

Skript zur Vorlesung
Knowledge Discovery in Databases
im Wintersemester 2007/2008

Kapitel 3: Klassifikation

Skript © 2003 Johannes Abfalg, Christian Böhm, Karsten Borgwardt,
Martin Ester, Eshref Januzaj, Karin Kailing, Peer Kröger, Jörg Sander und
Matthias Schubert

<http://www.dbs.ifi.lmu.de/Lehre/KDD>

3. Klassifikation

Inhalt dieses Kapitels

- 3.1 Grundbegriffe der Klassifikation
- 3.2 Bayes-Klassifikatoren
- 3.3 Nächste-Nachbarn-Klassifikatoren
- 3.4 Entscheidungsbaum-Klassifikatoren
- 3.5 Neuronale Netze
- 3.5 Support Vector Machines and Kernel Learning
- 3.6 Hierarchische Klassifikation

3.1 Grundbegriffe der Klassifikation

Das Klassifikationsproblem

Gegeben: eine Menge O von Objekten des Formats (o_1, \dots, o_d)

mit Attributen A_i , $1 \leq i \leq d$, und Klassenzugehörigkeit $c_i, c_i \in C = \{c_1, \dots, c_k\}$

Gesucht: die Klassenzugehörigkeit für Objekte aus $DB \setminus O$

ein Klassifikator $K : DB \rightarrow C$

Abgrenzung zum Clustering

Klassifikation: Klassen apriori bekannt

Clustering: Klassen werden erst gesucht

Verwandtes Problem: *Vorhersage* (Prediction)

gesucht ist der Wert für ein numerisches Attribut Methode z.B. Regression.



50

Einleitung

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

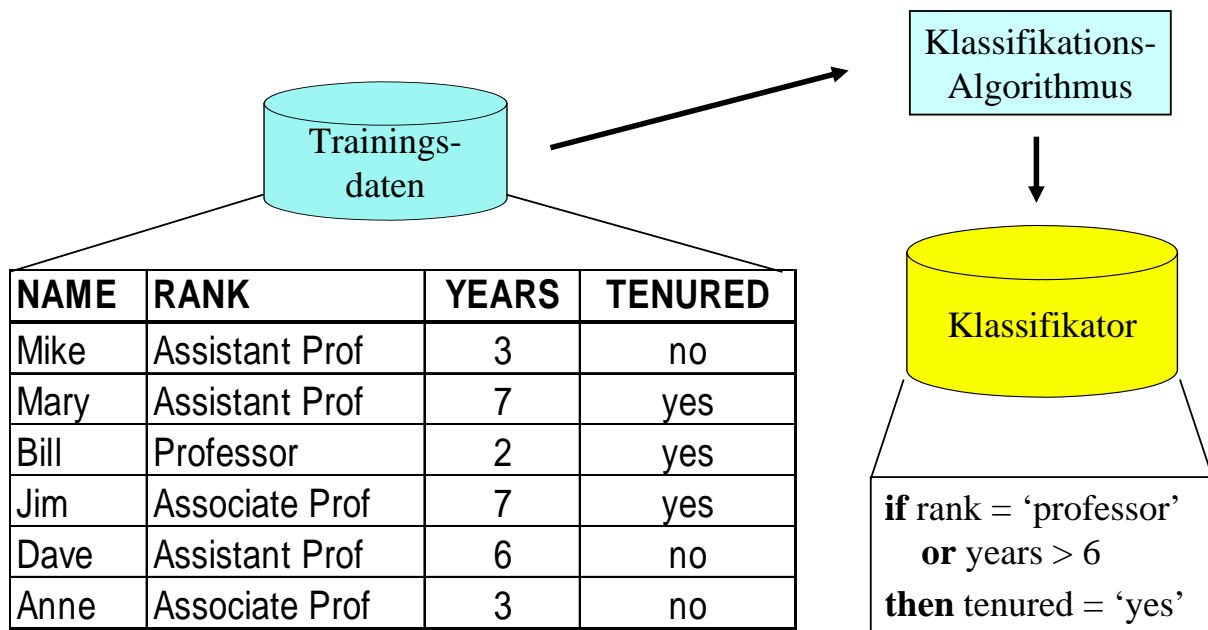
Einfacher Klassifikator

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

51

Der Prozess der Klassifikation

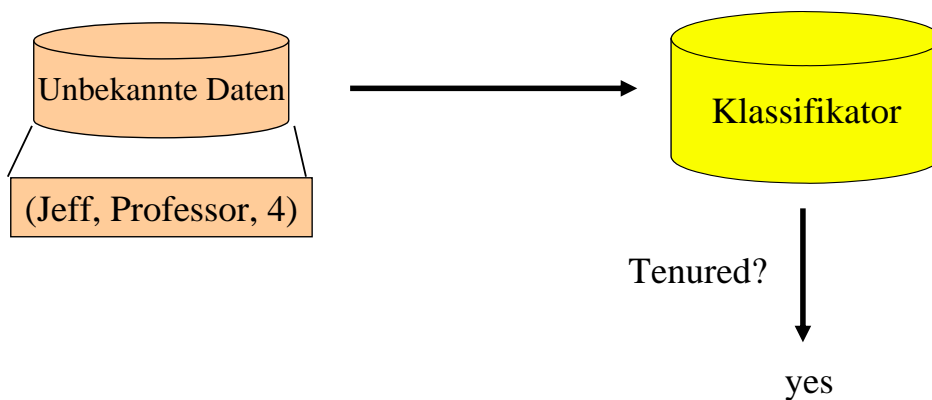
Konstruktion des Modells



52

Der Prozess der Klassifikation

Anwendung des Modells



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



53

Bewertung von Klassifikatoren

Grundbegriffe

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

Problem der Bewertung:

gewünscht ist gute Performanz auf ganz DB .

Klassifikator ist für TR optimiert.

Test auf TR erzeugt in der Regel viel bessere Ergebnisse, als auf $DB \setminus TR$.

Daher kein realistisches Bild der Performanz auf DB .

\Rightarrow *Overfitting*

54

Bewertung von Klassifikatoren

Train-and-Test

Bewertung ohne *Overfitting* durch Aufteilen von O in :

Trainingsmenge TR

zum Lernen des Klassifikators (Konstruktion des Modells)

Testmenge TE

zum Bewerten des Klassifikators

55

Bewertung von Klassifikatoren

Grundbegriffe

Train-and-Test nicht anwendbar, wenn nur wenige Objekte mit bekannter Klassenzugehörigkeit

Stattdessen: *m*-fache Überkreuz-Validierung (*m*-fold Cross-Validation)

m-fache Überkreuz-Validierung

teile die Menge *O* in *m* gleich große Teilmengen

verwende jeweils *m*-1 Teilmengen zum Training

und die verbleibende Teilmenge zur Bewertung

kombiniere die erhaltenen *m* Klassifikationsfehler

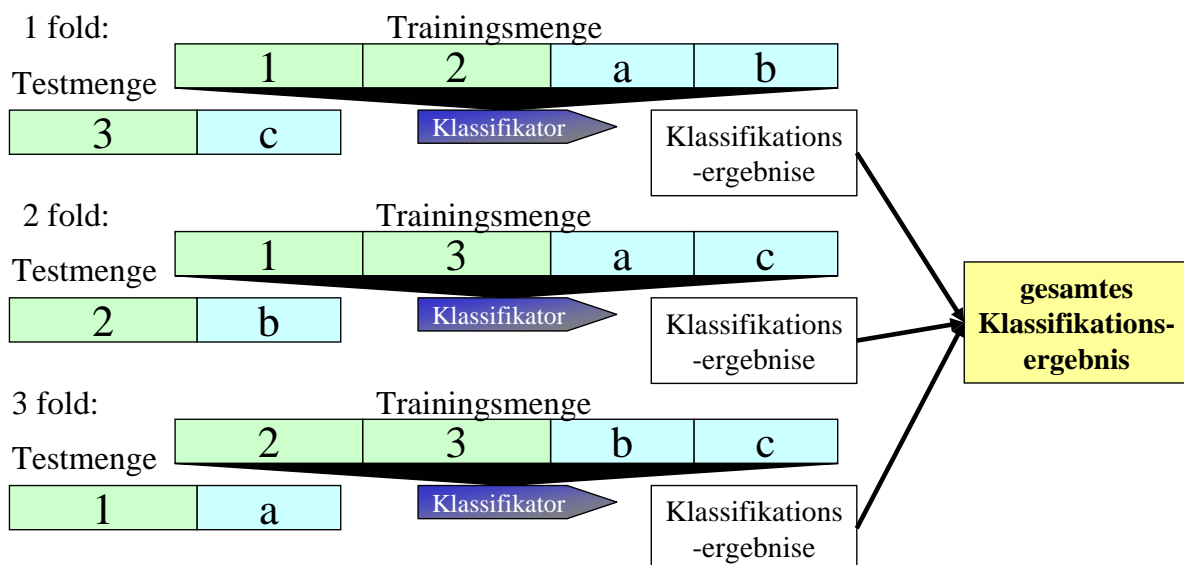
(und die *m* gefundenen Modelle!)

56

Bewertung von Klassifikatoren

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

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



57

Bewertung von Klassifikatoren

Ergebnis des Tests : Konfusionsmatrix (confusion matrix)

		klassifiziert als ...				
		Klasse 1	Klasse 2	Klasse 3	Klasse 4	other
tatsächliche Klasse ...	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 folgende Kennzahlen berechnen : Accuracy, Classification Error, Precision und Recall.

58

Bewertung von Klassifikatoren

Gütemaße für Klassifikatoren

Sei K ein Klassifikator, $TR \subseteq O$ die Trainingsmenge, $TE \subseteq O$ die Testmenge.

Bezeichne $C(o)$ die tatsächliche Klasse eines Objekts o .

Klassifikationsgenauigkeit (classification accuracy) von K auf TE :

$$G_{TE}(K) = \frac{|\{o \in TE \mid K(o) = C(o)\}|}{|TE|}$$

Tatsächlicher Klassifikationsfehler (true classification error)

$$F_{TE}(K) = \frac{|\{o \in TE \mid K(o) \neq C(o)\}|}{|TE|}$$

Beobachteter Klassifikationsfehler (apparent classification error)

$$F_{TR}(K) = \frac{|\{o \in TR \mid K(o) \neq C(o)\}|}{|TR|}$$

59

Bewertung von Klassifikatoren

Recall:

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

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

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

		Zugeordnete Klasse $K(o)$			
		1	2		
Tatsächl. Klasse $C(o)$	1				
	2				

• Precision:

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

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

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

60

Bewertung von Klassifikatoren

weitere Gütemaße für Klassifikatoren

Kompaktheit des Modells

z.B. Größe eines Entscheidungsbaums

Interpretierbarkeit des Modells

wieviel Einsichten vermittelt das Modell dem Benutzer?

Effizienz

- der Konstruktion des Modells
- der Anwendung des Modells

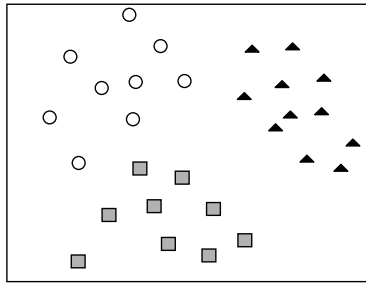
Skalierbarkeit für große Datenmengen für sekundärspeicherresidente Daten

Robustheit gegenüber Rauschen und fehlenden Werten

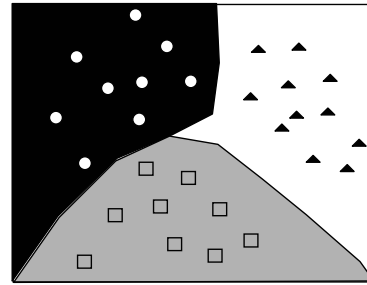
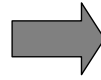
61

Überblick über Klassifikationsmethoden

Trainingsmenge mit 3 Klassen



3 Klassenbereiche (weiß, grau, schwarz)



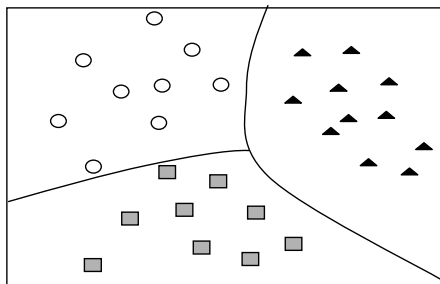
Alle Klassifikatoren legen beim Training Klassengrenzen fest.

Aber: Es gibt viele Methoden Klassengrenzen aus Trainingsdaten abzuleiten.

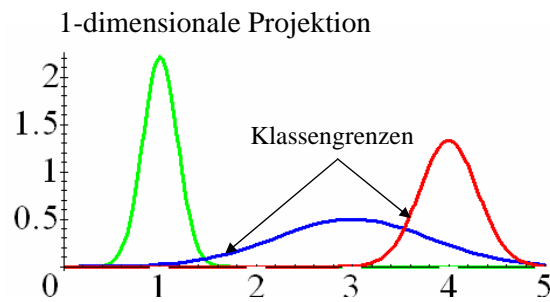
=> Unterschiedliche Klassifikatoren (statische Kl., Entscheidungsbäume, Support Vektor Maschinen, kNN Klassifikatoren, neuronale Netze, ...)

62

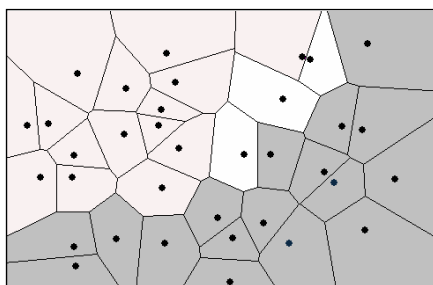
Motivation der Klassifikationsmethoden(1)



Bayes Klassifikatoren



Unterscheidung durch Dichtefunktionen.

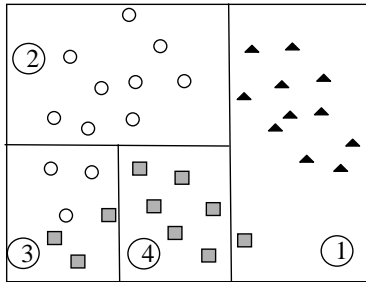


NN-Klassifikator

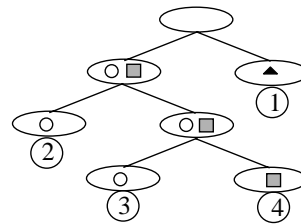
Unterscheidung durch Voronoi-Zellen
(1 nächster Nachbar Klassifikator)

63

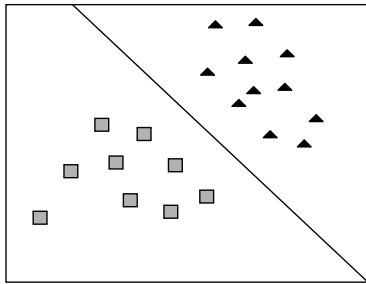
Motivation der Klassifikationsmethoden(2)



Entscheidungsbäume



Festlegen der Grenzen durch rekursive Unterteilung in Einzeldimension.



Support Vektor Maschinen

Grenzen über lineare Separation

64

Anwendungen Klassifikation

- Klassifikation von Risikoklassen bei Versicherungen und Kreditvergabe
- Funktionsvorhersage von Proteinen
- Gesichtserkennung
- Erkennen von relevanten Webinhalten
- Erkennen von Spam- Emails

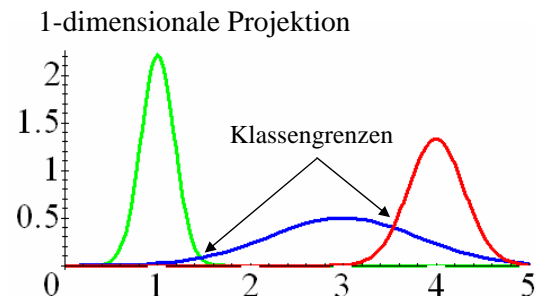
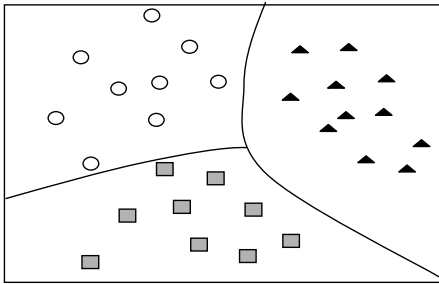
65

3.2 Bayes-Klassifikatoren

Was sind Bayes-Klassifikatoren?

Statistische Klassifikatoren

- Klassen werden durch statistische Prozesse beschrieben
- Beruht auf dem Satz von Bayes
- Bestimme Wahrscheinlichkeiten mit denen jeder Prozess das Objekt erklärt (Class-Membership-Probability)
- Vorhersage der wahrscheinlichsten Klasse (Maximum Likelihood Classification)



66

Überblick Bayes Klassifikatoren

1. Grundlagen statistischer Klassifikatoren

1. A-priori und A-posteriori Wahrscheinlichkeiten
2. Regel von Bayes
3. „Maximum Likelihood“ Klassifikation

• Klassifikatoren und Statistische Prozesse

- Naive Bayes
- Bayes Netzwerke
- LDA
- multivariate Gauss-Prozesse

67

Bayes-Klassifikatoren

Grundlagen

- Regeln und Fakten zur Klassifikation werden mit Hilfe des Satzes von Bayes als bedingte Wahrscheinlichkeiten formuliert
- A-Priori-Wahrscheinlichkeiten modellieren Faktenwissen über die Häufigkeit einer Klasse und das Auftreten von Merkmalen, z.B.
 - 20% der Objekte sind Äpfel
 - 30% sind Orangen
 - 50% der Objekte sind rund
 - 40% haben Farbe orange

} A-Priori Wahrsch. f. Klassenzugehörig.
} A-Priori Merkmalshäufigkeit
- Bedingte Wahrscheinlichkeiten („A-Posteriori“) modellieren Zusammenhänge zwischen Klassen und Merkmalen:
 - 100% der Orangen sind rund: $P(\text{rund} | \text{Orange}) = 100\%$
 - 100% der Äpfel sind rund: $P(\text{rund} | \text{Apfel}) = 100\%$
 - 90% der Orangen sind orange: $P(\text{orange} | \text{Orange}) = 90\%$

68

Bayes-Klassifikatoren

- Bei einem gegebenen Merkmals-Vektor M lässt sich die Wahrscheinlichkeit der Klassenzugehörigkeit zu Klasse C_i mit dem Satz von Bayes ermitteln:

$$P(C_i | M) = \frac{P(M | C_i) \cdot P(C_i)}{P(M)} = \frac{P(M | C_i) \cdot P(C_i)}{\sum_{c_j \in C} P(C_j) \cdot P(M | C_j)}$$

- Im Beispiel: Wahrscheinlichkeit, dass ein oranges Objekt eine Orange ist:

$$P(\text{Orange} | \text{orange}) = \frac{P(\text{orange} | \text{Orange}) \cdot P(\text{Orange})}{P(\text{orange})} = \frac{0.9 \cdot 0.3}{0.4} = 0.675$$

Die entsprechenden Wahrscheinlichkeiten werden aus den Trainingsdaten geschätzt

69

Bayes-Klassifikation

- Der Bayes-Klassifikator schätzt die Wahrscheinlichkeit der Klassenzugehörigkeit eines Merkmalsvektors
- Zur eindeutigen Zuordnung eines Klassen-Labels geht man meist nach dem Prinzip „Maximum Likelihood“ vor:

$$C = \underset{C_i}{\operatorname{argmax}} P(C_i | M) = \underset{C_i}{\operatorname{argmax}} \frac{P(M | C_i) \cdot P(C_i)}{P(M)} = \underset{C_i}{\operatorname{argmax}} P(M | C_i) \cdot P(C_i)$$

- Da $P(M)$ bei allen C_i gleich ist, ist nur das Produkt zu optimieren
- Beispiel:

$$\left. \begin{array}{l} \bullet P(\text{Apfel} \mid M) = 32\% \\ \bullet P(\text{Orange} \mid M) = 32\% \\ \bullet P(\text{Kiwi} \mid M) = 36\% \end{array} \right\} \Rightarrow C = \text{Kiwi}$$

70

Schätzung der Wahrscheinlichkeiten

- **„A-priori“ Wahrscheinlichkeiten**

Meistens: relative Häufigkeit in den Trainingsdaten.

Bsp: 7 Orangen, 2 Äpfel, 1 Stein $\Rightarrow P(\text{Orange}) = \frac{7}{7+2+1} = 70\%$

- **„A-posteriori“ Wahrscheinlichkeiten**

- Statistischer Prozess modelliert Zusammenhänge zwischen Merkmalen und einer Klasse
- Unterschiede verwendeter Prozesse:
 - Abhängigkeit der Merkmale (Korrelation oder Unabhängigkeit)
 - Verwendete Verteilungsfunktionen der Merkmalswerte (diskret, Normalverteilung, Multinomial...)
 - Beschaffenheit der Objekte (Vektor, Sequenz...)

71

1-dimensionale Verteilungen

Diskrete Merkmale

Auszählen relativer Häufigkeiten

Bsp:

$$P(\text{Form} = \text{rund} \mid A) = \frac{3}{4} = 75\%$$

$$P(\text{Farbe} = \text{grün} \mid A) = \frac{2}{4} = \frac{1}{2} = 50\%$$

$$P(\text{Form} = \text{oval} \mid A) = \frac{0}{4} = 0\%$$

ID	Form	Farbe	Klasse
1	rund	orange	A
2	rund	grün	A
3	rund	gelb	A
4	eckig	grün	A
5	oval	weiß	B

Problem: (Form = oval) => Klasse ≠ A

Man verwendet häufig „Smoothing“, d.h. $P(x \mid \text{Klasse}) > \varepsilon$.

mit $0 < \varepsilon \ll 1$.

D.h.
$$P(\text{Form} = \text{oval} \mid A) = \max\left(\frac{0}{4}, \varepsilon\right) = \varepsilon$$

1-dimensionale Verteilungen

Kontinuierliche metrische Attribute

diskrete Approximation

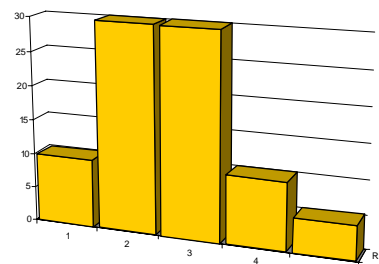
$$P(9.0 < \text{Durchmesser} \leq 9.5 \mid \text{Orange}) = 10\%$$

$$P(9.5 < \text{Durchmesser} \leq 10.0 \mid \text{Orange}) = 30\%$$

$$P(10.0 < \text{Durchmesser} \leq 10.5 \mid \text{Orange}) = 30\%$$

$$P(10.5 < \text{Durchmesser} \leq 11.0 \mid \text{Orange}) = 10\%$$

$$P(11.0 < \text{Durchmesser} \leq 11.5 \mid \text{Orange}) = 5\%$$



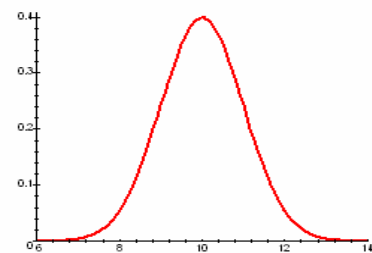
Wahrscheinlichkeits-Dichtefunktionen

z.B. Orangen haben einen Durchmesser von 10 ± 1 cm:

$$p(\text{Durchmesser} \mid \text{Orange}) = N(10, 1)$$

meist Berechnung nach Normalverteilung:

$$P(x \mid C) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad \text{wobei} \quad \mu = \frac{\sum_{x \in TR} x}{|TR|} \quad \text{und} \quad \sigma = \sqrt{\frac{\sum_{x \in TR} (x-\mu)^2}{|TR|}}$$



Motivation

Bei hochdimensionalen Merkmalsvektoren schwierige Schätzung der bedingten Wahrscheinlichkeiten $P(M | C)$ und damit $P(C | M)$:

- M besteht aus vielen einzelnen Komponenten, die UND-verknüpft sind:

$$P(C | M_1 \wedge M_2 \wedge \dots) = \frac{P(M_1 \wedge M_2 \wedge \dots | C) \cdot P(C)}{P(M_1 \wedge M_2 \wedge \dots)}$$

- Bei d verschiedenen Merkmalen und jeweils r verschiedenen Werten ergeben sich r^d verschiedene Merkmalskombinationen

Probleme:

- Die Wahrscheinlichkeiten lassen sich nicht mehr abspeichern
- Man bräuchte $\gg r^d$ Trainingsdatensätze, um die Wahrscheinlichkeit der einzelnen Merkmalskombinationen überhaupt ermitteln zu können

74

Naive Bayes-Klassifikation

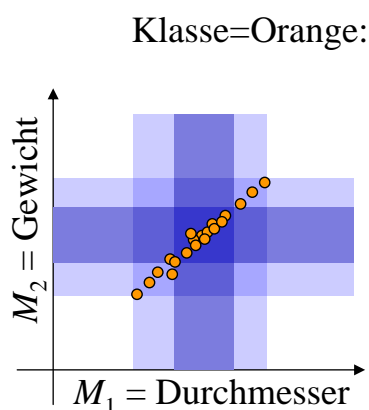
Lösung dieses Problems beim naiven Bayes-Klassifikator:

Annahme der Bedingten Unabhängigkeit

d.h. bei jeder einzelnen Klasse werden die Merkmale so behandelt als wären sie voneinander statistisch unabhängig:

$$P(M_1 \wedge M_2 | C) = P(M_1 | C) \cdot P(M_2 | C)$$

Was bedeutet dies?



- Annahme kann falsch sein
- Dies führt *nicht* unbedingt dazu, dass die Klassifikation versagt
- Aber schlechte Leistung, wenn...
 - alle Merkmale bei mehreren Klassen etwa gleich verteilt sind
 - Unterschiede nur in „Relationen“ der Merkmale zueinander

75

Naive Bayes-Klassifikation

Damit ist die Wahrscheinlichkeit der Zugehörigkeit zu Klasse C_i :

$$\begin{aligned} P(C_i | M_1 \wedge M_2 \wedge \dots) &= \frac{P(C_i) \cdot P(M_1 \wedge M_2 \wedge \dots | C_i)}{P(M_1 \wedge M_2 \wedge \dots)} \\ &= \frac{P(C_i) \cdot \prod_j P(M_j | C_i)}{\sum_k P(C_k) \prod_j P(M_j | C_k)} \end{aligned}$$

Auch hier ist der Nenner für alle Klassen gleich, so dass nur der Zähler zu maximieren ist:

$$C = \operatorname{argmax}_{C_i} \{P(C_i) \cdot \prod_j P(M_j | C_i)\}$$

76

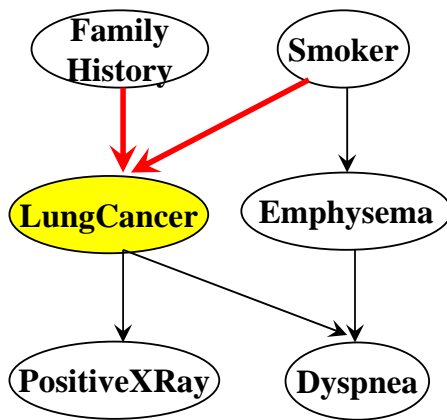
Bayes-Netzwerke

Grundbegriffe

- Graph mit Knoten = *Zufallsvariable* und Kante = *bedingte Abhängigkeit*
- Jede Zufallsvariable ist bei gegebenen Werten für die Vorgänger-Variablen bedingt unabhängig von allen Zufallsvariablen, die keine Nachfolger sind.
- Für jeden Knoten (Zufallsvariable): Tabelle der bedingten Wahrscheinlichkeiten
- Trainieren eines Bayes-Netzwerkes
 - bei gegebener Netzwerk-Struktur und allen bekannten Zufallsvariablen
 - bei gegebener Netzwerk-Struktur und teilweise unbekanntem Zufallsvariablen
 - bei apriori unbekannter Netzwerk-Struktur

77

Beispiel



	FH,S	FH, ¬S	¬FH,S	¬FH, ¬S
LC	0.8	0.5	0.7	0.1
~LC	0.2	0.5	0.3	0.9

bedingte Wahrscheinlichkeiten für LungCancer

bei gegebenen Werten für FamilyHistory und Smoker liefert der Wert für Emphysema keine zusätzliche Information über LungCancer

Lineare Diskriminanz Analyse

- Modelliere alle Klassen als multivariate Normalverteilungen
- Berücksichtigt Korrelationen der Attribute
- Varianzen und Korrelationen für alle Klassen gleich

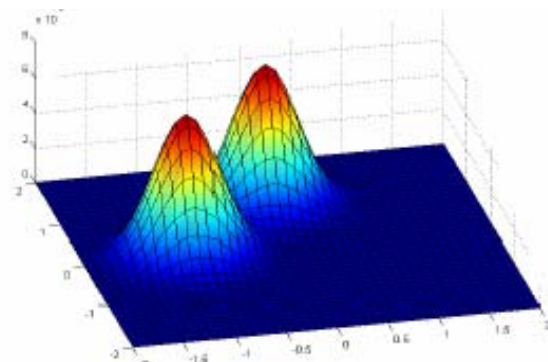
Basis multivariate Normalverteilung (Gauss-Verteilung)

$$P(x|C) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} e^{-\frac{1}{2} \cdot (x-\mu)^T \cdot (\Sigma)^{-1} \cdot (x-\mu)}$$

$$\text{Erwartungsvektor: } \mu = \frac{\sum_{x \in TR} x}{|TR|}$$

Kovarianzmatrix :

$$\Sigma(i, j) = \frac{\sum_{x \in TR} (x_i - \mu_i) \cdot (x_j - \mu_j)}{|TR|}$$



Eigenschaften:

- Korrelation zwischen i und j
- Varianz in der Diagonalen

Lineare Diskriminanz Analyse

Training:

1. Bestimme μ_C und Σ_C für alle Klassen C.
2. Middle globale Kovarianzmatrix Σ .
(Gewichteter Durchschnitt der Kovarianzmatritzen aller Klassen) $\Sigma = \frac{\sum_{C_i \in C} \Sigma_{C_i}}{|C|}$

Klassifikation:

$$\begin{aligned} \arg \max_{C_i \in C} P(x | C_i) &= \arg \max_{C_i \in C} \left(\frac{1}{\sqrt{(2\pi)^d |\Sigma|}} e^{-\frac{1}{2} \cdot (x - \mu_{C_i})^T \cdot (\Sigma)^{-1} \cdot (x - \mu_{C_i})} \cdot P(C_i) \right) \\ &= \arg \max_{C_i \in C} \left(-\frac{1}{2} \cdot (x - \mu_{C_i})^T \cdot (\Sigma)^{-1} \cdot (x - \mu_{C_i}) + \log(P(C_i)) \right) \\ &= \arg \max_{C_i \in C} \left(\underbrace{x^T (\Sigma)^{-1} \mu_{C_i} - \frac{1}{2} \mu_{C_i}^T (\Sigma)^{-1} \mu_{C_i} + \log(P(C_i))}_{\text{Lineare Diskriminanzfunktion}} \right) = \sigma_{C_i}(x) \end{aligned}$$

80

Lineare Diskriminanz Analyse

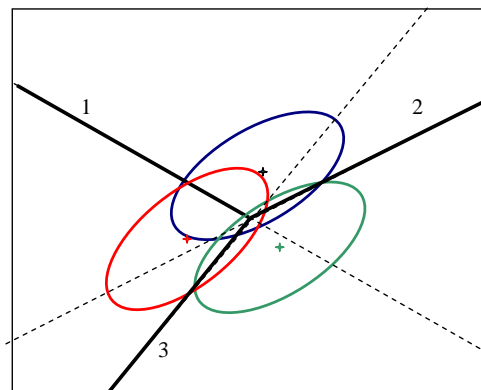
Beobachtung: Da nur Erwartungswerte unterschiedlich
 \Rightarrow Lineare Separation

Man muss nicht die Wahrscheinlichkeit berechnen.

Es reicht die Auswertung der folgenden Diskriminanzfunktion:

$$\sigma_{C_i}(x) = x^T \Sigma^{-1} \mu_{C_i} - \frac{1}{2} \mu_{C_i}^T \Sigma^{-1} \mu_{C_i} + \log P(C_i)$$

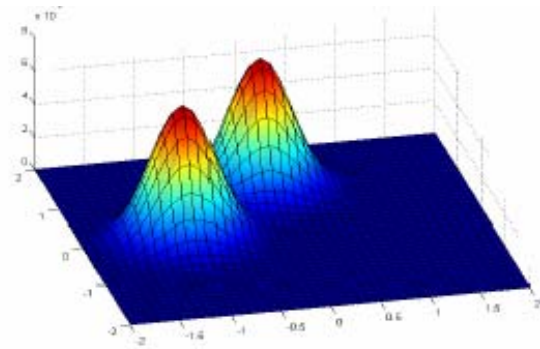
Klasse mit maximalem $\sigma_C(x)$ wird vorhergesagt.



81

Multivariate Gauss-Prozesse

- Modelliere jede Klasse als multivariate Normalverteilung (Vektoren im R^d)
- Berücksichtigt Korrelationen der Attribute
- Hier: Varianzen und Korrelationen für alle Klassen **individuell !!!!!**
- Berechnung der Wahrscheinlichkeiten zur Klassifikation (Maximum Likelihood)



Probleme:

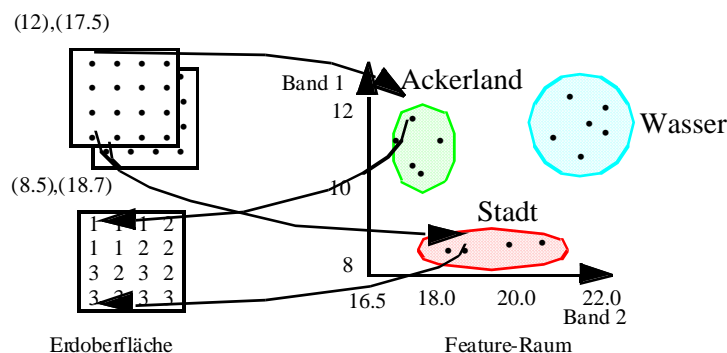
Braucht sehr viel Trainingsobjekte für jede Klasse, um signifikante Korrelationswerte zu bestimmen.

82

Interpretation von Rasterbildern

Motivation

- *automatische* Interpretation von d Rasterbildern eines bestimmten Gebiets für jedes Pixel ein d -dimensionaler Grauwertvektor (o_1, \dots, o_d)
- verschiedene Oberflächenbeschaffenheiten der Erde besitzen jeweils ein charakteristisches Reflexions- und Emissionsverhalten

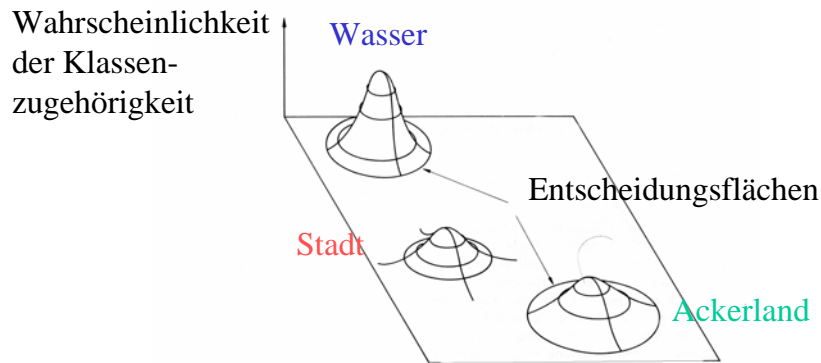


83

Interpretation von Rasterbildern

Grundlagen

- Anwendung des Bayes-Klassifikators mit Gauss Prozess
- Schätzung der $P(o | c)$ ohne Annahme der bedingten Unabhängigkeit
- Annahme einer d -dimensionalen Normalverteilung für die Grauwertvektoren einer Klasse



84

Interpretation von Rasterbildern

Methode

- Zu schätzen aus den Trainingsdaten

μ_i : d -dimensionaler Mittelwertvektor aller Feature-Vektoren der Klasse c_i

Σ_i : Kovarianzmatrix der Klasse c_i
 $d \cdot d$

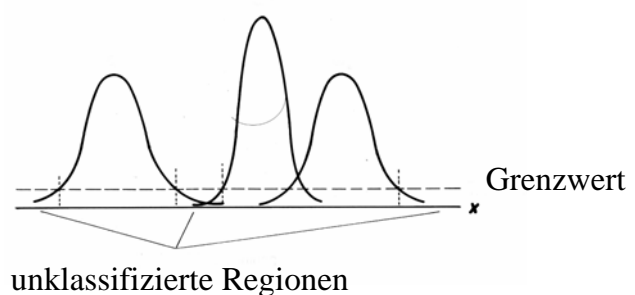
- Probleme der Entscheidungsregel

- Likelihood für die gewählte

Klasse sehr klein

- Likelihood für mehrere

Klassen ähnlich



85

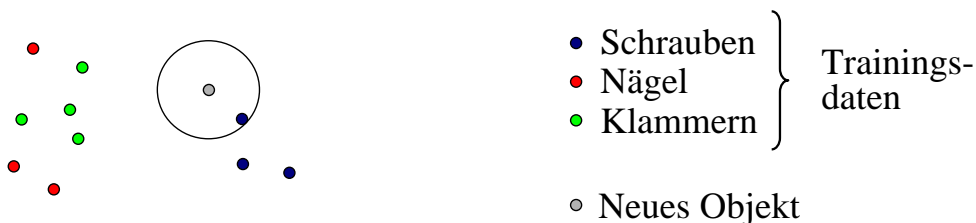
Bayes-Klassifikatoren

Diskussion

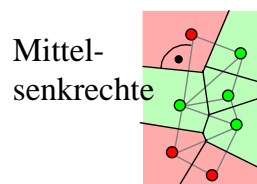
- + hohe Klassifikationsgenauigkeit in vielen Anwendungen
- + Inkrementalität
Klassifikator kann einfach an neue Trainingsobjekte adaptiert werden
- + Einbezug von Anwendungswissen
- Anwendbarkeit
die erforderlichen bedingten Wahrscheinlichkeiten sind oft unbekannt
- Ineffizienz
bei sehr vielen Attributen
insbesondere Bayes-Netzwerke

86

3.3 Nächste-Nachbarn-Klassifikatoren

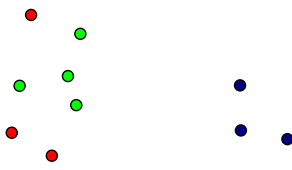


- Instanzbasiertes Lernen (*instance based learning*)
- Einfachster Nächste-Nachbar-Klassifikator:
Zuordnung zu der Klasse des nächsten Nachbarpunkts
- Im Beispiel: Nächster Nachbar ist eine Schraube
- Regionen der Klassenzuordnung können als *Voronoi-Diagramme* dargestellt werden:



87

Nächste-Nachbarn-Klassifikatoren

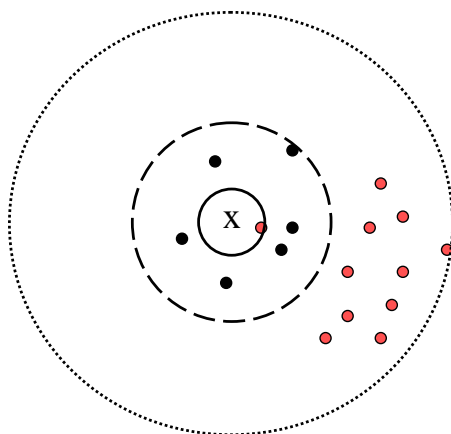


- Problem: Punkt rechts oben wahrscheinlich nur Ausreißer
- Besser: Betrachte mehr als nur einen Nachbarn
→ k -Nächste-Nachbarn-Klassifikator
- *Entscheidungsmenge*
die Menge der zur Klassifikation betrachteten k -nächsten Nachbarn
- *Entscheidungsregel*
wie bestimmt man aus den Klassen der Entscheidungsmenge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

88

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 $1 \ll k < 10$



- Entscheidungsmenge für $k = 1$
- Entscheidungsmenge für $k = 7$
- Entscheidungsmenge für $k = 17$

x: zu klassifizieren

89

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!)

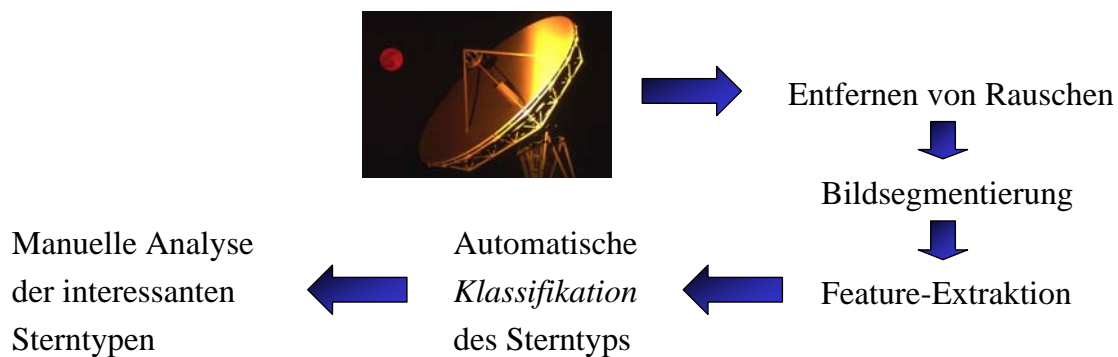
Problem: Klasse mit zu wenig Instanzen ($< k/2$) in der Trainingsmenge bekommt keine Chance, ausgewählt zu werden, selbst bei optimaler Distanzfunktion

- Klasse A: 95 %, Klasse B 5 %
- Entscheidungsmenge = {A, A, A, A, B, B, B}
- Standardregel \Rightarrow A, gewichtete Regel \Rightarrow B

90

Klassifikation von Sternen

Analyse astronomischer Daten



Klassifikation des Sterntyps mit Nächste-Nachbarn-Klassifikator
basierend auf dem Hipparcos-Katalog

91

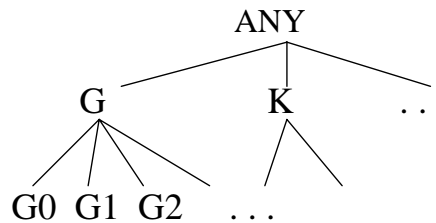
Klassifikation von Sternen

Hipparcos-Katalog [ESA 1998]

- enthält ca. 118 000 Sterne
- mit 78 Attributen (Helligkeit, Entfernung, Farbe, . . .)
- Klassenattribut: Spektraltyp (Attribut H76)

z.B.

H76: G0
 H76: G7.2
 H76: KIII/IV



- Werte des Spektraltyps sind vage
- Hierarchie von Klassen
 ➔ benutze die erste Ebene der Klassenhierarchie

Klassifikation von Sternen

Verteilung der Klassen

Klasse	#Instanzen	Anteil Instanzen	
K	32 036	27.0	} häufige Klassen
F	25 607	21.7	
G	22 701	19.3	
A	18 704	15.8	
B	10 421	8.8	
M	4 862	4.1	
O	265	0.22	} seltene Klassen
C	165	0.14	
R	89	0.07	
W	75	0.06	
N	63	0.05	
S	25	0.02	
D	27	0.02	

Klassifikation von Sternen

Experimentelle Untersuchung [Poschenrieder 1998]

Distanzfunktion

- mit 6 Attributen (Farbe, Helligkeit und Entfernung)
- mit 5 Attributen (ohne Entfernung)
⇒ beste Klassifikationsgenauigkeit mit 6 Attributen

Anzahl k der Nachbarn

⇒ beste Klassifikationsgenauigkeit für $k = 15$

Entscheidungsregel

- Gewichtung nach Distanz
- Gewichtung nach Klassenverteilung
⇒ beste Klassifikationsgenauigkeit bei Gewichtung nach Distanz
aber nicht nach Klassenverteilung

94

Klassifikation von Sternen

Klasse	Falsch klassifiziert	Korrekt klassifiziert	Klassifikationsgenauigkeit
K	408	2338	85.1%
F	350	2110	85.8%
G	784	1405	64.2%
A	312	975	75.8%
B	308	241	43.9%
M	88	349	79.9%
C	4	5	55.6%
R	5	0	0%
W	4	0	0%
O	9	0	0%
N	4	1	20%
D	3	0	0%
S	1	0	0%
Total	2461	7529	75.3%



hohe Klassifikationsgenauigkeit für die häufigen Klassen, schlechte Genauigkeit für die seltenen Klassen

die meisten seltenen Klassen besitzen weniger als $k / 2 = 8$ Instanzen!

95

Nächste-Nachbarn-Klassifikatoren

Diskussion

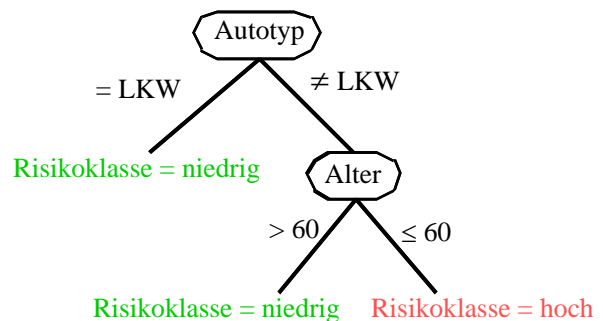
- + Anwendbarkeit
 - erfordert als Eingabe nur die Trainingsdaten
- + hohe Klassifikationsgenauigkeit
 - in vielen Anwendungen
- + inkrementell
 - Klassifikator kann sehr einfach an neue Trainingsobjekte adaptiert werden
- + auch zur Vorhersage einsetzbar
- Ineffizienz bei der Auswertung des “Modells”
 - erfordert k -nächste-Nachbarn Anfrage an die Datenbank
- liefert kein explizites Wissen über die Klassen

96

3.4 Entscheidungsbaum-Klassifikatoren

Motivation

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



finden explizites Wissen



Entscheidungsbäume sind für die meisten Benutzer verständlich

97

Grundbegriffe

- Ein *Entscheidungsbaum* ist ein Baum mit folgenden Eigenschaften:
 - ein innerer Knoten repräsentiert ein Attribut,
 - eine Kante repräsentiert einen Test auf dem Attribut des Vaterknotens,
 - ein Blatt repräsentiert eine der Klassen.
- Konstruktion eines Entscheidungsbaums
anhand der Trainingsmenge
Top-Down
- Anwendung eines Entscheidungsbaums
Durchlauf des Entscheidungsbaum von der Wurzel zu einem der Blätter
 ➡ eindeutiger Pfad
 Zuordnung des Objekts zur Klasse des erreichten Blatts

98

Konstruktion eines Entscheidungsbaums

Basis-Algorithmus

- Anfangs gehören alle Trainingsdatensätze zur Wurzel.
- Das nächste Attribut wird ausgewählt (Splitstrategie).
- Die Trainingsdatensätze werden unter Nutzung des Splitattributs partitioniert.
- Das Verfahren wird rekursiv für die Partitionen fortgesetzt.

➡ lokal optimierender Algorithmus

Abbruchbedingungen

- keine weiteren Splitattribute
- alle Trainingsdatensätze eines Knotens gehören zur selben Klasse

99

Entscheidungsbaum-Klassifikatoren

Beispiel

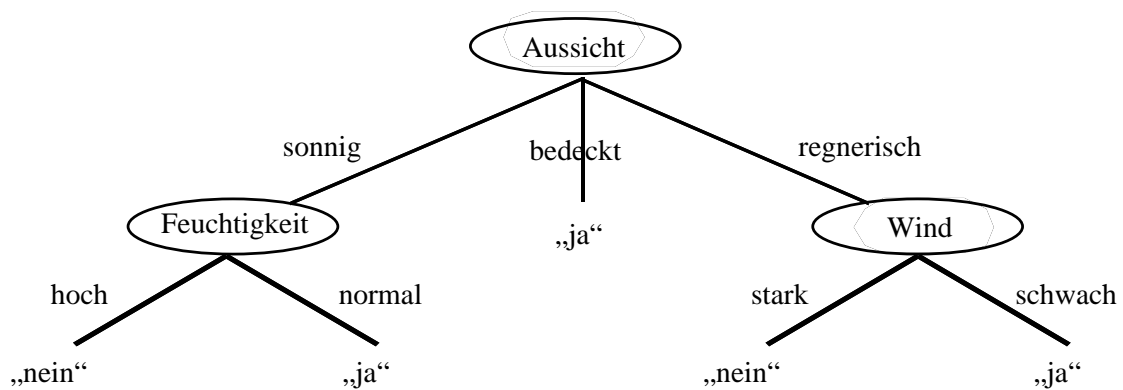
Tag	Aussicht	Temperatur	Feuchtigkeit	Wind	Tennispielen
1	sonnig	heiß	hoch	schwach	nein
2	sonnig	heiß	hoch	stark	nein
3	bedeckt	heiß	hoch	schwach	ja
4	regnerisch	mild	hoch	schwach	ja
5	regnerisch	kühl	normal	schwach	ja
6	regnerisch	kühl	normal	stark	nein
7

Ist heute ein Tag zum Tennispielen?

100

Entscheidungsbaum-Klassifikatoren

Beispiel



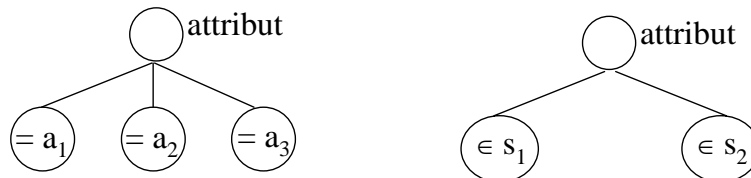
101

Splitstrategien

Typen von Splits

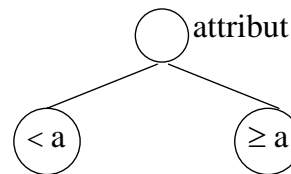
Kategorische Attribute

- Splitbedingungen der Form „ $attribut = a$ “ or „ $attribut \in set$ “
- viele mögliche Teilmengen



Numerische Attribute

- Splitbedingungen der Form „ $attribut < a$ “
- viele mögliche Splitpunkte



102

Splitstrategien

Qualitätsmaße für Splits

Gegeben

- eine Menge T von Trainingsobjekten
- eine disjunkte, vollständige Partitionierung T_1, T_2, \dots, 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, \dots, T_m , der dieses Maß der Unreinheit *minimiert*
 - ➔ Informationsgewinn, Gini-Index

103

Splitstrategien

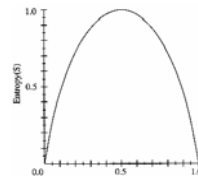
Informationsgewinn

- 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

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

$$\text{entropie}(T) = 0, \text{ falls } p_i = 1 \text{ für ein } i$$

$$\text{entropie}(T) = 1 \text{ für } k = 2 \text{ Klassen mit } p_i = 1/2$$



- Das Attribut A habe die Partitionierung T_1, T_2, \dots, T_m erzeugt.
- Der *Informationsgewinn* des Attributs A in Bezug auf T ist definiert als

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

104

Splitstrategien

Gini-Index

- *Gini-Index* für eine Menge T von Trainingsobjekten

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

kleiner Gini-Index \Leftrightarrow geringe Unreinheit,

großer Gini-Index \Leftrightarrow hohe Unreinheit

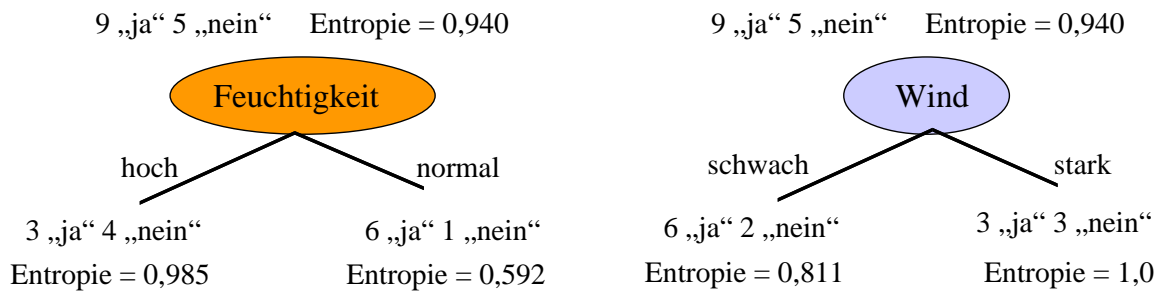
- Das Attribut A habe die Partitionierung T_1, T_2, \dots, T_m erzeugt.
- *Gini-Index* des Attributs A in Bezug auf T ist definiert als

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

105

Splitstrategien

Beispiel



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

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

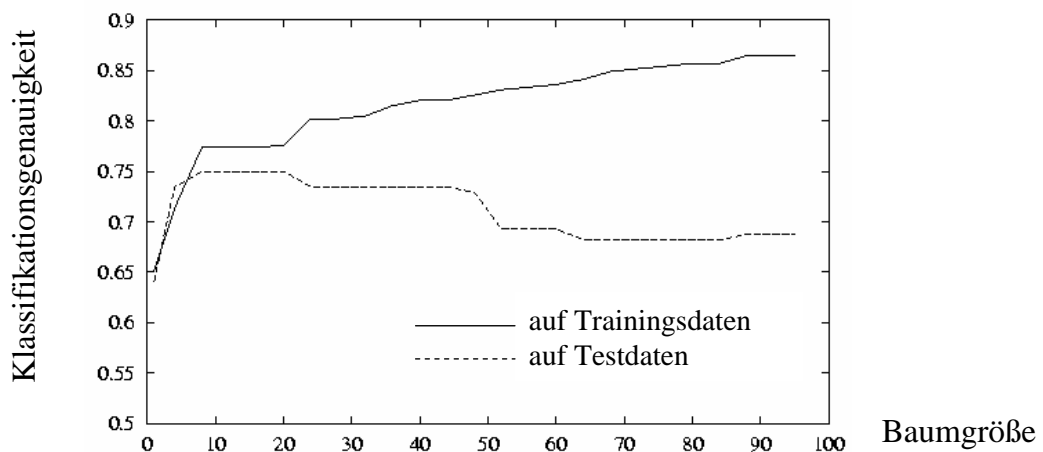
➔ Feuchtigkeit liefert den höheren Informationsgewinn

Overfitting

Einführung

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 $\gg 1$

108

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* $\ll 100\%$
- Blätter können auch fehlerhafte Datensätze oder Rauschen „absorbieren“
- nachträgliches Pruning des Entscheidungsbaums
 - Abschneiden der überspezialisierten Äste

109

Pruning von Entscheidungsbäumen

Fehlerreduktions-Pruning [Mitchell 1997]

- 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



110

Numerische Splitgrenzen

Wo sollen diskrete Attribute gesplittet werden?

=> An den Stellen, die den Information Gain maximieren.

Idee: Ordnen der numerischen Attributwerte

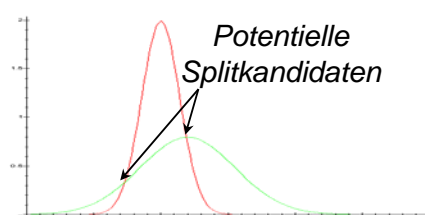
Wert	0.9	0.8	0.65	0.5	0.45	0.3	0.15	0.01
Klasse	A	A	B	B	B	A	A	A

←—————→
Potentielle Splitkandidaten

Teste die Kombination, die den höchsten Information Gain erzielen.

Schnellere Methode:

- Bilde Gauß-Kurve über alle Klassen
- Wähle Schnittpunkte der Gauß-Kurven als Kandidaten.



111

Diskussion

- + Interpretation des gefundenen Baumes relativ einfach
- + Implizite Gewichtung der Attribute
- + Leistungsfähiger Klassifikator, häufig in der Praxis verwendet
- + Effiziente Auswertung des gefundenen Modells

- Finden eines optimalen Entscheidungsbaums ist exponentiell
- Heuristische Methoden können nur lokales Optimum finden
- Anfällig für Overfitting

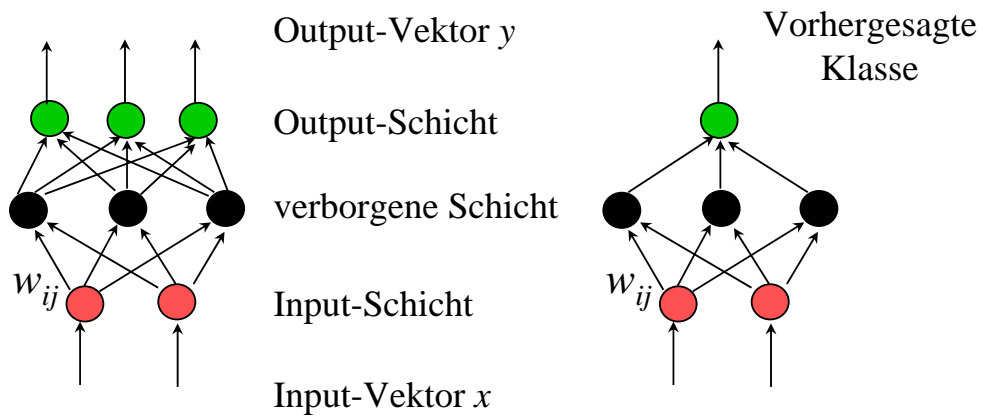
3.4 Neuronale Netze

Grundlagen [Bigus 1996], [Bishop 1995]

- Paradigma für ein Maschinen- und Berechnungsmodell
- Funktionsweise ähnlich der von biologischen Gehirnen
- *Neuronales Netz*: Menge von Neuronen, über Kanten miteinander verbunden
- *Neuron*: entspricht biologischem Neuron
 - Aktivierung durch Input-Signale an den Synapsen
 - Erzeugung eines Output-Signals, das zu anderen Neuronen weitergeleitet wird
- Organisation eines neuronalen Netzes
 - Input-Schicht, verborgene Schichten, Output-Schicht*
 - Knoten einer Schicht mit allen Knoten der vorhergehenden Schicht verbunden

Grundlagen

- Kanten besitzen *Gewichte*
- Funktion eines neuronalen Netzes



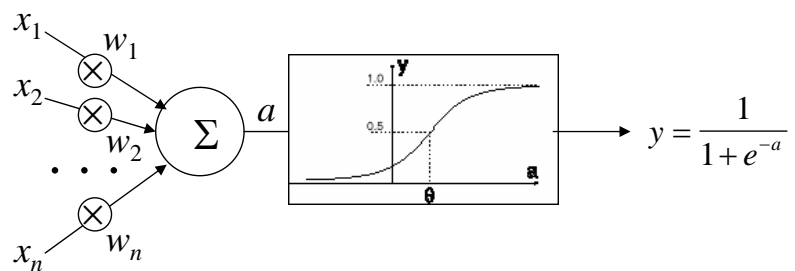
114

Neuronen

- allgemeines Neuron

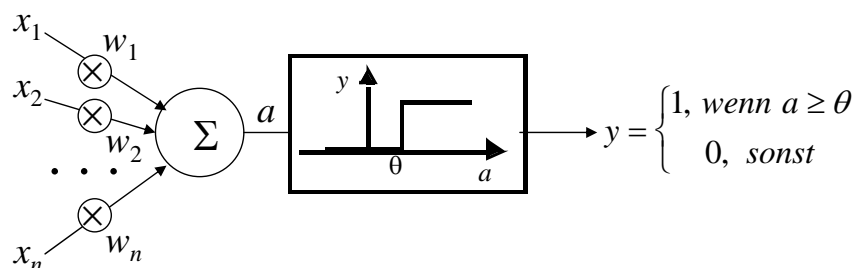
a : Aktivierungswert

$$a = \sum_{i=1}^n w_i \cdot x_i$$



- *Threshold Logic Unit*

(TLU)



115

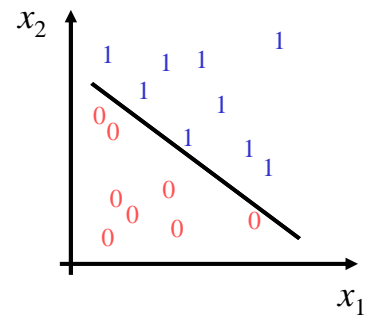
Neuronen

- Klassifikation mit Hilfe einer TLU

repräsentiert eine (Hyper-)Ebene $\sum_{i=1}^n w_i \cdot x_i = \theta$

links von der Ebene: Klasse 0

rechts von der Ebene: Klasse 1



- Trainieren einer TLU

Lernen der „richtigen“ Gewichte zur Unterscheidung der zwei Klassen

Iterative Anpassung der Gewichte w_{ij}

Rotation der durch w und θ gegebene Hyperebene um einen kleinen Betrag in Richtung v , wenn v noch nicht auf der richtigen Seite der Ebene liegt

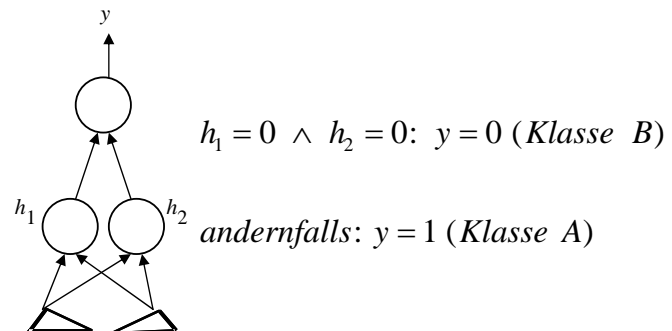
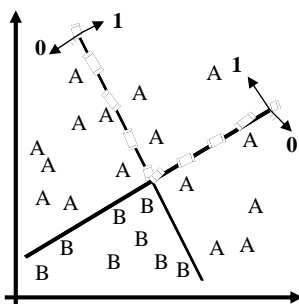
116

Kombination mehrerer Neuronen

- zwei Klassen, die nicht linear separierbar sind:

➡ zwei innere Knoten und ein Output-Knoten

- Beispiel



117

Lernalgorithmus für komplexe Neuronale Netze

- bei Abweichung von vorhergesagter und tatsächlicher Klasse:
 - Anpassung der Gewichte mehrerer Knoten
- Frage
 - in welchem Maße sind die verschiedenen Knoten an dem Fehler beteiligt?
- Anpassung der Gewichte
 - durch Gradientenverfahren, das den Gesamtfehler minimiert
 - Gesamtfehler*: Summe der (quadratischen) Abweichungen des tatsächlichen Outputs y vom gewünschten Output t für die Menge der Inputvektoren
 - Voraussetzung: Output y *stetige* Funktion der Aktivierung a

118

Algorithmus Backpropagation

für jedes Paar (v, t) // $v = \text{Input}, t = \text{gewünschter Output}$

„*forward pass*“:

Bestimme den tatsächlichen Output y für Eingabe v ;

„*backpropagation*“:

Bestimme den Fehler $(t - y)$ der Output-Einheiten
und passe die Gewichte der Output-Einheiten in die
Richtung an, die den Fehler minimiert;

Solange der Input-Layer nicht erreicht ist:

Propagiere den Fehler auf die nächste Schicht
und passe auch dort die Gewichte der
Einheiten in fehlerminimierender Weise an;

119

Design der Netztopologie

- Bestimmung von
 - Anzahl der Input-Knoten
 - Anzahl der inneren Schichten und jeweilige Anzahl der Knoten
 - Anzahl der Output-Knotenstarker Einfluß auf die Klassifikationsgüte
- zu wenige Knoten
 - ➡ niedrige Klassifikationsgüte
- zu viele Knoten
 - ➡ Overfitting

120

Bestimmung der Netztopologie

nach [SPSS Clementine 2000]

- Statische Topologie
 - Topologie wird apriori festgelegt
 - eine verborgene Schicht reicht in vielen Anwendungen aus
- Dynamische Topologie
 - dynamisches Hinzufügen von Neuronen (und verborgenen Schichten)
 - solange Klassifikationsgüte signifikant verbessert wird
- Multiple Topologien
 - Trainieren mehrerer dynamischer Netze parallel
 - z.B. je ein Netz mit 1, 2 und 3 verborgenen Schichten

121

Bestimmung der Netztopologie

- Pruning

Trainieren eines Netzes mit statischer Topologie
nachträgliches Entfernen der unwichtigsten Neuronen
solange Klassifikationsgüte verbessert wird

Schlußfolgerung

- statische Topologie: niedrige Klassifikationsgüte, aber relativ schnell.
- Pruning: beste Klassifikationsgüte, aber sehr hoher Laufzeitaufwand zum Training.

122

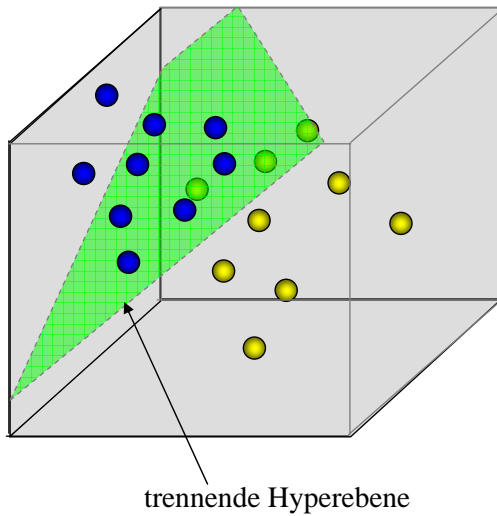
Diskussion

- + im Allgemeinen sehr hohe Klassifikationsgüte
beliebig komplexe Entscheidungsflächen
- + robust gegen Rauschen in den Trainingsdaten
- + Effizienz der Anwendung
- schlechte Verständlichkeit
(lernt nur Gewichte, aber keine Klassenbeschreibung)
- Ineffizienz des Lernens (sehr lange Trainingszeiten)
- keine Integration von Hintergrundwissen

123

3.5. Support Vector Machines

Motivation: Lineare Separation



- Vektoren in \mathcal{R}^d repräsentieren Objekte.
- Objekte gehören zu genau einer von je 2 Klassen

Klassifikation durch lineare Separation:

- Suche Hyperebene, die beide Klassen „maximal stabil“ voneinander trennt.
- ordne unbekannte Elemente der Seite der Ebene zu, auf der sie sich befinden.

124

Support Vector Machines

Probleme bei linearer Separation:

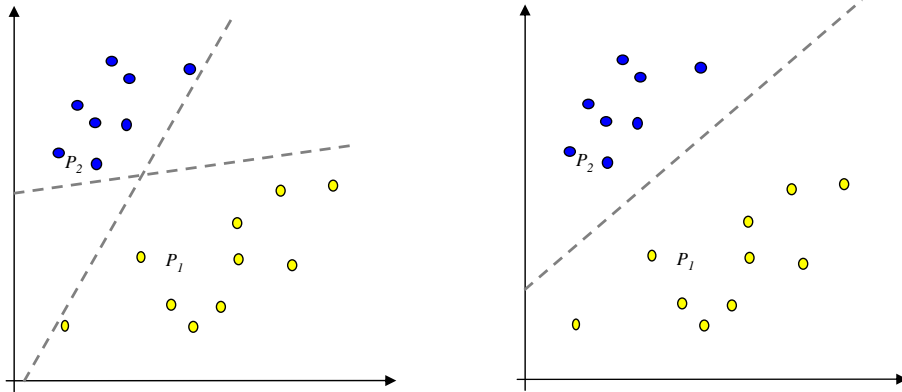
- Was ist die „maximal stabile“ Hyperebene und wie berechnet man Sie effizient?
- Klassen nicht immer linear trennbar.
- Berechnung von Hyperebenen nach Auswahl sehr aufwendig.
- Einschränkung auf 2 Klassen.
- ...

⇒ *Lösungen dieser Probleme mit Support Vector Machines (SVMs) [Vapnik 1979 u. 1995].*

125

Maximum Margin Hyperplane

Problem: Hyperebene die P_1 und P_2 trennt ist nicht eindeutig.
 \Rightarrow Welche Hyperebene ist für die Separation die Beste ?

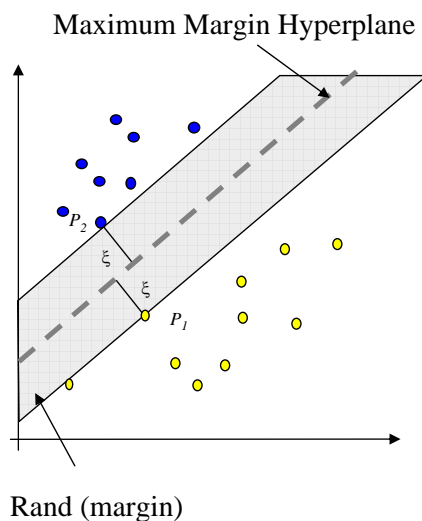


Kriterien:

- Stabilität beim Einfügen
- Abstand zu den Objekten beider Klassen

Maximum Margin Hyperplane

Lineare Separation mit der „Maximum Margin Hyperplane“



- Abstand zu Punkten aus beiden Mengen ist maximal, d.h. mind. ξ .
- Wahrscheinlichkeit, dass beim Einfügen die trennende Hyperebene verschoben werden muss, ist minimal.
- generalisiert am besten.

\Rightarrow Maximum Margin Hyperplane (MMH) ist „maximal stabil“

MMH ist nur von Punkten P_i abhängig, die Abstand ξ zur Ebene aufweisen.

$\Rightarrow P_i$ heißt Support Vector

Maximum Margin Hyperplane

Zusammenfassung der Schreibweisen der benötigten algebraischen Konstrukte für Featurespace FS :

- Skalarprodukt zweier Vektoren: $\langle \vec{x}, \vec{y} \rangle, \vec{x}, \vec{y} \in FS$

$$\text{z.B. } \langle \vec{x}, \vec{y} \rangle = \sum_{i=1}^d (x_i \cdot y_i) \quad \text{kanonisches Skalarprodukt}$$

- Beschreibung einer Hyperebene: $H(\vec{w}, b) = \left\{ \vec{x} \in FS \mid 0 = \langle \vec{w}, \vec{x} \rangle + b \right\}$

- Abstand eines Vectors zur Ebene: $\text{dist}(\vec{x}, H(\vec{w}, b)) = \left| \frac{1}{\sqrt{\langle \vec{w}, \vec{w} \rangle}} \langle \vec{w}, \vec{x} \rangle + b \right|$

128

Maximum Margin Hyperplane

Berechnung der Maximum Margin Hyperplane

1. Bedingung: kein Klassifikationsfehler (Klasse 1: $y_i=1$, Klasse 2: $y_i=-1$)

$$\left. \begin{array}{l} (y_i = -1) \Rightarrow [\langle \vec{w}, \vec{x}_i \rangle + b] < 0 \\ (y_i = 1) \Rightarrow [\langle \vec{w}, \vec{x}_i \rangle + b] > 0 \end{array} \right\} \Leftrightarrow y_i [\langle \vec{w}, \vec{x}_i \rangle + b] > 0$$

2. Bedingung: Maximaler Rand (Margin)

$$\text{maximiere: } \xi = \min_{x_i \in TR} \left| \frac{1}{\sqrt{\langle \vec{w}, \vec{w} \rangle}} (\langle \vec{w}, \vec{x}_i \rangle + b) \right| \quad (\text{Abstand von } x_i \text{ zur Ebene } H(\vec{w}, b))$$

oder

$$\text{maximiere: } \xi, \text{ so dass } \left[y_i \left(\frac{1}{\sqrt{\langle \vec{w}, \vec{w} \rangle}} [\langle \vec{w}, \vec{x}_i \rangle + b] \right) \geq \xi \right] \quad \text{für } \forall i \in [1..n]$$

129

Maximum Margin Hyperplane

maximiere ξ in $\left[y_i \left(\frac{1}{\sqrt{\langle \vec{w}, \vec{w} \rangle}} \left[\langle \vec{w}, \vec{x}_i \rangle + b \right] \right) \geq \xi \right]$, für $\forall i \in [1..n]$

Setze $\frac{1}{\sqrt{\langle \vec{w}, \vec{w} \rangle}} = \xi : \max. \frac{1}{\sqrt{\langle \vec{w}, \vec{w} \rangle}}$, mit $\left(y_i \cdot \left(\xi \cdot \left(\langle \vec{w}, \vec{x}_i \rangle + b \right) \right) \geq \xi \right) \quad \forall i \in [1..n]$

$\Rightarrow \max. \frac{1}{\sqrt{\langle \vec{w}, \vec{w} \rangle}}$, mit $\left(y_i \left(\langle \vec{w}, \vec{x}_i \rangle + b \right) \geq 1 \right) \quad \forall i \in [1..n]$

Statt $\frac{1}{\sqrt{\langle \vec{w}, \vec{w} \rangle}}$ invertiere, quadriere und minimiere das Ergebnis:

Primäres OP: minimiere $J(\vec{w}, b) = \langle \vec{w}, \vec{w} \rangle$

unter Nebenbedingung für $\forall i \in [1..n]$ sei $\left(y_i \left(\langle \vec{w}, \vec{x}_i \rangle + b \right) \geq 1 \right)$

130

Maximum Margin Hyperplane

Zur Berechnung wird das primäre Optimierungsproblem in ein duales OP überführt (Umformulierung in Form mit Lagrange Multiplikatoren).

Duales OP: maximiere $L(\vec{\alpha}) = \left(\sum_{i=1}^n \alpha_i \right) - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \cdot \alpha_j \cdot y_i \cdot y_j \cdot \langle \vec{x}_i, \vec{x}_j \rangle$

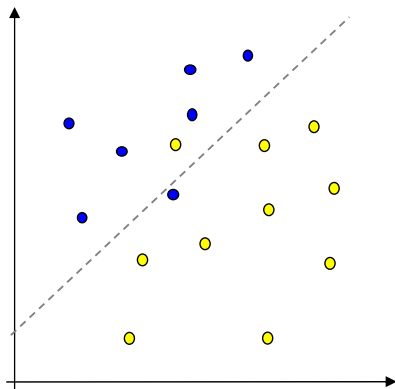
unter Bedingung $\sum_{i=1}^n \alpha_i \cdot y_i = 0$, $0 \leq \alpha_i$ und $\vec{\alpha} \in \mathfrak{R}^n$

- \Rightarrow Lösung des Problems mit Algorithmen aus der Optimierungstheorie
- \Rightarrow bis jetzt nur linear separierbarer Fall: Soft Margin Optimierung
- \Rightarrow Einführung von Kernelfunktionen zur Steigerung der Kapazität

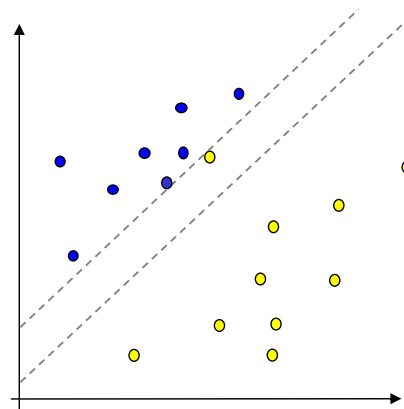
131

Soft Margin

Behandlung nicht linear trennbarer Daten: *Soft Margin Optimierung*



Daten nicht separierbar

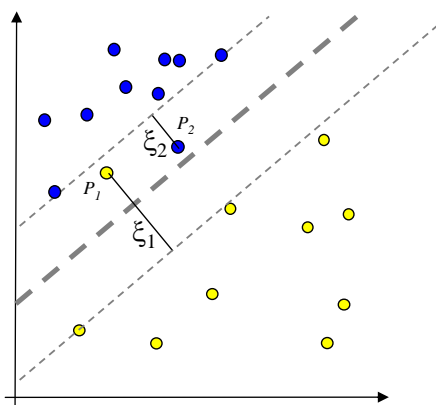


vollständige Separation ist nicht optimal

⇒ Trade-Off zwischen Trainingsfehler und Breite des Randes

Soft Margin

Betrachte beim Optimieren zusätzlich noch die Anzahl der Trainingsfehler.



- ξ_i ist der Abstand von P_i zum Rand (wird auch *Slack-Variable* genannt)
- C reguliert den Einfluss eines einzelnen Trainingsvektors

Primäres OP : minimiere $J(\vec{w}, b, \xi) = \frac{1}{2} \langle \vec{w}, \vec{w} \rangle + C \cdot \sum_{i=1}^n \xi_i$
 unter Nebenbedingung für $\forall i \in [1..n]$ sei $y_i \langle \vec{w}, \vec{x}_i \rangle + b \geq 1 - \xi_i$ und $\xi_i \geq 0$

⇒ Primäres Optimierungsproblem unter weichen Grenzen (Soft Margin)

Soft Margin

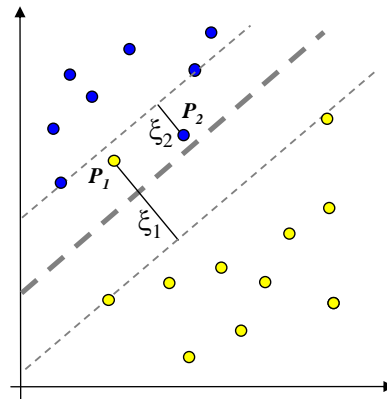
Das duale OP mit Langrange Multiplikatoren verändert sich wie folgt:

Duales OP: maximiere $L(\vec{\alpha}) = \left(\sum_{i=1}^n \alpha_i \right) - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \cdot \alpha_j \cdot y_i \cdot y_j \cdot \langle \vec{x}_i, \vec{x}_j \rangle$
 mit Bedingung $\sum_{i=1}^n \alpha_i \cdot y_i = 0$ und $0 \leq \alpha_i \leq C$

- $0 < \alpha_i < C \Leftrightarrow$ Support Vektor mit $\xi_i = 0$
- $\alpha_i = C \Leftrightarrow$ Support Vektor mit $\xi_i > 0$
- $\alpha_i = 0$ sonst

Entscheidungsregel:

$$h(\vec{x}) = \text{sign} \left(\sum_{x_i \in \text{SV}} \alpha_i \cdot y_i \langle \vec{x}_i, \vec{x} \rangle + b \right)$$

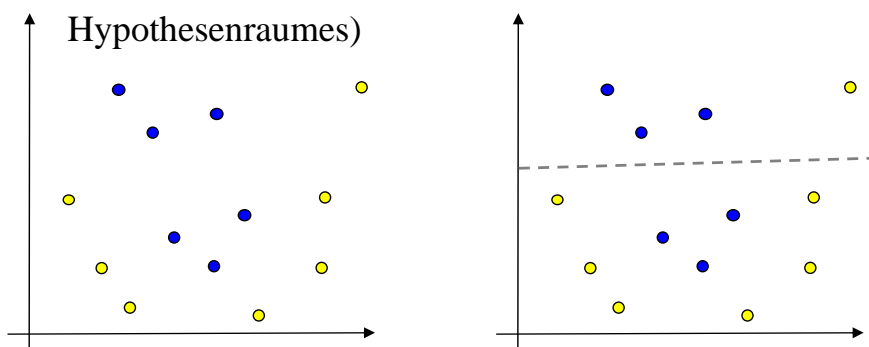


Kernel Machines

Lernen bei nicht linear trennbaren Datenmengen

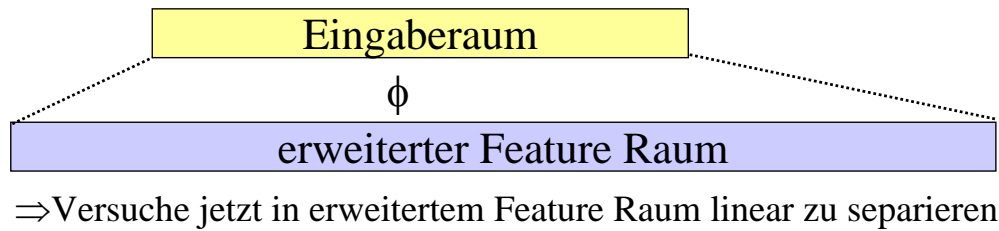
Problem: Bei realen Problemen ist häufig keine lineare Separation mit hoher Klassifikationsgenauigkeit mehr möglich.

Idee: Transformiere Daten in einen nicht linearen Feature-Raum und versuche sie im neuen Raum linear zu separieren. (Erweiterung des

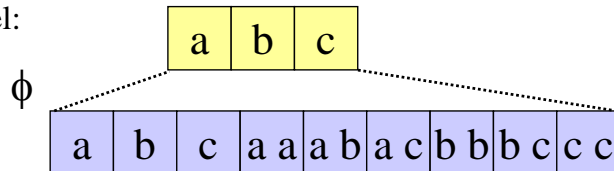


Beispiel: quadratische Transformation

Erweiterung der Hypothesenraumes



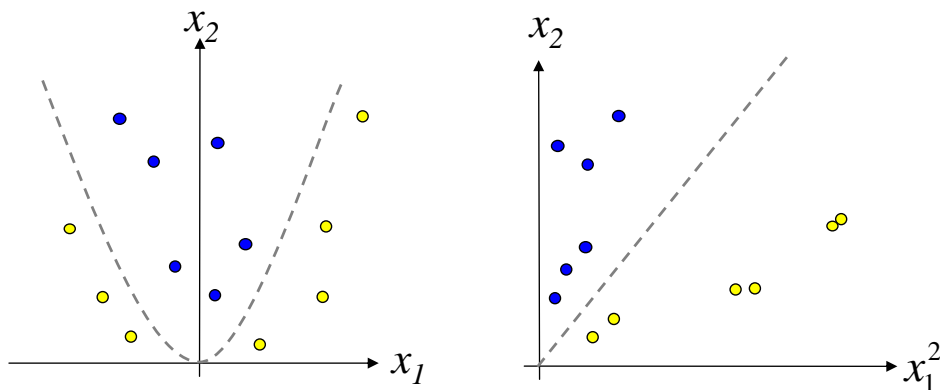
Beispiel:



hier: Eine Hyperebene im erweiterten Feature Raum ist ein Polynom 2. Grades im Eingaberaum.

Eingaberaum: $\vec{x} = (x_1, x_2)$ (2 Attribute)

im erweiterten Raum: $\phi(\vec{x}) = (x_1^2, x_2^2, \sqrt{2} \cdot x_1, \sqrt{2} \cdot x_2, \sqrt{2} \cdot x_1 \cdot x_2, 1)$
(6 Attribute)



Kernel Machines

Einführung eines Kernels \Leftrightarrow (Implizite) Featuretransformation mittels

$$\phi(\vec{x}): FS_{alt} \longrightarrow FS_{neu}$$

Duales OP: maximiere

$$L(\vec{\alpha}) = \left(\sum_{i=1}^n \alpha_i \right) - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \cdot \alpha_j \cdot y_i \cdot y_j \cdot \langle \phi(\vec{x}_i), \phi(\vec{x}_j) \rangle$$

mit Bedingung $\sum_{i=1}^n \alpha_i \cdot y_i = 0$ und $0 \leq \alpha_i \leq C$

Zusätzliche Featuretransformation wirkt sich nur auf das Skalarprodukt der Trainingsvektoren aus. \Rightarrow Kernel K ist eine Funktion mit:

$$K_{\phi}(\vec{x}_i, \vec{x}_j) = \langle \phi(\vec{x}_i), \phi(\vec{x}_j) \rangle$$

138

Kernel Machines

Wann ist eine Funktion $K(x,y)$ ein Kernel ?

Wenn die **Kernel-Matrix** (Gram Matrix) KM

$$KM(K) = \begin{pmatrix} K(\vec{x}_1, \vec{x}_1) & \dots & K(\vec{x}_1, \vec{x}_n) \\ \dots & \dots & \dots \\ K(\vec{x}_n, \vec{x}_1) & \dots & K(\vec{x}_n, \vec{x}_n) \end{pmatrix}$$

positiv (semi)definit ist, also keine negativen

Eigenwerte besitzt, dann ist $K(x,y)$ ein Kernel (siehe Mercer's Theorem)

Notwendige Bedingungen:

- $K_{\phi}(\vec{x}, \vec{y}) = \langle \phi(\vec{x}), \phi(\vec{y}) \rangle = \langle \phi(\vec{y}), \phi(\vec{x}) \rangle = K_{\phi}(\vec{y}, \vec{x})$ (Symmetrie)
- $K_{\phi}(\vec{x}, \vec{y})^2 \leq K_{\phi}(\vec{x}, \vec{x}) \cdot K_{\phi}(\vec{y}, \vec{y})$ (Cauchy-Schwarz)

Symmetrie und Cauchy-Schwarz sind keine hinreichenden Bedingungen!

139

Kernel Machines

einige Regeln zur Kombination vom Kerneln:

$$K(\vec{x}, \vec{y}) = K_1(\vec{x}, \vec{y}) \cdot K_2(\vec{x}, \vec{y})$$

$$K(\vec{x}, \vec{y}) = K_1(\vec{x}, \vec{y}) + K_2(\vec{x}, \vec{y})$$

$$K(\vec{x}, \vec{y}) = a \cdot K_1(\vec{x}, \vec{y})$$

$$K(\vec{x}, \vec{y}) = \vec{x}^T \cdot \mathbf{B} \cdot \vec{y}$$

für K_1, K_2 Kernelfunktionen, a eine positive Konstante und \mathbf{B} eine symmetrische positiv semi-definite Matrix.

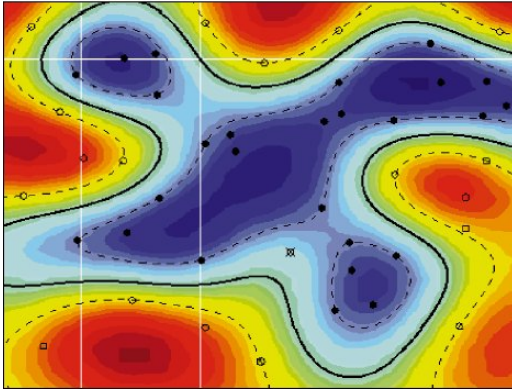
140

Kernel Machines

Beispiele für verwendete Kernel-Funktionen:

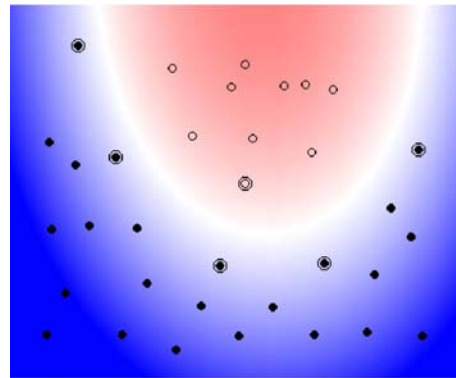
- linear: $K(\vec{x}, \vec{y}) = \langle \vec{x}, \vec{y} \rangle$
- polynomiell: $K(\vec{x}, \vec{y}) = \left(\langle \vec{x}, \vec{y} \rangle + c \right)^d$
- Radiale Basisfunktionen: $K(\vec{x}, \vec{y}) = \exp\left(-\gamma \cdot \|\vec{x} - \vec{y}\|^2\right)$
- Gauss Kernel: $K(\vec{x}, \vec{y}) = \exp\left(-\frac{\|\vec{x} - \vec{y}\|^2}{2\sigma^2}\right)$
- Sigmoid: $K(\vec{x}, \vec{y}) = \tanh\left(\gamma \cdot (\vec{x} - \vec{y}) + c\right)$

141



Radial Basis Kernel

Polynomieller Kernel (Grad 2)



Training einer SVM

zu lösen ist folgendes Problem:

$$\text{Duales OP: maximiere } L(\vec{\alpha}) = \left(\sum_{i=1}^n \alpha_i \right) - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \cdot \alpha_j \cdot y_i \cdot y_j \cdot K(\vec{x}_i, \vec{x}_j)$$

mit Bedingung $\sum_{i=1}^n \alpha_i \cdot y_i = 0$ und $0 \leq \alpha_i \leq C$

oder

$$\max \begin{bmatrix} 1 \\ \dots \\ 1 \end{bmatrix}^T \cdot \begin{bmatrix} \alpha_1 \\ \dots \\ \alpha_n \end{bmatrix} - \frac{1}{2} \begin{bmatrix} \alpha_1 \\ \dots \\ \alpha_n \end{bmatrix}^T \cdot \begin{bmatrix} y_1 y_1 K(\vec{x}_1, \vec{x}_1) & \dots & y_1 y_n K(\vec{x}_1, \vec{x}_n) \\ \dots & \dots & \dots \\ y_n y_1 K(\vec{x}_n, \vec{x}_1) & \dots & y_n y_n K(\vec{x}_n, \vec{x}_n) \end{bmatrix} \cdot \begin{bmatrix} \alpha_1 \\ \dots \\ \alpha_n \end{bmatrix}$$

mit Bedingung $\sum_{i=1}^n \alpha_i \cdot y_i = 0$ und $0 \leq \alpha_i \leq C$

Training einer SVM

zur Lösung:

- Standardalgorithmen aus der Optimierungstheorie für *konvexe quadratische Programme*
- für große Trainingsmengen numerische Algorithmen notwendig
- es existieren einige Spezialalgorithmen für *SVM*-Training:
 - Chunking / Decomposition
 - Sequential Minimal Optimisation (*SMO*)

144

Multi-Class SVMs

Bisher SVMs nur anwendbar auf 2 Klassen Probleme !!

Idee: Kombination mehrere 2-Klassen SVMs zu Klassifikatoren, die beliebig viele Klassen unterscheiden können.

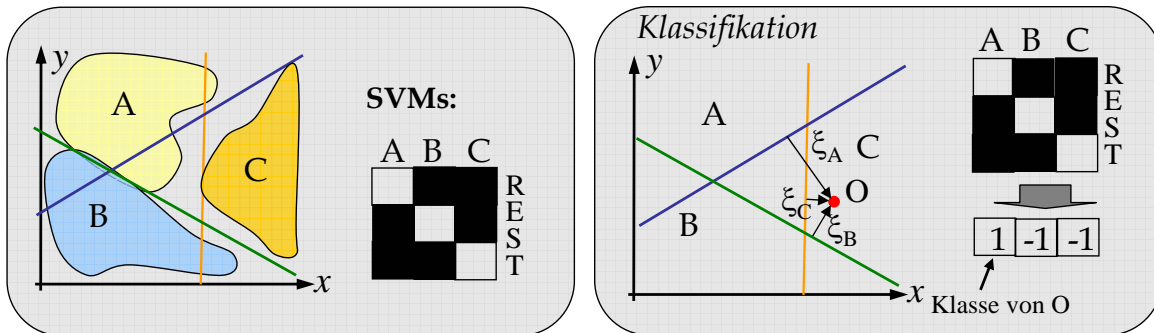
⇒ Multi-Class SVMs

2 klassische Ansätze:

1. Unterscheide jede Klasse von allen anderen (1-versus-Rest)
2. Unterscheide je 2 Klassen (1-versus-1)

145

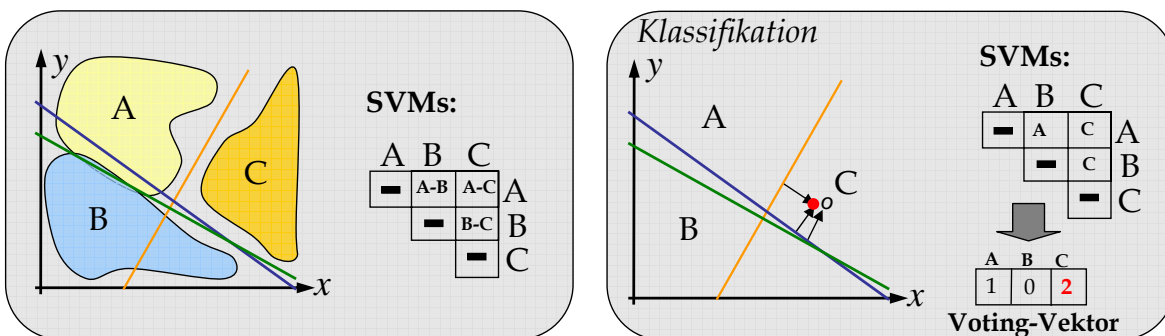
1-versus-Rest Ansatz



- 1-versus-Rest Ansatz : 1 SVM für jede Klasse.
- SVM trennt jedes Klasse von Vereinigung aller anderen Klassen ab
- Klassifiziere O mit allen Basis-SVMs.
 - ⇒ **Multiple Klassenzugehörigkeit möglich (Multi-Classification)** oder
 - Entscheidung für die Klasse, bei der Abstand ξ_i am größten ist.**

146

1-versus-1 Ansatz



- 1-versus-1 Ansatz : 1 SVM für jedes Paar von Klassen.
- Klassifikation von Objekt O:
 1. Klassifiziere O mit allen Basis-SVMs.
 2. Zähle "Stimmen"(Votes) für jede Klasse.
- Maximale Anzahl an Votes => Ergebnis.

147

Vergleich 1-versus-rest und 1-versus-1

Kriterium	1-versus-rest	1-versus-1
Aufwand Training	linear zur Anzahl der Klassen ($O(/K/)$)	Quadratisch zur Anzahl der Klassen ($O(/K^2/)$)
Aufwand Klassifikation	linear zur Anzahl der Klassen ($O(/K/)$)	Quadratisch zur Anzahl der Klassen ($O(/K^2/)$) Verbesserung: „Decision Directed Acyclic Graphs“ Klassifikation in $O(/K/)$ [Platt,Christianini 1999]
Genauigkeit	tendenziell schlechter	tendenziell höher, (nutzt wissen über unterschiedliche Klassen besser aus)

148

Anwendungen von SVM

SVM zur Textklassifikation [Joachims98]

2 Datensätze

- **Reutersdatensatz:** Nachrichtenagenturtexte
9603 Trainingsdokumente, 3299 Testdokumente, 90 Kategorien
- **Ohsumed corpus:** Medizinische Texte
jeweils 10.000 Test- und Trainingsdokumente, 23 Kategorien
(unterschiedliche Krankheiten)

Experimente durch Vergleich zwischen:

- *Naive Bayes*
- *C4.5*(Entscheidungsbaum)
- *Rocchio* (Relevance Feedback)
- *k-NN* Klassifikator
- *SVM* (polynomieller und „radial basis function“-Kernel)

149

Anwendungen von SVM

Ergebnisse:

	Naive Bayes	Rocchio	C4.5	k-NN	SVM (poly.)	SVM (rbf)
durchschn. Genauigkeit	72.0	79.9	79.4	82.3	86.0	86.4
max. pro Kl.	95.9	96.1	96.1	97.3	98.5	98.5

(Klassifikation mit 1 Klassifikator pro Klasse, der Zugehörigkeit prüft.)

Ergebnis:

⇒ SVM lieferten deutlich bessere Ergebnisse

⇒ *k-NN* Klassifikator bester der etablierten Methoden

⇒ Wahl des Kernel/der Kernelparameter rel. unkritisch

Bsp.: Grad des Polynoms : 84.2% (d=2) - 86.2% (d=4)

150

Anwendungen von SVM

weitere Anwendungsbeispiele:

- Bilderkennung [Pontil, Verri 98]
- Buchstabenerkennung [Boser, Guyon, Vapnik 92]
- Bioinformatik
 - Genexpressionsanalyse [Brown et al. 99]
 - Erkennen homologer Proteine [Jaakkola, Haussler 98]

151

Support Vector Machines

Diskussion

- + erzeugt Klassifikatoren mit hoher Genauigkeit
- + verhältnismäßig schwache Tendenz zu Overfitting
(Begründung durch Generalisierungstheorie)
- + effiziente Klassifikation neuer Objekte
- + kompakte Modelle

- unter Umständen lange Trainingszeiten
- aufwendige Implementierung
- gefundene Modelle schwer zu deuten

152

Support Vector Machines

Literatur:

- C. Cortes, V. Vapnik: Support-vector networks.
Machine Learning, 20:273-297, November 1995.
- C.J.C. Burges: A tutorial on support vector machines for pattern recognition.
Data Mining and Knowledge Discovery, 2(2):121-167,1998.
- T. Joachims: Text categorisation with Support Vector Machines.
in Proceedings of European Conference on Machine Learning (ECML),
1998.
- N. Cristianini, J Shawne-Taylor: An Introduction to Support Vector
Machines and other kernel-based learning methods.
Cambridge University Press 2000.

153

3.6 Hierarchische Klassifikation

Bisher: Flacher Klassenraum $C = \{C_1, \dots, C_n\}$

Beispiel: Eine Email ist Spam oder nicht .

Häufig: Hierarchischer Klassenraum

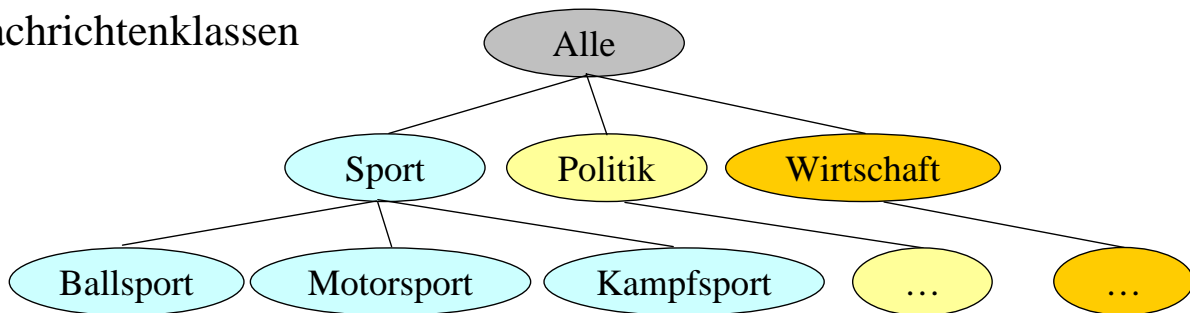
Beispiel: Nachrichten über Fußball sind ein Teil der Sportnachrichten.

⇒ Hierarchische Klassifikation berücksichtigt Beziehungen der Klassen zueinander.

154

Beispiel zur hierarchischen Klassifikation

Nachrichtenklassen



- Klassen sind in einer **Taxonomie** organisiert! (is-a Beziehungen)
- Es gilt: Gehört ein (Trainings-)Objekt o zu Klasse C_1 , dann gehört o auch zu allen Klassen C_i , die Oberklassen von C_1 sind.
 - ⇒ es reicht aus, wenn die Trainingsdokumente, ihrer speziellsten Klasse zugeordnet sind.
- Top-Down Klassifikation in der Taxonomie:
Damit ein Objekt zur Klasse gehört, muss es erstmal zur Vaterklasse gehören.
- **Achtung:** Es gibt auch andere Arten von hierarchischen Klassifikatoren, die Klassen-Ähnlichkeiten ausnützen!

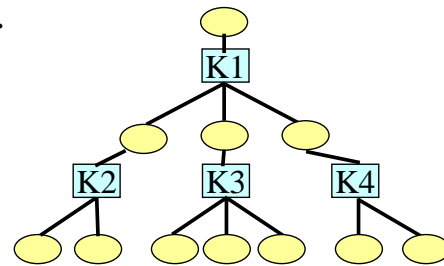
155

Aufbau von hierarchischen Klassifikatoren

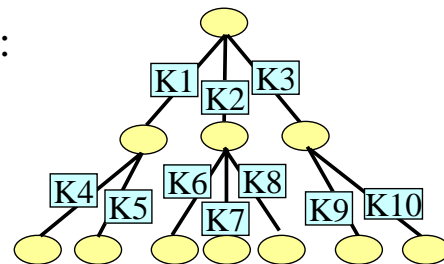
Für die Klassifikatoren K_i ist prinzipiell jedes Klassifikationsverfahren anwendbar.

Aufbau des Klassifikationssystems:

- 1 Objekt hat genau eine Klasse:
 \Rightarrow Pro inneren Knoten wird ein Klassifikator trainiert.



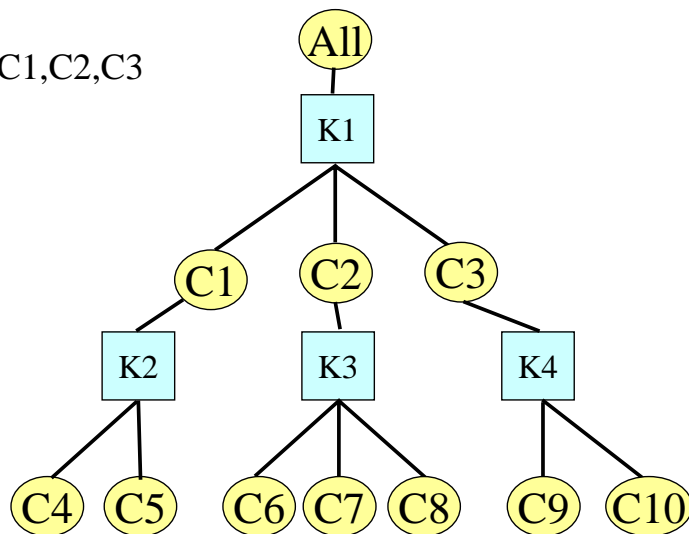
- 1 Objekt kann mehrere Klassen haben:
 \Rightarrow Pro Kante wird ein Klassifikator trainiert.



Training von hierarchischen Klassifikatoren

Top-Down Training für eindeutige Klassenzugehörigkeit:
 Trainingsmenge TR

- Trainiere K1 mit TR für Klassen C1, C2, C3
- Trainiere K2 mit $TR_{C1} = \{x \in TR \mid \text{class}(x) = C1\}$ für Klassen C4 und C5
- Trainiere K3 mit $TR_{C2} = \{x \in TR \mid \text{class}(x) = C2\}$ für Klassen C6, C7, C8
- ...



Training von hierarchischen Klassifikatoren

Top-Down Training für mehrfache Klassenzugehörigkeit:

Trainingsmenge TR, $\text{Class}(o) = \{\text{Menge der Klassen von } o\}$

- Trainiere K1 mit TR für Klassen C1, Other
wobei

$$\text{TR}_{C1} = \{x \in \text{TR} \mid C1 \in \text{Class}(x)\}$$

$$\text{TR}_{\text{other}} = \{x \in \text{TR} \mid C1 \notin \text{Class}(x)\}$$

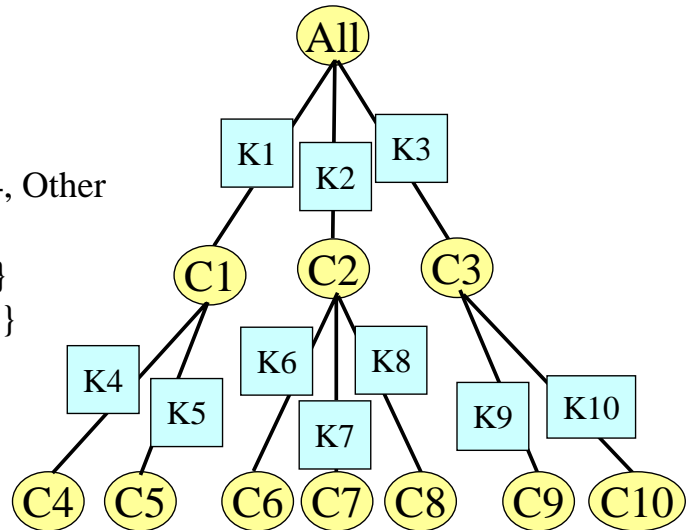
....

- Trainiere K4 mit TR für Klassen C4, Other
wobei

$$\text{TR}_{C4} = \{x \in \text{TR}_{C1} \mid C4 \in \text{Class}(x)\}$$

$$\text{TR}_{\text{other}} = \{x \in \text{TR}_{C1} \mid C4 \notin \text{Class}(x)\}$$

...



158

Klassifikation mit hierarchischen Klassifikatoren

Greedy-Ansatz:

Klassifikation von Objekt o

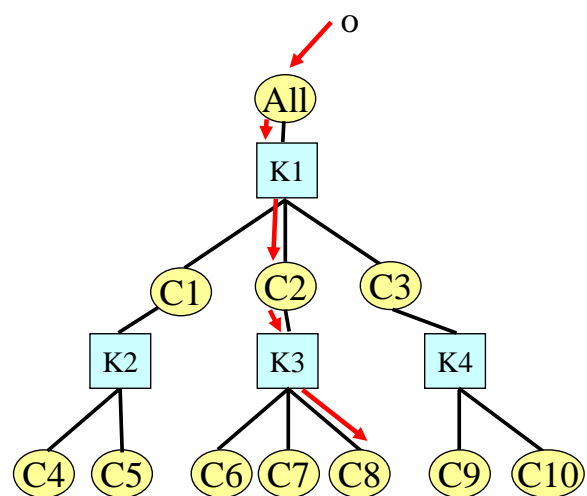
1. Klassifiziere o mit K1

=> $C_t \in \{C1, C2, C3\}$

2. Gehe zu C_t und klassifiziere o mit K_t : ...

Klassifikation entlang des Pfades auf dem die jeweils vorhergesagte Klasse liegt.

Abbruch wenn Blatt erreicht ist.



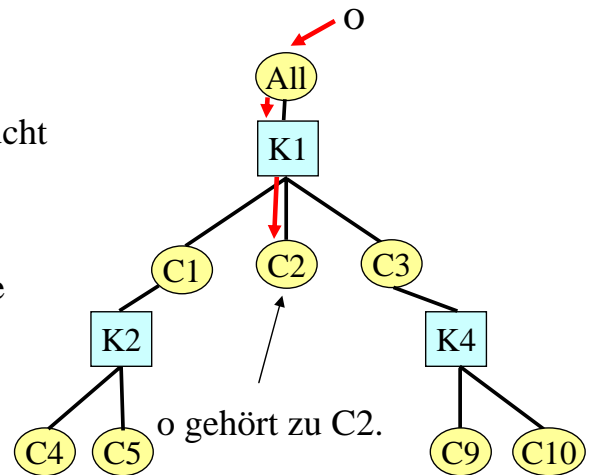
o gehört zu C8.

159

Klassifikation mit hierarchischen Klassifikatoren

Vorteile des Greedy-Ansatz:

1. Effizienz
Nur ein Pfad in der Taxonomie muß besucht werden.
2. Taxonomie beliebig
Blätter dürfen auf unterschiedlicher Höhe liegen.
3. Beliebige Klassifikatoren anwendbar.



Nachteil des Greedy-Ansatzes:

Fehler bei Vaterknoten können nicht durch korrekte Behandlung bei den Söhnen ausgeglichen werden. D.h.

Falls K1 nur 55 % Genauigkeit leistet, kann Klassifikationsgüte des gesamten Klassifikators nur schlechter werden.

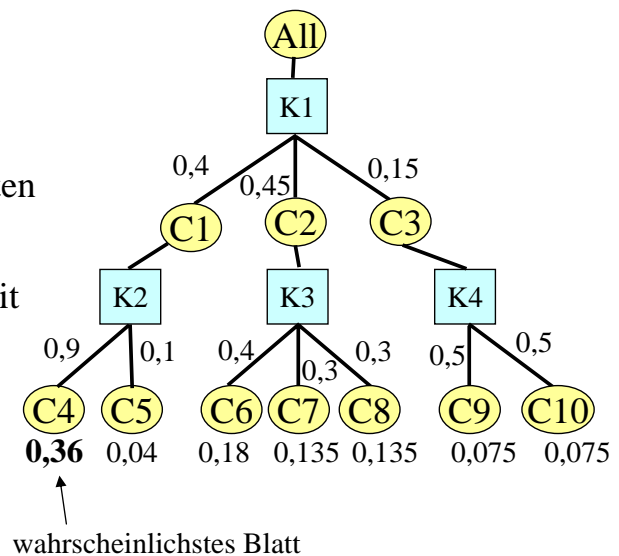
Klassifikation mit hierarchischen Klassifikatoren

Vollständige hierarchische Klassifikation

Bedingungen:

1. alle Blätter auf selber Höhe.
 2. Klassifikator liefert Wahrscheinlichkeiten /Konfidenzwerte für jede Klasse.
- ⇒ Berechne Konfidenz/Wahrscheinlichkeit jeder Blattklasse.

Bsp: $P(C4|o) = P(C1|o) \cdot P(C4|o \in C1)$
 $= 0,4 \cdot 0,9 = 0,36$



Ausgleich bei Fehlklassifikation möglich.

Nachteile: - schlechte Effizienz
 - von der Güte der berechneten Konfidenzwerte abhängig.

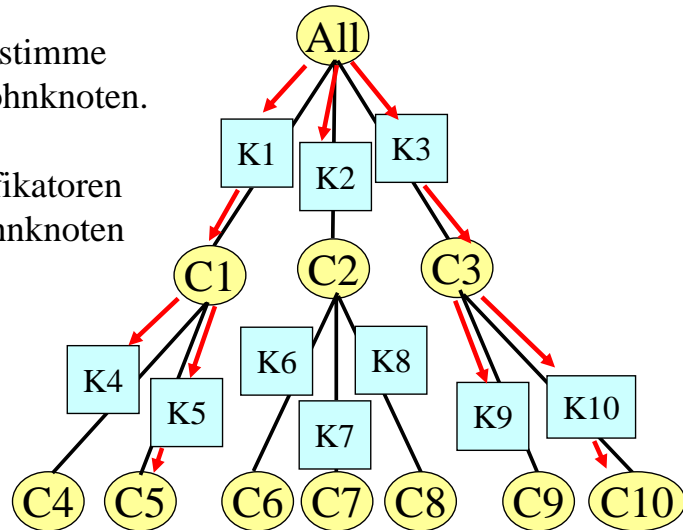
Klassifikation bei mehrfacher Klassenzugehörigkeit

Bestimme alle Klassen im Baum zu denen Objekt o gehört.

1. Klassifiziere o mit K1, K2, K3:
Falls K_i Klasse C_i vorhersagt, gehe zu Knoten C_i .
{C1, C3}
2. Für alle erreichten Knoten C_i , bestimme
Klassifikatoren auf Kanten zu Sohnknoten.
{K4, K5, K9, K10}
3. Klassifiziere o mit diesen Klassifikatoren
und bestimme alle erreichten Sohnknoten

...

- Sagt nicht nur Blätter sondern beliebige Klassen vorher !!!
- Fehler in allgemeinen Knoten können viele falsche Klassen im Ergebnis bewirken.



$\text{Class}(o) = \{C5, C10\}$

162

Diskussion hierarchische Klassifikation

Ziel: Miteinbeziehen von Taxonomien für schnellere und genauere Klassifikation.

⇒ Anwendungsbereich: Probleme mit vielen Klassen die bereits in Taxonomie organisiert sind.

(z.B. Webpages nach Yahoo-Taxonomie, Proteinklassen,..)

⇒ schnelle Klassifikation mit Greedy-Ansatz.

⇒ vollständige hierarchische Klassifikation sehr aufwendig.

⇒ Steigerung der Klassifikationsgüte hängt von Anwendung ab.

In der Regel mäßige Verbesserung der Genauigkeit im Vergleich zu flachen Klassensystemen.

⇒ Klassifikation mit mehrfachen Klassenzugehörigkeiten ordnet Objekt eine Teilmenge aller möglichen Klassen zu.

163