A Tutorial on Support Vector Machines for Pattern Recognition

Autor: Christopher J.C. Burges, 1998

Quelle: Data Mining and Knowledge Discovery, 2, Seiten 121-167

Referat von: Marcel Fitzner

Kapitel zum SVM-Tutorial

1. Einführungen zur SVM
2. Grundlagen zur SVM
3. Lineare SVMs
4. Nicht-lineare SVMs
5. Fazit
6. SVM in Action!

❖ Motivation

1. SVMs zur **Muster-Erkennung** (Pattern Recognition)

Erkennung / Identifikation von
- handgeschriebene Ziffern (bzw. Zeichen)
- Objekten
- Sprechern
- Gesichtern / Fußgänger (in Autosoftware)
- Text-Kategorien
- u.v.m.

2. SVMs zur **Regressions-Schätzung** (Regression Estimation)

- Benchmark Zeitreihen-Vorhersage-Tests
- „Boston Housing“ - Problem
- „PET Operator Inversion“ - Problem
- Dichte Schätzung (Density Estimation)

❖ Arbeitsweise der SVM (Fall Muster-Erkennung)

1. Lernen der SVM anhand einer Menge von Trainings-Daten

2. Die angelernte SVM soll neue Daten möglichst fehlerfrei klassifizieren
   (SVM: „Ist es das Objekt, auf das ich trainiert wurde?“)

Dilemma:
Um gute „Generalisierungs-Performanz“ zu erhalten,

Geeignete Wahl:
- **Maß an Genauigkeit (Accuracy) bezüglich genau dieser Trainingsdaten**
  bzw.
- **geeignete generelle „Kapazität“* (Capacity) der Lern-Maschine**

(*Fähigkeit der Maschine, beliebige Trainingsdaten ohne Fehler zu lernen)

❖ Arbeitsweise der SVM (Beispiel Muster-Erkennung)

⇒ SVM: „Ist es das Objekt, auf das ich trainiert wurde?“

Dilemma:
Accuracy bzgl. genau dieser Trainingsdaten / generelle Capacity*

Beispiel: SVM soll in einem Bild einen Baum erkennen

Trainings-Beispiel:

Neues Beispiel 1:

SVM 1 mit zu geringer Capacity:

„Ist ein Baum“ (enthält ja grün)

=> Underfitting

(*Fähigkeit der Maschine, **beliebige** Trainingsdaten ohne Fehler zu lernen)

❖ Arbeitsweise der SVM (Beispiel Muster-Erkennung)

⇒ SVM: „Ist es das Objekt, auf das ich trainiert wurde?“

Dilemma:
Accuracy bzgl. genau dieser Trainingsdaten / generelle Capacity*

Beispiel: SVM soll in einem Bild einen Baum erkennen

Trainings-Beispiel:

Neues Beispiel 2:

SVM 2 mit zu hoher Capacity:

„Ist kein Baum“ (zu viele Blätter)

=> Overfitting

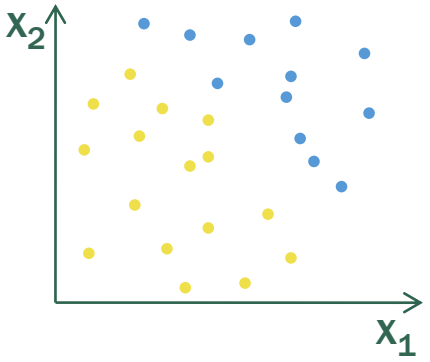
(*Fähigkeit der Maschine, **beliebige** Trainingsdaten ohne Fehler zu lernen)

Kapitel zum SVM-Tutorial

1. Einführungen zur SVM
2. Grundlagen zur SVM
3. Lineare SVMs
4. Nicht-lineare SVMs
5.
6. SVM in Action!

❖ Überwachtes Lernen

Beispiel: $x_i \in R^2$
● : $y_i = 1$
● : $y_i = -1$



Gegeben: *l* Trainings-Daten:
$\{(x_i,y_i) \in P(x,y) \mid x_i \in R^d$, Labels $y_i \in \{-1,1\}$, $i=1,..,l\}$
ansonsten existiert kein Vorwissen

Ziel: **Lernen einer Zuordnungs-Funktion**

$\mathbf{f(x,\alpha) : x_i \rightarrow y_i}$,

mit noch unbekannten Parametern des Vektors α,
die den Erwartungswert des Zuordnungsfehlers
minimiert:

$$R(\alpha) = \frac{1}{2}\int |y - f(x,\alpha)| dP(x,y)$$

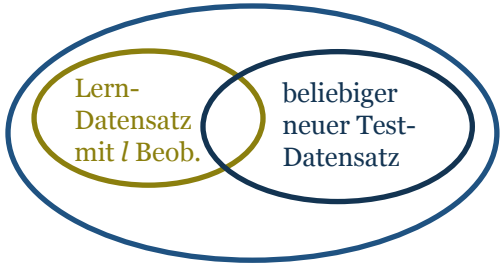
mit $\frac{1}{2}|y_i - f(x_i,\alpha)| \in [0;1]$ als Test-Fehler $R_{test}(\alpha)$ bei neuen Daten.

Der **tatsächliche Fehler R(α)** lässt sich jedoch nicht berechnen,
da die Verbund-Verteilung ***P(x,y)*** unbekannt bzw. ein Schätzer für die Verteilung fehlt.

❖ Überwachtes Lernen

Gegeben: *l* Trainings-Daten:
$\{(x_i,y_i) \in P(x,y) \mid x_i \in R^d$, Labels $y_i \in \{-1,1\}$, $i=1,..,l\}$
ansonsten existiert kein Vorwissen

Stochastischer Prozess mit unbekannter Verteilung P(x,y)



Empirischer Fehler $R_{emp}(\alpha)$

Test-Fehler $R_{test}(\alpha)$ (Generalisierungs-fähigkeit)

Ziel: **Lernen einer Zuordnungs-Funktion**

$\mathbf{f(x,\alpha) : x_i \rightarrow y_i}$,

mit noch unbekannten Parametern des Vektors α,
die den Erwartungswert des Zuordnungsfehlers
minimiert:

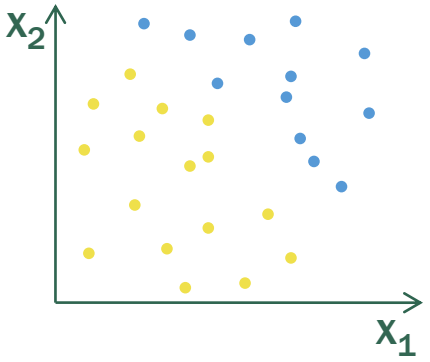
$$R(\alpha) = \frac{1}{2}\int |y - f(x,\alpha)| dP(x,y)$$

mit $\frac{1}{2}|y_i - f(x_i,\alpha)| \in [0;1]$ als Test-Fehler $R_{test}(\alpha)$ bei neuen Daten.

Der **tatsächliche Fehler R(α)** lässt sich jedoch nicht berechnen,
da die Verbund-Verteilung ***P(x,y)*** unbekannt bzw. ein Schätzer für die Verteilung fehlt.

Messen des Trainings-Fehlers

Beispiel: $x_i \in R^2$
● : $y_i = 1$
● : $y_i = -1$



Tatsächlicher Fehler:

$$R(\alpha) = \frac{1}{2}\int |y - f(x,\alpha)| dP(x,y)$$

Bei *l* gegebenen Trainings-Daten erhalten wir:

Trainings-Fehler:

$$R_{emp}(\alpha) = \frac{1}{2l}\sum_{i=1}^{l} |y_i - f(x_i,\alpha)|$$

⇒ Zur Berechnung des **empirischen Fehlers $R_{emp}(\alpha)$** wird keine Verteilung ***P(x,y)*** benötigt!

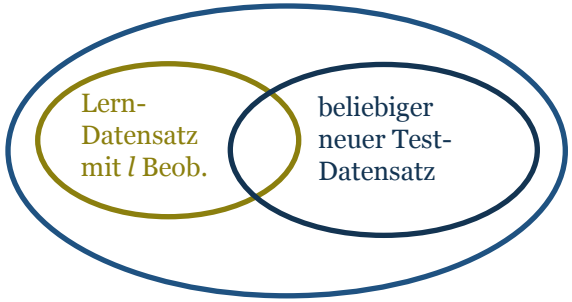
Folgende Fragen stellen sich uns:
- Wie nah ist man nach *l* Trainingsdaten am **tatsächlichen Fehler R(α)**?
- Wie gut kann man aus dem **empirischen Fehler $R_{emp}(\alpha)$** den **tatsächlichen Fehler R(α)** abschätzen?

Antworten liefert hierzu die Lerntheorie von Vapnik-Chervonenkis (VC-Theorie)

❖ Obergrenze für die Generalisierungsfähigkeit des Lernens nach der VC-Theorie

Erinnerung: $R(\alpha) = \frac{1}{2}\int |y - f(x,\alpha)| dP(x,y)$

Stochastischer Prozess mit unbekannter Verteilung P(x,y)



Empirischer Fehler $R_{emp}(\alpha)$

Test-Fehler $R_{test}(\alpha)$ (Generalisierungs-fähigkeit)

Wir wählen nun ein beliebiges $\eta \in [0;1]$
Die folgende obere Abschätzung für den tatsächlichen Fehler $R(\alpha)$ gilt mit Wahrscheinlichkeit $(1-\eta)$:

$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\frac{h(\log(2l/h) - \log(\eta/4)}{l}}$$

VC-Konfidenz Φ

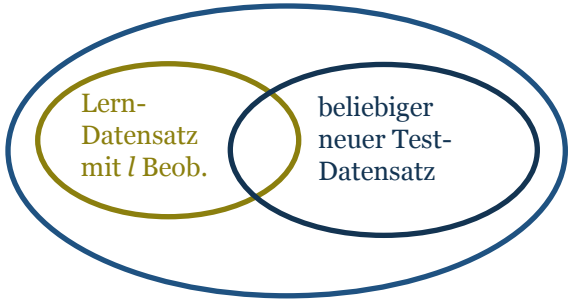
l – Anzahl Beobachtungen
h – VC-Dimension
η – wählbarer Parameter für Konfidenz-Niveau $(1-\eta)$

Abschätzung für $R(\alpha)$ unabhängig von zugrundeliegender Verbundverteilung P(x,y)!
Voraussetzung hierfür: Trainings-/ und Test-Daten werden unabhängig und identisch verteilt gezogen

❖ Obergrenze für die Generalisierungsfähigkeit des Lernens nach der VC-Theorie

Erinnerung: $R(\alpha) = \frac{1}{2}\int |y - f(x,\alpha)| dP(x,y)$

Stochastischer Prozess mit unbekannter Verteilung P(x,y)



Empirischer Fehler $R_{emp}(\alpha)$

Test-Fehler $R_{test}(\alpha)$ (Generalisierungs-fähigkeit)

Wir wählen nun ein beliebiges $\eta \in [0;1]$
Die folgende obere Abschätzung für den tatsächlichen Fehler $R(\alpha)$ gilt mit Wahrscheinlichkeit $(1-\eta)$:

$$R(\alpha) \le R_{emp}(\alpha) + \sqrt{\frac{h(\log(2l/h) - \log(\eta/4)}{l}}$$

VC-Konfidenz Φ

l – Anzahl Beobachtungen
h – VC-Dimension
η – wählbarer Parameter für Konfidenz-Niveau (1- η)

Bei Auswahl zwischen verschiedenen Lern-Maschinen (=Funktionen-Klasse $f(x,\alpha)$), wählt man diejenige LM, die bei gegebenem empirischen Fehler $R_{emp}(\alpha)$ das kleinste VC-Konfidenz-Maß Φ liefert.

❖ VC-Dimension (1/3)

- Ist eine Eigenschaft für Lern-Maschinen (= Menge von Funktionen-Klassen $\{f(\alpha)\}$)
  (α sei eine generische Menge von Parametern; die Wahl eines α spezifiziert eine bestimmte Funktion)

- kann für verschiedene Klassen von Funktionen definiert werden; das Tutorial betrachtet jedoch ausschließlich den Zweiklassen-Fall, also mit den Labels $f(x,\alpha) \in \{-1;1\}\ \forall x,\alpha$

  Bei l Punkten im Trainings-Datensatz sind das 2^l mögliche Funktionen-Klassen, in die die Punkte aufgeteilt werden können.

Begriffs-Einführung:
Findet sich eine Funktion aus der Menge $\{f(\alpha)\}$, so dass alle Labels korrekt zugewiesen werden können, so wird dies als **„Zerschmettern“** der Punkte durch die gefundene Funktionen-Klasse bezeichnet.

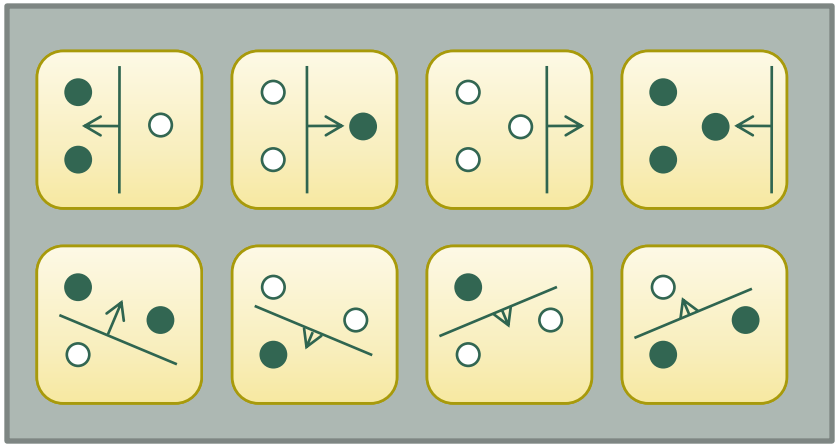
❖ VC-Dimension (2/3)

Definition: VC-Dimension

Für eine Menge von Funktionen {f(α)} ist die VC-Dimension *h* definiert als die **maximale Anzahl von Trainings-Punkten**, die diese Menge zerschmettert.

Beispiel:
- 3 Punkte im R^2
- {f(α)} besteht aus Funktionen, die ausgerichtete Geraden beschreiben.

=> Es können im R^2 3 Punkte mit der Menge {f(α)} zerschmettert werden, nicht jedoch 4!



Bemerkung:
Ist die VC-Dimension *h*, so ***existiert mindestens eine Menge von h Punkten***, die zerschmettert werden können. Im Allgemeinen müssen aber nicht alle (2^l) Mengen zerschmetterbar sein.

❖ VC-Dimension (3/3)

Korrolar:

Die VC-Dimension einer Menge von ausgerichteten Hyper-Ebenen im R^n ist n+1.

Es können stets n+1 Punkte gewählt werden, aus denen ein beliebiger Punkt als Ursprungspunkt fungiere, so sind die n verbleibenden Punkte linear unabhängig.

Jedoch können keine n+2 Punkte gewählt werden, da bereits im R^n keine n+1 Vektoren linear unabhängig sind.

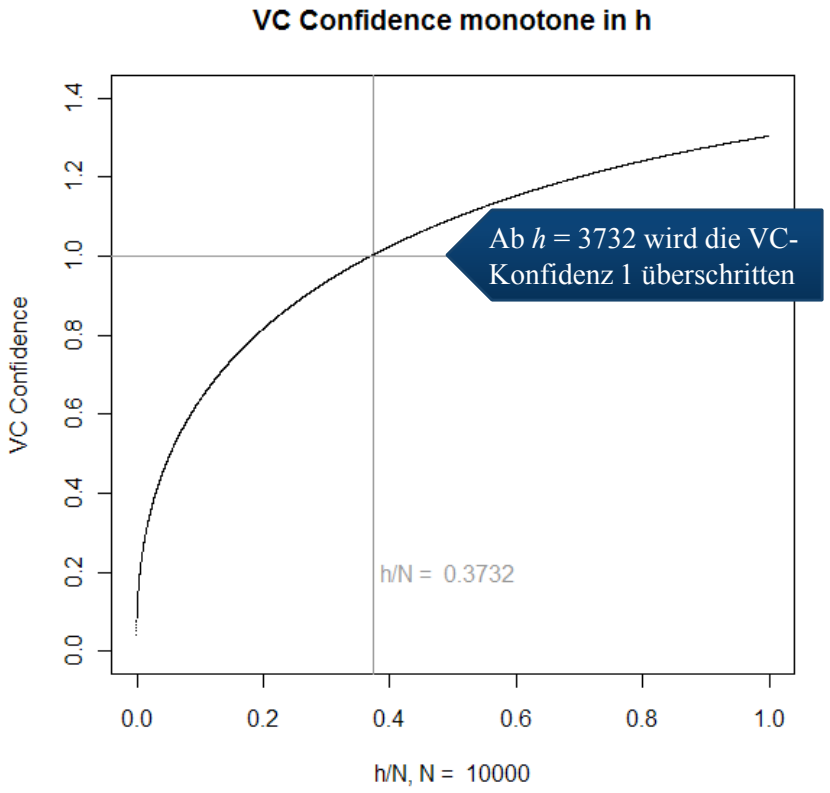
❖ Minimierung der Obergrenze durch Minimierung von h

Erinnerung:
Die folgende obere Abschätzung für den tatsächlichen Fehler R(α) gilt mit Wahrscheinlichkeit (1-η):

$$R(\alpha) \le R_{emp}(\alpha) + \underbrace{\sqrt{\frac{h(\log(2l/h) - \log(\eta/4)}{l}}}_{\text{VC-Konfidenz } \Phi}$$

Beispiel:
Für ein Konfidenz-Niveau von 95% wird η=0.05 gesetzt.
Bei N=*l*=10.000 Trainingspunkte und variierender VC-Dimension *h* erhalten wir anschaulich die VC-Konfidenz:

Wie gestaltet sich die Wahl einer geeigneten LM?



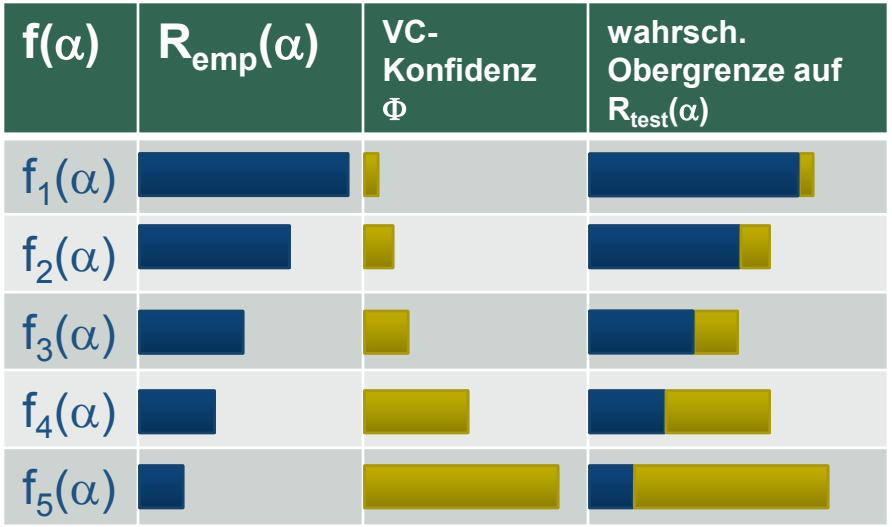
❖ Minimierung der Obergrenze durch Minimierung von h

Erinnerung:
Die folgende obere Abschätzung für den tatsächlichen Fehler R(α) gilt mit Wahrscheinlichkeit (1-η):

$$R(\alpha) \le R_{emp}(\alpha) + \underbrace{\sqrt{\frac{h(\log(2l/h) - \log(\eta/4)}{l}}}_{\text{VC-Konfidenz } \Phi}$$

Wahl-Strategie:
Wähle Lern-Maschine $\{f_i(\alpha)\}$ mit kleinster oberen Schranke auf tatsächlichem Fehler R(α)

- **Für LMs mit $R_{emp}(\alpha) = 0$** wählen wir die LM, deren Menge von Funktionen minimale VC-Dimension *h* haben
- **Für LMs mit $R_{emp}(\alpha) \neq 0$** wird diejenige LM gewählt, die die Obergrenze minimiert.



❖ Minimierung der Obergrenze durch Minimierung von h

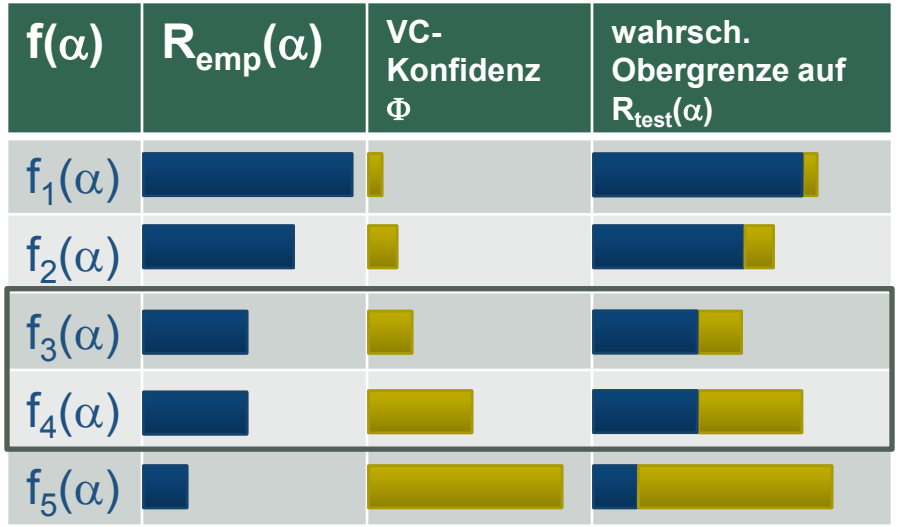
Erinnerung:
Die folgende obere Abschätzung für den tatsächlichen Fehler R(α) gilt mit Wahrscheinlichkeit (1-η):

$$R(\alpha) \le R_{emp}(\alpha) + \underbrace{\sqrt{\frac{h(\log(2l/h) - \log(\eta/4)}{l}}}_{\text{VC-Konfidenz } \Phi}$$

Jedoch:
Die VC-Dimension *h* ist nicht der einzige ausschlaggebende Einflussfaktor!

Bei LMs mit gleichen $R_{emp}(\alpha)$ kann diejenige LM mit höherem *h* trotzdem eine größere Performanz aufweisen!

Am Rande: Für LM's mit unendlichem h (Bspl. kNN-Classifier mit k=1) ist die Obergrenze ungültig und kann nicht herangezogen werden!

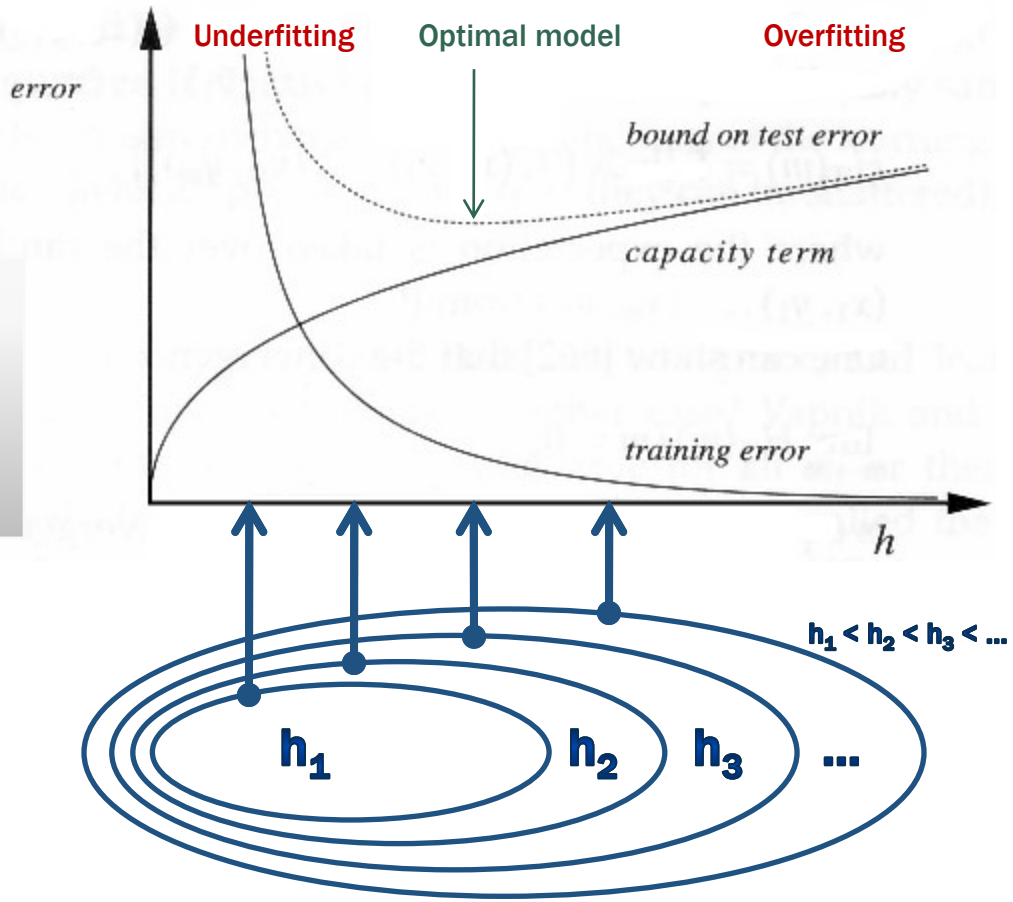


❖ Strukturelle Risiko Minimierung (SRM)

$$R(\alpha) \le R_{emp}(\alpha) + \sqrt{\frac{h(\log(2l/h) - \log(\eta/4)}{l}}$$

VC-Konfidenz Φ

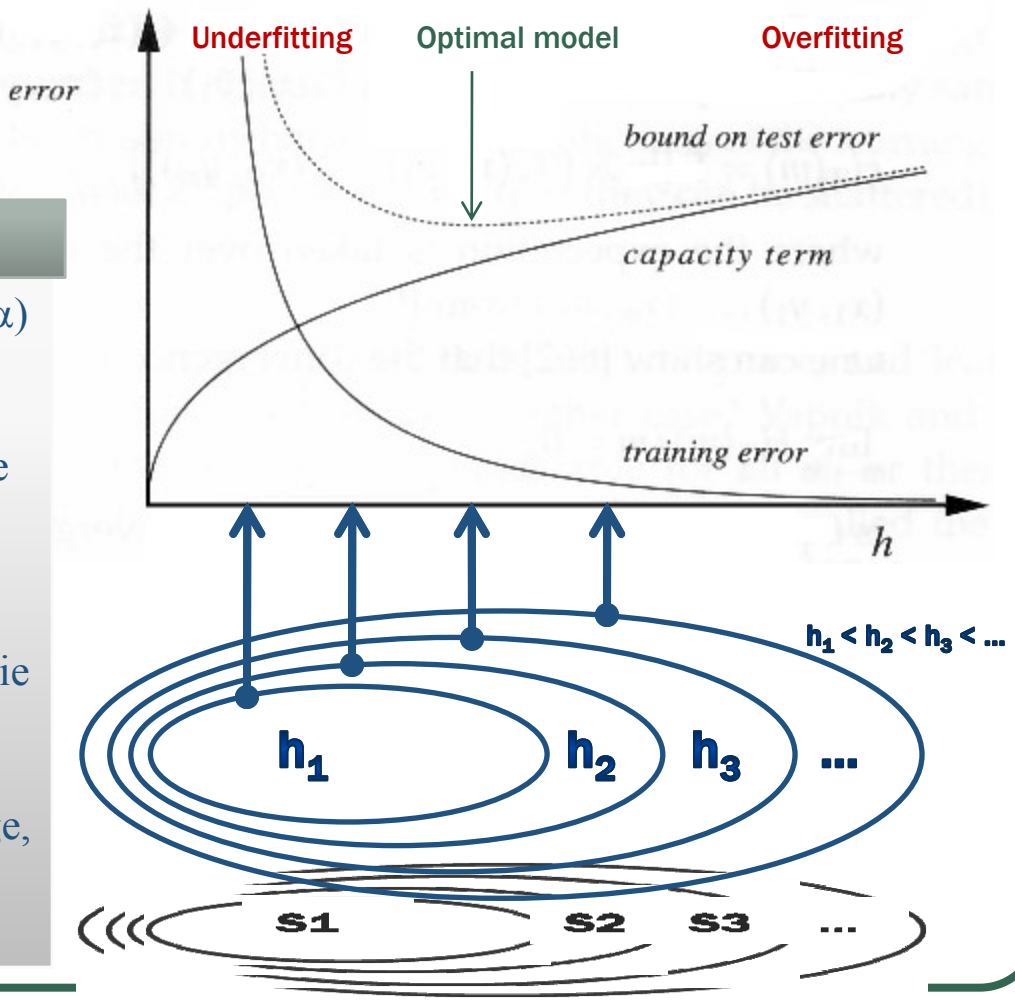
- VC-Konfidenz hängt von der gewählten Funktionen-Klasse f(α) ab
- $R_{emp}(\alpha)$ sowie $R_{test}(\alpha)$ hängen von einer bestimmten Funktion ab, die in der Trainingsprozedur gewählt wurde



❖ Strukturelle Risiko Minimierung (SRM)

$$R(\alpha) \le R_{emp}(\alpha) + \sqrt{\frac{h(\log(2l/h) - \log(\eta/4))}{l}}$$

VC-Konfidenz Φ



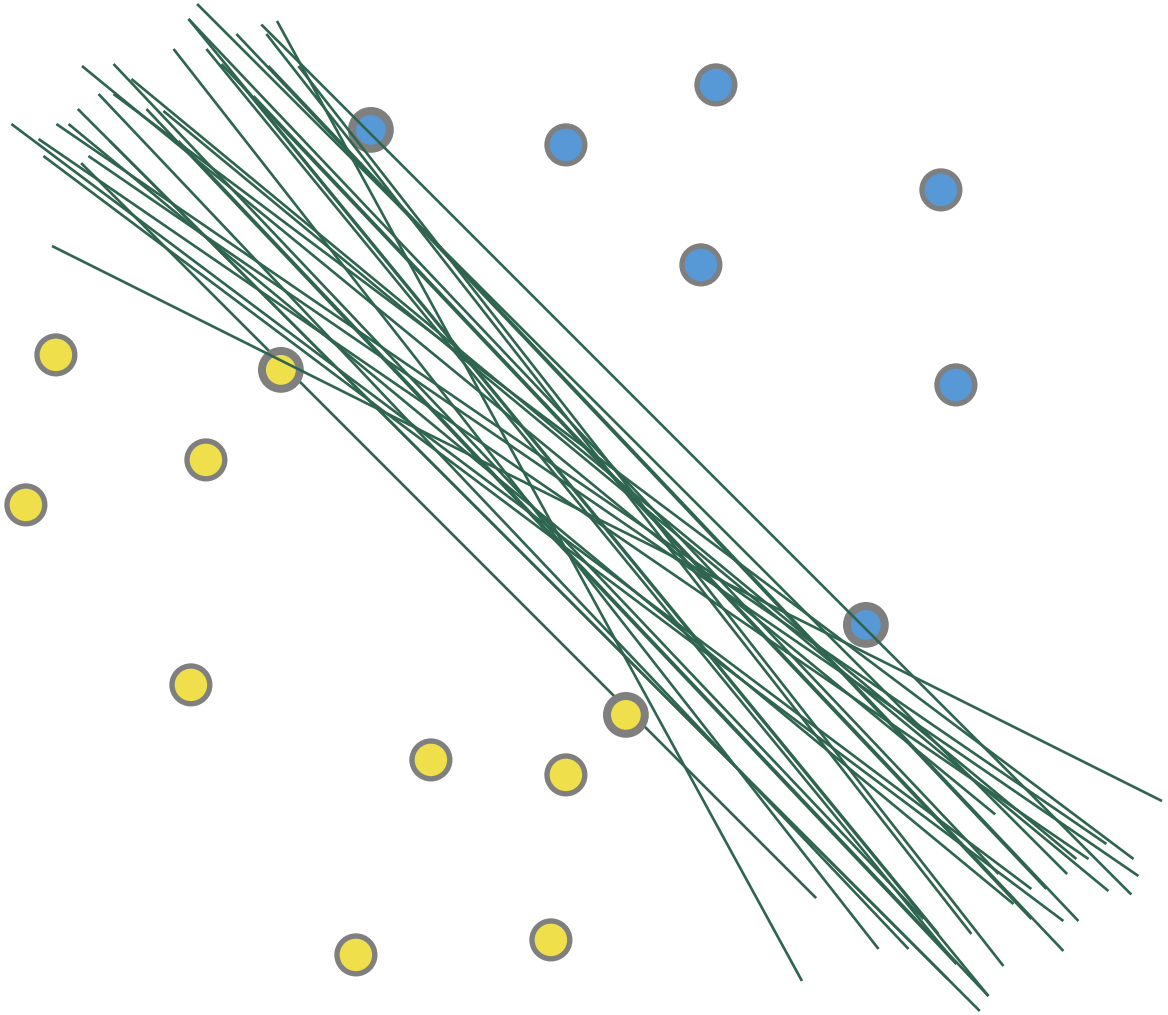
Prinzip der SRM:

1. Unterteilung der Funktionen-Klasse f(α) in verschachtelte Untermengen S_i.
2. Berechnung von h für jede Untermenge oder Ermittlung einer Schranke für h
3. Trainiere für jede Untermenge eine Reihe von LMs, deren einfaches Ziel die Minimierung von $R_{emp}(\alpha)$ ist.
4. Wähle aus der Reihe von LMs diejenige, deren Summe aus $R_{emp}(\alpha)$ und VC-Konfidenz minimal ist.

Kapitel zum SVM-Tutorial

1. Einführungen zur SVM
2. Grundlagen zur SVM
3. Lineare SVMs
4. Nicht-lineare SVMs
5. Fazit
6. SVM in Action!

❖ Lernen auf trennbaren Daten

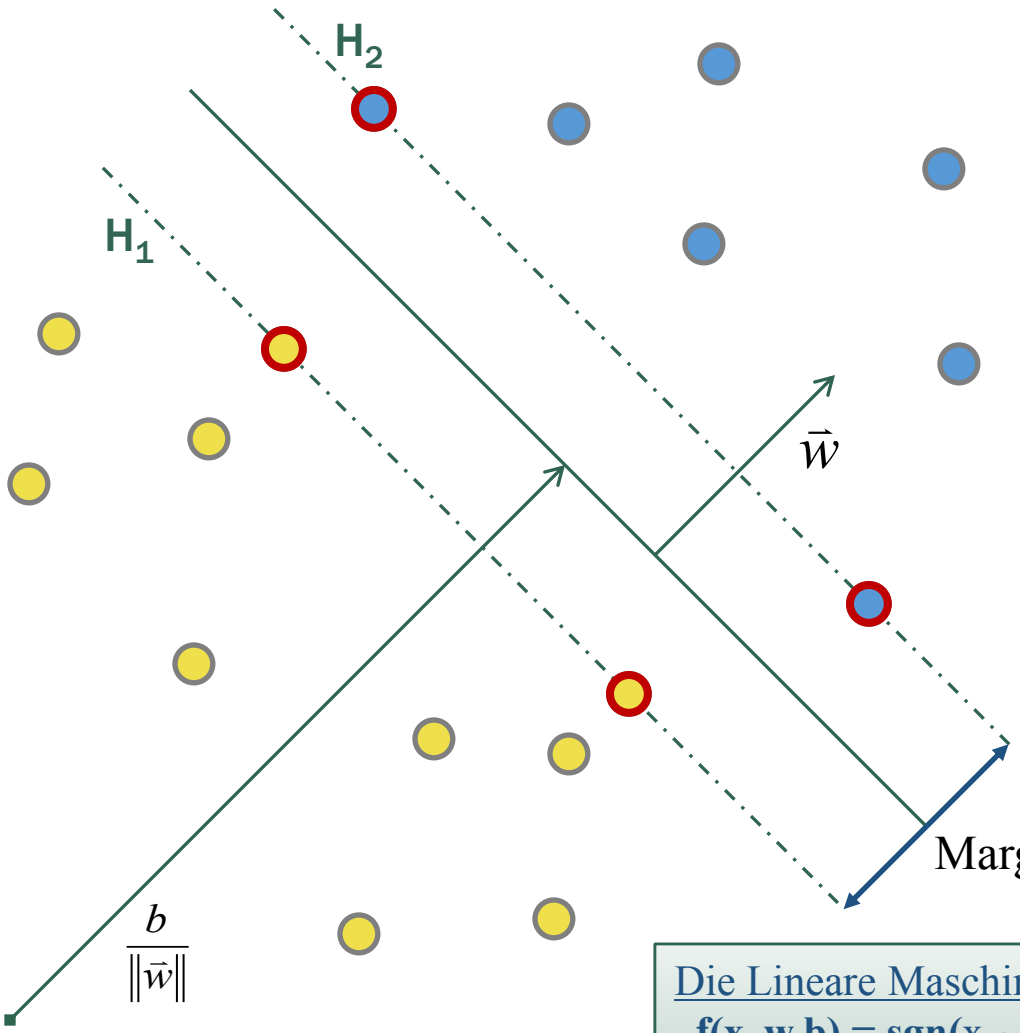


Trainings-Daten:
$\{x_i, y_i\}$, i=1,..,l
mit $x_i \in R^d$
$y_i \in \{-1,1\}$
Beispiel:
$x_i \in R^2$
●: $y_i = -1$
○: $y_i = 1$

Trennende Hyper-Ebene H
$x_i \cdot w + b = 0$
○ Punkte zw. 0 und H:
$x_i \cdot w + b \geq 0$
● Punkte „jenseits“ von H:
$x_i \cdot w + b \leq 0$
.

Welche ist die richtige Hyper-Ebene und wie kann die LM sie finden?

3. Lineare SVMs



Trainings-Daten:
$\{x_i, y_i\}$, i=1,..,l
mit $x_i \in R^d$
$y_i \in \{-1,1\}$
Beispiel:
$x_i \in R^2$
●: $y_i = -1$
○: $y_i = 1$

Hyper-Ebenen
H_1: $x_i \cdot w + b = 1$
H_2: $x_i \cdot w + b = -1$

Skalarprodukt: $\langle x_i, w \rangle$

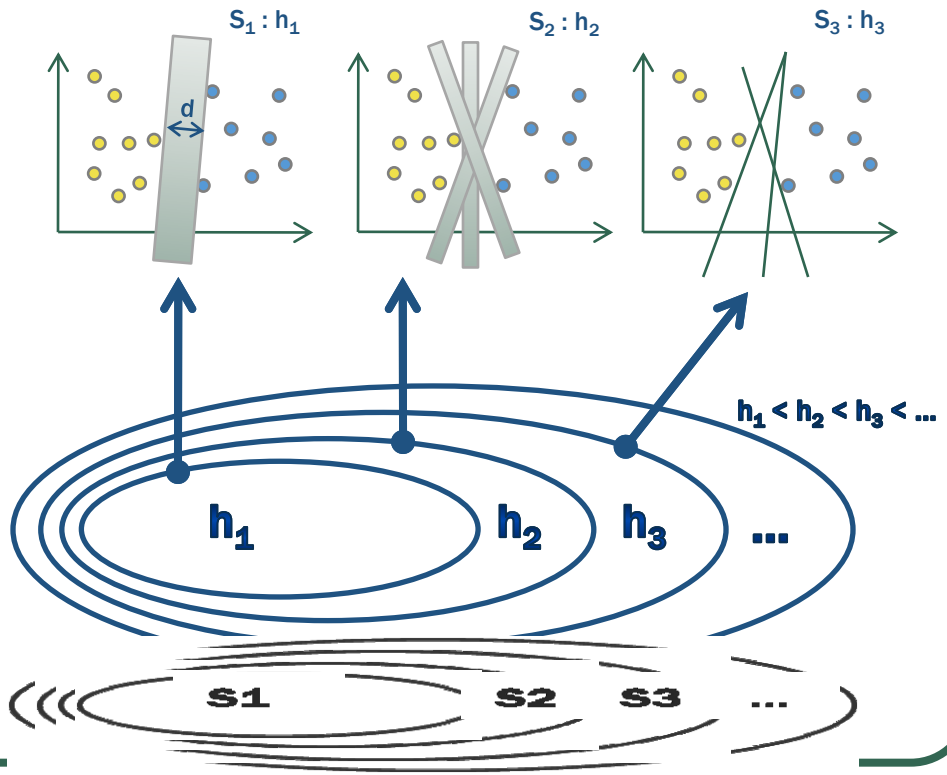
Die Lineare Maschine soll also lernen:
$f(x_i,w,b) = sgn(x_i \cdot w + b)$
so,dass alle Punkte korrekt klassifiziert werden

❖ Entscheidungs-Grenze mit größtem Margin

Trennende Hyperebene mit maximalem Rand ⟷ **Trennende Hyperebene mit minimaler VC-Dimension h**

Prinzip der SRM:

- Geht man von einem linear separierbarem Zweiklassenproblem aus, so erreichen alle Geraden, welche beide Klassen trennen einen empirischen Fehler Null
- Die VC-Konfidenz wird minimiert durch ein Polynom minimaler VC-Dimension h, nämlich einer Hyperebene
- Die VC-Dimension h kann weiter abgesenkt werden durch „breite Hyperebenen“ (**large margin hyperplanes**)



❖ Lernen auf trennbaren Daten

Eine lineare SVM klassifiziert Trainings-Daten, indem sie
- eine trennende Hyperebene derart findet,
- dass sie den Abstand zu den nächsten positiven / negativen Beispielen maximiert

Für $y_i = +1$ gilt: $x_i \cdot w + b \geq +1$

Für $y_i = -1$ gilt: $x_i \cdot w + b \leq -1$

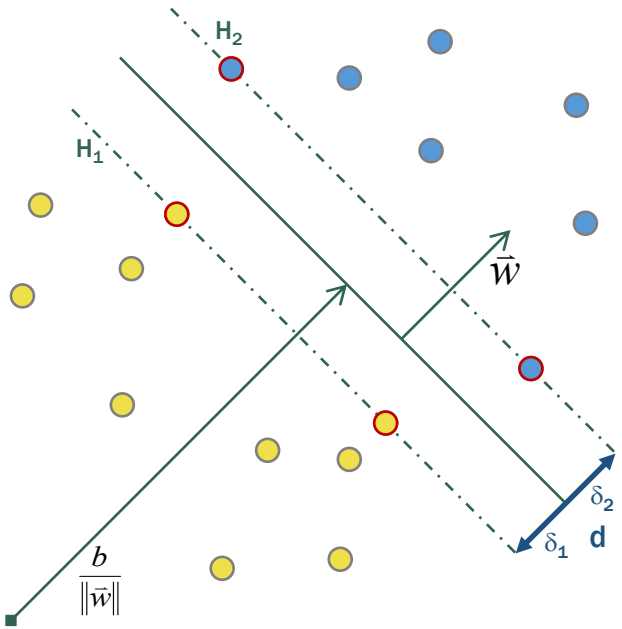
$$y_i(x_i \cdot w + b) - 1 \geq 0 \quad \forall i$$

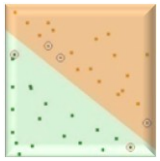
Zentrale Frage: Wie kann eine SVM diesen Margin maximieren?

- Der Abstand von 0 zur trennenden Hyper-Ebene ist: b / ||w||
- Der Abstand vom Ursprung zu H_1 ist: |b-1| / ||w||
- Der Abstand vom Ursprung zu H_2 ist: |b+1| / ||w||

$\Rightarrow$ Somit ist der Margin $d=\delta_1+\delta_1 = 2 / ||w||$

Somit können wir den **Margin maximieren**,
indem wir ||w|| **minimieren**.
Statt ||w|| können wir auch einfach ½ ||w||² minimieren.





❖ Optimierungsproblem mit Nebenbedingungen

Optimierungsproblem: ½ ||w||² minimieren unter der N.B.: $y_i(x_i \cdot w + b) - 1 \geq 0$ ∀i

Lagrange-Multiplikatoren-Regel:

Um die Nebenbedingungen in das Optimierungsproblem zu integrieren

1. wird für alle i Nebenbedingungen (≥0) je ein **Lagrange-Multiplikator $\alpha_i \geq 0$** eingeführt
2. α_i wird mit der Nebenbedingung (nur linke Seite) multipliziert $\alpha_i y_i (x_i \cdot w + b) - \alpha_i$
3. der jeweilige Lagrange-Term wird von der Zielfunktion subtrahiert, die wir L_p, nennen.

   Wir erhalten $$L_p \equiv \frac{1}{2}\|w\|^2 - \sum_{i=1}^{l} \alpha_i y_i (x_i \cdot w + b) + \sum_{i=1}^{l} \alpha_i$$

Für den Fall der Gleichheit in den N.B. sind die α_i unbeschränkt: $0 \leq \alpha_i \leq \infty$

Minimierung des **primalen Optimierungs-Problems L_p bezüglich w und b:**

$$\frac{\partial L_p}{\partial w} \overset{!}{=} 0 \Leftrightarrow w = \sum_{i=1}^{l} \alpha_i y_i x_i$$

$$\frac{\partial L_p}{\partial b} \overset{!}{=} 0 \Leftrightarrow \sum_{i=1}^{l} \alpha_i y_i = 0$$

Problem, denn die α_i müssen verschwinden

❖ Karush-Kuhn-Tucker-Bedingungen (KKT-Bedingungen)

Das primale Problem soll minimiert werden bezüglich w und b :

$$L_p \equiv \frac{1}{2}\|w\|^2 - \sum_{i=1}^{l} \alpha_i y_i (x_i \cdot w + b) + \sum_{i=1}^{l} \alpha_i \quad \forall i$$

Die so genannten **Karush-Kuhn-Tucker-Bedingungen**, als Verallgemeinerung der Lagrange-Multiplikatoren, sind **notwendige Bedingungen**, um eine optimale Lösung in der nichtlinearen Optimierung zu gewährleisten.

Es gibt eine optimale Lösung, da alle N.B. linear beschränkt sind.

Darüberhinaus sind die KKT-Bedingungen sogar **hinreichend**, da die Zielfunktion konvex ist, und alle N.B. einen konvexen Gültigkeitsbereich ergeben.

Antwort:
⇒Ja, es gibt eine optimale Lösung für w,b und α.

KKT-Bedingungen:

$$\frac{\partial}{\partial w_\nu} L_p = w_\nu - \sum_{i=1}^{l} \alpha_i y_i x_i = 0 \quad \text{mit } \nu = 1,\ldots,d \text{ (Dimension)}$$

$$\frac{\partial}{\partial b} L_p = \sum_{i=1}^{l} \alpha_i y_i = 0$$

$$y_i (x_i \cdot w + b) - 1 \geq 0 \quad \text{mit } i = 1,\ldots,l$$

$$\alpha_i \geq 0 \quad \forall i$$

$$\alpha_i (y_i (x_i \cdot w + b) - 1) = 0 \quad \forall i$$

❖ Duales Optimierungs-Problem

Ursprüngliches O.P.: ½ ||w||² minimieren unter der N.B.: $y_i(x_i \cdot w + b) - 1 \geq 0 \quad \forall i$

Primales Optimierungs-Problem:
Die Lagrange-Funktion des **primalen O.P. L_P bezüglich w und b minimeren;**
Gradienten mit α sollten verschwinden.

$$L_p \equiv \frac{1}{2}\|w\|^2 - \sum_{i=1}^{l} \alpha_i y_i (x_i \cdot w + b) + \sum_{i=1}^{l} \alpha_i$$

N.B.: $\vec{w} = \sum_{i=1}^{l} \alpha_i y_i x_i$ $\quad \sum_{i=1}^{l} \alpha_i y_i = 0$

⇒ Zu viele freie Parameter!

Duales Optimierungs-Problem:

Wir können daher das **duale O.P. L_D** formulieren:
Dazu setzen wir einfach den Ausdruck für Vektor w in L_P ein und können nun **L_D bezüglich α_i maximieren**, wobei nur noch Ableitungsterme mit α_i vorkommen.

$$L_D \equiv \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j x_i x_j$$

N.B.: $0 \leq \alpha_i \leq \infty$ $\quad \sum_{i=1}^{l} \alpha_i y_i = 0$

❖ Ermittlung der Stützvektoren (1/2)

$$L_D \equiv \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j x_i x_j$$

N.B.: $0 \le \alpha_i \le \infty$ $\quad \sum_{i=1}^{l} \alpha_i y_i = 0$

Mit $\frac{\partial L_D}{\partial \alpha_i} \overset{!}{=} 0$

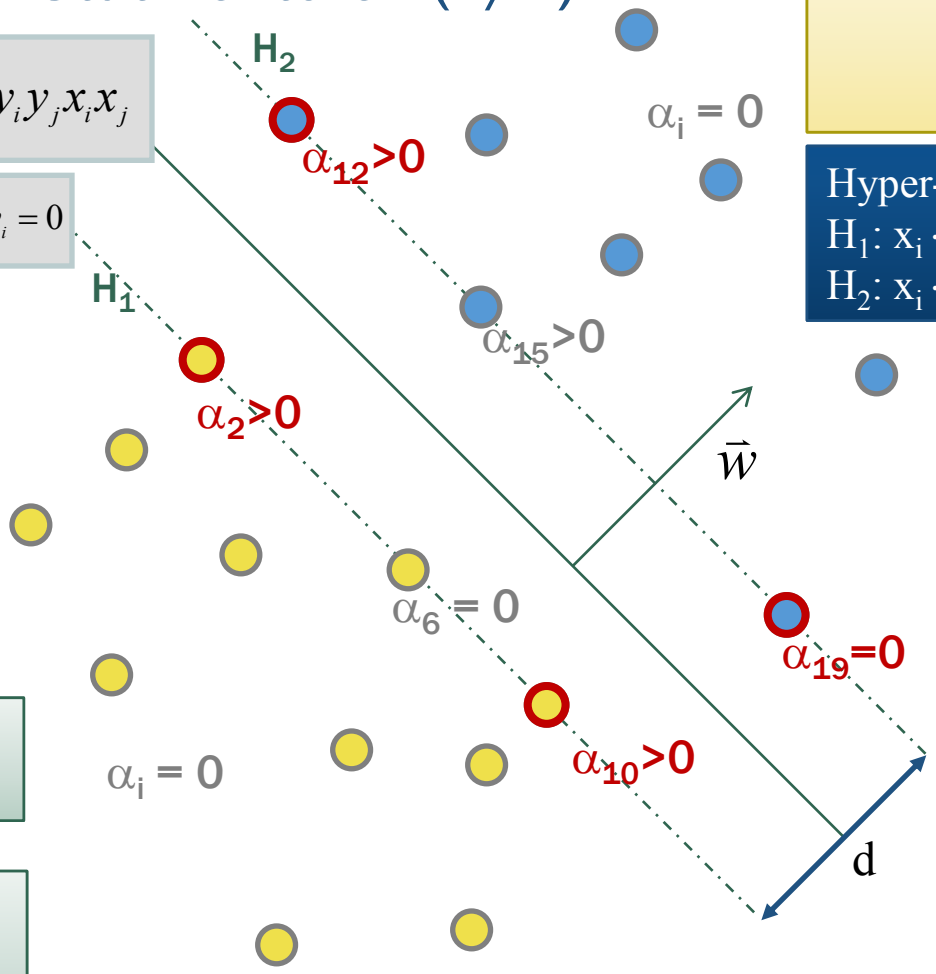
erhalten wir Lösungen für *l* α-Vektoren, mit:

$\alpha_i > 0$, also Stützvektoren (entweder auf H_1 oder H_2)

$\alpha_i = 0$, keine S.V. (auf H_1, H_2 oder in einer der beiden Halbräume)

Bspl.-Trainings-Daten:
$x_i \in R^2$
●: $y_i = -1$
○: $y_i = 1$

Hyper-Ebenen
H_1: $x_i \cdot w + b = 1$
H_2: $x_i \cdot w + b = -1$



❖ Ermittlung der Stützvektoren (2/2)

$$L_D \equiv \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j x_i x_j$$

N.B.: $0 \le \alpha_i \le \infty$ $\quad \sum_{i=1}^{l} \alpha_i y_i = 0$

$\vec{w}$ ist dann mithilfe des Vektors α lösbar:

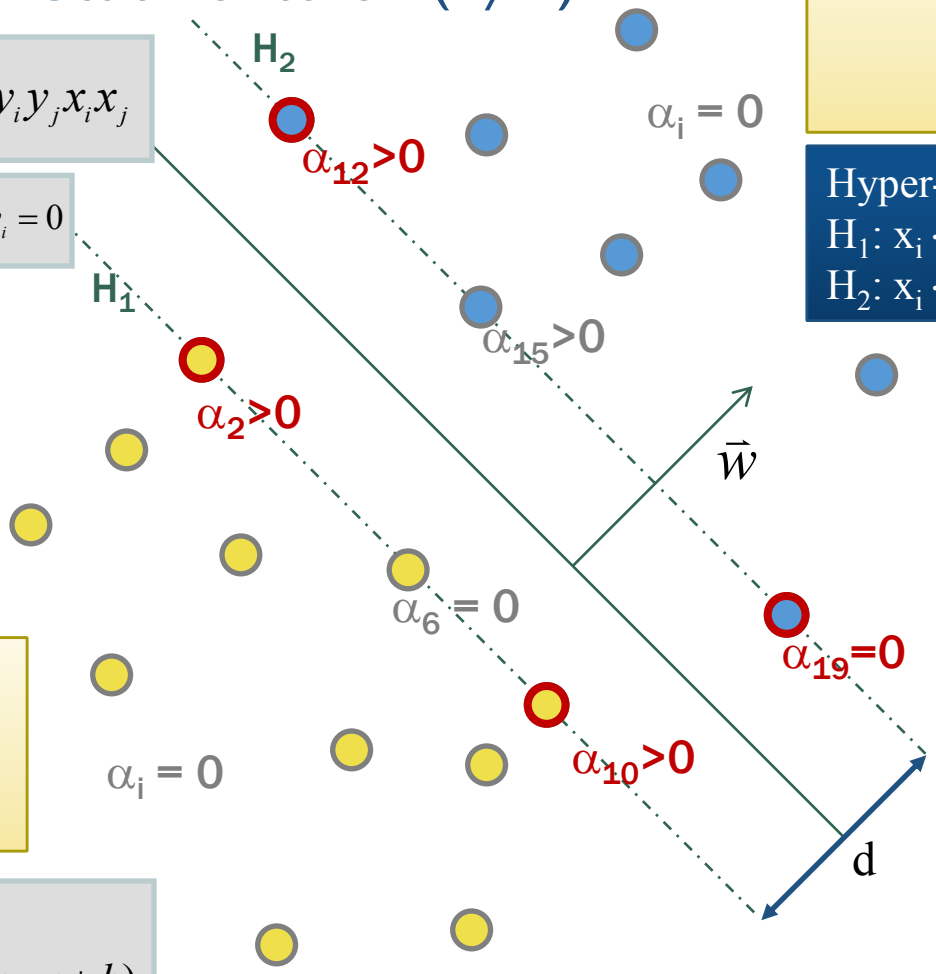
$$\vec{w} = \sum_{i=1}^{|SV|} \alpha_i y_i x_i$$

Damit gestaltet sich die **Entscheidungs-Grenze als Linear-Kombination von Stützvektoren**:

$$f(x, w, b) = \mathrm{sgn}(x \cdot w + b)$$
$$\Leftrightarrow f(x, w, b) = \mathrm{sgn}\left(\sum_{i \in SV} \alpha_i y_i x_i \cdot x + b\right)$$

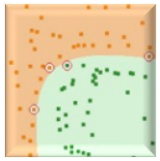
Bspl.-Trainings-Daten:
$x_i \in R^2$
●: $y_i = -1$
○: $y_i = 1$

Hyper-Ebenen
H_1: $x_i \cdot w + b = 1$
H_2: $x_i \cdot w + b = -1$

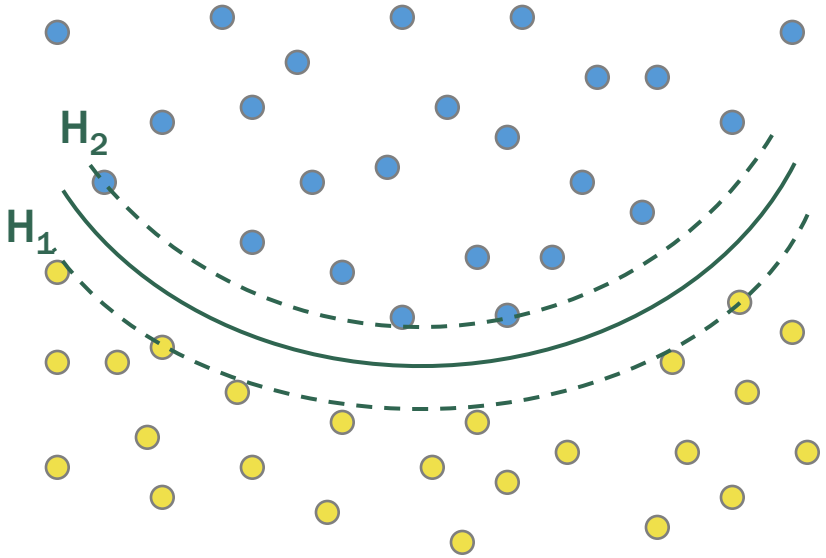


Kapitel zum SVM-Tutorial

1. Einführungen zur SVM
2. Grundlagen zur SVM
3. Lineare SVMs
4. Nicht-lineare SVMs
5. Fazit
6. SVM in Action!



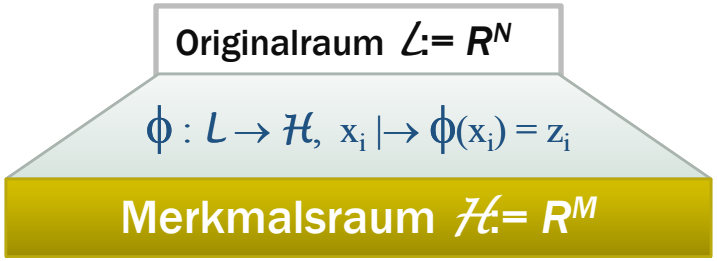
❖ Nicht-linear trennbare Daten



Trainings-Daten:
$\{x_i, y_i\}$, i=1,..,l
mit $x_i \in R^d$
$y_i \in \{-1,1\}$
Beispiel:
$x_i \in R^2$
●: $y_i = -1$
○: $y_i = 1$

Hyper-Ebene:
H_1: $\phi(x_i) \cdot w + b = 1$
H_2: $\phi(x_i) \cdot w + b = -1$

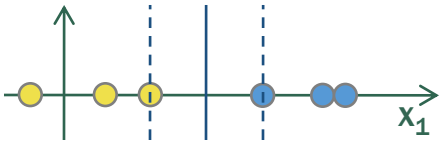
Verbiegen der Hyper-Ebenen?
→ Nein
Ausweg: Abbildung der Daten in einen höher-dimensionalen Raum:



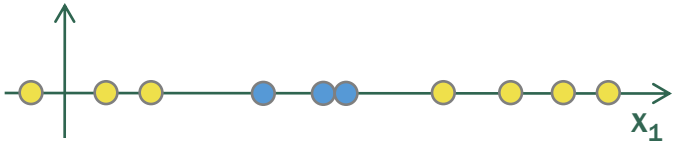
$dim\ \mathcal{H} = M >> N,\ M \leq \infty$

❖ Beispiel: Basis-Erweiterung für Daten im R^1

Separierbarer Fall:

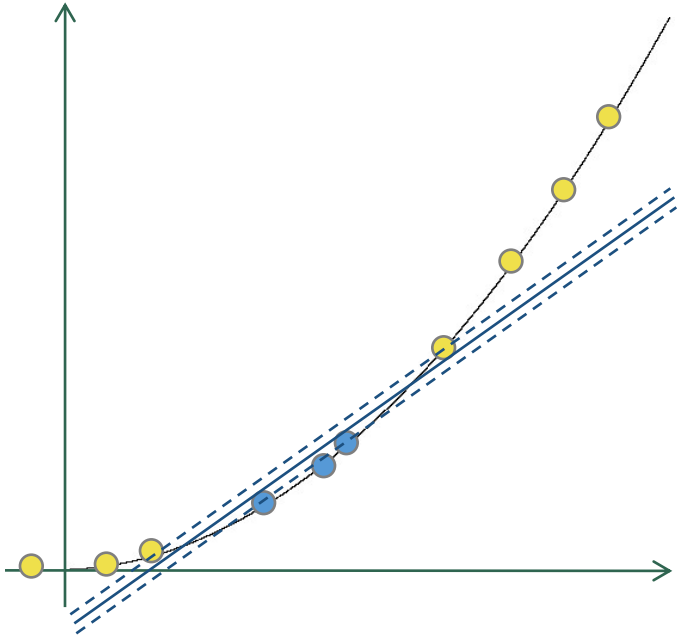


Linear nicht separierbar Fall:



Ausweg: Abbildung des Originalraums $\mathcal{L}$ in $\mathcal{H}=R^2$:

$$\Phi : R^1 \to R^2 \qquad x_1 \mapsto \Phi(x_1) = \vec{z} \qquad \vec{z} = \begin{pmatrix} z_1 = x_1 \\ z_2 = x_1^2 \end{pmatrix}$$



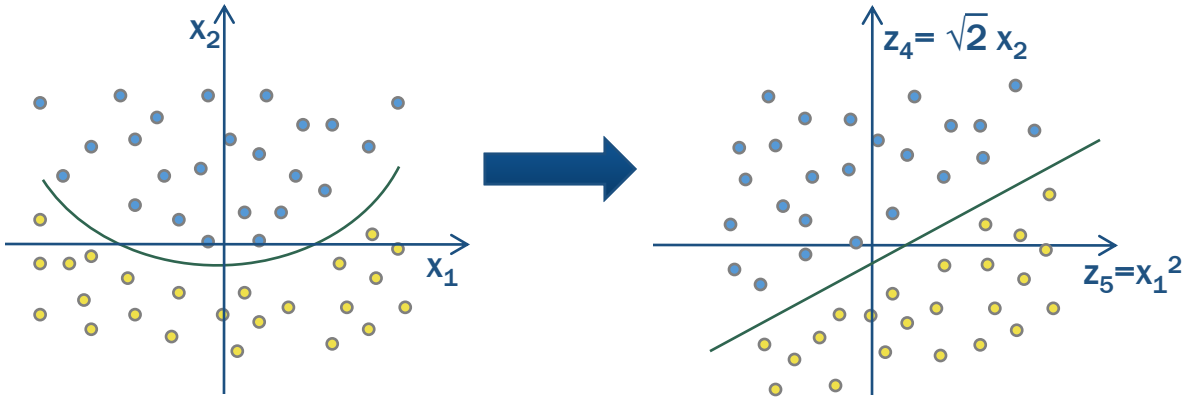
z_k = „polynomielle Terme von x_k vom Grad 1 und 2“

❖ Nicht-lineare Erweiterung im R^2

Original-Raum: $x=(x_1,x_2)$

Merkmalsraum $\mathcal{H}$:

$$\Phi(x)=\vec{z}=\begin{pmatrix} z_1=1 \\ z_2=\sqrt{2}x_1x_2 \\ z_3=\sqrt{2}x_1 \\ z_4=\sqrt{2}x_2 \\ z_5=x_1^2 \\ z_6=x_2^2 \end{pmatrix}$$



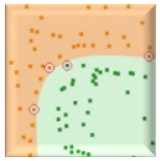
Nicht-Lineare Separation:

- $x_2 > x_1^2$
- $x_2 < x_1^2$

Lineare Separation:

- $z_4 > z_5$
- $z_4 < z_5$

Wie bauen wir diese Erweiterung in unsere SVM ein?



❖ Nicht-lineare Erweiterung im R²

Zu maximieren war ja das duale O.P.:

$$L_D \equiv \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j x_i x_j$$

N.B.: $0 \le \alpha_i \le C$ $\quad \sum_{i=1}^{l} \alpha_i y_i = 0$

Jetzt mit nicht-linearer Erweiterung: $\phi(x_i)$ für jedes x_i:

$$L_D \equiv \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j Q_{ij}$$

mit $Q_{ij} = y_i y_j (\Phi(x_i) \cdot \Phi(x_j))$

Codierung der Matrix Q_{ij} mit
½ l² Skalarprodukten
Wie viele Operationen benötigt ein Skalar-Produkt maximal?

Bei Auswahl von 2 Termen und Dimension m: $\binom{m+2}{2} = \frac{(m+2)(m+1)}{2} \approx m^2/2$

Mit Lösung:

$$\vec{w} = \sum_{i=1}^{|SV|} \alpha_i y_i x_i$$

$$f(x, w, b) = \operatorname{sgn}(x \cdot w + b)$$

Mit Lösung:

$$\vec{w} = \sum_{i=1}^{|SV|} \alpha_i y_i \Phi(x_i)$$

$$f(x, w, b) = \operatorname{sgn}(\Phi(x) \cdot w + b)$$

Menge aller möglicher quadratischer Terme:

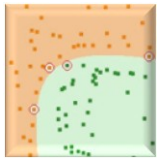
$$\Phi(x) = \begin{pmatrix} 1 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ \vdots \\ \sqrt{2}x_m \\ x_1^2 \\ x_2^2 \\ \vdots \\ x_m^2 \\ \sqrt{2}x_1x_2 \\ \sqrt{2}x_1x_3 \\ \vdots \\ \sqrt{2}x_1x_m \\ \sqrt{2}x_2x_3 \\ \vdots \\ \sqrt{2}x_2x_m \\ \vdots \\ \sqrt{2}x_{m-1}x_m \end{pmatrix}$$

konstante Terme

lineare Terme

reine quadratische Terme

Gemischte quadratische Terme



❖ Nicht-lineare Erweiterung im R²

Zu maximieren war ja das duale O.P.:

$$L_D \equiv \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j y_i y_j x_i x_j$$

N.B.: $0 \le \alpha_i \le C$ $\quad \sum_{i=1}^{l} \alpha_i y_i = 0$

Jetzt mit nicht-linearer Erweiterung: $\phi(x_i)$ für jedes x_i:

$$L_D \equiv \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} \alpha_i \alpha_j Q_{ij}$$

mit $Q_{ij} = y_i y_j (\Phi(x_i) \cdot \Phi(x_j))$

Matrix mit ½ l^2 *Skalarprodukten*
Jedes Sk.Prod. benötigt $m^2/2$ Operationen
=> Insgesamt: ¼ l^2 m^2 Operationen!!

Argh!

Bei Auswahl von 2 Termen und Dimension m:

$$\binom{m+2}{2} = \frac{(m+2)(m+1)}{2} \approx m^2/2$$

Mit Lösung:

$$\vec{w} = \sum_{i=1}^{|SV|} \alpha_i y_i x_i$$

$$f(x, w, b) = \operatorname{sgn}(x \cdot w + b)$$

Mit Lösung:

$$\vec{w} = \sum_{i=1}^{|SV|} \alpha_i y_i \Phi(x_i)$$

$$f(x, w, b) = \operatorname{sgn}(\Phi(x) \cdot w + b)$$

Menge aller möglicher quadratischer Terme:

$$\Phi(x) = \begin{pmatrix} 1 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ \vdots \\ \sqrt{2}x_m \\ x_1^2 \\ x_2^2 \\ \vdots \\ x_m^2 \\ \sqrt{2}x_1x_2 \\ \sqrt{2}x_1x_3 \\ \vdots \\ \sqrt{2}x_1x_m \\ \sqrt{2}x_2x_3 \\ \vdots \\ \sqrt{2}x_2x_m \\ \vdots \\ \sqrt{2}x_{m-1}x_m \end{pmatrix}$$

konstante Terme

lineare Terme

reine quadratische Terme

Gemischte quadratische Terme

❖ „Kernel-Trick“ – hier: polynomielle Kernel-Funktionen

$$\Phi(a)\cdot\Phi(b) = \begin{pmatrix} 1 \\ \sqrt{2}a_1 \\ \sqrt{2}a_2 \\ \vdots \\ \sqrt{2}a_m \\ a_1^2 \\ a_2^2 \\ \vdots \\ a_m^2 \\ \sqrt{2}a_1a_2 \\ \sqrt{2}a_1a_3 \\ \vdots \\ \sqrt{2}a_1a_m \\ \sqrt{2}a_2a_3 \\ \vdots \\ \sqrt{2}a_2a_m \\ \vdots \\ \sqrt{2}a_{m-1}a_m \end{pmatrix} \cdot \begin{pmatrix} 1 \\ \sqrt{2}b_1 \\ \sqrt{2}b_2 \\ \vdots \\ \sqrt{2}b_m \\ b_1^2 \\ b_2^2 \\ \vdots \\ b_m^2 \\ \sqrt{2}b_1b_2 \\ \sqrt{2}b_1b_3 \\ \vdots \\ \sqrt{2}b_1b_m \\ \sqrt{2}b_2b_3 \\ \vdots \\ \sqrt{2}b_2b_m \\ \vdots \\ \sqrt{2}b_{m-1}b_m \end{pmatrix} \begin{matrix} 1 \\ + \\ \sum_{i=1}^{m} 2a_ib_i \\ + \\ \sum_{i=1}^{m} a_i^2b_i^2 \\ + \\ \sum_{i=1}^{m}\sum_{j=i+1}^{m} 2a_ia_jb_ib_j \end{matrix}$$

allgemein auf m Dimensionen

$$\Phi(a)\cdot\Phi(b) = \sum_{i=1}^{m}\sum_{j=i+1}^{m} 2a_ia_jb_ib_j + \sum_{i=1}^{m} a_i^2b_i^2 + \sum_{i=1}^{m} 2a_ib_i + 1$$

$$= \sum_{i=1}^{m}\sum_{j=1}^{m} a_ia_jb_ib_j + \sum_{i=1}^{m} 2a_ib_i + 1$$

$$= \left(\sum_{i=1}^{m} a_ib_i\right)^2 + \sum_{i=1}^{m} 2a_ib_i + 1$$

$$= (a\cdot b)^2 + 2a\cdot b + 1$$

$$= (a\cdot b + 1)^2$$

Ha! Eine quadratische Funktion über ein Skalarprodukt!
Statt also, wie bisher $O(n^2)$ Operationen, benötigen wir jetzt nur noch $O(n)$!

$$L_D \equiv \sum_{i=1}^{l}\alpha_i - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_jQ_{ij}$$

$$Q_{ij} = y_iy_j(\Phi(x_i)\cdot\Phi(x_j))$$

Wir benötigen keine expliziten ϕ-Werte mehr, sondern lediglich die Kernel-Funktion $K(x_1,x_2) = \phi(x_1) \bullet \phi(x_2)$
Wir können also das Skalar-Produkt zwischen den x_i‘s direkt berechnen!

❖ „Kernel-Trick“ – hier: polynomielle Kernel-Funktionen

$$\Phi(a)\cdot\Phi(b) = \begin{pmatrix} 1 \\ \sqrt{2}a_1 \\ \sqrt{2}a_2 \\ \vdots \\ \sqrt{2}a_m \\ a_1^2 \\ a_2^2 \\ \vdots \\ a_m^2 \\ \sqrt{2}a_1a_2 \\ \sqrt{2}a_1a_3 \\ \vdots \\ \sqrt{2}a_1a_m \\ \sqrt{2}a_2a_3 \\ \vdots \\ \sqrt{2}a_2a_m \\ \vdots \\ \sqrt{2}a_{m-1}a_m \end{pmatrix} \cdot \begin{pmatrix} 1 \\ \sqrt{2}b_1 \\ \sqrt{2}b_2 \\ \vdots \\ \sqrt{2}b_m \\ b_1^2 \\ b_2^2 \\ \vdots \\ b_m^2 \\ \sqrt{2}b_1b_2 \\ \sqrt{2}b_1b_3 \\ \vdots \\ \sqrt{2}b_1b_m \\ \sqrt{2}b_2b_3 \\ \vdots \\ \sqrt{2}b_2b_m \\ \vdots \\ \sqrt{2}b_{m-1}b_m \end{pmatrix} \begin{matrix} 1 \\ + \\ \sum_{i=1}^{m} 2a_ib_i \\ + \\ \sum_{i=1}^{m} a_i^2b_i^2 \\ + \\ \sum_{i=1}^{m}\sum_{j=i+1}^{m} 2a_ia_jb_ib_j \end{matrix}$$

$$\Phi(a)\cdot\Phi(b) = \sum_{i=1}^{m}\sum_{j=i+1}^{m} 2a_ia_jb_ib_j + \sum_{i=1}^{m} a_i^2b_i^2 + \sum_{i=1}^{m} 2a_ib_i + 1$$

$$= \sum_{i=1}^{m}\sum_{j=1}^{m} a_ia_jb_ib_j + \sum_{i=1}^{m} 2a_ib_i + 1$$

$$= \left(\sum_{i=1}^{m} a_ib_i\right)^2 + \sum_{i=1}^{m} 2a_ib_i + 1$$

$$= (a\cdot b)^2 + 2a\cdot b + 1$$

$$= (a\cdot b + 1)^2$$

Ha! Eine quadratische Funktion über ein Skalarprodukt!
Statt also, wie bisher $O(n^2)$ Operationen, benötigen wir jetzt nur noch $O(n)$!

$$L_D \equiv \sum_{i=1}^{l}\alpha_i - \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_jQ_{ij}$$

$$Q_{ij} = y_iy_j(x_i\cdot x_j + 1)^2$$

Wir benötigen keine expliziten ϕ-Werte mehr, sondern lediglich die Kernel-Funktion $K(x_1,x_2) = \phi(x_1) \bullet \phi(x_2)$
Wir können also das Skalar-Produkt zwischen den x_i‘s direkt berechnen!

❖ „Kernel-Trick“ – hier: polynomielle Kernel-Funktionen

$$\Phi(a)\cdot\Phi(b)=\begin{pmatrix}1\\ \sqrt{2}a_1\\ \sqrt{2}a_2\\ \vdots\\ \sqrt{2}a_m\\ a_1^2\\ a_2^2\\ \vdots\\ a_m^2\\ \sqrt{2}a_1a_2\\ \sqrt{2}a_1a_3\\ \vdots\\ \sqrt{2}a_1a_m\\ \sqrt{2}a_2a_3\\ \vdots\\ \sqrt{2}a_2a_m\\ \vdots\\ \sqrt{2}a_{m-1}a_m\end{pmatrix}\cdot\begin{pmatrix}1\\ \sqrt{2}b_1\\ \sqrt{2}b_2\\ \vdots\\ \sqrt{2}b_m\\ b_1^2\\ b_2^2\\ \vdots\\ b_m^2\\ \sqrt{2}b_1b_2\\ \sqrt{2}b_1b_3\\ \vdots\\ \sqrt{2}b_1b_m\\ \sqrt{2}b_2b_3\\ \vdots\\ \sqrt{2}b_2b_m\\ \vdots\\ \sqrt{2}b_{m-1}b_m\end{pmatrix}\begin{matrix}1\\ +\\ \sum_{i=1}^{m}2a_ib_i\\ +\\ \sum_{i=1}^{m}a_i^2b_i^2\\ +\\ \sum_{i=1}^{m}\sum_{j=i+1}^{m}2a_ia_jb_ib_j\end{matrix}$$

$$\Phi(a)\cdot\Phi(b)=\sum_{i=1}^{m}\sum_{j=i+1}^{m}2a_ia_jb_ib_j+\sum_{i=1}^{m}a_i^2b_i^2+\sum_{i=1}^{m}2a_ib_i+1$$

$$=\sum_{i=1}^{m}\sum_{j=1}^{m}a_ia_jb_ib_j+\sum_{i=1}^{m}2a_ib_i+1$$

$$=\left(\sum_{i=1}^{m}a_ib_i\right)^2+\sum_{i=1}^{m}2a_ib_i+1$$

$$=(a\cdot b)^2+2a\cdot b+1$$

$$=(a\cdot b+1)^2$$

$$L_D\equiv\sum_{i=1}^{l}\alpha_i-\frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_jQ_{ij}$$

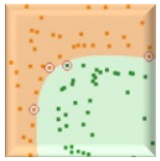
$$Q_{ij}=y_iy_j(x_i\cdot x_j+1)^2$$

Die **Lern-Funktion** ist dann:

$$f(x,w,b)=\mathrm{sgn}(\Phi(x)\cdot w+b)$$

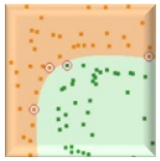
$$=\mathrm{sgn}\left(\sum_{i=1}^{|SV|}\alpha_iy_i\Phi(x_i)\cdot\Phi(x)+b\right)$$

$$=\mathrm{sgn}\left(\sum_{i=1}^{|SV|}\alpha_iy_iK(x_i,x)+b\right)$$



❖ Kostenvergleich – Direkte Berechnung vs. Kernel-Funktion bei Polynomen höherer Ordnung

Polynom	$\phi(x)$	Kosten der Matrix Q_{ij} (Berechnung einzelner ϕ)	Kosten bei Dimension 100	$K(x_1,x_2) = \phi(x_1) \cdot \phi(x_2)$	Kosten der Matrix Q_{ij} mit Kernel-Fkt.	Kosten bei Dimension 100
quadrat.	$m^2/2$	$l^2m^2/4$	2500 l^2	$(x_1 \cdot x_2+1)^2$	$l^2m/2$	50 l^2
kubisch	$m^3/6$	$l^3m^3/12$	83000 l^3	$(x_1 \cdot x_2+1)^3$	$l^3m/2$	50 l^2
quartisch	$m^4/24$	$l^4m^4/48$	$1{,}96*10^6*l^4$	$(x_1 \cdot x_2+1)^4$	$l^4m/2$	50 l^2



❖ Andere Kernel-Funktionen

Eine kleine Übersicht über andere bekannte Kernel-Funktionen:

Lineare Kernel: $K(x_1,x_2) = \langle \phi(x_1), \phi(x_2) \rangle = \langle x_1,x_2 \rangle$

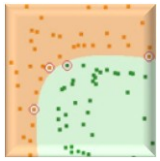
Polynomielle Kernel: $K(x_1,x_2) = \langle \phi(x_1), \phi(x_2) \rangle = \langle x_1,x_2 \rangle^d$, Grad d

Gauss / RBF-Kernel: $K(x_1,x_2) = \exp(-\frac{\|x_1 - x_2\|^2}{2\sigma^2})$

Neuronal-Netz-Kernel / Sigmoid – Kernel: $K(x_1,x_2) = \tanh(\kappa\, x_1 \cdot x_2 - \delta)$

Kapitel zum SVM-Tutorial

1. Einführungen zur SVM
2. Grundlagen zur SVM
3. Lineare SVMs
4. Nicht-lineare SVMs
5. Fazit
6. SVM in Action!



❖ Zusammenfassung:

Was haben wir gelernt?

- Maximieren der Margin einer separierenden Hyper-Ebene (***large margin hyperplanes***) ergibt optimale trennende Hyper-Ebene

- Erweiterung des Originalraums auf den Merkmalsraum behandelt Nichtlinearität

- Die SVM kann ***trotz möglichem unendlichem Raum*** mithilfe der Kernel-Funktionen mit sehr guter Performanz klassifizieren.

- Der erwartete Testfehler ist beschränkt und hängt nicht von der Dimensionalität des Raumes ab.

Kapitel zum SVM-Tutorial

1. Einführungen zur SVM
2. Grundlagen zur SVM
3. Lineare SVMs
4. Nicht-lineare SVMs
5. Fazit
6. SVM in Action!

Vielen Dank für Ihre Aufmerksamkeit!