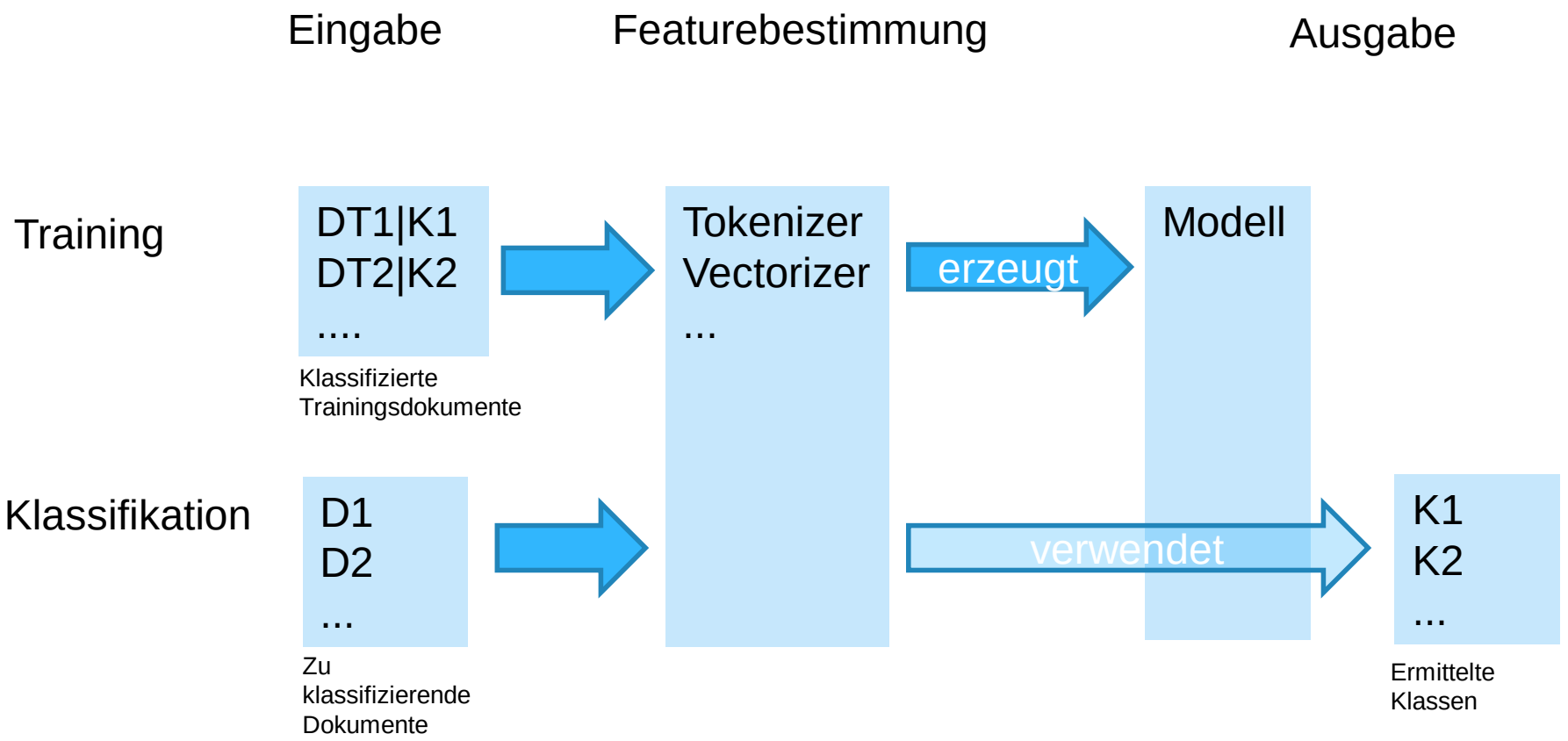
SEMINAR
KLASSIFIKATION & CLUSTERING
ÜBERSICHT ALGORITHMEN

Stefan Langer
CIS – Universität München
Wintersemester 2022/23
stefan.langer@cis.uni-muenchen.de

Klassifikation



Einteilung von Algorithmen

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

Überwachte vs. nicht-überwachte Verfahren

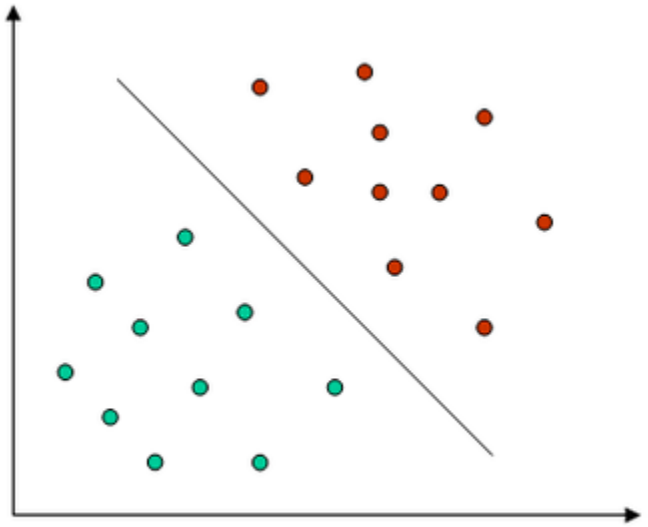
Überwacht	Nicht-überwacht
Trainingsdaten sind vorklassifiziert/vorgeclustert	Unklassifizierte Trainingsdaten
Klassen sind vorgegeben	Klassen bzw. Cluster müssen erlernt werden
Klassifikation	Clustering
z.B. K-Nächster-Nachbar-Klassifikation	z. B. K-Means-Clustering

Parametrische und nicht-parametrische Klassifikationsverfahren

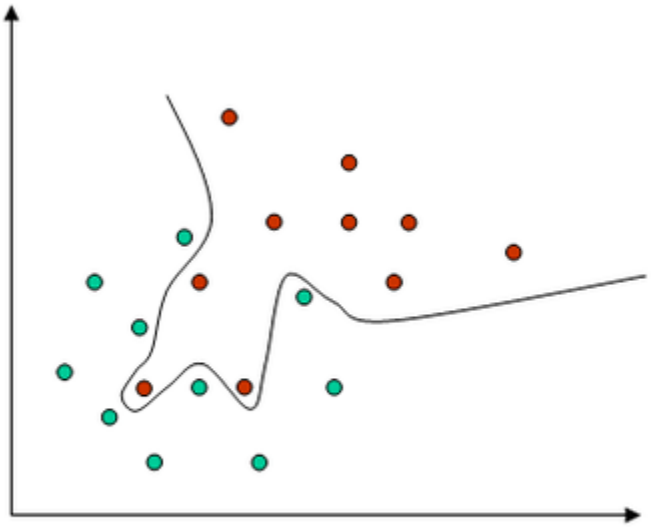
Parametrisch	Nicht-parametrisch (verteilungsfrei)
Geht von Annahmen über die Verteilung der Daten aus (Wahrscheinlichkeitsverteilung)	Annahmen werden aus den Daten ermittelt
Modellstruktur liegt fest	Modellstruktur wird aus den Daten ermittelt
Ermittelt wird die Wahrscheinlichkeit der Zugehörigkeit zu 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

Lineare Multi-Klassen-Klassifikation

- Probem: Ein binärer Klassifikator trennt zwei Klassen
  - Z.B. Perceptron, logistic regression, support vector machines
- Frage: Wie bilde ich dies auf ein Mehrklassenproblem ab

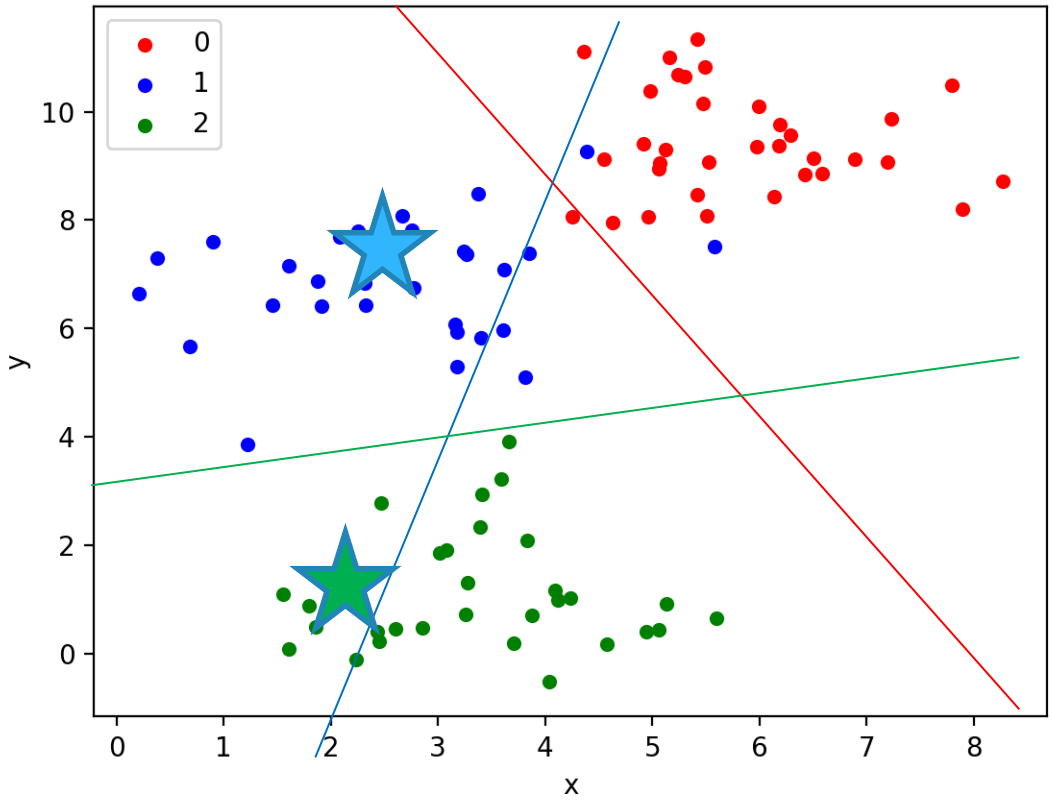
- Option A: Einer gegen alle anderen (one versus rest)
- Option B: Einer gegen einen (one versus one)

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

Entscheidungsbäume:

- Entscheidungsbäume (decision trees)
- Random Forest

Lineare Klassikatoren:

- Ridge classifier, logistic regression
- Naïve Bayes
- Support Vector

Parameterfreie Klassifikation:

- K-nearest neighbor

Neuronale Netze:

- Perceptron
- Neural networks; Transformer

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:

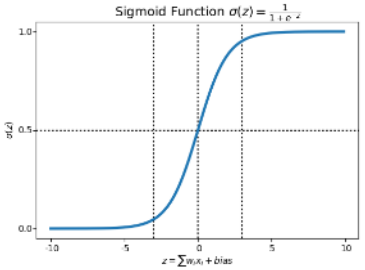
$$\text{argmax}\ P(K|D) = \text{argmax}\, P(D|K)P(K)$$

Nimmt man an, das alle Klassen gleich wahrscheinlich sind:

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

Logistic regression

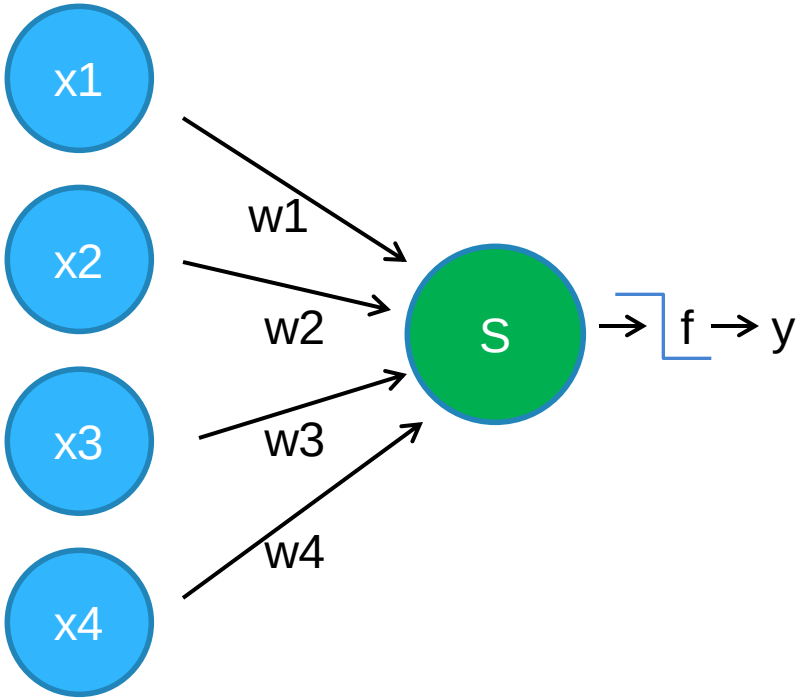
- Basis ist die logit (log-odds)-Funktion für die Wahrscheinlichkeit p, dass der Wert eine binären Klasse 1 ist:
  - logit (p) = log p/(1-p) = b0 + b1*x1 + b2*x2 ....
  - Durch Umformung:
    - p = 1 / (1 + 10 $^{-(b0 + b1*x1 + b2*x2)}$) → Sigmoid-Funktion



- Training: Parameter-Ermittlung von b0...bn

PERZEPTRON

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



x1 – x4: Eingabedaten

w1 – w4: Gewichtung

S: Knoten mit Summenfunktion

f: Schwellenwertfunktion

y: Output (0 oder 1)

Zu lernen: w1-w4

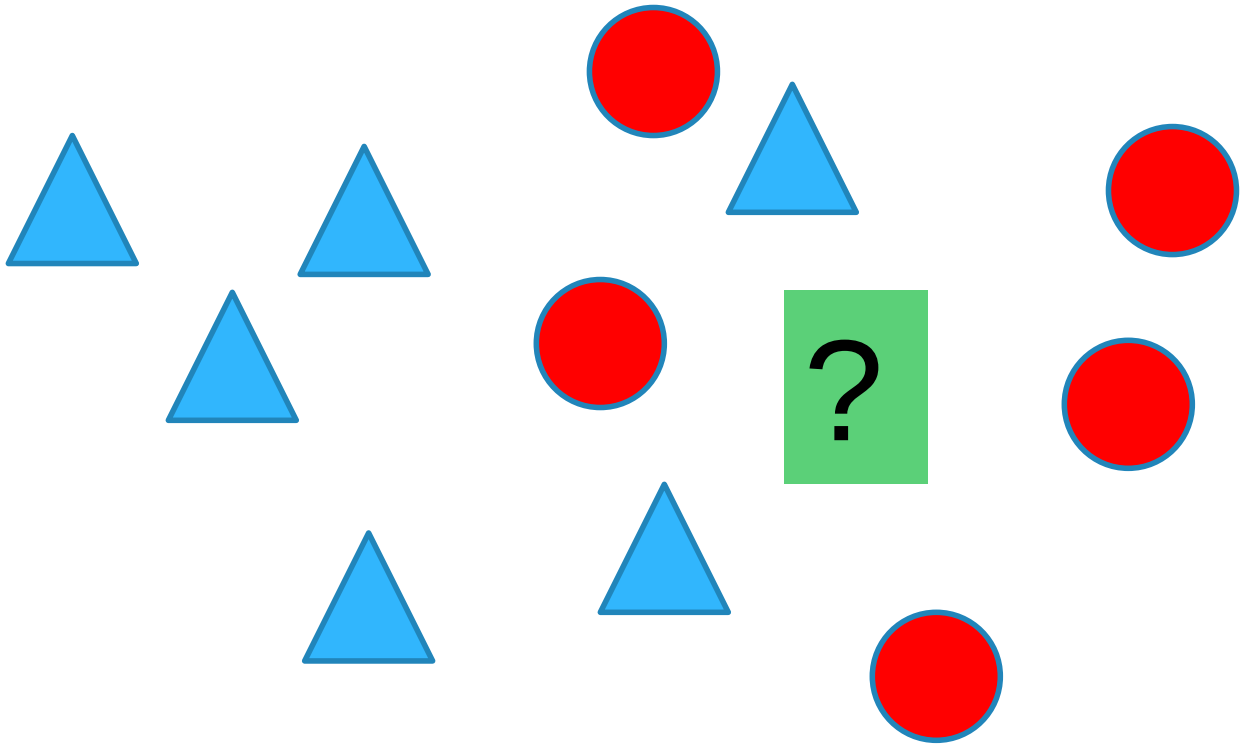
Eigenschaften: Linearer Klassifikator (Daten sollten über Hyperebene separierbar sein)

Nächste-Nachbar-Klassifikation

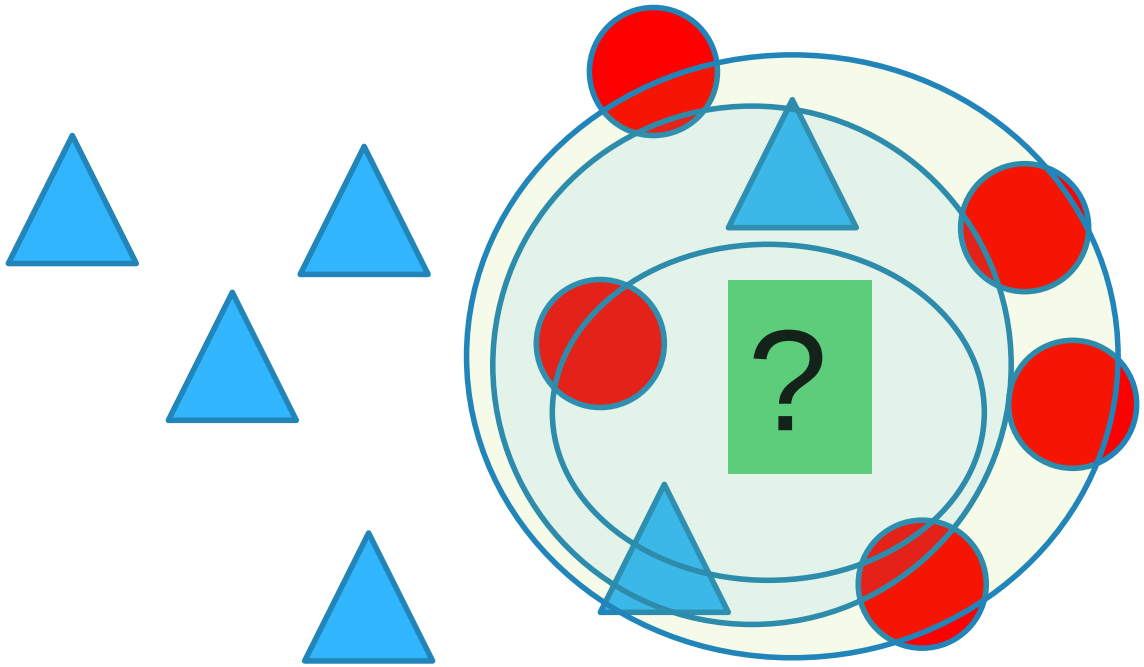
Grundidee:

- 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’

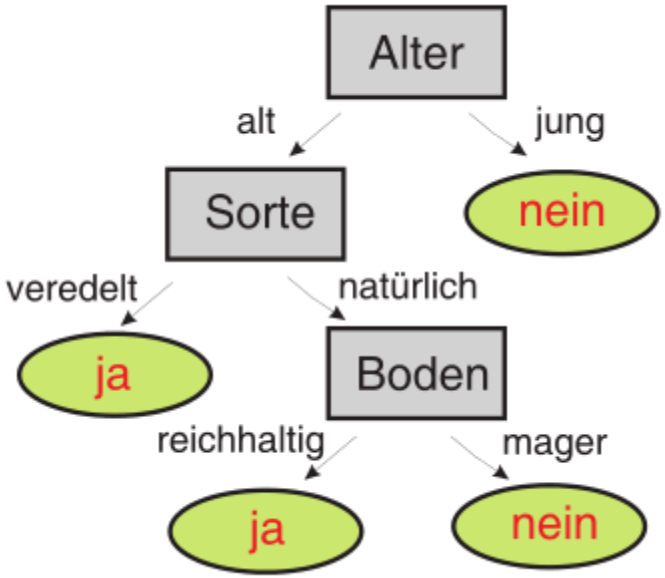
Beispiel (zwei-dimensionaler Raum)



Beispiel k = 1; 3; 5



Decision trees / Entscheidungsbäume

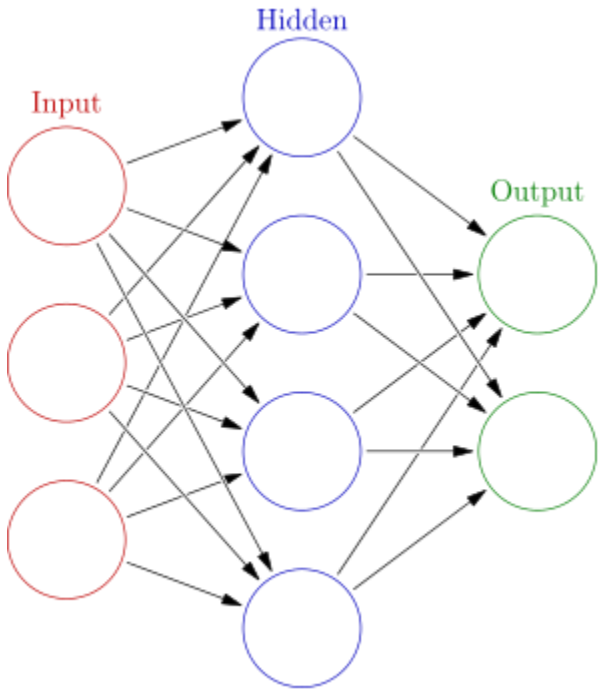


Quelle: de.wikipedia.org

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

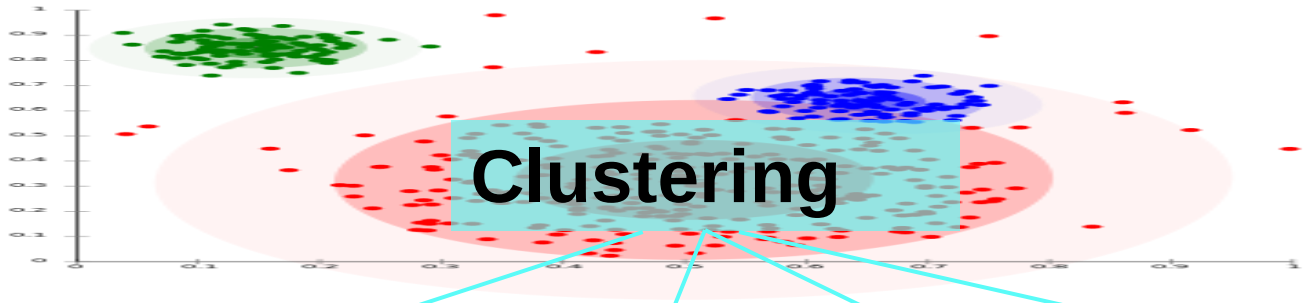
→ Übernächste Sitzung: Neuronale Architekturen

Clustering vs. Klassifikation

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

Ähnlichkeit – was ist das?

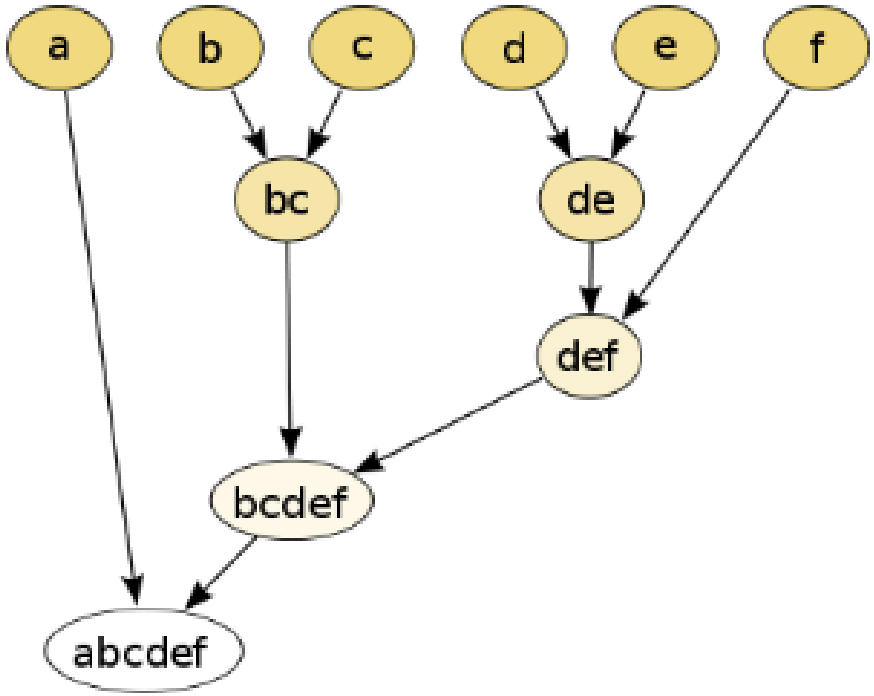
- Dokumente
  - Ähnliches Thema
  - Ähnliche Protagonisten
    - Z.B. Sportler im Sportteil; Sportler im Vermischten; Ortsnamen …
  - Ähnliches Genre (z.B. Literatur, Nachrichtentext, Social Media)
  - Gleiche Quelle (z.B. Wikipedia-Artikel; Artikel aus der Süddeutschen Zeitung)
  - AutorIn (z.B. bei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



Clustering

- hierarchisch
  - divisiv
    - Divisive Analysis Clustering (DIANA)
  - agglomerativ
    - single linkage, complete linkage …
- partitionierend
  - K-Means
- dichtebasiert
  - DBScan, Optics
- neuronale Netze
  - Deep embedded clustering (DEC)

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;

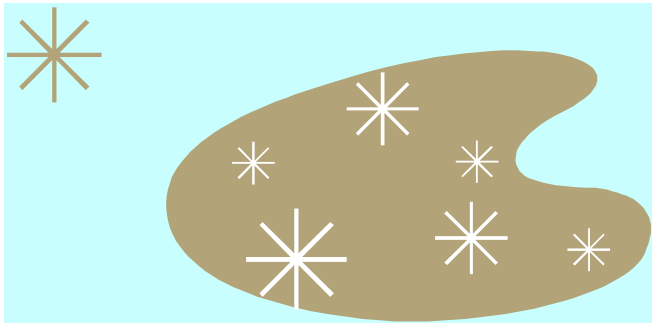
D(X,Y) ist die Distanz zwischen zwei Clustern;

- 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

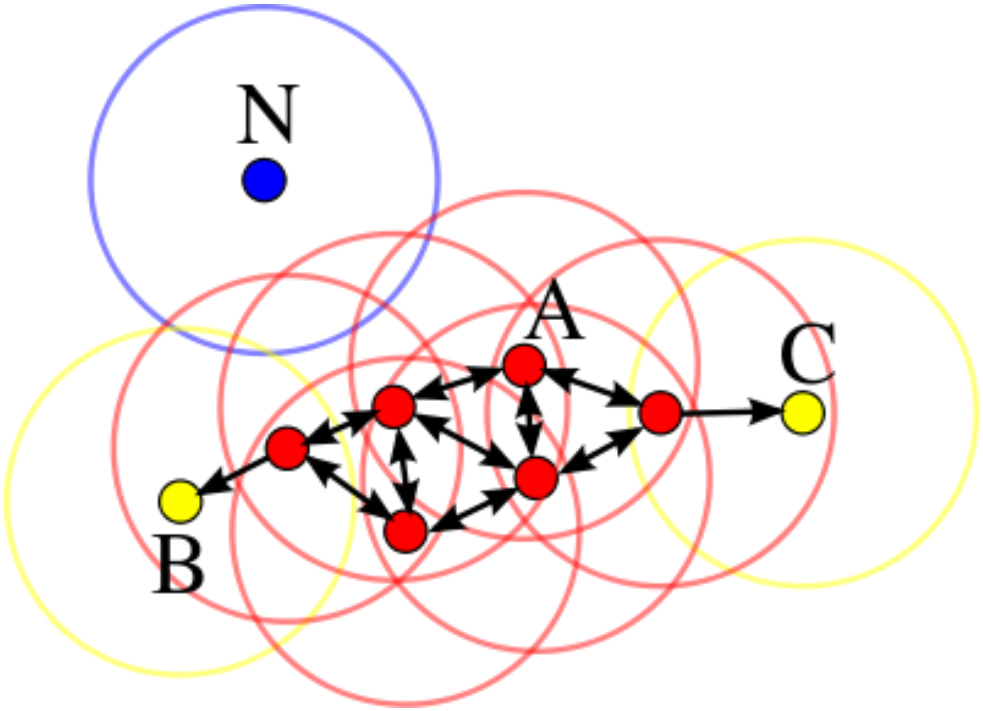
- Centroid-basiert
- Jedes Cluster wird von einem zentralen Vektor repräsentiert
- Minimiere die quadrierten Abstände
- Initiale Clusterzentren zufällig gewählt
- Zahl der Cluster wird vorgegeben

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 (from Wikipedia)

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

- It starts with an arbitrary starting point that has not been visited. This point's epsilon-neighborhood is retrieved, and if it contains sufficiently many points, a cluster is started. Otherwise, the point is labeled as noise.
  - Note that this point might later be found in a sufficiently sized epsilon-environment of a different point and hence be made part of a cluster.

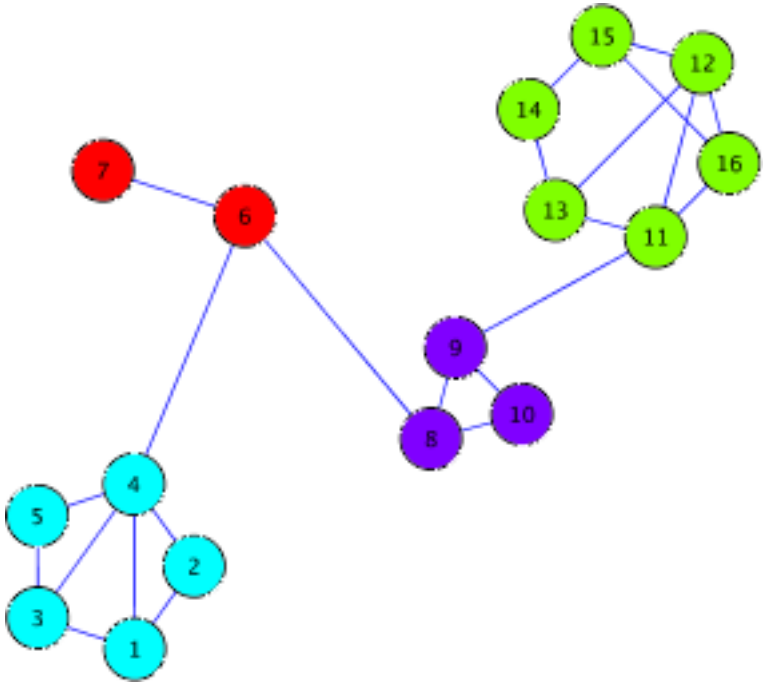
- If a point is found to be a dense part of a cluster, its epsilon is also part of that cluster.
- Hence, all points that are found within the epsilon-neighborhood are added, as is their own epsilon-neighborhood when they are also dense.
- This process continues until the density-connected cluster is completely found.
- Then, a new unvisited point is retrieved and processed, leading to the discovery of a further cluster or noise.

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 ρ-champion von *C* wenn er höchsten ρ|C| Nachbarn außerhalb von *C* hat
- Wenn $\rho < 2\beta - 1 - \alpha$, kann jeder Knotenρ-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 und Clustering

- Scikit learn
- TensorFlow
- Keras / KTrain
- pytorch
- Apache Mahout - machine learning - clustering, classification and collaborative filtering
- Carrot2 – clustering
- ELKI (LMU, Kriegel)
  - Algorithms: http://elki.dbs.ifi.lmu.de/wiki/Algorithms
- WEKA
- Orange
  - http://docs.orange.biolab.si/widgets/rst/index.html
- Stanford classifier (maximum entropy classifier)
- KNIME (Generic data processing framework)