Neurocomputing 134 (2014) 280–288

Contents lists available at ScienceDirect

Neurocomputing

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

Incremental two-dimensional kernel principal component analysis

Yonghwa Choi [a], Seiichi Ozawa [b], Minho Lee [a,*]

[a] School of Electrical Engineering, Kyungpook National University, 1370 Sankyuk-Dong, Puk-Gu, Taegu 702-701, Republic of Korea
[b] Graduate School of Engineering, Kobe University Faculty of Engineering, Kobe University, 1-1 Rokko-Dai, Nada-ku, Kobe 657-8501, Japan

ARTICLE INFO

Article history:
Received 2 May 2013
Received in revised form
29 July 2013
Accepted 30 August 2013
Available online 25 January 2014

Keywords:
Principal component analysis (PCA)
Kernel method
Incremental learning
Two-dimensional PCA

ABSTRACT

In this paper, we propose a new online non-linear feature extraction method, called the incremental two-dimensional kernel principal component analysis (I2DKPCA), not only to reduce the computational cost but also to provide good feature representation. Batch type feature extraction methods such as principal component analysis (PCA) and two-dimensional PCA (2DPCA) require more computational time and memory usage, as they collect the entire training data to extract the basis vectors. Also, these linear feature extraction methods could not effectively represent the non-linear distribution of input data. Therefore, by adopting a non-linear kernel approach with chunk concept, the KPCA and 2DKPCA can effectively address the non-linear feature representation problem by adaptively changing the feature spaces. However, this kernel approach requires more computational time for processing images with high dimensional input data. In order to solve these problems, we combined the 2DKPCA with incremental learning for (1) solving the non-linear problem and (2) reducing the memory usage with computational time. In order to evaluate the performance of I2DKPCA, several experiments have been performed using well-known face and object image databases.

1. Introduction

Feature extraction is one of the most important processing steps in pattern recognition [1], as it allows high-dimensional image data to be represented by low-dimensional feature vectors. The principal component analysis (PCA) is a well-known feature extraction and data representation technique widely used in the areas of pattern recognition, computer vision and signal processing [2]. Sirovich and Kirby first used PCA to efficiently represent human faces [3,4]. Studies related to PCA have extended this process in many ways, resulting in algorithms such as two-dimensional PCA (2DPCA), incremental PCA (IPCA), kernel PCA (KPCA), etc. [8,9,11].

Conventionally, a 2-D face image is transformed into a 1-D face image vector in the column or row direction to apply the PCA. Thus, Zhang and Zhou proposed a 2DPCA method to alleviate the large computational load in dealing with a large covariance matrix [7]. The size of the covariance matrix in the 2DPCA method is smaller than that found in conventional PCA, since 2-D images are not required to be converted to a 1-D vector. Moreover, they showed that 2DPCA has better recognition accuracy for several face databases than conventional PCA [8,9]. However, the PCA and 2DPCA could not effectively represent non-linear distribution of input data. In order to solve the non-linear problem, various approaches with regards to kernel functions have also been extensively studied as extensions to the PCA [4,5]. In KPCA, samples are mapped to a high-dimensional kernel space to convert the non-linear distribution of input data into a linear distribution of input data before conducting the PCA [5,6]. By combining 2DPCA with the kernel approach, Zhang et al. [10] proposed the 2DKPCA method based on the 2DPCA. However, the kernel method has the following two disadvantages: First, this approach is not efficient in real-time practical systems because the conventional KPCA uses the entire data set with heavy computational load as the data are provided sequentially in small chunks. Second, when a new sample is given, the conventional KPCA method needs to refer to the previously acquired entire data set to update the eigenvectors. As a result, it requires a large amount of memory usage because all of the previously acquired input data are needed to extract new feature spaces. In order to solve this problem, an incremental KPCA (IKPCA) algorithm was proposed by Ozawa et al. [11], whereby the eigenspace is updated using only the new input data without previously acquired data. But, the learning time of IKPCA increases by adding new sample data because the conventional IKPCA is easy to increase the feature dimension. By combining the advantages of both the 2D approach with kernel functions and incremental learning, we develop a new incremental two-dimensional kernel principal component analysis (I2DKPCA) method for 2-D input data, which requires less computation time than the conventional IKPCA and a smaller memory usage than the conventional 2DKPCA.

* Corresponding author. Tel.: +82 53 950 6436; fax: +82 53 950 5505.
E-mail addresses: yhchoi@ee.knu.ac.kr (Y. Choi), ozawasei@kobe-u.ac.jp (S. Ozawa), mholee@gmail.com (M. Lee).

0925-2312/$ - see front matter © 2014 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.neucom.2013.08.045

The rest of this paper is organized as follows. In Section 2, we describe the proposed incremental feature extraction algorithm called I2DKPCA. Then, we present the performance evaluation of the proposed I2DKPCA on a face recognition system in Section 3. Finally, we draw our conclusions in Section 4.

2. Incremental two-dimensional kernel PCA (I2DKPCA)

First, we briefly introduce the batch type linear feature extraction method such as PCA and 2DPCA. Then, we discuss kernel functions to enhance the non-linear feature representation ability of KPCA and 2DKPCA. Finally, we introduce the incremental feature extraction methods such as IKPCA and the proposed I2DKPCA.

2.1. Family of batch type PCA methods

2.1.1. Linear feature extraction methods

The conventional PCA is a simple way to effectively reduce the feature dimension [2]. Suppose there are N training data samples vector $x^s \quad (s=1,2,\dots,N)$ converted from image data denoted by $m\times n$ matrix where m and n are the number of row and column in an image. The covariance matrix of PCA is given as follows:

$$Q=\frac{1}{N}\sum_{s=1}^{N}(x^s-c)(x^s-c)^T \tag{1}$$

where c is the mean vector of entire training data which is calculated by $c=(1/N)\sum_{s=1}^{N}x^s$. Eigenvalue problem of conventional PCA using covariance matrix Q is given by

$$Qz=z\lambda \tag{2}$$

By solving this eigenvalue problem and projecting the data onto eigenvectors z, data can be reduced into small feature dimension. To extend PCA, 2DPCA algorithm considers two-dimensional data [8]. Suppose there are N training data samples $X^s \quad (s=1,2,\dots,N)$ each of which is denoted by an m by n matrix, X^s consists of n column vectors $x_j^s \quad (j=1,\dots,n)$. of size m by 1

$$X^s=[x_1^s,\dots,x_n^s] \tag{3}$$

The covariance matrix of 2DPCA for the column-direction is given as follows:

$$\begin{aligned}Q&=\frac{1}{N}\sum_{j=1}^{n}\sum_{s=1}^{N}(x_j^s-c_j)(x_j^s-c_j)^T\\&=\frac{1}{N}\sum_{s=1}^{N}(X^s-c)(X^s-c)^T\end{aligned} \tag{4}$$

where N is the number of training data samples and $c=(1/N)\sum_{s=1}^{N}X^s$. It is the same notation as PCA. The eigenvalue problem of conventional 2DPCA is then obtained in the same way as PCA.

2.1.2. Non-linear feature extraction methods such as KPCA and 2DKPCA

The KPCA algorithm uses kernel to map high-dimensional space [5]. Suppose there are N training data samples $x^s \quad (s=1,2,\dots,N)$, where x^s is an $(m\times n)\times 1$ dimensional vector. We express the kernel function for mapping the vector data as $\phi(\bullet)$ and $\Phi(\bullet)$ for matrix of data. The training data in a high-dimensional feature space is denoted as

$$\Phi(X)=[\phi(x^1),\dots,\phi(x^N)] \tag{5}$$

where $\phi(x^s)$ is a mapping function, x^s is the sth data and Φ is the matrix of mapped data. The covariance matrix of KPCA is given as follows:

$$Q=\frac{1}{N}\sum_{s=1}^{N}(\phi(x^s)-c)(\phi(x^s)-c)^T \tag{6}$$

However, computing (6) will require a very high computational load. So we simply calculate kernel using $k(x,y)=\langle\phi(x),\phi(y)\rangle$ where $\langle\cdot,\cdot\rangle$ is indeed an inner product [17]. To avoid the explicit mapping, the trick is to use learning algorithms that only require dot products between the vectors of training data, and choose the mapping such that these high-dimensional dot products can be computed within the original space by means of a kernel function. Calculating the eigenproblem of KPCA is similar to the conventional PCA algorithm.

In the conventional KPCA method, a kernel-induced mapping function maps the data vector from the original input space to a higher or even infinite dimensional feature space. Suppose there are N training two-dimensional data samples $X^s \quad (s=1,2,\dots N)$, each of which is denoted as an m by n matrix. The image data in a high-dimensional feature space is denoted as

$$\Phi(X^s)=[\phi(x_1^s),\dots,\phi(x_n^s)] \tag{7}$$

where $j(x_j^s)$ is a mapping function and x_j^s is the jth column vector of the sth image x_j^s. The column wise covariance matrix of 2DKPCA is given as follows:

$$Q=\frac{1}{N}\sum_{s=1}^{N}(\Phi(X^s)-c)(\Phi(X^s)-c)^T \tag{8}$$

where N is the number of 2D-samples and $c=(1/N)\sum_{s=1}^{N}\Phi(X^s)$. Eigenproblem of conventional PCA is calculated as

$$Qz=z\lambda \tag{9}$$

where z is the matrix of eigenvectors and λ is the vector of eigenvalues. The eigenspace is obtained by solving the eigenvalue problem as done in the conventional PCA using the covariance matrix Q. Practically, however, this calculation is hardly ever carried out because the number of dimensions in the feature space is generally very high and could possibly be infinite. To avoid the explicit calculation in the feature space, the so-called "kernel trick" is applied.

Without loss of generality, we can assume that a set of r linearly independent samples $\{\phi(D_1),\dots,\phi(D_r)\} \quad (r\le(N\times n))$ in (10) spans the space for $N\times n$ training samples obtained from n column vectors of each image

$$z_i=[\phi(D_1),\dots,\phi(D_r)]\begin{bmatrix}\alpha_{i1}\\ \vdots\\ \alpha_{r1}\end{bmatrix}=\Phi_r\alpha_i \tag{10}$$

where $\alpha_i=[\alpha_{i1},\dots,\alpha_{r1}]^T \quad (i=1,\dots,r)$ is a coefficient vector and independent sample D_i, denoted by $[D_1,\dots,D_r]$, is obtained by selecting independent vectors from all the column vectors such as

$$[\phi(x_1^1),\dots,\phi(x_n^1),\dots,\phi(x_n^N),\dots,\phi(x_n^N)].$$

Using Algorithm l, we can find independent sample D from all column vectors.

Algorithm 1. Calculating independent columns

```
$D \leftarrow [\varnothing]$
$Y \leftarrow [\phi(x_1^1),\dots,\phi(x_n^1),\dots,\phi(x_n^N),\dots,\phi(x_n^N)]$
$r \leftarrow 0$
while $Y$ is not empty
  randomly select one column $Y_j$
  $H \leftarrow \Phi([DY_i])$
  if $rank(H)$ is independent
      $D \leftarrow [DY_i]$
      $r \leftarrow r+1$
  end if
  $Y_j \leftarrow [\phi]$
end while
```