Feature Reduction and Metric Learning

Idea: Instead of deleting features, try to find a low dimensional feature space generating the original space as accurate as possible:

- Redundant features are summarized
- Irrelevant features are weighted by small values

Methods being discussed in the course:

- Reference point embedding
- Principal component analysis (PCA)
- Singular value decomposition(SVD)
- Fischer-Faces (FF) and Relevant Component Analysis(RCA)
- Large Margin Nearest Neighbor (LMNN)

Reference Point Embedding

Idea: Describe the position of each object by their distances to a set of reference points.

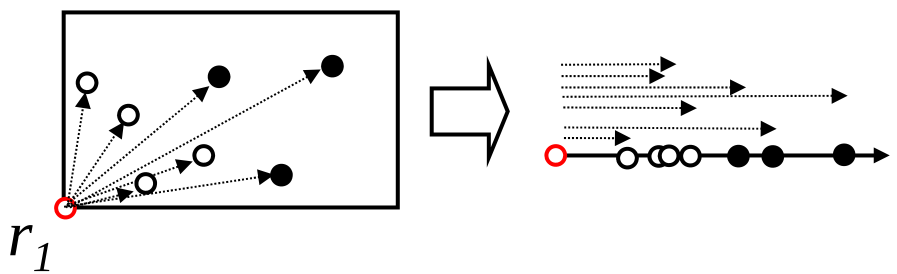
Given: Vector space F =$D_1 \times .. \times D_n$ where D =$\{D_1,..,D_n\}$.

Target: A *k*-dimensional space *R* which yields optimal solutions to given data mining task.

Method: For each reference point $R = \{r_1,..,r_k\}$ and a distance measure $d(\bullet,\bullet)$:

Transform vector x ∈ F:

$$r_R(x) = \begin{pmatrix} d(r_1, x) \\ \vdots \\ d(r_k, x) \end{pmatrix}$$



Reference Point Embedding

- Distance measure is usually determined by the application.
- Selection of reference points:
  - use centroids of the classes or cluster-centroids
  - using points on the margin of the data space

Advantages :

- Simple approach which is easy to implement
- The transformed vectors yields lower and upper bounds of the exact distances

Disadvantages:

- Even using d reference points does not reproduce a d-dimensional feature space
- Selecting good reference points is relevant and difficult

Preliminaries:

- Inner product of vectors x,y:

$$x \cdot y^T = \begin{pmatrix} x_1 & \cdots & x_d \end{pmatrix} \cdot \begin{pmatrix} y_1 \\ \vdots \\ y_d \end{pmatrix} = \langle x, y \rangle = \sum_{i=1}^{d} x_i \cdot y_i$$

- Outer product of vectors x,y:

$$x^T \cdot y = \begin{pmatrix} x_1 \\ \vdots \\ x_d \end{pmatrix} \cdot \begin{pmatrix} y_1 & \cdots & y_d \end{pmatrix} = \begin{pmatrix} x_1 y_1 & \cdots & x_1 y_d \\ \vdots & \ddots & \vdots \\ x_d y_1 & \cdots & x_d y_d \end{pmatrix}$$

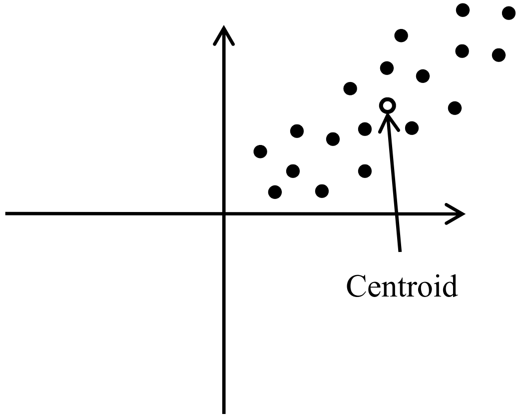
- Matrix product:

$$\begin{pmatrix} \vec{x}_1 \\ \vdots \\ \vec{x}_n \end{pmatrix} \cdot \begin{pmatrix} \vec{y}_1 & \cdots & \vec{y}_m \end{pmatrix} = \begin{pmatrix} \langle \vec{x}_1, \vec{y}_1 \rangle & \cdots & \langle \vec{x}_1, \vec{y}_m \rangle \\ \vdots & \ddots & \vdots \\ \langle \vec{x}_n, \vec{y}_1 \rangle & \cdots & \langle \vec{x}_n, \vec{y}_m \rangle \end{pmatrix}$$

Data Matrix

- Given n Vectors $v_i \in IR^d$, $n \times d$ matrix

$$D = \begin{pmatrix} v_1 \\ \vdots \\ v_d \end{pmatrix} = \begin{pmatrix} v_{1,1} & \cdots & v_{1,d} \\ \vdots & \ddots & \vdots \\ v_{n,1} & \cdots & v_{n,d} \end{pmatrix}$$ is called data matrix

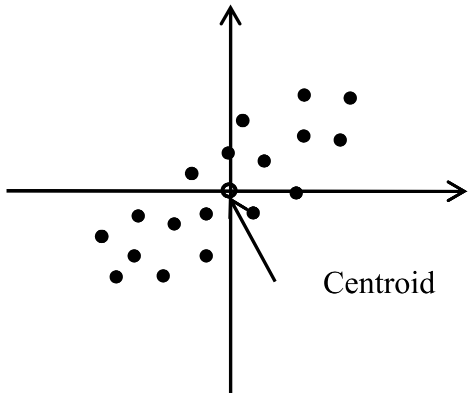


- Centroid/mean vector of D:

$$\vec{\mu} = \frac{1}{n} \cdot \sum_{i=1}^{n} v_i$$

- Centered data matrix:

$$D_{cent} = \begin{pmatrix} v_1 - \vec{\mu} \\ \vdots \\ v_d - \vec{\mu} \end{pmatrix}$$



Mahalanobis Distance

- Quadratic forms or Mahalanobis distance:

$$d_A(x,y) = \left((x-y)A(x-y)^T\right)^{\frac{1}{2}} = \sqrt{(x-y)\begin{pmatrix} A_{1,1} & \cdots & A_{1,d} \\ \vdots & \ddots & \vdots \\ A_{d,1} & \cdots & A_{d,d} \end{pmatrix}(x-y)^T} = \sqrt{\sum_{i=1}^{d}\sum_{j=1}^{d}(x_i-y_i)A_{i,j}(x_j-y_j)}$$

  Remark: If A symmetric and positive definite then d_M is a metric.

- Weighted Euclidian Distance: A is a diagonal matrix with $A_i > 0$:

$$d_A(x,y) = \sqrt{(x-y)\begin{pmatrix} A_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & A_d \end{pmatrix}(x-y)^T} = \sqrt{\sum_{i=1}^{d} A_i(x_i-y_i)^2}$$

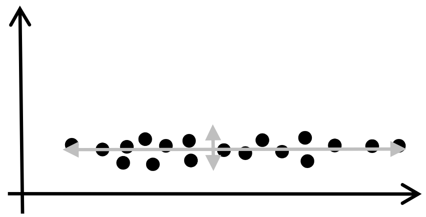
- **Connection to basis transformation** :

  If there is a symmetric decomposition $A = B \cdot B^T$ then the Mahalanobis distance is equivalent to the Euclidian distance under basis transformation B:

$$d_M(x,y) = \left((x-y)B \cdot B^T(x-y)^T\right)^{\frac{1}{2}} = \left((xB-yB)\cdot(xB-yB)^T\right)^{\frac{1}{2}} = d_{eucl}(xB, yB)$$

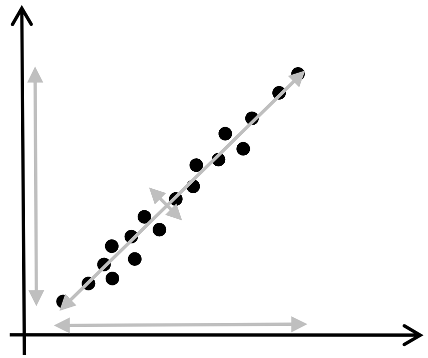
Variance Analysis

- Which attributes are the most important to the distance ?
  => attributes with strongly varying value differences $|x_i - y_i|$
  => distance to the mean value is large $|x_i - \mu_i|$
  => Variance is large:: $\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_i)^2$



Idea: Variance Analysis (= unsupervised feature selection)

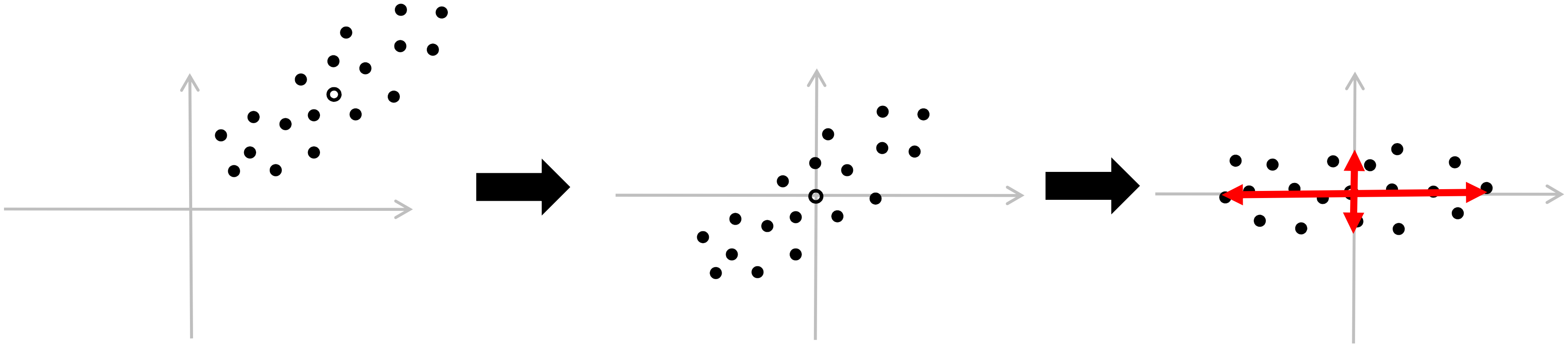
- Attributes with large variance allow strong distinction between objects
- Attributes with small variance: difference between objects are negligible
- Method:
  - Determine the variance between the values in each dimension
  - Sort all features w.r.t. to the variance
  - Select k features having the strongest variance



Beware: Even linear correlation can distribute one strong feature over arbitrarily many other dimension!!!

Principal Component Analysis (PCA)

Idea: Rotate the data space in a way that the principal components are placed along the main axis of the data space
=> Variance analysis based on principal components



- Rotate the data space in a way that the direction with the largest variance is places on an axis of the data space
- Rotation is equivalent to a basis transformation by an orthonormal basis
  - Mapping is equal of angle and preserves distances:

$$x \cdot B = x\left(b_{*,1},\ldots,b_{*,d}\right) = \left(\langle x, b_{*,1}\rangle,\ldots,\langle x, b_{*,d}\rangle\right) \quad mit \quad \underset{i \neq j}{\forall} \langle b_i, b_j \rangle = 0 \wedge \underset{1 \leq i \leq d}{\forall} \|b_i\| = 1$$

- B is built from the largest variant direction which is orthogonal to all previously selected vectors in B.

- Covariance matrix:
  - Describes the variance of all features and the pairwise correlations between them

$$VAR(X) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2 \qquad COV(X,Y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_x)(y_i - \mu_y)$$

  - Covariance matrix Σ_D *($d \times d$)* for the n$\times$*d data matrix D*:

$$\Sigma_D = \begin{pmatrix} VAR(X_1) & \cdots & COV(X_1, X_d) \\ \vdots & \ddots & \vdots \\ COV(X_d, X_1) & \cdots & VAR(X_d) \end{pmatrix} = \frac{1}{n} D_{cent}^T D_{cent}$$

- Eigenvalue λ_i and eigenvector v_i of matrix $d \times d$ D: $D \cdot v_i = \lambda_i \cdot v_i$
- Eigenvalue decomposition $M = V \Lambda V^T$

$$V = (v_1, \cdots, v_d) \;\; mit \underset{i \neq j}{\forall} \langle v_i, v_j \rangle = 0 \;\; und \overset{d}{\underset{i=1}{\forall}} \|v_i\| = 1$$

$$\Lambda = \begin{pmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_d \end{pmatrix}$$

Eigen value decomposition and Covariance matrix

- Applying the eigenvalue decomposition to the covariance matrix:

$$\Sigma_D = V \Lambda V^T = (v_1, \cdots, v_d) \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \lambda_d \end{pmatrix} \begin{pmatrix} v_1 \\ \vdots \\ v_d \end{pmatrix}$$

- v_i: Orthogonal principal components (eigenvectors)
- λ_i: Variance along each direction (eigenvalues)

Beware: λ_i=0 means that the corresponding direction is a linear combination of other principal components.

=> Depending on the algorithm completely redundant dimension cause problems

Workaround: Add a diagonal matrix with very small values δ_i to Σ_D.

Feature reduction using PCA

1. Compute the covariance matrix Σ
2. Compute the eigenvalues and the corresponding eigenvectors of Σ
3. Select the k biggest Eigenvalues and their eigenvectors (v‘)
4. The *k* selected eigenvectors represent an orthogonal basis
5. Transform the $n \times d$ data matrix D with the $d \times k$ basis V‘:

$$D \cdot V' = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} (v'_1, \cdots, v'_k) = \begin{pmatrix} \langle x_1, v'_1 \rangle & \cdots & \langle x_1, v'_k \rangle \\ \vdots & \ddots & \vdots \\ \langle x_n, v'_1 \rangle & \cdots & \langle x_n, v'_k \rangle \end{pmatrix} = D$$

Singular Value Decomposition (SVD)

Generalization of the eigenvalue decomposition

Decomposition of an $n \times d$ matrix into 2 orthogonal matrixes O,A and 1 diagonal matrix S containing the singular values.

$$D = OSA^T$$

$$= \begin{bmatrix} x_{1,1} & \cdots & x_{1,d} \\ \vdots & \ddots & \vdots \\ x_{n,1} & \cdots & x_{n,d} \end{bmatrix} = \begin{bmatrix} o_{1,1} & \cdots & o_{1,k} \\ \vdots & \ddots & \vdots \\ o_{n,1} & \cdots & t_{n,k} \end{bmatrix} \cdot \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_k \end{bmatrix} \cdot \begin{bmatrix} a_{1,1} & \cdots & a_{1,d} \\ \vdots & \ddots & \vdots \\ a_{k,1} & \cdots & a_{k,d} \end{bmatrix}$$

O : n×k left singular vectors, orthogonal column matrix
S : k×k *diagonal matrix* containing singular values
A : k×d right singular vectors, orthogonal column matrix
k : Rank of D (max. Amount of independent rows/columns)

Decomposition based on numerical algorithms.

Connection between SVD and PCA

Apply SVD to the covariance data:

$$D_{cent} = OSA^T$$

$$\Sigma_D = \frac{1}{n} D_{cent}^T D_{cent} = \left(OSA^T\right)^T OSA^T = AS^T\left(O^T O\right)SA^T = A\left(S^T S\right)A^T = A\begin{pmatrix} \lambda_1^2 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_k^2 \end{pmatrix} A^T$$

- Here: A is a matrix of eigenvectors
- Eigenvalues of the covariance matrix = squared singular values of D
- O is a $n \times k$ matrix of orthonormal column vectors=> $O^T O$ is the identity matrix E

Conclusion: Eigenvalues and Eigenvectors of the covariance matrix Σ can be determined by the SVD of the data matrix D.

$\Rightarrow$ SVD is sometimes a better way to perform PCA (Large dimensionalities e.g., text data)
$\Rightarrow$ SVD can cope is dependent dimensions (k<d is an ordinary case in SVD)

Kernel PCA

Connection between the orthonormal busies O und A: $D = OSA^T$

- A is a k-dimensional basis of eigenvectors of $D^T \cdot D$ *(cf. previous slide)*
- *Analogously: O is a* k-dimension basis of Eigenvectors $D \cdot D^T$
  - $D \cdot D^T$ is a kernel matrix for the linear kernell <x,y> (cf. SVMs in KDD I)
  - The vectors of A and O are connected in the following way:

$$D_{cent} = OSA^T \Rightarrow O^T D_{cent} = O^T OSA^T = SA^T \Rightarrow S^{-1} O^T D_{cent} = A^T$$

$$\Rightarrow a_j = \sum_{i=1}^{n} o_{i,j} x_i$$

The j^{th} *d*-dimensional eigenvector in A is a linear combination of the vectors in D based on k-dimensional j^{th} eigenvectors as weighting vector (the i^{th} values is the weight for vector d_i)

$\Rightarrow$ A basis in vector space corresponds to a basis in the kernel space

$\Rightarrow$ A PCA can be computed for any kernel space based on the kernel matrix (Kernel PCA allows PCA in a non-linear transformation of the original data)

Kernel PCA

Let $K(x,y) = \langle \Phi(x), \Phi(y) \rangle$ a kernel for the non-linear transformation Φ(x).

Assume: *K(x,y)* is known, but Φ(x) is not explicitly given.

- Let K be the kernel matrix of D w.r.t. *K(x,y)* :

$$K = \begin{pmatrix} K(x_1, x_1) & \cdots & K(x_i, x_n) \\ \vdots & \ddots & \vdots \\ K(x_n, x_1) & \cdots & K(x_n, x_n) \end{pmatrix}$$

- The eigenvalue decomposition of K : $K = VSV^T$
  where V is a n-dimensional basis from eigenvectors of K
- To map *D* w.r.t. *V* the principal components in the target space the vectors x_i in *D must be transformed using* the kernel *K(x,y)*.

$$y' = \begin{pmatrix} \left\langle \Phi(y), \sum_{i=1}^{n} v_{i,1} \Phi(x_i) \right\rangle \\ \vdots \\ \left\langle \Phi(y), \sum_{i=1}^{n} v_{i,k} \Phi(x_i) \right\rangle \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} v_{i,1} \langle \Phi(y), \Phi(x_i) \rangle \\ \vdots \\ \sum_{i=1}^{n} v_{i,k} \langle \Phi(y), \Phi(x_i) \rangle \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} v_{i,1} K(y, x_i) \\ \vdots \\ \sum_{i=1}^{n} v_{i,k} K(y, x_i) \end{pmatrix}$$

Matrix factorization as an Optimization Task

SVD and PCA are standard problem in algebra.

- Matrix decomposition can be formulated as a optimization task.
- This allows a computation via numerical optimization algorithms
- In this formulation the diagonal matrix is often distributed to both basis matrixes

$$D = ASB^T = \left(A \begin{pmatrix} \sqrt{\lambda_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sqrt{\lambda_k} \end{pmatrix} \right) \left(\begin{pmatrix} \sqrt{\lambda_1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \sqrt{\lambda_k} \end{pmatrix} B^T \right) = UV^T$$

- As an optimization problem: $L(U,V) = \left\| D - UV^T \right\|_f^2$

(squared Frobenius Norm of a matrix) $\left\| M \right\|_f^2 = \sum_{i=1}^{n} \sum_{j=1}^{m} \left| m_{i,j} \right|^2$

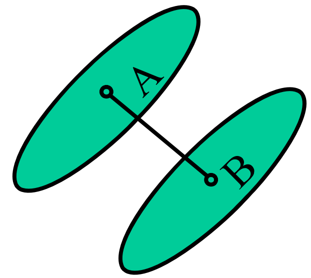
subject to: $\underset{i \neq j}{\forall} : \langle v_i, v_j \rangle = 0 \wedge \langle u_i, u_j \rangle = 0$

Fischer Faces

Idea: Use examples to increase the discriminative power of the target space.

$$\Sigma_b = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

$$\Sigma_w = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$



Target:

- Minimize the similarity between objects from different classes.
  (between class scatter matrix: Σ_b)

  Σ_b: Covariance matrix of the class centroids

$$\overline{\mu} = \frac{1}{|C|} \sum_{c \in C} \mu_c$$

$$\Sigma_b = \frac{1}{|C|} \begin{bmatrix} \mu_1 - \overline{\mu} \\ .. \\ \mu_m - \overline{\mu} \end{bmatrix}^T \cdot \begin{bmatrix} \mu_1 - \overline{\mu} \\ .. \\ \mu_m - \overline{\mu} \end{bmatrix}$$

- Maximize similarity between objects belonging to the same class
  (within class scatter matrix Σ_w)

  Σ: Average covariance matrix of all classes.

$$\Sigma_w = \frac{\sum_{C_i \in C} \Sigma_{C_i}}{|C|}$$

Fischer Faces

Determine basis x_i *in a way that* $S = \frac{x_i^T \cdot \Sigma_b \cdot x_i}{x_i^T \cdot \Sigma_w \cdot x_i}$ is maximized subject to $i \neq j : \langle x_i, x_j \rangle = 0$

Computation: Determine a orthonormal basis with dimensionality d‘ < d. Reduction to the eigenvalue decomposition.

$$\lambda_i \cdot x_i = \lambda_i \cdot \Sigma_w^{-1} \cdot \Sigma_b$$

Remark: The vector having the largest eigenvalue corresponds to the normal vector of the separating hyper plane in linear discriminant analysis or Fisher‘s discriminant analysis. (cf. KDD I)

Relevant Component Analysis (RCA)

Fischer Faces are limited due to nature of Σ_b and Σ_w :

Assumption of mono-modal classes:
each class is assumed to follow a multivariate

=> distribution of class centroids Σ_b

=> within correlation in Σ_w

Conclusion: Multi-modal or non-Gaussian distribution are not modeled well

Relevant Component Analysis:

- Remove linear dependent features (e.g. with SVD)
- Given: chunks data which are known to consist of similar objects.
  => replace Σ_w with an within-chunk matrix: $$\Sigma_{wc} = \frac{1}{|C|}\sum_{C_i \in C}\frac{1}{|C_i|}C_i^T C_i$$
- The covariance of all data objects is dominated by dissimilarity
  => replace Σ_b with the covariance matrix of D $$\Sigma = \frac{1}{|D|}D^T D$$

Large Margin Nearest Neighbor (LMNN)

Observation: Objects in a class might vary rather strongly.

Idea: Define an optimization problem only considering the distances the most similar objects from the same and other classes.

Define: $y_{i,j}=1$ *if* x_i and x_j *are from the same class else* $y_{i,j}=0$

- *Target:* $L: IR^d \rightarrow IR^d$ linear transformation of the vector space: $D(x,y) = \|L(x) - L(y)\|^2$
- Target neighbors: T_x k-nearest neighbors from the same class

  $\eta_{i,j}=1$: x_j is a target neighbor of x_i *else* $\eta_{i,j}=0$
- Training by minimizing the following error function:

$$E(L) = \sum_{i=1}^{n}\sum_{j=1}^{n} \eta_{i,j} \|L(x_i) - L(x_j)\|^2 + c \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{l=1}^{n} \eta_{i,j}(1 - y_{i,l})\left[1 + \|L(x_i) - L(x_j)\|^2 - \|L(x_i) - L(x_l)\|^2\right]_+$$

  where $[z]_+ = \max(z, o)$
- Problem is a ***semi-definite program***
  => Standard optimization problem where the optimization paramters must form a semi-definite matrix. Here the matrix is the basis transformation L(x).

Summary

- Linear basis transformation yield a rich framework to optimize feature spaces
- Unsupervized Methods delete low variant dimensions
  (PCA und SVD)
- Kernel PCA allows to compute PCA in non-linear kernel spaces
- Supervized methods try to minimize the within class distances while maximizing between class distances
- Fischer Faces extend linear discriminant analysis based on the assumption that all classes follow Gaussian distributions
- Relevant Component Analysis(RCA) generalize this notion and only minimize the distances between chunks of similar objects
- Large Margin Nearest Neighbor(LMNN) minimizes the distances to the nearest target neighbors and punish small distances to non-target neighbors in other classes

Literature

- S. Deerwester, S. Dumais, R. Harshman: *Indexing by Latent Semantic Analysis,* Journal of the American Society of Information Science,Vol. 41, 1990
- L. Yang and R. Jin. Distance metric learning: A comprehensive survey. Technical report, Department of Computer Science and Engineering, Michigan State University, 2006.
- K. Q. Weinberger and L. K. Saul. Distance metric learning for large margin nearest neighbor classication. Journal of Machine Learning Research, 10:207,244, 2009.
- P. Comon. Independent component analysis, a new concept? Signal Processing, 36(3):287{314, 1994.
- J. Davis, B. Kulis, S. Sra, and I. Dhillon. Information theoretic metric learning. In in NIPS 2006 Workshop on Learning to Compare Examples, 2007.
- A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall. Learning distance functions using equivalence relations. In Proceedings of the 20th International Conference on Machine Learning (ICML), Washington, DC, USA, pages 11{18, 2003.