Neurocomputing 113 (2013) 58–66

Contents lists available at SciVerse ScienceDirect

Neurocomputing

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

Adaptive kernel subspace method for speeding up feature extraction

Zhaoyang Zhang [a,*], Zheng Tian [a,b], Xifa Duan [a], Xiangyun Fu [a]

[a] School of Science, Northwestern Polytechnical University, Xi'an 710129, China
[b] State Key Laboratory of Remote Sensing Science, Institute of Remote Sensing Applications, Chinese Academy of Sciences, Beijing 100101, China

ARTICLE INFO

Article history:
Received 11 May 2012
Received in revised form
19 December 2012
Accepted 5 January 2013
Communicated by: Xiaofei He
Available online 1 March 2013

Keywords:
Kernel principal component analysis (KPCA)
Feature extraction
Adaptive kernel subspace
Spanning vectors

ABSTRACT

KPCA can extract nonlinear features of data set. However, its efficiency is in inverse proportion to the size of the training sample set. In this paper, we proposed an adaptive kernel subspace method to extract features efficiently. The method is methodologically consistent with KPCA, and can improve the efficiency by adaptively selecting the spanning vectors of the kernel principal components, meanwhile, not affect the accuracy much. Experiments on two-dimensional data, MNIST dataset and USPS dataset show that the feature extraction method is more efficient than that associated with KPCA and reference methods.

1. Introduction

Kernel principal component analysis (KPCA) [1,2], which is a nonlinear extension of principal component analysis (PCA) [3], is widely used to extract nonlinear features of data set. The core idea of KPCA is to first map the input space into a feature space using a nonlinear mapping and then compute the principal components in the feature space. As a result, the extracted kernel principal component of the mapped data is nonlinear with regards to the original input space, and the extracted kernel principal components are expanded in terms of all training samples in the feature space. Thus, if we use KPCA to extract features of a sample, all the kernel functions between this sample and the total training samples in the feature space should be computed. As a result, the larger the size of training sample set is, the lower the efficiency of feature extraction becomes. This is called the batch nature of the KPCA. The batch nature binders KPCA in terms of computation and memory demand as the data size increases. Especially, for real world applications with large numbers of training samples, the KPCA based feature extraction is inefficient and even unfeasible. Indeed, other kernel approaches also suffer from similar problem, such as kernel Fisher discriminant analysis [4], kernel nonlinear regression [5] and Kernel Canonical Correlation Analysis [6].

In recent years, some algorithms have been proposed to deal with the low efficiency induced by the batch nature of kernel method in feature extracting. Generally, these algorithms primarily root in the following two ideas. The first idea is based on the supposition that at least one training sample in the feature space can be exactly expressed as a linear combination of the others. The second idea is that the principal component may be expanded approximately in terms of some training samples in the feature space, the number of which is fewer than the number of total training samples. For the first idea, it is only reasonable and feasible for linearly dependent training samples because there is at least one training sample that can be exactly expressed as a linear combination of the others. However, for some real world applications associated with Gaussian kernel function, the training samples in the feature space do not meet the supposition of the first idea. For the second idea, it develops only with the viewpoint of numerically approximating the principal component ground truth in the feature space. The typical works are expectation maximization approach [7] proposed by Rosipal and Girolami and improved KPCA (IKPCA) method [8] proposed by Yong Xu et al.. The expectation method [7] improves the implementation efficiency of KPCA with a large number of data points, though this approach is not able to improve KPCA-based feature extraction. The IKPCA method [8] is methodologically consistent with the essence of KPCA, and it can speed up the feature extraction. However, the training samples used to expand principal component compose a fixed proportion of the total training samples. Moreover, these training samples could not be chosen adaptively. Kim et al. [9] first propose to kernelize the generalized Hebbian algorithm, which is an iterative self-organizing computation procedure for linear PCA, to estimate principal components in the feature space. Later, Gunter et al. [10] developed gain

* Corresponding author.
E-mail addresses: zhzj0218@yahoo.com.cn (Z. Zhang), zhtian@nwpu.edu.cn (Z. Tian), 11700575@qq.com (X. Duan), 372342663@qq.com (X. Fu).

0925-2312/$ - see front matter Crown Copyright © 2013 Published by Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2013.01.035

adaptation methods to improve convergence of the kernel Hebbian algorithm by incorporating the reciprocal of the eigenvalue as a part of the principal component. While these two methods can reduce the time complexity of computing kernel principal component, it is not clear how the added novel data can be incorporated to update the kernel principal component. Incremental KPCA [11–13] can reduce the time complexity of computing principal component. Extraordinarily, Chin and Suter [12,13] show how to obtain the kernel principal component by incrementally updating singular value decomposition (SVD) of the mapped data in feature space. To maintain constant update speed and memory usages, the kernel principal component representations was compressed by constructing reduced-set expansion, which is computationally expensive. Greedy KPCA[14,15] was employed to approximate the principal component by a prior filtering of the training data. However, one drawback is that the filtering could be computationally expensive by itself. Changshui Zhang et al. [16] proposed a general kernelization framework based on PCA and KPCA for feature extraction, in which the low-rank KPCA is adopted to remove noise in the feature space. However, the batch nature still exists because of using all training samples as spanning vectors. Adaptive KPCA (AKPCA) [17–19] is with rapid and accurate computation for extracting kernel principal components. However, it does not show clearly about how often to update the principal components to achieve a certain tradeoff between its computation efficiency and update ability. Yi Yang et al. [20] proposed an unsupervised feature selection method to select the most discriminative feature subset from the whole feature set in batch mode. However, it is not suitable for selecting spanning vector in KPCA related problems. An approximate linear dependence condition is proposed in ALD method [21] to select training samples with a given approximation accuracy error. However, the number of selected samples which are used to expand kernel principal components is mainly impacted by the given approximation accuracy error.

In this paper, an adaptive kernel subspace method is proposed, which is still subject to the KPCA methodology and chooses the training sample adaptively to approximately linearly expand kernel principal components in the feature space. The proposed method is derived directly from the KPCA methodology, and the feature extraction process using the proposed method is more efficient than that using the KPCA, IKPCA [8] and AKPCA [17]. The rest of this paper is organized as follows. KPCA is briefly introduced in Section 2. Then the adaptive kernel subspace method is presented on Section 3, followed by the experiment results show in Section 4. Finally, the conclusion is presented in Section 5.

2. Nonlinear extension of PCA based on a kernel function

As a nonlinear method, KPCA is nothing but the PCA in the feature space [22]. We assume that x_i, $i=1,2,\ldots,N$, $x_i \in R^n$ are training samples in input space. KPCA nonlinearly maps x_i into a higher dimensional space F by a nonlinear function $\phi: R^n \to F, x_i \to \phi(x_i)$, and subsequently performs linear PCA in F. Assuming that the mapped data is centered in the feature space, its covariance matrix is given by

$$C_\phi = \frac{1}{N}\sum_{i=1}^{N}\phi(x_i)\phi(x_i)^T$$

The map ϕ is induced by a kernel function $k(\bullet,\bullet)$ that allows us to evaluate inner products in F: $<\phi(x_i),\phi(x_j)> = k(x_i,x_j)$ $i,j=1,2,\cdots,N$. Suppose that C_ϕ has an eigendecomposition $C_\phi = 1/N(V_N \Lambda_N V_N^T)$where $V_N=[v^1,v^2,\ldots,v^N]$, $\Lambda_N = \mathrm{diag}(\lambda_1,\lambda_2,\ldots\lambda_N)$, v^w and corresponding λ_w satisfy

$$C_\phi v^w = \lambda_w v^w \tag{1}$$

Given that the mapping function ϕ is implicit, this eigendecomposition can not be performed on C_ϕ to compute the kernel principal components. KPCA circumvents the kernel principal component by a dual eigendecomposition problem for kernel Gram matrix $N\lambda_w a^w = Ka^w$, in which $a^w=[a_1^w,a_2^w,\ldots,a_N^w]^T$ is the normalized eigenvector associated with the w-th largest eigenvalues. Then the kernel principal components in the feature space take the form of

$$V_r=[v^1,v^2,\ldots,v^N]=[\phi(x_1),\phi(x_2),\ldots,\phi(x_N)][a^1,a^2,\ldots,a^N]$$

For $v^w=\Sigma_{i=1}^N a_i^w\phi(x_i)$, we have $v^w \in \mathrm{span}\{\phi(x_1),\phi(x_2),\cdots,\phi(x_N)\}$, and $\phi(x_1),\phi(x_2),\ldots,\phi(x_N)$ are named as the spanning vectors of kernel principal component v^w.

For simplicity, we have made the above assumption that the mapped data are zero-mean. However, the assumption is often invalid. Denoting the mean of the mapped data as $\mu=(1/N)\Sigma_{i=1}^N\phi(x_i)$, we still can get the kernel principal components as linear combination of centered data as

$$\tilde{v}^w=\sum_{i=1}^{N}\tilde{a}^w{}_i\tilde{\phi}(x_i)=[(\phi(x_1)-\mu),(\phi(x_2)-\mu),\cdots,(\phi(x_N)-\mu)]\tilde{a}^w$$

where $\tilde{a}^w$ is the normalized eigenvector associated with the eigendecomposition problem for centered kernel Gram matrix

$$\tilde{K}\tilde{a}^w=(K-1_NK-K1_N+1_NK1_N)\tilde{a}^w=N\lambda_w\tilde{a}^w$$

where $\tilde{K}$ is the centered kernel Gram matrix, and 1_N is a $N\times N$ matrix with all entries equal to (1).

It is easy to know that for the sample $\phi(x)$ in the feature space, the most representative m dimensional features extracted using KPCA form the following vector:

$$Y=\left[\frac{\sum_{i=1}^{N}a_i^1k(x_i,x)}{\sqrt{\lambda_1}},\frac{\sum_{i=1}^{N}a_i^2k(x_i,x)}{\sqrt{\lambda_2}},\cdots,\frac{\sum_{i=1}^{N}a_i^mk(x_i,x)}{\sqrt{\lambda_m}}\right]^T \tag{2}$$

According to the essence of the KPCA methodology, the feature extraction procedure based on (2) is theoretically able to produce the minimum reconstruction error.

3. Kernel subspace method for feature extraction

3.1. The idea of adaptive kernel subspace method

The former feature extraction using KPCA from (2) indicates that to obtain features of a sample in the feature space, we should calculate all the kernel functions between this sample and the total training samples. It further means that the implementation is inefficient when the training data set is large. The idea of kernel subspace method is that the kernel principal components for feature extraction can be expressed approximately as a linear combination of some of the training samples. These samples, which can be chosen adaptively, span a subspace of feature space F. This means that the spanning vectors of the extracted kernel principal components of kernel subspace method are a subset of that of KPCA. Assume that $v^w=\Sigma_{i=1}^M b_i^w\phi(x_i^*)$, where $M<N$ and $\phi(x_1^*),\phi(x_2^*),\ldots,\phi(x_M^*)$ are spanning vectors of v^w, we get

$$\lambda_w\left(\sum_{i=1}^{M}b_i^w<\phi(x_k^*),\phi(x_i^*)>\right)=\frac{1}{N}\sum_{i=1}^{N}<\phi(x_k^*),\phi(x_i)><\phi(x_i),\sum_{j=1}^{M}b_j^w\phi(x_j^*)>,$$

$$k=1,2,\ldots,M$$

based on the methodology of KPCA and Eq. (1).Given that $M\times N$ matrix $K_{1i,j}:=<\phi(x_i^*),\phi(x_j)>=k(x_i^*,x_j)$ and $M\times M$ matrix $K_{2i,j}:=<\phi(x_i^*),\phi(x_j^*)>=k(x_i^*,x_j^*)$, the former set of equations can be expressed as a matrix form:

$$N\lambda_w K_2 b^w = K_1K_1^Tb^w \tag{3}$$