Grundlagen der Künstlichen Intelligenz - Informatik

Rapp, DHBW Lörrach

24.11.2023

Inhaltsübersicht

1. Bias-Varianz-Dilemma
2. Clustering
3. Agglomerativ
4. k-Means
5. DBSCAN
6. Erwartungs-Maximierung

Lernziele

Meine 3 Lernziele für heute

1. Ich kenne den grundlegenden Aufbau der vorgestellten Clustering-Algorithmen des maschinellen Lernens.
2. Ich verstehe die Vorgehensweise zur Strukturierung von Clustering-Programmen.
3. Ich kann die Ergebnisqualität eines Clustering-Algorithmus interpretieren und daraus Maßnahmen zur Verbesserung des Agentenprogramms ableiten.

Bias-Varianz-Dilemma

Bias-Varianz-Zerlegung im überwachten Lernen

Ziel des überwachten Lernens

Berechnung der Modellfunktion $\hat{f}(x)$, die

1. die wahre Funktion $y = f(x)$ so gut wie möglich annähert und
2. sich anschließend auf ungesehene Testdaten verallgemeinern lässt.

Leider ist es i.d.R. **unmöglich,** beides gleichzeitig zu tun $\rightarrow$**Dilemma!**

Wir berechnen den **erwarteten quadratischen Fehler** $E[(y - \hat{f}(x))^2]$ zwischen der **wahren Funktion** y und der **Modellfunktion** $\hat{f}(x)$ für eine ungesehene Stichprobe x:

Bias-Varianz-Zerlegung

$$Err(x) = E[(y - \hat{f}(x))^2] = Bias[\hat{f}(x)]^2 + Variance[\hat{f}(x)] + \sigma^2$$

Trainingsdaten $x_1, ..., x_n$ mit dazugehörigen reellen Werten $y_1, ..., y_n$.

Terme der Bias-Varianz-Zerlegung

Verzerrung *(engl. „Bias“)*

- Systematischer Fehler aufgrund **vereinfachenden Annahmen** innerhalb des Modells.
- **Beispiele**
  - Hoher Bias: Lineare Regression, z.B. nichtlineare Funktion $y = f(x)$ wird durch lineare Funktion $\hat{f}(x)$ approximiert.
  - Niedriger Bias: Entscheidungsbäume, k-Nearest Neighbours und Support-Vector-Machines

Varianz

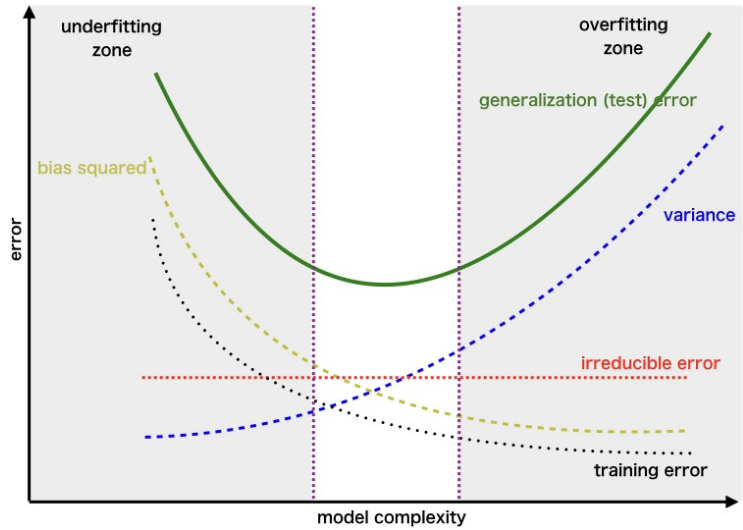
- Fehler ausgehend von der **Empfindlichkeit** der Lernmethode $\hat{f}(x)$ auf kleine Schwankungen in den Trainingsdaten.
- Mathematisch: *Schwankungsbreite* des Modells $\hat{f}(x)$ um seinen Erwartungswert
- **Beispiele**
  - Hohe Varianz: Entscheidungsbäume, k-Nearest Neighbours und Support-Vector-Machines
  - Niedrige Varianz: Lineare Regression

Irreduzibler Fehler σ

- Außerhalb unserer Kontrolle
- z.B. aufgrund **statistischem Rauschen** der Beobachtungen.
- stellt untere Schranke für die erwartete Abweichung auf ungesehenen Testdaten dar, da alle drei Terme nicht negativ sind.

Graphische Visualisierung

Bereich optimaler Kapazität: Niedrige Bias und Varianz bedeuten gute Generalisierbarkeit.

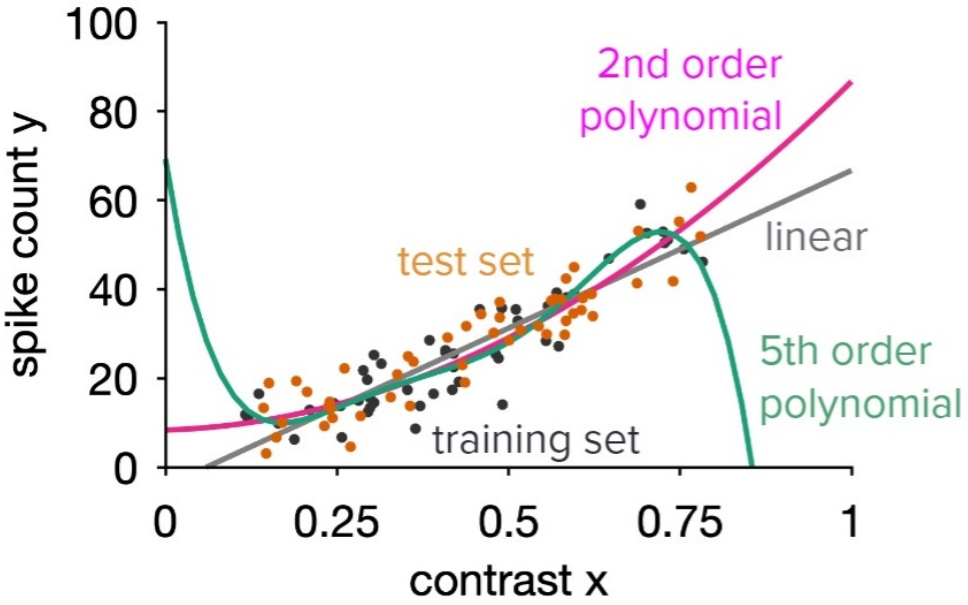


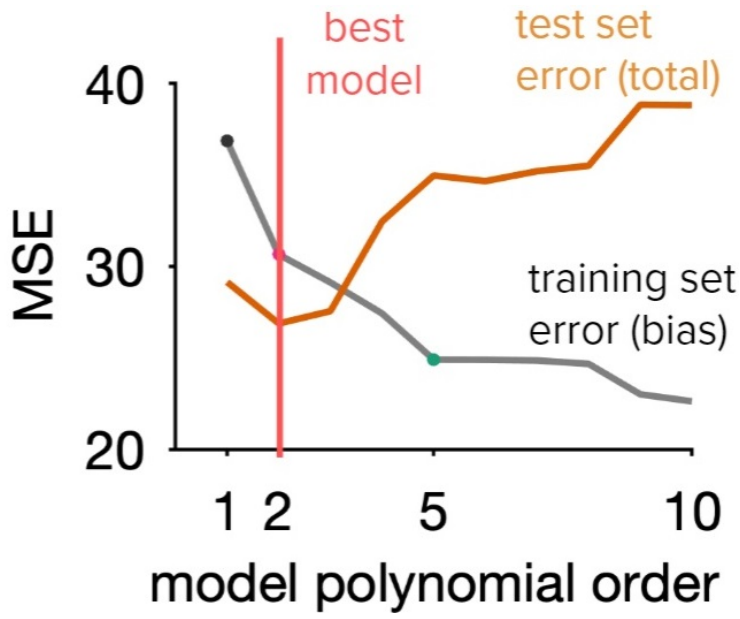
Schematischer Verlauf der Fehlerterme aus Bias-Varianz-Zerlegung als Funktion der Modell-Komplexität (=Kapazität) mit den Bereichen Unter- (links) und Überanpassung (rechts) und dem optimalen Bereich (Mitte).

generalization (test) error: $Err(x) = Bias^2 + Variance + irreducible\ error$

- **Hoher Bias und niedrige Varianz:** z.B. lineares Modell approximiert nichtlineare Funktion $\Rightarrow$ **Unteranpassung**
- **Niedriger Bias und hohe Varianz:** Rauschen in den Trainingsdaten statt der vorgesehenen Ausgabe wird modelliert $\Rightarrow$ **Überanpassung**
- Mit **steigender Kapazität** (=Anzahl der Aspekte der Daten, welche im Modell berücksichtigt werden) passt sich das Modell an die Trainingsdaten an $\Rightarrow$ Bias sinkt $\Rightarrow$ Risiko für **Überanpassung** steigt
- Modell mit **geringerer Kapazität** kann wahren Zusammenhang nicht mehr erfassen $\Rightarrow$ Bias steigt $\Rightarrow$ Risiko für **Unteranpassung** steigt

Polynombeispiel





- **Schwarz:** Trainingsdaten für das Einlernen (engl. „fitting“) des Modells.
- **Orange:** Testdaten, die das Modell während des Trainings nicht gesehen hat.
- **Grau:** Lineares Modell mit großem Trainingsfehler.
- **Lila:** Quadratisches Polynom bietet den besten Bias-Varianz Kompromiss und sollte daher ausgewählt werden.
- **Grün:** Polynom 5. Ordnung hat den geringsten Trainingsfehler, da es fast alle Punkte mit geringer Abweichung abbilden kann. Jedoch ist die Varianz groß, weil sich Polynome 5. Ordnung stark voneinander unterscheiden, wodurch dieses Modell eine hohe Anfälligkeit für Schwankungen in den Trainingsdaten besitzt.

Beispiele allgemeiner Polynomformeln n-ter Ordnung

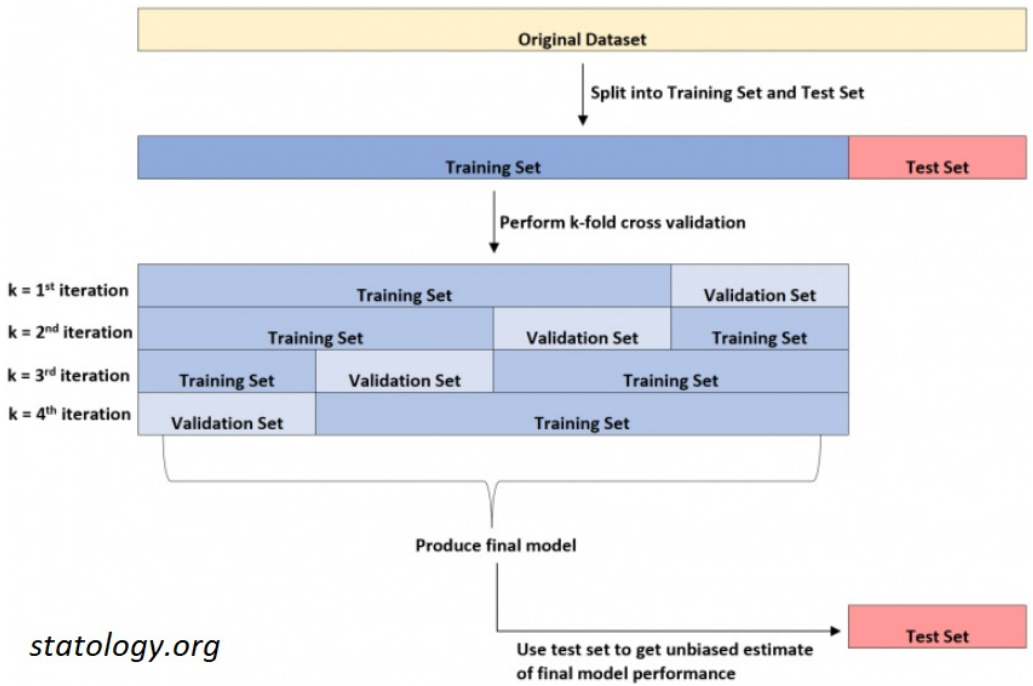
- **Lineare Regression:** $y = \theta_1 x + \theta_0$
  $\rightarrow$ Unteranpassung
- **Quadratisch:** $y = \theta_2 x^2 + \theta_1 x + \theta_0$
  $\rightarrow$ Bestes Modell
- **Fünfte Ordnung:** $y = \sum_{p=0}^{5} \theta_p x^p$
  $\rightarrow$ Überanpassung

Kreuzvalidierung

Aus: medium.com, Koppert-Anisimova

Vorgehen der Kreuzvalidierung

k-fache Kreuzvalidierung: Partitionierung des ursprünglichen Datensatzes in k zufällig ausgewählte Teilmengen *(„folds“)* gleicher Größe:



Bei jeder der k Iterationen wird eine andere Teilmenge des Trainingsdatensatzes zum Validieren des Modells verwendet.

- **Vorteile:** Over- und Underfitting werden reduziert.
- **Nachteil:** Vorgang ist rechenintensiv, da Training und Validierung mehrmals durchgeführt werden müssen.

Clustering

Clustering-Algorithmen

Cluster-Analyse *(auch: Clustering-Algorithmen)*

Verfahren zur Aufdeckung von Ähnlichkeitsstrukturen in großen Datenbeständen.

Charakterisierung

- Verfahren des unüberwachten Lernens → keine Label für Datenpunkte vorhanden

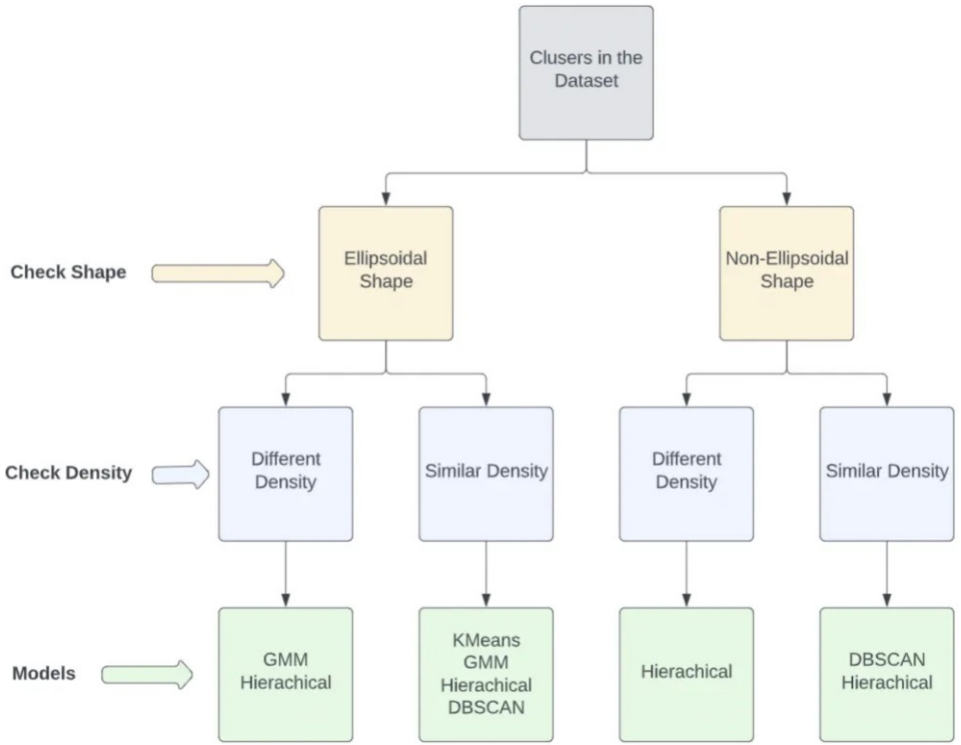
Ziel

Organisation von Daten in Gruppen, um eine **hohe Intra-Cluster** und **niedrige Inter-Cluster Ähnlichkeit** zu erzielen.

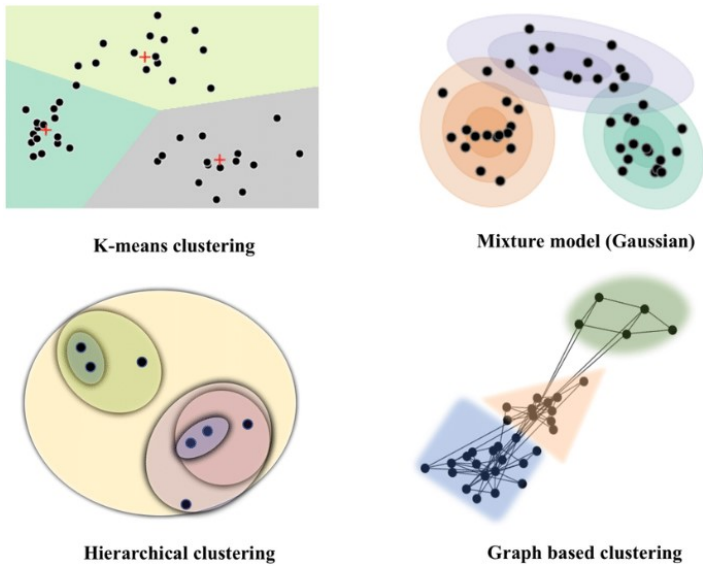
Abgrenzung

- **Klassifikation:** Daten werden bestehenden Klassen zugeordnet
- **Cluster-Analyse:** Neue Gruppen in Daten identifizieren

Übersicht Clustering Algorithmen



GrabNGoInfo.com



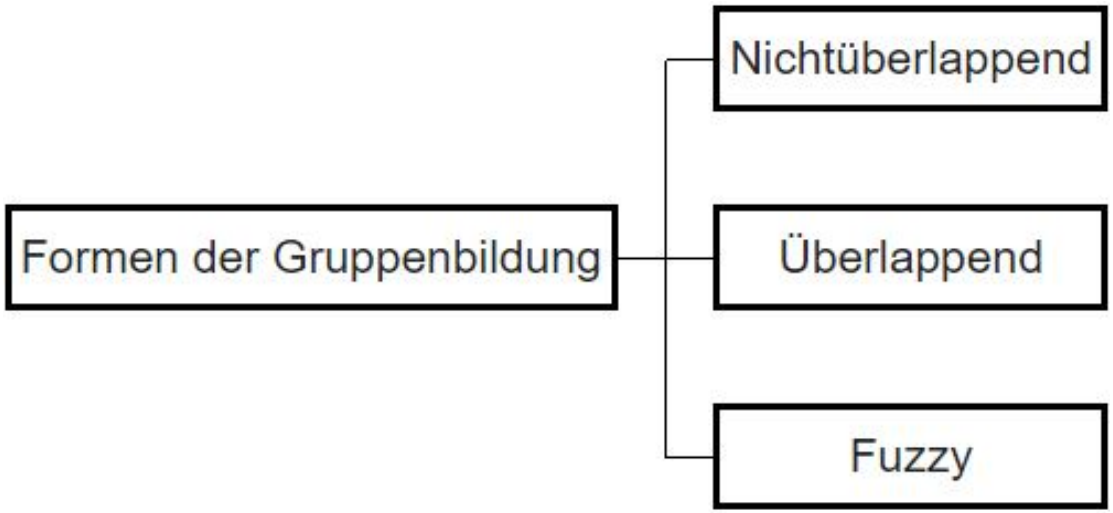
researchgate.net, Meenakshi

Unterscheidungsmerkmale Clustering-Algorithmen

1) Ähnlichkeits- und Gruppenbegriff
2) Cluster-Modell
3) Algorithmisches Vorgehen
4) Toleranz gegenüber Störungen in den Daten

Formen der Gruppenbildung

Drei unterschiedliche Formen der **Gruppenbildung** bzw. -zugehörigkeit sind möglich:



Bei den **nichtüberlappenden Gruppen** wird jedes Objekt nur einer Gruppe (Segment, Cluster) zugeordnet, bei den **überlappenden Gruppen** kann ein Objekt mehreren Gruppen zugeordnet werden, und bei den **Fuzzygruppen** gehört ein Element jeder Gruppe mit einem bestimmten Grad des Zutreffens an.

Hierarchische Clusterverfahren

Hierarchische Clusterverfahren

- Familie **distanzbasierter Verfahren** zur Clusteranalyse
- Unterscheidung der Verfahren nach den verwendeten **Distanzmaßen** (zwischen Objekten, aber auch zwischen ganzen Clustern) und ihrer **Berechnungsvorschrift.**

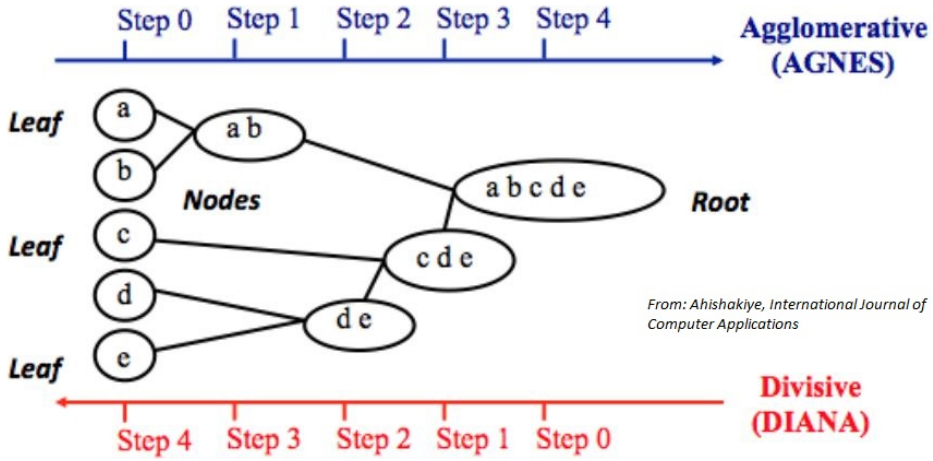
Aufbau

1. Startet mit der feinsten (agglomerativ bzw. bottom-up) bzw. gröbsten (divisiv bzw. top-down) Partition
   - **Agglomerative Verfahren** kommen in der Praxis (z.B. Marktsegmentierung im Marketing) viel häufiger vor
   - Feinste Partition: enthält lediglich ein Element bzw. jedes Element bildet seine eigene Gruppe/Partition.
   - Gröbste Partition: entspricht der Gesamtheit aller Elemente
2. Clusterbildung durch anschließendes Zusammenfassen bzw. Aufteilen

Cluster

Gruppierung von Objekten, die zueinander eine **geringere Distanz** (d.h. höhere Ähnlichkeit) aufweisen als zu Objekten anderer Gruppen.

Berechnungsvorschrift



Agglomeratives Clusterverfahren *(auch: Bottom-up-Verfahren)*

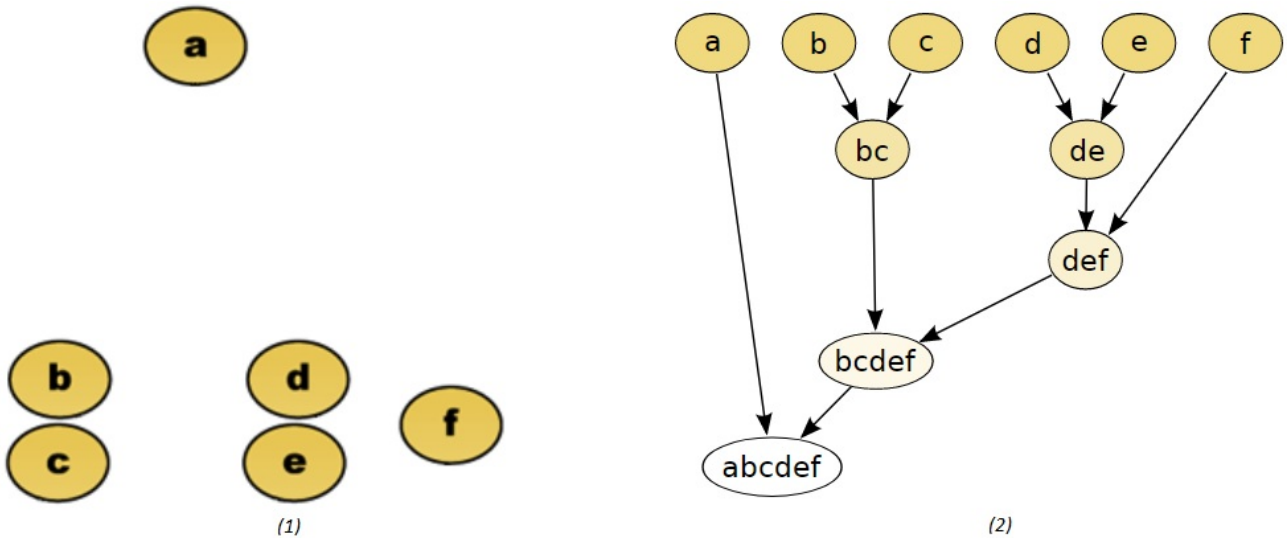
Zunächst bildet jedes Objekt einen Cluster und dann werden schrittweise die bereits gebildeten Cluster zu immer größeren zusammengefasst, bis alle Objekte zu einem Cluster gehören.

- An der entstandenen Hierarchie kann man nicht erkennen, wie sie berechnet wurde.
- Strikte Clusterhierarchie: Einmal gebildete Cluster können nicht mehr verändert werden.

Visualisierung von hierarchischem Clustering

Dendogramm

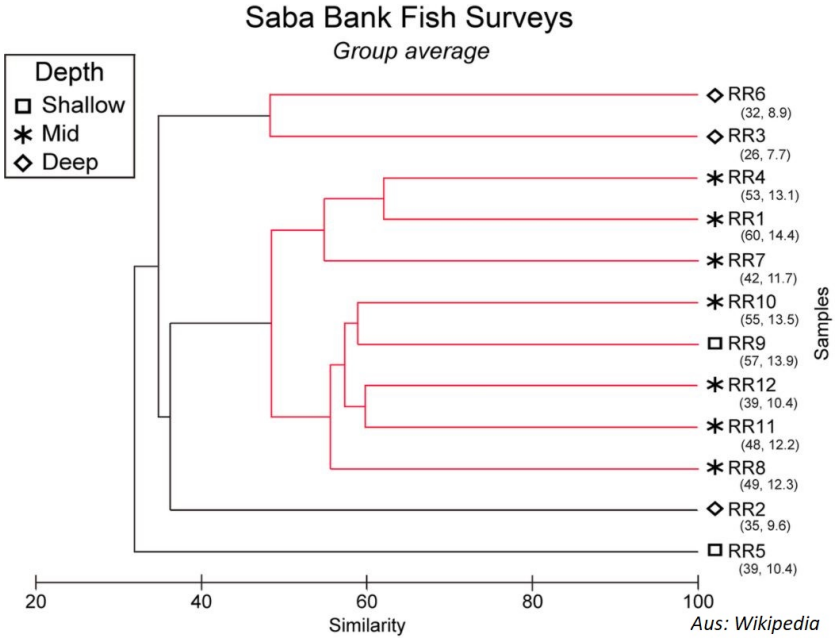
Baumstruktur als Darstellungsform für die hierarchische Zerlegung der Datenmenge **O** in immer kleinere Teilmengen.



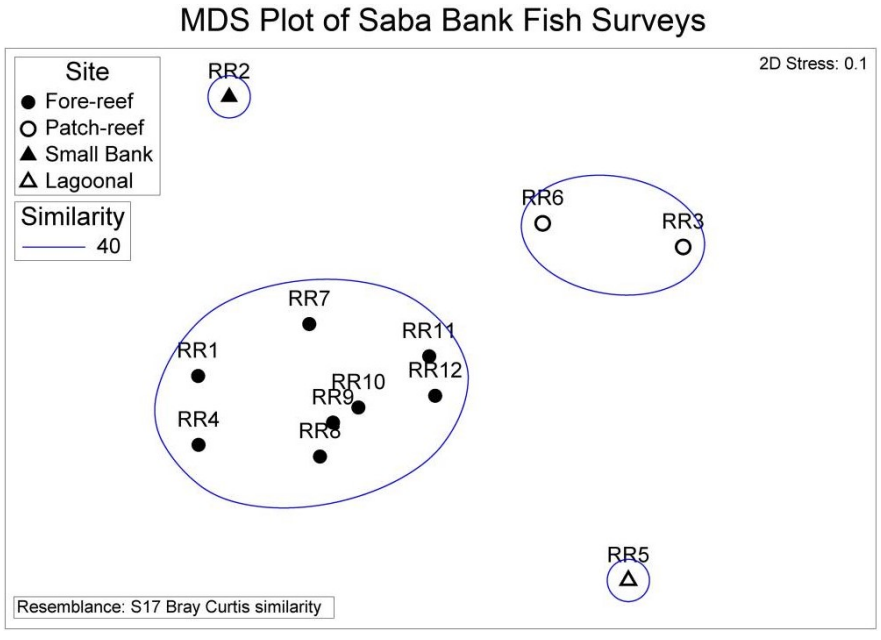
(1) Beispieldatensatz: Die Objekte b und c sowie d und e liegen dicht zusammen.
(2) Dendogramm für Single-Linkage: b und c sowie d und e werden als erstes zusammengefasst.

- **Wurzel:** einzelner Cluster, der die gesamte Menge **O** enthält
- **Blätter:** Cluster, in denen sich je ein einzelnes Objekt der Datenmenge befindet
- **Innerer Knoten:** Vereinigung aller Kinderknoten
- **Kante:** Distanz als Attribut zwischen Knoten und Kindknoten

Dendogramm Beispiel



Dendogramm: *Die Achse repräsentiert die Ähnlichkeit (bzw. Distanz) zweier Cluster. Beispiel: RR1 und RR4 wurden beim Ähnlichkeitsmaß 62 zusammengefügt.*



MDS (Non-metric multidimensional-scaling ordination) Plot 12 verschiedener Fish-Survey Stationen als **Visualisierungshilfe.**

- In der Darstellung als **Dendogramm** kann man eine gewünschte Zahl von Clustern auswählen, indem man das Dendrogramm auf einer geeigneten Höhe durchschneidet.
- Typischerweise sucht man eine Stelle, wo es zwischen zwei Fusionierungen einen großen Sprung der Distanz oder (Un-)ähnlichkeit gibt, z. B. im obigen Dendrogramm auf der Höhe 40.
- Dann ergeben sich vier Cluster, von denen 2 nur einzelne Objekte enthalten (RR2, RR5), ein Cluster enthält zwei Objekte (RR3 und RR6) und der letzte Cluster enthält die übrigen Objekte.
- Gibt es hierarchische Cluster mit deutlich unterschiedlichen Größen, so kann es notwendig sein, auf unterschiedlichen Höhen zu zerlegen: während ein Cluster auf einer Höhe noch mit seinen Nachbarn verbunden ist, zerfällt ein anderer ("dünnerer") Cluster auf dieser Höhe schon in einzelne Objekte.

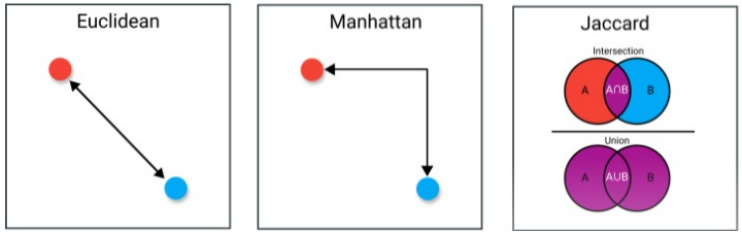
Distanz- und Ähnlichkeitsmaße

Sowohl in der agglomerativen als auch bei den divisiven hierarchischen Clusteranalysen ist es notwendig, Abstände bzw. (Un-)ähnlichkeiten zwischen zwei Objekten, einem Objekt und einem Cluster oder zwei Clustern zu berechnen. Je nach **Skalenniveau** der zugrunde liegenden Variablen kommen verschiedene Maße zum Einsatz:

- **Metrische Variablen: Distanzmaße.** Z.B. bedeutet ein Wert von 0, dass die Objekte einen Abstand von 0, also maximale Ähnlichkeit haben.
- **Kategoriale (nominale und ordinale) Variablen: Ähnlichkeitsmaße.** Z.B. bedeutet ein Wert von 0, dass die Objekte maximale Unähnlichkeit haben.

Beispiele

- Distanzmaße
  - Euklidisch: $\sqrt{\sum_{k=1}^{p}(x_{ik} - x_{jk})^2}$
  - Manhattan: $\sum_{k=1}^{p} |x_{ik} - x_{jk}|$
- Jaccard Ähnlichkeitsmaß: $\frac{n_{11}}{n_{01}+n_{10}+n_{11}}$



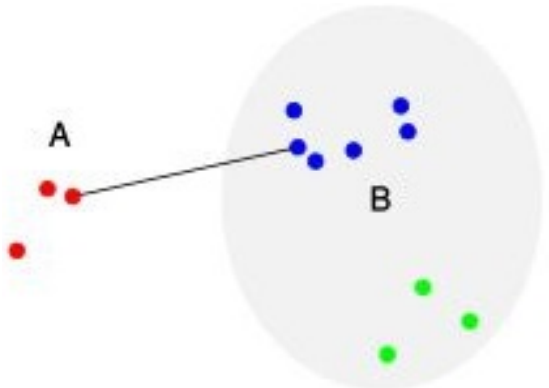
Fusionierungsalgorithmen

Der Abstand D zwischen Cluster A und dem neuen Cluster B wird oft über den Abstand oder die Unähnlichkeit d von zwei Objekten berechnet. Der neue Cluster B ist aus der Fusion des "grünen" und "blauen" Clusters entstanden:

Single-Linkage *(z.B. SLINK)*

Minimaler Abstand aller Elementpaare aus den beiden Clustern:

$$D_{single-linkage}(A, B) := \min_{a \in A, b \in B} \{d(a, b)\}$$

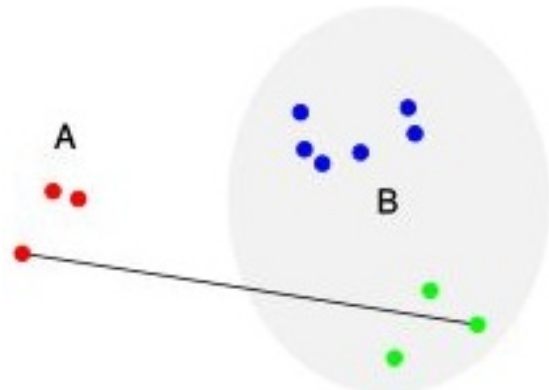


→ *Dieses Verfahren neigt zur Kettenbildung*

Complete-Linkage *(z.B. CLINK)*

Maximaler Abstand aller Elementpaare aus den beiden Clustern:

$$D_{complete-linkage}(A, B) := \max_{a \in A, b \in B} \{d(a, b)\}$$



→ *Neigt zur Bildung kleiner Gruppen*

Distanzmatrix

Bei der Verwendung eines bestimmten Distanzmaßes werden im ersten Schritt die beiden einander nächsten Objekte zu einem Cluster fusioniert. Dies kann wie folgt als **Distanzmatrix** *(auch: Proximity Matrix, links)* dargestellt werden:

Distanz zw.	Cluster1 Objekt1	Cluster2 Objekt2	Cluster3 Objekt3	Cluster4 Objekt4
Objekt1	0			
Objekt2	4	0		
Objekt3	7	5	0	
Objekt4	8	10	9	0

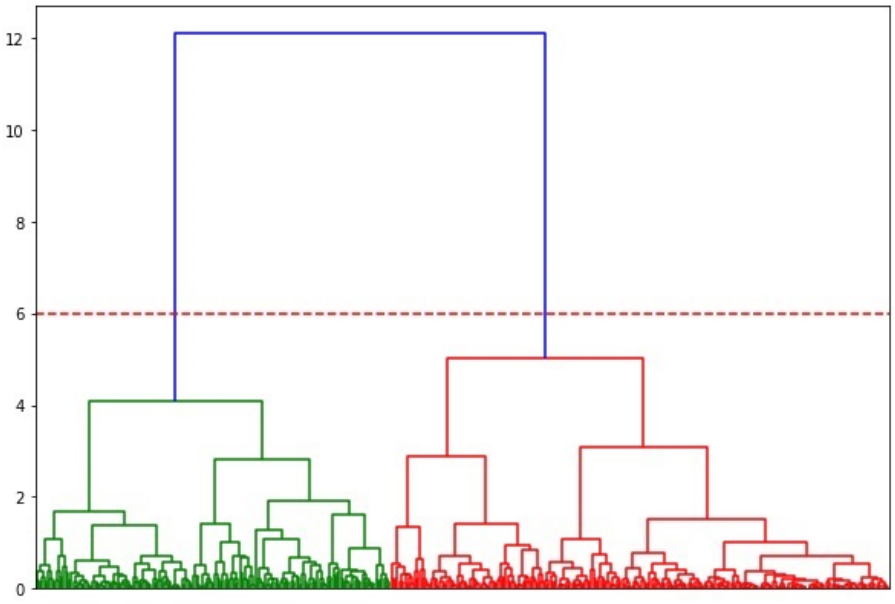
Distanz zw.	Cluster1 Objekt1&2	Cluster2 Objekt3	Cluster3 Objekt4
Objekt1&2	0		
Objekt3	7 o. 5	0	
Objekt4	8 o. 10	9	0

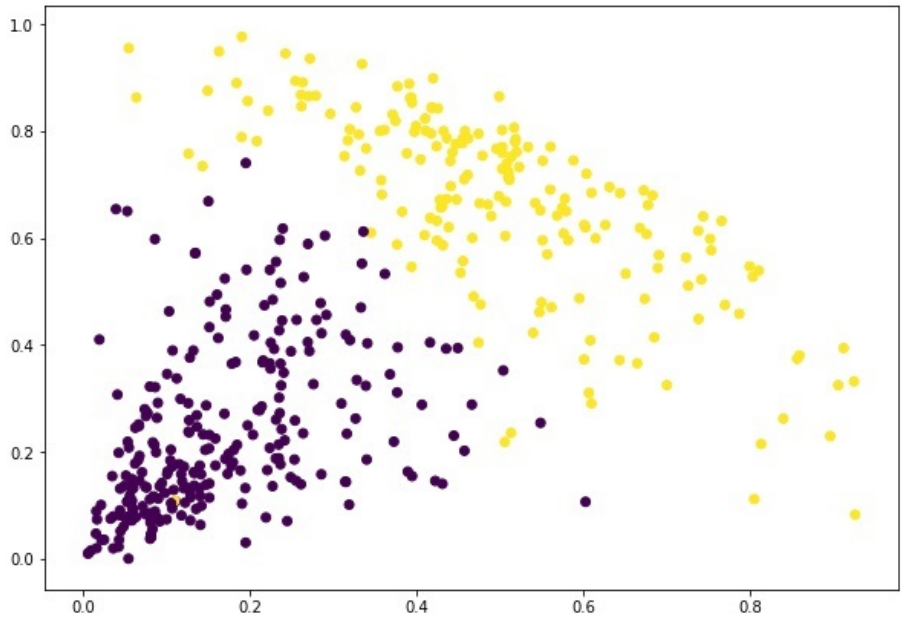
Die kleinste Distanz findet sich zwischen dem Objekt1 und Objekt2 (rot in der Distanzmatrix) und man würde daher Objekt1 und Objekt2 zu einem Cluster zusammenfassen (fusionieren). Im zweiten Schritt muss die Matrix nun neu erstellt werden (*rechts,* "o." steht für oder), das heißt die Distanz zwischen dem neuen Cluster und Objekt3 bzw. Objekt4 muss neu berechnet werden (gelb in der Distanzmatrix).
Welcher der beiden Werte für die Distanzbestimmung relevant ist, bestimmt das Verfahren:

- **Single-Linkage:** $min(5, 7) = 5,\ min(8, 10) = 8$
- **Complete-Linkage:** $max(5, 7) = 7,\ max(8, 10) = 10$

Code-Beispiel

Kundensegmentierung mit hierarchischem Clustering in Python





https://www.analyticsvidhya.com/blog/2019/05/beginners-guide-hierarchical-clustering/

Das Setzen der gestrichelten Trennlinie zwischen Clustern mit maximaler Distanz resultiert in **2 Clustern.**

Entsprechende Zuordnung der Datenpunkte zu den beiden Clustern im **Feature-Vektorraum.**

Gruppenarbeit

Verteilen Sie sich auf insgesamt 3 Gruppen und benennen jeweils eine/n Gruppensprecher/in.

Gruppe 1: Analyse der Programmstruktur

Analysieren Sie den strukturellen Aufbau des Programms **Clustering_Algorithms.ipynb** und präsentieren Ihre Ergebnisse.

Gruppe 2: Programmausführung

1. Kopieren Sie das Jupyter Notebook **Clustering_Algorithms.ipynb** in eine Jupyter Entwicklungsumgebung (z.B. Google Colab).
2. Führen Sie das Notebook aus und stellen die Ergebnisse kurz vor.

Gruppe 3: Gegenüberstellung von Cluster-Algorithmen

Vergleichen Sie die 4 Clusteralgorithmen k-Means, agglomerativ, Erwartungs-Maximierung und DBSCAN präsentieren Ihre Ergebnisse.

Partitionierende Clusterverfahren

Partitionierende Clusterverfahren

Verfahrensbeschreibung

1. Vorherige Definition der Clusterzahl k
2. Bestimmung von k Clusterzentren
3. Minimierung einer vorgegebenen Fehlerfunktion führt zu iterativer Verschiebung der Clusterzentren solange, bis sich die Zuordnung der Beobachtungen zu den k Clusterzentren nicht mehr verändert.
4. Objekte können während der Verschiebung der Clusterzentren ihre Clusterzugehörigkeit wechseln

Vergleich zu hierarchischem Clustering

- 1. $\rightarrow$ *Nachteil*
- 4. $\rightarrow$ *Vorteil*

k-Means

k-Means Algorithmus

Charakterisierung

- eine der am häufigsten verwendeten Techniken zur Gruppierung von Objekten, da schnell die Zentren der Cluster gefunden werden
- Gruppen mit geringer Varianz und ähnlicher Größe werden bevorzugt
- starke Ähnlichkeit zum EM-Algorithmus

Funktionsprinzip

Aus einer Menge von ähnlichen Objekten wird eine **vorher bekannte Anzahl von** k **Gruppen** gebildet.

Anwendungsbeispiel Bildverarbeitung

- In der Bildverarbeitung wird der k-Means-Algorithmus oft zur Segmentierung verwendet.
- Als Entfernungsmaß ist die euklidische Distanz häufig nicht ausreichend und es können andere Abstandsfunktionen, basierend auf z.B. Pixelintensitäten und Pixelkoordinaten verwendet werden.
- Die Ergebnisse können z.B. zur Trennung von Vordergrund und Hintergrund und zur Objekterkennung benutzt werden.

Lloyd Algorithmus

Der am häufigsten verwendete k-Means-Algorithmus ist der **Lloyd-Algorithmus** in folgenden 3 Schritten:

1. **Initialisierung:** Wähle k zufällige Mittelwerte *("Means")* $m_1^{(1)}, ..., m_k^{(1)}$ aus dem Datensatz.
2. **Zuordnung:** Jedes Datenobjekt wird demjenigen Cluster zugeordnet, bei dem die Cluster-Varianz am wenigsten erhöht wird:

$$S_i^{(t)} = \{x_j : |x_j - m_i^{(t)}|^2 \leq |x_j - m_{i^*}^{(t)}|^2 \text{ für alle } i^* = 1, ..., k\}$$

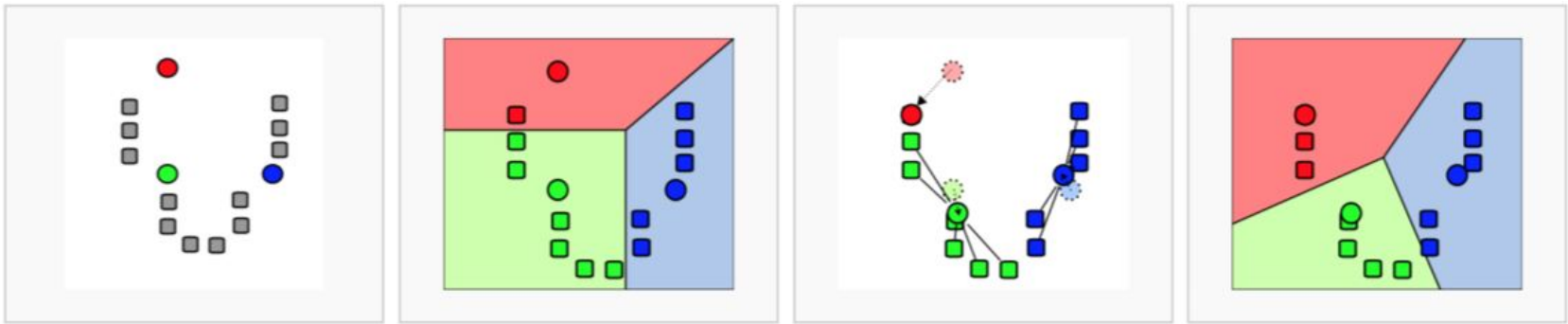
3. **Aktualisierung:** Berechne die Mittelpunkte der Cluster neu:

$$m_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{x_j \in S_i^{(t)}} x_j$$

Die Schritte 2–3 werden dabei so lange wiederholt, bis sich die Zuordnungen nicht mehr ändern.

Beispiel für $k = 3$

Die folgenden Bilder zeigen exemplarisch einen Durchlauf eines k-Means-Algorithmus zur Bestimmung von drei Gruppen:



1. *k* initial "means" (in this case *k*=3) are randomly generated within the data domain (shown in color).

2. *k* clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means.

3. The centroid of each of the *k* clusters becomes the new mean.

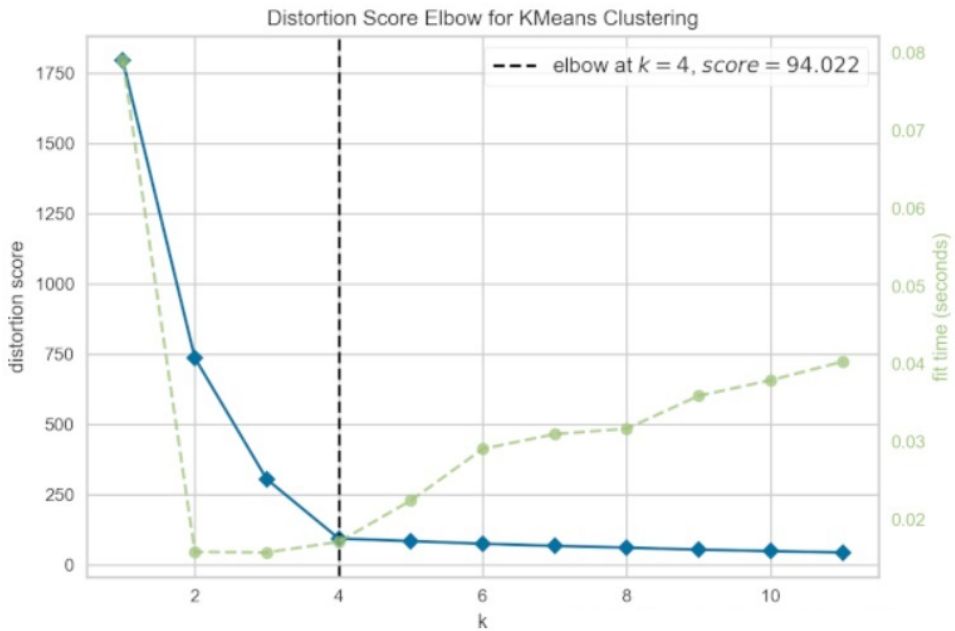
4. Steps 2 and 3 are repeated until convergence has been reached.

1. Drei Clusterzentren wurden zufällig gewählt.
2. Die durch Rechtecke repräsentierten Objekte (Datenpunkte) werden jeweils dem Cluster mit dem nächsten Clusterzentrum zugeordnet.
   - **Centroid:** Mittelwert aller Punkte des Clusters *(auch: Schwerpunkt)*
3. Die Schwerpunkte (bzw. Zentren) der Cluster werden neu berechnet.
4. Die Objekte werden neu verteilt und erneut dem Cluster zugewiesen, dessen Zentrum am nächsten ist.

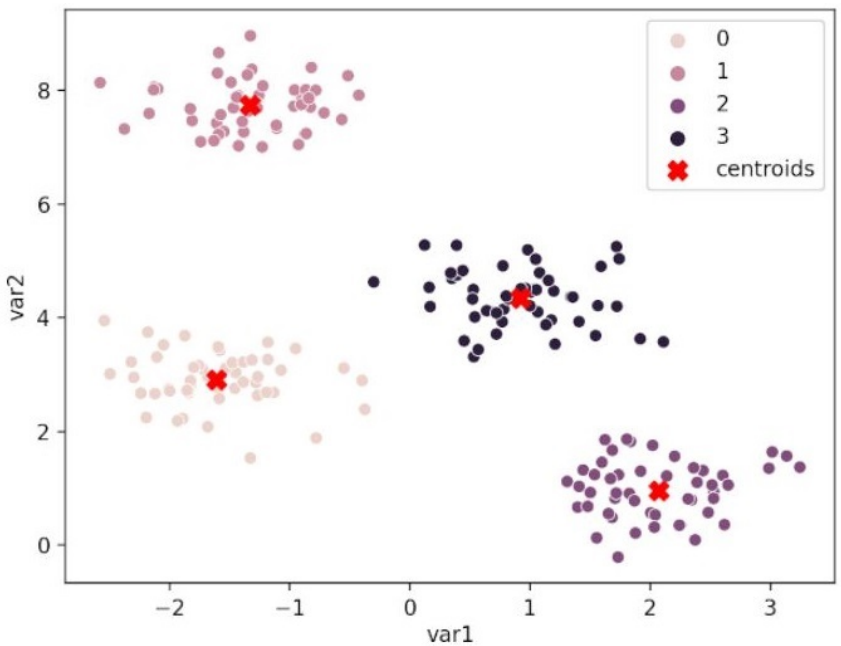
Ellbogen Methode mit Code-Beispiel

Ellbogen Methode

Heuristik zur Berechnung der Summe aller quadrierten Abstände vom Clusterzentrum zu den Datenpunkten verschiedener k-Werte (z.B. 1-10), für die Bestimmung des **optimalen k-Werts.**



Ellbogen Methode für die optimale Wahl der Clusterzahl k. Distortion *(=Verzerrung):* mittlere quadratische Entfernung der Cluster zum jeweiligen Clusterzentrum.

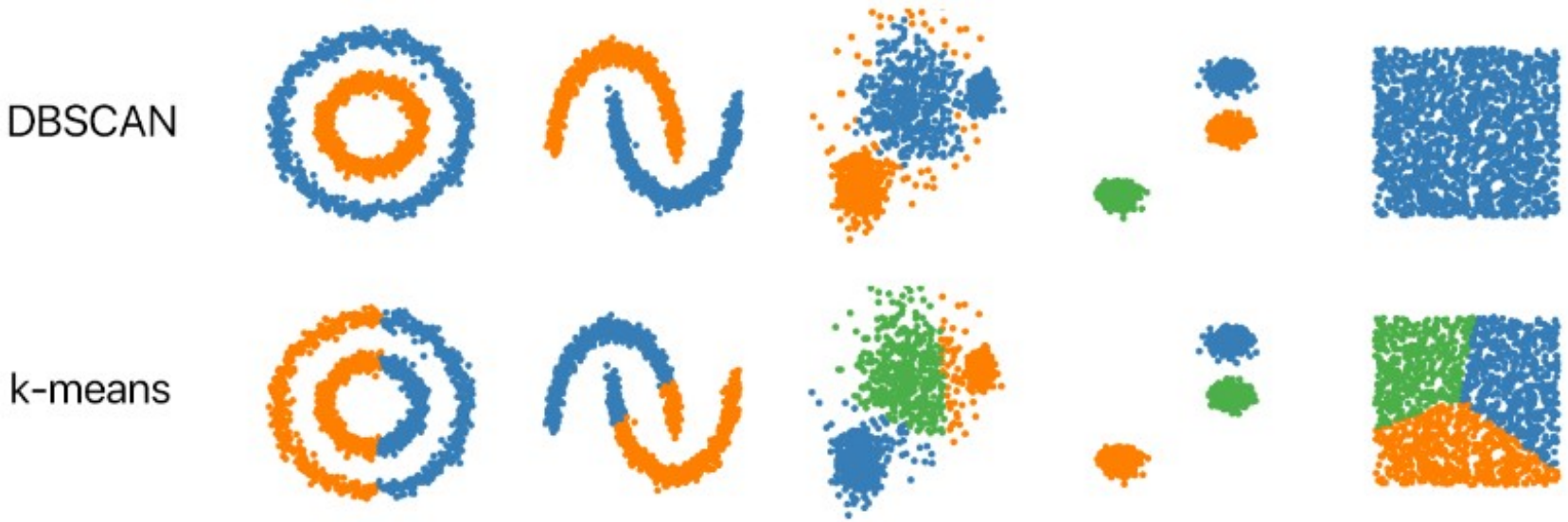


k-means Cluster mit Schwerpunkten (centroids). *Aus: https://www.reneshbedre.com/blog/kmeans-clustering-python.html*

DBSCAN

DBSCAN vs. k-means

Bei **dichtebasiertem Clustering** werden Cluster als Objekte in einem d-dimensionalen Raum betrachtet, welche dicht beieinander liegen, getrennt durch Gebiete geringerer Dichte.

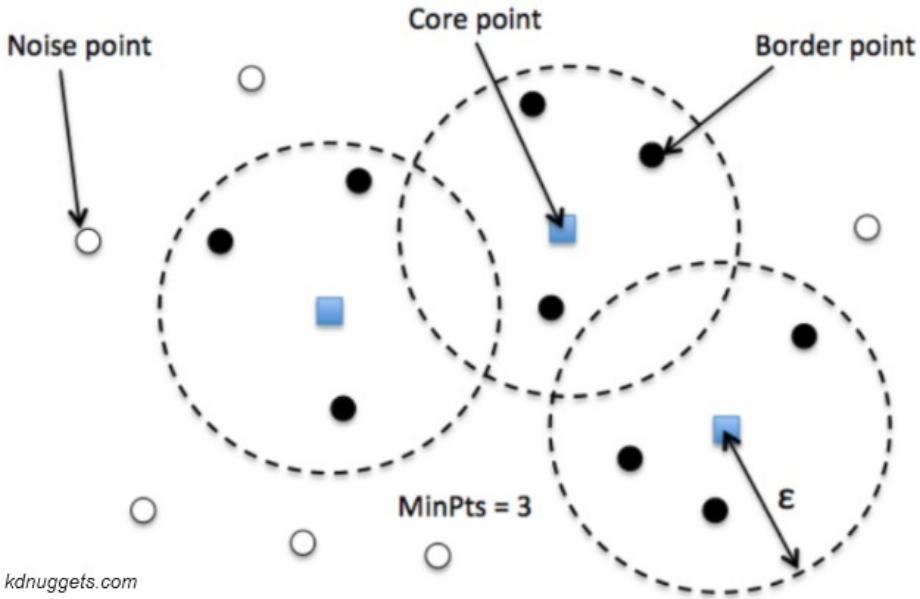


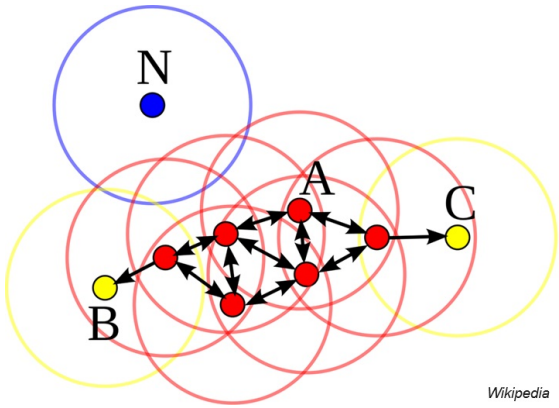
DBSCAN (Density-Based Spatial Clustering of Applications with Noise) vs. k-means Cluster-Beispiele.
Aus: https://github.com/NSHipster/DBSCAN

Beim **DBSCAN** Algorithmus muss im Gegensatz zu **k-Means** im vornherein nicht bekannt sein, wie viele Cluster existieren.

DBSCAN Parameter und Punkttypen - Idee

DBSCAN identifiziert Cluster unterschiedlicher Formen und Größen in großen Datenmengen, die Rauschen und Ausreißer enthalten können.





Punkte bei A sind Kernpunkte mit MinPts = 3. Punkte B und C sind dichte-erreichbar von A und dadurch dichte-verbunden und bilden den Rand des Clusters. Punkt N ist weder Kernpunkt noch dichte-erreichbar, also Rauschen.

Parameter

- **MinPts:** Minimale Punktzahl für einen „dichten“ Punkt (bzw. Cluster-Region).
- ϵ**:** Distanzmaß für die Lokalisierung von Nachbar-Punkten.

Punkttypen

- **Kern (Core):** Dichter Punkt mit $\geq$ *MinPts* Punkten innerhalb der Distanz *n*.
- **Rand (Border):** Nicht-dichter Punkt mit ≥ 1 Kern-Punkt innerhalb der Distanz *n*.
- **Rauschen (Noise):** Außerhalb der Kern- und Rand-Punkte mit $<$ *MinPts* Punkten innerhalb Distanz *n*.

Erwartungs-Maximierungs-Clustering

Erwartungs-Maximierungs-Clustering

EM-Clustering

- Verfahren zur Clusteranalyse, das die Daten mit einem Gaußschen Mischmodell (englisch gaussian mixture model, kurz: GMM) – also als **Überlagerung von Normalverteilungen** repräsentiert.
- Zufällige oder heuristische Initialisierung und anschließende Optimierung mit dem EM-Algorithmus
  $\rightarrow$ analoge Vorgehensweise zum **k-Means** Algorithmus

Übersicht

Erwartungs-Maximierungs-Algorithmus

- Start mit k Clustern mit jeweils zufällig gewählter Verteilungsfunktion *(=Modell)*
- Verbesserung des Modells durch abwechselnden
  1. **Erwartungsschritt (E-Schritt)**
     Zuordnung der Daten zu den einzelnen Teilen des Modells
  2. **Maximierungsschritt (M-Schritt)**
     Anpassung der Parameter des Modells an die neueste Zuordnung

In beiden Schritten wird dabei die **Qualität des Ergebnisses** verbessert: Im **E-Schritt** werden die Punkte besser zugeordnet, im **M-Schritt** wird das Modell so verändert, dass es besser zu den Daten passt. Findet keine wesentliche Verbesserung mehr statt, beendet man das Verfahren.

Hinweis

Das Verfahren findet typischerweise nur *lokale* Optima. Dadurch ist es oft notwendig, das Verfahren **mehrfach aufzurufen** und das beste gefundene Ergebnis auszuwählen.

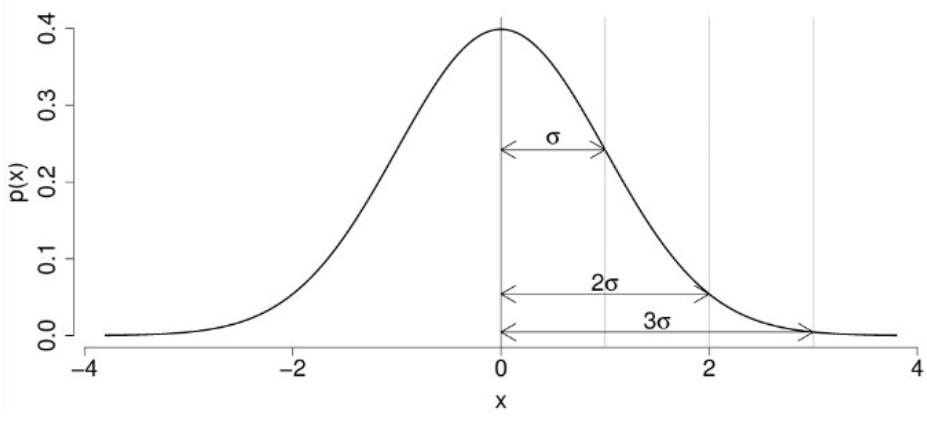
Formulierung als Zufallsexperiment

Die **Wahrscheinlichkeitsdichte** eines Zielwertes bei der Annahme einer **Normalverteilung mit konstanter Varianz** σ^2 der Zufallsvariablen lässt sich darstellen als:

$$p(y_i|h) \propto e^{-\frac{1}{2\sigma^2}\sum_{j=1}^{n} w_{ij}(y_i-\mu_j)^2}$$

Parameter

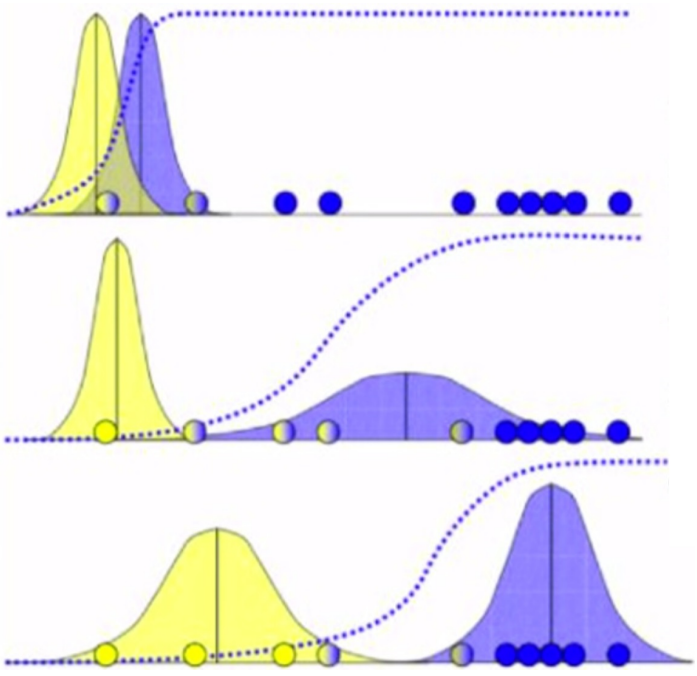
- y_i : Wert der i-ten Zielgröße
- h : Stichprobe
- n : Anzahl der Gewichte
- w_{ij} : Gewicht der j-ten Zufallsvariable für den i-ten Wert der Zielgröße
- μ_j : Erwartungswert der j-ten Zufallsvariable



Normalverteilung für $\mu = 0$ und $\sigma = 1$ in 1 Dimension. Aus: https://learnche.org/

Iterationsschritte

Das EM-Clustering besteht aus mehreren Iterationen der Schritte **Erwartung** und **Maximierung:**



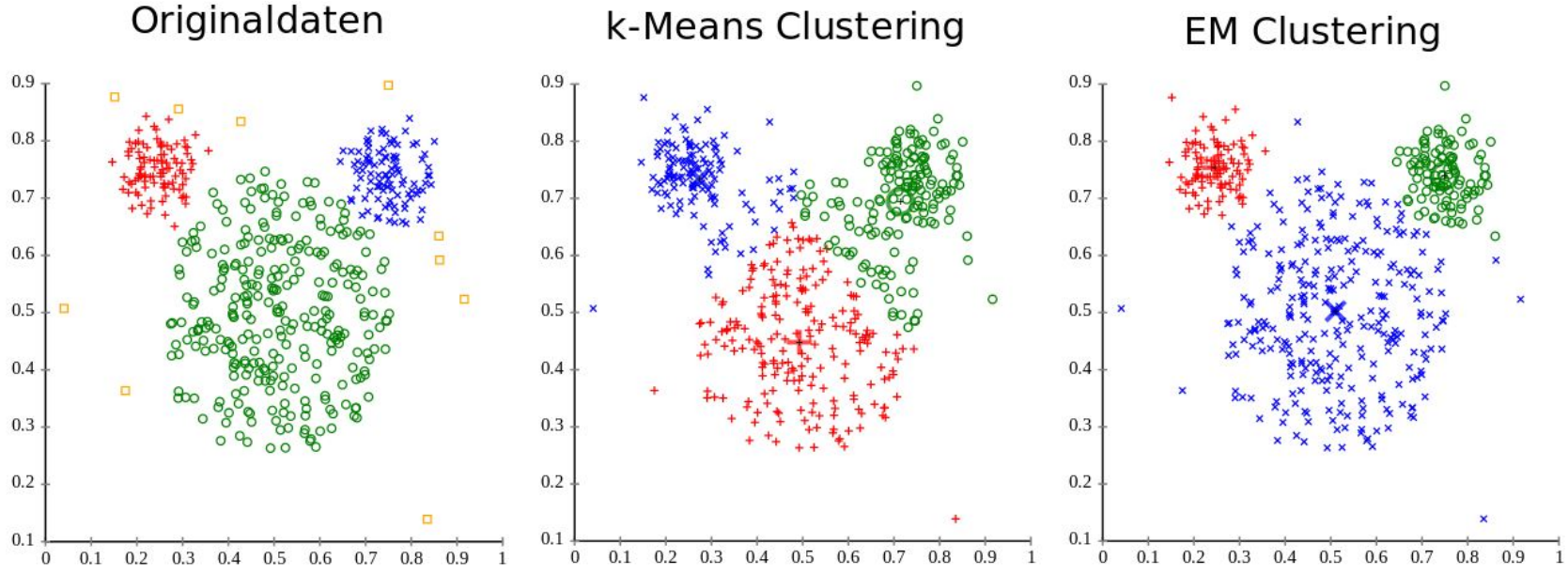
Aus: Lavrenko, https://www.youtube.com/watch?v=iQoXFmbXRJA

- Jeder Cluster wird durch einen Gauss Modus repräsentiert (Bsp. $k = 2$)
- Dargestellt sind die Wahrscheinlichkeitsdichte des ersten **(gelb)** und zweiten **(blau)** Clusters.
- Die gestrichelte Linie **(blaue)** stellt die Verteilungsfunktion (d.h. das Integral der Wahrscheinlichkeitsdichte) des zweiten Clusters dar.

Durch Wiederholung der **Erwartungs-** und **Maximierungsschritte** werden die Parameter ihren tatsächlichen Werten angenähert.

Vorteile EM Clustering

EM Clustering: Modellierung der Daten als **Normalverteilungen**

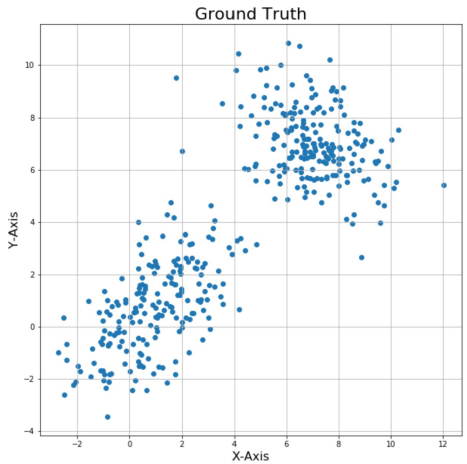


Clustering-Ergebnisse auf dem "Maus"-Datensatz. Durch Verwendung von Varianzen kann EM die unterschiedlichen Normalverteilungen akkurat beschreiben, während k-Means die Daten in ungünstige Voronoi-Zellen aufteilt.

- Erlaubt Cluster unterschiedlicher Größe (im Sinne von Streuung – k-Means ignoriert die Varianz der Cluster).
- Kann korrelierte Cluster erkennen und repräsentieren durch Kovarianzmatrix
- Kann gut mit unvollständigen Beobachtungen umgehen!

Code-Beispiel

Ground Truth



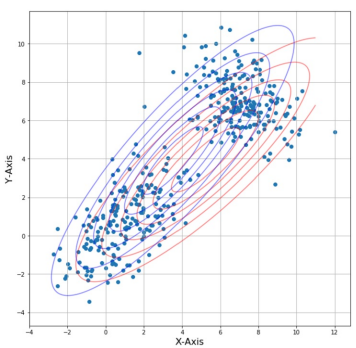
Zufällig generierter Datensatz jeweils einer Normalverteilung (200 Datenpunkte) mit $\mu_1 = [1, 1]$ *und* $cov1 = [[3, 2], [2, 3]]$ *sowie einer weiteren Normalverteilung (200 Datenpunkte) mit* $\mu_2 = [7, 7]$ *und* $cov2 = [[2, -1], [-1, 2]]$. *Aus: medium.com*

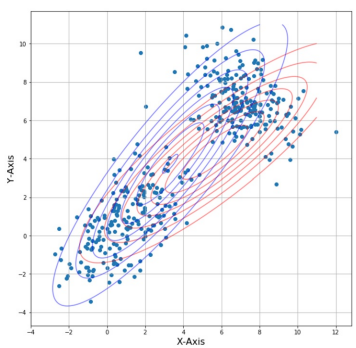
Code für synthetische Erzeugung der Ground Truth

```
x = np.random.multivariate_normal(m1, cov1, size=(200,))
y = np.random.multivariate_normal(m2, cov2, size=(200,))
d = np.concatenate((x, y), axis=0)
```

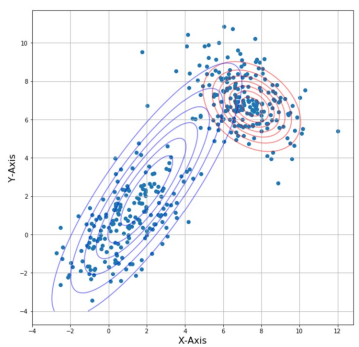
Plot der Iterationen

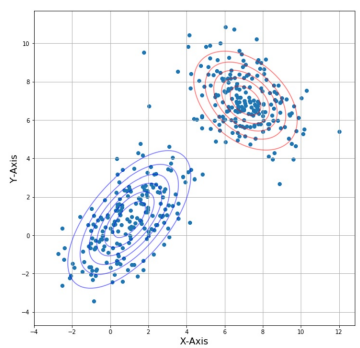
```
iterations = 20
lis1=[m1,m2,cov1,cov2,pi]
for i in range(0,iterations):
    lis2 = Mstep(Estep(lis1))
    lis1=lis2
    if(i==0 or i == 4 or i == 9 or i == 14 or i == 19):
        plot(lis1)
```





• • •





Lernkontrolle

Meine 3 Lernziele für heute waren

1. Ich kenne den grundlegenden Aufbau der vorgestellten Clustering-Algorithmen des maschinellen Lernens.
2. Ich verstehe die Vorgehensweise zur Strukturierung von Clustering-Programmen.
3. Ich kann die Ergebnisqualität eines Clustering-Algorithmus interpretieren und daraus Maßnahmen zur Verbesserung des Agentenprogramms ableiten.