Neurocomputing 207 (2016) 560–567

Contents lists available at ScienceDirect

Neurocomputing

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

Hessian semi-supervised extreme learning machine

Ganesh Krishnasamy, Raveendran Paramesran*

Department of Electrical Engineering, Faculty of Engineering, Universiti Malaya, Lembah Pantai, 50603 Kuala Lumpur, Malaysia

A R T I C L E I N F O

Article history:
Received 2 October 2015
Received in revised form
22 April 2016
Accepted 13 May 2016
Communicated by G.-B. Huang
Available online 20 May 2016

Keywords:
Extreme learning machine
Semi-supervised learning
Manifold learning
Hessian regularization

A B S T R A C T

Extreme learning machine (ELM) has emerged as an efficient and effective learning algorithm for classification and regression tasks. Most of the existing research on the ELMs mainly focus on supervised learning. Recently, researchers have extended ELMs for semi-supervised learning, in which they exploit both the labeled and unlabeled data in order to enhance the learning performances. They have incorporated Laplacian regularization to determine the geometry of the underlying manifold. However, Laplacian regularization lacks extrapolating power and biases the solution towards a constant function. In this paper, we present a novel algorithm called Hessian semi-supervised ELM (HSS-ELM) to enhance the semi-supervised learning of ELM. Unlike the Laplacian regularization, the Hessian regularization that favors functions whose values vary linearly along the geodesic distance and preserves the local manifold structure well. This leads to good extrapolating power. Furthermore, HSS-ELM maintains almost all the advantages of the traditional ELM such as the significant training efficiency and straight forward implementation for multiclass classification problems. The proposed algorithm is tested on publicly available data sets. The experimental results demonstrate that our proposed algorithm is competitive with the state-of-the-art semi-supervised learning algorithms in term of accuracy. Additionally, HSS-ELM requires remarkably less training time compared to semi-supervised SVMs/regularized least-squares methods.

1. Introduction

The extreme learning machine (ELM) is a relatively new training algorithm for a single-hidden layer feedforward network (SLFN) that enables fast training of the network [1]. Many existing SLFN algorithms such as the back-propagation algorithm [2] and the Levenberg–Marquardt algorithm [3], utilize gradient descent optimization to adjust the weights and biases of the neurons at both the hidden and output layers of the network.

Support vector machine (SVM) is considered one of the most successful algorithms for training SLFNs, which is a maximal margin classifier established under the framework of structural risk minimization [4,5]. The formulation used in SVM can be solved conveniently since the dual problem of SVM is a quadratic programming. SVMs have been applied extensively in many applications mainly due to simplicity and stable generalization performances [6,7,8,9].

Recently, Huang et al. [1,10,11] proposed a new algorithm termed as extreme learning machine (ELM) to train SLFNs. Unlike the conventional approaches, ELM only needs to analytically calculate the output weights while the input weights and hidden layer biases are randomly generated. Despite this simplicity, however, ELM not only reaches the smallest training error, but also the smallest norm of output weights that leads to a good generalization performance [12]. Recent research studies show that ELM has comparable or even better performances in prediction accuracy compared to SVM [10,11,13]. In recent years, many extension of basic ELMs have been tailored for solving specific problems, e.g. online sequential data [14,15,16], imbalanced data [17,18] and noisy/missing data [19,20,21].

Most of the existing works in ELM mainly focus on supervised learning that requires large number of labeled patterns for classification and regression tasks. In practice, it is cumbersome to collect a large amount of labeled data as it is both expensive and time consuming. While, obtaining unlabeled data is both easier and more cost effective. To circumvent the problem faced in supervised learning, semi-supervised learning (SSL) algorithms have been introduced. SSL algorithms take advantage of both labeled and unlabeled data in order to improve the prediction accuracy while saving the labor cost for annotating large amount of label data [22,23].

Manifold regularization based approaches have been widely applied in semi-supervised learning algorithms. Manifold regularization enhances the performances of semi-supervised learning by trying to explore the geometry of intrinsic data probability

* Corresponding author. Tel.: +60 3 79675253.
E-mail addresses: krishnasamy.ganesh@gmail.com (G. Krishnasamy), ravee@um.edu.my (R. Paramesran).

http://dx.doi.org/10.1016/j.neucom.2016.05.039
0925-2312/© 2016 Elsevier B.V. All rights reserved.

of distribution. One of the most popular manifold regularization is the Laplacian regularization [24,25], in which it utilizes graph Laplacian to determine the geometry of the underlying manifold. Laplacian regularization has been implemented in various fields such as in sparse coding [26,27], classification [25,28] and feature selection [29,30]. Recently, Laplacian regularization has been also applied in ELM for semi-supervised learning tasks [31,32,33,34,35].

Although, semi-supervised learning methods based on Laplacian regularization produce good performance, they suffer from few drawbacks. Its performance worsens when there are only few labeled data available as it lacks extrapolating power. Furthermore, it has been reported that Laplacian regularization biases the solution towards a constant function due to its constant null space and cannot preserve well the local topology [36].

In contrast, Hessian regularization has a richer null space and can favor the learned functions whose values vary linearly along the data manifold. Furthermore, it can exploit the intrinsic local geometry of the data manifold well and has a better extrapolating power [36,37,38,39]. Thus, Hessian regularization is more suited for semi-supervised learning compared to Laplacian regularization. Hessian regularization has been extensively implemented in many semi-supervised applications such as in kernel regression [36], classification [40,41,42], sparse coding [43,44] and feature selection [45].

In this paper, we extend the ELM to handle semi-supervised learning problems by introducing Hessian regularization into ELM. Unlike the Laplacian regularization which was used in the previous semi-supervised ELM algorithms, Hessian regularization favors functions whose values vary linearly with respect to geodesic distance and preserves the local manifold well. Therefore, Hessian regularization enhances the performance of ELM in semi-supervised learning. Furthermore, the proposed algorithm inherits the computational efficiency and learning capability of traditional ELM, especially for multiclass classification problems. We conducted several experiments using the standard data sets to evaluate our algorithm against state-of-the-art semi-supervised algorithms. The results show that the proposed algorithm is competitive with other semi-supervised algorithms in terms of accuracy and requires much less training time compared to semi-supervised SVMs/regularized least-square based methods for multiclass classification problems.

This paper is organized as follows: Section 2 contains the description of ELM, manifold regularization and semi-supervised ELM (SS-ELM). In Section 3, we present our proposed framework which consists of Hessian regularization and HSS-ELM formulation. Experimental results are presented in Section 4. Section 5 concludes the study.

2. Related work

In this section, we present a brief description of ELM, manifold regularization and semi-supervised extreme learning (SS-ELM), which are the underlying basis of our work.

2.1. ELM

We briefly describe the ELM algorithm as proposed in Huang et al. [1,10]. In a supervised learning, a training set with N samples is denoted by $\{\boldsymbol{X},\boldsymbol{T}\}=\{\boldsymbol{x}_i,\boldsymbol{t}_i\}_{i=1}^{N}$, where $\boldsymbol{x}_i \in \Re^D$. The corresponding class labels are given by $\boldsymbol{t}_i=[t_{i1},t_{i2},\ldots,t_{iK}]$, where K represents the number of classes. We set $t_{ik}=1$ if $\boldsymbol{x}_i$ belongs class k and $t_{ik}=0$, otherwise. We utilize training samples $\{\boldsymbol{X},\boldsymbol{T}\}$ in order to train the ELM. This SLFN consists of n_D input neurons, n_H hidden neurons and n_K output neurons.

Training of ELM is divided into two stages. In the first stage, the hidden layer is constructed by using a fixed number of randomly generated mapping neurons by using activation functions such as Sigmoid function (1) and Gaussian function (2):

$$g(\boldsymbol{x};\theta)=\frac{1}{1+\exp(-(\boldsymbol{a}^T\boldsymbol{x}+b))} \tag{1}$$

$$g(\boldsymbol{x};\theta)=\exp(-b\|\boldsymbol{x}-\boldsymbol{a}\|) \tag{2}$$

where $\theta=\{\boldsymbol{a},b\}$ are the parameters of the mapping function and $\|\cdot\|$ represents the Euclidean norm.

These parameters are randomly generated using any continuous probability distribution, for example a uniform distribution of $(-1,1)$. Since these parameters are randomly generated, the output weights between the hidden and output neurons can be analytically calculated. Therefore, ELM is considerably more efficient than learning with back-propagation and training SVMs.

These hidden neurons map the input data into n_H dimension of random feature space, thus the network output is expressed as:

$$\boldsymbol{f}(\boldsymbol{x_i})=\boldsymbol{\phi}(\boldsymbol{x_i})\boldsymbol{\beta},\quad i=1,2,\ldots N \tag{3}$$

where $\boldsymbol{\phi}(\boldsymbol{x_i})\in\Re^{1\times n_H}$, output of hidden layer corresponding to training vector $\boldsymbol{x_i}$ and $\boldsymbol{\beta}\in\Re^{n_H\times n_K}$, the output weights between the hidden layer with the output layer.

In the next stage, ELM calculates the output weights by minimizing the norm of the output weights as follows:

$$\begin{aligned}&\min_{\boldsymbol{\beta}\in\Re^{n_H\times n_K}}\ \frac{1}{2}\|\boldsymbol{\beta}\|^2+\frac{C}{2}\sum_{i=1}^{N}\|\boldsymbol{e_i}\|^2\\ &\text{s.t.}\quad \boldsymbol{\phi}(\boldsymbol{x_i})\boldsymbol{\beta}=\boldsymbol{t}_i^T-\boldsymbol{e}_i^T,\quad i=1,2,\ldots N\end{aligned} \tag{4}$$

where $\boldsymbol{e}_i\in\Re^{n_K}$ denotes the error vector corresponding to $\boldsymbol{x}_i$ and C is a penalty function coefficient on the training errors.

The unconstrained optimization formulation is obtained by substituting the constraints into the objective function:

$$\min_{\boldsymbol{\beta}\in\Re^{n_H\times n_K}} L_{ELM}=\frac{1}{2}\|\boldsymbol{\beta}\|^2+\frac{C}{2}\|\boldsymbol{T}-\boldsymbol{\Phi\beta}\|^2 \tag{5}$$

where $\boldsymbol{\Phi}=[\boldsymbol{\phi}(\boldsymbol{x}_1)^T,\boldsymbol{\phi}(\boldsymbol{x}_2)^T,\ldots,\boldsymbol{\phi}(\boldsymbol{x}_N)^T]^T\in\Re^{N\times n_H}$.

We set the gradient of L_{ELM} with respect $\boldsymbol{\beta}$ to zero to obtain the following formulation:

$$\nabla L_{ELM}=\boldsymbol{\beta}+C\boldsymbol{\Phi}^T(\boldsymbol{T}-\boldsymbol{\Phi\beta})=0 \tag{6}$$

If the number of training samples is larger than the number of hidden neurons, Eq. (6) will be overdetermined. In a such case, $\boldsymbol{\Phi}$ has more rows than columns and full column rank. Therefore, we have a close-form solution for (5):

$$\boldsymbol{\beta}^*=\left(\boldsymbol{\Phi}^T\boldsymbol{\Phi}+\frac{\boldsymbol{I}_{n_H}}{C}\right)\boldsymbol{\Phi}^T\boldsymbol{T} \tag{7}$$

where $\boldsymbol{I}_{n_H}$ is an n_H-by-n_H identity matrix.

On the other hand, $\boldsymbol{\Phi}$ will have more columns than rows if the number of hidden neurons is larger than the number of training samples, which might lead to an under-determined least-squares problem. In such a case, the output weight $\boldsymbol{\beta}$ might have an infinite number of solutions.

In order to address this problem, $\boldsymbol{\beta}$ will be restricted to be a linear combination of the rows of $\boldsymbol{\Phi}$: $\boldsymbol{\beta}=\boldsymbol{\Phi}^T\boldsymbol{\alpha}$, where $\boldsymbol{\alpha}\in\Re^{N\times n_K}$ [33]. $\boldsymbol{\Phi\Phi}^T$ is invertible when $\boldsymbol{\Phi}$ has more columns than rows and full row rank. We multiply both side of $(\boldsymbol{\Phi\Phi}^T)^{-1}\ \boldsymbol{\Phi}$ to get:

$$\boldsymbol{\alpha}+C(\boldsymbol{T}-\boldsymbol{\Phi\Phi}^T\boldsymbol{\alpha})=0 \tag{8}$$

This leads to:

$$\boldsymbol{\beta}^*=\boldsymbol{\Phi}^T\boldsymbol{\alpha}^*=\boldsymbol{\Phi}^T\left(\boldsymbol{\Phi\Phi}^T+\frac{\boldsymbol{I_N}}{C}\right)^{-1}\boldsymbol{T} \tag{9}$$

where $\boldsymbol{I}_N$ is an N-by-N identity matrix.