Machine Learning

B. Unsupervised Learning
B.2 Dimensionality Reduction

Lars Schmidt-Thieme, Nicolas Schilling

Information Systems and Machine Learning Lab (ISMLL)
Institute for Computer Science
University of Hildesheim, Germany

Outline

1. Principal Components Analysis

2. Non-linear Dimensionality Reduction

3. Supervised Dimensionality Reduction

Outline

1. Principal Components Analysis

2. Non-linear Dimensionality Reduction

3. Supervised Dimensionality Reduction

The Dimensionality Reduction Problem

Given

- a set $\mathcal{X}$ called **data space**, e.g., $\mathcal{X} := \mathbb{R}^m$,
- a set $X \subseteq \mathcal{X}$ called **data**,
- a function

$$D : \bigcup_{X \subseteq \mathcal{X}, K \in \mathbb{N}} (\mathbb{R}^K)^X \to \mathbb{R}_0^+$$

  called **distortion** where $D(P)$ measures how bad a low dimensional representation $P : X \to \mathbb{R}^K$ for a data set $X \subseteq \mathcal{X}$ is, and
- a number $K \in \mathbb{N}$ of latent dimensions,

find a low dimensional representation $P : X \to \mathbb{R}^K$ with K dimensions with minimal distortion $D(P)$.

Distortions for Dimensionality Reduction (1/2)

Let $d_{\mathcal{X}}$ be a distance on $\mathcal{X}$ and d_Z be a distance on the latent space $\mathbb{R}^K$, usually just the Euclidean distance

$$d_Z(v, w) := ||v - w||_2 = (\sum_{i=1}^{K}(v_i - w_i)^2)^{\frac{1}{2}}$$

Multidimensional scaling aims to find latent representations P that **reproduce the distance measure $d_{\mathcal{X}}$** as good as possible:

$$D(P) := \frac{2}{|X|(|X|-1)} \sum_{\substack{x,x' \in X \\ x \neq x'}} (d_{\mathcal{X}}(x, x') - d_Z(P(x), P(x')))^2$$

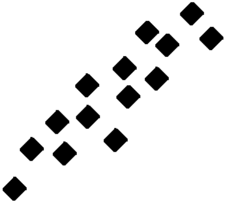
$$= \frac{2}{n(n-1)} \sum_{i=1}^{n} \sum_{j=1}^{i-1} (d_{\mathcal{X}}(x_i, x_j) - ||z_i - z_j||)^2, \quad z_i := P(x_i)$$

Distortions for Dimensionality Reduction (2/2)

Feature reconstruction methods aim to find latent representations P and reconstruction maps $r : \mathbb{R}^K \rightarrow \mathcal{X}$ from a given class of maps that **reconstruct features** as good as possible:

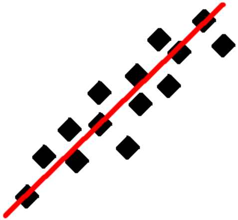
$$D(P, r) := \frac{1}{|X|} \sum_{x \in X} d_{\mathcal{X}}(x, r(P(x)))$$
$$= \frac{1}{n} \sum_{i=1}^{n} d_{\mathcal{X}}(x_i, r(z_i)), \quad z_i := P(x_i)$$

PCA: Intuition



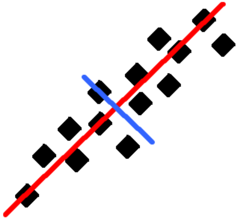
- We want to find an **orthogonal basis** which represent directions of **maximum variance**

PCA: Intuition



- we want to find an **orthogonal basis** which represent directions of **maximum variance**
- red line indicates direction of maximal variance within the data

PCA: Intuition



- we want to find an **orthogonal basis** which represent directions of **maximum variance**
- red line indicates direction of maximal variance within the data
- blue line represents direction of maximal variance given that it has to be orthogonal to the red line

PCA: Task

We assume that the data has mean Zero:

$$\sum_{i=1}^{n} x_i = \mathbf{0}$$

Then PCA can be accomplished by finding the top K eigenvalues of the covariance matrix $X^\top X$ of the data!

- Mean has to be zero to account for different units in the data (Degrees C vs Degrees F)
- Can also be accomplished through a **singular value decomposition** of the data matrix X as we will see

Singular Value Decomposition (SVD)

Theorem (Existence of SVD)

For every $A \in \mathbb{R}^{n\times m}$ there exist matrices

$$U \in \mathbb{R}^{n\times k}, V \in \mathbb{R}^{m\times k}, \Sigma := diag(\sigma_1, \ldots, \sigma_k) \in \mathbb{R}^{k\times k}, \qquad k := \min\{n, m\}$$

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > \sigma_{r+1} = \cdots = \sigma_k = 0, \qquad r := \text{rank}(A)$$

$$U, V \textit{ orthonormal, i.e., } U^T U = I, V^T V = I$$

with

$$A = U \Sigma V^T$$

σ_i *are called* **singular values of** A.

Note: $I := \text{diag}(1, \ldots, 1) \in \mathbb{R}^{k\times k}$ denotes the unit matrix.

Singular Value Decomposition (SVD; 2/2)

It holds:

a) σ_i^2 are eigenvalues and V_i eigenvectors of $A^T A$:

$$(A^T A)V_i = \sigma_i^2 V_i, \quad i = 1, \ldots, k, V = (V_1, \ldots, V_k)$$

b) σ_i^2 are eigenvalues and U_i eigenvectors of AA^T:

$$(AA^T)U_i = \sigma_i^2 U_i, \quad i = 1, \ldots, k, U = (U_1, \ldots, U_k)$$

Singular Value Decomposition (SVD; 2/2)

It holds:

a) σ_i^2 are eigenvalues and V_i eigenvectors of $A^T A$:

$$(A^T A)V_i = \sigma_i^2 V_i, \quad i = 1, \ldots, k, V = (V_1, \ldots, V_k)$$

b) σ_i^2 are eigenvalues and U_i eigenvectors of AA^T:

$$(AA^T)U_i = \sigma_i^2 U_i, \quad i = 1, \ldots, k, U = (U_1, \ldots, U_k)$$

proof:

$$\text{a) } (A^T A)V_i = V\Sigma^T U^T\, U\Sigma V^T V_i = V\Sigma^2 e_i = \sigma_i^2 V_i$$

$$\text{b) } (AA^T)U_i = U\Sigma^T V^T\, V\Sigma^T U^T U_i = U\Sigma^2 e_i = \sigma_i^2 U_i$$

Truncated SVD

Let $A \in \mathbb{R}^{n \times m}$ and $U\Sigma V^T = A$ its SVD. Then for $k' \leq \min\{n, m\}$ the decomposition

$$A = U'\Sigma' V'^T$$

with

$$U' := (U_{,1}, \ldots, U_{,k'}), V' := (V_{,1}, \ldots, V_{,k'}), \Sigma' := \text{diag}(\sigma_1, \ldots, \sigma_{k'})$$

is called **truncated SVD with rank** k'.

Low Rank Approximation

Let $A \in \mathbb{R}^{n \times m}$. For $k \leq \min\{n, m\}$, any pair of matrices

$$U \in \mathbb{R}^{n \times k}, V \in \mathbb{R}^{m \times k}$$

is called a **low rank approximation of A with rank k**.
The matrix

$$UV^T$$

is called the **reconstruction of A by U, V** and the quantity

$$||A - UV^T||_F$$

the **L2 reconstruction error**.

Low Rank Approximation

Let $A \in \mathbb{R}^{n \times m}$. For $k \leq \min\{n, m\}$, any pair of matrices

$$U \in \mathbb{R}^{n \times k}, V \in \mathbb{R}^{m \times k}$$

is called a **low rank approximation of A with rank k**.
The matrix

$$UV^T$$

is called the **reconstruction of A by U, V** and the quantity

$$||A - UV^T||_F = \sum_{i=1}^{n} \sum_{j=1}^{m} (A_{i,j} - U_i^T V_j)^2$$

the **L2 reconstruction error**.

Note: $||A||_F$ is called Frobenius norm.

Optimal Low Rank Approximation is Truncated SVD

Theorem (Low Rank Approximation; Eckart-Young theorem)

Let $A \in \mathbb{R}^{n \times m}$. For $k' \leq \min\{n, m\}$, the optimal low rank approximation of rank k' (i.e., with smallest reconstruction error)

$$(U^*, V^*) := \underset{U \in \mathbb{R}^{n \times k'}, V \in \mathbb{R}^{m \times k'}}{\arg\min} ||A - UV^T||^2$$

is the truncated SVD.

Note: As U, V do not have to be orthonormal, one can take $U := U'\Sigma'$, $V := V'$ for the SVD $A = U'\Sigma'V'^T$.

Principal Components Analysis (PCA)

Let $X := \{x_1, \ldots, x_n\} \subseteq \mathbb{R}^m$ be a data set and $K \in \mathbb{N}$ the number of latent dimensions ($K \leq m$).

PCA finds

- K principal components $v_1, \ldots, v_K \in \mathbb{R}^m$ and
- latent weights $z_i \in \mathbb{R}^K$ for each data point $i \in \{1, \ldots, n\}$,

such that the linear combination of the principal components

$$x_i \approx \sum_{k=1}^{K} z_{i,k} v_k$$

reconstructs the original features x_i as good as possible:

Principal Components Analysis (PCA)

Let $X := \{x_1, \ldots, x_n\} \subseteq \mathbb{R}^m$ be a data set and $K \in \mathbb{N}$ the number of latent dimensions ($K \leq m$).

PCA finds

- K principal components $v_1, \ldots, v_K \in \mathbb{R}^m$ and
- latent weights $z_i \in \mathbb{R}^K$ for each data point $i \in \{1, \ldots, n\}$,

such that the linear combination of the principal components reconstructs the original features x_i as good as possible:

$$
\begin{aligned}
\arg\min_{\substack{v_1,\ldots,v_K \\ z_1,\ldots,z_n}} \sum_{i=1}^n ||x_i - \sum_{k=1}^K z_{i,k} v_k||^2 \\
= \sum_{i=1}^n ||x_i - V z_i||^2, \quad V := (v_1, \ldots, v_K)^T
\end{aligned}
$$

Principal Components Analysis (PCA)

Let $X := \{x_1, \ldots, x_n\} \subseteq \mathbb{R}^m$ be a data set and $K \in \mathbb{N}$ the number of latent dimensions ($K \leq m$).

PCA finds

- K principal components $v_1, \ldots, v_K \in \mathbb{R}^m$ and
- latent weights $z_i \in \mathbb{R}^K$ for each data point $i \in \{1, \ldots, n\}$,

such that the linear combination of the principal components reconstructs the original features x_i as good as possible:

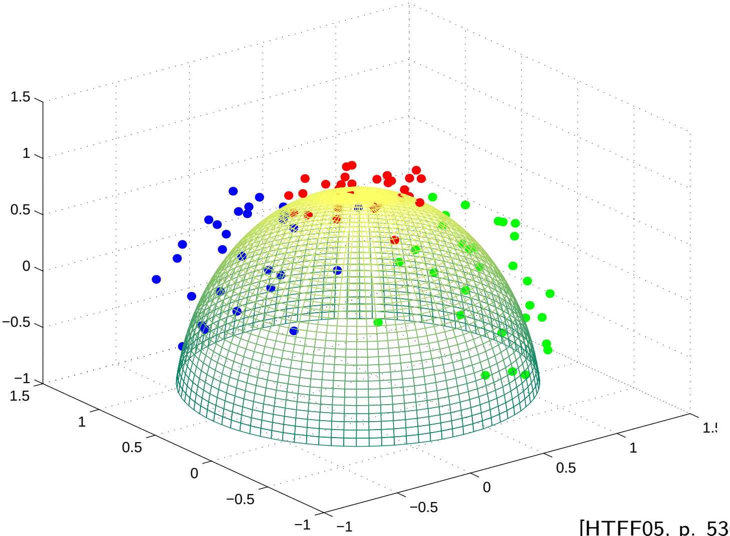
$$\begin{aligned}
\underset{\substack{v_1,\ldots,v_K \\ z_1,\ldots,z_n}}{\arg\min} \sum_{i=1}^{n} ||x_i - \sum_{k=1}^{K} z_{i,k} v_k||^2 \\
= \sum_{i=1}^{n} ||x_i - V z_i||^2, \quad V := (v_1, \ldots, v_K)^T \\
= ||X - Z V^T||_F^2, \quad X := (x_1, \ldots, x_n)^T, Z := (z_1, \ldots, z_n)^T
\end{aligned}$$

thus PCA is just the SVD of the data matrix X.

PCA Algorithm

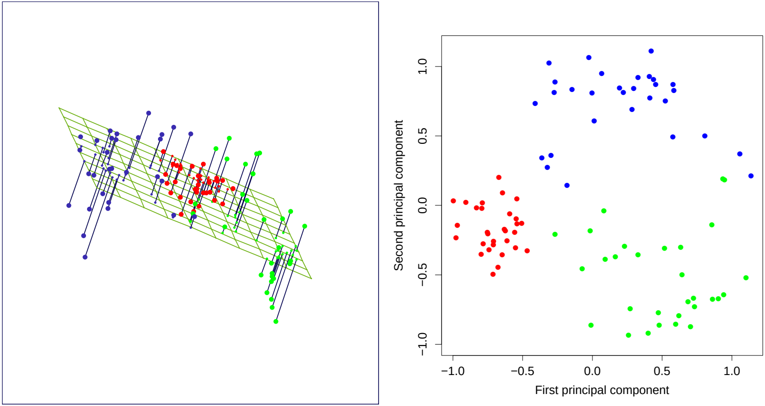
1: **procedure** DIMRED-PCA($\mathcal{D} := \{x_1, \ldots, x_N\} \subseteq \mathbb{R}^M, K \in \mathbb{N}$)
2: $X := (x_1, x_2, \ldots, x_N)^T$
3: $(U, \Sigma, V) := \text{svd}(X)$
4: $Z := U_{.,1:K} \cdot \Sigma_{1:K,1:K}$
5: **return** $\mathcal{D}^{\text{dimred}} := \{Z_{1,.}, \ldots, Z_{N,.}\}$

Principal Components Analysis (Example 1)



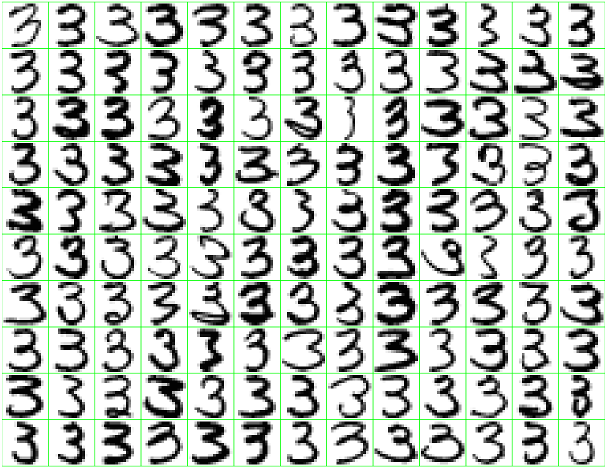
[HTFF05, p. 530]

Principal Components Analysis (Example 1)



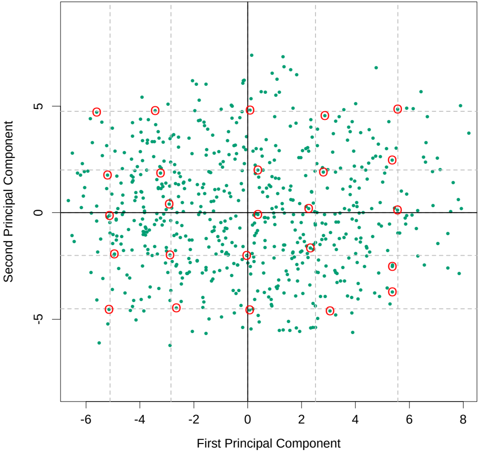
[HTFF05, p. 536]

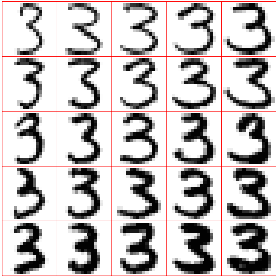
Principal Components Analysis (Example 2)



[HTFF05, p. 537]

Principal Components Analysis (Example 2)





[HTFF05, p. 538]

Outline

1. Principal Components Analysis

2. Non-linear Dimensionality Reduction

3. Supervised Dimensionality Reduction

Linear Dimensionality Reduction

Dimensionality reduction accomplishes two tasks:

1. compute lower dimensional representations for **given data points** x_i
   - for PCA:

$$u_i = \Sigma^{-1} V^T x_i, \quad U := (u_1, \ldots, u_n)^T$$

Linear Dimensionality Reduction

Dimensionality reduction accomplishes two tasks:

1. compute lower dimensional representations for **given data points** x_i
   - for PCA:

$$u_i = \Sigma^{-1} V^T x_i, \quad U := (u_1, \ldots, u_n)^T$$

2. compute lower dimensional representations for **new data points** x (often called "fold in")
   - for PCA:

$$u := \arg\min_u ||x - V\Sigma u||^2 = \Sigma^{-1} V^T x$$

Linear Dimensionality Reduction

Dimensionality reduction accomplishes two tasks:

1. compute lower dimensional representations for **given data points** x_i
   - for PCA:

$$u_i = \Sigma^{-1} V^T x_i, \quad U := (u_1, \dots, u_n)^T$$

2. compute lower dimensional representations for **new data points** x (often called "fold in")
   - for PCA:

$$u := \arg\min_u ||x - V\Sigma u||^2 = \Sigma^{-1} V^T x$$

PCA is called a **linear dimensionality reduction technique** because the latent representations u depend linearly on the observed representations x.

Kernel Trick

Represent (conceptionally) non-linearity by linearity in a higher dimensional embedding

$$\phi : \mathbb{R}^m \rightarrow \mathbb{R}^{\tilde{m}}$$

but compute in lower dimensionality for methods that depend on x only through a scalar product

$$\tilde{x}^T \tilde{\theta} = \phi(x)^T \phi(\theta) = k(x, \theta), \quad x, \theta \in \mathbb{R}^m$$

if k can be computed without explicitly computing ϕ.

Kernel Trick / Example

Example:

$$\phi : \mathbb{R} \rightarrow \mathbb{R}^{1001},$$
$$x \mapsto \left(\binom{1000}{i}^{\frac{1}{2}} x^i \right)_{i=0,\dots,1000} = \begin{pmatrix} 1 \\ 31.62\, x \\ 706.75\, x^2 \\ \vdots \\ 31.62\, x^{999} \\ x^{1000} \end{pmatrix}$$

$$\tilde{x}^T \tilde{\theta} = \phi(x)^T \phi(\theta) = \sum_{i=0}^{1000} \binom{1000}{i} x^i \theta^i = (1 + x\theta)^{1000} =: k(x, \theta)$$

Naive computation:

- 2002 binomial coefficients, 3003 multiplications, 1000 additions.

Kernel computation:

- 1 multiplication, 1 addition, 1 exponentiation.

Kernel PCA

$$
\begin{aligned}
\phi :& \mathbb{R}^m \rightarrow \mathbb{R}^{\tilde{m}}, \quad \tilde{m} \gg m \\
\tilde{X} :=& \begin{pmatrix} \phi(x_1) \\ \phi(x_2) \\ \vdots \\ \phi(x_n) \end{pmatrix} \\
\tilde{X} \approx& U \Sigma \tilde{V}^T
\end{aligned}
$$

We can compute the columns of U as eigenvectors of $\tilde{X}\tilde{X}^T \in \mathbb{R}^{n \times n}$ without having to compute $\tilde{V} \in \mathbb{R}^{\tilde{m} \times k}$ (which is large!):

$$
\tilde{X}\tilde{X}^T U_i = \sigma_i^2 U_i
$$

Kernel PCA / Removing the Mean

Issue 1: The $\tilde{x}_i := \phi(x_i)$ may not have zero mean and thus distort PCA.

$$\tilde{x}'_i := \tilde{x}_i - \frac{1}{n} \sum_{i=1}^{n} \tilde{x}_i$$

Kernel PCA / Removing the Mean

Issue 1: The $\tilde{x}_i := \phi(x_i)$ may not have zero mean and thus distort PCA.

$$\begin{aligned}\tilde{x}_i' :=& \tilde{x}_i - \frac{1}{n}\sum_{i=1}^{n} \tilde{x}_i \\ =& \tilde{X}^T(I - \frac{1}{n}\mathbb{1}) \\ \tilde{X}' :=& (\tilde{x}_1', \ldots, \tilde{x}_n')^T = (I - \frac{1}{n}\mathbb{1})\tilde{X}^T\end{aligned}$$

Note: $\mathbb{1} := (1)_{i=1,\ldots,n, j=1,\ldots,n}$ vector of ones,
$I := (\delta(i = j))_{i=1,\ldots,n, j=1,\ldots,n}$ unity matrix.

Kernel PCA / Removing the Mean

Issue 1: The $\tilde{x}_i := \phi(x_i)$ may not have zero mean and thus distort PCA.

$$
\begin{aligned}
\tilde{x}'_i :=& \tilde{x}_i - \frac{1}{n}\sum_{i=1}^{n} \tilde{x}_i \\
=& \tilde{X}^T (I - \frac{1}{n}\mathbb{1}) \\
\tilde{X}' :=& (\tilde{x}'_1, \ldots, \tilde{x}'_n)^T = (I - \frac{1}{n}\mathbb{1})\tilde{X}^T \\
K' :=& \tilde{X}'\tilde{X}'^T = (I - \frac{1}{n}\mathbb{1})\tilde{X}^T\tilde{X}(I - \frac{1}{n}\mathbb{1}) \\
=& HKH, \quad H := (I - \frac{1}{n}\mathbb{1}) \text{ centering matrix}
\end{aligned}
$$

Thus, the kernel matrix K' with means removed can be computed from the kernel matrix K without having to access coordinates.

Kernel PCA / Fold In

Issue 2: How to compute projections u of new points x (as $\tilde{V}$ is not computed)?

$$u := \arg\min_u ||x - \tilde{V}\Sigma u||^2 = \Sigma^{-1}\tilde{V}^T x$$

With

$$\tilde{V} = \tilde{X}^T U \Sigma^{-1}$$
$$u = \Sigma^{-1}\tilde{V}^T x = \Sigma^{-1}\Sigma^{-1}U^T\tilde{X}x = \Sigma^{-2}U^T(k(x_i, x))_{i=1,\ldots,n}$$

u can be computed with access to kernel values only (and to U, Σ).

Kernel PCA / Summary

Given:

- data set $X := \{x_1, \ldots, x_n\} \subseteq \mathbb{R}^m$,
- kernel function $k : \mathbb{R}^m \times \mathbb{R}^m \rightarrow R$.

task 1: Learn latent representations U of data set X:

$$K := (k(x_i, x_j))_{i=1,\ldots,n, j=1,\ldots,n} \tag{0}$$

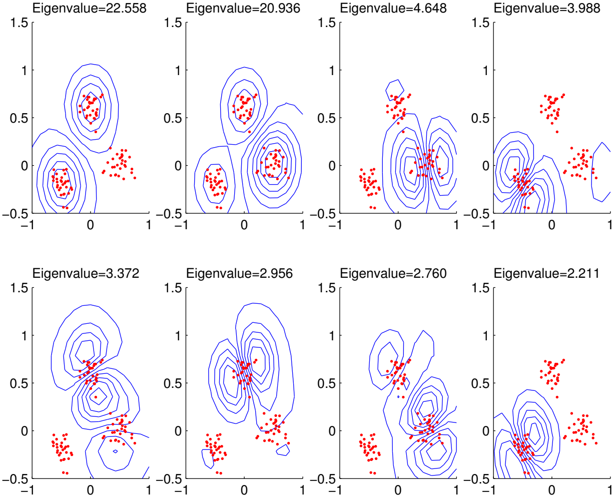
$$K' := HKH, \quad H := (I - \frac{1}{n}\mathbb{1}) \tag{1}$$

$$(U, \Sigma) := \text{eigen decomposition } (K') \tag{2}$$

task 2: Learn latent representation u of new point x:

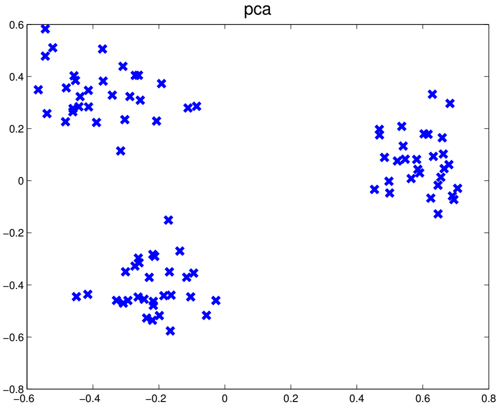
$$u := \Sigma^{-2} U^T (k(x_i, x))_{i=1,\ldots,n} \tag{3}$$

Kernel PCA: Example 1



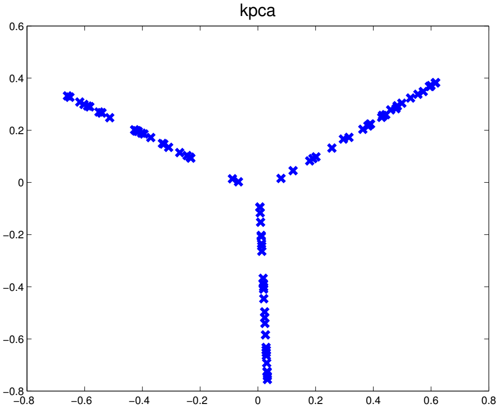
[Mur12, p. 493]

Kernel PCA: Example 2



[Mur12, p. 495]

Kernel PCA: Example 2



[Mur12, p. 495]

Outline

1. Principal Components Analysis

2. Non-linear Dimensionality Reduction

3. Supervised Dimensionality Reduction

Dimensionality Reduction as Pre-Processing

Given a prediction task and

a data set $\mathcal{D}^{\text{train}} := \{(x_1, y_1), \ldots, (x_n, y_n)\} \subseteq \mathbb{R}^m \times \mathcal{Y}$.

1. compute latent features $z_i \in \mathbb{R}^K$ for the objects of a data set by means of dimensionality reduction of the predictors x_i.
   - e.g., using PCA on $\{x_1, \ldots, x_n\} \subseteq \mathbb{R}^m$
2. learn a prediction model

   $$\hat{y} : \mathbb{R}^K \to \mathcal{Y}$$

   on the latent features based on

   $$\mathcal{D}'^{\text{train}} := \{(z_1, y_1), \ldots, (z_n, y_n)\}$$

3. treat the number K of latent dimensions as hyperparameter.
   - e.g., find using grid search.

Dimensionality Reduction as Pre-Processing

Advantages:

- simple procedure
- generic procedure
  - works with any dimensionality reduction method and prediction method as component methods.
- usually fast

Dimensionality Reduction as Pre-Processing

Advantages:

- simple procedure
- generic procedure
  - works with any dimensionality reduction method and prediction method as component methods.
- usually fast

Disadvantages:

- dimensionality reduction is **unsupervised**, i.e., not informed about the target that should be predicted later on.
  - leads to the very same latent features regardless of the prediction task.
  - likely not the best task-specific features are extracted.

Supervised PCA

$$
\begin{aligned}
p(z) &:= \mathcal{N}(z; 0, 1) \\
p(x \mid z; \mu_x, \sigma_x^2, W_x) &:= \mathcal{N}(x; \mu_x + W_x z, \sigma_x^2 I) \\
p(y \mid z; \mu_y, \sigma_y^2, W_y) &:= \mathcal{N}(y; \mu_y + W_y z, \sigma_y^2 I)
\end{aligned}
$$

- like two PCAs, coupled by shared latent features z:
  - one for the predictors x.
  - one for the targets y.
- latent features act as **information bottleneck**.
- also known as **Latent Factor Regression** or **Bayesian Factor Regression**.

Supervised PCA: Discriminative Likelihood

A simple likelihood would put the same weight on

- reconstructing the predictors and
- reconstructing the targets.

A weight $\alpha \in \mathbb{R}_0^+$ for the reconstruction error of the predictors should be introduced (**discriminative likelihood**):

$$L_\alpha(\Theta; x, y, z) := \prod_{i=1}^{n} p(y_i \mid z_i; \Theta) p(x_i \mid z_i; \Theta)^\alpha p(z_i; \Theta)$$

α can be treated as hyperparameter and found by grid search.

Supervised PCA: EM

- The M-steps for μ_x, σ_x^2, W_x and μ_y, σ_y^2, W_y are exactly as before.
- the coupled E-step is:

$$z_i = \left(\frac{1}{\sigma_y^2} W_y^T W_y + \alpha \frac{1}{\sigma_x^2} W_x^T W_x\right)^{-1}$$
$$\left(\frac{1}{\sigma_y^2} W_y^T (y_i - \mu_y) + \alpha \frac{1}{\sigma_x^2} W_x^T (x_i - \mu_x)\right)$$

Conclusion (1/3)

- Dimensionality reduction aims to find a **lower dimensional representation of data** that **preserves the information** as much as possible. — "Preserving information" means
  - to preserve **pairwise distances between objects** (**multidimensional scaling**).
  - to be able to reconstruct the original object features (**feature reconstruction**).

- The **truncated Singular Value Decomposition (SVD)** provides the best **low rank factorization** of a matrix in two factor matrices.
  - SVD is usually computed by an algebraic factorization method (such as QR decomposition).

Conclusion (2/3)

- **Principal components analysis (PCA)** finds latent object and variable features that provide the **best linear reconstruction** (in L2 error).
  - PCA is a truncated SVD of the data matrix.

- **Probabilistic PCA** (PPCA) provides a probabilistic interpretation of PCA.
  - PPCA adds a **L2 regularization** of the object features.
  - PPCA is learned by the **EM algorithm**.
  - Adding L2 regularization for the linear reconstruction/variable features on top leads to **Bayesian PCA**.
  - Generalizing to variable-specific variances leads to **Factor Analysis**.
  - For both, Bayesian PCA and Factor Analysis, EM can be adapted easily.

Conclusion (3/3)

- To capture a **nonlinear relationship** between latent features and observed features, PCA can be kernelized (**Kernel PCA**).
  - Learning a Kernel PCA is done by an eigen decomposition of the kernel matrix.
  - Kernel PCA often is found to lead to "unnatural visualizations".
  - But Kernel PCA sometimes provides better classification performance for simple classifiers on latent features (such as 1-Nearest Neighbor).

Readings

- Principal Components Analysis (PCA)
  - [HTFF05], ch. 14.5.1, [Bis06], ch. 12.1, [Mur12], ch. 12.2.
- Probabilistic PCA
  - [Bis06], ch. 12.2, [Mur12], ch. 12.2.4.
- Factor Analysis
  - [HTFF05], ch. 14.7.1, [Bis06], ch. 12.2.4.
- Kernel PCA
  - [HTFF05], ch. 14.5.4, [Bis06], ch. 12.3, [Mur12], ch. 14.4.4.

Further Readings

- (Non-negative) Matrix Factorization
  - [HTFF05], ch. 14.6
- Independent Component Analysis, Exploratory Projection Pursuit
  - [HTFF05], ch. 14.7 [Bis06], ch. 12.4 [Mur12], ch. 12.6.
- Nonlinear Dimensionality Reduction
  - [HTFF05], ch. 14.9, [Bis06], ch. 12.4

References

Christopher M. Bishop.
Pattern recognition and machine learning, volume 1.
springer New York, 2006.

Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin.
The elements of statistical learning: data mining, inference and prediction, volume 27.
2005.

Kevin P. Murphy.
Machine learning: a probabilistic perspective.
The MIT Press, 2012.