] 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 [30] in comparing several ensemble approaches. Plane has been used by Ridgeway *et al.* [31] in exploring the performance of boosted naive Bayesian regressors. The Boston house data set is obtained from the University of California at Irvine (UCI, Irvine, CA) machine learning repository [32]. 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 have been preprocessed and organized by Rätsch *et al.*[7] These data sets include one synthetic set (banana) along with 12 other real-world data sets coming from the UCI [32], DELVE,[8] 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 Rätsch's implementation and we generate additional 80 folds by random sampling to make the experiments consistent). In addition, every instance is normalized dimensionwise to have zero mean and unit standard deviation.

Table III reports the performance of these algorithms on the eight benchmark regression data sets. According to the tables, RNCL performs quite favorably in these data sets. For example,

[6]The regularization parameter in single RBF network is optimized by Bayesian inference.

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

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

TABLE I
SUMMARY OF REGRESSION DATA SETS

Data Sets	Function	Variable	Training Points	Test Points
Mexican Hat	$y = sinc\|x\| = \frac{\sin\|x\|}{\|x\|}$	$x \sim U[-2\pi, 2\pi]$	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

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

RNCL outperforms the other methods in seven out of eight data sets, in which four wins are significant against NCL_{CV} and three wins are significant against $\text{Bagging}_{\text{reg}}$.

The performance of RNCL, NCL_{CV}, $\text{NCL}_{\lambda=1}$, $\text{Bagging}_{\text{unreg}}$, and $\text{Bagging}_{\text{reg}}$, $\text{RBF}_{\text{Bayes}}$ on classification problems has been reported in Table IV. Based on the table, RNCL performs well since RNCL outperforms all the other methods in nine out of 13 data sets, and comes second in four cases.

According to these results, $\text{NCL}_{\lambda=1}$ and NCL_{CV} achieve similar outperform as RNCL in these cases: image, ringnorm, and twonorm, which are both synthetic data with few noise (see the lower error rate). This observation also validates that $\text{NCL}_{\lambda=1}$ achieves good results when noise is small and RNCL is more robust to noise than NCL.

From the table, $\text{Bagging}_{\text{reg}}$ outperforms $\text{Bagging}_{\text{unreg}}$ in most of cases. This empirical results support the statement that Bagging benefits from combining simple learners to succeed [10], [33] since regularized networks are simpler than unregularized ones.

We also notice that $\text{RBF}_{\text{Bayes}}$ is not as competitive as other ensemble methods, though it uses Bayesian inference to optimize the regularization parameter. This also confirms that the application of Bayesian inference on ensemble systems and the decomposition of ensemble training into a group of subtasks are beneficial.

D. Comparison of Bayesian Inference With Cross Validation

In the previous experiments, we notice that RNCL achieves better generalization than NCL_{CV} in most cases, especially on the data sets with nontrivial noise. In this section, we would like to compare two RNCL versions. The first is $\text{RNCL}_{\text{Bayes}}$, whose regularization parameters are optimized by Bayesian inference and another one is RNCL_{CV},[9] whose regularization parameter is selected by cross validation.

First, we conduct the experiments on two regression and classification data sets, which have been used in Section IV-B, i.e., the sinc function with 0.2 Gaussian noise and Friedman data set with 2 Gaussian noise for regression problems and synth and Gaussian with 20% noise for classification problems.

Since it is computationally costly to search several $\alpha_i s$ by cross validation, we employ the single α in $\text{RNCL}_{\text{Bayes}}$ (i.e., $\alpha_i = \alpha_j = \alpha$) to facilitate comparison. The experimental results are illustrated in Fig. 7.

It can be observed from Fig. 7 that the error rate or MSE is reduced with the increase of α in the beginning but after α reaches the minimum and increases, the error is increased with α. The reason is that the learner might *overfit* the noise with a small regularization term in the beginning. When a large α is presented, the system might be over-regularized and thus *under-fit* the data. $\text{RNCL}_{\text{Bayes}}$ is able to find the optimal or near optimal α values automatically.

In Table V,[10] we compare the performance of $\text{RNCL}_{\text{Bayes}}$ and RNCL_{CV} on eight data sets based on the average results of 100 runs. In RNCL_{CV}, the search ranges are $\alpha \in \{0, 0.01, 0.02 \ldots, 0.3\}$ for regression problems (the data has been normalized to unit standard deviation) and $\alpha \in \{0, 0.1, 0.2 \ldots, 3\}$ for classification problems, respectively.

Based on this table, we observe that $\text{RNCL}_{\text{Bayes}}$ obtains similar performance as RNCL_{CV} while $\text{RNCL}_{\text{Bayes}}$ costs much less time than RNCL_{CV}. Another potential problem with cross-validation search is that for different problems, the optimal α values are in different ranges. Once an inappropriate search range, either too small or too large, is specified, the obtained regularization parameter would not work and we need to respecify the search range and rerun the cross validation.

Based on Fig. 7 and Table V, the Bayesian inference proposed in this paper is able to select an appropriate regularization term according to different problems and thus improve the performance of NCL.

[9]In RNCL_{CV}, we assume that these networks have the same regularization parameter (i.e., $\alpha_i = \alpha_j = \alpha$) to reduce the search space of cross validation.

[10]To be consistent, we add additive Gaussian noise [one-third of standard deviation of the target $y(x)$] to benchmark regression data sets, except Boston house data set. Since the noise levels of sinc and Friedman in Table V are less than those in Fig. 7 (0.2 noise level for sinc and 2 noise level for Friedman). The MSEs in Table V are smaller than those in Fig. 7 for sinc and Friedman data sets.

TABLE III
COMPARISON OF RNCL, NCL_{CV}, $\text{NCL}_{\lambda=1}$, $\text{Bagging}_{\text{reg}}$, $\text{Bagging}_{\text{unreg}}$, AND $\text{RBF}_{\text{Bayes}}$ ON EIGHT REGRESSION DATA SETS, BY MSE (STANDARD DEVIATION). A WIN–LOSS–TIE SUMMARIZATION BASED ON MEAN VALUE AND t-TEST (95% SIGNIFICANCE LEVEL) IS ATTACHED AT THE BOTTOM OF THE TABLE. THESE RESULTS ARE AVERAGES OF 100 RUNS

Data Sets	RNCL	NCL_{CV}	$\text{NCL}_{\lambda=1}$	Bagging_{reg}	Bagging_{unreg}	RBF_{Bayes}
Mexican Hat	**6.4e-3(9.1e-4)**	6.9e-3(1.0e-3)	7.4e-3(9.4e-4)	6.5e-3(1.1e-3)	7.2e-3(1.3e-3)	7.0e-3(1.0e-3)
Friedman	**0.82(0.21)**	0.93(0.21)	1.02(0.23)	0.90(0.21)	1.13(0.31)	0.97(0.27)
Gabor	**0.008(0.003)**	0.009(0.003)	0.012(0.002)	**0.008(0.004)**	0.009(0.004)	0.010(0.003)
Multi	**0.028(0.006)**	0.036(0.008)	0.042(0.009)	0.030(0.010)	0.038(0.010)	0.033(0.012)
Plane	1.21e-4(4.6e-5)	1.22e-4(5.2e-5)	1.28e-4(5.2e-5)	**1.18e-4(4.8e-5)**	1.26e-4(5.4e-5)	1.42e-4(6.9e-5)
Polynomial	**0.073(0.018)**	0.091(0.023)	0.128(0.031)	0.083(0.018)	0.118(0.025)	0.086(0.029)
Sinc	**2.09e-3(4.6e-4)**	2.21e-3(5.1e-4)	2.32e-3(5.4e-4)	2.19e-3(4.4e-4)	2.41e-3(5.3e-4)	2.21e-3(4.6e-4)
Boston House	**10.86(2.74)**	11.06(2.62)	14.06(3.05)	11.41(3.71)	12.17(3.46)	11.52(3.59)
mean W-L-T	-	0-8-0	0-8-0	1-6-1	0-8-0	0-8-0
Significant W-L-T	-	0-4-4	0-7-1	0-3-5	0-6-2	0-5-3

TABLE IV
COMPARISON OF RNCL, NCL_{CV}, $\text{NCL}_{\lambda=1}$, $\text{Bagging}_{\text{reg}}$, $\text{Bagging}_{\text{unreg}}$, AND $\text{RBF}_{\text{Bayes}}$ ON 13 BENCHMARK DATA SETS, BY PERCENT ERROR (STANDARD DEVIATION). A WIN–LOSS–TIE SUMMARIZATION BASED ON MEAN VALUE AND t-TEST (95% SIGNIFICANCE LEVEL) IS ATTACHED AT THE BOTTOM OF THE TABLE. THESE RESULTS ARE AVERAGES OF 100 RUNS

	RNCL	NCL_{CV}	$\text{NCL}_{\lambda=1}$	Bagging_{reg}	Bagging_{unreg}	RBF_{Bayes}
Banana	**10.40(0.63)**	10.86(0.68)	11.22(0.68)	10.94(0.69)	11.41(0.78)	11.21(0.42)
Cancer	**26.27(4.77)**	26.89(4.59)	28.12(4.61)	27.36(4.87)	28.13(4.87)	27.64(4.71)
Diabetics	**23.16(1.62)**	24.38(1.68)	25.47(1.96)	24.68(1.78)	25.23(1.83)	25.31(1.88)
Solar	**33.86(1.71)**	34.64(1.88)	35.42(1.79)	34.97(1.51)	35.73(1.51)	34.71(1.96)
German	**24.01(2.23)**	24.63(2.35)	25.88(2.19)	24.76(2.07)	24.91(2.13)	25.27(2.38)
Heart	16.43(3.11)	17.82(3.78)	18.28(3.68)	**16.32(3.10)**	18.71(2.93)	17.85(3.31)
Image	**2.65(0.68)**	2.73(0.46)	2.79(0.84)	2.76(0.72)	2.68(0.69)	3.32(0.65)
Ringnorm	1.63(0.19)	**1.62(0.24)**	1.69(0.26)	1.71(0.24)	1.79(0.31)	1.80(0.21)
Splice	**10.31(0.74)**	11.03(0.83)	11.64(0.71)	10.41(0.69)	11.62(0.62)	11.15(0.82)
Thyroid	**4.03(2.11)**	4.45(2.37)	4.87(3.13)	4.33(2.16)	4.48(2.36)	4.52(2.12)
Titanic	22.42(1.23)	22.42(1.24)	23.91(1.78)	**22.31(1.41)**	23.98(1.37)	23.62(1.34)
Twonorm	2.66(0.21)	**2.61(0.26)**	2.72(0.32)	2.84(0.34)	2.75(0.31)	2.94(0.29)
Waveform	**9.91(0.48)**	11.14(0.78)	12.26(0.63)	10.68(0.61)	11.68(0.62)	10.86(1.12)
mean W-L-T	-	2-10-1	0-13-0	2-11-0	0-13-0	0-13-0
Significant W-L-T	-	0-6-7	0-10-3	0-5-8	0-8-5	0-7-6

TABLE V
COMPARISON OF $\text{RNCL}_{\text{Bayes}}$ AND RNCL_{CV} ON FOUR REGRESSION PROBLEMS AND FOUR CLASSIFICATION PROBLEMS IN TERMS OF MSE/ERROR RATE AND COMPUTATIONAL TIME

Data Sets	SinC	Friedman	Gabor	House	Banana	Cancer	Diabetics	Solar
RNCL_{CV}	0.0031	0.84	0.008	10.82	10.45	26.23	23.30	33.94
Time_{CV}	102.6	142.7	96.3	314.2	105.1	139.4	256.8	320.9
RNCL_{Bayes}	0.0029	0.82	0.008	10.86	10.40	26.27	23.16	33.86
Time_{Bayes}	21.1	36.2	30.8	70.6	12.4	15.5	27.2	34.7

E. Statistical Comparisons Over Multiple Data Sets

In Section IV-D, we have conducted the statistical tests on single data sets. Statistical tests on multiple data sets for multiple algorithms are preferred for comparing different algorithms over multiple data sets [34]. In this section, we will conduct statistical tests over multiple data sets by using the Friedman test [35] with the corresponding post-hoc tests.

The Friedman test is a nonparametric 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. This paper uses an improved Friedman test proposed by Iman and Davenport [36].

The Friedman test [35] is carried out to test*whether all the algorithms are equivalent*. If the test result rejects the null hypothesis, i.e., these algorithms are not 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 the RNCL as the control classifier to be compared with. Since the baseline classification RBF algorithms are not comparable to RNCL and other ensemble algorithms, this section will analyze only three algorithms: NCL_{CV}, $\text{NCL}_{\lambda=1}$, and $\text{Bagging}_{\text{reg}}$ against the control algorithm RNCL.

The Bonferroni–Dunn test [37] 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[11] differ by at least the critical

[11]We rank these algorithms based 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. Refer to Table VI for the mean rank of these algorithms.

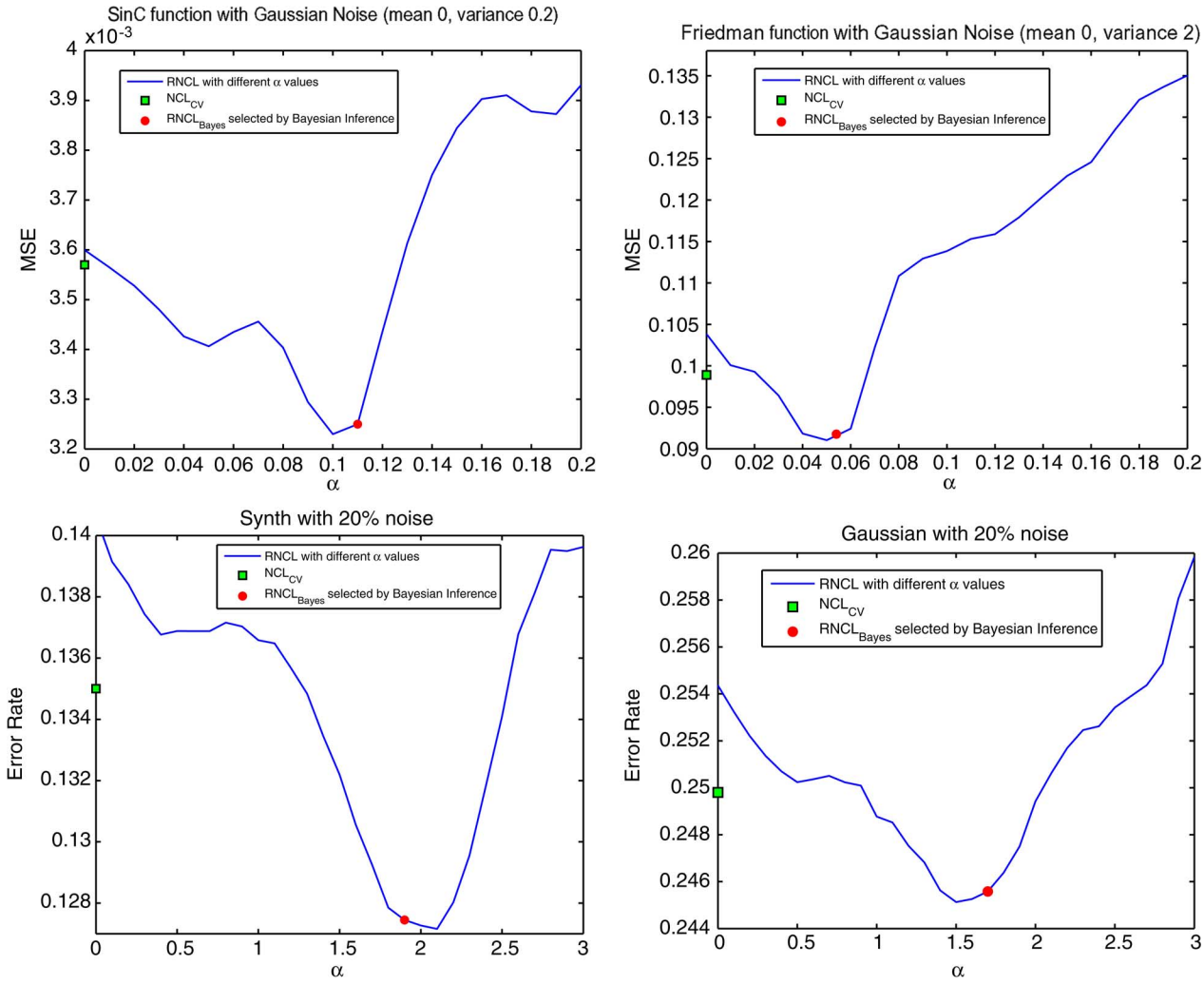

Fig. 7. Illustration of RNCL with different α values and the selected α value (dots) by Bayesian inference using four data sets. The error of NCL_{CV} (square circle) is shown to facilitate comparison.

TABLE VI
MEAN RANK OF RNCL, NCL_{CV}, $\text{NCL}_{\lambda=1}$, AND $\text{Bagging}_{\text{reg}}$

Mean Rank	RNCL	NCL_{CV}	$\text{NCL}_{\lambda=1}$	Bagging_{reg}
Regression	1.1875	2.8750	4.0000	1.9375
Classification	1.3462	2.1923	3.8462	2.6154

TABLE VII
FRIEDMAN TESTS WITH THE CORRESPONDING POST-HOC TESTS, BONFERRONI–DUNN, TO COMPARE ESTIMATORS AND CLASSIFIERS FOR MULTIPLE DATA SETS. THE THRESHOLD IS 0.10 AND $q_{0.10} = 2.128$

Algorithms	Friedman test	$\text{CD}_{0.10}$	NCL_{CV}	$\text{NCL}_{\lambda=1}$	Bagging_{reg}
Regression	0.000	1.3736	1.6875	2.8125	0.7500
Classification	0.000	1.0776	0.8461	2.5000	1.2692

difference

$$CD = q_\alpha \sqrt{\frac{j(j+1)}{6T}} \tag{16}$$

where j is the number of algorithms, T is the number of data sets, and critical values q_α can be found in [34]. For example, when $j = 4$, $q_{0.10} = 2.128$, where the subscript 0.10 is the threshold value.

Table VI lists the mean rank of these algorithms using different ensemble training algorithms. Table VII gives the Friedman test results. Since we employ the same threshold 0.10 for these ensemble training algorithms, the critical differences are $CD = 1.3736$ (where $j = 4$ and $T = 8$) and $CD = 1.0776$ (where $j = 4$ and $T = 13$) for regression and classification problems, respectively. Several observations can be made from our results.

First, for regression problems, the differences between RNCL versus $\text{NCL}_{\lambda=1}$, and RNCL versus NCL_{CV} are greater than the critical difference, so the differences are significant, which means that the RNCL is significantly better than $\text{NCL}_{\lambda=1}$ and NCL_{CV} in the current experimental settings. We could not detect any significant difference between $\text{Bagging}_{\text{reg}}$ and RNCL. The correct statistical statement would be that *the experimental data are not sufficient to reach any conclusion regarding the difference between RNCL and* $\text{Bagging}_{\text{reg}}$ for regression problems.

Second, for classification problems, RNCL significantly outperforms $\text{NCL}_{\lambda=1}$ and $\text{Bagging}_{\text{reg}}$. Since the difference between RNCL and NCL_{CV} is smaller than the critical dif-

TABLE VIII
RUNNING TIME OF RNCL, NCL_{CV}, $\text{NCL}_{\lambda=1}$, $\text{Bagging}_{\text{reg}}$, $\text{Bagging}_{\text{unreg}}$, AND $\text{RBF}_{\text{Bayes}}$ ON REGRESSION DATA SETS IN SECONDS. RESULTS ARE AVERAGED OVER 100 RUNS

Data Sets	Mexican Hat	Friedman	Gabor	Multi	Plane	Polynomial	Sinc	House
RBF_{Bayes}	0.6	0.3	0.3	0.4	0.8	0.9	0.6	2.3
Bagging_{unreg}	3.9	3.3	3.2	1.4	3.3	2.6	3.8	4.6
Bagging_{reg}	4.3	3.6	3.3	1.6	3.5	2.8	4.0	4.7
$\text{NCL}_{\lambda=1}$	3.4	4.7	5.9	5.0	2.9	4.4	3.3	10.3
NCL_{CV}	34.1	51.2	55.3	56.7	30.2	40.6	35.2	110.6
RNCL	21.3	36.2	30.8	38.2	10.6	17.4	20.1	70.6

TABLE IX
RUNNING TIME OF RNCL, NCL_{CV}, $\text{NCL}_{\lambda=1}$, $\text{Bagging}_{\text{reg}}$, $\text{Bagging}_{\text{unreg}}$, AND $\text{RBF}_{\text{Bayes}}$ ON CLASSIFICATION DATA SETS IN SECONDS. RESULTS ARE AVERAGED OVER 100 RUNS

	RNCL	NCL_{CV}	$\text{NCL}_{\lambda=1}$	Bagging_{reg}	Bagging_{unreg}	RBF_{Bayes}
Banana	12.4	39.2	3.4	2.6	2.2	0.3
Cancer	15.5	46.3	4.4	2.8	2.6	1.0
Diabetics	27.2	91.6	8.4	3.1	3.0	1.1
Solar	34.7	110.2	10.6	3.6	3.5	0.6
German	90.4	194.2	18.6	6.7	6.6	1.2
Heart	18.2	58.9	5.2	3.1	3.1	0.4
Image	90.5	300.7	29.1	6.9	6.2	1.5
Ringnorm	46.5	120.6	11.8	2.6	2.4	0.9
Splice	489.7	518.6	46.1	10.2	10.2	2.4
Thyroid	13.1	34.6	3.2	1.8	1.7	0.4
Titanic	12.4	31.4	2.6	1.7	1.6	0.4
Twonorm	60.2	136.4	8.3	2.3	2.3	0.6
Waveform	40.3	72.3	6.8	2.7	2.7	0.7

ference, we cannot draw any conclusion about the difference between RNCL and NCL_{CV} for classification problems in our experimental settings.

There are three major reasons why the RNCL performs better than others.

1) RNCL inherits the advantages of NCL and encourages the cooperation among ensemble members to solve one problem.
2) RNCL regularizes the complexity of the ensemble using an additional regularization term. The adequate regularization term controls the model complexity, and thus improves the model generalization.
3) RNCL incorporates an efficient parameter optimization procedure based on Bayesian inference. This procedure not only saves the effort to do cross validation but also improves the performance.

F. Computational Complexity and Running Time

Based on the algorithm in Fig. 1, RNCL consists of two main parts: neural network training using NCL and Bayesian parameter optimization.

In the first part, for each component neural network, in total M neural networks in the ensemble, one needs to train the network with an amount of epochs. Since the SCG 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 eigendecomposition of the Hessian matrix. The calculation of Hessian matrix will cost $O(NW^2)$, where N is the number of training points and W is the total number of weights in the ensemble. As discussed in Section III-C2, the computational cost for the eigendecomposition problem is $O(W^3)$. Therefore, the total computational cost is $O(NW^2 + W^3)$. For small sample problems when $N \approx W$, the cost to calculate the Hessian matrix is similar to the calculation of eigendecomposition. For moderate to large sample problems when $N \gg W$, the calculation of Hessian matrix costs more than the eigendecomposition. This calculation becomes computationally expensive for large ensembles, i.e., large W, with applications to large sample problems. In order to make the calculation efficient, a medium or small ensemble is suitable for large sample applications. To further reduce the computational complexity of the eigendecomposition, one can choose to calculate only the largest eigenvalues using an expectation–maximization approach [27].

RNCL is an iterative algorithm to update the regularization parameters, and in most of time it will converge in less than eight 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 Tables VIII and IX, we show the average running time of RNCL and other algorithms over 100 runs. The computational environment is windows XP with Intel Core 2 Duo 1.66G CPU and 2-GB RAM. These algorithms including RNCL, NCL_{CV}, $\text{NCL}_{\lambda=1}$, $\text{Bagging}_{\text{reg}}$, $\text{Bagging}_{\text{unreg}}$, and $\text{RBF}_{\text{Bayes}}$ are implemented in MATLAB and C language. The C language is used to implement RBF network training algorithm.

V. CONCLUSION

This paper analyzes NCL and points out that NCL is prone to overfitting the noise since NCL does not regularize its complexity. In the following, the paper analyzes this problem and proposes the RNCL which incorporates an additional regularization term into 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 a component neural network. RNCL inherits the advantages

of NCL and its 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 automatic procedure to optimize regularization parameters 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 behavior 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 nontrivial in data sets. Second, we carry out extensive experiments on eight benchmark regression and 13 benchmark classification data sets to compare the performance of RNCL, $\mathrm{NCL}_{\mathrm{CV}}$, $\mathrm{NCL}_{\lambda=1}$, $\mathrm{Bagging}_{\mathrm{reg}}$, $\mathrm{Bagging}_{\mathrm{unreg}}$, and $\mathrm{RBF}_{\mathrm{Bayes}}$. To compare classifiers on multiple data sets, the Friedman test with the corresponding post-hoc test has been used to statistically compare these classifiers over multiple data sets. This paper also analyzes the computational complexity of RNCL.

In order to validate the efficiency of Bayesian inference, in Section IV-D, we employ the cross validation to search the regularization term α. The results show that Bayesian inference can find appropriate regularization parameters according to different problems and are more efficient than cross validation.

Our results confirm that RNCL has shown an excellent performance on these data sets. For the $\mathrm{NCL}_{\mathrm{CV}}$, it appears that only tuning the correlation coefficient λ cannot overcome the overfitting problem. The differences between RNCL and $\mathrm{NCL}_{\mathrm{CV}}/\mathrm{NCL}_{\lambda=1}$ show that adopting regularization term and optimizing the regularization parameters by Bayesian inference are beneficial.

The RNCL algorithm is built on the NCL algorithm with the correlation parameter $\lambda = 1$. In this case, the summation of the negative correlation term and MSE term becomes the MSE of ensemble, so the negative correlation term disappears. We retain the name of RNCL here for consistency and historical reasons although negative correlation disappears when $\lambda = 1$. Our future work will consider including the correlation parameter λ within the Bayesian framework and optimizing λ and the regularization parameter α by Bayesian inference.

Although there is a lot of work in using MSE for classification problems [20], [38] and RNCL seems to work well for classification problems according to the empirical results, our theoretical analysis, including the Bayesian inference, is justified only for regression problems. More rigorous analysis needs to be done in the future for classification problems.

Chen *et al.* [39] 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 RNCL to generate more compact ensembles.

In general, we could conclude that the RNCL is an ensemble learning algorithm that addresses the substantial drawbacks of NCL. RNCL incorporates an efficient parameter optimization procedure, not only saving the effort to do cross validation but also improving the performance. The noise-robustness characteristic of RNCL is especially important when the training data are contaminated with noise.

Appendix I
Further Details of Gaussian Posterior

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

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

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

$$J_1(\mathbf{w}) \approx J_1(\mathbf{w}_{MP}) + \frac{1}{2}(\mathbf{w} - \mathbf{w}_{MP})^T A(\mathbf{w} - \mathbf{w}_{MP})$$

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

$$\begin{aligned} A = \nabla\nabla J_1 &= \nabla\nabla\left(\sum_{i=1}^{M}\frac{\mu_i}{2}\mathbf{w}_i^T\mathbf{w}_i + \frac{\beta}{2}\sum_{n=1}^{N}e_n^2\right) \\ &= \mathrm{diag}(\Lambda) + \beta\nabla\nabla\left(\frac{1}{2}\sum_{n=1}^{N}e_n^2\right) \end{aligned} \tag{17}$$

where

$$\Lambda = \left[\mu_1^{(1)}, \ldots \mu_1^{(n_1)}, \mu_2^{(1)}, \ldots \mu_2^{(n_2)}, \ldots, \mu_M^{(1)}, \ldots \mu_M^{(n_M)}\right]^T$$

and the superscript indicates the repetition number of μ_i.

We use two notations μ_i $(i = 1, \ldots, M)$ and Λ_j $(j = 1, \ldots, W)$ for μ_i, where M is the ensemble size, W is the number of weighs in ensemble, and

$$\Lambda = \left[\mu_1^{(1)}, \ldots \mu_1^{(n_1)}, \mu_2^{(1)}, \ldots \mu_2^{(n_2)}, \ldots, \mu_M^{(1)}, \ldots \mu_M^{(n_M)}\right]^T$$

wherein the superscript indicates the repetition number of μ_i. Because μ_i is the prior parameter for each network i, $\mu_i^{(n_i)}$ corresponds to each weight in network i. In Bayesian inference, we need to calculate the Hessian matrix of $J_1(\mathbf{w})$; the Hessian matrix A, whose dimension is $W \times W$, can be represented by $A = \mathrm{diag}(\Lambda) + \beta\nabla\nabla((1/2)\sum_{n=1}^{N} e_n^2)$. This is the reason why we use the second notation Λ for μ.

The integral can be computed as

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

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

$$\begin{aligned} P(\mathbf{w}|D) &= \frac{\exp\left(-J_1(\mathbf{w})\right)}{\int \exp\left(-J_1(\mathbf{w})\right) d\mathbf{w}} \\ &\approx \frac{\exp\left(-\frac{1}{2}(\mathbf{w}-\mathbf{w}_{MP})^T A(\mathbf{w}-\mathbf{w}_{MP})\right)}{(2\pi)^{W/2}\det A^{-\frac{1}{2}}}. \end{aligned}$$

APPENDIX II
DETAILS OF PARAMETER UPDATES

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

$$J_2 = \frac{1}{2}\sum_{i=1}^{M}\mu_i \mathbf{w}_{i,MP}^T\mathbf{w}_{i,MP} + \frac{1}{2}\beta\sum_{n=1}^{N}e_{n,MP}^2 - \frac{1}{2}\sum_{i=1}^{M}n_i\log\mu_i - \frac{1}{2}N\log\beta + \frac{1}{2}\log\det A. \quad (18)$$

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

Since $\det A = \prod_{j=1}^{W}(\beta\zeta_j+\Lambda_j)$, where ζ_j is the eigenvalue of the Hessian matrix $\nabla\nabla((1/2)\sum_{n=1}^{N}e_n^2)$ and W is the number of weighs in ensemble

$$\frac{\partial}{\partial\mu_i}\log\det A = \frac{\partial}{\partial\mu_i}\log\left(\prod_{j=1}^{W}(\beta\zeta_j+\Lambda_j)\right) = \sum_{j\in n_i}\frac{1}{\beta\zeta_j+\mu_i}$$

$$\frac{\partial}{\partial\beta}\log\det A = \frac{\partial}{\partial\beta}\log\left(\prod_{j=1}^{W}(\beta\zeta_j+\Lambda_j)\right) = \sum_{j}\frac{\zeta_j}{\beta\zeta_j+\Lambda_j}$$

where

$$\Lambda = \left[\mu_1^{(1)},\ldots\mu_1^{(n_1)},\mu_2^{(1)},\ldots\mu_2^{(n_2)},\ldots,\mu_M^{(1)},\ldots\mu_M^{(n_M)}\right]^T$$

the superscript indicates the repetition number of μ_i, and $j\in n_i$ indicates the range $[\sum_{t=1}^{i-1}n_t+1,\ldots,\sum_{t=1}^{i}n_t]$.

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

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

$$\frac{\partial J_2}{\partial\beta} = \frac{1}{2}\sum_{n=1}^{N}e_{n,MP}^2 - \frac{N}{2\beta} + \frac{1}{2}\sum_{j}\frac{\zeta_j}{\beta\zeta_j+\Lambda_j}.$$

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

$$\mu_i^{MP} = \frac{1}{\mathbf{w}_{i,MP}^T\mathbf{w}_{i,MP}}\left(n_i - \sum_{j\in n_i}\frac{\mu_i}{\beta\zeta_j+\mu_i}\right) \quad (19)$$

$$\beta^{MP} = \frac{1}{\sum_{n=1}^{N}e_{n,MP}^2}\left(N - \sum_{j=1}^{W}\frac{\beta\zeta_j}{\beta\zeta_j+\Lambda_j}\right). \quad (20)$$

After the most probable μ_i^{MP} and β^{MP}[12] have been obtained by (19) and (20), the most probable α_i^{MP} can be easily obtained by using the relation $\alpha_i^{MP} = \mu_i^{MP}/\beta^{MP}$.

[12]The equality can be viewed as the Bayesian estimate of the variance $(\beta^{MP})^{-1} = \sum_{n=1}^{N}e_{n,MP}^2/(N-\sum_{j=1}^{W}(\beta\zeta_j/\beta\zeta_j+\Lambda_j))$ of the noise. The term $\sum_{j=1}^{W}(\beta\zeta_j/\beta\zeta_j+\Lambda_j)$ is the effective number of parameters [19], [24], [26].

In the following, we give a formal approach by reformulating the optimization problem, i.e., J_2 in (18), in μ_i, and β into a scalar optimization problem in $\alpha_i = \mu_i/\beta$.

Based on (19) and (20), the following relations can be obtained:

$$\sum_{i=1}^{M}\mu_i^{MP}\mathbf{w}_{i,MP}^T\mathbf{w}_{i,MP} = \sum_{i=1}^{M}n_i - \sum_{i=1}^{M}\sum_{j\in n_i}\frac{\mu_i}{\beta\zeta_j+\mu_i} = W - \sum_{j=1}^{W}\frac{\Lambda_j}{\beta\zeta_j+\Lambda_j} \quad (21)$$

$$\beta^{MP}\sum_{n=1}^{N}e_{n,MP}^2 = N - \sum_{j=1}^{W}\frac{\beta\zeta_j}{\beta\zeta_j+\Lambda_j}. \quad (22)$$

Combining both (21) and (22) and the relation $\alpha_i^{MP} = \mu_i^{MP}/\beta^{MP}$, we obtain the following equations:

$$\beta^{MP}\left[\sum_{i=1}^{M}\alpha_i^{MP}\mathbf{w}_{i,MP}^T\mathbf{w}_{i,MP} + \sum_{n=1}^{N}e_{n,MP}^2\right] = N. \quad (23)$$

In the following, we reformulate the optimization problem (18) 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 (23) also holds in the scalar optimization, we search for the optimum only along this curve in the (α_i and β) space.

By elimination of β from (23), the minimization problem from J_2 is obtained in a straightforward way

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

where $\hat{\alpha}_j = \Lambda_j/\beta$ and

$$\Lambda = \left[\mu_1^{(1)},\ldots\mu_1^{(n_1)},\mu_2^{(1)},\ldots\mu_2^{(n_2)},\ldots,\mu_M^{(1)},\ldots\mu_M^{(n_M)}\right]^T.$$

Setting $\partial J_3/\partial\alpha_i = 0$, the update rule α_i^{MP} can be obtained as follows:

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

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

APPENDIX III
DETAILS OF RBF NETWORKS

The component network in the ensemble is an RBF network. The output of RBF network is computed as a linear combination of n_i basis functions

$$f_i(\mathbf{x}) = \sum_{k=1}^{n_i}w_k\phi_k(\mathbf{x}) = \Phi^T\mathbf{w}_i$$

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

$$\phi_k(\mathbf{x}) = \exp\left(\frac{\|\mathbf{x} - \rho_k\|^2}{2\sigma_k^2}\right)$$

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

$$\min e_i = \frac{1}{M}\sum_{n=1}^{N}(f_i(\mathbf{x}_n) - y_n)^2 - \frac{1}{M}\sum_{n=1}^{N}(f_i(\mathbf{x}_n) - f_{ens}(\mathbf{x}_n))^2 + \alpha_i \sum_{k=1}^{n_i} w_k^2. \quad (24)$$

The derivative of (24) with respect to w_k is

$$\frac{\partial e_i}{\partial w_k} = \frac{2}{M}\sum_{n=1}^{N}(f_i(\mathbf{x}_n) - y_n)\frac{\partial f_i(\mathbf{x}_n)}{\partial w_k} - \frac{2}{M}\sum_{n=1}^{N}(f_i(\mathbf{x}_n) - f_{ens}(\mathbf{x}_n))\left(1 - \frac{1}{M}\right)\frac{\partial f_i(\mathbf{x}_n)}{\partial w_k} + 2\alpha_i w_k. \quad (25)$$

In order to fine-tune the centers and widths, we simultaneously adjust the output weights and the RBF centers and variances. Taking the derivative of (24) with respect to RBF means ρ_k and variances σ_k^2, and we obtain

$$\frac{\partial e_i}{\partial \rho_k} = \frac{2}{M}\sum_{n=1}^{N}(f(\mathbf{x}_n) - y_n)\frac{\partial f_i(\mathbf{x}_n)}{\partial \rho_k} - \frac{2}{M}\sum_{n=1}^{N}(f_i(\mathbf{x}_n) - f_{ens}(\mathbf{x}_n))\left(1 - \frac{1}{M}\right)\frac{\partial f_i(\mathbf{x}_n)}{\partial \rho_k} \quad (26)$$

with $\partial f_i(\mathbf{x}_n)/\partial \rho_k = w_k((\mathbf{x}_n - \rho_k)/\sigma_k^2)\phi_k(\mathbf{x}_n)$ and

$$\frac{\partial e_i}{\partial \sigma_k} = \frac{2}{M}\sum_{n=1}^{N}(f(\mathbf{x}_n) - y_n)\frac{\partial f_i(\mathbf{x}_n)}{\partial \sigma_k} - \frac{2}{M}\sum_{n=1}^{N}(f_i(\mathbf{x}_n) - f_{ens}(\mathbf{x}_n))\left(1 - \frac{1}{M}\right)\frac{\partial f_i(\mathbf{x}_n)}{\partial \sigma_k} \quad (27)$$

with $\partial f_i(\mathbf{x}_n)/\partial \sigma_k = w_k(\|\mathbf{x} - \rho_k\|^2/\sigma_k^3)\phi_k(\mathbf{x}_n)$. These three derivatives are employed in the minimization of (24) by an SCG descent.

REFERENCES

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

[2] Y. Liu and X. Yao, "Simultaneous training of negatively correlated neural networks in an ensemble," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 29, no. 6, pp. 716–725, Dec. 1999.

[3] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 10, pp. 993–1001, Oct. 1990.

[4] F. J. Huang, T. Chen, Z. Zhou, and H. Zhang, "Pose invariant face recognition," in *Proc. 4th IEEE Int. Conf. Autom. Face Gesture Recognit.*, Washington, DC, 2000, pp. 245–250.

[5] L. K. Hansen, L. Liisberg, and P. Salamon, "Ensemble methods for handwritten digit recognition," in *Proc. IEEE Workshop Neural Netw. Signal Process.*, Helsingoer, Denmark, 1992, pp. 333–342.

[6] K. J. Cherkauer, "Human expert level performance on a scientific image analysis task by a system using combined artificial neural networks," in *Proc. AAAI Workshop Integrating Multiple Learned Models Improving Scaling Mach. Learn. Algorithms*, Menlo Park, CA, 1996, pp. 15–21.

[7] X. Yao, M. Fischer, and G. Brown, "Neural network ensembles and their application to traffic flow prediction in telecommunications networks," in *Proc. Int. Joint Conf. Neural Netw.*, 2001, pp. 693–698.

[8] Y. Liu, X. Yao, and T. Higuchi, "Evolutionary ensembles with negative correlation learning," *IEEE Trans. Evol. Comput.*, vol. 4, no. 4, pp. 380–387, Nov. 2000.

[9] G. Brown, J. Wyatt, and P. Tino, "Managing diversity in regression ensembles," *J. Mach. Learn. Res.*, vol. 6, pp. 1621–1650, 2005.

[10] L. Breiman, "Bagging predictors," *Mach. Learn.*, vol. 24, no. 2, pp. 123–140, 1996.

[11] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 8, pp. 832–844, Aug. 1998.

[12] R. E. Schapire, "A brief introduction to boosting," in *Proc. 16th Int. Joint Conf. Artif. Intell.*, 1999, pp. 1401–1406.

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

[14] H. Chen and X. Yao, "Evolutionary random neural ensemble based on negative correlation learning," in *Proc. IEEE Congr. Evol. Comput.*, 2007, pp. 1468–1474.

[15] H. Chen and X. Yao, "Multi-objective neural network ensembles based on regularized negative correlation learning," *IEEE Trans. Knowl. Data Eng.*, 2009, to be published.

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

[17] A. E. Hoerl and R. W. Kennard, "Ridge regression: Biased estimation for nonorthogonal problems," *Technometrics*, vol. 42, no. 1, pp. 80–86, 2000.

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

[19] D. J. C. MacKay, "Bayesian interpolation," *Neural Comput.*, vol. 4, no. 3, pp. 415–447, 1992.

[20] T. V. Gestel, J. A. K. Suykens, G. Lanckriet, A. Lambrechts, B. D. Moor, and J. Vandewalle, "Bayesian framework for least-squares support vector machine classifiers, Gaussian processes, and kernel fisher discriminant analysis," *Neural Comput.*, vol. 14, no. 5, pp. 1115–1147, 2002.

[21] M. E. Tipping, "Sparse Bayesian learning and the relevance vector machine," *J. Mach. Learn. Res.*, vol. 1, no. 3, pp. 211–244, 2001.

[22] H. Chen, P. Tiňo, and X. Yao, "Probabilistic classification vector machine," *IEEE Trans. Neural Netw.*, vol. 20, no. 6, pp. 901–914, Jun. 2009.

[23] J. Kwok, "Moderating the outputs of support vector machine classifiers," *IEEE Trans. Neural Netw.*, vol. 10, no. 5, pp. 1018–1031, Sep. 1999.

[24] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford, U.K.: Oxford Univ. Press, 1996.

[25] M. F. Møller, "A scaled conjugate gradient algorithm for fast supervised learning," *Neural Netw.*, vol. 6, no. 4, pp. 525–533, 1993.

[26] R. M. Neal, *Bayesian Learning for Neural Networks*. New York: Springer-Verlag, 1996, vol. 118.

[27] R. Rosipal and M. Girolami, "An expectation-maximization approach to nonlinear component analysis," *Neural Comput.*, vol. 13, no. 3, pp. 505–510, 2001.

[28] B. D. Ripley, *Pattern Recognition and Neural Networks*. Cambridge, U.K.: Cambridge Univ. Press, 1996.

[29] J. Weston, M. Stitson, A. Gammerman, V. Vovk, and V. Vapnik, "Experiments with support vector machines," Royal Holloway Univ. London, London, U.K., Tech. Rep. CSD-TR-96-19, 1996.

[30] J. Hansen, "Combining predictors: Meta machine learning methods and bias/variance and ambiguity decompositions," Ph.D. dissertation, Dept. Comput. Sci., Univ. Aarhus, Aarhus, Denmark, 2000.

[31] G. Ridgeway, D. Madigan, and T. Richardson, "Boosting methodology for regression problems," in *Proc. Artif. Intell. Statist.*, 1999, pp. 152–161.
[32] A. Asuncion and D. Newman, UCI Machine Learning Repository, Univ. California Irvine, Irvine, CA, 2007 [Online]. Available: http://www.archive.ics.uci.edu/ml
[33] P. Buhlmann and B. Yu, "Analyzing Bagging," *Ann. Statist.*, vol. 30, no. 4, pp. 927–961, 2002.
[34] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, 2006.
[35] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *J. Amer. Statist. Assoc.*, vol. 32, pp. 675–701, 1937.
[36] R. L. Iman and J. M. Davenport, "Approximations of the critical region of the Friedman statistic," *Commun. Statist.*, vol. 9, no. 6, pp. 571–595, 1980.
[37] O. J. Dunn, "Multiple comparisons among means," *J. Amer. Statist. Assoc.*, vol. 56, pp. 52–64, 1961.
[38] J. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Process. Lett.*, vol. 9, no. 3, pp. 293–300, 1999.
[39] H. Chen, P. Tiňo, and X. Yao, "Predictive ensemble pruning by expectation propagation," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 7, pp. 999–1013, Jul. 2009.

Huanhuan Chen (S'06–M'07) received the B.Sc. degree from the University of Science and Technology of China, Hefei, China, in 2004 and Ph.D. degree in computer science from the University of Birmingham, Birmingham, U.K., in 2008.

He is a Research Fellow with the Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA), 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 CPHC/British Computer Society (BCS) Distinguished Dissertations Award as the runner up (2009) for his Ph.D. dissertation "Diversity and regularization in neural network ensembles," 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, China, in 1982, the M.Sc. degree from the North China Institute of Computing Technology, Beijing, China, 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, Australia, 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, Australia, 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. He moved to the University of Birmingham, U.K., as a Professor (Chair) of Computer Science 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. 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.

Dr. Yao was the Editor-in-Chief of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION (2003–2008), an Associate Editor or editorial board member of 12 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. 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.