Machine Learning

C. Unsupervised Learning
C.2 Dimensionality Reduction

Lars Schmidt-Thieme

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

Syllabus

Fri. 25.10.	(1)	0. Introduction
		A. Supervised Learning: Linear Models & Fundamentals
Fri. 1.11.	(2)	A.1 Linear Regression
Fri. 8.11.	(3)	A.2 Linear Classification
Fri. 15.11.	(4)	A.3 Regularization
Fri. 22.11.	(5)	A.4 High-dimensional Data
		B. Supervised Learning: Nonlinear Models
Fri. 29.11.	(6)	B.1 Nearest-Neighbor Models
Fri. 6.12.	(7)	B.2 Neural Networks
Fri. 13.12.	(8)	B.3 Decision Trees
Fri. 20.12.	(9)	B.4 Support Vector Machines
		— *Christmas Break* —
Fri. 10.1.	(10)	B.5 A First Look at Bayesian and Markov Networks
		C. Unsupervised Learning
Fri. 17.1.	(11)	C.1 Clustering
Fri. 24.1.	(12)	C.2 Dimensionality Reduction
Fri. 31.1.	(13)	C.3 Frequent Pattern Mining
Fri. 7.2.	(14)	Q&A

Outline

Outline

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 1 / Multidimensional Scaling

Let $d_{\mathcal{X}}$ be a distance on $\mathcal{X}$ and
$d_{\mathcal{Z}}$ be a distance on the latent space $\mathcal{Z} := \mathbb{R}^K$
— usually just the Euclidean distance:

$$d_{\mathcal{Z}}(v, w) := ||v - w||_2 = (\sum_{k=1}^{K}(v_k - w_k)^2)^{\frac{1}{2}}$$

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

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

$$= \frac{1}{N(N-1)} \sum_{n=1}^{N} \sum_{\substack{m=1 \\ m \neq n}}^{N} (d_{\mathcal{X}}(x_n, x_m) - ||z_n - z_m||)^2, \quad z_n := P(x_n)$$

Distortions 2 / Reconstruction

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 well as possible:

$$D(P, r) := \frac{1}{|X|} \sum_{x \in X} d_{\mathcal{X}}(x, r(P(x)))$$

$$= \frac{1}{N} \sum_{n=1}^{N} d_{\mathcal{X}}(x_n, r(z_n)), \quad z_n := P(x_n)$$

Reconstruction / Make it simple

- allow **only linear maps for reconstructions** r:

$$r(z) := Wz, \quad W \in \mathbb{R}^{M \times K}$$

- use **squared Euclidean distance** to assess the reconstruction error:

$$d_{\mathcal{X}}(x, x') := ||x - x'||_2^2$$

$$\rightsquigarrow \quad D(P, r) = \frac{1}{N} \sum_{n=1}^{N} d_{\mathcal{X}}(x_n, r(z_n)),$$

$$\propto \sum_{n=1}^{N} ||x_n - Wz_n||_2^2$$

$$= ||X - WZ||_F$$

i.e., find the **best low rank approximation** w.r.t. the Frobenius norm.

$$||A||_F := \sum_{n=1}^{N} \sum_{m=1}^{M} A_{n,m}^2$$

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

$$||A - UV^T||_F = (\sum_{n=1}^{N} \sum_{m=1}^{M} (A_{n,m} - U_n^T V_m)^2)^{\frac{1}{2}}$$

the **L2 reconstruction error**.

Note: $||A||_F$ is called **Frobenius norm**.
(Do not confuse it with the L2 norm $|| \cdot ||_2$ for matrices.)

Singular Value Decomposition (SVD)

Theorem (Existence of SVD)

For every matrix $A \in \mathbb{R}^{N \times M}$ there exist matrices

- $U \in \mathbb{R}^{N \times K}$
- $V \in \mathbb{R}^{M \times K}$
  - *both with orthonormal columns, i.e.,* $U^T U = I, V^T V = I$
- $\Sigma \in \mathbb{R}^{K \times K}$,
  - *diagonal, i.e.* $\Sigma := diag(\sigma_1, \ldots, \sigma_K)$
  - $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_R > \sigma_{R+1} = \cdots = \sigma_K = 0$,

with $K := \min\{N, M\}, \quad R := \text{rank}(A)$
such that

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

σ_r *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_k^2 are eigenvalues and V_k eigenvectors of $A^T A$:

$$(A^T A)V_k = \sigma_k^2 V_k, \quad k = 1, \ldots, K, V = (V_1, \ldots, V_K)$$

b) σ_k^2 are eigenvalues and U_k eigenvectors of AA^T:

$$(AA^T)U_k = \sigma_k^2 U_k, \quad k = 1, \ldots, K, U = (U_1, \ldots, U_K)$$

Singular Value Decomposition (SVD; 2/2)

It holds:

a) σ_k^2 are eigenvalues and V_k eigenvectors of $A^T A$:

$$(A^T A) V_k = \sigma_k^2 V_k, \quad k = 1, \ldots, K, V = (V_1, \ldots, V_K)$$

b) σ_k^2 are eigenvalues and U_k eigenvectors of AA^T:

$$(AA^T) U_k = \sigma_k^2 U_k, \quad k = 1, \ldots, K, U = (U_1, \ldots, U_K)$$

proof:

$$\text{a) } (A^T A) V_k = V \Sigma^T U^T \, U \Sigma V^T V_k = V \Sigma^2 e_k = \sigma_k^2 V_k$$

$$\text{b) } (AA^T) U_k = U \Sigma^T V^T \, V \Sigma^T U^T U_k = U \Sigma^2 e_k = \sigma_k^2 U_k$$

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

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

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'**.

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)

$$(U^*, V^*) := \underset{U \in \mathbb{R}^{N \times K}, V \in \mathbb{R}^{M \times K}}{\arg\min} ||A - UV^T||_F^2$$

is the truncated SVD.

Note: As U, V do not have to be orthonormal, one can take $U := U'\Sigma'$, $V := V'$ for the K-truncated 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
$K \in \mathbb{N}$ called **number of latent dimensions** ($K \leq M$).

PCA finds

- K **principal components** $v_1, \ldots, v_K \in \mathbb{R}^M$ and
- **latent weights** $z_n \in \mathbb{R}^K$ for each data point $n \in \{1, \ldots, N\}$,

such that the linear combination of the principal components reconstructs the original features x_n as well as possible:

$$x_n \approx \sum_{k=1}^{K} z_{n,k} v_k$$

Principal Components Analysis (PCA)

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

PCA finds

- K **principal components** $v_1, \ldots, v_K \in \mathbb{R}^M$ and
- **latent weights** $z_n \in \mathbb{R}^K$ for each data point $n \in \{1, \ldots, N\}$,

such that the linear combination of the principal components reconstructs the original features x_n as well as possible:

$$\arg\min_{\substack{v_1,\ldots,v_K \\ z_1,\ldots,z_N}} \sum_{n=1}^{N} ||x_n - \sum_{k=1}^{K} z_{n,k} v_k||^2$$

$$= \sum_{n=1}^{N} ||x_n - V z_n||^2, \quad V := (v_1, \ldots, v_K)^T$$

Principal Components Analysis (PCA)

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

PCA finds

- K **principal components** $v_1, \ldots, v_K \in \mathbb{R}^M$ and
- **latent weights** $z_n \in \mathbb{R}^K$ for each data point $n \in \{1, \ldots, N\}$,

such that the linear combination of the principal components reconstructs the original features x_n as well as possible:

$$\arg\min_{\substack{v_1,\ldots,v_K \\ z_1,\ldots,z_N}} \sum_{n=1}^{N} ||x_n - \sum_{k=1}^{K} z_{n,k} v_k||^2$$

$$= \sum_{n=1}^{N} ||x_n - V z_n||^2, \quad V := (v_1, \ldots, v_K)^T$$

$$= ||X - Z V^T||_F^2, \quad X := (x_1, \ldots, x_N)^T, Z := (z_1, \ldots, z_N)^T$$

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

PCA Algorithm

dimred-pca$(\mathcal{D} := \{x_1, \ldots, x_N\} \subseteq \mathbb{R}^M, K \in \mathbb{N})$:

$X := (x_1, x_2, \ldots, x_N)^T$

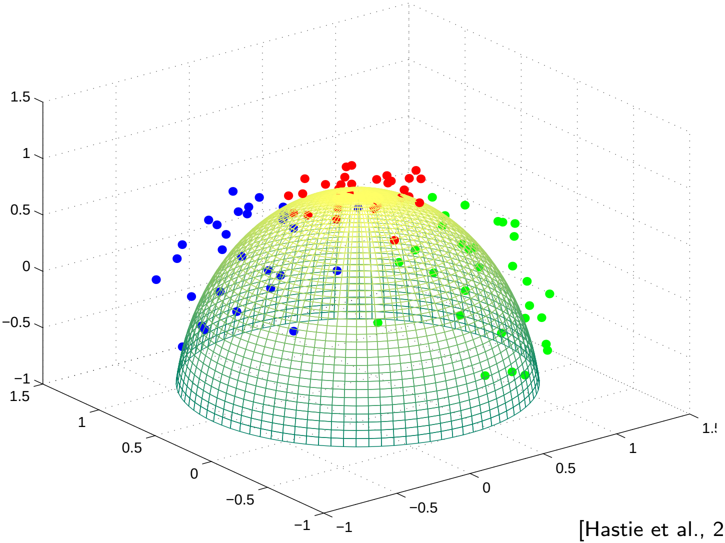
$(U, \Sigma, V) := \text{svd}(X)$

$Z := U_{.,1:K} \cdot \Sigma_{1:K,1:K}$

return $\mathcal{D}^{\text{dimred}} := \{Z_{1,.}, \ldots, Z_{N,.}\}$

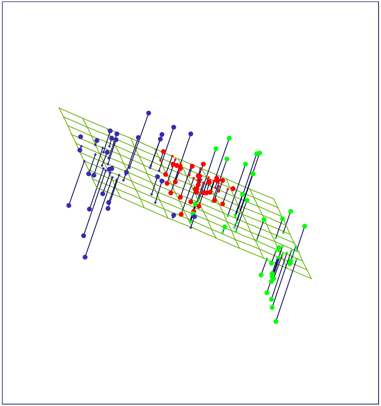
- X usually is normalized s.t. its columns have zero mean.

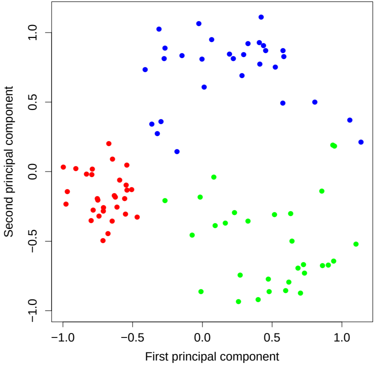
Principal Components Analysis (Example 1)



[Hastie et al., 2005, p. 53

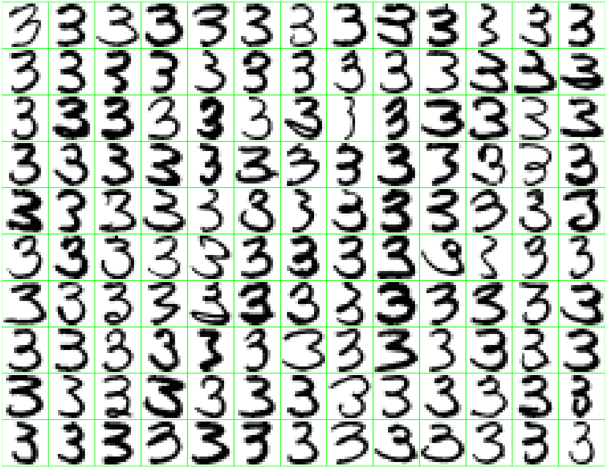
Principal Components Analysis (Example 1)





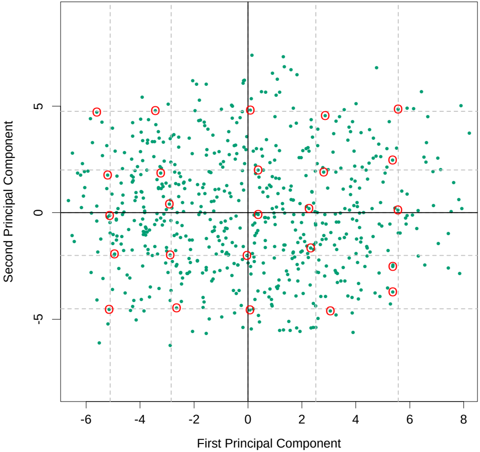
[Hastie et al., 2005, p. 53

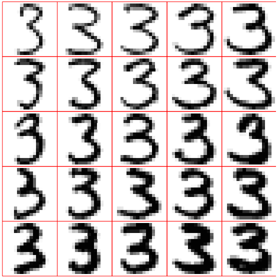
Principal Components Analysis (Example 2)



[Hastie et al., 2005, p. 53

Principal Components Analysis (Example 2)





[Hastie et al., 2005, p. 53

Outline

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:

$$p(z) := \mathcal{N}(z; 0, I)$$
$$p(x \mid z; \mu, \sigma^2, W) := \mathcal{N}(x; \mu + Wz, \sigma^2 I)$$

Probabilistic PCA Loglikelihood

$$\ell(X, Z; \mu, \sigma^2, W)$$
$$= \sum_{i=1}^{n} \ln p(x_i \mid z_i; \mu, \sigma^2, W) + \ln p(z_i)$$

Probabilistic PCA Loglikelihood

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

Probabilistic PCA Loglikelihood

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

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

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

PCA vs Probabilistic PCA

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

- as PCA: Decompose with minimal L2 loss

$$x_i \approx \mu + \sum_{k=1}^{K} z_{i,k} v_k$$
$$\text{with } v_k := W_{.,k}$$

- 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**: $\forall i$
$$\frac{\partial \ell}{\partial z_i} \stackrel{!}{=} 0 \qquad \rightsquigarrow z_i = \ldots \qquad (0)$$
2. **minimization step**:
$$\frac{\partial \ell}{\partial \mu} \stackrel{!}{=} 0 \qquad \rightsquigarrow \mu = \ldots \qquad (1)$$
$$\frac{\partial \ell}{\partial \sigma^2} \stackrel{!}{=} 0 \qquad \rightsquigarrow \sigma^2 = \ldots \qquad (2)$$
$$\frac{\partial \ell}{\partial W} \stackrel{!}{=} 0 \qquad \rightsquigarrow W = \ldots \qquad (3)$$

EM / Block Coordinate Descent

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

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

EM / Block Coordinate Descent

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

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

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

EM / Block Coordinate Descent

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

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

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

EM / Block Coordinate Descent

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

$$\frac{\partial \ell}{\partial W} = -\frac{1}{\sigma^2} \sum_i 2W z_i z_i^T - 2x_i z_i^T + 2\mu z_i^T \stackrel{!}{=} 0$$
$$W(\sum_i z_i z_i^T) = \sum_i (x_i - \mu) z_i^T$$
$$W = \sum_i (x_i - \mu) z_i^T (\sum_i z_i z_i^T)^{-1} \quad (3)$$

EM / Block Coordinate Descent: Summary

alternate until convergence:

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

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

2. **minimization step**:

$$\mu = \frac{1}{n} \sum_i x_i - W z_i \tag{1}$$

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

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

Matrix Notation

$$\hat{X} := \mathbb{1}\mu + WZ^T$$

$$\log \ell(W, Z, \mu, \sigma^2; X) \propto -\frac{1}{2} n \log \sigma^2 - \frac{1}{2\sigma^2} \operatorname{tr}(X - \mathbb{1}\mu - WZ^T)(X - \mathbb{1}\mu - WZ^T)^T - \frac{1}{2} ZZ^T$$

1. **expectation step**:

$$Z^T = (W^T W + \sigma^2 I)^{-1} W^T (X - \mathbb{1}\mu) \quad (0)$$

2. **minimization step**:

$$\mu^T = \frac{1}{n} \mathbb{1}^T (X - WZ^T) \quad (1)$$

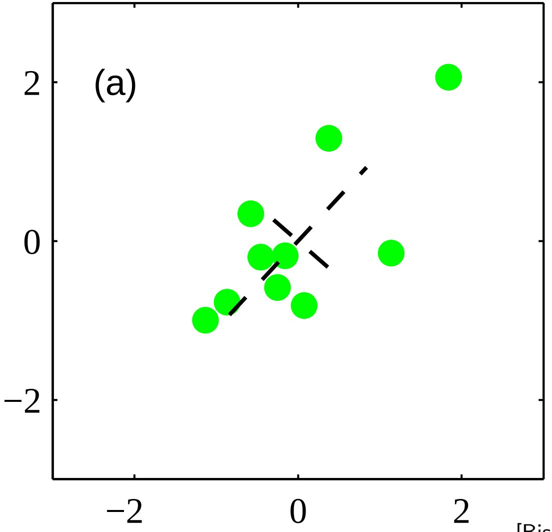
$$\sigma^2 = \frac{1}{n} \operatorname{tr}(X - \mathbb{1}\mu - WZ^T)(X - \mathbb{1}\mu - WZ^T)^T \quad (2)$$

$$W^T = (Z^T Z)^{-1} Z^T (X - \mathbb{1}\mu) \quad (3)$$

Probabilistic PCA Algorithm (EM)

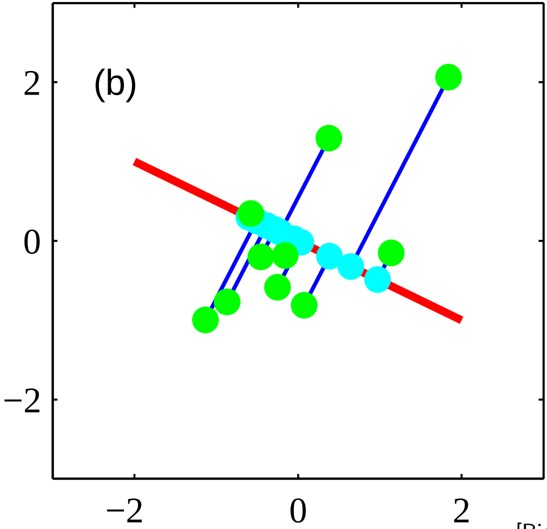
dimred-ppca$(\mathcal{D} := \{x_1, \ldots, x_N\} \subseteq \mathbb{R}^M, K \in \mathbb{N}, \epsilon \in \mathbb{R}^+)$:
  allocate $z_1, \ldots, z_N := 0 \in \mathbb{R}^K, \mu := 0 \in \mathbb{R}^M, W := 0 \in \mathbb{R}^{N \times K}, \sigma^2 := 1 \in \mathbb{R}$
  repeat
    $\sigma^2_{\text{old}} := \sigma^2, z^{\text{old}} := z$
    for $n := 1, \ldots, N$:
      $z_n := (W^T W + \sigma^2 I)^{-1} W^T (x_n - \mu)$
    $\mu_{\text{old}} := \mu$
    $\mu := \frac{1}{N} \sum_n x_n - W z_n$
    $\sigma^2 := \frac{1}{N} \sum_n (x_n - \mu_{\text{old}} - W z_n)^T (x_n - \mu_{\text{old}} - W z_n)$
    $W := \sum_n (x_n - \mu_{\text{old}}) z_n^T (\sum_n z_n z_n^T)^{-1}$
  until $\frac{1}{N} \sum_{n=1}^N ||z_n - z_n^{\text{old}}|| < \epsilon$
  return $\mathcal{D}^{\text{dimred}} := \{z_1, \ldots, z_N\}$

EM / Block Coordinate Descent: Example



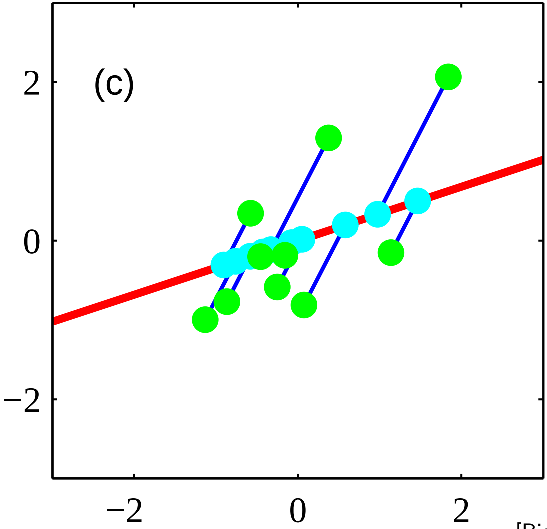
[Bishop, 2006, p. 581]

EM / Block Coordinate Descent: Example



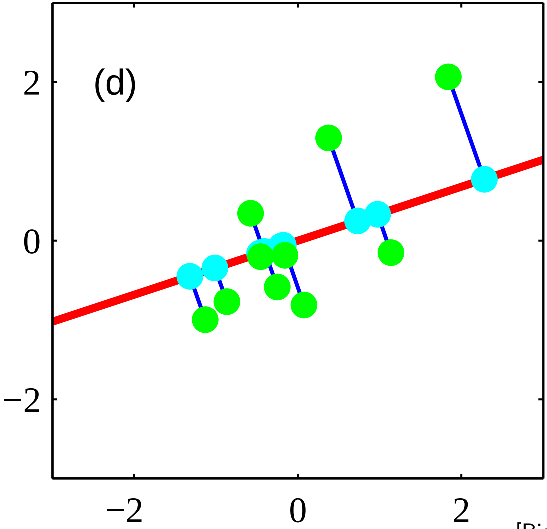
[Bishop, 2006, p. 581]

EM / Block Coordinate Descent: Example



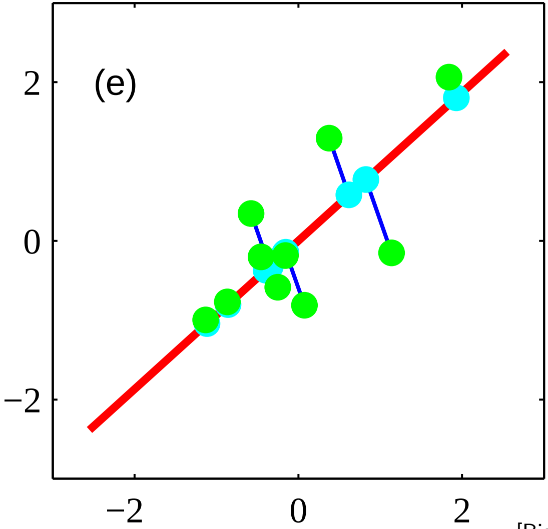
[Bishop, 2006, p. 581]

EM / Block Coordinate Descent: Example



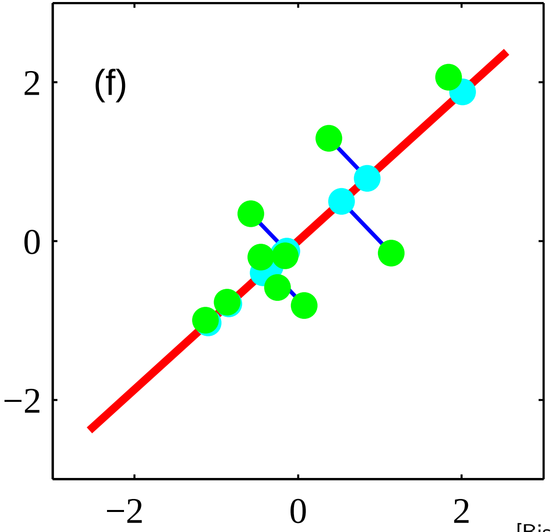
[Bishop, 2006, p. 581]

EM / Block Coordinate Descent: Example



[Bishop, 2006, p. 581]

EM / Block Coordinate Descent: Example



[Bishop, 2006, p. 581]

Regularization of Column Features W

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

Regularization of Column Features W

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

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

Regularization of Column Features W

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

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

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

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

Regularization of Column Features W

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

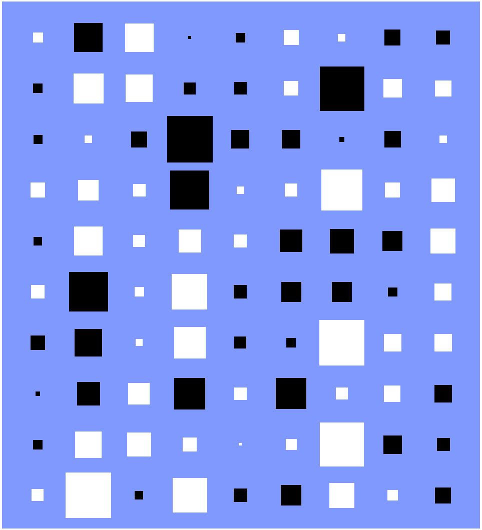
$$\rightsquigarrow \ell = \ldots + \sum_{j=1}^{m} -K \log \tau_j^2 - \frac{1}{2\tau_j^2} w_j^T w_j$$

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

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

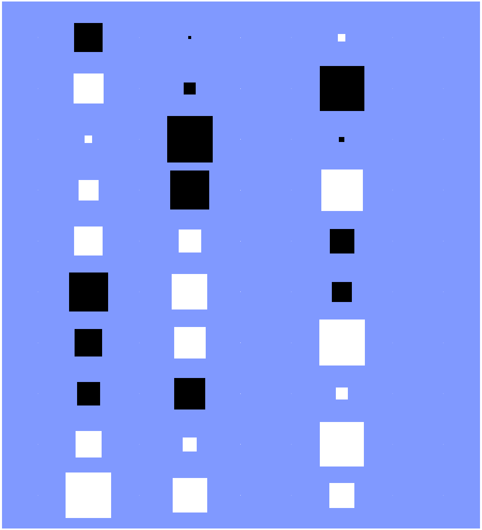
This variant of probabilistic PCA is called **Bayesian PCA**.

Bayesian PCA: Example



[Bishop, 2006, p. 584]

Bayesian PCA: Example



[Bishop, 2006, p. 584]

Factor Analysis

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

Factor Analysis

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

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

Factor Analysis

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

EM:

$$z_i = (W^T \Sigma^{-1} W + I)^{-1} W^T \Sigma^{-1} (x_i - \mu) \qquad (0')$$
$$\mu = \frac{1}{n} \sum_i x_i - W z_i \qquad (1)$$
$$\Sigma_{j,j} = \frac{1}{n} \sum_i ((x_i - \mu_i - W z_i)_j)^2 \qquad (2')$$
$$W = \sum_i (x_i - \mu) z_i^T (\sum_i z_i z_i^T)^{-1} \qquad (3)$$

Note: See appendix for derivation of EM formulas.

Outline

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:

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

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

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

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:

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

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

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

Naive computation:

- 2002 binomial coefficients, 3003 multiplications, 1000 additions.

Kernel computation:

- 1 multiplication, 1 addition, 1 exponentiation.

Kernel PCA

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

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.

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

Note: $\mathbb{1} := (1)_{i=1,\ldots,n,j=1,\ldots,n}$ matrix 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.

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

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 $k : \mathbb{R}^m \times \mathbb{R}^m \to R$.

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

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

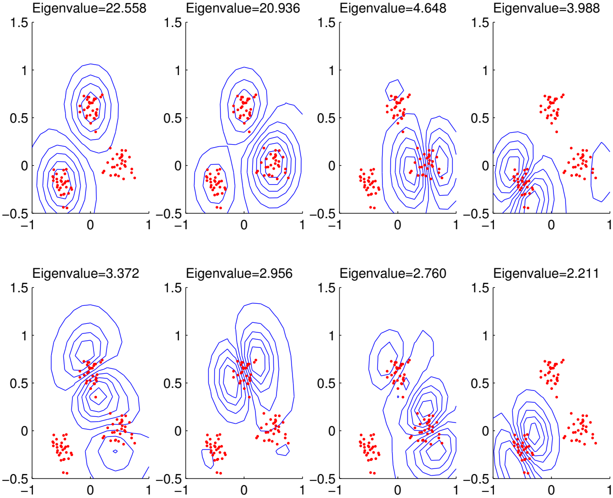
$$K' := HKH, \quad H := (I - \frac{1}{n}\mathbb{1}) \quad (1)$$

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

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

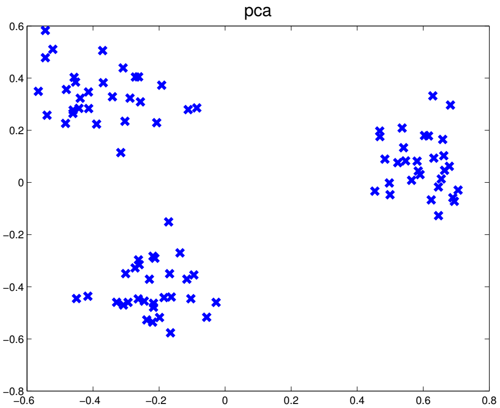
$$u := \Sigma^{-2} U^T (k(x_i, x))_{i=1,\ldots,n} \quad (3)$$

Kernel PCA: Example 1



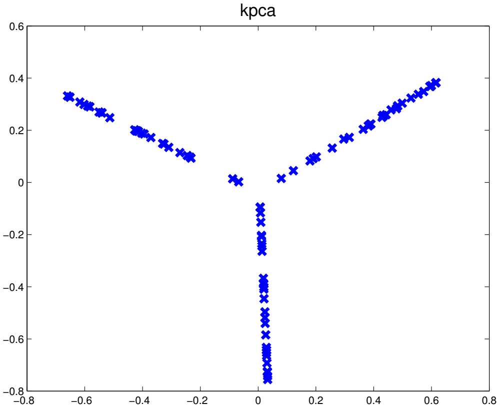
[Murphy, 2012, p. 493]

Kernel PCA: Example 2



[Murphy, 2012, p. 495]

Kernel PCA: Example 2



[Murphy, 2012, p. 495]

Outline

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

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

   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 any prediction method as component methods.
- usually fast

Dimensionality Reduction as Pre-Processing

Advantages:

- simple procedure
- generic procedure
  - works with any dimensionality reduction method and any 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

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

- 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 (1/4)

- 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 (2/4)

- **Principal components analysis (PCA)** finds latent object features and latent 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 (3/4)

- 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).

Conclusion (4/4)

- To learn a latent representation that is **useful for a given supervised task**, either
  - a two-stage approach can be taken (**PCA regression**):
    1. to learn a PCA (unsupervised) and
    2. to learn a supervised model based on the PCA features.
    - treating the PCA dimensionality K as hyperparameter, or
  - the PCA and the regression model can be combined into one model learned jointly (**supervised PCA**)
    - yields features optimized for the supervised task at hand.

Readings

- Principal Components Analysis (PCA)
  - Hastie et al. [2005], ch. 14.5.1, Bishop [2006], ch. 12.1, Murphy [2012], ch. 12.2.
- Probabilistic PCA
  - Bishop [2006], ch. 12.2, Murphy [2012], ch. 12.2.4.
- Factor Analysis
  - Hastie et al. [2005], ch. 14.7.1, Bishop [2006], ch. 12.2.4.
- Kernel PCA
  - Hastie et al. [2005], ch. 14.5.4, Bishop [2006], ch. 12.3, Murphy [2012], ch. 14.4.4.

Further Readings

- (Non-negative) Matrix Factorization
  - Hastie et al. [2005], ch. 14.6
- Independent Component Analysis, Exploratory Projection Pursuit
  - Hastie et al. [2005], ch. 14.7 Bishop [2006], ch. 12.4 Murphy [2012], ch. 12.6.
- Nonlinear Dimensionality Reduction
  - Hastie et al. [2005], ch. 14.9, Bishop [2006], ch. 12.4
- Very influential paper about visualization:
  - van der Maaten and Hinton [2008]

Factor Analysis: Loglikelihood

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

Factor Analysis: Loglikelihood

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

Factor Analysis: Loglikelihood

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

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

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

Factor Analysis: EM / Block Coordinate Descent

$$\ell(X, Z; \mu, \Sigma, W)$$
$$\propto -\sum_i \log|\Sigma| + (x_i^T\Sigma^{-1}x_i + \mu^T\Sigma^{-1}\mu + z_i^T W^T\Sigma^{-1}Wz_i - 2x_i^T\Sigma^{-1}\mu$$
$$- 2x_i^T\Sigma^{-1}Wz_i + 2\mu^T\Sigma^{-1}Wz_i) + z_i^T z_i$$

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

Factor Analysis: EM / Block Coordinate Descent

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

$$\frac{\partial \ell}{\partial \mu} = -\sum_i 2\mu^T \Sigma^{-1} - 2 x_i^T \Sigma^{-1} + 2 z_i^T W^T \Sigma^{-1} \stackrel{!}{=} 0$$
$$\mu = \frac{1}{n} \sum_i x_i - W z_i \qquad (1')$$

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

Factor Analysis: EM / Block Coordinate Descent

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

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

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

Factor Analysis: EM / Block Coordinate Descent

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

$$\frac{\partial \ell}{\partial W} = -\sum_i 2\Sigma^{-1} W z_i z_i^T - 2\Sigma^{-1} x_i z_i^T + 2\Sigma^{-1} \mu z_i^T \stackrel{!}{=} 0$$
$$W(\sum_i z_i z_i^T) = \sum_i (x_i - \mu) z_i^T$$
$$W = \sum_i (x_i - \mu) z_i^T (\sum_i z_i z_i^T)^{-1} \qquad (3'')$$

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*, volume 27. 2005.

Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of machine learning research*, 9(Nov): 2579–2605, 2008.

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

$$\mathrm{tr} : \bigcup_{n \in \mathbb{N}} \mathbb{R}^{n \times n} \to \mathbb{R}$$

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

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

a) invariance under permutations of factors:

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

b) invariance under basis change:

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

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:

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

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

Frobenius Norm

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

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

is called **Frobenius norm**.

Frobenius Norm

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

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

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

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

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

a) trace representation:

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

b) invariance under orthonormal transformations:

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

proof:

$$\text{a)}\ \mathrm{tr}(A^T A) = \sum_i \sum_j A_{j,i} A_{j,i} = ||A||_2^2$$

$$\text{b)}\ ||UAV||_F^2 = \mathrm{tr}(VA^T U^T UAV^T) = \mathrm{tr}(VA^T AV^T)$$

$$= \mathrm{tr}(A^T AV^T V) = \mathrm{tr}(A^T A) = ||A||_F^2$$

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

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