Syntaktische und Statistische Mustererkennung

VO 1.0 840.040
(UE 1.0 840.041)

Bernhard Jung

bernhard@jung.name
http://bernhard.jung.name/VUSSME/

Rückblick

- Entscheidungstheorie
- Bayes'sche Klassifikation
- Dichteschätzung

Naive Bayes

Naive Dichteschätzung

- Die gemeinsame Verteilung spiegelt nur die Trainingsdaten wieder
- Wir benötigen Generalisierungsfähigkeit

Naive Schätzung

- Annahme:
  - Jedes Attribut ist unabhängig von allen anderen verteilt

Unabhängigkeit

$$P(x_i = v | x_1 = u_1, x_2 = u_2, \ldots, x_{i-1} = u_{i-1}, x_{i+1} = u_{i+1}, \ldots, x_M = u_M) =$$
$$= P(x_i = v)$$

für alle $i, v, u_{1,} \ldots, u_M$

Auch geschrieben als:

$$x_i \perp \{x_{1,} x_{2,} \ldots, x_{i-1}, x_{i+1}, \ldots, x_M\}$$

A und B sind unabhängig, genau dann wenn:
P(A|B) = P(A)

Naiver Dichteschätzer

Sind x_1 ... x_M unabhängig, dann gilt:

$$P(x_1 = u_1, x_2 = u_2, \ldots, x_M = u_M) = \prod_{k=1}^{M} P(x_k = u_k)$$

Lernen

$$\hat{P}(x_i = u_i) = \frac{\text{Anzahl Records mit } x_i = u_i}{\text{Gesamtanzahl an Records}}$$

Naive Bayes Classifier

$Y^{predict}$ = argmax $P(X_1 = u_1, \ldots, X_m = u_m | Y = v) P(Y = v)$

= argmax $P(Y = v) \prod_{i=1}^{m} P(X_i = u_i | Y = v)$

Um numerische Instabilität zu vermeiden:

$Y^{predict}$ = argmax $\log P(Y = v) + \sum_{i=1}^{m} \log P(X_i = u_i | Y = v)$

Naive Bayes Classifier Eigenschaften

- Verschiedene Dichteschätzer können integriert werden
- Dichteschätzung mit reellen Attributen möglich
- Bayessche Klassifizierer sind nicht maximal diskriminativ
- Null Wahrscheinlichkeiten sind ein Problem für Naive und Gemeinsame Verteilungen (Dirichlet Prior)
- Performanter Algorithmus: Naiver Bayes mit 10,000 Attributen möglich!

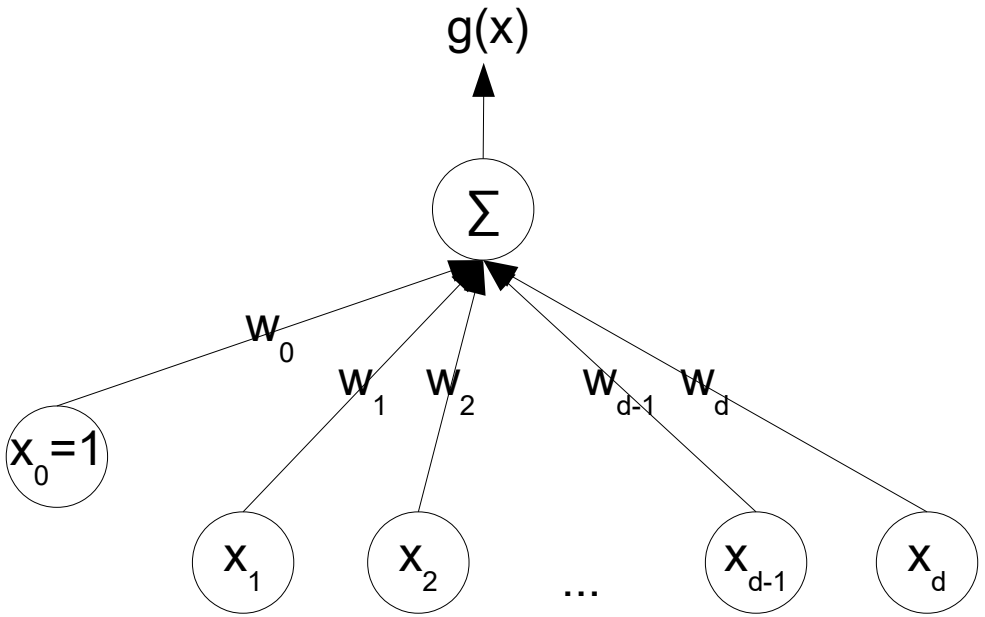
Lineare Klassifikation

Lineare Klassifikation

Lineare Diskriminantenfunktion

$$g(x) = w^\top x + w_0$$

Gewichtsvektor w, Bias/Schwellwert w_0



nicht-lineare Beispiele für g(x)

- Bayesscher Klassifizierer
  g(x) = p(w1 | x) – p(w2 | x)
- Log-likelihood

$$g(x) = \ln \frac{p(x|w_1)}{p(x|w_2)} + \ln \frac{p(w_1)}{p(w_2)}$$

vgl.

$p(x \mid w_1)\ p(w_1) > p(x \mid w_2)\ p(w_2)$

$$\frac{p(x \mid w_1)}{p(x \mid w_2)} > \frac{p(w_2)}{p(w_1)}$$

Polynomische Diskriminanzfunktionen

- Statt (linear)

  $g(x) = a_0 + a_1 * x_1 + ... + a_n * x_n = a^t * y$

- Polynomische Diskriminanzfunktion r-ten Grades

  $g(x) = a_0 + a_1 * f_1(x) + ... + a_m * f_m(x)$

  mit $f_j(x) = x_1^{k1} * x_2^{k2} * ... * x_n^{kn}$ $(\sum k_i = r)$

  - Für N=2 und r = 2 (quadratisch)

    $f_0(x) = x_1^0 * x_2^0 = 1$

    $f_1(x) = x_1^1 * x_2^0 = x_1$

    $f_2(x) = x_1^0 * x_2^1 = x_2$

    $f_3(x) = x_1^1 * x_2^1 = x_1 * x_2$

    $f_4(x) = x_1^2 * x_2^0 = x_1^2$

    $f_5(x) = x_1^0 * x_2^2 = x_2^2$

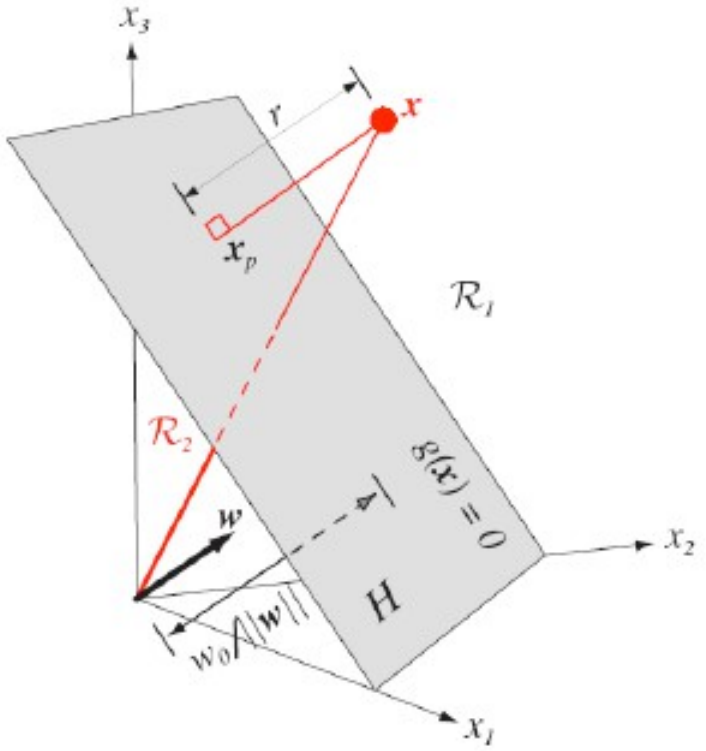
- Statt Polynomen: stückweise-lineare Funktionen

Lineare Entscheidungsgrenzen

- Entscheidungsregel: Entscheide für Klasse
  w_1 wenn $g(x) > 0$, für
  w_2 wenn $g(x) < 0$,
  keine Entscheidung wenn $g(x) = 0$
- Gleichung $g(x) = 0$ stellt die Entscheidungsfläche dar.
  Weil $g(x)$ linear ist => Entscheidungs(hyper)ebene H.
- Der Gewichtsvektor w ist der Normalvektor von H.
- H teilt den Raum in 2 Halbräume:
  - R_1 für w_1 bzw. $g(x) > 0$ (positive Seite, w zeigt in R_1)
  - R_2 für w_2 bzw. $g(x) < 0$ (negative Seite)

Abstand zur Entscheidungsebene

- Die Diskriminantenfunktion g(x) misst den Abstand zur Entscheidungsebene

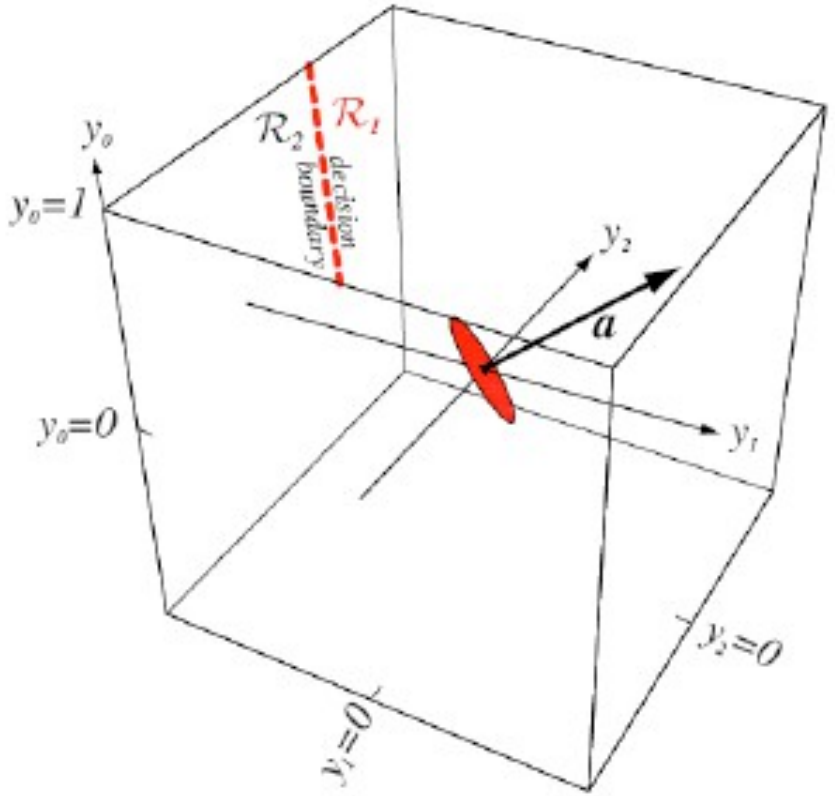


$$x = x_p + r\frac{w}{||w||}$$

$$\Rightarrow \quad r = \frac{g(x)}{||w||}$$

Erweiterte Merkmalsvektoren

- Einführung eines zusätzlichen konstanten Merkmals $x_0 = 1$ zur Vereinfachung der Notation



$$g(x) = w^\top x + w_0 = w_0 + \sum_{i=1}^{n} w_i x_i = \sum_{i=0}^{n} w_i x_i$$

$$y = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} 1 \\ x \end{bmatrix} \qquad a = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} w_0 \\ w \end{bmatrix}$$

$$g(x) = a^\top y$$

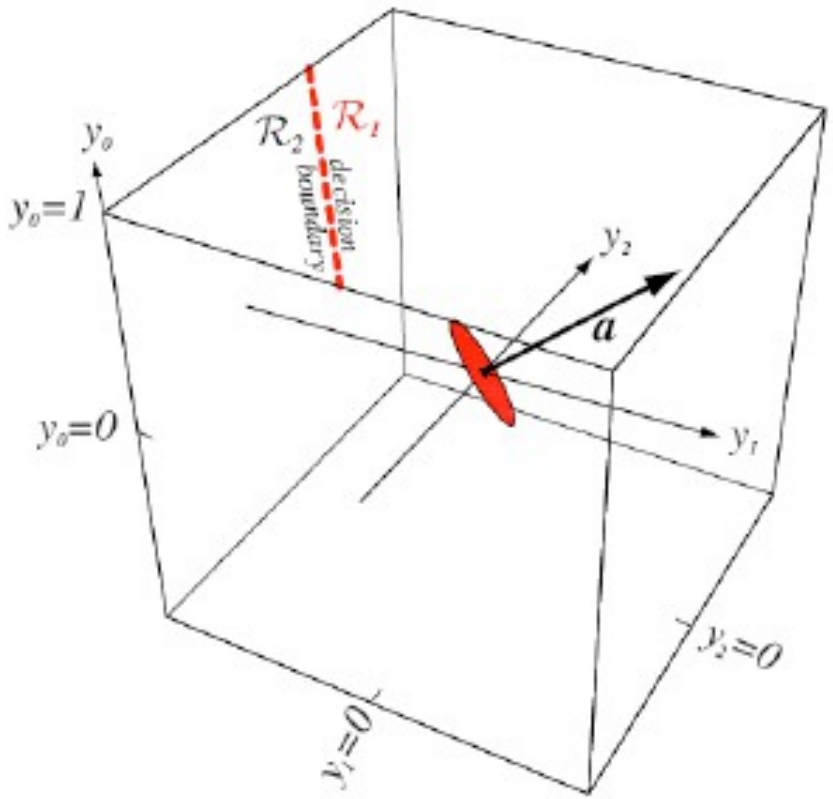
Perzeptron-Lernen

- Entscheidungsfunktion:

$$g(x) = a^\top y$$

> 0 -> Klasse 1

< 0 -> Klasse 2



Algorithmus:

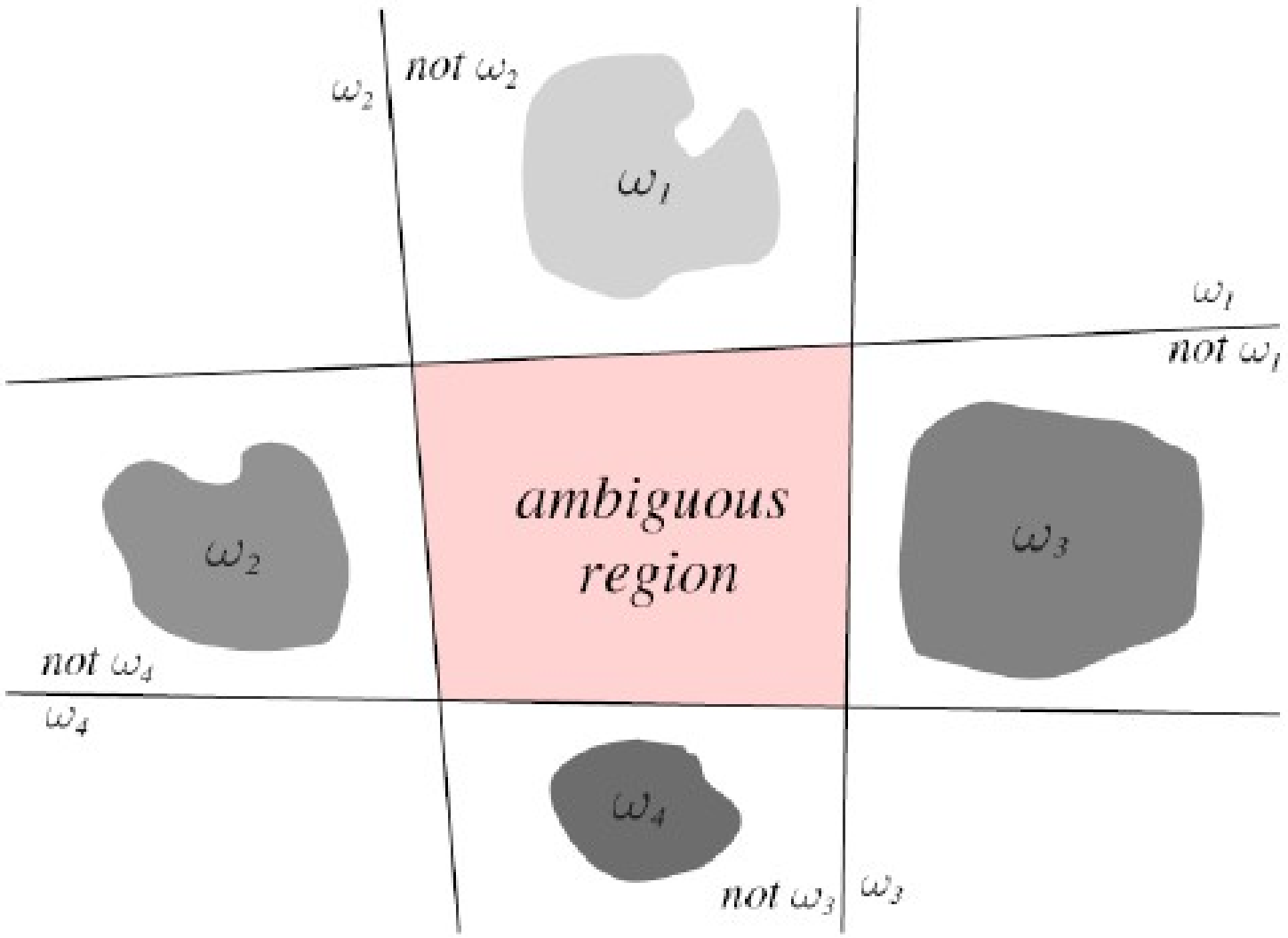
Iteriere (wiederholt) über alle Trainingsbespiele bis Verfahren konvergiert

- Berechne g(x)
- Wenn vorhergesagte Klasse falsch:
  - Anpassen (verbiegen) des Gewichtsvektors:
    $a_{neu} = a_{alt} + c * y$
  - c beliebige positive Zahl (beinflusst, wie rasch das Verfahren konvergiert)

Mehrklassenprobleme (mit binären Klassifikatoren bzw. Diskriminantenfunktionen)

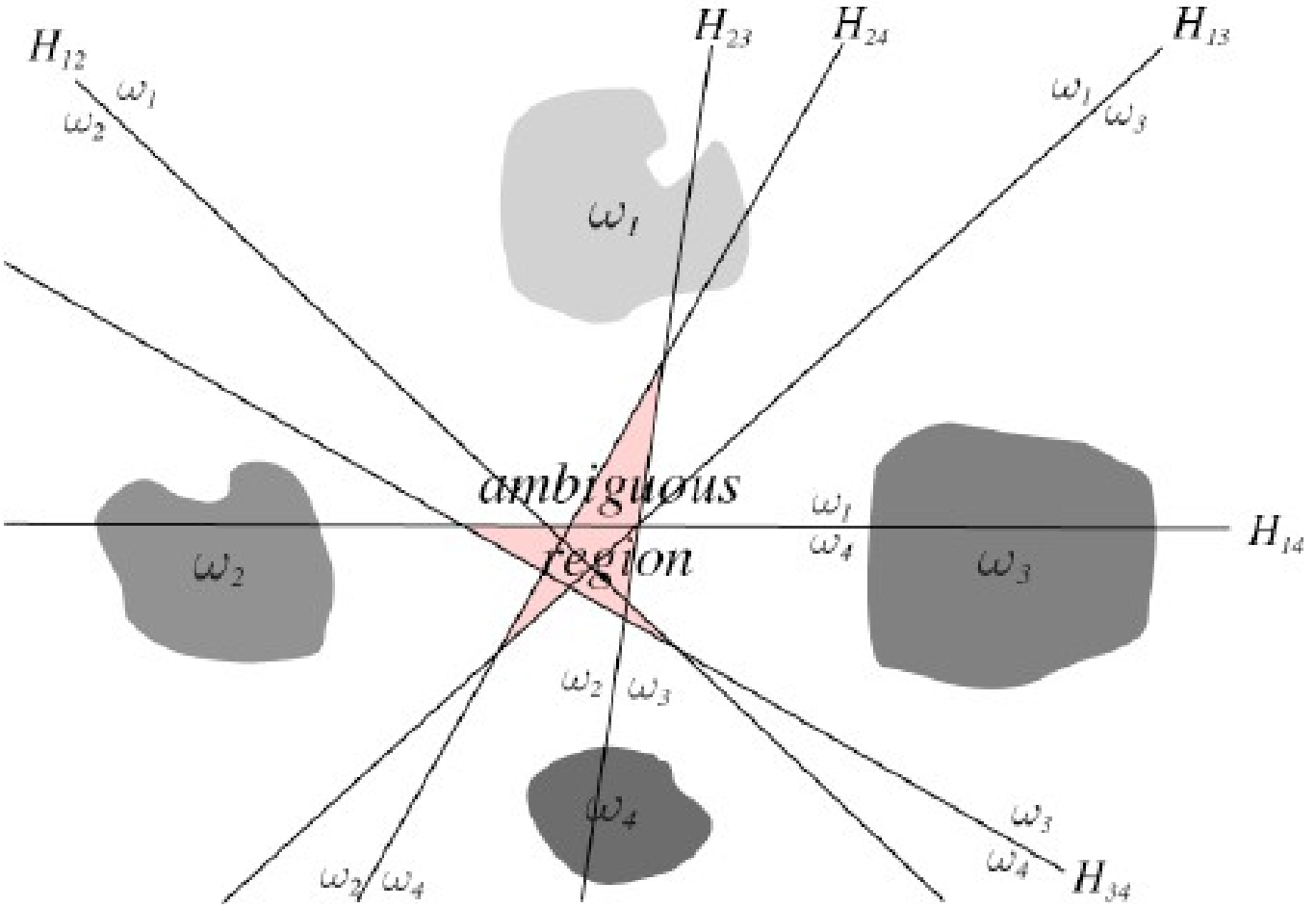
Mehrklassenprobleme

1-vs-all



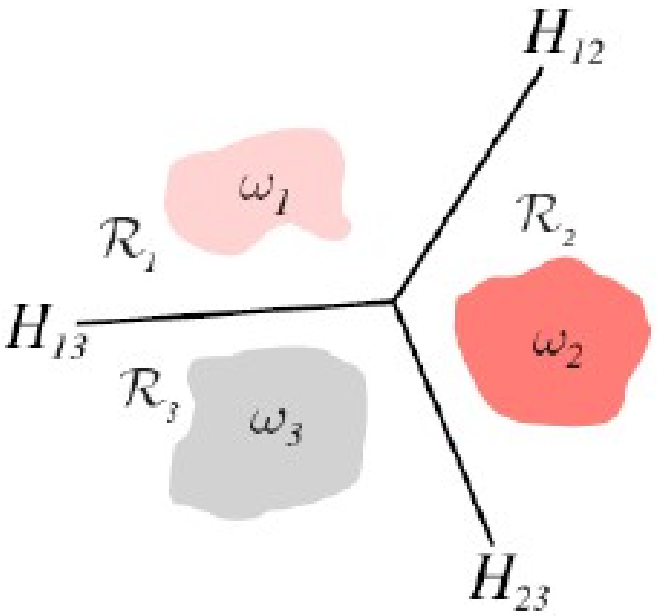
Mehrklassenprobleme

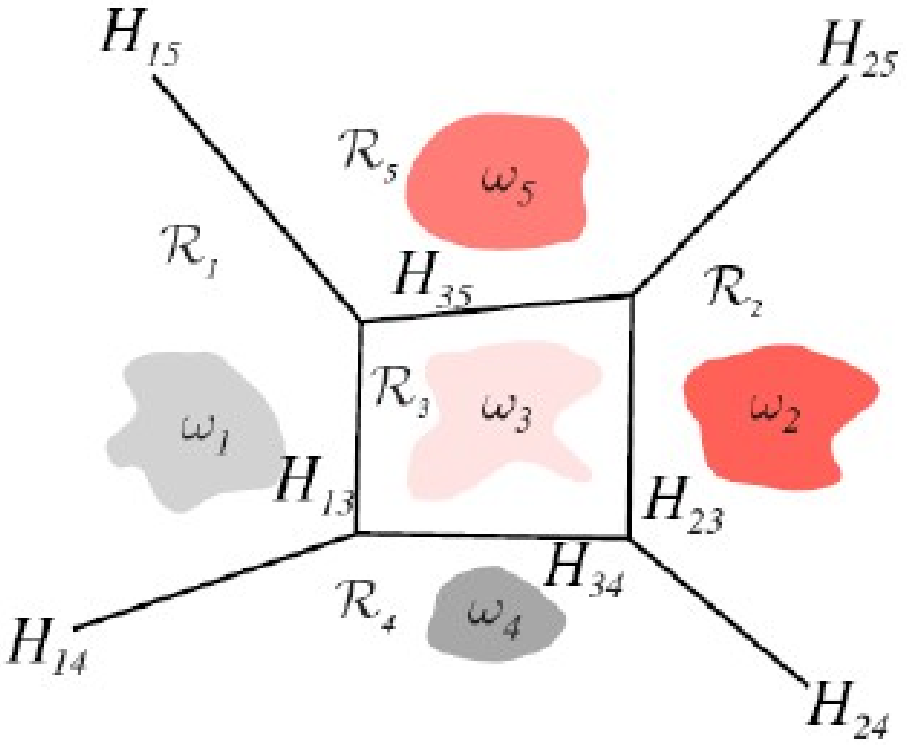
1-vs-1



Lineare Maschinen

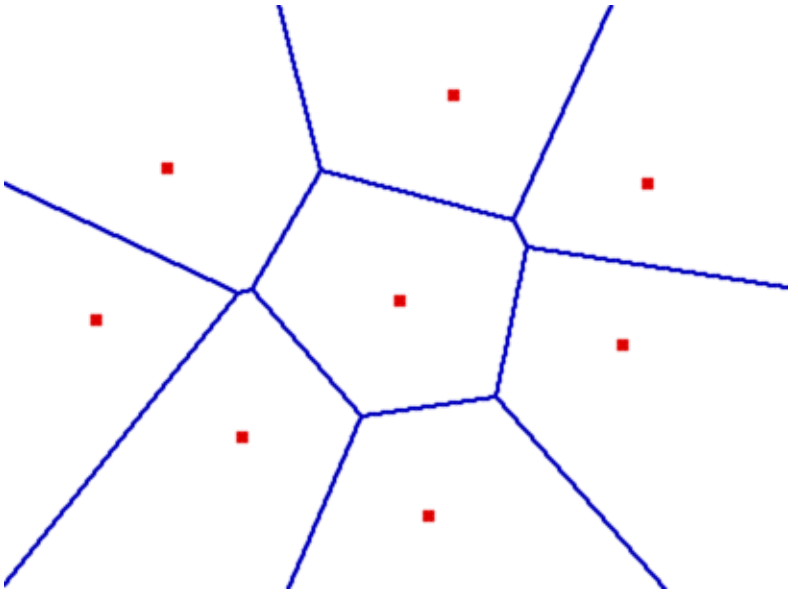
- Für c Klassen werden c lineare Diskriminantenfunktionen $g_i(x)$ verwendet
- Entscheidung für w_i wenn $g_i(x) > g_j(x)$ für alle $j \neq i$





nearest neighbour

- Prototyp-basierter Klassifikator (partitionierend)
  - Berechne $d(x,p_j)$ für zu klassifizierendes x und die Prototypen p_j aller Klassen
  - Diskriminanzfunktion: $g_j(x) = -d(x,p_j)$
  - Entscheidungsgrenzen: Mittelsenkrechte zwischen den Prototypen (Voronoi-Tesselation)



nearest neighbour

- Woher stammen die Prototypen?
  - z.B. Mittelung über Stichprobe
  - Erweiterung von einzelnem Prototypen für eine Klasse zu Prototypenmenge
    (Extremfall: jedes Trainingsbeispiel ein Prototyp)
- Man kann zeigen: für jede Metrik und (nahezu) beliebige bedingte Wahrscheinlichkeitsdichteverteilungen gilt:
  $P_{bayes} <= P_{NN} <= P_{bayes}*(2-P_{bayes}*M/(M-1))$
  - Fehlerwahrscheinlichkeit des NN-Klassifikators ist höchstens doppelt so groß wie die des optimalen Bayes'schen Klassifikators

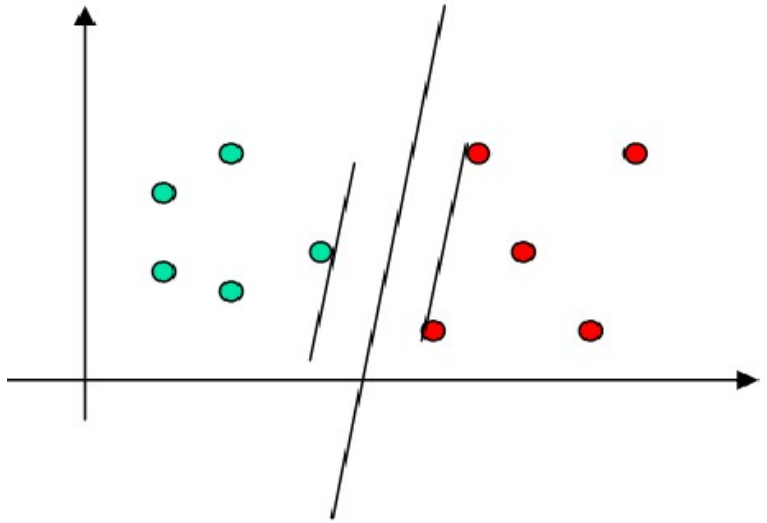
k-nearest neighbour

- Betrachtung der k nächsten Nachbarn von x
- Vorhersage des häufigsten Labels dieser k Nachbarn

- Eigenschaften von k-nearest neighbour:
  - Vorteil: Sehr einfach, kein Training
  - Nachteil: Nicht effizient bei der Klassifikation

Support Vector Machines

Idee: Trennende Hyperebene mit größtem Rand

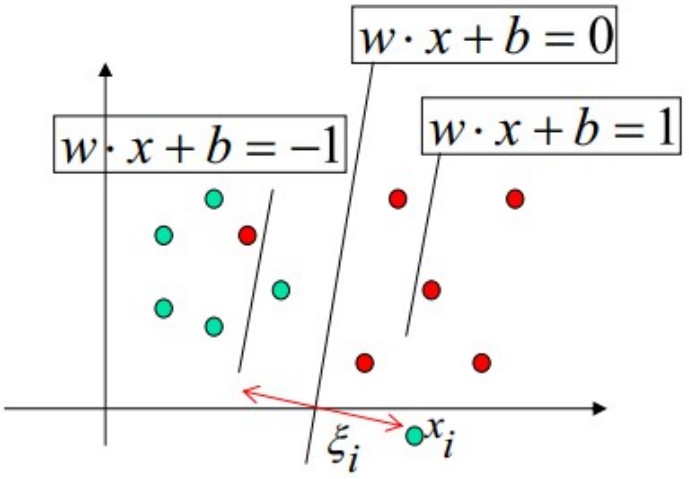


Wenn linear separierbar:

- Eindeutigkeit der Ebene, globale Lösung

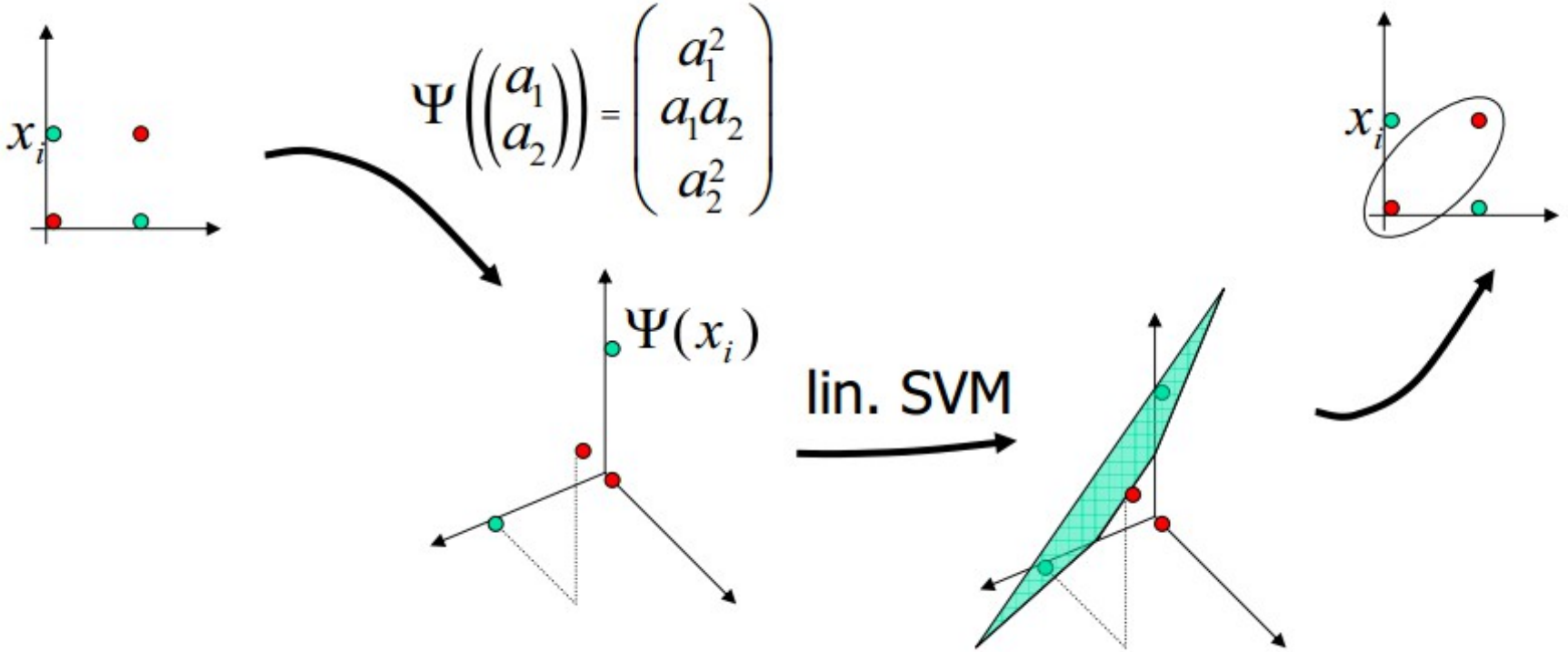
Support Vector Machines

Nicht linear separierbare Daten:



Bestrafung von Randverletzungen durch Lockerungsvariablen → wieder optimale Lösung möglich

nicht lineare SVM



nicht lineare SVM – Kernel Trick

- Problem: Berechnung von $\Psi(x_i)$
- Aber, nur Skalarprodukte: $\Psi(x_i) \cdot \Psi(x_j)$
- → Effizient berechenbare Kernelfunktion:

$$K(x_i, x_j) = \Psi(x_i) \cdot \Psi(x_j)$$

- Kernelfunktionen, z.B.

Linear	$K(x, y) = x \cdot y$
Polynomial	$K(x, y) = (x \cdot y + c)^d$
Sigmoid	$K(x, y) = \tanh(\kappa(x \cdot y) + \theta)$
Gauß	$K(x, y) = \exp\left(-\|x - y\|^2 / (2\sigma^2)\right)$

SVM - Eigenschaften

- Komplexitäten:
  - Training: Speicher $O(n^2)$, Laufzeit $O(n^3)$
  - Klassifikation: Speicher $O(n_s)$, Laufzeit $O(n_s*d)$
- Schwächen:
  - Langsames Lernen und Klassifizieren
  - Keine offensichtliche Multiklassenerweiterung
  - Keine allgemeinen Kriterien zur Wahl des Kernels
  - Fehlende Aussagen zur Klassifikationssicherheit

SVM - Eigenschaften

- Stärken:
  - Empirisch hervorragend
  - Schnelle Implementierungen verfügbar (z.b. libsvm)
  - Leichte Handhabbarkeit, wenig Parameter, keine Modellwahl, kein a-priori Wissen
  - Anschauliche Funktionsweise

Sequentielle Klassifizierung

- Motivation: Merkmalsmessung sehr teuer/schwer vorzunehmen
- Erhebe Merkmal n (von 1 bis N)
  - Quotient $\tau_n = p(x_n|\omega_1) / p(x_n|\omega_2)$
  - Vergleich mit zwei Grenzwerten A und B

    $A = (1-e_{21})/e_{12}$ und $B = e_{21}/(1-e_{12})$

    e_{kj} ... Wahrscheinlichkeit (einer Falschklassifikation), dass man für w_k obwohl w_j zutrifft

    $\tau_n >= A$ Klasse ω_1

    $\tau_n <= B$ Klasse ω_2

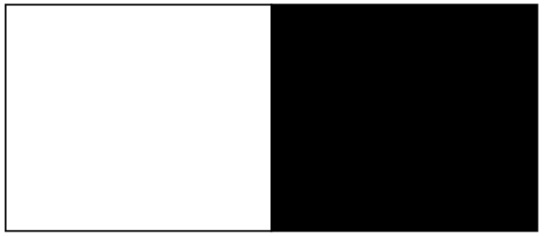
    $B < \tau_n < A$ weiteres Merkmal muss untersucht werden

Boosting

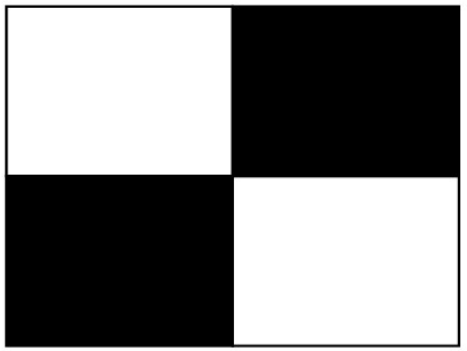
- Meta-Algorithmus zur Verbesserung der Performanz anderer Klassifikationsalgorithmen
  - Kombiniert „schwache“ Klassifizierer zu einem „starken“
- Algorithmus:
  - Lernbeispielen werden gewichtet
  - Wiederhole T mal:
    - Lerne Klassifizierer unter Berücksichtigung der Gewichte
    - Passe Gewichte basierend auf Performanz des gelernten Klassifizieres an
  - Kombiniere „schwache“ Klassifizierer
- Siehe ADABoost, LPBoost, etc.

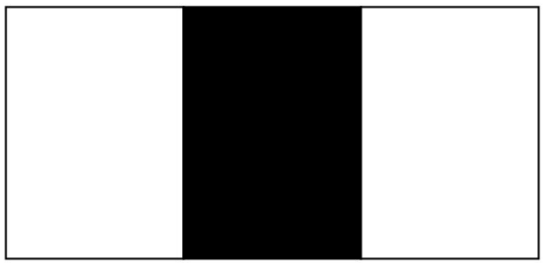
Boosted Rectangle Filters

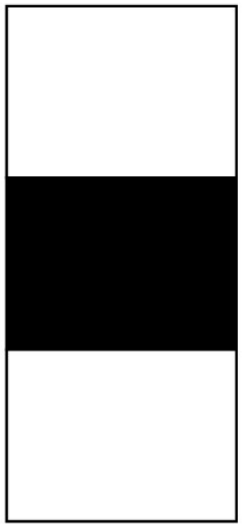
- 5 verschiedene Basisformen

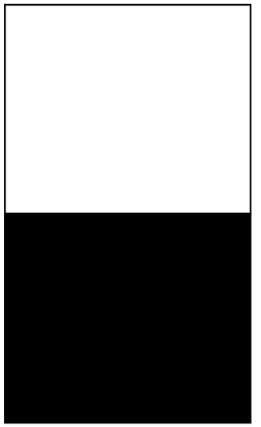


white: value = 1
black: value = -1







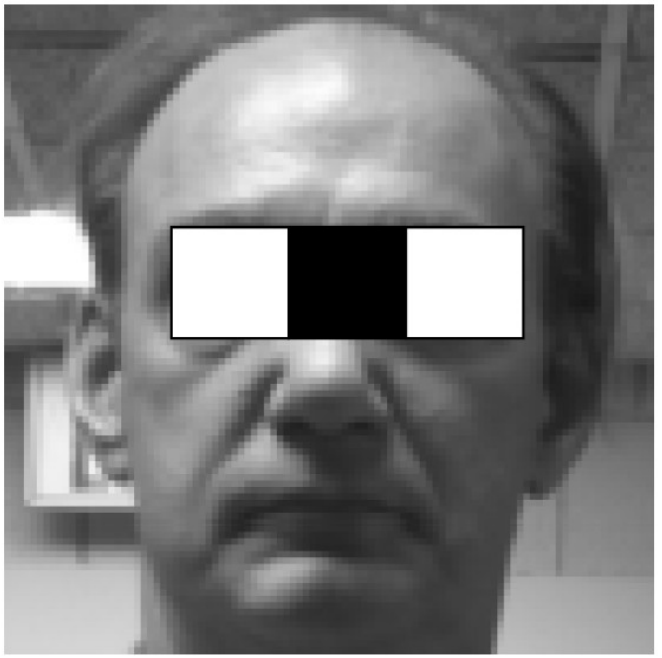


Rechtsecksfilter

- Einfach zu definieren und zu berechnen
- Zahlreich vorhanden
- Schnell und effizient zu berechnen
  - Integralbilder

Klassifikation mit Rechtsecksfilter

- Klassifikator definiert über:
  - Filtertyp
  - Rechtecksgröße
  - Fensteroffset
  - Schwellwert



- Anzahl möglicher Klassifikatoren?
  - #pixel * #thresholds

Auswahl schwacher Klassifizierer

- Zufällige Auswahl von ein paar tausend „soft“ Klassifizierern, definiert über:
  - Filtertyp
  - Rechtecksgröße
  - Fensteroffset
- Vorberechnung der Antwortfunktion der „soft classifiers“
  - Anzahl möglicher schwacher Klassifizierer: 1000 * # thresholds
- Suche nach optimalem Schwellwert für weak classifier

Boosting

- Jede Menge schwacher Klassifizierer h(pattern): besser als zufälliges Raten, Fehlerrate < 0.5
- Binäres Problem: Gesicht Ja oder Nein
- Starker Klassifikator:

  $H(p) = \alpha_1 * h_1(\text{pattern}) + \ldots + \alpha_d * h_d(\text{pattern})$

Boosting

- Aktualisierung der Gewichte:
  weights(i) = weights(i) * exp(-alpha * h1(i) * labels(i))
  - Korrekt klassifiziert, Multiplikation mit < 1
  - Falsch klassifiziert, Multiplikation mit > 1
- Normalisierung der Gewichte: weights = weights / sum(weights)

Boosting

- Auswahl des besten Klassifizierers (geringste Fehlerrate)
- Bestimmung von alpha:
  alpha = direction * 0.5 * log((1 - error) / error)
- Trainingsgewichte:
  Jedes Trainingsbeispiel hat ein Gewicht,
  zu Beginn gleichförming: 1/#samples

Boosting

- Auswahl des nächsten besten Klassifizieres unter Berücksichtigung der geänderten Gewichte
- Auswahl von alpha_i
- Anpassung der Gewichte
- … und wieder von vorne ...

Klassifikationsmethoden

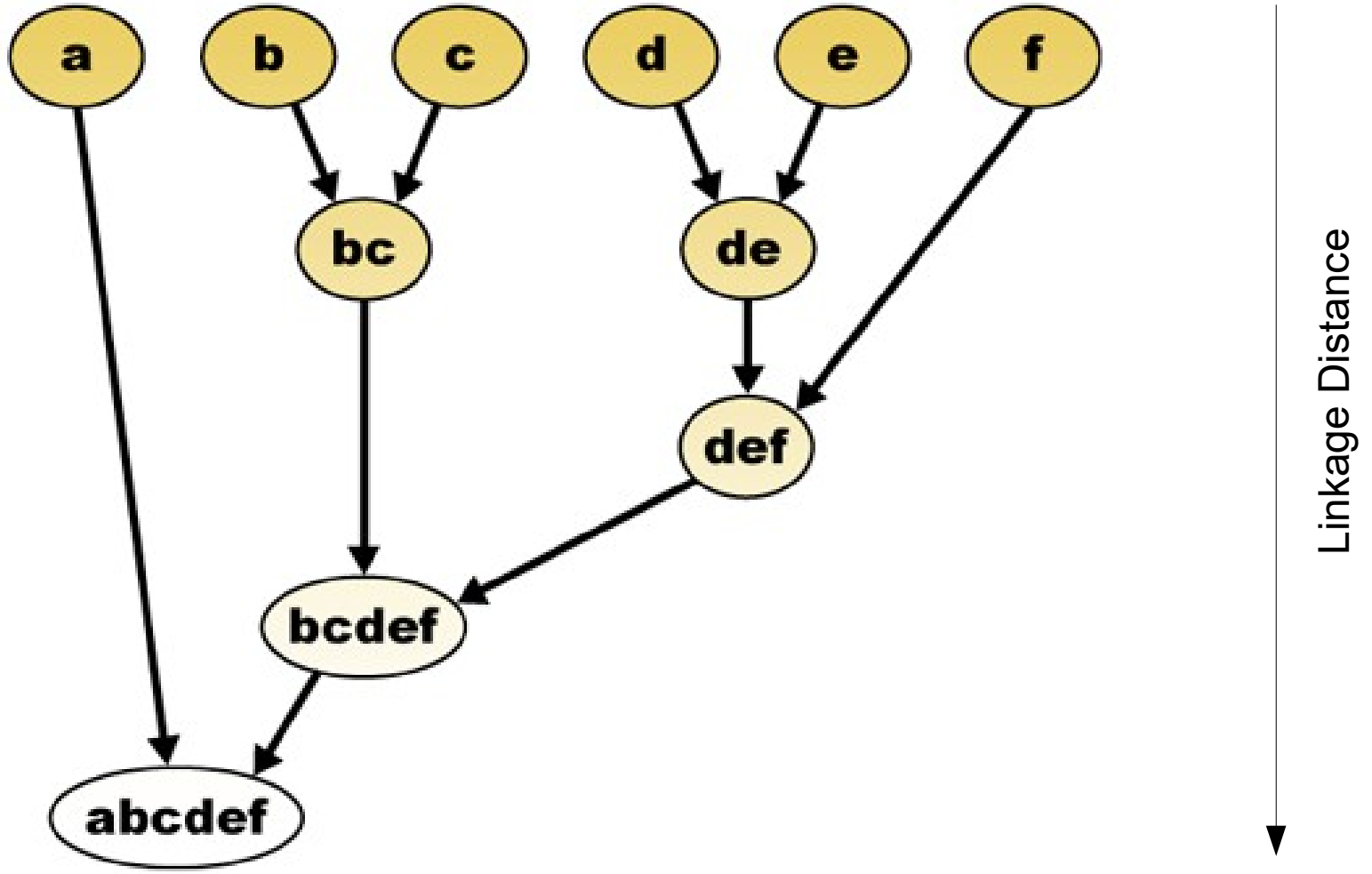
- Generative Modelle (parametrisch, nicht-parametrisch)
  Lernen von Wahrscheinlichkeitsdichtefunktionen
  - Naive Bayes classifier
  - Linear Discriminant Analysis
  - ...
- Diskriminative Modelle
  Maximierung der Qualität des Klassifikators
  auf dem Trainingsset
  - Logistic Regression
  - Perceptron
  - Support Vector Machines (SVM)
  - ...

Clustering

- Hierarchisch
  - Agglomerierend (bottom-up, merging)
  - Unterteilend (top-down, splitting)
- Partitionierende Verfahren
  - K-nearest Neighbour
  - K-means
  - Expectation Maximization
  - Inkrementelle ϵ-Umgebung

Hierarchisches Clustering

Ergebnis: Binärbäume (Dendrogramm)



Distanzen zwischen Klassen

- Single Linkage (nearest neighbour)
  - $D(B1,B2) = \min d(x,y)$ (über alle $x \in B_1$, $y \in B_2$)
    - „langgestreckte Klassen“
- Average Linkage
  - $D(B1,B2) = 1/(n_1 * n_2) * \sum\sum d(x,y)$
    (über alle Paare $x \in B_1$, $y \in B_2$)

Distanzen zwischen Klassen

- Ward's Method
  - $D(B1,B2) = 1/(n_1 * n_2) * \sum d(z,m)$

    ($z \in B_1 \cup B_2$ und m Mittelwert über alle z)
- Complete Linkage (furthest neighbour)
  - $D(B1,B2) = \max d(x,y)$
    - Größte Unähnlichkeit zwischen 2 Clustern
      (Kompakte, annähernd kugelförmige Klassen; gleiche Größe)

Hierchariches Clustering

Agglomerierend

- M Objekte bilden 1-elementige Klassen B1 ... Bm
- Schrittweise Vereinigung von 2 Klassen, deren Distanz am geringsten ist

Unterteilend

- Alle Objekte bilden eine gemeinsame Klasse
- Schrittweises Aufteilung einer Klasse in jene 2 Klassen mit der größten Distanz

Clustering (Hierarchisch)

- Clustering Schedule Graph
  (Plot Clustering steps gegen Linkage Distance)
  - Zur Beurteilung von ausgewogenen (optimalen) Cluster-Größen
    (Anstieg der Linkage Distance)

- Ergebnis des Clusterings stark abhängig von verwendeter Metrik und Skalierung der Merkmale
- Merging einfacher und schneller als splitting (und ähnliche Ergebnisse)

Skalierung des Merkmalsraums

- Skalierung eines Merkmals mit Spannweite a_n
  - $a_n = \max x_n - \min x_n$ (über alle Muster)
  - $x_{n'} = x_n / a_n$
- Skalierung eines Merkmals mit Standardabweichung
  - $X_{n'} = x_n / \sigma_n$
- Problem:
  - Clustering benötigt geeignetes Abstandmaß
  - Abstandsmaß hängt von Standardisierung ab
  - Um richtige Standardisierung durchzuführen, muss man die Clusterzugehörigkeit kennen

Ausblick

- Nächster Termin:

Donnerstag, 1.12.2016 13-15 (c.t.)

Graphical Models