Neurocomputing ∎ (∎∎∎∎) ∎∎∎–∎∎∎

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

ARTICLE INFO

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

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

ABSTRACT

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/

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:

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

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:

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

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:

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

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]. $\boldsymbol{\Phi}\boldsymbol{\Phi}^T$ is invertible when $\boldsymbol{\Phi}$ has more columns than rows and full row rank. We multiply both side of $(\boldsymbol{\Phi}\boldsymbol{\Phi}^T)^{-1}\boldsymbol{\Phi}$ to get:

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

This leads to:

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

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

Please cite this article as: G. Krishnasamy, R. Paramesran, Hessian semi-supervised extreme learning machine, Neurocomputing (2016), http://dx.doi.org/10.1016/j.neucom.2016.05.039

Hence, we use (7) to calculate output weights when the number of training sample is larger than the number of hidden neurons, or otherwise (9) is used.

2.2. Manifold regularization

Two important assumptions are made to develop the semi-supervised learning framework:

- Both labeled data $\boldsymbol{X}_l$ and unlabeled data $\boldsymbol{X}_u$ are drawn from the same marginal distribution P_X.
- The conditional probabilities of $P(\boldsymbol{y}|\boldsymbol{x}_1)$ and $P(\boldsymbol{y}|\boldsymbol{x}_2)$ should be similar if points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are close to each other.

The second assumption is known as the smoothness assumption and is widely utilized in machine learning. By applying this assumption on the data, the framework of the manifold regularization proposes to minimize the following objective function:

$$L_m = \frac{1}{2}\sum_{i,j} w_{i,j} \| (P(\boldsymbol{y}|\boldsymbol{x}_i) - P(\boldsymbol{y}|\boldsymbol{x}_j)) \|^2 \tag{10}$$

where w_{ij} measures the similarity between two points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. The similarity weight w_{ij} is usually defined by the heat kernel function $\exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2/2\sigma^2)$ or fixed to 1.

From (10), we can observe that $P(\boldsymbol{y}|\boldsymbol{x})$ is required to vary smoothly along the geodesics in $P(\boldsymbol{x})$ since a small change in $\boldsymbol{x}$ will penalize a large variation in the conditional probability of $P(\boldsymbol{y}|\boldsymbol{x})$.

We can approximate (10) to the following expression since its cumbersome to compute $P(\boldsymbol{y}|\boldsymbol{x})$:

$$\hat{L}_m = \frac{1}{2}\sum_{ij} w_{ij} \| \hat{\boldsymbol{y}}_i - \hat{\boldsymbol{y}}_j \|^2 \tag{11}$$

where $\hat{\boldsymbol{y}}_i$ and $\hat{\boldsymbol{y}}_j$ are predicted labels of $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$.

Eq. (11) can be further simplified into a matrix form of:

$$\hat{L}_m = \mathrm{Tr}(\hat{\boldsymbol{Y}}\boldsymbol{L}\hat{\boldsymbol{Y}}) \tag{12}$$

where $\mathrm{Tr}(\cdot)$ represents the trace of a matrix. Graph Laplacian, $\boldsymbol{L}$ is defined by $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W}$, where $\boldsymbol{D}$ is a diagonal matrix with its diagonal elements $D_{ii} = \sum_{j=1}^{l+u} w_{ij}$ and similarity matrix, $\boldsymbol{W} = [w_{ij}]$. We can also normalize the Laplacian by $\boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{L}\boldsymbol{D}^{-\frac{1}{2}}$ and iterate it to the degree of integer p based on some prior knowledge [46].

2.3. Semi-supervised extreme learning machine (SS-ELM)

In semi-supervised environment, we have only few labeled data and plenty of unlabeled data. Denote $\{\boldsymbol{X}, \boldsymbol{T}\} = \{\boldsymbol{x}_l, \boldsymbol{t}_l\}_{i=1}^{l}$ as labeled data and $\{\boldsymbol{X}_u\} = \{\boldsymbol{x}_u\}_{i=1}^{u}$ as unlabeled data, where l and u are the number of labeled and unlabeled data.

Recently, Huang et al. [33] proposed a new semi-supervised learning algorithm called as SS-ELM, which incorporates the graph Laplacian into ELM to leverage the unlabeled data to improve the classification accuracy when labeled data is sparse. SS-ELM is formulated as:

$$\begin{aligned} \min_{\boldsymbol{\beta} \in \Re^{n_H \times n_K}} \quad & \frac{1}{2}\|\boldsymbol{\beta}\|^2 + \frac{1}{2}\sum_{i=1}^{l} C_i \|\boldsymbol{e}_i\| + \frac{\lambda}{2}\mathrm{Tr}(\boldsymbol{F}^T\boldsymbol{L}\boldsymbol{F}) \\ \text{s.t.} \quad & \boldsymbol{\phi}(\boldsymbol{x}_i)\boldsymbol{\beta} = \boldsymbol{t}_i^T - \boldsymbol{e}_i^T, \quad i = 1, 2, \ldots, l \\ & \boldsymbol{f}_i = \boldsymbol{\phi}(\boldsymbol{x}_i)\boldsymbol{\beta}, \quad i = 1, 2, \ldots, l+u \end{aligned} \tag{13}$$

where the graph Laplacian, $\boldsymbol{L} \in \Re^{(l+u)\times(l+u)}$ is built from the labeled and unlabeled data and $\boldsymbol{F} \in \Re^{(l+u)\times n_K}$ is the matrix of the output network with its ith row equal to $\boldsymbol{f}(\boldsymbol{x}_i)$. $\boldsymbol{\beta} \in \Re^{n_H \times n_K}$ represents the output weights while λ is a tradeoff parameter.

3. Proposed framework

In this section, we present our algorithm HSS-ELM in detail. Our algorithm incorporates the Hessian energy as a regularizer that takes account of local manifold structure of the data space.

3.1. Hessian regularization

We adopt the Hessian regularization proposed by Kim et al. [36] into our semi-supervised extreme learning machine frame.

Given a smooth manifold $M \subset R^n$, we can define the tangent space $T_x(M) \subset R^n$ for point $\boldsymbol{x} \in M$. Let $\varphi(\boldsymbol{x}_i) = f_i$ be a mapping function that produces the predictive label of $\boldsymbol{x}_i$, then Hessian energy $E_H(\varphi)$ can be expressed as follows:

$$E_H(\varphi) = \int_M \| \nabla_a \nabla_b \varphi \|^2_{T^*_x M \otimes T^*_x M} dV(\boldsymbol{x}) \tag{14}$$

where $\nabla_a \nabla_b \varphi$ is the second covariant derivative of φ and $dV(\boldsymbol{x})$ is the natural volume element [47].

In a special coordinate system on M, the so-called normal coordinates, the Hessian energy function (14) can be evaluated quite easily. Normal coordinates at given point $\boldsymbol{x}$ are coordinates on M such that the manifold looks as Euclidean as possible around $\boldsymbol{x}$. Thus, in normal coordinates x^r is centered at $\boldsymbol{x}$, we have:

$$\| \nabla_a \nabla_b \varphi \|^2_{T^*_x M \otimes T^*_x M} = \sum_{r,s=1}^{l} \left(\frac{\partial^2 \varphi}{\partial x^r \partial x^s} \right)^2 \tag{15}$$

So that at a given point $\boldsymbol{x}$, the norm of the second covariant derivative is just the Frobenius norm of Hessian of φ in normal coordinates.

Let $N_k(\boldsymbol{x}_i)$ represent the set of k nearest neighbors of $\boldsymbol{x}_i$. In order to estimate the local tangent space $T^*_{\boldsymbol{x}_i} M$ (seen as an l-dimension affine space of R^d), we perform PCA on the points in $N(\boldsymbol{x}_i)$, and then the l leading eigenvectors correspond to an orthogonal basis of $T^*_{\boldsymbol{x}_i} M$. Having the exact tangent space $T^*_{\boldsymbol{x}_i} M$, one can determine the normal coordinates x^r of a point $\boldsymbol{x}_t \in N_k(\boldsymbol{x}_i)$ $(1 \le t \le k)$. Given the function values $\varphi(x_t) = f_t$ on $N_k(\boldsymbol{x}_i)$ the Hessian of φ at $\boldsymbol{x}_i$ can be approximated as follows:

$$\left. \frac{\partial^2 \varphi}{\partial x^r \partial x^s} \right|_{\boldsymbol{x}_i} \approx \sum_{t=1}^{k} Z^{(i)}_{rst} f_t \tag{16}$$

where $Z^{(i)}_{rst}$ is a local Hessian operator of sample $\boldsymbol{x}_i$ in the normal coordinates x^r of a point $\boldsymbol{x}_t \in N_k(\boldsymbol{x}_i)$ and it can be computed by fitting a second-order polynomial using linear least squares.

Fitting a second-order polynomial $q(x)$ in normal coordinates to $\varphi(x_t)_{t=1}^{k}$

$$q^{(i)}(x) = \varphi(x_i) + \sum_{r=1}^{l} H_r x^r + \sum_{r=1}^{l}\sum_{s=r}^{l} N_{rs} x^r x^s \tag{17}$$

where the zeroth-order term is fixed at $\varphi(x_i)$. In the limit as the neighbor size tends to zero, $q^{(i)}(x)$ becomes the second-order Taylor expansion of φ around $\boldsymbol{x}_i$, that is:

$$H_r = \frac{\partial \varphi}{\partial x^r}|_{\boldsymbol{x}_i}, \quad N_{rs} = \frac{1}{2}\frac{\partial^2 \varphi}{\partial x^r \partial x^s}|_{\boldsymbol{x}_i} \tag{18}$$

with $N_{rs} = N_{sr}$. In order to fit polynomial we use standard linear least squares

$$\underset{v \in R^p}{\mathrm{argmin}} \quad \sum_{t=1}^{k} \left((\varphi(x_t) - \varphi(x_i)) - (\Gamma v)_t \right)^2 \tag{19}$$

where $\boldsymbol{\Gamma} \in R^{k \times P}$ is the design matrix with $P = l + l(l+1)/2$. The corresponding basis function γ is monomial of the normal coordinates (centered $\boldsymbol{x}_i$) of $x_t \in N_k(\boldsymbol{x}_i)$ up to second order. The solution $v \in R^p$ is $v = \boldsymbol{\Gamma}^+ \boldsymbol{f}$, where $\mathbf{f} = (\mathbf{f_1}, \mathbf{f_2}, \ldots, \mathbf{f_k})^T \in \mathbf{R^k}$ and $\boldsymbol{f}_t = \varphi(x_t)$ with $x_t \in N_k(\boldsymbol{x}_i)$ and $\boldsymbol{\Gamma}^+$ denotes the pseudo-inverse of $\boldsymbol{\Gamma}$.

Please cite this article as: G. Krishnasamy, R. Paramesran, Hessian semi-supervised extreme learning machine, Neurocomputing (2016), http://dx.doi.org/10.1016/j.neucom.2016.05.039

Note, that the last $l(l+1)/2$ components of ν correspond to coefficients N_{rs} of the polynomial (up to rescaling for the diagonal components) and thus with Eq. (18), we obtain the desired $Z^{(i)}_{rst}$. Let $\varphi(x_\alpha)=\mathbf{f}_\alpha$, an estimate of the Frobenius norm of the Hessian of φ at $\boldsymbol{x}_i$ be given by:

$$\|\nabla_a\nabla_b\varphi\|^2 \approx \sum_{r,s=1}^{l}\left(\sum_{\alpha=1}^{k} Z^{(i)}_{rs\alpha}\mathbf{f}_{\boldsymbol{\alpha}}\right)^2 = \sum_{\alpha,\beta=1}^{k}\mathbf{f}_{\boldsymbol{\alpha}}f_\beta H^{(i)}_{\alpha,\beta} \tag{20}$$

where $H^{(i)}_{\alpha,\beta}=\sum_{r,s=1}^{l} Z^{(i)}_{rs\alpha}Z^{(i)}_{rs\beta}$ and finally the Hessian energy can be approximated as:

$$\hat{E}_H(\varphi)=\sum_{i=1}^{n}\sum_{\alpha\in N_k(x_i)}\sum_{\beta\in N_k(x_i)}\mathbf{f}_{\boldsymbol{\alpha}}\mathbf{f}_{\boldsymbol{\beta}}H^{(i)}_{\alpha\beta}=\boldsymbol{F}^T\boldsymbol{H}\boldsymbol{F} \tag{21}$$

where $\boldsymbol{H}=\sum_{i=1}^{n}H^{(i)}$ and n is the number of training data.

Hessian regularizer favors functions whose values vary linearly along the geodesic distance and it preserves the local manifold structure better compared to Laplacian regularizer. Hence, we incorporate Hessian regularizer into our framework to enhance the performance of semi-supervised learning.

3.2. HSS-ELM formulation

Following the framework of SS-ELM [33], we present our algorithm HSS-ELM based on Hessian regularization for semi-supervised learning. By modifying the formulation of SS-ELM, we can write the HSS-ELM objective function as:

$$\begin{aligned}\underset{\boldsymbol{\beta}\in\Re^{n_H\times n_K}}{\operatorname{argmin}}\quad & =\frac{1}{2}\|\boldsymbol{\beta}\|^2+\frac{1}{2}\sum_{i=1}^{l}C_i\|\boldsymbol{e}_i\|^2+\frac{\lambda}{2}\mathrm{Tr}(\boldsymbol{F}^T\boldsymbol{H}\boldsymbol{F})\\ \text{s.t.}\quad & \boldsymbol{\phi}(\boldsymbol{x}_i)\boldsymbol{\beta}=\boldsymbol{t}_i^T-\boldsymbol{e}_i^T,\quad i=1,2,\ldots,l\\ & \boldsymbol{f}_i=\boldsymbol{\phi}(\boldsymbol{x}_i)\boldsymbol{\beta}\quad i=1,2,\ldots,l+u\end{aligned} \tag{22}$$

where $\boldsymbol{H}\in\Re^{(l+u)\times(l+u)}$ is the Hessian operator derived from both labeled and unlabeled data. $\boldsymbol{F}\in\Re^{(l+u)\times n_K}$ is the matrix of output network with its ith row equal to $\boldsymbol{f}(\boldsymbol{x}_i)$. $\boldsymbol{\beta}\in\Re^{n_H\times n_K}$ represents the output weights while λ is a tradeoff parameter. In this paper, we utilize different coefficient C_i on the prediction errors for each sample from different classes similar to the works of Zong et al. [17] and Huang et al. [33]. As reported in Huang et al. [33], the authors found that in a skewed data set, certain classes might have more training data compared to other classes, which can lead to poor generalization. Thus, we compute the penalty coefficient as $C_i=C_0/N_{t_i}$, where N_{t_i} is the number of training samples in class t_i (if training sample x_i belongs to class t_i) and C_0 is the user-defined parameter as in ELM [17,33]. This computation method ascertains that the algorithm will not overfit the training samples from dominant classes and at the same time does not neglect the classes with less samples.

By substituting the constraints into the objective function, we can re-write (22) as:

$$\underset{\boldsymbol{\beta}\in\Re^{n_H\times n_K}}{\operatorname{argmin}}\quad =\frac{1}{2}\|\boldsymbol{\beta}\|^2+\frac{1}{2}\|\boldsymbol{C}^{\frac{1}{2}}(\boldsymbol{T}-\boldsymbol{\Phi}\boldsymbol{\beta})\|^2+\frac{\lambda}{2}\mathrm{Tr}(\boldsymbol{\beta}^T\boldsymbol{\Phi}^T\boldsymbol{H}\boldsymbol{\Phi}\boldsymbol{\beta}) \tag{23}$$

where $\boldsymbol{T}\in\Re^{(l+u)\times n_K}$ is the training target, whose first l rows are equal to T_l and the remaining rows are set to 0. $\boldsymbol{C}$ is a diagonal matrix of $(l+u)\times(l+u)$ where its first l diagonal elements $[\boldsymbol{C}]_{ii}=C_i$ and the remaining diagonal elements are set to 0.

Then, the gradient of the objective function (23) is computed to give:

$$\nabla L_{HSS-ELM}=\boldsymbol{\beta}+\boldsymbol{\Phi}^T\boldsymbol{C}(\boldsymbol{T}-\boldsymbol{\Phi}\boldsymbol{\beta})+\lambda\boldsymbol{\Phi}^T\boldsymbol{H}\boldsymbol{\Phi}\boldsymbol{\beta} \tag{24}$$

By setting the gradient to zero, the solution of HSS-ELM can be expressed as:

$$\boldsymbol{\beta}^*=(\boldsymbol{I}_{n_h}+\boldsymbol{\Phi}^T\boldsymbol{C}\boldsymbol{\Phi}+\lambda\boldsymbol{\Phi}^T\boldsymbol{H}\boldsymbol{\Phi})^{-1}\boldsymbol{\Phi}^T\boldsymbol{C}\boldsymbol{T} \tag{25}$$

If the number of data is less than the number of hidden nodes, we can write the alternative solution for HSS-ELM as:

$$\boldsymbol{\beta}^*=\boldsymbol{\Phi}^T(\boldsymbol{I}_{l+u}+\boldsymbol{C}\boldsymbol{\Phi}\boldsymbol{\Phi}^T+\lambda\boldsymbol{H}\boldsymbol{\Phi}\boldsymbol{\Phi}^T)^{-1}\boldsymbol{C}\boldsymbol{T} \tag{26}$$

where $\boldsymbol{I}_{l+u}$ is an $(l+u)$-by-$(l+u)$ identity matrix.

By observing (25) and (26), setting $\lambda=0$ and the diagonal matrix of C_i to a same constant, HSS-ELM solutions are reduced to a traditional ELM, which are expressed in (7) and (9). The detailed procedure of HSS-ELM is stated in Algorithm 1.

Regarding the time complexity of the proposed algorithm, we have two the following cases:

- When the number of data is larger than the number of hidden nodes, HSS-ELM solution is given in (25). One needs to calculate the Hessian regularization matrix $\boldsymbol{H}$ first, having the time complexity of $O(DN^2)$, where N is the number of data and D is the dimensionality of the data. Subsequently, a matrix inversion and three matrix multiplication steps having the time complexity of $O(n_H^3+n_HN^2)$ are required. Keeping only the higher order terms, the time complexity of (25) is equal to $O(N^3+n_HN^2)$
- When the number of hidden nodes is larger than the number of data, HSS-ELM solution is given in (26). As in the previous case, the Hessian matrix, $\boldsymbol{H}$ needs to be computed which requires a time complexity of $O(DN^2)$. Afterwards, a matrix inversion and three matrix multiplication steps having time complexity of $O(N^3+n_HN^2)$ are required. Keeping only the higher order terms, the time complexity of (26) is equal to $O(N^3+n_HN^2)$.

Algorithm 1. HSS-ELM algorithm.

Input:
Labeled data, $\{\boldsymbol{X}_l,\boldsymbol{T}_l\}=\{\boldsymbol{x}_i,\boldsymbol{t}_i\}_{i=1}^{l}$
Unlabeled data $\{\boldsymbol{X}_u\}=\{\boldsymbol{x}\}_{i=1}^{u}$
Output:
The mapping function of HSS-ELM: $\boldsymbol{f}:\Re^{n_D}\rightarrow\Re^{n_K}$
Step 1: Compute the Hessian operator, $\boldsymbol{H}$ from both labeled and unlabeled data
Step 2: Set up an ELM network of n_H hidden neurons with random input weights and biases
Step 3: Compute the output of the hidden neurons, $\boldsymbol{\Phi}\in\Re^{(l+u)\times n_H}$
Step 4:
if $\boldsymbol{n_H}\leq N$ **then**
Compute output weights $\boldsymbol{\beta}$ using (25)
else
Compute output weights $\boldsymbol{\beta}$ using (26)
Return The mapping function $\boldsymbol{f}(\boldsymbol{x})=\boldsymbol{\Phi}(x)\boldsymbol{\beta}$

4. Experimental results

In this section, we present the results of the experiments conducted to evaluate the performance of our proposed algorithm against the state-of-the-art semi-supervised algorithms such as the transductive SVM (TSVM) [48], LapSVM [25], LapRLS [25], HesSVM [40] and SS-ELM [33].

4.1. Data sets and experiment setup

We implement these algorithms on five standard data sets that are widely used for semi-supervised algorithm evaluation. These data sets are described as follows:

- The G50C is an artificial binary data set, where each class is generated from two unit covariance normal distribution with

Please cite this article as: G. Krishnasamy, R. Paramesran, Hessian semi-supervised extreme learning machine, Neurocomputing (2016), http://dx.doi.org/10.1016/j.neucom.2016.05.039

equal probabilities. The class means are modified so that the true Bayes error is 5%.

- The Columbia object image library (COIL20) is a set of 1440 gray-scale images of 20 different objects. Each sample represents a 32×32 gray scale image of an object acquired from a specific view. The COIL20(B) is a binary data set generated by grouping the first 10 objects in COIL20 to class 1 and the remaining objects to class 2.
- The USPST data set is a collection of hand-written digits from the USPS postal system. Each digit image is represented by a resolution of 16×16 pixels. The USPST(B) is a binary data set which was built by grouping the first 5 digits to class 1 and the remaining digits to class 2.

We followed the same experimental protocol as reported in Melacci and Belkin [28] and Huang et al. [33] in order to evaluate these semi-supervised algorithms. Specifically, we performed a 4-fold cross-validation on each data set, in which one of the 4-folds was used for testing (denoted as T) and the remaining folds were used for training. The training set was further divided into labeled data (L), unlabeled data (U) and validation data (V). We use the validation data (V) only for fine-tuning the hyperparameters of the classifier such as C and λ for the HSS-ELM algorithm. We repeated this random fold generation 3-times, resulting in a total of 12-splits. The details of the data sets are shown in Table 1.

The sigmoid activation function was used as the non-linear mapping for HSS-ELM. A uniform distribution of $(-1, 1)$ was utilized to generate the input weights and biases for HSS-ELM algorithm. The number of hidden neurons n_H was set to 1000 for the G50C data set and 2000 for the remaining data sets. The hyperparameters C and λ were chosen from an exponential sequence of $\{10^{-6}, 10^{-5}, \ldots, 10^{6}\}$ based on the validation errors.

Table 1
Description of the data sets.

Data sets	$\|L\|$	$\|U\|$	$\|V\|$	$\|T\|$
G50C	50	314	50	136
COIL20(B)	40	1000	50	360
USPST(B)	50	1409	50	498
COIL20	40	1000	50	360
USPST	50	1409	50	498

Table 2
Training times (seconds) for LapRLS, LapSVM, SS-ELM and HSS-ELM.

Data set	LapRLS	LapSVM	HesSVM	SS-ELM		HSS-ELM	
				$N \le n_H$	$N \ge n_H$	$N \le n_H$	$N \ge n_H$
G50C	0.367	0.381	0.426	0.253	0.173	0.358	0.235
COIL20(B)	1.061	1.015	1.205	1.127	0.285	1.325	0.495
USPST(B)	0.915	1.109	1.275	1.051	0.702	1.383	0.787
COIL20	11.938	11.213	12.877	1.263	0.527	2.416	0.764
USPST	15.483	16.635	17.512	1.089	0.474	1.624	0.618

4.2. Computational complexity analysis

Table 2 shows the training time for HSS-ELM, SS-ELM, LapRLS, LapSVM and HesSVM. These algorithms were implemented using Matlab R2012a on a 2.9 GHz machine with 8 GB memory. For both SS-ELM and HSS-ELM, we have considered for both cases when $N \le n_H$ and $N \ge n_H$, where N is the number of data and n_H is the number of hidden nodes. In the case when $N \le n_H$, n_H was set to 1000 for G50C data set while for the remaining four data sets, it was set to 2000. On the other hand, for the case when $N \ge n_H$, we have set n_H to 100 for G50C and 500 for the rest of the four data sets.

For the case when $N \le n_H$, the training times of HSS-ELM are slower when compared to SS-ELM, LapSVM, LapRLS and HesSVM for all binary data sets except for G50C. HesSVM achieved the slowest training time for G50C data set. This is due to the computation of the Hessian operator $\boldsymbol{H}$ in both HSS-ELM and HesSVM that takes a longer time to compute than the Laplacian $\boldsymbol{L}$ operator in LapRLS, LapSVM and SS-ELM. However, for the case when $N \ge n_H$, HSS-ELM takes lesser training time when compared to LapRLS, LapSVM and HesSVM.

In the case of multiclass data sets of COIL20 and USPST, training times of both ELM-based classifiers, SS-ELM and HSS-ELM are much faster than LapRLS, LapSVM and HesSVM for both cases. As expected, ELM-based classifiers are usually more computationally efficient than the SVM-based classifiers for multi-class classification [10,33]. However, HSS-ELM takes longer training time than SS-ELM for multiclass data sets due to the computation complexity of the Hessian operator.

Table 3
Performance comparison of the HSS-ELM algorithm with pure supervised algorithms and semi-supervised algorithms.

Data sets	Subset	SVM	ELM	TSVM	LapRLS	LapSVM	SS-ELM	HesSVM	HSS-ELM
G50C	U	9.33 ± 2.00	8.91 ± 2.29	5.43 ± 1.11	6.03 ± 1.32	5.52 ± 1.15	5.24 ± 1.17	5.23 ± 1.07	5.05 ± 0.62
	V	9.83 ± 3.46	8.20 ± 3.05	4.67 ± 1.81	6.17 ± 3.66	5.67 ± 2.67	4.07 ± 0.95	4.00 ± 2.31	3.67 ± 1.27
	T	10.06 ± 2.80	9.20 ± 2.07	4.96 ± 1.37	6.54 ± 2.11	5.51 ± 1.65	4.96 ± 1.53	4.54 ± 1.34	4.15 ± 0.76
COIL20	U	16.23 ± 2.63	11.28 ± 1.95	13.68 ± 3.09	8.07 ± 2.05	8.31 ± 2.19	6.04 ± 1.26	5.83 ± 0.71	5.55 ± 1.01
	V	18.54 ± 6.20	11.30 ± 2.29	12.25 ± 3.99	7.92 ± 3.96	8.13 ± 4.01	5.95 ± 1.68	5.50 ± 2.23	4.20 ± 1.51
	T	15.93 ± 3.00	12.22 ± 2.00	15.19 ± 2.52	8.59 ± 1.90	8.68 ± 2.04	7.33 ± 2.15	6.19 ± 0.49	5.86 ± 1.54
USPST(B)	U	17.00 ± 2.74	17.49 ± 2.44	22.75 ± 2.68	8.87 ± 1.88	8.84 ± 2.20	9.24 ± 2.79	8.36 ± 0.40	9.13 ± 1.51
	V	18.17 ± 5.94	17.40 ± 2.01	21.40 ± 3.78	10.17 ± 4.55	8.67 ± 4.38	9.40 ± 2.07	8.67 ± 0.94	8.80 ± 0.44
	T	17.10 ± 3.21	17.98 ± 2.86	22.06 ± 2.52	9.42 ± 2.51	9.68 ± 2.48	10.98 ± 2.99	9.24 ± 0.62	9.83 ± 1.64
COIL20	U	29.49 ± 2.24	29.23 ± 1.68	24.61 ± 3.16	10.35 ± 2.30	10.51 ± 2.06	10.92 ± 2.05	10.37 ± 0.55	10.34 ± 1.39
	V	31.46 ± 7.79	29.25 ± 2.81	21.25 ± 3.63	9.79 ± 4.94	9.79 ± 4.94	10.75 ± 3.02	9.50 ± 0.92	8.75 ± 1.76
	T	28.98 ± 2.74	29.19 ± 3.61	23.83 ± 3.61	11.30 ± 2.17	11.44 ± 2.39	11.16 ± 2.39	10.31 ± 0.35	10.19 ± 1.83
USPST	U	23.84 ± 3.26	23.91 ± 2.36	18.92 ± 2.81	15.12 ± 2.90	14.36 ± 2.55	13.51 ± 1.89	13.22 ± 1.51	13.20 ± 1.43
	V	24.67 ± 4.54	24.00 ± 2.33	17.53 ± 4.29	14.67 ± 3.94	15.17 ± 4.04	13.47 ± 2.65	14.75 ± 0.71	12.83 ± 1.76
	T	23.60 ± 2.32	23.85 ± 2.71	18.92 ± 3.19	16.44 ± 3.53	14.91 ± 2.83	13.85 ± 2.41	13.81 ± 1.39	13.78 ± 1.11

Please cite this article as: G. Krishnasamy, R. Paramesran, Hessian semi-supervised extreme learning machine, Neurocomputing (2016), http://dx.doi.org/10.1016/j.neucom.2016.05.039

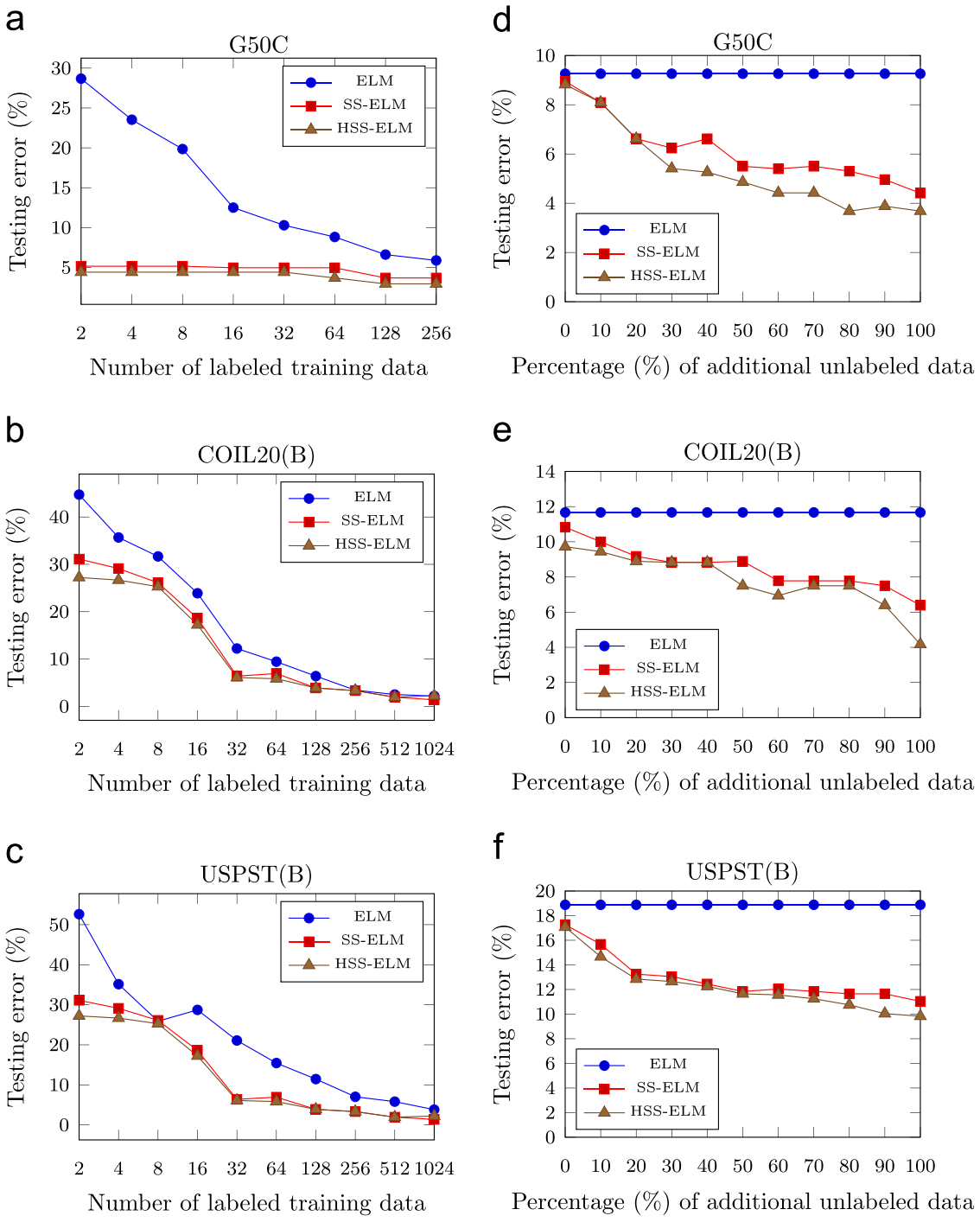

Fig. 1. Testing error with different number of labeled data: (a) G50C; (b) COIL20(B); and (c) USPST(B) and testing error with different number of unlabeled data: (d) G50C; (e) COIL20(B); and (f) USPST(B).

4.3. *Performance comparison with related algorithms*

The performance of our proposed algorithm on these selected data sets was compared with two supervised algorithms and five state-of-the-art semi-supervised algorithms. Two supervised algorithms, ELM and SVM were included as the baseline classifiers. The following semi-supervised algorithms were used: TSVM, LapRLS, LapSVM, HesSVM and HSS-ELM. We used the classification error rate ($\pm$ standard deviation) on unlabeled set (U), validation set (V) and test set (T) to evaluate the performances of these algorithms. Table 3 shows the results for these algorithms. The results for ELM, SVM, TSVM, LapRLS, LapSVM and SS-ELM were obtained from Huang et al. [33].

4.3.1. *Comparison between the proposed algorithm and supervised algorithms*

Our proposed method was compared with two supervised algorithms, ELM and SVM. From Table 3, the results show that HSS-ELM performs better than SVM and ELM for all data sets. These results demonstrate that the proposed algorithm is able to efficiently leverage the unlabeled data to yield a better performance when compared to supervised algorithms such as SVM and

Please cite this article as: G. Krishnasamy, R. Paramesran, Hessian semi-supervised extreme learning machine, Neurocomputing (2016), http://dx.doi.org/10.1016/j.neucom.2016.05.039

ELM. For example, the test error obtained using HSS-ELM is 4.15% while the test errors for ELM and SVM are 9.20% and 10.06% respectively for the G50C data set.

4.3.2. Comparison between the proposed algorithm and semi-supervised algorithms

The proposed method was compared with five semi-supervised algorithms. The results in Table 3 show that our proposed algorithm performed better than the other semi-supervised algorithms for all data sets except for USPST(B). For the USPST(B) data set, HesSVM gave the lowest classification error for unlabeled set, validation set and test set.

Table 3 also reveals that HSS-ELM performed better than SS-ELM for all data sets. These results indicate that Hessian regularization favors functions whose values vary linearly with respect to geodesic distances and it preserves the local manifold structure better than Laplacian regularization, hence it can boost ELMs semi-supervised learning performances.

4.4. Performance with different number of labeled and unlabeled training data

In this section, the performances of HSS-ELM, SS-ELM and ELM using different numbers of label and unlabeled data are investigated. Fig. 1(a), (b) and (c) shows the performances of these algorithms on G50C, COIL20(B) and USPST(B) with different number of labeled data respectively. Using the same experimental setup as reported in Huang et al. [33], we varied the proportion of the labeled and unlabeled data in the training set. From Fig. 1, it can be seen that HSS-ELM performed better than SS-ELM and ELM when a small amount of labeled data were used in the experiments. This shows that Hessian regularization based approaches are better than Laplacian regularization based approaches when the number of labeled data is small.

The performances of HSS-ELM, SS-ELM and ELM with different numbers of unlabeled data are shown in Fig. 1(d), (e) and (f) respectively. Again, using the same experimental protocol as in [33], we incrementally added the unlabeled samples to the unlabeled set (U) in increments of 10%, while the labeled set (L), test set (T) and validation set (V) remained unchanged. From Fig. 1(d)–(f), it can observed that the test error drops significantly when more unlabeled samples are added to the unlabeled set (U). Even without any labeled data, HSS-ELM performed better than ELM. This phenomenon is consistent with Belkin et al. [25], where manifold regularization is also effective for pure supervised learning.

5. Conclusion

In this paper, we proposed a new algorithm called HSS-ELM, where the traditional ELM was extended for semi-supervised learning. We have incorporated Hessian regularization into ELM, which has more favorable properties for semi-supervised learning than Laplacian regularization. The Hessian regularization allows functions that extrapolate, i.e. functions whose values are not limited to the range of the training output. This extrapolation ability leads to significant improvement on performance especially when only few labeled data are available. Additionally, the proposed algorithm inherits almost all the advantages of ELM such as the exceptional training efficiency and a straight forward implementation for multiclass classification problems. Experimental results show that HSS-ELM is competitive with other state-of-the-art semi-supervised algorithms and it requires significantly less training time when compared to LapRLS, LapSVM and HesSVM on the multiclass classification problems. Our proposed algorithm can easily be applied for pattern recognition tasks such as human action recognition, image annotation and speech recognition that require a semi-supervised learning algorithm.

Acknowledgment

This work was supported by the HIR-MOHE Grant No. UM.C/625/1/HIR/MOHE/ENG/42.

References

[1] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: a new learning scheme of feedforward neural networks, in: Proceedings of the 2004 IEEE International Joint Conference on Neural Networks, vol. 2, 2004, pp. 985–990.
[2] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagation errors, Nature 323 (1986) 533–536.
[3] M. Hagan, M. Menhaj, Training feedforward networks with the Marquardt algorithm, IEEE Trans. Neural Netw. 5 (1994) 989–993.
[4] V.N. Vapnik, The Nature of Statistical Learning Theory, Springer-Verlag New York, Inc, New York, NY, USA, 1995.
[5] C. Cortes, V. Vapnik, Support-vector networks, Mach. Learn. 20 (1995) 273–297.
[6] J.A.K. Suykens, J. Vandewalle, Least squares support vector machine classifiers, Neural Process. Lett. 9 (1999) 293–300.
[7] Y. Tian, M. Fu, F. Wu, Steel plates fault diagnosis on the basis of support vector machines, Neurocomputing 151 (2015) 296–303 (Part 1).
[8] S. Yin, X. Zhu, C. Jing, Fault detection based on a robust one class support vector machine, Neurocomputing 145 (2014) 263–268.
[9] X. Huang, S. Mehrkanoon, J.A. Suykens, Support vector machines with piece-wise linear feature mapping, Neurocomputing 117 (2013) 118–127.
[10] G.-B. Huang, H. Zhou, X. Ding, R. Zhang, Extreme learning machine for regression and multiclass classification, IEEE Trans. Syst. Man Cybern. Part B: Cybern. 42 (2012) 513–529.
[11] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: theory and applications, Neurocomputing 70 (2006) 489–501 (Neural Networks Selected Papers from the 7th Brazilian Symposium on Neural Networks, SBRN'04).
[12] P. 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 Trans. Inf. Theory 44 (1998) 525–536.
[13] L.-C. Shi, B.-L. Lu, Eeg-based vigilance estimation using extreme learning machines, Neurocomputing 102 (2013) 135–143 (Advances in Extreme Learning Machines, ELM 2011).
[14] S. Wong, K. Yap, H. Yap, S. Tan, A truly online learning algorithm using hybrid fuzzy artmap and online extreme learning machine for pattern classification, Neural Process. Lett. (2014) 1–18.
[15] J. Zhao, Z. Wang, D.S. Park, Online sequential extreme learning machine with forgetting mechanism, Neurocomputing 87 (2012) 79–89.
[16] X. Jia, R. Wang, J. Liu, D.M. Powers, A semi-supervised online sequential extreme learning machine method, Neurocomputing 174 (2016) 168–178 (Part A).
[17] W. Zong, G.-B. Huang, Y. Chen, Weighted extreme learning machine for imbalance learning, Neurocomputing 101 (2013) 229–242.
[18] K. Li, X. Kong, Z. Lu, L. Wenyin, J. Yin, Boosting weighted ELM for imbalanced learning, Neurocomputing 128 (2014) 15–21.
[19] P. Horata, S. Chiewchanwattana, K. Sunat, Robust extreme learning machine, Neurocomputing 102 (2013) 31–44 (Advances in Extreme Learning Machines, ELM 2011).
[20] Q. Yu, Y. Miche, E. Eirola, M. van Heeswijk, E. Séverin, A. Lendasse, Regularized extreme learning machine for regression with missing data, Neurocomputing 102 (2013) 45–51 (Advances in Extreme Learning Machines, ELM 2011).
[21] S.Y. Wong, K.S. Yap, H.J. Yap, A constrained optimization based extreme learning machine for noisy data regression, Neurocomputing 171 (2016) 1431–1443.
[22] O. Chapelle, B. Schlkopf, A. Zien, Semi-Supervised Learning, 1st ed., The MIT Press, 2010.
[23] X. Zhu, Semi-Supervised Learning Literature Survey (2006).
[24] M. Belkin, P. Niyogi, Laplacian eigenmaps and spectral techniques for embedding and clustering, in: T.G. Dietterich, S. Becker, Z. Ghahramani (Eds.), Advances in Neural Information Processing Systems, vol. 14, 2001, pp. 585–591.
[25] M. Belkin, P. Niyogi, V. Sindhwani, Manifold regularization: a geometric framework for learning from labeled and unlabeled examples, J. Mach. Learn. Res. 7 (2006) 2399–2434.
[26] S. Gao, I. Tsang, L.-T. Chia, P. Zhao, Local features are not lonely-Laplacian sparse coding for image classification, in: 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010, pp. 3555–3561.
[27] S. Gao, I.-H. Tsang, L.-T. Chia, Laplacian sparse coding, hypergraph Laplacian sparse coding, and applications, IEEE Trans. Pattern Anal. Mach. Intell. 35 (2013) 92–104.
[28] S. Melacci, M. Belkin, Laplacian support vector machines trained in the primal, J. Mach. Learn. Res. 12 (2011) 1149–1184.

[29] Z. Ma, F. Nie, Y. Yang, J. Uijlings, N. Sebe, A. Hauptmann, Discriminating joint feature analysis for multimedia data understanding, IEEE Trans. Multimed. 14 (2012) 1662–1672.
[30] Y. Wang, S. Chen, H. Xue, Z. Fu, Semi-supervised classification learning by discrimination-aware manifold regularization, Neurocomputing 147 (2015) 299–306 (Advances in Self-Organizing Maps Subtitle of the special issue: Selected Papers from the Workshop on Self-Organizing Maps 2012, WSOM 2012).
[31] J. Liu, Y. Chen, M. Liu, Z. Zhao, Selm: semi-supervised ELM with application in sparse calibrated location estimation, Neurocomputing 74 (2011) 2566–2572.
[32] A. Iosifidis, A. Tefas, I. Pitas, Semi-supervised classification of human actions based on neural networks, in: 2014 22nd International Conference on Pattern Recognition, ICPR, 2014, pp. 1336–1341.
[33] G. Huang, S. Song, J. Gupta, C. Wu, Semi-supervised and unsupervised extreme learning machines, IEEE Trans. Cybern. 44 (2014) 2405–2417.
[34] Y. Zhou, B. Liu, S. Xia, B. Liu, Semi-supervised extreme learning machine with manifold and pairwise constraints regularization, Neurocomputing 149 (2015) 180–186 (Part A, Advances in Neural Networks, Selected Papers from the Tenth International Symposium on Neural Networks, ISNN 2013, Advances in Extreme Learning Machine, Selected Articles from the International Symposium on Extreme Learning Machines, ELM 2013).
[35] Y. Gu, Y. Chen, J. Liu, X. Jiang, Semi-supervised deep extreme learning machine for wi-fi based localization, Neurocomputing 166 (2015) 282–293.
[36] K.I. Kim, F. Steinke, M. Hein, Semi-supervised regression using hessian energy with an application to semi-supervised dimensionality reduction, in: Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, A. Culotta (Eds.), Advances in Neural Information Processing Systems, vol. 22, Curran Associates, Inc., 2009, pp. 979–987.
[37] J. Eells, L. Lemaire, Selected Topics in Harmonic Maps, Published for the Conference Board of the Mathematical Sciences by the American Mathematical Society Providence, R.I., 1983.
[38] D.L. Donoho, C. Grimes, Hessian eigenmaps: new locally linear embedding techniques fo high-dimensional data, Proc. Nat. Acad. Sci. USA 100 (10), 2003, 5591–5596.
[39] F. Steinke, M. Hein, Non-parametric regression between manifolds, in: D. Koller, D. Schuurmans, Y. Bengio, L. Bottou (Eds.), Advances in Neural Information Processing Systems, vol. 21, Curran Associates, Inc., 2009, pp. 1561–1568.
[40] D. Tao, L. Jin, W. Liu, X. Li, Hessian regularized support vector machines for mobile image annotation on the cloud, IEEE Trans. Multimed. 15 (2013) 833–844.
[41] W. Liu, D. Tao, Multiview Hessian regularization for image annotation, IEEE Trans. Image Process. 22 (2013) 2676–2687.
[42] W. Liu, Y. Li, X. Lin, D. Tao, Y. Wang, Hessian-regularized co-training for social activity recognition, PLoS ONE 9 (2014) 1–10.
[43] M. Zheng, J. Bu, C. Chen, Hessian sparse coding, Neurocomputing 123 (2014) 247–254 (Contains Special Issue Articles: Advances in Pattern Recognition Applications and Methods).
[44] W. Liu, D. Tao, J. Cheng, Y. Tang, Multiview Hessian discriminative sparse coding for image annotation, Comput. Vis. Image Underst. 118 (2014) 50–60 (Cited By 80).
[45] C. Shi, Q. Ruan, G. An, R. Zhao, Hessian semi-supervised sparse feature selection based on $l_{2,1/2}$-matrix norm, IEEE Trans. Multimed. 17 (2015) 16–28.
[46] V. Sindhwani, P. Niyogi, M. Belkin, Beyond the point cloud: from transductive to semi-supervised learning, in: Proceedings of the 22nd International Conference on Machine Learning, ICML'05, ACM, New York, NY, USA, 2005, pp. 824–831.
[47] J.M. Lee, Riemannian Manifolds – An Introduction to Curvature, Springer, New York, NY, USA, 1997.
[48] T. Joachims, Transductive inference for text classification using support vector machines, in: Proceedings of the Sixteenth International Conference on Machine Learning, ICML'99, Morgan Kaufmann Publishers Inc, San Francisco, CA, USA, 1999, pp. 200–209.

Ganesh Krishnasamy received the B.Eng. and M.Eng. degrees in electrical and electronic engineering from the Universiti Kebangsaan Malaysia in 2004 and 2007 respectively. He is currently pursuing the Ph.D. degree in electrical engineering with the Department of Electrical Engineering, University of Malaya, Malaysia. His current research interests include the field of computer vision, machine learning and optimization.

Raveendran Paramesran received the B.Sc. and M.Sc. degrees in electrical engineering from South Dakota State University, Brookings, South Dakota, USA in 1984 and 1985 respectively. He was a systems designer with Daktronics, USA before joining the Department of Electrical Engineering at University of Malaya, Kuala Lumpur, as a lecturer in 1986. In 1992, he received a Ronpaku scholarship from Japan to pursue Doctorate in Engineering, which he completed in 1994 at University of Tokushima, Japan. He was promoted as associate professor in 1995 and was promoted as professor in 2003. His research areas include image and video analysis, formulation of new image descriptors for image analysis, fast computation of orthogonal moments, analysis of EEG signals, and data modeling of substance concentration acquired from non-invasive methods. His contributions can be seen in the form of journal publications, conference proceedings, chapters in books and an international patent to predict blood glucose levels using non-parametric model. He has successfully supervised the completion of 10 Ph.D. students and 12 students in M.Eng.Sc. (Masters by research). His current research interests include image and video analysis, formulation of new image descriptors for image analysis, fast computation of orthogonal moments, analysis of electroencephalography signals, and data modeling of substance concentration acquired from non-invasive methods.

Please cite this article as: G. Krishnasamy, R. Paramesran, Hessian semi-supervised extreme learning machine, Neurocomputing (2016), http://dx.doi.org/10.1016/j.neucom.2016.05.039