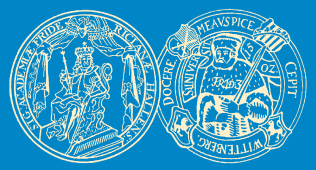
Principal Component Analysis (PCA)

- Motivation: Klassifikation mit der PCA
- Berechnung der Hauptkomponenten
- Theoretische Hintergründe
- Anwendungsbeispiel: Klassifikation von Gesichtern
- Weiterführende Bemerkungen



Birgit Möller & Denis Williams
AG Bioinformatik & Mustererkennung
Institut für Informatik
Martin-Luther-Universität Halle-Wittenberg

Motivation

Rückblick: Klassifikation von Mustern

Allgemeine Vorgehensweise:

1. Berechnung von *geeigneten* Merkmalen aus den Mustervektoren
2. Training eines Klassifikators, z.B. NN-Klassifikator
3. Zuordnung neuer Merkmalsvektoren zu einer Klasse

$\Rightarrow$ offene Frage: Wie findet man geeignete Merkmale zur Klassifikation?

- für Bilder als Mustervektoren bewährt:

  Hauptkomponentenanalyse (PCA)

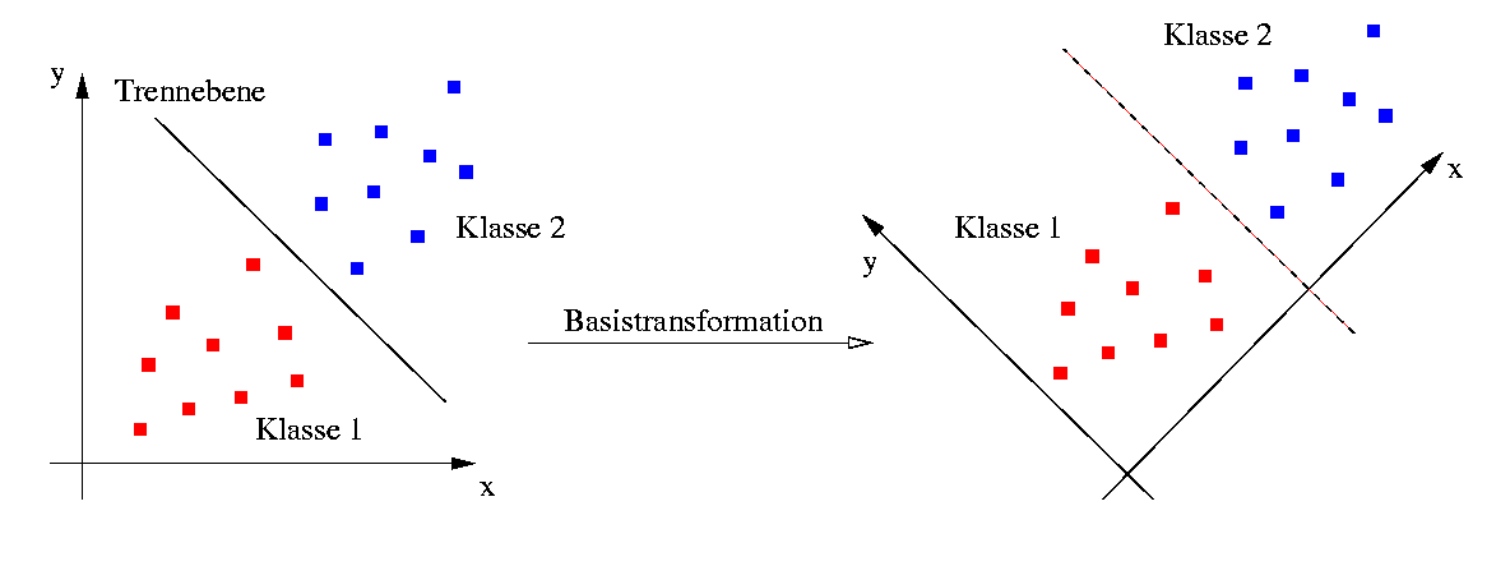
Motivation

Grundidee:

”Ermögliche Klassifikation durch Transformation der Mustervektoren in einen niedriger-dimensionalen Unterraum, in dem der Hauptteil der Datenvariation liegt.”

Annahme dabei:

Variation in den Daten entspricht einem hohen Informationsgehalt!



Algorithmus

- gegeben seien mittelwert-freie Mustervektoren $\vec{x}^\alpha, \alpha = 1 \ldots N, \vec{x}^\alpha \in R^d$:

$$\frac{1}{N} \sum_{\alpha=1}^{N} \vec{x}^\alpha = 0$$

Algorithmus:

1. berechne die Autokorrelationsmatrix C^{xx} der Datenmenge:

$$C_{ij}^{xx} = \frac{1}{N} \sum_{\alpha=1}^{N} \vec{x}_i^\alpha \vec{x}_j^\alpha \tag{1}$$

   - C^{xx} ist positiv-definit und symmetrisch.

2. berechne die Eigenwerte λ_i und die Eigenvektoren $\hat{v}_i$ von C^{xx}:

$$C^{xx} \cdot \hat{v}_i = \lambda_i \cdot \hat{v}_i \quad \text{(Eigenwertgleichung)} \tag{2}$$

   - aufgrund der Symmetrie gilt: $\hat{v}_i \cdot \hat{v}_j := \delta_{ij}$

     $\Rightarrow$ die Eigenvektoren bilden eine Orthonormal-Basis des R^d

Es gilt nun:

- jeder Datenvektor $\vec{x}^{\alpha}$ besitzt die Eigenvektorzerlegung

$$\vec{x}^{\alpha} = \sum_{i=1}^{d} t_i^{\alpha} \cdot \hat{v}_i \qquad \Longleftrightarrow \qquad t_j^{\alpha} = \vec{x}^{\alpha} \cdot \hat{v}_j \tag{3}$$

- die t_i^{α} sind zentriert und paarweise unkorreliert
- die Eigenwerte λ_i liefern die Varianz in den t_i^{α}:

$$\frac{1}{N} \sum_{\alpha=1}^{N} t_i^{\alpha} t_j^{\alpha} = \lambda_i \cdot \delta_{ij} \quad , \tag{4}$$

denn

$$\frac{1}{N} \sum_{\alpha=1}^{N} t_i^{\alpha} t_j^{\alpha} \;=\; \frac{1}{N} \sum_{\alpha=1}^{N} \hat{v}_i \, \vec{x}^{\alpha} \, (\vec{x}^{\alpha})^T \, \hat{v}_j \;=\; \hat{v}_i \cdot C^{xx} \cdot \hat{v}_j \;=\; \lambda_i \cdot \delta_{ij} \tag{5}$$

Interpretation

- die Eigenvektorzerlegung beschreibt jeden Vektor $\vec{x}^{\alpha}$ durch einen neuen Parametervektor (Merkmalsvektor!)

$$\vec{t}^{\alpha} = (t_1^{\alpha}, t_2^{\alpha}, \cdots, t_d^{\alpha})^T$$

- die t_i^{α} gehen durch lineare Transformation aus den $\vec{x}^{\alpha}$ hervor:

$$t_j^{\alpha} = \vec{x}^{\alpha} \cdot \hat{v}_j$$

- die Eigenwerte λ_i liefern die Varianzen in den einzelnen t_i^{α}

$\Rightarrow$ **Dimensionsreduktion** durch Auswahl einer Teilmenge der Basisvektoren bei der Transformation

Dimensionsreduktion

- die Eigenwerte seien absteigend sortiert:

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$$

- Abbruch der Eigenvektorzerlegung nach dem k-ten Term liefert Approximation $\tilde{x}^\alpha$ für $\vec{x}^\alpha$:

$$\tilde{x}^\alpha = \sum_{j=1}^{k} t_j^\alpha \cdot \hat{v}_j$$

- Approximationsfehler $\delta\tilde{x}^\alpha$:

$$\delta\tilde{x}^\alpha = \vec{x}^\alpha - \tilde{x}^\alpha = \sum_{j=k+1}^{d} t_j^\alpha \cdot \hat{v}_j$$

Statistische Analyse

Frage: Wie groß ist der Approximationsfehler im Mittel?

- Berechnung des Erwartungswertes des quadratischen Fehlers:

$$
\begin{aligned}
< (\delta\tilde{x}^{\alpha})^2 >_{\alpha} &= \frac{1}{N}\sum_{\alpha}(\delta\tilde{x}^{\alpha})^2 \\
&= \frac{1}{N}\sum_{\alpha}\sum_{i>k}\sum_{j>k} t_i^{\alpha} t_j^{\alpha} \hat{v}_i \hat{v}_j \\
&= \frac{1}{N}\sum_{\alpha}\sum_{i>k}\sum_{j>k} t_i^{\alpha} t_j^{\alpha} \delta_{ij} \\
&= \frac{1}{N}\sum_{\alpha}\sum_{i>k}(t_i^{\alpha})^2 = \sum_{i>k} < (t_i^{\alpha})^2 >_{\alpha} = \sum_{i>k}\lambda_i
\end{aligned}
$$

$\Rightarrow$ der mittlere Approximationsfehler ist gleich der Summe unberücksichtigter Eigenwerte!

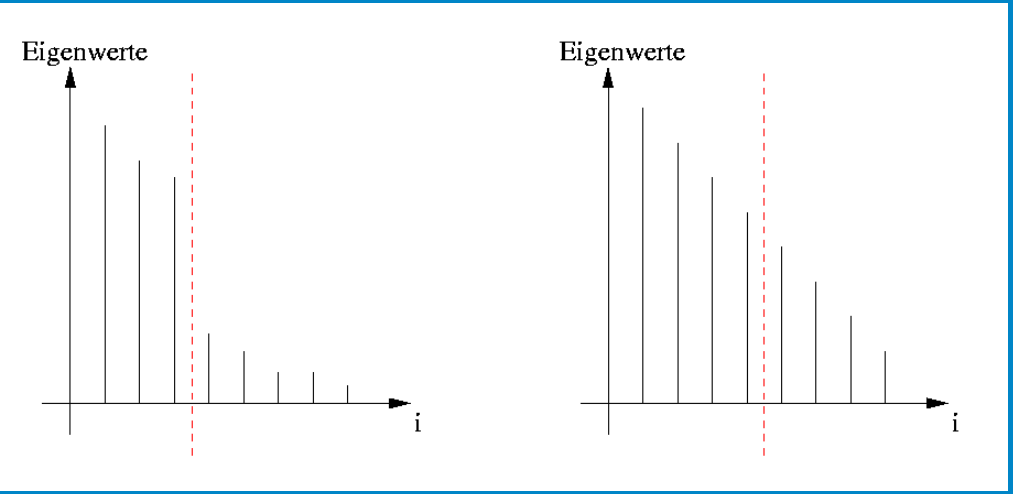
Statistische Analyse

Fazit:

- Mitnahme der k größten Eigenvektoren führt zu Minimierung des mittleren Approximationsfehlers unter allen Projektionen auf k-dimensionale Unterräume
- Dimensionsreduktion auch bekannt als **Karhunen-Loeve-Entwicklung**

Offene Frage: Wie wählt man k geschickt?!

$\Rightarrow$ anhand der Eigenwertverteilung von C^{xx}



Zwischenfazit

- Eigenwertanalyse gibt Aufschluß über intrinsische Datendimensionalität
- PCA macht keine Aussage über semantischen Gehalt der Daten

$\Rightarrow$ Achtung bei starkem Rauschen in den Daten!!!

Fazit:

- PCA fokussiert durch Dimensionsreduktion auf spezifische Charakteristika der zu klassifizierenden Daten
- der entstehende (niedrig-dimensionale) Datenraum beschreibt die Mustercharakteristik optimal bei gewählter Dimension k

Klassifikation von Gesichtern - Eigenfaces

Grundidee:

- Repräsentation der gesicht-spezifischen Merkmale von Bildern in einem geeigneten Unterraum
- Klassifikation eines unbekannten Musters durch Auswertung seiner Projektion in den gewählten Unterraum

Der Klassifikator unterscheidet zwei Modi:

1. Systeminitialisierung:

   Training des Klassifikators auf einer Trainingsmenge
2. Arbeitsphase:

   Klassifikation unbekannter Muster (mit optionalem Update)

Phase I - Initialisierung

- gegeben eine Menge von M Trainingsmustern $\vec{x}_\alpha \in R^{N^2}, \alpha = 1 \cdots M$
  (fasse $N \times N$-dimensionales Bild als N^2-dimensionalen Vektor auf)
- berechne *"Facespace"* durch Auswahl von L Eigenvektoren der Korrelationsmatrix C^{xx} als Basis des gesuchten Unterraums R^L
- berechne Merkmalsvektoren $\vec{\omega}_\alpha$ der Trainingsvektoren $\vec{x}_\alpha$ als Repräsentanten der einzelnen Klassen Ω_i (NN-Klassifikator)

Berechnung der Eigenvektoren:

$$C^{xx} = \frac{1}{M} \sum_{\alpha=1}^{M} \vec{x}_\alpha (\vec{x}_\alpha)^T = \frac{1}{M} A\, A^T \text{ mit } A = [\vec{x}_1 \cdots \vec{x}_M] \tag{6}$$

Problem:

$\vec{x}_i \in R^{N^2}$, d.h. für Bilder der Größe 256 $\times$ 256 folgt $N^2 = 65536$ und $A \in R^{N^2 \times N^2}$!!!

Phase I - Initialisierung

"Trick 17":

Ist die Anzahl der Datenpunkte M sehr viel kleiner als ihre Dimension N^2, dann lassen sich nur maximal $M-1$ aussagekräftige Eigenvektoren finden!

$\Rightarrow$ leite Eigenvektoren aus niedrig-dimensionalem Unterraum ab!

- die Eigenvektoren $\hat{v}_i$ von $A^T A$ sind gegeben durch

$$A^T A \hat{v}_i = \lambda_i \hat{v}_i$$

- ferner gilt: $AA^T A\hat{v}_i = \lambda_i A\hat{v}_i$

$\Rightarrow$ die $A\hat{v}_i$ entsprechen den Eigenvektoren $\hat{u}_i$ von $C^{xx} = AA^T$

- $A^T A$ hat die Dimension $M \times M$, mit $(A^T A)_{mn} = \vec{x}_m^T \vec{x}_n$

Phase I - Initialisierung

Algorithmus - auf einen Blick:

1. berechne Matrix $A^T A$
2. berechne die Eigenvektoren $\hat{v}_i$ von $A^T A$
3. berechne die Eigenvektoren u_i von AA^T (“Eigenfaces“) aus $u_i = A\hat{v}_i$
4. projiziere die Trainingsmuster $\vec{x}_\alpha$ in den Unterraum (“Facespace“) und verwende deren Projektionen $\vec{\omega}_\alpha$ als Repräsentanten für einen intuitiven Klassifikator

Phase II - Klassifikation

Gegeben ein unbekanntes Muster $\vec{x}$.

- berechne die Eigenvektorzerlegung des Eingabemusters

  (Projektion in den *"Facespace"*)

$$t_k = \hat{u}_k \cdot (\vec{x} - \bar{x}) \text{ mit } \bar{x} = \frac{1}{M} \sum_{\alpha=1}^{M} \vec{x}_\alpha$$

- klassifiziere über Distanzen zu den Repräsentanten:

$$\epsilon_i = \| \vec{t} - \vec{\omega}_i \|^2 < \theta_1 \text{ mit } \vec{\omega}_i \text{ Repräsentant der Klasse } \Omega_i$$

- Problem:

  auch Nicht-Gesichter werden unter Umständen auf Merkmalsvektoren

  nahe den Repräsentanten abgebildet!

Phase II - Klassifikation

Rückweisungskriterium:

$$\psi = \| \vec{x} - \tilde{x} \|^2 > \theta_2 \text{ mit } \tilde{x} = \sum_{i=1}^{L} t_i^{\vec{x}} \hat{u}_i$$

Damit resultieren drei Fälle, die zu unterscheiden sind:

a) $\psi > \theta_2$: Rückweisung

b) $\psi < \theta_2, \epsilon > \theta_1$: unbekanntes Gesicht

c) $\psi < \theta_2, \epsilon < \theta_1$: bekanntes Gesicht

Abschliessende Bemerkungen

- Trainingsmenge:

  je größer, desto besser, aber auch desto aufwändiger! (30 bis 40 Bilder)

- Eigenfaces sind nicht
  - skalierungsinvariant

    $\Rightarrow$ Ausschnitte gleicher Größe verwenden oder explizit skalieren!

    (erst Lokalisation anhand alternativer Kriterien, dann Skalierung)
  - beleuchtungsinvariant

    $\Rightarrow$ zu große Varianz in der Beleuchtung vermeiden!

- weiteres Problem: Hintergrund!

  $\Rightarrow$ je mehr Varianz dort, desto störender!

  (einfarbigen Hintergrund verwenden oder Randbereiche maskieren)

Abschliessende Bemerkungen

Suche nach Gesichtern in großen Bildern:

- projiziere jeweils Ausschnitte in den ”Facespace” und klassifiziere gemäß

$$\psi = \| \vec{x} - \tilde{x} \|^2 > \theta_2 \text{ mit } \tilde{x} = \sum_{i=1}^{L} t_i^{\vec{x}} \hat{u}_i$$

- Aber Achtung: mitunter sehr aufwändig!!!
- Verbesserungen:
  - Offline-Berechnung einzelner Terme
  - Auflösungspyramide
  - Groblokalisation, z.B. durch Farbe, dann Verifikation
  - Kalmanfilter beim Tracking