

SEMINAR KLASSIFIKATION & CLUSTERING ÜBERSICHT ALGORITHMEN

Stefan Langer

CIS – Universität München

Wintersemester 2023/24

stefan.langer@cis.uni-muenchen.de

Klassifikation

Eingabe

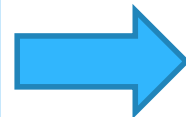
Featurebestimmung

Ausgabe

Training

DT1|K1
DT2|K2
....

Klassifizierte
Trainingsdokumente



Tokenizer
Vectorizer
...

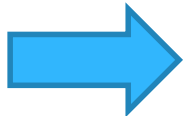


Modell

Klassifikation

D1
D2
...

Zu
klassifizierende
Dokumente



K1
K2
...

Ermittelte
Klassen

Einteilung von Algorithmen

- Überwachte / nicht überwachte Verfahren (supervised/unsupervised)
- Parametrische und nicht-parametrische Verfahren
- Lineare vs. nichtlineare Klassifikatoren

Überwachte vs. nicht-überwachte Verfahren

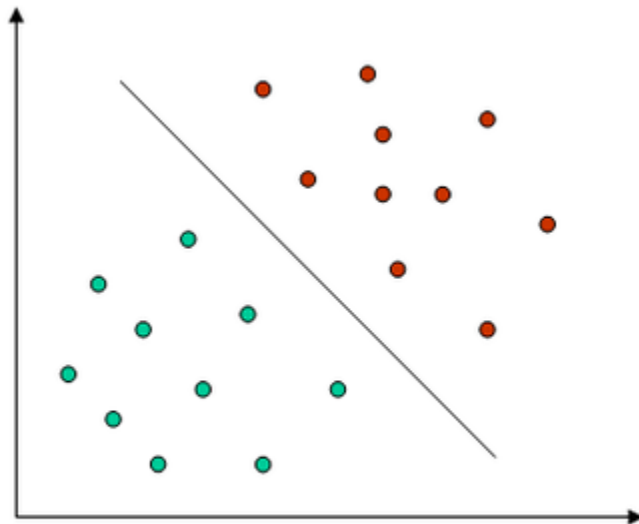
Überwacht	Nicht-überwacht
Trainingsdaten sind vorklassifiziert/vorgeclustert	Unklassifizierte Trainingsdaten
Klassen sind vorgegeben	Klassen bzw. Cluster müssen erlernt werden
Klassifikation	Clustering
z.B. K-Nächster-Nachbar- Klassifikation	z. B. K-Means-Clustering

Parametrische und nicht-parametrische Klassifikationsverfahren

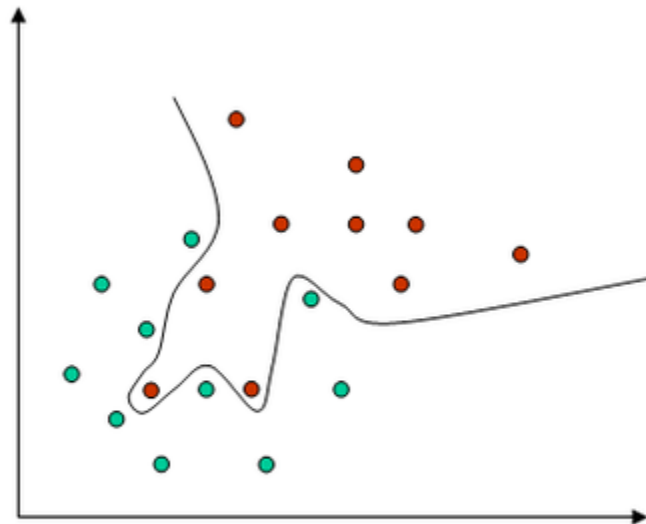
Parametrisch	Nicht-parametrisch (verteilungsfrei)
Geht von Annahmen über die Verteilung der Daten aus (Wahrscheinlichkeitsverteilung)	Annahmen werden aus den Daten ermittelt
Modellstruktur liegt fest	Modellstruktur wird aus den Daten ermittelt
Ermittelt wird die Wahrscheinlichkeit der Zugehörigkeit zu einer Klasse	Ja/Nein Entscheidung
Bsp. Ridge classifier, Perceptron	Bsp K-Nächster-Nachbar

Lineare / nicht-lineare Klassifikatoren

- Lineare Klassifikatoren trennen Klassen durch eine Hyperebene



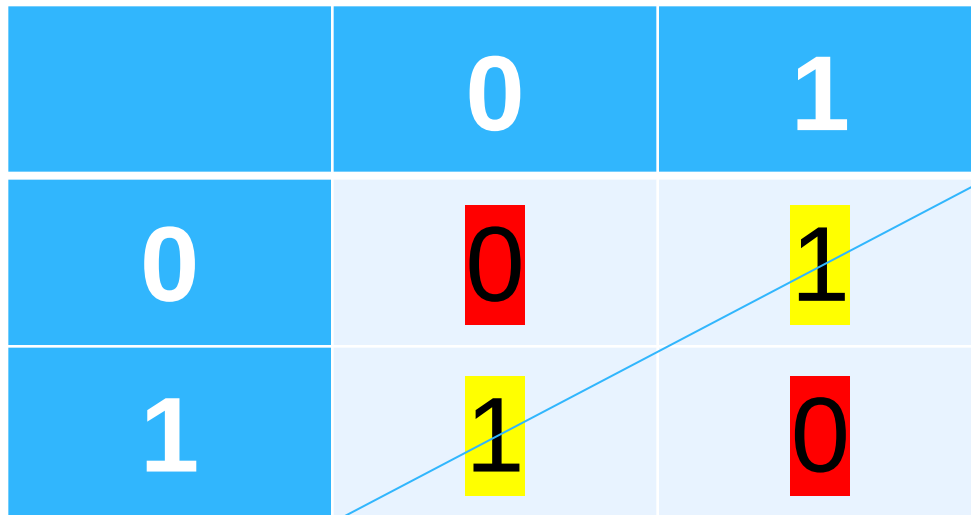
linear trennbar



nicht linear trennbar

Linear & X-OR

	0	1
0	0	1
1	1	0



- Naive training algorithms for e.g. the Perceptron algorithm cannot cope with data which is not linear separable
- → Training algorithms need to be able to cope with non-linear data

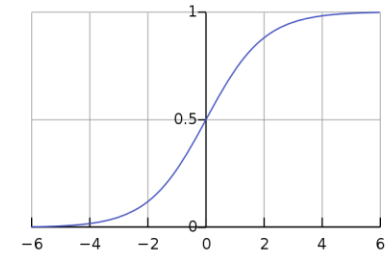
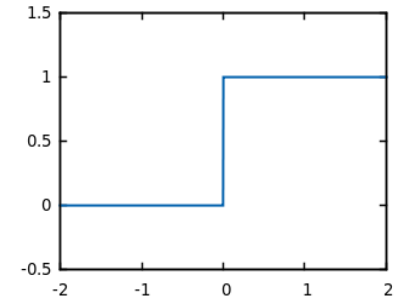
Grundidee lineare Funktion

- $F(x, w) = w_0 \sum_1^n w_i * x_i$

- x : Features
- w : Gewichte

Lineare Klassifikatoren

- Perzeptron:
 - Schwellenwertfunktion + lineare Grundfunktion
- Logistic Regression:
 - Logistische Funktion (sigmoid), logit-Funktion



Loss function – cost function

- Anwendung: Training
- Funktion, welche ein Maß für die Abweichung von den Trainingdaten berechnet
- Verlustfunktion: Pro Datenpunkt
- Kostenfunktion: Für alle Datenpunkte

Ziel beim Training ist es, den Wert der Kostenfunktion zu minimieren.

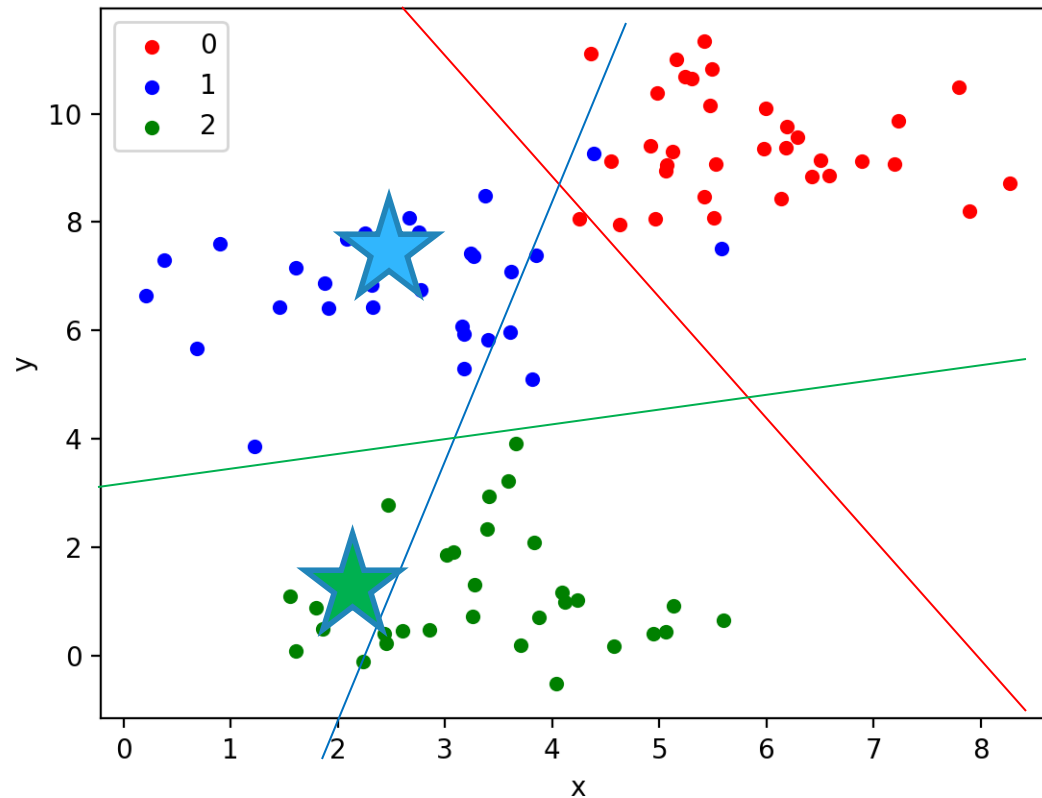
Lineare Multi-Klassen-Klassifikation

- Problem: Ein binärer Klassifikator trennt zwei Klassen
 - Z.B. Perceptron, logistic regression, support vector machines
- Frage: Wie bilde ich dies auf ein Mehrklassenproblem ab
- Option A: Einer gegen alle anderen (one versus rest)
- Option B: Einer gegen einen (one versus one)

One versus Rest

- Bei N Klassen, trainiere N-1 Klassifikatoren, jeweils 1-Klasse gegen alle anderen (i.e. N-1 binäre Probleme)
- Ausgabe: Wähle die Klasse mit dem höchsten Konfidenzwert
- Anforderungen / Probleme:
 - Konfidenzwert/Ausgabescore muss vergleichbar sein (bei linearen Klassifikatoren: Abstand zur Hyperebene)
 - Nicht-ausgewogenes Trainingsset für jeden Classifier (in der Regel mehr negative Beispiele pro Klassifikator)

Lineare Klassifikatoren – multiclass 1:all



One versus One

- Bei N Klassen, trainiere jeweils für jedes Paar von Klassen einen Klassifikator
- Ausgabe: Wähle die Klasse, die bei den meisten Vergleichen gewinnt
- Anforderungen / Probleme:
 - Sehr viele Klassifikatoren: $(N * (N - 1)) / 2$
 - Pattsituationen möglich, wenn der Konfidenzwert nicht berücksichtigt wird

Multi-Klassen für einige Algorithmen

Classifier	Inhärent Multi	One vs. One One vs. all	Alternativen / Varianten für Multiklassen
Logistic Regression	-	+	Multinomial logistic regression
K-Nearest Neighbour	+	n/a	n/a
SVM	-	+	Crammer-Singer multiclass SVM
Decision trees & Random Forest	+	n/a	n/a
Perceptron	-	+	(multilayer perceptron)
Multilayer perceptron, neural networks	+	n/a	n/a

Algorithmen - Klassifikation

Lineare Klassikatoren:

- Perceptron
- Ridge classifier, logistic regression
- Support Vector (SVM)
- Naïve Bayes

Entscheidungsbäume:

- Entscheidungsbäume (decision trees)
- Random Forest

Parameterfreie Klassifikation:

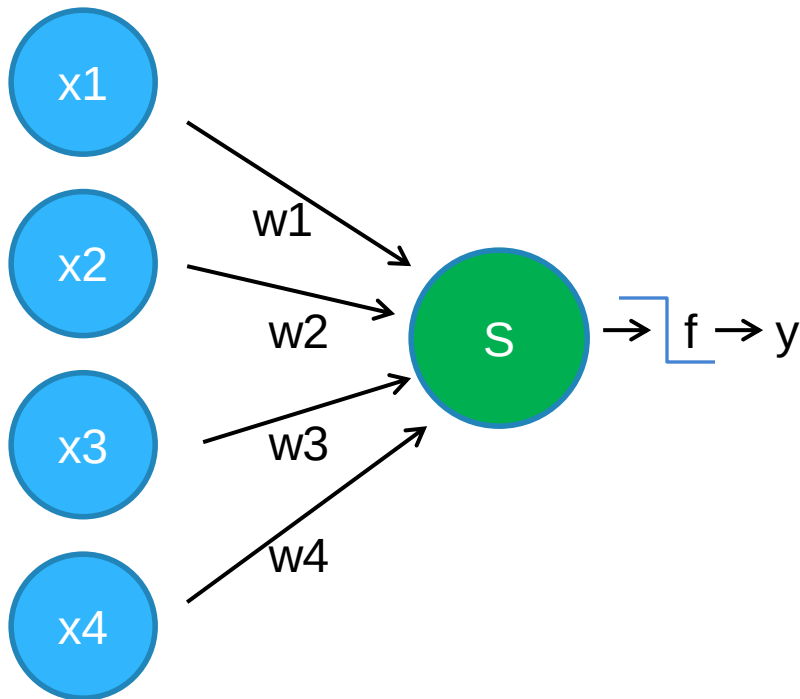
- K-nearest neighbor

Neuronale Netze:

- Perceptron
- Neural networks; Transformers, large language models

PERZEPTRON

Ein einfaches Perzeptron ist ein einfaches neuronales Netz das nur aus gewichteten Eingaben und einem Summen/Ausgabeknoten



$x_1 - x_4$: Eingabedaten

$w_1 - w_4$: Gewichtung

S : Knoten mit Summenfunktion

f : Schwellenwertfunktion

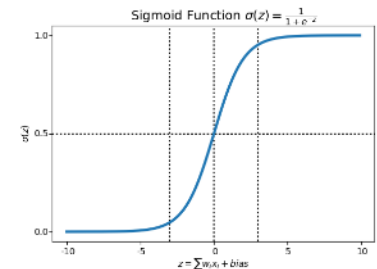
y : Output (0 oder 1)

Zu lernen: $w_1 - w_4$

Eigenschaften: Linearer
Klassifikator (Daten sollten über
Hyperebene separierbar sein)

Logistic regression

- Basis ist die logit (log-odds)-Funktion für die Wahrscheinlichkeit p , dass der Wert eine binären Klasse 1 ist:
 - $\text{logit}(p) = \log p/(1-p) = b_0 + b_1 \cdot x_1 + b_2 \cdot x_2 \dots$
 - Durch Umformung:
 - $p = 1 / (1 + 10^{-(b_0 + b_1 \cdot x_1 + b_2 \cdot x_2)}) \rightarrow \text{Sigmoid-Funktion}$
- Training: Parameter-Ermittlung von $b_0 \dots b_n$



Bayes

$$P(K|D) = \frac{P(D|K)P(K)}{P(D)}$$

Wir wollen nun die beste Klasse ermitteln.

$P(D)$ ist für ein gegebenes Dokument konstant. Damit:

$$\operatorname{argmax} P(K|D) = \operatorname{argmax} P(D|K)P(K)$$

Nimmt man an, dass alle Klassen gleich wahrscheinlich sind:

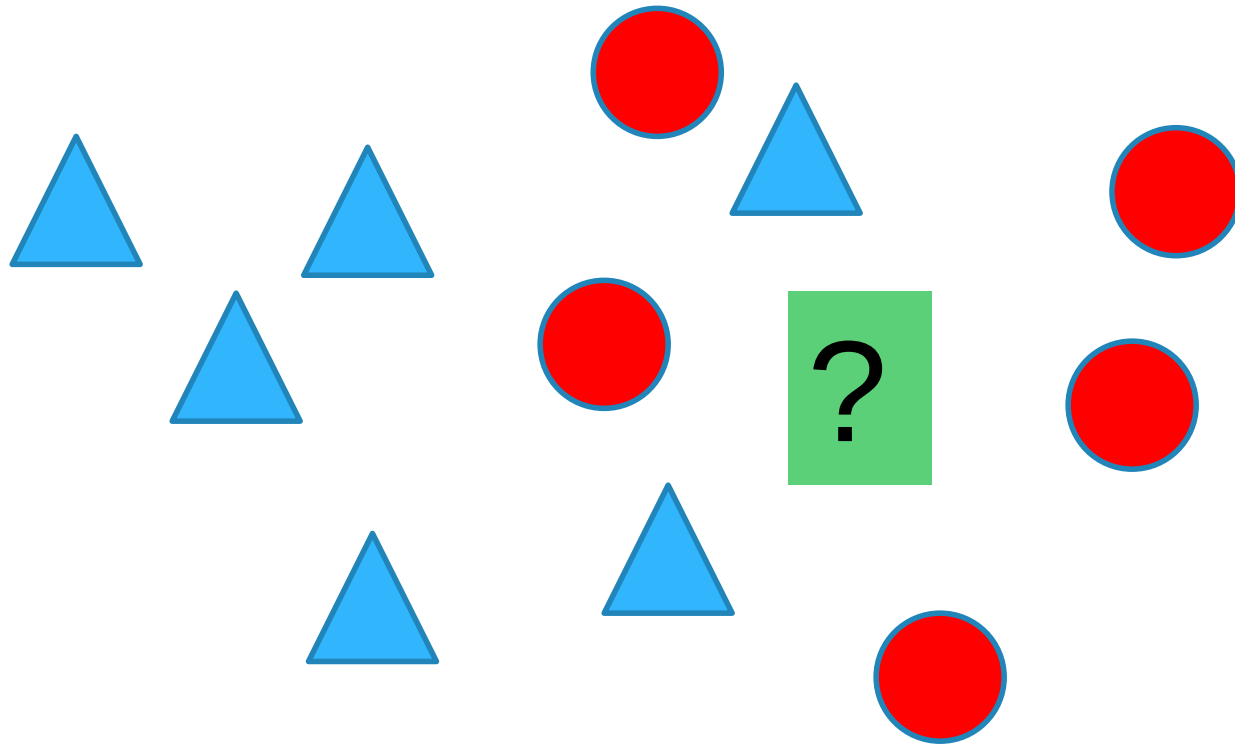
$$= \operatorname{argmax} P(D|K)$$

Nächste-Nachbar-Klassifikation

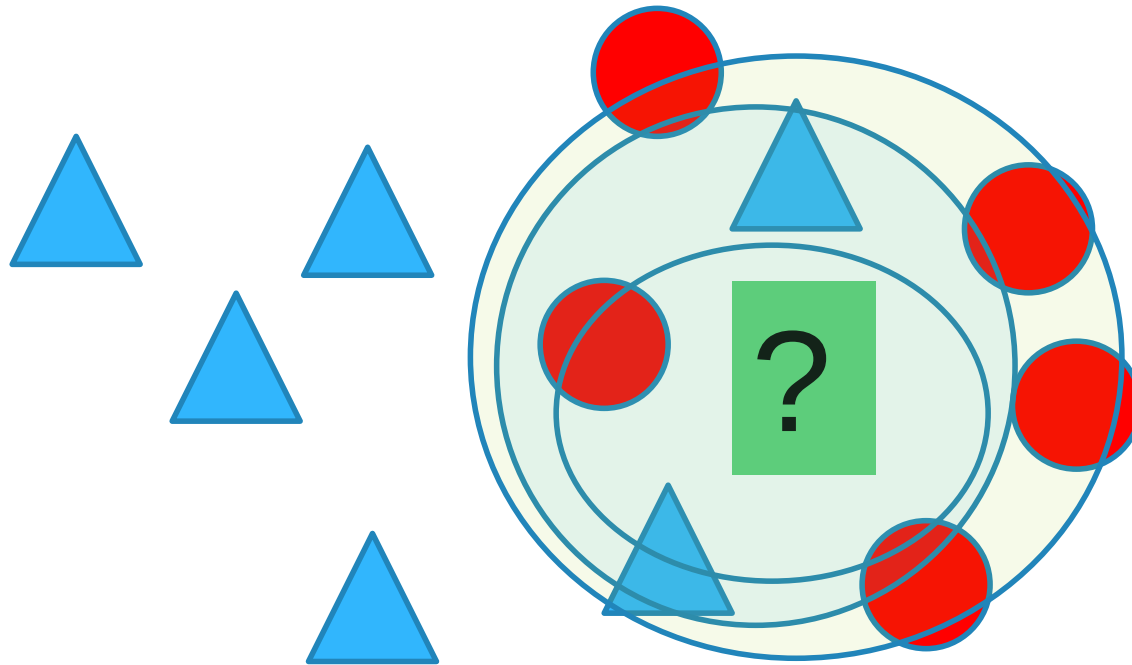
Grundidee:

- Klassifizierung eines Objekts aufgrund der Klassen der k nächsten Objekte.
- Training: Abspeichern aller Elemente des Trainingskorpus
- Klassifizierung: Berechnen der nächsten Elemente für ein neu zu klassifizierendes Element
- Möglichkeiten:
 - Vektorenabstandsmaße
 - Word embeddings: 'word movers distance'

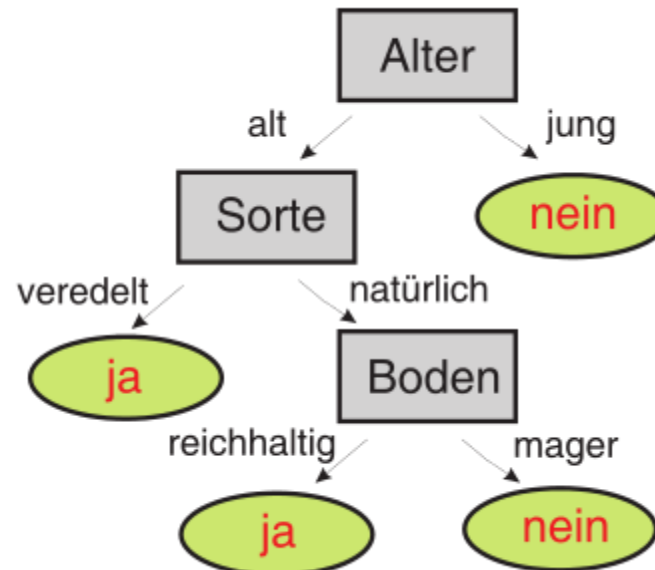
Beispiel (zwei-dimensionaler Raum)



Beispiel $k = 1; 3; 5$



Decision trees / Entscheidungsbäume



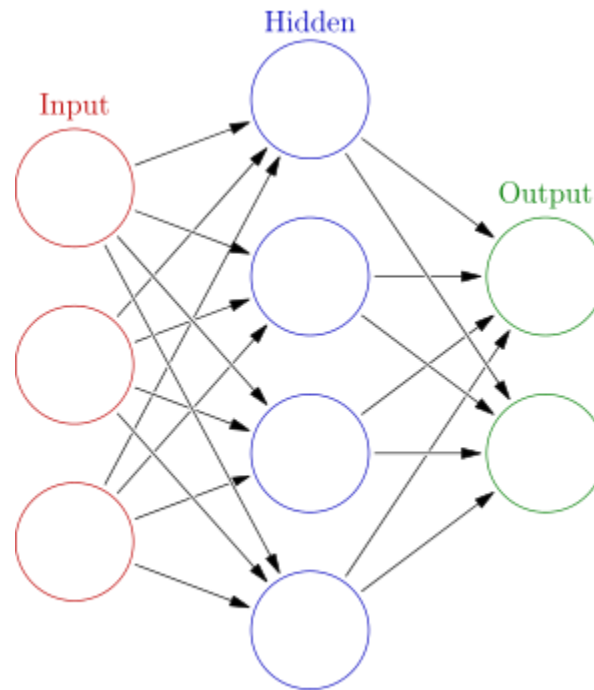
Quelle: de.wikipedia.org

Random forest



- Kombiniere mehrere Entscheidungsbäume
 - Untermengen von Merkmalen und Trainingdaten für jeden Baum
- Mehrheitsentscheidung

Neural networks / neuronale Netze



Künstliches Neuronales Netz (Quelle: Wikipedia - en)

Deep learning

- Scikit learn: Multi-layer-Perceptron
- Basis: neuronale Netze
- + verbesserte Algorithmen
- + mehr Zwischenschichten
- + mehr Rechenleistung
- Transformer
- Large Language Models – GPT, Llama2 ...

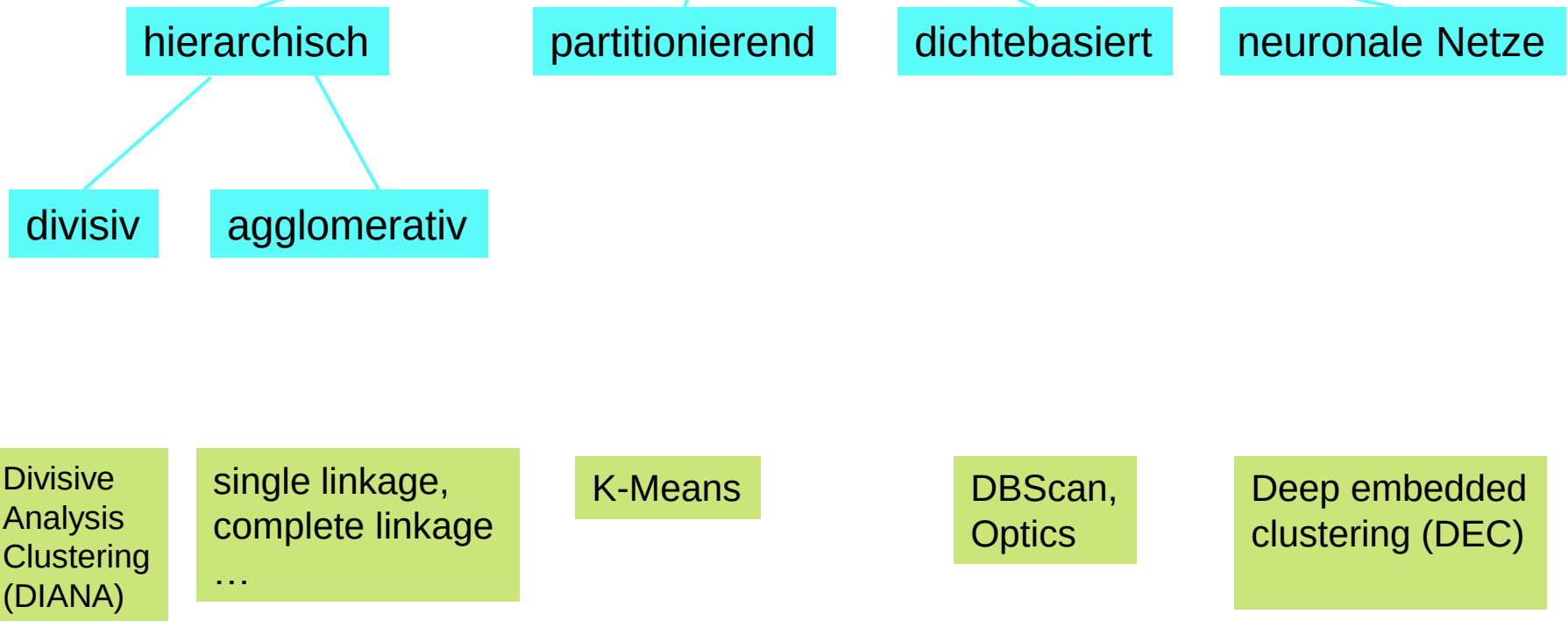
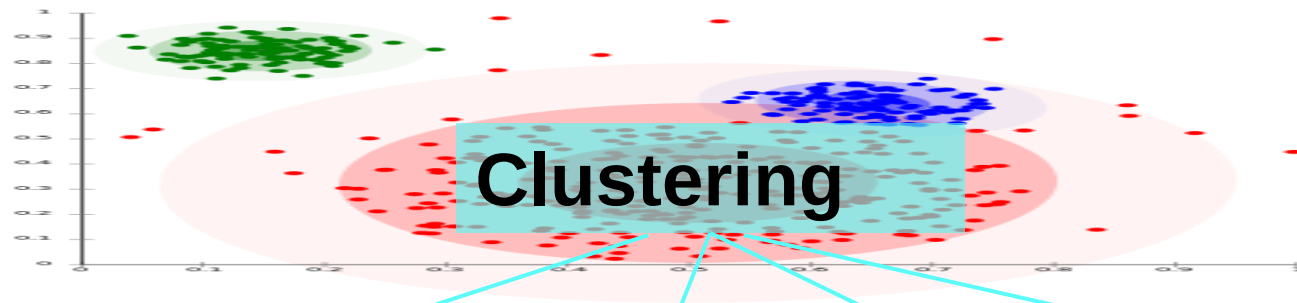
→ Eine der nächsten Sitzungen: Neuronale Architekturen

Clustering vs. Klassifikation

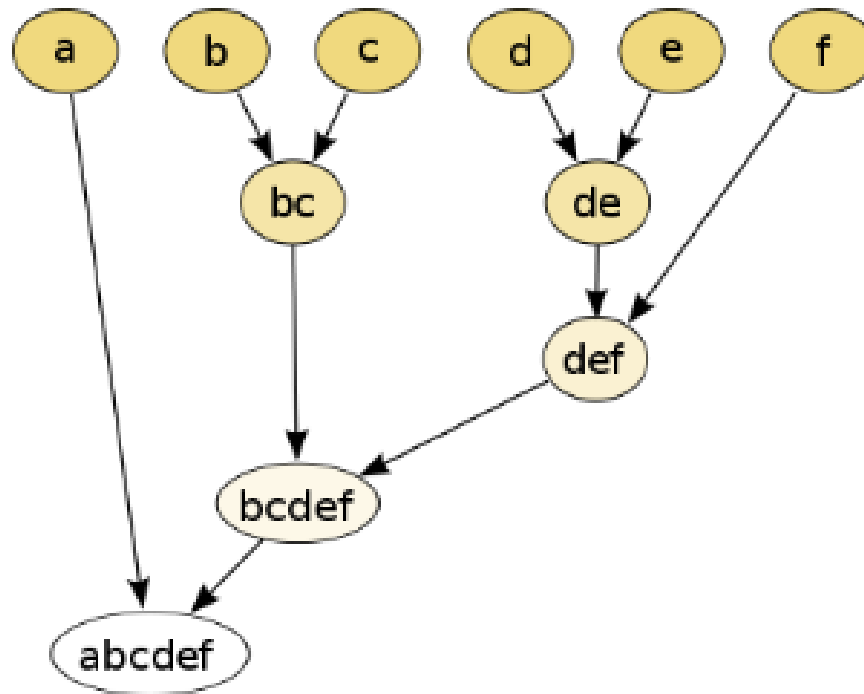
Klassifikation	Clustering
Vordefinierte Klassen	Cluster werden algorithmisch erzeugt
Trainingskorporus erforderlich – Überwachtes Lernen (supervised)	Kein Trainingskorporus erforderlich – Unüberwachtes Lernen (unsupervised)
Basis: Ähnlichkeitsmaß zwischen Objekten	Basis: Ähnlichkeitsmaß zwischen Objekten
Hierarchisch oder flach	Hierarchisch oder flach
Einfache Algorithmen verfügbar (z.B. Perceptron)	Relativ komplexe und aufwändige Algorithmen
Evaluation nach Recall/Precision u.ä. Maßen; Testkorporus erforderlich	Evaluation schwierig; Testkorporus erforderlich; Erstellungskriterien unklar -

Ähnlichkeit – was ist das?

- Dokumente
 - Ähnliches Thema
 - Ähnliche Protagonisten
 - Z.B. Sportler im Sportteil; Sportler im Vermischten; Ortsnamen ...
 - Ähnliches Genre (z.B. Literatur, Nachrichtentext, Social Media)
 - Gleiche Quelle (z.B. Wikipedia-Artikel; Artikel aus der Süddeutschen Zeitung)
 - AutorIn (z.B. bei Briefen)
- Wörter
 - Semantisch – taxonomisch - ontologisch
 - Ontologiebasierte Beispiele : Berufsbezeichnungen, Tiere
 - Semantische Felder
 - Ähnliche Distribution
 - Wortarten



Clustering-Algorithmus: Hierarchisches Clustering



(Graphik aus en.wikipedia)

Hierarchische Clustering

Algorithmen zur Abstandbestimmung zwischen zwei Clustern

$d(x,y)$ ist die Distanz zwischen zwei Objekten;

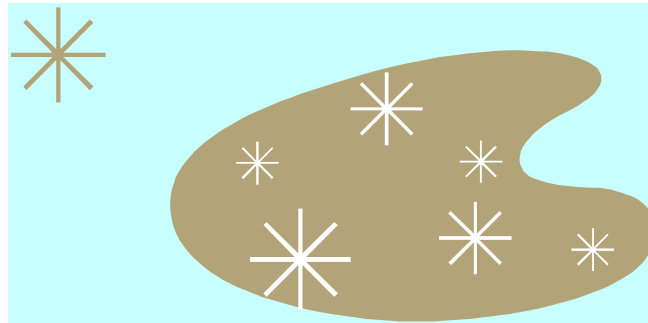
$D(X,Y)$ ist die Distanz zwischen zwei Clustern;

- Single Linkage - $D(X,Y) = \min(d(x,y))$
- Complete Linkage - $D(X,Y) = \max(d(x,y))$
- Average Linkage - $D(X,Y) = \text{average}(d(x,y))$

K-means Clustering

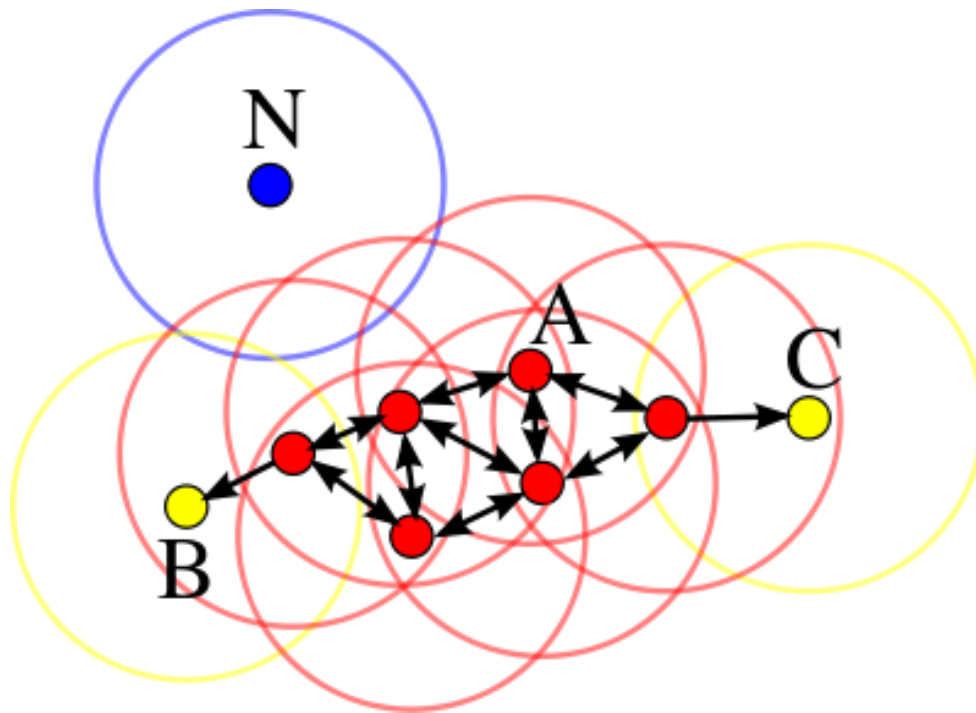
- Centroid-basiert
- Jedes Cluster wird von einem zentralen Vektor repräsentiert
- Minimiere die quadrierten Abstände
- Initiale Clusterzentren zufällig gewählt
- Zahl der Cluster wird vorgegeben

Dichtebasiertes Clustering



- Grundidee: Bereiche mit hoher Dichte bilden ein Cluster
- Bekannteste Methode ist DBSCAN
- Gewisse Ähnlichkeiten mit k-Nearest Neighbor in der Klassifikation

DBSCAN(Source: Wikipedia)



DBSCAN – Algorithm

Zwei Parameter: Epsilon (maximaler Abstand zwischen zwei Punkten) und die Minimalzahl von Elementen in einem Cluster (minPts).

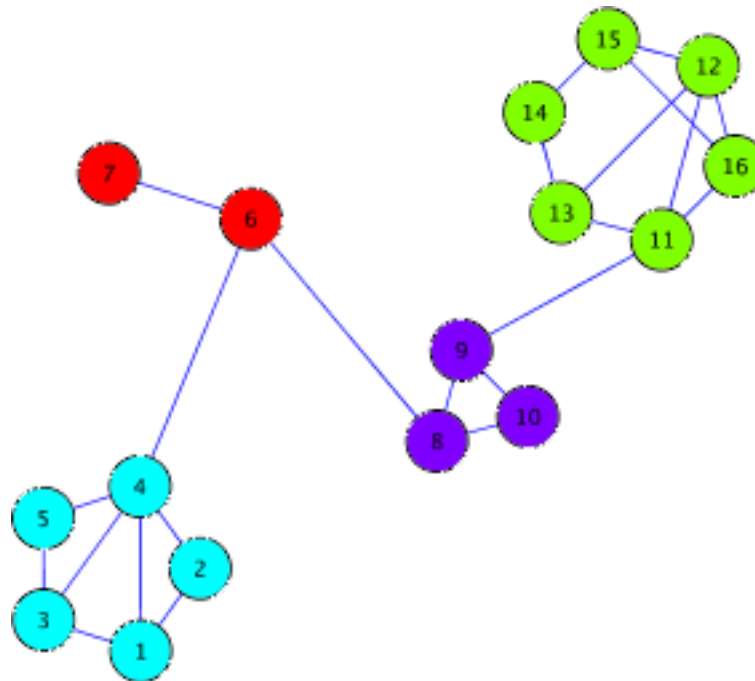
- Initialisierung mit einem beliebigen Startpunkt, der noch nicht besucht wurde.
 - Die Epsilon-Nachbarschaft dieses Punktes wird abgerufen, und wenn sie ausreichend viele Punkte enthält, wird ein Cluster gebildet. Andernfalls wird der Punkt als Rauschen eingestuft. Es ist zu beachten, dass dieser Punkt später in einer ausreichend großen Epsilon-Umgebung eines anderen Punktes gefunden werden kann und somit Teil eines Clusters wird.
 - Wenn ein Punkt als dichter Teil eines Clusters gefunden wird, ist sein Epsilon ebenfalls Teil dieses Clusters.
 - Daher werden alle Punkte, die sich innerhalb der Epsilon-Nachbarschaft befinden, hinzugefügt, ebenso wie ihre eigene Epsilon-Nachbarschaft, wenn sie ebenfalls dicht sind.
 - Dieser Prozess wird so lange fortgesetzt, bis der mit der Dichte assoziierte Cluster vollständig gefunden ist.
- Dann wird ein neuer, nicht besuchter Punkt abgerufen und verarbeitet, was zur Entdeckung eines weiteren Clusters oder Rauschens führt.

DBSCAN - Erweiterungen

- OPTICS - Ordering Points To Identify the Clustering Structure
- Shared-Nearest-Neighbor-Clustering - Finding Clusters of Different Sizes, Shapes, and Densities in Noisy, High Dimensional Data
- PreDeCon - Density Connected Clustering with Local Subspace Preferences
- SubClu - Density connected Subspace Clustering for High Dimensional Data
- 4C - Computing Clusters of Correlation Connected Objects
- ERiC - Exploring Complex Relationships of Correlation Clusters

Graphen-Clustering

- Aufgabe: Identifizierung von Clustern in einem Graphen
- Beispiel: Facebook social graph



Graph-Clustering

- Clustering im Graph: Grundlagen
 - Ziel: besonders eng verbundene Elemente im Graph als Cluster identifizieren
 - Parameter
 - Interne Dichte eines Clusters
 - Anteil Knoten im Cluster, mit denen ein Knoten verbunden ist
 - Externe Dichte
 - Zahl der Verbindung von Knoten außerhalb des Clusters zu Knoten innerhalb des Clusters
 - Cluster sind zusammenhängende Knoten mit hoher interner Dichte und geringer externer Dichte

Graph-Clustering

- α/β -Cluster
- **Interne Dichte:** Jeder Knoten im Cluster hat mindestens einen Anteil von β Verbindungen innerhalb des Clusters
- **Geringe externe Dichte:** Jeder Knoten außerhalb des Clusters hat höchstens Verbindung zum Anteil α innerhalb des Clusters
- Es gilt stets das $\alpha < \beta$

Zusätzlich:

- Ein Knoten ist ein p -champion von C wenn er höchstens $p|C|$ Nachbarn außerhalb von C hat
- Wenn $p < 2\beta - 1 - \alpha$, kann jeder Knoten p -Champion in einem Cluster sein.
- MISHRA, Nina, et al. Clustering social networks. In: *Algorithms and Models for the Web-Graph*. Springer Berlin Heidelberg, 2007. S. 56-67

Open source – Klassifikation & Clustering

Generic

- Scikit learn

Deep learning

- TensorFlow
- Keras
- pytorch

Some others

- Apache Mahout - machine learning - clustering, classification
- Carrot2 – Clustering
- ELKI (LMU, Kriegel) – Schwerpunkt Clustering
 - <https://elki-project.github.io/>
- WEKA
- Orange
 - <https://orangedatamining.com/docs/>
- Stanford classifier (maximum entropy classifier)
- KNIME (Generic data processing framework)