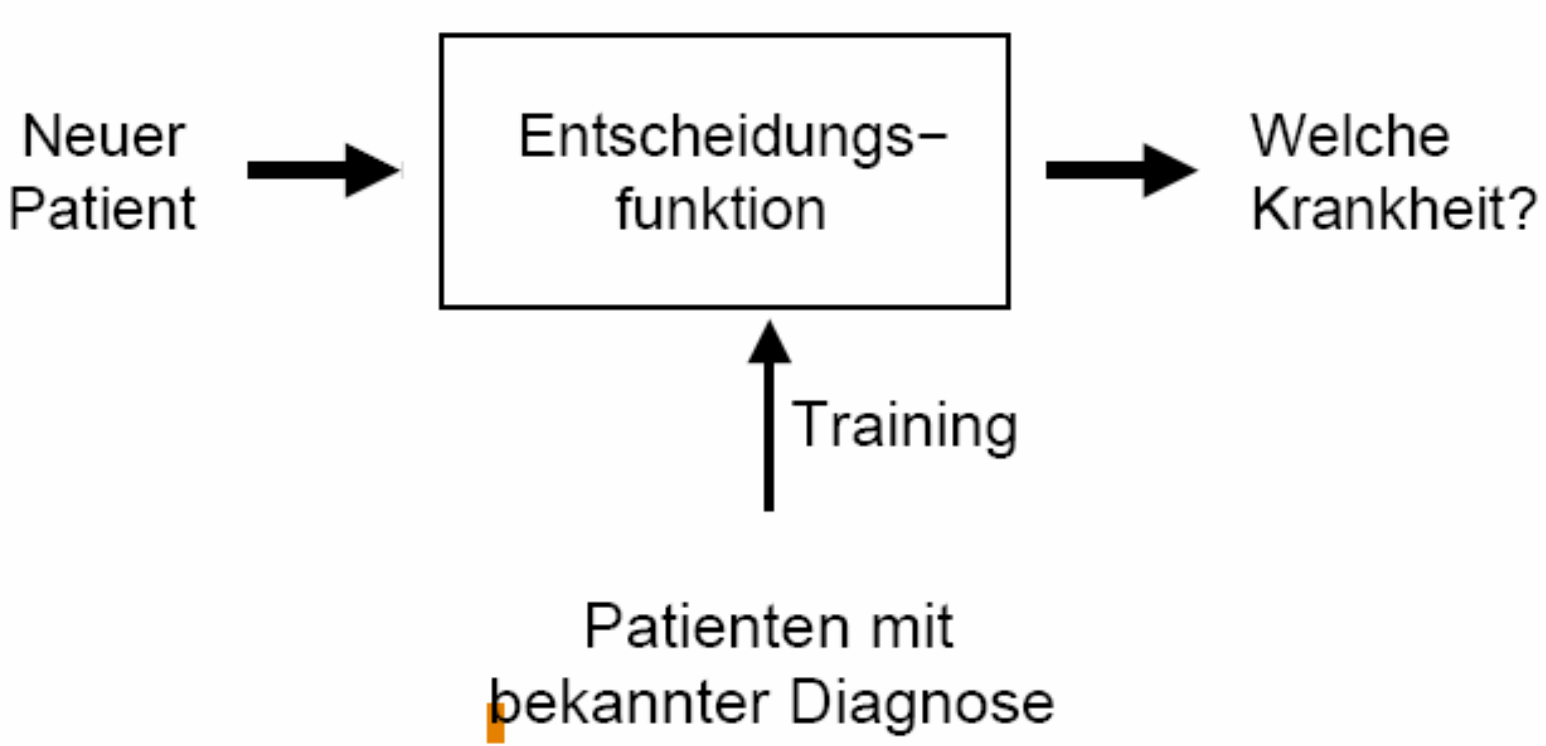
Chapter 12:
Support Vector Machines

Computergestützte Diagnostik



Überwachtes Lernen / Supervised Learning

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.

Das ist nicht schwierig, denn wir kennen ja die Diagnose der Trainingsdaten.

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

Ziel: finde eine Entscheidungsregel mit hoher Generalisierungsfähigkeit!

Lernen aus Beispielen

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

bestehend aus

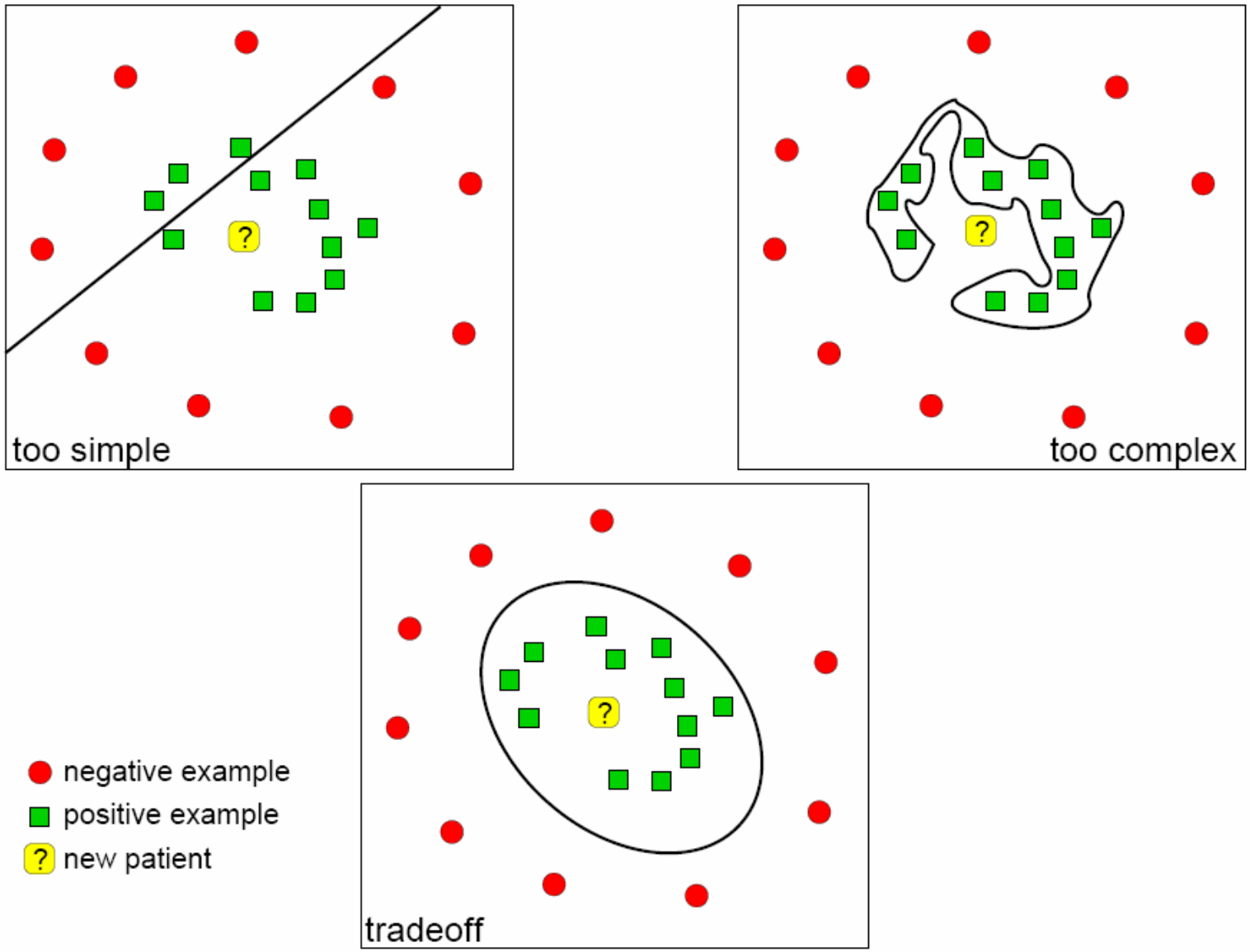
$x_i \in \mathbb{R}^g$	Punkte	Expressionsprofile
$y_i \in \{+1, -1\}$	Klassen	2 Arten von Krebs

Entscheidungsfunktion

$$f_{\mathcal{X}} : \quad \mathbb{R}^g \mapsto \{+1, -1\}$$

Diagnose = $f_{\mathcal{X}}$ (neuer Patient)

„Underfitting“ und „Overfitting“



Lineare Trennung der Trainingsdaten

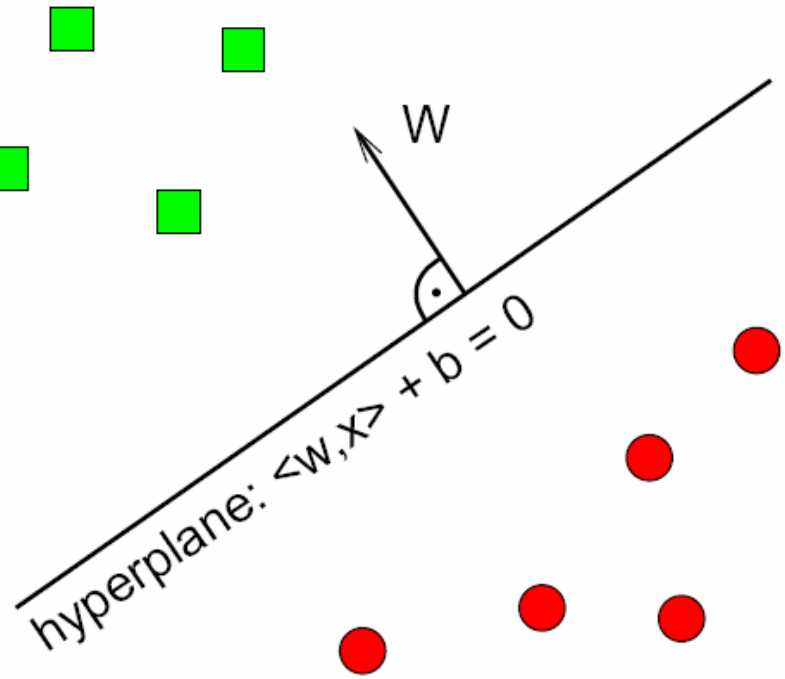
Wir fangen mit linearer Trennung an und vergrößern die Komplexität in einem zweiten Schritt durch Kernel-Funktionen.

Eine trennende Hyperebene ist definiert durch

- den Normalenvektor *w* und
- die Verschiebung *b*:

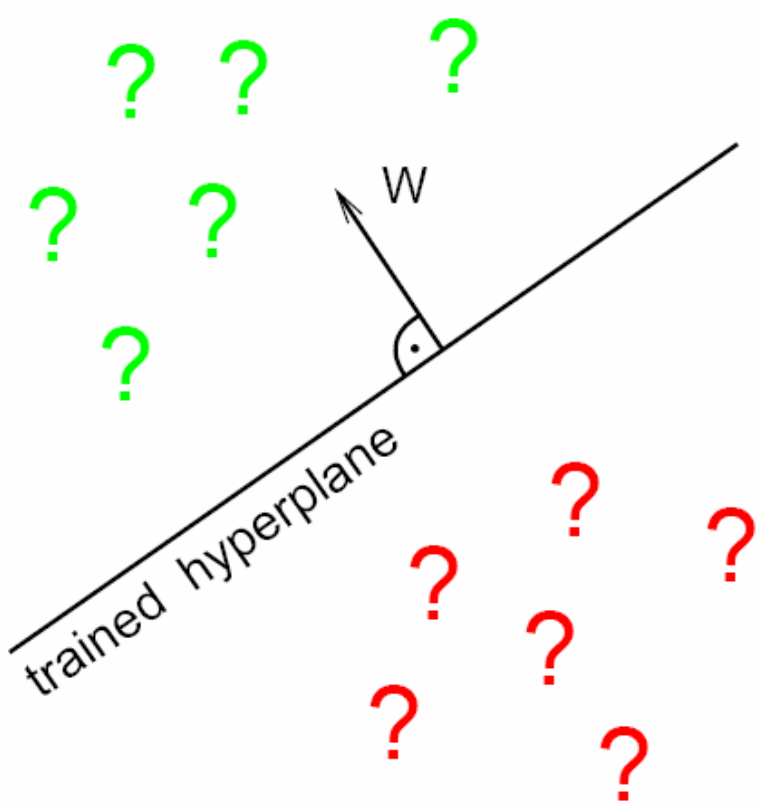
$$\text{Hyperebene } \mathcal{H} = \{ x \mid \langle w, x \rangle + b = 0 \}$$

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



Vorhersage der Klasse eines neuen Punktes

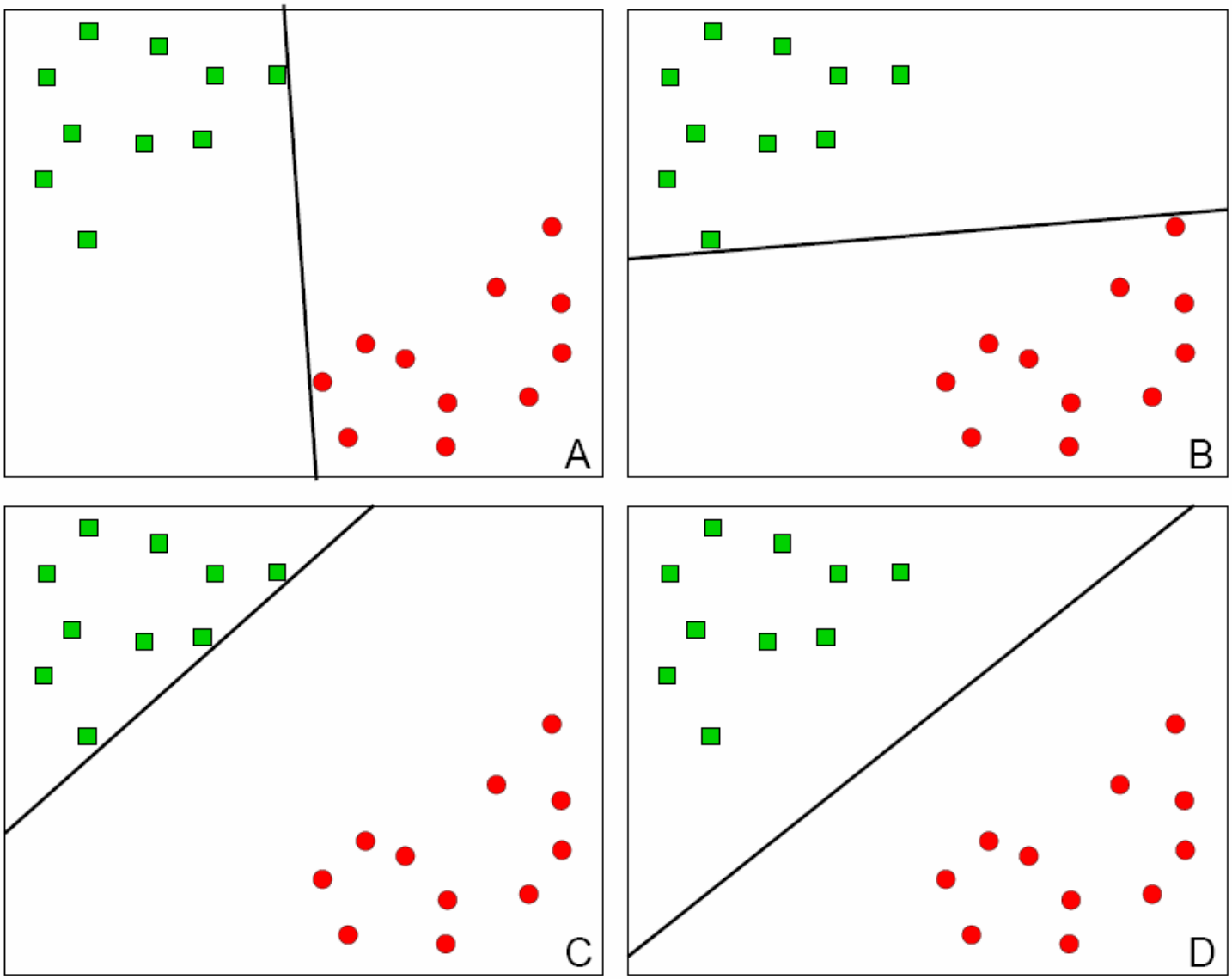
Training: Wähle *w* und *b* so dass die Hyperebene die Trainingsdaten trennt.
Vorhersage: Auf welcher Seite der Hyperebene liegt der neue Punkt?



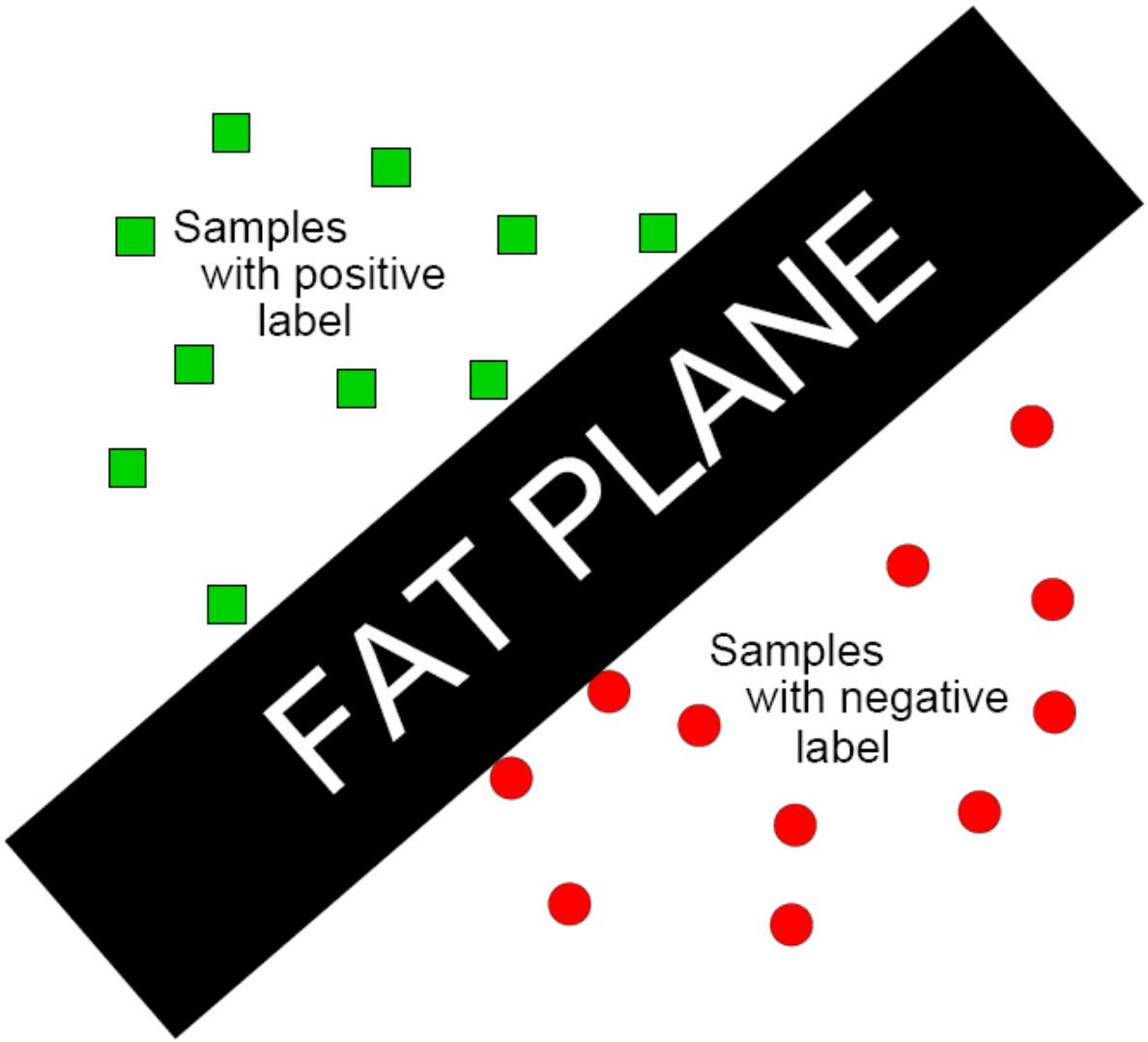
Punkte in Richtung des Normalenvektors diagnostizieren wir als **POSITIV**.

Punkte auf der anderen Seite diagnostizieren wir als **NEGATIV**.

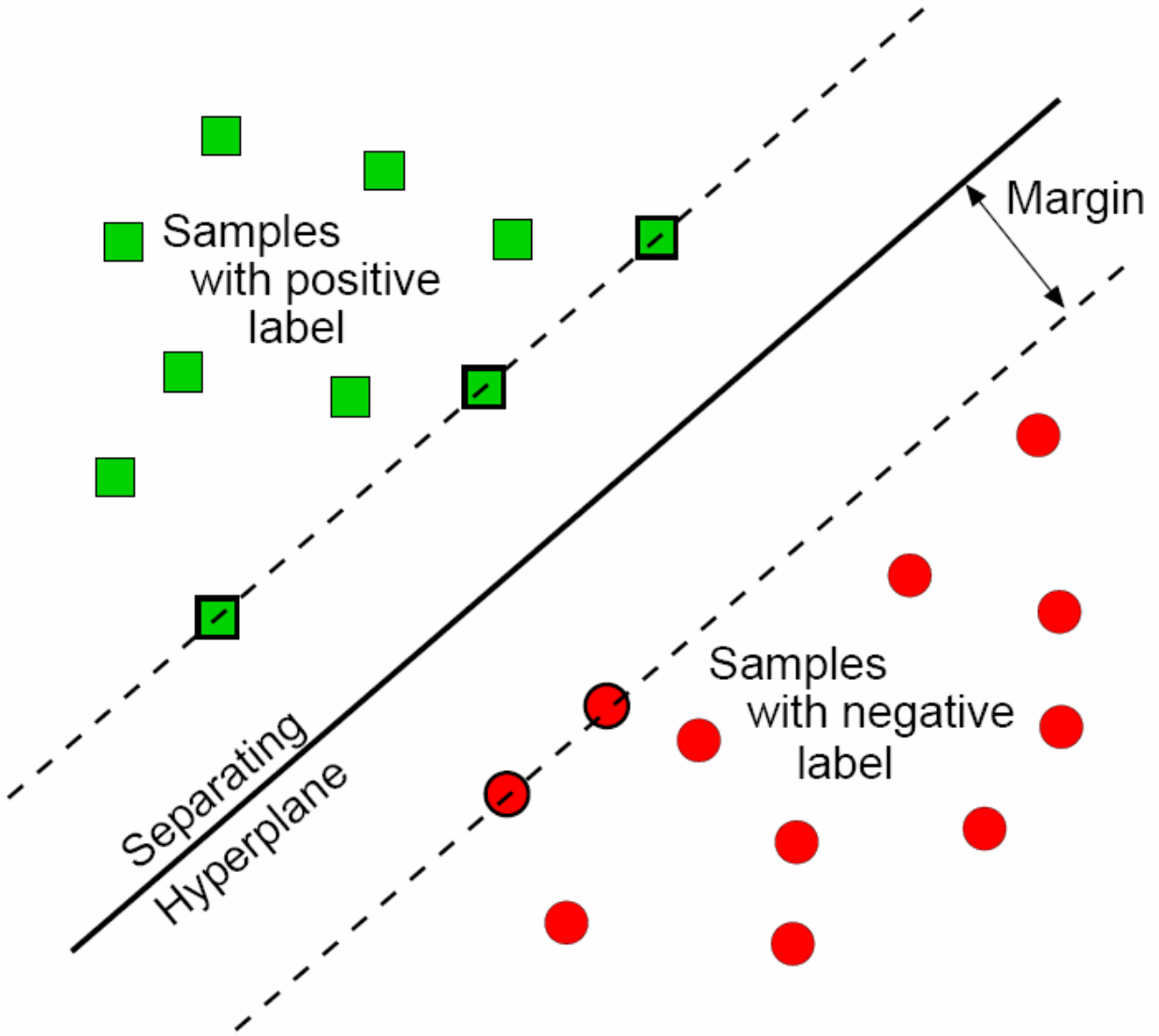
Welche Hyperebene ist die Beste? Und warum?



Kein scharfes Messer, sondern eine …

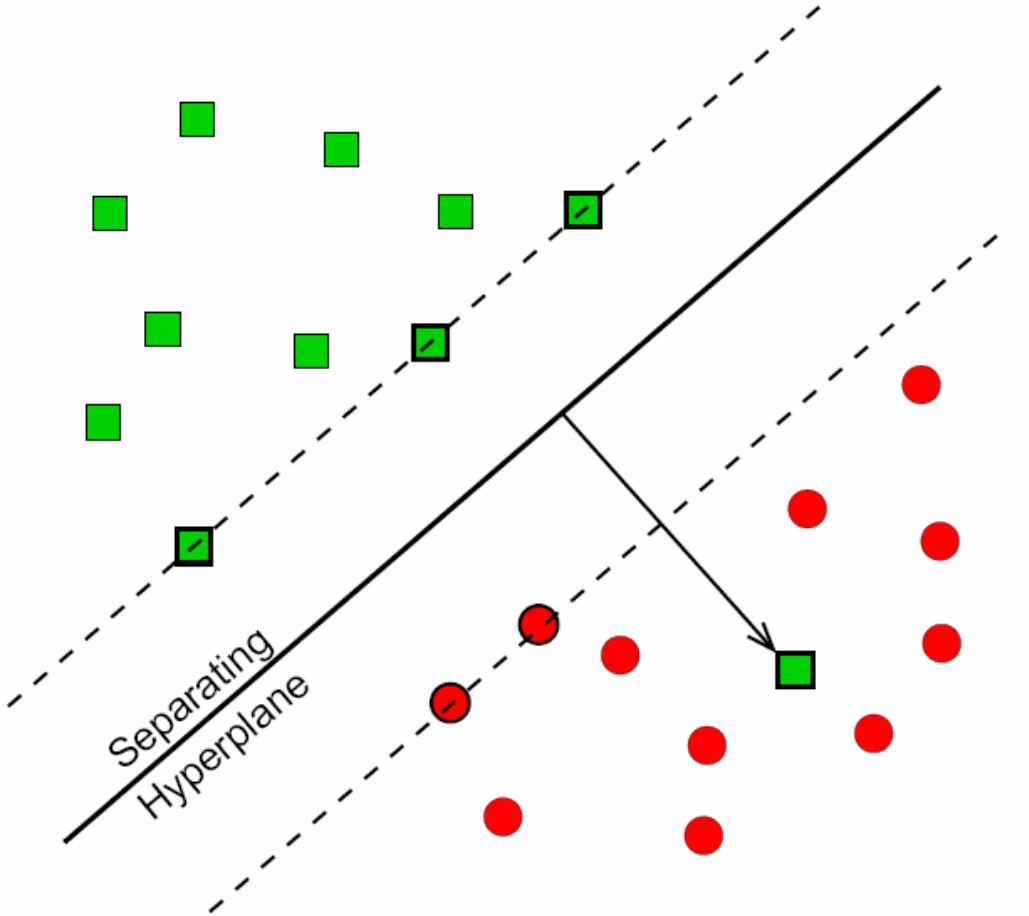


Trenne die Trainingsdaten mit maximaler Trennspanne



Nicht-trennbare Trainingsdaten

Versuche eine lineare Trennung, aber lass Fehler zu:



Strafe für Fehler: Abstand zur Hyperebene multipliziert mit *Fehlergewicht C.*

Konstruktion einer Trennebene mit max. Trennspanne

1. Eindeutigkeit durch Skalieren

2. Wie groß ist die Trennspanne?

3. Optimierungsproblem formulieren

4. Lagrange-Ansatz

5. Duales Optimierungsproblem

6. Lösung

Skalierbarkeit

Vor dem Maximieren müssen wir noch folgendes Problem beseitigen:

$$\text{Für } c \neq 0 \text{ gilt } \{\, x \mid \langle w, x \rangle + b = 0 \} \;=\; \{\, x \mid \langle cw, x \rangle + cb = 0 \}.$$

Also: (*cw*, *cb*) beschreibt die gleiche Hyperebene wie (*w*, *b*). Die Hyperebene lässt sich nicht eindeutig beschreiben.

Deshalb skalieren wir (*w*, *b*) relativ zu den Trainingsdaten $\mathcal{X} = \{x_i, y_i\}_{i=1}^{n}$, so dass

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

Eine so skalierte Hyperebene heißt auch eine kanonische Hyperebene.

Trennspanne

Der Abstand $d(x_i, \mathcal{H})$ eines Punktes x_i der Klasse $y_i \in \{+1, -1\}$ zu einer Hyperebene $\mathcal{H} = \{ x \mid \langle w, x \rangle + b = 0\}$ lässt sich berechnen als:

$$d(\mathcal{H}, x_i) = y_i(\langle \frac{w}{\|w\|}, x_i \rangle + \frac{b}{\|w\|})$$

Eben haben wir so skaliert, dass für Trainingspunkte $(x_1, +1)$ und $(x_2, -1)$ nahe an der Hyperebene gilt:

$$\langle\ w, x_1\ \rangle + b = +1 \qquad \text{und} \qquad \langle\ w, x_2\ \rangle + b = -1$$

$$\Rightarrow \langle\ \frac{w}{\|w\|}, x_1\ \rangle + \frac{b}{\|w\|} = \frac{1}{\|w\|} \qquad \text{und} \qquad \langle\ \frac{w}{\|w\|}, x_2\ \rangle + \frac{b}{\|w\|} = \frac{-1}{\|w\|}$$

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

Optimierungsproblem

Die Trennspanne maximieren heißt $\|w\|$ minimieren. Das ist gleichbedeutend mit:

$$\textbf{Minimiere } \|w\|^2$$

Um sicherzustellen, dass die Hyperebene die Trainingsdaten auch korrekt trennt, führen wir Nebenbedingungen ein:

$$y_i \cdot (\langle\, w, x_i \,\rangle + b) \;\geq\; 1 \qquad \text{für } i = 1, \ldots, n$$

Lagrange-Ansatz

Wir führen Lagrange-Multiplikatoren $\alpha_1 \geq 0$ ein und fassen das Optimierungsproblem in der sog. **Lagrange-Funktion** $L(w, b, \alpha)$ zusammen:

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

$L(w, b, \alpha)$ wird minimiert für w und b und maximiert für die α_i.

Wir setzen

$$\frac{\partial}{\partial b} L(w, b, \alpha) = 0 \qquad \text{und} \qquad \frac{\partial}{\partial w} L(w, b, \alpha) = 0,$$

und erhalten

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

Duales Problem

Wenn wir das Ergebnis wieder in $L(w, b, \alpha)$ einsetzen und ein bisschen umformen, bekommen wir das duale Problem

Maximiere:

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

unter den Nebenbedingungen:

$$\alpha_i \geq 0 \qquad \text{und} \qquad \sum_{i=1}^{n} \alpha_i y_i = 0 \ .$$

Lösung

1. Wir lösen das duale Problem und erhalten die α_i, die $W(\alpha)$ maximieren.

2. Damit berechnen wir den Normalenvektor w mit der Formel $w = \sum \alpha_i y_i x_i$ und haben die Trennebene mit maximaler Trennspanne gefunden.

3. Die Entscheidungsfunktion ist dann:

$$\begin{aligned} f(x_{neu}) &= \text{sign}(\langle w, x_{neu} \rangle + b) \\ &= \text{sign}(\sum_{i=1}^{n} \alpha_i y_i \langle x_i, x_{neu} \rangle + b) \end{aligned}$$

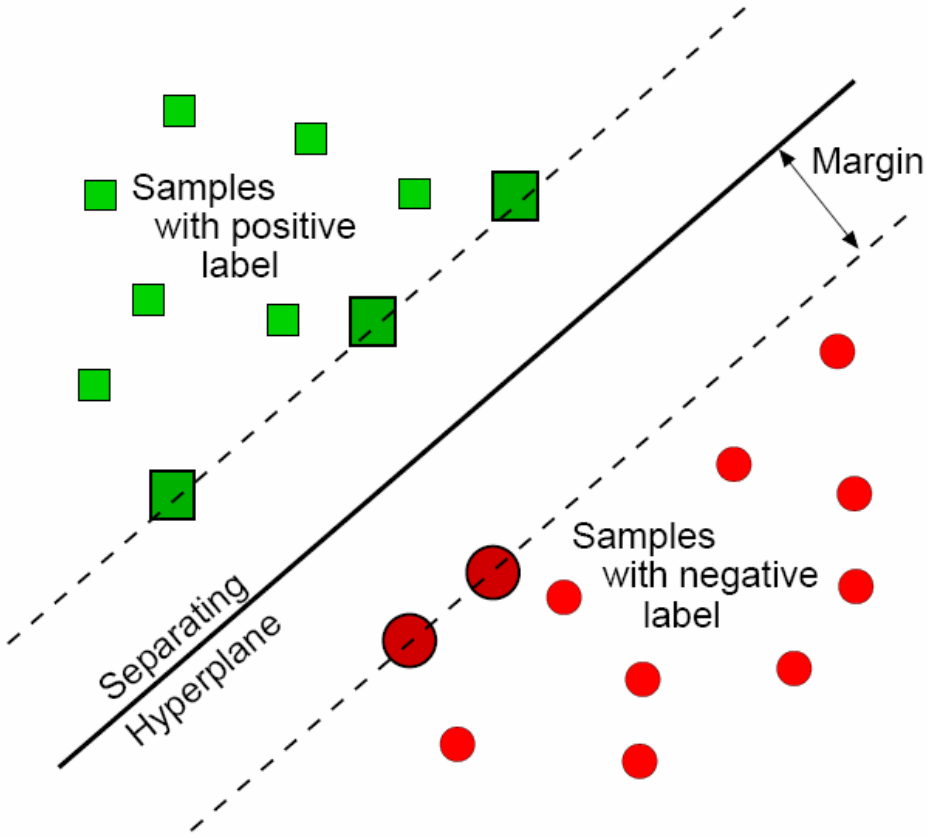
Was sind „Support Vectors“?

Die Punkte, die der Hyperebene am nächsten liegen, nennt man Support Vectors („tragende Vektoren“).

Sie allein bestimmen die Lage der Hyperebene. **Alle anderen Punkte haben keinen Einfluß!**

Nur die support vectors haben $\alpha_i \neq 0.$

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



Skalarprodukte

Wichtige Beobachtung: Sowohl im dualen Optimierungsproblem,

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

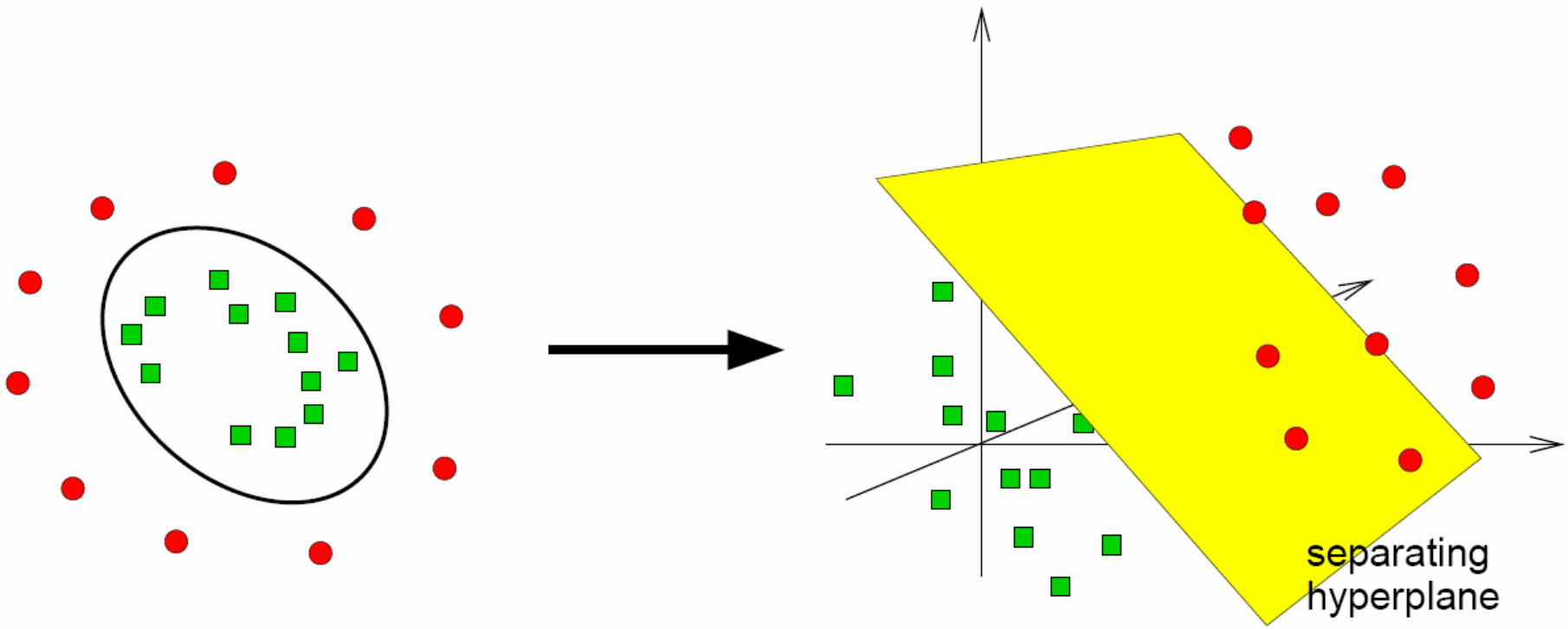
als auch in der Entscheidungsfunktion

$$f(x_{neu}) = \text{sign}(\sum_{i=1}^{n} \alpha_i y_i \langle x_i, x_{neu} \rangle + b)$$

tauchen die Trainingspunkte x_i nur in Skalarprodukten auf.

Das werden wir uns jetzt zu Nutze machen...

In hohen Dimensionen kann man einfacher trennen



Die Trennung ist …

… **komplex in 2 Dimensionen.**

… **einfach in 3 Dimensionen.**

Feature spaces

Wir verarbeiten die Datenpunkte zuerst mit einer Funktion Φ ("feature map")

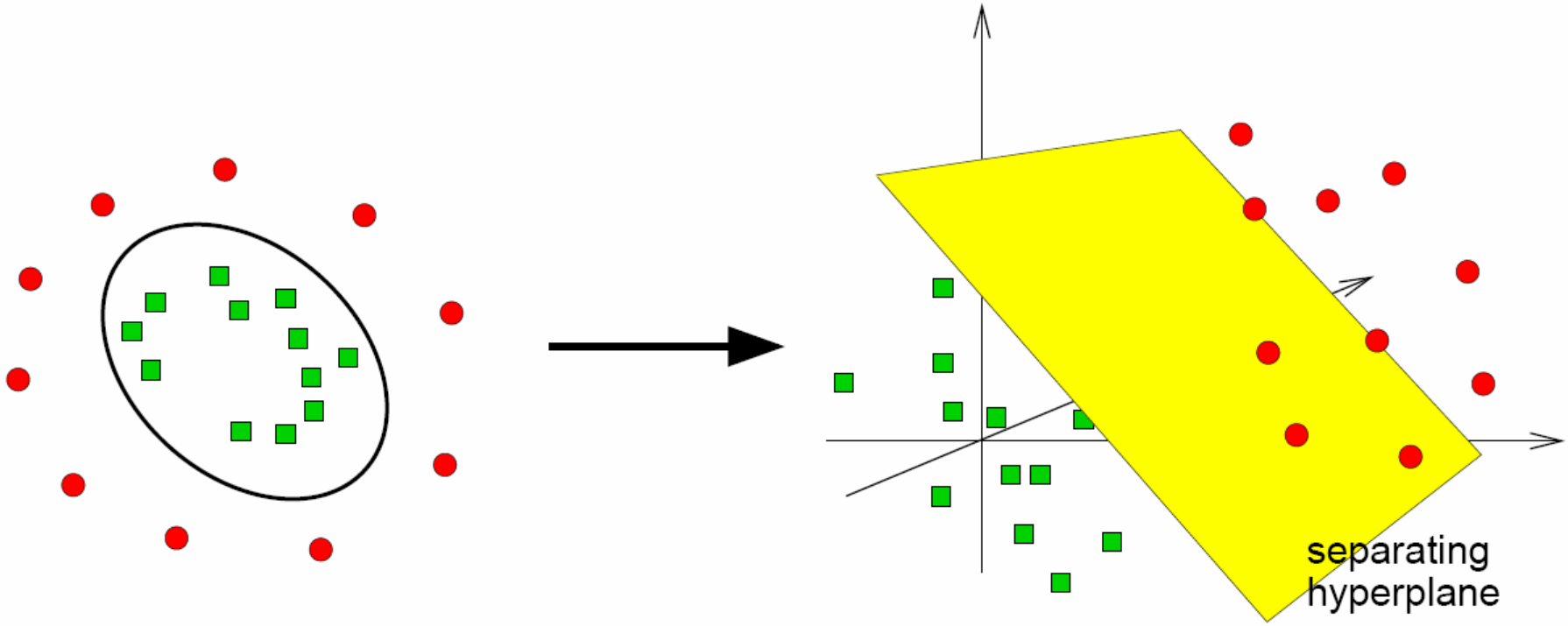
$$\begin{aligned} \Phi : \mathbb{R}^g &\rightarrow \mathcal{H} \\ x &\mapsto \Phi(x), \end{aligned}$$

wobei $\mathcal{H}$ ein Raum ist, in dem ein Skalarprodukt erklärt ist,

und dann trennen wir die Punkte $\Phi(x)$.

$\mathcal{H}$ nennt man auch den "feature space". Seine Dimension ist oft viel größer als die Anzahl der Gene (und das sind meist schon sehr viele!).

Ein Beispiel für Φ



$$\begin{aligned} \Phi : \quad \mathbb{R}^2 &\rightarrow \mathbb{R}^3 \\ (g_1, g_2) &\mapsto (z_1, z_2, z_3) := (g_1^2, \sqrt{2} g_1 g_2, g_2^2) \end{aligned}$$

Der Kernel Trick

Hyperebene mit maximaler Trennspanne im feature space

Wenn Klassifikation in hoch-dimensionalen Räumen wirklich einfacher ist, wollen wir die trennende Hyperebene dort konstruieren.

Dann müssen wir Skalarprodukte der Form $\langle\ \Phi(p), \Phi(q)\ \rangle$ ausrechnen.

Das kann sehr schwierig / unmöglich sein, wenn die Dimension von $\mathcal{H}$ zu groß wird.

Ausweg

Statt des Skalarproduktes benutzen wir eine sog. Kernel Funktion $\mathcal{K}$, die im $\mathbb{R}^g$ lebt, sich aber wie ein Skalarprodukt in $\mathcal{H}$ verhält.

$$\mathcal{K}(p, q) = \langle\ \Phi(p), \Phi(q)\ \rangle$$

Kernel-Funktionen

Ähnlichkeit im Gen-Raum: SKALARPRODUKT

$$\langle p, q \rangle = \sum_{i=1}^{g} p_i q_i = p_1 q_1 + p_2 q_2 + \ldots + p_g q_g$$

Ähnlichkeit im feature space: KERNEL FUNKTION

linear	$\mathcal{K}(p, q) = \langle p, q \rangle$
polynomiell	$\mathcal{K}(p, q) = (\gamma \langle p, q \rangle + c_0)^d$
radial basis function	$\mathcal{K}(p, q) = \exp\left(-\gamma \|p - q\|^2\right)$

Wir rechnen ein Beispiel:

Gegeben zwei Expressionsprofile $p = (g1, g2)$ und $q = (h_1, h_2)$.
Wir wenden auf p und q die Abbildung $\Phi : (g_1, g_2) \mapsto (g_1^2,\ \sqrt{2}g_1g_2,\ g_2^2)$ an und dann berechnen wir das Skalarprodukt.

$$\begin{aligned}
\langle\ \Phi(p), \Phi(q)\ \rangle &= (g_1^2, \sqrt{2}g_1g_2, g_2^2)(h_1^2, \sqrt{2}h_1h_2, h_2^2)^t \\
&= g_1^2h_1^2\ +\ 2\ g_1h_1\ g_2h_2\ +\ g_2^2h_2^2 \\
&= (g_1h_1\ +\ g_2h_2)^2 \\
&= \langle\ p, q\ \rangle^2\ =:\ \mathcal{K}(p, q)
\end{aligned}$$

Wir können also das Skalarprodukt zwischen $\Phi(p)$ und $\Phi(q)$ ausrechnen, ohne die Funktion Φ anzuwenden. Es reicht, das Quadrat des Skalarproduktes von p und q im $\mathbb{R}^2$ zu berechnen.

Wann ist eine Funktion eine Kernel-Funktion?

Die Antwort gibt das **Theorem von Mercer**.

1. Die Funktion $\mathcal{K}$ muß symmetrisch sein: $\mathcal{K}(p_i, p_j) = \mathcal{K}(p_j, p_i)$,

2. die Kernel-Matrix K mit

$$K_{ij} := \mathcal{K}(p_i, p_j)$$

ist positiv (semi-)definit für alle Trainingsdaten $p_1, p_2, \ldots, p_n$, d. h.

$$a^t K a = \sum_{i,j} a_i a_j K_{ij} \geq 0 \qquad \forall a \in \mathbb{R}^n$$

Warum ist das ein Trick?

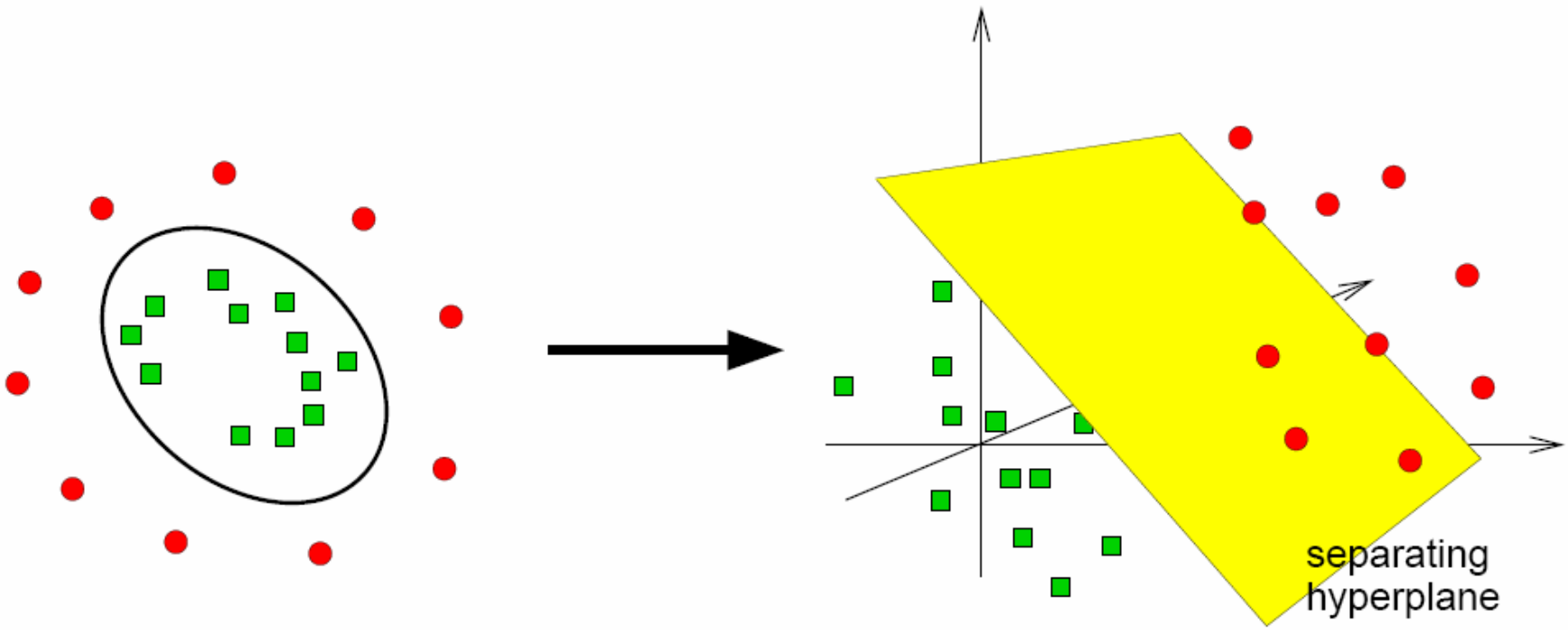
Wir müssen nicht wissen, wie der feature space wirklich aussieht wir brauchen nur die Kernel-Funktion als ein Maß der Ähnlichkeit.

Das hat was von schwarzer Magie: wir wissen nicht, was im Kernel passiert, wir interessieren uns nur für das Ergebnis.

Trotzdem haben wir die geometrische Interpretation der trennenden Hyperebene:

SVMs sind transparenter als z. B. künstliche Neuronale Netze.

Kernel Trick: Zusammenfassung



Nichtlineare Trennung
von Vektoren
im Gen-Raum
mit Kernel-Funktion

=

Lineare Trennung
von Vektoren
im feature space
mit Skalarprodukt

Support Vector Machines

Eine *Support Vector Machine* ist

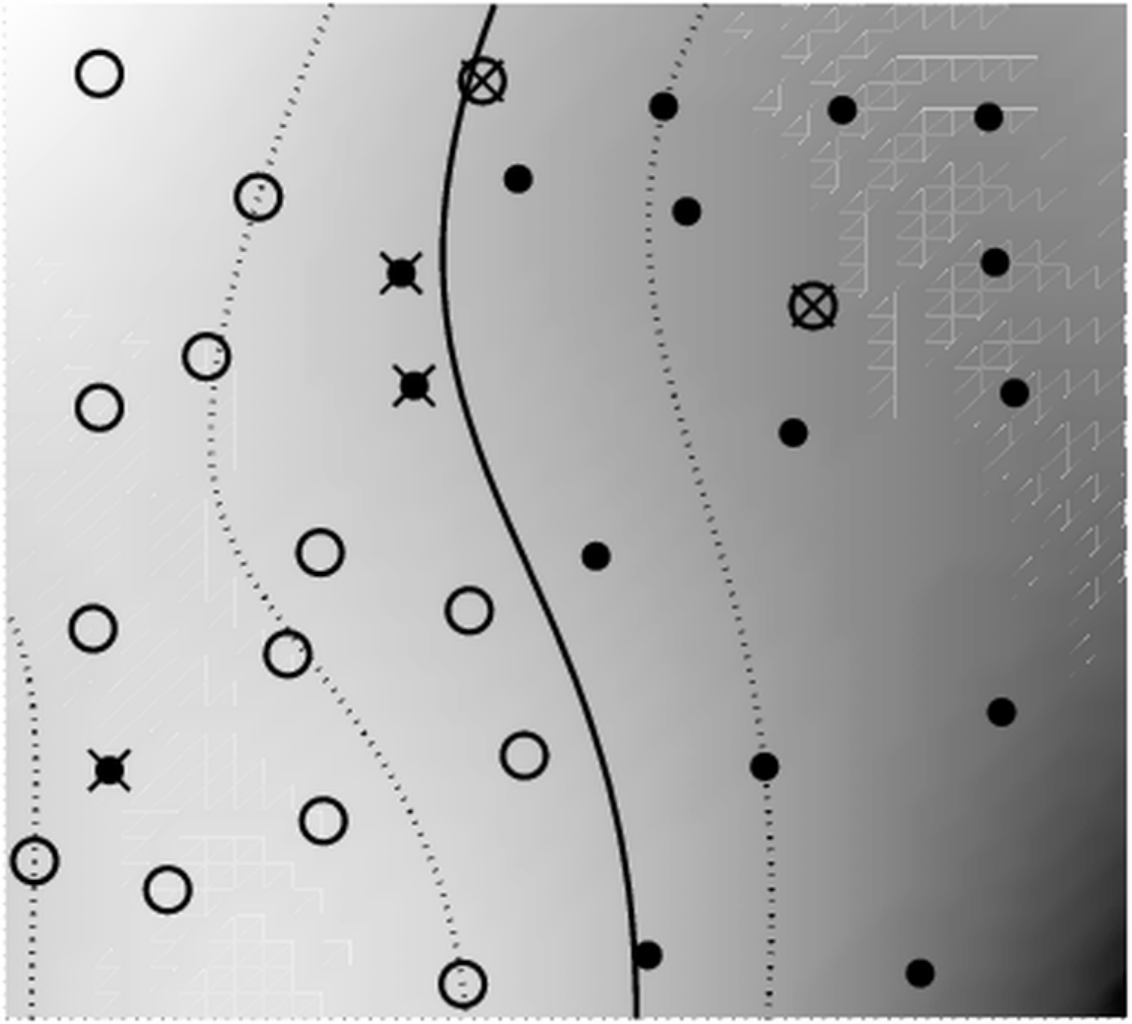
1. eine Hyperebene mit maximaler Trennspanne im *feature space*

2. konstruiert im Gen-Raum durch eine Kernel-Funktion.

Parameters of SVM

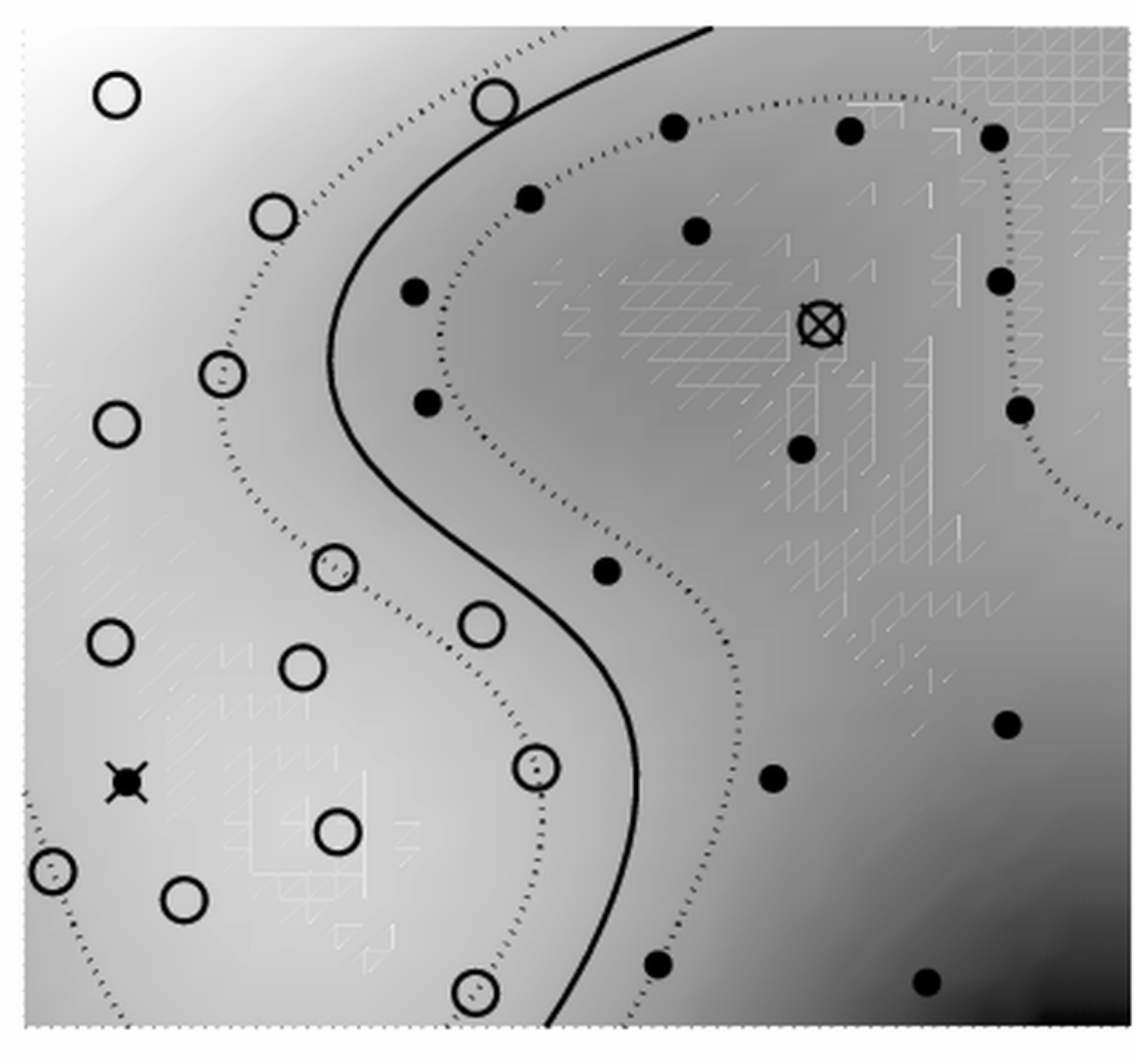
Kernel Parameter	γ:	Weite der gaußschen Glocken Koeff. im Polynom [$= 1$]
	d:	Grad des Polynoms [$= 3$]
	c_0	additive Konstante im Polynom [$= 0$]
Fehlergewicht	C:	Einfluss von Trainingsfehlern

SVM@work: niedrige Komplexität



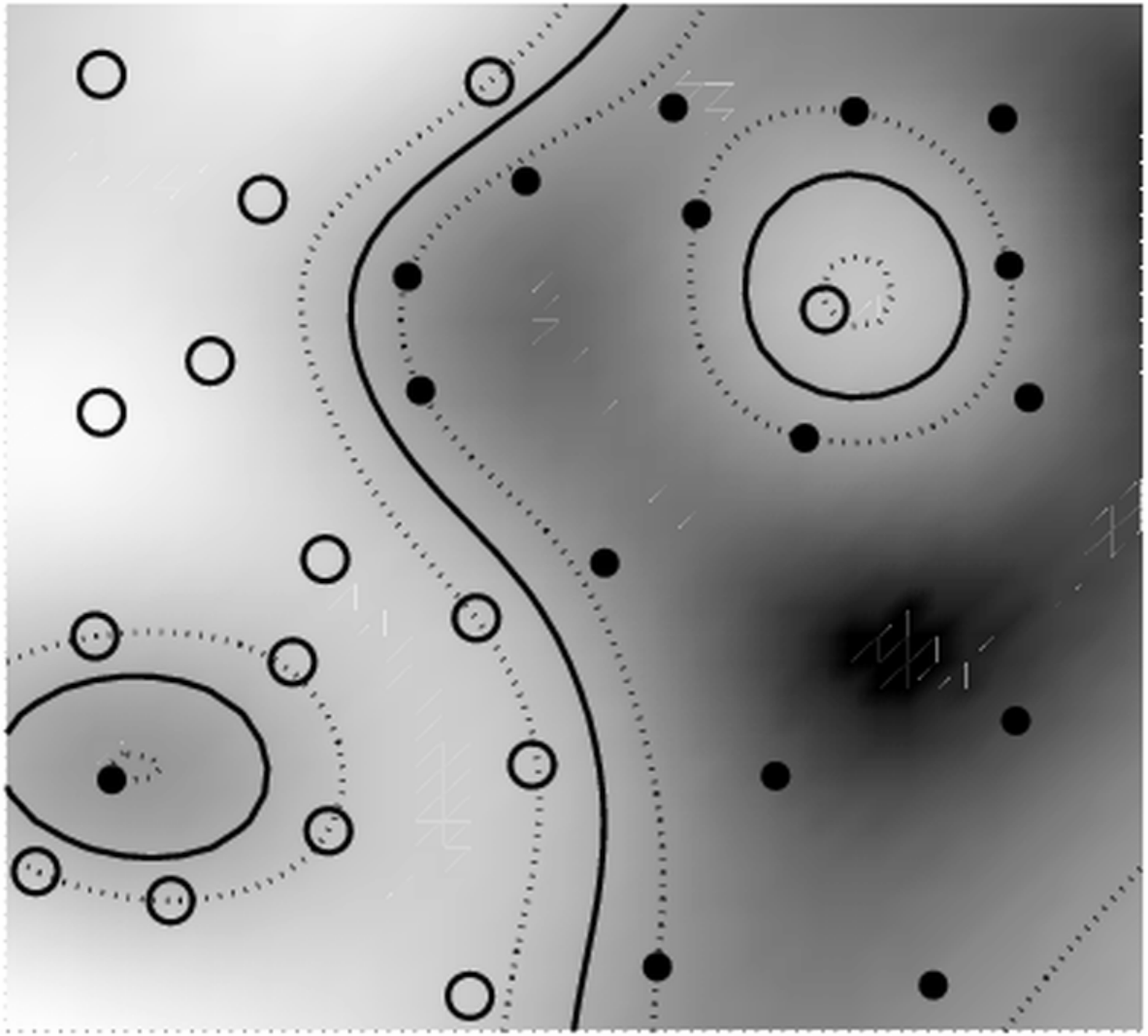
aus: SCHÖLKOPF and SMOLA, *Learning with Kernels*, MIT Press 2002, p217

SVM@work: mittlere Komplexität



aus: SCHÖLKOPF and SMOLA, *Learning with Kernels*, MIT Press 2002, p217

SVM@work: hohe Komplexität



aus: SCHÖLKOPF and SMOLA, *Learning with Kernels*, MIT Press 2002, p217