5.6 Support Vector Maschines (SVM)

übernommen von
Stefan Rüping, Katharina Morik
Universität Dortmund
Vorlesung Maschinelles Lernen und Data Mining
WS 2002/03

Erinnerung: Funktionslernen

Gegeben:

Beispiele X in LE

- die anhand einer Wahrscheinlichkeitsverteilung P auf X erzeugt wurden und
- mit einem Funktionswert Y = t(X) versehen sind (alternativ: Eine Wahrscheinlichkeitsverteilung P(Y|X) der möglichen Funktionswerte - verrauschte Daten).

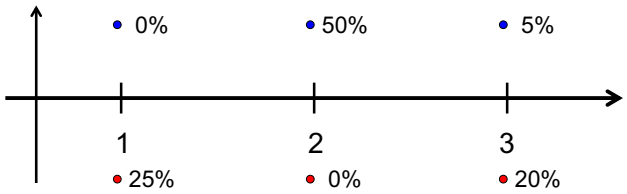
H die Menge von Funktionen in LH.

Ziel: Eine Hypothese h(X) ∈ H, die das erwartete Fehlerrisiko R(h) minimiert.

Risiko:

$$R(h) = \sum_x Q(x,h)P(x)$$

Beispiel: Funktionenlernen



H = { f_a | $f_a(x) = 1$, für $x \geq a$, $f_a(x) = -1$ sonst, $a \in \Re$}

$R(f_0)$	= 0,25 + 0 + 0,20	= 0,45
$R(f_{1,5})$	= 0 + 0 + 0,20	= 0,20
$R(f_{3,5})$	= 0 + 0,5 + 0,05	= 0,55

Reale Beispiele

Klassifikation: Q(x,h) = 0, falls t(x) = h(x), 1 sonst

- Textklassifikation (x = Worthäufigkeiten)
- Handschriftenerkennung (x = Pixel in Bild)
- Vibrationsanalyse in Triebwerken (x = Frequenzen)
- Intensivmedizinische Alarmfunktion (x = Vitalzeichen)

Regression: $Q(x,h) = (t(x)-h(x)))^2$

- Zeitreihenprognose (x = Zeitreihe, t(x) = nächster Wert)

Erinnerung: Minimierung des beobachteten Fehlers

Funktionslernaufgabe nicht direkt lösbar. Problem:

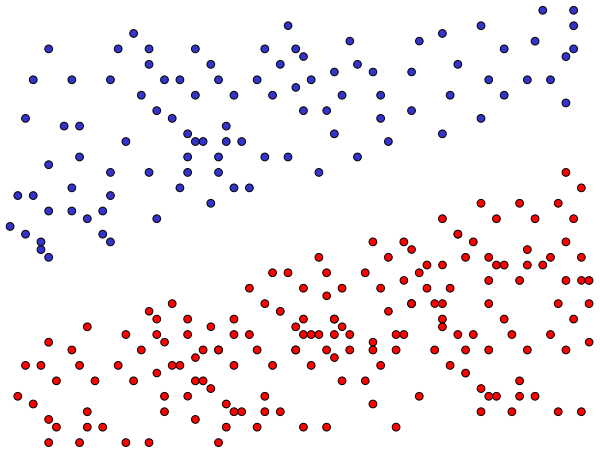
- Die tatsächliche Funktion t(X) ist unbekannt.
- Die zugrunde liegende Wahrscheinlichkeit ist unbekannt.

Ansatz:

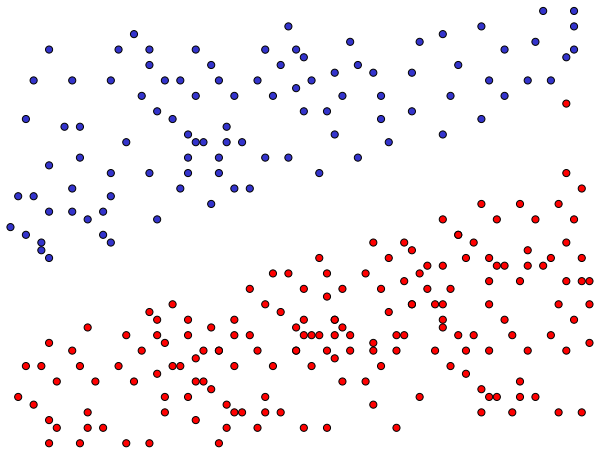
- eine hinreichend große Lernmenge nehmen und für diese den Fehler minimieren.

$\Rightarrow$ Empirical Risk Minimization

Beispiel



Beispiel II

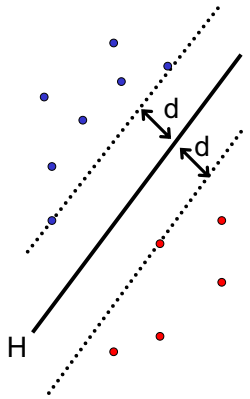


Probleme der ERM

- Aufgabe ist nicht eindeutig beschrieben: Mehrere Funktionen mit minimalem Fehler existieren. Welche wählen?
- Overfitting: Verrauschte Daten und zu wenig Beispiele führen zu falschen Ergebnissen.

Die optimale Hyperebene

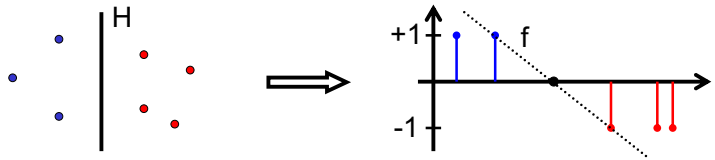
- Beispiele heißen linear trennbar, wenn es eine Hyperebene H gibt, die die positiven und negativen Beispiele voneinander trennt.
- H heißt optimale Hyperebene, wenn ihr Abstand d zum nächsten positiven und zum nächsten negativen Beispiel maximal ist.
- Satz: Es existiert eine eindeutig bestimmte optimale Hyperebene.



Berechnung der opt. Hyperebene

- Hyperebene
  $H = \{x \mid w*x+b = 0\}$
- H trennt (x_i,y_i), $y_i \in \{\pm 1\}$
- H ist optimale Hyperebene

- Entscheidungsfunktion $f(x) = w*x+b$
- $f(x_i) > 0 \Leftrightarrow y_i > 0$
- $||w||$ minimal und
  $f(x_i) \geq 1$, wenn $y_i = 1$
  $f(x_i) \leq -1$, wenn $y_i = -1$

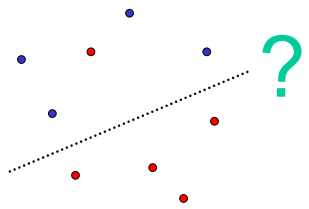


Optimierungsaufgabe der SVM

- Minimiere $||w||^2$
- so dass für alle i gilt:
  $f(x_i) = w*x_i+b \geq 1$ für $y_i = 1$ und
  $f(x_i) = w*x_i+b \leq -1$ für $y_i = -1$
- Äquivalente Nebenbedingung: $y_i*f(x_i) \geq 1$
- Konvexes, quadratisches Optimierungs-problem $\Rightarrow$ eindeutig in $O(n^3)$ lösbar.
- Satz: $||w|| = 1/d$, d = Abstand der optimalen Hyperebene zu den Beispielen.

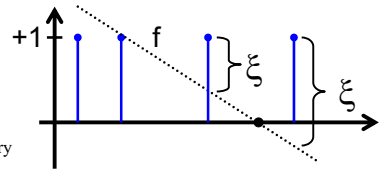
Nicht linear trennbare Daten

- In der Praxis sind linear trennbare Daten selten.
- 1. Ansatz: Entferne eine minimale Menge von Datenpunkten, so dass die Daten linear trennbar werden (minimale Fehlklassifikation).
- Problem: Algorithmus wird exponentiell.



Weich trennende Hyperebene

- Wähle $C \in \Re_{>0}$ und minimiere $\|w\|^2 + C\sum_{i=1}^{n} \xi_i$
- so dass für alle i gilt:
  $f(x_i) = w*x_i+b \geq 1-\xi_i$ für $y_i = 1$ und
  $f(x_i) = w*x_i+b \leq -1+\xi_i$ für $y_i = -1$
- Äquivalent: $y_i*f(x_i) \geq 1- \xi_i$



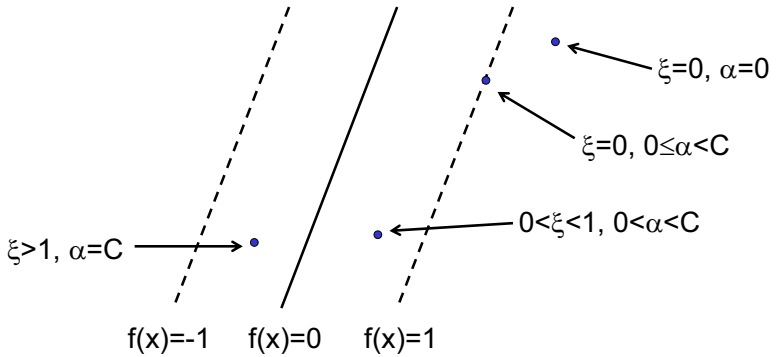
Duales Optimierungsproblem

- Umformung mit Lagrange-Multiplikatoren liefert einfacheres Optimierungsproblem:
- Maximiere

$$W(\alpha) = \sum_{i=1}^{n} \alpha_i - \tfrac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j \left(x_i * x_j\right)$$

- unter $0 \leq \alpha_i \leq C$ für alle i und $\sum \alpha_i y_i = 0$
- Es gilt $w = \sum \alpha_i y_i x_i$, also $f(x) = \sum \alpha_i y_i (x_i * x) + b$

Bedeutung von ξ und α



Beispiele x_i mit $\alpha_i>0$ heißen Stützvektoren $\Rightarrow$ SVM

Optimierungsalgorithmus

```
s = Gradient von W(α)                 // s_i = ∑α_j(x_j*x_i)
while(nicht konvergiert(s))           // auf  ε genau
    WS = working_set(s)               // suche k „gute“ Variablen
    α‘ = optimiere(WS)                // k neue α-Werte
    s = update(s, α‘)                 // s = Gradient von W(α‘)
```

- Gradientensuchverfahren
- Trick: Stützvektoren allein definieren Lösung
- Weitere Tricks: Shrinking, Caching von x_i*x_j

Was wissen wir jetzt über SVM‘s?

- Funktionslernen als allgemeine Lernaufgabe
- Minimierung des empirischen Risikos als Lösungsstrategie
- Optimale Hyperebene präzisiert die ERM
- Praxis: weich trennende Hyperebene
- Berechnung mittels SVM und dualem Problem
- **Offene Fragen:** Generelles Prinzip hinter der optimalen Hyperebene? Nicht lineare Daten?