Neurocomputing 128 (2014) 96–103

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

# Fast sparse approximation of extreme learning machine

Xiaodong Li, Weijie Mao*, Wei Jiang

*State Key Laboratory of Industrial Control Technology, Institute of Cyber-Systems and Control Zhejiang University, Yuquan Campus, Hangzhou 310027, PR China*

ARTICLE INFO

*Article history:*
Received 1 September 2012
Received in revised form
17 December 2012
Accepted 15 January 2013
Available online 26 October 2013

*Keywords:*
Fast greedy algorithm
Extreme learning machine (ELM)
Sparse approximation

ABSTRACT

We introduce a fast sparse approximation schemes of extreme learning machine (ELM) named FSA-ELM of extreme learning machine (ELM). Our algorithms have two compelling features: low complexity and sparse solution. Experiments on benchmark data sets show that the proposed algorithm obtains sparse classifiers at a rather low complexity without sacrificing the generalization performance. As validated by the simulation results, FSA-ELM tends to have better scalability and achieves similar or much better generalization performance with much faster learning speed than the traditional ELM algorithm.



## 1. Introduction

In the classification problem, along with corresponding class labels, a set of samples of input vectors is given, and the task is to find a deterministic function that best represents the relation between input vectors and class labels. A successful method for solving the classification problem is the least squares support vector machine (LS-SVM) [1,2], which attempts to minimize the least square error on the training samples while simultaneously maximizing the margin between two classes. Extensive empirical comparisons [3] show that LS-SVM achieves good performance on various classification problems, but two obvious limitations persist. First, the training procedure of LS-SVM amounts to solving a set of linear equations [4]. Although the training problem is solvable, it is not easy to solve for a large-scale data set using the classical techniques, e.g., the Gaussian elimination, because their computational complexity usually scales cubically with the size of the training samples. Secondly, the solution of LS-SVM lacks sparseness and hence, the test speed is significantly slower than that of other algorithms, such as the support vector machine (SVM) [5,6] and neural networks (NNs) [7,8,18,21].

Many researchers have attempted to develop methods to overcome these aforementioned two limitations. As a fast algorithms for LS-SVM, Suykens et al. [9] presented a conjugate gradient algorithm, and Chu et al. [10] proposed an improved conjugate gradient algorithm. Although these algorithms achieve low complexity, their resulting solutions are not sparse. Hence, the second limitation persists. To address the sparseness of LS-SVM, Suykens et al. [11,12,19] adopted a simple approach; they introduced sparseness by sorting the support value spectrum (SVS), i.e., the absolute value of the solution of LS-SVM. Kruif and Vries [13] presented a more complicated pruning mechanism that omits the sample bearing the least error after it has been omitted. Zeng and Chen [14] used an SMO-based pruning method. However, these algorithms require a set of linear equations (gradually decreasing in size) to be solved multiple times, and hence involves high computational costs. Therefore, the first limitation persists. Moreover, Suykens et al. [15] proposed a fixed-size LS-SVM fast-finding algorithm for obtaining the sparse approximate solution of LS-SVM. Recently, Hoegaerts et al. [16] proposed a similar approach to solving the kernel partial least squares regression. Jiao et al. [17] presented a fast greedy algorithm for LS-SVM that attempts to overcome the two limitations simultaneously. Huang et al. investigated the convex incremental ELM and the enhanced random search-based incremental ELM in [24] respectively. Huang et al. in [25] showed that single SLFNs with randomly generated additive or RBF nodes with a widespread of piecewise continuous activation functions can universally approximate any continuous target function on any compact subspace of the Euclidean space. Furthermore, Feng et al. in [26] addressed the error minimized ELM with growth of hidden nodes. Previous studies have investigated error minimized ELM (EM-ELM) as an error minimization-based method in which the number of hidden nodes can grow one-by-one or group-by-group until optimal. By modifying the classic forward selection algorithm, a constructive hidden nodes selection method for ELM (CS-ELM) [27] was proposed, which is less greedy and has no matrix decompositions. At each step of CS-ELM, the hidden node with an output that has the highest

* Corresponding author. Tel.: +86 136 0581 4511.
*E-mail address:* wjmao@iipc.zju.edu.cn (W. Mao).

0925-2312/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2013.01.064

correlation with the current residual is selected. The standard ELM algorithm proposed in [29,30] can approximate any target continuous function and classify any disjoint regions created by the designer of the classifier network. Huang et al. [23] showed that both LS-SVM and PSVM can be simplified by removing the term bias $b$; the resultant learning algorithms are unified with ELM. Instead of different variants being requested for different types of applications, ELM can be applied in regression and multiclass classification applications directly. ELM can work with a wide variety of feature mappings (including Sigmoid networks, RBF networks, trigonometric networks, threshold networks, fuzzy inference systems, fully complex neural networks, high-order networks, and ridge polynomial networks). In contrast to the algorithms mentioned previously, FSA-ELM can solve the sparseness issue, and the test speed is very high. The proposed algorithm tends to have better scalability and achieves a similar or much better generalization performance at much higher learning speed than the traditional ELM algorithm.

## 2. A brief review of ELM

This section briefly reviews the ELM proposed in [9,10]. A key principle of ELM is that one may randomly choose and fix the hidden node parameters. After the hidden nodes parameters have been chosen randomly, SLFN becomes a linear system where the output weights of the network can be determined analytically using a simple generalized inverse operation of the hidden layer output matrices.

### 2.1. Extreme learning machine (ELM)

The output of an SLFN with $N$ hidden nodes (additive or RBF nodes) can be represented by

$$f_N(x)=\sum_{i=1}^{N}\beta_i G(\mathbf{x};\mathbf{w}_i,b_i),\quad \mathbf{x}\in\mathbf{R}^n,\ \ b_i\in R^n, \tag{1}$$

where $\mathbf{w}_i$ and $b_i$ are the learning parameters of hidden nodes, $\beta_i$ is the weight connecting the $i$th hidden node to the output node, and $G(\mathbf{x};\mathbf{w}_i,b_i)$ is the output of the $i$th hidden node with respect to the input $\mathbf{x}$. For additive hidden nodes with the sigmoid or threshold activation function $G(\mathbf{x};\mathbf{w}_i,b_i)$, $g(\mathbf{x}):\mathbf{R}\to\mathbf{R}$, is given by

$$G(\mathbf{x};\mathbf{w}_i,b_i)=g(\mathbf{w}_i\mathbf{x}+b_i),\quad a_i\in\mathbf{R} \tag{2}$$

where $\mathbf{a}_i$ is the weight vector connecting the input layer to the $i$th hidden node, $b_i$ is the bias of the $i$th hidden node, and $\mathbf{w}_i\mathbf{x}$ denotes the inner product of vectors $\mathbf{a}_i$ and $\mathbf{x}$ in $R^n$. For RBF hidden nodes with the Gaussian or triangular activation function $g(\mathbf{x}):\mathbf{R}\to\mathbf{R}$, $G(\mathbf{x};\mathbf{w}_i,b_i)$ is given by

$$G(\mathbf{x};\mathbf{w}_i,b_i)=g(b_i\|\mathbf{x}-\mathbf{w}_i\|),\quad b_i\in R^+, \tag{3}$$

where $\mathbf{w}_i$ and $b_i$ are the center and the impact factor of $i$th RBF node. $R^+$ indicates the set of all positive real values.

For $N$ arbitrary distinct samples $(\mathbf{x}_k,\mathbf{t}_k)\in\mathbf{R}^n\times\mathbf{R}^m$ if a SLFN with $N$ hidden nodes can approximate these $N$ samples with zero error, it then implies that there exist $\beta_i$, $\mathbf{w}_i$ and $b_i$ such that

$$\sum_{i=1}^{N}\beta_i G(\mathbf{x}_k;\mathbf{w}_i,b_i)=\mathbf{t}_k,\quad k=1,\ldots,N \tag{4}$$

Eq. (4) can be written compactly as

$$\mathbf{H}\beta=\mathbf{T}, \tag{5}$$

where

$$\mathbf{H}(\mathbf{w}_1,\ldots,\mathbf{w}_N,\ \mathbf{x}_1,\ldots,\mathbf{x}_N,\ b_1,\ldots,b_N)$$

$$=\begin{bmatrix} G(\mathbf{x}_1;\mathbf{w}_1,b_1) & \cdots & G(\mathbf{x}_1;\mathbf{w}_N,b_N)\\ \vdots & \cdots & \vdots\\ G(\mathbf{x}_N;\mathbf{w}_1,b_1) & \cdots & G(\mathbf{x}_N;\mathbf{w}_N,b_N)\end{bmatrix}_{N\times N} \tag{6}$$

$$\beta=\begin{bmatrix}\beta_1^T\\ \vdots\\ \beta_N^T\end{bmatrix}_{N\times M}\quad\text{and}\quad \mathbf{T}=\begin{bmatrix}\mathbf{t}_1^T\\ \vdots\\ \mathbf{t}_N^T\end{bmatrix}_{N\times M}. \tag{7}$$

$\mathbf{H}$ is called the hidden layer output matrix of the network [22,23]; the $i$th column of $\mathbf{H}$ is the $i$th hidden node's output vector with respect to inputs $\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_N$ and the $k$th row of $\mathbf{H}$ is the output vector of the hidden layer with respect to input $\mathbf{x}_k$.

Huang et al. [20,24,25,28] proved that for SLFNs with additive or RBF hidden nodes, one may randomly choose and fix the hidden node parameters and then analytically determine the output weights when approximating any continuous target function. It should be noted that the proof of universal approximation is shown to be valid for SLFNs with a general type of hidden nodes, i.e., in the form of sigmoid, RBF, or a hybrid of both [24,25].

In this regard, for $N$ arbitrary distinct samples $(\mathbf{x}_k,\mathbf{t}_k)$, in order to obtain an arbitrarily small non-zero training error, one may randomly generate hidden nodes (with random parameters $(\mathbf{w}_i,b_i)$). Eq. (5) then becomes a linear system and the output weights $\beta$ are estimated as

$$\hat{\beta}=\mathbf{H}^{\dagger}\mathbf{T}, \tag{8}$$

where $\mathbf{H}^{\dagger}$ is the Moore–Penrose generalized inverse [30] of the hidden layer output matrix $\mathbf{H}$. The output weights are calculated in a single step here. This avoids any lengthy training procedure where the network parameters are adjusted iteratively with appropriately chosen control parameters (learning rate, learning epochs, etc.). The three-step ELM algorithm [27] can be summarized as follows:

**ELM Algorithm**. Given a training set $\mathbb{N}=\{(\mathbf{x}_k,\mathbf{t}_k)|\mathbf{x}_k\in\mathbf{R}^n,\ \mathbf{t}_k\in|\mathbf{R}^m,\ k=1,\ldots,N\}$, activation function $g(x)$ and hidden nodes $\tilde{N}$,

Step 1: Randomly assign hidden node parameters $(\mathbf{w}_i,b_i),\quad i=1,\ldots,\tilde{N}$.
Step 2: Calculate the hidden layer output matrix $\mathbf{H}$.
Step 3: Calculate the output weight $\beta:\beta=\mathbf{H}^{\dagger}\mathbf{T}$.

where $\mathbf{H}$, $\beta$, and $\mathbf{T}$ are defined as Formulas (6) and (7).

### 2.2. Minimum norm least squares (LS) solution of SLFNs

It is very interesting that, unlike the most common understanding that all the parameters of SLFNs need to be adjusted, the input weights $\mathbf{w}_i$ and the hidden layer biases $b_i$ are in fact not necessarily tuned and the hidden layer output matrix $\mathbf{H}$ can actually remain unchanged once random values have been assigned to these parameters in the beginning of learning. For fixed input weights $\mathbf{w}_i$ and the hidden layer biases $b_i$, seen in Eq. (9), to train an SLFN is simply equivalent to finding a least squares solution $\hat{\beta}$ of the linear system $\mathbf{H}\beta=\mathbf{T}$

$$\|\mathbf{H}(\hat{\mathbf{w}}_1,\ldots,\hat{\mathbf{w}}_{\tilde{N}},\hat{b}_1,\ldots,\hat{b}_{\tilde{N}})\hat{\beta}-\mathbf{T}\|$$
$$=\min_{\mathbf{w}_i,b_i,\beta}\|\mathbf{H}(\mathbf{w}_1,\ldots,\mathbf{w}_{\tilde{N}},\hat{b}_1,\ldots,\hat{b}_{\tilde{N}})\beta-\mathbf{T}\| \tag{9}$$

$$\|\mathbf{H}(\mathbf{w}_1,\ldots,\mathbf{w}_{\tilde{N}},b_1,\ldots,b_{\tilde{N}})\hat{\beta}-\mathbf{T}\|$$
$$=\min_{\beta}\|\mathbf{H}(\mathbf{w}_1,\ldots,\mathbf{w}_{\tilde{N}},b_1,\ldots,b_{\tilde{N}})\beta-\mathbf{T}\|. \tag{10}$$

If the number $\tilde{N}$ of hidden nodes is equal to the number $N$ of distinct training samples, $\tilde{N}=N$, matrix $\mathbf{H}$ is square and invertible when the input weight vectors $\mathbf{w}_i$ and the hidden biases $b_i$ are