# Support Vector Machines (SVM)

Jasmin Fischer

Universität Ulm

12. Juni 2007

# Inhalt

# Grundlagen

**Ausgangslage:**

N Trainingsdaten $(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)$

mit $x_i \in \mathbb{R}^g$

und $y_i \in \{+1, -1\}$ (i=1,2,...,N)

**Idee:**

Unterteile eine Menge von Objekten durch eine Hyperebene in zwei Klassen

# Grundlagen

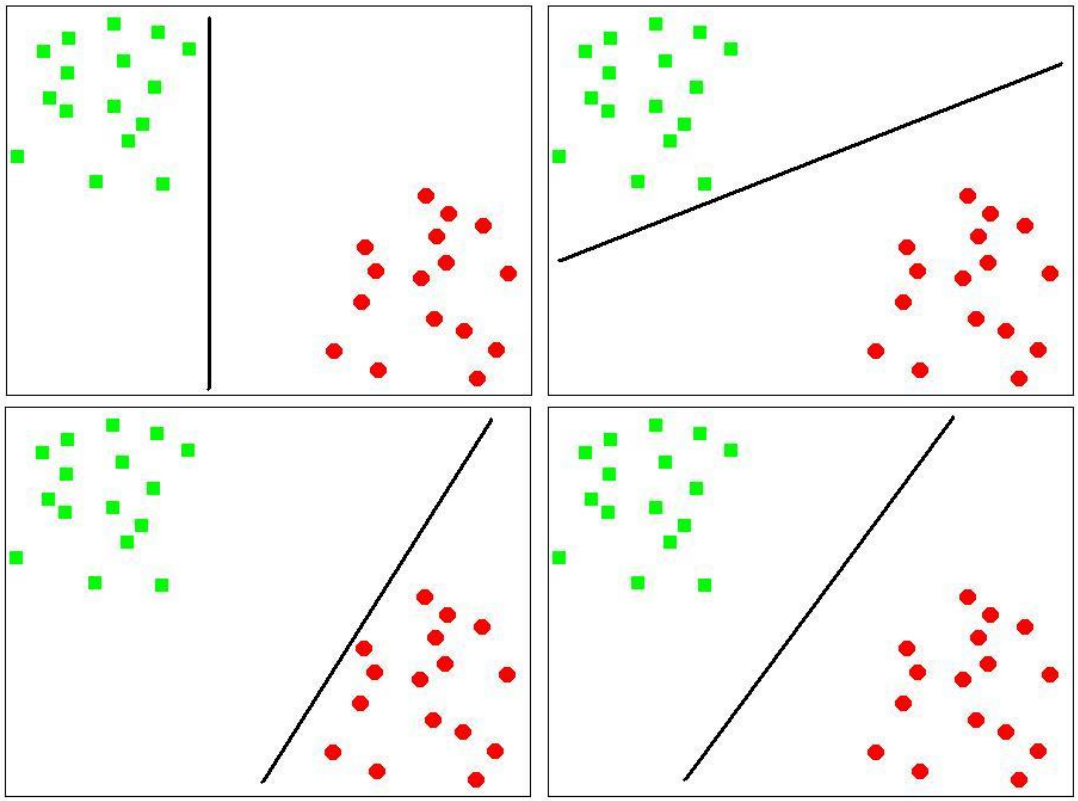
## Vorgehensweise

1. Suche $f : \mathbb{R}^g \rightarrow \{-1, +1\}$, so dass $f(x_i) = y_i$
   - im Fall der Trennbarkeit $\quad \forall\ i = 1, ..., N$
   - sonst für zumindest „viele“ i

   erfüllt ist.
2. Klassen-Zuordnung neuer Punkte $x_{neu}$ durch $f(x_{neu})$

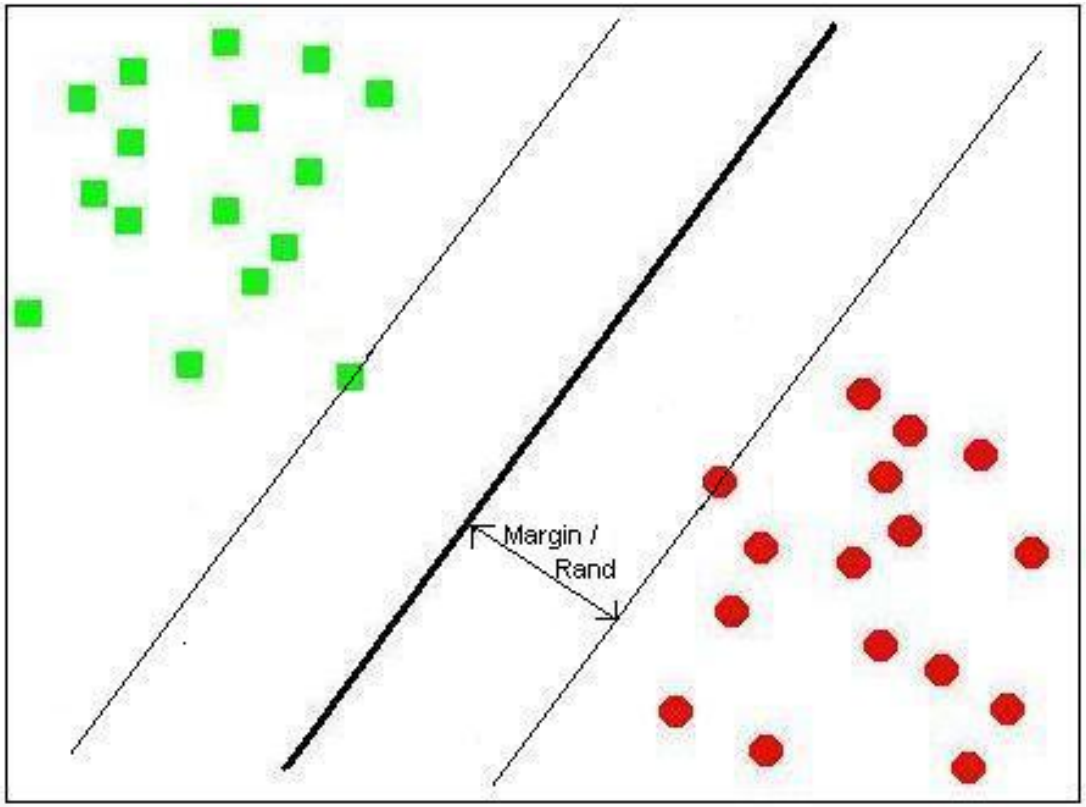
# Fragestellung

**Voraussetzung:** Die Trainingsdaten sind linear trennbar.
**Aber:** Wie genau ist die Hyperebene zu wählen?



# Idee

Erhalte einen möglichst breiten Rand um die Klassengrenzen herum
$\Rightarrow$ „*large - margin - classification*“



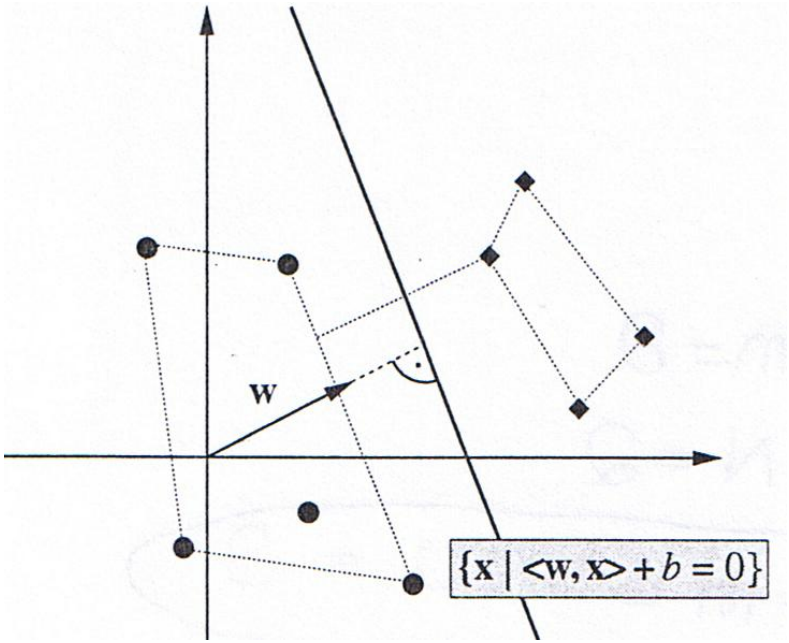
# Definition der Hyperebene

Eine **trennende Hyperebene** $\mathcal{H}$ ist folgendermaßen definiert:

$$\mathcal{H} := \{x \in \mathbb{R}^g | \langle w, x \rangle + b = 0\}$$

mit den bestimmenden Elementen

-w $\in \mathbb{R}^g$ orthogonal zu $\mathcal{H}$
-b $\in \mathbb{R}$ (Verschiebung)



# Trennende Hyperebene - Skalierung

**Problem:** Keine eindeutige Beschreibung der Hyperebene:

$$\mathcal{H} = \{x \in \mathbb{R}^g | \langle aw, x \rangle + ab = 0\} \quad \forall\, a \in \mathbb{R} \setminus \{0\}$$

$\Rightarrow$ Ausweg durch **Skalierung**:

$(w, b) \in \mathbb{R}^g \times \mathbb{R}$ heißt **kanonische Form der Hyperebene**, wenn gilt:

$$\min_{i=1,..,N} |\langle w, x_i \rangle + b| = 1$$

# Definition Rand

Als *Rand* bezeichnet man nun den Abstand der kanonischen Hyperebene zu dem Punkt, der ihr am nächsten liegt.

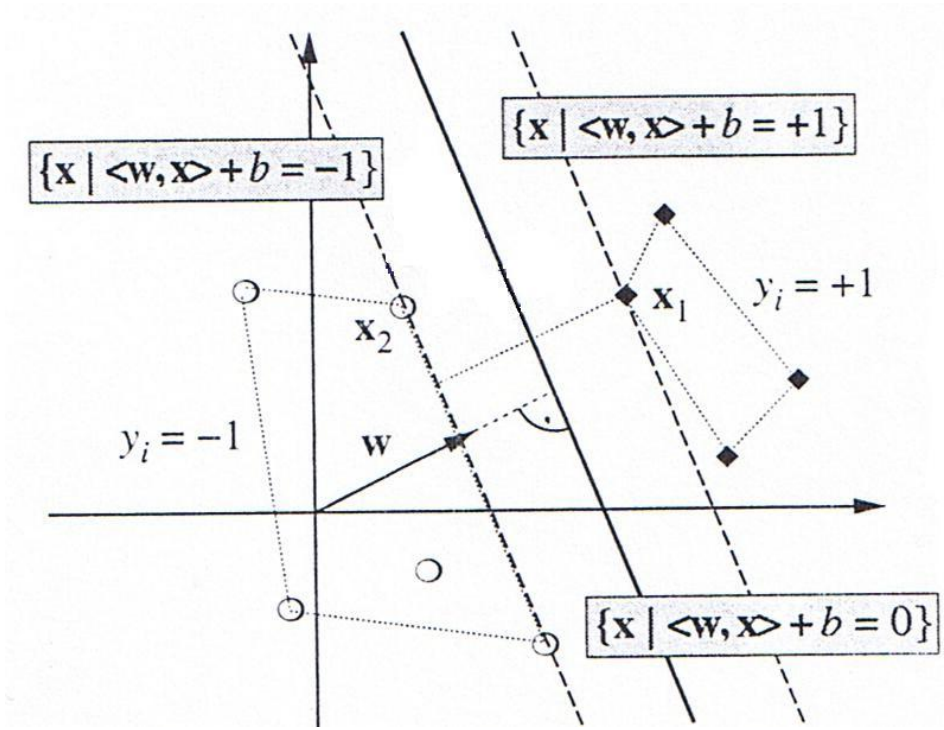
Er lässt sich zu $\frac{1}{\|w\|}$ berechnen.
Beweis:

$$\langle w, x_1 \rangle + b = +1$$

$$\langle w, x_2 \rangle + b = -1$$

$$\Rightarrow \langle w, (x_1 - x_2) \rangle = 2$$

$$\Rightarrow \langle \frac{w}{\|w\|}, (x_1 - x_2) \rangle = \frac{2}{\|w\|}$$



# Das Optimierungsproblem

**Aufgabenstellung:**

- maximiere den Rand $\leftrightarrow$ minimiere $\|w\|^2$
- die Entscheidungsfunktion $f(x) = sgn(\langle w, x\rangle + b)$ erfülle

$$f(x_i) = y_i \iff y_i(\langle x_i, w\rangle + b) \geq 1 \quad \forall\ i = 1, ..., N$$

**Primales Programm:**

$$\underset{w\in\mathbb{R}^g, b\in\mathbb{R}}{\text{minimiere}} \quad \frac{1}{2}\|w\|^2$$

NB: $y_i(\langle x_i, w\rangle + b) \geq 1 \quad \forall\ i = 1, ..., N$

# Lagrange-Funktion

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{N} \alpha_i (y_i(\langle x_i, w \rangle + b) - 1)$$

mit $\alpha = (\alpha_1, ..., \alpha_N)$ und $\alpha_i \geq 0$ (Lagrange Multiplikatoren)

- bezüglich $\alpha$ zu maximieren
- bezüglich w und b zu minimieren, d.h.

$$\frac{\partial}{\partial b} L(w, b, \alpha) = 0, \quad \frac{\partial}{\partial w} L(w, b, \alpha) = 0$$

Damit folgt

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \quad \text{und} \quad w = \sum_{i=1}^{N} \alpha_i y_i x_i$$

# Definition: Support Vektoren

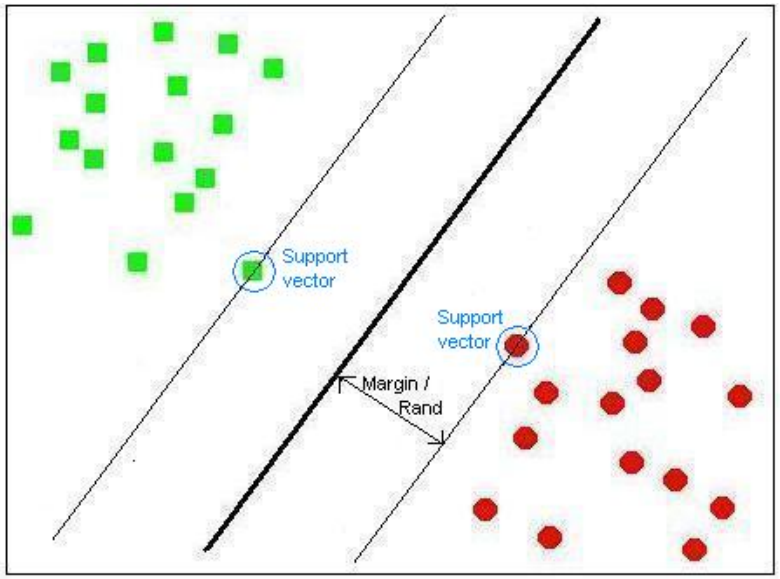
Sattelpunkt-Bedingungen laut Kuhn-Tucker:

$$\alpha_i[y_i(\langle x_i, w\rangle + b) - 1] = 0 \quad \forall\ i = 1, ..., N$$

Damit gilt:

- Punkte mit $\alpha_i > 0$ liegen direkt auf dem Rand.
  (*Support Vectors („Stützvektoren“)*)
- Die restlichen Trainingspunkte haben keinen Einfluss auf $\mathcal{H}$ ($\alpha_i = 0$)

$$\Rightarrow \quad w = \sum_{\{i \in \{1,...,N\} : x_i \ \textit{Support vector}\}} \alpha_i y_i x_i$$



# Das Duale Programm

Zugehöriges duales Programm:

$$\underset{\alpha \in \mathbb{R}^N}{\text{maximiere}} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle$$

unter den Bedingungen

$$\begin{aligned} \alpha_i &\geq 0 \quad \forall\ i = 1, ..., N \\ \sum_{i=1}^{N} \alpha_i y_i &= 0 \end{aligned}$$

# Vorgehensweise einer SVM

1. Berechne die Lagrange-Multiplikatoren $\alpha_i$ der Support Vektoren durch das duale Programm
2. Bestimme damit den Vektor $w = \sum_{i=1}^{N} \alpha_i y_i x_i$ der kanonischen Hyperebene
3. Die Verschiebung ergibt sich zu $b = y_j - \sum_{i=1}^{N} y_i \alpha_i \langle x_j, x_i \rangle$
4. Stelle die gesuchte Entscheidungsfunktion $f(x) = sgn(\langle w, x \rangle + b)$ folgendermaßen auf:

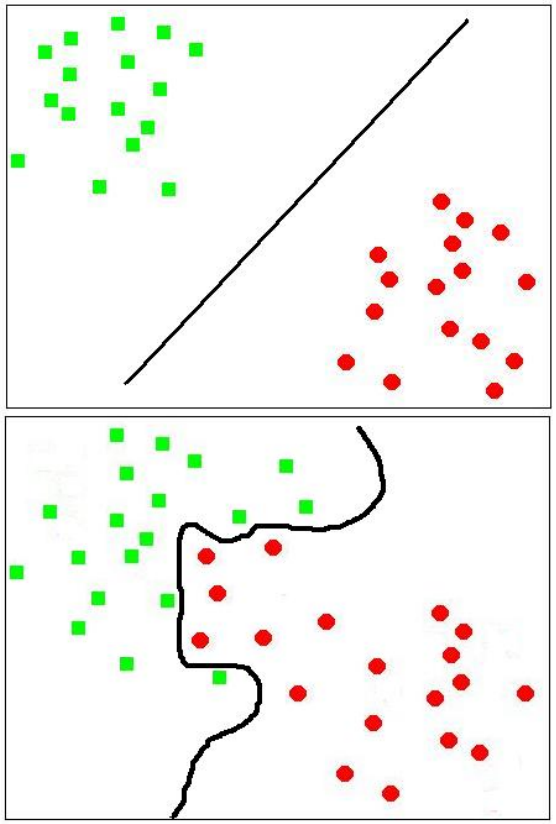
$$f(x) = sgn\left( \sum_{i=1}^{N} \alpha_i y_i \langle x, x_i \rangle + b \right)$$

**Voraussetzung:** nicht linear trennbare Trainingsdaten

**Idee:**

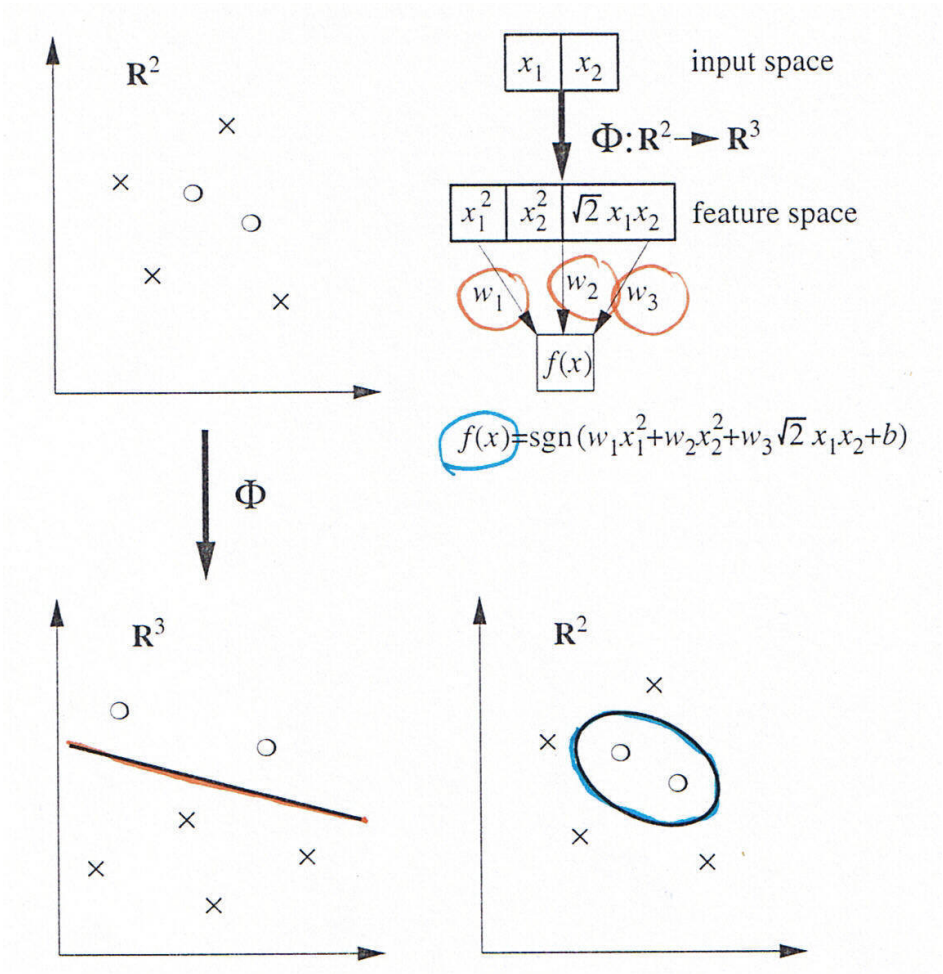
- Überführe die Trainingsdaten in einen Raum $\mathcal{M}$ mit so hoher Dimension, dass sich die Trainingsdaten dort linear trennen lassen.
- Die kanonische trennende Hyperebene kann in $\mathcal{M}$ bestimmt werden.

$\Rightarrow$ Bei der Rücktransformation in den ursprünglichen Raum wird die Hyperebene zu einer nicht-linearen Trennfläche.



# Beispiel

ursprüngliche Daten




höher dimensionaler Raum $\mathcal{M}$ *(hier:* $\mathcal{M} = \mathbb{R}^3$*)*

nichtlineare Entscheidungs-fläche im ursprünglichen Raum.

# Der Kern-Trick

**Problematik:**

Im höherdimensionalen Raum sind Skalarprodukt-Berechnungen der Form $\langle \Phi(x_i), \Phi(x_j) \rangle$ nötig.

$\Rightarrow$ Sehr komplex und rechenlastig.

**Ausweg:**

Benutze eine sog. *Kern-Funktion* $k$, die sich wie ein Skalarprodukt in $\mathcal{M}$ verhält:

$$k(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle$$

# Wdh.: Eigenschaften der Kern-Funktion

- $k : \mathbb{R}^g \times \mathbb{R}^g \to \mathcal{M}$
  wobei $\mathcal{M}$ mit einem Skalarprodukt versehen sein soll
- k symmetrisch und positiv definit
- typische Funktionen:
  - POLYNOMIELL VOM GRAD d: $k(x_i, x_j) = (c + \langle x_i, x_j \rangle)^d$
    für c konstant
  - RADIAL BASIS: $k(x_i, x_j) = \exp(-\frac{\|x_i - x_j\|^2}{c})$ für $c > 0$
  - NEURONALES NETZWERK: $k(x_i, x_j) = \tanh(\kappa \langle x_i, x_j \rangle + \theta)$
    wobei $\kappa > 0$ und $\theta \in \mathbb{R}$

# Beispiel

Betrachte

- zwei Trainingsdaten $(x_1, y_1)$ und $(x_2, y_2)$, wobei $y_1, y_2 \in \{\pm 1\}$ und $x_1, x_2 \in \mathbb{R}^2$, d.h. $x_1 = (x_{11}, x_{12})$ und $x_2 = (x_{21}, x_{22})$
- polynomieller Kern zweiten Grades mit $c = 1$

Dann gilt:

$$\begin{aligned} k(x_1, x_2) &= (1 + \langle x_1, x_2 \rangle)^2 \\ &= (1 + x_{11}x_{21} + x_{12}x_{22})^2 \\ &= 1 + 2x_{11}x_{21} + 2x_{12}x_{22} + (x_{11}x_{21})^2 + (x_{12}x_{22})^2 + 2x_{11}x_{21}x_{12}x_{22} \end{aligned}$$

Mit $\Phi(x_1) = \Phi((x_{11}, x_{12})) \mapsto (1, \sqrt{2}x_{11}, \sqrt{2}x_{12}, x_{11}^2, x_{12}^2, \sqrt{2}x_{11}x_{12})$ folgert:

$$\langle \Phi(x_1), \Phi(x_2) \rangle = k(x_1, x_2)$$

# Nicht-lineare Lösung

Damit ergibt sich nun

- Der Raum $\mathcal{M}$ muss nicht bekannt sein. Die Kern-Funktion als Maß der Ähnlichkeit ist für alle Berechnungen ausreichend.
- Die Lösung des optimalen Programms ergibt sich durch Ersetzen des ursprl. Skalarproduktes durch die Kern-Funktion.
- Die Entscheidungsfunktion hat dann die folgende Form:

$$f(x) = sgn\left(\sum_{i=1}^{N} \alpha_i y_i k(x, x_i) + b\right)$$

# Bisherige Problematik

Ein einzelner Ausreißer in den Trainingsdaten kann die Ausprägung der Hyperebene stark beeinflussen
($\rightsquigarrow$ höherdimensionale Berechnungen)

$\Rightarrow$ Oft nützlich eine bestimmte Anzahl an Ausreißern/ Fehlern zuzulassen

**Idee:**

Erlaube, aber *bestrafe* derartige Fehleinordnungen

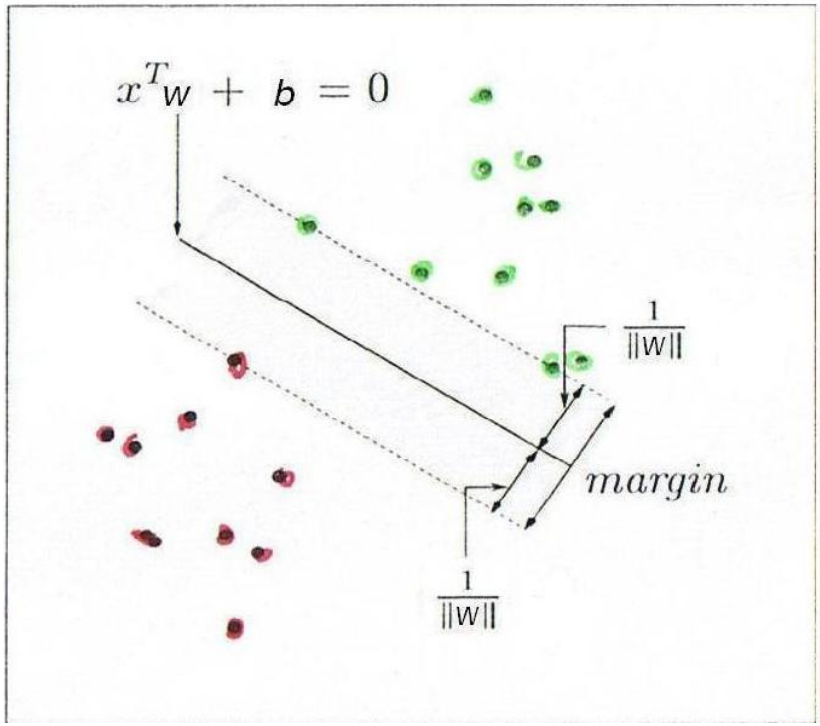
# Grundidee

**Vorgehen:**

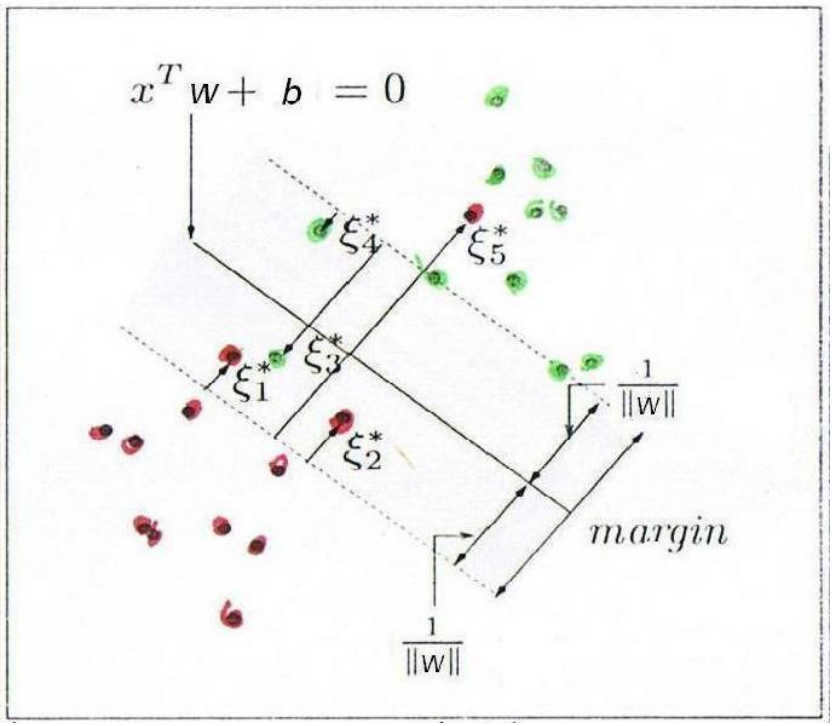
- Schwäche die Randbedingung ab,
  d.h. führe die sogenannten *Schlupfvariablen* $\xi_i \geq 0$ ein mit

$$y_i(\langle x_i, w \rangle + b) \geq 1 - \xi_i \quad \forall i = 1, ..., N$$

- Lege eine Strafe in Form des Kostenterms $\gamma \xi_i$ fest.
  $\gamma$ kann als *Fehlergewicht* interpretiert werden.




trennbare Daten




überlappende Daten

- $\xi_i = 0$ für korrekt klassifizierte Trainingsdaten
- $0 < \xi_i \leq 1$ für korrekt klassifizierte Daten innerhalb des Randes
- $\xi_i > 1$ für Trainingsdaten auf der falschen Seite von $\mathcal{H}$

# Bemerkungen

- Schlupfvariablen drücken folgendes aus:
  - Bevorzugung eines Randes, der die Trainingsdaten korrekt klassifiziert
  - Abschwächung der Nebenbedingungen, so dass im nicht-trennbaren Fall die Strafe propotional zum Ausmaß der Misklassifikation ist
- $\gamma$ kontrolliert die Gewichtung zwischen den konkurrierenden Zielen
  - breiter Rand mit großen Fehlern
  - kleine Fehler, aber schmaler Rand

# Zugehöriges Optimierungsproblem

Aufnahme des Strafterms in das Minimierungsproblem führt zu

$$\underset{w\in\mathcal{M},b\in\mathbb{R},\xi\in\mathbb{R}^N}{\text{minimiere}} \quad \frac{1}{2}\|w\|^2 + \gamma\sum_{i=1}^{N}\xi_i$$

unter den Bedingungen

$$\begin{aligned} \xi_i &\geq 0 \\ y_i(\langle x_i, w\rangle + b) &\geq 1-\xi_i \quad \forall i = 1,...,N \end{aligned}$$

Minimierung der Lagrange-Funktion

$$L(w, b, \alpha, \mu) = \frac{1}{2}\|w\|^2 + \gamma \sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \alpha_i (y_i(\langle x_i, w \rangle + b) - (1 - \xi_i)) - \sum_{i=1}^{N} \mu_i \xi_i$$

bezüglich w, b und $\xi_i$ ergibt analog zu oben die Lösung:

$$\begin{aligned} w &= \sum_{i=1}^{N} \alpha_i x_i y_i \\ 0 &= \sum_{i=1}^{N} \alpha_i y_i \\ \alpha_i &= \gamma - \mu_i \qquad \forall\ i = 1, ..., N \end{aligned}$$

wobei die $\alpha_i$ durch Lösen des quadratischen Programmes

$$\underset{\alpha \in \mathbb{R}^N}{\text{maximiere}} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle$$

NB:

$$0 \leq \alpha_i \leq \gamma \quad \forall\ i = 1, ..., N$$

$$\sum_{i=1}^{N} \alpha_i y_i = 0$$

bestimmt werden können.

# Support Vektoren

Mit den Kuhn-Tucker-Bedingungen

$$\begin{aligned} \alpha_i[y_i(\langle x_i, w\rangle + b) - (1 - \xi_i)] &= 0 \\ \mu_i \xi_i &= 0 \\ y_i(\langle x_i, w\rangle + b) - (1 - \xi_i) &\geq 0 \qquad \forall i = 1, .., N \end{aligned}$$

ergeben sich *zwei* mögliche Arten von **Support Vektoren**:

- Punkte direkt auf dem Rand
  (mit $\xi_i = 0$ und daraus folgend $0 < \alpha_i < \gamma$)
- Punkte jenseits ihres Randes
  (mit $\xi_i > 0$ und $\alpha_i = \gamma > 0$)

# Bemerkungen

- die Schlupfvariablen verschwinden aus dem dualen Problem
- die Konstante $\gamma$ taucht dort nur noch als zusätzliche Beschränkung der Lagrange-Multiplikatoren $\alpha_i$ auf
- auch im Fall der Soft-Margin-Klassifikation kann der Kern-Trick angewendet werden
- Entscheidungsfunktion f und Verschiebung b bestimmen sich analog zu oben.

# Die Konstante $\gamma$

Bisher wurde keine Aussage über die Wahl von $\gamma$ gemacht.

- $\gamma$ groß $\rightsquigarrow$ hohes Fehlergewicht $\rightsquigarrow$ kleiner Rand $\rightsquigarrow$ Fokusierung auf Punkte nahe $\mathcal{H}$
- $\gamma$ klein $\rightsquigarrow$ schwaches Fehlergewicht $\rightsquigarrow$ breiter Rand $\rightsquigarrow$ Einbeziehung ferner Punkte

$\Rightarrow$ Intuitive Bestimmung von $\gamma$ schwierig

Üblicherweise wird dazu *Kreuzvalidierung* genutzt.

# Mehrklassige SVM

$\rightsquigarrow$ Einteilung in M Klassen (mit $M > 2$)

- **One Versus the Rest**
  - Bilde eine Klassifikatoren-Menge $f^1, ..., f^M$ durch jeweiliges Trennen einer Klasse von den Restlichen
  - $f(x) := \underset{j=1,...,M}{\arg\max}\ g^j(x)$, wobei $g^j(x) = \sum_{i=1}^{N} y_i \alpha_i^j k(x, x_i) + b^j$

- **Paarweise Klassifikation**
  - Bilde Klassifikatoren für jedes mögliche Paar von Klassen ($\rightsquigarrow \frac{M(M-1)}{2}$ Stück)
  - Einordnung eines neuen Datenpunktes in diejenige Klasse, die die höchste Anzahl an *Stimmen* (d.h. Klassifikatoren, die den Datenpunkt in diese Klasse einordnen) aufweisen kann

- **Error-Correcting Output Coding**
  - Generiere L binäre Klassifikatoren $f^1, ..., f^L$ durch Aufteilung der ursprünglichen Trainingsdaten in jeweils zwei disjunkte Klassen
  - Die Auswertung eines Datenpunktes anhand aller L Funktionen bestimmt seine Klasse eindeutig
    ($\rightsquigarrow$ Jede Klasse entspricht einem eindeutigen Vektor in $\{\pm 1\}^L$)
  - Für M Klassen ergibt sich damit die sogenannte *decoding matrix* $\mathcal{D} \in \{\pm 1\}^{M \times L}$
  - Ein neuer Datenpunkt wird durch Vergleich von dessen L-dimensionalem Vektor mit den Zeilen der Matrix $\mathcal{D}$ einer Klasse zugeteilt.

# Zusammenfassung

**Vorteile:**

- Klassifikation sehr schnell möglich
  ($\rightsquigarrow$ basierend auf wenigen Support Vektoren)
- hohe Generalisierungsfähigkeit
  ($\rightsquigarrow$ gute Anwendbarkeit auf reale Probleme)
- Arbeiten in hohen Dimensionen möglich

# Zusammenfassung

**Nachteile:**

- neues Training für neue (verschiedene) Eingabedaten erforderlich
- Umgang mit nicht-linear separierbaren Problemen trickreich
  ($\rightsquigarrow$ Größe der Dimension)
- Wahl des Kerns schwierig
  ($\rightsquigarrow$ muss empirisch gesucht werden)

# Fragen?

Vielen Dank für die Aufmerksamkeit.