A Probabilistic Ensemble Pruning Algorithm

Huanhuan Chen, Peter Tino and Xin Yao
Cercia, School of Computer Science
University of Birmingham
Birmingham, United Kingdom B15 2TT
{H.Chen, P.Tino, X.Yao}@cs.bham.ac.uk

Abstract

An ensemble is a group of learners that work together as a committee to solve a problem. However, the existing ensemble training algorithms sometimes generate unnecessary large ensembles, which consume extra computational resource and may degrade the performance. Ensemble pruning algorithm aims 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 non-pruned ensemble. This paper will introduce a probabilistic ensemble pruning algorithm by choosing a set of "sparse" combination weights, most of which are zero, to prune the large ensemble. In order to obtain the set of sparse combination weights and satisfy the non-negative restriction of the combination weights, a left-truncated, non-negative, Gaussian prior is adopted over every combination weight. Expectation-Maximization algorithm is employed to obtain maximum a posterior (MAP) estimation of weight vector. Four benchmark regression problems and another four benchmark classification problems have been employed to demonstrate the effectiveness of the method.

1 Introduction

Ensemble of multiple learning machines, i.e. a group of learners that work together as a committee, has received a lot of research interests because it is thought as a good approach to improve the generalization ability [6]. Because of the simple and effective properties, ensemble has become a hot topic in the machine learning communities. There have been many approaches to train ensemble, such as Bagging [2], Boosting [10], negative correlation learning and evolutionary computation based algorithms [8][7].

In general, the training of ensemble can be decomposed into two steps, i.e., training a number of ensemble members and then combining their predictions. In the second step, most ensemble training algorithms employ all of the available ensemble members to constitute an ensemble, which is sometimes unnecessarily "large" and thus consumes extra computational resource and may degrade the performance. Some theoretical and empirical evidences have also shown that ensembling many of them may be better than ensembling all of them [12][2].

Motivated by this point, this paper will develop a probabilistic ensemble pruning algorithm, which is based on Bayesian framework. By introducing a sparseness-inducing prior over combination weight vector $\mathbf{w}$, many of the posteriors of weights are sharply distributed at zero, leading to pruning those irrelevant learning machines. A left-truncated, non-negative prior will be adopted for w_i since it is reasonable to force the combination weight w_i to be non-negative [2]. Based on the Bayesian framework and expectation-maximization (EM) algorithm, the Maximum a posteriori (MAP) estimation of weights can be obtained. An empirical study on several regression/classification benchmark data sets also shows that our algorithm utilizes far less component learners but performs as well as, or better than, the non-pruned ensemble.

The rest of this paper is organized as follows. The sparseness-inducing prior is introduced in Section 2. Section 3 will present the probabilistic algorithm for regression problems and Section 4 is proposed for classification problems. Experimental results are presented in Section 5. Finally, Section 6 will conclude the paper.

2 Sparseness-inducing and Non-negative Prior

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

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

where $\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_M)^T$ and $N_t(w_i|0, \alpha_i^{-1})$ is a left-truncated Gaussian distribution. This can be formalized in

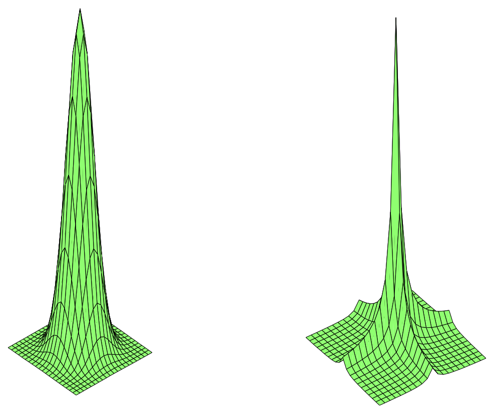

Figure 1. A two dimensional Gaussian prior $p(\mathbf{w}|\alpha)$ and the student-t prior $p(\mathbf{w})$.

Equation (2).

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

To follow the Bayesian inference, hierarchical hyperpriors over α will be defined. With Gamma hyperprior [11],

$$p(\alpha) = \Pi_{i=1}^{M} \Gamma(a)^{-1} b^a \alpha_i^{a-1} e^{-b\alpha_i}, \quad (3)$$

where $\Gamma(a) = \int_0^\infty t^{a-1}e^{-t}dt$ is the gamma function. The complete prior can be obtained by marginalizing with respect to each α_i:

$$\begin{aligned} & p(w_i|a,b) \\ = & \begin{cases} \frac{2b^a\Gamma(a+\frac{1}{2})}{\sqrt{2\pi}\Gamma(a)}(\frac{w_i^2}{2}+b)^{-(a+\frac{1}{2})} & \text{if } w_i \geq 0 \\ 0 & \text{if } w_i < 0 \end{cases}. \quad (4)\end{aligned}$$

Equation (4) shows that the hierarchical prior is equivalent to a truncated student-t prior, which is sharply peaked at zero and more peaky than a Gaussian prior, as illustrated in Figure 1.

3 Ensemble Pruning for Regression Problems

In this section, we will present the model specification of ensemble pruning algorithm for regression problems and detail expectation-maximization [3] procedures.

In the standard regression model, we are given a data set of input-target training pairs $\{\mathbf{x}_n, t_n\}_{n=1}^N$, where t_n is a scalar. To follow the standard probabilistic formulation, we assume the ensemble output is corrupted by an i.i.d. additive Gaussian noise $\epsilon_n = N(0,\sigma^2)$ with its mean zero and variance σ^2:

$$t_n = \sum_{i=1}^{M} w_i f_i(x_n) + \epsilon_n, \quad (5)$$

where $f_i(x_n)$, $i = 1\cdots M$ is the output of ensemble member f_i at point x_n. According to Equation (5), the true value t_n is distributed as a Gaussian distribution with mean $\sum_{i=1}^{M} w_i f_i(x_n)$ and variance σ^2. Based on the assumption of independence of t_n, the likelihood can be represented by:

$$p(\mathbf{t}|\mathbf{w},\sigma^2) = (2\pi\sigma^2)^{-N/2}\exp\{-\frac{1}{2\sigma^2}\|\mathbf{t}-\mathbf{F}\mathbf{w}\|^2\}. \quad (6)$$

where $\mathbf{t} = (t_1\cdots t_N)^T$, $\mathbf{w} = (w_1\cdots w_M)^T$ and $\mathbf{F} = (\mathbf{F}(\mathbf{x_1})^\mathbf{T}, \mathbf{F}(\mathbf{x_2})^\mathbf{T}, \cdots, \mathbf{F}(\mathbf{x_N})^T)^T$ is a $N\times M$ matrix, wherein $\mathbf{F}(\mathbf{x_n}) = (f_1(\mathbf{x_n}), f_2(\mathbf{x_n}), \cdots, f_M(\mathbf{x_n}))$.

Since the prior α over weight vector $\mathbf{w}$ is a truncated Gaussian, the integral in the standard Bayesian inference is intractable. In this paper, expectation-maximization algorithm will be employed to get the MAP estimation of $\mathbf{w}$ and σ^2, where the parameter α is regarded as hidden variable in the EM algorithm. With the definition of hidden variable, the complete log-posterior is obtained in Equation (7) by incorporating Equation (6) and Equation (1).

$$\begin{aligned} & \log p(\mathbf{w},\sigma^2|\mathbf{t},\alpha) \\ \propto & -N\log\sigma^2 - \frac{1}{\sigma^2}\|\mathbf{t}-\mathbf{F}\mathbf{w}\|^2 - \mathbf{w}^T A\mathbf{w}, \quad (7)\end{aligned}$$

where A is a diagonal matrix: $A = diag(\alpha_1,\alpha_2,\cdots,\alpha_M)$.

Expectation-step: After obtaining the log-posterior, the expectation step [1] can be done in the following equation:

$$\begin{aligned} & Q(\mathbf{w},\sigma^2|\mathbf{w}^{old},(\sigma^2)^{old}) \\ = & E_\alpha[\log p(\mathbf{w},\sigma^2|\mathbf{t},\alpha)|\mathbf{t},\mathbf{w}^{old},(\sigma^2)^{old}] \quad (8) \\ = & -N\log\sigma^2 - \frac{1}{\sigma^2}\|\mathbf{t}-\mathbf{F}\mathbf{w}\|^2 - \mathbf{w}^T E_\alpha[A|\mathbf{t},\mathbf{w}^{old},(\sigma^2)^{old}]\mathbf{w},\end{aligned}$$

where the expectation is with respect to the hidden value α. The computation the expectation: $E_\alpha[A|\mathbf{t},\mathbf{w}^{old},(\sigma^2)^{old}]$ which can be decomposed to a diagonal matrix Ω with its elements as $\Omega = diag(E_{\alpha_i}[\alpha_i|\mathbf{t},\mathbf{w}^{old},(\sigma^2)^{old}])$, since A is a diagonal matrix: $A = diag(\alpha_1,\alpha_2,\cdots,\alpha_M)$.

$$\begin{aligned} & \bar{\alpha}_i = E_{\alpha_i}[\alpha_i|\mathbf{t},\mathbf{w}^{old},(\sigma^2)^{old}] \\ = & \frac{\int_0^\infty \alpha_i\cdot p(w_i|\alpha_i)p(\alpha_i)d\alpha_i}{\int_0^\infty p(w_i|\alpha_i)p(\alpha_i)d\alpha_i} = \frac{a+1/2}{w_i^2+b}. \quad (9)\end{aligned}$$

Based on Equation (9), the Q function is reorganized as follows:

$$\begin{aligned} & Q(\mathbf{w},\sigma^2|\mathbf{w}^{old},(\sigma^2)^{old}) \\ = & -N\log\sigma^2 - \frac{1}{\sigma^2}\|\mathbf{t}-\mathbf{F}\mathbf{w}\|^2 - \mathbf{w}^T\Omega\mathbf{w}. \quad (10)\end{aligned}$$

Maximization-step: In this step, the optimal values of $\mathbf{w}$ and σ^2 can be given by analyzing the derivative of Equation (10). By setting the derivatives to zero, we can get the update $(\sigma^2)^{new}$:

$$(\sigma^2)^{new} = \frac{\|\mathbf{t}-\mathbf{F}\mathbf{w}\|^2}{N}, \quad (11)$$

and $\mathbf{w}^{new}$:

$$\mathbf{w}^{new} = (\mathbf{F}^T\mathbf{F} + (\sigma^2)^{new}\Omega)^{-1}\mathbf{F}^T\mathbf{t}. \qquad (12)$$

In order to obtain a parameter-free model, the parameters a, b will be set to zero. However, in this situation, the evaluation of $\bar{\alpha}_i (= 1/(2w_i^2))$ is unstable when w_i approaches zero and a minor modification [4] is adopted on Equation (12).

$$\mathbf{w}^{new} = \mathbf{M}(\mathbf{M}\mathbf{F}^T\mathbf{F}\mathbf{M} + (\sigma^2)^{new}\mathbf{I})^{-1}\mathbf{M}\mathbf{F}^T\mathbf{t}, \qquad (13)$$

where the diagonal elements in the diagonal matrix $\mathbf{M} = diag(m_1, m_2, \cdots, m_M)$ are

$$m_i = (\bar{\alpha}_i)^{-1/2} = \begin{cases} \sqrt{2}w_i & \text{if } w_i \geq 0 \\ 0 & \text{if } w_i < 0 \end{cases}. \qquad (14)$$

4 Ensemble Pruning for Classification Problems

In the standard classification model setting, we are given a data set of input-target training pairs $\{\mathbf{x}_n, y_n\}_{n=1}^N$, considering two-class classification only, i.e. $y_i \in \{-1, +1\}$. Probit link function will be used to allow a steep and smooth transition between two classes.

$$\Phi(x) = \int_{-\infty}^{x} N(t|0,1)dt, \qquad (15)$$

where $\Phi(x)$ is the Gaussian cumulative distribution function. After incorporating the probit link function, the ensemble model becomes:

$$l(\mathbf{x}; \mathbf{w}) = \Phi(\sum_{i=1}^{M} w_i f_i(\mathbf{x})) = \Phi(\mathbf{F}(\mathbf{x})\mathbf{w}). \qquad (16)$$

where l is the probabilistic output, which is bounded by the interval $l \in [0, 1]$, we can map l to $[-1, +1]$ by $y = 2l - 1$.

We follow the standard probabilistic formulation and assume that $\sum_{i=1}^{M} w_i f_i(\mathbf{x}_n)$ is corrupted with an additive random noise ϵ_n, where $\epsilon_n \sim N(0, 1)$. According to the probit link model, if $h(\mathbf{x}_n) = \mathbf{F}(\mathbf{x_n})\mathbf{w} + \epsilon_n \geq 0$, y_n will be set to 1 and if $h(\mathbf{x}_n) = \mathbf{F}(\mathbf{x_n})\mathbf{w} + \epsilon_n < 0$, $y_n = -1$, otherwise. We can reconstruct the probit mode by the random noise ϵ_n.

$$\begin{aligned} p(y_n &= 1|\mathbf{x_n}, \mathbf{w}) \\ &= p(\mathbf{F}(\mathbf{x_n})\mathbf{w} + \epsilon_n \geq 0) = \Phi(\mathbf{F}(\mathbf{x_n})\mathbf{w}). \end{aligned} \qquad (17)$$

From Equation (17), $h(\mathbf{x}_n)$ is a hidden variable because ϵ_n is an unobserved variable. If $h(\mathbf{x}_n)$ is known, the likelihood of $\mathbf{w}$ can be given by standard probabilistic formulation: $p(h(\mathbf{x}_n)|\mathbf{w}) = N(h(\mathbf{x}_n)|\mathbf{F}(\mathbf{x_n})\mathbf{w}, 1)$. Take the $N \times M$ matrix form $\mathbf{F} = (\mathbf{F}(\mathbf{x_1})^\mathbf{T}, \mathbf{F}(\mathbf{x_2})^\mathbf{T}, \cdots, \mathbf{F}(\mathbf{x_N})^T)^T$, wherein $\mathbf{F}(\mathbf{x_n}) = (f_1(\mathbf{x_n}), f_2(\mathbf{x_n}), \cdots, f_M(\mathbf{x_n}))$, and vector form for $H = (h(\mathbf{x}_1), h(\mathbf{x}_2), \cdots, h(\mathbf{x}_N))^T$, we obtain

$$p(H|\mathbf{w}) = (2\pi)^{-N/2}\exp\{-\frac{1}{2}\|H - \mathbf{F}\mathbf{w}\|^2\}. \qquad (18)$$

In order to obtain the complete log-posterior of $\mathbf{w}$, we need two hidden variables: $H = (h(\mathbf{x}_1), h(\mathbf{x}_2), \cdots, h(\mathbf{x}_N))^T$ and $\alpha = (\alpha_1, \alpha_2, \cdots, \alpha_M)^T$.

With the definitions of hidden variables, the complete log-posterior is obtained in Equation (19).

$$\log p(\mathbf{w}|\mathbf{y}, H, \alpha) \propto \mathbf{w}^T\mathbf{F}^T(2H - \mathbf{F}\mathbf{w}) - \mathbf{w}^T A\mathbf{w}, \quad (19)$$

Expectation-step: After obtaining the log-posterior, the expectation step [1] can be done in the following equation:

$$\begin{aligned} Q(\mathbf{w}|\mathbf{w}^{old}) &= 2\mathbf{w}^T\mathbf{F}^T E[H|y, \mathbf{w}^{old}] - \mathbf{w}^T\mathbf{F}^T\mathbf{F}\mathbf{w} \\ &- \mathbf{w}^T E[A|y, \mathbf{w}^{old}]\mathbf{w}, \end{aligned} \qquad (20)$$

where the expectation is with respect to the hidden variables: H, α. The computation of Q function reduces to compute the expectations $E[H|y, \mathbf{w}^{old}]$ and $E[A|y, \mathbf{w}^{old}]$.

$$\begin{aligned} \bar{h}_n &= E[h(\mathbf{x}_n)|y_n, \mathbf{w}^{old}] \\ &= \begin{cases} z_n\Phi(z_n) + N(z_n|0,1) & \text{if } y_n = +1 \\ z_n\Phi(-z_n) - N(z_n|0,1) & \text{if } y_n = -1 \end{cases} \end{aligned} \qquad (21)$$

where $z_n = \mathbf{F}(\mathbf{x}_n)\mathbf{w}$.

Note that y_n in Equation (21) is to restrict the integral bound: when $y_n = +1$, $p(h(\mathbf{x}_n)|y_n, \mathbf{w}^{old})$ is distributed as a left-truncated Gaussian from zero to infinity with mean $\mathbf{F}(\mathbf{x}_n)\mathbf{w}$ and when $y_n = -1$, $p(h(\mathbf{x}_n)|y_n, \mathbf{w}^{old})$ is distributed as a right-truncated Gaussian from negative infinity to zero with mean $\mathbf{F}(\mathbf{x}_n)\mathbf{w}$.

Since A is a diagonal matrix: $A = diag(\alpha_1, \alpha_2, \cdots, \alpha_M)$, the expectation $E[A|y, \mathbf{w}^{old}, b^{old}]$ can be decomposed to a diagonal matrix Ω with its elements as $\Omega = diag(E[\alpha_i|y, \mathbf{w}^{old}])$.

$$\begin{aligned} \bar{\alpha}_i &= E[\alpha_i|y, \mathbf{w}^{old}] \\ &= \frac{\int_0^\infty \alpha_i \cdot p(w_i|\alpha_i)p(\alpha_i)d\alpha_i}{\int_0^\infty p(w_i|\alpha_i)p(\alpha_i)d\alpha_i} = \frac{a + 1/2}{w_i^2 + b}, \end{aligned} \qquad (22)$$

Based on Equations (21) and (22), the Q function is organized as follows:

$$Q(\mathbf{w}|\mathbf{w}^{old}) = 2\mathbf{w}^T\mathbf{F}^T\bar{H} - \mathbf{w}^T\mathbf{F}^T\mathbf{F}\mathbf{w} - \mathbf{w}^T\Omega\mathbf{w} \quad (23)$$

where $\bar{H}$ is a vector or $\bar{h}_n$: $\bar{H} = (\bar{h}_1, \bar{h}_2, \cdots, \bar{h}_N)^T$.

Maximization-step: In this step, the gradient of $\mathbf{w}$ can be given by analyzing the derivative of Equation (23). By setting the derivative to zero, we get the update weight vectors:

$$\mathbf{w}^{new} = (\mathbf{F}^T\mathbf{F} + \Omega)^{-1}F^T\bar{H} \qquad (24)$$

Table 1. Average Test MSE, Standard Deviation and their normalized mean for four Benchmark Data sets based on 30 runs.

	SincG	SincU	Fried.	Hous.	Mean
Non-Pruned	0.0028	0.0183	7.4546	27.8699	**1**
S.D.	0.0008	0.0056	0.7663	8.5879	**1**
Pruned	0.0015	0.0173	5.7161	17.5307	**0.72**
S.D.	0.0006	0.0058	0.5676	6.3944	**0.82**

Table 2. Size of Training Set, Size of Pruned Ensemble and Computational Time of Our Algorithm.

	SincG	SincU	Friedman	House
Training Size	100	100	200	400
Size	2.7±0.8	2.5±0.6	4.7±1.4	4.7±1.2
Time (s)	0.016	0.017	0.031	0.03

As same as regression problem, a minor modification [4] is adopted on Equation (24) to allow a stable numerical computation in practice.

$$\mathbf{w}^{new} = \mathbf{M}(\mathbf{M}\mathbf{F}^T\mathbf{F}\mathbf{M} + \mathbf{I})^{-1}\mathbf{M}\mathbf{F}^T\bar{H}, \qquad (25)$$

where the diagonal elements in the diagonal matrix $\mathbf{M} = diag(m_1, m_2, \cdots, m_M)$ are same as Equation 14.

5 Experimental Results

This section will present the experimental results of our algorithm for regression problems and classification problems, respectively.

Neural networks with single hidden layer, which has 5 hidden units, are employed as ensemble members. The training set for these neural network is generated via bootstrap sampling from the training set. An ensemble is composed of 25 neural networks. In the training process, the generalization error is estimated by a validation set, which is bootstrap sampled from the training set, every five epoches. Once the estimated generalization error does not change or tends to increases, the training process will be terminated to avoid overfitting. The average weights, i.e. the weights of Bagging, will be adopted as the initialized weight vector $\mathbf{w}$ for expectation-maximization algorithm.

5.1 Results for Regression Problems

The four benchmark regression data sets will be employed in our paper. The first two data sets are $sinc = \sin(x)/x$ with different noises, where 100 x's are equally sampled in the interval $[-10, 10]$. The first data set is *Sinc-G* with zero-mean Gaussian noise and standard deviation 0.1; The second is *Sinc-U* with a uniform noise in $[-0.1, 0.1]$. In both cases, the test sets are consisted of 1000 noise-free data points. The third problem is the synthetic *Friedman* function [5]. For the data set, 200 training points and 1000 noise-free test points are generated randomly. The last data set is *Boston Housing* from UCI Machine Learning Repository [9]. For this data set, 400 training points and the remaining 106 test points are sampled randomly.

For every data set, we run thirty times and record the average mean squared error (MSE) and the standard deviation (S.D.) on test set for the non-pruned ensemble and the pruned ensemble. By way of summary, the our algorithm statistics are also normalized by those of the non-pruned ensemble and the overall average is displayed in Table 1. For our algorithm, the measure of "sparseness", i.e. the average number of neural networks in the ensemble and the standard deviation, has also been recorded.

Table 1 shows the performance of non-pruned ensemble versus pruned ensemble based on thirty independent runs. We also show the number of selected ensemble members and running time versus the size of training set for these four data sets in Table 2. The performance of pruned ensemble on these four benchmark problems is far better than non-pruned ensemble in terms of generalization ability and sparsity. Pruned ensemble achieves better performance by employing only a few of the available neural networks (25 in total).

5.2 Sparse Classification Ensemble

In this subsection, we will use four benchmark classification data sets: waveform, diabetics, titanic and credit card, to compare the performance of pruned ensemble with non-pruned ensemble. All of the data sets are obtained from UCI Machine Learning Repository [9].

The training set and test set are generated randomly from these four data sets and the number of training and test points are 1000, 4000 for waveform, 400, 368 for diabetics, 500, 1701 for titanic and 400, 390 for credit card. For every data set, we run thirty times and record the average error rate and the standard deviation on test set of the two ensemble: pruned ensemble and non-pruned ensemble. By way of summary, the our algorithm statistics are also normalized by those of the non-pruned ensemble and the overall average is displayed in Table 3. For our algorithm, the

Table 3. Average Test error, Standard Deviation and their normalized mean for four Benchmark Data sets based on 30 runs.

	wave	diab.	Titanic	Card	Mean
NonPruned	0.1013	0.2305	0.2172	0.1416	**1**
S.D.	0.0078	0.0183	0.0085	0.0237	**1**
Pruned	0.0994	0.2304	0.2170	0.1443	**1.00**
S.D.	0.0072	0.0163	0.0087	0.0239	**0.96**

Table 4. Size of Training Set, Size of Pruned Ensemble and Computational Time of Our Algorithm.

	wave	diabe.	Titanic	Card
Training Size	1000	400	500	400
Size	6.4±0.8	6.4±0.8	5.3±3.1	4.2±1.0
Time (s)	0.266	0.172	0.203	0.109

measure of "sparseness", i.e. the average number of neural networks in the ensemble and the standard deviation, also has been recorded.

Table 3 shows the performance comparison of non-pruned ensemble versus our algorithm based on thirty independent runs and we also show the number of selected ensemble members and running time versus the size of training set for these four data sets in Table 4. The error rate of our algorithm is comparable with non-pruned ensemble but employs fewer component neural networks. The ensemble pruning algorithm can achieve the sparseness in ensemble without hurting the generalization ability. It provides a way to reduce the computational complexity and make the ensemble more compact.

6 Conclusion

In this paper, a probabilistic ensemble pruning algorithm has been proposed in order to get a set of sparse combination weights to prune the ensemble by introducing a left-truncated, non-negative, Gaussian prior over every combination weight. Our algorithm 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 performs as well as, or better than, the non-pruned ensemble; b) The highly spare model is obtained by the sparseness-inducing prior and behaves optimally compact; c) No parameters to tune.

However, the present algorithm is not applicable to large ensemble (e.g. roughly $M > 1000$) because the EM update rules involve an inverse operation of matrix, which requires $O(M^3)$ complexity, where M is the number of selected ensemble members. Although the pruning process in EM algorithm will reduce M to a manageable size in most problems, M may be very large at initialization when the non-pruned ensemble is very large and this will cost a lot of training times.

Acknowledgment

This work is partially supported by a Dorothy Hodgkin Postgraduate Scholarship to the first author and an AWM grant to the third author.

References

[1] J. Bilmes. A gentle tutorial of the em algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. Technical report, University of California at Berkeley, 1997.

[2] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[3] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society Series B*, 39(1):1–38, 1977.

[4] M. A. T. Figueiredo. Adaptive sparseness for supervised learning. *IEEE Transactions Pattern Analysis and Machine Intellgence*, 25(9):1150–1159, 2003.

[5] J. H. Friedman. Multivariate adaptive regression splines. *he Annals of Statistics*, 19(1):1–141, 1991.

[6] L. K. Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990.

[7] M. M. Islam, X. Yao, and K. Murase. A constructive algorithm for training cooperative neural network ensembles. *IEEE Transaction on Neural Networks*, 14(4):820–834, 2003.

[8] Y. Liu and X. Yao. Ensemble learning via negative correlation. *Neural Networks*, 12(10):1399–1404, 1999.

[9] D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz. UCI repository of machine learning databases, 1998.

[10] R. E. Schapire. A brief introduction to boosting. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 1401–1406, 1999.

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

[12] Z. Zhou, J. Wu, and W. Tang. Ensembling neural networks: many could be better than all. *Artificial Intelligence*, 137(1-2):239–263, 2002.