A Double-Layer ELM with Added Feature Selection Ability using a Sparse Bayesian Approach

Farkhondeh Kiaee[a], Christian Gagné[b], Hamid Sheikhzadeh[a]

[a]*Department of Electrical Engineering, Amirkabir University of Technology (Tehran Polytechnic), Tehran 15914, Iran*
(e-mail: f.kiaee@aut.ac.ir;hsheikh@aut.ac.ir).
[b]*Département de génie électrique et de génie informatique, Université Laval, Québec, QC, Canada G1V 0A6. (e-mail: christian.gagne@ulaval.ca)*

Abstract

The Sparse Bayesian Extreme Learning Machine (SBELM) has been recently proposed to reduce the number of units activated on the hidden layer. To deal with high-dimensional data, a novel sparse Bayesian Double-Layer ELM (DL-ELM) is proposed in this paper. The first layer of the proposed DL-ELM is based on a set of SBELM subnetworks which are separately applied to the features on the input layer. The second layer consists in a set of weight parameters to determine the contribution of each feature in the output. Adopting a Bayesian approach and using Gaussian priors, the proposed method is sparse in both the hidden layer (of SBELM subnetworks) and the input layer. Sparseness in the input layer (i.e. pruning of irrelevant features) is achieved by decaying weights on the second layer to zero, such that the contribution of the corresponding input feature is deactivated. The proposed framework then enables simultaneous feature selection and classifier design at the training time. Experimental comparisons on real benchmark data sets show that the proposed method benefits from efficient feature selection ability while providing a compact classification model of good accuracy and generalization properties.

Keywords: Extreme learning machine, Feature selection, Sparse Bayesian learning, Subnetwork architecture, Binary classification.

1. Introduction

Sparse learning methods allow the automatic selection and use of only the most important features in the input set. The human mind performs operations analogous to feature selection and the application of a machine learning algorithm to a reduced feature set [1]. It is known for example that the human brain is able to reconstruct a total 3D model of a previously known object, by observing that object only from one point of view.

Support Vector Machine (SVM) [2] and Relevance Vector Machine (RVM) [3] have been shown to be efficient for classification, while being the base of several approaches facilitating sparseness over the input features. By the input sparsity we mean that feature selection property is enabled for the derived approaches. Some embedded feature selection methods have been derived from SVM classifiers [4, 5, 6]. The joint classifier and feature optimization (JCFO) model of Krishnapuram *et al.* (2004) performs feature selection using adjustment of the kernel scaling parameter [6]. Nguyen *et al.* (2010) proposed a weighted SVM relying on a convex energy-based framework for joint feature selection and classification [5].

Few embedded methods for joint feature selection and classifier design based on RVM have been proposed in the literature [7, 8]. The negative log-likelihood loss function in the proposed joint feature selection and classifier learning (JFSCL) method of Lapedriza *et al.* (2008) is augmented with two terms: a regularization term on the parameters of RVM and a term for feature selection [7]. An extended version of RVM known as the Relevance Sample-Feature Machine (RSFM) was recently introduced by Mohsenzadeh *et al.* (2013), which is sparse in both observation space and input feature space [8]. However, the drawbacks of this method is that the training of the RVM-based models is slow given its high computational complexity due to matrix inversions. Moreover, the performance of RVM-based models is sensitive to the selection of kernel-width, whose determination requires using the inaccurate and time consuming cross-validation procedure.

The Extreme learning machine (ELM) [9, 10, 11] has recently gained increasing interest from various research fields [12, 13, 14, 15]. ELM is a single hidden layer feed-forward neural network with random weight hidden neurons. However, the accuracy of ELM is highly sensitive to the number of hidden neurons. The approach developed in [15], uses some optimization algorithms to find the appropriate number of neurons. However, it does not benefit from an embedded feature selection scheme in which relevant features

are selected during the learning process, and the learning effectiveness can be improved dramatically.

A more general form of ELM in which a hidden node itself can be a subnetwork of several hidden nodes has been recently introduced [16]. The construction of subnets plays distinctive roles in the performance of different learning networks. In a recent deep network architecture, subnets based on multilayer perceptron are successfully applied to replace the linear convolution filter for better modelling of highly nonlinear data [17]. In the approach presented in [18], a set of subnets based on auto-associative neural networks (AANN) are employed to reduce input dimension. However, researchers who are trying to use subnets in the configuration of their learning networks are still facing following questions.

What is the smallest proper number of hidden subnets required without affecting learning effectiveness. In order to reduce the size of hidden layer, a couple of incremental learning algorithms have been proposed [16, 19], in which subnetworks of hidden nodes are added one by one until achieving the expected training error. It is proved by Yang *et al.* (2015) that efficient updates of some hidden nodes during the growth of the networks results in a better ability of achieving minimum error than other incremental networks.

What should be the number of nodes used in the input layer of each individual subnet. In the incremental methods [16, 19], there are as many input nodes for the subnets as the number of attributes in the corresponding data. In other researches, the set of features that is used as the input of subnets are subjected to quantitative or qualitative modification. The qualitative modifications involve a change in the feature space such as the hierarchical ELM method proposed in [18]. In this method a set of AANN subnets are used as a feature transformations stage before an ELM block. However, this method does not provide the feature selection ability for the ELM method. A quantitative modification to input features is conducted in [20] using a method called *Evolutionary WRAPPER*. This method attempts the possible combination with the attributes subnets. The presented method is complex and much time consuming in implementation.

Sparse Bayesian learning (SBL) is a family of Bayes methodologies that aims to find a sparse estimate of output weights, by imposing a hierarchical-independent hyperparameters for the priors of different weights. It is shown that SBL retains a desirable property of the 0-norm optimization (i.e., the global optimum point is achieved at the maximally sparse solution) [21, 22].

As alternative technologies to incremental methods and in order to automatic selection of an appropriate number of neurons, sparse Bayesian versions of ELM method are also introduced for both regression [23] and classification [24] tasks. The Sparse Bayesian ELM (SBELM) [24] is a derivative of SBL with automatic relevance determination (ARD) prior. ARD prior methodology assumes a zero-mean Gaussian prior with a different variance for each weight and a flat Gamma prior on the hyperparameters (variance of the weights). This hierarchical prior gives rise to an implicit (marginalized) prior over the weight parameters with a Student-t distribution which is packed around zero and hence gains sparsity by tuning some weights to zeros.

Inspired by [16], a SBELM block itself can be used as a subnetwork of a larger modular network. In this paper, we propose a double layer feedforward neural network. The first layer consists of a set of SBELM subnets with equal weight parameters, where each of them are applied to an input feature. Then a second layer of weights is added to the network to specify the contribution of each feature in modelling the data. The proposed DL-ELM method aims to fill the gap between ELM and feature selection methods by efficient combination of SBELM subnetworks (see Fig. 1). We propose the sparse Bayesian inference approach for learning the model parameters under the evidence framework.

Briefly, the main contributions of the proposed DL-ELM can be summarized as:

- Different from other ELM methods with subnet architecture, which are only able to reduce the size of hidden layer, DL-ELM is a sparse model in both the hidden layer and input layer;
- Sparseness in the input layer provides DL-ELM with robustness against irrelevant features and low system complexity;
- Fast convergence due to the use of the evidence approximation to efficiently compute the optimum value of model hyperparameters;
- Bayesian inference provides probabilistic predictions in contrast to previously presented feature selection methods on SVM in the literature that only produce a point estimation.

Experimental results on 24 binary datasets and one handwritten digits dataset indicate that the accuracy of the proposed DL-ELM method is

superior or very close to the SBELM model while being sparser in the hidden layer. Furthermore, DL-ELM is sparse in the input layer by successfully pruning irrelevant features. Comparison of the proposed DL-ELM with other state-of-the-art methods (such as JCFO, JFSCL, RSFM, and weighted SVM) shows that the DL-ELM method allows small models to be obtained while being able to accurately classify the datasets.

The organization of this paper is as follows. Section 2 provides a short review on SBELM for binary classification. Our proposed DL-ELM model, as an extension of SBELM for joint feature selection and classification based on double-layer network, is described in Section 2. The evaluation of the performance of the proposed DL-ELM are conducted in Section 3. Finally, we conclude our paper in Section 4.

2. Classification Methods

Consider a set of input-target pairs $\{\boldsymbol{x}_n, t_n\}_{n=1}^N$ as the samples for model training, where t_n represents the target output of the $n-th$ sample and $\boldsymbol{x}_n$ represents the corresponding input vector. We assume that $\boldsymbol{x}_n$ is composed of d features, $\boldsymbol{x}_n = \{x_{n,1}, x_{n,2}, \ldots, x_{n,d}\}$.

In the case of binary classification, every training sample can be considered as an independent Bernoulli event. Then the log-likelihood function of all the training samples is:

$$\log p(\boldsymbol{t}|\boldsymbol{M}) = \sum_{n=1}^{N} [t_n \log \sigma(y_n) + (1 - t_n) \log(1 - \sigma(y_n))]. \tag{1}$$

where $\sigma(\cdot)$ is the sigmoid function, $\sigma(x) = 1/(1 + \exp(-x))$. $\boldsymbol{M}$ is the set of unknown model parameters and y_n is a function of $\boldsymbol{M}$ which is defined precisely for each model in the following.

2.1. Spare Bayesian Extreme Learning Machine (SBELM)

A SBELM [24] is defined by L hidden neurons and an activation function $h(\cdot)$. Thus, the set of unknown parameters in (1) is $\boldsymbol{M} = \{\boldsymbol{w}\}$ and the term y_n is represented as

$$y_n = \boldsymbol{w}^T \boldsymbol{h}(\boldsymbol{\Theta}, \boldsymbol{x}_n), \tag{2}$$

where $\boldsymbol{h}(\boldsymbol{\Theta}, \boldsymbol{x}_n) = [1, h_1(\theta_1, \boldsymbol{x}_n), \ldots, h_L(\theta_L, \boldsymbol{x}_n)]^T$ is the hidden feature mapping with respect to input $\boldsymbol{x}_n$ and $\boldsymbol{\Theta} = [\theta_1, \ldots, \theta_L]$ is the vector of randomly generated parameters of the hidden layer. The vector $\boldsymbol{w} = [w_0, w_1, \ldots, w_L]^T$

includes the output weights between the nodes of the hidden layer and the output node. The structure of a standard SBELM block is shown in Fig. 2.

The relation can be written in matrix form as

$$\boldsymbol{y} = \boldsymbol{w}^T \boldsymbol{H}, \tag{3}$$

where the $(L+1) \times N$ hidden layer feature-mapping matrix, $\boldsymbol{H}$, can be expressed as

$$\boldsymbol{H} = \begin{pmatrix} 1 & \dots & 1 \\ h_1(\theta_1, \boldsymbol{x}_1) & \dots & h_1(\theta_1, \boldsymbol{x}_N) \\ \vdots & \ddots & \vdots \\ h_L(\theta_L, \boldsymbol{x}_1) & \dots & h_L(\theta_L, \boldsymbol{x}_N) \end{pmatrix}. \tag{4}$$

More specifically, the prior is defined in a hierarchical way. Some independent zero-mean Gaussian prior distributions are considered on weight parameters, leading to the following joint prior density:

$$p(\boldsymbol{w}|\boldsymbol{\alpha}) = \prod_{i=0}^{L} \mathcal{N}(w_i|0, \alpha_i^{-1}) = \frac{|\boldsymbol{A}|^{\frac{1}{2}}}{(2\pi)^{\frac{L}{2}}} \exp\left(-\frac{\boldsymbol{w}^T \boldsymbol{A} \boldsymbol{w}}{2}\right), \tag{5}$$

where each inverse variance (precision) hyperparameter α_i is associated with the prior of w_i parameter and $\boldsymbol{\alpha} = [\alpha_0, \dots, \alpha_L]^T$. $\boldsymbol{A}$ is defined as $\boldsymbol{A} = \text{diag}(\boldsymbol{\alpha})$. In addition a Gamma distribution prior with zero parameters is also imposed on the hyperparameters.

The two-stage hierarchical prior on the weight parameters is actually a Student's t-distribution and is called *sparse*, since it enforces most of the α_is to be large, thus the corresponding weights are set to zero and pruned from the model. This way, the complexity of the model is controlled in an automatic way and over-fitting is avoided.

In the SBELM model, the optimum weight parameters, $\boldsymbol{w}^{MAP}$, are achieved by maximizing the following objective function in which the log-likelihood of the model (1) is augmented with a penalty term that corresponds to the logarithm of the prior,

$$\mathcal{L}(\boldsymbol{w}, \boldsymbol{\alpha}) = \log p(\boldsymbol{t}, \boldsymbol{w}|\boldsymbol{\alpha}) = \log p(\boldsymbol{t}|\boldsymbol{w}) - \frac{1}{2}\boldsymbol{w}^T \boldsymbol{A} \boldsymbol{w}. \tag{6}$$

To find the optimum values of the hyperparameters $\boldsymbol{\alpha}$ in SBELM, the differential of the marginal log-likelihood function with respect to the logarithm of each α_i is set to zero, which results in the updating equations:

$$\alpha_i^{new} = \frac{\gamma_i}{({w_i}^{MAP})^2}; \qquad \gamma_i = 1 - \alpha_i \Sigma_{ii}. \tag{7}$$

2.2. Double-Layer Extreme Learning Machine (DL-ELM)

The proposed double-layer ELM extends the SBELM with adjustable parameters $\boldsymbol{M} = \{\boldsymbol{w}, \boldsymbol{\beta}\}$ such that the following relation is satisfied:

$$y_n = \sum_{j=1}^{d} (\boldsymbol{w}^T \boldsymbol{h}(\boldsymbol{\Theta}, x_{n,j}))\, \beta_j = \boldsymbol{w}^T \boldsymbol{H_n} \boldsymbol{\beta}, \tag{8}$$

where $\boldsymbol{h}(\boldsymbol{\Theta}, x_{n,j}) = [1, h_1(\theta_1, x_{n,j}), \ldots, h_L(\theta_L, x_{n,j}]^T$ is the output vector of the hidden layer with respect to the j^{th} input feature of the n^{th} sample $x_{n,j}$ and $\boldsymbol{\Theta} = [\Theta_1, \ldots, \Theta_L]$ is the vector of randomly generated parameters of the hidden layer.

The structure of the proposed double-layer ELM as shown in Fig. 3 includes d standard SBELM subnetworks. $\boldsymbol{w} = [w_1, \ldots, w_L]^T$ is the weight vector of hidden neurons to the output neuron that is shared in all SBELM subnetsworks. The outputs of SBELM subnets are combined with an added set of unknown weight parameters $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_d]^T$ to give the final output.

The matrix $\boldsymbol{H}_n$, in the matrix form representation in (8), can be expressed as

$$\boldsymbol{H}_n = \begin{pmatrix} 1 & \ldots & 1 \\ h_1(\theta_1, x_{n,1}) & \ldots & h_1(\theta_1, x_{n,d}) \\ \vdots & \ddots & \vdots \\ h_L(\theta_L, x_{n,1}) & \ldots & h_L(\theta_L, x_{n,d}) \end{pmatrix}. \tag{9}$$

Hence, in terms of matrices, a formulation of (8) that includes all of the N training samples can be written as

$$\boldsymbol{y} = \boldsymbol{H}_w \boldsymbol{\beta} = \boldsymbol{H}_\beta \boldsymbol{w}, \tag{10}$$

in which $\boldsymbol{H}_w$ and $\boldsymbol{H}_\beta$ are defined as $\boldsymbol{H}_w = [\boldsymbol{H}_1^T \boldsymbol{w}, \boldsymbol{H}_2^T \boldsymbol{w}, \ldots, \boldsymbol{H}_N^T \boldsymbol{w}]^T$ and $\boldsymbol{H}_\beta = [\boldsymbol{H}_1^T \boldsymbol{\beta}, \boldsymbol{H}_2^T \boldsymbol{\beta}, \ldots, \boldsymbol{H}_N^T \boldsymbol{\beta}]^T$, respectively.

Similar to (5), zero-mean Gaussian prior distributions are considered on weight parameters

$$p(\boldsymbol{w}|\boldsymbol{\alpha_w}) = \prod_{i=0}^{L} \mathcal{N}(w_i|0, \alpha_{w_i}^{-1}) = \frac{|\boldsymbol{A_w}|^{\frac{1}{2}}}{(2\pi)^{\frac{L+1}{2}}} \exp\left(-\frac{\boldsymbol{w}^T \boldsymbol{A_w} \boldsymbol{w}}{2}\right), \tag{11}$$

$$p(\boldsymbol{\beta}|\boldsymbol{\alpha_\beta}) = \prod_{i=1}^{d} \mathcal{N}(\beta_i|0, \alpha_{\beta_i}^{-1}) = \frac{|\boldsymbol{A_\beta}|^{\frac{1}{2}}}{(2\pi)^{\frac{d}{2}}} \exp\left(-\frac{\boldsymbol{\beta}^T \boldsymbol{A_\beta} \boldsymbol{\beta}}{2}\right), \tag{12}$$

where each inverse variance (precision) hyperparameters α_{w_i} and α_{β_i} are associated with the prior of w_i and β_i, respectively, and $\boldsymbol{\alpha_w} = [\alpha_{w_0}, \ldots, \alpha_{w_L}]^T$, $\boldsymbol{\alpha_\beta} = [\alpha_{\beta_1}, \ldots, \alpha_{\beta_d}]^T$. Matrices $\boldsymbol{A_w}$ and $\boldsymbol{A_\beta}$ are defined as $\boldsymbol{A_w} = \text{diag}(\boldsymbol{\alpha}_w)$ and $\boldsymbol{\beta} = \text{diag}(\boldsymbol{\alpha}_\beta)$. Analogous to the approach presented in the SBELM, the hyperparameters are assumed to follow a flat Gamma prior distribution.

The posterior distribution of unknown parameters of the proposed model can be factorized as

$$p(\boldsymbol{w}, \boldsymbol{\beta}, \boldsymbol{\alpha}_w, \boldsymbol{\alpha}_\beta | \boldsymbol{t}) = p(\boldsymbol{w}, \boldsymbol{\beta} | \boldsymbol{t}, \boldsymbol{\alpha}_w, \boldsymbol{\alpha}_\beta) p(\boldsymbol{\alpha}_w, \boldsymbol{\alpha}_\beta | \boldsymbol{t}). \tag{13}$$

where the term in the right-hand side of (13) is the posterior over the weights multiplied by the hyperparameter posterior. The details of the optimization of the weights and hyperparameters is described in the next sections.

2.2.1. Inference of the weight parameters

Using the Bayes' rule, the posterior over the weights (the first term on the right hand side of (13)) can be extended as

$$p(\boldsymbol{w}, \boldsymbol{\beta} | \boldsymbol{t}, \boldsymbol{\alpha}_w, \boldsymbol{\alpha}_\beta) \propto p(\boldsymbol{t} | \boldsymbol{w}, \boldsymbol{\beta}) p(\boldsymbol{w} | \boldsymbol{\alpha}_w) p(\boldsymbol{\beta} | \boldsymbol{\alpha}_\beta). \tag{14}$$

Replacing the first term on the right hand side of (14) by (1) gives rise to

$$\begin{aligned} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\beta}, \boldsymbol{\alpha}_w, \boldsymbol{\alpha}_\beta) &= \log p(\boldsymbol{w}, \boldsymbol{\beta} | \boldsymbol{t}, \boldsymbol{\alpha}_w, \boldsymbol{\alpha}_\beta) \\ &= p(\boldsymbol{t} | \boldsymbol{w}, \boldsymbol{\beta}) - \frac{1}{2} \boldsymbol{w}^T \boldsymbol{A}_w \boldsymbol{w} - \frac{1}{2} \boldsymbol{\beta}^T \boldsymbol{A}_\beta \boldsymbol{\beta}. \end{aligned} \tag{15}$$

A Gaussian distribution is then approximated for the objective function $\mathcal{L}$ using the Laplace approximation approach.

$$\begin{aligned} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\beta}, \boldsymbol{\alpha}_w, \boldsymbol{\alpha}_\beta) = \mathcal{L}(\boldsymbol{w}^{MAP}, \boldsymbol{\beta}^{MAP}, \boldsymbol{\alpha}_w, \boldsymbol{\alpha}_\beta) \\ + (\boldsymbol{w} - \boldsymbol{w}^{MAP}) \boldsymbol{\Sigma}_w^{-1} (\boldsymbol{w} - \boldsymbol{w}^{MAP}) \\ + (\boldsymbol{\beta} - \boldsymbol{\beta}^{MAP}) \boldsymbol{\Sigma}_\beta^{-1} (\boldsymbol{\beta} - \boldsymbol{\beta}^{MAP}), \end{aligned} \tag{16}$$

where the mean $\boldsymbol{w}^{MAP}$ and $\boldsymbol{\beta}^{MAP}$ of the approximated Gauss are the Laplace's mode around $\boldsymbol{w}$ and $\boldsymbol{\beta}$, respectively. The covariance matrix $\boldsymbol{\Sigma}_w$ and $\boldsymbol{\Sigma}_\beta$ of the approximated Gaussian are negated inverse Hessian matrices (presented next in (20) to (23), respectively).

In general, it is efficient to find the Laplace's mode $\boldsymbol{w}^{MAP}$ and $\boldsymbol{\beta}^{MAP}$ using the iterative reweighted least squares (IRLS) method [25]. The IRLS involves

computation of the gradient and Hessian of the model

$$\nabla_w \mathcal{L} = \boldsymbol{H}_\beta^T[\boldsymbol{t} - \sigma(\boldsymbol{y})] - \boldsymbol{A}_w \boldsymbol{w},\ \nabla_{\boldsymbol{w}}^2 \mathcal{L} = -\boldsymbol{H}_\beta^T \boldsymbol{B} \boldsymbol{H}_\beta - \boldsymbol{A}_w, \tag{17}$$

$$\nabla_\beta \mathcal{L} = \boldsymbol{H}_w^T[\boldsymbol{t} - \sigma(\boldsymbol{y})] - \boldsymbol{A}_\beta \boldsymbol{\beta},\ \nabla_{\boldsymbol{\beta}}^2 \mathcal{L} = -\boldsymbol{H}_w^T \boldsymbol{B} \boldsymbol{H}_w - \boldsymbol{A}_\beta, \tag{18}$$

where $\boldsymbol{B}$ is an $N \times N$ diagonal matrix with element $b_i = \sigma(y_i)(1 - \sigma(y_i))$.

It can be seen from (17) and (18) that the gradient vector and Hessian matrix corresponding to $\boldsymbol{w}$ will depend on $\boldsymbol{\beta}$. Similarly, the gradient and Hessian with respect to $\boldsymbol{\beta}$ will depend on $\boldsymbol{w}$. To solve this problem, we use an iterative algorithm so that in each step one of the variables $\boldsymbol{w}$ or $\boldsymbol{\beta}$ is considered to be constant and the covariance matrix and mean vector of the other variable is computed.

Therefore, using the IRLS method, $\boldsymbol{w}$ is estimated from the following equation

$$\boldsymbol{w}_{new} = \boldsymbol{w}_{old} - (\nabla_w^2 \mathcal{L})^{-1} \nabla_w \mathcal{L} = (\boldsymbol{H}_\beta^T \boldsymbol{B} \boldsymbol{H}_\beta + \boldsymbol{A})^{-1} \boldsymbol{H}_\beta^T \boldsymbol{B} \hat{\boldsymbol{t}}, \tag{19}$$

where $\hat{\boldsymbol{t}}_w = \boldsymbol{H}_\beta \boldsymbol{w} + \boldsymbol{B}^{-1}(\boldsymbol{t} - \boldsymbol{y})$, giving rise to the following update equations for the mean parameter $\boldsymbol{w}^{MAP}$ and covariance matrix $\boldsymbol{\Sigma}_w$ of the Laplace approximated Gaussian distribution over $\boldsymbol{w}$,

$$\boldsymbol{w}^{MAP} = \boldsymbol{\Sigma}_w \boldsymbol{H}_\beta^T \boldsymbol{B} \hat{\boldsymbol{t}}_w, \tag{20}$$

$$\boldsymbol{\Sigma}_w = -(\nabla_{\boldsymbol{w}}^2 \mathcal{L} \mid \boldsymbol{w} = \boldsymbol{w}^{MAP})^{-1} = (\boldsymbol{A}_w + \boldsymbol{H}_\beta^T \boldsymbol{B} \boldsymbol{H}_\beta)^{-1}. \tag{21}$$

Similarly, the update equations for the mean parameter $\boldsymbol{\beta}^{MAP}$ and covariance matrix $\boldsymbol{\Sigma}_\beta$ are

$$\boldsymbol{\beta}^{MAP} = \boldsymbol{\Sigma}_\beta \boldsymbol{H}_w^T \boldsymbol{B} \hat{\boldsymbol{t}}_\beta, \tag{22}$$

$$\boldsymbol{\Sigma}_\beta = -(\nabla_{\boldsymbol{\beta}}^2 \mathcal{L} \mid \boldsymbol{\beta} = \boldsymbol{\beta}^{MAP})^{-1} = (\boldsymbol{A}_\beta + \boldsymbol{H}_w^T \boldsymbol{B} \boldsymbol{H}_w)^{-1}, \tag{23}$$

where $\hat{\boldsymbol{t}}_\beta = \boldsymbol{H}_w \boldsymbol{\beta} + \boldsymbol{B}^{-1}(\boldsymbol{t} - \boldsymbol{y})$.

2.2.2. Hyperparameters optimisation

For the calculation of the hyperparameter posterior, considering the approximation $p(\boldsymbol{\alpha}_w, \boldsymbol{\alpha}_\beta | \boldsymbol{t}) \propto p(\boldsymbol{t} | \boldsymbol{\alpha}_w, \boldsymbol{\alpha}_\beta) p(\boldsymbol{\alpha}_w) p(\boldsymbol{\alpha}_\beta)$ allows finding that the mode of the hyperparameter posterior $p(\boldsymbol{\alpha}_w, \boldsymbol{\alpha}_\beta | \boldsymbol{t})$ is approximately equivalent to maximizing the marginal likelihood $p(\boldsymbol{t} | \boldsymbol{\alpha}_w, \boldsymbol{\alpha}_\beta)$, which is also known as the evidence of the hyperparameters [26]:

$$p(\boldsymbol{t} | \boldsymbol{\alpha}_w, \boldsymbol{\alpha}_\beta) = \int p(\boldsymbol{t}, \boldsymbol{w}, \boldsymbol{\beta} \,|\, \boldsymbol{\alpha}_w, \boldsymbol{\alpha}_\beta)\, d\boldsymbol{w}\, d\boldsymbol{\beta}. \tag{24}$$

Applying the Laplace approximation in (16) to the term inside the integral, the evidence can be analytically determined as

$$
\begin{aligned}
p(\boldsymbol{t}|\boldsymbol{\alpha}_w, \boldsymbol{\alpha}_\beta) = \int & p(\boldsymbol{t}, \boldsymbol{w}^{MAP}, \boldsymbol{\beta}^{MAP} \,|\, \boldsymbol{\alpha}_w, \boldsymbol{\alpha}_\beta) \\
& \exp\left[-\frac{(\boldsymbol{w} - \boldsymbol{w}^{MAP})^T \boldsymbol{\Sigma}_w^{-1} (\boldsymbol{w} - \boldsymbol{w}^{MAP})}{2} \right. \\
& \left. -\frac{(\boldsymbol{\beta} - \boldsymbol{\beta}^{MAP})^T \boldsymbol{\Sigma}_\beta^{-1} (\boldsymbol{\beta} - \boldsymbol{\beta}^{MAP})}{2} \right] d\boldsymbol{w}\, d\boldsymbol{\beta}, \\
\approx & \, p(\boldsymbol{t}, \boldsymbol{w}^{MAP}, \boldsymbol{\beta}^{MAP} \,|\, \boldsymbol{\alpha}_w, \boldsymbol{\alpha}_\beta)\, (2\pi)^N \sqrt{|\boldsymbol{\Sigma}_w||\boldsymbol{\Sigma}_\beta|}. \qquad (25)
\end{aligned}
$$

This gives rise to the Gaussian distribution approximation of the posterior distribution for the weights

$$
\begin{aligned}
p(\boldsymbol{w}, \boldsymbol{\beta} \,|\, \boldsymbol{t}, \boldsymbol{\alpha}_w, \boldsymbol{\alpha}_\beta) &= \frac{p(\boldsymbol{t}, \boldsymbol{w}, \boldsymbol{\beta} \,|\, \boldsymbol{\alpha}_w, \boldsymbol{\alpha}_\beta)}{p(\boldsymbol{t} \,|\, \boldsymbol{\alpha}_w \boldsymbol{\alpha}_\beta)} \\
&\approx \mathcal{N}(\boldsymbol{w}^{MAP}, \boldsymbol{\Sigma}_w)\, \mathcal{N}(\boldsymbol{\beta}^{MAP}, \boldsymbol{\Sigma}_\beta). \qquad (26)
\end{aligned}
$$

Using evidence approximation in (25) yields the following objective function

$$
\begin{aligned}
\mathcal{C} = \mathcal{L} &+ N \log(2\pi) + \frac{1}{2} \log(|\boldsymbol{A}_w|) \\
&+ \frac{1}{2} \log(|\boldsymbol{A}_\beta|) + \frac{1}{2} \log(|\boldsymbol{\Sigma}_w|) + \frac{1}{2} \log(|\boldsymbol{\Sigma}_\beta|). \qquad (27)
\end{aligned}
$$

Ultimately, using the method presented in [3] and differentiating $\mathcal{C}$ with respect to $\log \alpha_w$ and $\log \alpha_\beta$, we obtain the following update equations

$$
\alpha_{w_i}^{new} = \frac{\gamma_{w_i}}{({w_i}^{MAP})^2}, \qquad \gamma_{w_i} = 1 - \alpha_{w_i} \Sigma_{w_{i,i}}, \qquad (28)
$$

$$
\alpha_{\beta_i}^{new} = \frac{\gamma_{\beta_i}}{({\beta_i}^{MAP})^2}, \qquad \gamma_{\beta i} = 1 - \alpha_{\beta_i} \Sigma_{\beta_{i,i}}, \qquad (29)
$$

where $\boldsymbol{\Sigma}_{w_{i,i}}$ and $\boldsymbol{\Sigma}_{\beta_{i,i}}$ denote the i^{th} diagonal element of covariance matrices $\boldsymbol{\Sigma}_w$ and $\boldsymbol{\Sigma}_\beta$, respectively. Re-estimation stops when maximum changes in two consecutive iterations are less than a small threshold value.

Therefore the proposed EM training procedure iterates between estimating the unknown weights using (20) to (23) and updating hyperparameters

Algorithm 1 Learning procedure for the proposed DL-ELM model

$\boldsymbol{\alpha}_{\beta}^{new} = [1, \ldots, 1]_{d\times 1}^{T}, \quad \boldsymbol{\alpha}_{w}^{new} = [1, \ldots, 1]_{(L+1)\times 1}^{T}$
function DL-ELM($\{t_i, \boldsymbol{x}_i\},\ i = 1, \ldots, N$)
  do
    $\boldsymbol{\alpha}_{\beta} = \boldsymbol{\alpha}_{\beta}^{new}, \quad \boldsymbol{\alpha}_{w} = \boldsymbol{\alpha}_{w}^{new}$
    Estimate SBELM block weights $\boldsymbol{w}^{MAP}$ from (20)
    Estimate SBELM covariance matrix $\boldsymbol{\Sigma}_{w}$ from (21)
    Estimate feature weights $\boldsymbol{\beta}$ from (22)
    Estimate feature covariance matrix $\boldsymbol{\Sigma}_{\beta}$ from (23)
    Prune hidden neurons with large $\boldsymbol{\alpha}_{w}$ weights
    Prune features with large $\boldsymbol{\alpha}_{\beta}$ weights
    Compute $\boldsymbol{\alpha}_{w}^{new}$ based on (28)
    Compute $\boldsymbol{\alpha}_{\beta}^{new}$ based on (29)
  while $\max_{\beta}(|\boldsymbol{\alpha}_{\beta}^{new} - \boldsymbol{\alpha}_{\beta}|) < \delta \wedge \max_{w}(|\boldsymbol{\alpha}_{w}^{new} - \boldsymbol{\alpha}_{w}|) < \delta$
return $\boldsymbol{w}$, $\boldsymbol{\beta}$
end function

(28) and (29), sequentially. During the learning steps, many of α_{w_i}s and α_{β_i}s tend to infinity, and thus the corresponding weights are shrunk toward zero and sparsity is realized in hidden and input layers. The outline of the DL-ELM approach is summarized in Alg. 1.

3. Experimental Results

In this section, the performance of the proposed DL-ELM is evaluated and compared with other state-of-the-art models. In all of the experiments, the simulations are carried out in MATLAB R2011 executed under Windows 7 on a 3 GHz CPU with 4 GB RAM PC.

3.1. Experiments on Benchmark Datasets

In the first set of experiments, the performance of DL-ELM is evaluated using a set of benchmark data sets from the UCI machine learning repository. The properties of the 24 evaluated datasets are listed in Table 1. The datasets are preprocessed by applying a linear scaling to each feature to the range $[-1, 1]$. The number of samples of covtype-binary is cut to the first 50,000 instances. Instances with unknown values are removed from all datasets.

Table 1: Description of the datasets used for the experiments

Dataset	# instances	# features
ijcnn1	191681	22
w8a	64702	300
covtype-binary	50000	54
adult	45222	14
a8a	32561	123
magic04	19020	22
mushroom	8124	21
krvskp	3196	36
splice	3175	60
madelon	2600	500
svmguide3	1284	21
German	1000	24
australian	690	14
Breast-cancer	683	10
WBC	569	30
Pima	532	7
congress	435	16
ionosphere	351	34
heart	270	13
Spect	267	22
Liver	345	6
Crabs	200	5
parkinsons	195	22
Colon	62	2000

For all of the datasets, the generalization performance of the methods is investigated using ten random holdout procedures with $4N/5$ instances for the training set and the remaining $N/5$ instances for the test set.

The experiment was first run with a varying number of initial hidden neurons L. For each number of hidden neurons L, the results are averaged over 10 holdout repetitions of train sets. In the experiments, the generation of hidden layer random parameters may have an impact on the performance. Hence, the process is repeated for the different seeds $s \in [1, 2, \ldots, 10]$ to generate uniformly random hidden parameters and the seed corresponding to the best mean performance is selected for predictions of the test sets.

The accuracy performance of the SBELM and DL-ELM methods is shown in Fig. 4. It can be observed from this figure that the accuracy performance of both models in most of the datasets are comparable. Furthermore, the sparse nature of SBELM and DL-ELM models causes the redundant hidden neurons to be deactivated during the training stage. Hence, for SBELM and DL-ELM models, the initial hidden neuron number used in Fig. 4 may not be equal to the number of active hidden neurons at the end of training. Fig. 5 shows the number of active hidden neurons for SBELM and DL-ELM at the end of training, against different initial number of hidden neurons. Fig. 6 shows the number of features selected by DL-ELM (remember that SBELM works on the complete input feature space).

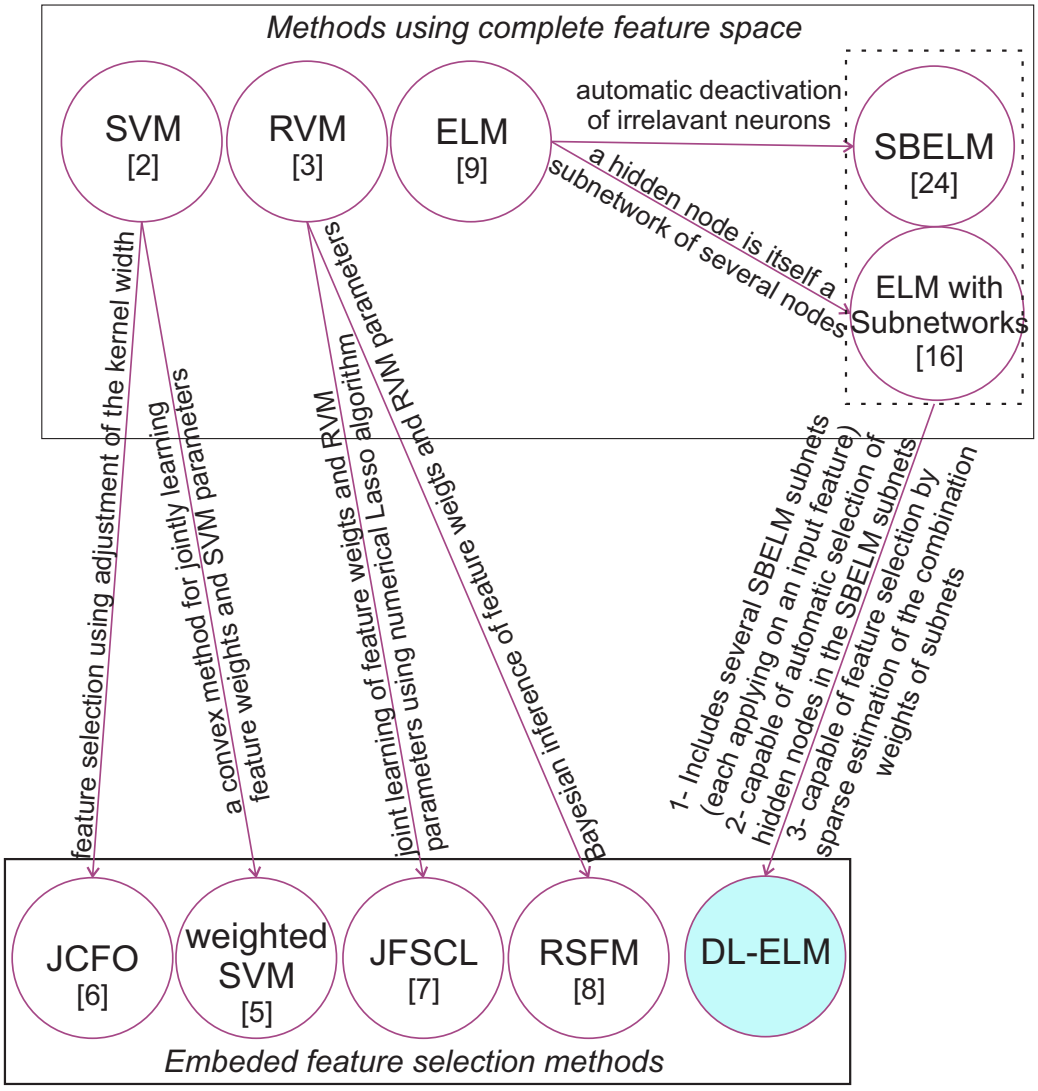

Figure 1: Relationship and distinction between our proposed method and other relevant techniques.

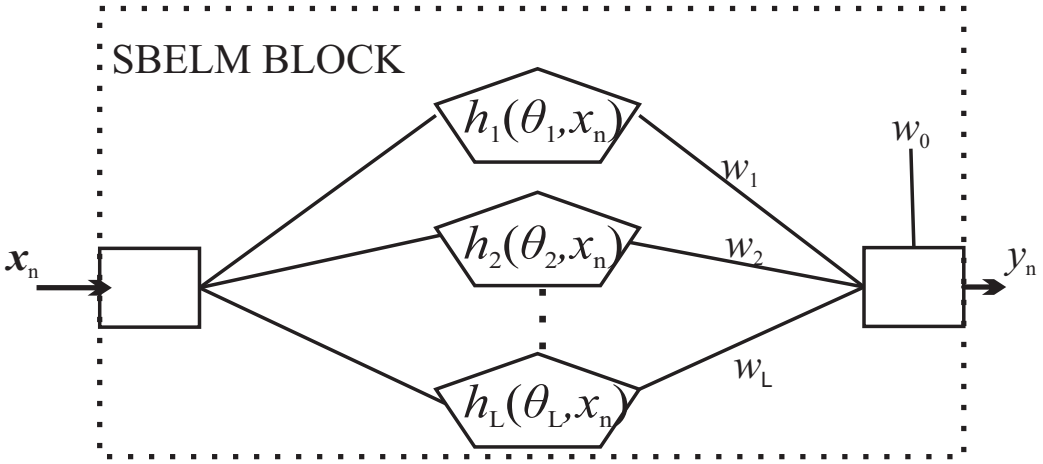

Figure 2: Standard SBELM model.

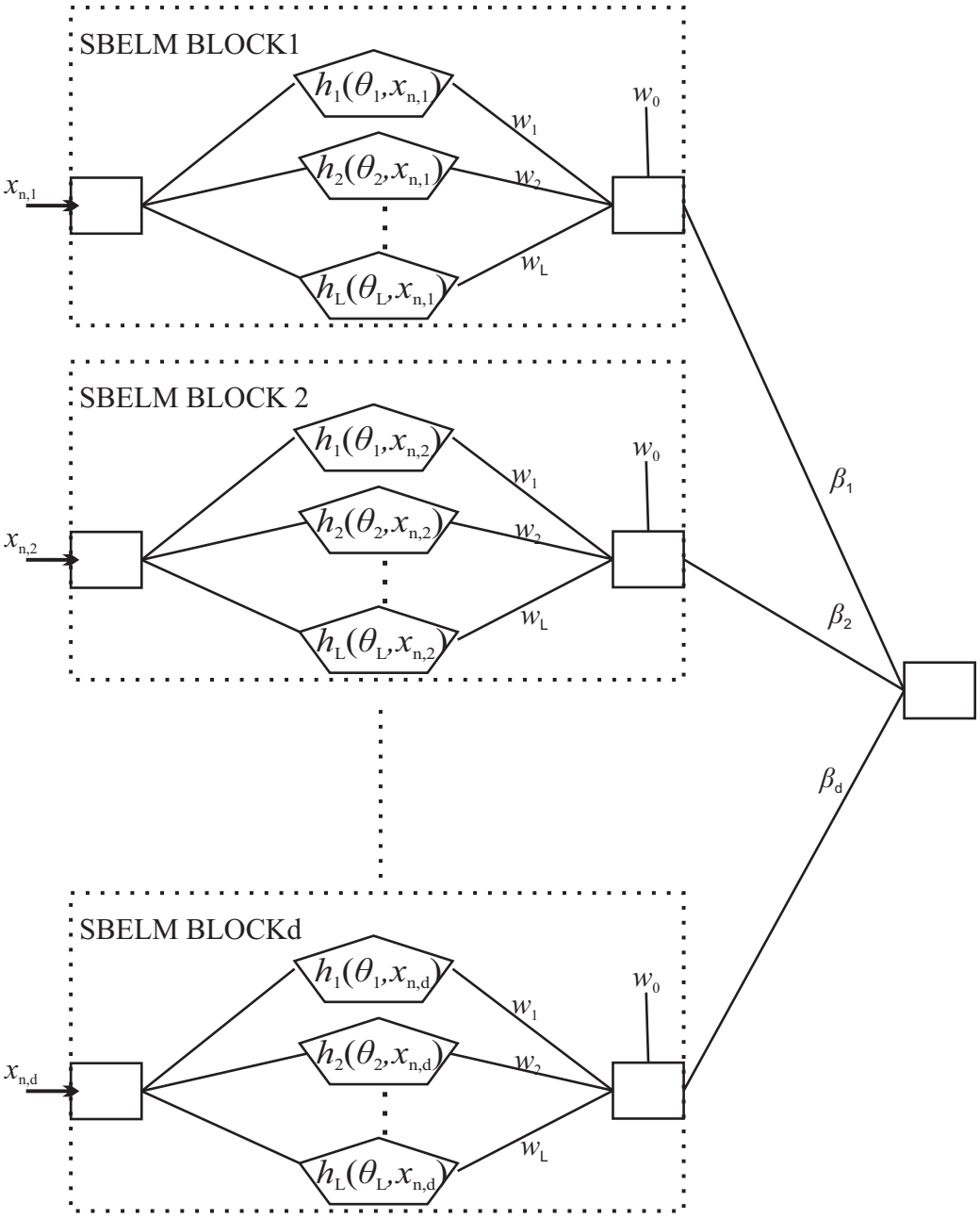

Figure 3: Proposed Double Layer ELM (DL-ELM) model

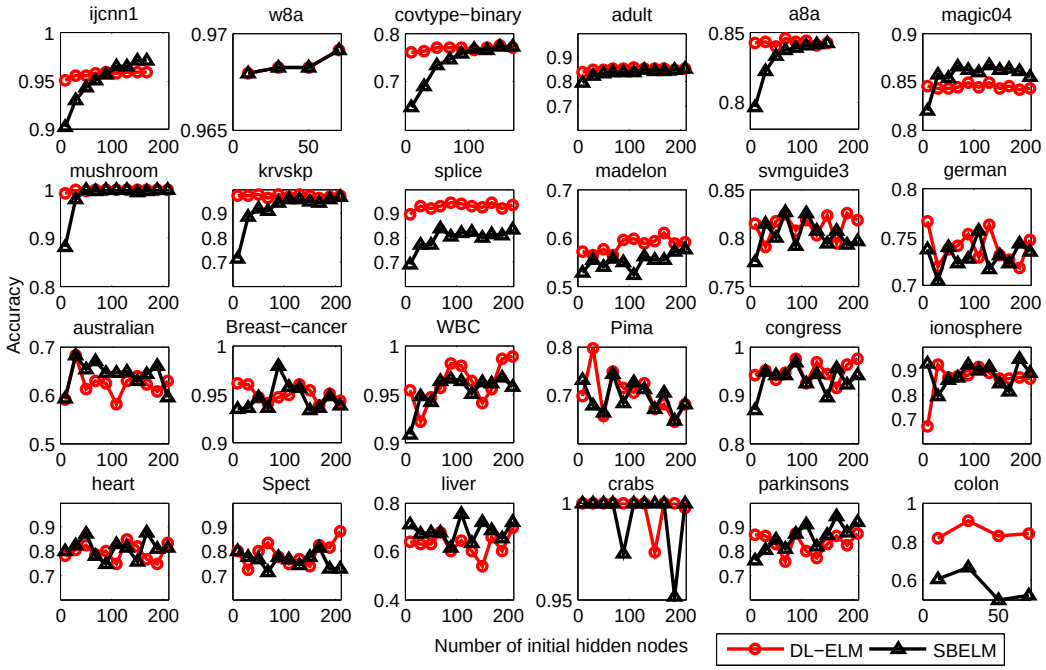

Figure 4: Accuracy of DL-ELM and SBELM for different initial number of hidden neurons

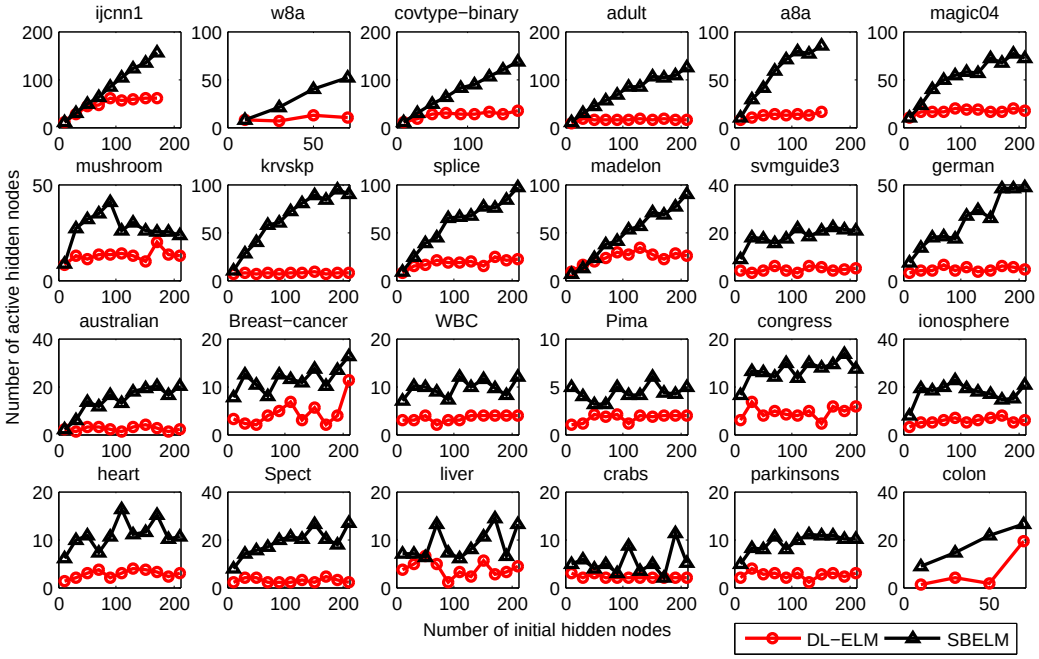

Figure 5: Number of final active hidden neurons with DL-ELM and SBELM for different initial number of hidden neurons

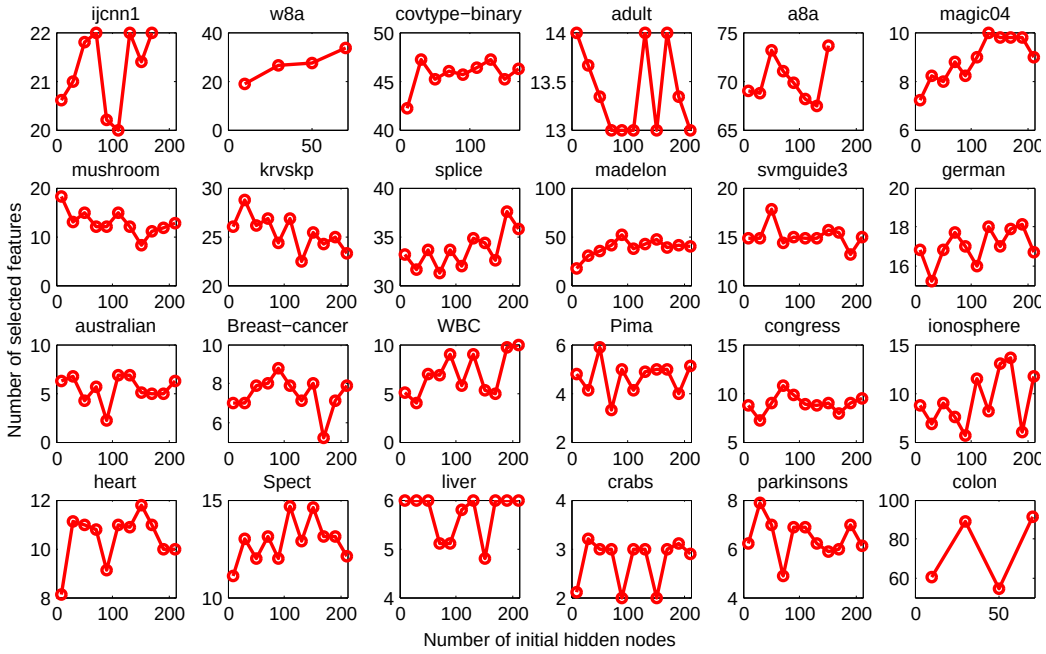

Figure 6: Number of selected features with DL-ELM for different initial number of hidden neurons

The best initial number of nodes in the hidden layer is then selected based on the training error. Then the accuracy performance as well as the number of final active neurons (AN) of the methods on the test set are reported in Table 2.

In order to check the statistical significance of the results, we conduct a pairwise Wilcoxon signed-rank test at a significance level of 0.05. The null hypothesis of the test is that the median performance of the SBELM is equal to that of the proposed DL-ELM, while the one-tailed alternate hypothesis is that the median performance of the SBELM method is worse than that of the DL-ELM. The counted wins / losses / ties across datasets are reported in Table 2.

One may notice from Table 2 that the proposed DL-ELM method offers the best accuracy rate on 12 data sets while the hidden layer sparsity performance of the proposed DL-ELM method is significantly better than SBELM on almost all datasets.

In order to compare the overall performance of the methods on all datasets, the one-tailed Wilcoxon signed-rank test is again conducted pairwisely on the performance of the two methods on all 24 datasets [27]. The p-values are reported in the last row of Table 2. As Wilcoxon p-values suggest, the accuracy of DL-ELM in the trained model is nearly the same as the accuracy for SBELM, while its hidden layer sparsity of DL-ELM significantly outperforms the SBELM method. Moreover, from Fig. 6, it can be seen that DL-ELM is able to reduce the input dimensionality significantly for many datasets. Therefore, the main advantages of the proposed DL-ELM method in comparison to the SBELM can be summarized as: 1) joint features selection and classifier learning, 2) better generalization and thus avoidance of overfitting, and 3) lower computational cost during the test stage.

Afterwards, the performance of the proposed DL-ELM method is compared with the other state-of-the-art algorithms in this field. For a fair comparison of our model with existing results in [6], the leave-one-out cross-validation procedure is used to compute the accuracy of the DL-ELM method on the gene expression Colon dataset. The accuracy performance of the proposed DL-ELM method is superior to most of the methods in Table 3 but not better than that of the JCFO method. However, as reported in [6], JCFO suffers from high computational complexity (compared to RVM and SVM) which makes it impractical for large datasets, while the computational complexity of the proposed DL-ELM method is at the same order of SBELM, and so it is much better than JCFO method.

Table 2: Comparison of the DL-ELM and the SBELM model. (The $\surd/\times$ symbol indicates that, according to the Wilcoxon signed rank test, the SBELM method is significantly better/worse than our proposed DL-ELM. The $\sim$ symbol stands for the methods without significance differences.) The p-values of the Wilcoxon signed rank test over all datasets are presented in the last row.

Datasets	Accuracy		#AN	
	DL-ELM	SBELM	DL-ELM	SBELM
ijcnn1	0.96	$0.97\surd$	60.00	$135.20\times$
w8a	0.97	$0.97\sim$	7.40	$7.20\sim$
cov-binary	0.77	$0.77\times$	28.60	$137.20\times$
adult	0.85	$0.85\times$	18.67	$108.00\times$
a8a	0.84	$0.84\surd$	12.20	$84.60\times$
magic04	0.84	$0.86\surd$	19.40	$72.40\times$
mushroom	1.00	$1.00\times$	12.90	$35.00\times$
krvskp	0.97	$0.97\times$	7.90	$89.40\times$
splice	0.93	$0.83\times$	22.40	$97.20\times$
madelon	0.59	$0.57\times$	27.00	$76.20\times$
svmguide3	0.81	$0.80\times$	4.00	$20.80\times$
german	0.72	$0.72\surd$	6.80	$48.10\times$
australian	0.58	$0.63\surd$	1.20	$19.40\times$
Breast-can	0.94	$0.94\times$	11.40	$16.40\times$
WBC	0.99	$0.96\times$	4.00	$9.70\times$
Pima	0.72	$0.65\times$	2.00	$4.20\times$
congress	0.92	$0.94\surd$	3.90	$13.80\times$
ionosphere	0.87	$0.82\times$	7.80	$14.50\times$
heart	0.78	$0.82\surd$	3.80	$10.60\times$
Spect	0.72	$0.73\sim$	3.90	$17.90\times$
liver	0.60	$0.65\surd$	2.20	$6.60\times$
crabs	1.00	$1.00\surd$	2.90	$5.00\times$
parkinsons	0.83	$0.92\surd$	2.80	$10.10\times$
colon	0.82	$0.52\times$	1.20	$26.30\times$
win/tie/loss	-	12/2/10	-	23/1/0
p-value	-	0.3484	-	0.0000

Table 3: Accuracy of diagnostic classification for gene expression Colon dataset

Classifier	accuracy (%)
Adaboost (Decision stumps) [28]	72.6
SVM (Linear Kernel) [28]	77.4
SVM (Quadratic Kernel) [28]	74.2
RSFM [8]	88.7
Logistic regression [6]	71.0
RVM [6]	88.7
Sparse probit regression [6]	88.7
Sparse probit regression (Linear Kernel) [6]	91.9
Sparse probit regression (Quadratic Kernel) [6]	84.6
JCFO (Linear Kernel) [6]	96.8
JCFO (Quadratic Kernel) [6]	88.7
DL-ELM	91.9

Furthermore, similarly to the reported experiments in the literature [6, 7, 29], we used 200, 300, and 80 samples for training and 332, 269, and 120 samples for testing, in the Pima Indians diabetes, Wisconsin breast cancer (WBC), and Crabs datasets, respectively. The accuracy measure for the proposed method and other state-of-the-art algorithms is reported in Table 4. The performance of the proposed method in Table 4 is comparable to the best results for each dataset. DL-ELM benefits from the sparsity in both input layer and hidden layer. DL-ELM selected 5 out of 100 hidden neurons and 5 out of 7 input features in the Pima dataset. For the WBC dataset, 6 out of 100 hidden neurons and 7 out of 30 input features are selected, while for the Crabs dataset 2 out of 80 hidden neurons and 3 out of 5 features have been retained with the DL-ELM approach.

Table 4: Number of instances erroneously classified for Pima, WBC, and Crabs datasets

Classifier	Pima	WBC	Crabs
Linear discriminant [30]	67	19	3
Neural network [31]	75	N/A	3
Gaussian process [31]	67	8	3
SVM [30]	64	9	4
Logistic regression [31]	66	N/A	4
RVM [3]	65	9	0
RSFM [8]	66	9	0
Sparse probit regression [29]	62	9	0
JCFO [6]	64	8	0
DL-ELM	65	9	0

Table 5: Performance of DL-ELM, weighted SVM, and RSFM on MNIST dataset for the digits 4 Versus 9 and 8 Versus 0 digits. AN: number of active neurons; AF: number of active features; SV: number of support vectors; SF: number of support features; RV: number of relevance vectors; RF: number of relevance features.

		DL-ELM				Weighted SVM				RSFM			
	#	Acc. (%)	Time	AN	AF	Acc. (%)	Time	SV	SF	Acc. (%)	Time	RV	RF
9 versus 4	100	89.31	0.63	8	38	89.50	2.38	55	60	88.25	0.72	9	42
	200	89.83	1.21	16	52	89.95	6.83	72	76	88.80	1.42	18	84
	300	91.86	1.82	17	86	89.85	8.22	56	108	91.46	2.31	20	112
	400	90.42	2.01	18	72	90.96	13.21	48	94	90.06	2.34	20	84
	600	90.58	2.99	26	71	90.10	25.36	71	127	90.39	3.49	31	84
	1000	91.59	5.18	27	83	93.92	68.65	81	97	91.62	5.77	33	98
8 versus 0	100	97.62	0.27	5	32	97.75	2.76	21	24	97.34	0.36	5	56
	200	97.35	0.38	5	35	97.29	6.38	23	25	97.13	0.59	6	70
	300	97.84	0.88	5	29	98.47	10.72	26	29	97.65	1.18	5	42
	400	98.80	1.02	7	27	98.16	17.30	26	56	98.41	1.42	9	42
	600	98.15	2.71	5	25	98.87	32.43	26	57	97.96	3.20	5	28
	1000	97.84	4.61	6	24	97.29	86.41	41	60	97.75	5.22	6	28

3.2. Results on Handwritten Digit Recognition

Our second set of experiments were conducted on the large publicly available MNIST dataset. The dataset contains $60,000$ handwritten examples of digits 0 to 9 for training and $10,000$ examples for testing. Each class of a digit contains 6000 training instances and 1000 instances for testing. Similarly to the reported experiments in [8], two separate evaluations are considered. The first one is the evaluation of the DL-ELM classifier for discriminating the digits 8 and 0, where a portion of the dataset composed of digits 8 and 0 is selected in order to conduct the evaluation. Similarly, the second evaluation involves the more challenging digits 9 and 4. In order to investigate the ability of the proposed DL-ELM method to prune the irrelevant input features, 200 irrelevant normally distributed features are added to each sample, increasing the input space to a total of 984 features.

We split each dataset into a training (with varying number of training samples N as shown in Table 5) and test (2000 instances) portion. Moreover, another N random samples are chosen as validation in order to tune the number of initial hidden nodes and the seed for generating random parameters of the hidden layer in the DL-ELM method. The average performance of the DL-ELM method on 20 repetitions of the described training/test procedure is presented in Table 5. Also presented in Table 5 are the results reported in [8] for the weighted SVM and RSFM models.

The experiments are executed on a similar computer to the one used in [8]. The results in Table 5 show that the accuracy (Acc) of the DL-ELM in discriminating 4 and 9 is comparable with the weighted SVM and RSFM. The number of final active neurons (AN) in the DL-ELM is smaller than the number of support vectors (SV) in the weighted SVM and comparable to the number of relevance vectors (RV) in the RSFM model, suggesting that DL-ELM can achieve a sparser model compared to the weighted SVM model. Moreover, the number of active features (AF) for the proposed DL-ELM method is smaller than the number of relevance features (RF) for RSFM and support features (SF) for weighted SVM models, implying that the proposed DL-ELM model can achieve to a better feature selection property.

Comparison of the training times of both algorithms, demonstrate that the training time with DL-ELM is smaller than for the two other methods, particularly for larger training sets. The experimental results (Figs. 3-5) indicate that , DL-ELM tends to reach the best performance at a small number of hidden neurons L. Hence, by setting a small initial L, the training time of Dl-ELM can be significantly reduced, while still obtaining an accurate

model. From Table 5, it can be seen that the evaluation of the methods to distinguish the digit 8 versus digit 0 gives the same result based on the accuracy and complexity properties of the methods.

4. Conclusion and Future Work

In this paper, we introduced a novel neural model with two layers which is sparse in both the hidden layer and input layer. A set of sparse Bayesian ELM subnets with shared weights construct the hidden layer. Each subnet is applied on a single feature. The output of the subnets are combined in the second layer with a sparse set of weights which gives rise to input layer sparsity. As a result of sparseness in the input layer, the proposed DL-ELM model is capable of automatic input feature selection. We presented the Bayesian inference of the proposed model and the learning algorithm based on the EM procedure. The proposed DL-ELM model was evaluated on 24 real benchmark datasets and one handwritten digits dataset. While the accuracy of the proposed DL-ELM method is comparable to that of the SBELM model, its hidden layer sparsity outperforms that of the SBELM, not to mention that DL-ELM is sparse in the input layer by successfully pruning irrelevant features. The DL-ELM method in comparison to the other state-of-the-art methods namely JCFO, JFSCL, RSFM, and weighted SVM, achieves compact model while is able to accurately classify the datasets.

In many practical applications, different neural and kernel based models are proposed which are not sparse in the input feature space and depend on all input explanatory variables. Extending the DL-ELM model for the classification problem to other applications such as those presented in [12, 32] is our future plan.

Acknowledgements

The authors are grateful to Annette Schwerdtfeger for proofreading this manuscript.

[1] P. Shenoy, K. J. Miller, J. G. Ojemann, R. P. Rao, Generalized features for electrocorticographic BCIs, IEEE Transactions on Biomedical Engineering 55 (1) (2008) 273–280.

[2] V. Vapnik, S. E. Golowich, A. J. Smola, Support vector method for function approximation, regression estimation and signal processing, in: Neural Information Processing Systems (NIPS), 1997, pp. 281–287.

[3] M. E. Tipping, Sparse Bayesian learning and the relevance vector machine, Journal of Machine Learning Research 1 (2001) 211–244.

[4] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, V. Vapnik, Feature selection for SVMs, in: Neural Information Processing Systems (NIPS), 2000, pp. 668–674.

[5] M. H. Nguyen, F. De la Torre, Optimal feature selection for support vector machines, Pattern Recognition 43 (3) (2010) 584–591.

[6] B. Krishnapuram, A. Harternink, L. Carin, M. A. Figueiredo, A Bayesian approach to joint feature selection and classifier design, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (9) (2004) 1105–1111.

[7] À. Lapedriza, S. Seguí, D. Masip, J. Vitrià, A sparse Bayesian approach for joint feature selection and classifier learning, Pattern Analysis and Applications 11 (3-4) (2008) 299–308.

[8] Y. Mohsenzadeh, H. Sheikhzadeh, A. M. Reza, N. Bathaee, M. M. Kalayeh, The relevance sample-feature machine: A sparse Bayesian learning approach to joint feature-sample selection, IEEE Transactions on Cybernetics 43 (6) (2013) 2241–2254.

[9] G.-B. Huang, H. Zhou, X. Ding, R. Zhang, Extreme learning machine for regression and multiclass classification, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 42 (2) (2012) 513–529.

[10] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (1) (2006) 489–501.

[11] G.-B. Huang, An insight into extreme learning machines: random neurons, random features and kernels, Cognitive Computation 6 (3) (2014) 376–390.

[12] F. Kiaee, H. Sheikhzadeh, S. E. Mahabadi, Sparse bayesian mixed-effects extreme learning machine, an approach for unobserved clustered heterogeneity, Neurocomputing 175 (2016) 411–420.

[13] N. Wang, M. J. Er, M. Han, Parsimonious extreme learning machine using recursive orthogonal least squares, Neural Networks and Learning Systems, IEEE Transactions on 25 (10) (2014) 1828–1841.

[14] J. Cao, T. Chen, J. Fan, Landmark recognition with compact bow histogram and ensemble elm, Multimedia Tools and Applications 75 (5) (2016) 2839–2857.

[15] J. Cao, Y. Zhao, X. Lai, M. E. H. Ong, C. Yin, Z. X. Koh, N. Liu, Landmark recognition with sparse representation classification and extreme learning machine, Journal of The Franklin Institute 352 (10) (2015) 4528–4545.

[16] Y. Yang, Q. Wu, Extreme learning machine with subnetwork hidden nodes for regression and classification, IEEE Transactions on Cybernetics In-press (2015) 1–14.

[17] M. Lin, Q. Chen, S. Yan, Network In Network, ArXiv e-printsarXiv:1312.4400.

[18] Y.-L. He, Z.-Q. Geng, Y. Xu, Q.-X. Zhu, A hierarchical structure of extreme learning machine (helm) for high-dimensional datasets with noise, Neurocomputing 128 (2014) 407–414.

[19] Y. Yang, Y. Wang, X. Yuan, Bidirectional extreme learning machine for regression problem and its learning effectiveness, Neural Networks and Learning Systems, IEEE Transactions on 23 (9) (2012) 1498–1505.

[20] D. Chyzhyk, A. Savio, M. Graña, Evolutionary elm wrapper feature selection for alzheimer's disease cad on anatomical brain mri, Neurocomputing 128 (2014) 73–80.

[21] A. Tipping, A. Faul, Analysis of sparse bayesian learning, Advances in neural information processing systems 14 (2002) 383–389.

[22] D. P. Wipf, B. D. Rao, Sparse bayesian learning for basis selection, Signal Processing, IEEE Transactions on 52 (8) (2004) 2153–2164.

[23] K. I. Wong, C. M. Vong, P. K. Wong, J. Luo, Sparse Bayesian extreme learning machine and its application to biofuel engine performance prediction, Neurocomputing 149 (2015) 397–404.

[24] J. Luo, C.-M. Vong, P.-K. Wong, Sparse Bayesian extreme learning machine for multi-classification, IEEE Transactions on Neural Networks and Learning Systems 25 (4) (2014) 836–843.

[25] I. T. Nabney, Efficient training of RBF networks for classification, International Journal of Neural Systems 14 (3) (2004) 201–208.

[26] D. J. MacKay, The evidence framework applied to classification networks, Neural Computation 4 (5) (1992) 720–736.

[27] J. Demšar, Statistical comparisons of classifiers over multiple data sets, The Journal of Machine Learning Research 7 (2006) 1–30.

[28] A. Ben-Dor, L. Bruhn, N. Friedman, I. Nachman, M. Schummer, Z. Yakhini, Tissue classification with gene expression profiles, Journal of Computational Biology 7 (3-4) (2000) 559–583.

[29] M. A. Figueiredo, Adaptive sparseness for supervised learning, IEEE Transactions on Pattern Analysis and Machine Intelligence 25 (9) (2003) 1150–1159.

[30] M. Seeger, Bayesian model selection for support vector machines, Gaussian processes and other kernel classifiers, in: Neural Information Processing Systems (NIPS), 2000, pp. 603–609.

[31] C. K. Williams, D. Barber, Bayesian classification with Gaussian processes, IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (12) (1998) 1342–1351.

[32] F. Kiaee, H. Sheikhzadeh, S. Eftekhari Mahabadi, Relevance vector machine for survival analysis, IEEE Transactions on Neural Networks and Learning Systems, unpublished.