Available online at www.sciencedirect.com
ScienceDirect
Pattern Recognition 40 (2007) 339–342
PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY
www.elsevier.com/locate/patcog

Rapid and brief communication

Two-dimensional locality preserving projections (2DLPP) with its application to palmprint recognition

Dewen Hu, Guiyu Feng*, Zongtan Zhou

Department of Automatic Control, College of Mechatronics and Automation, National University of Defense Technology, Changsha, Hunan, 410073, PR China

Received 26 December 2005; received in revised form 20 June 2006; accepted 22 June 2006

Abstract

This paper proposes a novel algorithm for image feature extraction, namely, the two-dimensional locality preserving projections (2DLPP), which directly extracts the proper features from image matrices based on locality preserving criterion. Experimental results on the PolyU palmprint database show the effectiveness of the proposed algorithm.

Keywords: Locality preserving projections (LPP); Principal component analysis (PCA); Linear discriminant analysis (LDA); Palmprint recognition

1. Introduction

Locality preserving projections (LPP) is a recently proposed method in image recognition for feature extraction and dimension reduction. The objective of LPP is to preserve the local structure of the image space by explicitly considering the manifold structure, which is in fact to solve a generalized eigenvalue problem [1]

$$XLX^{\mathrm{T}}a = \lambda XDX^{\mathrm{T}}a. \tag{1}$$

A difficulty in using the LPP method for image recognition is the high-dimensional nature of the image space, in such a space, the XDX^{T} matrix is always singular, which makes the direct implementation of the LPP algorithm impossible.

One possible solution to attack this problem is to utilize the principal component analysis (PCA) as a preprocessing step to reduce the dimensionality of the vector space, which is known as Laplacianface algorithm and has been applied successfully to face representation and recognition [1]. However, in the existing Laplacianface (PCA + LPP) algorithm, several disadvantages should be pointed out:

(1) The 2D image matrices must be previously transformed into 1D image vectors. The resulting image vectors usually lead to a high-dimensional image vector space, where it is difficult to calculate the bases to represent the original images, which is also called the "curse of dimensionality" problem. This problem is more apparent in small-sample-size cases such as image recognition.

(2) Such a matrix-to-vector transform may cause the loss of some structural information residing in original 2D images.

(3) In the PCA step of the Laplacianface algorithm, how to determine the numbers of principal components is a hard problem.

(4) In the Laplacianface algorithm, after all the image vectors are projected into the subspace spanned by the principal components, the LPP algorithm is then performed. However, since the objective of the PCA and that of LPP are essentially different, the preprocessing procedure to reduce the dimensionality using the PCA could result in the loss of some important information for the LPP algorithm that

* Corresponding author. Tel.: +86 731 4574977; fax: +86 731 4574992.
E-mail address: smartfgy@yahoo.com.cn (G. Feng).

0031-3203/$30.00
doi:10.1016/j.patcog.2006.06.022

follows the PCA. To illustrate this problem, a brief proof is given:

In [2], a locality preserving function f was defined as follows:

$$f(a) = \frac{a^{\mathrm{T}} X L X^{\mathrm{T}} a}{a^{\mathrm{T}} X D X^{\mathrm{T}} a}. \tag{2}$$

The locality preserving function $f(a)$ evaluates the locality preserving power of the projective map a. Actually, in the LPP algorithm $a^{\mathrm{T}} X D X^{\mathrm{T}} a = 1$, then Eq. (2) can be reduced to

$$f(a) = a^{\mathrm{T}} X L X^{\mathrm{T}} a. \tag{3}$$

In fact, XLX^{T} is also singular [1], which implies that the null space of XLX^{T} contains valuable discriminatory information; however, the PCA step may discard such projection directions which satisfy $f(a) = 0$. Now we can see that some important information for the following LPP algorithm may have been lost in the PCA step.

Inspired by Yang et al. [3], in this paper, an alternative way is proposed to handle the above problems by directly projecting the image matrix under a specific projection criterion, rather than using the stretched image vector. Our algorithm proposed here is a straightforward manner based on locality preserving criterion and the image matrix projection. Experimental results on the PolyU palmprint database show that the 2DLPP algorithm outperforms the conventional PCA, PCA+LDA and PCA+LPP algorithms in terms of the recognition performance rate. Our work will fit into the scene for a better picture about LPP-based methods for image recognition.

2. Two-dimensional locality preserving projections (2DLPP)

2.1. The algorithm of 2DLPP

Like that of the vector-based LPP [1], the objective function of 2DLPP is defined as

$$\min \sum_{i,j} \|Y_i - Y_j\|^2 S_{ij}, \tag{4}$$

where Y_i is the n-dimensional representation of $m \times n$ matrix X_i, the matrix S is a similarity matrix, and $\|\bullet\|$ means the L_2 norm. A possible way of defining S is as follows: $S_{ij} = \exp(-\|X_i - X_j\|^2/t)$, if X_i is among k nearest neighbors of X_j or X_j is among k nearest neighbors of X_i, otherwise, $S_{ij} = 0$.

Here, k defines the local neighborhood. The objective function with this choice of symmetric weights S_{ij} incurs a heavy penalty if neighboring points X_i and X_j are mapped far apart, i.e., if $\|Y_i - Y_j\|^2$ is large. Therefore, minimizing Eq. (4) is an attempt to ensure that, if X_i and X_j are "close", then Y_i and Y_j are close as well. Following some matrix analysis steps, we can get

$$\begin{aligned}
&\frac{1}{2} \min \sum_{i,j} \|Y_i - Y_j\|^2 S_{ij} \\
&\quad = \frac{1}{2} \sum_{ij} \|a^{\mathrm{T}} X_i - a^{\mathrm{T}} X_j\|^2 S_{ij} \\
&\quad = \frac{1}{2} \sum_{ij} (a^{\mathrm{T}} X_i - a^{\mathrm{T}} X_j)(a^{\mathrm{T}} X_i - a^{\mathrm{T}} X_j)^{\mathrm{T}} S_{ij} \\
&\quad = \frac{1}{2} \sum_{ij} a^{\mathrm{T}} (X_i - X_j)(X_i^{\mathrm{T}} - X_j^{\mathrm{T}}) a S_{ij} \\
&\quad = \frac{1}{2} \sum_{ij} a^{\mathrm{T}} (X_i X_i^{\mathrm{T}} + X_j X_j^{\mathrm{T}} - X_j X_i^{\mathrm{T}} - X_i X_j^{\mathrm{T}}) a S_{ij} \\
&\quad = \sum_{ij} a^{\mathrm{T}} X_i S_{ij} X_i^{\mathrm{T}} a - \sum_{ij} a^{\mathrm{T}} X_i S_{ij} X_j^{\mathrm{T}} a \\
&\quad = a^{\mathrm{T}} X D X^{\mathrm{T}} a - a^{\mathrm{T}} X S X^{\mathrm{T}} a \\
&\quad = a^{\mathrm{T}} X (D - S) X^{\mathrm{T}} a \\
&\quad = a^{\mathrm{T}} \left(\sum_{i,j=1}^{k} X_i X_j^{\mathrm{T}} L_{ij} \right) a \\
&\quad = a^{\mathrm{T}} X L X^{\mathrm{T}} a,
\end{aligned} \tag{5}$$

where $X = [X_1, X_2, \ldots, X_k]$, and D is a diagonal matrix; its entries are column or row sums of S. $L = D - S$ is the Laplacian matrix. Obviously, a trivial solution exists $a = 0$. Therefore, a constraint $a^{\mathrm{T}} X D X^{\mathrm{T}} a = 1$ is added.

Then the minimization problem becomes

$$\arg\min_a \quad a^{\mathrm{T}} X L X^{\mathrm{T}} a \quad \text{S.t. } a^{\mathrm{T}} X D X^{\mathrm{T}} a = 1. \tag{6}$$

The Lagrange multiplier can be employed to transform the above objective function to include the constraint

$$g(a, \lambda) = a^{\mathrm{T}} X L X^{\mathrm{T}} a + \lambda (1 - a^{\mathrm{T}} X D X^{\mathrm{T}} a). \tag{7}$$

The solution of Eq. (6) can be found by letting $\partial g / \partial a = 0$.

Thus we can get

$$X L X^{\mathrm{T}} a = \lambda X D X^{\mathrm{T}} a. \tag{8}$$

Now the transformation vector a that minimizes the objective function is given by the minimum eigenvalue solution to this generalized eigenvalue problem.

It should be pointed out that Eqs. (8) and (1) may look like the same to each other in that the calculating of S, D and L is in the same way, however, they are quite different in essence: in Eq. (8) X is matrix-based, while in Eq. (1) X is vector-based.

2.2. Feature extraction

Let the column vectors $a_0, a_1, \ldots, a_{l-1}$ be the solutions of Eq. (8), ordered according to their eigenvalues $\lambda_0 < \lambda_1 < \cdots < \lambda_{l-1}$. Thus the ith embedding is as follows:

$$Y_i = a_i^{\mathrm{T}} X_i, \tag{9}$$