Nonlinear Component Analysis as a Kernel Eigenvalue Problem

Bernhard Schölkopf Alexander Smola Klaus-Robert Müller

Technical Report No. 44, Max-Planck-Institut für biologische Kybernetik, 1996

Sungsu Lim

March 8, 2011

Outline

Introduction

PCA

- PCA (Principal Component Analysis) is a powerful technique for extracting structure from possibly high-dimensional data sets.
- It is often the case that only small number of principle component is sufficient to account for most of the structure.

Kernel trick

- We generalizes PCA to the case where we are not interested in principal components in “input space”, but rather in principal components of “features”, which are nonlinearly related to the input variables.

PCA in Feature Space

Review of PCA

- (THM) A matix is orthogonally diagonallyzable if and only if it is symmetric.
- (THM) A symmetric matrix is diagonalized by a matrix of its orthonormal eigenvectors.
- The data set is $X = [\mathbf{x}_1 \dots \mathbf{x}_M]^T$. Since XX^T is symmetric, it provides: $XX^T = PDP^T$ where D is a diagonal matrix and P is an orthonormal matrix of eigenvectors of XX^T arranged as columns.
- The principal components of X are the eigenvectors of XX^T, and each diagonal value of D is the variance of X along the corresponding principal component.

PCA in Feature Space

Review of PCA

- Given a set of M centered observations $\mathbf{x}_k$, $k = 1, \ldots, M$, $\mathbf{x}_x \in \mathbb{R}^N$, $\sum_{k=1}^{M} \mathbf{x}_k = 0$, the covariance matrix

$$C = \frac{1}{M} \sum_{j=1}^{M} \mathbf{x}_j \mathbf{x}_j^T.$$

- PCA diagonalizes C; we solve $\lambda \mathbf{v} = C\mathbf{v}$ for $\lambda \geq 0$, $\mathbf{v} \neq 0$.
- Since the variance of any real-valued random variable is nonnegative, and the symmetry of the covariance matrix, only a p.s.d. matrix can be a covariance matrix.
- (THM) A is p.s.d. if and only if all eigenvalues of A are nonnegative.

PCA in Feature Space

Review of PCA

- (THM) The eigenvectors lie in the span of $\mathbf{x}_1, \ldots, \mathbf{x}_M$.
- (Proof)

$$C\mathbf{v} = \lambda\mathbf{v} \Rightarrow \mathbf{v} = \frac{1}{M\lambda}\sum_{j=1}^{M}(\mathbf{x}_j \cdot \mathbf{v})\mathbf{x}_j.$$

  But, $(\mathbf{x}_j \cdot \mathbf{v})$ is just scalar, so all solutions $\mathbf{v}$ with $\lambda \neq 0$ lies in $span\{\mathbf{x}_1, \ldots, \mathbf{x}_M\}$.
- Thus, there exist $\alpha_i's$ such that $\mathbf{v} = \sum_{i=1}^{M} \alpha_i \mathbf{x}_i$, and it sufficient to solve $\lambda(\mathbf{x}_k \cdot \mathbf{v}) = (\mathbf{x}_k \cdot C\mathbf{v})$ for all $k = 1, \ldots, M$, to solve the eigenvalue problem.

PCA in Feature Space

PCA in feature space

- Let $\Phi : \mathbb{R}^N \to F$ be a (nonlinear) map, where F is another dot product space. We refer to F as the feature space.
- Assuming $\sum_{k=1}^{M} \Phi(\mathbf{x}_k) = 0$, the covariance matrix is

$$\bar{C} = \frac{1}{M} \sum_{j=1}^{M} \Phi(\mathbf{x}_j)\Phi(\mathbf{x}_j)^T.$$

- We have to find $\lambda \geq 0$ and $\mathbf{V} \in F \backslash \{0\}$ satisfying $\lambda \mathbf{V} = \bar{C}\mathbf{V}$.
- It suffices to solve $\lambda(\Phi(\mathbf{x}_k) \cdot \mathbf{V}) = (\Phi(\mathbf{x}_k) \cdot \bar{C}\mathbf{V})$ for all k.

PCA in Feature Space

PCA in feature space

- $\lambda(\Phi(\mathbf{x}_k) \cdot \mathbf{V}) = (\Phi(\mathbf{x}_k) \cdot \bar{C}\mathbf{V})$ for all k.
  (LHS)= $\lambda \sum_{i=1}^{M} \alpha_i (\Phi(\mathbf{x}_k) \cdot \Phi(\mathbf{x}_i))$
  (RHS) = $\frac{1}{M} \sum_{i=1}^{M} \sum_{j=1}^{M} \alpha_i (\Phi(\mathbf{x}_k) \cdot \Phi(\mathbf{x}_j))(\Phi(\mathbf{x}_j) \cdot \Phi(\mathbf{x}_i))$
- Defining $K := \left(\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)\right)_{i,j}$,
  $M\lambda K\alpha = K^2\alpha$, where $\alpha = [\alpha_1 \dots \alpha_M]^T$
  $\Leftrightarrow M\lambda\alpha = K\alpha$ (since K is symmetric)
- Since $K = [\Phi(\mathbf{x}_1) \dots \Phi(\mathbf{x}_M)]^T \cdot [\Phi(\mathbf{x}_1) \dots \Phi(\mathbf{x}_M)]$,
  $(X \cdot KX) \geq 0$ for all $X \in F$, so K is positive semidefinite.
  Thus, all eigenvalues of K are nonnegative, and are exactly give the solutions $M\lambda$.
- Therefore, we only need to diagonalize K.

PCA in Feature Space

PCA in feature space

- Let $\lambda_1 \leq \ldots \leq \lambda_M$ be the eigenvalues, and $\alpha^1, \ldots, \alpha^M$ the corresponding eigenvectors.
- Let λ_p be the first nonzero eigenvalue. We normalize $\alpha^p, \ldots, \alpha^M$ so that $V^k \cdot V^k = 1$ for all $k = p, \ldots, M$.
- $1 = V^k \cdot V^k = \sum_{i,j=1}^{M} \alpha_i^k \alpha_j^k K_{ij} = \lambda_k(\alpha^k \cdot \alpha^k)$.
- Let $\mathbf{x}$ be a test point, with an image $\Phi(\mathbf{x})$ in F, then $(V^k \cdot \Phi(\mathbf{x})) = \sum_{i=1}^{M} \alpha_i^k (\Phi(\mathbf{x_i}) \cdot \Phi(\mathbf{x}))$ may be called its nonlinear principal components corresponding to Φ.

PCA in Feature Space

Summary

The necessary steps to compute the principal components.

1. Compute the dot product matrix K.
   K is called the kernel matrix, whose ij-th element is the inner-product kernel $K(\mathbf{x}_i, \mathbf{x}_j)$.
2. Compute its eigenvectors and normalize them in F.
3. Compute projections of a test point onto the eigenvectors.

Computing Dot Products in Feature Space

Kernel representation

- In order to compute dot products of the form $\Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})$, we use kernel representations of the form $k(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})$.
- The choice of k implicitly determines the mapping Φ and the feature space F.
- If F is high-dimensional, we would like to be able to find a closed form expression for k which can be efficiently computed.

Computing Dot Products in Feature Space

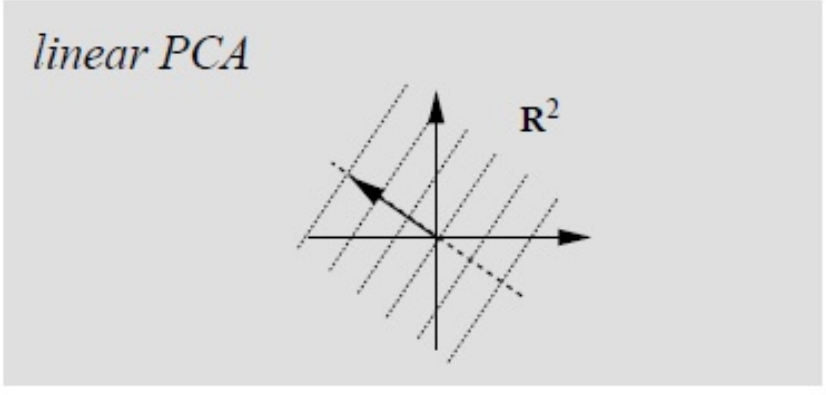
Example (Bad choice of k)

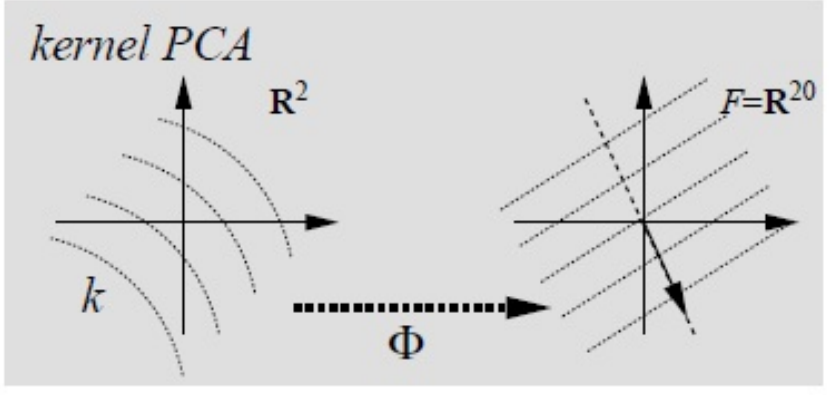
- $(\mathbf{x} \cdot \mathbf{y})^d = (C_d(\mathbf{x}), C_d(\mathbf{y}))$ where C_d maps $\mathbf{x}$ to the vector $C_d(\mathbf{x})$ whose entries are all possible n-th degre ordered products of the entries of $\mathbf{x}$.
- If $\mathbf{x} = (x_1, x_2)$, then $C_2(\mathbf{x}) = (x_1^2, x_2^2, x_1 x_2, x_2 x_1)$.
  For N dimensional data, there exist $\frac{(N+d-1)!}{d!(N-1)!}$ different monomials. For example, 16×16 pixel input images with a monomial degree $d = 5$ yields a dimension of almost 10^{10}.

Kernel PCA

The algorithm

1. Compute $K_{ij} = (k(\mathbf{x}_i, \mathbf{x}_j))_{ij}$.
2. Solve $M\lambda\alpha = K\alpha$ by diagonalizing K, and normalizing the eigenvalue expansion coefficients α^n by solving $1 = \lambda_n(\alpha^n \cdot \alpha^n)$.
3. Extract the principal components (corresponding to the kernel k) of a test point $\mathbf{x}$: compute $(kPC)_n(\mathbf{x}) = (V^n \cdot \Phi(\mathbf{x})) = \sum_{i=1}^{M} \alpha_i^n k(\mathbf{x}_i, \mathbf{x})$.





Kernel PCA

Dimensional reduction

- Kernel PCA allows the extraction of a number of principal components which can exceed the input dimensionality. (This is not necessary a dimensional reduction)

Computational complexity

- Kernel functions may easy to compute. However, extracting principal components does take more work, because we have to evaluate the kernel functions M times for each extracted principal components, instead of just one dot product as in linear PCA.
- It has a disadvantage. e.g. extract principal components as a preprocessing step for classification.

Kernel PCA

Computational complexity

- We can speed up the extraction; approximate each eigenvector $V = \sum_{i=1}^{\ell} \alpha_i \Phi(\mathbf{x}_i)$ by $\tilde{V} = \sum_{j=1}^{m} \beta_j \Phi(\mathbf{z}_j)$, where $||V - \tilde{V}||^2$ is minimized, for some chosen $m < \ell$.

SVM

- If we replaces $\mathbf{x}_i \cdot \mathbf{x}_j$ by $K(\mathbf{x}_i, \mathbf{x}_j)$, then in test phase a linear SVM is used by computing sign of

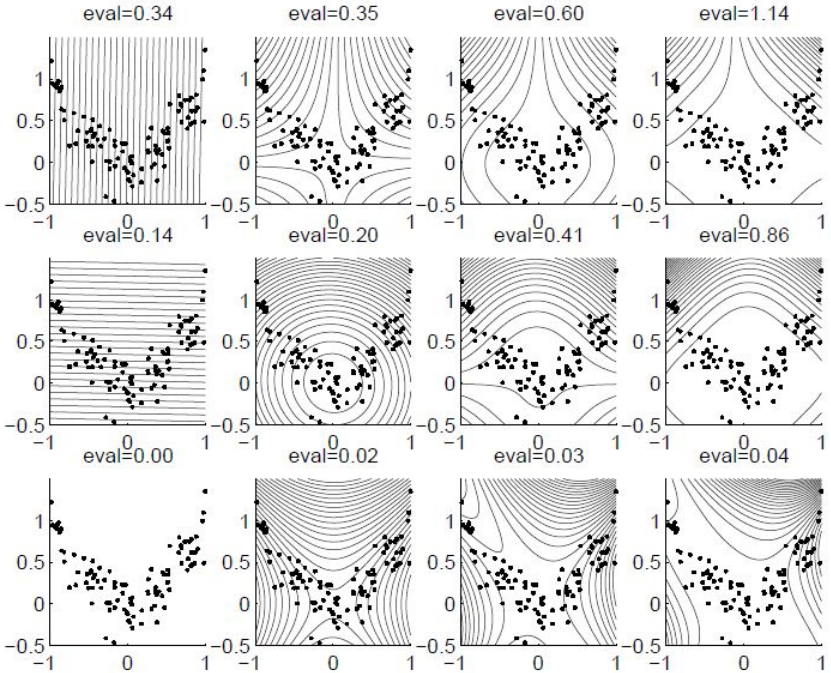
$$f(\mathbf{x}) = \sum_{i=1}^{N_s} \alpha_i y_i \Phi(\mathbf{s}_i) \cdot \Phi(\mathbf{x}) + b = \sum_{i=1}^{N_s} \alpha_i y_i K(\mathbf{s}_i, \mathbf{x}) + b$$

where $\mathbf{s}_i$ are the support vectors. We can avoid computing $\Phi(\mathbf{x})$ and use $K(\mathbf{s}_i, \mathbf{x})$ instead. It can be done fast.

Experiments

Two-dimensional toy examples

- $x_i \in [-1, 1]$, $y_i = x_i^2 + \xi$, where $\xi \sim N(0, 0.2)$.
- From left to right, we use $k(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{x})^d$, $d = 1, \ldots, 4$.
- From top to bottom, the first 3 eigenvectors are shown.



Nonlinear variants of other algorithms

Kernel-k-means clustering

- Consider k-means clustering. Let $M_{iv} = 1$ if $\mathbf{x}_i$ belongs to cluster v, 0 otherwise.
- We are trying to find k centers $\mathbf{m}_v$. Clearly, the centers should lie in $span\{\Phi(\mathbf{x}_1), \ldots, \Phi(\mathbf{x}_M)\}$. If not, we could project $\mathbf{m}_v$ to the above span.
- We have $\mathbf{m}_v = \sum_{j=1}^{M} \gamma_{vj}\Phi(\mathbf{x}_j)$. Initially, we can set $\gamma_{ij} = \delta_{ij}$. The squared distance between $\mathbf{m}_v$ and a mapped pattern $\Phi(\mathbf{x})$ can be expressed as $||\Phi(\mathbf{x}) - \mathbf{m}_v||^2$.
- The kernel-k-means proceeds as follows: each new data point $\mathbf{x}_{t+1}$, assigned to the closest mean $\mathbf{m}_v$.
  Update: $\mathbf{m}_v^{t+1} = \mathbf{m}_v^t + \zeta(\Phi(\mathbf{x_{t+1}}) - \mathbf{m}_v^t)$, where
  ζ =1/#(points in cluster v).

Nonlinear variants of other algorithms

Classification and image indexing

- Clearly, distance-based algorithms like k-NN can be easily recast in the nonlinear kernel framework.
- In addition, it would be desirable to develop nonlinear forms of discriminant analysis based on kernels.
- PCA has the shortcoming of only being able to take into account second order correlation between pixels. Nonlinear component analysis take into account higher order correlations.