2.3 Featurereduktion

Idee: Anstatt Features einfach wegzulassen, generiere einen neuen niedrigdimensionalen Feature-Raum aus allen Features:

- Redundante Features können zusammengefasst werden
- Irrelevantere Features haben einen entsprechend kleineres Gewicht in den neuen Features

Lösungsansätze:

- Referenzpunktansatz
- Hauptkomponentenanalyse (PCA)
- Single-Value-Decompusition (SVD)
- Fischer-Faces (FF) und Relevant Component Analysis(RCA)
- Large Margin Nearest Neighbor (LMNN)

1. Referenzpunkt Transformation

Idee:

Position eines Objekts kann häufig recht gut über den Abstand zu anderen Objekten beschrieben werden.

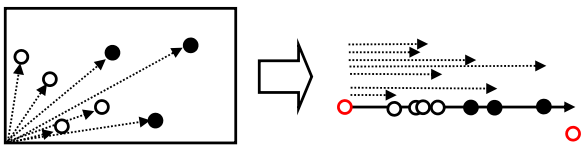
Wähle k Referenzpunkte und beschreibe Objekte über den k-dimensionalen Vektor der Abstände zu den Referenzpunkten.

Gegeben: Vektorraum $F = D_1 \times .. \times D_n$ mit $D = \{D_1,..,D_n\}$.

Gesucht: *k*-dimensionaler Raum *R*, der für ein gegebenes Data Mining Problem eine optimale Lösung erlaubt.

Methode: Für die Menge der Referenzpunkte $R = \{r_1,..,r_k\}$ und Distanzmaß $d()$:.

Transformation von Vektor $x \in F$:



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

1. Referenzpunkt Transformation

- Abstandsmaß ist meist durch Applikation gegeben.
- Auswahl der Referenzpunkte:
  - Wähle Centroide der Klassen oder Cluster-Centroide als Referenzpunkte
  - Häufig Wahl der Referenzpunkte am Rand und möglichst weit weg von allen Datenobjekten.

Vorteile:

- leicht umzusetzender Ansatz

Nachteil:

- selbst bei gleicher Feature Anzahl ist die Abbildung nicht eindeutig
- Performanz stark von der Wahl der Referenzpunkte abhängig.

Latente Räume und Matrix Faktorisierung

Basics über Vektoren und Matritzen:

- Inneres Produkt von zwei Vektoren x,y:

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

- Äußeres Produkt vonzwei Vektoren x,y:

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

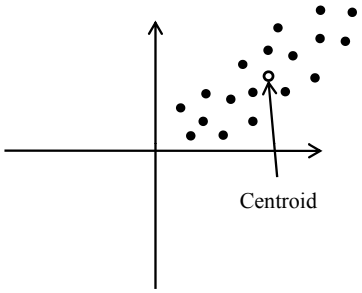
- Matrix-Multiplikation:

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

Datenmatrix

- Gegeben eine Menge von *n* Vektoren $v_i \in IR^d$, die *n*×*d* Matrix

$$D = \begin{pmatrix} v_1 \\ \vdots \\ v_d \end{pmatrix} = \begin{pmatrix} v_{1,1} & \cdots & v_{1,d} \\ \vdots & \ddots & \vdots \\ v_{n,1} & \cdots & v_{n,d} \end{pmatrix}$$ heißt Datenmatrix

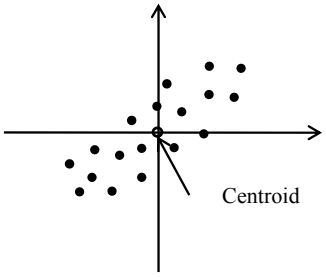


- Centroid/Erwartungsvektor einer Datenmenge:

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

- Zentrierte Datenmatrix:

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



Mahalanobis Distanz

- Quadratische Form oder Mahalanobis Distanz:

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

Bermerkung: Ist A symmetrisch und positiv definit dann ist d_M eine Metrik.

- Gewichtete euklydische Distanz: A ist eine Diagonalmatrix mit $A_i > 0$:

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

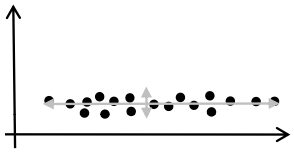
- **Zusammenhang Basistransformation**:

Wenn es ein symmetrische Zerlegung von $A = B \cdot B^T$ gibt, dann entspricht die Mahalanobis Distanz der euklydischen Distanz im mit *B* transformierten Raum:

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

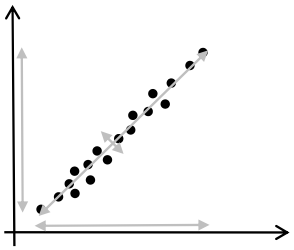
Varianzanalyse

- Welche Attribute sind die wichtigsten für die Bestimmung der Distanz?
  => Attribute mit stark unterschiedlichen Werten $|x_i-y_i|$
  => Abstand zum Erwartungswert ist groß $|x_i-\mu_i|$
  => Varianz ist groß: $\frac{1}{n}\sum_{i=1}^{n}(x_i-\mu_i)^2$



Idee: Varianzanalyse (eigtl. unsupervised Feature Selection)

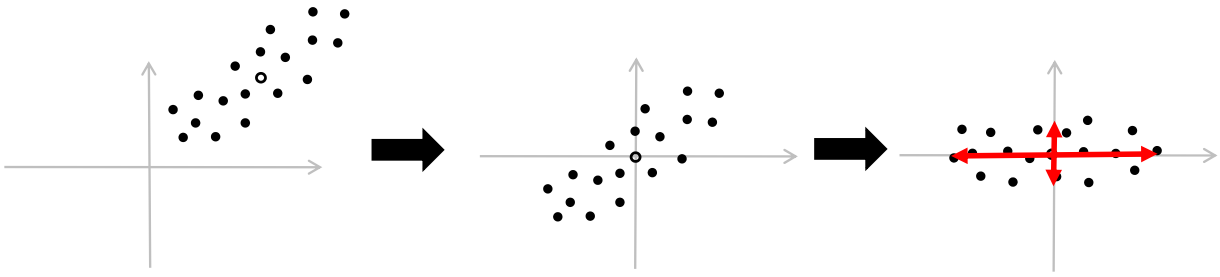
- Attribute mit großer Varianz ermöglichen Unterscheidung von Objekten
- Attribute mit kleiner Varianz: alle Objekte sind ähnlich unterschiedlich
- Vorgehen
  - Bestimme Varianz der Werteverteilung in allen Dimensionen
  - Sortiere alle Features absteigend nach der Varianz
  - Wähle die *k* mit der höchsten Varianz



ACHTUNG: Selbst linear Korrelation kann eindimensionale starke Varianzen auf beliebig viele Dimensionen aufteilen !

Principal Component Analysis (PCA)

Idee: Rotiere die Datenmenge so, dass die Hauptausdehnungsrichtungen auf den Hauptachsen angetragen werden und erlaube so ein Varianzanalyse auf den Hauptkomponenten (Principal Components).



- Drehe so, dass die Richtung mit der höchsten Varianz auf eine Hauptachse gelegt wird.
- Rotation entspricht einer Basistransformation mit einer Orthonormalbasis
  - Abbildung ist winkeltreu und abstandserhaltend:

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

- *B* entsteht aus der Richtung die jeweils die höchste Varianz aufweist und orthogonal zu allen bisher ausgewählten Basisvektoren ist.

Eigenwertzerlegung und Kovarianzmatrix

- Kovarianzmatrix:
  - Beschreibt Varianzen und Korrelationen(Kovarianzen)

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

  - Kovarianzmatrix Σ_D ($d \times d$) der $n \times d$ *Datenmatrix D*:

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

- Eigenwert λ_i und Eigenvektor v_i einer Matrix $d \times d$ D: $D \cdot v_i = \lambda_i \cdot v_i$
- Eigenwertzerlegung: $M = V \Lambda V^T$

$$V = (v_1, \cdots, v_d) \quad mit \forall_{i \neq j} \langle v_i, v_j \rangle = 0 \quad und \bigwedge_{i=1}^{d} \|v_i\| = 1$$

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

Eigenwertzerlegung der Kovarianzmatrix

- Wendet man die Eigenwertzerlegung auf die Kovarianzmatrix an:

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

- v_i: Orthogonale Hauptkomponenten
- λ_i: Varianzen in den Dimensionen

Achtung: Bei nullwertigen sind bestimmte Dimensionen komplett als Linearkombination anderer Dimensionen darstellbar.

=> Abhilfe man addiert eine Diagonalmatrix mit einem kleinen Wert δ_i auf alle Diagonalelemente

Ablauf PCA

Dimensionsreduktion via PCA

1. Berechne Kovarianzmatrix Σ
2. Berechne Eigenwerte und Eigenvektoren von Σ
3. Wähle die *k* größten Eigenwerte und deren Eigenvektoren (v‘)
4. Die *k* resultierenden Eigenvektoren bilden eine Orthonormalbasis
5. Transformiere die $n \times d$ Datenmatrix D mit der neuen $d \times k$ Basis V‘:

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

Singular Value Decomposition (SVD)

Verallgemeinerung der Eigenwertzerlegung

Zerlegung einer beliebigen $n \times d$ Matrix in 2 orthogonale Matritzen O,A und 1 Diagonalmatrix S aus Singulärwerten.

$$D = OSA^T$$

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

O : n $\times k$ links-singuläre Vektoren, orthogonale Spaltenmatrix
S : k $\times k$ Diagonalmatrix aus Singulär-Werten
A : k $\times d$ rechts-singuläre Vektoren, orthogonale Spaltenmatrix
k : Rang der Datenmatrix D (max. Anzahl an unabhängigen Zeilen/Spalten)
Zerlegung mittels numerischer Algorithmen zur Matrix-Faktorisierung.

Zusammenhang SVD und PCA

- Anwendung der SVD auf die Kovarianzmatrix

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

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

- In diesem Fall stellt A die Matrix der Eigenvektoren dar.
- Die Eigenwerte der Kovarianzmatrix sind die quadrierten Singulärwerte von D
- Da O eine n×k Matrix auf orthonormalen Spaltenvektoren ergibt O^TO die Einheitsmatrix E

Fazit: Sowohl die Eigenwerte als auch die Eigenvektoren der Kovarianzmatrix können auch über die SVD der Datenmatrix bestimmt werden.

⇒ Je nach Dimensionalität und eingesetzten Berechnungsalgorithmen ist SVD eine bessere alternative

⇒ SVD ist auch bei nullwertigen Eigenwerten einsetzbar (k<d ist bei SVD erlaubt)

Kernel PCA

Zusammenhang zwischen den Orthnormalbasen O und A: $D = OSA^T$

- O enthält eine k-dimensionale Basis aus Eigenvektoren von $D^T \cdot D$ *(vorherige Folie)*
- *Analog: A enthält eine* k-dimensionale Basis aus Eigenvektoren $D \cdot D^T$
  - $D \cdot D^T$ ist die Kernelmatrix zum linearen Kernel <x,y> (vgl. SVM aus KDD I)
  - Die Vektoren von A und O hängen dabei wie folgt zusammen

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

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

Der *j*-te *d*-dimensionale Eigenvektor kann als Linearkombination der Datenmenge mit dem Gewichtungsvektor des k-dimensionalen j-ten Eigenvektors dargestellt werden.

=> Hat man die Basis im Kernelraum kann man die Basis im Featureraum bestimmen

=> Diese Beobachtung erlaubt die Berechnung der PCA in beliebigen Kernelräumen (Kernel PCA)

Kernel PCA

Sei $K(x,y) = \langle \Phi(x), \Phi(y) \rangle$ eine Kernelfunktion zur nicht-linearen Transformation Φ(x).
Annahme: K(x,y) ist bekannt aber Φ(x) ist nicht explizit gegeben.

- Sei K die Kernelmatrix über D: $K = \begin{pmatrix} K(x_1,x_1) & \cdots & K(x_i,x_n) \\ \vdots & \ddots & \vdots \\ K(x_n,x_1) & \cdots & K(x_n,x_n) \end{pmatrix}$
- Die Eigenwertzerlegung von K ergibt dann: $K = VSV^T$
  wobei V eine n-dimensionale Basis aus Eigenvektoren von k ist
- Um D bzgl. V auf die Hauptkomponenten des Zielraums abzubilden
  müssen die Vektoren x_i in D mittels der Kernelfunktion K transformiert werden.

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

Matrixfaktorisierung als Optimierungsproblem

SVD und Eigenwertzerlegung sind Standardverfahren aus der Mathematik.

- Matrixzerlegungen können aber auch als Optimierungsproblem formuliert werden.
- dies ermöglicht häufig eine Berechnung über numerischen Optimierung
- Bei der Formulierung als Optimierungsproblem wird die Diagonalmatrix häufig auf die resultierenden Matritzen aufgeteilt.

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

- Als Optimierungsproblem: $L(U,V) = \left\| D - UV^T \right\|_f^2$
  mit $\|M\|_f^2 = \sum_{i=1}^{n} \sum_{j=1}^{m} |m_{i,j}|^2$ (quadriete Frobeniusnorm einer Matrix)
  Nebenbedingungen: $\forall_{i \neq j} : \langle v_i, v_j \rangle = 0 \wedge \langle v_i, v_j \rangle = 0$

Fischer Faces

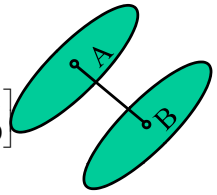
Idee: Nutze Klasseninformationen um relevanten Teil des Raumes zu erhalten.

Ziel:

- Minimiere die Ähnlichkeit zwischen Objekten unterschiedlicher Klassen
  (Between Class Scatter Matrix: Σ_b)
  Σ_b: Kovarianzmatrix der Klassencentroide

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

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



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

- Maximiere die Ähnlichkeit zwischen Objekten derselben Klasse
  (Within Class Scatter Matrix Σ_w)
  Σ: Durchschnittliche Kovarianzmatrix innerhalb der Klassen

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

Fischer Faces

Suche Basisvektoren x_i so dass $S = \frac{x_i^T \cdot \Sigma_b \cdot x_i}{x_i^T \cdot \Sigma_w \cdot x_i}$ maximal wird unter der Bedingung $i \neq j : \langle x_i, x_j \rangle = 0$

Berechnung: Gesucht orthnomale Basis der Dimension d‘ < d.

Rückführung des Problems auf Eigenwertproblem.

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

Bemerkung: Der Vektor mit der dem größten Eigenwert entspricht der Normalen der trennenden Hyperebene einer LDA (Fisher‘s Diskriminanzanalyse)

Relevant Component Analysis (RCA)

Fischer Faces haben Nachteile bzgl. Σ_b und Σ:

- Annahme mono-modaler Klassen:
  jede Klasse lässt sich durch 1 Normalverteilung darstellen
  => Verteilung der Centroide modeliert Σ_b
  => Verteilung der Klasse basiert auf Kovarianz Σ für Klasse C
- Bei nicht-normalverteilten Klassen schlechte Darstellung der Daten

Relevant Component Analysis:

- Entfernen vollständig abhängiger Dimensionen (z.B. PCA mit Hilfe der SVD)
- Es existieren Chunks von denen man, weiß das Sie ähnlich sind.
  => ersetze Σ_w durch Within-Chunk-Matrix: $\Sigma_{wc} = \frac{1}{|C|}\sum_{C_i \in C}\frac{1}{|C_i|}C_i^T C_i$
- Betrachtet man die Kovarianz aller Daten ist diese überwiegend durch unähnliche Objekte bestimmt:
  $$\Sigma = \frac{1}{|D|} D^T D$$

Large Margin Nearest Neighbor (LMNN)

Beobachtung: Nicht alle Elemente einer Klasse müssen gleich sein.

Idee: Bilde ein Optimierungsproblem, dass nur die Distanz zu den *k*-nächsten Nachbarn der gleichen Klasse zu minimieren.

- $y_{i,j}$=*1* falls x_i und x_j in der gleichen Klasse. $y_{i,j}$=*0 sonst*
- *Gesucht: L:IR^d→IR^d* lineare Transformation des Raums: $D(x,y) = \|L(x) - L(y)\|^2$
- Targetneighbors: T_x k-nächste Nachbarn innerhalb der gleichen Klasse
  $\eta_{i,j}$ =1 : x_j ist Targetneighbor von x_i, $\eta_{i,j}$ =0 *sonst*
- Lernen der Transformation über Minimieren der folgenden Fehlerfunktion:
  $$E(L) = \sum_{i=1}^{n}\sum_{j=1}^{n}\eta_{i,j}\|L(x_i) - L(x_j)\|^2 + c\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{l=1}^{n}\eta_{i,j}(1-y_{i,l})\left[1 + \|L(x_i) - L(x_j)\|^2 - \|L(x_i) - L(x_l)\|^2\right]_+$$
  mit $[z]_+ = \max(z, o)$
- Optimierung über semidefinites Programm
  (Optimierungsproblem bei denen die zu optimierenden Paramter eine semidefinite Matrix bilden. hier L die Basis des Zielraumes)

Zusammenfassung

- Lineare Basistransformationen bilden ein reiches Framework zur Optimierung von Feature-Räumen
- Es gibt unsupervised Methoden, die niedrig-variante Faktoren eleminieren (PCA und SVD)
- Kernel PCA erlaubt die Berechnung der PCA in nicht-linearen Kernelräumen
- Es gibt Supervised Verfahren, die versuchen Distanzen zwischen Objekten der gleichen Klasse zu minimieren und Distanzen zwischen Objekten unterschiedlicher Klassen zu maximieren.
- Fischer Faces erweitern Lineare Diskriminanz Analyse basieren aber auf der Annahme normalverteilter Klassen
- Relevant Component Analysis(RCA) verallgemeinert diese Annahme und minimiert nur die Distanzen innerhalb von Chunks
- Large Margin Nearest Neighbor(LMNN) minimiert Distanzen zu den nächsten Nachbarn und bestraft kleine Distanzen zu nicht Nichtarge-Neighbors, die zu anderen Klassen gehören.

Literatur

- 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.