Principal Component Analysis

Algorithmic Learning, Teil 3c

Norman Hendrich

University of Hamburg
MIN Faculty, Dept. of Informatics
Vogt-Kölln-Str. 30, D-22527 Hamburg
hendrich@informatik.uni-hamburg.de

27/06/2012

Outline

Overview

- datasets in high-dimensional spaces. . .
- the "*curse of dimensionality*"

- reminder: linear algebra
- PCA: Principal Component Analysis
- ICA: Independent Component Analysis

- LLE: Locally Linear Embedding
- IsoMap, . . .

The *Curse of Dimensionality*

- problem: the analysis/classification of datasets literally "explodes" with the dimension n of the data

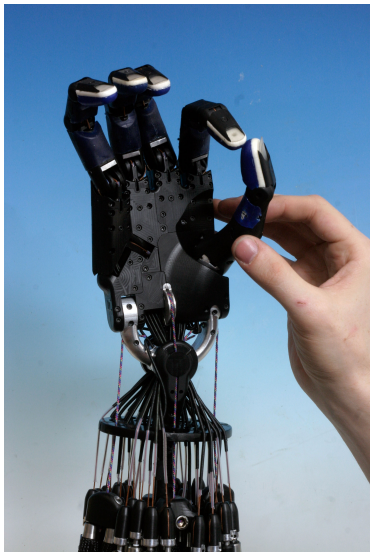
- volume of a n-dim. space increases exponentially with n
- basic example:
  - 1D: 10^2 points (100) are sufficient to cover the 1D-unit-interval [0..1] with a precision of 0.01
  - 10D: 10^{20} points required to cover the 10D-unit-cube with a precision of 0.01
  - in other words: the 10-D unit-cube seems to be larger than the 1-D unit-interval by a factor of 10^{18} at the requested precision
  - requested precision of 10^{-6}: 10^6 (1D) vs. 10^{60} (10D) etc.

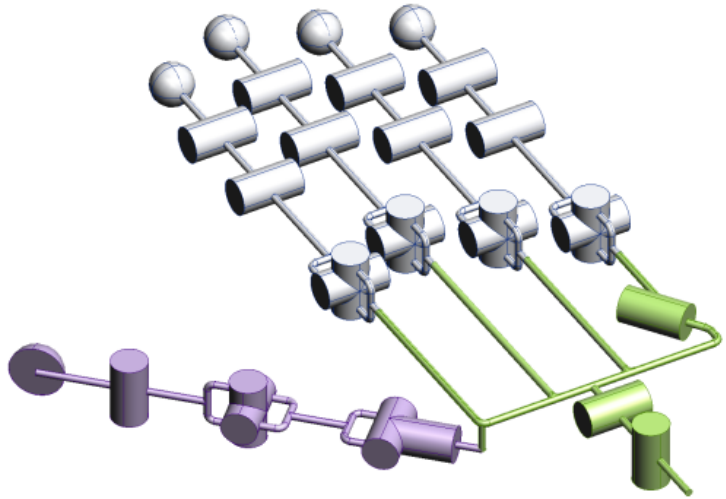
(term *curse of dimensionality* introduced by Richard Bellman, 1961)

Typical number of dimensions?!

- robot-arm: typically 6-7 DOF, or 6-7 dimensions
- human hand: finger-kinematics approximated with 24-DOF, grasps require additional 6-DOF to describe hand-object pose
- audio-clips: 16-bit, 44.1 kHz, 10 sec: 441000 dimensions
- 10-Mpixel photo, RGB: $3 \cdot 10 \cdot 10^6$ dimensions
- QVGA, gray-scale: $320x240 = 76800$ dimensions per frame
- VGA, RGB-video: $640 \cdot 480 \cdot 3 = 921600$ dimensions per frame
- VGA RGB video, 25 fps, 5 minutes $= 6.9 \cdot 10^9$ values
- DNA sequence: typically $10^6..10^9$ base-pairs
- text analysis: $10^3..10^7$ characters ...
- etc.

Example: robot grasping





- hand-model: finger-kinematics using 24-DOF
- grasps require additional 6-DOF to describe hand-object pose
- but: finger movements not at all independent

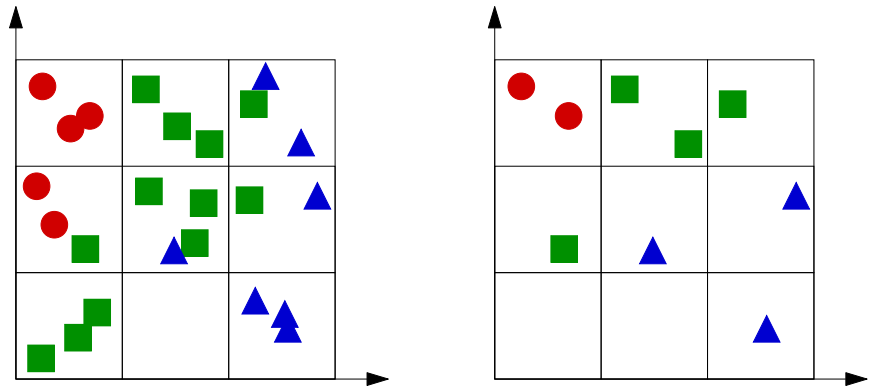
Example: sorting data into three classes, 1D

- divide input space into evenly spaced intervals
- sort all training-data into intervals

- count population per interval
- classify based on the dominant class per interval

- but: not meaningful, too many overlaps between classes

Example: sorting data into three classes, 2D

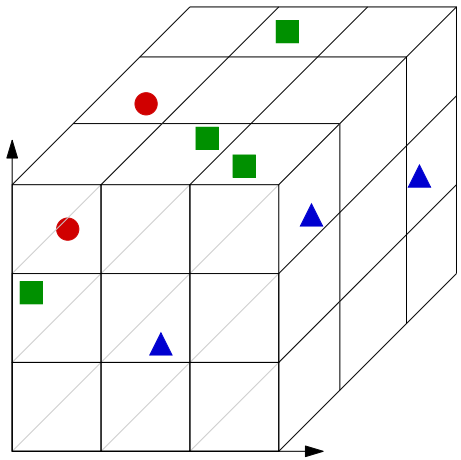


- divide input space into evenly spaced intervals
- sort all training-data into intervals
- count population per interval

now: two independent axes/variables

- at constant number of data samples (left)?
- at constant density of data samples (right)?

Example: sorting data into three classes, 3D



- divide input space into evenly spaced intervals

transition to 3D or nD brings out the problem:

- constant number of samples: space almost empty
- constant density of samples: enormous number of data required

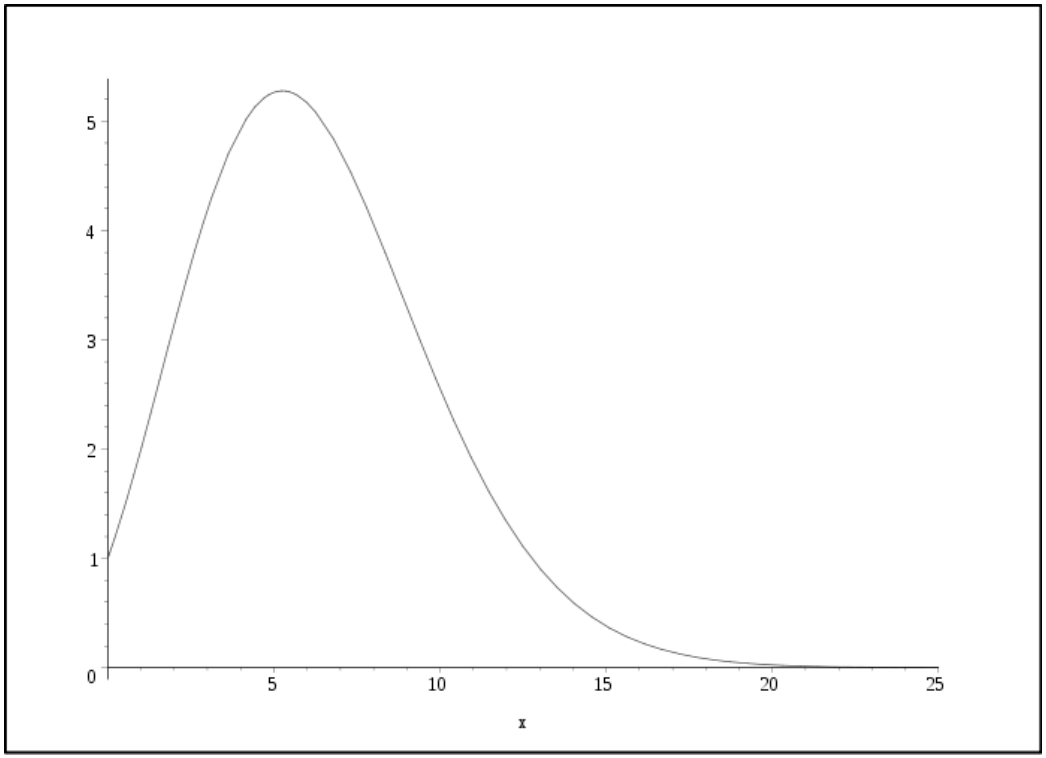
Volume of unit-cube and unit-sphere

- unit-cube: n-dimensional cube with side-length 1
- $V_{\mathrm{c}}(n) = 1^n$
- unit-cube around the unit-sphere: $V_{2\mathrm{c}}(n) = 2^n$

- unit-sphere: n-dimensional sphere with radius 1
- $V_{\mathrm{s}}(n) = \frac{\pi^{n/2}}{\Gamma(1+n/2)}$

- example $n = 10$: $V_{2c}(10) = 1024$, $V_s(10) = \pi^5/120 \approx 2.55$
- at increasing n, the sphere becomes insignificant to the cube
- rephrase: almost all points are far from the center in n-D
- the n-dimensional space is dominated by its edges/corners

Relative volume of the unit-sphere



- note: the unit-sphere formula can be evaluated for all real values of n

(Wikipedia: Ball volume in n dimensions)

How to solve this dilemma?

- exploit pre-knowledge about the problem
- suitable pre-processing of the data samples
- reduce the (effective) dimensionality of the problem

- limit the number of features/properties
- at given number of training samples

beyond that:

- loss of precision due to dimensionality problem
- analysis or learning impossible due to insufficient number/density of samples

Only hope: actual complexity is smaller than n

- representation of a problem defines the dimensionality
- but actual complexity can be (much) smaller

Example: record pendulum with multi-camera system

- actual movement is basic 1D/2D/3D oscillation
- equations of motions are known and can be solved easily

- sensor data from multiple cameras
- different view-points, pendulum and background
- requires: extraction of the significant parameters/dimensions from the (enormous number) of recorded measurements

Summary

- volume of the n-dim. space increases exponentially with n

- exponential count of data samples to fill the n-dim. space
- exponential complexity of n-dim. goal-functions
- exponential number of training data to learn a goal function

- 1D: many different density function known and analysed
- nD: usually, only Gauss-functions suitable/tractable

- hope: actual complexity is *not* proportional to number of input dimensions n: reduction of dimensionality

Reminder: Linear Algebra

- Matrices
- Vector spaces
- Linear transformations
- Subspaces, span, and basis
- Eigenvalues and eigenvectors
- ...

- http://en.wikipedia.org/wiki/Matrix_(mathematics)
- http://en.wikipedia.org/wiki/Linear_algebra
- http://en.wikipedia.org/wiki/Singular_value_decomposition

Eigenvalues and Eigenvectors

- Matrix $\mathbf{A}$
- Eigenvector x with Eigenvalue λ, iff:

$$\mathbf{A}x = \lambda x$$

- of course, requires $x \neq 0$
- note: any linear scaling of the equation is possible

Eigenvalues and Eigenvectors

- rewriting the equation gives

$$(\mathbf{A} - \mathbf{I}\lambda)x = 0$$
$$\det|\mathbf{A} - \mathbf{I}\lambda)| = 0$$

- determinant results in n-th degree polynom

- up to n real zeroes (Eigenvalues)
- always n complex zeroes (Eigenvalues)

Singular Value Decomposition

- formally, the singular value decomposition of an $m \times n$ real or complex matrix M is a factorization of the form

$$M = U\Sigma V^*$$

  where U is an $m \times m$ real or complex unitary matrix,
  Σ is an $m \times n$ rectangular diagonal matrix with nonnegative real numbers on the diagonal,
  and $V*$ (the conjugate transpose of V) is an $n \times n$ real or complex unitary matrix.
- diagonal entries $\Sigma_{i,i}$ are the singular values of M.
- The m columns of U and the n columns of V are called the left singular vectors and right singular vectors of M, respectively.
- the left singular vectors of M are eigenvectors of MM^*
- the right singular vectors of M are eigenvectors of M^*M

Principal Component Analysis

- simple mathematical transformation
- rotation/translation of the coordinate system
- replace correlated variables by uncorrelated variables
- sort new variables (coordinates) by relevance (= variance)
- remove insignificant variables: reduction of effective dimensionality

- aka Karhunen-Loève transformation
- aka Hotelling-transformation
- aka *proper orthogonal decomposition*

(Wikipedia: en.wikipedia.org/wiki/Principal_component_analysis, Karl Pearson 1901)

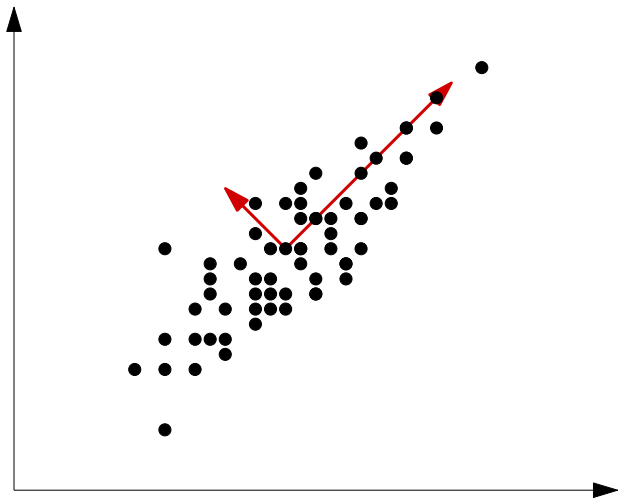
Principal Component Analysis: Basic idea

- select a new coordinate system for your data
- step 1: calculate and remove the mean of the data
- step 2: rotate the coordinate axes
- step 3: sort coordinates by the variance of the data

- calculation via Eigenvalues of the covariance matrix
- or SVD of the data matrix

- large variance indicates significant structure in the data
- coordinates with small variance are not important: ignore them

Example: 2D-dataset



- data distribution tilted along the original coordinates
- new first coordinate x_1 along the main variance of the data
- second new coordinate x_2 orthogonal to x_1

Solving the dimensionality problem with PCA

Basic idea: projection of the original problem into a new subspace, sorted by the variance of the data

dimensionality reduction: due to the projection

$\Rightarrow$ in the new subspace (coordinate system), fewer attributes are sufficient to describe the data

Example: image classification

Example problem:

- training data: training images $\vec{x}$ with $\vec{x} = [x_1, x_2, \ldots, x_t]$
- model: input images correspond to one of k-classes
- task: classify new input images according ot the training images

Problem: image classification

- typical camera images are very large,
- here: larger than 76800 (320x240) pixels

$\Rightarrow$ direct comparison between input images is computationally expensive

$\Rightarrow$ dimensionality problem

- no generalization

PCA: step 1

- input data:
  We need suitable training-samples $\vec{x}_M$
- remove the mean of the data:
  This step removes the mean values of the training samples

$$\vec{x} - \vec{\mu} \quad \text{mit} \tag{1}$$

$$\vec{\mu} = \frac{1}{M} \sum_{i=1}^{M} \vec{x}_M \tag{2}$$

PCA: step 2

- calculation of the covariance of the data:
  Calculate the covariance-matrix from the mean-value corrected samples:

$$\mathbf{Q} = \mathbf{P}\mathbf{P}^T \quad \text{mit} \tag{3}$$

$$\mathbf{P} = [\vec{x}_1 - \vec{\mu}, \vec{x}_2 - \vec{\mu}, \ldots, \vec{x}_M - \vec{\mu}] \tag{4}$$

- calculation of the Eigenvalues:
  the Eigenvalues of the covariance-matrix are calculated as:

$$\lambda_i \vec{e}_i = \mathbf{Q}\vec{e}_i \quad \text{where} \tag{5}$$

$$\lambda_i \quad i = 1 \ldots m \quad \text{Eigenvalues and}$$

$$\vec{e}_i \quad i = 1 \ldots m \quad \text{Eigenvectors}$$

PCA: step 3

- dimensionality reduction:
  sort the Eigenvectors $\vec{e}_i$ according to the Eigenvalues:

$$\lambda_1 > \lambda_2 > \ldots > \lambda_m$$

  the "information content" decreases with increasing index i.
- the mean-corrected rotated data still contains the full information of the original data
- to reduce the dimensionality, we only keep the first n Eigenvectors ($n < m$), and drop the rest of the Eigenvectors which contain little useful information.

PCA: step 4

- the transformation matrix:

$$\mathbf{A} = (\vec{e}_1 \dots \vec{e}_n)^T$$

transforms input data (test samples) into the Eigenspace:

$$\vec{p}_i = \mathbf{A} \cdot \vec{x}_i \qquad dim(\vec{p}_i) = n$$

- back-projection:
  A is square and orthogonal, therefore:

$$\mathbf{A}^{-1} = \mathbf{A}^T$$

The back-transformation from the Eigenspace into the original data-space is given by:

$$\vec{x}_i = \mathbf{A}^{-1}\vec{p}_i = \mathbf{A}^T\vec{p}_i$$

Dimensionality reduction

- Selecting a large value of n:
  $\Rightarrow$ little information loss, but also little reduction of dimensionality
- Selecting a large value of n:
  $\Rightarrow$ potentially large loss of information from the original training data, but also signification reduction of the dimensionality (complexity), so subsequent operations are easier

Automatic selection of n :

$$\frac{\sum_{i=1}^{n} \lambda_i}{\sum_{i=1}^{m} \lambda_i} \geq T$$

PCA: implicit covariance (1)

calculation of the covariance-matrix is potentially expensive:
image with 320×240 pixels $\Rightarrow$ vectors with dimension 76800
covariance-matrix :

$$\mathbf{Q} = \mathbf{P}\mathbf{P}^T \qquad \mathbf{Q} \in M_{t \times t}$$

$\Rightarrow 76800^2 = 5,89824 * 10^9$ elements
$\Rightarrow$ even with 1 byte per element: $\approx 5,5\,\mathrm{GB}$
$\Rightarrow$ correspondingly worse with increasing n

PCA: implicit covariance (2)

Usually, we only have $M \ll dim(\vec{x}_i)$ training samples, which limits the number of potential Eigenvectors to M.
Implicit covariance:

$$\widetilde{\mathbf{Q}} = \mathbf{P}^T\mathbf{P} \qquad \widetilde{\mathbf{Q}} \in M_{M\times M}$$

Example: 100 input images with 320×240 pixels each:
$\Rightarrow 100^2 = 10000$ Elemente
$\Rightarrow$ at 1-byte per element: $\approx 10\,\text{kB}$

Eigenvalues and Eigenvectors of the implicit covariance-matrix

Die Eigenwerte von $\mathbf{Q}$ und ihre korrespondierenden Eigenvektoren lassen sich aus den Eigenwerten und -vektoren von $\widetilde{\mathbf{Q}}$ berechnen:

$$\begin{aligned} \lambda_i &= \widetilde{\lambda}_i \\ \vec{e}_i &= \widetilde{\lambda}_i^{-\frac{1}{2}} \mathbf{P} \widetilde{\vec{e}}_i \end{aligned}$$

Klassifikation und PCA (1)

Beispiel aus „Turk and Pentland: Eigenfaces for Recognition “

- Projektion der Klassen in den Eigenraum:

$$\vec{\Omega}_c = \mathbf{A}^T(\vec{x}_c - \vec{\mu}) \qquad c = 1, \ldots, k \tag{6}$$

- Bestimme maximalen Abstand zwischen Klassen:

$$\theta_l = \frac{1}{2} \max_{j,k} \{\|\vec{\Omega}_j - \vec{\Omega}_i\|\} \qquad j, i = 1, \ldots, k \tag{7}$$

Klassifikation und PCA (2)

- Klassifikation eines neuen Bildes $\vec{x}$:
  - Projektion in den Eigenraum :

$$\vec{\Omega} = \mathbf{A}^T(\vec{x} - \vec{\mu}) \tag{8}$$

  - Klassenabstand bestimmen:

$$\epsilon_c = \|\vec{\Omega} - \vec{\Omega}_c\| \tag{9}$$

  - Bestimmung des Abstands zwischen Eingabe und Rückprojektion:

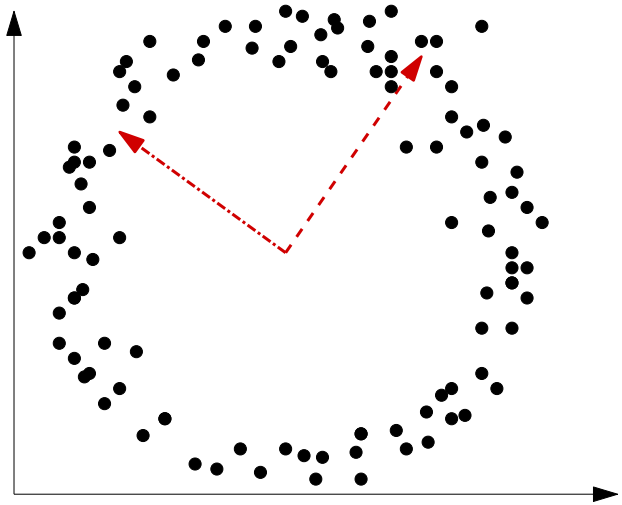
$$\epsilon = \|\vec{x} - \vec{x}_r\| \quad \text{mit} \tag{10}$$

$$\vec{x}_r = \mathbf{A}\vec{\Omega} + \vec{\mu} \tag{11}$$

Klassifikation und PCA (3)

- Klassifikation
  - Falls $\epsilon \geq \theta_l$
    Input ist kein Gesicht.
  - Falls $\epsilon < \theta_l$ und $\forall c, \epsilon_c \geq \theta_l$
    Input ist ein unbekanntes Gesicht.
  - Falls $\epsilon < \theta_l$ und $\epsilon_{c^*} = \min_c\{\epsilon_c\} < \theta_l$
    Input enthält ein Gesicht von Person c^*

Problemfälle für PCA

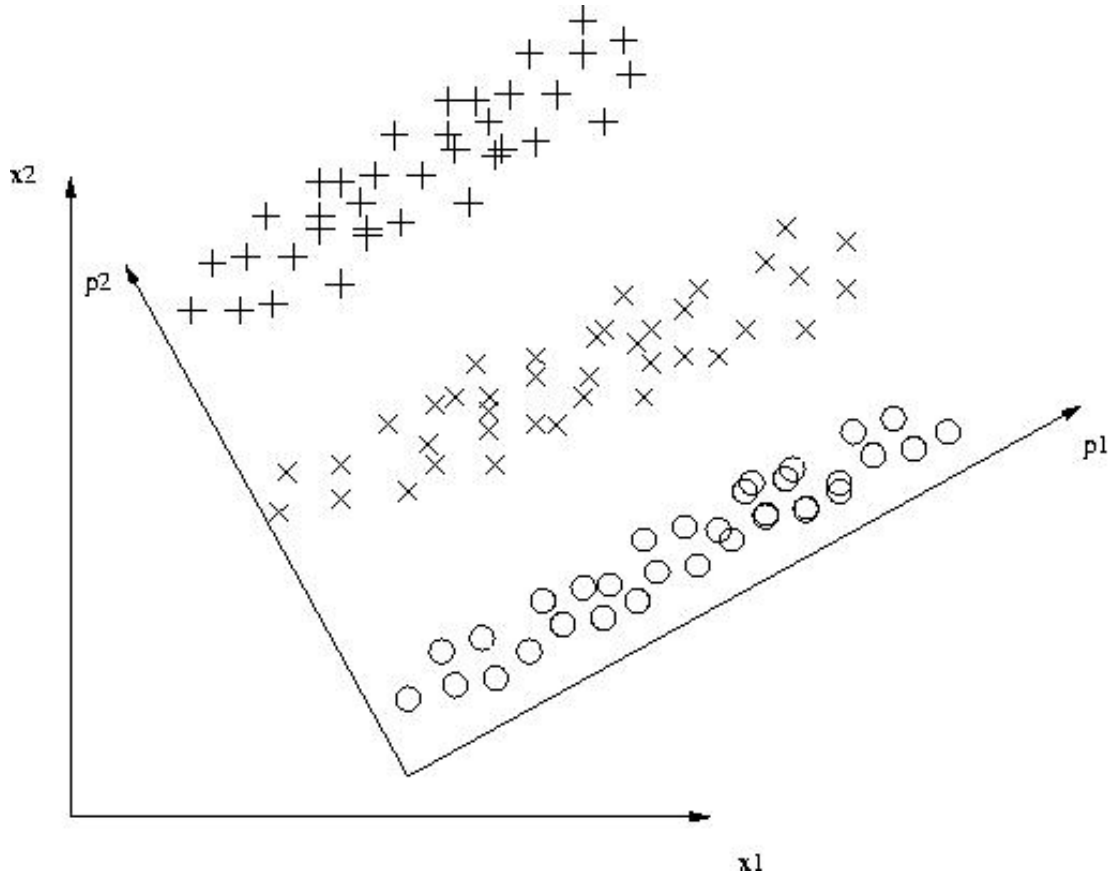


- Daten werden nach Varianz sortiert
- implizite Annahme einer Gauss-Verteilung

- aber ungeeignet für Daten mit anderer Verteilung
- z.B. bi-modale oder multi-modale Verteilungen

Problemfälle für PCA

Adidas-Problem:



PCA und Neuronale Netze

Die erste Hauptkomponente der PCA lässt sich auch über die Hebb-Regel von einem einschichtigen Perzeptronnetzwerk lernen. Regel von Yuille et al. :

$$\begin{aligned} \vec{y} &= \sum \omega_i \xi_j = \mathbf{w}^T \vec{x} = \vec{x}^T \mathbf{w} \\ \Delta\omega_j &= \eta(Vx_j - \omega_j |w|^2) \end{aligned}$$

Der Gewichtsvektor $\vec{w}$ zeigt im Konvergenzfall in die Richtung des Eigenvektors der Kovarianzmatrix mit der größten Varianz.

Oja-Algorithmus

- *start*: Menge X von n-dimensionalen Eingabevektoren
- Vektor w zufällig initialisiert ($w \neq 0$)
- Lernrate γ mit $0 < \gamma \leq 1$

- *update*: wähle zufälligen Vektor x aus X
- berechne Skalarprodukt $\Phi = x \cdot w$
- neuer Gewichtsvektor ist $w + \gamma\Phi(x - \Phi w)$
- gehe zu *update*, reduziere γ

(Oja 1982, Rojas 5.3.1)

Independent Component Analysis

- Folie wird nachher komplettiert
- ICA Webseite: http://www.cis.hut.fi/projects/ica/

Independent Component Analysis (1)

Folgender Sachverhalt verdeutlicht die Arbeitsweise der Independent Component Analysis (kurz ICA):

In einem Raum sind zwei Lautsprecher aufgestellt, die zwei verschiedene Tonsignale $s_1(t)$ und $s_2(t)$ ausgeben.

Die zwei Tonsignale werden von zwei verschiedenen Mikrofonen an verschiedenen Stellen aufgenommen.

Independent Component Analysis (2)

Die beiden Mikrofone nehmen zwei unterschiedliche Mischungen $x_1(t)$ und $x_2(t)$ der Originaldaten auf, wobei:

$$x_1(t) = a_{11}s_1(t) + a_{12}s_2(t) \tag{12}$$

$$x_2(t) = a_{21}s_1(t) + a_{22}s_2(t) \tag{13}$$

Die Faktoren $a_{11}, a_{12}, a_{21}, a_{22}$ und die Originaldaten sind aus der Perspektive der Mikrofone unbekannt. In Vektornotation kann die obige Gleichung allgemein wie folgt geschrieben werden:

$$\vec{x}(t) = \mathbf{A} \cdot \vec{s}(t) \tag{14}$$

wobei $\vec{x}(t), \vec{s}(t) \in R^n$ sind und $\mathbf{A}$ eine $n \times n$-Matrix ist.

Independent Component Analysis (3)

Falls die Originaldaten folgenden Bedingungen genügen, kann die Mischungsmatrix **A** mit der ICA bestimmt werden:

1. Die Originaldaten müssen statistisch unabhängig sein
2. Die Originaldaten müssen stationär sein
3. Maximal eine Originalquelle darf gaußverteilt sein

Die Originaldaten können aus der inversen Mischungsmatrix $W = A^{-1}$ der Matrix A berechnet werden:

$$\mathbf{s}(t) = W \cdot \mathbf{x}(t) \tag{15}$$

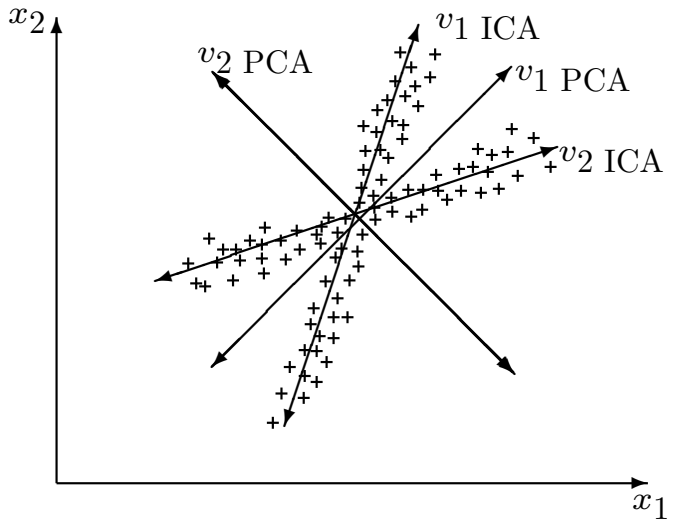
Independent Component Analysis (4)

Die Wiederherstellung der ursprünglichen Daten unterliegt allerdings zwei Einschränkungen:

1. Die Energie oder die Varianzen der einzelnen Originalquellen können nicht wiederhergestellt werden.
2. Die Reihenfolge der Originalsignale $s_i(t)$ kann ebenfalls nicht rekonstruiert werden.

Vergleich ICA mit PCA

Gezeichnet sind die Vektoren zweier korrelierter, nicht-normalverteilter Zeitserien. Die PCA projiziert auf eine Basis, deren Achsen orthogonal sind, wobei die 1. Achse in Richtung der größten Varianz zeigt. Die Achsen der ICA müssen nicht orthonormal sein, so daß die Varianz für beide Achsen maximiert werden kann. Dieses führt zu einer günstigeren Dekorrelation.



Dimensionsreduktion mit der ICA (1)

Bei der PCA ist die Reihenfolge der zu selektierenden Principal Components durch die Größe der Eigenwerte vorgegeben. Für die gefundenen Independent Components existiert diese Reihenfolge nicht.
Es gibt deshalb verschiedene Ansätze, die ICA zur Dimensionsreduktion zu benutzen:

- Ordnung der Zeilen in der Mischungsmatrix A nach der euklidischen L_2-Norm. Die Zeilen von A mit der größten L_2-Norm haben die größte Energie und somit haben die Quellen die zu diesen Zeilen gehören, einen größeren Einfluß auf die beobachteten gemischten Signale $\mathbf{x}(t)$.

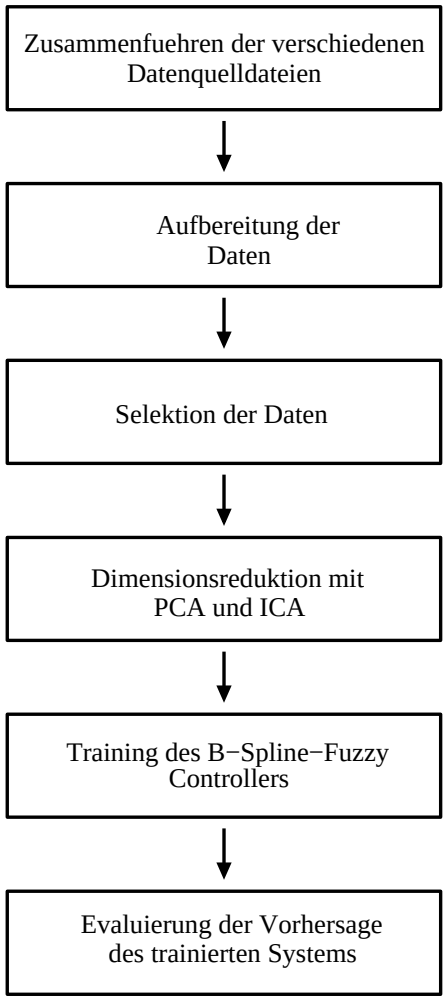
Dimensionsreduktion mit der ICA (2)

- Selektion der m Quellen mit der größten Amplitude, also der Komponenten $s_i(t)$ der Vektoren $\mathbf{s}(t)$ mit der größten L_∞-Norm.
- Eine weitere Möglichkeit ist die Berechnung der Independent Component Analysis auf Principal Components. Bei diesem Verfahren werden die n-dimensionalen Eingangsvektoren $\mathbf{x}(t)$ mit Hilfe der PCA auf m-dimensionale Vektoren reduziert. Auf diesen reduzierten Vektoren wird dann die ICA berechnet. (Geschwindigkeitsvorteil).
- Als weitere Möglichkeit bietet sich Input Selection an, um die Komponenten mit den interessantesten Informationen zu selektieren.

Anwendungsgebiete ICA

- Filtern von MEG/EEG Daten
- Reduzierung von Rauschen in natürlichen Bildern
- Telekommunikation
- Auffinden von versteckten Faktoren in Finanzdaten

Aufbau des Vorhersagesystems



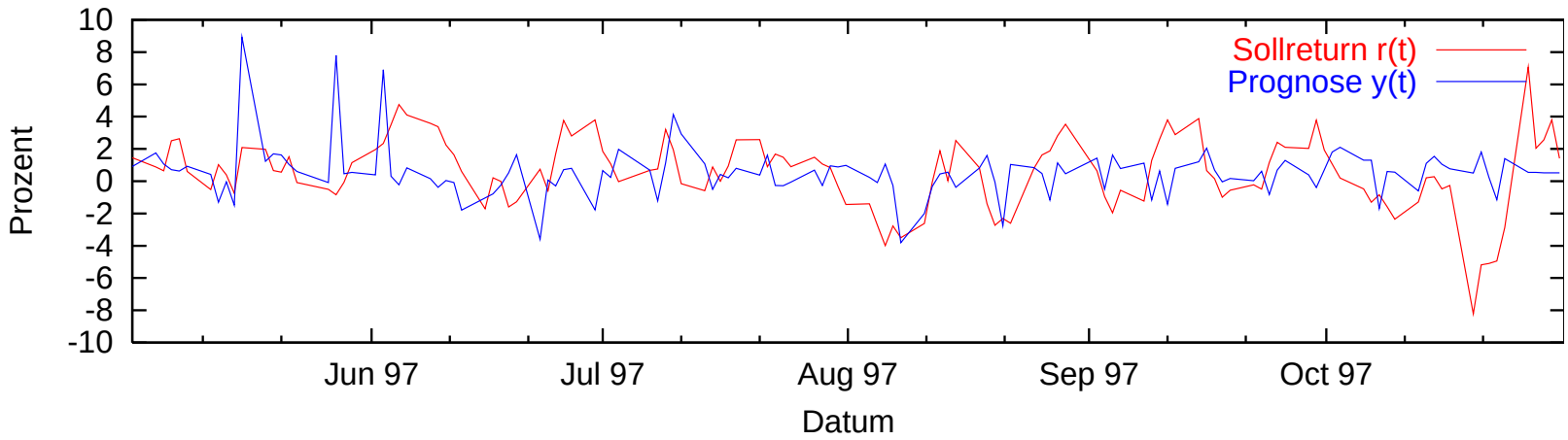
Vorhersage ohne fundamentale Daten (1)

Bei der ersten Testreihe soll für zwei verschiedene Paare von Trainings- und Testmengen das unterschiedliche Verhalten des Vorhersagesystems für die unterschiedlichen Methoden der Dimensionsreduktion analysiert werden.
Der Vorhersagehorizont h ist 5 und die drei Paare von Trainings- und Testmengen sind Standardintervalle. Für jeden Tag wurde der Vektor $\mathbf{x}(t)$ aus den Returns in Prozent $r_h(t)$ für den S&P 500 Index wie in Gleichung 16 berechnet.

$$\mathbf{x}(t) = (r_{-1}(t), r_{-2}(t), \ldots, r_{-50}(t))^T \tag{16}$$

Vorhersage ohne fundamentale Daten (2)

Als Dimensionsreduktionsverfahren werden in dieser Testreihe die PCA, die ICA mit der L_2-Norm als Selektionskriterium gegenübergestellt.
Die Sollreturns $r(t)$ und die prognostizierten Returns $y(t)$ für die Vorhersage mit der ICA:



Vorhersage mit fundamentalen Daten (1)

Mit dieser Methode soll herausgefunden werden, ob sich die Vorhersagen, basierend auf der Zeitreihenanalyse mit ICA, durch Hinzunahme der fundamentalen Daten verbessern lassen.
Der Vektor $\mathbf{p}(t)$ besteht hier aus 6 mittels L_2-Norm berechneten Independent Components, die aus k-Day-Returns des S&P 500 Index berechnet wurden.
Als 7. Komponente kommt zusätzlich der jeweils aktuelle Wert aus einer der 10 Zeitserien f_i mit den fundamentalen Daten hinzu. Der Vorhersagehorizont h ist 5 und der Vektor $\mathbf{x}(t)$ wie in Gleichung.

$$\mathbf{x}(t) = (r_{-1}(t), r_{-2}(t), \ldots, r_{-50}(t), f_i(t))^T. \tag{17}$$

Vorhersage mit fundamentalen Daten (2)

Der Vektor $\mathbf{p}(t)$ wird mittels ICA aus den Return-Komponenten $r\__i(t)$ des Vektors $\mathbf{x}(t)$ berechnet. Die letzte Komponente $f_i(t)$ geht direkt in den Vektor $\mathbf{p}(t)$ ein.
In einer der folgenden Testreihen wird dann die Performance des Vorhersagesystems mit den Independent Components der Zeitreihenanalyse und den Kombinationen aller vielversprechenden Kandidaten untersucht.
Für das Vorhersagesystem wurden US-amerikanische Indikatoren und Aktienindizes verwendet. Das System sollte sich aber auch auf andere Märkte übertragen lassen, sofern für die jeweiligen Märkte genügend Datenmaterial öffentlich zugänglich ist.

Vorhersage mit fundamentalen Daten (3)

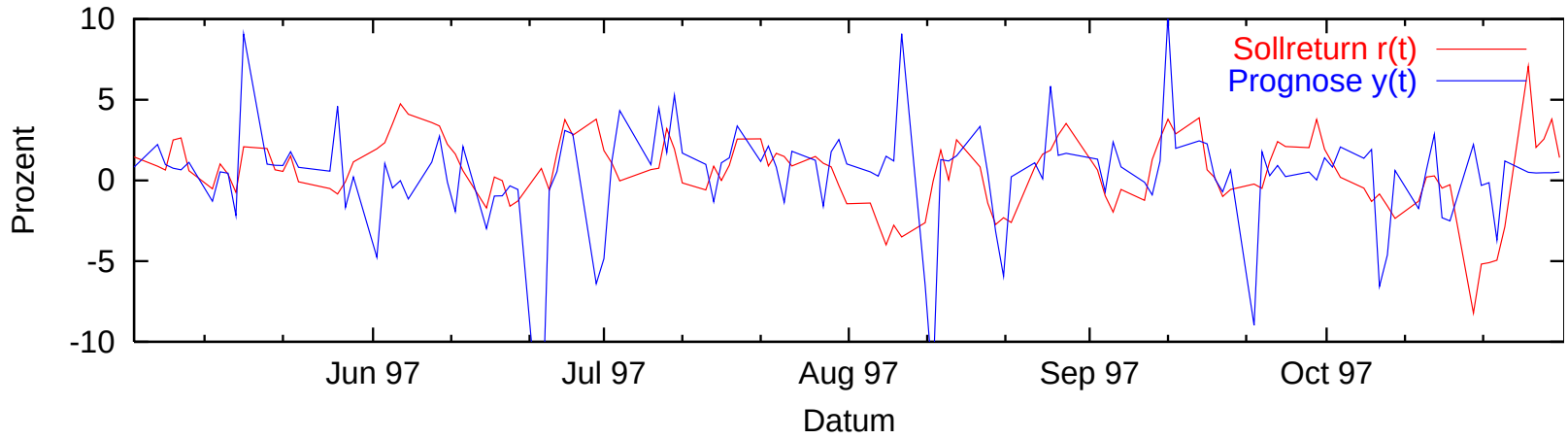
Das entwickelte Prognosemodell ermöglicht die Kombination von Zeitreihenanalyse mit der Analyse fundamentaler Daten. Die folgende Tabelle gibt eine Übersicht über die benutzten fundamentalen Daten:

Name	Kürzel
U.S. Weekly Leading Index	wli
U.S. Coincident Leading Index	usci
M3 Money Stocks	m3
Exchange Rate Swiss Franc/US$	exszus
Consumer Price Index	cpiaucsl
Civilian Unemployment Rate	unrate
Consumer Sentiment Index	umcsent
Real Disposable Personal Income	dspic96
NAPM Manufact. Composite Index	napm
Manufacturers' New Orders	neworder

Überblick über die fundamentalen Daten

Vorhersage mit fundamentalen Daten (4)

Die Geldmenge M3 als zusätzlicher Eingang hat die Vorhersage auf allen drei Intervallen verbessert. Die Indikatoren "Manufacturers New Orders" (neworder), "Arbeitslosenquote" (unrate) und "Real Disposal Personal Income" (dspic96) verbessern die Vorhersage bezüglich des "Mean Profit per Trade" Kriteriums auf jeweils zwei der drei Intervalle.



Die Sollreturns $r(t)$ und die prognostizierten Returns $y(t)$ für die Vorhersage aus 6 Independent Components und den Auftragseingängen (neworder) als zusätzlichem Eingang auf der Testmenge des Intervalls 2.