Available online at www.sciencedirect.com

SCIENCE DIRECT®

Pattern Recognition 38 (2005) 1784 – 1787

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

Rapid and brief communication

Kernel ICA: An alternative formulation and its application to face recognition

Jian Yang[a, b, *], Xiumei Gao[b], David Zhang[a], Jing-yu Yang[b]

[a]Department of Computing, Hong Kong Polytechnic University, Kowloon, Hong Kong
[b]Department of Computer Science, Nanjing University of Science and Technology, Nanjing 210094, PR China

Received 30 December 2004; accepted 28 January 2005

Abstract

This paper formulates independent component analysis (ICA) in the kernel-inducing feature space and develops a two-phase kernel ICA algorithm: whitened kernel principal component analysis (KPCA) plus ICA. KPCA spheres data and makes the data structure become as linearly separable as possible by virtue of an implicit nonlinear mapping determined by kernel. ICA seeks the projection directions in the KPCA whitened space, making the distribution of the projected data as non-gaussian as possible. The experiment using a subset of FERET database indicates that the proposed kernel ICA method significantly outperform ICA, PCA and KPCA in terms of the total recognition rate.

Keywords: Kernel-based methods; Independent component analysis (ICA); Principal component analysis (PCA); Feature extraction; Face recognition

1. Introduction

Over the last few years, independent component analysis (ICA) has aroused wide research interests and become a popular tool for blind source separation and feature extraction. From the feature extraction point of view, ICA has a close relationship with projection pursuit since both of them aim to find the directions such that the projections of the data into those directions have maximally “non-gaussian” distributions. These projections are interesting and considered more useful for classification [1]. Bartlett [2] and Liu [3] applied ICA to face representation and recognition and found that it outperforms PCA when cosine distance was used as the similarity measure.

ICA, however, fails to separate the nonlinearly mixed source due to its intrinsic linearity. Likewise, for feature extraction, ICA-based linear projection is incompetent to represent the data with nonlinear structure. To address this problem, our idea is to nonlinearly map the data into a feature space, in which the data has a linear structure (as linearly separable as possible). Then we perform ICA in feature space and make the distribution of data as non-gaussian as possible. We will use “kernel tricks” to solve the computation of independent projection directions in high-dimensional feature space and ultimately convert the problem of performing ICA in feature space into a problem of implementing ICA in the kernel principal component analysis (KPCA) transformed space.

It should be mentioned that kernel ICA formulation in this paper is different from that in [5]. In [5], the kernel trick is used for computation and optimization of a canonical

[*] Corresponding author. Department of Computing, Hong Kong Polytechnic University, Kowloon, Hong Kong. Tel.: +852 2766 7312; fax: +852 2774 0842.

E-mail addresses: csjyang@comp.polyu.edu.hk (J. Yang), gao_xiumei@sina.com (X. Gao), csdzhang@comp.polyu.edu.hk (D. Zhang), yangjy@mail.njust.edu.cn (Jing-yu Yang).

0031-3203/$30.00 © 2005 Pattern Recognition Society. Published by Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2005.01.023

correlation (Φ-correlation) based contrast function, while in this paper, "kernel" is introduced to realize an implicit nonlinear mapping which makes the data linearly structured in feature space.

2. Kernel ICA

2.1. ICA model in feature space

Given a random vector $\mathbf{x}$, which is possibly nonlinearly mixed, we map it into its image in the feature space H by the following nonlinear mapping:

$$\Phi : \mathbb{R}^n \to H, \quad x \mapsto \Phi(x). \tag{1}$$

As a result, a pattern in the original observation space (input space) $\mathbb{R}^n$ is mapped into a potentially much higher dimensional feature vector in the *feature space* H.

Assume that after the nonlinear mapping, the data have a linearly separable structure in feature space H. Our task is to find a linear operator $\mathbf{W}^{\Phi}$ in H to recover the independent components from $\Phi(x)$ by the following linear transformation:

$$\mathbf{s} = \mathbf{W}^{\Phi}\Phi(\mathbf{x}). \tag{2}$$

Note that if the feature space is finite-dimensional, the operator $\mathbf{W}^{\Phi}$ is a matrix. Now, the problem is how to determine the linear transformation $\mathbf{s} = \mathbf{W}^{\Phi}\Phi(\mathbf{x})$ in the high-dimensional (possibly infinite-dimensional) feature space.

2.2. Algorithm

Let us recall the implementation of ICA in observation space. Before applying an ICA algorithm on the data, it is usually very useful to do some preprocessing work (e.g. sphering or whitening data). The preprocessing can make the problem of ICA estimation simpler and better conditioned [1]. Generally, a whitened PCA is used to sphere the data and make the transformed data $\mathbf{y}$ satisfy

$$E\{\mathbf{y}\mathbf{y}^{\mathrm{T}}\} = \mathbf{I},$$
where $\mathbf{I}$ is an identity matrix. (3)

Similarly, we can perform PCA in feature space H for data whitening. Note that *performing PCA in feature space can be equivalently implemented in input space* (*observation space*) *by virtue of kernels, i.e., performing KPCA based on the observation data*. Based on this idea, we will develop a concise kernel ICA algorithm to implement ICA in feature space.

2.2.1. Sphering of data using KPCA

Given an observation sequence $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M$ in $\mathbb{R}^n$, let us assume their images are centered in feature space, i.e., $\sum_{j=1}^{M}\Phi(\mathbf{x}_j) = \mathbf{0}$ (This is just for simplicity, we will deal with the centering problem later). The *covariance operator* on the *feature space* H can be constructed by

$$\mathbf{S}_t^{\Phi} = \frac{1}{M}\sum_{j=1}^{M}\Phi(\mathbf{x}_j)\Phi(\mathbf{x}_j)^{\mathrm{T}}. \tag{4}$$

In a finite-dimensional feature space, this operator is generally called covariance matrix. It is not hard to show $\mathbf{S}_t^{\Phi}$ is a positive operator, so non-zero eigenvalues of $\mathbf{S}_t^{\Phi}$ are all positive. It is these positive eigenvalues that are of interest to us. Schölkopf et al. [4] has suggested a way to find them. Let $\mathbf{Q} = [\Phi(\mathbf{x}_1), \ldots, \Phi(\mathbf{x}_M)]$, then $\mathbf{S}_t^{\Phi}$ can be expressed by $\mathbf{S}_t^{\Phi} = (1/M)\,\mathbf{Q}\mathbf{Q}^{\mathrm{T}}$. Let us form the Gram matrix $\mathbf{R} = \mathbf{Q}^{\mathrm{T}}\mathbf{Q}$, which is an $M \times M$ matrix and its elements can be determined by virtue of the given kernel function $\mathrm{k}(\mathbf{x}, \mathbf{y})$, i.e.,

$$\mathbf{R}_{ij} = \Phi(\mathbf{x}_i)^{\mathrm{T}}\Phi(\mathbf{x}_j) = (\Phi(\mathbf{x}_i)\cdot\Phi(\mathbf{x}_j)) = \mathrm{k}(\mathbf{x}_i, \mathbf{y}_j). \tag{5}$$

Calculate the orthonormal eigenvectors $\boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2, \ldots, \boldsymbol{\gamma}_m$ of $\mathbf{R}$ corresponding to m largest positive eigenvalues $\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_m$. Then, the m largest positive eigenvalues of $\mathbf{S}_t^{\Phi}$ are $\lambda_1/M, \lambda_2/M, \ldots, \lambda_m/M$, and the associated orthonormal eigenvectors $\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \ldots, \boldsymbol{\beta}_m$ can be expressed by

$$\boldsymbol{\beta}_j = \frac{1}{\sqrt{\lambda_j}}\mathbf{Q}\boldsymbol{\gamma}_j, \quad j = 1, \ldots, m. \tag{6}$$

Denote $\mathbf{V} = (\boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2, \ldots, \boldsymbol{\gamma}_m)$, $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_m)$, and $\mathbf{B} = (\boldsymbol{\beta}_1, \boldsymbol{\beta}_2, \ldots, \boldsymbol{\beta}_m) = \mathbf{Q}\mathbf{V}\Lambda^{-1/2}$, then

$$\mathbf{B}^{\mathrm{T}}\mathbf{S}_t^{\Phi}\mathbf{B} = \mathrm{diag}\left(\frac{\lambda_1}{M}, \frac{\lambda_2}{M}, \ldots, \frac{\lambda_m}{M}\right) = \frac{1}{M}\Lambda. \tag{7}$$

Going one step further, let $\mathbf{P} = \mathbf{B}(\frac{1}{M}\Lambda)^{-1/2} = \sqrt{M}\mathbf{Q}\mathbf{V}\Lambda^{-1}$, then

$$\mathbf{P}^{\mathrm{T}}\mathbf{S}_t^{\Phi}\mathbf{P} = \mathbf{I}. \tag{8}$$

Thus, we obtain the whitening matrix $\mathbf{P}$. The mapped data in feature space can be whitened by the following transformation:

$$\mathbf{y} = \mathbf{P}^{\mathrm{T}}\Phi(\mathbf{x}). \tag{9}$$

Specifically,

$$\begin{aligned}\mathbf{y} &= \sqrt{M}\Lambda^{-1}\mathbf{V}^{\mathrm{T}}\mathbf{Q}^{\mathrm{T}}\Phi(\mathbf{x}) \\ &= \sqrt{M}\Lambda^{-1}\mathbf{V}^{\mathrm{T}}[\mathrm{k}(\mathbf{x}_1, \mathbf{x}), \mathrm{k}(\mathbf{x}_2, \mathbf{x}), \ldots, \mathrm{k}(\mathbf{x}_M, \mathbf{x})]^{\mathrm{T}} \\ &= \sqrt{M}\Lambda^{-1}\mathbf{V}^{\mathrm{T}}\mathbf{R}_x.\end{aligned} \tag{10}$$

Now, let us go back to the problem of centering data. Data centering in input space is easy, but it is difficult to do so in feature space because we cannot explicitly compute the mean of the mapped data in H. Fortunately, a slight modification of the above process can achieve this.