Predictive Ensemble Pruning by Expectation Propagation

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

Abstract—An ensemble is a group of learners that work together as a committee to solve a problem. The existing ensemble learning algorithms often generate unnecessarily large ensembles, which consume extra computational resource and may degrade the generalization performance. Ensemble pruning algorithms aim to find a good subset of ensemble members to constitute a small ensemble, which saves the computational resource and performs as well as, or better than, the unpruned ensemble. This paper introduces a probabilistic ensemble pruning algorithm by choosing a set of "sparse" combination weights, most of which are zeros, to prune the ensemble. In order to obtain the set of sparse combination weights and satisfy the non-negative constraint of the combination weights, a left-truncated, non-negative, Gaussian prior is adopted over every combination weight. Expectation propagation (EP) algorithm is employed to approximate the posterior estimation of the weight vector. The leave-one-out (LOO) error can be obtained as a byproduct in the training of EP without extra computation and is a good indication for the generalization error. Therefore, the LOO error is used together with the Bayesian evidence for model selection in this algorithm. An empirical study on several regression and classification benchmark data sets shows that our algorithm utilizes far less component learners but performs as well as, or better than, the unpruned ensemble. Our results are very competitive compared with other ensemble pruning algorithms.

Index Terms—Machine learning, Probabilistic algorithms.

1 Introduction

Ensemble of multiple learning machines, i.e. a group of learners that work together as a committee, has attracted a lot of research interests because it is a good approach to improve the generalization ability [1]. Because of their simplicity and effectiveness, ensembles have become a hot topic in machine learning. This technique originates from Hansen and Salamon's work [1], which shows that the generalization ability of a neural network can be significantly improved through ensembling a number of neural networks. Ensembles have already been successfully applied to many areas and there have been many ensemble learning algorithms in the literature, such as Bagging [2], Boosting [3], Arcing [4], random forests [5], rotation forests [6], COPEN (pairwise COnstraints Projection based ENsemble) [7], negative correlation learning and evolutionary computation based algorithms [8], [9].

However, the existing ensemble learning algorithms often generate unnecessarily large ensembles. These large ensembles are memory demanding. Obtaining a prediction for a *fresh* data point can be done expensively in large ensembles. Although these extra costs may seem to be negligible when dealing with small data sets, they may become serious when the ensemble method is applied to a large scale data set.

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

Manuscript received August 21, 2008; revised October 27, 2009; accepted February 13, 2009.

In addition, it is not always true that the larger the size of an ensemble, the better it is. Some theoretical and empirical evidences have shown that small ensembles can be better than large ensembles [2], [10], [11]. For example, the boosting ensembles, Adaboosting [3] and Arcing [4], pay more attention to those training samples that are misclassified by former learning machines in the training of next machines and finally reduce the training error to zero. In this way, Boosting ensembles are prone to overfitting the noise in the training set [12]. In these circumstances, it is necessary to prune some overfitting individuals to achieve good generalization.

In the last decades, several ensemble pruning algorithms have been proposed, such as Kappa pruning [13], concurrency pruning [14]. However, these algorithms all resort to greedy search, which is without either theoretical or empirical quality guarantees.

Yao et al. [10] first adopted a global optimization approach, genetic algorithm (GA), to weigh the ensemble members by constraining the weighs to be positive. Zhou et al. [11] later also proved that small ensembles can be better than large ensembles. A similar genetic algorithm approach can be found in [15]. However, these GA based algorithms try to obtain the optimal combination weights by minimizing the training error and in this way these algorithms become sensitive to noise.

Motivated by the above reasons, we modeled the ensemble pruning as a probabilistic model with truncated Gaussian prior for both regression and classification problems [16]. The Expectation-Maximization (EM) algorithm is used to infer the combination weights and our algorithm shows good performance in both generalization error and pruned ensemble size. However, the

EM algorithm is sensitive to the initialization and it does not guarantee to obtain the optimal solutions due to converging to local maxima.

In order to resolve the unstable issue of EM, this paper extends our previous work and proposes a probabilistic ensemble pruning algorithm based on expectation propagation (EP). The algorithm treats ensemble pruning as a weight-based optimization, aiming to improve the generalization performance of the ensemble by tuning the weight of each ensemble member. By introducing a sparseness-inducing prior for each combination weight, many of the posteriors of weights are sharply distributed around zero, leading to pruning unimportant learning machines. As negative combination weights are unreliable and not intuitive [17]–[19], we follow this constraint and employ a left-truncated prior to prevent negative values in the combination weights.

By incorporating the truncated prior, the normalization integral in Bayesian inference becomes intractable. This paper uses expectation propagation to approximate the posterior of weights. EP [20] is a deterministic algorithm for approximating Bayesian inference that extends assumed-density filtering (ADF) to incorporate iterative refinement of the approximations. As the leave-one-out (LOO) error can be obtained as a byproduct in the training of EP without extra computation, the LOO error is used together with Bayesian evidence for model selection.

An empirical study on several regression and classification benchmark data sets shows that our algorithm utilizes far less component learners yet performs as well as, or better than, the unpruned ensemble. The results are very competitive compared with other existing ensemble pruning algorithms.

The original contribution of this paper over the previous paper [16] is the use of Expectation Propagation instead of Expectation-Maximization to overcome stability problem of EM (sensitivity to initialization and convergence to local maxima), and a comprehensive empirical study. The benefit of obtaining a LOO estimate in the training of EP is an additional advantage of this approach.

Ensemble pruning algorithms can improve accuracy significantly as shown by the theoretical and empirical studies in this paper. A smaller ensemble will also reduce test (application) time after the ensemble is trained and pruned. However, such benefits come with the cost of a longer training time. There is a trade-off between training and application times when considering ensemble pruning. When an application has a very large training set, ensemble pruning can be computationally expensive, or even prohibitive. For applications that are characterized by relatively small training sets but large test ones, e.g., in the field of transductive learning, ensemble pruning algorithms are highly appropriate as smaller ensembles pruned by an ensemble pruning algorithm can improve the accuracy and reduce the application time in the test stage. In this paper, we provide our approach on a poker hand problem characterized by a relatively small training set and a large test set.

The rest of this paper is organized as follows. After the background introduction in Section 2, Section 3 presents the EP pruning algorithm for regression and classification problems, respectively, followed by experimental results and analysis in Section 4. Finally, Section 5 concludes the paper and presents some future work.

There are many symbols used in this paper. Figure 1 reports the meaning of these notations. We use lowercase letters for vectors and uppercase for matrices. Scalar quantities will be typed in normal.

$f_i(\cdot)$ – the i-th individual learner in an ensemble
$f_{ens}(\cdot)$ – the function of an ensemble
w_i – the combination weight of the learner $f_i(\cdot)$.
$\mathbf{w}$ – the weight vector of an ensemble $\mathbf{w} = (w_1, \cdots, w_M)^T$.
$\mathbf{F}(\mathbf{x})$ – the vector of individual learners $\mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), \cdots, f_M(\mathbf{x}))^T$.
$\mathbf{x}$ – input vector from a d-dimensional space, usually $\mathbb{R}^d$.
y – the output corresponding to a given input $\mathbf{x}$.
$\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$ – the training data set.
$p(\mathbf{x})$ – the distribution of $\mathbf{x}$
$\mathbf{C}$ – correlation matrix with elements indexed as $C_{ij} = \int p(\mathbf{x})(f_i(\mathbf{x}) - y)(f_j(\mathbf{x}) - y)d\mathbf{x}$
N – the number of samples.
M – the number of individual models in the ensemble, i.e. the size of ensemble.
α_i – the inverse variance of weight w_i
$\alpha = (\alpha_1, \cdots, \alpha_M)^T$ – the inverse variance of weight vector $\mathbf{w}$.
$N_t(w_i|0, \alpha_i^{-1})$ – a left-truncated Gaussian distribution with mean 0 and variance α_i^{-1}.
$N(w_i|0, \alpha_i^{-1})$ – a Gaussian distribution with mean 0 and variance α_i^{-1}.
ϵ_n – Gaussian noise $\epsilon_n = N(0, \sigma^2)$ with mean zero and variance σ^2.
$\Theta(w_i)$ – the step function $\Theta(w_i) = \begin{cases} 1 & \text{if } w_i > 0 \\ 0 & \text{if } w_i \leq 0 \end{cases}$.
$t_i(\mathbf{w})$ – the exact term $t_i(\mathbf{w}) = \Theta(w_i)$ in EP.
$\tilde{t}_i(\mathbf{w})$ – the term to approximate the step function $t_i(\mathbf{w}) = \Theta(w_i)$ in EP.
g_n – the exact likelihood term $g_n = p(y_n|\mathbf{x}_n, \mathbf{w})$ in EP.
$\tilde{g}_n$ – the term to approximate the likelihood term $g_n = p(y_n|\mathbf{x}_n, \mathbf{w})$.
m_i – the mean of the approximating term $\tilde{t}_i(\mathbf{w})$.
v_i – the variance of the approximating term $\tilde{t}_i(\mathbf{w})$.
s_i – the normalization factor of the approximating term $\tilde{t}_i(\mathbf{w})$.
$q(\mathbf{w})$ – the approximated posterior of $\mathbf{w}$.
$q^{\setminus n}(\mathbf{w})$ – the approximated *leave-one-out* posterior, i.e. remove the approximation term $\tilde{g}_n$ from the posterior $q(\mathbf{w})$.
$\Phi(x) = \int_{-\infty}^{x} N(t|0, 1)dt$ – Gaussian cumulative distribution function.

Fig. 1. Notations used in this paper.

2 BACKGROUND

The goal of ensemble pruning is to reduce the size of ensemble without compromising its performance. The pruning algorithms can be classified into two categories, selection-based and weight-based pruning algorithms. In the following, we review the two kinds of strategies, respectively.

2.1 Selection based Ensemble Pruning

The selection-based ensemble pruning algorithms do not weigh each leaner by a weighting coefficient, and they either select or reject the learner.

A straightforward method is to rank the learners according to their individual performance on a validation set and pick the best ones [21]. This simple approach may sometimes work well but is theoretically unsound

since this strategy is greedy and not guaranteed to be globe optimum.

Margineantu et al. [13] proposed four heuristic approaches to prune ensembles generated by Adaboost. Of them, KL-divergence pruning [13] and Kappa pruning [13] aim at maximizing the pair-wise difference between the selected ensemble members. Kappa-error convex hull pruning [13] is a diagram-based heuristic targeting at a good accuracy-divergence trade-off among the selected subsets. Back-fitting pruning [13] essentially enumerates all the possible subsets, which is computationally too costly for large ensembles. Then, Prodromidis et al. invented several pruning algorithms for their distributed data mining system [22]. One of the two algorithms they implemented is based on a diversity measure they defined, and the other is based on the class speciality metrics. The major problem with the above algorithms is that they all resort to greedy search, which is usually without either theoretical or empirical quality guarantees.

Zhang et al. [23] formulated ensemble pruning as a quadratic integer programming problem. By applying semi-definite programming (SDP) as a solution technique, their SDP-based pruning algorithm outperformed other heuristics. However, the algorithm did not achieve better performance than the unpruned ensemble due to the following limitations: 1) some parameters need to be specified in the algorithm and 2) a better objective function is needed for more accurate solutions.

2.2 Weight based Ensemble Pruning

The more general weight-based ensemble optimization aims at improving the generalization performance of the ensemble by tuning the weight on each ensemble member.

For regression ensembles, the optimal combination weights can be calculated analytically [11], [18]. The study has been covered in other research areas as well, such as operational research [24]. According to [11], the optimal weights can be obtained as:

$$w_i = \frac{\sum_{j=1}^{M}(\mathbf{C}^{-1})_{ij}}{\sum_{k=1}^{M}\sum_{j=1}^{M}(\mathbf{C}^{-1})_{kj}},$$

where $\mathbf{C}$ is the correlation matrix with elements indexed as $C_{ij} = \int p(\mathbf{x})(f_i(\mathbf{x}) - y)(f_j(\mathbf{x}) - y)dx$ that is the correlation between the i^{th} and the j^{th} component learner, wherein $p(\mathbf{x})$ is the distribution of $\mathbf{x}$. The correlation matrix $\mathbf{C}$ cannot be computed analytically without knowing the distribution $p(\mathbf{x})$ but can be approximated with a training set, as follows:

$$C_{ij} \approx \frac{1}{N}\sum_{n=1}^{N}\left[(y_n - f_i(\mathbf{x}_n))(y_n - f_j(\mathbf{x}_n))\right].$$

However, this approach rarely works well in real-world applications. This is because when a number of estimators are available, there are often some estimators that are quite similar in performance, which makes the correlation matrix $\mathbf{C}$ ill-conditioned, hampering the least square estimation. Other issues of this formulation include (1) the optimal combination weights are computed from the training set, which often overfits the noise and (2) in most cases the optimal solution does not reduce the ensemble size.

In this paper, a numerically stable algorithm, which is applicable to rank deficient cases (*lscov* in MATLAB), is used to calculate the least square solution for ensemble pruning. The algorithm, called least square (LS) pruning in our experiments, acts as a baseline. The LS pruning is applicable to binary classification problems by modeling the classification problem as a regression problem with its target set to -1 or +1. However, the LS pruning often produces negative combination weights. Strategies allowing negative combination weights were shown to be unreliable [25], [26].

Demiriz et al. [27] employed mathematical programming to look for good weighting schemes. Those optimization approaches are effective in performance enhancement according to empirical results and are sometimes able to significantly reduce the ensemble size [27]. However, ensemble size reduction is not explicitly built into those programs and the final size of the ensemble can still be very large in some cases.

In fact, the weight based ensemble pruning can be viewed as a sparse Bayesian learning problem by applying Tipping's relevance vector machine (RVM) [28]. RVM is an application of Bayesian automatic relevance determination (ARD) and it prunes most of the ensemble members by employing a Gaussian prior and updating the hyperparameters in an iterative way. However, *ARD pruning* allows negative combination weights and the solution is not optimal according to the current research [25], [26].

To address the problem of *ARD pruning*, Chen et al. [16] modeled the ensemble pruning as a probabilistic model with truncated Gaussian prior for both regression and classification problems. The Expectation-Maximization (EM) algorithm is used to infer the combination weights and our algorithm shows good performance in both generalization error and pruned ensemble size. However, EM algorithm is sensitive to the initialization and it does not guarantee to obtain the optimal solutions due to converging to local maxima.

The paper extends the previous work and employs the deterministic expectation propagation (EP) [20] to approximate the posterior of weights. Conveniently, an estimate of the leave-one-out (LOO) error can be obtained in the training of EP. The LOO error is used together with the Bayesian evidence for model selection in this algorithm.

3 Ensemble Pruning Algorithms by Expectation Propagation

In this section, we describe our ensemble pruning algorithm and present the detailed expectation propagation

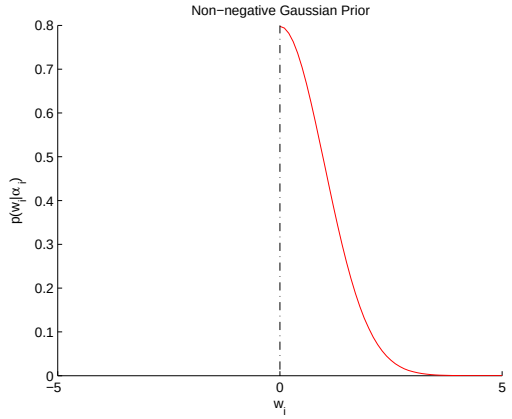


Fig. 2. The left-truncated Gaussian Prior

procedures.

3.1 Sparseness-introduction and Truncated Prior

In the weight-based ensemble pruning algorithm, the ensemble is formulated as a linear combination of the individual learners:

$$f_{ens}(\mathbf{x};\mathbf{w}) = \sum_{i=1}^{M} w_i f_i(\mathbf{x}) = \mathbf{w}^T\mathbf{F}(\mathbf{x}),$$

where $\mathbf{w} = (w_1, \cdots, w_M)^T$ is the weight vector of the ensemble, and $\mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), \cdots, f_M(\mathbf{x}))^T$ is the vector of individual learners. The pruning algorithm is to adjust the parameters $\mathbf{w}$, setting many w_i to zeros, but not degrade the generalization performance of the ensemble. As negative weight vectors are neither intuitive nor reliable [17]–[19], this paper constraints the weight vector to be non-negative.

To encourage the sparsity of weight vector $\mathbf{w}$ and to satisfy the non-negative restriction, a left-truncated Gaussian prior is introduced for each weight w_i:

$$p(\mathbf{w}|\alpha) = \prod_{i=1}^{M} p(w_i|\alpha_i) = \prod_{i=1}^{M} N_t(w_i|0,\alpha_i^{-1}), \qquad (1)$$

where $\alpha = (\alpha_1, \cdots, \alpha_M)^T$ is the inverse variance of weight $\mathbf{w}$ and $N_t(w_i|0,\alpha_i^{-1})$ is a left-truncated Gaussian distribution. This is formalized in equation (2) and illustrated in Figure 2.

$$p(w_i|\alpha_i) = \begin{cases} 2N(w_i|0,\alpha_i^{-1}) & \text{if } w_i \geq 0 \\ 0 & \text{if } w_i < 0 \end{cases}. \qquad (2)$$

3.2 Expectation Propagation

Expectation propagation (EP) [20] is a deterministic algorithm for approximating Bayesian inference that extends assumed-density filtering (ADF) to incorporate iterative refinement of the approximations. Expectation propagation assumes that the joint distribution $p(D,\mathbf{w})$, where the data $D = \{\mathbf{x}_n, y_n\}_{n=1}^N$ has been observed and $\mathbf{w}$ is a parameter vector, can be factored into some simple terms:

$$p(D,\mathbf{w}) = \prod_{i=1}^{M} p(w_i) \prod_{n=1}^{N} p(y_n|\mathbf{w},x_n) = \prod_{i=1}^{M} t_i \prod_{n=1}^{N} g_n,$$

where $p(w_i)$ is the prior distribution of w_i, M is the number of weights and $\prod_{n=1}^{N} p(y_n|\mathbf{w},x_n)$ is the likelihood. The intuitive meaning of the equation is that the prior $p(\mathbf{w})$ and the likelihood $\prod_{n=1}^{N} p(y_n|\mathbf{w},x_n)$ can be factored into a number of terms t_i and g_n, which can then be approximated by corresponding $\tilde{t}_i$ and $\tilde{g}_n$ in EP.

EP adopts a family of exponential functions ($\tilde{t}_i$, $\tilde{g}_n$) to approximate each term (t_i, g_n) by minimizing the KL-divergence between the exact term and the approximation term, and then combines these approximations analytically to obtain a Gaussian posterior $q(\mathbf{w})$ on $\mathbf{w}$.

3.3 Expectation Propagation for Regression Ensembles

In the regression ensemble model, we train M individual estimators using the training set $\{\mathbf{x}_n, y_n\}_{n=1}^N$, where y_n is a scalar. We assume the ensemble output is corrupted by an i.i.d. additive Gaussian noise $\epsilon_n \sim N(0,\sigma^2)$ with mean zero and variance σ^2:

$$y_n = \mathbf{w}^T\mathbf{F}(\mathbf{x}_n) + \epsilon_n. \qquad (3)$$

According to equation (3), the true value y_n is distributed as a Gaussian distribution with mean $\mathbf{w}^T\mathbf{F}(\mathbf{x}_n)$ and variance σ^2. Based on the assumption of independence of training points, the likelihood can be expressed as:

$$p(\mathbf{y}|\mathbf{w},\mathbf{x}_n,\sigma^2) = (2\pi\sigma^2)^{-N/2}\exp\{-\frac{1}{2\sigma^2}\|\mathbf{y}^T - \mathbf{w}^T\mathbf{F}\|^2\}. \qquad (4)$$

where $\mathbf{y} = (y_1, \cdots, y_N)^T$, $\mathbf{w} = (w_1, \cdots, w_M)^T$ and $\mathbf{F} = (\mathbf{F}(\mathbf{x_1}), \cdots, \mathbf{F}(\mathbf{x_N}))$ is a $M \times N$ matrix, where $\mathbf{F}(\mathbf{x_n}) = (f_1(\mathbf{x_n}), \cdots, f_M(\mathbf{x_n}))^T$.

The posterior of the weight vector $\mathbf{w}$ is denoted by

$$p(\mathbf{w} \mid \mathbf{x},\mathbf{y},\alpha) \propto \prod_{i=1}^{M} p(w_i|\alpha_i) \prod_{n=1}^{N} p(y_n|\mathbf{x}_n,\mathbf{w}). \qquad (5)$$

According to equation (2), the prior $p(w_i|\alpha_i)$ can be written as:

$$p(w_i|\alpha_i) = 2N(w_i|0,\alpha_i^{-1})\Theta(w_i), \qquad (6)$$

where $\Theta(w_i) = \Theta(\mathbf{w}^T\mathbf{e}_i) = \begin{cases} 1 & \text{if } w_i > 0 \\ 0 & \text{if } w_i \leq 0 \end{cases}$ prevents the weights from negative values and $\mathbf{e}_i = (0, \cdots, 1, 0, \cdots, 0)^T$ is used to obtain the weight w_i ($w_i = \mathbf{w}^T\mathbf{e}_i$ and $\Theta(w_i) = \Theta(\mathbf{w}^T\mathbf{e}_i)$).

Based on equation (6), equation (5) can be written as

$$p(\mathbf{w} \mid \mathbf{x},y,\alpha) \propto \prod_{i=1}^{M} 2N(w_i|0,\alpha_i^{-1}) \prod_{i=1}^{M} \Theta(w_i) \prod_{n=1}^{N} p(y_n|\mathbf{x}_n,\mathbf{w}). \qquad (7)$$

Input: the training set $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, the ensemble function $\mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), \cdots, f_M(\mathbf{x}))^T$, the number of sample N, the size of ensemble M and the inverse variance of weight vector $\alpha = (\alpha_1, \cdots, \alpha_M)^T$.

1. Initialize the prior term: $q(\mathbf{w}) = N(\mathbf{w}|0, \alpha^{-1})$ and the approximating terms to 1, $\tilde{t}_i = 1$: $m_i = 0$, $v_i = \infty$ and $s_i = 1$.
2. Until both $\tilde{g}_n$ and $\tilde{t}_i$ converge: Loop $n = 1, \ldots, N$, and $i = 1, \ldots, M$;
   (a) Remove the approximation term $\tilde{g}_n$ from the posterior $q(\mathbf{w})$ to obtain the *leave-one-out* posterior $q^{\setminus n}(\mathbf{w})$: $N(m_{\mathbf{w}}^{\setminus n}, V_{\mathbf{w}}^{\setminus n})$. Since $q^{\setminus n}(\mathbf{w}) \propto q(\mathbf{w})/\tilde{g}_n$:
   $$\begin{aligned} V_{\mathbf{w}}^{\setminus n} &= V_{\mathbf{w}} + \frac{(V_{\mathbf{w}} F_n)(V_{\mathbf{w}} F_n)^T}{v_n - F_n^T V_{\mathbf{w}} F_n}, \\ m_{\mathbf{w}}^{\setminus n} &= m_{\mathbf{w}} + (V_{\mathbf{w}}^{\setminus n} F_n) v_n^{-1} (F_n^T m_{\mathbf{w}} - m_n). \end{aligned}$$
   (b) Combine $q^{\setminus n}(\mathbf{w})$ and the exact term $g_n(\mathbf{w})$ to get the new posterior $q(\mathbf{w})$.
   $$V_{\mathbf{w}} = (\sigma^2 F_n^T F_n + V_{\mathbf{w}}^{\setminus n})^{-1}, \quad m_{\mathbf{w}} = V_{\mathbf{w}}(\sigma^2 F_n^T y_n + V_{\mathbf{w}}^{\setminus n} m_{\mathbf{w}}^{\setminus n}).$$
   (c) Update the approximation term $\tilde{g}_n = Z_n \frac{q(\mathbf{w})}{q^{\setminus n}(\mathbf{w})}$:
   $$v_n = \sigma^2, \quad m_n = y_n.$$
   (d) Remove the approximation term $\tilde{t}_i$ from the posterior $q(\mathbf{w})$ to obtain the leave-one-out posterior $q^{\setminus i}(\mathbf{w})$: $N(m_{\mathbf{w}}^{\setminus i}, V_{\mathbf{w}}^{\setminus i})$. Refer to the equations in step (a).
   (e) Combine $q^{\setminus i}(\mathbf{w})$ and the exact term $t_i(\mathbf{w})$ to get $\hat{p}(\mathbf{w}) \propto q^{\setminus i}(\mathbf{w}) t_i(\mathbf{w})$ and minimize the KL-divergence between $\hat{p}(\mathbf{w})$ and new posterior $q(\mathbf{w})$.
   $$\begin{aligned} m_{\mathbf{w}} &= m_{\mathbf{w}}^{\setminus i} + V_{\mathbf{w}}^{\setminus i} e_i \rho_i, \quad V_{\mathbf{w}} = V_{\mathbf{w}}^{\setminus i} + (V_{\mathbf{w}}^{\setminus i} e_i)(\frac{\alpha_i e_i^T m_{\mathbf{w}}}{e_i^T V_{\mathbf{w}}^{\setminus i} e_i})(V_{\mathbf{w}}^{\setminus i} e_i)^T, \\ Z_i &= \int_{\mathbf{w}} q^{\setminus i}(\mathbf{w}) g_i(\mathbf{w}) = \Phi(z_i), \end{aligned}$$
   where
   $$z_i = \frac{(m_{\mathbf{w}}^{\setminus i})^T e_i}{\sqrt{e_i^T V_{\mathbf{w}}^{\setminus i} e_i}}, \qquad \alpha_i = \frac{1}{\sqrt{e_i^T V_{\mathbf{w}}^{\setminus i} e_i}} \frac{N(z_i; 0, 1)}{\Phi(z_i)}.$$
   (f) Update the approximation term $\tilde{t}_i = Z_i \frac{q(\mathbf{w})}{q^{\setminus i}(\mathbf{w})}$:
   $$\begin{aligned} v_i &= e_i^T V_{\mathbf{w}}^{\setminus i} e_i (\frac{1}{\alpha_i e_i^T m_{\mathbf{w}}} - 1), \quad m_i = (m_{\mathbf{w}}^{\setminus i})^T e_i + (v_i + e_i^T V_{\mathbf{w}}^{\setminus i} e_i)\alpha_i, \\ s_i &= \Phi(z_i) \sqrt{e_i^T V_{\mathbf{w}}^{\setminus i} e_i v_i^{-1} + 1} \exp(\frac{1}{2} \frac{e_i^T V_{\mathbf{w}}^{\setminus i} e_i}{e_i^T m_{\mathbf{w}}} \alpha_i). \end{aligned}$$

Output: The approximated posterior of the weight vector $\mathbf{w}$
$$p(\mathbf{w}|\mathbf{x}, y, \alpha) \approx q(\mathbf{w}) = N(m_{\mathbf{w}}, V_{\mathbf{w}}).$$

Fig. 3. Expectation Propagation for Regression Ensembles

In equation (7), the likelihood $p(y_n|\mathbf{x}_n, \mathbf{w})$ and the terms $N(w_i|0, \alpha_i^{-1})$ are both Gaussians, i.e. $p(y_n|\mathbf{x}_n, \mathbf{w}) = (2\pi\sigma^2)^{-1/2} \exp(-\frac{1}{2\sigma^2}(\mathbf{w}^T \mathbf{F}(\mathbf{x}_n) - y_n)^2)$. Since EP approximates each non-Gaussian term by a Gaussian, only the terms $\Theta(w_i)$ are non-Gaussions in Equation (7). In this case, we only need to approximate the terms $\Theta(w_i)$ in calculating the posterior. Denote the exact terms $\Theta(w_i)$ by $t_i(\mathbf{w})$, and the approximate terms by $\tilde{t}_i(\mathbf{w}) = s_i \exp(-\frac{1}{2v_i}(\mathbf{w}^T \mathbf{e}_i - m_i)^2)$ which are parameterized by (m_i, v_i, s_i). Since the likelihood terms $p(y_n|\mathbf{x}_n, \mathbf{w})$ are Gaussians, we represent these terms $p(y_n|\mathbf{x}_n, \mathbf{w})$ by $\tilde{g}_n(\mathbf{w}) = s_n \exp(-\frac{1}{2v_n}(\mathbf{w}^T \mathbf{F}(\mathbf{x}_n) - y_n)^2)$ to facilitate EP training, where $v_n = \sigma^2$ and $m_n = y_n$. After approximating every term as an exponential family distribution, the resulting distribution will be Gaussian: $p(\mathbf{w}|\mathbf{x}, \mathbf{y}, \alpha) \approx q(\mathbf{w}) = N(\mathbf{m}_{\mathbf{w}}, \mathbf{V}_{\mathbf{w}})$. The EP algorithm for regression ensembles is described Figure 3 (to simplify notations, $\mathbf{F}_n$ stands for $\mathbf{F}(\mathbf{x}_n)$).

Input: the training set $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$, the ensemble function $\mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), \cdots, f_M(\mathbf{x}))^T$, the number of sample N, the size of ensemble M and the inverse variance of weight vector $\alpha = (\alpha_1, \cdots, \alpha_M)^T$.

1. Initialization the prior term: $q(\mathbf{w}) = p(\mathbf{w}|\alpha)$. Also initialize the approximating terms to 1: $\tilde{g}_n = 1$ and $\tilde{t}_i = 1$: $m = 0$, $v = \infty$ and $s = 1$.
2. Until both $\tilde{g}_n$ and $\tilde{t}_i$ converge: Loop $n = 1, \ldots, N$, and $i = 1, \ldots, M$;
   (a) Remove the approximation term $\tilde{g}_n$ from the posterior $q(\mathbf{w})$ to obtain the leave-one-out posterior $q^{\setminus n}(\mathbf{w})$: $N(m_{\mathbf{w}}^{\setminus n}, V_{\mathbf{w}}^{\setminus n})$. The Equation is exactly the same as the regression EP. (Please refer to the equations in step (a) in Figure 2)
   (b) Combine $q^{\setminus n}(\mathbf{w})$ and the exact term $g_n(\mathbf{w})$ to get $\hat{p}(\mathbf{w}) \propto q^{\setminus n}(\mathbf{w}) g_n(\mathbf{w})$ and minimize the KL-divergence between $\hat{p}(\mathbf{w})$ and new posterior $q(\mathbf{w})$.
   $$\begin{aligned} m_{\mathbf{w}} &= m_{\mathbf{w}}^{\setminus n} + V_{\mathbf{w}}^{\setminus n} F_n \rho_n, \quad V_{\mathbf{w}} = V_{\mathbf{w}}^{\setminus n} + (V_{\mathbf{w}}^{\setminus n} F_n) \frac{\rho_n (F_n^T m_{\mathbf{w}} + \rho_n)}{F_n^T V_{\mathbf{w}}^{\setminus n} F_n + 1} (V_{\mathbf{w}}^{\setminus n} F_n)^T, \\ Z_i &= \int_{\mathbf{w}} q^{\setminus n}(\mathbf{w}) g_n(\mathbf{w}) = \Phi(z_n), \end{aligned}$$
   where
   $$z_n = \frac{(m_{\mathbf{w}}^{\setminus n})^T F_n}{\sqrt{F_n^T V_{\mathbf{w}}^{\setminus n} F_n + 1}}, \quad \rho_n = \frac{1}{\sqrt{F_n^T V_{\mathbf{w}}^{\setminus n} F_n + 1}} \frac{N(z_n; 0, 1)}{\Phi(z_n)}.$$
   (c) Update the approximation term $\tilde{g}_n = Z_n \frac{q(\mathbf{w})}{q^{\setminus n}(\mathbf{w})}$:
   $$\begin{aligned} v_n &= F_n^T V_{\mathbf{w}}^{\setminus n} F_n (\frac{1}{\rho_n (F_n^T m_{\mathbf{w}} + \rho_n)} - 1) + \frac{1}{\rho_n (F_n^T m_{\mathbf{w}} + \rho_n)}, \\ m_n &= (m_{\mathbf{w}}^{\setminus n})^T F_n + (v_n + F_n^T V_{\mathbf{w}}^{\setminus n} F_n)\rho_n, \\ s_n &= \Phi(z_n) \sqrt{F_n^T V_{\mathbf{w}}^{\setminus n} F_n v_n^{-1} + 1} \exp(\frac{1}{2} \frac{F_n^T V_{\mathbf{w}}^{\setminus n} F_n + 1}{F_n^T m_{\mathbf{w}} + \rho_n} \rho_n). \end{aligned}$$
   (d) The remaining steps are the same as the regression ensemble. Please refer to EP pruning for regression ensemble steps 2(d)-(f) in Figure 3.

Output: The approximated posterior of the weight vector $\mathbf{w}$
$$p(\mathbf{w}|\mathbf{x}, y, \alpha) \approx q(\mathbf{w}) = N(m_{\mathbf{w}}, V_{\mathbf{w}}).$$

Fig. 4. Expectation Propagation for Classification Ensembles

Leave-one-out Estimation: A nice property of EP is that it can easily obtain an estimate of the leave-one-out error without any extra computation. In each iteration, EP computes the parameters of the approximate leave-one-out posterior $q^{\setminus n}(\mathbf{w})$ (step 2(a) in Figure 3) that does not depend on the n^{th} data point. So we can use the mean $m_{\mathbf{w}}^{\setminus n}$ to approximate an estimator trained on the other $(N-1)$ data points, thus an estimate of the leave-one-out MSE error can be computed as

$$err_{loo} = \frac{1}{N} \sum_{n=1}^{N} ((m_{\mathbf{w}}^{\setminus n})^T \mathbf{F}(\mathbf{x}_n) - y_n)^2. \tag{8}$$

3.4 Expectation Propagation for Classifier Ensembles

For classifier ensembles, the ensemble output is a linear combination of individual classifiers passed through a link function

$$f_{ens}(\mathbf{x}_n) = \Phi\left(\sum_{i=1}^{M} w_i f_i(\mathbf{x}_n)\right),$$

where $\Phi(x) = \int_{-\infty}^{x} N(t|0, 1)dt$ is the Gaussian cumulative distribution function.

Given the data set $D = \{\mathbf{x}_n, y\}_{n=1}^N$, the likelihood for

the combination weight $\mathbf{w}$ can be written as

$$p(\mathbf{y} \mid \mathbf{x}, \mathbf{w}) = \prod_{n=1}^{N} p(y_n|\mathbf{x}_n, \mathbf{w}) = \prod_{n=1}^{N} \Phi\left(y_n \sum_{i=1}^{M} w_i f_i(\mathbf{x}_n)\right).$$

By incorporating the prior with likelihood, the posterior of weight vectors $\mathbf{w}$ is denoted by

$$\begin{aligned} p(\mathbf{w}|\mathbf{x},y,\alpha) &\propto p(\mathbf{w}|\alpha) \prod_{n=1}^{N} p(y_n|\mathbf{x}_n, \mathbf{w}) \\ &= \prod_{i=1}^{M} 2N(w_i|0, \alpha_i^{-1})\Theta(w_i) \prod_{n=1}^{N} p(y_n|\mathbf{x}_n, \mathbf{w}). \end{aligned}$$

In EP algorithm, we need to approximate both the likelihood term $p(y_n|\mathbf{x}_n, \mathbf{w}) = \Phi\left(y_n \sum_{i=1}^{M} w_i f_i(\mathbf{x}_n)\right)$ and the $\Theta(w_i)$ term. Denote the exact terms $g_n(\mathbf{w}) = p(y_n|\mathbf{x}_n, \mathbf{w})$ and $t_i(\mathbf{w}) = \Theta(w_i) = \Theta(\mathbf{w}^T\mathbf{e}_i)$, and the approximate terms by $\tilde{g}_n(\mathbf{w}) = s_n \exp(-\frac{1}{2v_n}(y_n\mathbf{w}^T F_n - m_n)^2)$ and $\tilde{t}_i(\mathbf{w}) = s_i \exp(-\frac{1}{2v_i}(\mathbf{w}^T e_i - m_i)^2)$. The EP algorithm for classification ensembles is described in Figure 4 (to simplify notations, $y_n\mathbf{F}(\mathbf{x}_n)$ is written as $\mathbf{F}_n$).

An estimate of the leave-one-out error can be obtained by

$$err_{loo} = \frac{1}{N} \sum_{n=1}^{N} \Theta(-y_n(m_{\mathbf{w}}^{\backslash n})^T \mathbf{F}(\mathbf{x}_n)), \tag{9}$$

where $\Theta(\cdot)$ is a step function.

3.5 Hyperparameters Optimization for Expectation Propagation

The previous sections present the training algorithm of EP with fixed hyperparameter α. In this subsection, we update the hyperparameter α based on the type-II marginal likelihood, also known as the evidence [29]. According to the updated value of α, we choose to add one learner to the ensemble, delete one ensemble member or re-estimate the hyperparameter α.

As described in previous sections, expectation propagation approximates each term as a Gaussian distribution, leading to the situation that the likelihood of every point in a classification ensemble has similar forms as a regression likelihood term. The likelihood of each data point in classifier ensemble can be obtained as

$$p(\mathbf{m}|\mathbf{w}, \mathbf{x}) = \frac{\exp(-\frac{1}{2}(\mathbf{w}^T\mathbf{F} - \mathbf{m})^T\Lambda^{-1}(\mathbf{w}^T\mathbf{F} - \mathbf{m}))}{(2\pi)^N|\Lambda|^{1/2}},$$

where $\mathbf{m} = (m_1, \ldots, m_N)$ denotes the target point vector, $\Lambda = diag(v_1, \ldots, v_N)$, v_n represents the variance of the noise for the training point n. EP actually maps a classification problem into a regression problem where (m_n, v_n) defines the virtual observation data point with mean m_n and variance v_n.

Note that we can compute analytically the posterior distribution of the weights. The posterior distribution of the weight vector is thus given by:

$$\begin{aligned} p(\mathbf{w}|\mathbf{x}, \mathbf{m}, \alpha) &= \frac{p(\mathbf{m}|\mathbf{w}, \mathbf{x})p(\mathbf{w}|\alpha)}{p(\mathbf{m}|\alpha, \mathbf{x})} \\ &= \frac{\exp(-\frac{1}{2}(\mathbf{w} - \mu)^T\mathbf{\Sigma}^{-1}(\mathbf{w} - \mu))}{(2\pi)^N|\mathbf{\Sigma}|^{1/2}}, \end{aligned} \tag{10}$$

where the posterior covariance and mean are:

$$\mathbf{\Sigma} = (A + \mathbf{F}\Lambda^{-1}\mathbf{F}^T)^{-1}, \tag{11}$$

$$\mu = \mathbf{\Sigma}\mathbf{F}\Lambda^{-1}\mathbf{m}. \tag{12}$$

where $A = diag(\alpha_1, \cdots, \alpha_M)$.

For regression ensembles, the posterior of weights can be easily obtained by replacing classification likelihood terms with regression likelihood terms. The posterior of weights has the similar equations as (10), (11) and (12) but with different $\mathbf{m}$ (i.e. $\mathbf{y}$) and $\mathbf{\Lambda}$ (i.e. σ).

In order to sequentially update α, we can maximize the type-II marginal likelihood $p(D|\alpha)$. The fast algorithm to optimize the type-II marginal likelihood is to decompose $p(D|\alpha)$ into two parts, one part denoted by $p(D|\alpha_{\backslash i})$, that does not depend on α_i and another that does, i.e.,

$$p(D|\alpha) = p(D|\alpha_{\backslash i}) + l(\alpha_i), \tag{13}$$

where $l(\alpha_i)$ is a function that depends on α_i.

The updating rule for α_i can be obtained with the derivation of marginal likelihood [29]. The details have been presented in appendix A.

3.6 Algorithm Description

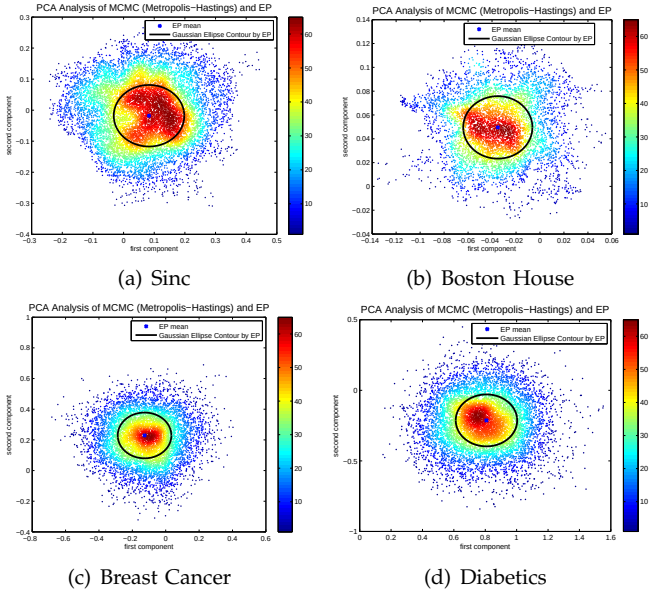
Based on the above subsections, the predictive ensemble pruning algorithm by expectation propagation is summarized as follows:

1) Include a number of learning machines in the ensemble and initialize the hyperparameters α.
2) Train EP algorithm with the current hyperparameters α and sequentially update α by maximizing the type-II marginal likelihood $p(D|\alpha)$. Based on the updated values of α, we choose to add one learner to the ensemble, delete one existing ensemble member or re-estimate the hyperparameter α. Repeat this process until the algorithm converges.
3) Choose the ensemble from the sequential updates with the minimum leave-one-out error estimation. As the leave-one-out error is discrete, so in case of a tie, choose the first ensemble in the tie, i.e., the one with the smaller marginal likelihood[1].

3.7 Comparison of Expectation Propagation with Markov Chain Monte Carlo

Expectation Propagation is a kind of integral approximation technique. It is better to know the difference between the approximation and the exact distribution.

1. Qi et al. [30] pointed out that optimization of marginal likelihood can lead to over-fitting and leave-one-out error with smaller marginal likelihood is a better choice for model selection.



(a) Sinc (b) Boston House

(c) Breast Cancer (d) Diabetics

Fig. 5. The posteriors of combination weights calculated by MCMC (30000 sampling points) and EP. The color bar indicates the density (the number of overlapping points) in each place.

As the truncated Gaussian prior is used in this paper, the exact posterior distribution is unknown. In this section, we employ Markov Chain Monte Carlo (MCMC) method to simulate the exact posterior distribution for the comparison with EP.

MCMC methods [31] are a class of algorithms for sampling from probability distributions based on constructing a Markov chain that has the desired distribution as its equilibrium distribution. MCMC may be too slow for many practical applications, but has the advantage that it becomes exact in the limit of long runs. Thus, MCMC can provide a standard way to measure the accuracy of integral approximation methods, such as expectation propagation in this paper.

This paper uses one of the most well-known MCMC algorithms, Metropolis-Hastings algorithm [31] to investigate regression and classification ensembles, respectively. In our experiments, a Bagging ensemble with 100 Classification And Regression Trees (CARTs) is generated. MCMC and EP with the hyperparameters optimization algorithm are employed for ensemble pruning.

In most of the cases, the pruned ensemble size is larger than 2 or 3 which makes it inconvenient to directly visualize the resulting distribution. To facilitate the visualization, principal components analysis (PCA) is performed and the first two components are used for visualization. Figure 5 illustrates the first two components, calculated by PCA, of the posterior of weighs calculated by MCMC and EP for regression and classification ensembles.

Figure 5 illustrates the posteriors of combination weights calculated by MCMC (30000 sampling points) and EP. We use sinc (with 0.1 Gaussian noise), Boston house, breast cancer and diabetics data sets in this figure. Note that the hyperparameters and noise terms are estimated in the hyperparameters optimization step by maximizing the marginal likelihood in both EP and MCMC methods. The posteriors of weights calculated by MCMC have irregular boundaries for these problems and EP approximates the posteriors well by picking a Gaussian to cover the densest area, although the distribution are not Gaussians, for both regression and classification problems.

TABLE 1
The pruned ensemble size, error rate and computational time of MCMC, EP and unpruned ensembles.

Regression	Sinc			House		
	Size	MSE	Time	size	MSE	Time
MCMC	7	0.0082	343.1s	11	11.4892	398.5s
EP	8	0.0087	8.7s	11	11.5725	11.6s
Unpruned	100	0.0103	-	100	11.8464	-
Classification	**Cancer**			**Diabetics**		
	Size	%error	Time	size	error	Time
MCMC	10	26.34	676.2s	19	24.58	986.3s
EP	11	26.93	19.1s	18	24.73	62.6s
Unpruned	100	27.64	-	100	24.65	-

The pruned ensemble sizes and the error for both regression and classification problems are shown in Table 1. From the table, EP and MCMC achieves similar performance in terms of both accuracy and ensemble size. EP uses much less time than MCMC.

Both figures and table indicate that EP approximates the posterior well in this ensemble pruning model with truncated Gaussian priors for regression and classification problems.

4 Experimental Results

This section presents the experimental results of expectation propagation pruning algorithm for regression problems and classification problems, respectively.

4.1 Synthetic Data Sets

As the first experiment, we compare EP-pruned ensembles with original ensembles on some synthetic data sets, including one regression data set, sinc, and two classification data sets: synth and banana.

Figure 6 shows the output of EP pruning and original Bagging ensembles, which consists of 100 neural networks. We notice that EP pruning is a little better than the original ensemble in the left tail of sinc function. With respect to ensemble size, EP pruning only picks 9 neural networks vs. 100 neural networks in the original ensemble.

In the following synthetic classification data sets, we select the Adaboost of neural networks as the ensemble algorithm because large Adaboost is prone to overfitting the noise in the training set. Figure 7 illustrates the decision boundaries of EP pruning and original Adaboost for both problems. Not surprisingly, Adaboost with 100

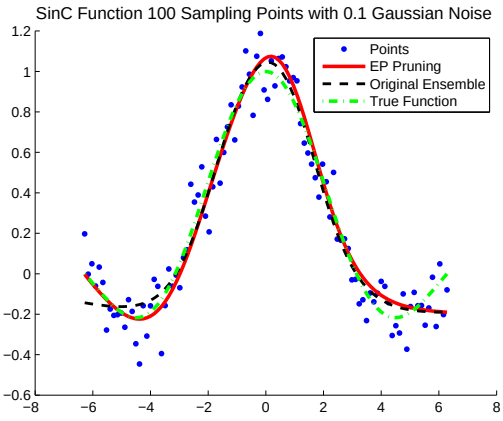
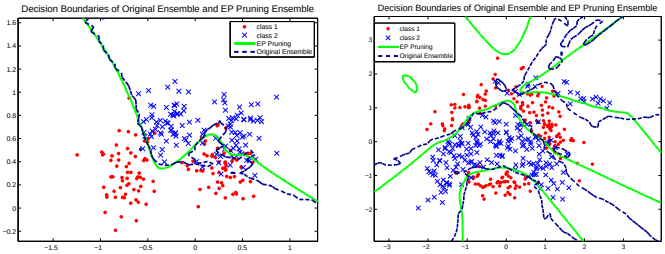
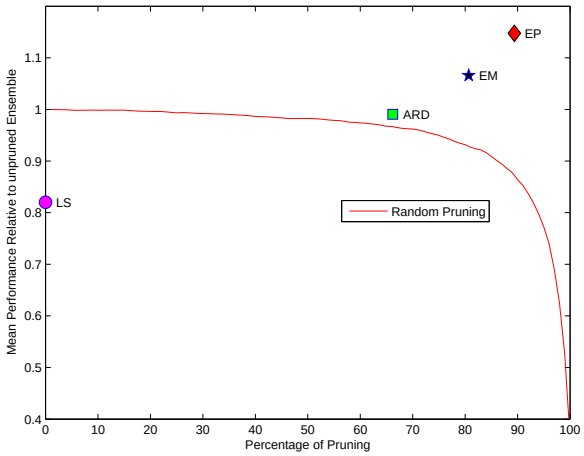


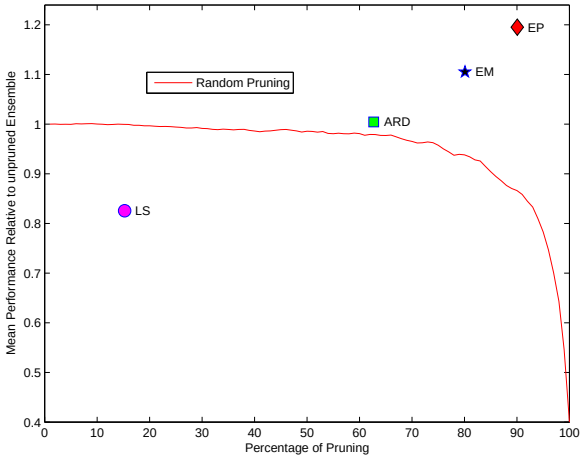
Fig. 6. Comparison of EP-pruned ensembles and original Bagging ensembles on sinc data set. The sinc data set is generated by sampling 100 data points with 0.1 Gaussian noise from the sinc function. The Bagging ensemble consists of 100 three-layered neural networks (MLP) with random selected hidden nodes (3-6 nodes). The weights in these MLPs are randomly initialized.



(a) Synth (b) Banana

Fig. 7. Comparison of EP-pruned ensembles and unpruned Adaboost ensembles on Synth and banana data sets. The Adaboost ensemble consists of 100 neural networks with random selected hidden nodes (3-6 nodes).





(b) Random Forests

Fig. 8. Relative performance of each pruning method averaged across the 7 problems as a function of the amount of pruning. Note that the EP, ARD, EM and LS pruning appear as single points, since the amount of pruning will be determined by these methods. A performance greater than 1.0 indicates that the pruned ensemble actually performed better than unpruned ensemble.

neural networks over-fits the noise and generates the twisty boundaries. With small ensemble size (16 for synth and 12 for banana), EP pruning removes those overfitting individuals and generates better (smoother) decision boundaries for both classification problems.

According to these initial experiments with synthetic data, we observe that EP pruning performs better than the original ensembles by utilizing a small amount of individuals.

4.2 Results for Regression Problems

The experiments in this section will investigate EP pruning for benchmark problems. We utilize decision trees, i.e. classification and regression trees (CART), as base learners to generate different kinds of ensembles, i.e. Bagging, Adaboost and Random Forest (Adaboost is used for classification ensembles only). Each ensemble consists of 100 CARTs.

In section 2, we reviewed a number of ensemble pruning algorithms, such as least-square (LS) pruning, Bayesian automatic relevance determination (ARD) pruning, EM pruning an so on. These algorithms are employed to compared with our EP pruning algorithm in this section.

The information on the data sets used for regression is tabulated in Table 2. The Friedman was used by Breiman [2] in testing the performance of Bagging. Gabor, Multi and Sinc were used by Hansen [32] in comparing several ensemble approaches. Plane was used by Ridgeway et al. [33] in evaluating the performance of boosted naive Bayesian regressors. The constraints of the variables are also shown in Table 2, where $U[x, y]$ means a uniform distribution over the interval determined by x and y. Note that in our experiments additive zero-mean Gaussian noise, whose variance is one-third of the standard deviation of the target $y(x)$, is generated. The Boston House data set is obtained from UCI machine learning

TABLE 2
Summary of Regression Data Sets

Data Sets	Function	Variable	Training Points	Test Points
Sinc	$y = sincx = \frac{\sin x}{x}$	$x \sim U[-2\pi, 2\pi]$	250	1000
Friedman	$y = 10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5$	$x_i \sim U[0,1]$	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
Boston House	—	-	400	106

TABLE 3
Average Test MSE, Standard Deviation for seven regression Benchmark Data sets based on 100 runs for Bagging and random forests. EP, ARD, EM, LS, Random stand for EP pruning, ARD pruning, EM pruning, least square pruning and random pruning (when it has 25 ensemble members), respectively.

Bagging	EP	ARD	EM	LS	Random	Unpruned
Sinc	**0.0087±0.0019**	0.0158±0.0026	0.0098±0.0041	0.0254±0.0036	0.0138±0.0018	0.0102±0.0017
Friedman	**4.4765±0.4287**	4.6327±0.4079	4.5816±0.5436	4.9594±0.4355	4.7711±0.4107	4.6094±0.4196
Gabor	**0.0272±0.0094**	0.0289±0.0080	0.0294±0.0130	0.0311±0.0087	0.0518±0.0118	0.0497±0.0010
Multi	**0.1472±0.0206**	0.1606±0.0198	0.1569±0.0315	0.1994±0.0236	0.1753±0.0206	0.1537±0.0190
Plane	0.0011±0.0002	0.0014±0.0003	0.0014±0.0003	0.0026±0.0004	0.0016±0.0002	**0.0010±0.0002**
Polynomial	**0.3261±0.0522**	0.3614±0.0499	0.3276±0.0531	0.4813±0.0664	0.3765±0.0515	0.3349±0.0487
House	**11.5725±4.0741**	11.5803±4.3361	11.7561±4.6472	12.3202±4.8322	12.2844±4.8322	11.8464±4.4938
W-L-T	-	0-7-0	0-7-0	0-7-0	0-7-0	1-6-0
Significant	-	0-4-3	0-3-4	0-6-1	0-6-1	0-2-5
Random Forests	EP	ARD	EM	LS	Random	Unpruned
Sinc	**0.0094±0.0020**	0.0141±0.0022	0.0113±0.0038	0.0164±0.0028	0.0141±0.0034	0.0133±0.0022
Friedman	**4.3263±0.4998**	4.7465±0.4927	4.5816±0.5436	4.8629±0.5867	5.6519±0.7071	5.0567±0.5756
Gabor	**0.0298±0.0096**	0.0313±0.0086	0.0304±0.0104	0.0397±0.0109	0.0504±0.0123	0.0434±0.0010
Multi	**0.1452±0.0213**	0.1672±0.0198	0.1509±0.0237	0.2179±0.0365	0.1762±0.0318	0.1493±0.0210
Plane	**0.0010±0.0003**	0.0015±0.0003	0.0013±0.0003	0.0024±0.0004	0.0020±0.0002	0.0013±0.0002
Polynomial	0.3287±0.0521	0.3714±0.0493	**0.3276±0.0606**	0.4969±0.0805	0.4032±0.2586	0.3452±0.0510
House	**11.3569±4.2070**	12.2112±4.6749	11.7561±4.6472	12.6589±4.7024	13.3276±5.1642	11.5672±4.3652
W-L-T	-	0-7-0	1-6-0	0-7-0	0-7-0	0-7-0
Significant	-	0-4-3	0-3-4	0-6-1	0-7-0	0-2-5

repository [34]. In the 100 runs, we randomly select 400 data points for the training set and the rest 106 points are used for testing.

For every data set, we run independently 100 times and record the average mean squared error (MSE) and the standard deviation on the test set for different algorithms. Table 3 reports the performance of EP and other four algorithms for Bagging and random forests, respectively. A win-loss-tie summary based on mean values and t-test (95% significance level) is attached at the bottom of the table.

For the random pruning algorithm, we report the performance in table when the pruned ensemble has 25 members since previous empirical research suggests that, in most cases, most or all of the generalization gain in a well-constructed ensemble comes from the first 25 learners added [2], [35]. The performance of random pruning algorithm with different pruning levels is also reported in Figure 8.

We also illustrate the mean normalized performance of each pruning method averaged over the seven data sets in Figure 8. A performance greater than 1.0 indicates that the pruned ensemble actually performed better than unpruned ensemble.

TABLE 4
Size of Pruned Ensemble with standard deviation for Different Algorithms for Bagging and random forests. The results are based on 100 runs. Bag and RF stand for Bagging and random forests, respectively.

Bagging	EP	ARD	EM	LS
Sinc	7.9±1.7	21.4±5.3	10.2±3.3	100
Friedman	12.2±1.9	36.3±5.3	16.9±3.9	100
Gabor	9.6±2.0	44.3±4.6	20.4±4.2	100
Multi	13.6±1.7	34.9±5.0	21.6±5.0	100
Plane	9.3±1.5	24.4±6.3	21.8±3.3	100
Polynomial	11.2±2.1	31.3±5.3	20.5±4.9	100
House	10.5±1.5	44.0±4.4	23.8±5.0	100
Random Forest	EP	ARD	EM	LS
Sinc	8.8±1.9	18.8±4.7	12.1±3.6	51.7±4.8
Friedman	13.4±0.9	45.2±8.3	20.3±4.7	98.1±1.7
Gabor	9.3±2.1	43.9±8.6	22.1±6.4	75.8±4.0
Multi	9.7±0.9	41.3±8.0	23.0±7.2	97.9±1.8
Plane	8.6±1.2	28.4±9.3	17.3±5.8	75.2±4.2
Polynomial	10.4±0.7	35.6±10.5	19.6±7.0	94.5±2.3
House	9.3±1.4	47.9±4.9	24.9±7.3	100

TABLE 5
Summary of Classification Data Sets.

Data	Banana	Cancer	Diabetics	Solar	German	Heart	Image	Ringnorm	Splice	Thyroid	Titanic	Twonorm	Waveform
Train	400	200	468	666	700	170	1300	400	1000	140	150	400	400
Test	4900	77	300	400	300	100	1010	7000	2175	75	2051	7000	4600
Input Dim	2	9	8	9	20	13	18	20	60	5	3	20	21

From tables and Figure 8, it is observed that EP pruning outperforms all the other methods in six out of seven data sets, comes second in other one case. Although ARD pruning uses Bayesian inference for ensemble pruning as well, it seems that adopting the negative combination weights leads to inferior results. The baseline algorithm, random pruning, fails to compete with EP and ARD pruning. In most situations, least square algorithm is worse than other algorithm, which gives the empirical evidence that least square algorithm often over-fits the noise and does not work well in practice.

Another interesting point is that EP pruning achieves better performance by employing only a few of the ensemble members, as shown by Table 4 and Figure 8. From these tables and Figure 8, EP pruning consistently uses much fewer ensemble members than other algorithms, including ARD pruning. This observation goes in accordance with the algorithm of EP pruning. In EP pruning, we employ a sparse prior and sequentially add or delete individuals in the ensemble based on the estimation of type II marginal likelihood. The sparse prior and the implementation lead to spare ensembles.

In general, the performance of EP pruning on these benchmark problems is better than unpruned ensembles in terms of generalization ability and sparsity.

4.3 Results for Classification Problems

For classifier ensembles, we use the data sets, which have been preprocessed and organized by Rätsch et al.[2] to do binary classification tests. These data sets include one synthetic set (banana) along with 12 other real-world data sets from the UCI [34] and DELVE[3]. The characteristics of the data set are summarized in Table 5.

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 test instances (Splice and Image have only 20 splits in the Rätsch's implementation and we generate additional 80 splits by random sampling to make our experiments consistent.) In addition, every instance was input-normalized dimension-wise to have zero mean and unit standard deviation.

In order to compare our algorithm with others, we have implemented ARD pruning, EM pruning, kappa pruning, concurrency pruning, least square pruning and random pruning.

Table 6 reports the performance of these algorithms on the 13 benchmark data sets with Bagging, Adaboosting and Random forests. The size of the ensembles also has been recorded in Tables 7. Figure 9 shows the mean normalized performance of each pruning method averaged over the thirteen datasets as a function of the amount of pruning. A performance greater than 1.0 indicates that the pruned ensemble actually performed better than unpruned ensemble.

According to these tables and Figure 9, EP pruning compares quite favorably against these different ensemble algorithms. For example, for Bagging EP pruning outperforms all the other methods, including the unpruned ensemble on eight out of thirteen data sets, comes second in two cases and third in the remaining three. Comparing with the original ensemble, EP pruning employs much fewer ensemble members but performs better. Take the Adaboost as an example, EP pruning performs better than unpruned ensemble in eight out of thirteen cases, in which four wins are statistically significant; EP pruning loses five times, where two losses are statistically significant.

EM pruning always loses when comparing with EP pruning, though they have similar models. The unstable problems of EM (sensitiveness to initialization and convergence to local maxima) do degrade the performance. EP pruning is more stable than EM pruning based on the results.

Least square (LS) pruning, which minimizes the training error, performs well only on data set with little noise, for example Image. In most situations, LS pruning does not reduce the size of an ensemble. Random pruning, which serves as the baseline algorithm, is not comparable to the original ensemble and other pruning algorithms.

According to these tables, the previous finding that Adaboost is prune to overfitting the noise in the training set is also confirmed as Adaboost performs well on data sets with little noise, such as Image, Twonorm, but worse on noise-corrupted data sets.

In these tables, we only report the performance of Kappa, CP, random pruning with 25 ensemble members. In order to illustrate the relative performance of these algorithms with different pruning levels, we have reported the relative performance of each pruning method averaged across all the problems as a function of the amount of pruning. The experiment was repeated 100 times and the results were averaged.

Based on these tables and Figure 9, EP pruning achieves significant sparseness in ensembles and performs better or as well as the original ensemble. It provides a way to reduce the computational complexity

2. http://ida.first.fraunhofer.de/projects/bench/benchmarks.htm
3. http://www.cs.toronto.edu/~delve/data/datasets.html

TABLE 6
Average Test error, Standard Deviation for 13 classification Benchmark Data sets based on 100 runs for Bagging, Adaboosting and random forests algorithm. EP, ARD, EM, Kappa, CP, LS, Random stand for EP pruning, ARD pruning, EM pruning, kappa pruning, concurrency pruning, lease square pruning and random pruning.

Bagging	EP	ARD	EM	Kappa	CP	LS	Random	Unpruned
Banana	**12.74±0.78**	13.14±0.67	12.88±0.83	13.74±0.73	13.32±0.87	14.54±0.94	13.93±0.85	12.75±0.79
Cancer	**26.81±4.74**	30.96±4.73	27.68±7.31	28.81±4.54	30.30±4.54	34.51±5.03	29.35±4.33	27.42±4.54
Diabetics	24.88±1.93	25.76±1.90	24.83±2.26	26.30±1.79	25.08±1.97	26.15±1.89	26.89±1.95	**24.62±1.81**
Solar	**35.18±1.80**	36.33±1.93	36.11±2.04	36.80±1.98	36.15±1.77	37.15±1.93	37.88±1.82	35.96±1.82
German	**22.15±2.21**	24.19±2.23	22.94±2.68	24.12±2.21	24.67±2.30	25.01±2.27	25.80±2.05	23.63±2.17
Heart	19.13±3.64	22.17±3.56	19.26±3.71	20.01±3.99	20.71±4.11	26.42±3.98	21.08±3.75	**19.09±3.89**
Image	**2.04±0.48**	2.20±0.47	2.31±0.58	2.40±0.54	2.31±0.48	2.35±0.46	2.40±0.53	2.32±0.50
Ringnorm	8.68±1.16	10.44±1.53	8.95±1.56	8.88±1.16	9.26±1.42	**8.00±1.20**	9.14±1.36	8.08±1.37
Splice	5.03±0.67	5.18±0.69	5.04±0.69	5.13±0.75	5.15±0.69	5.04±0.70	5.41±0.79	**5.02±0.77**
Thyroid	**6.27±3.04**	7.03±5.14	6.37±3.89	6.87±3.04	7.35±2.95	9.19±3.25	7.49±3.21	6.83±3.20
Titanic	**22.36±1.31**	23.72±1.61	22.60±1.52	22.56±1.67	24.00±1.64	22.49±1.21	24.50±0.96	22.57±1.01
Twonorm	7.24±1.04	12.55±6.48	7.34±1.96	7.98±0.88	8.47±1.60	7.18±0.98	8.52±1.04	**6.55±1.34**
Waveform	**13.12±0.57**	14.35±0.48	13.89±0.91	14.10±0.65	13.52±0.63	13.84±0.89	14.24±0.68	13.67±0.71
W-L-T	-	0-13-0	0-13-0	0-13-0	0-13-0	2-11-0	0-13-0	5-8-0
Significant	-	0-7-6	0-5-8	0-5-8	0-7-6	0-8-5	0-11-2	1-3-9
Adaboosting	EP	ARD	EM	Kappa	CP	LS	Random	Unpruned
Banana	**13.49±0.65**	14.19±0.76	13.81±0.82	16.35±1.48	13.40±0.72	14.23±0.67	16.13±0.69	13.51±0.60
Cancer	31.88±4.15	31.64±6.21	32.41±5.32	37.40±6.58	34.86±5.87	32.70±4.97	36.94±6.42	**31.16±4.47**
Diabetics	**25.72±2.42**	28.78±2.33	26.47±2.71	28.21±2.52	28.13±2.07	26.25±1.92	29.15±2.12	26.06±1.99
Solar	**34.28±1.87**	36.37±1.98	36.28±2.01	39.46±2.38	39.95±2.38	38.59±1.97	40.25±5.43	36.26±1.78
German	24.37±2.55	27.39±2.43	25.55±2.89	26.73±2.35	25.21±2.57	24.21±2.05	28.05±2.44	**24.06±2.19**
Heart	**18.40±4.25**	23.33±3.41	20.62±5.13	28.33±4.68	22.65±4.00	21.74±3.76	22.79±3.98	20.82±3.97
Image	1.23±0.54	1.78±0.72	1.83±0.71	1.46±0.55	1.23±0.37	1.15±0.35	1.40±0.42	**1.12±0.35**
Ringnorm	**3.82±0.53**	4.65±0.61	4.27±0.81	5.14±1.08	4.43±0.78	4.37±0.49	6.39±0.75	4.09±0.47
Splice	4.17±0.85	4.40±0.62	4.22±0.92	6.4±0.68	4.10±0.58	3.94±0.49	6.12±0.77	**3.55±0.54**
Thyroid	5.09±2.70	7.24±2.90	6.51±3.15	8.39±8.04	5.58±2.57	7.41±3.24	7.00±3.38	**4.69±2.40**
Titanic	**21.40±0.79**	23.76±0.82	22.05±0.93	28.98±0.84	26.12±0.79	23.22±0.96	27.36±1.01	21.98±0.70
Twonorm	**3.52±0.41**	5.49±0.52	4.01±0.72	6.08±0.52	5.01±0.83	4.14±0.44	5.85±0.39	3.79±0.30
Waveform	**10.47±0.52**	11.88±0.62	11.29±0.90	14.12±0.64	12.43±0.70	11.58±0.59	14.48±0.47	11.42±0.50
W-L-T	-	1-12-0	0-13-0	0-13-0	1-12-0	3-10-0	0-13-0	5-8-0
Significant	-	0-7-5	0-5-9	0-10-3	0-9-4	0-5-7	0-12-1	2-4-7
Random Forest	EP	ARD	EM	Kappa	CP	LS	Random	Unpruned
Banana	12.76±0.44	13.83±0.42	13.12±0.76	13.70±0.49	13.19±0.57	15.86±0.86	16.24±0.79	**12.72±0.48**
Cancer	**24.86±4.66**	26.66±4.65	26.58±6.42	26.86±4.69	26.18±4.47	34.79±4.69	28.43±4.21	24.92±4.10
Diabetics	24.67±1.98	**24.35±2.14**	25.17±2.59	30.79±2.38	25.12±1.76	29.81±2.21	29.45±2.18	25.20±1.74
Solar	34.90±1.79	36.31±1.90	35.76±2.84	36.78±2.76	39.70±4.60	37.37±2.06	38.85±2.31	**34.59±1.91**
German	**23.64±2.37**	24.96±2.38	23.96±2.81	27.23±2.48	24.51±2.29	29.72±2.29	28.18±2.24	24.32±2.33
Heart	18.08±4.16	18.31±4.27	18.21±4.79	19.34±4.20	19.63±4.06	26.32±4.11	19.71±3.87	**17.43±3.72**
Image	1.81±0.40	1.83±0.58	1.83±0.71	2.37±0.81	1.98±0.48	**1.67±0.42**	2.43±0.34	1.84±0.44
Ringnorm	**3.71±0.63**	4.07±0.76	4.19±0.96	5.50±0.65	6.07±0.98	5.19±0.68	6.00±0.85	4.63±0.68
Splice	3.63±0.51	3.70±0.68	3.66±0.74	3.64±1.42	3.99±0.48	3.42±0.39	3.84±0.48	**2.91±0.35**
Thyroid	**5.25±2.84**	6.13±2.83	6.07±3.10	8.36±2.87	5.78±2.54	8.49±3.92	8.42±3.14	5.71±2.78
Titanic	22.44±1.31	22.76±2.97	23.11±2.42	24.19±2.20	24.23±2.11	**22.41±1.17**	23.96±2.73	23.55±2.15
Twonorm	**3.89±0.51**	4.21±0.81	4.14±0.88	5.93±0.56	5.34±0.75	5.39±0.53	6.06±0.60	4.31±0.40
Waveform	11.98±0.75	12.06±0.85	12.14±1.24	13.06±0.85	12.65±0.76	13.88±0.76	12.59±0.72	**11.57±0.63**
W-L-T	-	1-12-0	0-12-1	0-13-0	0-13-0	3-10-0	0-13-0	5-8-0
Significant	-	0-5-8	0-7-6	0-9-4	0-7-6	0-9-4	0-7-6	1-2-10

at the *test* stage and make the ensemble more compact. There are two possible reasons to explain the success of EP pruning.

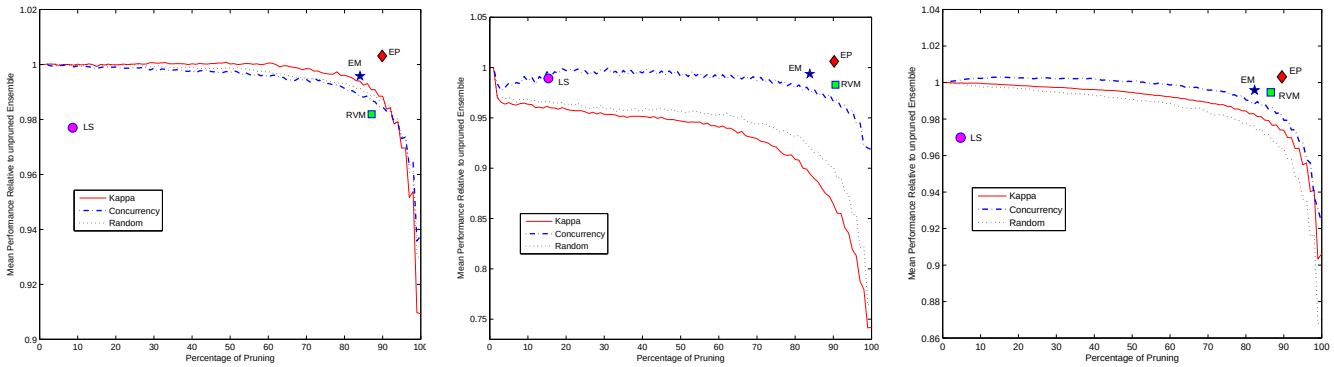
1) EP pruning benefits from the truncated Gaussian priors. As negative combination weights are neither intuitive nor reliable [17]–[19], the truncated Gaussian prior not only satisfies the constraint but also leads to a sparse ensemble. This prior controls the complexity by generating appropriate sparseness, and thus improves the generalization.
2) EP pruning employs the leave-one-out error together with the Bayesian evidence as the criterion for model selection, which is more effective than the other algorithms.

4.4 Statistical Comparisons over Multiple Data Sets

In the previous subsection, 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 [36]. In this section, we will conduct statistical tests over multiple data sets by using the Friedman test [37] with the corresponding post-hoc tests.

TABLE 7
Size of Pruned Ensemble with standard deviation with Different Algorithms for Bagging, Adaboosting and random forests The results are based on 100 runs. Bag, Ada and RF stand for Bagging, Adaboosting and random forests, respectively.

	Bag EP	Bag ARD	Bag EM	Bag LS	Ada EP	Ada ARD	Ada EM	Ada LS	RF EP	RF ARD	RF EM	RF LS
Banana	10.2±2.7	12.7±1.7	16.3±4.6	100	10.9±1.9	10.6±1.8	16.8±4.3	100	14.1±2.4	15.1±4.7	20.3±4.1	76.0±4.5
Cancer	9.8±2.7	17.5±2.0	14.5±3.8	100	11.4±2.3	13.2±1.7	17.3±5.8	65.0±19.2	6.7±1.6	19.3±3.1	13.1±4.0	98.5±1.4
Diabetics	17.5±2.3	18.6±1.8	20.4±3.6	100	12.5±1.9	12.4±2.1	21.8±4.0	100	16.8±4.2	20.3±2.0	24.6±5.6	99.9±0.1
Solar	5.7±1.8	7.0±1.0	11.3±4.1	69.4±4.3	8.3±2.8	11.8±1.7	16.2±5.1	49.8±17.1	7.1±2.6	11.9±2.0	10.2±4.0	84.5±3.6
German	17.9±3.7	25.0±2.5	26.7±5.1	100	12.4±1.8	12.3±1.2	18.8±4.3	100	16.0±4.8	27.1±2.2	23.4±5.3	100
Heart	10.1±1.7	10.1±1.5	12.9±2.2	100	10.9±1.6	11.0±2.1	17.0±4.1	100	11.1±1.6	11.2±1.6	18.6±3.1	100
Image	9.4±1.4	9.5±1.5	16.3±3.9	100	8.8±2.3	6.2±2.7	14.2±4.4	100	10.1±1.5	8.9±2.9	19.3±3.3	100
Ringnorm	10.1±1.6	8.0±2.4	22.3±5.1	100	10.4±2.7	8.7±2.5	21.7±5.3	100	10.7±1.4	6.4±2.6	21.6±5.2	100
Splice	11.4±2.4	12.2±1.7	15.7±3.7	100	8.9±1.2	8.3±2.2	14.3±3.9	87.3±12.6	12.8±1.9	12.1±2.3	18.4±4.2	100
Thyroid	5.1±1.2	4.9±2.2	10.2±4.2	43±6.2	5.9±0.9	5.1±2.9	11.3±3.9	87.3±18.3	5.6±1.2	4.9±2.0	11.3±5.1	82.5±5.9
Titanic	3.3±1.1	25.9±22.5	8.7±3.5	74.8±1.4	4.9±0.9	5.6±0.7	9.6±2.4	10.7±1.8	3.7±1.5	22.3±18.1	10.2±4.5	96.8±1.4
Twonorm	10.6±1.5	5.3±3.5	17.2±4.9	100	11.3±1.6	8.6±1.4	16.3±3.1	100	10.8±1.4	4.8±2.9	17.3±3.2	100
Waveform	10.3±1.8	10.8±1.2	14.6±3.7	100	10.5±2.0	9.1±2.9	14.7±3.0	100	11.1±1.4	10.4±2.2	21.7±4.4	100



(a) Bagging (b) Adaboosting (c) Random Forests

Fig. 9. Relative performance of each pruning method averaged across the 13 problems as a function of the amount of pruning. Note that the EP, ARD, EM and LS pruning appear as single points, since the amount of pruning will be determined by these methods.

TABLE 8
The mean rank of these algorithms with different ensemble algorithms and unpruned ensemble.

Mean Rank	EP	ARD	EM	Unpruned
(Regression) Bagging	1.1429	3.3571	2.6429	2.8571
(Regression) RF	1.1429	3.7143	2.2429	2.9286
Bagging	1.4615	3.8462	2.7692	1.9231
Adaboosting	1.4615	3.7692	3.0000	1.7692
Random Forest	1.4615	3.1923	2.8846	2.4615

The Friedman test is a non-parametric equivalence of the repeated-measures analysis of variance (ANOVA) under the null hypothesis that all the algorithms are equivalent and so their ranks should be equal. This paper uses an improved Friedman test proposed by Iman and Davenport [38].

The Friedman test [37] is carried out to test *whether all the algorithms are equivalent*. If the test result rejects the null hypothesis, i.e. these algorithms are equivalent, we can proceed to a post-hoc test. The power of the post-hoc test is much greater when all classifiers are compared with a control classifier and not among themselves. We do not need to make pairwise comparisons when we in fact only test whether a newly proposed method is better than the existing ones.

Based on this point, we would like to choose the EP pruning algorithm as the control classifier to be compared with. Since the baseline classification algorithms are not comparable to EP, ARD and EM, this section will analyze only three algorithms: ARD, EM and unpruned ensembles against the control algorithm EP.

The Bonferroni-Dunn test [39] 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[4] differ by at least the critical difference

$$CD = q_\alpha \sqrt{\frac{j(j+1)}{6T}}, \tag{14}$$

where j is the number of algorithms, T is the number of data sets and critical values q_α can be found in [36]. For

4. We rank these algorithms based on the metric on each data set and record the ranking of each algorithm as 1, 2 and so on. Average ranks are assigned in case of ties. The average rank of one algorithm is obtained by averaging over all of data sets. Please refer to Table 8 for the mean rank of these algorithms under different metrics.

TABLE 9
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 $q_{0.10} = 2.128$.

Algo.	Frie.test	$CD_{0.10}$	ARD	EM	Unpruned
Reg.BAG	0.000	1.47	2.2142	1.5000	1.7142
Reg.RF	0.000	1.47	2.5714	1.1000	1.7857
BAG	0.000	1.08	2.3847	1.3077	0.4616
ADA	0.000	1.08	2.3077	1.5385	0.3077
RF	0.002	1.08	1.7308	1.4231	1.0000

TABLE 10
Running Time of EP pruning, ARD pruning abd EM pruning on Regression Data Sets in seconds. Results are averaged over 100 runs.

Time	Sinc	Fried.	Gabor	Multi	Plane	Poly.	House
EP	8.7s	9.3s	8.6s	7.6s	7.2s	9.2	11.6s
EM	0.6s	0.5s	0.7s	0.6s	0.3s	0.6s	1.1s
ARD	0.3s	0.2s	0.3s	0.3s	0.1	0.3s	0.4s

example, when $j = 4$, $q_{0.10} = 2.128$, where the subscript 0.10 is the threshold value.

Table 8 lists the mean rank of these algorithms using different ensemble training algorithms. Table 9 gives the Friedman test results. Since we employ the same threshold 0.10 for these ensemble training algorithms, the critical differences are $CD = 1.47$ (where $j = 4$ and $T = 7$) and $CD = 1.08$ (where $j = 4$ and $T = 13$) for regression and classification, respectively. Several observations can be made from our results.

Firstly, the null hypothesis that all the algorithms are equivalent is rejected for each algorithms in Table 8.

Secondly, for the Bagging regression problems, the differences between EP and other algorithms including ARD, unpruned ensemble are greater than the critical difference, so the differences are significant, which means the EP is significantly better than EP and Unpruned ensemble in this current experimental setting. The difference (1.1000) between EP and EM for random forests is below the critical difference. We could not detect any significant difference between EM and EP. The correct statistical statement would be that *the experimental data are not sufficient to reach any conclusion regarding the difference between EP and EM for random forests in regression problems.*

Thirdly, for classification problems, EP significantly outperforms ARD and EM. Since the differences between EP and unpruned ensemble are smaller than the critical difference, we cannot draw any conclusion about the difference between EP vs. unpruned ensemble for classification problems in our experimental settings.

4.5 Computational Complexity and Running Time

The improved performance of our algorithm comes with a price: more computation time during the *training* stage. Tables 10 and 11 show the average running time of EP pruning and other pruning algorithms over 100 runs for regression and classification problems, respectively. The computational environment is Windows XP with Intel Core 2 Duo 1.66G CPU and 2G RAM. These algorithms are implemented in MATLAB.

According to the algorithm in section 3, EP pruning is an iterative algorithm and it consists of two major parts: EP training and sequential update of hyperparameters α.

In the first part, EP processes each data point in $O(M^2)$ time, where M is the size of *current* ensemble. Assuming the number of iterations is constant, which seems to be true in practice, the computational complexity of EP training in the first part is $O(NM^2)$, where N is the number of training points. In the second step, the major running time is consumed in calculating vector products, which can be done quickly. Most of the computation time is consumed in the first part.

Although we cannot prove the convergence of EP, in our experiments it always converges for ensemble pruning with Gaussian (for regression) or probit (for classification) likelihood. In practice, 200 iterations have been adopted in our ensemble pruning algorithm. Therefore, the total estimated computational complexity of EP pruning is around $O(iter * NM^2)$, where $iter$ is the number of iterations.

As indicated in the introduction, ensemble pruning algorithms can improve accuracy and reduce the test time, but will lead to a longer training time. Ensemble pruning algorithms are particularly suited to the applications that are characterized by small training but large test sets. In this subsection, we will provide an example with a relatively small training set and a large test set, namely the poker hand data set from the UCI machine learning repository [34].

The data set consists of 25010 training examples and 1 million test examples. Each example represents a hand holding five playing cards drawn from a standard deck of 52. Each card is described by two attributes (suit and rank), for a total of 10 predictive attributes (corresponding to the 5 cards). There are ten classes in the data set and we merge the ten classes into 2 classes. One class means nothing[5] in hand and another class means that there is something (one pair, two pairs, flush, royal flush, etc.) in hand. The percentages of the two classes are nearly balanced in both training and test data sets. Although a manually-specified rule can successfully classify the data set, this task is not easy for a machine learning algorithm operating on the provided vectorial feature representation.

We use 100 CART trees to generate a Bagging ensemble and we use different ensemble pruning algorithms

5. In the five cards, there is *no* one pair, two pairs, three of a kind, straight (five cards, sequentially ranked with no gaps), flush (five cards with the same suit), full house (pair plus different rank three of a kind), four of a kind (four equal ranks within five cards), straight flush (straight plus flush) or royal flush (Ace, King, Queen, Jack, Ten plus flush).

TABLE 11
Running Time of EP pruning, ARD pruning, EM pruning, Kappa pruning and concurrency pruning on Classification Data Sets in seconds. Results are averaged over 100 runs.

Time	Banana	Cancer	Diabetics	Solar	German	Heart	Image	Ringnorm	Splice	Thyroid	Titanic	Twonorm	Waveform
EP	56.3s	19.1s	62.6s	42.2s	88.4s	21.4s	184.4s	83.5s	136.2s	21.7s	3.1s	84.6s	82.5s
EM	1.6s	1.1s	3.4s	2.6s	4.9s	1.7s	6.3s	4.9s	3.8s	1.3s	0.9s	4.7s	5.3s
ARD	0.7s	0.3s	1.3s	0.7s	2.0s	0.4s	2.5s	1.8s	1.6s	0.4s	0.2s	1.8s	1.8s
Kappa	0.8s	0.7s	0.9s	1.0s	1.0s	0.7s	1.5s	0.9s	1.3s	0.7s	0.6s	0.8s	0.8s
CP	1.2s	0.6s	1.3s	2.1s	2.7s	0.6s	7.2s	1.2s	4.1s	0.5s	0.5s	1.2s	1.2s

to prune the ensemble. The experimental results are summarized in Table 12. According to Table 12, the EP pruning algorithm has improved both accuracy and efficiency. For example, the pruned ensemble outperformed the unpruned ensemble in terms of accuracy (error rate 25.68% vs. 27.94%). The total time (2806.1 seconds) including the EP training time and the application time for the pruned ensemble is much smaller than the application time for the unpruned ensemble (6154.9 seconds).

TABLE 12
Summary of EP, EM, ARD, LS, Kappa, CP, random and other unpruned ensembles with poker hand problem (Train points 25010 and Test points 1 mil.). The results are averaged over ten runs.

Summary	Error %	Size	Train Time	Test Time	Total Time
Unpruned	27.94	100	-	6154.9s	6154.9s
EP	**25.68**	**11.6**	2129.4s	677.4s	2806.8s
EM	27.84	19.2	2463.7s	1181.7s	3645.4s
ARD	28.42	29.6	2236.2s	1829.6s	4065.8s
LS	29.13	100	6.8s	6189.2s	6196.0s
Kappa	31.15	25	195.7s	1455.6s	1651.3s
CP	28.73	25	458.6s	1479.3s	1937.9s
Random	31.92	25	1.1s	1464.3s	1465.4s

Figure 10 illustrates the total time as a function of the number of test examples. According to the figure, though EP needed more time in training, the pruned ensemble is much smaller and thus consumed considerably less time in testing than the unpruned ensemble. When the number of test examples increased, EP used less time than the unpruned ensemble.

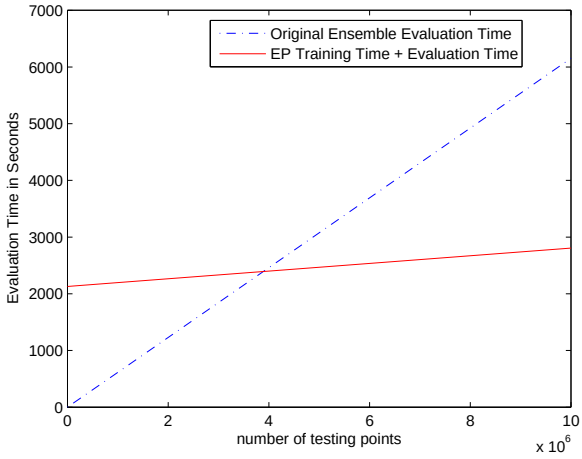


Fig. 10. Comparison of evaluation time of each pruning method averaged.

5 Conclusion

Given that large ensemble may not always be better than small ensemble [10], [11] and large ensemble consumes much more computational resources, this paper propose a probabilistic ensemble pruning algorithm in order to get a set of sparse combination weights to prune an ensemble.

As the previous research implies that negative combination weights in the ensemble may degrade the performance, we introduce a left-truncated, non-negative, Gaussian prior over every combination weight in this probabilistic model to prevent negative weights. However, after incorporating the truncated Gaussian prior, the normalization integral becomes intractable and expectation propagation has been used to approximate the posterior calculation. As the leave-one-out (LOO) error can be obtained as a byproduct in EP training without extra computation, the LOO error is used together with Bayesian evidence for ensemble pruning.

An empirical study on several regression/classification benchmark data sets shows that our algorithm utilizes far less component learners but performs as well as, or better than, the unpruned ensemble. The results are very competitive compared with ARD pruning and some other heuristic algorithms.

EP pruning offers a way to estimate the combination weights and prune the ensemble with the following compelling advantages: a) Good generalization ability. Although our algorithm employs only a few of the ensemble members, they perform as well as, or better than, the unpruned ensemble; b) The highly spare model is obtained by the sparseness-inducing prior and behaves optimally compact; c) No parameters to tune.

Currently, most of the ensemble pruning algorithms, including the proposed algorithm in this paper, carry out post-pruning, i.e., pruning after all learners are generated, which is off-line pruning. Since the EP pruning algorithm operates sequentially by adding one learner, deleting one learner or re-estimating the hyperparameter α, which acts like an "online" pruning algorithm, it is possible to generalize the EP pruning algorithm to a pre-pruning scenario to reduce the computational complex-

ity. This will be our future work. Other further work for this study is to generalize the pruning algorithm to multi-class problems.

APPENDIX

.1 Further Details of Hyperparameters Optimization

The following analysis is based on the sequential analysis of sparse Bayesian learning [29]. Please refer to [29] for more details.

To have a sequential update on α_i, we explicitly decompose $p(D|\alpha)$ into two parts, one part denoted by $p(D|\alpha_{\setminus i})$, that does not depend on α_i and another that does, i.e.,

$$p(D|\alpha) = p(D|\alpha_{\setminus i}) + \frac{1}{2}(\log \alpha_i - \log(\alpha_i + r_i) + \frac{u_i^2}{\alpha_i + r_i}),$$

where $r_i = F_i C_{\setminus i}^{-1} F_i^T$, $u_i = F_i C_{\setminus i}^{-1} \mathbf{m}$, and $C_{\setminus i} = \Lambda^{-1} + \sum_{m \neq i} F_m^T F_m$. Here F_i and F_m are the i^{th} and the m^{th} rows of the ensemble matrix $\mathbf{F}$ respectively. Using the above equation, $p(D|\alpha)$ has a maximum with respect to α_i:

$$\alpha_i = \frac{r_i^2}{u_i^2 - r_i}, \quad \text{if } \eta_i > 0, \tag{15}$$

$$\alpha_i = \infty, \quad \text{if } \eta_i \leq 0, \tag{16}$$

where $\eta_i = u_i^2 - r_i$. Thus, in order to maximize the evidence, we introduce the i^{th} learner when $\alpha_i = \infty$ and $\eta_i > 0$, exclude the i^{th} learner when $\alpha_i < \infty$ and $\eta_i \leq 0$, and re-estimate α_i according to (15) when $\alpha_i < \infty$ and $\eta_i > 0$.

REFERENCES

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

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

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

[4] L. Breiman, "Arcing classifier," *Annals of Statistics*, vol. 26, no. 3, pp. 801–849, 1998.

[5] ——, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[6] J. J. Rodriguez, L. I. Kuncheva, and C. J. Alonso, "Rotation forest: A new classifier ensemble method," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 10, pp. 1619–1630, 2006.

[7] D. Zhang, S. Chen, Z. Zhou, and Q. Yang, "Constraint projections for ensemble learning," in *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI'08)*, 2008, pp. 758–763.

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

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

[10] X. Yao and Y. Liu, "Making use of population information in evolutionary artificial neural networks," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 28, no. 3, pp. 417–425, 1998.

[11] Z. Zhou, J. Wu, and W. Tang, "Ensembling neural networks: many could be better than all," *Artificial Intelligence*, vol. 137, no. 1-2, pp. 239–263, 2002.

[12] T. G. Dietterich, "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization," *Machine Learning*, vol. 40, no. 2, pp. 139–157, 2000.

[13] D. D. Margineantu and T. G. Dietterich, "Pruning adaptive boosting," in *Proceedings of the Fourteenth International Conference on Machine Learning*, 1997, pp. 211–218.

[14] R. E. Banfield, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer, "Ensemble diversity measures and their application to thinning," *Information Fusion*, vol. 6, no. 1, pp. 49–62, 2005.

[15] Y. Kim, W. N. Street, and F. Menczer, "Meta-evolutionary ensembles," in *Proceedings of the 2002 International Joint Conference on Neural Networks*, vol. 3, 2002, pp. 2791–2796.

[16] H. Chen, P. Tino, and X. Yao, "A probabilistic ensemble pruning algorithm," in *Workshops on Optimization-based Data Mining Techniques with Applications in Sixth IEEE International Conference on Data Mining*, 2006, pp. 878–882.

[17] L. Breiman, "Stacked regressions," *Machine Learning*, vol. 24, no. 1, pp. 49–64, 1996.

[18] S. Hashem, "Optimal linear combinations of neural networks," Ph.D. dissertation, Purdue University, 1993.

[19] M. LeBlanc and R. Tibshirani, "Combining estimates in regression and classification," *Journal of the American Statistical Association*, vol. 91, no. 436, pp. 1641–1650, 1996.

[20] T. P. Minka, "Expectation propagation for approximate bayesian inference," in *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, 2001, pp. 362–369.

[21] N. V. Chawla, L. O. Hall, K. W. Bowyer, and W. P. Kegelmeyer, "Learning ensembles from bites: A scalable and accurate approach," *Journal of Machine Learning Research*, vol. 5, pp. 421–451, 2004.

[22] A. Prodromidis and P. Chan, "Meta-learning in a distributed data mining system: Issues and approaches," in *Proceedings of the Fourteenth International Conference on Machine Learning*, 1998, pp. 211–218.

[23] Y. Zhang, S. Burer, and W. N. Street, "Ensemble pruning via semi-definite programming," *Journal of Machine Learning Research*, vol. 7, pp. 1315–1338, 2006.

[24] J. M. Bates and C. W. J. Granger, "The combination of forecasts," *Operations Research*, vol. 20, pp. 451–468, 1969.

[25] J. A. Benediktsson, J. R. Sveinsson, O. K. Ersoy, and P. H. Swain, "Parallel consensual neural networks," *IEEE Transaction on Neural Networks*, vol. 8, no. 1, pp. 54–64, 1997.

[26] N. Ueda, "Optimal linear combination of neural networks for improving classification performance," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 2, pp. 207–215, 2000.

[27] A. Demiriz, K. P. Bennett, and J. Shawe-Taylor, "Linear programming boosting via column generation," *Machine Learning*, vol. 46, no. 1–3, pp. 225–254, 2002.

[28] M. E. Tipping, "Sparse bayesian learning and the relevance vector machine," *Journal of Machine Learning Research*, vol. 1, pp. 211–244, 2001.

[29] A. Faul and M. Tipping, "Analysis of sparse bayesian learning," in *Advances in Neural Information Processing Systems 14*, 2002, pp. 383–389.

[30] Y. Qi, T. P. Minka, R. W. Picard, and Z. Ghahramani, "Predictive automatic relevance determination by expectation propagation," in *ICML '04: Proceedings of the Twenty-first International Conference on Machine Learning*, 2004, p. 85.

[31] C. Andrieu, N. d. Freitas, A. Doucet, and M. I. Jordan, "An introduction to mcmc for machine learning," *Machine Learning*, vol. 50, no. 1–2, pp. 5–43, 2003.

[32] 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.

[33] G. Ridgeway, D. Madigan, and T. Richardson, "Boosting methodology for regression problems," in *Proceedings of Artificial Intelligence and Statistics*, 1999, pp. 152–161.

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

[35] D. Opitz and R. Maclin, "Popular ensemble methods: An empirical study," *Journal of Artificial Intelligence Research*, vol. 11, pp. 169–198, 1999.

[36] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine learning research*, vol. 7, pp. 1–30, 2006.
[37] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *Journal of the American Statistical Association*, vol. 32, pp. 675–701, 1937.
[38] R. L. Iman and J. M. Davenport, "Approximations of the critical region of the friedman statistic," *Communications in Statistics*, pp. 571–595, 1980.
[39] O. J. Dunn, "Multiple comparisons among means," *Journal of the American Statistical Association*, vol. 56, pp. 52–64, 1961.

Huanhuan Chen received the B.Sc. degree from the University of Science and Technology of China, Hefei, China, in 2004, and Ph.D. degree, sponsored by Dorothy Hodgkin Postgraduate Award (DHPA), in computer science at the University of Birmingham, Birmingham, UK, in 2008.

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

Dr. Chen is the recipient of 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).

Peter Tiňo received the M.Sc. degree from the Slovak University of Technology, Bratislava, Slovakia, in 1988 and the Ph.D. degree from the Slovak Academy of Sciences, Slovakia, in 1997.

He was a Fullbright Fellow at the NEC Research Institute in Princeton, NJ, USA, from 1994 to 1995. He was a Postdoctoral Fellow at the Austrian Research Institute for AI in Vienna, Austria, from 1997 to 2000, and a Research Associate at the Aston University, UK, from 2000 to 2003. He is with the School of Computer Science, the University of Birmingham, UK, since 2003 and is currently a a senior lecturer. He is on the editorial board of several journals. His main research interests include probabilistic modeling and visualization of structured data, statistical pattern recognition, dynamical systems, evolutionary computation, and fractal analysis.

Dr. Tiňo is a recipient of the Fullbright Fellowship in 1994. He was awarded the Outstanding Paper of the Year for IEEE Transactions on Neural Networks with T. Lin, B.G. Horne, and C.L. Giles in 1998 for the work on recurrent neural networks. He won the 2002 Best Paper Award at the International Conference on Artificial Neural Networks with B. Hammer.

Xin Yao (M'91-SM'96-F'03)received the B.Sc. degree from the University of Science and Technology of China (USTC), Hefei, Anhui, in 1982, the M.Sc. degree from the North China Institute of Computing Technology, Beijing, in 1985, and the Ph.D. degree from USTC in 1990.

He was an Associate Lecturer and Lecturer from 1985 to 1990 at USTC, while working towards his Ph.D on simulated annealing and evolutionary algorithms. He took up a Postdoctoral Fellowship in the Computer Sciences Laboratory, Australian National University (ANU), Canberra, in 1990, and continued his work on simulated annealing and evolutionary algorithms. He joined the Knowledge-Based Systems Group, CSIRO (Commonwealth Scientific and Industrial Research Organisation) Division of Building, Construction and Engineering, Melbourne, in 1991, working primarily on an industrial project on automatic inspection of sewage pipes. He returned to Canberra in 1992 to take up a lectureship in the School of Computer Science, University College, University of New South Wales (UNSW), Australian Defence Force Academy (ADFA), where he was later promoted to a Senior Lecturer and Associate Professor. Attracted by the English weather, he moved to the University of Birmingham, U.K., as a Professor (Chair) of Computer Science on the April Fool's Day in 1999. Currently, he is the Director of the Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA) and a Changjiang (Visiting) Chair Professor (Cheung Kong Scholar) at the University of Science and Technology of China, Hefei. He was the Editor-in-Chief of the IEEE Transactions on Evolutionary Computation (2003-08), an associate editor or editorial board member of twelve other journals, and the Editor of the World Scientific Book Series on Advances in Natural Computation. He has given more than 50 invited keynote and plenary speeches at conferences and workshops worldwide. His major research interests include evolutionary artificial neural networks, automatic modularization of machine learning systems, evolutionary optimization, constraint handling techniques, computational time complexity of evolutionary algorithms, coevolution, iterated prisoner's dilemma, data mining, and real-world applications. He has more than 300 refereed publications. He was awarded the President's Award for Outstanding Thesis by the Chinese Academy of Sciences for his Ph.D. work on simulated annealing and evolutionary algorithms in 1989. He won the 2001 IEEE Donald G. Fink Prize Paper Award for his work on evolutionary artificial neural networks.