Machine Intelligence II

Kernel PCA

Tutorial

Neural Information Processing Group (Prof. Dr. Klaus Obermayer)

26.05.2016

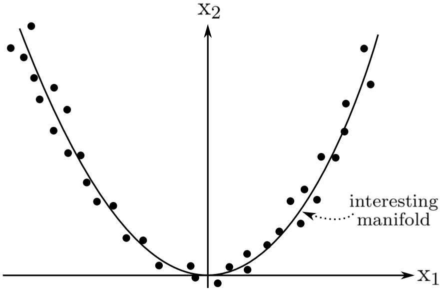
Projection Methods

- Principal Component Analysis (PCA)
- Online-PCA $\Rightarrow$ Hebbian Learning
- Nonlinear Structure $\Rightarrow$ Kernel PCA
- Source Separation
  - Model based $\Rightarrow$ Independent Component Analysis (ICA)
  - Cost Function Based $\Rightarrow$ Projection Pursuit

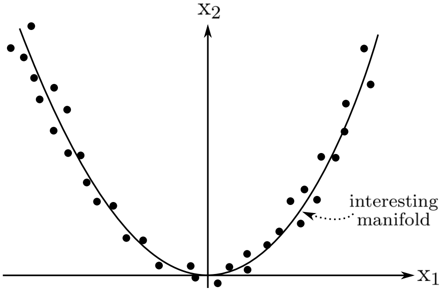
Kernel Principal Component Analysis

Goal: finding nonlinear regularities

Kernel Principal Component Analysis: Motivation



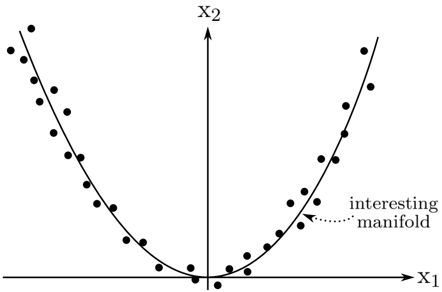
Kernel Principal Component Analysis: Motivation



in the original space:

- standard PCA: two directions with high variance
- "interesting" feature is a non-linear combination of elementary features

Kernel Principal Component Analysis: Motivation

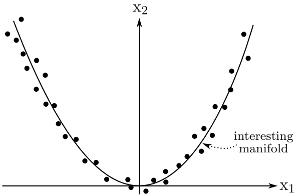


in the original space:

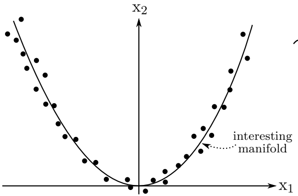
- standard PCA: two directions with high variance
- "interesting" feature is a non-linear combination of elementary features

$\Rightarrow$ problem for standard PCA

Kernel Principal Component Analysis: Intuition

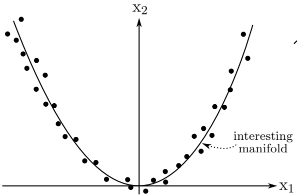


Kernel Principal Component Analysis: Intuition

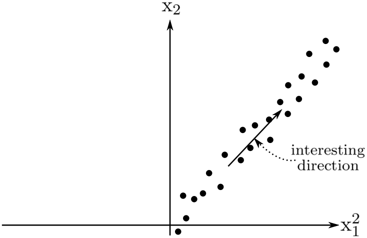


$\rightsquigarrow$

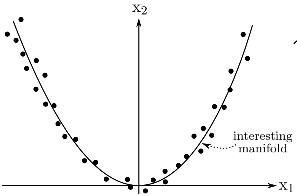
Kernel Principal Component Analysis: Intuition



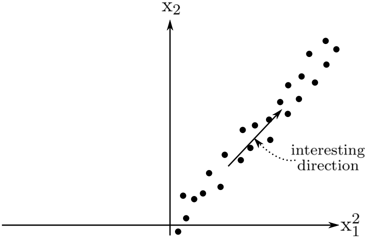
$\rightsquigarrow$



Kernel Principal Component Analysis: Intuition

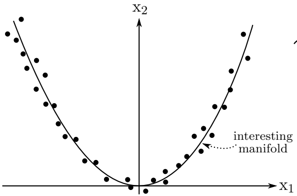


$\rightsquigarrow$

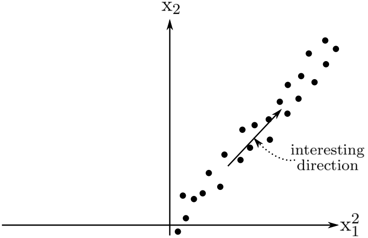


original (data) space $\rightsquigarrow$ transformed (feature) space:

Kernel Principal Component Analysis: Intuition



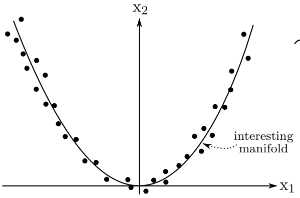
$\rightsquigarrow$



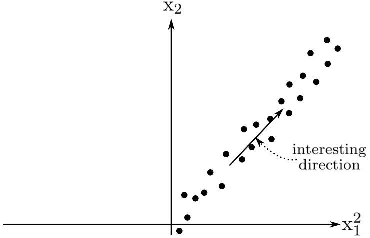
original (data) space $\quad\rightsquigarrow\quad$ transformed (feature) space:

$\rightsquigarrow$ "interesting" features correspond to directions of high variance

Kernel Principal Component Analysis: Intuition



$\rightsquigarrow$

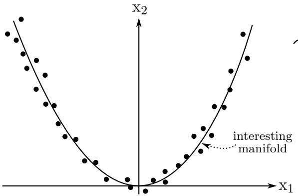


original (data) space $\rightsquigarrow$ transformed (feature) space:

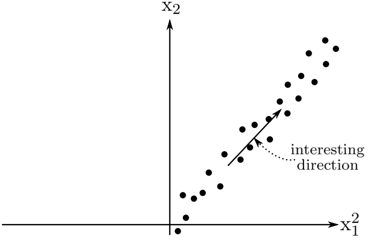
$\rightsquigarrow$ "interesting" features correspond to directions of high variance

Agenda

Kernel Principal Component Analysis: Intuition



$\rightsquigarrow$



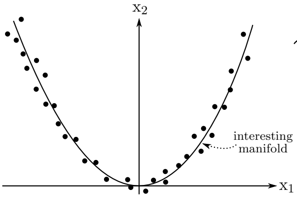
original (data) space $\rightsquigarrow$ transformed (feature) space:

$\rightsquigarrow$ "interesting" features correspond to directions of high variance

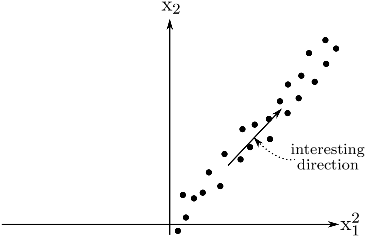
Agenda

1. non-linear preprocessing:

Kernel Principal Component Analysis: Intuition



$\rightsquigarrow$



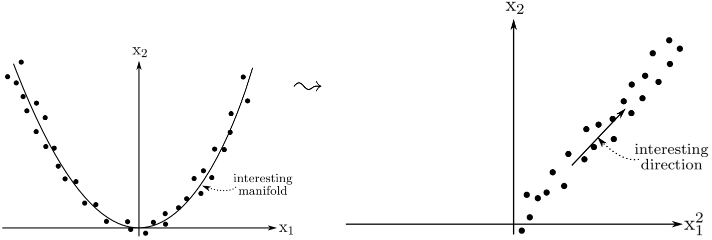
original (data) space $\rightsquigarrow$ transformed (feature) space:

$\rightsquigarrow$ "interesting" features correspond to directions of high variance

Agenda

1. non-linear preprocessing:
   transformation into an "appropriate' feature space $\underline{\phi} : \underline{\mathbf{x}} \rightarrow \underline{\phi}(\underline{\mathbf{x}})$

Kernel Principal Component Analysis: Intuition



original (data) space $\rightsquigarrow$ transformed (feature) space:

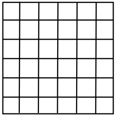
$\rightsquigarrow$ "interesting" features correspond to directions of high variance

Agenda

1. non-linear preprocessing:
   transformation into an "appropriate' feature space $\underline{\phi} : \underline{\mathbf{x}} \rightarrow \underline{\phi}(\underline{\mathbf{x}})$
2. apply standard linear methods

Why we need the Kernel Trick

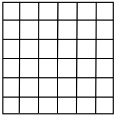
Problem: Some feature spaces may be extremely high-dimensional



pixel image

Why we need the Kernel Trick

Problem: Some feature spaces may be extremely high-dimensional

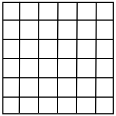


pixel image

- interesting structure is hidden in correlations between pixel values

Why we need the Kernel Trick

Problem: Some feature spaces may be extremely high-dimensional



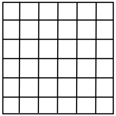
pixel image

- interesting structure is hidden in correlations between pixel values
- suitable feature space: space spanned by all d^{th}-order monomials

$$\text{e.g. } d = 2 : \mathrm{x}_1^2, \mathrm{x}_1\mathrm{x}_2, \mathrm{x}_2^2, \mathrm{x}_1\mathrm{x}_3, \mathrm{x}_2\mathrm{x}_3, \mathrm{x}_3^2, \ldots$$

Why we need the Kernel Trick

Problem: Some feature spaces may be extremely high-dimensional



pixel image

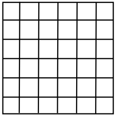
- interesting structure is hidden in correlations between pixel values
- suitable feature space: space spanned by all d^{th}-order monomials

$$\text{e.g. } d = 2 : \mathrm{x}_1^2, \mathrm{x}_1\mathrm{x}_2, \mathrm{x}_2^2, \mathrm{x}_1\mathrm{x}_3, \mathrm{x}_2\mathrm{x}_3, \mathrm{x}_3^2, \ldots$$

- dimensionality $O(N^d)$ prohibits "direct" application of this idea

Why we need the Kernel Trick

Problem: Some feature spaces may be extremely high-dimensional



pixel image

- interesting structure is hidden in correlations between pixel values
- suitable feature space: space spanned by all d^{th}-order monomials

$$\text{e.g. } d = 2 : \mathrm{x}_1^2, \mathrm{x}_1\mathrm{x}_2, \mathrm{x}_2^2, \mathrm{x}_1\mathrm{x}_3, \mathrm{x}_2\mathrm{x}_3, \mathrm{x}_3^2, \ldots$$

- dimensionality $O(N^d)$ prohibits “direct” application of this idea

$\rightsquigarrow$ **kernel trick** allows to avoid this problem

Insight: PCA can be reformulated to use Scalar Products

PCA involves solution of the eigenvalue problem of the covariance matrix

$$\underline{\mathbf{C}}\mathbf{e}_k = \lambda_k \underline{\mathbf{e}}_k$$

Insight: PCA can be reformulated to use Scalar Products

PCA involves solution of the eigenvalue problem of the covariance matrix

$$\underline{\mathbf{C}}\mathbf{e}_k = \lambda_k \underline{\mathbf{e}}_k \qquad \rightsquigarrow \qquad \underline{\mathbf{K}}\mathbf{a}_k = p\lambda_k \underline{\mathbf{a}}_k$$

Insight: PCA can be reformulated to use Scalar Products

PCA involves solution of the eigenvalue problem of the covariance matrix

$$\underline{\mathbf{C}}\mathbf{e}_k = \lambda_k \underline{\mathbf{e}}_k \qquad \rightsquigarrow \qquad \underline{\mathbf{K}}\mathbf{a}_k = p\lambda_k \underline{\mathbf{a}}_k$$

where $a_k^{(\beta)}$ describe an expansion of the eigenvectors in terms of data points

Insight: PCA can be reformulated to use Scalar Products

PCA involves solution of the eigenvalue problem of the covariance matrix

$$\underline{\mathbf{C}}\underline{\mathbf{e}}_k = \lambda_k \underline{\mathbf{e}}_k \qquad \rightsquigarrow \qquad \underline{\mathbf{K}}\underline{\mathbf{a}}_k = p\lambda_k \underline{\mathbf{a}}_k$$

where $a_k^{(\beta)}$ describe an expansion of the eigenvectors in terms of data points

$$\underline{\mathbf{e}}_k = \sum_{\beta=1}^{p} a_k^{(\beta)} \underline{\mathbf{x}}^{(\beta)}$$

Insight: PCA can be reformulated to use Scalar Products

PCA involves solution of the eigenvalue problem of the covariance matrix

$$\underline{\mathbf{C}}\underline{\mathbf{e}}_k = \lambda_k \underline{\mathbf{e}}_k \qquad \rightsquigarrow \qquad \underline{\mathbf{K}}\underline{\mathbf{a}}_k = p\lambda_k \underline{\mathbf{a}}_k$$

where $a_k^{(\beta)}$ describe an expansion of the eigenvectors in terms of data points

$$\underline{\mathbf{e}}_k = \sum_{\beta=1}^{p} a_k^{(\beta)} \underline{\mathbf{x}}^{(\beta)}$$

always possible: PCs "live" in the subspace spanned by the (centered) data

Insight: PCA can be reformulated to use Scalar Products

PCA involves solution of the eigenvalue problem of the covariance matrix

$$\underline{\mathbf{C}}\underline{\mathbf{e}}_k = \lambda_k \underline{\mathbf{e}}_k \qquad \rightsquigarrow \qquad \underline{\mathbf{K}}\underline{\mathbf{a}}_k = p\lambda_k \underline{\mathbf{a}}_k$$

where $a_k^{(\beta)}$ describe an expansion of the eigenvectors in terms of data points

$$\underline{\mathbf{e}}_k = \sum_{\beta=1}^{p} a_k^{(\beta)} \underline{\mathbf{x}}^{(\beta)}$$

always possible: PCs "live" in the subspace spanned by the (centered) data

$\Rightarrow$ We can write EV-problem s.t. data-points occur only in scalar products!

Insight: PCA can be reformulated to use Scalar Products

PCA involves solution of the eigenvalue problem of the covariance matrix

$$\underline{\mathbf{C}}\underline{\mathbf{e}}_k = \lambda_k \underline{\mathbf{e}}_k \qquad \rightsquigarrow \qquad \underline{\mathbf{K}}\underline{\mathbf{a}}_k = p\lambda_k \underline{\mathbf{a}}_k$$

where $a_k^{(\beta)}$ describe an expansion of the eigenvectors in terms of data points

$$\underline{\mathbf{e}}_k = \sum_{\beta=1}^{p} a_k^{(\beta)} \underline{\mathbf{x}}^{(\beta)}$$

always possible: PCs "live" in the subspace spanned by the (centered) data

$\Rightarrow$ We can write EV-problem s.t. data-points occur only in scalar products!

note: Eigenvectors $\underline{\mathbf{e}}_k \in \mathbb{R}^N$ but coefficients $\underline{\mathbf{a}}_k \in \mathbb{R}^p$

Why Scalar Products?

- If PCA doesn't work nonlinear pre-processing can simplify the problem

Why Scalar Products?

- If PCA doesn't work nonlinear pre-processing can simplify the problem

$$\underline{\phi} : \underline{\mathbf{x}} \xrightarrow{\text{non-linear transformation}} \underline{\phi}(\underline{\mathbf{x}})$$

Why Scalar Products?

- If PCA doesn't work nonlinear pre-processing can simplify the problem

$$\underline{\phi} : \underline{\mathbf{x}} \xrightarrow{\text{non-linear transformation}} \underline{\phi}(\underline{\mathbf{x}})$$

Mercer's theorem

Every **positive definite** kernel k corresponds to a scalar product in some metric feature space (*cf. MI I*).

Why Scalar Products?

- If PCA doesn't work nonlinear pre-processing can simplify the problem

$$\underline{\phi} : \underline{\mathbf{x}} \xrightarrow{\text{non-linear transformation}} \underline{\phi}(\underline{\mathbf{x}})$$

Mercer's theorem

Every **positive definite** kernel k corresponds to a scalar product in some metric feature space (*cf. MI I*).

Kernel trick

$\Rightarrow$ avoid direct transformation $\underline{\phi}$

Why Scalar Products?

- If PCA doesn't work nonlinear pre-processing can simplify the problem

$$\underline{\phi} : \underline{\mathbf{x}} \xrightarrow{\text{non-linear transformation}} \underline{\phi}(\underline{\mathbf{x}})$$

Mercer's theorem

Every **positive definite** kernel k corresponds to a scalar product in some metric feature space (*cf. MI I*).

Kernel trick

$\Rightarrow$ avoid direct transformation $\underline{\phi}$

$\Rightarrow$ replace all scalar products by "kernel functions"

$$\underline{\phi}^T_{(\underline{\mathbf{x}})} \underline{\phi}_{(\underline{\mathbf{x}}')} \longleftrightarrow k(\underline{\mathbf{x}}, \underline{\mathbf{x}}')$$

Why Scalar Products?

- If PCA doesn't work nonlinear pre-processing can simplify the problem

$$\underline{\phi} : \underline{\mathbf{x}} \xrightarrow{\text{non-linear transformation}} \underline{\phi}(\underline{\mathbf{x}})$$

Mercer's theorem

Every **positive definite** kernel k corresponds to a scalar product in some metric feature space (*cf. MI I*).

Kernel trick

$\Rightarrow$ avoid direct transformation $\underline{\phi}$

$\Rightarrow$ replace all scalar products by "kernel functions"

$$\underline{\phi}^T_{(\underline{\mathbf{x}})}\underline{\phi}_{(\underline{\mathbf{x}}')} \longleftrightarrow k(\underline{\mathbf{x}}, \underline{\mathbf{x}}')$$

---

if an algorithm can be formulated solely in terms of scalar products, a non-linear version can be derived "without actually projecting" into the (highdimensional) feature space

Kernels and Applications

Popular Kernel Functions

$$k_{(\underline{\mathbf{x}},\underline{\mathbf{x}}')} = \left(\underline{\mathbf{x}}^T \underline{\mathbf{x}}' + 1\right)^d$$

polynomial kernel of degree d
image processing (pixel correlation)

$$k_{(\underline{\mathbf{x}},\underline{\mathbf{x}}')} = \exp\left\{ -\frac{\left(\underline{\mathbf{x}}-\underline{\mathbf{x}}'\right)^2}{2\sigma^2} \right\}$$

RBF-kernel with range σ
infinite dimensional feature space

$$k_{(\underline{\mathbf{x}},\underline{\mathbf{x}}')} = \tanh\left\{ K\underline{\mathbf{x}}^T \underline{\mathbf{x}}' + \theta \right\}$$

neural network kernel with parameters K and θ
not necessarily positive definite

Other Kernelizable Methods

- Support Vector Machines (MI I)
- Fisher Discriminant Analysis, Canonical Correlation Analysis
- K-Means Clustering, Self-Organizing Maps (both MI II)

1. calculate the un-normalized kernel matrix $\underline{\widetilde{\mathbf{K}}} \in \mathbb{R}^{p,p}$

$$\widetilde{K}_{\alpha\beta} = k\left(\underline{\mathbf{x}}^{(\alpha)}, \underline{\mathbf{x}}^{(\beta)}\right)$$

1. calculate the un-normalized kernel matrix $\underline{\widetilde{\mathbf{K}}} \in \mathbb{R}^{p,p}$

$$\widetilde{K}_{\alpha\beta} = k\left(\underline{\mathbf{x}}^{(\alpha)}, \underline{\mathbf{x}}^{(\beta)}\right)$$

2. center the kernel matrix ($\Longrightarrow$ centered data in feature space)

$$K_{\alpha\beta} = \widetilde{K}_{\alpha\beta} - \frac{1}{p}\sum_{\delta=1}^{p} \widetilde{K}_{\alpha\delta} - \frac{1}{p}\sum_{\gamma=1}^{p} \widetilde{K}_{\gamma\beta} + \frac{1}{p^2}\sum_{\gamma,\delta=1}^{p} \widetilde{K}_{\gamma\delta}$$

1. calculate the un-normalized kernel matrix $\underline{\widetilde{\mathbf{K}}} \in \mathbb{R}^{p,p}$

$$\widetilde{K}_{\alpha\beta} = k\left(\underline{\mathbf{x}}^{(\alpha)}, \underline{\mathbf{x}}^{(\beta)}\right)$$

2. center the kernel matrix ($\Longrightarrow$ centered data in feature space)

$$K_{\alpha\beta} = \widetilde{K}_{\alpha\beta} - \frac{1}{p}\sum_{\delta=1}^{p} \widetilde{K}_{\alpha\delta} - \frac{1}{p}\sum_{\gamma=1}^{p} \widetilde{K}_{\gamma\beta} + \frac{1}{p^2}\sum_{\gamma,\delta=1}^{p} \widetilde{K}_{\gamma\delta}$$

3. solve the eigenvalue problem $\boxed{\frac{1}{p}\underline{\mathbf{K}\widetilde{\mathbf{a}}}_k = \lambda_k \underline{\widetilde{\mathbf{a}}}_k}$

1. calculate the un-normalized kernel matrix $\underline{\widetilde{\mathbf{K}}} \in \mathbb{R}^{p,p}$

$$\widetilde{K}_{\alpha\beta} = k\left(\underline{\mathbf{x}}^{(\alpha)}, \underline{\mathbf{x}}^{(\beta)}\right)$$

2. center the kernel matrix ($\Longrightarrow$ centered data in feature space)

$$K_{\alpha\beta} = \widetilde{K}_{\alpha\beta} - \frac{1}{p}\sum_{\delta=1}^{p} \widetilde{K}_{\alpha\delta} - \frac{1}{p}\sum_{\gamma=1}^{p} \widetilde{K}_{\gamma\beta} + \frac{1}{p^2}\sum_{\gamma,\delta=1}^{p} \widetilde{K}_{\gamma\delta}$$

3. solve the eigenvalue problem $\boxed{\frac{1}{p}\underline{\mathbf{K}}\underline{\widetilde{\mathbf{a}}}_k = \lambda_k \underline{\widetilde{\mathbf{a}}}_k}$

4. normalize eigenvectors to unit length ($\Longrightarrow$ $\underline{\mathbf{e}}_k^2 = 1$ in feature space)

$$\underline{\mathbf{a}}_k = \frac{1}{\sqrt{p\lambda_k}\,\|\underline{\widetilde{\mathbf{a}}}_k\|}\underline{\widetilde{\mathbf{a}}}_k$$

1. calculate the un-normalized kernel matrix $\widetilde{\underline{\mathbf{K}}} \in \mathbb{R}^{p,p}$

$$\widetilde{K}_{\alpha\beta} = k\left(\underline{\mathbf{x}}^{(\alpha)}, \underline{\mathbf{x}}^{(\beta)}\right)$$

2. center the kernel matrix ($\Longrightarrow$ centered data in feature space)

$$K_{\alpha\beta} = \widetilde{K}_{\alpha\beta} - \frac{1}{p}\sum_{\delta=1}^{p} \widetilde{K}_{\alpha\delta} - \frac{1}{p}\sum_{\gamma=1}^{p} \widetilde{K}_{\gamma\beta} + \frac{1}{p^2}\sum_{\gamma,\delta=1}^{p} \widetilde{K}_{\gamma\delta}$$

3. solve the eigenvalue problem $\boxed{\frac{1}{p}\underline{\mathbf{K}}\widetilde{\underline{\mathbf{a}}}_k = \lambda_k \widetilde{\underline{\mathbf{a}}}_k}$

4. normalize eigenvectors to unit length ($\Longrightarrow$ $\underline{\mathbf{e}}_k^2 = 1$ in feature space)

$$\underline{\mathbf{a}}_k = \frac{1}{\sqrt{p\lambda_k}\,\|\widetilde{\underline{\mathbf{a}}}_k\|}\widetilde{\underline{\mathbf{a}}}_k$$

5. calculate projections of data points $\underline{\mathbf{x}}^{(\alpha)}$ onto eigenvectors $\underline{\mathbf{e}}_k$

$$u_k\left(\underline{\mathbf{x}}^{(\alpha)}\right) = \sum_{\beta=1}^{p} a_k^{(\beta)} K_{\beta\alpha} \quad \leftarrow$$ use centered kernel matrix and normalized eigenvector!

Projections of new points onto PCs in feature space

For data points $\underline{\mathbf{x}}^{(\alpha)}$ we have onto the k-th PC:

$$u_k\left(\underline{\mathbf{x}}^{(\alpha)}\right) = \sum_{\beta=1}^{p} a_k^{(\beta)} K_{\beta\alpha} \quad \leftarrow \text{use centered kernel matrix and normalized eigenvector!}$$

Projections of new points onto PCs in feature space

For data points $\underline{\mathbf{x}}^{(\alpha)}$ we have onto the k-th PC:

$$u_k\left(\underline{\mathbf{x}}^{(\alpha)}\right) = \sum_{\beta=1}^{p} a_k^{(\beta)} K_{\beta\alpha} \quad \leftarrow \begin{array}{c}\text{use centered kernel matrix}\\ \text{and normalized eigenvector!}\end{array}$$

More generally, for arbitrary $\underline{\mathbf{x}}$ the projection is computed as:

$$\begin{aligned} u_k\left(\underline{\mathbf{x}}\right) &= \sum_{\beta=1}^{p} a_k^{(\beta)} \underline{\phi}^T_{\left(\underline{\mathbf{x}}^{(\beta)}\right)} \underline{\phi}_{(\underline{\mathbf{x}})} \qquad \leftarrow \text{centered feature vectors} \\ &= \sum_{\beta=1}^{p} a_k^{(\beta)} \left(k(\underline{\mathbf{x}}^{(\beta)}, \underline{\mathbf{x}}) - \frac{1}{p}\sum_{\delta=1}^{p} \widetilde{K}_{\beta\delta} - \frac{1}{p}\sum_{\gamma=1}^{p} k(\underline{\mathbf{x}}^{(\gamma)}, \underline{\mathbf{x}}) + \frac{1}{p^2}\sum_{\gamma,\delta=1}^{p} \widetilde{K}_{\gamma\delta} \right) \end{aligned}$$

KPCA: Additional Thoughts

- kernel-PCA $\widehat{=}$ PCA in feature space (!!!)

KPCA: Additional Thoughts

- kernel-PCA $\widehat{=}$ PCA in feature space (!!!)
  - $\rightsquigarrow$ projections onto PCs are uncorrelated
  - $\rightsquigarrow$ λ_l: variance of the data along principal component k (in feature space)
  - $\rightsquigarrow$ "minimal" mean-squared approximation of data (in feature space)

KPCA: Additional Thoughts

- kernel-PCA $\widehat{=}$ PCA in feature space (!!!)
  - $\rightsquigarrow$ projections onto PCs are uncorrelated
  - $\rightsquigarrow$ λ_l: variance of the data along principal component k (in feature space)
  - $\rightsquigarrow$ "minimal" mean-squared approximation of data (in feature space)
- #PCs can exceed #dimensions in the original space

KPCA: Additional Thoughts

- kernel-PCA $\widehat{=}$ PCA in feature space (!!!)
  - $\rightsquigarrow$ projections onto PCs are uncorrelated
  - $\rightsquigarrow$ λ_l: variance of the data along principal component k (in feature space)
  - $\rightsquigarrow$ "minimal" mean-squared approximation of data (in feature space)
- #PCs can exceed #dimensions in the original space
- kernel matrices may be very large
  - $\rightsquigarrow$ only first few eigenvectors (largest eigenvalues) are of interest
  - $\rightsquigarrow$ use specialized (iterative) routines (e.g. ARPACK via `eigs`)