Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2019) 27: 4256 – 4268

doi:10.3906/elk-1808-75

Sparse Bayesian approach to fast learning network for multiclassification

Zhibiao ZHAO, Haoran LIU*, Chao WU, Bin LIU
School of Information Science and Engineering, Yanshan University, Hebei, China

Received: 08.08.2018 • **Accepted/Published Online:** 13.07.2019 • **Final Version:** 26.11.2019

Abstract: This paper proposes a novel artificial neural network called sparse-Bayesian–based fast learning network (SBFLN). In SBFLN, sparse Bayesian regression is used to train the fast learning network (FLN), which is an improved extreme learning machine (ELM). The training process of SBFLN is to randomly generate the input weights and the hidden layer biases, and then find the probability distribution of other weights by the sparse Bayesian approach. SBFLN calculates the predicted output through Bayes estimator, so it can provide a natural marginal possibility for classification problems and can solve the overfitting problem caused by the least-squares estimation in FLN. In addition, the sparse Bayesian approach can automatically trim most redundant neurons in hidden layer, which makes the network more compact and accurate. To verify the effectiveness of the improvements in this paper, the results of SBFLN are evaluated in 15 benchmark classification problems. The experimental results show that SBFLN is not sensitive to the number of neurons in the hidden layer, and the performance of SBFLN is competitive or superior to some other state-of-the-art algorithms.

Key words: Extreme learning machines, fast learning network, Bayesian learning, sparse Bayesian, multiclassification

1. Introduction

Artificial neural networks (ANNs) have been widely used in industrial, financial, and natural fields due to their ability to obtain potential nonlinear mappings from data [1–3]. Extreme learning machine (ELM) is one of the most popular ANNs with its simple structure and powerful approximation capability [4, 5]. In ELM, the input weights are randomly assigned, and the output weights are calculated by the least squares. This method overcomes the slow learning process of the ANNs and the local minimum problem [6]. However, ELM requires more hidden neurons than traditional neural network learning algorithms in some regression or classification applications, which may result in trained models spending more reaction time on unknown test samples [7]. In addition, due to the random assignment of input weights, the stability and repeatability of ELM are not very good. In [7], Li Guoqiang proposes an improved architecture of ELM called fast learning network (FLN). FLN is a dual parallel architecture consisting of a single hidden layer feedforward neural network and a single-layer linear perceptron. It passes the received input information to the hidden layer and the output layer. Therefore, FLN not only has the nonlinear approximation capability like general ANNs, but also reflects a linear mapping between input and output. This combination allows FLN to achieve better accuracy, generalization performance, and stability with the same hidden neurons [8]. More importantly, FLN inherits the ELM's advantage that it does not require iterative calculations. Similarly with original ELM, FLN transforms network training into solving linear least-squares problems, and then computes the output weights through the Moore-

*Correspondence: haoranliuysu@163.com

Penrose generalized inverse [9], so the training speed is very fast. Due to these advantages, FLN has been successfully applied in the real world problem [10–12]. In recent years, various improved training algorithms have been proposed for ELM to improve the pseudoinverse operation in ELM training [13, 14]. In order to solve the overfitting problem of ELM, an enhanced ridge regression algorithm is proposed in [14] to train ELM (Ridge-ELM). Ridge-ELM introduces the L2 norm penalty of the output weights on the training objective function, which sacrifices a small amount of precision to obtain more reliable generalization ability. Miche [15] proposes an optimally pruned ELM called OP-ELM, which introduces L1 norm penalty into the training objective function and uses the minimum angle regression training method to obtain sparse output weights. Another L1 norm penalty ELM, called Newton linear programming ELM [16], uses the Newton-Armijo training algorithm. Compared with the L2 regularization regression, the L1 regularization regression has a better effect in filtering a small number of abnormal samples, and can shield the irrelevant features in the hidden layer. However, the accuracy of the L1 regularization regression tends to be worse than the L2 regularization regression. In [17] and [18], both L1 and L2 norm penalties, which consider both sparseness and accuracy of the network, are introduced to deal with the linear mapping from the hidden layer to the output layer in ELM. The improved networks are called adaptive elastic ELM (AEELM) and Tikhonov regularization OP-ELM (TROP-ELM), respectively. In contrast, training methods of FLN are relatively backward. Recently, the Bayesian approaches of the neural network models become very intense in recent research and show their suitability in different fields [19]. These algorithms introduce a probability distribution on the network parameters and the predictive outputs. Some related classic algorithms are probabilistic versions of self-organizing maps [19] and the relevance support vector machine (RVM) [20] which is the probabilistic approach for support vector machines. The Bayesian approach has also been introduced into the training of ELM to solve regression and classification problems for higher generalization [21, 22].

This paper proposes a sparse Bayesian learning approach to FLN (SBFLN), which not only has excellent performance with low computational cost in multiclassification, but also finds sparse representation for the output weights. The paper is organized as follows: Section 2 describes the original FLN structure and its learning process. Section 3 explains the sparse Bayesian learning process of FLN and the automatic relevance determination (ARD) process for generating sparse priors. SBFLN performance is evaluated in Section 4 using different benchmarks datasets that are commonly used in machine learning. Section 5 summarizes the conclusions of this paper.

2. Review of fast learning network (FLN)

The FLN is a bidirectional parallel forward neural network based on the least square algorithm. As shown in Figure 1, FLN is formed by adding a direct connection between the input layer and the output layer in a single hidden-layer feed-forward neural network.

Assume that there are N observation samples $\{(x_i, y_i), i = 1, 2, \ldots, N\}$, where $x_i \in \Re^n$ represents the feature vector and $y_i \in \Re^l$ is output vector. If there are m hidden layer neurons in the FLN, then W^{in} is a matrix of $m \times n$. The hidden layer units have a threshold vector of $b = [b_1, b_2, \cdots, b_m]^{\mathrm{T}}$. The weight matrix W^{oh} has a dimension of $l \times m$. The weight matrix W^{oi} is a matrix of $l \times n$. The mathematical model of the output neurons of the fast learning network is:

$$\boldsymbol{Y} = \boldsymbol{W}^{oi}\boldsymbol{X} + \boldsymbol{W}^{oh}\boldsymbol{G} = \begin{bmatrix}\boldsymbol{W}^{oi} & \boldsymbol{W}^{oh}\end{bmatrix}\begin{bmatrix}\boldsymbol{X}\\ \boldsymbol{G}\end{bmatrix} = \boldsymbol{W}\begin{bmatrix}\boldsymbol{X}\\ \boldsymbol{G}\end{bmatrix}, \tag{1}$$

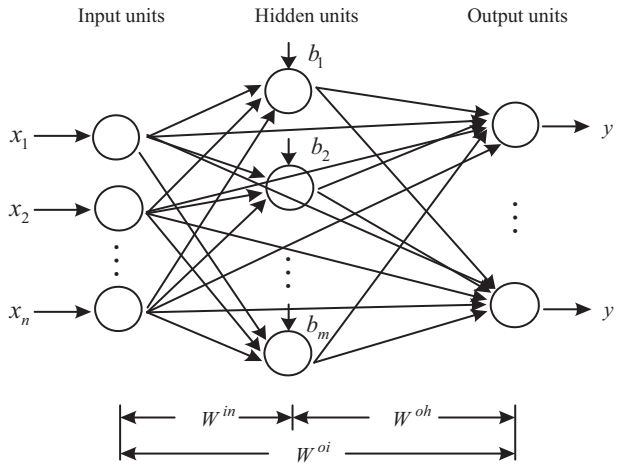

Figure 1. Structure of the fast learning network.

where $\boldsymbol{Y}$ is the desired output matrix, the combination matrix $W = \begin{bmatrix} W^{oi} & W^{oh} \end{bmatrix}$ is called the output weight matrix, $\boldsymbol{G}$ is the FLN hidden layer output matrix:

$$\boldsymbol{G}\left(\boldsymbol{W}_1^{in}, \cdots, \boldsymbol{W}_m^{in}, b_1, \cdots, b_m, \boldsymbol{x}_1, \cdots, \boldsymbol{x}_N\right) \tag{2}$$

$$= \begin{bmatrix} g\left(\boldsymbol{W}_1^{in}\boldsymbol{x}_1 + b_1\right) & \cdots & g\left(\boldsymbol{W}_1^{in}\boldsymbol{x}_N + b_1\right) \\ \vdots & \ddots & \vdots \\ g\left(\boldsymbol{W}_m^{in}\boldsymbol{x}_1 + b_m\right) & \cdots & g\left(\boldsymbol{W}_m^{in}\boldsymbol{x}_N + b_m\right) \end{bmatrix}_{m\times N}. \tag{3}$$

As already mentioned, the input weights $\boldsymbol{W}^{in}$ and hidden neuron thresholds $b = [b_1, b_2, \cdots, b_m]$ of FLN can be randomly assigned values. Thus, the output weights can be obtained with Moore–Penrose's generalized inverse:

$$\hat{\boldsymbol{W}} = Y\begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{G} \end{bmatrix}^{+} = \boldsymbol{Y}\boldsymbol{H}^{+} \tag{4}$$

$$\begin{cases} \boldsymbol{W}^{oi} = \hat{\boldsymbol{W}}\left(1 : l, 1 : n\right) \\ \boldsymbol{W}^{oh} = \hat{\boldsymbol{W}}\left(1 : l, n+1 : (n+m)\right) \end{cases} \tag{5}$$

where $\boldsymbol{H} = \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{G} \end{bmatrix}$, $\boldsymbol{H}^{\dagger} = \left(\boldsymbol{H}^{\mathrm{T}}\boldsymbol{H}\right)^{-1}\boldsymbol{H}$ is a pseudo-inverse matrix.

3. Sparse Bayesian learning for FLN

This paper optimizes output weights $\boldsymbol{W}$ based on Bayesian approach. To make it clear, the inputs for Bayesian learning are expressed as $\boldsymbol{H} \in \Re^{N\times(n+m)}$, in which $\boldsymbol{H} = [\boldsymbol{h}_1, \boldsymbol{h}_2, \cdots, \boldsymbol{h}_N]^{\mathrm{T}}$ and $\boldsymbol{h_i} = \begin{bmatrix} \boldsymbol{x}(i) \\ \boldsymbol{g}(i) \end{bmatrix}$ is the combination matrix of the input and hidden layer output, $\boldsymbol{x}(i) = [x_1(i), \ldots, x_n(i)]^{\mathrm{T}}, \boldsymbol{g}(i) = [g(W_1^{in}x_1(i) + b_1), \ldots, g(W_m^{in}x_m(i) + b_m)]^{\mathrm{T}}, i = 1, \cdots, N$. Suppose that $t_i = y_i + \varepsilon_i = \boldsymbol{W}\boldsymbol{h}_i + \varepsilon_i$, where ε_i is the independent noise.

We first consider binary classification, the purpose of which is to predict the posterior probability of a classes (0 or 1) for a given input x. Every sample can be treated as an independent Bernoulli event. The likelihood function $p(t|W)$ can be analyzed using a Bernoulli distribution as equation (6). The case can then be assigned to the class with the greatest likelihood.

$$p(\boldsymbol{t}|\boldsymbol{W}) = \prod_{k=1}^{N} \sigma\left\{y(\boldsymbol{h}_k;\boldsymbol{W})\right\}^{\boldsymbol{t}_k} \left[1-\sigma\left\{y(\boldsymbol{h}_k;\boldsymbol{W})\right\}\right]^{1-\boldsymbol{t}_k}, \tag{6}$$

where $\boldsymbol{t}$ is defined as the class labels, $\sigma\{y(\boldsymbol{h};\boldsymbol{W})\}$ is the sigmoid function $\frac{1}{1+e^{-y(\boldsymbol{h};\boldsymbol{W})}}$, $y(\boldsymbol{h};\boldsymbol{W}) = \boldsymbol{h}(\boldsymbol{x})\boldsymbol{W}$, and $\{\boldsymbol{x}_k\}_{k=1}^{N}$ is the training input. To calculate output probability distribution, the Bayesian predictive framework is used to infer the posterior distribution over $\boldsymbol{W}$.

$$p(\boldsymbol{W}|\boldsymbol{t},\boldsymbol{\alpha}) = \frac{p(\boldsymbol{t}|\boldsymbol{W})p(\boldsymbol{W}|\boldsymbol{\alpha})}{p(\boldsymbol{t}|\boldsymbol{\alpha})}, \tag{7}$$

where $p(\boldsymbol{W}|\boldsymbol{\alpha})$ is the prior distribution of $\boldsymbol{W}$ defined using the principle of automatic relevance determination (ARD) to set the inverse variance hyperparameter as $\boldsymbol{\alpha}$. A Gaussian prior distribution with a mean of zero is calculated by:

$$p(W_k|\alpha_k) = \mathcal{N}(W_k|0,\alpha_k^{-1}), \tag{8}$$

$$p(\boldsymbol{W}|\boldsymbol{\alpha}) = \prod_{k=1}^{m+n} \frac{\alpha_k}{\sqrt{2\pi}} \exp\left(-\frac{\alpha_k {W_k}^2}{2}\right). \tag{9}$$

The hyperparameter $\boldsymbol{\alpha}$ is implicitly related to $\boldsymbol{W}$. We can use ARD iteration to get the best $\boldsymbol{\alpha}$ and $\boldsymbol{W}$. Note that each α_k independently controls the prior distribution of the associated W_k, which causes the ARD to reduce the weight of the unrelated hidden layer neurons to zero, resulting in a sparse model. This is because some hyperparameters α approach infinity during inference, the posterior distributions of their associated weights are peaked around zero[20].

The following describes the specific process of determining $\boldsymbol{\alpha}$ and $\boldsymbol{W}$. As shown in equation (7), since $p(\boldsymbol{W}|\boldsymbol{t},\boldsymbol{\alpha}) \propto p(\boldsymbol{t}|\boldsymbol{W})p(\boldsymbol{W}|\boldsymbol{\alpha})$, maximizing the logarithm of $p(\boldsymbol{t}|\boldsymbol{W})p(\boldsymbol{W}|\boldsymbol{\alpha})$ is equivalent to maximizing $p(\boldsymbol{W}|\boldsymbol{t},\boldsymbol{\alpha})$. Thus, the objective function can be written as:

$$\begin{aligned} E(\boldsymbol{W}) =& \ln p(\boldsymbol{t}|\boldsymbol{W})p(\boldsymbol{W}|\boldsymbol{\alpha}) \\ =& \ln\left\{\prod_{k=1}^{N} \gamma^{t_k}(1-\gamma)^{1-t_k}\right\} + \ln\left\{\prod_{k=1}^{m+n} \frac{\alpha_k}{\sqrt{2\pi}} \exp\left(-\frac{\alpha_k {W_k}^2}{2}\right)\right\} \\ =& \sum_{k=1}^{N} \left[t_k \ln\gamma_k + (1-t_k)\ln(1-\gamma_k)\right] - \frac{1}{2}\boldsymbol{W}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{W} + C, \end{aligned} \tag{10}$$

where $\gamma_k = \sigma\{y(\boldsymbol{h}_k,\boldsymbol{W})\}$, $\boldsymbol{A} = \mathrm{diag}(\alpha_1,\cdots,\alpha_{m+n})$, $C = \sum_{k=1}^{m+n} \ln\alpha_k - \frac{1}{2}\ln(2\pi)$. Equation (10) is a penalized logistic log-likelihood function and it can be iterated to maximize. The iterative reweighted least squares (IRLS)

algorithm [22] is used to find the $\boldsymbol{W}$ to maximize Equation (10). The first and second derivatives of equation (10) with respect to $\boldsymbol{W}$ can be written as

$$\frac{\partial E(\boldsymbol{W})}{\partial \boldsymbol{W}} = \boldsymbol{H}^T(\boldsymbol{t}-\boldsymbol{\gamma}) - \boldsymbol{A}\boldsymbol{W} \tag{11}$$

$$\frac{\partial^2 E(\boldsymbol{W})}{\partial \boldsymbol{W}\partial \boldsymbol{W}} = -(\boldsymbol{H}^T\boldsymbol{B}\boldsymbol{H} + \boldsymbol{A}) \tag{12}$$

Where $\boldsymbol{B} = \mathrm{diag}(\beta_1, \cdots, \beta_N)$ and $\beta_i = \gamma_i(1-\gamma_i)$. The Laplace approximation approach provides a Gaussian approximation of the posterior weights $\boldsymbol{W}$ with center $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ [23].

$$\boldsymbol{\mu} = \boldsymbol{\Sigma}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{B}\hat{\boldsymbol{t}}, \tag{13}$$

$$\boldsymbol{\Sigma} = (\boldsymbol{H}^{\mathrm{T}}\boldsymbol{B}\boldsymbol{H} + \boldsymbol{A})^{-1}, \tag{14}$$

where $\hat{\boldsymbol{t}} = \boldsymbol{H}\boldsymbol{\mu} + \boldsymbol{B}^{-1}(\boldsymbol{t}-\boldsymbol{\gamma})$, that is, $\boldsymbol{\mu}$ be the iterative update formula for $\boldsymbol{W}$, with a step of $(\frac{\partial^2 E(\boldsymbol{W})}{\partial \boldsymbol{W}\partial \boldsymbol{W}})^{-1}\frac{\partial E(\boldsymbol{W})}{\partial \boldsymbol{W}}$. Therefore, $p(\boldsymbol{t}|\boldsymbol{W})p(\boldsymbol{W}|\boldsymbol{\alpha}) \propto \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. The α_k is determined by maximizing the marginal likelihood $p(\boldsymbol{t}|\boldsymbol{\alpha}, \boldsymbol{H})$

$$p(\boldsymbol{t}|\boldsymbol{\alpha}, \boldsymbol{H}) = \int p(\boldsymbol{t}|\boldsymbol{W}, \boldsymbol{H})p(\boldsymbol{W}|\boldsymbol{\alpha})\mathrm{d}\boldsymbol{W} \tag{15}$$

As we have obtained equation (10), the log marginal likelihood of equation (15) can be written as:

$$\ln p(\boldsymbol{t}|\boldsymbol{\alpha}, \boldsymbol{H}) = -\frac{1}{2}\left[N\ln(2\pi) + \ln\left|\boldsymbol{B} + \boldsymbol{H}\boldsymbol{A}\boldsymbol{H}^T\right| + \hat{\boldsymbol{t}}^T(\boldsymbol{B} + \boldsymbol{H}\boldsymbol{A}\boldsymbol{H}^T)^{-1}\hat{\boldsymbol{t}}\right]. \tag{16}$$

To locate a peak point, let the differential of $\ln p(\boldsymbol{t}|\boldsymbol{\alpha}, \boldsymbol{H})$ with respect to α_k be zero:

$$\frac{\partial \ln p(\boldsymbol{t}|\boldsymbol{\alpha}, \boldsymbol{H})}{\partial \alpha_k} = \frac{1}{2\alpha_k} - \frac{1}{2}\Sigma_{kk} - \frac{1}{2}\mu_k^2 = 0. \tag{17}$$

Equation (17) can be written in a fixed-point recursive form

$$\alpha_k^* = \frac{1-\alpha_k\Sigma_{kk}}{\mu_k^2}. \tag{18}$$

This completes the calculation of all formulas. Firstly, $\boldsymbol{W}$ and $\boldsymbol{\alpha}$ are initialized, then according to Equations (13) and (14), set $\boldsymbol{W}^* = \boldsymbol{\mu}$ to update the weight $\boldsymbol{W}$, and then update $\boldsymbol{\alpha}$ with equation (18). Iterate through the loop until the algorithm converges or reaches the maximum number of iterations. The convergence criterion is generally that the difference between $\ln(a_k)$ of two iterations is less than a preset precision.

For multiclassification problems, we can generalize equation (6) by a similar approach with [24]. Define the number of categories as l, we have

$$p(\boldsymbol{t}|\boldsymbol{W}) = \prod_{k=1}^{N}\prod_{i=1}^{l}\sigma\{y_i(\boldsymbol{h_k}; \boldsymbol{W_i})\}^{t_{ki}}, \tag{19}$$

where t_{ki} is the indicator variable for the case k to be a member of class i, and y_i is the predictor for the class i. According to the principle of multinominal logistic regression, Equation (19) can be alternated by

$$p(\boldsymbol{t}|\boldsymbol{W}) = \prod_{k=1}^{N}\prod_{i=1}^{l}\sigma\{y_i; y_1, y_2, \cdots, y_l\}^{t_{ki}}, \tag{20}$$

where the class predictors y_i are coupled in the multinominal softmax function

$$\sigma\{y_i; y_1, y_2, \cdots, y_l\} = \frac{e^{y_i}}{e^{y_1} + e^{y_2} + \cdots + e^{y_l}}. \tag{21}$$

Similar to the binary classification problem, since $p(\boldsymbol{W}|\boldsymbol{t}, \boldsymbol{\alpha}) \propto p(\boldsymbol{t}|\boldsymbol{W})p(\boldsymbol{W}|\boldsymbol{\alpha})$, maximizing the log of $p(\boldsymbol{t}|\boldsymbol{W})p(\boldsymbol{W}|\boldsymbol{\alpha})$

$$\begin{aligned} E(\boldsymbol{W}) &= \ln p(\boldsymbol{t}|\boldsymbol{W})p(\boldsymbol{W}|\boldsymbol{\alpha}) \\ &= \ln p(\boldsymbol{t}|\boldsymbol{W}) + \ln p(\boldsymbol{W}|\boldsymbol{\alpha}) \\ &= \sum_{k=1}^{N}\sum_{j=1}^{l} t_{kj}\ln\gamma_{kj}) - \frac{1}{2}\sum_{i=1}^{m+n}\sum_{j=1}^{l}(\alpha_{ij}{W_{ij}}^2 - \ln\alpha_{ij}) + C. \end{aligned} \tag{22}$$

Remark $\boldsymbol{H_i} = (\boldsymbol{h}_{1i}, \boldsymbol{h}_{2i}, \cdots, \boldsymbol{h}_{(m+n)i})$ and $\boldsymbol{H} = \mathrm{diag}(\boldsymbol{H}_1, \cdots, \boldsymbol{H}_l)$, and $\{\alpha_1, \cdots, \alpha_k, \cdots, \alpha_{(m+n)\times l}\}$ are the elements of diagonal matrix $\boldsymbol{A}$, which corresponds to each element in matrix $\boldsymbol{W}$. Similarly, $\boldsymbol{B}$ is a block matrix consisting of $\boldsymbol{B}_j$ with element of $\beta_{ij} = \gamma_{ij}(1-\gamma_{ij})$. Then the first and second derivatives of Equation (22) with respect to $\boldsymbol{W}$ have same format with Equations (13) and (14), which can be provided to IRLS procedure as in the ordinary logistic regression. The subsequent steps are the same as the binary classification, the $\boldsymbol{W}$ and corresponding $\boldsymbol{\alpha}$ can be iteratively obtained using Equations (13), (14), and (18).

From the network structure of FLN in Figure 1 and the introduction of the softmax function of Equation (21), it can be seen that SBFLN is equivalent to the parallel of a linear classifier and a neural network classifier, which can comprehensively utilize the linear features of the original data and the nonlinear features of the neuron mapping. Good classification performance can be expected for this structure by the sparse Bayesian approach. Specifying independent hyperparameters α_k is the key to sparsity. In the process of carrying out the iteration of $\boldsymbol{\alpha}$ to maximize the marginal likelihood, most α_k grow to infinity. As a result, the corresponding means and variance of W_k become zero through Equations (13) and (14).

4. Experiments and evaluation

In this section, 15 benchmark classification problems are chosen from the UCI machine learning repository [25] to evaluate the proposed SBFLN, including seven binary classification problems and eight multiclassification problems. These are common problems for evaluating the effectiveness and performance of classifiers such as ELM, SVM, and FLN. The specifications of the benchmark datasets are listed in Table 1. As is shown, each dataset is split into training set and testing set according to the number of samples. The samples of missing attributes are eliminated and all the attributes of each dataset are linearly scaled to $[-1, 1]$.

The SBFLN is compared with FLN [7], kernel ELM (KELM) [16], TROP-ELM [18], adaptive elastic ELM (AEELM) [17], RVM [20], and sparse Bayesian ELM (SBELM) [22] to verify the performance of various problems of the algorithm. The evaluation index is the accuracy (correctness) and size of the model. In order to

Table 1. The specifications of the benchmark datasets.

Datasets	Samples		Attributes	Classes
	Training	Testing		
Colon	30	32	2000	2
Diabetes	384	384	8	2
Geman	500	500	24	2
Liver	175	170	6	2
Mushroom	2800	2844	21	2
Australian	345	345	14	2
Spect heart	130	137	22	2
Balance	310	315	4	3
Glass	100	114	10	6
lris	75	75	4	3
Segment	1150	1160	19	7
Vehicle	420	426	18	4
Wine	89	89	13	3
Satimage	2200	2235	36	6
Vowel	495	495	13	11

reduce randomness, each experiment is repeated 30 times, and the mean and standard deviation are compared. In particular, FLN and SBFLN are individually compared in 15 benchmark problems by varying the number of hidden units from 20 to 200 in 20 increments, which helps to analyze whether SBFLN is insensitive to number of hidden units. The activation functions of FLN, SBELM, AEELM, TROP-ELM, and SBFLN are the simple logistic sigmoid functions $g(x) = \dfrac{1}{1+\exp(-x)}$, and the kernel of the KELM and RVM chosen is Gaussian RBF. We use grid search method to search the combination of parameters C and the kernel parameters γ in RVM and KELM, $C = [2^{-2}, 2^{-1}, \cdots, 2^{11}, 2^{12}]$ and $\gamma = [2^{-10}, 2^{-9}, \cdots, 2^{3}, 2^{4}]$, so the algorithms search parameter combination $15 \times 15 = 225$ times to achieve the best results. The same method is used as the other model to determine the number of hidden units. The number of hidden units is increased from 20 to 200 in steps of 20. Optimization process of hidden units number or kernel parameters is also counted as part of the training process and the training time for each algorithm is recorded. The RVM handles multiclassification problems by the same approach with SBFLN. The accuracies of the five classifiers on each problem are shown in Table 2. Superior results are highlighted in bold.

As shown in Table 2, SBFLN achieves five best results out of the 15 benchmark problems. SBELM and RVM obtain two best results respectively. AEELM achieves the best result for only one problem. FLN and TROP-ELM fail to obtain a better result. KELM achieves five optimal results. It can be seen that SBFLN and KELM get more optimal results, indicating that their applicability of different problems is relatively ahead of other algorithms. Compared with the original FLN, SBFLN achieves better results in all test problems, which proves that the sparse Bayesian learning algorithm in this paper significantly improves the generalization ability of FLN. SBFLN has obtained ten similar or better results than SBELM. The linear relationship between features and labels in these datasets is easier to handle in SBFLN. Sparse Bayesian algorithm preserves the

Table 2. Comparisons of SBFLN with other models in the average of testing accuracy (%).

Datasets	SBFLN	FLN	SBELM	TROP-ELM	AEELM	RVM	KELM
Colon	**88.65**	85.64	88.57	78.10	82.33	79.52	84.38
Diabetes	78.78	77.69	78.66	76.97	78.77	**78.79**	78.52
Geman	77.34	75.33	77.30	75.51	76.90	**77.50**	76.10
Liver	74.20	72.40	**74.21**	70.72	71.59	73.91	74.20
Mushroom	93.65	90.93	**94.33**	89.93	90.12	90.40	88.84
Australian	67.78	67.62	67.83	67.42	67.27	67.83	**68.26**
Spect heart	**86.25**	84.66	86.12	84.64	84.96	84.25	84.25
Balance	98.85	90.32	98.72	87.21	97.15	98.07	**100.00**
Glass	**95.35**	86.96	95.26	88.91	90.73	93.67	89.91
lris	98.00	94.33	98.00	96.67	**98.03**	96.67	98.00
Segment	97.36	97.05	97.40	90.43	97.06	97.23	**97.45**
Vehicle	85.12	73.42	85.17	70.00	78.01	83.66	**86.75**
Wine	**99.41**	95.36	99.41	96.58	98.33	97.76	99.41
Satimage	91.02	89.78	90.00	84.28	89.54	91.21	**92.60**
Vowel	**65.35**	59.35	65.35	58.06	63.74	64.75	59.60

useful connection weights between the labels and the features, leaving the extraneous features filtered and left to the nonlinear hidden layer neurons for processing. The other two sparse models (TROP-ELM, AEELM) are less accurate than SBFLN in most of the test problems. The result of AEELM in Iris problem is better than SBFLN, but the difference is not obvious. FLN and SBFLN that with double parallel structure are better in the Colon problem. The possible reason is that the linear classifier plays a role in the datasets with high input dimension.

The best results of all binary classification datasets belong to SBELM, SBFLN, or RVM, indicating that the Bayesian method is more effective in binary classification problem. KELM excels in most multiclassification problems due to the native multiclassification method. Moreover, from the results obtained by KELM, the accuracy of KELM is positively correlated with the number of samples, because the larger the number of measured samples in KELM is, the higher the dimension of the kernel matrix and the ability of nonlinear mapping are. However, it is worth mentioning that the high-dimensional calculation of the kernel matrix in KELM has higher requirements for computer memory and often takes more computing time. The average training time (in terms of seconds) of all algorithms is shown in Table 3. In terms of training time, FLN and SBELM need the least training time in several methods, followed by SBFLN and TROP-ELM. The training speed of SBFLN differs greatly from that of SBELM only in Colon datasets which has high input feature dimension, while the training speed of other datasets is similar. Compared with FLN, SBFLN takes some time to compute hyperparameters, so it is slower than FLN. RVM and KELM using kernel methods are the slowest. Especially in datasets with large number of samples, the training time of RVM and KELM is dozens of times higher than those of other algorithms. Since the dimension of the kernel matrix is $N \times N$, a large amount of computer memory is consumed when N is large. If the memory of the computer is insufficient, the training time will increase exponentially, which is contrary to the original intention of fast learning. Table 4 lists the standard deviation of all testing accuracy. Compared to original FLN and SBELM, most standard deviation of

SBFLN is lower, showing that SBFLN significantly improved the stability and repeatability. SBFLN obtains five minimum standard deviations out of 15 problems, and the number of optimal results is the highest among all algorithms, indicating that SBFLN is the most robust of all the algorithms compared. In general, SBFLN is competitive or advantageous in seven models.

Table 3. The average training time (s).

Datasets	SBFLN	FLN	SBELM	TROP-ELM	RVM	KELM
Colon	0.3410	0.9541	**0.0031**	0.0047	0.1152	0.0568
Diabetes	0.0867	**0.055**	0.0885	0.1078	1.1000	0.2874
Geman	0.0715	**0.0682**	0.0684	0.0983	0.9923	0.2207
Liver	0.0078	**0.0066**	0.0081	0.0149	0.1437	0.0812
Mushroom	1.2631	**0.9249**	1.2397	1.2745	22.6797	6.7808
Australian	0.0692	**0.0437**	0.0633	0.1162	1.0693	0.1338
Spect heart	0.0088	0.0091	**0.0083**	0.0105	0.0843	0.0836
Balance	0.0043	0.0062	**0.0037**	0.0057	0.0646	0.0606
Glass	0.0037	0.0033	0.0038	0.0050	0.0075	**0.0024**
lris	**0.0028**	0.0029	0.0036	0.0068	0.0053	0.0029
Segment	1.0952	**0.8812**	1.1559	1.3694	16.4044	9.5824
Vehicle	0.1176	**0.0921**	0.1158	0.1744	1.9166	1.4878
Wine	0.0027	0.0031	0.0021	**0.0021**	0.0023	0.0029
Satimage	3.7074	5.3256	**3.6337**	4.8975	53.2604	14.7737
Vowel	0.0266	0.0545	**0.0230**	0.0289	0.2552	0.0481

Table 4. Standard deviation of testing accuracy.

Datasets	SBFLN	FLN	SBELM	TROP-ELM	AEELM	RVM	KELM
Colon	**4.26**	5.43	4.81	13.11	7.26	10.36	5.38
Diabetes	3.57	3.84	3.58	5.48	1.94	**1.77**	1.87
Geman	**2.59**	2.59	2.59	3.95	4.17	3.66	4.87
Liver	4.14	4.35	4.15	5.65	**2.63**	4.81	4.51
Mushroom	0.04	0.04	**0.03**	0.09	0.07	0.06	0.07
Australian	**0.12**	0.13	0.12	1.97	0.38	0.12	1.32
Spect heart	2.64	4.33	2.64	2.49	5.27	3.26	**2.30**
Balance	0.70	0.83	0.72	0.82	1.58	1.24	**0.00**
Glass	2.61	2.64	3.58	**2.32**	2.74	6.17	2.80
lris	**2.56**	2.56	2.98	13.33	13.65	3.33	2.98
Segment	0.83	0.85	**0.55**	0.76	0.50	0.76	0.86
Vehicle	2.53	2.54	3.15	1.48	1.59	1.87	**1.19**
Wine	1.33	4.53	**1.32**	13.86	1.32	2.44	1.32
Satimage	**0.95**	1.34	0.96	1.60	1.13	1.05	1.60
Vowel	1.39	3.40	1.40	15.48	3.60	1.05	**0.42**

Table 5. The number of nonzero weights.

Datasets	SBFLN	FLN	SBELM	TROP-ELM	AEELM
Colon	321	2108	9	7	10
Diabetes	11	160	10	71	67
Geman	24	72	17	56	54
Liver	11	76	10	59	59
Mushroom	13	209	10	154	153
Australian	4	34	3	13	10
Spect heart	10	42	7	18	14
Balance	25	372	24	168	168
Glass	19	780	19	558	554
lris	9	142	9	177	176
Segment	40	1533	39	1309	1312
Vehicle	61	736	57	524	526
Wine	13	495	12	252	249
Satimage	127	1344	115	1020	1016
Vowel	81	2343	79	2189	2178

The number of nonzero weights of several the nonkernel models (SBFLN, FLN, SBELM, TROP-ELM, AEELM) is listed in Table 5. It can be seen that SBELM has the least nonzero weights. SBFLN has less nonzero weights in the datasets with low dimensional features (Diabetes, Liver, Balance, Iris, etc.), and has more nonzero weights in the datasets with high dimensional features (Colon). This also shows that SBFLN has utilized original features. Although TROP-ELM and AEELM pruned many redundant neurons of the network, the number of nonzero weights is greater than SBELM and SBFLN, indicating that the model is not as compact as SBELM and SBFLN.

To test whether SBFLN is insensitive to the number of hidden units, we compare the accuracy of SBFLN and FLN under different numbers of hidden units. As shown in Figure 2, FLN accuracy is not as good as SBFLN in most cases. The highest accuracy of the two models is very close in some problems (Colon, Vehicle, Vowel). However, in most other problems (Diabetes, Geman, Liver, Spect heart, Glass), the accuracy of FLN is worse than SBFLN, and it decreases as the number of hidden units increases. The reason may be that the FLN is overfitting in these datasets, resulting in poor generalization. The SBFLN exhibits stable high accuracy under different number of hidden units, and is insensitive to the number of hidden units. The objective function of SBFLN contains a penalty term for the weight, which improves the overfitting problem caused by collinearity in the feature. Moreover, the method of this paper automatically calculates the hyperparameter of the penalty term, so the overfitting problem in the case of large-scale network is not prominent. Moreover, the ideal accuracy can be achieved when the number of hidden units is only about $60 \sim 80$ and the real number of effectively hidden units can be even much smaller under the pruning effect, so the calculation cost can be significantly reduced. If the FLN is to achieve an accuracy comparable to SBFLN only by increasing the number of hidden nodes (sometimes possible, such as Vehicle problem in this paper), it not only leads to an increase in network complexity and computational cost, but also increases the chance of model overfitting. In summary, SBFLN is an efficient model algorithm with low computational cost and insensitivity to the number of hidden units.

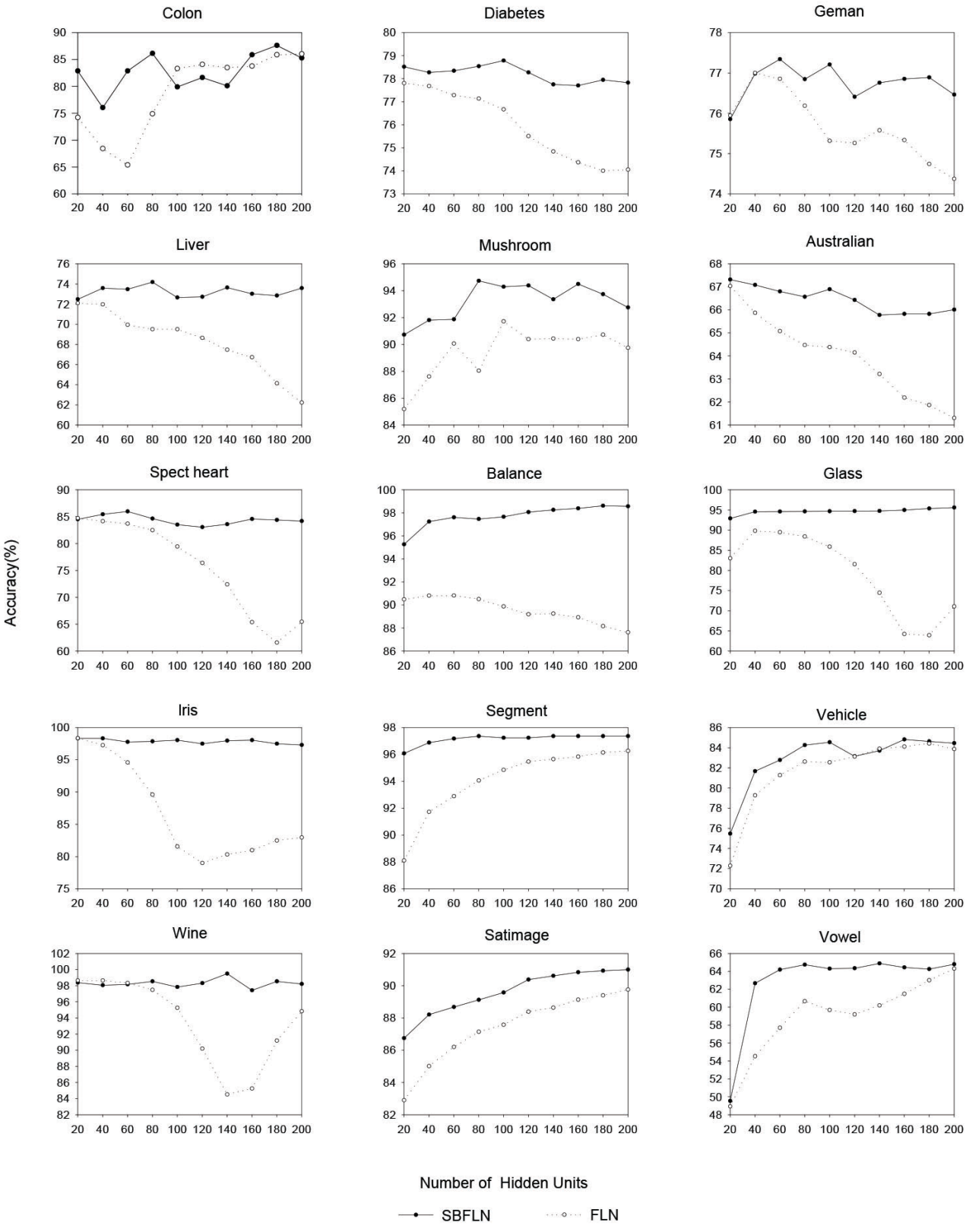

Figure 2. Structure of the fast learning network.

5. Conclusions

This paper proposes a sparse Bayesian approach to FLN for multiclassification. SBFLN estimates the output weights of FLN using a sparse Bayesian approach, which not only generates a natural probabilistic distribution for uncertain outputs, but also uses independent hyperparameters α to control the probability distribution of the output weights, automatically adjust most of the output weights to zero, thereby obtaining sparsity. In addition, by maximizing the marginal likelihood instead of minimizing training error, SBFLN does not suffer from overfitting problem and has excellent generalization ability. From the structure, SBFLN is a parallel of linear classifier and ELM classifier. Better precision and repeatability can be achieved compared to the SBELM in some datesets with linear features or high dimension. Finally, SBFLN is insensitive to the number of hidden layer nodes, which means that an accuracy of close to large-scale networks can be obtained with a very small number of neurons. These advantages have been verified in experimental results obtained from 15 different classification problems.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No. 51641609) and the Natural Science Foundation of Hebei Province (Grant No. F2019203320).

References

[1] He W, Qian F, Cao J. Pinning-controlled synchronization of delayed neural networks with distributed-delay coupling via impulsive control. Neural Networks 2017; 85: 1-9.

[2] Kalinić H, Mihanović H, Cosoli S, Tudor M, Vilibić I. Predicting ocean surface currents using numerical weather prediction model and kohonen neural network: a northern adriatic study. Neural Computing and Applications 2017; 28: 611-620.

[3] Liu Z, Wang R, Tao M. Smoteadanl: a learning method for network traffic classification. Journal of Ambient Intelligence and Humanized Computing 2016; 7: 121-130.

[4] Huang GB, Zhu QY, Siew CK. Extreme learning machine: theory and applications. Neurocomputing 2006; 70: 489-501.

[5] Huang GB, Wang DH, Lan Y. Extreme learning machines: a survey. International Journal of Machine Learning and Cybernetics 2011; 2: 107-122.

[6] Ding S, Zhao H, Zhang Y, Xu X, Nie R. Extreme learning machine: algorithm, theory and applications. Artificial Intelligence Review 2015; 44: 103-115.

[7] Li G, Niu P, Duan X, Zhang X. Fast learning network: a novel artificial neural network with a fast learning speed. Neural Computing and Applications 2014; 24: 1683-1695.

[8] Li G, Qi X, Chen B, Ma Y, Niu P et al. Fast learning network with parallel layer perceptrons. Neural Processing Letters 2018; 47: 1-16.

[9] Rao CR, Mitra SK. Generalized Inverse of Matrices and Its Applications. New York, NY, USA: Wiley, 1971.

[10] Li G, Niu P, Wang H, Liu Y. Least square fast learning network for modeling the combustion efficiency of a 300WM coal-fired boiler. Neural Networks 2014; 51: 57-66.

[11] Niu P, Chen K, Ma Y, Li X, Liu A et al. Model turbine heat rate by fast learning network with tuning based on ameliorated krill herd algorithm. Knowledge-Based Systems 2017; 118: 80-92.

[12] Ali MH, Al Mohammed BAD, Ismail A, Zolkipli MF. A new intrusion detection system based on fast learning network and particle swarm optimization. IEEE Access 2018; 6: 20255-20261.

[13] Lu S, Wang X, Zhang G, Zhou X. Effective algorithms of the moore-penrose inverse matrices for extreme learning machine. Intelligent Data Analysis 2015; 19: 743-760.

[14] Li G, Niu P. An enhanced extreme learning machine based on ridge regression for regression. Neural Computing and Applications 2013; 22: 803-810.

[15] Miche Y, Sorjamaa A, Bas P, Simula O, Jutten C et al. Op-elm: optimally pruned extreme learning machine. IEEE Transactions on Neural Networks 2010; 21: 158-162.

[16] Balasundaram S, Gupta D. 1-norm extreme learning machine for regression and multiclass classification using newton method. Neurocomputing 2014; 128: 4-14.

[17] Luo X, Chang X, Ban X. Regression and classification using extreme learning machine based on l1-norm and l2-norm. Neurocomputing 2016; 174: 179-186.

[18] Miche Y, Van Heeswijk M, Bas P, Simula O, Lendasse A. Trop-elm: a double-regularized elm using lars and tikhonov regularization. Neurocomputing 2011; 74: 2413-2421.

[19] Haykin SS. Neural Networks and Learning Machines. Upper Saddle River, NJ, USA: Prentice Hall, 2009.

[20] Tipping ME. Sparse bayesian learning and the relevance vector machine. Journal of Machine Learning Research 2001; 1: 211-244.

[21] Soria-Olivas E, Gomez-Sanchis J, Martin JD, Vila-Frances J, Martinez M et al. Belm: Bayesian extreme learning machine. IEEE Transactions on Neural Networks 2011; 22: 505-509.

[22] Luo J, Vong CM, Wong PK. Sparse bayesian extreme learning machine for multi-classification. IEEE Transactions on Neural Networks and Learning Systems 2014; 25: 836-843.

[23] MacKay DJ. The evidence framework applied to classification networks. Neural Computation 1992; 4: 720-736.

[24] Zhang H, Malik J. Selecting shape features using multi-class relevance vector machine. Technical Rep No UCB/EECS-2005 2005; 6.

[25] Blake CL, Merz CJ. UCI Repository of Machine Learning Database (Machine Readable Data Repository). Department of Information and Computer Science, University of California, Irvine, 1999.