LMU LUDWIG-MAXIMILIANS-UNIVERSITY MUNICH
DEPARTMENT INSTITUTE FOR INFORMATICS
DATABASE SYSTEMS GROUP

Skript zur Vorlesung:

Einführung in die Informatik: Systeme und Anwendungen

Sommersemester 2011

Kapitel 4: Data Mining

Vorlesung: Prof. Dr. Christian Böhm
Übungen: Annahita Oswald

http://www.dbs.ifi.lmu.de/cms/
Einführung_in_die_Informatik_Systeme_und_Anwendungen

Überblick

Motivation


Kreditkarten


Scanner-Kassen


Datenbanken


Telefongesellschaft


Astronomie

- Riesige Datenmengen werden in Datenbanken gesammelt
- Analysen können nicht mehr manuell durchgeführt werden

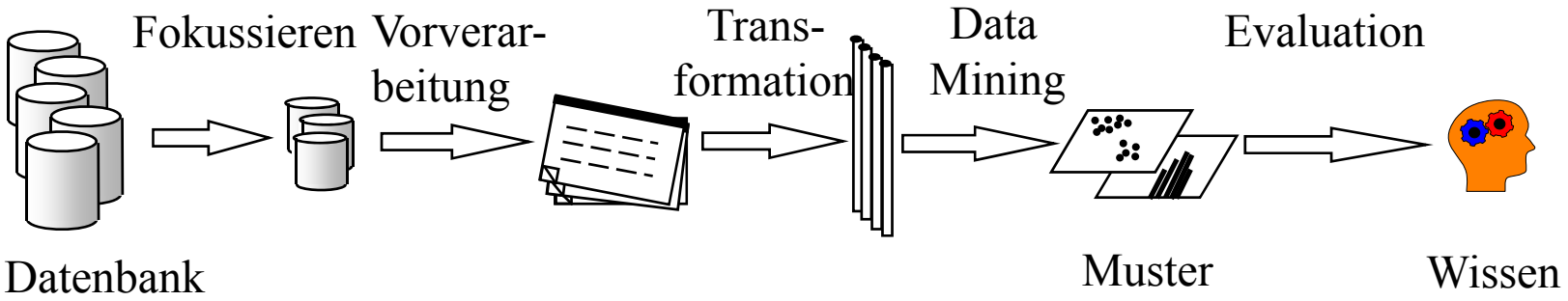
Definition KDD

- *Knowledge Discovery in Databases* (*KDD*) ist der Prozess der (semi-) automatischen Extraktion von Wissen aus Datenbanken, das
  - *gültig*
  - *bisher unbekannt*
  - und *potentiell nützlich* ist.

- Bemerkungen:
  - (*semi-*) *automatisch*: im Unterschied zu manueller Analyse. Häufig ist trotzdem Interaktion mit dem Benutzer nötig.
  - *gültig*: im statistischen Sinn.
  - *bisher unbekannt*: bisher nicht explizit, kein „Allgemeinwissen“.
  - *potentiell nützlich*: für eine gegebene Anwendung.

Der KDD-Prozess (Modell)

Prozessmodell nach Fayyad, Piatetsky-Shapiro & Smyth



Fokussieren:
- Beschaffung der Daten
- Verwaltung (File/DB)
- Selektion relevanter Daten

Vorverarbeitung:
- Integration von Daten aus unterschiedlichen Quellen
- Vervollständigung
- Konsistenzprüfung

Transformation
- Diskretisierung numerischer Merkmale
- Ableitung neuer Merkmale
- Selektion relevanter Merkm.

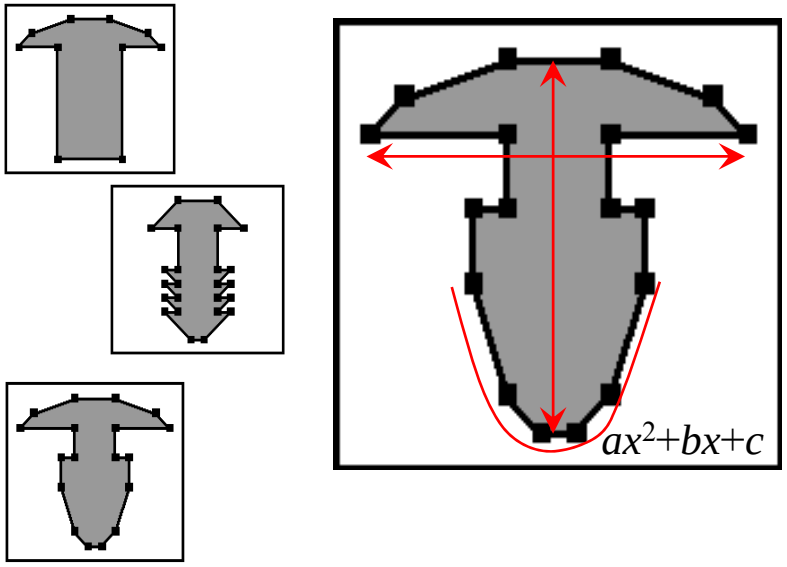
Data Mining
- Generierung der Muster bzw. Modelle

Evaluation
- Bewertung der Interessantheit durch den Benutzer
- Validierung: Statistische Prüfung der Modelle

Objekt-Merkmale (Feature)

- Oft sind die betrachteten Objekte komplex
- Eine Aufgabe des KDD-Experten ist dann, geeignete Merkmale (*Features*) zu definieren bzw. auszuwählen, die für die Unterscheidung (Klassifikation, Ähnlichkeit) der Objekte relevant sind.

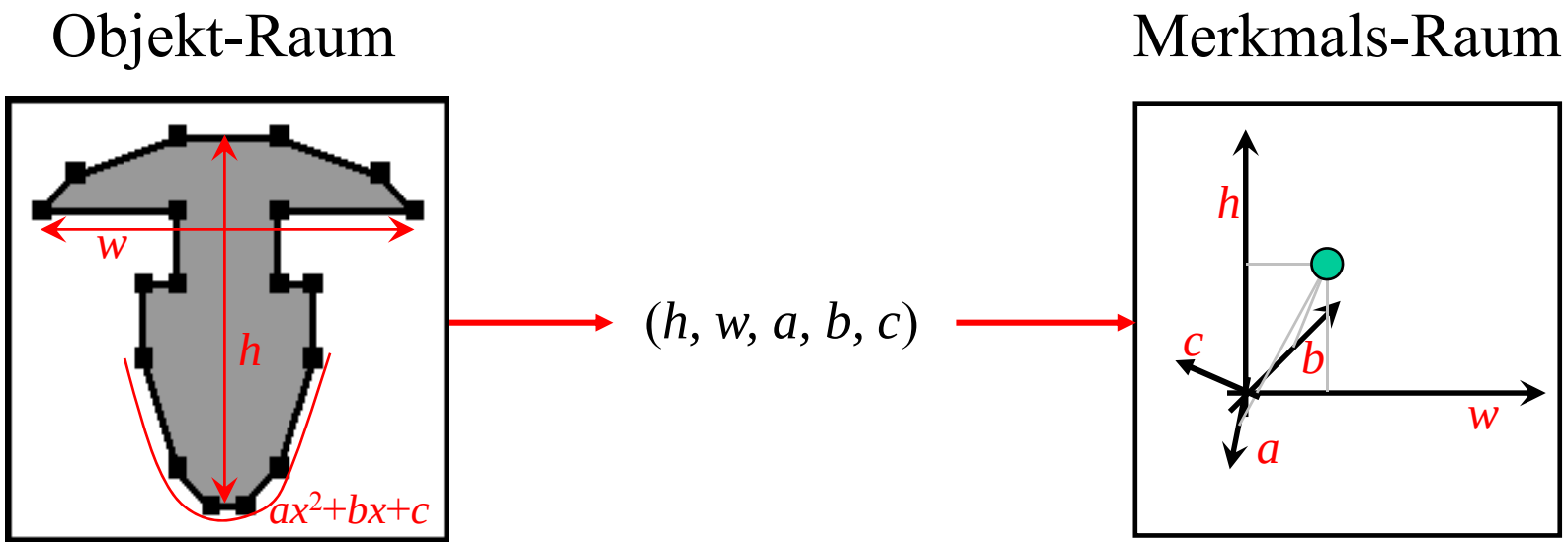
Beispiel: CAD-Zeichnungen:



Mögliche Merkmale:

- Höhe h
- Breite w
- Kurvatur-Parameter (a,b,c)

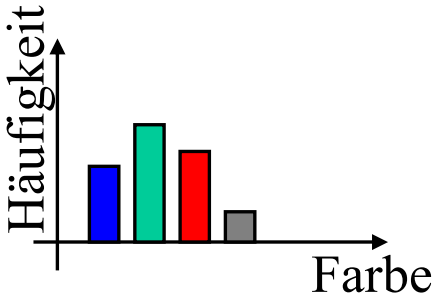
Feature-Vektoren



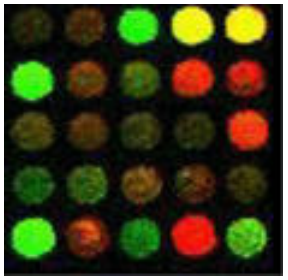
- Im Kontext von statistischen Betrachtungen werden die Merkmale häufig auch als *Variablen* bezeichnet
- Die ausgewählten Merkmale werden zu Merkmals-Vektoren (*Feature Vector*) zusammengefasst
- Der Merkmalsraum ist häufig hochdimensional (im Beispiel 5-dim.)

Feature-Vektoren (weitere Beispiele)

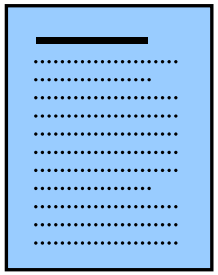
Bilddatenbanken:
Farbhistogramme



Gen-Datenbanken:
Expressionslevel



Text-Datenbanken:
Begriffs-Häufigkeiten



Data	25
Mining	15
Feature	12
Object	7
...	

Der Feature-Ansatz ermöglicht einheitliche Behandlung von Objekten verschiedenster Anwendungsklassen

Feature: verschiedene Kategorien

Nominal (kategorisch)

Charakteristik:

Nur feststellbar, ob der Wert gleich oder verschieden ist. Keine Richtung (besser, schlechter) und kein Abstand.
Merkmale mit nur zwei Werten nennt man *dichotom*.

Beispiele:

Geschlecht (dichotom)
Augenfarbe
Gesund/krank (dichotom)

Ordinal

Charakteristik:

Es existiert eine Ordnungsrelation (besser/schlechter) zwischen den Kategorien, aber kein einheitlicher Abstand.

Beispiele:

Schulnote (metrisch?)
Güteklasse
Altersklasse

Metrisch

Charakteristik:

Sowohl Differenzen als auch Verhältnisse zwischen den Werten sind aussagekräftig. Die Werte können diskret oder stetig sein.

Beispiele:

Gewicht (stetig)
Verkaufszahl (diskret)
Alter (stetig oder diskret)

Ähnlichkeit von Objekten

- Spezifiziere Anfrage-Objekt $q \in DB$ und…
  - … suche ähnliche Objekte – Range-Query (Radius ε)

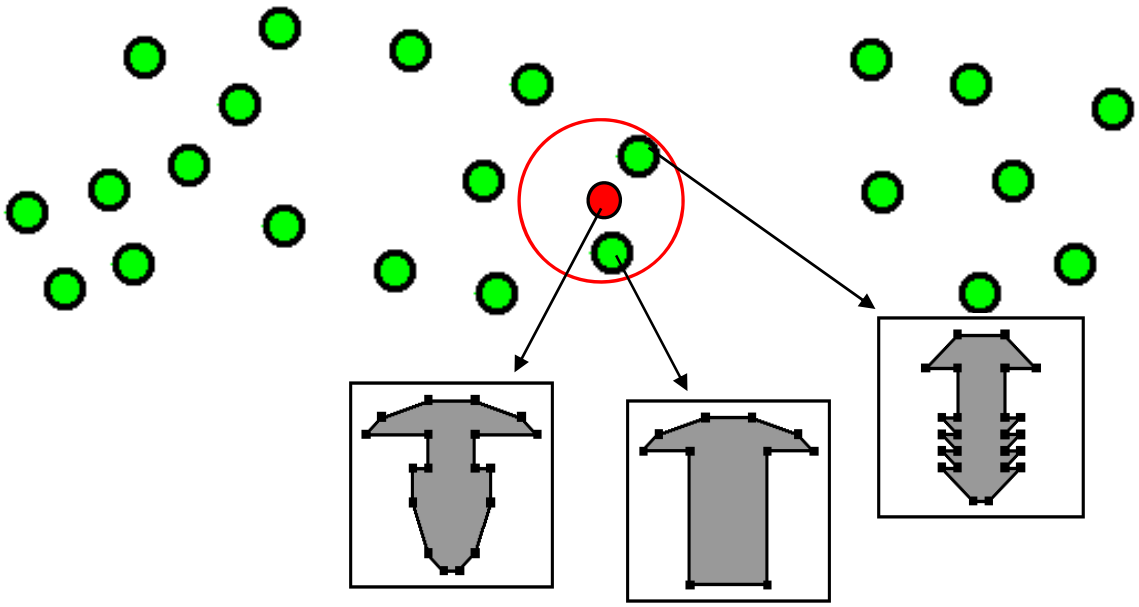
$$RQ(q,\varepsilon) = \{ o \in DB \mid \delta(q,o) \leq \varepsilon \}$$

  - … suche die k ähnlichsten Objekte – Nearest Neighbor Query

$$NN(q,k) \subseteq DB \text{ mit } k \text{ Objekten, sodass}$$

$$\forall o \in NN(q,k), p \in DB \backslash NN(q,k) : \delta(q,o) \leq \delta(q,p)$$

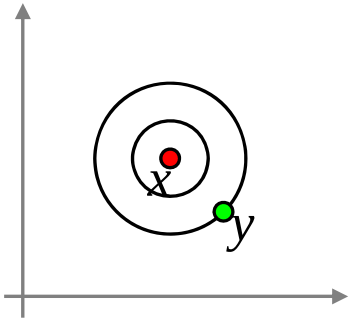
alternative Schreibweise für Mengendifferenz:
$A \backslash B = A - B$



Ähnlichkeitsmaße im Feature-Raum

Euklidische Norm (L_2):

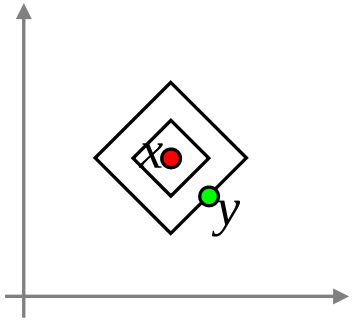
$\delta_1(x,y) = ((x_1-y_1)^2+(x_2-y_2)^2+...)^{1/2}$



Abstand in Euklidischen Raum (natürliche Distanz)

Manhattan-Norm (L_1):

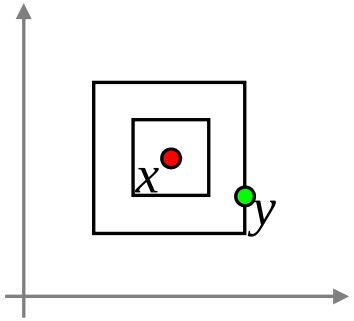
$\delta_2(x,y) = |x_1-y_1|+|x_2-y_2|+...$



Die Unähnlichkeiten der einzelnen Merkmale werden direkt addiert

Maximums-Norm (L_∞):

$\delta_\infty(x,y) = \max\{|x_1-y_1|, |x_2-y_2|,...\}$



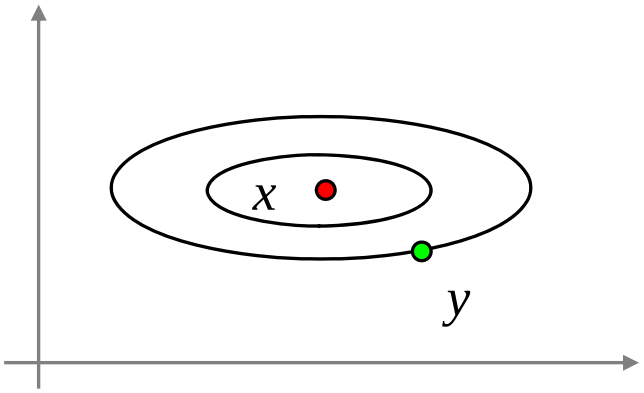
Die Unähnlichkeit des am wenigsten ähnlichen Merkmals zählt

Verallgemeinerung L_p-Abstandsmaß:

$$\delta_p(x,y) = (|x_1-y_1|^p + |x_2-y_2|^p + ...)^{1/p}$$

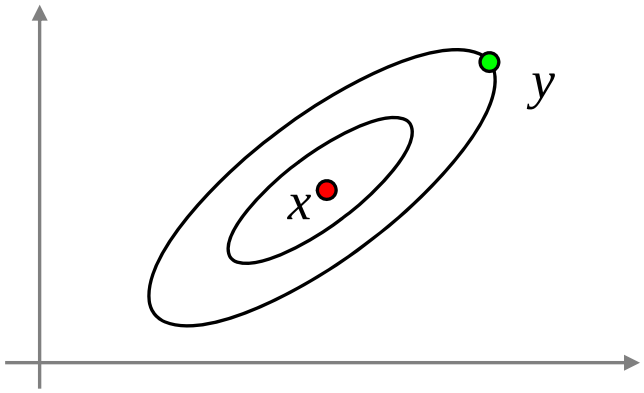
Gewichtete Ähnlichkeitsmaße

- Viele Varianten gewichten verschiedene Merkmale unterschiedlich stark.



Gewichtete Euklidische Distanz

$$\delta_{p,w}(x,y) = \sqrt[p]{\sum_{i=1}^{d} w_i \cdot |x_i - y_i|^p}$$



Mahalanobis Distanz

$$\delta_{\Sigma}(x,y) = \sqrt{(x-y)^T \cdot \Sigma^{-1} \cdot (x-y)}$$

Σ = Kovarianz-Matrix

Kategorien von Data Mining

- Wichtigste Data-Mining-Verfahren auf Merkmals-Vektoren:
  - Clustering
  - Outlier Detection

  } normalerweise unsupervised

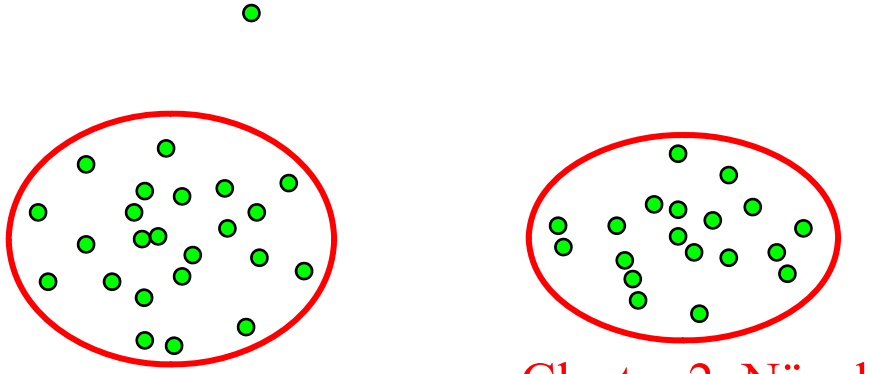
  - Klassifikation
  - Regression

  } normalerweise supervised

- Supervised: In Trainingsphase wird eine Funktion gelernt, die in der Testphase angewandt wird.
- Unsupervised: Es gibt keine Trainingsphase. Die Methode findet Muster, die einem bestimmten Modell entsprechen.

- Darüber hinaus gibt es zahlreiche Verfahren, die nicht auf Merkmalsvektoren, sondern direkt auf Texten, Mengen, Graphen usw. arbeiten.

Clustering



Ein Grundmodell des Clustering ist:

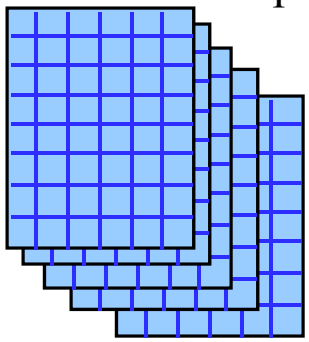
Zerlegung (Partitionierung) einer Menge von Objekten (bzw. Feature-Vektoren) in Teilmengen (Cluster), so dass

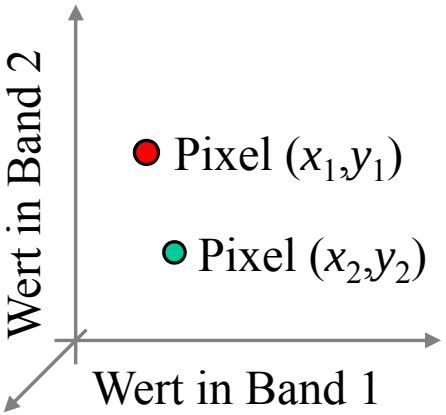
- die Ähnlichkeit der Objekte innerhalb eines Clusters maximiert
- die Ähnlichkeit der Objekte verschiedener Cluster minimiert wird

Idee: Die verschiedenen Cluster repräsentieren meist unterschiedliche Klassen von Objekten; bei evtl. unbek. Anzahl und Bedeutung der Klassen

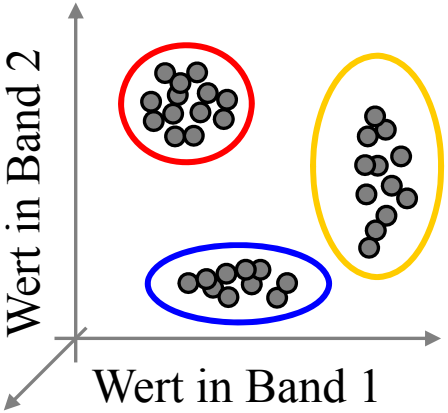
Anwendung: Thematische Karten

Aufnahme der Erdoberfläche
in 5 verschiedenen Spektren



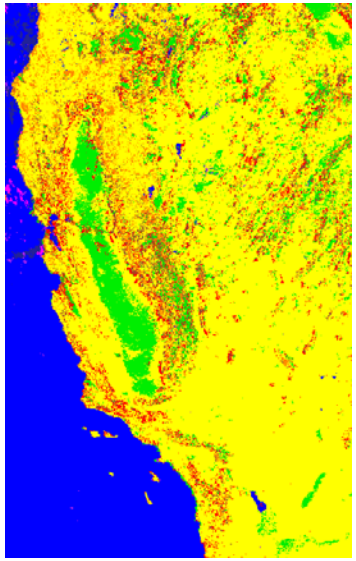


Cluster-Analyse

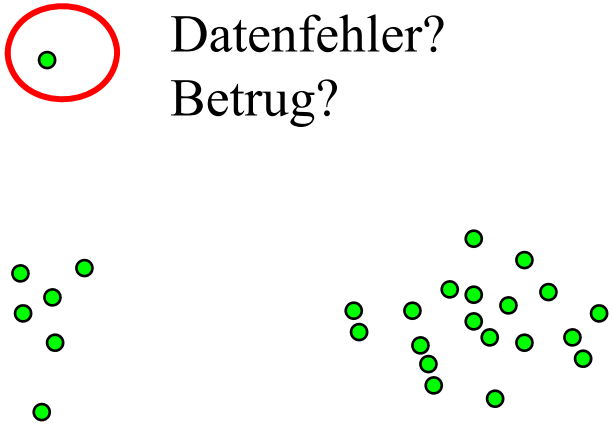


Rücktransformation
in *xy*-Koordinaten

Farbcodierung nach
Cluster-Zugehörigkeit



Outlier Detection



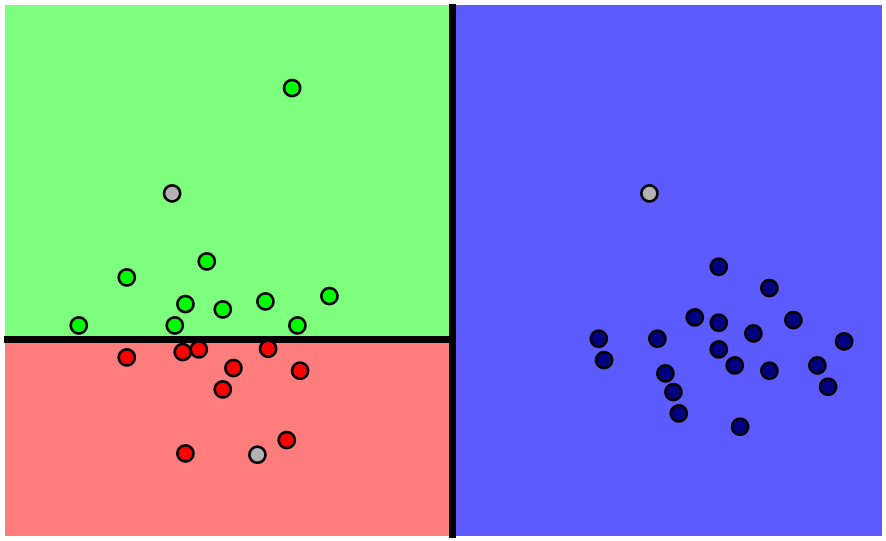
Outlier Detection bedeutet:

Ermittlung von untypischen Daten

Anwendungen:

- Entdeckung von Missbrauch etwa bei
  - Kreditkarten
  - Telekommunikation
- Datenbereinigung (Messfehler)

Klassifikation



- Schrauben
- Nägel
- Klammern

} Trainings-daten

- Neue Objekte

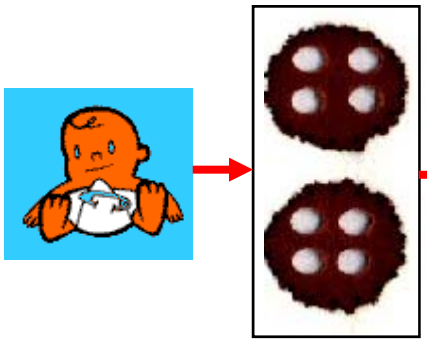
Aufgabe:

Lerne aus den bereits klassifizierten *Trainingsdaten* die *Regeln*, um neue Objekte nur aufgrund der Merkmale zu klassifizieren

Das Ergebnismerkmal (Klassenvariable) ist nominal (*kategorisch*)

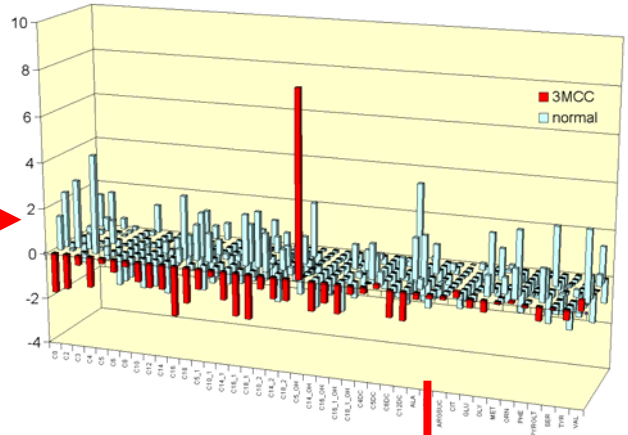
Anwendung: Neugeborenen-Screening

Blutprobe des Neugeborenen



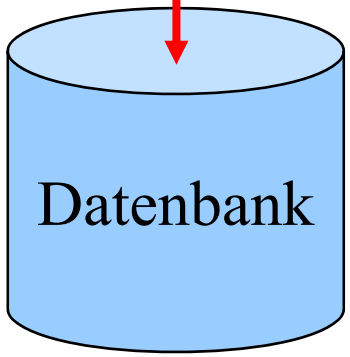
Massenspektrometrie

Metabolitenspektrum

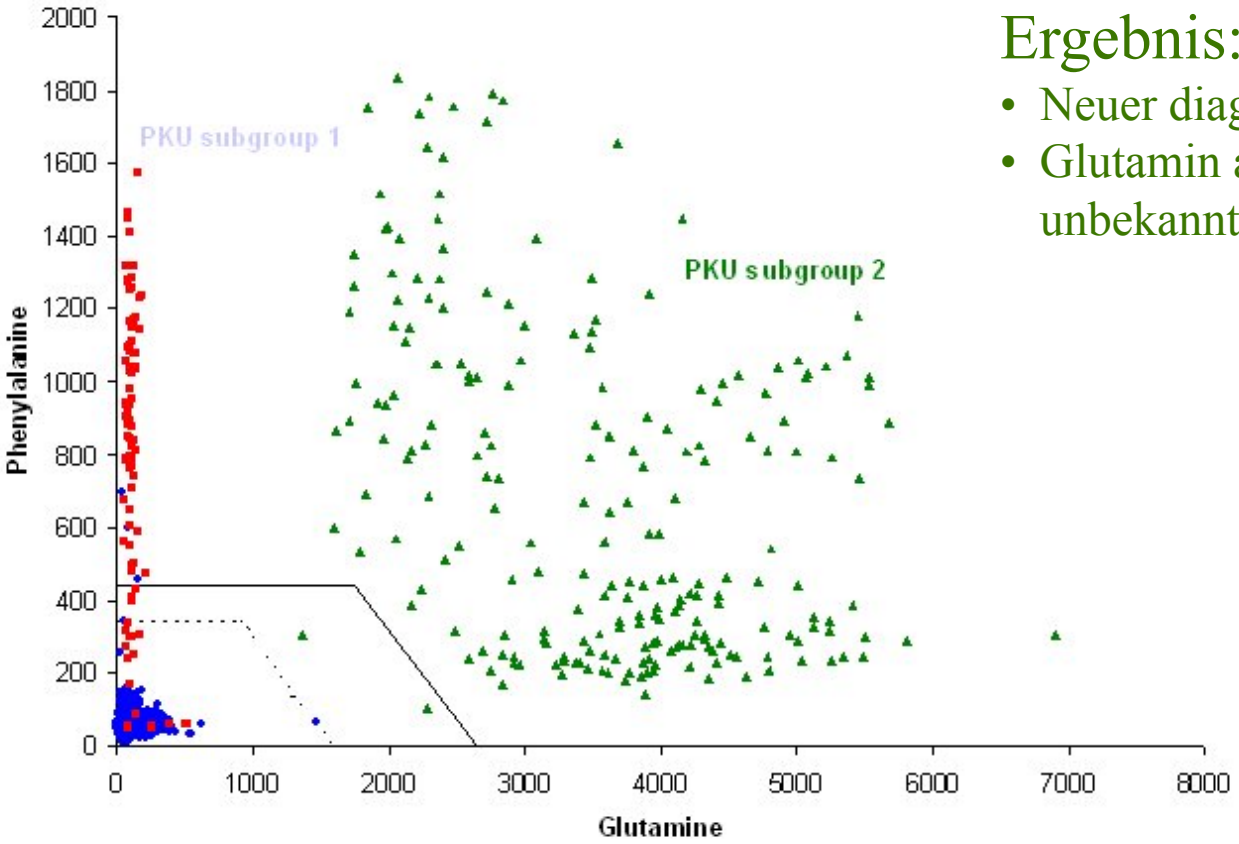


14 analysierte Aminosäuren:

alanine
arginine
argininosuccinate
citrulline
glutamate
glycine
methionine
phenylalanine
pyroglutamate
serine
tyrosine
valine
leuzine+isoleuzine
ornitine



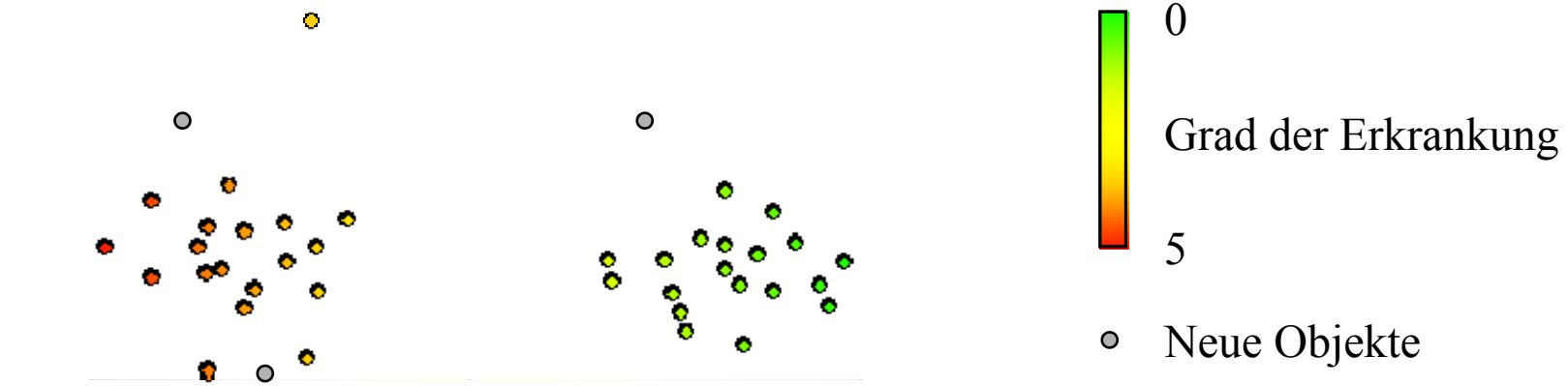
Anwendung: Neugeborenen-Screening



Ergebnis:

- Neuer diagnostischer Test
- Glutamin als bisher unbekannter Marker

Regression



Aufgabe:

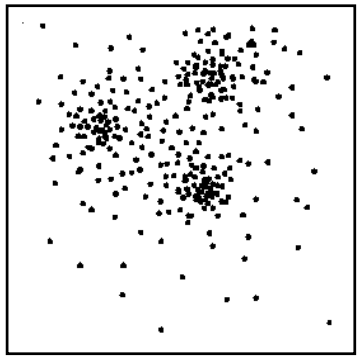
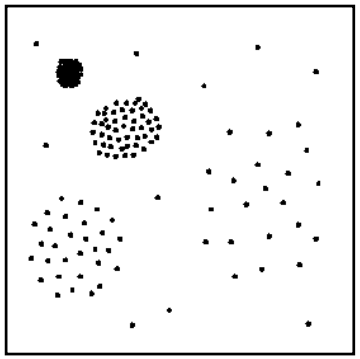
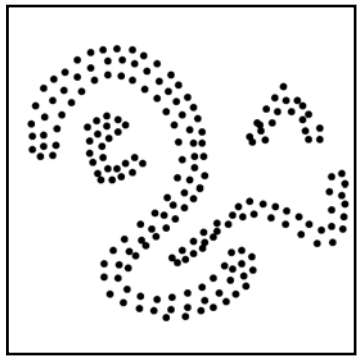
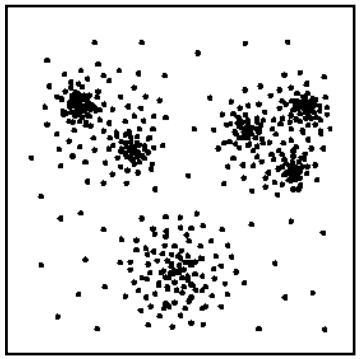
Ähnlich zur Klassifikation, aber das Ergebnis-Merkmal, das gelernt bzw. geschätzt werden soll, ist *metrisch*.

Überblick

Ziel des Clustering

Ziel: Identifikation einer endlichen Menge von Kategorien, Klassen oder Gruppen (*Cluster*) in den Daten

- Objekte im *gleichen* Cluster sollen möglichst ähnlich sein
- Objekte aus *verschiedenen* Clustern sollen möglichst unähnlich zueinander sein

Herausforderungen:

- Cluster unterschiedlicher Größe, Form und Dichte
- hierarchische Cluster
- Rauschen (Noise)

=> unterschiedliche Clustering-Algorithmen

Typen von Clustering-Verfahren

- Partitionierende Verfahren
  - Parameter: Anzahl *k* der Cluster, Distanzfunktion
  - sucht ein „flaches“ Clustering in *k* Cluster mit minimalen Kosten
- Dichtebasierte Verfahren
  - Parameter: minimale Dichte in einem Cluster, Distanzfunktion
  - erweitert Punkte um ihre Nachbarn solange Dichte groß genug

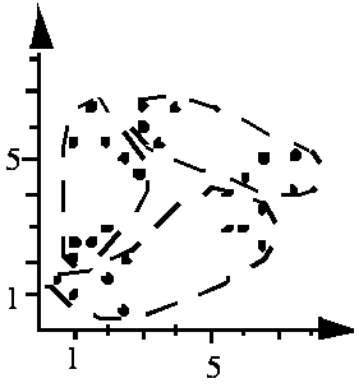
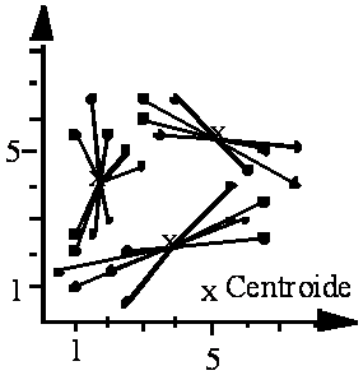
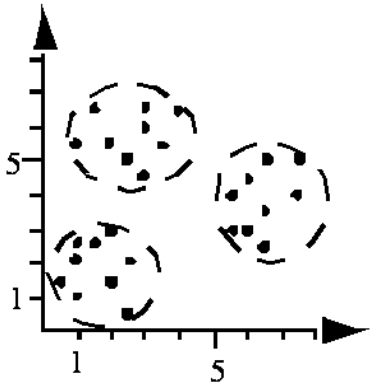
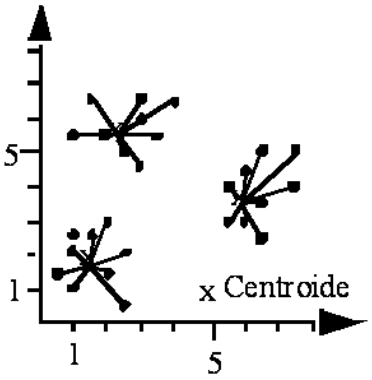
Partitionierende Verfahren

Grundlagen

- Ziel
  - Partitionierung in *k* Cluster so dass eine Kostenfunktion minimiert wird (Gütekriterium)
- Lokal optimierendes Verfahren
  - wähle *k* initiale Cluster-Repräsentanten
  - optimiere diese Repräsentanten iterativ
  - ordne jedes Objekt seinem ähnlichsten Repräsentanten zu
- Typen von Cluster-Repräsentanten
  - Mittelwert des Clusters (*Centroid*)
  - Element des Clusters (*Medoid*)
  - Wahrscheinlichkeitsverteilung des Clusters (*Erwartungsmaximierung*)

Partitionierende Verfahren

Beispiel

	Cluster	Cluster-Repräsentanten
schlechtes Clustering		
optimales Clustering		

Partitionierende Verfahren: k-means

Grundbegriffe

- Objekte sind Punkte $p=(p_1, ..., p_d)$ in einem euklidischen Vektorraum
- euklidische Distanz
- *Centroid* μ_C: Mittelwert aller Punkte im Cluster *C*
- *Maß für die Kosten* (Kompaktheit) *eines Clusters C*

$$TD^2(C) = \sum_{p \in C} dist(p, \mu_C)^2$$

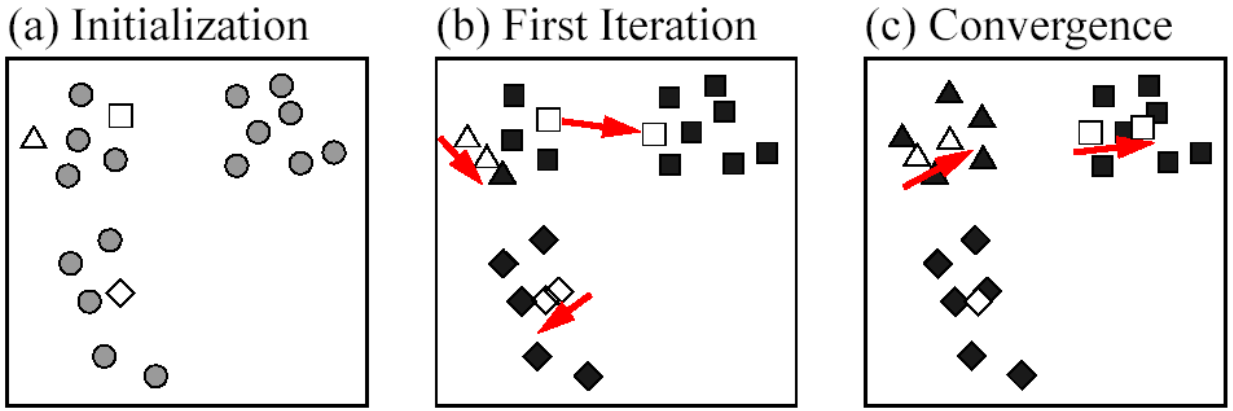
- *Maß für die Kosten* (Kompaktheit) *eines Clustering*

$$TD^2 = \sum_{i=1}^{k} TD^2(C_i)$$

Partitionierende Verfahren: k-means

Idee des Algorithmus

- Algorithmus startet z.B. mit zufällig gewählten Punkten als Cluster-Repräsentanten
- Der Algorithmus besteht aus zwei alternierenden Schritten:
  - Zuordnung jedes Datenpunktes zum räumlich nächsten Repräsentanten
  - Neuberechnung der Repräsentanten (Centroid der zugeordneten Punkte)
- Diese Schritte werden so lange wiederholt, bis sich keine Änderung mehr ergibt



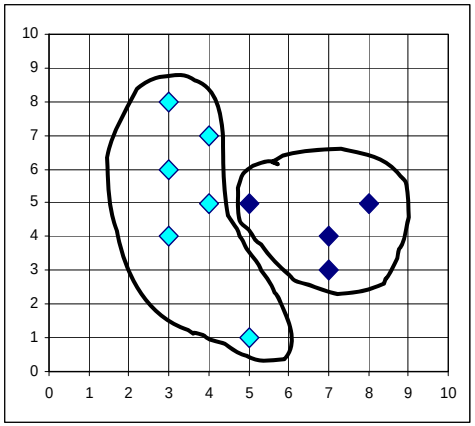
Partitionierende Verfahren: k-means

Algorithmus

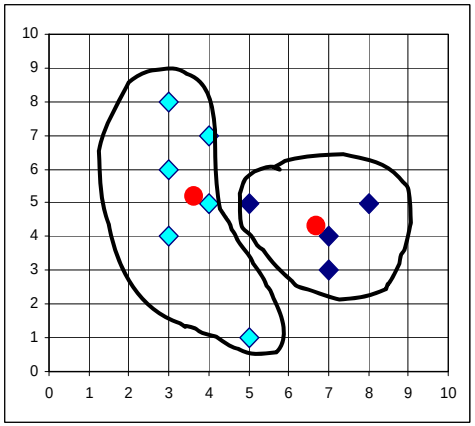
```
ClusteringDurchVarianzMinimierung(Punktmenge D, Integer k)
   Erzeuge eine „initiale“ Zerlegung der Punktmenge D in k Klassen;
   Berechne die Menge C'={C_1, ..., C_k} der Zentroide für die k Klassen;
   C = {};
   repeat
      C = C';
      Bilde k Klassen durch Zuordnung jedes Punktes zum
        nächstliegenden Zentroid aus C;
      Berechne die Menge C'={C'_1, ..., C'_k} der Zentroide für die neu
        bestimmten Klassen;
   until C = C';
   return C;
```

Partitionierende Verfahren: k-means

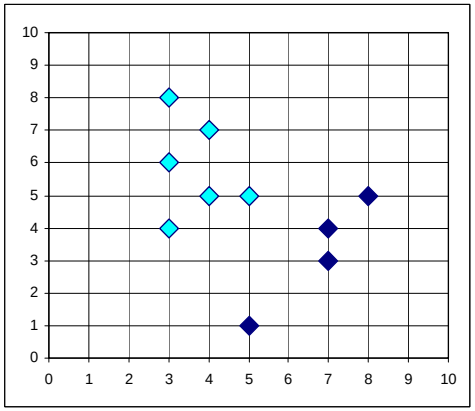
Beispiel



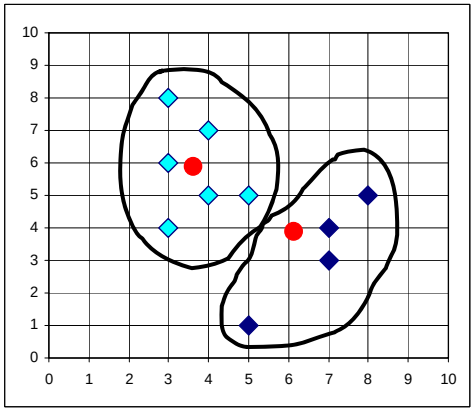
Berechnung der neuen Zentroide



Zuordnung zum nächsten Zentroid



Berechnung der neuen Zentroide



Partitionierende Verfahren: k-means

Diskussion

+ Effizienz:
  - Anzahl der Iterationen ist im allgemeinen klein (~ 5 - 10).
+ einfache Implementierung:
  - *k*-means ist das populärste partitionierende Clustering-Verfahren

– Anfälligkeit gegenüber Rauschen und Ausreißern
(alle Objekte gehen ein in die Berechnung des Zentroids)
– Cluster müssen konvexe Form haben
– die Anzahl *k* der Cluster muss bekannt sein
– starke Abhängigkeit von der initialen Zerlegung
(sowohl Ergebnis als auch Laufzeit)

Dichtebasiertes Clustering

LMU

Grundlagen

- Idee
  - Cluster als Gebiete im *d*-dimensionalen Raum, in denen die Objekte dicht beieinander liegen
  - getrennt durch Gebiete, in denen die Objekte weniger dicht liegen
- Anforderungen an dichtebasierte Cluster
  - für jedes Objekt eines Clusters überschreitet die lokale Punktdichte einen gegebenen Grenzwert
  - die Menge von Objekten, die den Cluster ausmacht, ist räumlich zusammenhängend

Dichtebasiertes Clustering

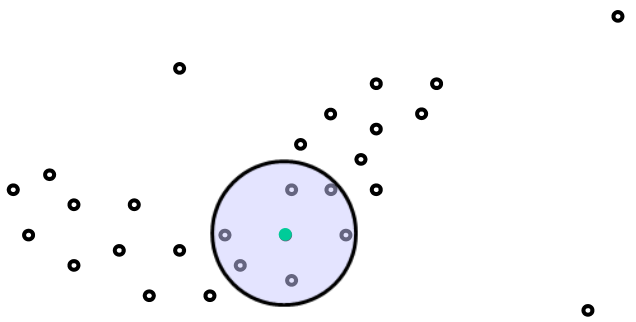
Grundlagen

Parameter:

- $\varepsilon \in \mathbf{R}$
- $minPts \in \mathbf{N}$
- Distanzfunktion dist

ε $minPts = 4$

- ε-*Umgebung eines Punktes p*: $N_\varepsilon(p) = \{q \mid \text{dist}(p,q) \leq \varepsilon\}$



Dichtebasiertes Clustering

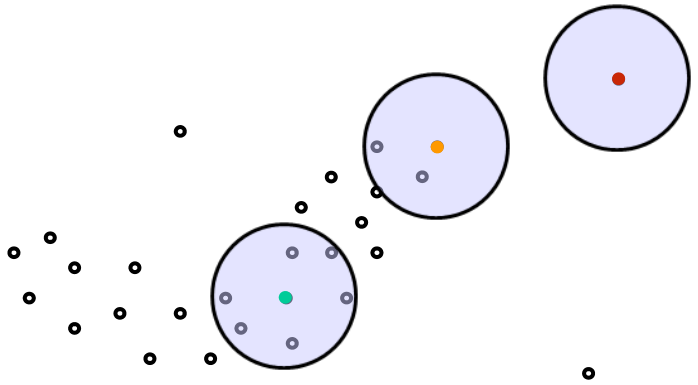
Grundlagen

Parameter:

- $\varepsilon \in \mathbf{R}$
- $minPts \in \mathbf{N}$
- Distanzfunktion

ε $\quad$ $minPts = 4$

- ε-*Umgebung eines Punktes p*: $N_\varepsilon(p) = \{q \mid \text{dist}(p,q) \leq \varepsilon\}$
- *Kernpunkt p*: $|N_\varepsilon(p)| \geq minPts$
- *Randpunkt / Rauschen ???*



Dichtebasiertes Clustering

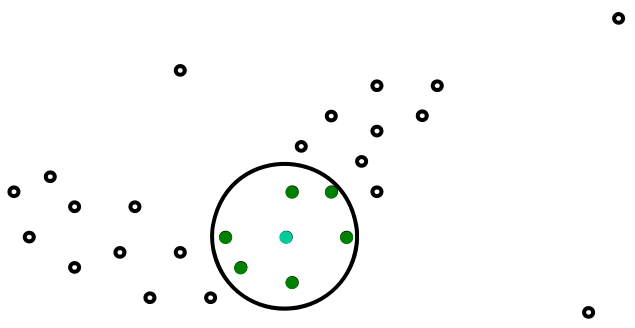
Grundlagen

Parameter:

- $\varepsilon \in \mathbf{R}$
- $minPts \in \mathbf{N}$
- Distanzfunktion

ε $\quad$ $minPts = 4$

- ε-*Umgebung eines Punktes p*: $N_\varepsilon(p) = \{q \mid \text{dist}(p,q) \leq \varepsilon\}$
- *Kernpunkt p*: $|N_\varepsilon(p)| \geq minPts$
- *Randpunkt / Rauschen ???*
- *Dichte-Verbundenheit: sammle rekursiv alle Punkte, die in der ε-Umgebung eines Kernpunktes liegen auf*



Dichtebasiertes Clustering

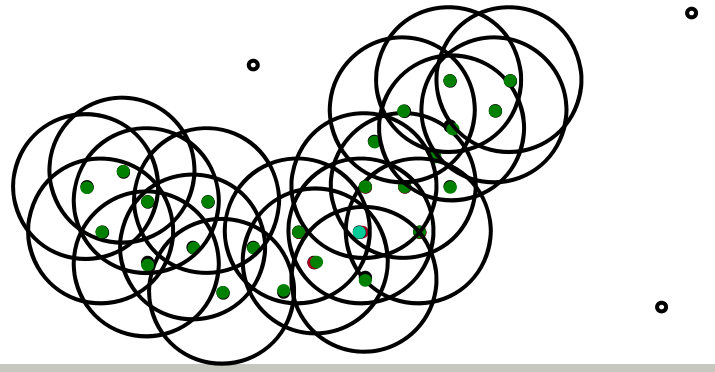
Grundlagen

Parameter:

- $\varepsilon \in \mathbf{R}$
- $minPts \in \mathbf{N}$
- Distanzfunktion

ε $\quad$ $minPts = 4$

- *ε-Umgebung eines Punktes p:* $N_\varepsilon(p) = \{q \mid \text{dist}(p,q) \leq \varepsilon\}$
- *Kernpunkt p:* $|N_\varepsilon(p)| \geq minPts$
- *Randpunkt / Rauschen ???*
- *Dichte-Verbundenheit: sammle rekursiv alle Punkte, die in der ε-Umgebung eines Kernpunktes liegen auf*



Dichtebasiertes Clustering

Algorithmus DBSCAN

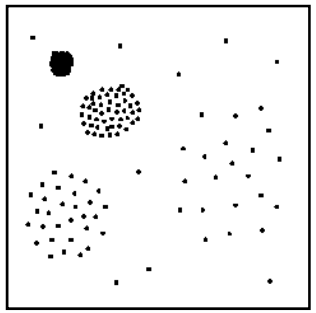
- Entdecken eines Clusters:
  - Finde einen beliebigen Kernpunkt des Clusters
  - Expandiere das Cluster, d.h. finde alle dichteverbundenen Punkte, indem *rekursiv* alle Punkte, die in der ε-Umgebung eines Kernpunkts liegen, aufgesammelt werden
- Algorithmus:
  - Prüfe jeden noch nicht markierten Punkt ob es ein Kernpunkt ist
  - Wenn ja, expandiere einen neuen Cluster (alle Punkte des Clusters werden mit der entspr. Cluster-ID markiert)
  - Wenn nein, markiere Punkt als Rauschen (Randpunkte werden bei der Expansion des entspr. Clusters später richtig markiert)

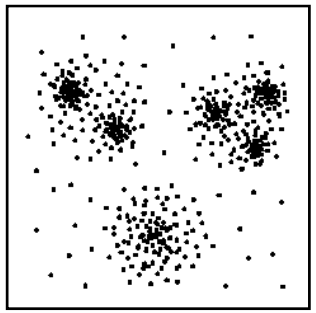
Dichtebasiertes Clustering

Diskussion

+ Effizienz:
  - ε-Umgebungen i.d.R. effizient auswertbar
+ einfache Implementierung:
+ keine Anfälligkeit gegenüber Rauschen und Ausreißern
+ Cluster unterschiedlicher Form erkennbar
+ Anzahl der Cluster wird automatisch bestimmt

– Parameter (Dichtegrenzwert) nicht immer einfach zu bestimmen (Cluster unterschiedlicher Dichte)





Überblick

Klassifikationsproblem

Gegeben:

- eine Menge $O \subseteq D$ von Objekten $o = (o_1, \ldots, o_d) \in O$ mit *Attributen* A_i, $1 \leq i \leq d$
- eine Menge von Klassen $C = \{c_1, \ldots, c_k\}$
- Klassenzuordnung $T : O \rightarrow C$

Gesucht:

- die Klassenzugehörigkeit für Objekte aus $D \setminus O$
- ein *Klassifikator* $K : D \rightarrow C$

Abgrenzung zum Clustering

- Klassifikation: Klassen a priori bekannt
- Clustering: Klassen werden erst gesucht

Verwandtes Problem: *Regression*

- gesucht ist der Wert für ein numerisches Attribut

Beispiel

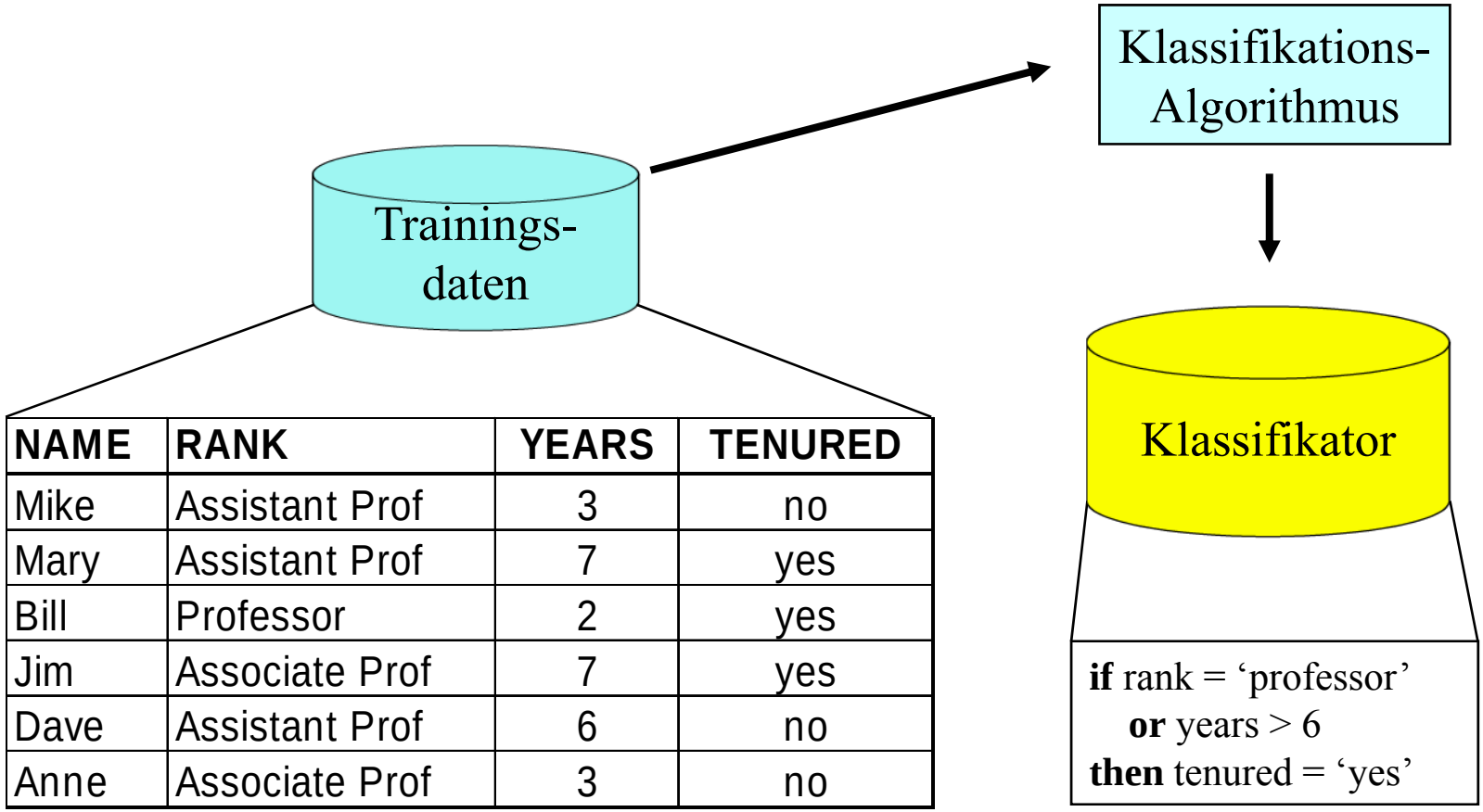
ID	Alter	Autotyp	Risiko
1	23	Familie	hoch
2	17	Sport	hoch
3	43	Sport	hoch
4	68	Familie	niedrig
5	32	LKW	niedrig

Einfacher Klassifikator

```
if Alter > 50 then Risikoklasse = Niedrig;
if Alter ≤ 50 and Autotyp=LKW then
Risikoklasse=Niedrig;
if Alter ≤ 50 and Autotyp ≠ LKW
                then Risikoklasse = Hoch.
```

Klassifikations-Prozess

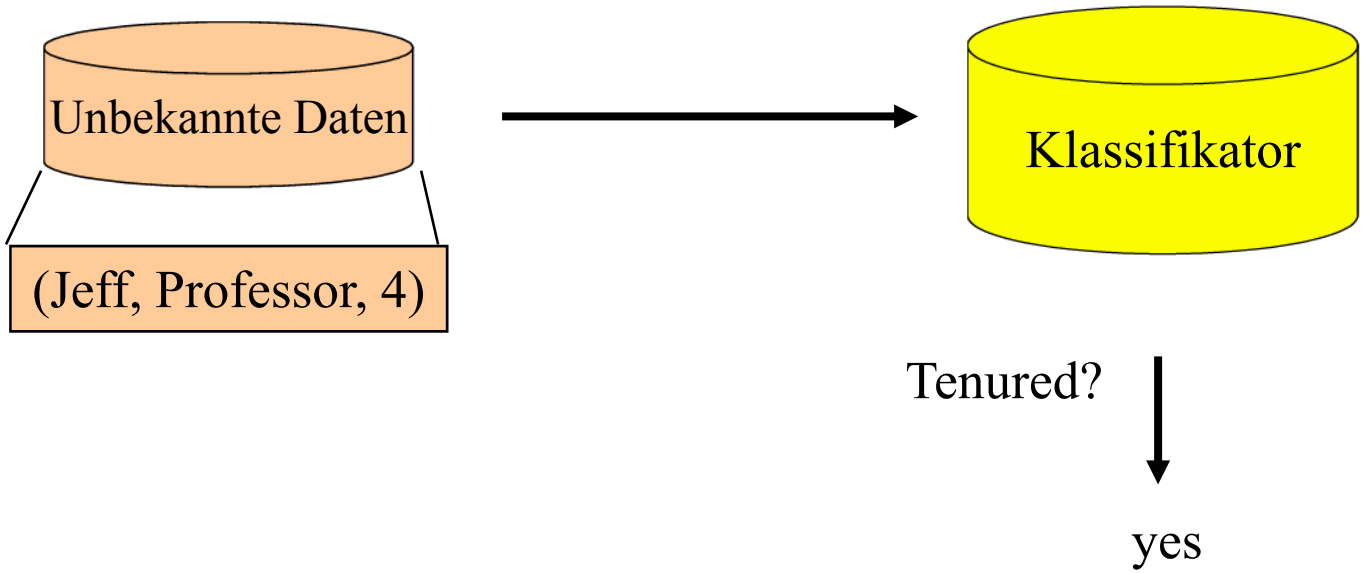
Konstruktion des Modells



NAME	RANK	YEARS	TENURED
Mike	Assistant Prof	3	no
Mary	Assistant Prof	7	yes
Bill	Professor	2	yes
Jim	Associate Prof	7	yes
Dave	Assistant Prof	6	no
Anne	Associate Prof	3	no

Klassifikations-Prozess

Anwendung des Modells



manchmal: keine Klassifikation unbekannter Daten
sondern „nur“ besseres Verständnis der Daten

Bewertung von Klassifikatoren

Grundbegriffe

- Sei K ein Klassifikator und sei $TR \subseteq O$ die Trainingsmenge. $O \subseteq D$ ist die Menge der Objekte, bei denen die Klassenzugehörigkeit bereits bekannt ist .

- **Problem der Bewertung:**
  - gewünscht ist gute Performanz auf ganz D.
  - Klassifikator ist für TR optimiert.
  - Test auf TR erzeugt in der Regel viel bessere Ergebnisse, als auf $D \setminus TR$.

    Daher kein realistisches Bild der Performanz auf D.

    $\Rightarrow$ *Overfitting*

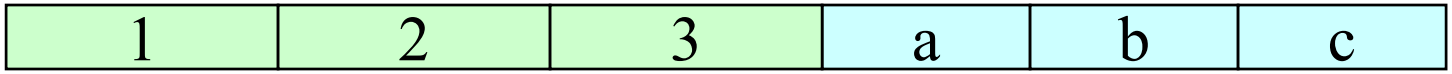
Bewertung von Klassifikatoren

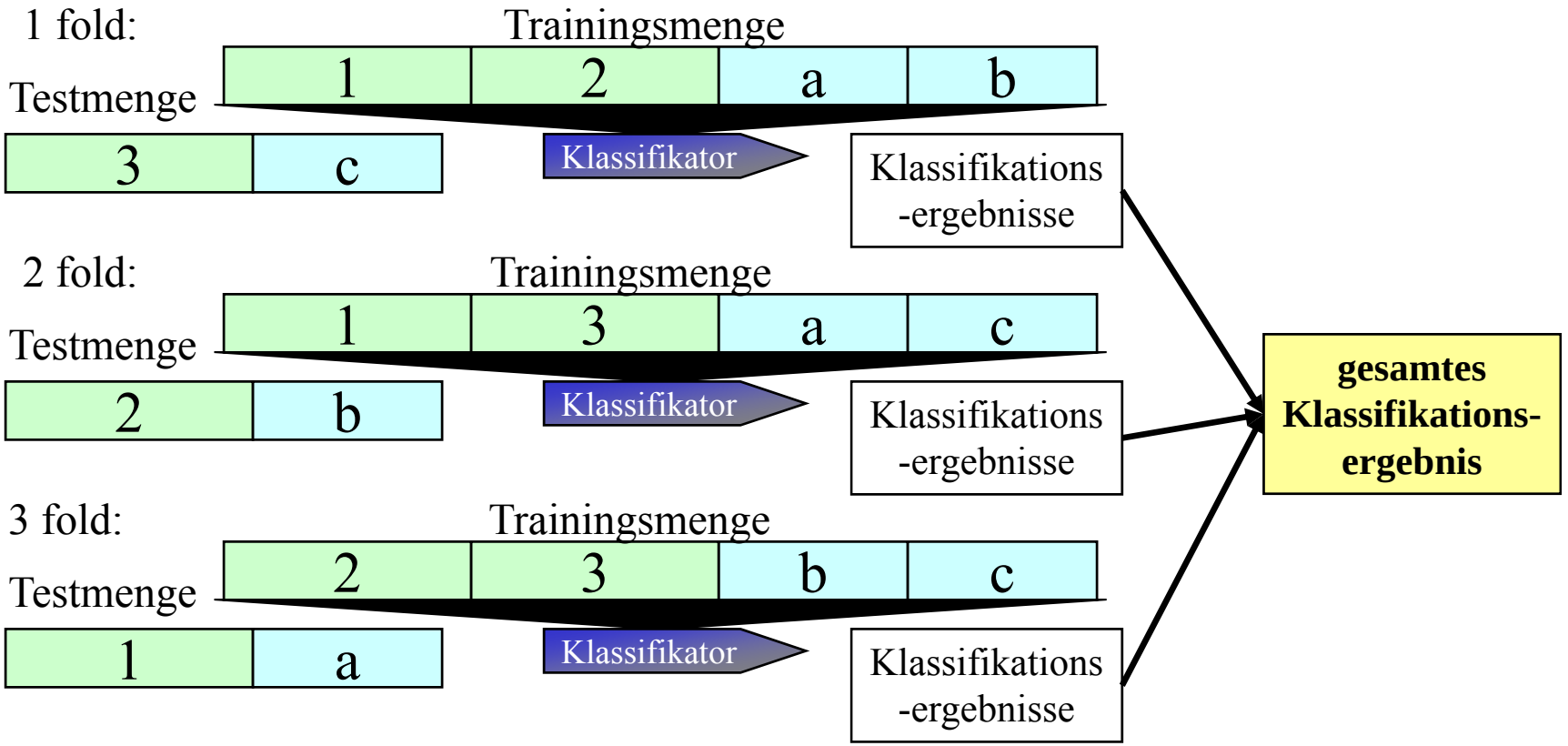
- Abschätzung der Vorhersagequalität auf unbekannten Daten: k-fache Kreuzvalidierung (k-fold cross-validation)
  - Teile Trainingsmenge $TR \subseteq O$ in k Partitionen $TR_1,\ldots,TR_k$ ein.
  - für $i = 1\ldots k$:
    - trainiere einen Klassifikator K_i auf $TR \setminus TR_i$
    - teste K_i auf TR_i
  - Mittle die k beobachteten Fehlerraten

Bewertung von Klassifikatoren

Ablauf 3-fache Überkreuzvalidierung (3-fold Cross Validation)

Sei n = 3 : Menge aller Daten mit Klasseninformation die zur Verfügung stehen





Bewertung von Klassifikatoren

Ergebnis des Tests : Konfusionsmatrix (confusion matrix)

klassifiziert als ...

tatsächliche Klasse ...	Klasse1	Klasse 2	Klasse 3	Klasse 4	other
Klasse 1	35	1	1	1	4
Klasse 2	0	31	1	1	5
Klasse 3	3	1	50	1	2
Klasse 4	1	0	1	10	2
other	3	1	9	15	13

korrekt klassifizierte Objekte

Aus der Konfusionsmatrix lassen sich diverse Kennzahlen berechnen, z.B. Accuracy, Classification Error, Precision und Recall.

Bewertung von Klassifikatoren

• *Gütemaße für Klassifikatoren*

• Sei K ein Klassifikator, $TR \subseteq O$ die Trainingsmenge, $TE \subseteq O$ die Testmenge. Bezeichne $T(o)$ die tatsächliche Klasse eines Objekts o.

- *Klassifikationsgenauigkeit* (classification accuracy) von K auf TE:

$$G_{TE}(K) = \frac{|\{o \in TE : K(o) = T(o)\}|}{|TE|}$$

- *Tatsächlicher Klassifikationsfehler* (true classification error)

$$F_{TE}(K) = \frac{|\{o \in TE : K(o) \neq T(o)\}|}{|TE|}$$

- *Beobachteter Klassifikationsfehler* (apparent classification error)

$$F_{TR}(K) = \frac{|\{o \in TR : K(o) \neq T(o)\}|}{|TR|}$$

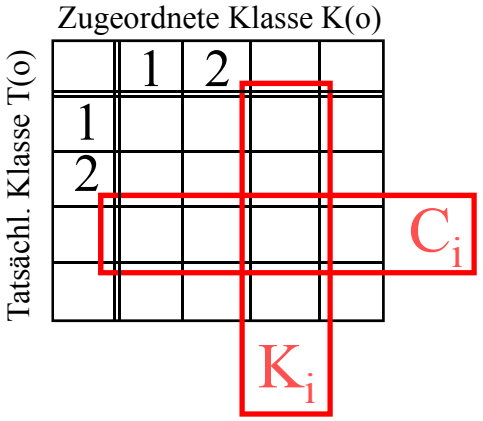
Bewertung von Klassifikatoren

Recall:

Anteil der Testobjekte einer Klasse *i*, die richtig erkannt wurden.

Sei $C_i = \{o \in TE : T(o) = i\}$, dann ist

$$\text{Recall}_{TE}(K,i) = \frac{|\{o \in C_i : K(o) = T(o)\}|}{|C_i|}$$



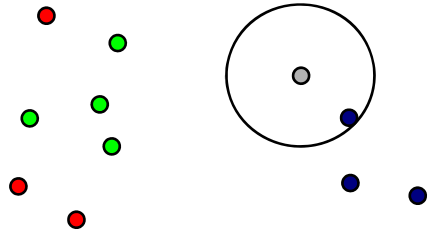
Precision:

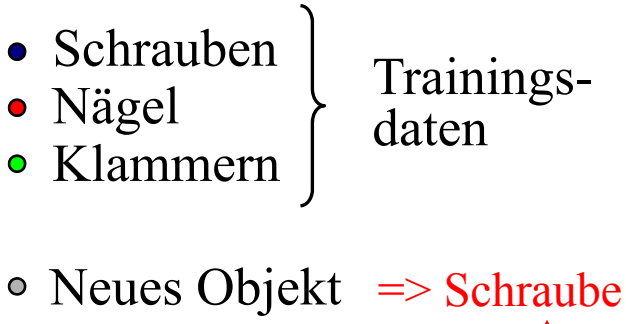
Anteil der zu einer Klasse i zugeordneten Testobjekte, die richtig erkannt wurden.

Sei $K_i = \{o \in TE : K(o) = i\}$, dann ist

$$\text{Precision}_{TE}(K,i) = \frac{|\{o \in K_i : K(o) = T(o)\}|}{|K_i|}$$

Nächste-Nachbarn-Klassifikatoren





- Instanzbasiertes Lernen (*instance based learning*)
- Einfachster Nächste-Nachbar-Klassifikator:
  Zuordnung zu der Klasse des nächsten Nachbarpunkts
- Im Beispiel: Nächster Nachbar ist eine Schraube

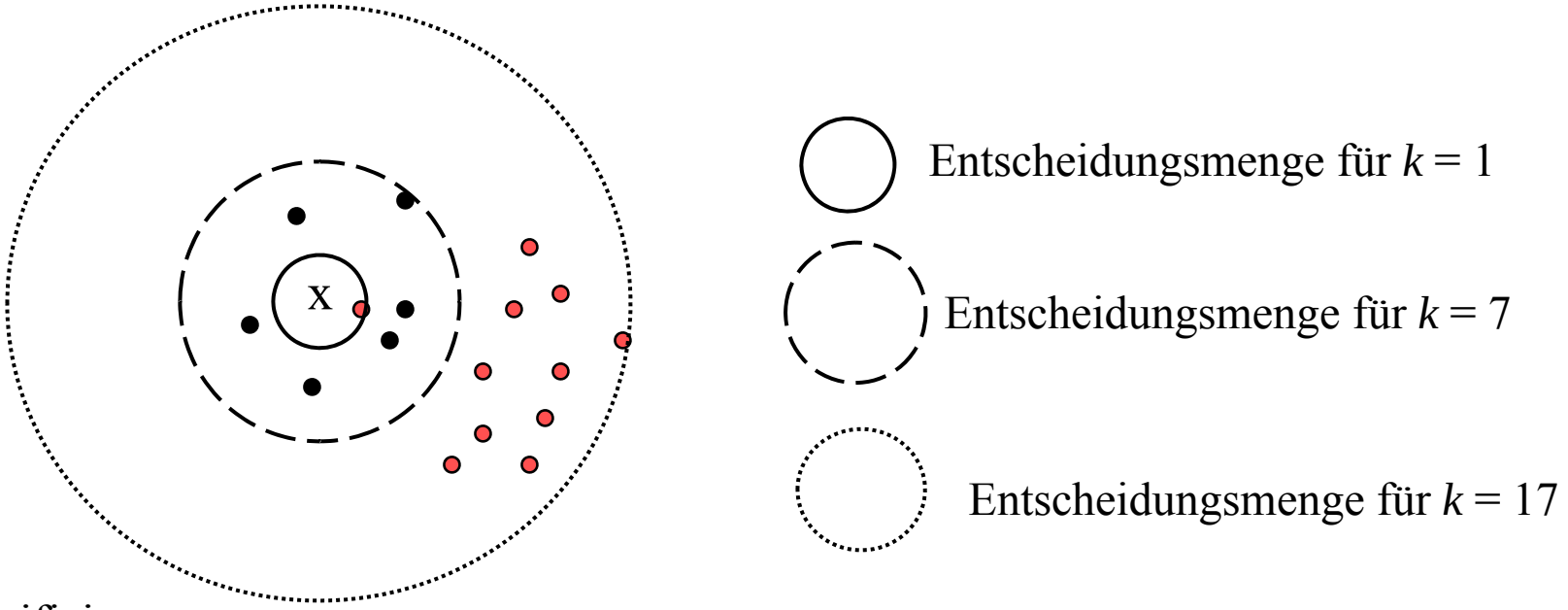
Nächste-Nachbarn-Klassifikatoren

- Problem: Punkt links oben wahrscheinlich nur Ausreißer
  => neues Objekt vermutlich grün statt rot
- Besser: Betrachte mehr als nur einen Nachbarn
  → *k*-Nächste-Nachbarn-Klassifikator
- *Entscheidungsmenge*
  die Menge der zur Klassifikation betrachteten *k*-nächsten Nachbarn
- *Entscheidungsregel*
  wie bestimmt man aus den Klassen der Entscheidungsmenge die Klasse des zu klassifizierenden Objekts?
  - Interpretiere Häufigkeit einer Klasse in der Entscheidungsmenge als Wahrscheinlichkeit der Klassenzugehörigkeit
  - Maximum-Likelihood-Prinzip: Mehrheitsentscheidung
  - Ggf. Gewichtung

Nächste-Nachbarn-Klassifikatoren

Wahl des Parameters k

- „zu kleines“ *k*: hohe Sensitivität gegenüber Ausreißern
- „zu großes“ *k*: viele Objekte aus anderen Clustern (Klassen) in der Entscheidungsmenge.
- mittleres *k*: höchste Klassifikationsgüte, oft $1 << k < 10$



x: zu klassifizieren

Nächste-Nachbarn-Klassifikatoren

Entscheidungsregel

- Standardregel
  - wähle die Mehrheitsklasse der Entscheidungsmenge
- Gewichtete Entscheidungsregel

  gewichte die Klassen der Entscheidungsmenge
  - nach Distanz, meist invers quadriert: $weight(dist) = 1/dist^2$
  - nach Verteilung der Klassen (oft sehr ungleich!)
    Problem: Klasse mit zu wenig Instanzen ($< k/2$) in der Trainingsmenge bekommt keine Chance, ausgewählt zu werden, selbst bei optimaler Distanzfunktion
    - Klasse A: 95 %, Klasse B 5 %
    - Entscheidungsmenge = {A, A, A, A, B, B, B}
    - Standardregel $\Rightarrow$ A, gewichtete Regel $\Rightarrow$ B

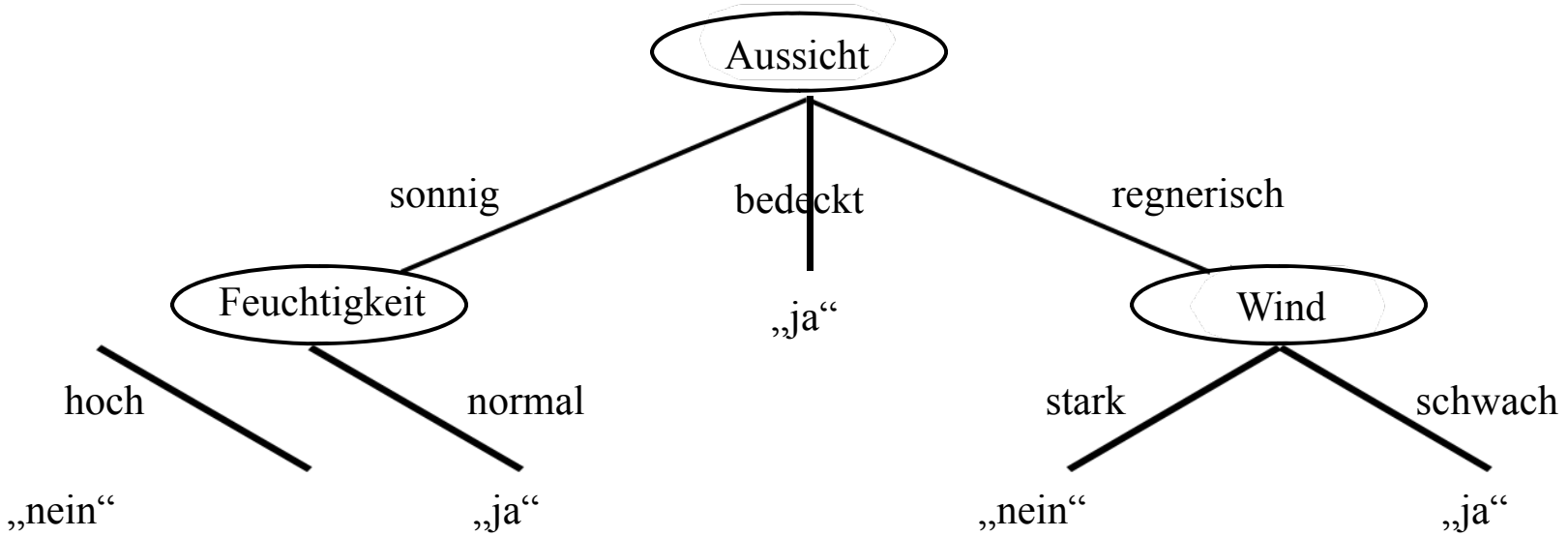
Entscheidungsbaum-Klassifikatoren

Ist heute ein Tag zum Tennisspielen?

Tag	Aussicht	Temperatur	Feuchtigkeit	Wind	Tennispielen
1	sonnig	heiß	hoch	schwach	nein
2	sonnig	heiß	hoch	stark	nein
3	bedeckt	heiß	hoch	schwach	ja
4	regnerisch	mild	hoch	schwach	ja
5	regnerisch	kühl	normal	schwach	ja
6	regnerisch	kühl	normal	stark	nein
7	bedeckt	kühl	normal	stark	ja
8	sonnig	mild	hoch	schwach	nein
9	sonnig	kühl	normal	schwach	ja
10	regnerisch	mild	normal	schwach	ja
11	sonnig	mild	normal	stark	ja
12	bedeckt	mild	hoch	stark	ja
13	bedeckt	heiß	normal	schwach	ja
14	regnerisch	mild	hoch	stark	nein

Entscheidungsbaum-Klassifikatoren

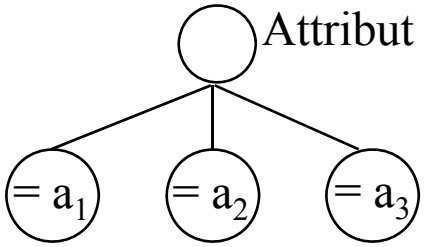
Möglicher Entscheidungsbaum

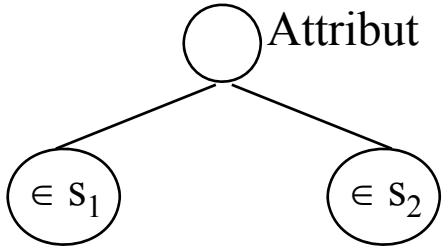


Entscheidungsbaum-Klassifikatoren

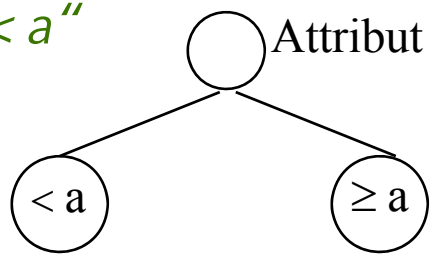
Typen von Splits

- Kategorische Attribute
  Split-Bedingungen der Form „*Attribut = a*“ or „*Attribut ∈ set*“
  viele mögliche Teilmengen





- Numerische Attribute
  Split-Bedingungen der Form „*Attribut < a*“
  viele mögliche Split-Punkte



Entscheidungsbaum-Klassifikatoren

- Wo sollen diskrete Attribute gesplittet werden?
- => An den Stellen, die die Qualität maximieren.
- Idee: Ordnen der numerischen Attributwerte

Wert	0.9	0.8	0.65	0.5	0.45	0.3	0.15	0.01
Klasse	A	A	B	B	B	A	A	A

Potentielle Splitkandidaten

- Teste die Kombination, die den höchsten Information Gain erzielen.

Entscheidungsbaum-Klassifikatoren

Qualitätsmaße für Splits

Gegeben

- eine Menge T von Trainingsobjekten
- eine disjunkte, vollständige Partitionierung $T_1, T_2, \ldots, T_m$ von T
- p_i die relative Häufigkeit der Klasse c_i in T

Gesucht

- ein Maß der *Unreinheit* einer Menge S von Trainingsobjekten in Bezug auf die Klassenzugehörigkeit
- ein Split von T in $T_1, T_2, \ldots, T_m$, der dieses Maß der Unreinheit *minimiert*

  $\Rightarrow$ Informationsgewinn, Gini-Index

Entscheidungsbaum-Klassifikatoren

Informationsgewinn

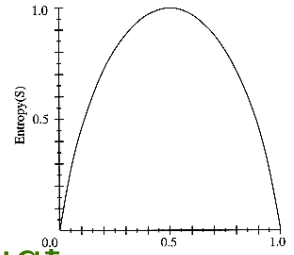
Entropie: minimale Anzahl von Bits zum Codieren der Nachricht, mit der man die Klasse eines zufälligen Trainingsobjekts mitteilen möchte.

Die *Entropie* für eine Menge T von Trainingsobjekten ist definiert als

$$entropie(T) = -\sum_{i=1}^{k} p_i \cdot \log p_i$$

entropie(T) = 0, falls $p_i = 1$ für ein i

entropie(T) = 1 für $k = 2$ Klassen mit $p_i = ½$



Das Attribut A habe die Partitionierung $T_1, T_2, \ldots, T_m$ erzeugt.

Der *Informationsgewinn* des Attributs A in Bezug auf T ist definiert als

$$Informationsgewinn(T, A) = entropie(T) - \sum_{i=1}^{m} \frac{|T_i|}{|T|} \cdot entropie(T_i)$$

Entscheidungsbaum-Klassifikatoren

Gini-Index

Gini-Index für eine Menge T von Trainingsobjekten

$$gini(T) = 1 - \sum_{j=1}^{k} p_j^2$$

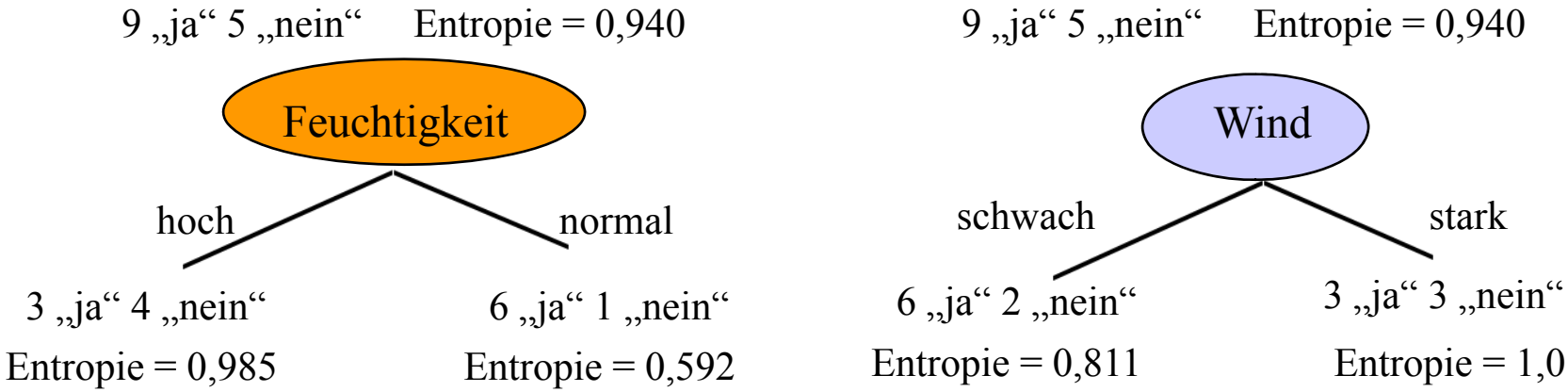
- kleiner Gini-Index ⇔ geringe Unreinheit,
- großer Gini-Index ⇔ hohe Unreinheit

Das Attribut A habe die Partitionierung $T_1, T_2, \ldots, T_m$ erzeugt.
Gini-Index des Attributs A in Bezug auf T ist definiert als

$$gini_A(T) = \sum_{i=1}^{m} \frac{|T_i|}{|T|} \cdot gini(T_i)$$

Entscheidungsbaum-Klassifikatoren

Beispiel



$$Informationsgewinn(T, Feuchtigkeit) = 0{,}94 - \frac{7}{14} \cdot 0{,}985 - \frac{7}{14} \cdot 0{,}592 = 0{,}151$$

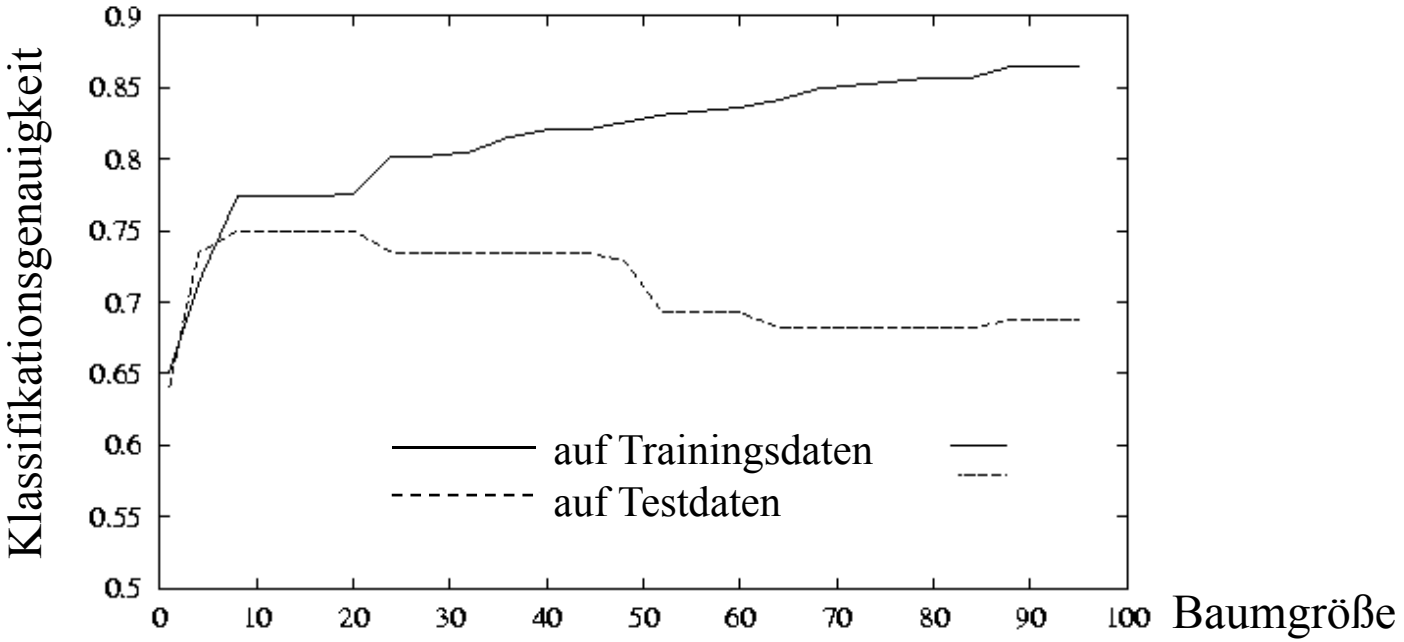
$$Informationsgewinn(T, Wind) = 0{,}94 - \frac{8}{14} \cdot 0{,}811 - \frac{6}{14} \cdot 1{,}0 = 0{,}048$$

$\Rightarrow$ Feuchtigkeit liefert den höheren Informationsgewinn

Overfitting

Overfitting bei der Konstruktion eines Entscheidungsbaums, wenn es zwei Entscheidungsbäume *E* und *E'* gibt mit

- *E* hat auf der Trainingsmenge eine kleinere Fehlerrate als *E'*,
- *E'* hat auf der Grundgesamtheit der Daten eine kleinere Fehlerrate als *E*.



Overfitting

Ansätze zum Vermeiden von Overfitting

Entfernen von fehlerhaften Trainingsdaten

- insbesondere widersprüchliche Trainingsdaten

Wahl einer geeigneten Größe der Trainingsmenge

- nicht zu klein, nicht zu groß

Wahl einer geeigneten Größe des minimum support

- *minimum support*:
  - Anzahl der Datensätze, die mindestens zu einem Blattknoten des Baums gehören müssen

minimum support >> 1

Overfitting

Ansätze zum Vermeiden von Overfitting

Wahl einer geeigneten Größe der minimum confidence

minimum confidence: Anteil, den die Mehrheitsklasse eines Blattknotens mindestens besitzen muss.

minimum confidence << 100%

Blätter können auch fehlerhafte Datensätze oder Rauschen „absorbieren“

nachträgliches Pruning des Entscheidungsbaums

Abschneiden der überspezialisierten Äste

Overfitting

Fehlerreduktions-Pruning

- Aufteilung der klassifizierten Daten in Trainingsmenge und Testmenge
- Konstruktion eines Entscheidungsbaums *E* für die Trainingsmenge
- Pruning von *E* mit Hilfe der Testmenge *T*
  - bestimme denjenigen Teilbaum von *E*, dessen Abschneiden den Klassifikationsfehler auf *T* am stärksten reduziert
  - entferne diesen Teilbaum
  - fertig, falls kein solcher Teilbaum mehr existiert

nur anwendbar, wenn genügend viele klassifizierte Daten zu Verfügung stehen

Ausblick

Data Mining und andere Wissenschaften

- Data Mining lebt von der Anwendung und muss für viele Anwendungsszenarien und Probleme zugeschnitten werden.

- Data Mining kann im Anwendungsgebiet (z.B. einer anderen Wissenschaft – Geographie, BWL, Kunst, Sprachwissenschaft, Physik, Biologie,…) zu neuen Erkenntnissen führen.

- Umgekehrt bietet ein konkretes Anwendungsszenario oft interessante Herausforderungen für die Forschung im Bereich Data Mining.