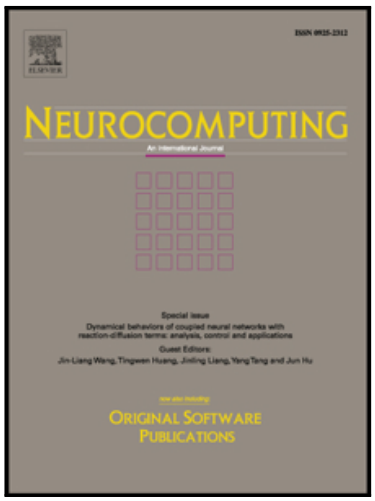


Sparse Coding Extreme Learning Machine for Classification

Yuanlong Yu, Zhenzhen Sun

PII: S0925-2312(17)30207-2
DOI: 10.1016/j.neucom.2016.06.078
Reference: NEUCOM 18006

To appear in: *Neurocomputing*

Received date: 30 September 2015
Revised date: 25 May 2016
Accepted date: 16 June 2016

Please cite this article as: Yuanlong Yu, Zhenzhen Sun, Sparse Coding Extreme Learning Machine for Classification, *Neurocomputing* (2017), doi: 10.1016/j.neucom.2016.06.078

Sparse Coding Extreme Learning Machine for Classification

Yuanlong Yu[a], Zhenzhen Sun[a]

[a] *College of Mathematics and Computer Science, Fuzhou University Fuzhou, Fujian, 350116, China*

Abstract

As one of supervised learning algorithms, extreme learning machine (ELM) has been proposed for training single-hidden-layer feedforward neural networks and shown great generalization performance. ELM randomly assigns the weights and biases between input and hidden layers and only learns the weights between hidden and output layers. Physiological research has shown that neurons at the same layer are laterally inhibited to each other such that outputs of each layer are sparse. However, it is difficult for ELM to accommodate the lateral inhibition by directly using random feature mapping. Therefore, this paper proposes a sparse coding ELM (ScELM) algorithm, which can map the input feature vector into a sparse representation. In this proposed ScELM algorithm, an unsupervised way is used for sparse coding and dictionary is randomly assigned rather than learned. Gradient projection based method is used for the sparse coding. The output weights are trained through the same supervised way as ELM. Experimental results on the benchmark datasets have shown that this proposed ScELM algorithm can outperform other state-of-the-art methods in terms of classification accuracy.

Keywords: `Sparse coding, extreme learning machine, gradient projection`

*Corresponding author: Yuanlong Yu, Email: yu.yuanlong@fzu.edu.cn.

**This work is supported by National Natural Science Foundation of China under grant 61473089.

1. Introduction

During the past decades, neural networks are widely studied in the areas of machine learning, pattern recognition and robotics since they are able to approximate complex nonlinear functions so as to provide much higher classification accuracy. Many learning algorithms have been proposed for training neural networks, for example, support vector machine (SVM) [1, 2] for single-hidden-layer neural networks (SLNN), back-propagation (BP) algorithm and deep learning algorithms [3, 4, 5, 6, 7] for multiple-hidden-layer neural networks (MLNN).

SVM can be seen as a training method for SLNN based on standard optimization method by maximizing the margin between two classes. However, it is difficult for SVM to deal with large-scale data since the quadratic programming required to obtain the optimal solution is computationally expensive when the number of training samples is too large.

Further efforts have been put on training MLNNs. BP algorithm is a pioneer for this type of efforts. It minimizes the training errors based on gradient descent strategy and the errors are back-propagated from the output layer to previous hidden layers. However, in real applications, BP algorithm has not shown great performance for neural networks with many hidden layers. This is because that the gradients become smaller and smaller with the back-propagation process from the top to lower layers such that the updates are weak at lower layers. Recently, several deep learning algorithms have been proposed, e.g., deep Boltzmann machine (DBM) [4, 6, 7], deep belief network (DBN) [5], convolutional neural network (CNN) [3], stacked denoise autoencoder (SDAE) [8, 9, 10] and stacked sparse autoencoder (SSAE) [11, 12]. The underlying idea of deep learning is that feature extraction and classification are combined together in a unified MLNN architecture. In these algorithms, learning of connection weights is basically divided into two processes. The first one is bottom-up layer-wise pre-training through unsupervised ways with a common objective function that output and input are as close as possible between two neighboring layers. For example, DBM performs Gibbs sampling to maximize the log-likelihood of training

data and SSAE performs self-taught sparse coding. The second one is top-down fine-tuning of connection weights through a supervised way based on gradient descent strategy. However, the gradient descent based pre-training and fine-tuning is likely to converge to a local optimum.

Recently, extreme learning machine (ELM) was proposed for training SLNNs [13, 14, 15]. One contribution of ELM is that the weights and bias between input and hidden layers are randomly generated such that only the weights between hidden and output layers require training. The other contribution of ELM is that it obtains an optimal output weights by minimizing not only the training errors but also the norm of output weights such that better generalization performance is achieved [16]. This objective function is solved by using Lagrange multiplier method. Theoretically, ELM can obtain a global optimum [17] and therefore it is unlikely to fall into a local optimum. In terms of computation, the training cost of ELM is much lower than other state-of-the-art learning methods.

However, it is difficult to accommodate the lateral inhibition between neurons by directly using random feature mapping in ELM. Physiological research has shown that neurons at the same layer are laterally inhibited to each other such that the outputs of each layer are sparse [18]. Therefore, this paper proposes a sparse coding ELM (ScELM) algorithm which uses sparse coding technique to map the inputs to the hidden layer instead of the random mapping used by ELM. The gradient projection (GP) based method with l_1 norm optimization [19] is used in the encoding stage while the output weights between hidden and output layers are learned by using Lagrange multiplier algorithm. The contribution of this proposed ScELM is that the sparsity makes hidden-layer feature representations more salient and distinctive resulting that they can contribute more for classification.

Some pioneer work has been proposed by combining l_1 norm optimization with ELM. One method uses l_1 norm optimization to obtain the sparse output weights [20], but hidden-layer feature representations are not sparse. Given original features, another method first calculates their sparse representations and then use such sparse representations as the inputs of ELM based SLNN [21].

In other words, feature's sparse coding routine is beyond the neural network. Compared with the above existing methods, this proposed ScELM algorithm uses sparse mapping instead of random mapping between input and hidden layers. It is important to note that randomness are somewhat remained in the sense that the based vectors (i.e., directory) for sparse coding are randomly assigned in the proposed ScELM.

The remainder of this paper is organized as follows. Section 2 reviews some related work in sparse coding. Section 3 presents details of this proposed ScELM algorithm. Experiment results are shown in section 4.

2. Related Work in Sparse Coding

2.1. Sparse Coding Algorithms

By exploring the receptive fields of simple neurons in the visual stripe cortex of cats, Hubel and Wiesel posited that the receptive field of primary visual cortex (i.e., V1 neurons) can produce a sparse representation for visual signal [22]. The electrophysiological experiments further validated the sparse coding principle existed in the visual cortex [23]. These findings inspired engineering community to develop sparse coding algorithms for signal processing.

There have been various algorithms which attempt to encode signal's sparse representation by using optimization techniques. Most of them seek the sparse codes with the fewest number of non-zero coefficients given a dictionary by minimizing l_0 norm. Unfortunately, l_0 norm optimization problem is NP-hard.

Two types of methods have been proposed to find acceptable suboptimal solutions to the aforementioned optimization problem. The first type employs iterative greedy strategy to build up sparse codes [24]. The canonical examples of such greedy strategy are known as matching pursuit (MP) [25] and orthogonal matching pursuit (OMP) [26]. The underlying idea of MP algorithm is to find a dictionary element (i.e., atom) that can best approximate the current signal residual at each iteration. The sparse coefficient corresponding to such

atom is represented as the inner-product between this atom and signal residual. Then the residual is updated by removing the item reconstructed from this sparse coefficient. The iterative process stops until some convergence condition is satisfied. The OMP algorithm is a refinement of MP in the sense that it can guarantee that the selected atoms and the current residual are orthogonal at each iteration. As a result, OMP algorithm can converge much faster than MP algorithm.

The second type replaces l_0 norm by l_1 norm, which is called basis pursuit strategy [27]. It has been proved that basis pursuit strategy can obtain a similar solution to the l_0 norm optimization problem if the signal's sparsity is very close to the number of non-orthogonal atoms [28]. For noisy signals, a modified algorithm called basis pursuit de-noising (BPDN) [27] was proposed to make a tradeoff between sparsity and reconstruction error. BPDN algorithm can achieve the sparsest approximation of l_0 norm minimization given a reconstruction quality. Many extended algorithms inspired by BPDN were further proposed, e.g., gradient projection (GP) [19] and feature-sign search (FSS) [29]. BPDN algorithm is much more effective and computationally efficient compared against the greedy strategy based methods, so this algorithm and its extensions have been widely applied.

2.2. Applications of Sparse Coding

Sparse coding techniques have been used for feature extraction in recent years. A sparse representation coding (SRC) algorithm was proposed for face recognition [30]. The SRC algorithm initializes the dictionary by using all training samples and there is no further training for dictionary. Given a new test sample, it would approximately lie in the linear span of the training samples from the same class which the test sample belongs to. Thus its coefficient is sparse, i.e., most of the coefficients are zeros. Even in the case of noise, occlusion and corruption, SRC algorithm can also achieve high recognition accuracy. The training process of SRC is also much faster than other face recognition methods. Other object detection and recognition methods are also proposed by

using sparse coding [31, 32].

A sparse coding spatial pyramid matching (ScSPM) [33] method was also proposed for image feature extraction. It is an extension of the spatial pyramid matching (SPM) algorithm by generalizing vector quantization to sparse coding. It then uses a linear SVM classifier for classification. The experimental results have shown that the sparse coding can significantly outperform other types of feature mapping in terms of recognition accuracy.

Sparse coding is also integrated into deep neural networks, e.g, SSAE [11]. SSAE uses an unsupervised learning algorithm for feature extraction by imposing a sparsity constraint on the hidden units.

3. Exposition of the Proposed ScELM Algorithm

3.1. Architecture

As shown in Fig. 1, this proposed ScELM aims to train a single-hidden-layer neural network. Between input and hidden layers, it uses sparse coding technique to map input features into a mid-level feature space. Given an input feature vector, the hidden layer outputs its sparse representation. Since the dictionary used for sparse coding is randomly assigned using uniform distribution in this proposed ScELM algorithm, training is not required between input and hidden layers. The calculation between input and hidden layers is called *encoding stage*. In this paper, GP algorithm [19] is used to calculate sparse codes. The output weights between hidden and output layers are learned in a supervised manner by minimizing the training error as well as the norm of output weights.

3.2. Encoding Stage

In this stage, GP algorithm is used to calculate the sparse representations of input features since the GP algorithm is faster than other state-of-the-art approaches, e.g., iterative shrinkage-threshold (IST) algorithm [34, 35] and recently proposed l_1_ls package [36].

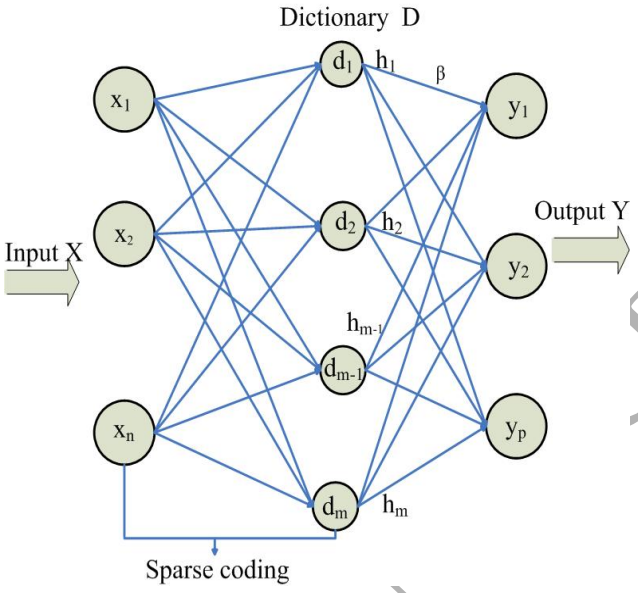


Figure 1: Framework of this proposed ScELM algorithm.

The GP algorithm aims to minimize the reconstruction error as well as l_1 norm of the sparse representation given dictionary $\mathbf{D}$:

$$\textbf{Minimize:} \quad \frac{1}{2}\|\mathbf{x} - \mathbf{D}\mathbf{h}\|_2^2 + \lambda\|\mathbf{h}\|_1 \tag{1}$$

where $\mathbf{x} \in R^n$ is the input feature whose dimension number is n, $\mathbf{D} = [\mathbf{d}_1, \mathbf{d}_2, ..., \mathbf{d}_m] \in R^{n\times m}$ is the over-complete dictionary that includes m atoms, $\mathbf{h} \in R^m$ is the sparse representation of $\mathbf{x}$ and λ is the regularization factor. It is important to note that $m \gg n$ in order to achieve the sparsity of input feature.

The above cost function as shown in (1) can be expressed as a quadratic optimization program. This is implemented by splitting the vector $\mathbf{h}$ into positive and negative parts:

$$\mathbf{h} = \mathbf{u} - \mathbf{v}, \tag{2}$$

where $\mathbf{u} \in R^m$ and $\mathbf{v} \in R^m$.

The above equation is satisfied when $u_i = (h_i)_+$ and $v_i = (-h_i)_+$ for all $i = 1, 2, ..., m$, where $(h_i)_+ = max\{0, h_i\}$. Thus $\|\mathbf{h}\|_1 = \mathbf{1}_m^T\mathbf{u} + \mathbf{1}_m^T\mathbf{v}$, where

$\mathbf{1}_m = [1, 1, ..., 1]^T$. As a result, (1) can be rewritten as the following bound-constrained quadratic program (BCQP):

$$\begin{aligned} \min_{\mathbf{u},\mathbf{v}} \quad & \frac{1}{2}||\mathbf{x} - \mathbf{D}(\mathbf{u} - \mathbf{v})||_2^2 + \lambda \mathbf{1}_m^T \mathbf{u} + \lambda \mathbf{1}_m^T \mathbf{v}, \\ \text{s.t.} \quad & \mathbf{u} \geq 0, \mathbf{v} \geq 0. \end{aligned} \tag{3}$$

Problem (3) can be further written in a standard BCQP format:

$$\begin{aligned} \min_{\mathbf{z}} \quad & \mathbf{c}^T \mathbf{z} + \frac{1}{2} \mathbf{z}^T \mathbf{B} \mathbf{z} \equiv F(\mathbf{z}) \\ \text{s.t.} \quad & \mathbf{z} \geq 0. \end{aligned} \tag{4}$$

where

$$\begin{aligned} \mathbf{z} = \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix}, \quad \mathbf{b} = \mathbf{D}^T \mathbf{x}, \quad \mathbf{c} = \lambda \mathbf{1}_{2m} + \begin{bmatrix} -\mathbf{b} \\ \mathbf{b} \end{bmatrix}, \\ \mathbf{B} = \begin{bmatrix} \mathbf{D}^T \mathbf{D} & -\mathbf{D}^T \mathbf{D} \\ -\mathbf{D}^T \mathbf{D} & \mathbf{D}^T \mathbf{D} \end{bmatrix}. \end{aligned} \tag{5}$$

In the GP algorithm, given two step factors α and γ, $\mathbf{z}$ can be iteratively updated as follows

$$\mathbf{w}^k = (\mathbf{z}^k - \alpha^k \nabla F(\mathbf{z}^k))_+ \tag{6}$$

$$\mathbf{z}^{k+1} = \mathbf{z}^k + \gamma^k (\mathbf{w}^k - \mathbf{z}^k). \tag{7}$$

where k denotes the iteration index.

The basic implementation of GP algorithm searches $\mathbf{z}^k$ in each iteration along the negative gradient orientation, i.e., $-\nabla F(\mathbf{z}^k)$, and performs a backtracking line search until a sufficient decrease of F is attained.

An initial guess of α^k can be determined as follows. First, GP algorithm defines a vector $\mathbf{g}^k = [g_1^k, ..., g_j^k, ...]^T$:

$$g_i^k = \begin{cases} \nabla F(z^k)_i & \text{if } z_i^k > 0 \text{ or } \nabla F(z^k)_i < 0 \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

Then the initial guess can be computed as

$$\alpha_0^k = \frac{(\mathbf{g}^k)^T \mathbf{g}^k}{(\mathbf{g}^k)^T \mathbf{B} \mathbf{g}^k}. \tag{9}$$

Algorithm 1 Gradient projection algorithm for encoding stage

1: **(Initialization)** Given $\mathbf{z}^0$, choose parameters $\rho \in (0,1)$ and $\mu \in (0, \frac{1}{2})$; Set $k = 0$;

2: Compute α_0^k using (9);

3: **(Backtracking Line Search)** Choose α^k to be the first number in the sequence $\alpha_0^k, \rho\alpha_0^k, \rho^2\alpha_0^k, ...$ that can satisfy $F((\mathbf{z}^k - \alpha^k \nabla F(\mathbf{z}^k))_+) \leq F(\mathbf{z}^k) - \mu \nabla F(\mathbf{z}^k)^T(\mathbf{z}^k - (\mathbf{z}^k - \alpha^k \nabla F(\mathbf{z}^k))_+)$; Set $\mathbf{z}^{k+1} = (\mathbf{z}^k - \alpha^k \nabla F(\mathbf{z}^k))_+$;

4: If $F(\mathbf{z}^k) - F(\mathbf{z}^{k+1}) < \epsilon$, approximate solution $\mathbf{z}^{k+1}$ is achieved; Otherwise, set $k = k + 1$ and return to Step 2.

The complete routine of GP algorithm is shown in Algorithm 1.

The parameter λ in (1) is chosen as suggested in [36]:

$$\lambda = 0.1||\mathbf{D}^T\mathbf{x}||_\infty \tag{10}$$

It is important to note that the zero vector is the unique optimal solution of (1) for $\lambda \geq 0.1||\mathbf{D}^T\mathbf{x}||_\infty$ [36, 37].

3.3. Training of Output Weight β

For a training sample $(\mathbf{x}, \mathbf{t})$, the output function of ScELM is

$$\mathbf{y} = \sum_{i=1}^{m} \boldsymbol{\beta}_i h_i(\mathbf{x}) = \mathbf{h}(\mathbf{x})\boldsymbol{\beta} \tag{11}$$

where $\boldsymbol{\beta} = [\boldsymbol{\beta}_1, ..., \boldsymbol{\beta}_m]^T$ is the output weight matrix from the hidden layer to the output layer. $\mathbf{h}(\mathbf{x}) = [h_1(\mathbf{x}), ..., h_m(\mathbf{x})]$ denotes the sparse representation (i.e., output of hidden layer) with respect to input $\mathbf{x}$.

The supervised training of $\boldsymbol{\beta}$ is based on N training sample pairs. Each training sample consists of a feature vector $h(\mathbf{x}_k)$ and its binary class label vector (i.e., ground truth) $\mathbf{t}_k = [t_{k,1}, \cdots, t_{k,p}]$, where $k = 1, ..., N$ and p denotes the number of output nodes. In the label vector, each entry indicates whether or not the sample $h(\mathbf{x}_k)$ belongs to the corresponding class. All labels can form a matrix denoted as $\mathbf{T} = [\mathbf{t}_1, \cdots, \mathbf{t}_N]^T$.

Let $\mathbf{y}_k$ denote the actual output vector for the input $\mathbf{x}_k$. Taking all training samples $\{h(x_k)\}_{k=1,\ldots,N}$ into (11) can form a linear representation:

$$\mathbf{H}\boldsymbol{\beta} = \mathbf{Y} \tag{12}$$

where

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}(\mathbf{x}_1) \\ \vdots \\ \mathbf{h}(\mathbf{x}_N) \end{bmatrix} \tag{13}$$

$$\boldsymbol{\beta} = \begin{bmatrix} \boldsymbol{\beta}_1 \\ \vdots \\ \boldsymbol{\beta}_m \end{bmatrix} = \begin{bmatrix} \beta_{1,1} & \cdots & \beta_{1,p} \\ \vdots & \ddots & \vdots \\ \beta_{m,1} & \cdots & \beta_{m,p} \end{bmatrix} \tag{14}$$

and

$$\mathbf{Y} = \begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_N \end{bmatrix} = \begin{bmatrix} y_{1,1} & \cdots & y_{1,p} \\ \vdots & \ddots & \vdots \\ y_{N,1} & \cdots & y_{N,p} \end{bmatrix} \tag{15}$$

The training process aims to minimize the training error $\|\mathbf{T} - \mathbf{H}\boldsymbol{\beta}\|_2^2$ and the norm of output weight $\|\boldsymbol{\beta}\|$. So the training process can be represented as a constrained-optimization problem:

$$\begin{aligned} &\text{Minimize:} && \Psi(\boldsymbol{\beta}, \boldsymbol{\xi}) = \tfrac{1}{2}\|\boldsymbol{\beta}\|_2^2 + \tfrac{C}{2}\|\boldsymbol{\xi}\|_2^2 \\ &\text{Subject to:} && \mathbf{H}\boldsymbol{\beta} = \mathbf{T} - \boldsymbol{\xi} \end{aligned} \tag{16}$$

where constant C is used as a regularization factor to control the trade-off between the closeness to the training data and the smoothness of the decision function such that generalization performance is improved.

Lagrange multiplier technique is used to solve the above optimization problem by constructing the lagrangian:

$$L(\boldsymbol{\beta}, \boldsymbol{\xi}, \boldsymbol{\alpha}) = \frac{1}{2}\|\boldsymbol{\beta}\|_2^2 + \frac{C}{2}\|\boldsymbol{\xi}\|_2^2 - \boldsymbol{\alpha} \cdot (\mathbf{H}\boldsymbol{\beta} - \mathbf{T} + \boldsymbol{\xi}) \tag{17}$$

where $\boldsymbol{\alpha}$ is a Lagrange multiplier matrix with dimensions of $N \times p$. Since the constraints in (16) are all equalities, $\boldsymbol{\alpha}$ can be either positive or negative.

Based on KKT optimization conditions, gradients of L with respect to $\boldsymbol{\beta}$, $\boldsymbol{\xi}$ and $\boldsymbol{\alpha}$ can be obtained as follows:

$$\begin{cases} \dfrac{\partial L}{\partial \boldsymbol{\beta}} = 0 \quad \rightarrow \quad \boldsymbol{\beta} = \mathbf{H}^T \boldsymbol{\alpha} & \text{(18a)} \\ \dfrac{\partial L}{\partial \boldsymbol{\xi}} = 0 \quad \rightarrow \quad \boldsymbol{\alpha} = C\boldsymbol{\xi} & \text{(18b)} \\ \dfrac{\partial L}{\partial \boldsymbol{\alpha}} = 0 \quad \rightarrow \quad \boldsymbol{\xi} = \mathbf{T} - \mathbf{H}\boldsymbol{\beta} & \text{(18c)} \end{cases}$$

By substituting (18b) and (18c) into (18a), the above equations can be written as

$$(\frac{\mathbf{I}}{C} + \mathbf{H}^T\mathbf{H})\boldsymbol{\beta} = \mathbf{H}^T\mathbf{T} \tag{19}$$

If matrix $(\frac{\mathbf{I}}{C} + \mathbf{H}^T\mathbf{H})$ is not singular, solution $\hat{\boldsymbol{\beta}}$ can be obtained as

$$\hat{\boldsymbol{\beta}} = (\frac{\mathbf{I}}{C} + \mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{T} \tag{20}$$

By substituting (18a) and (18b) into (18c), the above equations can be also written as

$$(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T)\boldsymbol{\alpha} = \mathbf{T} \tag{21}$$

If matrix $(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T)$ is not singular, solution $\hat{\boldsymbol{\beta}}$ can be obtained by substituting (21) into (18a)

$$\hat{\boldsymbol{\beta}} = \mathbf{H}^T(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T)^{-1}\mathbf{T} \tag{22}$$

It can be seen that the dimensions of $(\frac{\mathbf{I}}{C} + \mathbf{H}^T\mathbf{H})$ is $m \times m$ while $(\frac{\mathbf{I}}{C} + \mathbf{H}\mathbf{H}^T)$ is $N \times N$. Therefore, if the number of training samples is huge, the solution in (20) can be used to decrease the computational cost; Otherwise, the solution in (22) can be used.

4. Experiments

4.1. Experimental Setup

Our experiments use a total of 16 data sets, including 8 binary-classification cases and 8 multi-classification cases, to evaluate this proposed ScELM algorithm. Most of the data sets are taken from *UCI Machine Learning Repository*

Table 1: Dataset Configurations

Datasets	Training	Testing	Features	Classes	Random Perm
LiverDisorders	230	115	6	2	Yes
Diabetes	512	256	8	2	Yes
Australian Credit	460	230	14	2	Yes
Diabetic	767	384	19	2	Yes
Breast Cancer	379	190	30	2	Yes
Musk	6598	476	166	2	No
Madelon	2000	600	500	2	No
Colon	30	32	2000	2	No
Iris	100	50	4	3	Yes
Wine	118	60	13	3	Yes
Vehicle	564	282	18	4	Yes
Glass	142	72	9	6	Yes
Satimage	4435	2000	36	6	No
Shuttle	43500	12800	9	7	No
Image segmentation	1540	770	19	7	Yes
Ecoli	224	112	7	8	Yes

[38]. The details of these data sets are shown in Table 1. In this table, column "Random Perm" shows whether the training and test data are randomly assigned. For each data set, there are a total of 50 collections of randomly assigned training-test data partitions. In each data partition, the ratio between training and test samples is $2:1$. It can be also seen that the dimensionality of features in these data sets covers a wide range from 4 to 2000.

This proposed ScELM algorithm is also compared with ELM and SVM (Gaussian kernel) algorithms. Implementation of ELM algorithm is based on its MATLAB codes and the implementation codes of SVM are downloaded from

website [39].

A standard PC is used in our experiments and its hardware configuration is as follows:

1. CPU: Intel(R) Pentium(R) CPU G2030 @3.00GHz;
2. Memory: 8.00GB;
3. Graphics Processing Unit (GPU): None.

4.2. Tuning Parameters

Table 2: Parameter Settings of SVM, ELM and ScELM

Datasets	SVM		ELM		ScELM	
	C	γ	L	C	L	C
LiverDisorders	2^{7}	$^{-5}$	1000	2^{-6}	70	2^{2}
Diabetes	2^{4}	2^{-6}	400	2^{-8}	600	2^{7}
Australian Credit	2^{4}	2^{-15}	900	2^{7}	1000	2^{10}
Diabetic	2^{16}	2^{-15}	700	2^{-5}	1000	2^{-5}
Breast Cancer	2^{-1}	2^{-20}	1000	2^{-3}	800	2^{8}
Musk	2^{5}	2^{-1}	2000	2^{-2}	1700	2^{-8}
Madelon	2^{3}	2^{-6}	900	2^{0}	900	2^{0}
Colon	2^{12}	2^{6}	1000	2^{-9}	1000	2^{0}
Iris	2^{15}	2^{-15}	100	2^{-6}	30	2^{3}
Wine	2^{10}	2^{-5}	800	2^{3}	200	2^{0}
Vehicle	2^{14}	2^{2}	1700	2^{-2}	100	2^{-3}
Glass	2^{-15}	2−7	1000	2^{5}	40	2^{7}
Satimage	2^{4}	2^{0}	1000	2^{-5}	1000	2^{10}
Shuttle	2^{10}	2^{-2}	900	2^{1}	50	2^{-2}
Image segmentation	2^{18}	2^{5}	1900	2^{-3}	1400	2^{2}
Ecoli	2^{16}	2^{5}	200	2^{2}	1000	2^{3}

This proposed ScELM algorithm has two tuning parameters like ELM: One is the number of hidden nodes L and the other is regularization factor C. In

our experiments, L is tuned from 100 to 2000 and C is tuned from 2^{-10} to 2^{10}. In SVM algorithm, there are also two tuning parameters: The cost parameter C and the kernel parameter γ. Parameter C is tuned from 2^{-20} to 2^{20} and γ is tuned from 2^{-20} to 2^{20}. For each parameter, it is tuned in the above given range and then the value that can get the highest training accuracy is selected. The settings of these tuning parameters are shown in Table 2. The 'sigmoid' function is used as the activation function in ELM algorithm. In ScELM, the uniform distribution with mean of 0 and variance of 1 is used to randomly set the dictionary.

4.3. Performance Evaluation

Maximal recognition accuracy and average recognition accuracy are used to evaluate SVM, ELM and ScELM algorithms. Average accuracy is obtained by running each algorithm on 50 training-test data partitions. Table 3 and Table 4 respectively show the maximal accuracy and average accuracy.

In terms of the maximal accuracy, ScELM algorithm outperforms ELM and SVM algorithm on half of datasets and shows comparable performance on the rest datasets. For example, the maximal accuracy obtained by ScELM is 19% higher than that obtained by ELM on the vehicle dataset.

In terms of the average accuracy, ScELM algorithm shows comparable performance on most datasets while obtains much higher accuracy on a few datasets (e.g., vehicle and wine datasets) compared with SVM and ELM algorithm.

In terms of the training speed, ScELM is slower than ELM but is faster than SVM. Furthermore, it is much faster than SVM when the number of training samples is huge.

Furthermore, as shown in Table 2, the number of hidden nodes used by ScELM algorithm is much smaller than that of ELM algorithm. It indicates that ScELM algorithm can use a more compact structure to achieve higher or comparable performance compared with ELM algorithm.

Finally, the hidden feature representations are also compared between ScELM and ELM algorithms. As shown in Fig. 2, the hidden feature representations

Table 3: Maximal accuracy obtained by SVM, ELM and ScELM

Datasets	SVM (%)	ELM (%)	ScELM (%)
LiverDisorders	77.59	**80.87**	80.00
Diabetes	**79.38**	75.39	74.61
Australian Credit	89.18	90.04	**90.48**
Diabetic	**78.39**	77.86	78.13
Breast Cancer	96.32	96.32	**98.42**
Musk	86.54	90.34	**92.86**
Madelon	56.39	61.33	**62.83**
Colon	84.38	**87.50**	84.38
Iris	98.00	**100**	**100**
Wine	95.24	93.33	**100**
Vehicle	82.14	80.14	**100**
Glass	75.32	77.78	**81.94**
Satimage	77.24	**80.45**	78.60
Shuttle	**99.74**	99.51	96.10
Image segmentation	**96.14**	95.32	91.30
Ecoli	90.42	**93.75**	91.96

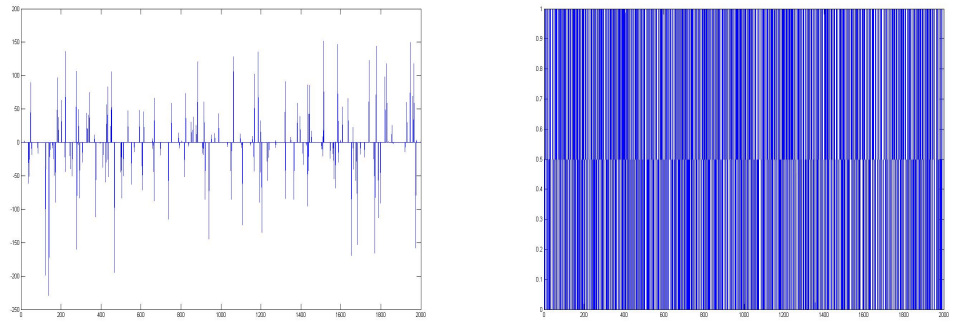
obtained by ScELM are much sparser than those obtained by ELM. The sparsity can partially contribute to improvement of classification performance.

5. Conclusions

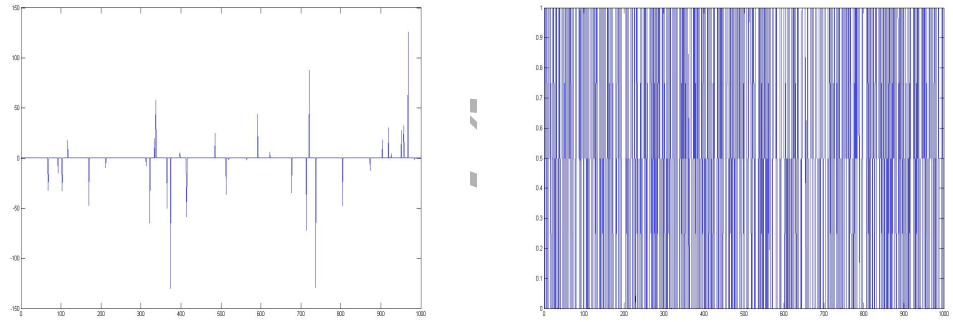
This paper proposes a new method for learning SLNNs, called ScELM. It uses sparse coding technique to map the input features to hidden feature representations such that it can improve the classification performance. This paper conducts extensive experiments on publicly available databases to evaluate this proposed ScELM algorithm and the results show that the ScELM gets better performance than ELM and SVM in terms of classification. Future work in-

Table 4: Average accuracy obtained by SVM, ELM and ScELM

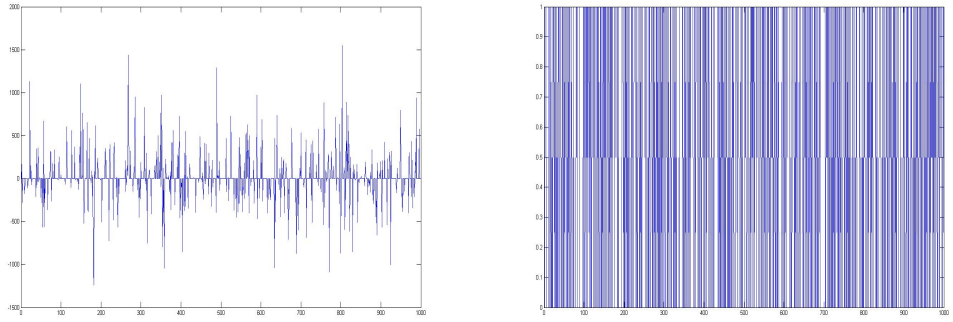
Datasets	SVM		ELM		ScELM	
	Testing Rate (%)	Training Time(s)	Testing Rate (%)	Training Time(s)	Testing Rate (%)	Training Time(s)
LiverDisorders	67.53±3.85	0.52	**76.61±1.89**	0.15	73.9± 3.03	0.47
Diabetes	68.62±4.26	0.61	**72.79±1.28**	0.05	69.88±1.90	0.58
Australian Credit	85.64±2.08	0.26	**85.71±2.08**	0.18	85.65±1.96	0.24
Diabetic	71.24±3.42	8.23	**74.25±1.30**	0.21	74.24±1.64	4.96
Breast Cancer	93.15±1.84	0.24	95.03±0.60	0.44	**96.26±1.29**	0.59
Musk	86.54±0.0	13014	87.41±1.35	12.29	**89.50±1.52**	60.1
Madelon	56.39±0.0	415.2	56.95±1.85	0.66	**57.56±1.70**	87.3
Colon	**84.38±0.0**	0.16	84.19±2.40	0.13	84.00±1.03	0.21
Iris	95.12±2.45	0.34	**98.08±2.49**	0.01	96.88±4.75	0.05
Wine	80.19±6.19	0.18	82.67±4.99	0.23	**96.20±2.88**	0.32
Vehicle	75.37± 1.65	1.51	73.99± 2.06	3.07	**96.33± 2.40**	5.34
Glass	67.83±4.83	0.29	**74.44±1.48**	0.12	72.17±5.40	0.44
Satimage	77.24±0.0	702.4	**79.19±0.58**	3.85	76.68±0.83	164.0
Shuttle	**99.74±0.0**	2864	99.19±0.14	11.47	93.33±1.35	125.1
Image segmentation	**95.23±0.64**	16.5	94.29±0.49	2.12	89.78±0.98	5.44
Ecoli	86.56±3.65	0.25	**93.75±0.0**	0.01	86.12±2.63	0.21



(a)



(b)



(c)

Figure 2: The hidden layer feature representations obtained by ScELM and ELM respectively. Left column: Obtained by ScELM; Right column: Obtained by ELM. (a): Musk dataset. (b): Satimage dataset. (c): Madelon dataset.

cludes using other sparse coding algorithms and autonomously finding an over-complete dictionary.

References

[1] C. Cortes, V. N. Vapnik, Support vector networks, Machine Learning 20 (1995) 273–297.

[2] T. Hastie, S. Rosset, R. Tibshirani, J. Zhu, The entire regularization path for the support vector machine, Journal Machine Learning Research 5 (2004) 1391–1415.

[3] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324.

[4] G. E. Hinton, R. R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504–507.

[5] G. E. Hinton, S. Osindero, Y. W. Teh, A fast learning algorithm for deep belief nets, Neural Computation 22 (2006) 781–796.

[6] R. Salakhutdinov, G. Hinton, Deep boltzmann machine, Journal of Machine Learning Research 5 (2009) 448–455.

[7] R. Salakhutdinov, G. Hinton, An efficient learning procedure for deep boltzmann machines, Neural Computation 24 (8) (2012) 1967–2006.

[8] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, Greedy layer-wise training of deep networks, in: Proceedings of Advances in Neural Information Processing Systems, Vol. 19, 2006, pp. 153–160.

[9] P. Vincent, H. Larochelle, Y. Bengio, P. A. Manzagol, Extracting and composing robust features with denoising autoencoders, in: Proceedings of International Conference on Machine Learning, 2008, pp. 1096–1103.

[10] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion, Journal of Machine Learning Research 11 (2010) 3371–3408.

[11] A. Coates, Y. N. Andew, Search machine learning repository: The importance of encoding versus training with sparse coding and vector quantization, in: Proceedings of International Conference on Machine Learning, 2011, pp. 921–928.

[12] H. Lee, C. Ekanadham, A. Y. Ng, Sparse deep belief net model for visual area v2, in: Proceedings of Advances in Neural Information Processing Systems, Vol. 20, 2008, pp. 1–8.

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

[14] G.-B. Huang, X.-J. Ding, H.-M. Zhou, Optimization method based extreme learning machine for classification, Neurocomputing 74 (2010) 155–163.

[15] G.-B. Huang, H.-M. Zhou, X.-J. 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.

[16] P. L. Bartlett, The sample complexity of pattern classification with neural networks: The size of the weights is more important than the size of the network, IEEE Transactions on Information Theory 44 (2) (1998) 525–536.

[17] G.-B. Huang, L. Chen, C.-K. Siew, Universal approximation using incremental constructive feedforward networks with random hidden nodes, IEEE Transactions on Neural Networks 17 (4) (2006) 879–892.

[18] B. A. Olshausen, D. J. Field, Sparse coding with an overcomplete basis set: A strategy employed by v1?, Visual Research 37 (23) (1997) 3311–3325.

[19] M. A. T. Figueiredo, R. D. Nowak, S. J. Wright, Gradient projection for sparse representation: Application to compressed sensing and other inverse problems, IEEE Journal on Selected Topics In Signal Processing 1 (4) (2007) 586–597.

[20] Y. B. Wang, D. Li, Y. Du, Z. S. Pan, Anomaly detection in traffic using l1-norm minimization extreme learning machine, Neurocomputing 149 (2015) 415–425.

[21] S. Shojaeilangari, W. Y. Yau, K. Nandakumar, J. Li, K. Teoh, Robust representation and recognition of facial emotions using extreme sparse learning, IEEE Transactions on Image Processing 24 (7) (2015) 2140–2152.

[22] D. H. Hubel, T. N. Wiesel, Receptive fields of signal neurons in the cat's striate cortex, Journal of Physiology 148 (1959) 574–591.

[23] E. T. Roll, M. J. Tovee, Sparseness of the neuronal representation of stmuli in the primate temporal visual cortex, Journal of Neurophysiology 173 (1992) 713–726.

[24] J. Tropp, Greed is good: Algorithmic results for sparse approximation., IEEE Transactions on Information Theory 50 (10) (2004) 2231–2242.

[25] S. Mallat, Z. Zhang, Matching pursuits with time-frequency dictionaries., IEEE Transactions on Signal Processing 41 (12) (1993) 3397–3415.

[26] Y. C. Pati, R. Rezaiifar, P. Krishnaprasad, Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition., in: IEEE International Conference on Signal, Systems snd Computers, Vol. 1, 1993, pp. 40–44.

[27] S. Chen, D. Donoho, M. Saunders, Atomic decomposition by basis pursuit, SIAM Review 43 (1) (2001) 129–159.

[28] D. Donoho, M. Elad, Optimally sparse representation in general (nonorthogonal) dictionaries via l^1 minimization., in: Proceeding of the

National Academy of Sciences of the United States of America, Vol. 100, 2003, pp. 2197–2202.

[29] H. Lee, A. Battle, R. Raina, A. Ng, Efficient sparse coding algorithm, in: Advances in neural Information Processing Systems, 2007, pp. 801–808.

[30] J. Wright, A. Y. Yang, A. Ganesh, Robust face recognition via sparse representation, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (2009) 210–227.

[31] H. Liu, Y. Liu, F. Sun, Robust exemplar extraction using structured sparse coding, IEEE Transactions on Neural Networks and Learning Systems 26 (8) (2015) 1816–1821.

[32] H. Liu, D. Guo, F. Sun, Object recognition using tactile measurements: Kernel sparse coding methods, IEEE Transactions on Instrumentation and Measurement 65 (3) (2016) 656–665.

[33] J. Yang, K. Yu, Y. Gong, T. Huang, Linear spatial pyramid matching using sparse coding for image classification, in: IEEE International Conference on Computer Vision and Pattern Recognition, 2009, pp. 1794–1801.

[34] I. Daubechies, M. Defrise, C. D. Mol, An iterative thresholding algorithm for linear inverse problems with a sparsity constraint, Communications on Pure & Applied Mathematics 57 (11) (2003) 1413–1457.

[35] M. A. T. Figueiredo, R. Nowak, An em algorithm for wavelet-bases image restoration, IEEE Transactions on Image Processing 12 (8) (2003) 906–916.

[36] S. J. Kim, K. Koh, S. Boyd, An interior-point method for large-scale l_1 -regularized least squares, IEEE Journal on Selected Topics in Signal Processing 1 (4) (2007) 606–617.

[37] J. J.Fuchs, On sparse representations in arbitrary redundant bases, IEEE Transactions on Information Theory 50 (6) (2004) 1341–1344.

[38] M. Lichman, UCI machine learning repository (2013).
URL http://archive.ics.uci.edu/ml

[39] S. Canu, Y. Grandvalet, V. Guigue, A. Rakotomamonjy, Svm and kernel methods matlab toolbox (2005).
URL http://asi.insarouen.fr/enseignants/~arakotom/toolbox/index.html

Yuanlong Yu received the B.Eng. degree in automatic control in 2000 from the Beijing Institute of Technology, Beijing, China, the M.Eng. degree in computer applied technology in 2003 from Tsinghua University, Beijing, and the Ph.D. degree in electrical engineering in 2010 from Memorial University of Newfoundland, St. Johns, NL, Canada. After completing his doctoral studies, he worked as a Postdoctoral Fellow at Memorial University of Newfoundland. Since September 2011, he has been with Dalhousie University, Halifax, NS, Canada, as a Postdoctoral Fellow. Since 2013, he worked as a Professor at Fuzhou University, China. His main interests are computer vision, pattern recognition, machine learning, visual attention, autonomous mental development and cognitive robotics.

Zhenzhen Sun received the Bachelor's degree in computer science and technology in 2015 at Fuzhou University, Fuzhou, China. Currently, she is a master student at Fuzhou University. Her research interests include computer vision and machine learning.