Support-Vektor-Maschinen: Übersicht

- Lineare Separierbarkeit
- Kernels
- VC-Dimension
- strukturelle Risiko-Minimierung

Ansatz

- Projeziere Instanzen in hochdimensionalen Raum
- Lerne lineare Trennfunktionen mit maximalem Margin
- Lernen ist hier Optimierung

Vorteile:

- Exzellente empirische Ergebnisse bei Zeichenerkennung, Textklassifikation, ...
- Theoretische Fundierung (PAC)
- vermeiden Overfitting in hochdimensionalen Räumen
- Globale Optimierungsmethode, keine lokalen Minima

Nachteile:

- Anwendung des Klassifikators kann teuer sein

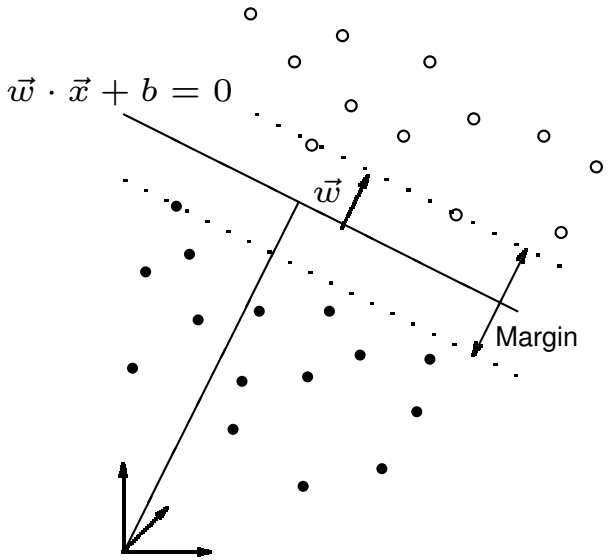
Lineare Separation

Suchen lineare Separation.
Dies kann als Constraint-Satisfaction-Problem angesehen werden:

$$\vec{x_i} \cdot \vec{w} + b \geq +1 \text{ falls } y_i = f(x_i) = +1$$
$$\vec{x_i} \cdot \vec{w} + b \geq -1 \text{ falls } y_i = f(x_i) = -1$$

Äquivalente Darstellung:

$$y_i(\vec{x_i} \cdot \vec{w} + b) - 1 \geq 0$$



Lineare Separation

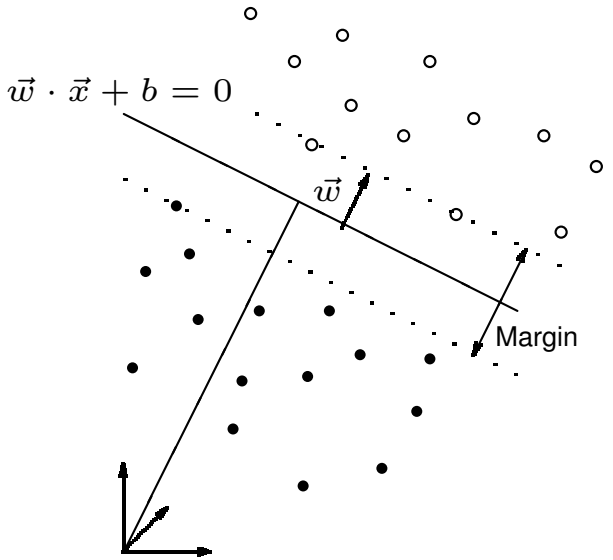
Suche Hyperebene mit maximalem Margin.

Maximaler Margin $\longrightarrow$ minimale Norm

Optimierungsproblem:

Minimiere $||\vec{w}||$ unter den Bedingungen

$$y_i(\vec{x_i} \cdot \vec{w} + b) - 1 \geq 0, \forall i$$



Duale Repräsentation:

$$f(\vec{x}) = sgn\left(\vec{w} \cdot \vec{x} + b\right) = sgn\left(\sum \alpha_i y_i \vec{x_i} \cdot \vec{x} + b\right)$$

$$\vec{w} = \sum \alpha_i y_i \vec{x_i}$$

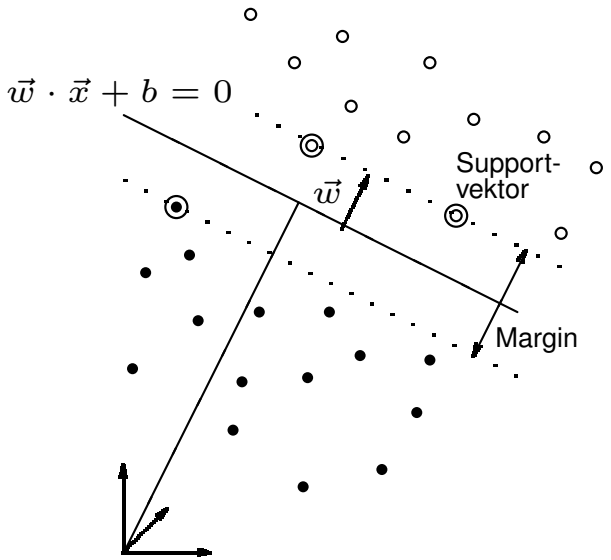
Lineare Separation

- reduzieren Optimierungsproblem auf Gewichte für die Beispiele
  - Fast alle Beispiele haben Gewicht 0

Margin wird nur durch wenige Beispiele bestimmt

- diese nennen wir *Support-Vektoren*
- Beispiele mit $\alpha_i > 0$
- Problem in Form der Support-Vektoren:

$$f(\vec{x}) = sgn\left(\sum_{s_i \in \text{support\_vectors}} \alpha_i y_i \vec{s_i} \cdot \vec{x} + b\right)$$



Nicht linear-trennbare Mengen

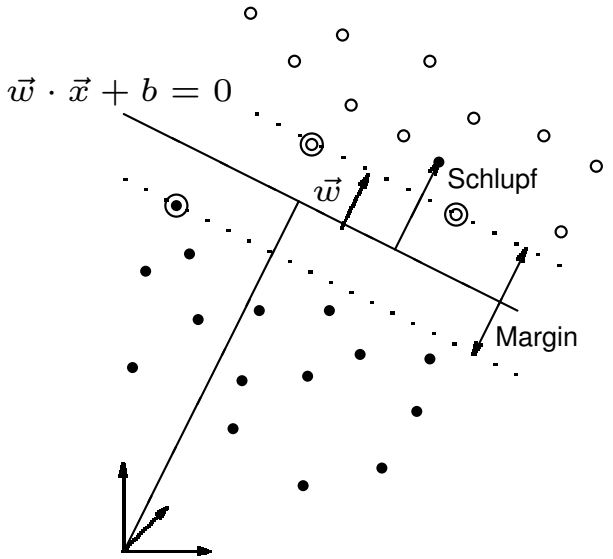
Füge ‘Schlupfvariable’ $\xi_i \geq 0$ für jedes Beispiel hinzu

Neues Optimierungsproblem:

Minimiere $||\vec{w}||^2 + C\left(\sum_i \xi_i\right)^2$ unter den Bedingungen

$$\vec{x_i} \cdot \vec{w} + b \geq +1 - \xi_i \text{ falls } y_i = +1$$
$$\vec{x_i} \cdot \vec{w} + b \geq -1 + \xi_i \text{ falls } y_i = -1$$

C Konstante, ‘von Hand’



Nicht-lineare SVMs

Angenommen, wir haben Beispiele aus $X = \mathcal{R}^{n_1}$ und benötigen nicht-lineare Separation.

$\rightarrow$ projeziere X in einen höherdimensionalen Raum $X' = \mathcal{R}^{n_2}$, in dem die Daten linear trennbar sind.

- sei $\Phi : X \rightarrow X'$ diese Transformation

Beobachtung:

- Lernen benutzt nur Punktprodukt $\Phi(\vec{x_i}) \cdot \Phi(\vec{y_i})$
- Falls wir Funktion K mit $K(\vec{x_i}, \vec{y_i}) = \Phi(\vec{x_i}) \cdot \Phi(\vec{y_i})$ finden, kann Lernen in $\mathcal{R}^{n_2}$ mit gleichem Aufwand wie in $\mathcal{R}^{n_1}$ durchgeführt werden.

  $\rightarrow$ K heißt *Kernel*

- Klassifikation: $f(\vec{x}) = sgn\left(\sum_{s_i \in \text{support\_vectors}} \alpha_i y_i K(\vec{s_i}, \vec{x}) + b\right)$

Beispiel

$X = \mathcal{R}^2, X' = \mathcal{R}^3$

$$\Phi(\vec{x}) = \Phi(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}) = \begin{pmatrix} x_1^2 \\ \sqrt{2}x_1x_2 \\ x_2^2 \end{pmatrix}$$

Dann kann folgender Kernel benutzt werden:

$$K(\vec{x_i}, \vec{x_j}) = \Phi(\vec{x_i}) \cdot \Phi(\vec{x_j}) = (\vec{x_i} \cdot \vec{x_j})^2$$

Wichtig:

- Transformation Φ muß nicht explizit ausgeführt werden

$\rightarrow$ Damit kann prinzipiell auch in unendlich-dimensional Räume transformiert werden

Nicht-lineare SVMs

Andere häufig benutzte Kernels:

- Polynomiale Klassifikation vom Grad p

$$K(\vec{x_i}, \vec{x_j}) = (\vec{x_i} \cdot \vec{x_j} + 1)^p$$

- Gaussche Radiale-Basisfunktion

$$K(\vec{x_i}, \vec{x_j}) = e^{-||\vec{x_i} \cdot \vec{x_j}||^2/2\sigma^2}$$

- Beispiel für Kernel, für den sich kein 'sinnvolles' Φ angeben läßt:

$$K(\vec{x_i}, \vec{x_j}) = \tanh(\kappa \vec{x_i} \cdot \vec{x_j} - \delta)$$

Vorteile von Kernels

- Trennung von allgemeinen Lernprinzipien und Domänenwissen
  - Kernels kodieren Wissen über Domäne $\longrightarrow$ 'Ähnlichkeit zwischen Instanzen
  - beliebige Lernverfahren benutzbar
  - sehr günstig für Software-Engineering und Analyse
- Kernels kombinierbar (Addition, Multiplikation, ...)
  - beliebig komplizierte Kernels konstruierbar
- über beliebigen Lernbereichen definierbar
  - Bilder, Texte, ...

Welchen Kernel wählen?

Einschub: VC-Dimension

- Es gibt (meistens) mehrere Hypothesen, die mit den gegebenen Daten konsistent sind

  $\longrightarrow$ Welche davon auswählen?

- Wähle einfachste Hypothese (Occams Razor)

  $\longrightarrow$ Welches ist die einfachste?

  – Minimum-Description-Length-Principle: Wähle die kürzeste

    $*$ Ist die kürzeste auch die einfachste?

    $\longrightarrow$ Hängt von der Kodierung ab.

Was tun?

Vapnik-Chervonenkis-Dimension

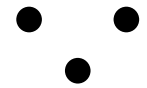
Definition: Sei X eine Menge von Objekten und H eine Menge von Funktionen $f : X \rightarrow \{0, 1\}$.

H *zerschmettert* X gdw. für jede Zuordnung von Objekten zu Labels $\{0, 1\}$ (sog. Dichotonomien) existiert eine Funktion $f \in H$, die diese Zuordnung repräsentiert.

Beispiel:

H Menge aller (gerichteten) Geraden im Raum

X folgende 3 Punkte:



Dichotonomien:

Wie sieht es mit 4 Punkten aus?

Vapnik-Chervonenkis-Dimension

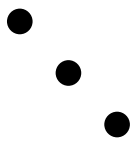
Definition: Sei X eine Menge von Objekten und H eine Menge von Funktionen $f : X \to \{0, 1\}$.

Die **VC-Dimension** von H (notiert als $\mathrm{VC}(H)$) ist die Kardinalität der größten endlichen Teilmenge von X, die H zerschmettert.

Wenn H beliebig große Teilmengen von X zerschmettert, dann setzen wir $\mathrm{VC}(H) = \infty$.

Achtung: $\mathrm{VC}(H) = d$ bedeutet lediglich, daß mindestens eine Teilmenge dieser Kardinalität existiert, die zerschmettert wird. Es bedeutet nicht, daß alle Teilmenge dieser Kardinalität zerschmettert werden.

Beispiel:



Vapnik-Chervonenkis-Dimension: Aufgaben

1. $X = \mathcal{R}$ alle reelen Zahlen, H = Menge aller geschlossenen endlichen Intervalle über $\mathcal{R}$

   Wie groß ist die VC-Dimension?

2. Wie müssen vier Punkte liegen, so so daß sie von beliebigen Geraden (analog dem Beispiel der vorigen Folien) zerschmettert werden?

   Wie groß ist die VC-Dimension beliebiger Geraden im 2-dimensionalen Raum?

3. Wie groß ist die VC-Dimension aller Polynome der Form $y = ax^2 + by + c$ im 2-dimensionalen Raum?

4. Wie groß ist die VC-Dimension aller achsenparalleler Rechtecke im 2-dimensionalen Raum?

Ende Einschub: VC-Dimension

- Es gibt (meistens) mehrere Hypothesen, die mit den gegebenen Daten konsistent sind

  $\longrightarrow$ Welche davon auswählen?

- Wähle einfachste Hypothese (Occams Razor)

  $\longrightarrow$ Welches ist die einfachste?

  $\longrightarrow$ Antwort: VC-Dimension beschreibt Einfachheit der Hypothesenräume

Structural Risc Minimization:

- Gliedere Hypothesenraum in (sinnvolle) Teilklassen
- unter allen konsistenten Hypothesen wähle eine aus einer Teilklasse mit minimaler VC-Dimension

Welchen Kernel wählen?

Aus PAC-Theorie wissen wir, daß mit Wahrscheinlichkeit $(1-\delta)$ gilt:

$$\mathrm{error}_{\mathcal{D}} \leq \mathrm{error}_T + \sqrt{\frac{\mathrm{VC}(H)(\log(2m/\mathrm{VC}(H))+1)-\log(\delta/4)}{m}}$$

- $\mathrm{error}_{\mathcal{D}}$: tatsächlicher Fehler, error_T: Trainingsfehler
- $\mathrm{VC}(H)$: VC-Dimension von H
- m Anzahl der Beispiele

Wähle nun denjenigen Kernel Φ, der obigen Ausdruck minimiert

$\longrightarrow$ *Strukturelle Risiko-Minimierung*

- Trade-Off zwischen error_T und $\mathrm{VC}(H)$ (analog MDL)

Zusammenfassung SVM

- Lerne lineare Separatoren
  - Wähle Separatoren, die den Margin maximieren
  - Schlupfvariablen für unseparierbare Daten
  - Standard-Algorithmen der quadratischen Optimierung
- Lernen von nicht-linearen Funktionen mittels Kernels
  - Projektion in höherdimensionalen Raum
  - Kernel-Funktionen führen diese Projektion implizit aus, ohne zusätzlichen Rechenaufwand
  - Wähle Hypothese (Support-Vektoren und Wahl von Φ), die strukturelles Risiko minimieren
- Fast alle (numerischen) Lernalgorithmen lassen sich mittels Kernels darstellen
  - Neuronale Netze, Nearest Neighbour, Naive Bayes, ...