Ostbayerische Technische Hochschule
Amberg-Weiden

Machine Learning

Prof. Dr. Fabian Brunner

<fa.brunner@oth-aw.de>

Amberg, 19. Dezember 2023

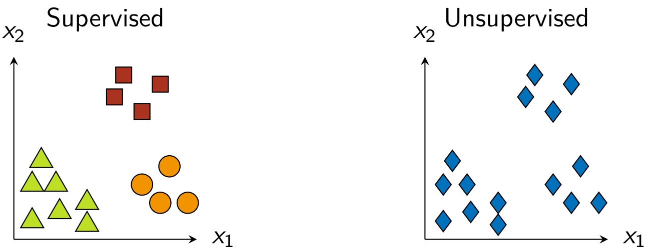
Supervised vs. Unsupervised Learning

Supervised Learning

- Gelabelte Trainingsdaten
- Erstellung von Prognosemodellen

Unsupervised Learning

- Ungelabelte Daten (oft besser verfügbar)
- Suche nach unbekannten Zusammenhängen



Clustering

Zielsetzung und Rahmenbedingungen

- Auffinden von Strukturen und Zusammenhängen in Daten
- keine korrekte Antwort (Labels) verfügbar

Wonach wird beim Clustering gesucht?

- Segmente (bzw. „Cluster“) der Daten
- Items in einem Cluster sollen ähnlich sein (Homogenität)
- Verschiedene Cluster sollen sich unterscheiden.
- Noch festzulegen: wie misst man Ähnlichkeit?

Typen von Clustering-Methoden

Hierarchisches Clustering

- Suche nach Hierarchien
- Baumbasierte Repräsentation der Objekte (z.B. Dendrogramm)

Partitionierungsmethoden

- Einteilung der Objekte in disjunkte Cluster
- Jedes Objekt gehört einem Cluster an
- Beispiel: k-Means-Clustering

Dichtebasierte Methoden

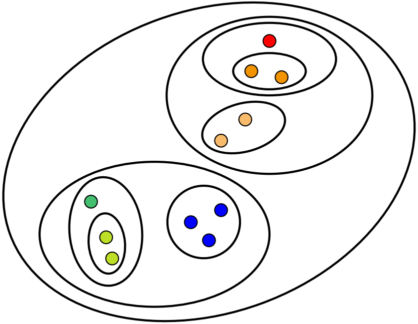
- Definition von Clustern mit Hilfe der Punktdichte
- Vorteil: Robustheit gegen Ausreißer/Rauschen
- Beispiel: DBSCAN-Algorithmus

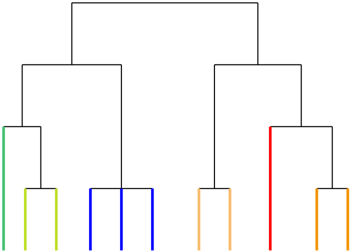
Fuzzy Clustering

- keine fixe Cluster-Zuordnung
- Bestimmung von Scores für Cluster-Zugehörigkeit
- Beispiel: Fuzzy-c-Means

Flaches vs. hierarchisches Clustering

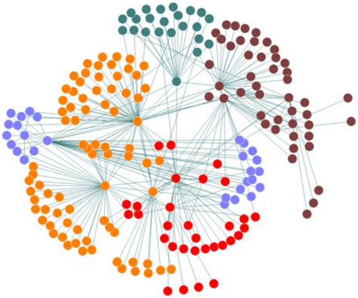
Mit Hilfe eines Dendrogramms (rechts) lassen sich hierarchische Cluster (links) darstellen.



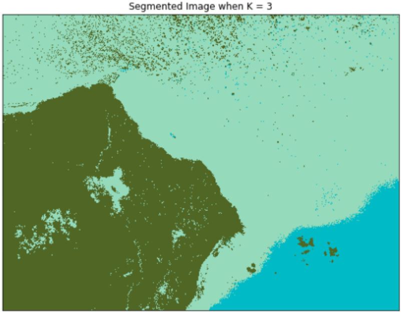


Anwendungen von Clustering

- Marketing (z.B. Kundensegmentierung)
- Bildsegmentierung
- Social Network Analyis
- Dokumentenklassifikation
- Feature-Generierung für Supervised Learning



Bildsegmentierung durch Clustering



Quelle: https://towardsdatascience.com/introduction-to-image-segmentation-with-k-means-clustering-83fd0a9e2fc3

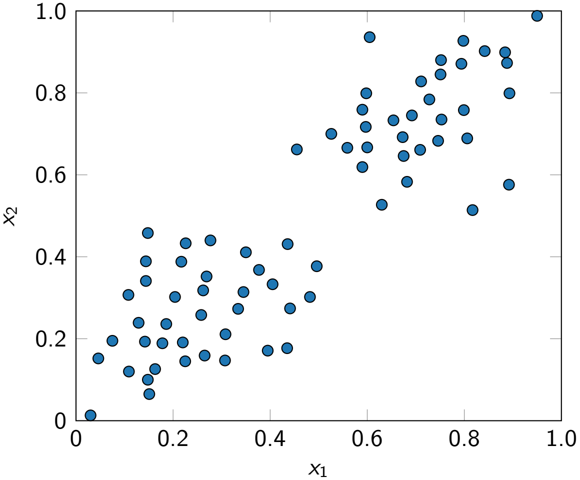
Das k-Means-Verfahren

Wir wollen den k-**Means-Algorithmus** untersuchen, das in der Praxis am weitesten verbreitet ist. Er wird auch **Lloyd-Algorithmus** genannt.

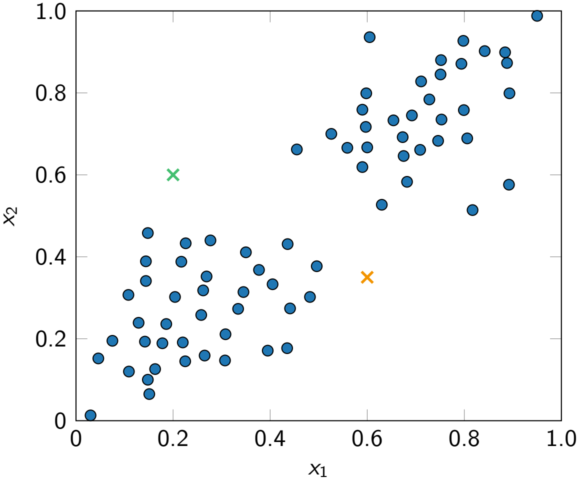
Grundidee und Eigenschaften:

- Lege zu Beginn die Anzahl der Cluster und zufällige Cluster-Zentren fest.
- Berechne die Clusterzugehörigkeit der einzelnen Punkte anhand des Abstands der Objekte zu den Clusterzentren.
- Berechne die Cluster-Zentren neu.

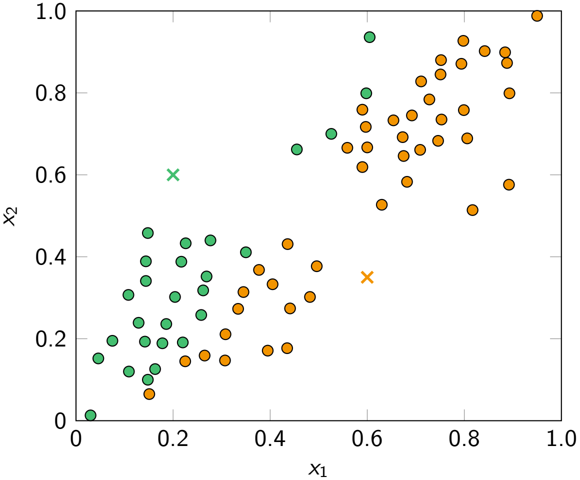
k-Means Clustering - Beispiel



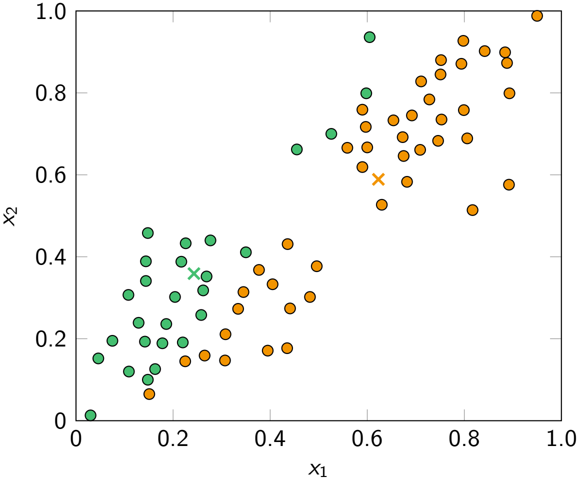
k-Means Clustering - Beispiel



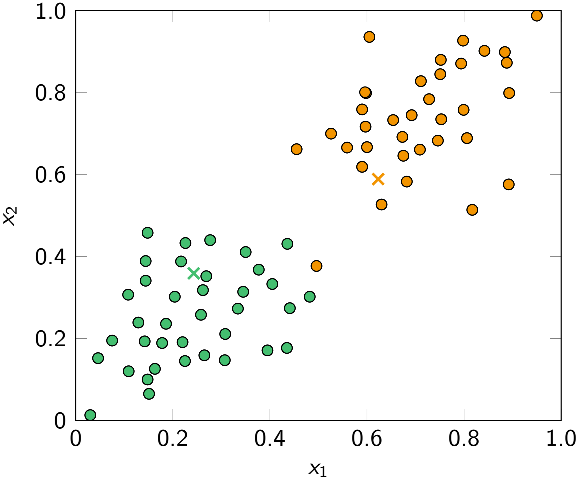
k-Means Clustering - Beispiel



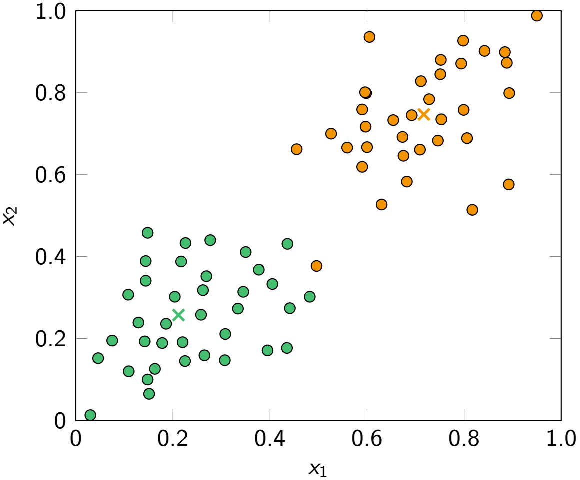
k-Means Clustering - Beispiel



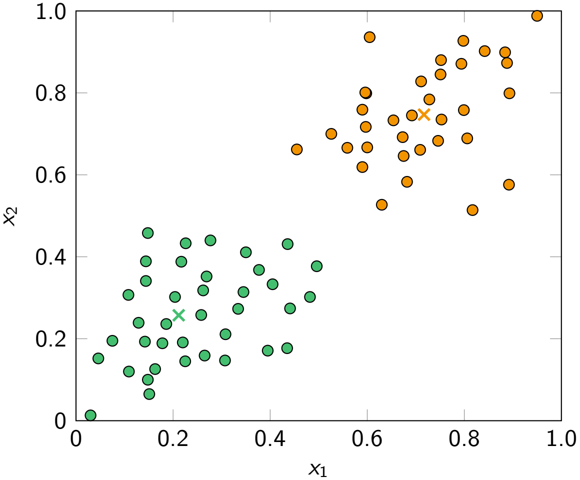
k-Means Clustering - Beispiel



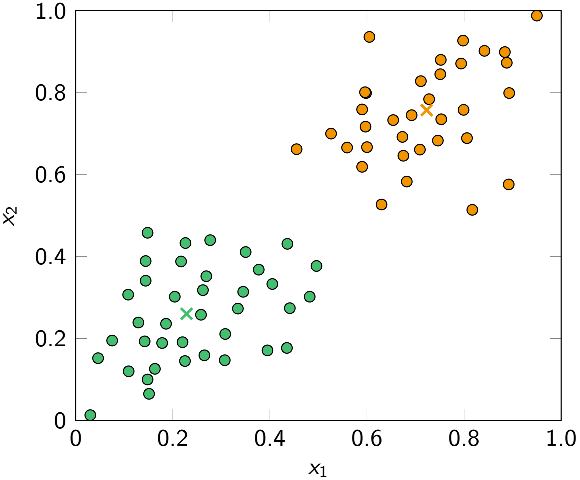
k-Means Clustering - Beispiel



k-Means Clustering - Beispiel



k-Means Clustering - Beispiel



k-Means Clustering

Notation

- Gegeben sei ein Datensatz S bestehend aus m Samples mit jeweils p numerischen Features: $S = \{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ mit $\boldsymbol{x}^{(i)} \in \mathbb{R}^p$ für $i = 1, \ldots, m$.
- Die Daten sollen in K diskunkte Cluster $C_1, \ldots, C_K$ partitioniert werden.

k-Means Algorithmus

1. **Initialisierung:** Bestimme K zufällige Cluster-Mittelpunkte $\mu_1^0, \ldots, \mu_K^0$.
2. **Cluster-Zuordnung:** ordne in der k-ten Iteration jeden Punkt $\boldsymbol{x}$ des Datensatzes S demjenigen Cluster zu, von dessen Mittelpunkt er den geringsten Abstand hat:
$$C_i^k := \{\boldsymbol{x} \in S \; : \; \mathrm{dist}(\boldsymbol{x}, \mu_i^k) \leq \mathrm{dist}(\boldsymbol{x}, \mu_j^k) \text{ für alle } j = 1, \ldots, K\} \; .$$
3. **Update der Mittelpunkte:** Berechne die Cluster-Mittelpunkte neu:
$$\mu_i^{k+1} = \frac{1}{|C_i^k|} \sum_{\boldsymbol{x} \in C_i^k} \boldsymbol{x} \; .$$

Die Schritte 2 und 3 werden solange wiederholt, bis sich keine Änderungen in den Zuordnungen mehr ergeben.

Wahl eines Distanzmaßes

- Wie bei der Nearest Neighbor-Klassifikation können verschiedene Distanzmaße zur Definition von k-Means Clustering verwendet werden. Diese müssen die folgenden Metrik-Eigenschaften besitzen:
  1. $\mathrm{dist}(\boldsymbol{x}, \boldsymbol{y}) \geq 0$ und $\mathrm{dist}(\boldsymbol{x}, \boldsymbol{y}) = 0 \;\Leftrightarrow\; \boldsymbol{x} = \boldsymbol{y}$ (positive Definitheit)
  2. $\mathrm{dist}(\boldsymbol{x}, \boldsymbol{y}) = \mathrm{dist}(\boldsymbol{y}, \boldsymbol{x})$ (Symmetrie)
  3. $\mathrm{dist}(\boldsymbol{x}, \boldsymbol{z}) \leq \mathrm{dist}(\boldsymbol{x}, \boldsymbol{y}) + \mathrm{dist}(\boldsymbol{y}, \boldsymbol{z})$ (Dreiecksungleichung)
- Häufigste Wahl ist die Euklidische Distanz

$$\mathrm{dist}(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{\|\boldsymbol{x} - \boldsymbol{y}\|_2^2} = \sqrt{\sum_{i=1}^{p} (x^{(i)} - y^{(i)})^2} \, .$$

  Diese wird im Folgenden betrachtet.
- Die Wahl des Distanzmaßes sollte abhängig von den Daten und der Anwendung getroffen werden.
- Für Zeichenketten könnte beispielsweise die Edit-Distanz herangezogen werden.
- Für Bit-Vektoren erscheint die Hamming-Distanz sinnvoll.

k-Means als Optimierungsaufgabe

Sei $c(i)$ der Index des Clusters, dem der Datenpunkt $\boldsymbol{x}^{(i)} \in S$ zugeordnet wird. Dann hat k-Means-Clustering das Ziel, die Gesamtvarianz

$$J(c(1), \ldots, c(m), \mu_1, \ldots, \mu_K) = \frac{1}{m} \sum_{i=1}^{m} \left\| \boldsymbol{x}^{(i)} - \mu_{c(i)} \right\|^2$$

zu minimieren, d.h. es wird die Lösung folgenden Optimierungsproblems bestimmt:

Optmierungsproblem bei k-Means-Clustering

$$\min_{\substack{c(1),\ldots,c(m) \\ \mu_1,\ldots,\mu_K}} J(c(1), \ldots, c(m), \mu_1, \ldots, \mu_K) \ .$$

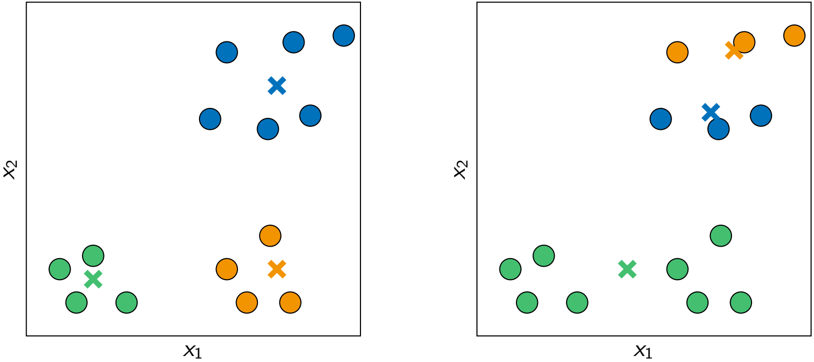
In jedem Iterationsschritt des k-Means-Verfahrens werden sowohl die Cluster-Zentren $\mu_1, \ldots, \mu_K$ als auch die Cluster-Zuordnungen $c(1), \ldots, c(m$ modifiziert.

k-Means Clustering in sklearn

```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans

X = ... #Definition des Datensatzes

kmeans = KMeans(n_clusters=15, random_state=0)
kmeans.fit(X)

#Clusterzurodnung für jeden Punkt in X
clusters = kmeans.predict(X)

# Ausgabe der Cluster-Zentren
print(kmeans.cluster_centers_)
```

Lokale Optima

- Das Ergebnis des k-Means-Clusterings hängt von den Startwerten ab.
- Es ist möglich, dass das Verfahren in einem lokalen Optimum terminiert.
- Folgendes Beispiel zeigt die berechneten Cluster für zwei verschiedene Start-Konfigurationen:



Initialisierung der Mittelpunkte

- Wähle K Trainingsdatenpunkte als Cluster-Zentren.
- Das Ergebnis des k-Means-Verfahrens hängt von den Initialwerten der Cluster-Zentren ab.
- Bei ungünstiger Wahl läuft das Verfahren in ein lokales Minimum.
- Um dies zu vermeiden kann man die Cluster-Zentren mehrfach zufällig initialisieren und die Cluster berechnen.
- Am Ende wird die Konfiguration gewählt, bei der der Zielfunktionswert $J(c(1), \ldots, c(m), \mu_1, \ldots, \mu_K)$ minimal ist.

Ex-post-Analyse der Cluster

Wie sehen die Cluster aus?

- Häufig ist man an einer Charakterisierung der Cluster interessiert.
- Beispiel Kundensegmentierung: wie sehen die Kundengruppen aus? Wodurch zeichnen sie sich aus? Was verbindet die Kunden in einer Kundengruppe? Worin unterscheiden sich verschiedene Kundengruppen?
- Um diesen Fragen nachzugehen, analysiert und vergleicht man die Cluster nachträglich („ex post“).

Ansätze zur ex-post-Analyse der Cluster

- Methoden der deskriptiven Statistik (univariat oder multivariat)
- Visualisierungen
- Training eines Entscheidungsbaums unter Verwendung der Clusternummer als Zielvariable. Anhand der Splits kann man ablesen, welche Merkmale für die einzelnen Cluster charakteristisch sind.
- Anwendung eines Modells des Supervised Learning mit Feature Selection, um die wichtigsten Features zu ermitteln, die bei der Clusterzuordnung eine Rolle spielen.

Beispiel: MNIST - Datensatz

Anwendung von k-Means-Clustering mit $K = 15$ auf den MNIST-Datensatz:

Geometrische Interpretation

Die Cluster beim k-Means Clustering haben die folgende Form

$$C_i = \{\boldsymbol{x} \in S : \mathrm{dist}(\boldsymbol{x}, \mu_i) \leq \mathrm{dist}(\boldsymbol{x}, \mu_j) \text{ für alle } j = 1, \ldots, K\} \ .$$

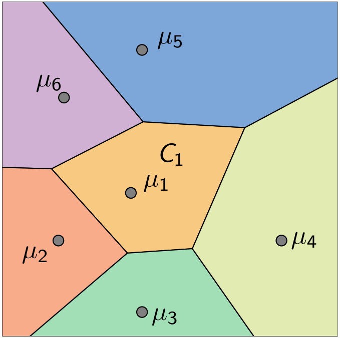
Wie wird der Raum durch die k Cluster-Zentren geometrisch partitioniert?

Geometrische Interpretation

Die Cluster beim k-Means Clustering haben die folgende Form

$$C_i = \{\boldsymbol{x} \in S : \mathrm{dist}(\boldsymbol{x}, \mu_i) \leq \mathrm{dist}(\boldsymbol{x}, \mu_j) \text{ für alle } j = 1, \ldots, K\} \ .$$

Wie wird der Raum durch die k Cluster-Zentren geometrisch partitioniert?

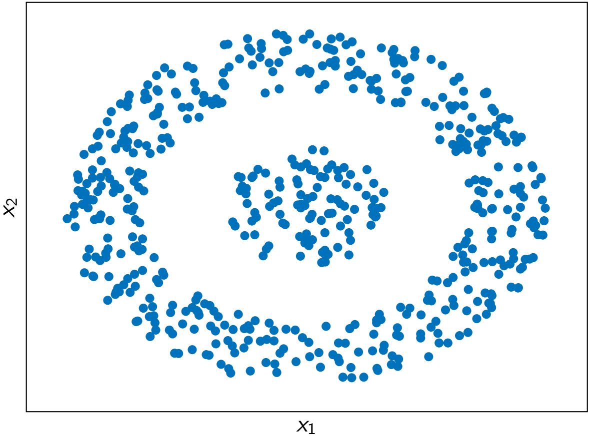


Voronoi-Zerlegung

- Punkte eines Clusters liegen in der vom Cluster-Zentrum erzeugten Voronoi-Zelle.
- Die Zuordnung eines neuen Objekts zu den bestehenden Clustern entspricht der Nearest Neighbor Klassifikation mit den Cluster-Zentren als Trainingsdatenpunkten.

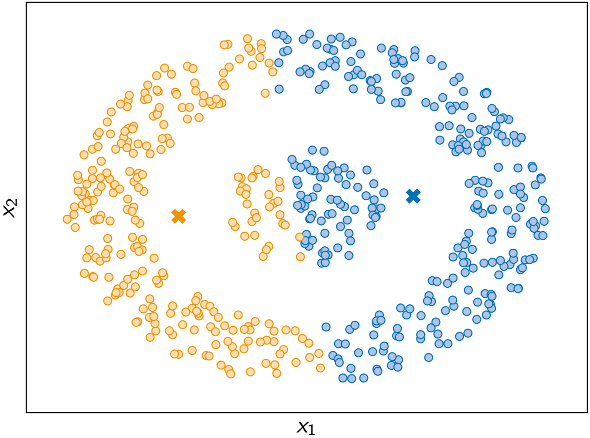
Grenzen von k-Means-Clustering

Wie würde k-Means mit $K = 2$ die folgenden Daten clustern?



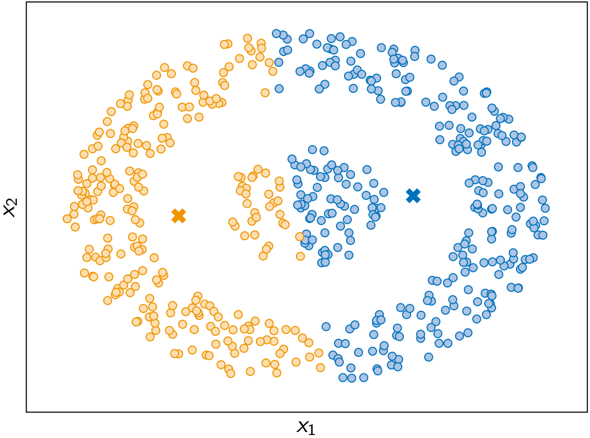
Grenzen von k-Means-Clustering

Wie würde k-Means mit $K = 2$ die folgenden Daten clustern?



Grenzen von k-Means-Clustering

Wie würde k-Means mit $K = 2$ die folgenden Daten clustern?



Bemerkung: in einer solchen Situation wäre ein dichtebasiertes Verfahren (z.B. DBSCAN) besser geeignet.

Vor- und Nachteile von k-means Clustering

Vorzüge

- Intuitiv
- Effizient
- oft akzeptable Ergebnisse

Nachteilige Eigenschaften

- Keine Behandlung/Berücksichtigung von Ausreißern und Rauschen
- Der Algorithmus findet ggf. nicht die optimale Lösung (lokale Optima).
- Die Clusteranzahl muss vorher festgelegt werden.
- Es werden immer Cluster berechnet, auch wenn die Daten in Wirklichkeit homogen sind.
- Cluster hängen stark von der Wahl der Initialwerte der Cluster-Zentren ab
- Reskalierung der Daten kann die Ergebnisse stark verändern. Es ist daher üblich, den Datensatz vor dem Clustering zu standardisieren.

Dimensionsreduktion

Häufig hat man hochdimensionale Daten mit sehr vielen Features gegeben.

Mögliche Probleme

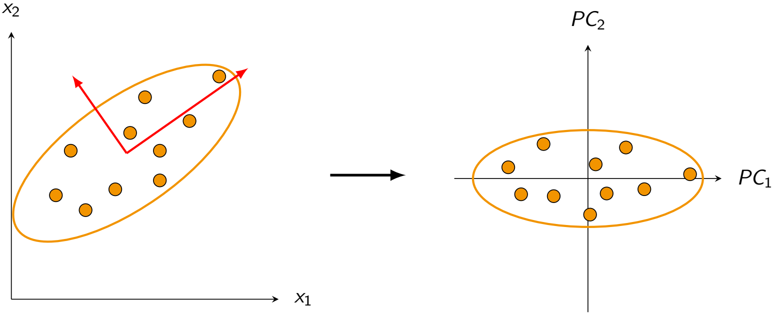
- Man hat nicht genug Trainingsdaten, um Modelle mit hinreichender Qualität zu trainieren („curse of dimensionality")
- Korrelierte Features können numerische Probleme beim Modelltraining bereiten.
- Das Modelltraining ist rechenintensiv.
- Die Ergebnisse sind schwierig zu interpretieren.

Dimensionsreduktion durch PCA (Principal Component Analysis)

- Reduktion der Dimension der Daten durch lineare Projektion der Features auf einen Feature-Raum geringerer Dimension
- Dabei: Maximierung der verbleibenden Varianz (Annahme: hohe Varianz bedeutet hohen Informationsgehalt)
- Mathematisches Werkzeug: Hauptkomponentenanalyse

Beispiel

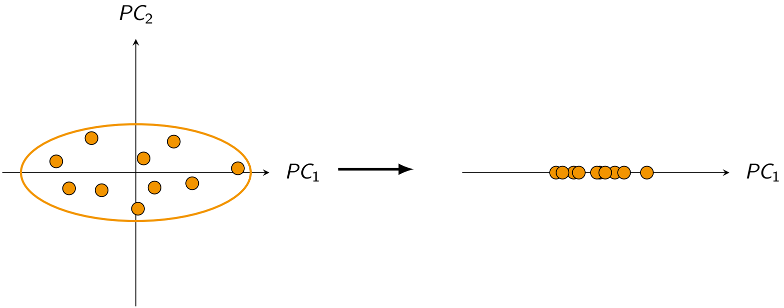
Schritt1: Transformation in anderes Koordinatensystem Transformiere in ein neues, orthogonales, Koordinatensystem mit möglichst großer Varianz entlang der Achsen:



Beispiel

Schritt2: Dimensionsreduktion

Üblicherweise erfolgt nach der Transformation eine Projektion in einen Raum niedrigerer Dimension, indem Dimensionen mit geringer Varianz weggelassen werden (Dimensionsreduktion):



PCA aus mathematischer Sicht

- Die neuen Koordinatenachsen (=Hauptkomponenten) ergeben sich als Eigenvektoren der empirischen Kovarianzmatrix der Daten.
- Sie stellen Linearkombinationen der ursprünglichen Koordinaten (=Features) dar (**lineare** Transformation).
- Die Kovarianzmatrix ist symmetrisch und deshalb diagonalisierbar und es gibt eine Orthonormalbasis aus Eigenvektoren (alle haben Länge 1 und sind orthogonal zueinander).
- Sortiert man die Eigenvektoren in absteigender Reihenfolge nach dem Betrag der zugehörigen Eigenwerte, wird bei Projektion der Daten auf den k-dimensionalen, von den ersten k Eigenvektoren aufgespannten Unterraum ($k < p$) die verbleibende Varianz maximiert. Es gibt also keine andere Projektion in einen k-dimensionalen Unterraum, sodass die transformierten Daten größere Varianz hätten als bei Projektion auf den von den ersten k Hauptkomponenten aufgespannten Unterraum.
- Man kann zeigen, dass die transformierten Features unkorreliert sind.

PCA für den Iris-Datensatz

Features: x_1 : sepal length, x_2: sepal width, x_3 : petal length, x_4: pedal width

Standardisierung:

$$x_{1,\mathrm{std}} = \frac{x_1 - 5.84}{0.825}\ , \quad x_{2,\mathrm{std}} = \frac{x_2 - 3.057}{0.434}\ , \quad x_{3,\mathrm{std}} = \frac{x_3 - 3.758}{1.759}\ , \quad x_{4,\mathrm{std}} = \frac{x_4 - 1.200}{0.760}\ .$$

Kovarianz-Matrix:

$$K = \begin{pmatrix} 1.007 & -0.118 & 0.878 & 0.823 \\ -0.118 & 1.007 & -0.431 & -0.369 \\ 0.878 & -0.431 & 1.007 & 0.969 \\ 0.823 & -0.369 & 0.969 & 1.007 \end{pmatrix}\ .$$

Eigenwerte und Eigenvektoren:

$$\boldsymbol{\lambda} = (2.92, 0.91, 0.15, 0.02)^T\ , \quad V = \begin{pmatrix} 0.521 & -0.377 & -0.720 & 0.261 \\ -0.269 & -0.923 & 0.244 & -0.124 \\ 0.580 & -0.024 & 0.142 & -0.801 \\ 0.565 & -0.067 & 0.634 & 0.524 \end{pmatrix}$$

Die ersten beiden Hauptkomponenten enthalten $\frac{2.92+0.91}{2.92+0.91+0.15+0.02} = 96\%$ der Gesamtvarianz der (standardisierten) Daten.

PCA für den Iris-Datensatz

Aus den Eigenvektoren ergibt sich die Transformation wie folgt:

	PC_1	PC_2	PC_3	PC_4
$x_{1,\text{std}}$	0.521	-0.377	-0.720	0.261
$x_{2,\text{std}}$	-0.269	-0.923	0.244	-0.124
$x_{3,\text{std}}$	0.580	-0.024	0.142	-0.801
$x_{4,\text{std}}$	0.565	-0.067	0.634	0.524

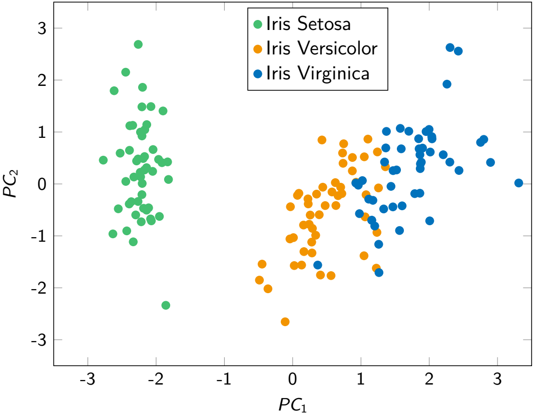
Da bereits die ersten beiden Hauptkomponenten bereits mehr als 95% der Gesamtvarianz enthalten, werden die beiden hinteren vernachlässigt Die zwei resultierenden Features lauten:

$$\begin{aligned} PC_1 &= 0.521 x_{1,\text{std}} - 0.269 x_{2,\text{std}} + 0.580 x_{3,\text{std}} + 0.565 x_{4,\text{std}} \\ PC_2 &= -0.377 x_{1,\text{std}} - 0.923 x_{2,\text{std}} - 0.024 x_{3,\text{std}} - 0.067 x_{4,\text{std}} \end{aligned}$$

Das Problem wurde also um zwei Dimensionen reduziert.

PCA für den Iris-Datensatz

- Bereits mit Hilfe der zwei Hauptkomponenten als Features können die drei Klassen sehr gut getrennt werden.
- Ein Logistisches Regressionsmodell liefert auf dem Trainingsdatensatz eine Accuracy von 92% bei Verwendung der Features PC_1 und PC_2, während bei Verwendung aller Features die Accuracy 98% beträgt.



Rechenbeispiel

Die Kovarianzmatrix zweier Features x_1 und x_2 laute

$$K = \begin{pmatrix} 2 & -1 \\ -1 & 1 \end{pmatrix} .$$

- Bestimmung der Eigenwerte und Sortierung absteigend nach Größe:

$$P(\lambda) = \det(K - \lambda \cdot E) = \det \begin{pmatrix} 2-\lambda & -1 \\ -1 & 1-\lambda \end{pmatrix}$$
$$= (2-\lambda)(1-\lambda) - 1 = (\lambda - 3)(\lambda - 1) \Rightarrow \lambda_1 = 3,\ \lambda_2 = 1 .$$

- Bestimmung der zugehörigen Eigenvektoren:

$$\begin{pmatrix} 2-\lambda_1 & -1 \\ -1 & 1-\lambda_1 \end{pmatrix} = \begin{pmatrix} -1 & -1 \\ -1 & -1 \end{pmatrix} \xrightarrow{\text{Gauß}} \begin{pmatrix} -1 & -1 \\ 0 & 0 \end{pmatrix}$$
$$\begin{pmatrix} 2-\lambda_2 & -1 \\ -1 & 1-\lambda_2 \end{pmatrix} = \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \xrightarrow{\text{Gauß}} \begin{pmatrix} 1 & -1 \\ 0 & 0 \end{pmatrix}$$

- Die Hauptkomponenten sind die normierten Eigenvektoren:

$$\boldsymbol{v}_1 = \frac{1}{\sqrt{2}}(1, -1)^T , \quad \boldsymbol{v}_2 = \frac{1}{\sqrt{2}}(1, 1)^T .$$

Rechenbeispiel

- Jedes $\boldsymbol{x} \in \mathbb{R}^2$ kann nun als Linearkombination der neuen Hauptkomponenten dargestellt werden:

$$\boldsymbol{x} = \lambda_1 \boldsymbol{v}_1 + \lambda_2 \boldsymbol{v}_2 = \underbrace{\begin{pmatrix} v_{11} & v_{21} \\ v_{12} & v_{22} \end{pmatrix}}_{=:V} \underbrace{\begin{pmatrix} \lambda_1 \\ \lambda_2 \end{pmatrix}}_{=:\boldsymbol{\lambda}} .$$

- Da $\boldsymbol{v}_1$ und $\boldsymbol{v}_2$ orthonormal sind, gilt $V^{-1} = V^T$, d.h. die Koordinaten $\boldsymbol{\lambda}$ bezüglich der Hauptkomponenten ergeben sich zu

$$\boldsymbol{\lambda} = (\lambda_1, \lambda_2)^T = V^T \boldsymbol{x} .$$

- Die Projektion auf einen niederdimensionalen Raum erfolgt, indem man die Koordinaten zu gewissen Hauptkomponenten (typischerweise diejenigen, die zu betragsmäßig kleinen Eigenwerten gehören) gleich Null setzt. In unserem Beispiel könnte man folgendermaßen auf den von $\boldsymbol{v}_1$ aufgespannten Unterraum U_1 projizieren:

$$P_{U_1}(\lambda_1 \boldsymbol{v}_1 + \lambda_2 \boldsymbol{v}_2) := \lambda_1 \boldsymbol{v}_1 .$$

- Dadurch hat man die Dimension reduziert.

PCA - Zusammenfassung

Beachte

- Die Größe der Eigenwerte und damit das Ergebnis der PCA hängt von den Einheiten der einzelnen Features ab.
- Bevor man eine PCA durchführt, sollten die Daten standardisiert werden.
- Dann entspricht die Kovarianzmatrix der Korrelationsmatrix.

Wann setzt man PCA ein?

- Wenn man die Anzahl der Features reduzieren möchte
- Wenn man unkorrelierte Features benötigt
- Wenn die Interpretierbarkeit der Features nachrangig ist.