# Maschinelles Lernen und Datenanalyse

*In der Mess- und Prüftechnik* PD Stefan Bosse

Universität Bremen - FB Mathematik und Informatik

# Daten- und Dimensionalitätsreduktion

Datenreduktion ist ein wichtiger Schritt in der Datenvorverarbeitung für ML

Ziel: Reduktion der Datenvariablen (Attribute) → Dimensionalitätsreduktion pro Instanz

Ziel: Reduktion der Dateninstanzen (durch kleine Anzahl von Repräsentanteninstanzen) → Datenvolumenreduktion

# Motivation für Datenreduktion

1. Die Daten sind sowohl hinsichtlich Dimensionalität des Eingabevektors $\boldsymbol{x}$ als auch hinsichtlich der Anzahl von Dateninstanzen $|\mathbf{D}|$ sehr groß

2. Es gibt Redundanzen

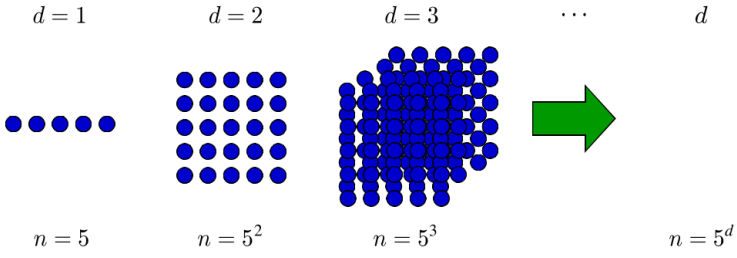
   - a. Von Datenvariablen (lineare Abhängigkeit)
   - b. Von Dateninstanzen (Redundanz und Überlappung)
   - Aber: Reduktion bei b. kann die geforderte Datenvarianz verschlechtern!

3. Trennung von wenig aussagekräftigen (schwachen) von aussagekräftigen (starken) Variablen

Wenn die Dimensionalität der Eingabedaten $\boldsymbol{x}$ zunimmt, wird jedes Lernproblem immer schwieriger und rechenintensiver!

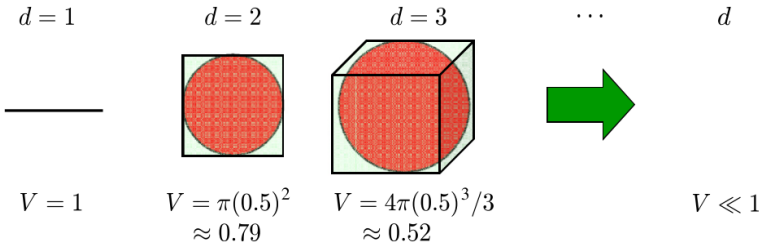
- Beispielsweise werden in regelmäßigen Abständen 5 Punkte von [0,1] abgetastet.
  - Das sammeln von Proben auf die gleiche Weise im d-dimensionalen Raum erfordert $5d$-Punkte, die exponentiell in Bezug auf $d$ wachsen.

[4]



Ein weiteres Problem bei hochdimensionalen Daten ist, dass unsere geometrische Interpretation von Daten irreführend sein kann.

- Betrachtet man zum Beispiel die ausfüllende Hypersphäre des Einheishyperwürfels im d-dimensionalen Raum
  - Wenn $d = 1$ ist, ist das Volumen einer eingepassten Hypersphäre 1, was dem Hyperwürfel entspricht.
  - Wenn $d$ auf 2 und 3 erhöht wird, ist das Volumen des Hyperkubus immer noch 1, aber das Volumen der Hypersphäre ist ungefähr 0,79 bzw. 0,52.




[4]

- Unsere geometrische Intuition ist falsch,
  - da obwohl die eingepasste Hypersphäre kleiner als der Hyperwürfel ist, ist die Hypersphäre nicht extrem klein.
  - Wenn jedoch *d* weiter erhöht wird, ist das Volumen des Hyperkubus immer noch 1, aber das Volumen der ausfüllenden Hypersphäre neigt zu 0.
  - Dies bedeutet, dass, wenn *d* groß ist, die Hypersphäre fast vernachlässigbar ist.

# Verfahren und Methoden

## Lineare Algebra

- Bestimmung von Eigenvektoren und Hauptkomponenten mit der **Principle Component Analysis**
  - Abbildung des Eingabedatenraums auf einen niedrigdimensionaleren Datenraum **unter Beibehaltung der Datenrepräsentation**
  - Ziel: Reduktion der Attribute, Beibehaltung der Dateninstanzen

Die PCA führt selber keine Reduktion durch! Transformation mit den Hauptkomponenten kann zur Reduktion führen.

## Clustering

- Bestimmung von Gruppen von Dateninstanzen (mit geometrischer Gemeinsamkeit durch "Nähe") durch dichtebasierte Clusteringverfahren (**DBSCAN**)
  - Ziel: Reduktion der Instanzmenge durch wenige repräsentative Instanzen; Beibehaltung der Datenvariablen
  - Repräsentative Instanzen können (aber müssen nicht) mehrheitlich die Instanzen der Gruppe mit einem bestimmten Wert der Zielvariable (sofern kategortisch oder intervallkodiert) verknüpfen
  - Hohe Zielvariablenwertdiversität in Gruppen zeigt schwache Korrelation von $\boldsymbol{x}$ mit $\boldsymbol{y}$!

# Lineare Dimensionalitätsreduktion

Lineare Dimensionalitätsreduktion transformiert die ursprünglichen *d*-dimensionalen Dateninstanzen $\{x_\mathrm{i}\}^\mathrm{n}$ in nieder *m*-dimensionale Ausdrücke $\{z_\mathrm{i}\}^\mathrm{n}$ durch eine lineare Transformation $\boldsymbol{T} \in \mathbb{R}^{m \times d}$

$$z_i = \mathbf{T} x_i$$

- $\boldsymbol{T}$ ist die Transformationsmatrix, *z* der reduzierte Datenvektor (pro Dateninstanz *i*) und *x* der ursprüngliche Datenvektor.

[4]

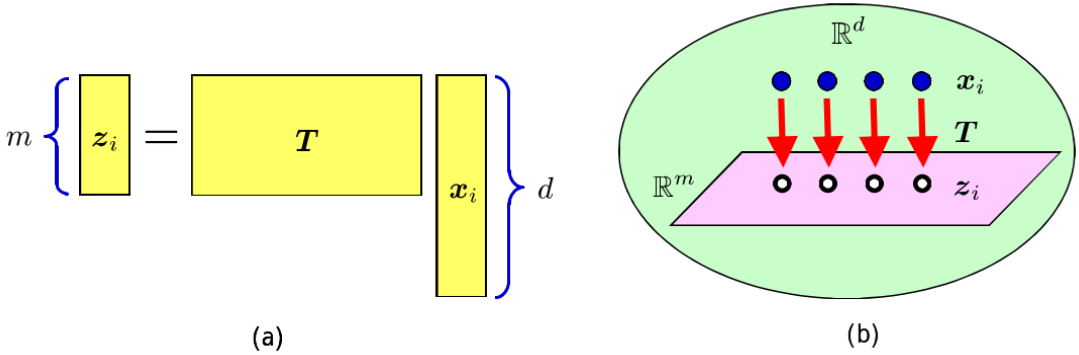


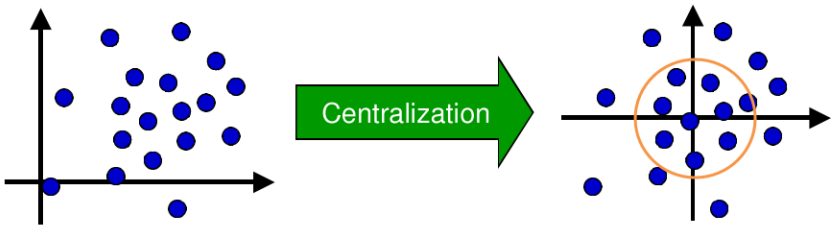

Abb. 1. (a) Lineare Dimensionalitätsreduktion (b) Projektion von Daten auf einen linearen Subraum

# Zentrierung von Daten

- Verschiebung der Daten in Richtung "Koordinatenursprung"

$$x_i \leftarrow x_i - 1/n \sum_j x_j$$

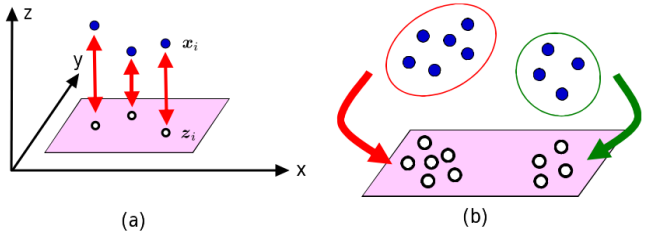
[4]



# Unüberwachte Dimensionsreduktion

## Hauptkomponentenanalyse (PCA)

- Reduktion der Datendimesionalität unter Beibehaltung der intrinsischen Information der Daten und der Datenstruktur




[4]

Abb. 2. PCA: (a) Reduktion der Dimensionalität 3 → 2 unter Beibehaltung der ursprünglichen geometrischen Eigenschaften der Punkte zueinander (Position) und der Gruppenzugehörigkeit (b)

- Das bedeutet dass $\boldsymbol{z}_\mathrm{i}$ eine **orthogonale Projektion** von $\boldsymbol{x}_\mathrm{i}$ ist
  - D.h. $\boldsymbol{T}\,\boldsymbol{T}^\mathrm{T} = \boldsymbol{I}_\mathrm{m}$ mit $\boldsymbol{I}_\mathrm{m}$ als Identitätsmatrix und a $^T$ die transponierte Matrix
- Die "Distanz" zwischen $\boldsymbol{x}$ und $\boldsymbol{z}$ kann aber (als Fehler) nicht unmittelbar bestimmt werden
  - Daher wird $\boldsymbol{z}$ wieder in den ursprünglich d-dimensionalen Datenraum durch $\boldsymbol{T}^\mathrm{T}$ zurück transformiert
- Die euklidische Distanz ist dann gegeben durch die totale statistische Streumatrix $\boldsymbol{C}$ und der Spur einer Matrix *tr*:

$$\sum_j ||\mathbf{T}^T\mathbf{T}x_i - x_i||^2 = -tr(\mathbf{T}\mathbf{C}\mathbf{T}^T) + tr(\mathbf{C})$$

- mit:

$$\mathbf{C} = \sum_j x_j x_j^T$$

- D.h. PCA ist ein Optimierungsproblem mit:

$$\max_{\mathbf{T}} tr(\mathbf{T}\mathbf{C}\mathbf{T}^T)$$

- Es gibt eine globale Lösung:

$$\mathbf{T} = (\mathbf{e}_1, \ldots, \mathbf{e}_m)^T$$

- mit $e_\mathrm{i}$ als Eigenvektor der Matrix $\boldsymbol{C}$ mit den Eigenwerten $\lambda_1 \geq \lambda_2 \geq .. \geq \lambda_\mathrm{d} \geq 0$.

- D.h. es gilt:

$$\mathbf{Ce} = \lambda \mathbf{e}$$

- Es wird "kleine" und "große" Eigenvektoren geben

> Die Transformationsmatrix $\boldsymbol{T}$ der PCA ist also eine orthogonale Projektion in einen Subraum der durch die "großen" Eigenvektoren aufgespannt wird,

- Die kleinen Eigenvektoren werden daher entfernt
- Und: PCA transformierte Variablen sind unkorreliert!

- PCA liefert aber i.A. keine Struktureigenschaften wie Cluster

[17]

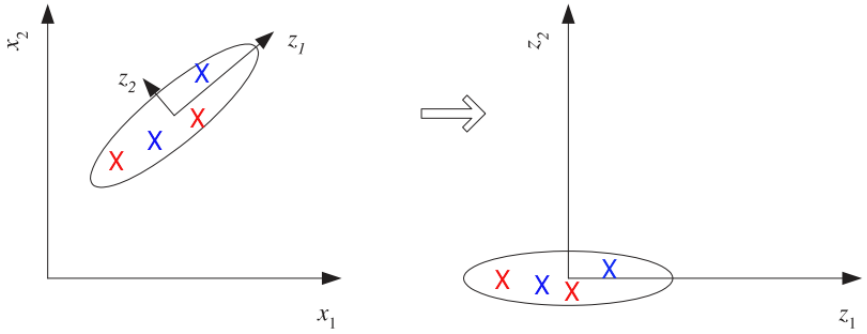



Abb. 3. Weiteres Beispiel von PCA: Die Analyse der Hauptkomponenten zentriert die Dateninstanzen und dreht dann die Achsen, um Sie mit den Richtungen der höchsten Varianz in Einklang zu bringen. Wenn die Varianz auf $z_2$ klein ist, kann Sie ignoriert werden und wir haben eine Dimensionalitätsreduktion von zwei auf eins.

# ML.pca

- Das ML Framework stellt ein PCA Modul zur Verfügung:
- Die Daten $\boldsymbol{D}$ müssen in Matrixform vorliegen (keine Recordtabellen)

1. Eigenvektoren berechnen

```
ML.pca.getEigenVectors = function(data:number [][]) → {
  eigenvalue: number, vector: number []
} [];
```

2. Transformation der Datentabelle berechnen

```
ML.pca.computeAdjustedData =
  function (data:number [][],
            eigenVector1:{},eigenVector2?:{},..)
→ {
  adjustedData: number [][],
  formattedAdjustedData: number [][],
  avgData: number [][],
  selectedVectors: number [][]
}
```

- Achtung: Das Datenformat von *adjustedData* ist transponiert und muss für eine Tabelle rekonstruiert werden

3. Rekonstruktion der Datentabelle aus der reduzierten Tabelle

```
ML.pca.computeOriginalData = function(
  formattedAdjustedData,
  selectedVectors,
  avgData) → {
  formattedOriginalData,
}
```

# PCA Beispiel

## Web WorkShell Live

CLEAR LOAD + - data eigen adjust re

# Lokalitätsbewahrende Projektion

- Ähnlichkeit zwischen Instanzen $\boldsymbol{x}_\mathrm{i}$ und $\boldsymbol{x}_\mathrm{j}$ wird untersucht und mit einem Wert $0 \leq W_{\mathrm{i,j}} \leq 1$ ausgedrückt.

## Ähnlichkeitsfunktionen

**Gaußfunktion**

$$W_{i,j} = exp(-\frac{||x_i - x_j||^2}{2t^2})$$

($t$ ist ein Anpassungsparameter)

**k-Nächster Nachbar (kNN)**

$$W_{ij} = \begin{cases} 1, (x_i \in N_k(x_j) \vee x_j \in N_k(x_i)) \\ 0 \end{cases}$$

- $N_{\mathrm{k}}(\boldsymbol{x})$ ist die Menge aller Nachbarschaftsinstanzen ($k$ ist die Anzahl der Menge)
- $k$ ist ein Anpassungsparameter der die Lokalität einstellt (Reichweite)

## Transformation

$$\min_T \sum_{i,j} W_{ij} ||\mathbf{T}x_i - \mathbf{T}x_j||^2$$

- Problem: ***T***=0 ist eine Lösung, aber nutzlos!
- Daher weitere Randbedingung für die Minimierung (Anpassung von ***T***):

$$\mathbf{TXDX}^T\mathbf{T}^T = \mathbf{I}_m$$

$$\mathbf{X} = (x_1, .., x_n), \mathbf{D}_{ij} = \begin{cases} \sum_k W_{ik}, (i = j) \\ 0, (i \neq j) \end{cases}$$

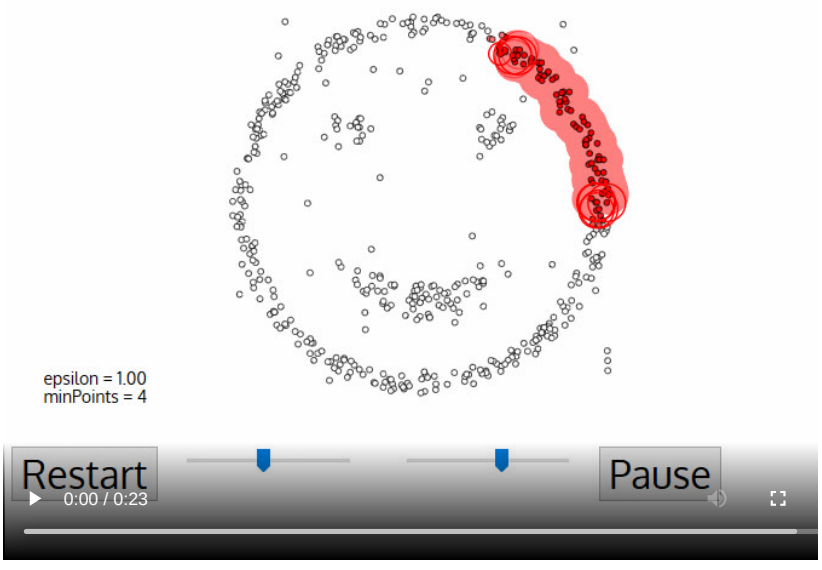
# Dichtebasiertes Clustering

- kNN kann ebenso als ein ML Clusterer mit Inferenz auf unbekannten Dateninstanzen eingesetzt werden
- Dichtebasierte Clusteringmethoden können primär für die Datenreduktion eingesetzt werden
    - Reduktion von einer Menge von Dateninstanzen auf Gruppen (wobei die Instanzen erhalten bleiben, aber in jeder Gruppe reduziert werden können)

# DBSCAN

> Dichtebasiertes Räumliches Clustering mit Rauschen (DBSCAN) ist ein Basisalgorithmus. Es kann Cluster unterschiedlicher Formen und Größen aus einer großen Datenmenge entdecken, die Rauschen und Ausreißer enthält.

[www.kdnuggets.com/2020/04/dbscan-clustering-algorithm-machine-learning.html]



## Algorithmische Schritte für DBSCAN Clustering

- Der Algorithmus iteriert über alle Punkte indem er willkürlich einen Punkt im Datensatz auswählt (bis alle Punkte besucht wurden)
- Wenn es mindestens *minPoint* - Punkte in einem Radius von ε bis zu dem Punkt gibt, werden alle diese Punkte als Teil desselben Clusters betrachet.
- Die Cluster werden dann erweitert, indem die Nachbarschaftsberechnung für jeden Nachbarpunkt rekursiv wiederholt wird.
- Eventuell gibt es Beschränkungen bezüglich der Dimension der Dateninstanzen (typisch 2: Clustering von Bildpunkten)

# Wahl der Parameter

> Jede Data Mining Aufgabe hat das Problem der Parameterwahl.

- Jeder Parameter beeinflusst den Algorithmus auf bestimmte Weise. Für DBSCAN werden die Parameter ε und *minPts* benötigt.

**minPts**

Das Minimum von *minPts* kann aus der Dimension des Datensatzes $\boldsymbol{D}$ abgeleitet werden, so dass minPts ≥ D + 1 gilt (D=|$\boldsymbol{D}$|).

**ε**

Der Wert für ε kann dann unter Verwendung eines k-Entfernungsgraphen ausgewählt werden, der den Abstand zum $k = minPts$-1 nächsten Nachbarn zeichnet, der vom größten zum kleinsten Wert geordnet ist.

- Bei guten Werten von ε gibt es eine ausgewogene Verteilung der Dateninstanzen in Gruppen
  - Wenn ε viel zu klein gewählt wird, wird ein großer Teil der Daten nicht gruppiert, während für
  - einen zu hohen Wert von ε Cluster zusammengeführt werden und sich die Mehrheit der Objekte im selben Cluster befinden.

Aufgrund *minPts* Randbedingung werden u.U. nicht alle Instanzen gruppiert!

Beispiel DBSCAN

# Web WorkShell Live

CLEAR LOAD + - data pca1 pca2 clus

# Zusammenfassung

- Datenreduktion ist ein wichtiger Schritt in der Datenvorverarbeitung
  - Datenreduktion bedeutet die Reduktion der Datenmenge und/oder ihrer Dimensionalität
  - Datenreduktion ist bereits eine Merkmalsselektion (Reduktion auf wesentliche Attribute und Werte)
- PCA ist geeignet um numerische Eingabedaten in ihrer Dimensionalität zu reduzieren (Reduktion der Datenvariablen und Ersatz mit transformierten unter Verwendung der Hauptkomponentenvektoren aus der PCA!)
- DBSCAN ist geeignet um Gruppen von Dateninstanzen zu finden um schliesslich repräsentative Instanzen daraus zu ermitteln