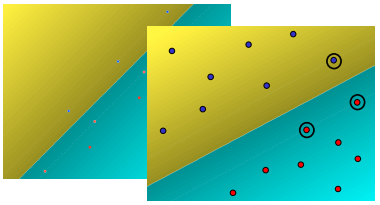
Einführung in Support-Vektor-Machinen

Einleitender Vortrag im Seminar
„Mustererkennung mit SVM“ WS 02/03

Übersicht

1. Lineare SVM:
   separabler Fall
2. Lineare SVM:
   nicht separabler Fall
3. Nichtlineare SVM
4. Eigenschaften
5. Anwendung
6. Zusammenfassung

1. Lineare SVM: separabler Fall



Annahmen

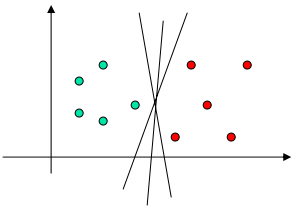
- Zweiklassenproblem
- Überwachtes Lernen:
  Lerndatensatz aus n Beobachtungen

$$(x_1, y_1), \ldots, (x_n, y_n) \in \mathbb{R}^d \times \{-1, +1\}$$

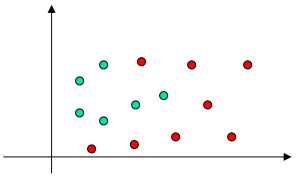
Linear separabler Fall

- Anschauliche Definition:
  - Daten linear separabel, wenn eine Hyperebene existiert, die die Daten separiert.
  - In $\mathbb{R}^2$:

linear separabel

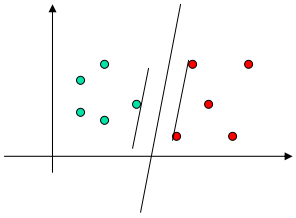


nicht linear separabel



Idee

- Trennende Hyperebene mit größtem Rand:

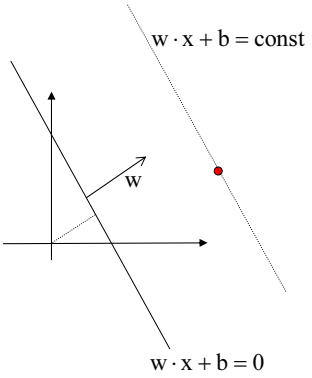


  - Anschaulich sinnvoll
  - Theoretisch sinnvoll
  - Lösung abhängig von wenigen Daten (Support Vektoren)

Formalisierung

- Hyperebene
  - Menge der Punkte x mit $w \cdot x + b = 0$
  - $w \in \mathbb{R}^d$ Normalenvektor
  - $b \in \mathbb{R}$ skalierter Abstand vom Ursprung
- Linearer Klassifikator

  $\hat{y}(x) = \text{sgn}(w \cdot x + b)$

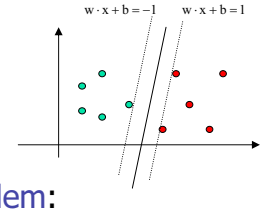


- Beobachtung: $w \cdot x + b = 0 \quad \Leftrightarrow \quad aw \cdot x + ab = 0$

  daher Normierung der Hyperebene:

  $\min_i |w \cdot x_i + b| = 1$

  hiermit ist Rand $\gamma = 2 / \|w\|$



- Ansatz als Optimierungsproblem:

  Minimieren von $\|w\|^2$

  mit $y_i(w \cdot x_i + b) \geq 1$ ($\Leftrightarrow$ korrekt klassifiziert und leere Randzone)

Optimierungsproblem

- Verallgemeinerte Lagrangefunktion

  Minimiere

$$L(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i [y_i (w \cdot x_i + b) - 1]$$

  mit $\alpha_i \geq 0$

- Duale Formulierung

  Maximiere

$$L'(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j (x_i \cdot x_j)$$

  mit $\alpha_i \geq 0$ und $\sum_{i=1}^{n} y_i \alpha_i = 0$

- numerische Lösung via Karush-Kuhn-Tucker-Bedingungen (KKT).

Lösung

- Lösung α des dualen Problems liefert gewünschte Hyperebene

$$w = \sum_{i=1}^{n} \alpha_i y_i x_i$$

$$b = -\frac{\max_{y_i=-1}(w \cdot x_i) + \min_{y_i=1}(w \cdot x_i)}{2}$$

- Beobachtungen:
  - Für alle Beispiele außerhalb des Randes ist $\alpha_i = 0$ => „spärliche" Darstellung der Lösung!!
  - $\alpha_i \neq 0$ => x_i Support-Vektor,
  - Eindeutigkeit der Ebene, globales Optimum.

Entscheidungsregel

- Einsetzen von w :

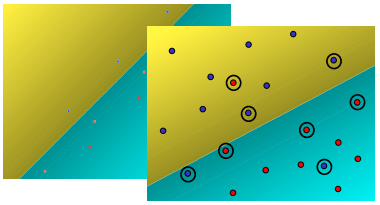
$$\hat{y}(x) = \operatorname{sgn}\left(w \cdot x + b\right)$$

$$= \operatorname{sgn}\left(\sum_{i:\alpha_i \neq 0} \alpha_i y_i (x_i \cdot x) + b\right)$$

Demo separabler Fall

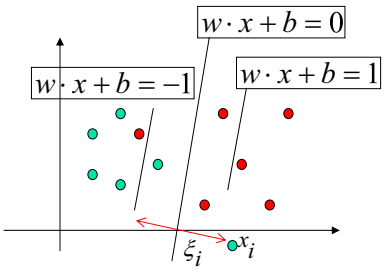
- svmtoy
  - Randzone ist leer
  - Unabhängigkeit von „äußeren" Daten
- Matlab toolbox
  - Kleiner Anteil der SV an Gesamtdaten
  - SV auf den Geraden $w \cdot x + b = \pm 1$

2. Lineare SVM: nicht separabler Fall



Nicht linear separabler Fall

- Idee: Bestrafen von Randverletzungen via Lockerungsvariablen ξ_i



Formalisierung

- Minimieren von $\|w\|^2 + C\sum_{i=1}^{n}\xi_i$ (C fest gewählt)

  mit $y_i(w \cdot x_i + b) \geq 1 - \xi_i$ und $\xi_i \geq 0$

- Duale Formulierung

  Maximiere

  $$L'(\alpha) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j (x_i \cdot x_j)$$

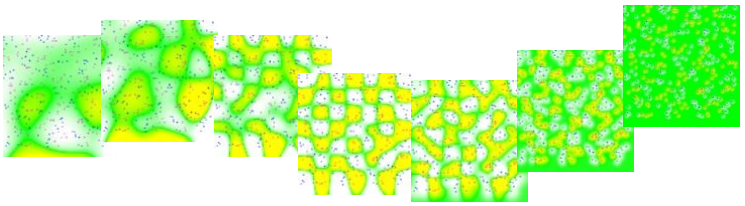
  mit $C \geq \alpha_i \geq 0$ und $\sum_{i=1}^{n} y_i \alpha_i = 0$

- Wieder garantiertes globales Optimum.

Demo nicht separabler Fall

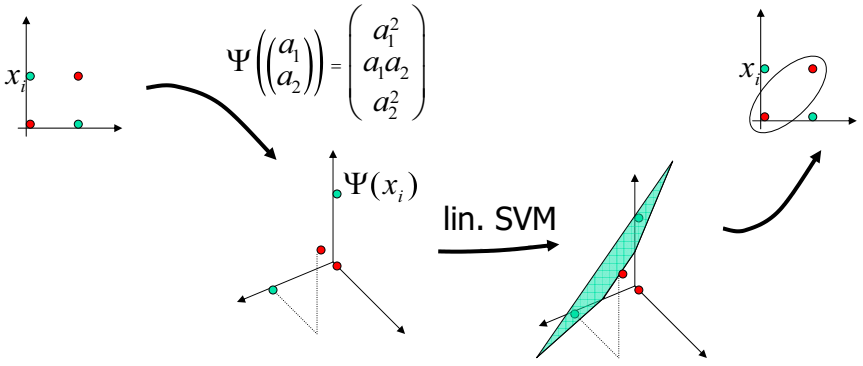
- svmtoy
  - Randzone nicht leer
  - Wachsendes C => kleinerer Rand
- Matlab toolbox
  - Support-Vektoren: Vektoren innerhalb Randzone und falsch klassifizierte
  - Qualitativ mehr SV als separabler Fall

3. Nichtlineare SVM



Idee

- Nichtlineare Abbildung $\Psi(x): \mathbb{R}^N \rightarrow \mathrm{H}$ anschließend lineare SVM
  - Beispiel XOR-Problem:



Kernel Trick

- Problem:
  Berechnungskomplexität bei Operieren auf $\Psi(x_i)$
- Beobachtung:
  in Training und Klassifikation treten nur Skalarprodukte $\Psi(\mathrm{x_i})\cdot\Psi(\mathrm{x_j})$ auf.
- Trick:
  Effizient berechenbare Kernfunktion
  $$\mathrm{K}(\mathrm{x_i},\mathrm{x_j}) = \Psi(\mathrm{x_i})\cdot\Psi(\mathrm{x_j})$$
  macht Kenntnis von Ψ und H überflüssig!
- Anwendbar auf Vielzahl linearer Algorithmen „kernelization"

Kernfunktionen

- Vorstellung:
  - Ähnlichkeitsmaß, verallgemeinertes Skalarprodukt
- Einfache Beispiele
  - Linear $\mathrm{K}(\mathrm{x},\mathrm{y}) = \mathrm{x}\cdot\mathrm{y}$
  - Polynomial $\mathrm{K}(\mathrm{x},\mathrm{y}) = (\mathrm{x}\cdot\mathrm{y}+\mathrm{c})^{\mathrm{d}}$
  - Sigmoid $\mathrm{K}(\mathrm{x},\mathrm{y}) = \tanh(\kappa(\mathrm{x}\cdot\mathrm{y})+\theta)$
  - Gauß $\mathrm{K}(\mathrm{x},\mathrm{y}) = \exp\left(-\|\mathrm{x}-\mathrm{y}\|^2/(2\sigma^2)\right)$
- Allgemeine Bedingung
  - Mercer Theorem oder
  - Positive Definitheit

Endformulierung

- Trainieren:

  maximiere

$$L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j K(x_i, x_j)$$

  mit $\sum_{i=1}^{l} y_i \alpha_i = 0, \quad C \geq \alpha_i \geq 0$

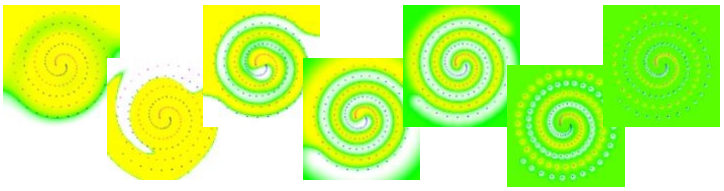
$$\left(b = -\frac{\max_{y_i=-1} \sum_{j=1}^{n} y_j \alpha_j K(x_i, x_j) + \min_{y_i=1} \sum_{j=1}^{n} y_j \alpha_j K(x_i, x_j)}{2} \right)$$

- Klassifizieren:

$$\hat{y}(x) = \operatorname{sgn}\left(\sum_{i:\alpha_i \neq 0} \alpha_i y_i K(x_i, x) + b \right)$$

Demo nichtlinearer Fall

- Rolle der Kerne:
  - Polynomialer Kern => spezieller Polynomklassifikator
  - Ähnliche SV-Mengen bei verschiedenen Kernen
  - Gamma-Variation bei Gauß-Kern



4. Eigenschaften

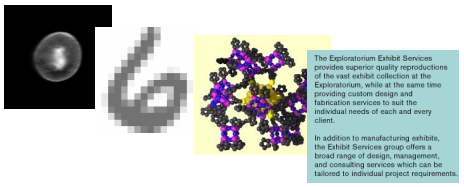
Komplexitäten

- Training:
  - Speicher $O(n^2)$
  - Laufzeit $O(n^3)$
- Klassifikation ($\hat{y}(x) = \operatorname{sgn}\left(\sum \alpha_i y_i K(x_i \cdot x) + b\right)$)
  - Speicher $O(n_s)$ mit $n_s :=$ Anzahl der SV
  - Laufzeit $O(n_s \cdot d)$ (d Dimension von x)

Generalisierungsfähigkeit

- Overfitting?
  - scheint nicht einzutreten
  - Vorurteil: viele Parameter im Klassifikator => schlechte Generalisierungsfähigkeit.
- Curse of Dimensionality?
  - Komplexitätsproblem: durch Kern-Trick irrelevant
  - Problem der Lerndatenmenge: empirisch irrelevant
- Statistische Lerntheorie:
  - Erklärung für gute Generalisierung

5. Anwendung



Anwendungen qualitativ

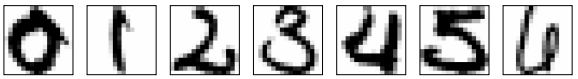
- Allgemein: beste bzw. state-of-the-art Ergebnisse
- Quelle für SVM-bezogene Arbeiten:
  - www.kernel-machines.org
- Gebiete [www.clopinet.com/isabelle/Projects/SVM/applist.html]
  - Medizin:
    - Brustkrebsdiagnose/Prognose
  - Biologie:
    - Protein-Anfangspunkterkennung in Nukleotidsequenzen
    - Voraussage funktionaler Eigenschaften von Proteinsequenzen
    - Pollenerkennung

Anwendungen qualitativ

- Physik:
  - Partikel-/Quarkfarben-Identifikation, Detektion von Motorklopfen
- Allgemeine Bild- / Signalverarbeitung:
  - Zeitreihenvoraussage
  - Bildklassifikation
  - 3D Objekterkennung
  - Gesichtserkennung in Bildern
- Dokumentenanalyse:
  - Handschriftenerkennung, Textkategorisierung

Anwendungen quantitativ

- Beispiel US Postal Service Digits
  - 16x16 Grauwertbilder
  - 7291 Training, 2007 Test-Beispiele



  - Ergebnisse:

Klassifikator	Fehlerrate
Mensch	2.5%
2-Schicht NN	5.9%
5-Schicht NN	5.1%
SVM (polynom grad 3)	4.0%
SVM + Invarianz	3.2%

Verfügbare Implementationen

- MATLAB-Toolboxen
- Svmlight:
  - 2 Programme, Skalierung Training $o(n^2)$
  - Datenübergabe per Datei
  - Strikte Lizenzregelung
  - Keine Multiklassenprobleme
- Libsvm:
  - Funktionalität/Effizienz vergleichbar mit Svmlight
  - Freie C++ Funktionensammlung
  - Multiklassenprobleme

Vorgehen bei konkreter Anwendung

- Geeignete Datenrepräsentation wählen und implementieren
- Geeigneten Kern wählen
  - Richtlinie: Kern muß Ähnlichkeit zwischen Mustern messen
  - Daten in Vektorraum -> Standardkerne vielversprechend
  - Sonst Design (und Implementation) eines problemspezifischen Kerns.
- Wählen/Variieren der Kernparameter + C
- Training und Evaluation
- Wahl der besten SVM

6. Zusammenfassung

Schwächen

- Langsames, speicherintensives Lernen
- Langsames Klassifizieren
- Keine offensichtliche Multiklassenerweiterung
- Fehlende Aussagen zu Klassifikationssicherheit
- Keine allgemeinen Kriterien zur Kernwahl

Stärken

- Empirisch hervorragend
- Schöne Theorie
  - Garantiert globales Optimum des Funktionals
  - Aussagen zur Generalisierungsfähigkeit
- Existenz schneller Implementationen
- Leichte Handhabbarkeit
  - Wenig Parameter per Hand zu setzen
  - (fast) keine Architektur-Wahl
  - Kaum a-priori-Wissen benötigt
- Spärliche Darstellung der Lösung
- Anschauliche Funktionsweise