[8] W. G. Teich and M. Seidl, "Code division multiple access communications: Multiuser detection based on a recurrent neural network structure," in *Proc. IEEE 4th ISSSTA*, vol. 3. Sep. 1996, pp. 979–984.
[9] M. Mostafa, W. G. Teich, and J. Lindner, "Global vs. local stability for recurrent neural networks as vector equalizer," in *Proc. ICSPCS*, Dec. 2011, pp. 1–5.
[10] M. Yoshida and T. Mori, "Global stability analysis for complex-valued recurrent neural networks and its application to convex optimization problems," in *Complex-Valued Neural Networks: Utilizing High-Dimensional Parameters*, T. Nitta, Ed. Hershey, PA, USA: IGI Global, 2009, ch. 5, pp. 104–114.
[11] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Nat. Acad. Sci. USA*, vol. 79, no. 8, pp. 2554–2558, Apr. 1982.
[12] E. Goles, E. Chacc, F. Fogelman-Soulié, and D. Pellegrin, "Decreasing energy functions as a tool for studying threshold functions," *Discrete Appl. Math.*, vol. 12, no. 3, pp. 261–277, 1985.
[13] F. Fogelman-Soulié and E. Goles, "Knowledge representation by automata networks," in *Computers and Computing* (Études et Recherches en Informatique), P. Chenin, C. D. Crescenzo, and F. Robert, Eds. Chichester, U.K.: Wiley, 1986, pp. 175–180.
[14] F. Fogelman-Soulié, C. Mejia, E. Goles, and S. Martinez, "Energy functions in neural networks with continuous local functions," *Complex Syst.*, vol. 3, no. 3, pp. 269–293, 1989.
[15] C. M. Marcus and R. M. Westervelt, "Dynamics of iterated-map neural networks," *Phys. Rev. A*, vol. 40, pp. 501–504, Jul. 1989.
[16] J. Si and A. N. Michel, "Analysis and synthesis of a class of discrete-time neural networks with multilevel threshold neurons," *IEEE Trans. Neural Netw.*, vol. 6, no. 1, pp. 105–116, Jan. 1995.
[17] G. Tanaka and K. Aihara, "Complex-valued multistate associative memory with nonlinear multilevel functions for gray-level image reconstruction," *IEEE Trans. Neural Netw.*, vol. 20, no. 9, pp. 1463–1473, Sep. 2009.
[18] Z. Yi and K. K. Tan, "Multistability of discrete-time recurrent neural networks with unsaturating piecewise linear activation functions," *IEEE Trans. Neural Netw.*, vol. 15, no. 2, pp. 329–336, Mar. 2004.
[19] A. Hirose, "Continuous complex-valued back-propagation learning," *Electron. Lett.*, vol. 28, no. 20, pp. 1854–1855, Sep. 1992.
[20] A. Hirose, "Nature of complex number and complex-valued neural networks," *Frontiers Electr. Electron. Eng. China*, vol. 6, no. 1, pp. 171–180, Mar. 2011.
[21] M. Agu, K. Yamanaka, and H. Takahashi, "A local property of the phasor model of neural networks," *IEICE Trans. Inf. Syst. E, Ser. D*, vol. 79, no. 8, pp. 1209–1211, Aug. 1996.
[22] S. Jankowski, A. Lozowski, and J. M. Zurada, "Complex-valued multistate neural associative memory," *IEEE Trans. Neural Netw.*, vol. 7, no. 6, pp. 1491–1496, Nov. 1996.
[23] D. L. Lee and W. J. Wang, "A multivalued bidirectional associative memory operating on a complex domain," *Neural Netw.*, vol. 11, no. 9, pp. 1623–1635, 1998.
[24] Y. Kuroe, N. Hashimoto, and T. Mori, "Qualitive analysis of a self-correlation type complex-valued associative memories," *Nonlinaer Anal.*, vol. 47, no. 9, pp. 5795–5806, 2001.
[25] W. Zhou, A. Lozowski, and J. Zurada, "Discrete-time recurrent neural networks with complex-valued linear threshold neurons," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 56, no. 8, pp. 669–673, Aug. 2009.
[26] C. Sgraja, A. Engelhart, W. G. Teich, and J. Lindner, "Equalization with recurrent neural networks for complex-valued modulation schemes," in *Proc. 3rd Workshop Kommunikationstechnik*, Schloss Reisensburg, Germany, Jul. 1999, pp. 7–12.
[27] M. Spivak, *Calculus on Mainfolds: A Modern Approach to Classical Theorems of Advanced Calculus*. Reading, MA, USA: Addison-Wesley, 1965, ch. 2, pp. 35–37.
[28] D. G. Zill, *A First Course in Differential Equations with Modeling Applications*. Stamford, CT, USA: Cengang Learning, 2009, ch. 9, pp. 340–345.
[29] Y. Kuroe, N. Hashimoto, and T. Mori, "On energy function for complex-valued neural networks and its applications," in *Proc. 9th ICONIP*, vol. 3. Nov. 2002, pp. 1079–1083.
[30] W. Zhao, W. Lin, R. Liu, and J. Ruan, "Asymptotical stability in discrete-time neural networks," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 49, no. 10, pp. 1516–1520, Oct. 2002.
[31] J. M. Zurada, I. Cloete, and W. van der Poel, "Generalized Hopfield networks for associative memories with multi-valued stable states," *Neurocomputing*, vol. 13, nos. 2–4, pp. 135–149, 1996.
[32] G. Tanaka and K. Aihara, "Complex-valued multistate associative memory with nonlinear multilevel functions for gray-level image reconstruction," *IEEE Trans. Neural Netw.*, vol. 20, no. 9, pp. 1463–1473, Sep. 2009.
[33] M. Mostafa, W. G. Teich, and J. Lindner, "Stability analysis of recurrent neural networks with time-varying activation functions," in *Proc. Joint 3rd INDS, 16th ISTET*, Klagenfurt, Austria, Jul. 2011, pp. 239–244.
[34] P. K. Sahoo and T. Riedel, *Mean Value Theorems and Functional Equations*. Singapore: World Scientific, 1998, ch. 4, pp. 127–130.

Sparse Bayesian Extreme Learning Machine for Multi-classification

Jiahua Luo, Chi-Man Vong, and Pak-Kin Wong

***Abstract*— Extreme learning machine (ELM) has become a popular topic in machine learning in recent years. ELM is a new kind of single-hidden layer feedforward neural network with an extremely low computational cost. ELM, however, has two evident drawbacks: 1) the output weights solved by Moore–Penrose generalized inverse is a least squares minimization issue, which easily suffers from overfitting and 2) the accuracy of ELM is drastically sensitive to the number of hidden neurons so that a large model is usually generated. This brief presents a sparse Bayesian approach for learning the output weights of ELM in classification. The new model, called Sparse Bayesian ELM (SBELM), can resolve these two drawbacks by estimating the marginal likelihood of network outputs and automatically pruning most of the redundant hidden neurons during learning phase, which results in an accurate and compact model. The proposed SBELM is evaluated on wide types of benchmark classification problems, which verifies that the accuracy of SBELM model is relatively insensitive to the number of hidden neurons; and hence a much more compact model is always produced as compared with other state-of-the-art neural network classifiers.**

***Index Terms*— Bayesian learning, classification, extreme learning machine (ELM), sparsity.**

I. Introduction

Extreme learning machine (ELM) was proposed in [1] and [12], which has become a popular research topic in machine learning in recent years [11]. It is proved that single-hidden layer feedforward neural networks with arbitrary

Manuscript received March 1, 2013; accepted September 10, 2013. Date of publication September 30, 2013; date of current version March 10, 2014. This work was supported by the Research Committee, University of Macau under Grant MYRG-075(Y2-L2)-FST12-VCM.

J. Luo and C.-M. Vong are with Department of Computer and Information Science, University of Macau, Taipa, Macao (e-mail: mb15457@umac.mo; cmvong@umac.mo).

P.-K. Wong is with Department of Electromechanical Engineering, University of Macau, Taipa, Macao (e-mail: fstpkw@umac.mo).

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

Digital Object Identifier 10.1109/TNNLS.2013.2281839

hidden parameters and continuous activation function can universally approximate to any continuous function [1]. ELM, however, suffers from two drawbacks: 1) ELM obtains the output weights by solving $\mathbf{Hw} = \mathbf{T}$ using Moore–Penrose generalized inverse $\mathbf{H}^{\dagger}$ [2]. This is a kind of least squares minimization learning, which can easily suffer from overfitting [7]. This problem becomes worse when the training data are not able to represent the characteristic of the learned dataset and 2) the accuracy of ELM is drastically influenced by the number of hidden neurons. Usually, an accurate ELM model can easily contain hundreds or even thousands of hidden neurons for a practical application [1], [3], [14], [24], [25]. Nevertheless, many systems for practical applications, such as onboard control and diagnostic systems, only provide a limited amount of expensive memory that may not be enough to store such ELM model. Therefore, it is essential to shrink the model size while maintaining high model accuracy.

For the first drawback, Deng *et al*. [14] proposed a L2-type regularized ELM to relieve it by penalizing the training error. The proposed model achieves higher generalization than the conventional ELM but usually takes a much higher number of hidden neurons to maintain a comparable accuracy so that a very complicated network is resulted [3], [14]. A kernel ELM with higher generalization was proposed in [3] in which the single hidden layer is generalized and need not be neuron alike. However, the computational cost and required memory of calculating the inverse of regularized square matrix like $(1/\mathrm{c} + \mathbf{\Omega}^T\mathbf{\Omega})^{-1}$ ($\mathbf{\Omega}$: basis matrix, c: regulator) in kernel ELM become a big challenge to deal with large scale problems, though some online learning approaches can be applied for kernel-based models to gain tradeoff for the complexity and accuracy [20], [22]. Thus, improvement of conventional ELM with compact model size is preferred.

Current methods [18], [19], [23], [25] to find the optimal number of hidden neurons are mainly based on an incremental learning methodology aiming to minimization of the training error. Such algorithms mainly focus on regression problems only. In addition, the size of trained model is still very large in some cases. Miche *et al.* [15] proposed an optimally pruned extreme learning machine (OP-ELM), which aims to prune the redundant hidden neurons and maintain high generalization. OP-ELM selects a group of neurons ranked with a least angle squares regression (L1-type regularization) to minimize training error and later improved with a cascade of L1- and L2-types regularization [16] (called Tikhonov Regularized OP-ELM, or TROP-ELM) by the same authors. Although the methodologies in OP-ELM and TROP-ELM outperform the conventional ELM for regression problems, there might be a large amount of hidden neurons selected in the trained model due to the minimization of training error in ranking neurons, resulting in a highly computational cost.

Recently, Bayesian methods are exploited [5] to learn the output weights of ELM to gain higher generalization. Bayesian methods have advantages for machine learning problems that they try to estimate the probability distribution of output values instead of fitting to data, and hence avoiding data overfitting. A nonsparse Bayesian approach for ELM (BELM) [5] for regression was proposed recently, where the output weights are with Gaussian priori distribution conditioned on a shared hyperprior parameter. Meanwhile, the classification task has not been solved in [5] due to a more complicated Bayesian learning process.

In this brief, we propose a sparse Bayesian learning (SBL) approach [7], [8] for ELM (called SBELM) that learns the output weights of ELM classifier, where the parameters of hidden layer are randomly generated as in the conventional ELM. SBELM finds sparse representatives for the output weights assumed with priori distribution instead of adding/deleting hidden neurons. The SBL is a family of Bayes methodology that aims to find a sparse estimate of output weights w_k, $k = 0$ to L (L is the number of hidden neurons), by imposing a hierarchical-independent hyperprior α_k on each w_k in which some w_k's are automatically tuned to zeros during learning phase. Derivatives of SBL mainly vary in the distribution of the hyperprior $p(\alpha_k)$ among which the automatic relevance determination (ARD) prior [8], which commonly assumes Gamma distribution as the learning strategy. A classic of such algorithm is relevance vector machine (RVM) [9], which is the Bayesian approach for support vector machine (SVM) [13]. The proposed SBELM assumes the output weights w_k are conditioned on ARD prior to gain sparsity by tuning some w_k to zeros, leading to pruning the corresponding hidden neurons. Hence, SBELM has the advantages of both SBL (high generalization and sparsity) and ELM (universal approximation and efficient learning speed).

The organization of this brief is as follows. Section II provides a short review on ELM, followed by the introduction of the proposed SBELM for binary classification and analysis of its sparse property in pruning redundant hidden neurons. An extension of SBELM for multicategory classification based on pairwise coupling approach [6], [10] is also described in Section II. The evaluation of the performance of SBELM and discussions on its complexity are conducted in Section III. Finally, we conclude our brief in Section IV.

II. Sparse Bayesian Approach to ELM

A. Short Review of ELM

We briefly review the ELM, whose detail is described elsewhere [1]. Given a set of N training dataset $\mathcal{D} = (\mathbf{x}_i, \mathbf{t}_i), i = 1$ to N with each $\mathbf{x}_i$ is a d-dimensional vector and $\mathbf{t}_i$ is the expectation output. The output function of ELM with L hidden neurons is represented by

$$\mathrm{f}(\mathbf{x}) = \sum_{k=0}^{L} \mathrm{w}_k h_k(\boldsymbol{\theta}_k; \mathbf{x}) = \mathbf{h}(\Theta; \mathbf{x})\mathbf{w} \tag{1}$$

where $\mathbf{h}(\mathbf{\Theta}; \mathbf{x}) = [1, h_1(\boldsymbol{\theta}_1; \mathbf{x}), \ldots, h_L(\boldsymbol{\theta}_L; \mathbf{x})]$ is the hidden feature mapping with respect to input $\mathbf{x}$. $\mathbf{\Theta} = [\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_L]$ are randomly generated parameters of hidden layer and $\mathbf{w}$ is the weight vector of all hidden neurons to an output neuron to be analytically analyzed. $h_k(\cdot)$ is the activation function of hidden layer. Equation (1) can be written as

$$\mathbf{HW} = \mathbf{T} \tag{2}$$

where $\mathbf{H}$ is the $N \times (L+1)$hidden layer feature-mapping matrix, whose elements are as follows:

$$\mathbf{H} = \begin{bmatrix} 1 & h_1(\boldsymbol{\theta}_1;\mathbf{x}_1) & \dots & h_L(\boldsymbol{\theta}_L;\mathbf{x}_1) \\ \vdots & \vdots & \vdots & \vdots \\ 1 & h_1(\boldsymbol{\theta}_1;\mathbf{x}_N) & \dots & h_L(\boldsymbol{\theta}_L;\mathbf{x}_N) \end{bmatrix} \tag{3}$$

the ith row of $\mathbf{H}$ is the hidden layer's output vector for an instance $\mathbf{x}$. Equation (2) is a linear system, which is solved by

$$\mathbf{W} = \mathbf{H}^{\dagger}\mathbf{T}, \qquad \mathbf{H}^{\dagger} = \left(\mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T \tag{4}$$

where $\mathbf{H}^{\dagger}$ is the Moore–Penrose generalized inverse [2] of matrix $\mathbf{H}$. A modified version of (4) is employed by adding a regulator value $1/\lambda$ to the diagonal of $\mathbf{H}^T\mathbf{H}$ in L2-regularized ELM.

B. Bayesian Inference to ELM for Classification

Different from ELM, the output weight $\mathbf{W}$ is learned by Bayesian method, instead of directly calculating $\mathbf{H}^{\dagger}$. The input for SBELM now becomes the hidden layer outputs $\mathbf{H}$. To make it clear, our inputs for Bayesian learning are expressed as $\mathbf{H} \in \mathcal{R}^{N\times(L+1)}$, in which $\mathbf{H} = [\mathbf{h}_1(\Theta;\mathbf{x}_1), \dots, \mathbf{h}_N(\boldsymbol{\Theta};\mathbf{x}_N)]^T$ with $\mathbf{h}_i(\boldsymbol{\Theta};\mathbf{x}_i) = [1, h_1(\boldsymbol{\theta}_1;\mathbf{x}_i), \dots, h_L(\boldsymbol{\theta}_L;\mathbf{x}_i)]$, $i=1\dots N$. We begin with binary classification, in which every training sample can be treated as an independent Bernoulli event, whose probability $p(t|\mathbf{x})$ is Bernoulli distribution. We write the likelihood as

$$p(\mathbf{t}|\mathbf{w},\mathbf{h}) = \prod_{i=1}^{N} \sigma\{\mathcal{Y}(\mathbf{h}_i;\mathbf{w})\}^{t_i}[1-\sigma\{\mathcal{Y}(\mathbf{h}_i;\mathbf{w})\}]^{1-t_i} \tag{5}$$

where $\sigma(\cdot)$ is sigmoid function

$$\sigma[\mathcal{Y}(\mathbf{h};\mathbf{w})] = \frac{1}{1+e^{-\mathcal{Y}(\mathbf{h};\mathbf{w})}} \tag{6}$$

and

$$\mathcal{Y}(\mathbf{h};\mathbf{w}) = \mathbf{h}\mathbf{w} \tag{7}$$

where $\mathbf{t} = (t_1 \dots t_N)^T$ $t_i \in \{0,1\}$ and $\mathbf{w} = (w_0 \dots w_L)^T$. A zero-mean Gaussian prior distribution over each parameter w_k conditions on an ARD hyperparameter α_k [7], [8] is given by

$$(w_k|\alpha_k) = \mathcal{N}\left(w_k|0,\alpha_k^{-1}\right) \tag{8}$$

$$p(\mathbf{w}|\alpha) = \prod_{k=0}^{L}\frac{\alpha_k}{\sqrt{2\pi}}\exp\left(-\frac{\alpha_k w_k^2}{2}\right) \tag{9}$$

where $\boldsymbol{\alpha} = [\alpha_0,\alpha_1,\alpha_2,\dots,\alpha_L]^T$. Importantly, there always exists an independent α_k associated with each w_k. The ARD prior selects the significant hidden neurons by controlling some values of w_k's to zero. Next step is to marginalize the likelihood over $\mathbf{t}$ conditioned on $\boldsymbol{\alpha}$ and $\mathbf{H}$. The values of $\boldsymbol{\alpha}$ are determined by maximizing the marginal likelihood by integrating the weight parameters $\mathbf{w}$

$$p(\mathbf{t}|\boldsymbol{\alpha},\mathbf{H}) = \int p(\mathbf{t}|\mathbf{w},\mathbf{H})\, p(\mathbf{w}|\boldsymbol{\alpha}) d\mathbf{w}. \tag{10}$$

The core learning procedure involves establishing the distribution of $p(\mathbf{t}|\boldsymbol{\alpha},\mathbf{H})$ to determine $\boldsymbol{\alpha}$ by maximizing the marginal likelihood. The integral of (10) is, however, intractable; ARD approximates a Gauss for it with Laplace approximation approach, which is achieved by evaluating a quadratic Taylor expansion of log-posterior function. The mean and covariance of the approximated Gauss are the Laplace's mode and negated inversed second derivative (Hessian matrix), respectively. Thus

$$\begin{aligned} &\ln\{p(\mathbf{t}|\mathbf{w},\mathbf{H})\, p(\mathbf{w}|\boldsymbol{\alpha})\} \\ &= \ln\left\{\prod_{i=1}^{N} y_i^{t_i}(1-y_i)^{1-t_i}\right\} + \ln\left\{\prod_{k=0}^{L}\frac{\alpha_k}{\sqrt{2\pi}}\exp\left(-\frac{\alpha_k w_k^2}{2}\right)\right\} \\ &= \sum_{i=1}^{N}\{t_i \ln y_i + (1-t_i)\ln(1-y_i)\} - \frac{1}{2}\mathbf{w}^{\mathbf{T}}\mathbf{A}\mathbf{w} + \text{const} \end{aligned} \tag{11}$$

where $y_i = \sigma\{\mathcal{Y}(\mathbf{h}_i;\mathbf{w})\}$ and $\mathbf{A} = \text{diag}(\boldsymbol{\alpha})$. Here, we let const$= \sum_{k=0}^{L}(\ln\alpha_k - 1/2\ln 2\pi)$, which is not associated with $\mathbf{w}$. Generally, it is efficient to find the Laplace's mode $\widehat{\boldsymbol{w}}$ using Newton–Raphson method iterative reweighted least squares (IRLS). Given (11), its gradient $\nabla\mathbf{E}$ and Hessian matrix Φ need to be figured out

$$\begin{aligned} \nabla\mathbf{E} &= \nabla_{\mathbf{w}}\ln\{p(\mathbf{t}|\mathbf{w},\mathbf{H})\, p(\mathbf{w}|\boldsymbol{\alpha})\} = \sum_{i=1}^{N}(t_i - y_i)\mathbf{h}_i - \mathbf{A}\mathbf{w} \\ &= \mathbf{H}^{\mathrm{T}}(\mathbf{t}-\mathbf{y}) - \mathbf{A}\mathbf{w} \end{aligned} \tag{12}$$

$$\Phi = \nabla_{\mathbf{w}}\nabla_{\mathbf{w}}\ln\{p(\mathbf{t}|\mathbf{w},\mathbf{H})\, p(\mathbf{w}|\boldsymbol{\alpha})\} = -(\mathbf{H}^{\mathrm{T}}\mathbf{B}\mathbf{H} + \mathbf{A}) \tag{13}$$

where $\mathbf{y} = [y_1, y_2, \dots, y_N]^T$, $\mathbf{B}$ is an $N \times N$ diagonal matrix with element $\beta_i = y_i(1-y_i)$ and $d\sigma/d\mathcal{Y} = \sigma(1-\sigma)$ Therefore, by using IRLS, $\widehat{w}$ is obtained by

$$\mathbf{w}_{\text{new}} = \mathbf{w}_{\text{old}} - \Phi^{-\mathbf{1}}\nabla\mathbf{E} = (\mathbf{H}^{\mathrm{T}}\mathbf{B}\mathbf{H} + \mathbf{A})^{-\mathbf{1}}\mathbf{H}^{\mathrm{T}}\mathbf{B}\hat{\boldsymbol{t}} \tag{14}$$

where $\hat{t} = \mathbf{H}\mathbf{w} + \mathbf{B}^{-1}(\mathbf{t}-\mathbf{y})$. The center $\widehat{\boldsymbol{w}}$ and covariance matrix $\boldsymbol{\Sigma}$ of Gauss distribution over $\mathbf{w}$ by Laplace approximation are

$$\widehat{\boldsymbol{w}} = \boldsymbol{\Sigma}\mathbf{H}^{\mathrm{T}}\mathbf{B}\hat{\boldsymbol{t}} \tag{15}$$

$$\boldsymbol{\Sigma} = (\mathbf{H}^{\mathrm{T}}\mathbf{B}\mathbf{H} + \mathbf{A})^{-\mathbf{1}}. \tag{16}$$

Therefore, we obtain $p(\mathbf{t}|\mathbf{w},\mathbf{H})\, p(\mathbf{w}|\boldsymbol{\alpha}) \propto \mathcal{N}(\widehat{\boldsymbol{w}},\Sigma)$. After gaining Gaussian approximation for $\mathbf{w}$, the integral in (10) becomes tractable. The log marginal likelihood is as follows:

$$\begin{aligned} \mathcal{L}(\alpha) &= \ln p(\mathbf{t}|\alpha,\mathbf{H}) \\ &= -\frac{1}{2}\left[N\ln(2\pi) + \ln\left|\mathbf{B} + \mathbf{H}\mathbf{A}\mathbf{H}^{\mathrm{T}}\right| + (\hat{\boldsymbol{t}})^T(\mathbf{B} + \mathbf{H}\mathbf{A}\mathbf{H}^{\mathrm{T}})^{-1}\hat{\boldsymbol{t}}\right] \\ &= -\frac{1}{2}\left[N\ln(2\pi) + \ln|\mathbf{C}| + \left(\hat{\boldsymbol{t}}\right)^T\mathbf{C}^{-1}\hat{\boldsymbol{t}}\right] \end{aligned} \tag{17}$$

where $\mathbf{C} = \mathbf{B} + \mathbf{H}\mathbf{A}\mathbf{H}^{\mathrm{T}}$. Set the differential of $\mathcal{L}(\boldsymbol{\alpha})$ with respect to $\boldsymbol{\alpha}$ to zero

$$\begin{aligned} \frac{\partial\mathcal{L}(\boldsymbol{\alpha})}{\partial\alpha_k} &= \frac{1}{2\alpha_k} - \frac{1}{2}\Sigma_{kk} - \frac{1}{2}\hat{w}_k^2 = 0 \\ &\Rightarrow \alpha_k^{new} = \frac{1-\alpha_k\Sigma_{kk}}{\hat{w}_k^2}. \end{aligned} \tag{18}$$

By setting w_k and α_k with initial values, $\widehat{\boldsymbol{w}}$ and $\boldsymbol{\Sigma}$ are updated from (15) and (16). Using these two values, $\boldsymbol{\alpha}$ is updated through (18) again, the operation continues to maximize the

marginal likelihood function until reaching the convergence criteria (e.g., when the difference between the maximum α_k (empirically, $\log(\alpha_k)$ is preferred) in two successive iterations is lower than a predefined accuracy, or the maximum number of iterations). After $\widehat{\boldsymbol{w}}$ converges, the probability distribution $p(t_{\text{new}}|\mathbf{x}_{\text{new}}, \widehat{\boldsymbol{w}})$ is predicted by (6) and (7).

C. Property of Sparsity

The mechanism of ARD prior tunes a part of output weights to zeros resulting in pruning their associated hidden neurons. Here, we analyze its sparsity property [17]. Decompose $\mathbf{C}$ based on the term α_k, $\mathcal{L}(\boldsymbol{\alpha})$ is rewritten as

$$\begin{aligned}\mathcal{L}(\boldsymbol{\alpha}) &= -\frac{1}{2}\Bigg[N\ln(2\pi) + \ln|\mathbf{C}_{-k}| + \left(\hat{\boldsymbol{t}}\right)^T \mathbf{C}_{-k}^{-1}\hat{\boldsymbol{t}} \\ &\quad -\ln\alpha_k + \ln\left(\alpha_k + \boldsymbol{\hbar}_k^T\mathbf{C}_{-k}^{-1}\boldsymbol{\hbar}_k\right) - \frac{\left(\boldsymbol{\hbar}_k^T\mathbf{C}_{-k}^{-1}\hat{\boldsymbol{t}}\right)^2}{\alpha_k + \boldsymbol{\hbar}_k^T\mathbf{C}_{-k}^{-1}\boldsymbol{\hbar}_k}\Bigg] \\ &= \mathcal{L}(\boldsymbol{\alpha}_{-k}) + \frac{1}{2}\left[\ln\alpha_k - \ln(\alpha_k + s_k) + \frac{q_k^2}{\alpha_k + s_k}\right] \quad (19)\end{aligned}$$

where $s_k = \boldsymbol{\hbar}_k^T\mathbf{C}_{-k}^{-1}\boldsymbol{\hbar}_k$, $q_k = \boldsymbol{\hbar}_k^T\mathbf{C}_{-k}^{-1}\hat{t}$, and $\boldsymbol{\hbar}_k$ is the kth column of $\mathbf{H}$. $\mathcal{L}(\boldsymbol{\alpha}_{-k})$ is the marginal likelihood with α_k omitted and $\mathbf{C}_{-k}$ represents the matrix $\mathbf{C}$ with term $\boldsymbol{\hbar}_k$ and α_k removed, thus $\mathcal{L}(\boldsymbol{\alpha}_{-k})$ is irrelevant to α_k but other components. By resetting the derivative of $\mathcal{L}(\boldsymbol{\alpha})$ with respect to α_k to zero, the stationary point is obtained

$$\widehat{\alpha}_k = \begin{cases} \frac{s_k^2}{q_k^2 - s_k}, & \text{if } q_k^2 > s_k \\ \infty, & \text{if } q_k^2 \le s_k. \end{cases}$$

In the process of carrying out the iteration of $\boldsymbol{\alpha}$ to maximize the marginal likelihood, some α_k's tend to grow to infinity, which influences the mean $\widehat{\boldsymbol{w}}$ and covariance Σ as

$$\lim_{\alpha_k\to\infty}\Sigma_{kk} = \lim_{\alpha_k\to\infty}\left\{\boldsymbol{\hbar}_k{}^{\mathrm{T}}\beta_i\boldsymbol{\hbar}_k + \alpha_k\right\}^{-1} = 0 \quad (20)$$

$$\lim_{\alpha_k\to\infty}\hat{w}_k = \lim_{\alpha_k\to\infty}\left\{\boldsymbol{\varepsilon}_k\boldsymbol{\hbar}_k^{\mathrm{T}}\beta_i\hat{t}_i\right\} = 0 \quad (21)$$

where $\boldsymbol{\varepsilon}_k$ is the kth row of matrix Σ. (20) and (21) indicates the $\widehat{\mathrm{w}}_k \sim \mathcal{N}(0, 0)$, equivalent to zero. Therefore, the corresponding $h_k(\boldsymbol{\theta}_k; \mathbf{x})$ is pruned, with some that can best contribute to the maximization of likelihood $p(\mathbf{t}|\boldsymbol{\alpha}, \mathbf{H})$ are left, which results in sparsity.

D. Extension to Multiclassification

The proposed SBELM in Section II-B. just solves the binary classification problem. Most of the practical problems are, however, multicategory. It is necessary to extend the SBELM to multiclassification. Here, we select state-of-the-art approach pairwise coupling proposed in [10], which combines all the outputs of every pair of classes to the overall reestimate probabilistic densities of all classes for a new instance. The pairwise coupling approach was also adopted in the well-known toolbox LIBSVM [4]. Here, we briefly introduce it.

Let r_{mn} be the predicted probability of a binary classifier for a new instance denoted as $r_{\text{mn}} = P\{t = m|t = m \text{ or } n, \mathbf{x}\}$, which is the sigmoid output of SBELM and $p_m, m = 1 \ldots K$ be the final probability to be estimated by the pairwise coupling strategy, where K is the label of class. Therefore, the problem of estimation of p_m is equivalent to solve an optimization problem

$$\begin{aligned}&\min\sum_{m=1}^{K}\sum_{n:n\neq m}(r_{mn}p_n - r_{nm}p_m)^2. \\ &\text{Subject to } \sum_{m=1}^{K}p_m = 1 \quad (22)\end{aligned}$$

where $r_{mn} \approx p_m/(p_m + p_n)$. Rewrite the objective function of (22) as

$$\min_p 2\mathrm{P}^T\mathrm{QP} \equiv \min_p \frac{1}{2}\mathrm{P}^T\mathrm{QP} \quad (23)$$

with

$$\mathrm{Q}_{mn} = \begin{cases} \sum_{s:s\neq m} r_{sm}^2, & \text{if } m = n \\ -r_{nm}r_{mn}, & \text{if } m \neq n. \end{cases} \quad (24)$$

Therefore, (23) is a linear equality–constrained convex quadratic programming problem, which can be solved by Lagrange multiplier method. The unique $\mathbf{p}$ is a global minimum if it satisfies the optimal condition. By importing the Lagrange multiplier ξ for the condition of (22), the optimization of (23) is changed to the following form:

$$\begin{bmatrix} Q & \mathbf{e} \\ \mathbf{e}^T & 0 \end{bmatrix}\begin{bmatrix} \mathbf{p} \\ \xi \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix} \quad (25)$$

where $\mathrm{Q}\mathbf{p}$ is a derivative of the right-hand side of (23), $\boldsymbol{e}$ and $\mathbf{0}$ are the $K\times 1$ vector of all ones and zeros, respectively. The linear system (25) provides the solution for (23).

III. Experiments and Evaluation

In this section, we evaluate the performance of SBELM on wide types of datasets from UCI machine learning repository [21]. The 16 evaluated datasets are composed of eight binary class problems and eight multicategory problems whose properties are listed in Table I. This section contains three parts: 1) a full analysis of SBELM's performance and its comparison with ELM are presented; 2) the comparison of SBELM is extended with RVM, SVM TROP-ELM (an improvement of OP-ELM [15] in sparsity and generalization), and BELM to verify its performance in terms of accuracy, model size, and training time. Note that we just simply measure the number of hidden neurons of learned network as the model size in this brief; and 3) its efficiency and complexity are discussed.

A. Setup

All the experiments are carried out via a fivefold cross validation strategy. Their mean accuracy and standard deviation is compared. The experiments are conducted in MATLAB environment running on a 2.93-GHz-CPU with 4-GB RAM PC. Some notes for the experiments are as follows.

1) All the features of each dataset are linearly scaled to $[-1, 1]$, except those whose all the values originally lie in this interval.

TABLE I
PROPERTIES OF TRAINING DATASET

Datasets	Samples	Features	Classes
Breast cancer	683	10	2
Colon	62	2000	2
Diabetes	768	8	2
Geman	1000	24	2
Liver	345	6	2
Mushroom	3000	21	2
Australian	690	14	2
Spect heart	267	22	2
*Balance	625	4	3
*Glass	214	10	6
*Iris	150	4	3
*Segment	2310	19	7
*Vehicle	846	18	4
*Wine	178	13	3
*Satimage	4435	36	6
*Vowel	990	13	11

Datasets with * ahead are multi-category classification problems

2) The hidden output function of SBELM and ELM is sigmoid $G(\mathbf{a}, b, \mathrm{x}) = 1/(1 + \exp(-(\mathbf{a}\mathrm{x} + b)))$, where $\mathbf{a}$, b are the random synapses and bias with uniform distribution within $[-1, 1]$, respectively. In practice, these random parameters may have impact on the performance. Thus, the seed of generating uniformly random $(\mathbf{a}, b)$ is also tuned via cross validation.
3) Fivefold cross validation for the number of hidden neurons L and the seed s of generating uniformly random hidden synapses and bias $[L, s]$ in SBELM and ELM are on points $[10, 30, 50, \ldots, 210] \times [1, 2, 3, \ldots, 10]$ with L incremented with step 20. In experiments, the samples at these 11 hidden neurons and 10 seeds are enough for cross validation to test the performance of ELMs (SBELM, ELM, BELM, and TROP-ELM). Note that each $[L, s]$ is fixed for the network even though the SBELM adopts pairwise coupling strategy to learn the output weights.
4) Each dataset is randomly permutated and split into five folds in advance and kept fixed for all experiments.

For each dataset, those samples are eliminated whose one or more features are lost, and the number of samples of mushroom is cut to 3000 instances.

B. Comparison of SBELM and ELM

Initially, the mean accuracies of SBELM and ELM are compared under different numbers of hidden neurons. For every given L, each dataset is trained on the 10 seeds and the best mean accuracy is chosen for analysis, as shown in Fig. 1. Observed from Fig. 1, the highest accuracy corresponding to the two models in most of the datasets are nearly the same except that the diabetes (binary) and four multicategory datasets balance, glass, segment and satimage in ELM are evidently lower than SBELM, whereas the accuracy series of ELM are with much higher fluctuation because of data overfitting, resulting in poor generalization. On the contrary, the series of SBELM on these 16 datasets are very stable

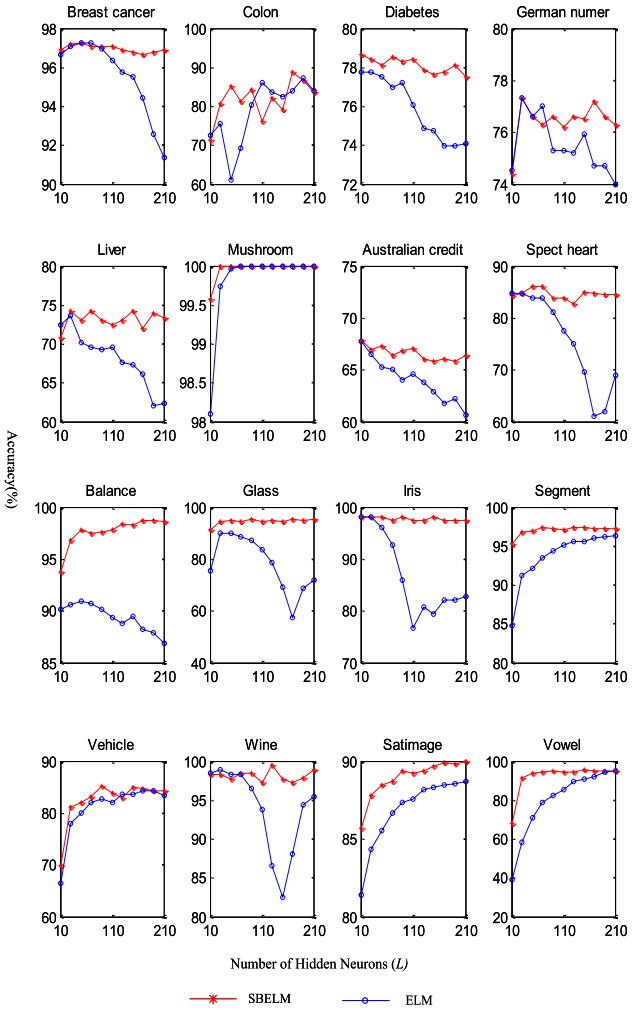

Fig. 1. Variations of accuracies of SBELM and ELM with increment of hidden neurons.

and with much higher accuracies, which denotes the two advantages of SBELM that: 1) it is relatively insensitive to the number of hidden neurons and 2) it tends to reach the best performance at small L, thus the computational cost can be significantly reduced, as shown in Fig. 1. For example, the (near-) best performance of SBELM tested on most of the datasets can be achieved at $L \approx 50$ (the real number of hidden neurons in the pruned model can be much smaller, see Section III-C.) in Fig. 1 and SBELM keeps stable subsequently. However, ELM fails, or needs much larger L, to get close to SBELM, e.g., segment, vehicle, and satimage and vowel on which $L > 200$, resulting in a very complicated network.

C. Comparison With RVM, SVM, TROP-ELM, and BELM

We here extend the comparison of SBELM with four related models RVM, SVM (by LIBSVM toolbox [4]), TROP-ELM) and BELM on these 16 datasets. Since BELM for classification has not been investigated, we develop it for binary classification problems and extend to multiclass case with the same pairwise coupling approach introduced in

TABLE II
COMPARISONS OF SBELM WITH OTHER MODELS

Datasets	SBELM		TROP-ELM		BELM		RVM	SVM
	Accuracy (%)	size	Accuracy (%)	size	Accuracy (%)	size	Accuracy (%)	Accuracy (%)
Breast cancer	97.22 ±1.21	5.6	97.07±0.69	26	97.08±0.87	30	97.37 ±0.83	97.36±0.86
Colon	88.57 ±4.81	7.8	78.10±13.11	7	87.14±7.26	130	79.52±10.36	64.76±4.26
Diabetes	78.66 ±3.58	8	76.97±5.48	71	78.79±1.94	210	78.79 ±1.77	78.52±1.87
Geman n.	77.30 ±2.59	14.4	75.5±3.95	66	76.90±4.17	70	77.50 ±3.66	76.10±4.87
Liver	74.20 ±4.15	8.6	70.72±5.65	89	71.59±2.63	50	73.91 ±4.81	74.20±4.51
Mushroom	100.00±0.0	8.2	99.93±0.09	154	100.0±0.00	30	100.00±0.0	100.00±0.00
Australian	67.83 ±0.12	2.2	67.42±1.97	13	67.97±0.38	10	67.83 ±0.12	68.26±1.32
Spect heart	86.12 ±2.64	6	84.64±2.49	18	84.96±5.27	130	84.25 ±3.26	84.25±2.30
*Balance	98.72 ±0.72	6.7×3	87.21±0.82	56	97.15±1.58	70×3	98.07 ±1.24	100.00±0.00
*Glass	95.26 ±3.58	2.6×15	88.91±2.32	93	95.76±2.74	110×15	93.67 ±6.17	96.26±2.80
*Iris	98.00 ±2.98	2.4×3	96.67±3.33	59	97.33±3.65	10×3	96.67 ±3.33	98.00±2.98
*Segment	97.40±0.55	4.6×21	90.43±0.76	187	97.06±0.50	150×21	97.23 ±0.76	97.45±0.36
*Vehicle	85.17±3.15	11.8×6	70.00±1.48	131	82.73±1.59	70×6	83.66 ±1.87	86.75±1.19
*Wine	99.41 ±1.32	3.3×3	96.58±3.86	84	99.41±1.32	30×3	97.76 ±2.44	99.41±1.32
*Satimage	90.0±0.96	16.0×15	84.28±1.60	170	89.54±1.13	210×15	91.21 ±1.05	92.60±0.60
*Vowel	95.35 ±1.40	6.0×55	66.57±5.48	203	93.74±3.60	210×55	94.75 ±1.05	99.60±0.42

The model sizes (i.e., the number of hidden neurons) of SBELM and BELM are calculated by $L_{remain} \times K(K\text{-}1)/2$, i.e., the number of pairwise classifiers. Note that each pairwise classifier in SBELM and BELM has just ONE output neuron, while TROP-ELM may have multiple output neurons for mutli-category problems.

TABLE III
COMPARISON OF TRAINING TIME (IN SECOND)

Classifiers	Breast C.	Colon	Diabetes	Geman n.	Liver	Mushroom	Australian C.	Spect heart
SVM	4.6E+1	4.9E+1	5.4E+2	3.6E+2	4.5E+1	1.6E+3	1.4E+2	3.6E+1
RVM	3.2E+3	9.4E+1	7.2E+3	8.4E+3	2.8E+3	1.2E+6	3.8E+3	7.4E+2
SBELM	6.1E+1	4.2E+1	7.2E+1	9.9E+1	3.6E+1	1.9E+2	6.3E+1	5.7E+1
BELM	2.2E+1	1.7E+1	2.8E+1	2.4E+1	2.1E+1	9.0E+1	2.1E+1	1.4E+1
ELM	9.9	2.9	1.0E+1	1.3E+1	7.4	2.7E+1	9.8	6.4
TROP-ELM	4.8E+2	5.2E+2	4.9E+2	5.0E+2	4.8E+2	9.9E+2	3.8E+2	4.5E+2

Classifiers	*Balance	*Glass	*Iris	*Segment	*Vehicle	*Wine	*Satimage	*Vowel
SVM	7.2E+1	3.6E+1	1.8E+1	1.2E+3	3.8E+2	1.8E+1	2.6E+3	4.2E+2
RVM	7.1E+3	1.2E+2	2.7E+2	6.6E+4	7.1E+3	1.4E+2	1.9E+5	1.4E+4
SBELM	2.2E+2	6.3E+2	1.1E+2	9.3E+2	4.2E+2	1.3E+2	1.7E+3	2.5E+3
BELM	1.4E+2	1.9E+2	3.9E+1	8.7E+2	2.2E+2	4.2E+1	1.1E+3	1.3E+3
ELM	8.9	5.3	3.0	2.1E+1	1.1E+1	4.1	3.6E+1	1.4E+1
TROP-ELM	4.6E+2	6.8E+2	4.6E+2	1.2E+3	8.5E+2	6.7E+2	1.1E+3	1.2E+3

Section II-D. RVM trains a kernel machine for a dataset and automatically prunes the irrelevant basis elements to gain sparsity. In this brief, RVM adopts the same pairwise coupling algorithm like SBELM to estimate the overall output probability. The hyperparameters $[c, \gamma]$ with regulator c and Gaussian kernel width γ in $\exp(\|x_i - x_j\|^2/\gamma)$ in SVM is conducted on points $[2^{-5}, 2^{-4}, \ldots, 2^{15}] \times [2^{-5}, 2^{-4}, \ldots, 2^{15}]$ and the same Gaussian kernel width range for RVM is also in $[2^{-5}, 2^{-4}, \ldots, 2^{15}]$.

The accuracies of the five classifiers on each dataset are shown in Table II and the final model sizes corresponding to the three ELMs (SBELM, TROP-ELM, and BELM) are listed as well. In terms of accuracy, SBELM is almost identical to SVM. In addition, it is also slightly better than RVM in most of the datasets, some are more evident, such as colon, spect heart, glass, iris, and vehicle and wine. Among the three ELM models, SBELM generally outperforms TROP-ELM in both accuracy and model size. Though the accuracy of BELM is close to SBELM, its model size is tens to hundreds of times larger than that of SBELM, as shown in Table II.

Fig. 2 is plotted showing the number of remaining hidden neurons L_{remain} with $L \in [10, 30, 50, \ldots, 210]$ over the 16 datasets after being pruned by SBELM and TROP-ELM. Note that $L_{\text{remain}}$of a multicategory dataset at a given L is the mean of $K(K-1)/2$ pairwise classifiers in SBELM. As observed from Fig. 2, L_{remain} in SBELM is usually around 10 for all datasets aside from German numer in which L_{remain} tends to reach 50 when L is close to 210, whereas its generalization (Fig. 1) is not significantly influenced. The stable result of L_{remain} verifies that SBELM is relatively insensitive to the initial number of L. On the contrary, L_{remain} in TROP-ELM linearly increases with L, this is caused by the issue that TROP-ELM selects the combination of ranked hidden neurons that minimize the training error.

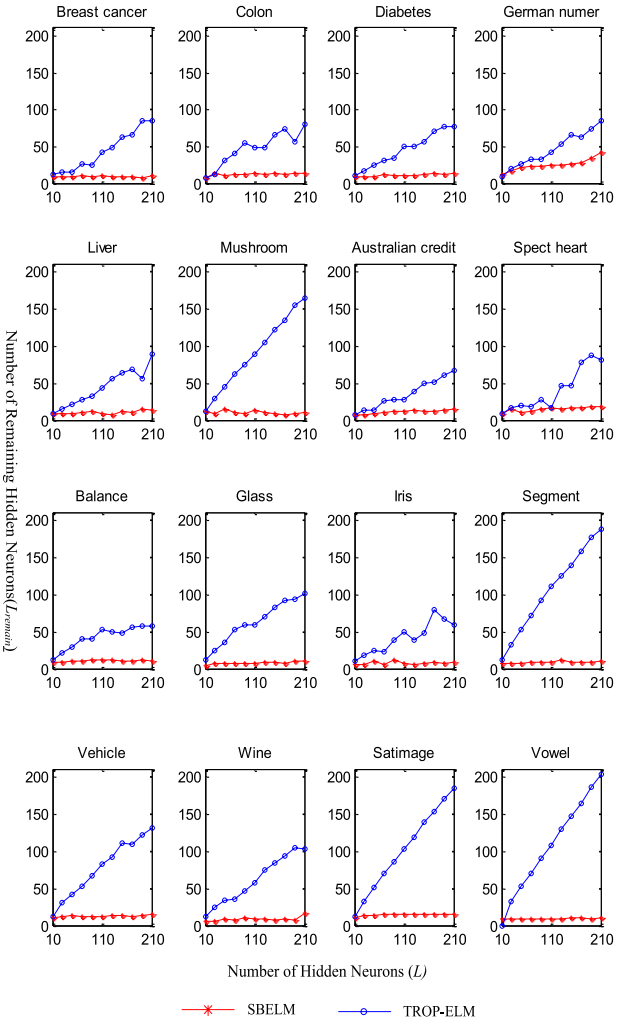

Fig. 2. Variations of number of remaining hidden neurons with increment of hidden neurons in SBELM and TROP-ELM.

Table III lists the training time of the six classifiers on the 16 benchmark problems. From Table III, the training time of SBELM over most of the datasets are in the same order of magnitude as SVM, except the relatively large datasets mushroom, segment and satimage, where SBELM is even faster. This is due to the large computational cost of SVM in dealing with large scale problems. In addition, SBELM is with much faster training time than RVM on almost all datasets, among which some are even two magnitudes faster. Likewise, SBELM is faster than TROP-ELM and in the same magnitude The model sizes (i.e., the number of hidden neurons) of SBELM and BELM are calculated by $L_{\text{remain}} \times K(K-1)/2$, i.e., the number of pairwise classifiers.

Note that each pairwise classifier in SBELM and BELM has just ONE output neuron, while TROP-ELM may have multiple output neurons for mutlicategory problems. as BELM, while SBELM achieves much better performance than TROP-ELM in accuracy and model size, as shown in Table II. From experimental results (Figs. 1 and 2), SBELM tends to be insensitive to the number of hidden neurons and obtains a smaller but very accurate model. Practically, by setting a small initial L as the upper bound, or even directly a specific L, the training time of SBELM can be significantly reduced, while still obtaining an accurate model. Therefore, SBELM provides a learning strategy in comparable accuracy, medium training time, but extremely sparse model size as compared with ELM and other state-of-the-art classifiers.

D. Discussion

As analyzed in Section III-B., although the number of hidden neurons is different, SBELM can still tune the learning process in the new feature space mapped by the parameters with same uniform distribution to select the most representative hidden neurons and prune the redundant ones. In this assumption, these new feature spaces will be with the same shape but with different dimensions, whereas the positions of remaining neurons and values after iteration might be different. Further mathematical proof is, however, required to support this assumption.

In Section III, under the same network structure, SBELM completely outperforms TROP-ELM in generalization. By ranking and selecting the number of hidden neurons for minimizing the training error, TROP-ELM might still contain a large number of hidden neurons (depending on the given upper bound L) in some cases. The pruned model may be very large, as shown in Fig. 2. In addition, SBELM is better than RVM in generalization, time, and model size. Due to a kernel-based learning model, the size of basis matrix in RVM is $N \times (N+1)$. Hence, RVM needs to compute the inverse of Hessian matrix like (16), which is frequently executed during the iteration of tuning w_k to maximize the marginal likelihood. The complexity of computing this Hessian matrix is $O(N^2)$, which hinders RVM to deal with large scale problems, while the complexity of SBELM is $O(L^2)$. From Section III, SBELM usually tends to gain the best performance with $L \approx 50$, which is generally much smaller than N, and also the L_{remain} decreases during iterations, which significantly reduces the execution time in calculating the inverse of Hessian matrix to gain much faster learning speed than RVM in large datasets. Therefore, SBELM is suitable for large scale problems.

IV. Conclusion

In this brief, we present a new Sparse Bayesian approach to ELM. SBELM uses the universal approximation capability of ELM and estimates the output weights by a sparse Bayesian approach, which generates a probabilistic distribution for outputs of uncertain data. Instead of focusing on explicitly adding/deleting hidden neurons in the conventional sparse ELMs, SBELM automatically tunes most of the output weights to zeros with an assumed prior distribution, thus gaining sparsity and achieving very high generalization. In addition, by maximizing the marginal likelihood instead of minimizing training error, SBELM does not suffer from overfitting the training data. From the experimental results over 16 benchmark datasets, the accuracy of SBELM is on par with SVM and even better than those of other compared methods. Nevertheless, SBELM provides a unique advantage over other classifiers: SBELM is relatively insensitive to the

number of hidden neurons so that the (near-)best accuracy can be achieved with a small number of hidden neurons (usually around 50 in the benchmark datasets). Under this property, the training time of SBELM can be greatly reduced by setting a small upper bound of hidden neurons, as verified in the experimental results. In a nutshell, SBELM can produce a compact classification model of high generalization with a relatively fast training time. Therefore, SBELM is suitable to many practical applications where compact model is required.

REFERENCES

[1] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, pp. 489–501, Dec. 2006.

[2] K. S. Banerjee, "Generalized inverse of matrices and its applications," *Technometrics*, vol. 15, pp. 197–202, Apr. 1973.

[3] G. B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Trans. Syst., Man, Cybern., Part B, Cybern.*, vol. 42, no. 2, pp. 513–529, Apr. 2012.

[4] C. C. Chang and C. J. Lin, "LIBSVM: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 27:1–27:27, 2011.

[5] E. Soria-Olivas, J. Gomez-Sanchis, J. D. Martin, J. Vila-Frances, M. Martinez, J. R. Magdalena, and A. J. Serrano, "BELM: Bayesian extreme learning machine," *IEEE Trans. Neural Netw.*, vol. 22, no. 3, pp. 505–509, Mar. 2011.

[6] C. M. Vong, P. K. Wong, and W. F. Ip, "A new framework of simultaneous-fault diagnosis using pairwise probabilistic multi-label classification for time-dependent patterns," *IEEE Trans. Ind. Electron.*, vol. 60, no. 8, pp. 3372–3385, Aug. 2013.

[7] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer-Verlag, 2006, ch. 4.

[8] D. J. C. MacKay, "Bayesian methods for backpropagation networks," *Models Neural Netw. III*, vol. 6, pp. 211–254, Jan. 1996.

[9] M. E. Tipping, "Sparse Bayesian learning and the relevance vector machine," *J. Mach. Learn. Res.*, vol. 1, pp. 211–244, May 2001.

[10] T. F. Wu, C. J. Lin, and R. C. Weng, "Probability estimates for multi-class classification by pairwise coupling," *J. Mach. Learn. Res.*, vol. 5, pp. 975–1005, Jan. 2004.

[11] G. B. Huang, D. H. Wang, and Y. Lan, "Extreme learning machines: A survey," *Int. J. Mach. Learn. Cybern.*, vol. 2, no. 2, pp. 107–122, 2011.

[12] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: A new learning scheme of feedforward neural networks," in *Proc. IEEE Int. Joint Conf. Neural Netw.*, vol. 2. Jul. 2004, pp. 985–990.

[13] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.

[14] W. Deng, Q. Zheng, and L. Chen, "Regularized extreme learning machine," in *Proc. IEEE Symp. Comput. Intell. Data Mining*, Apr. 2009, pp. 389–395.

[15] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, and A. Lendasse, "OP-ELM: Optimally pruned extreme learning machine," *IEEE Trans. Neural Netw.*, vol. 21, no. 1, pp. 158–162, Jan. 2010.

[16] Y. Miche, M. van Heeswijk, P. Bas, O. Simula, and A. Lendasse, "TROP-ELM: A double-regularized ELM using LARS and Tikhonov regularization," *Neurocomputing*, vol. 74, pp. 2413–2421, Sep. 2011.

[17] A. C. Faul and M. Tipping, "Analysis of sparse Bayesian learning," in *Proc. Conf. Neural Inf. Process. Syst.*, 2002, pp. 383–390.

[18] G. R. Feng, G. B. Huang, Q. P. Lin, and R. Gay, "Error minimized extreme learning machine with growth of hidden nodes and incremental learning," *IEEE Trans. Neural Netw.*, vol. 20, no. 8, pp. 1352–1357, Aug. 2009.

[19] R. Zhang, Y. Lan, G. B. Huang, and Y. C. Soh, "Extreme learning machine with adaptive growth of hidden nodes and incremental updating of output weights," *Auto. Intell. Syst.*, LNCS, vol. 6752, pp. 253–262, Jan. 2011.

[20] G. Li, C. Wen, Z. G. Li, A. Zhang, F. Yang, and K. Mao, "Model-based online learning with kernels," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 3, pp. 356–369, Mar. 2013.

[21] (2013). *University of California Irvine Machine Learning Repository* [Online]. Available: http://archive.ics.uci.edu/ml/

[22] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online learning with kernels," in *Proc. Adv. Neural Inf. Process. Syst.*, 2001, pp. 785–792.

[23] G. B. Huang and L. Chen, "Enhanced random search based incremental extreme learning machine," *Neurocomputing*, vol. 71, pp. 3460–3468, Oct. 2008.

[24] G. B. Huang, X. J. Ding, and H. M. Zhou, "Optimization method based extreme learning machine for classification," *Neurocomputing*, vol. 74, pp. 155–163, Dec. 2010.

[25] G. B. Huang, M. B. Li, L. Chen, and C. K. Siew, "Incremental extreme learning machine with fully complex hidden nodes," *Neurocomputing*, vol. 71, pp. 576–583, Jan. 2008.