Neurocomputing 149 (2015) 171–179

Contents lists available at ScienceDirect

Neurocomputing

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

Extreme spectral regression for efficient regularized subspace learning

Bing Liu, Shi-Xiong Xia*, Fan-Rong Meng, Yong Zhou

School of Computer Science and Technology, China University of Mining, and Technology, Jiangsu, Xuzhou 221116, China

ARTICLE INFO

Article history:
Received 6 August 2013
Received in revised form
26 September 2013
Accepted 30 September 2013
Available online 30 September 2014

Keywords:
Spectral regression
Extreme learning machine (ELM)
Dimensionality reduction
Out-of-sample extension

ABSTRACT

Traditional manifold learning algorithms, such as Locally Linear Embedding, Isomap and Laplacian Eigenmap, only provide the embedding results of training samples. Although many extensions of these approaches try to solve the out-of-sample extension problem, their computations cannot avoid eigen-decomposition of dense matrices which is expensive in both time and memory. To solve this problem, spectral regression (SR) casts the problem of learning an embedding function into a regression framework. Motivated by the effectiveness of extreme learning machine (ELM), in this paper, we solve the out-of-sample extension problem by seeking an embedding function in ELM feature space. An extreme spectral regression (ESR) algorithm is proposed to speed up kernel-based SR (KSR) further. In addition, it is proved that ESR is an approximation of KSR. Similar to SR, the proposed ESR algorithm can be performed in supervised, unsupervised and semi-supervised situation. Experimental results on classification and semi-supervised classification demonstrate the effectiveness and efficiency of our algorithm.

1. Introduction

Dimensionality reduction has been a key problem in many fields of information processing, such as machine learning, data mining, information retrieval, and pattern recognition. Practical algorithms usually behave badly when faced with many unnecessary features. A common way to attempt to resolve this problem is to use dimensionality reduction (DR) techniques, which include unsupervised, supervised and semi-supervised DR due to different assumptions about the data distribution or the availability of the data labeling.

Principal component analysis (PCA) [1] is one of the most popular unsupervised DR techniques, which finds a linear mapping by maximizing the projected variances. If the data is embedded in a linear subspace, PCA is guaranteed to discover the dimensionality of the subspace and produces a compact representation. In order to handle the data sampled from a nonlinear low dimensional manifold, many manifold learning techniques, such as ISOMAP [2], Locally Linear Embedding (LLE) [3] and Laplacian Eigenmap [4] have been proposed which reduce the dimensionality of a fixed training set in a way that can maximally preserve certain inter-point relationships. Given the data of each class have a Gaussian distribution, Linear discriminant analysis (LDA) first constructs the between-class scatter and the within-class scatter matrices by virtue of labeled data, then simultaneously maximizes the between-class scatter and minimizes the within-class scatter to obtain a projection. Alternatively, marginal Fisher analysis (MFA) [5] and local discriminant embedding (LDE) [6] exploit the assumption that the data of each class spread as a submanifold, and seek a discriminant embedding over these submanifolds. Semi-supervised DR, such as semi-supervised discriminant analysis (SDA) [7], utilizes partially labeled data while preserving the intrinsic geometric structures of the remaining. Generally, linear DR methods mentioned above can be kernelized into nonlinear ones. As shown in [8–10], the kernelized versions can achieve significant improvements.

Although some modified methods explicitly require an embedding function either linear or in the Reproducing Kernel Hilbert Space (RKHS) when minimizing the objective function [11,12], the computation of these methods involves eigen-decomposition of dense matrices which is expensive in both time and memory. It is almost infeasible to apply these approaches on large data sets. Spectral regression (SR), which is fundamentally based on regression and spectral graph analysis [13–16], casts the problem of learning an embedding function into a regression framework. In this method, an affinity graph over both labeled and unlabeled points is first constructed to discover the intrinsic discriminant structure in the data. The responses for both labeled and unlabeled points are then obtained by means of this graph. Finally, the ordinary regression is applied for learning the embedding function. Thus, SR avoids eigen-decomposition of dense matrices. Moreover, it can be performed either in supervised, unsupervised or semi-supervised situation.

Kernel SR (KSR) is the kernelized version of SR in the reproducing kernel Hilbert space (RKHS) into which data points are

* Corresponding author.
E-mail address: xiasx@cumt.edu.cn (S.-X. Xia).

http://dx.doi.org/10.1016/j.neucom.2013.09.073
0925-2312/

Table 1
Notations.

Notations	Descriptions
$\mathbb{R}^d$	the input d-dimensional Euclidean space
n	the number of total training data points
m	the number of classes that the samples belong to
$\boldsymbol{X}$	$\boldsymbol{X}=[\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n]\in\mathbb{R}^{d\times n}$ is the training data matrix
$k(\boldsymbol{x},\boldsymbol{y})$	Kernel function of variables $\boldsymbol{x}$ and $\boldsymbol{y}$
$\boldsymbol{K}$	Kernel matrix $\boldsymbol{K}=\{k(\boldsymbol{x}_i,\boldsymbol{x}_j)\}\in\mathbb{R}^{n\times n}$
$\|\ \|$	norm in the Hilbert space $\mathscr{H}$
$\boldsymbol{H}$	the hidden-layer output matrix
$\boldsymbol{\beta}$	the vector of the output weights between the hidden layer of L nodes and the output node
L	the graph Laplacian matrix

mapped. For large data sets, the computation of KSR involves an inversion of the kernel matrix of the training data, which is time-consuming in applications. To solve this problem, motivated by the fast learning speed of extreme learning machine (ELM), we solve the out-of-sample extension problem by seeking an embedding function in ELM feature space instead of RKHS. Thus, the final regression problem can be solved effectively by the ELM algorithm, which uses L_2 regularization to solve the over-fitting problem. The regularization methods for solving the regression problem of ELM have been studied extensively. Miche et al. in [17,18] proposed a double regularized ELM algorithm, which uses a cascade of two regularization penalties: first a L_1 penalty to rank the neurons of the hidden layer, followed by a L_2 penalty to prune the network accordingly. In this paper, we propose an extreme spectral regression (ESR) algorithm, which is based on ELM only using L_2 regularization for simplicity. In this case, it is proved that ESR is an approximation of KSR by discretizing the embedding functions in RKHS. Experimental results on classification and semi-supervised classification demonstrate the effectiveness and efficiency of our algorithm.

The paper is structured as follows. In Section 2, we briefly introduce the extreme learning machine model. Our extreme spectral regression algorithm is introduced in Section 3. In Section 4, we provide a theoretical and computational complexity analysis of our algorithm respectively. The experimental results are presented in Section 5. Finally, we give the related conclusions in Section 6. In order to avoid confusion, we give a list of the main notations used in this paper in Table 1.

2. Extreme learning machine

The output function of ELM for generalized SLFNs in the case of one output node is

$$f_L(\boldsymbol{x})=\sum_{i=1}^{L}\beta_i h_i(\boldsymbol{x})=\boldsymbol{h}(\boldsymbol{x})\boldsymbol{\beta}, \tag{1}$$

where $\boldsymbol{\beta}=[\beta_1,\ \ldots,\beta_L]^T$ is the vector of the output weights between the hidden layer of L nodes and the output node, and $\boldsymbol{h}(\boldsymbol{x})=[h_1(\boldsymbol{x}),\ \ldots,h_L(\boldsymbol{x})]$ is the output (row) vector of the hidden layer with respect to the input $\boldsymbol{x}$. In fact, $\boldsymbol{h}(\boldsymbol{x})$ maps the data from the d-dimensional input space to the L-dimensional hidden-layer feature space (ELM feature space) $\boldsymbol{H}$. ELM is to minimize the training error as well as the norm of the output weights [18–20]

$$\min_{\beta}\ \frac{C}{2}\|H\boldsymbol{\beta}-\boldsymbol{T}\|^2+\frac{1}{2}\|\boldsymbol{\beta}\|^2, \tag{2}$$

where C is a tradeoff parameter between the complexity and fitness of the decision function and $\boldsymbol{H}$ is the hidden-layer output matrix denoted by

$$\boldsymbol{H}=\begin{bmatrix}\boldsymbol{h}(\boldsymbol{x}_1)\\ \boldsymbol{h}(\boldsymbol{x}_2)\\ \vdots\\ \boldsymbol{h}(\boldsymbol{x}_n)\end{bmatrix}=\begin{bmatrix}h_1(\boldsymbol{x}_1) & \ldots & h_L(\boldsymbol{x}_1)\\ h_1(\boldsymbol{x}_2) & \ldots & h_L(\boldsymbol{x}_2)\\ \vdots & \vdots & \vdots\\ h_1(\boldsymbol{x}_n) & \ldots & h_L(\boldsymbol{x}_n)\end{bmatrix}. \tag{3}$$

Similar to support vector machine (SVM), to minimize the norm of the output weights $\|\boldsymbol{\beta}\|$ is actually to maximize the distance of the separating margins of the two different classes in the ELM feature space: $2/\|\boldsymbol{\beta}\|$, which actually controls the complexity of the function in the ELM feature space.

For completeness, we briefly introduce the multiclass classifiers of ELM.

(1) *Multiclass classifier with single output:* ELM can approximate any target continuous functions and the output of the ELM classifier $\boldsymbol{h}(\boldsymbol{x})\boldsymbol{\beta}$ can be as close to the class labels in the corresponding regions as possible. Thus the classification problem for ELM with a single-output node can be formulated as [21]:

$$\begin{aligned}&\text{Minimize}:\ L_{\text{ELM}}=\tfrac{1}{2}\|\boldsymbol{\beta}\|^2+\tfrac{C}{2}\sum_{i=1}^{n}\varepsilon_i^2\\ &\text{Subject to}:\ \boldsymbol{h}(\boldsymbol{x}_i)\boldsymbol{\beta}=t_i-_i,\ i=1,\ldots,n.\end{aligned} \tag{4}$$

For multiclass problems, among all the multiclass labels, the predicted class label of a given testing sample is the closest to the output of ELM classifier.

(2) *Multiclass classifier with multioutputs:* If ELM has multioutput nodes, an m-class classifier is corresponding to m output nodes. If the original class label is l, the expected output vector of the m output nodes is $\boldsymbol{t_i}=[\overbrace{0,\ldots,0,1}^{l},0,\ldots,0]^T$. That is, the lth element of $\boldsymbol{t_i}=[t_{i,1},\ldots,t_{i,m}]^T$ is one and the rest of the elements are zero. The classification problem for ELM with multioutput nodes is [21]:

$$\begin{aligned}&\text{Minimize}:\ L_{\text{ELM}}=\tfrac{1}{2}\|\boldsymbol{\beta}\|^2+\tfrac{C}{2}\sum_{i=1}^{n}\|\boldsymbol{\varepsilon}_i\|^2\\ &\text{Subject to}:\ \boldsymbol{h}(\boldsymbol{x}_i)\boldsymbol{\beta}=\boldsymbol{t}_i^T-\boldsymbol{\varepsilon}_i^T,\ i=1,\ldots,n.\end{aligned} \tag{5}$$

where $\boldsymbol{\varepsilon}_i=[\varepsilon_{i1},\ldots,\varepsilon_{im}]^T$ is the training error vector of the m output nodes with respect to the training sample $\boldsymbol{x_i}$.

If a feature mapping $\boldsymbol{h}(\boldsymbol{x})$ is unknown to users, the output function of ELM classifier is

$$f(x)=\boldsymbol{h}(\boldsymbol{x})\boldsymbol{H}^T\left(\frac{\boldsymbol{I}}{C}+\boldsymbol{H}\boldsymbol{H}^T\right)^{-1}\boldsymbol{T}=[k(\boldsymbol{x},\boldsymbol{x}_1),\ldots,k(\boldsymbol{x},\boldsymbol{x}_n)]\left(\frac{\boldsymbol{I}}{C}+\boldsymbol{M}\right)^{-1}\boldsymbol{T}, \tag{6}$$

where $\boldsymbol{M}=\boldsymbol{H}\boldsymbol{H}^T$, $m_{i,j}=k(\boldsymbol{x}_i,\boldsymbol{x}_j)$ and $k(\boldsymbol{x},\boldsymbol{y})$ is a positive semi-definite kernel function. If a feature mapping $\boldsymbol{h}(\boldsymbol{x})$ is known, we have $\boldsymbol{h}(\boldsymbol{x})=[G(a_1,b_1,\boldsymbol{x}),\ldots,G(a_L,b_L,\boldsymbol{x})]$, where $G(a,b,\boldsymbol{x})$ is a nonlinear piecewise continuous function satisfying ELM universal approximation capability theorems [22–26] and $\{(a_i,b_i)\}_{i=1}^{L}$ are randomly generated according to any continuous probability distribution. The output function of ELM classifier is

$$f(\boldsymbol{x})=\boldsymbol{h}(\boldsymbol{x})\boldsymbol{H}^T\left(\frac{\boldsymbol{I}}{C}+\boldsymbol{H}\boldsymbol{H}^T\right)^{-1}\boldsymbol{T}, \tag{7}$$

or

$$f(\boldsymbol{x})=\boldsymbol{h}(\boldsymbol{x})\left(\frac{\boldsymbol{I}}{C}+\boldsymbol{H}^T\boldsymbol{H}\right)^{-1}\boldsymbol{H}^T\boldsymbol{T}, \tag{8}$$