

# Business Analytics

## 3. Dimensionality Reduction

Lars Schmidt-Thieme

Information Systems and Machine Learning Lab (ISMLL)  
University of Hildesheim, Germany

# Outline

1. Principal Components Analysis
2. Probabilistic PCA & Factor Analysis
3. Non-linear Dimensionality Reduction
4. Independent Component Analysis
5. Supervised Dimensionality Reduction
6. Conclusion

# Outline

1. Principal Components Analysis
2. Probabilistic PCA & Factor Analysis
3. Non-linear Dimensionality Reduction
4. Independent Component Analysis
5. Supervised Dimensionality Reduction
6. Conclusion

# 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

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

called **distortion** where  $D(P)$  measures how bad a low dimensional representation  $P : X \rightarrow \mathbb{R}^K$  for a data set  $X \subseteq \mathcal{X}$  is, and

- ▶ a number  $K \in \mathbb{N}$  of latent dimensions,

find a low dimensional representation  $P : X \rightarrow \mathbb{R}^K$  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

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

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

$$\begin{aligned} D(P) &:= \frac{2}{|X|(|X| - 1)} \sum_{\substack{x, x' \in X \\ x \neq y}} (d_{\mathcal{X}}(x, x') - d_Z(P(x), P(x')))^2 \\ &= \frac{2}{n(n-1)} \sum_{i=1}^n \sum_{j=1}^{i-1} (d_{\mathcal{X}}(x_i, x_j) - \|z_i - z_j\|)^2, \quad z_i := P(x_i) \end{aligned}$$

## 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:

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

# Singular Value Decomposition (SVD)

## Theorem (Existence of SVD)

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

$$\begin{aligned}
 U \in \mathbb{R}^{n \times k}, V \in \mathbb{R}^{m \times k}, \Sigma := \text{diag}(\sigma_1, \dots, \sigma_k) \in \mathbb{R}^{k \times k}, & \quad k := \min\{n, m\} \\
 \sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > \sigma_{r+1} = \dots = \sigma_k = 0, & \quad r := \text{rank}(A) \\
 U, V \text{ orthonormal, i.e., } U^T U = I, V^T V = I &
 \end{aligned}$$

with

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

$\sigma_i$  are called **singular values of A**.

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

# Singular Value Decomposition (SVD; 2/2)

It holds:

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

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

b)  $\sigma_i^2$  are eigenvalues and  $U_i$  eigenvectors of  $AA^T$ :

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



# Singular Value Decomposition (SVD; 2/2)

It holds:

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

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

b)  $\sigma_i^2$  are eigenvalues and  $U_i$  eigenvectors of  $AA^T$ :

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

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

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

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

# Matrix Trace

The function

$$\text{tr} : \bigcup_{n \in \mathbb{N}} \mathbb{R}^{n \times n} \rightarrow \mathbb{R}$$

$$A \mapsto \text{tr}(A) := \sum_{i=1}^n a_{i,i}$$

is called **matrix trace**.

# Matrix Trace

The function

$$\text{tr} : \bigcup_{n \in \mathbb{N}} \mathbb{R}^{n \times n} \rightarrow \mathbb{R}$$

$$A \mapsto \text{tr}(A) := \sum_{i=1}^n a_{i,i}$$

is called **matrix trace**. It holds:

a) invariance under permutations of factors:

$$\text{tr}(AB) = \text{tr}(BA)$$

b) invariance under basis change:

$$\text{tr}(B^{-1}AB) = \text{tr}(A)$$

# Matrix Trace

The function

$$\text{tr} : \bigcup_{n \in \mathbb{N}} \mathbb{R}^{n \times n} \rightarrow \mathbb{R}$$

$$A \mapsto \text{tr}(A) := \sum_{i=1}^n a_{i,i}$$

is called **matrix trace**. It holds:

a) invariance under permutations of factors:

$$\text{tr}(AB) = \text{tr}(BA)$$

b) invariance under basis change:

$$\text{tr}(B^{-1}AB) = \text{tr}(A)$$

proof:

$$\text{a) } \text{tr}(AB) = \sum_i \sum_j A_{i,j} B_{j,i} = \sum_i \sum_j B_{j,i} A_{i,j} = \text{tr}(BA)$$

$$\text{b) } \text{tr}(B^{-1}AB) = \text{tr}(BB^{-1}A) = \text{tr}(A)$$

# Frobenius Norm

The function  $\|\cdot\|_F : \bigcup_{n,m \in \mathbb{N}} \mathbb{R}^{n \times m} \rightarrow \mathbb{R}_0^+$

$$A \mapsto \|A\|_F := \left( \sum_{i=1}^n \sum_{j=1}^m a_{i,j}^2 \right)^{\frac{1}{2}}$$

is called **Frobenius norm**.

# Frobenius Norm

The function  $\|\cdot\|_F : \bigcup_{n,m \in \mathbb{N}} \mathbb{R}^{n \times m} \rightarrow \mathbb{R}_0^+$

$$A \mapsto \|A\|_F := \left( \sum_{i=1}^n \sum_{j=1}^m a_{i,j}^2 \right)^{\frac{1}{2}}$$

is called **Frobenius norm**. It holds:

a) trace representation:

$$\|A\|_F = (\text{tr}(A^T A))^{\frac{1}{2}}$$

b) invariance under orthonormal transformations:

$$\text{tr}(UAV^T) = \text{tr}(A), \quad U, V \text{ orthonormal}$$

## Frobenius Norm

The function  $\|\cdot\|_F : \bigcup_{n,m \in \mathbb{N}} \mathbb{R}^{n \times m} \rightarrow \mathbb{R}_0^+$

$$A \mapsto \|A\|_F := \left( \sum_{i=1}^n \sum_{j=1}^m a_{i,j}^2 \right)^{\frac{1}{2}}$$

is called **Frobenius norm**. It holds:

a) trace representation:

$$\|A\|_F = (\text{tr}(A^T A))^{\frac{1}{2}}$$

b) invariance under orthonormal transformations:

$$\text{tr}(UAV^T) = \text{tr}(A), \quad U, V \text{ orthonormal}$$

proof:

$$\text{a) } \text{tr}(A^T A) = \sum_i \sum_j A_{j,i} A_{j,i} = \|A\|_F^2$$

$$\begin{aligned} \text{b) } \|UAV\|_F^2 &= \text{tr}(VA^T U^T UAV^T) = \text{tr}(VA^T AV^T) \\ &= \text{tr}(A^T AV^T V) = \text{tr}(A^T A) = \|A\|_F^2 \end{aligned}$$



## Frobenius Norm (2/2)

c) representation as sum of squared singular values:

$$\|A\|_F = \sum_{i=1}^{\min\{m,n\}} \sigma_i^2$$

## Frobenius Norm (2/2)

c) representation as sum of squared singular values:

$$\|A\|_F = \sum_{i=1}^{\min\{m,n\}} \sigma_i^2$$

proof:

c) let  $A = U\Sigma V^T$  the SVD of  $A$

$$\|A\|_F = \|U\Sigma V^T\|_F = \|\Sigma\|_F = \text{tr}(\Sigma^T \Sigma) = \sum_{i=1}^{\min\{m,n\}} \sigma_i^2$$

# 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**.

# 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)

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

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

PCA finds

- ▶  $K$  principal components  $v_1, \dots, v_K \in \mathbb{R}^m$  and
  - ▶ latent weights  $z_i \in \mathbb{R}^K$  for each data point  $i \in \{1, \dots, n\}$ ,
- such that the linear combination of the principal components

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

reconstructs the original features  $x_i$  as good as possible:

# Principal Components Analysis (PCA)

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

PCA finds

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

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

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

# Principal Components Analysis (PCA)

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

PCA finds

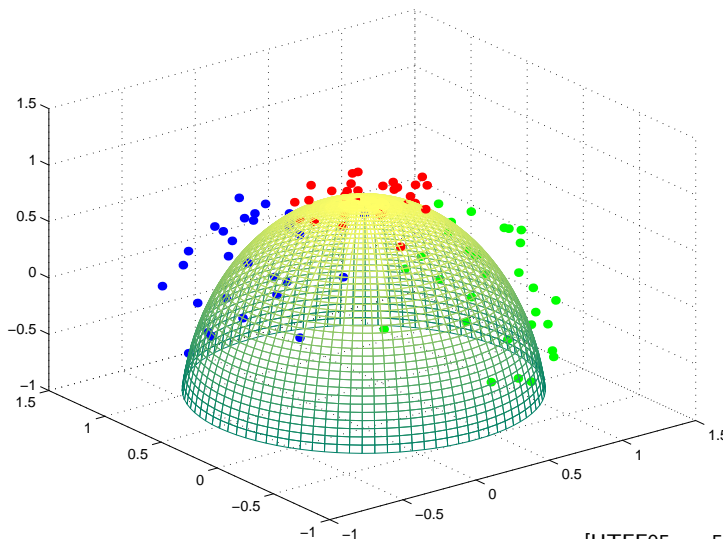
- ▶  $K$  principal components  $v_1, \dots, v_K \in \mathbb{R}^m$  and
- ▶ latent weights  $z_i \in \mathbb{R}^K$  for each data point  $i \in \{1, \dots, n\}$ ,

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

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

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

# Principal Components Analysis (Example 1)

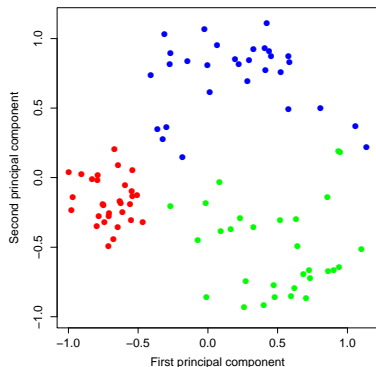
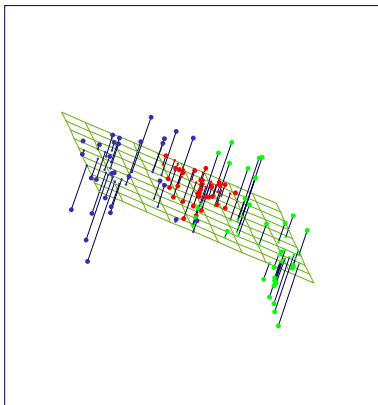


[HTFF05, p. 530]



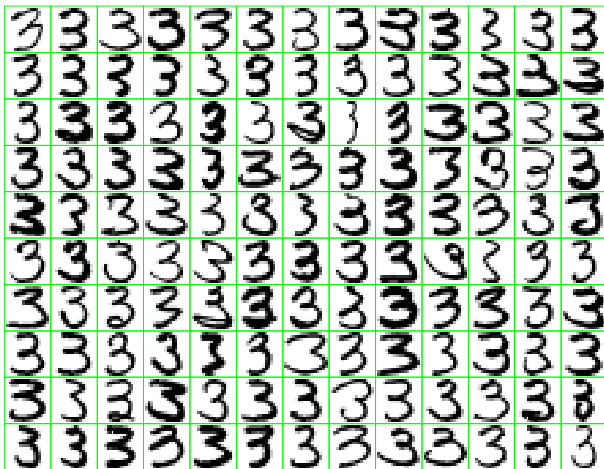


# Principal Components Analysis (Example 1)

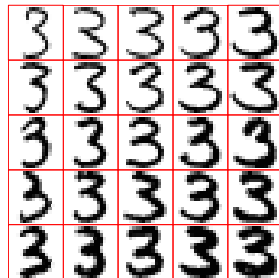
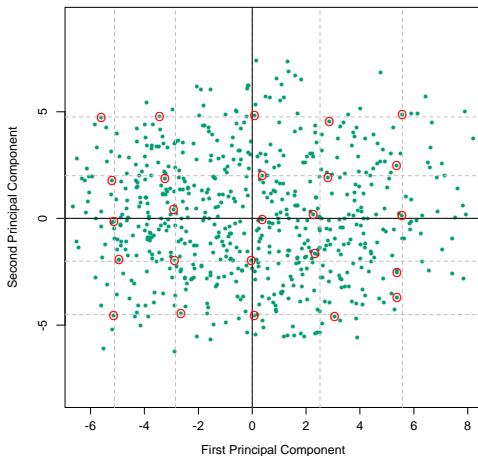


[HTFF05, p. 536]

# Principal Components Analysis (Example 2)



# Principal Components Analysis (Example 2)



# Outline

1. Principal Components Analysis
2. Probabilistic PCA & Factor Analysis
3. Non-linear Dimensionality Reduction
4. Independent Component Analysis
5. Supervised Dimensionality Reduction
6. Conclusion

# Probabilistic Model

Probabilistic PCA provides a probabilistic interpretation of PCA.

It models for each data point

- ▶ a multivariate normal distributed latent factor  $z$ ,
- ▶ that influences the observed variables linearly:

$$p(z) := \mathcal{N}(z; 0, I)$$

$$p(x | z; \mu, \sigma^2, W) := \mathcal{N}(x; \mu + Wz, \sigma^2 I)$$

# Probabilistic PCA Loglikelihood

$$\begin{aligned} \ell(X, Z; \mu, \sigma^2, W) \\ = \sum_{i=1}^n \ln p(x \mid z; \mu, \sigma^2, W) + \ln p(z) \end{aligned}$$

# Probabilistic PCA Loglikelihood

$$\begin{aligned}\ell(X, Z; \mu, \sigma^2, W) &= \sum_{i=1}^n \ln p(x_i | z_i; \mu, \sigma^2, W) + \ln p(z_i) \\ &= \sum_i \ln \mathcal{N}(x_i; \mu + Wz_i, \sigma^2 I) + \ln \mathcal{N}(z_i; 0, I)\end{aligned}$$

# Probabilistic PCA Loglikelihood

$$\begin{aligned}
 \ell(X, Z; \mu, \sigma^2, W) & \\
 &= \sum_{i=1}^n \ln p(x_i | z_i; \mu, \sigma^2, W) + \ln p(z_i) \\
 &= \sum_i \ln \mathcal{N}(x_i; \mu + Wz_i, \sigma^2 I) + \ln \mathcal{N}(z_i; 0, I) \\
 &\propto \sum_i -\frac{1}{2} \log \sigma^2 - \frac{1}{2\sigma^2} (x_i - \mu - Wz_i)^T (x_i - \mu - Wz_i) - \frac{1}{2} z_i^T z_i
 \end{aligned}$$

remember:  $\mathcal{N}(x; \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^m |\Sigma|}^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu)\Sigma^{-1}(x-\mu)}$ .



# Probabilistic PCA Loglikelihood

$$\begin{aligned}
 \ell(X, Z; \mu, \sigma^2, W) &= \sum_{i=1}^n \ln p(x_i | z_i; \mu, \sigma^2, W) + \ln p(z_i) \\
 &= \sum_i \ln \mathcal{N}(x_i; \mu + Wz_i, \sigma^2 I) + \ln \mathcal{N}(z_i; 0, I) \\
 &\propto \sum_i -\frac{1}{2} \log \sigma^2 - \frac{1}{2\sigma^2} (x_i - \mu - Wz_i)^T (x_i - \mu - Wz_i) - \frac{1}{2} z_i^T z_i \\
 &\propto - \sum_i \log \sigma^2 + \frac{1}{\sigma^2} (\mu^T \mu + z_i^T W^T W z_i - 2x_i^T \mu - 2x_i^T W z_i + 2\mu^T W z_i) + z_i^T z_i
 \end{aligned}$$

# PCA vs Probabilistic PCA

$$\begin{aligned} \ell(X, Z; \mu, \sigma^2, W) \\ \propto \sum_i -\frac{1}{2} \log \sigma^2 - \frac{1}{2\sigma^2} (x_i - \mu - Wz_i)^T (x_i - \mu - Wz_i) - \frac{1}{2} z_i^T z_i \end{aligned}$$

- ▶ as PCA: Decompose with minimal L2 loss

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

with  $v_k := W_{\cdot,k}$

- ▶ different from PCA: L2 regularized row features  $z$ .
  - ▶ cannot be solved by SVD. Use EM as learning algorithm!
- ▶ additionally also regularization of column features  $W$  possible (through a prior on  $W$ ).

## EM / Block Coordinate Descent: Outline

$$\ell(X, Z; \mu, \sigma^2, W)$$

$$\propto - \sum_i \log \sigma^2 + \frac{1}{\sigma^2} (\mu^T \mu + z_i^T W^T W z_i - 2x_i^T \mu - 2x_i^T W z_i + 2\mu^T W z_i + z_i^T z_i)$$

1. **expectation step:**  $\forall i$

$$\frac{\partial \ell}{\partial z_i} \stackrel{!}{=} 0 \quad \rightsquigarrow z_i = \dots \quad (0)$$

2. **minimization step:**

$$\frac{\partial \ell}{\partial \mu} \stackrel{!}{=} 0 \quad \rightsquigarrow \mu = \dots \quad (1)$$

$$\frac{\partial \ell}{\partial \sigma^2} \stackrel{!}{=} 0 \quad \rightsquigarrow \sigma^2 = \dots \quad (2)$$

$$\frac{\partial \ell}{\partial W} \stackrel{!}{=} 0 \quad \rightsquigarrow W = \dots \quad (3)$$

## EM / Block Coordinate Descent

$$\ell(X, Z; \mu, \sigma^2, W)$$

$$\propto - \sum_i \log \sigma^2 + \frac{1}{\sigma^2} (\mu^T \mu + z_i^T W^T W z_i - 2x_i^T \mu - 2x_i^T W z_i + 2\mu^T W z_i + z_i^T z_i)$$

$$\frac{\partial \ell}{\partial z_i} = -\frac{1}{\sigma^2} (2z_i^T W^T W - 2x_i^T W + 2\mu^T W) - 2z_i^T \stackrel{!}{=} 0$$

$$(W^T W + \sigma^2 I) z_i = W^T (x_i - \mu)$$

$$z_i = (W^T W + \sigma^2 I)^{-1} W^T (x_i - \mu) \quad (0)$$

## EM / Block Coordinate Descent

$$\ell(X, Z; \mu, \sigma^2, W)$$

$$\propto - \sum_i \log \sigma^2 + \frac{1}{\sigma^2} (\mu^T \mu + z_i^T W^T W z_i - 2x_i^T \mu - 2x_i^T W z_i + 2\mu^T W z_i + z_i^T z_i)$$

$$\frac{\partial \ell}{\partial \mu} = -\frac{1}{\sigma^2} \sum_i 2\mu^T - 2x_i^T + 2z_i^T W^T \stackrel{!}{=} 0$$

$$\mu = \frac{1}{n} \sum_i x_i - W z_i \quad (1)$$

Note: As  $\mathbb{E}(z_i) = 0$ ,  $\mu$  often is fixed to  $\mu := \frac{1}{n} \sum_i x_i$ .

## EM / Block Coordinate Descent

$$\ell(X, Z; \mu, \sigma^2, W)$$

$$\propto - \sum_i \log \sigma^2 + \frac{1}{\sigma^2} (\mu^T \mu + z_i^T W^T W z_i - 2x_i^T \mu - 2x_i^T W z_i + 2\mu^T W z_i + z_i^T z_i)$$

$$\frac{\partial \ell}{\partial \sigma^2} = -n \frac{1}{\sigma^2} + \frac{1}{(\sigma^2)^2} \sum_i \mu^T \mu + z_i^T W^T W z_i - 2x_i^T \mu - 2x_i^T W z_i + 2\mu^T W z_i + z_i^T z_i$$

$$\begin{aligned} \sigma^2 &= \frac{1}{n} \sum_i \mu^T \mu + z_i^T W^T W z_i - 2x_i^T \mu - 2x_i^T W z_i + 2\mu^T W z_i + z_i^T z_i \\ &= \frac{1}{n} \sum_i (x_i - \mu - W z_i)^T (x_i - \mu - W z_i) \end{aligned} \quad (2)$$

## EM / Block Coordinate Descent

$$\ell(X, Z; \mu, \sigma^2, W)$$

$$\propto - \sum_i \log \sigma^2 + \frac{1}{\sigma^2} (\mu^T \mu + z_i^T W^T W z_i - 2x_i^T \mu - 2x_i^T W z_i + 2\mu^T W z_i + z_i^T z_i)$$

$$\frac{\partial \ell}{\partial W} = -\frac{1}{\sigma^2} \sum_i 2W z_i z_i^T - 2x_i z_i^T + 2\mu z_i^T \stackrel{!}{=} 0$$

$$W \left( \sum_i z_i z_i^T \right) = \sum_i (x_i - \mu) z_i^T$$

$$W = \sum_i (x_i - \mu) z_i^T \left( \sum_i z_i z_i^T \right)^{-1} \quad (3)$$

# EM / Block Coordinate Descent: Summary

alternate until convergence:

1. **expectation step:**  $\forall i$

$$z_i = (W^T W + \sigma^2 I)^{-1} W^T (x_i - \mu) \quad (0)$$

2. **minimization step:**

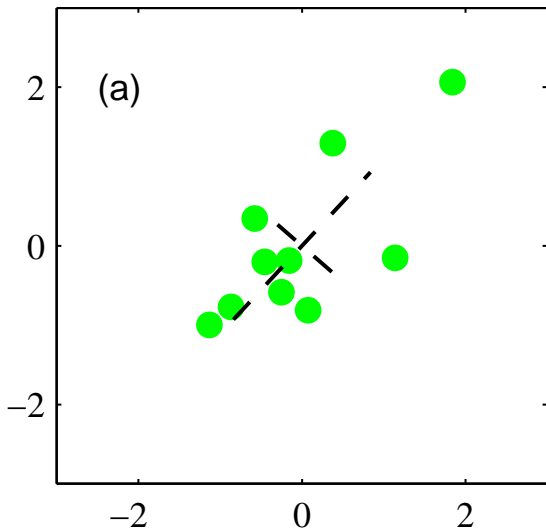
$$\mu = \frac{1}{n} \sum_i x_i - W z_i \quad (1)$$

$$\sigma^2 = \frac{1}{n} \sum_i (x_i - \mu - W z_i)^T (x_i - \mu - W z_i) \quad (2)$$

$$W = \sum_i (x_i - \mu) z_i^T \left( \sum_i z_i z_i^T \right)^{-1} \quad (3)$$

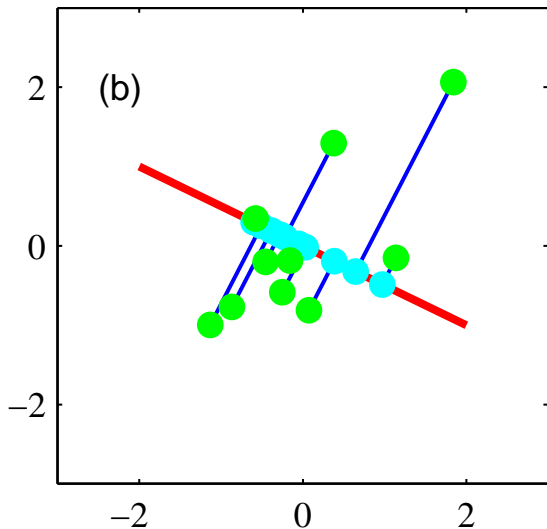


## EM / Block Coordinate Descent: Example



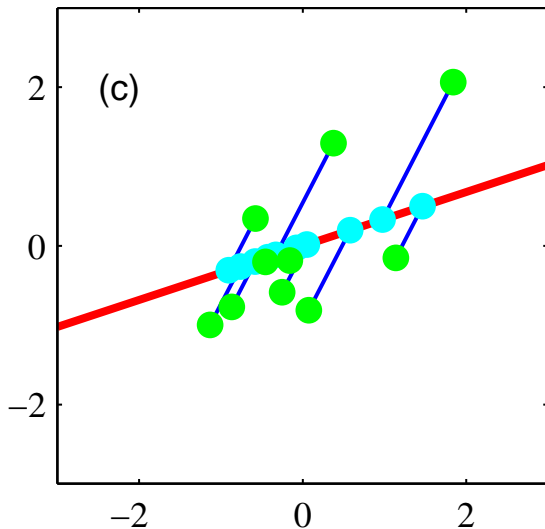
[Bis06, p. 581]

## EM / Block Coordinate Descent: Example



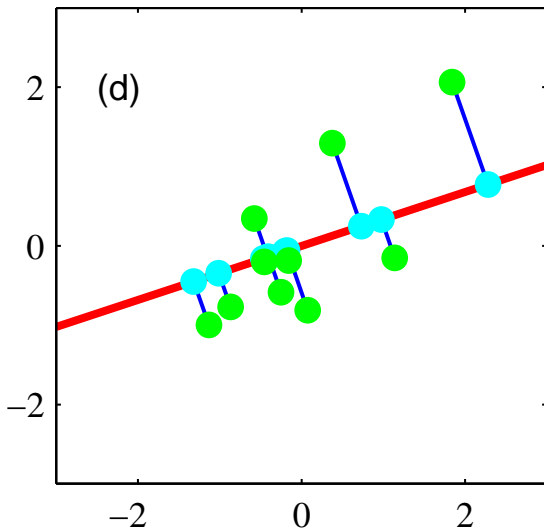
[Bis06, p. 581]

## EM / Block Coordinate Descent: Example



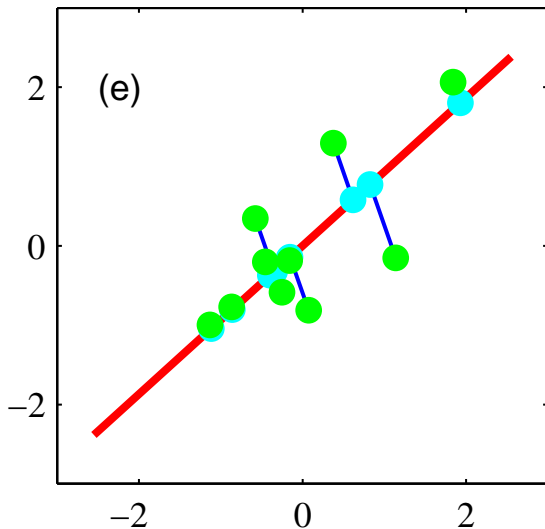
[Bis06, p. 581]

## EM / Block Coordinate Descent: Example



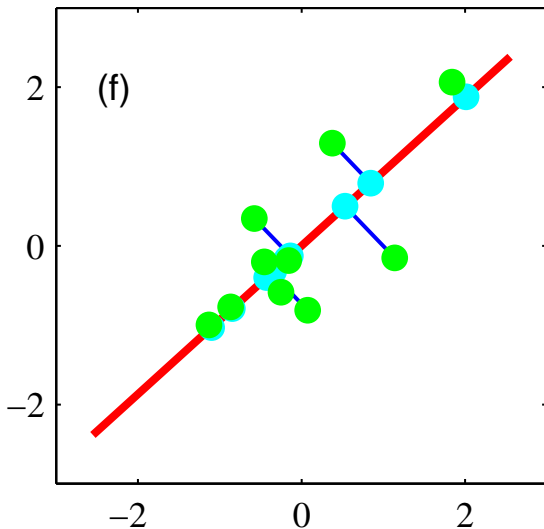
[Bis06, p. 581]

## EM / Block Coordinate Descent: Example



[Bis06, p. 581]

## EM / Block Coordinate Descent: Example



[Bis06, p. 581]

# Regularization of Column Features $W$

$$p(W) := \prod_{j=1}^m \mathcal{N}(w_j; 0, \tau_j^2 I), \quad W = (w_1, \dots, w_m)$$

# Regularization of Column Features $W$

$$p(W) := \prod_{j=1}^m \mathcal{N}(w_j; 0, \tau_j^2 I), \quad W = (w_1, \dots, w_m)$$
$$\rightsquigarrow \ell = \dots + \sum_{j=1}^m -K \log \tau_j^2 - \frac{1}{2\tau_j^2} w_j^T w_j$$



# Regularization of Column Features $W$

$$p(W) := \prod_{j=1}^m \mathcal{N}(w_j; 0, \tau_j^2 I), \quad W = (w_1, \dots, w_m)$$

$$\rightsquigarrow \ell = \dots + \sum_{j=1}^m -K \log \tau_j^2 - \frac{1}{2\tau_j^2} w_j^T w_j$$

$$\frac{\partial \ell}{\partial W} = \dots - W \operatorname{diag}\left(\frac{1}{\tau_1^2}, \dots, \frac{1}{\tau_m^2}\right)$$

$$W = \sum_i (x_i - \mu) z_i^T \left( \sum_i z_i z_i^T + \sigma^2 \operatorname{diag}\left(\frac{1}{\tau_1^2}, \dots, \frac{1}{\tau_m^2}\right) \right)^{-1} \quad (3')$$

# Regularization of Column Features $W$

$$p(W) := \prod_{j=1}^m \mathcal{N}(w_j; 0, \tau_j^2 I), \quad W = (w_1, \dots, w_m)$$

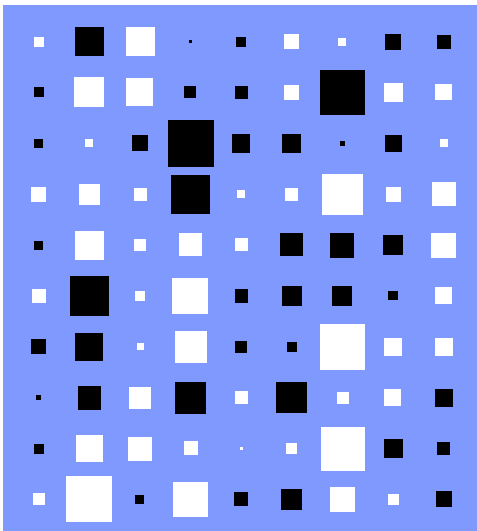
$$\rightsquigarrow \ell = \dots + \sum_{j=1}^m -K \log \tau_j^2 - \frac{1}{2\tau_j^2} w_j^T w_j$$

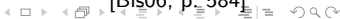
$$\frac{\partial \ell}{\partial \tau_j} = -K \frac{1}{\tau_j^2} + \frac{1}{(\tau_j^2)^2} w_j^T w_j \stackrel{!}{=} 0$$

$$\tau_j = \frac{1}{K} w_j^T w_j \tag{4}$$

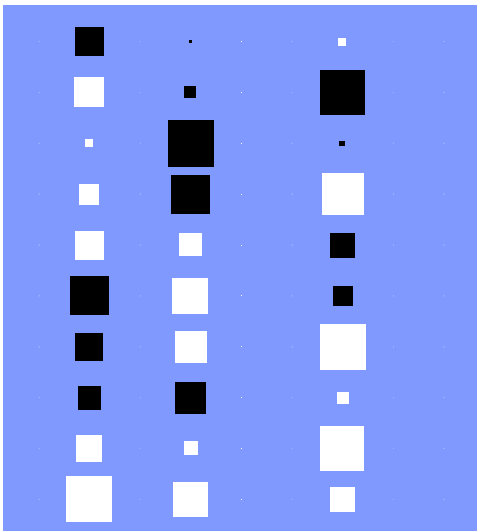
This variant of probabilistic PCA is called **Bayesian PCA**.

# Bayesian PCA: Example

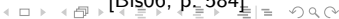


[Bis06, p. 584] 

# Bayesian PCA: Example



[Bis06, p. 584]



# Factor Analysis

$$p(z) := \mathcal{N}(z; 0, I)$$
$$p(x | z; \mu, \Sigma, W) := \mathcal{N}(x; \mu + Wz, \Sigma), \quad \Sigma \text{ diagonal}$$

# Factor Analysis

$$p(z) := \mathcal{N}(z; 0, I)$$
$$p(x | z; \mu, \Sigma, W) := \mathcal{N}(x; \mu + Wz, \Sigma), \quad \Sigma \text{ diagonal}$$

$$\ell(X, Z; \mu, \Sigma, W)$$

$$\propto \sum_i -\frac{1}{2} \log |\Sigma| - \frac{1}{2} (x_i - \mu - Wz_i)^T \Sigma^{-1} (x_i - \mu - Wz_i) - \frac{1}{2} z_i^T z_i$$

# Factor Analysis

$$p(z) := \mathcal{N}(z; 0, I)$$

$$p(x | z; \mu, \Sigma, W) := \mathcal{N}(x; \mu + Wz, \Sigma), \quad \Sigma \text{ diagonal}$$

EM:

$$z_i = (W^T \Sigma^{-1} W + I)^{-1} W^T \Sigma^{-1} (x_i - \mu) \quad (0')$$

$$\mu = \frac{1}{n} \sum_i x_i - W z_i \quad (1)$$

$$\Sigma_{j,j} = \frac{1}{n} \sum_i ((x_i - \mu)_j - W z_i)_j^2 \quad (2')$$

$$W = \sum_i (x_i - \mu) z_i^T \left( \sum_i z_i z_i^T \right)^{-1} \quad (3)$$

Note: See appendix for derivation of EM formulas.

# Outline

1. Principal Components Analysis
2. Probabilistic PCA & Factor Analysis
- 3. Non-linear Dimensionality Reduction**
4. Independent Component Analysis
5. Supervised Dimensionality Reduction
6. Conclusion



# Linear Dimensionality Reduction

Dimensionality reduction accomplishes two tasks:

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

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

# 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:

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

# 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  $\phi$ .

# Kernel Trick / Example

Example:

$$\phi : \mathbb{R} \rightarrow \mathbb{R}^{1001},$$

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

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

Naive computation:

- ▶ 2002 binomial coefficients, 3003 multiplications, 1000 additions.

Kernel computation:

- ▶ 1 multiplication, 1 addition, 1 exponentiation.

# Kernel PCA

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

$$\tilde{X} := \begin{pmatrix} \phi(x_1) \\ \phi(x_2) \\ \vdots \\ \phi(x_n) \end{pmatrix}$$

$$\tilde{X} \approx U \Sigma \tilde{V}^T$$

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.

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

$$\tilde{X}' := (\tilde{x}'_1, \dots, \tilde{x}'_n)^T = \left( I - \frac{1}{n} \mathbb{1} \right) \tilde{X}^T$$

Note:  $\mathbb{1} := (1)_{i=1, \dots, n, j=1, \dots, n}$  vector of ones,

$I := (\delta(i=j))_{i=1, \dots, n, j=1, \dots, n}$  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.

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

$$\tilde{X}' := (\tilde{x}'_1, \dots, \tilde{x}'_n)^T = \left( I - \frac{1}{n} \mathbb{1} \right) \tilde{X}^T$$

$$\begin{aligned}K' &:= \tilde{X}' \tilde{X}'^T = \left( I - \frac{1}{n} \mathbb{1} \right) \tilde{X}^T \tilde{X} \left( I - \frac{1}{n} \mathbb{1} \right) \\ &= HKH, \quad H := \left( I - \frac{1}{n} \mathbb{1} \right) \text{ centering matrix}\end{aligned}$$

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

$$\tilde{V} = \tilde{X}^T U \Sigma^{-1}$$

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

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

# Kernel PCA / Summary

Given:

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

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

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

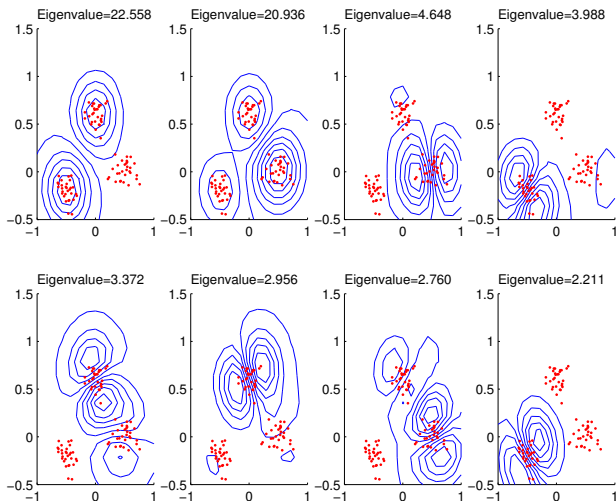
$$K' := HKH, \quad H := \left(I - \frac{1}{n}\mathbb{1}\right) \quad (1)$$

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

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

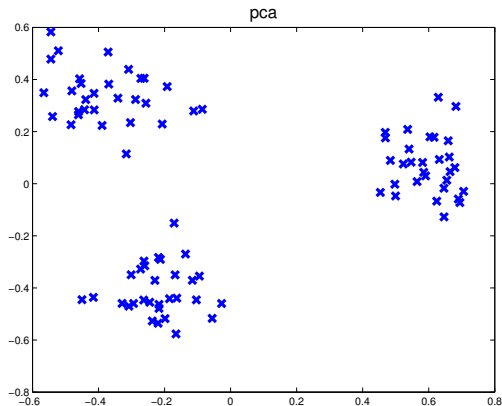
$$u := \Sigma^{-2}U^T(k(x_i, x))_{i=1, \dots, n} \quad (3)$$

# Kernel PCA: Example 1

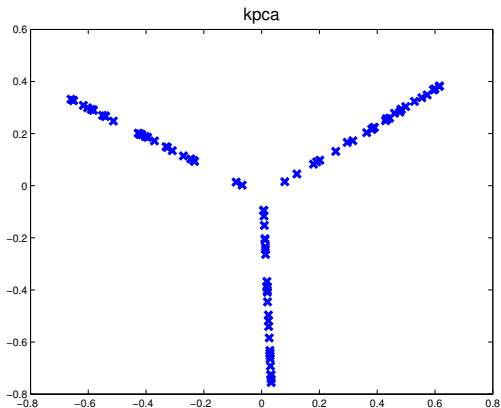


[Mur12, p. 493]

# Kernel PCA: Example 2



# Kernel PCA: Example 2



# Outline

1. Principal Components Analysis
2. Probabilistic PCA & Factor Analysis
3. Non-linear Dimensionality Reduction
- 4. Independent Component Analysis**
5. Supervised Dimensionality Reduction
6. Conclusion

# ICA Model

$$p(z) := \prod_{i=1}^K p(z_i)$$

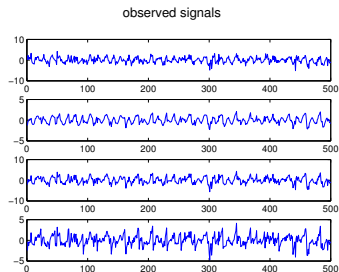
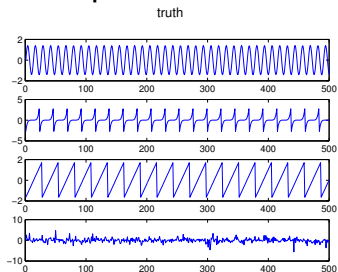
$$p(x | z; \mu, \sigma^2, W) := \mathcal{N}(x; \mu + Wz, \sigma^2 I)$$

- ▶ like PCA:
  - ▶  $x$  depends linearly on  $z$  with normal error.
  - ▶  $p(z)$  decomposes in a product of  $p(z_i)$ .
- ▶ different from PCA:
  - ▶  $p(z_i)$  are not Gaussians, but other distributions, e.g.,
    - ▶ logistic distribution:

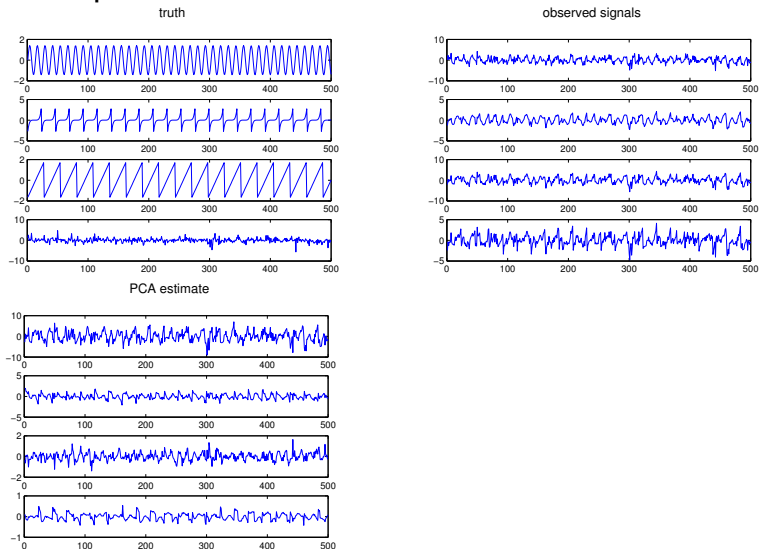
$$p(z_i) := \frac{e^{-z_i}}{(1 + e^{-z_i})^2}$$



# ICA: Example

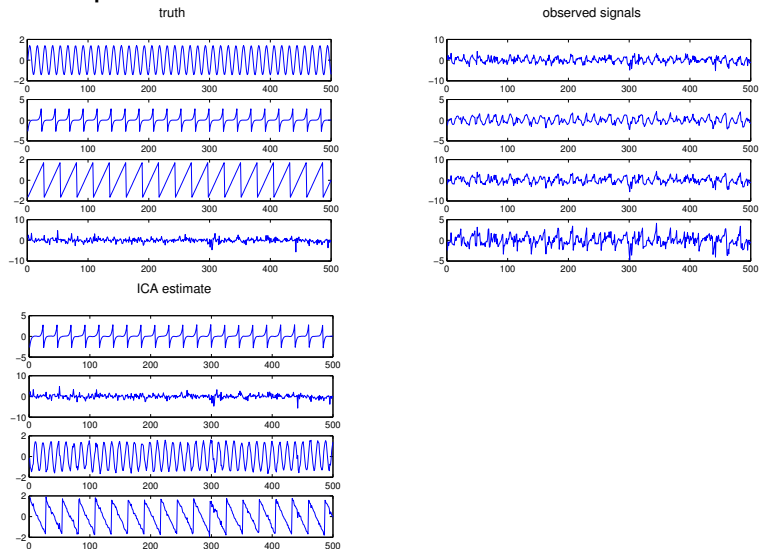


# ICA: Example



[Mur12, p. 408]

# ICA: Example



[Mur12, p. 408]

# Learning ICA

To learn ICA, different algorithms are used:

- ▶ gradient descent
  - ▶ usually slow
- ▶ EM
- ▶ approximate Newton (FastICA)

# Outline

1. Principal Components Analysis
2. Probabilistic PCA & Factor Analysis
3. Non-linear Dimensionality Reduction
4. Independent Component Analysis
- 5. Supervised Dimensionality Reduction**
6. Conclusion

# Dimensionality Reduction as Pre-Processing

Given a prediction task and

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

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  $\{x_1, \dots, x_n\} \subseteq \mathbb{R}^m$
2. learn a prediction model

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

on the latent features based on

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

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

$$p(z) := \mathcal{N}(z; 0, 1)$$

$$p(x | z; \mu_x, \sigma_x^2, W_x) := \mathcal{N}(x; \mu_x + W_x z, \sigma_x^2 I)$$

$$p(y | z; \mu_y, \sigma_y^2, W_y) := \mathcal{N}(y; \mu_y + W_y z, \sigma_y^2 I)$$

- ▶ 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**):

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

$\alpha$  can be treated as hyperparameter and found by grid search.

# Supervised PCA: EM

- ▶ The M-steps for  $\mu_x, \sigma_x^2, W_x$  and  $\mu_y, \sigma_y^2, W_y$  are exactly as before.
- ▶ the coupled E-step is:

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

# Outline

1. Principal Components Analysis
2. Probabilistic PCA & Factor Analysis
3. Non-linear Dimensionality Reduction
4. Independent Component Analysis
5. Supervised Dimensionality Reduction
- 6. Conclusion**

# 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).
- ▶ To learn models with non-normally distributed latent factors, **independent component analysis** (ICA) can be used.
  - ▶ ICA generalizes PCA to non-Gaussian distributions for the latent factors.
  - ▶ ICA can be learned by EM or approximate Newton (FastICA).

# 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

# Factor Analysis: Loglikelihood

$$\begin{aligned}\ell(X, Z; \mu, \Sigma, W) \\ &= \sum_{i=1}^n \ln p(x \mid z; \mu, \Sigma, W) + \ln p(z)\end{aligned}$$

# Factor Analysis: Loglikelihood

$$\begin{aligned}\ell(X, Z; \mu, \Sigma, W) &= \sum_{i=1}^n \ln p(x \mid z; \mu, \Sigma, W) + \ln p(z) \\ &= \sum_i \ln \mathcal{N}(x; \mu + Wz, \Sigma) + \ln \mathcal{N}(z; 0, I)\end{aligned}$$

# Factor Analysis: Loglikelihood

$$\begin{aligned}
 \ell(X, Z; \mu, \Sigma, W) &= \sum_{i=1}^n \ln p(x_i | z_i; \mu, \Sigma, W) + \ln p(z_i) \\
 &= \sum_i \ln \mathcal{N}(x_i; \mu + Wz_i, \Sigma) + \ln \mathcal{N}(z_i; 0, I) \\
 &\propto \sum_i -\frac{1}{2} \log |\Sigma| - \frac{1}{2} (x_i - \mu - Wz_i)^T \Sigma^{-1} (x_i - \mu - Wz_i) - \frac{1}{2} z_i^T z_i
 \end{aligned}$$

remember:  $\mathcal{N}(x; \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^m |\Sigma|}^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu)\Sigma^{-1}(x-\mu)}$ .

# Factor Analysis: Loglikelihood

$$\ell(X, Z; \mu, \Sigma, W)$$

$$= \sum_{i=1}^n \ln p(x_i | z_i; \mu, \Sigma, W) + \ln p(z_i)$$

$$= \sum_i \ln \mathcal{N}(x_i; \mu + Wz_i, \Sigma) + \ln \mathcal{N}(z_i; 0, I)$$

$$\propto \sum_i -\frac{1}{2} \log |\Sigma| - \frac{1}{2} (x_i - \mu - Wz_i)^T \Sigma^{-1} (x_i - \mu - Wz_i) - \frac{1}{2} z_i^T z_i$$

$$\propto - \sum_i \log |\Sigma| + (x_i^T \Sigma^{-1} x_i + \mu^T \Sigma^{-1} \mu + z_i^T W^T \Sigma^{-1} W z_i - 2x_i^T \Sigma^{-1} \mu - 2x_i^T \Sigma^{-1} W z_i + 2\mu^T \Sigma^{-1} W z_i) + z_i^T z_i$$

# Factor Analysis: EM / Block Coordinate Descent

$$\ell(X, Z; \mu, \Sigma, W)$$

$$\propto - \sum_i \log |\Sigma| + (x_i^T \Sigma^{-1} x_i + \mu^T \Sigma^{-1} \mu + z_i^T W^T \Sigma^{-1} W z_i - 2x_i^T \Sigma^{-1} \mu - 2z_i^T \Sigma^{-1} W z_i + 2\mu^T \Sigma^{-1} W z_i) + z_i^T z_i$$

$$\frac{\partial \ell}{\partial z_i} = -(2z_i^T W^T \Sigma^{-1} W - 2x_i^T W \Sigma^{-1} + 2\mu^T \Sigma^{-1} W) - 2z_i^T$$

$$(W^T \Sigma^{-1} W + I) z_i = W^T \Sigma^{-1} (x_i - \mu)$$

$$z_i = (W^T \Sigma^{-1} W + I)^{-1} W^T \Sigma^{-1} (x_i - \mu)$$

# Factor Analysis: EM / Block Coordinate Descent

$$\ell(X, Z; \mu, \Sigma, W)$$

$$\propto - \sum_i \log |\Sigma| + (x_i^T \Sigma^{-1} x_i + \mu^T \Sigma^{-1} \mu + z_i^T W^T \Sigma^{-1} W z_i - 2x_i^T \Sigma^{-1} \mu - 2x_i^T \Sigma^{-1} W z_i + 2\mu^T \Sigma^{-1} W z_i) + z_i^T z_i$$

$$\frac{\partial \ell}{\partial \mu} = - \sum_i 2\mu^T \Sigma^{-1} - 2x_i^T \Sigma^{-1} + 2z_i^T W^T \Sigma^{-1} \stackrel{!}{=} 0$$

$$\mu = \frac{1}{n} \sum_i x_i - W z_i \quad (1')$$

Note: As  $\mathbb{E}(z_i) = 0$ ,  $\mu$  often is fixed to  $\mu := \frac{1}{n} \sum_i x_i$ .

# Factor Analysis: EM / Block Coordinate Descent

$$\ell(X, Z; \mu, \Sigma, W)$$

$$\begin{aligned} \propto - \sum_i \log |\Sigma| + (x_i^T \Sigma^{-1} x_i + \mu^T \Sigma^{-1} \mu + z_i^T W^T \Sigma^{-1} W z_i - 2x_i^T \Sigma^{-1} \mu \\ - 2x_i^T \Sigma^{-1} W z_i + 2\mu^T \Sigma^{-1} W z_i) + z_i^T z_i \end{aligned}$$

$$\frac{\partial \ell}{\partial \Sigma_{j,j}} = -n \frac{1}{\Sigma_{j,j}} + \frac{1}{(\Sigma_{j,j})^2} \sum_i (x_i - \mu_i - W z_i)_j^2 \stackrel{!}{=} 0$$

$$\Sigma_{j,j} = \frac{1}{n} \sum_i ((x_i - \mu_i - W z_i)_j)^2 \quad (2')$$



# Factor Analysis: EM / Block Coordinate Descent

$$\ell(X, Z; \mu, \Sigma, W)$$




$$\propto - \sum_i \log |\Sigma| + (x_i^T \Sigma^{-1} x_i + \mu^T \Sigma^{-1} \mu + z_i^T W^T \Sigma^{-1} W z_i - 2x_i^T \Sigma^{-1} \mu - 2x_i^T \Sigma^{-1} W z_i + 2\mu^T \Sigma^{-1} W z_i) + z_i^T z_i$$

$$\frac{\partial \ell}{\partial W} = - \sum_i 2 \Sigma^{-1} W z_i z_i^T - 2 \Sigma^{-1} x_i z_i^T + 2 \Sigma^{-1} \mu z_i^T \stackrel{!}{=} 0$$

$$W \left( \sum_i z_i z_i^T \right) = \sum_i (x_i - \mu) z_i^T$$

$$W = \sum_i (x_i - \mu) z_i^T \left( \sum_i z_i z_i^T \right)^{-1} \quad (3'')$$

# 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.  
*The Mathematical Intelligencer*, 27(2):83–85, 2005.
-  Kevin P. Murphy.  
*Machine learning: a probabilistic perspective*.  
The MIT Press, 2012.