Neuronale Netze

Prof. Dr. Rudolf Kruse

Computational Intelligence
Institut für Intelligente Kooperierende Systeme
Fakultät für Informatik
rudolf.kruse@ovgu.de

Überwachtes Lernen / Support Vector Machines

Überwachtes Lernen, Diagnosesystem für Krankheiten

Trainingsdaten: Expressionsprofile von Patienten mit bekannter Diagnose

Durch die bekannte Diagnose ist eine Struktur in den Daten vorgegeben, die wir auf zukünftige Daten verallgemeinern wollen.

Lernen/Trainieren: Leite aus den Trainingsdaten eine Entscheidungsregel ab, die die beiden Klassen voneinander trennt.

Generalisierungsfähigkeit: wie gut ist die Entscheidungsregel darin, zukünftige Patienten zu diagnostizieren?

Ziel: finde eine Entscheidungsregel mit hoher Generalisierungsfähigkeit!

Lernen von Beispielen

Gegeben: $X = \{x_i, y_i\}_{i=1}^{n}$, Trainingsdaten von Patienten mit bekannter Diagnose

bestehend aus:

$$x_i \in \mathbb{R}^g \text{ (Punkte, Expressionsprofile)}$$

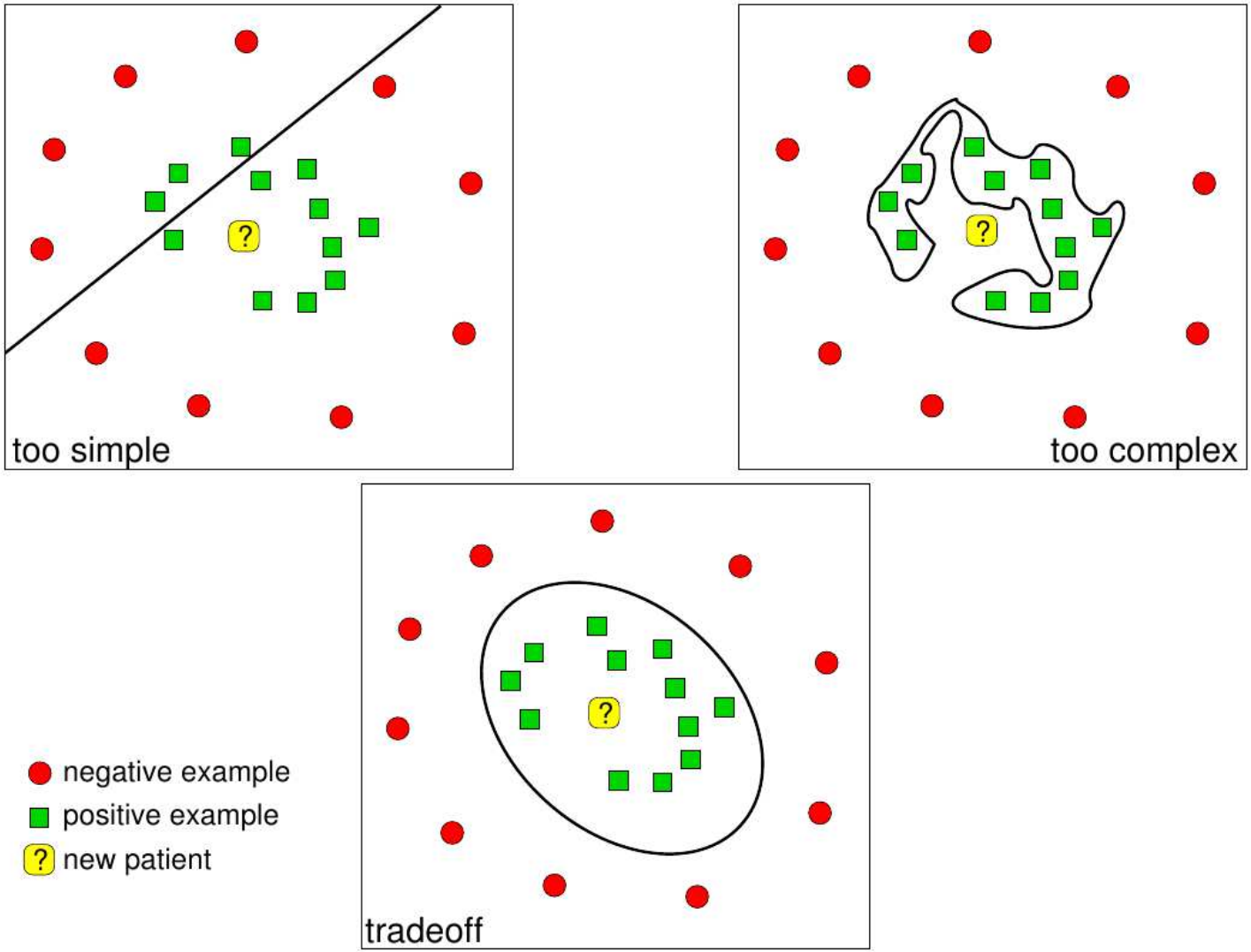
$$y_i \in \{+1, -1\} \text{ (Klassen, 2 Arten von Krebs)}$$

Entscheidungsfunktion:

$$f_X : \mathbb{R}^g \to \{+1, -1\}$$

$$\text{Diagnose} = f_X(\text{neuer Patient})$$

Underfitting / Overfitting



Lineare Trennung der Trainingsdaten

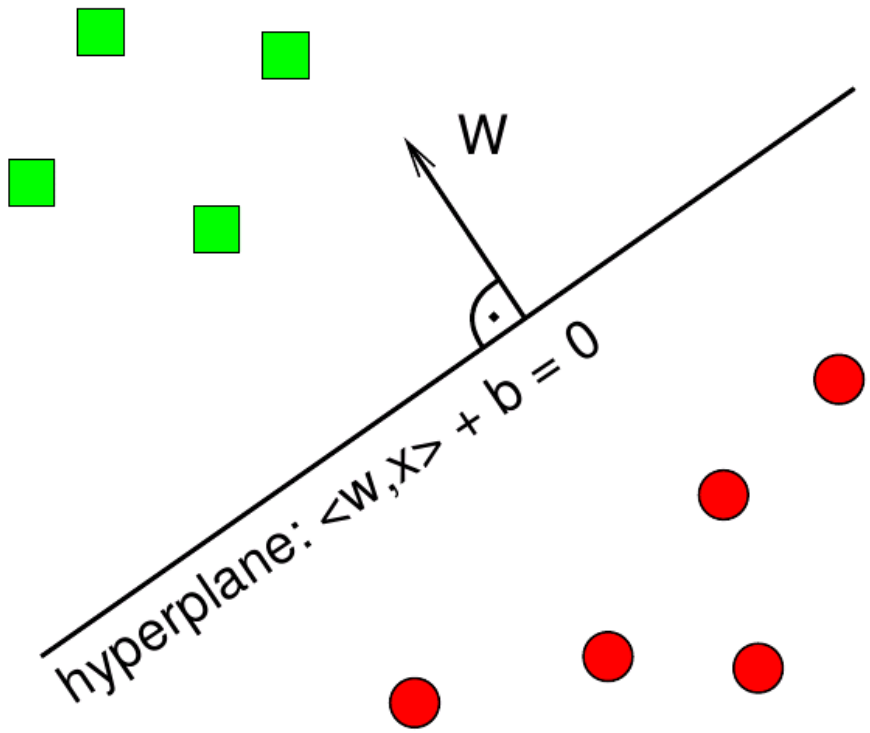
Wir fangen mit linearer Trennung an und vergrößern die Komplexität in einem zweiten Schritt durch Kernel-Funktionen.

Eine trennende Hyperebene ist definiert durch

den Normalenvektor w und
die Verschiebung b:

Hyperebene $\mathcal{H} = \{x | \langle w, x \rangle + b = 0\}$

$\langle \cdot, \cdot \rangle$ nennt man inneres Produkt
oder Skalarprodukt



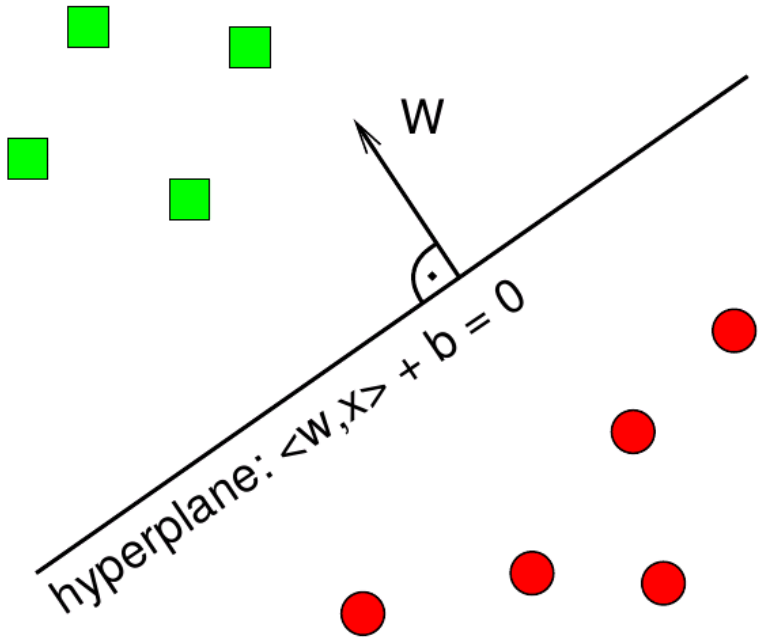
Vorhersage der Klasse eines neuen Punktes

Training: Wähle w und b so, daß die Hyperebene die Trainingsdaten trennt.

Vorhersage: Auf welcher Seite der Hyperebene liegt der neue Punkt?

Punkte in Richtung des Normalenvektors diagnostizieren wir als POSITIV.

Punkte auf der anderen Seite diagnostizieren wir als NEGATIV.



Motivation

Ursprung in statistischer Lerntheorie; Klasse optimaler Klassifikatoren

Zentrales Problem der statistischen Lerntheorie: Generalisierungsfähigkeit: Wann führt ein niedriger Trainingsfehler zu einem niedrigen echten Fehler?

Zweiklassenproblem:
Klassifikationsvorgang $\equiv$ Zuordnungsfunktion $f(x, u): \ x \to y \in \{+1, -1\}$
x: Muster aus einer der beiden Klassen
u: Parametervektor des Klassifikators

Lernstichprobe mit l Beobachtungen $x_1, x_2, \ldots, x_l$
mit entsprechender Klassenzugehörigkeit $y_1, y_2, \ldots, y_l$
$\to$ das empirische Risiko (Fehlerrate) für gegebenen Trainingsdatensatz:

$$R_{emp}(u) \ = \ \frac{1}{2l} \sum_{i=1}^{l} |y_i - f(x_i, u)| \ \in [0, 1]$$

Viele Klassifikatoren, z.B. neuronale Netze, minimieren das empirische Risiko

Motivation

Erwartungswert des Zuordnungsfehlers (expected risk):

$$R(u) = E\{R_{test}(u)\} = E\{\frac{1}{2}|y - f(x,u)|\} = \int \frac{1}{2}|y - f(x,u)|p(x,y)\, dxdy$$

$p(x,y)$: Verteilungsdichte aller möglichen Samples x mit entsprechender Klassenzugehörigkeit y (Dieser Ausdruck nicht direkt auswertbar, da $p(x,y)$ nicht zur Verfügung steht)

Optimale Musterklassifikation:
Deterministische Zuordnungsfunktion $f(x,u) : \; x \rightarrow y \in \{+1,-1\}$ gesucht, so dass das Expected Risk minimiert wird

Zentrale Frage der Musterklassifikation:

> Wie nah ist man nach l Trainingsbeispielen am *echten* Fehler? Wie gut kann aus dem empirischen Risiko $R_{emp}(u)$ das echte Risiko $R(u)$ abgeschätzt werden? (Structural Risk Minimization statt Empirical Risk Minimization)

Antwort durch Lerntheorie von Vapnik-Chervonenkis $\rightarrow$ SVM

SVM für linear trennbare Klassen

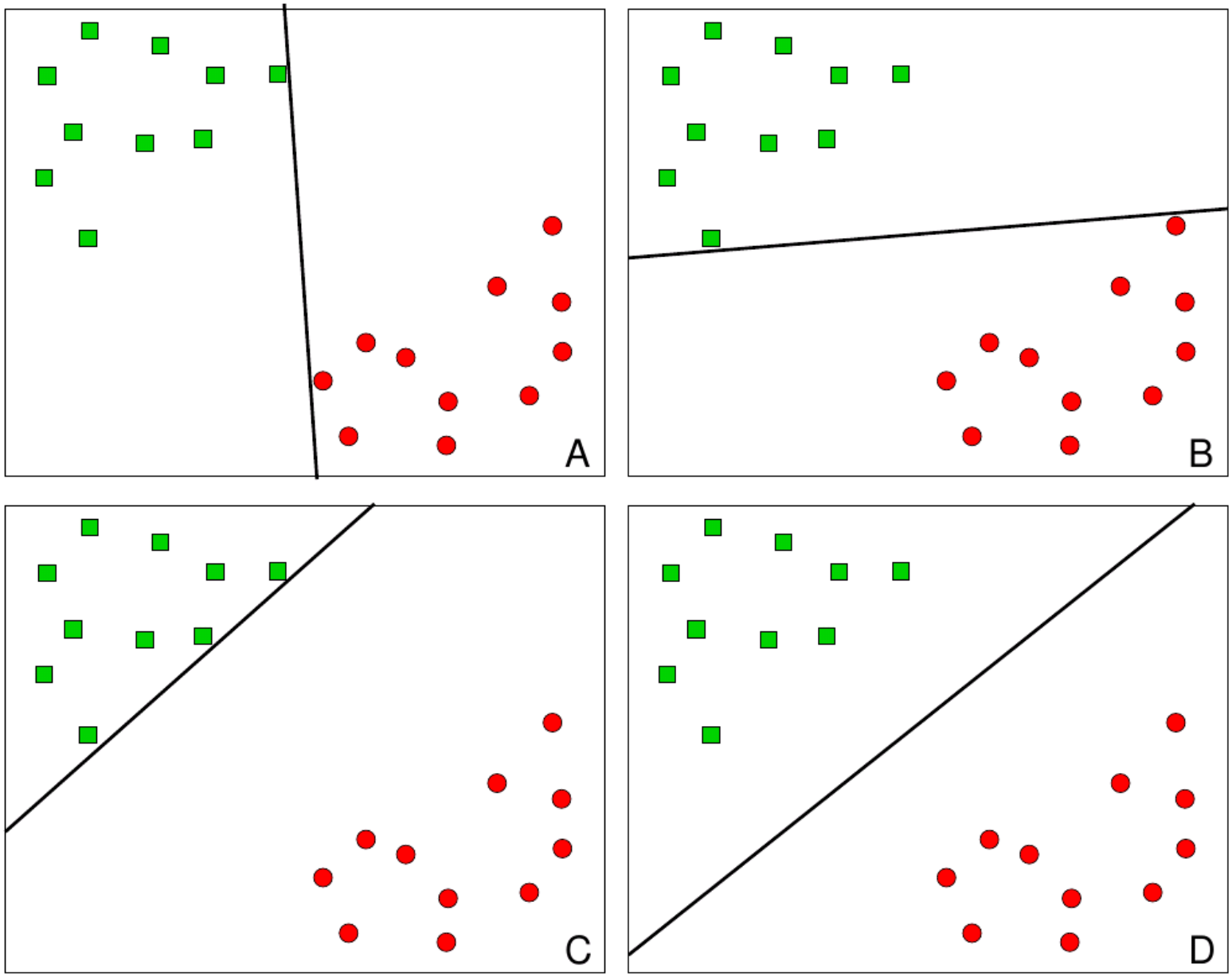
Bisherige Lösung:

Allgemeine Hyperebene: $wx + b = 0$

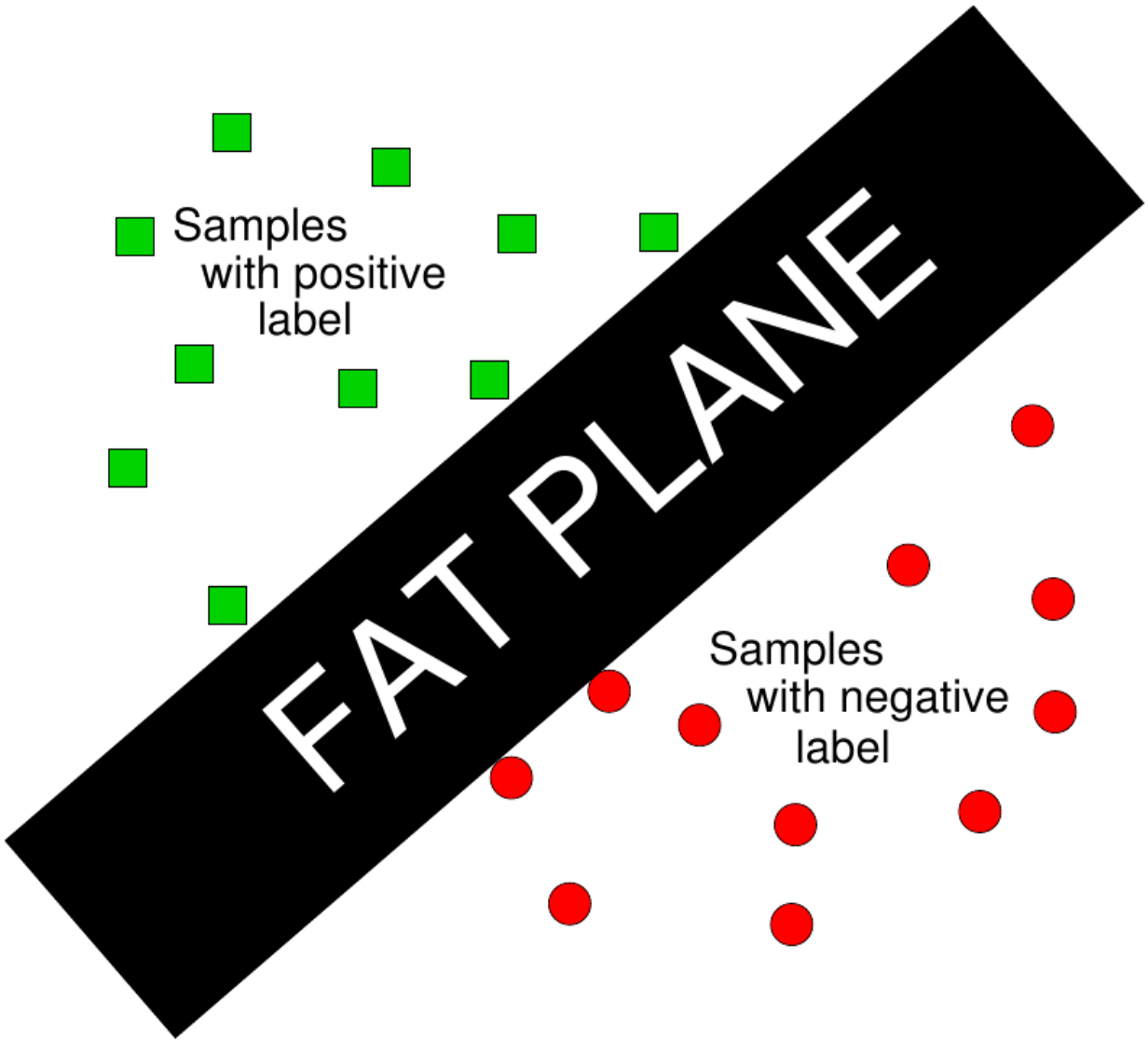
Klassifikation: $\text{sgn}(wx + b)$

Training z.B. mittels Perzeptron-Algorithmus (iteratives Lernen, Korrektur nach jeder Fehlklassifikation; keine Eindeuigkeit der Lösung)

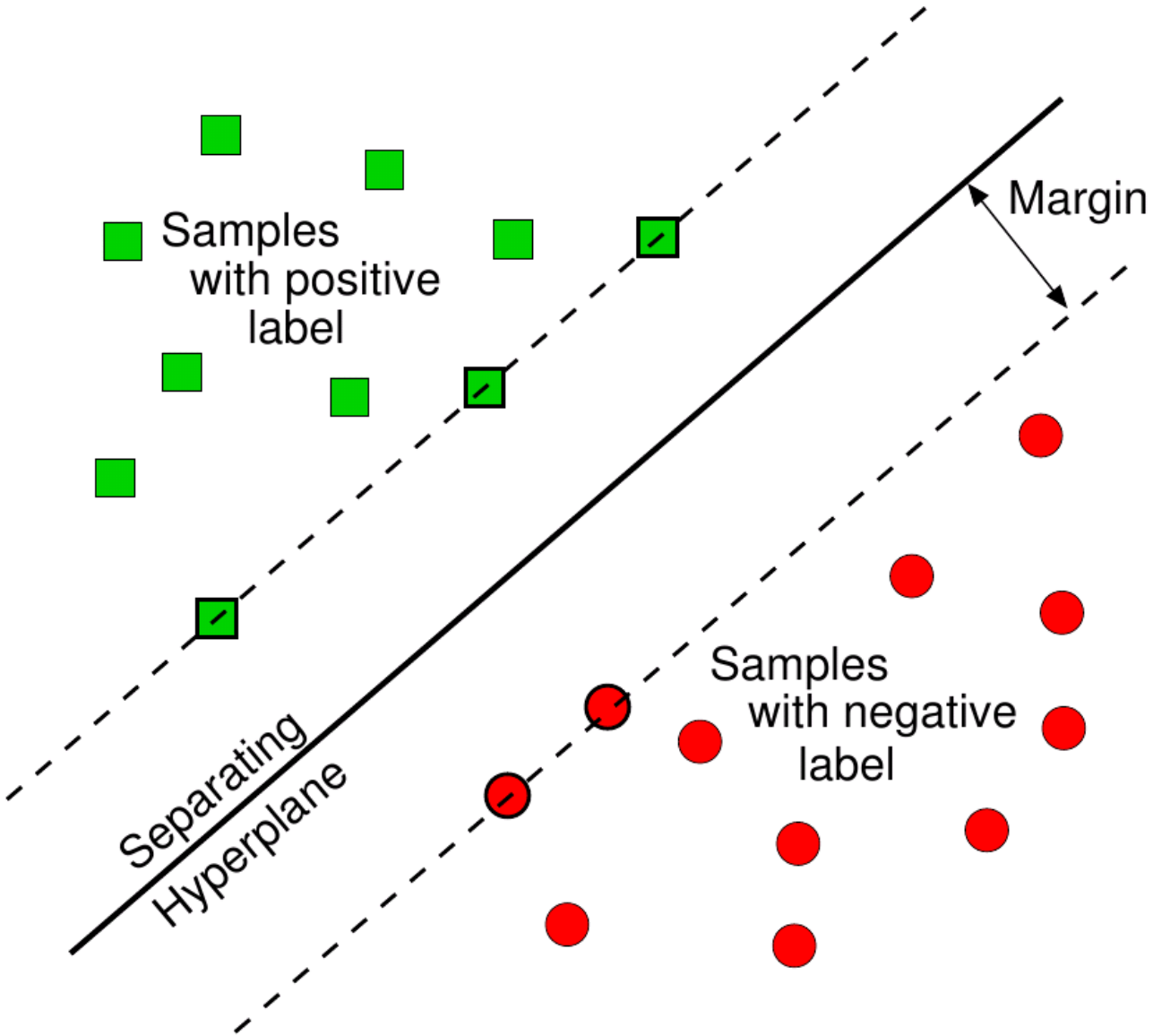
Welche Hyperebene ist die beste? Und warum?



Keine scharfe Trennung, sondern eine . . .

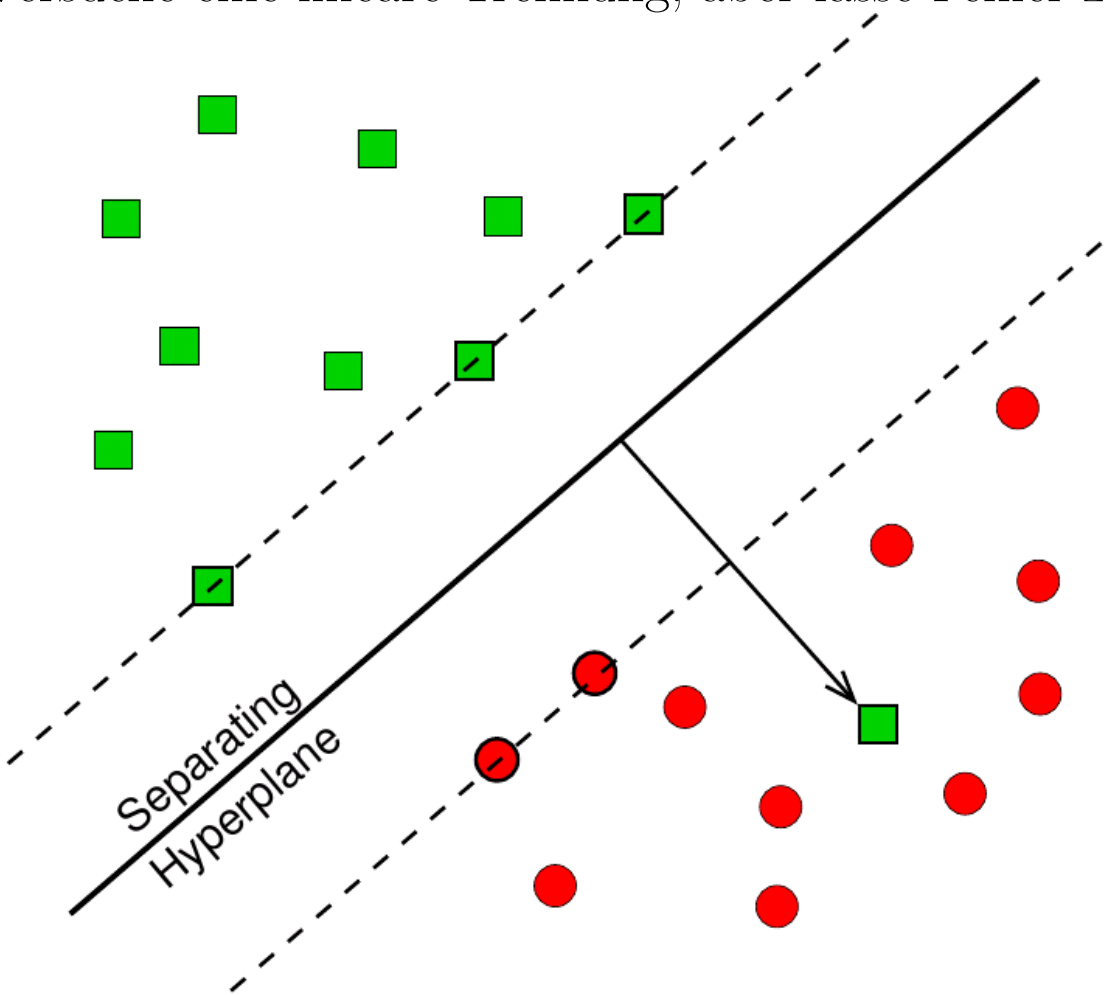


Trenne die Trainingsdaten mit maximaler Trennspanne



Trenne die Trainingsdaten mit maximaler Trennspanne

Versuche eine lineare Trennung, aber lasse Fehler zu:



Strafe für Fehler: Abstand zur Hyperebene multipliziert mit Fehlergewicht C

SVM für linear trennbare Klassen

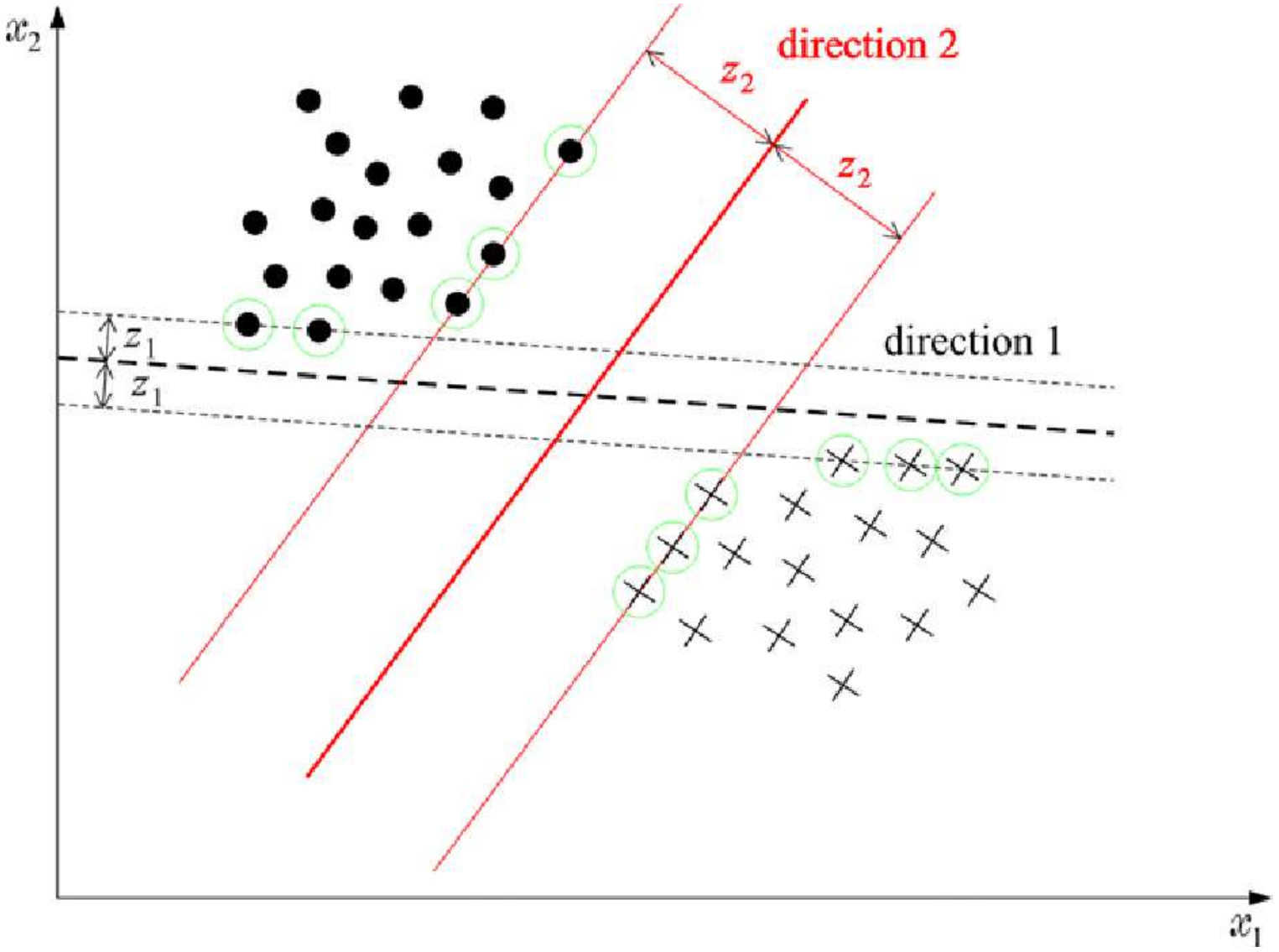
Bei SVM wird eine trennende Hyperebene mit maximalem Rand gesucht. *Optimal:* diejenige mit größtem 2δ von allen möglichen Trennebenen.

Anschaulich sinnvoll (Bei konstanter Intraklassenstreuung wächst Klassifikationssicherheit mit wachsendem Interklassenabstand)

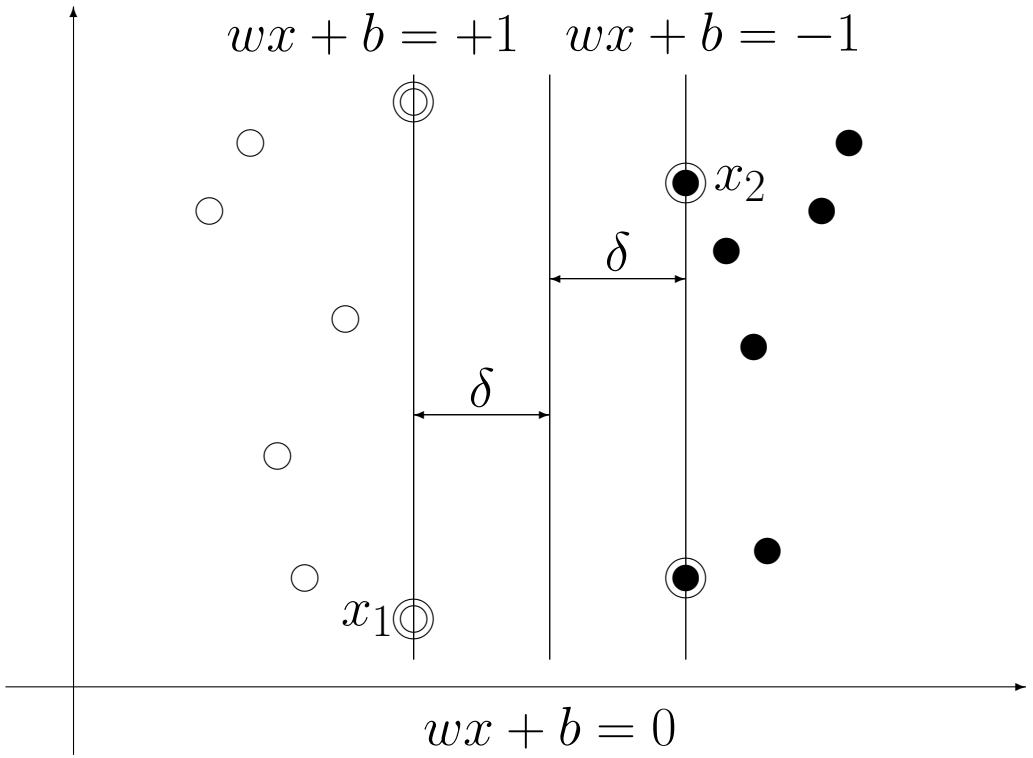
Theoretisch sind SVM durch statistische Lerntheorie begründet

SVM für linear trennbare Klassen

Large-Margin Klassifikator: Trennlinie 2 besser als Trennlinie 1



SVM für linear trennbare Klassen



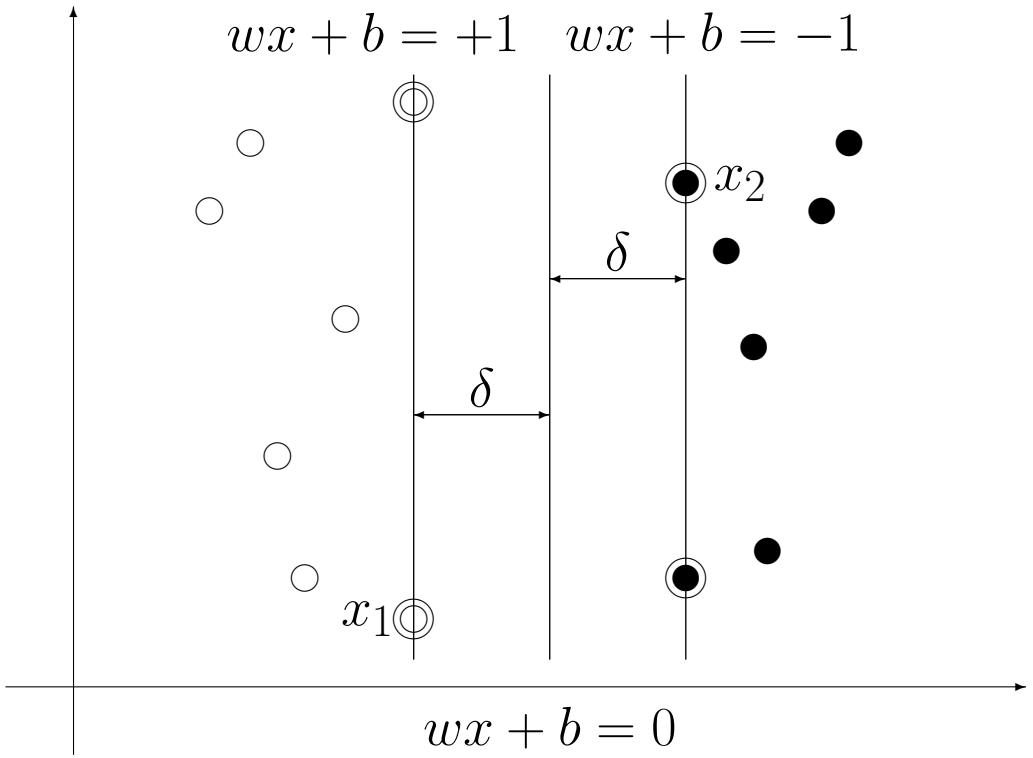
Problem:
Die optimale Hyperebene
$$\mathcal{H} = \{x | \langle w, x \rangle + b = 0\}$$
ist nicht eindeutig.

Für $c \neq 0 : \{x | \langle w, x \rangle + b = 0\} = \{x | \langle cw, x \rangle + cb = 0\}$
Also: (cw, cb) beschreibt die gleiche Hyperebene wie (w,b).
Deshalb: Skaliere (w,b) relativ zu den Trainingsdaten $X = \{x_i, y_i\}_{i=1}^n$, so dass

$$\min_{x_i \in X} |\langle w, x_i \rangle + b| = 1$$

Diese Hyperebene ist die **kanonische Hyperebene**

SVM für linear trennbare Klassen



Trainingsdaten werden dann korrekt klassifiziert, falls:

$$y_i(wx_i + b) \geq 1$$

wobei:

$$\begin{cases} wx_i + b = +1; \text{(Klasse mit } y_i = +1) \\ wx_i + b = -1; \text{(Klasse mit } y_i = -1) \end{cases}$$

Der Abstand zwischen den kanonischen Hyperebenen ergibt sich durch Projektion von $x_1 - x_2$ auf den Einheitsnormalenvektor $\frac{w}{||w||}$:

$$2\delta = \frac{2}{||w||}; \quad \text{d.h. } \delta = \frac{1}{||w||}$$

$\rightarrow$ Maximierung von $\delta \equiv$ Minimierung von $||w||^2$

SVM für linear trennbare Klassen

Optimale Trennebene durch Minimierung einer quadratischen Funktion unter linearen Nebenbedingungen:

Primales Optimierungsproblem:
minimiere: $J(w,b) = \frac{1}{2}||w||^2$
unter Nebenbedingungen $\forall i \; [y_i(wx_i + b) \geq 1], \; i = 1, 2, \ldots, l$

Einführung einer Lagrange-Funktion:

$$L(w,b,\alpha) = \frac{1}{2}||w||^2 - \sum_{i=1}^{l} \alpha_i [y_i(wx_i + b) - 1]; \quad \alpha_i \geq 0$$

führt zum *dualen Problem*: maximiere $L(w,b,\alpha)$ bezüglich α, unter den Nebenbedingungen:

$$\frac{\partial L(w,b,\alpha)}{\partial w} = 0 \quad \Longrightarrow \quad w = \sum_{i=1}^{l} \alpha_i y_i x_i$$

$$\frac{\partial L(w,b,\alpha)}{\partial b} = 0 \quad \Longrightarrow \quad \sum_{i=1}^{l} \alpha_i y_i = 0$$

SVM für linear trennbare Klassen

Einsetzen dieser Terme in $L(w, b, \alpha)$:

$$
\begin{aligned}
L(w,b,\alpha) &= \frac{1}{2}||w||^2 - \sum_{i=1}^{l} \alpha_i [y_i(wx_i + b) - 1] \\
&= \frac{1}{2} w \cdot w - w \cdot \sum_{i=1}^{l} \alpha_i y_i x_i - b \cdot \sum_{i=1}^{l} \alpha_i y_i + \sum_{i=1}^{l} \alpha_i \\
&= \frac{1}{2} w \cdot w - w \cdot w + \sum_{i=1}^{l} \alpha_i \\
&= -\frac{1}{2} w \cdot w + \sum_{i=1}^{l} \alpha_i \\
&= -\frac{1}{2} \sum_{i=1}^{l} \sum_{j=1}^{l} y_i y_j \alpha_i \alpha_j x_i x_j + \sum_{i=1}^{l} \alpha_i
\end{aligned}
$$

SVM für linear trennbare Klassen

Duales Optimierungsproblem:

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

unter Nebenbedingungen $\alpha_i \geq 0$ und $\sum_{i=1}^{l} y_i \alpha_i = 0$

Dieses Optimierungsproblem kann mithilfe der konvexen quadratischen Programmierung numerisch gelöst werden

SVM für linear trennbare Klassen

Lösung des Optimierungsproblems:

$$
\begin{aligned}
w^* &= \sum_{i=1}^{l} \alpha_i y_i x_i = \sum_{x_i \in SV} \alpha_i y_i x_i \\
b^* &= -\frac{1}{2} \cdot w^* \cdot (x_p + x_m) \\
&\quad \text{für beliebiges } x_p \in SV,\ y_p = +1,\ \text{und } x_m \in SV,\ y_m = -1
\end{aligned}
$$

wobei

$$SV = \{x_i \mid \alpha_i > 0,\ i = 1, 2, \ldots, l\}$$

die Menge aller Support-Vektoren darstellt

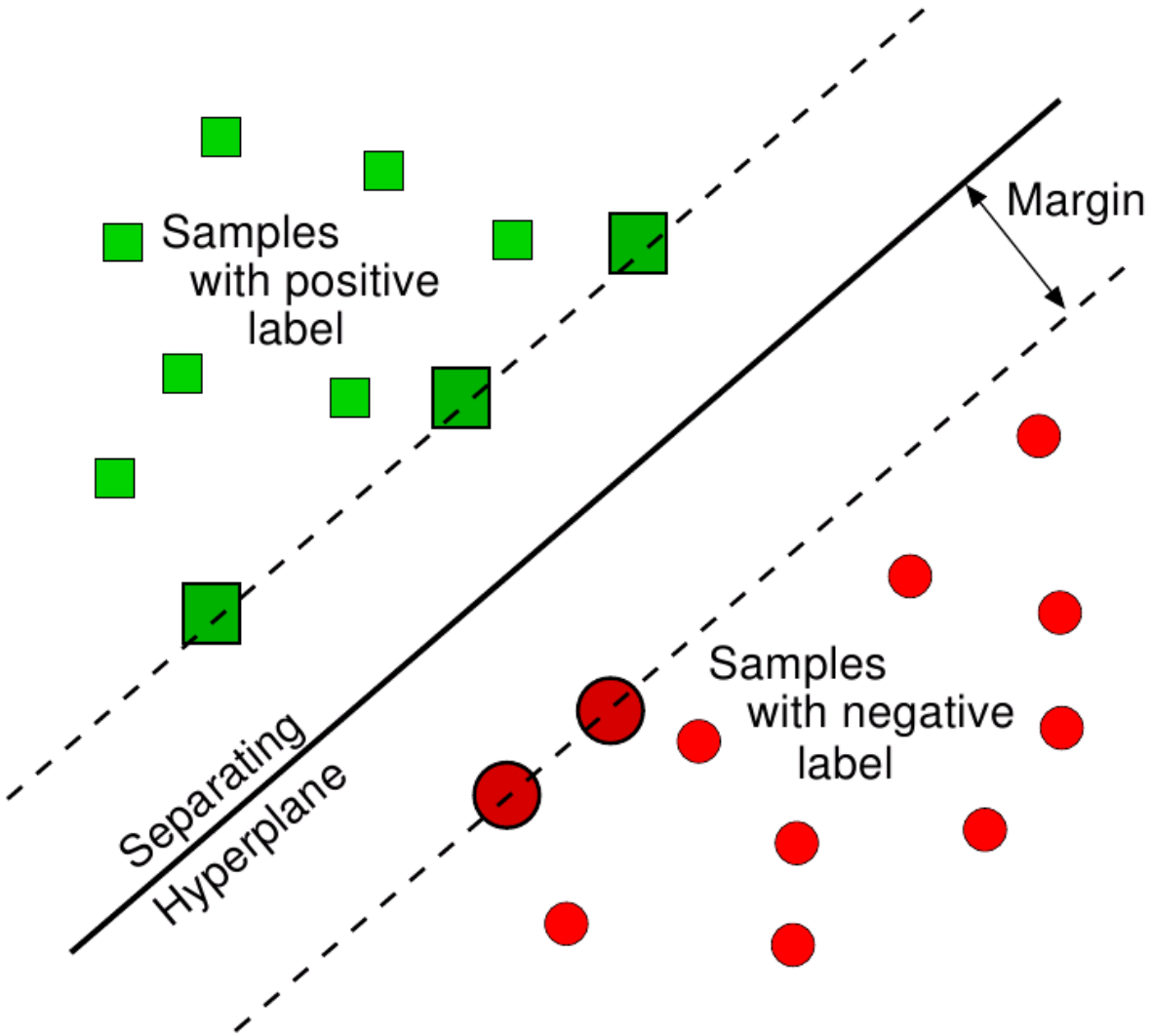
Klassifikationsregel:

$$\operatorname{sgn}(w^* x + b^*) = \operatorname{sgn}[(\sum_{x_i \in SV} \alpha_i y_i x_i) x + b^*]$$

Die Klassifikation hängt nur von den Support-Vektoren ab!

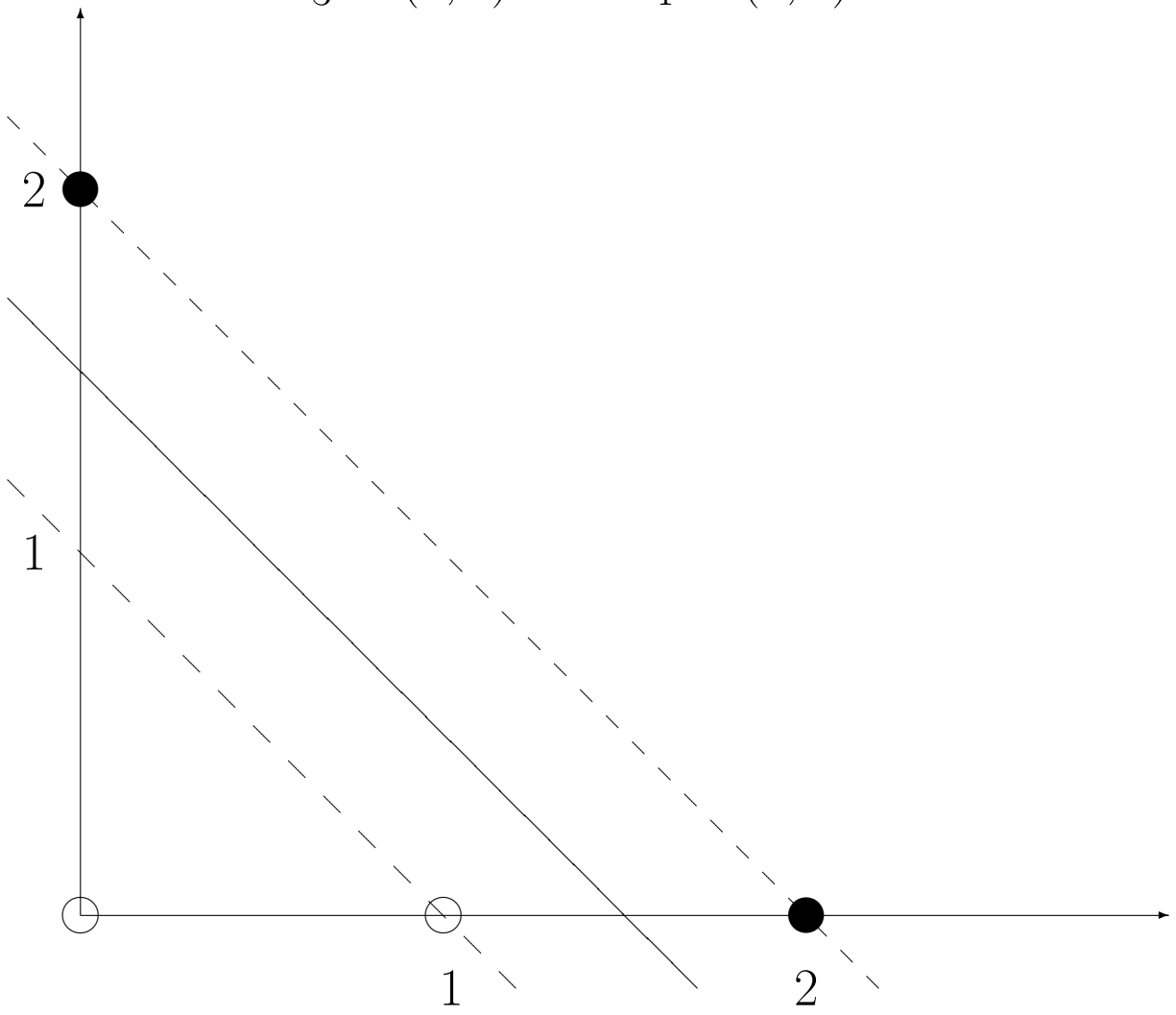
SVM für linear trennbare Klassen

Beispiel: Support-Vektoren



SVM für linear trennbare Klassen

Beispiel: Klasse +1 enthält $x_1 = (0, 0)$ und $x_2 = (1, 0)$;
Klasse -1 enthält $x_3 = (2, 0)$ und $x_4 = (0, 2)$



SVM für linear trennbare Klassen

Das duale Optimierungsproblem lautet:

maximiere: $L'(\alpha) = (\alpha_1 + \alpha_2 + \alpha_3 + \alpha_4) - \frac{1}{2}(\alpha_2^2 - 4\alpha_2\alpha_3 + 4\alpha_3^2 + 4\alpha_4^2)$
unter Nebenbedingungen $\alpha_i \geq 0$ und $\alpha_1 + \alpha_2 - \alpha_3 - \alpha_4 = 0$

Lösung:

$$\alpha_1 = 0, \quad \alpha_2 = 1, \quad \alpha_3 = \tfrac{3}{4}, \quad \alpha_4 = \tfrac{1}{4}$$
$$SV = \{(1,0),\ (2,0),\ (0,2)\}$$
$$w^* = 1 \cdot (1,0) - \tfrac{3}{4} \cdot (2,0) - \tfrac{1}{4} \cdot (0,2) = (-\tfrac{1}{2}, -\tfrac{1}{2})$$
$$b^* = -\tfrac{1}{2} \cdot (-\tfrac{1}{2}, -\tfrac{1}{2}) \cdot ((1,0) + (2,0)) = \tfrac{3}{4}$$

Optimale Trennlinie: $x + y = \frac{3}{2}$

SVM für linear trennbare Klassen

Beobachtungen:

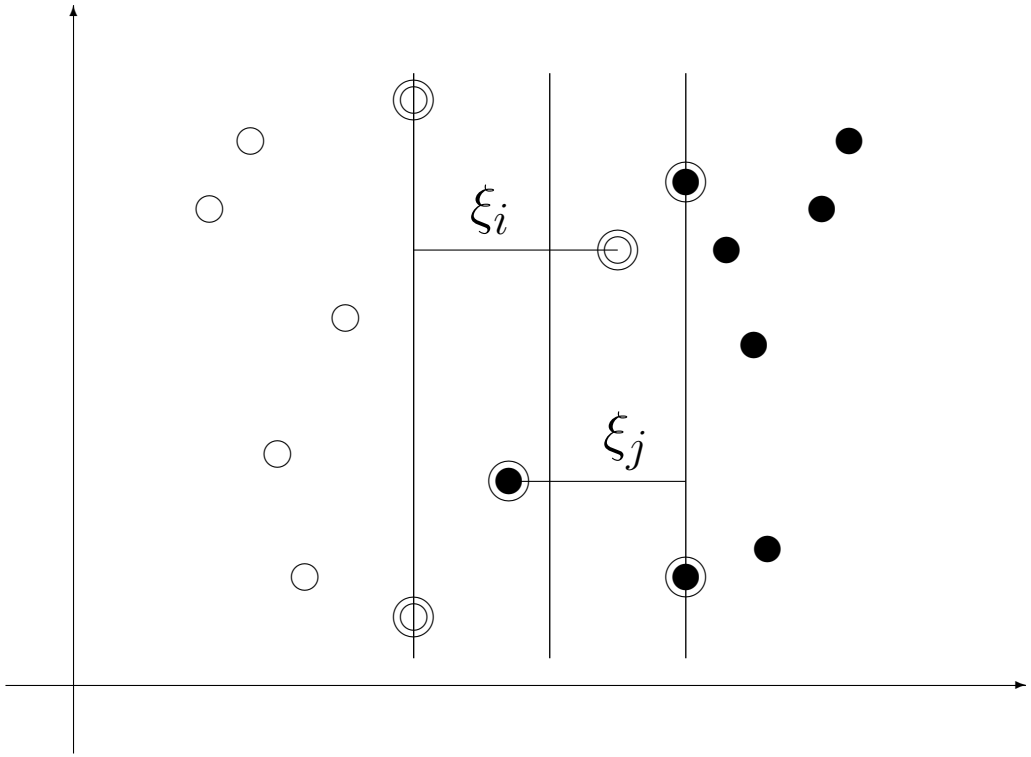
- Für die Support-Vektoren gilt: $\alpha_i > 0$
- Für alle Trainingsdaten außerhalb des Randes gilt: $\alpha_i = 0$
- Support-Vektoren bilden eine "sparse" Darstellung der Stichprobe und sind ausreichend für die Klassifikation
- Die Lösung entspricht dem globalen Optimum und ist eindeutig
- Der Optimierungsvorgang benötigt nur Skalarprodukte $x_i x_j$

SVM für nicht linear separierbare Klassen

Motivation für Verallgemeinerung:

- Keine Lösung mit bisherigem Ansatz für nicht trennbare Klassen
- Verbesserung der Generalisierung bei Ausreißern in der Randzone

Soft-Margin SVM: Einführung von Schlupf-Variablen



SVM für nicht linear separierbare Klassen

Bestrafen von Randverletzungen via "slack"-Variablen

Primales Optimierungsproblem:

minimiere: $J(w, b, \xi) = \frac{1}{2}||w||^2 + C \sum_{i=1}^{l} \xi_i$

unter Nebenbedingungen $\forall i \; [y_i(wx_i + b) \geq 1 - \xi_i, \; \xi_i \geq 0]$

Duales Optimierungsproblem:

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

unter Nebenbedingungen $0 \leq \alpha_i \leq C$ und $\sum_{i=1}^{l} y_i \alpha_i = 0$

(Weder die slack-Variablen noch deren Lagrange-Multiplier tauchen im dualen Optimierungsproblem auf!)

Einziger Unterschied zum linear trennbaren Fall: Konstante C in den Nebenbedingungen

SVM für nicht linear separierbare Klassen

Lösung des Optimierungsproblems:

$$\begin{aligned} w^* &= \sum_{i=1}^{l} \alpha_i y_i x_i = \sum_{x_i \in SV} \alpha_i y_i x_i \\ b^* &= y_k(1 - \xi_k) - w^* x_k; \quad k = \arg\max_i \alpha_i \end{aligned}$$

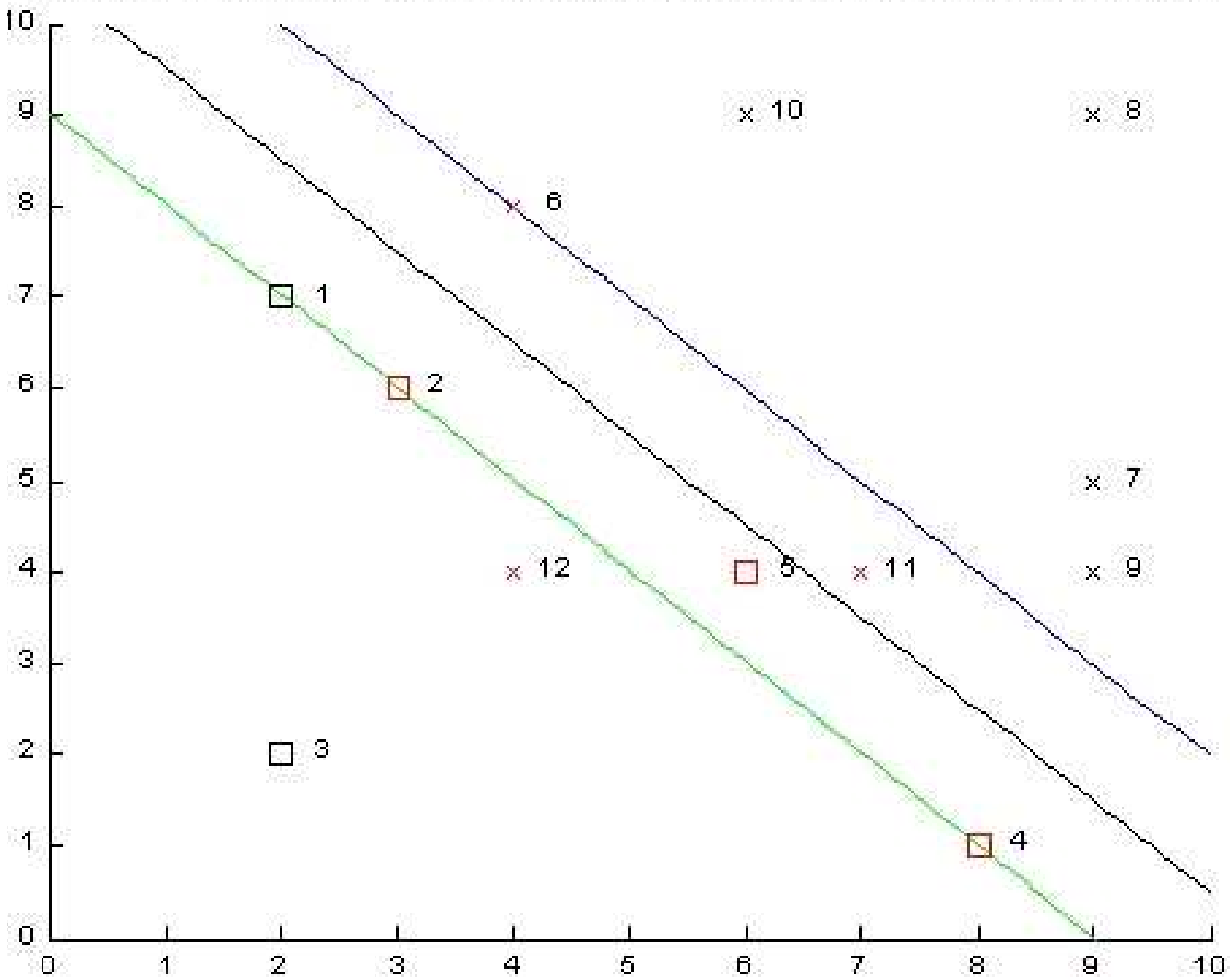
wobei

$$SV = \{x_i \mid \alpha_i > 0,\ i = 1, 2, \ldots, l\}$$

die Menge aller Support-Vektoren darstellt

SVM für nicht linear separierbare Klassen

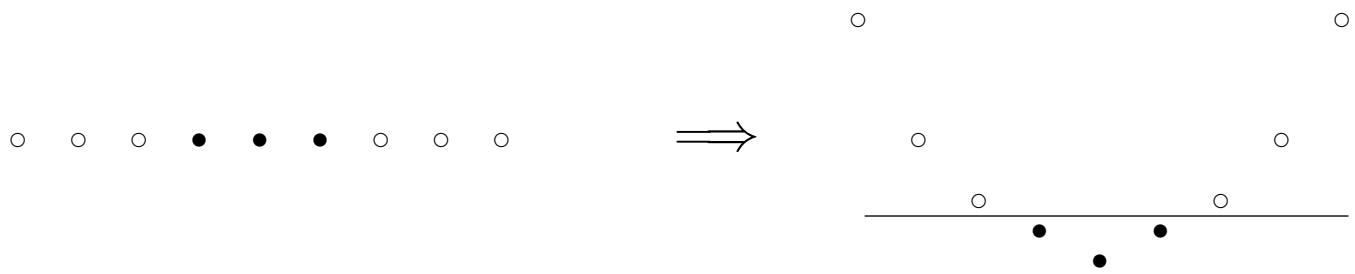
Beispiel: nicht linear trennbare Klassen



Nichtlineare SVM

Nichtlineare Klassengrenzen: Hyperebene → keine hohe Genauigkeit

Beispiel: Transformation $\Psi(x) = (x, x^2) \rightarrow C_1$ und C_2 linear trennbar



Idee: Merkmale $x \in \Re^n$ durch

$$\Psi : \quad \Re^n \longrightarrow \Re^m$$

in einen höherdimensionalen Raum $\Re^m$, $m > n$, transformieren und in $\Re^m$ eine optimale lineare Trennebene finden

Transformation Ψ steigert die lineare Trennbarkeit!

Trennende Hyperebene in $\Re^m \equiv$ nichtlineare Trennfläche in $\Re^n$

Beispiel: 2D → 3D

Nichtlineare SVM

Problem: Sehr hohe Dimension des Merkmalsraums $\Re^m$
Z.B. Polynome p-ten Grades über $\Re^n \rightarrow \Re^m$, $m = O(n^p)$

Trick mit Kernelfunktionen:

Ursprünglich in $\Re^n$: nur Skalarprodukte $x_i x_j$ erforderlich
neu in $\Re^m$: nur Skalarprodukte $\Psi(x_i)\Psi(x_j)$ erforderlich

Lösung:
$\Psi(x_i)\Psi(x_j)$ müssen nicht explizit ausgerechnet werden, sondern können mit reduzierter Komplexität mit Kernelfunktionen ausgedrückt werden

$$K(x_i, x_j) \;=\; \Psi(x_i)\Psi(x_j)$$

Nichtlineare SVM

Beispiel: Für die Transformation $\Psi : \Re^2 \longrightarrow \Re^6$

$$\Psi((y_1, y_2)) = (y_1^2,\ y_2^2,\ \sqrt{2}y_1,\ \sqrt{2}y_2,\ \sqrt{2}y_1y_2,\ 1)$$

berechnet die Kernelfunktion

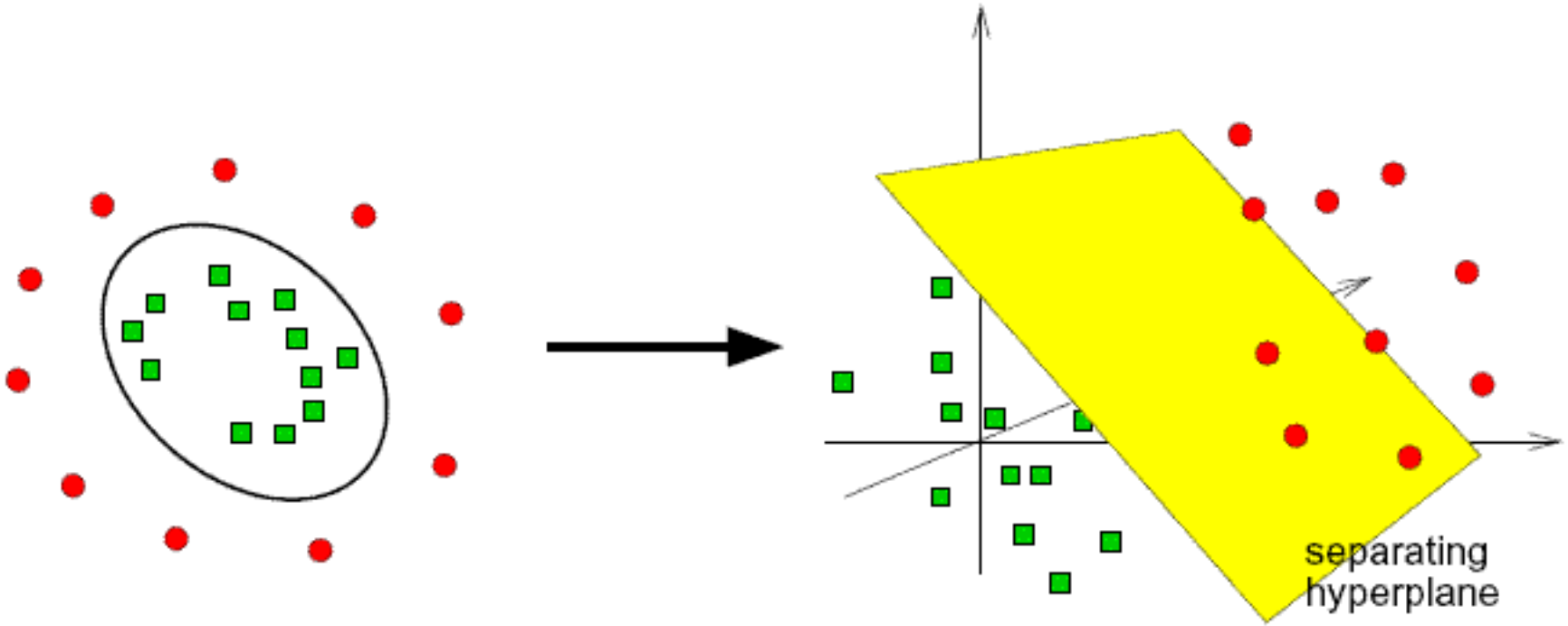
$$\begin{aligned} K(x_i, x_j) &= (x_i x_j + 1)^2 \\ &= ((y_{i1}, y_{i2}) \cdot (y_{j1}, y_{j2}) + 1)^2 \\ &= (y_{i1}y_{j1} + y_{i2}y_{j2} + 1)^2 \\ &= (y_{i1}^2, y_{i2}^2, \sqrt{2}y_{i1}, \sqrt{2}y_{i2}, \sqrt{2}y_{i1}y_{i2}, 1) \\ &\quad \cdot (y_{j1}^2, y_{j2}^2, \sqrt{2}y_{j1}, \sqrt{2}y_{j2}, \sqrt{2}y_{j1}y_{j2}, 1) \\ &= \Psi(x_i)\Psi(x_j) \end{aligned}$$

das Skalarprodukt im neuen Merkmalsraum $\Re^6$

Nichtlineare SVM

Beispiel: $\Psi : \Re^2 \longrightarrow \Re^3$

$$\Psi((y_1, y_2)) = (y_1^2, \sqrt{2}y_1y_2, y_2^2)$$



Die Kernelfunktion

$$K(x_i, x_j) = (x_i x_j)^2 = \Psi(x_i)\Psi(x_j)$$

berechnet das Skalarprodukt im neuen Merkmalsraum $\Re^3$. Wir können also das Skalarprodukt zwischen $\Psi(x_i)$ und $\Psi(x_j)$ ausrechnen, ohne die Funktion Ψ anzuwenden.

Nichtlineare SVM

Häufig verwendete Kernfunktionen:

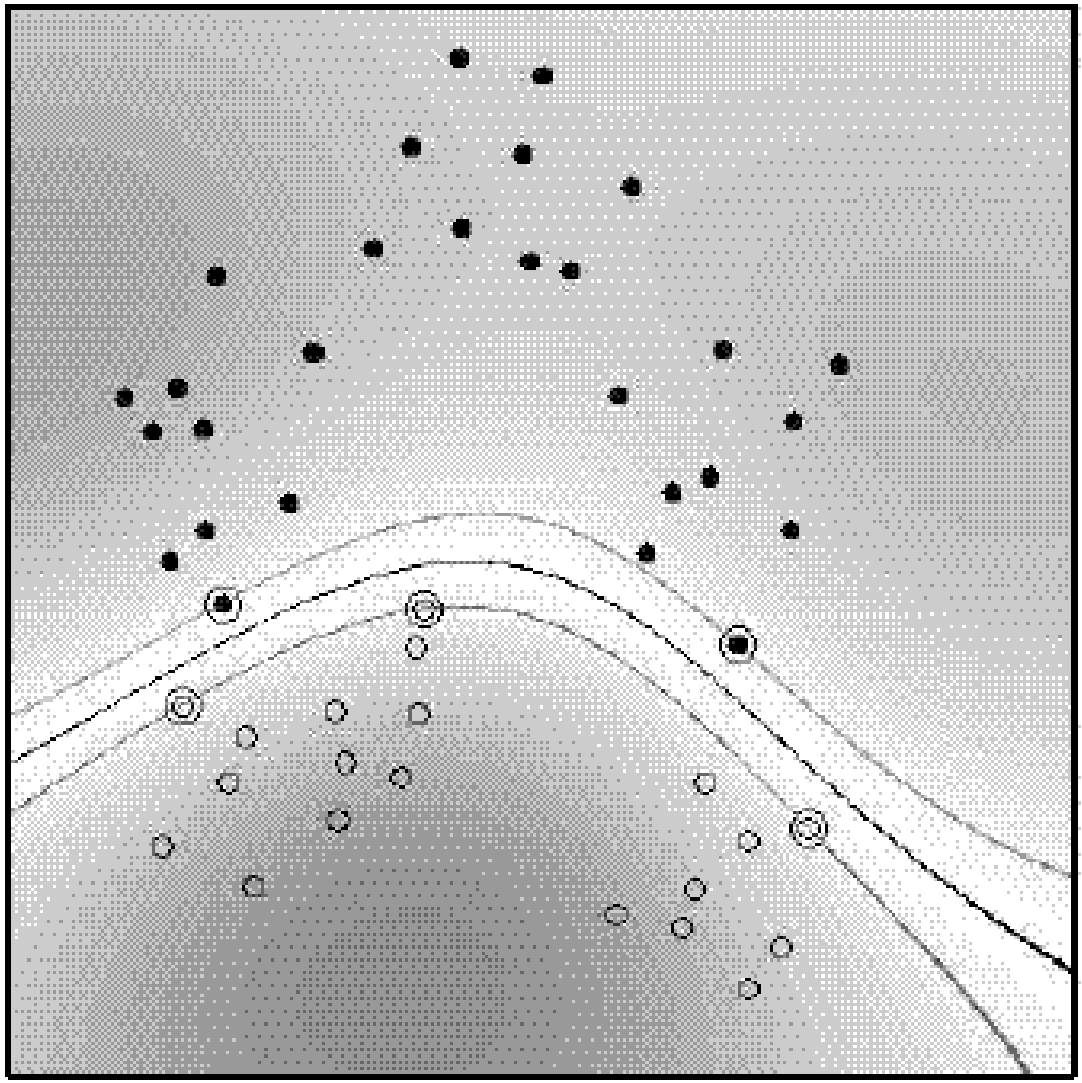
$$\begin{array}{lrcl} \text{Polynom-Kernel:} & K(x_i, x_j) & = & (x_i x_j)^d \\ \text{Gauss-Kernel:} & K(x_i, x_j) & = & e^{-\frac{||x_i - x_j||^2}{c}} \\ \text{Sigmoid-Kernel:} & K(x_i, x_j) & = & \tanh(\beta_1 x_i x_j + \beta_2) \end{array}$$

Lineare Kombinationen von gültigen Kernels $\rightarrow$ neue Kernelfunktionen

Wir müssen nicht wissen, wie der neue Merkmalsraum $\Re^m$ aussieht. Wir brauchen nur die Kernel-Funktion als ein Maß der Ähnlichkeit.

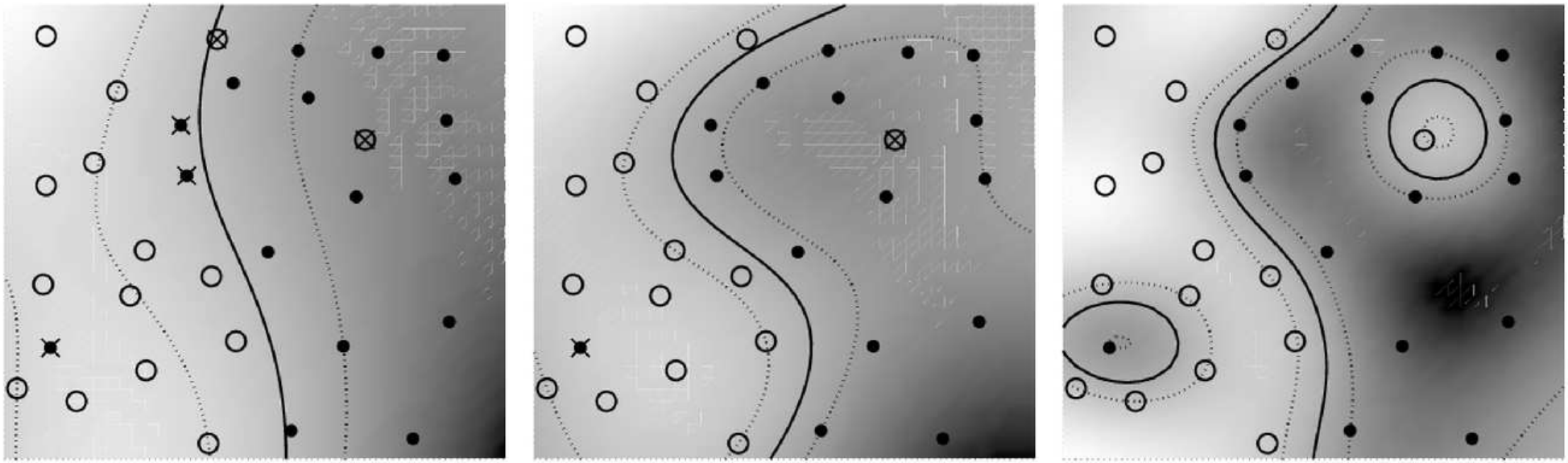
Nichtlineare SVM

Beispiel: Gauss-Kernel ($c = 1$). Die Support-Vektoren sind durch einen extra Kreis gekennzeichnet.



Nichtlineare SVM

Beispiel: Gauss-Kernel ($c = 1$) für Soft-Margin SVM.



Schlußbemerkungen

Stärken von SVM:

- SVM liefert nach derzeitigen Erkenntnissen sehr gute Klassifikationsergebnisse; bei einer Reihe von Aufgaben gilt sie als der Top-Performer
- Sparse-Darstellung der Lösung über Support-Vektoren
- Leicht anwendbar: wenig Parameter, kein a-priori-Wissen erforderlich
- Geometrisch anschauliche Funktionsweise
- Theoretische Aussagen über Ergebnisse: globales Optimum, Generalisierungsfähigkeit

Schwächen von SVM:

- Langsames speicherintensives Lernen
- "Tuning SVMs remains a black art: selecting a specific kernel and parameters is usually done in a try-and-see manner"

Schlußbemerkungen

Liste von SVM-Implementierungen unter
`http://www.kernel-machines.org/software`

LIBSVM ist die gängigste: `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`