Support Vector Machines, Kernels

Katja Kunze

13.01.04

Inhalt:

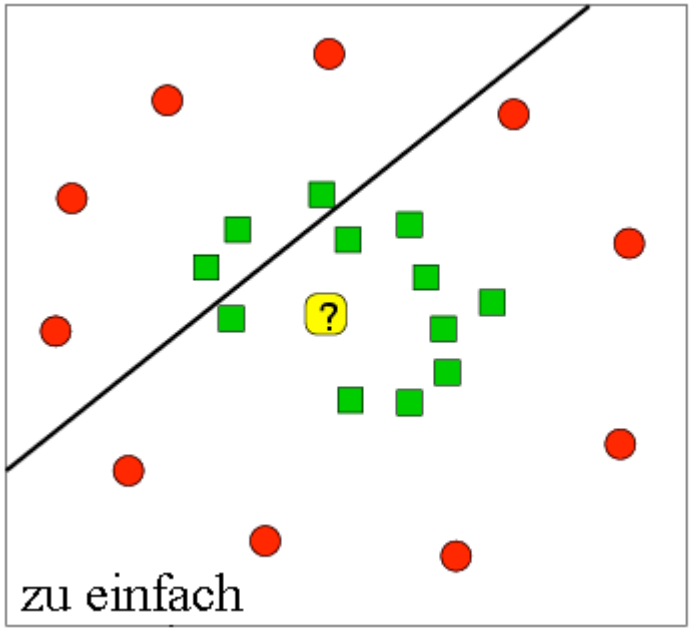
- Grundlagen/Allgemeines
- Lineare Trennung/Separation
  - Maximum Margin Hyperplane
  - Soft Margin SVM
- Kernels
- Praktische Anwendungen

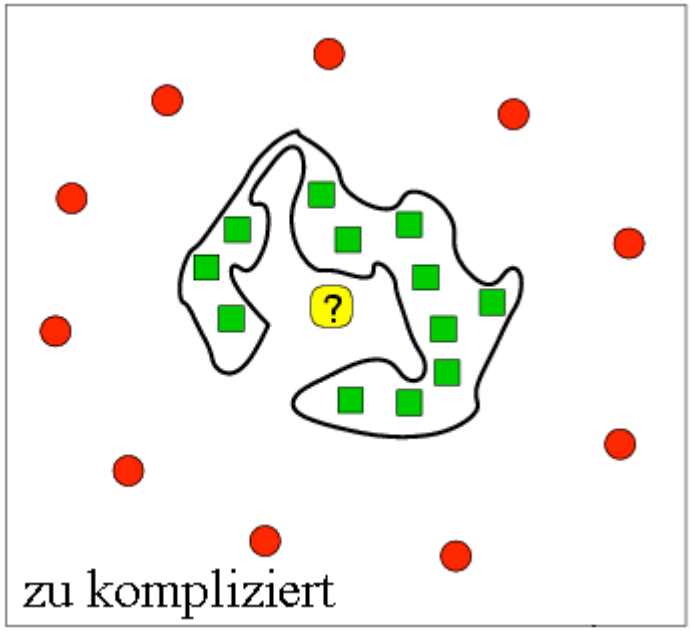
Allgemeines

- Entwickelt von V. Vapnik zur Klassifizierung von Daten
- Basiert auf der Statistischen ‚Lern-Theorie'

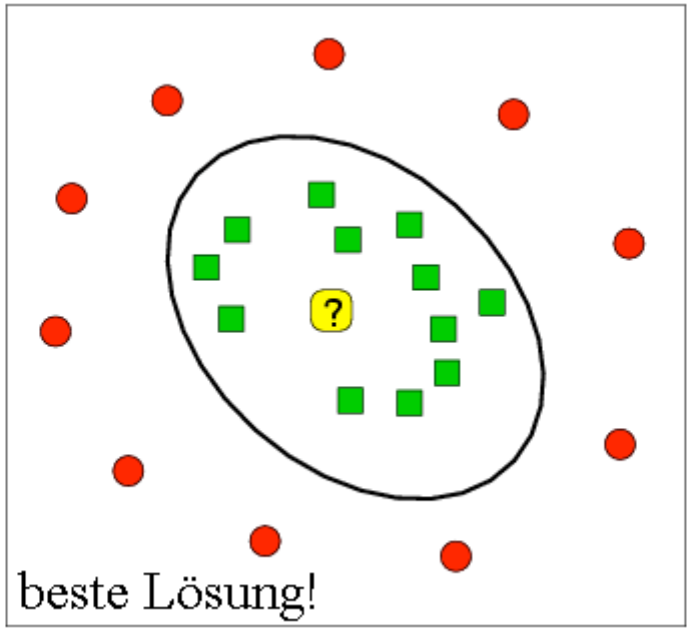
Speziell:

- Verfahren mit guter Generalisierungsmöglichkeit
- Komplexität der Optimierung unabhängig von der Anzahl der Features und deren Dimension





„Underfitting“



„Overfitting“

Vereinigung von 4 Konzepten in SVM

1. Minimierung und Fixierung des Fehlerrisikos, Daten falsch zu klassifizieren
2. Daten niedriger Dimension werden abgebildet auf einen neuen Featurespace _ Diese Variablen werden zum Lernen benutzt.
3. Anwendung von linearen Schätzern/ Näherungsfunktionen auf den neuen Feartureraum _ Minimierung des Fehlers bei der Näherung
4. Duales Optimierungsproblem

Begriffe:

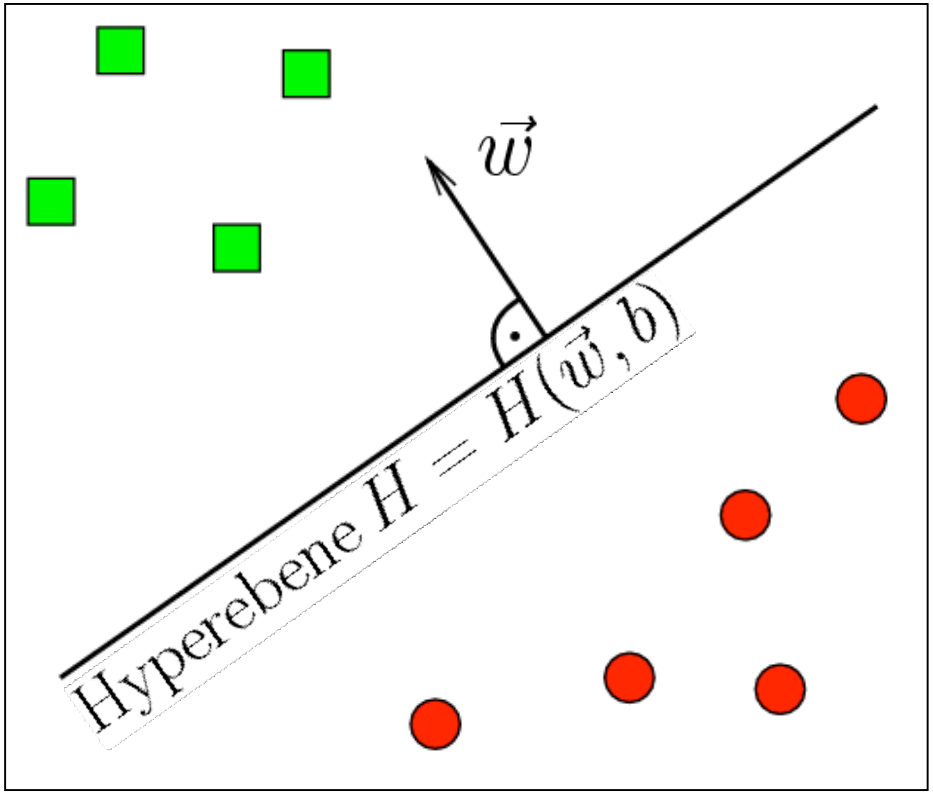
Trainigsdaten: Daten von denen bekannt ist, zu welcher Klasse sie gehören

Lernen/Trainieren: Ableitung einer Entscheidungsregel, die die beiden Klassen voneinander trennt.

Generalisierungsfähigkeit: wie gut ist die Entscheidungsregel bei neuen Daten?

_ **Ziel**: Finden einer Entscheidungsregel mit hoher Generalisierungsfähigkeit

Lineare Trennung der Trainingsdaten



Gegebene Trainingsdaten in Vektorform:

$$(\vec{x_1}, y_1)...(\vec{x_n}, y_n)$$

wobei $y_1 \in \{-1, 1\}$

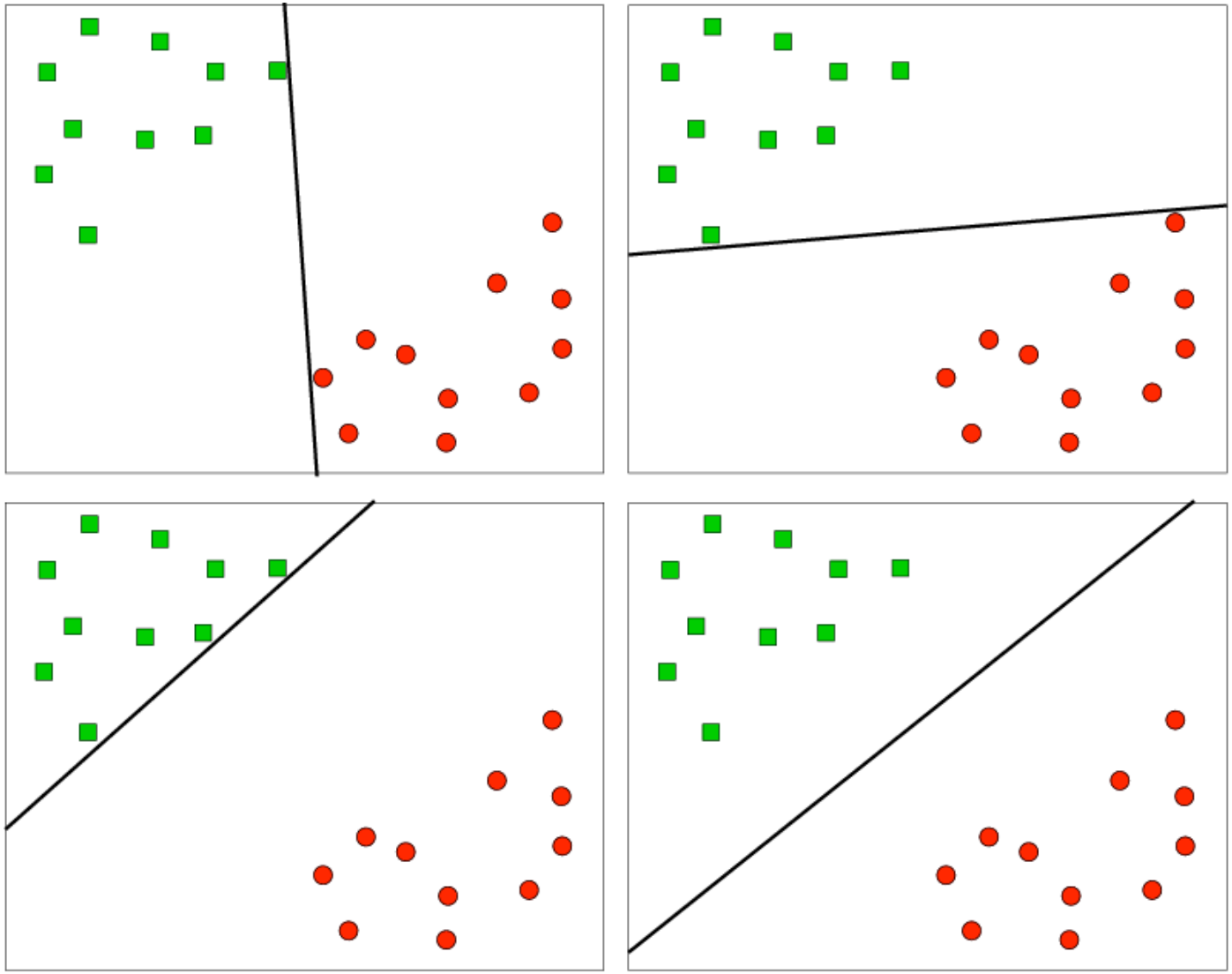
Hyperebene *H:*

$$H = H(\vec{w}, b) := \{\vec{x} \in FS | \langle \vec{x}, \vec{w} \rangle + b = 0\}$$

Abstand zur Ebene:

$$dist(\vec{x}, H) = \left| \frac{1}{||\vec{w}||} (\langle \vec{x}, \vec{w} \rangle + b = 0) \right|$$

Wie separiere ich die Klassen? _ Verschiede Möglichkeiten:



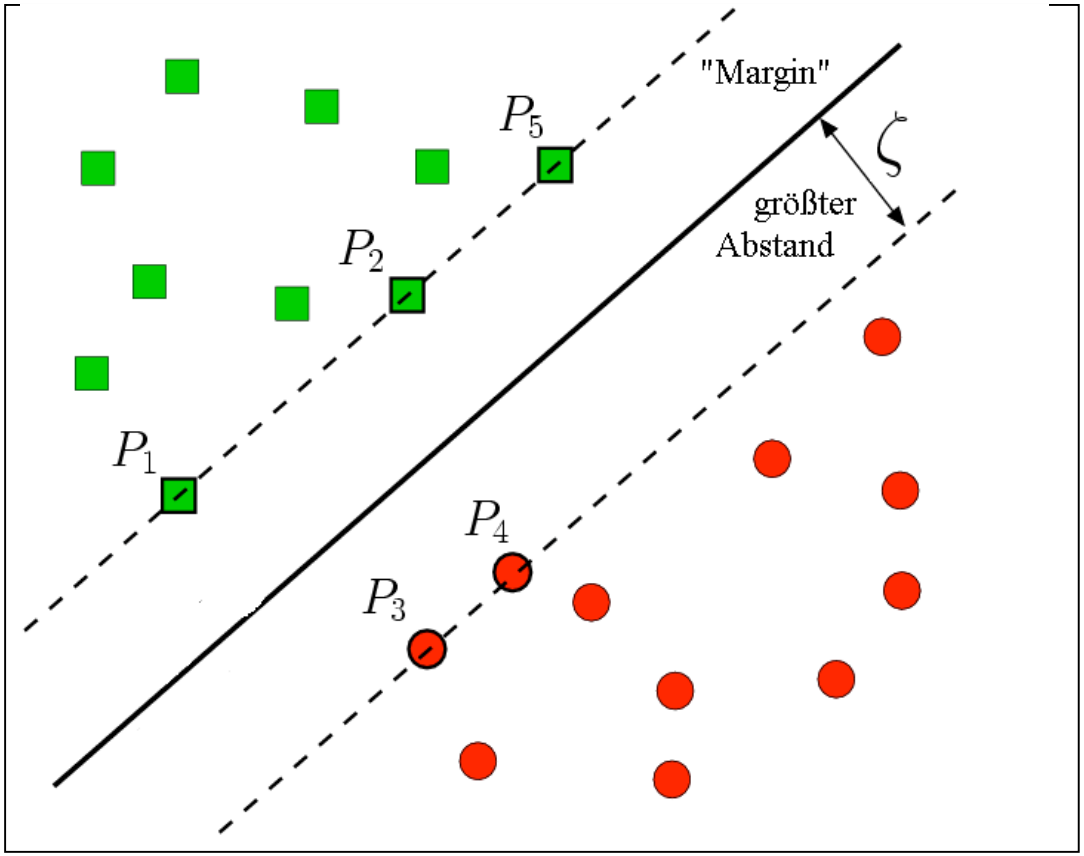
Welche Ebene ist optimal?

Linearer Fall:

Maximum Margin Hyperplane (MMH)

ζ = Minimaler Abstand eines Objektes zur Ebene

P_i = Objekte, für die ζ der kleinste Abstand ist = ***Support Vectors***



Optimale Ebene:

- in der Mitte des größten Spalts zwischen den Klassen
- wird nur durch die Support Vectors bestimmt

Vektorprodukt: $$\langle \vec{x}, \vec{y} \rangle = \sum_{i=1}^{d} x_i \cdot y_i$$

Darstellung der Ebene:

$$H = H(\vec{w}, b) := \{\vec{x} \in FS | \langle \vec{x}, \vec{w} \rangle + b = 0\}$$

Unterteilung in 2 Klassen:

$$y_i = sgn(\langle \vec{x}, \vec{w} \rangle + b)$$

wobei

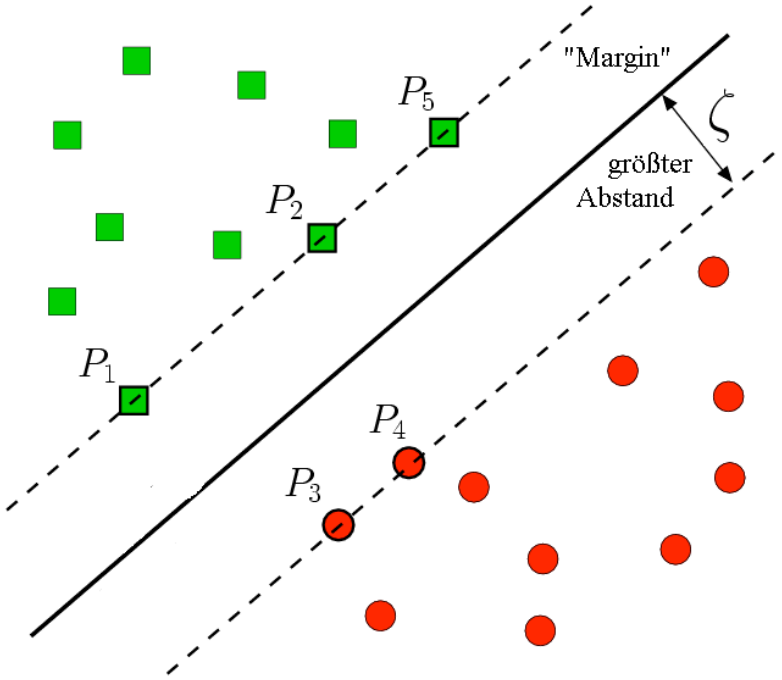
$$y_i = \begin{cases} 1 & = & \text{Objekt oberhalb der Hyperebene} \\ -1 & = & \text{Objekt unterhalb der Hyperebene} \end{cases}$$

Abstand :

$$\zeta = \min_{\vec{x_i}} \left| \frac{1}{||\vec{w}||}(\langle \vec{x}, \vec{w} \rangle + b = 0) \right|$$

Man erhält:

$$\zeta \leq y_i \frac{1}{||\vec{w}||}(\langle \vec{x}, \vec{w} \rangle + b = 0)$$



_ Die Gleichung hängt nur ab von ζ und $||\vec{w}||$.

Die Ebene ist kanonisch,

deshalb gilt:

$$\zeta \cdot ||\vec{w}|| = 1 \quad \Rightarrow \quad \zeta = \frac{1}{||\vec{w}||}$$

_ Maximierung von $\frac{1}{||\vec{w}||}$ unter der Bedingung

$$y_i(\langle \vec{x}, \vec{w} \rangle + b) \geq 1$$

oder

$$y_i(\langle \vec{x}, \vec{w} \rangle + b) - 1 < 0$$

_ entspricht der Minimierung von $\frac{1}{2}||\vec{w}||^2$

= Beschränktes Optimierungsproblem

Lösung mit den Lagrangemultiplikatoren
und der Lagrangefunktion:

Für $\alpha_i > 0$ gilt:

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

L kann maximiert werden für *b* und für $\vec{w}$:

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

Man erhält: $$\sum_{i=1}^{d} \alpha_i y_i = 0$$

und: $$\vec{w} = \sum_{i=1}^{d} \alpha_i y_i x_i$$

Durch Substitution in die Lagrangefunktion erhält man das ‚Duale Optimierungs-Problem‘:

$$maximize_{\alpha \in R^n}\ L(\vec{\alpha}) =$$
$$\sum_{i=1}^{d} \alpha_i - \frac{1}{2} \sum_{i=1}^{d} \sum_{i=1}^{d} \alpha_i \alpha_j y_i y_j \langle \vec{x_i}, \vec{x_j} \rangle$$

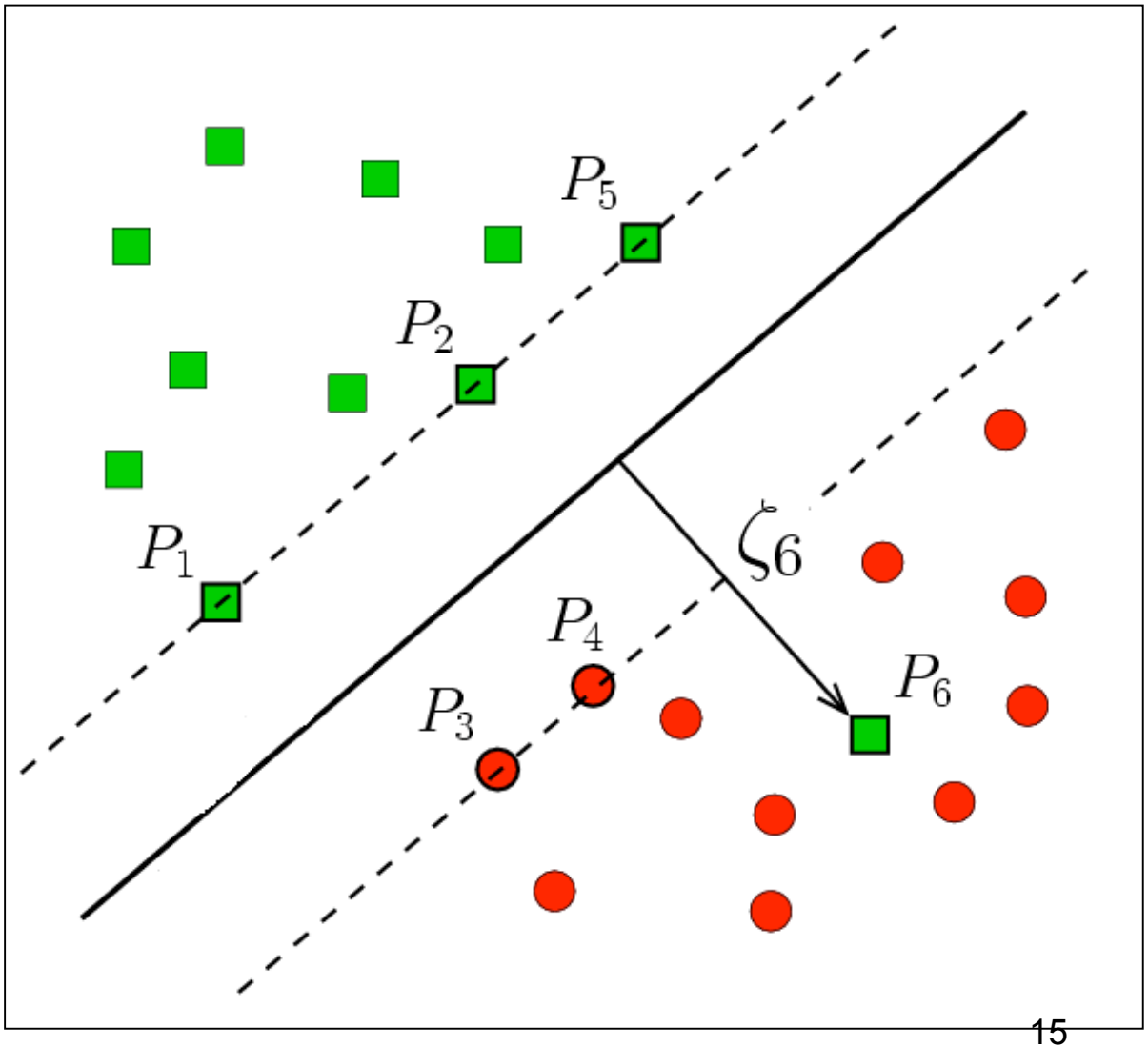
Lösung mit Algorithmen aus der Optimierungs-Theorie.

Nicht-Linearer Fall:

Soft Margin Hyperplane (SMH)

Einführung von Variablen ζ_i , die den Abstand eines falschpositionierten Punktes beschreiben

_ Zulassen eines Fehlers



_ Minimierung des Fehlers, der bei Linearer Seperation auftritt.

Abstand der Hyperebene multipliziert mit Fehlergewicht *C*:

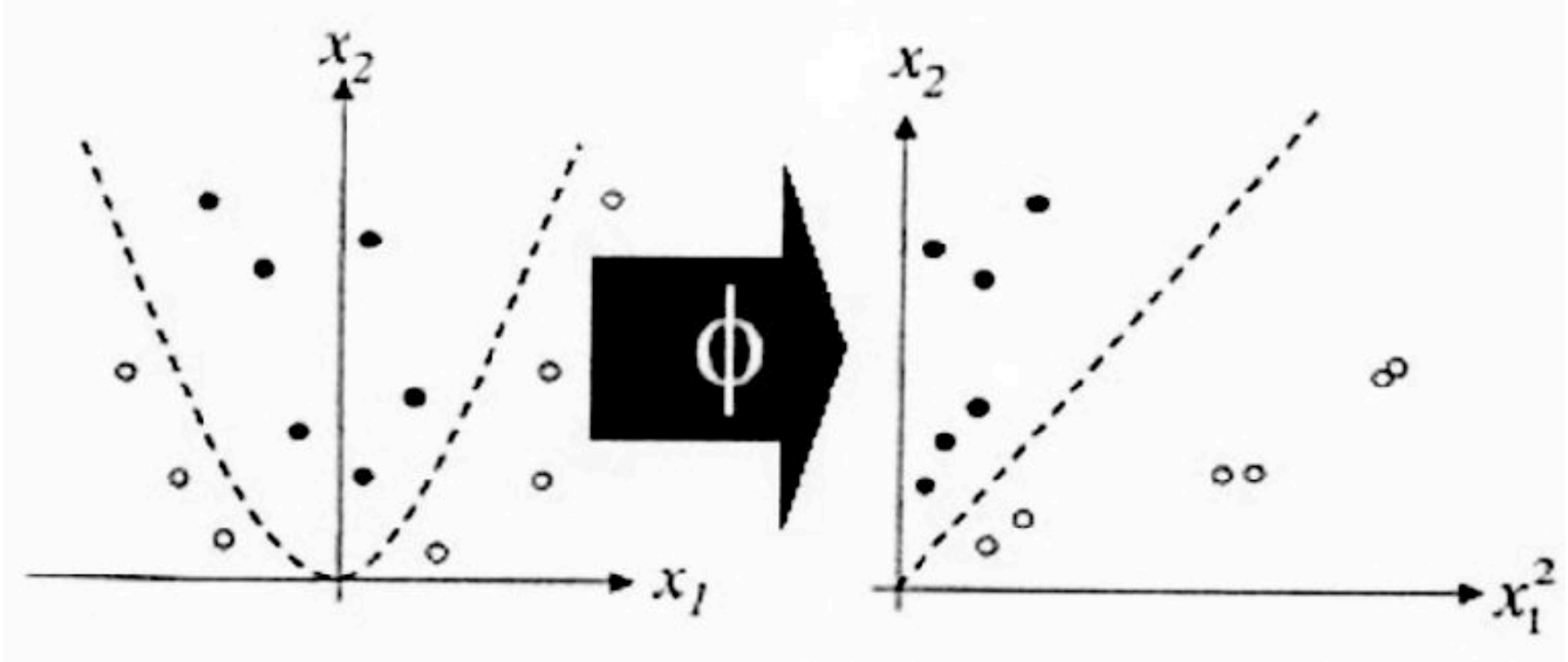
$$minimize\ \tfrac{1}{2}||\vec{w}||^2 + C\,\Sigma_{i=1}\,n\zeta_i$$

unter der Bedingung

$$y_i(\langle\vec{x},\vec{w}\rangle + b) \geq 1 - \zeta_i, \quad \zeta \geq 0\ \forall\ i$$

Kernels

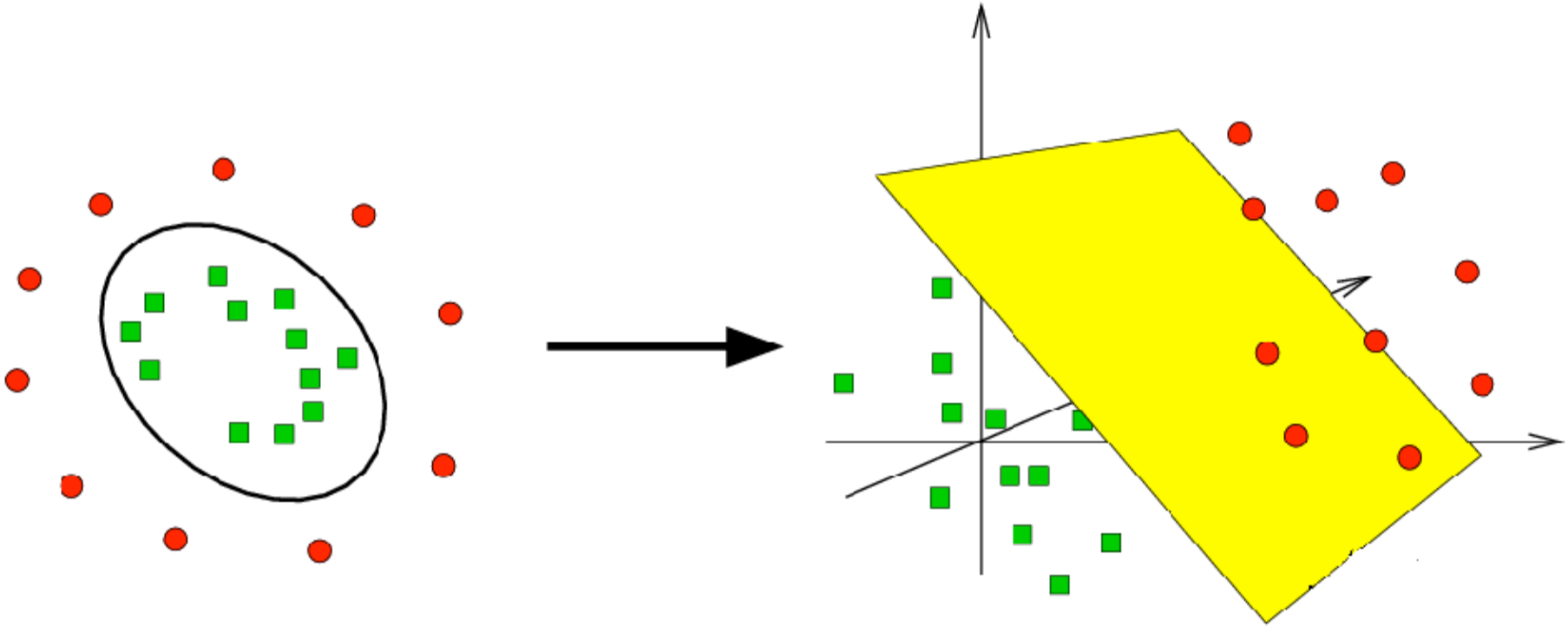
Verwendung von Kernelfunktionen bei Daten, die nicht durch eine lineare Seperation klassifiziert werden können:



1.Schritt: Transformation der Daten in einen neuen Raum, indem Lineare Trennung möglich ist. .

2.Schritt: Anwendung von Linearen Methoden auf die Daten.

Transformation in einen Raum höherer Dimension:



$$\mathbb{R}^2 \quad \rightarrow \quad \mathbb{R}^3$$

$$\Phi : (x_1, x_2) \mapsto (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$

_ Jetzt ist die Konstruktion einer Hyperebene einfach.

Jetzt ist aber Folgendes Skalarprodukt zu berechnen:

$$\langle \Phi(\vec{x}), \Phi(\vec{y}) \rangle$$

_ zu schwierig, wenn die Dimension sehr hoch ist!

Kerneltrick:

Statt Skalarprodukt benutzt man eine Kernelfunktion *K*, die sich wie ein Skalarprodukt verhält:

$$K_\Phi(\vec{x}, \vec{y}) := \langle \Phi(\vec{x}), \Phi(\vec{y}) \rangle$$

Rechenbeispiel

Gegeben: $\vec{x} = (x_1, x_2) \quad \vec{y} = (y_1, y_2)$

Wir wenden folgende Abbildung darauf an:

$$\Phi : (x_1, x_2) \mapsto (x_1^2, \sqrt{2}x_1x_2, x_2^2)$$

und berechnen das Skalarprodukt:

$$\begin{aligned}
\langle \Phi(\vec{x}), \Phi(\vec{y}) \rangle &= (x_1^2, \sqrt{2}x_1x_2, x_2^2)(y_1^2, \sqrt{2}y_1y_2, y_2^2) \\
&= x_1^2y_1^2 + 2x_1y_1x_2y_2 + x_2^2y_2^2 \\
&= (x_1y_1 + x_2y_2)^2 \\
&= \langle \vec{x}, \vec{y} \rangle^2 =: K(\vec{x}, \vec{y})
\end{aligned}$$

_ Man kann das Skalarprodukt ausrechnen, ohne die Funktion berechnen zu müssen. Es reicht aus, das Quadrat von *x* und *y* im $\mathbb{R}^2$ zu berechnen.

So ist auch das duale Optimierungsproblem lösbar:

$$maximize_{\alpha \in R^n} L(\vec{\alpha}) = \sum_{i=1}^{d} \alpha_i - \frac{1}{2} \sum_{i=1}^{d} \sum_{j=1}^{d} \alpha_i \alpha_j y_i y_j \langle \Phi(\vec{x_i}), \Phi(\vec{x_j}) \rangle$$

3 Kriterien müssen von einer Kernelfunktion K_Φ erfüllt werden:

1. Symmetrie: $K_\Phi(\vec{x}, \vec{y}) = K_\Phi(\vec{y}, \vec{x})$

2. Cauchy-Schwarz:

$$K_\Phi(\vec{x}, \vec{y})^2 \leq K_\Phi(\vec{x}, \vec{x}) \cdot K_\Phi(\vec{y}, \vec{y})$$

3. Matrix muss positiv (semi-definit) sein :

$$G(K) = \begin{pmatrix} K(\vec{x}_1, \vec{x}_1) & \dots & K(\vec{x}_1, \vec{x}_n) \\ \vdots & \ddots & \vdots \\ K(\vec{x}_n, \vec{x}_1) & \dots & K(\vec{x}_n, \vec{x}_n) \end{pmatrix}$$

Beispiele für häufig verwendete Kernelfunktionen:

linear: $K(\vec{x}, \vec{y}) := \langle \vec{x}, \vec{y} \rangle$

Radial-Basis-Kernel: $K(\vec{x}, \vec{y}) := exp(-\gamma \cdot |\vec{x} - \vec{y}|^2)$

polynomiell: $K(\vec{x}, \vec{y}) := (\langle \vec{x}, \vec{y} \rangle + 1)^d$

sigmoid: $K(\vec{x}, \vec{y}) := tanh(\gamma \cdot (\vec{x} - \vec{y}) + c)$

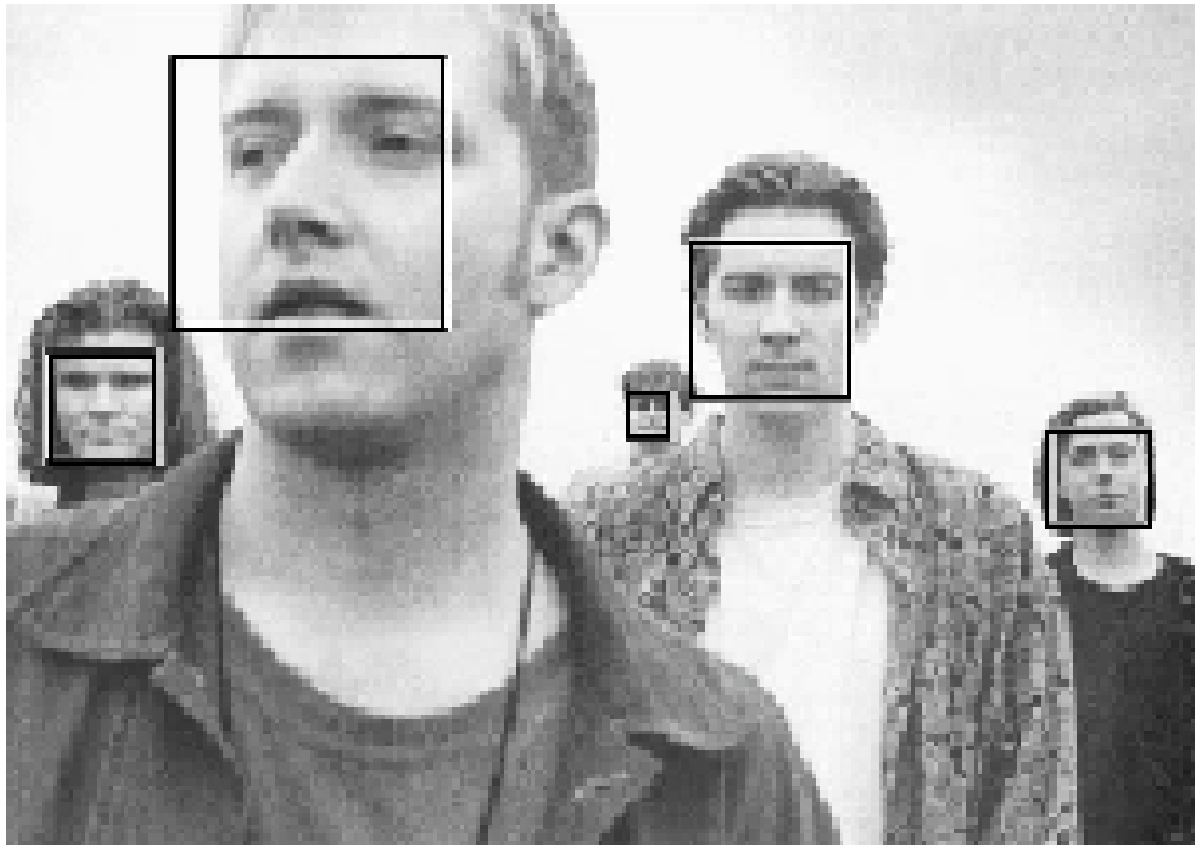
Eine **Support Vector Machine** ist

- eine *Hyperebene mit maximaler Trennspanne* im Feature Space.
- konstruiert im hochdimensionalen Raum durch eine *Kernelfunktion*.

Praktische Anwendungen:

Gesichtserkennung

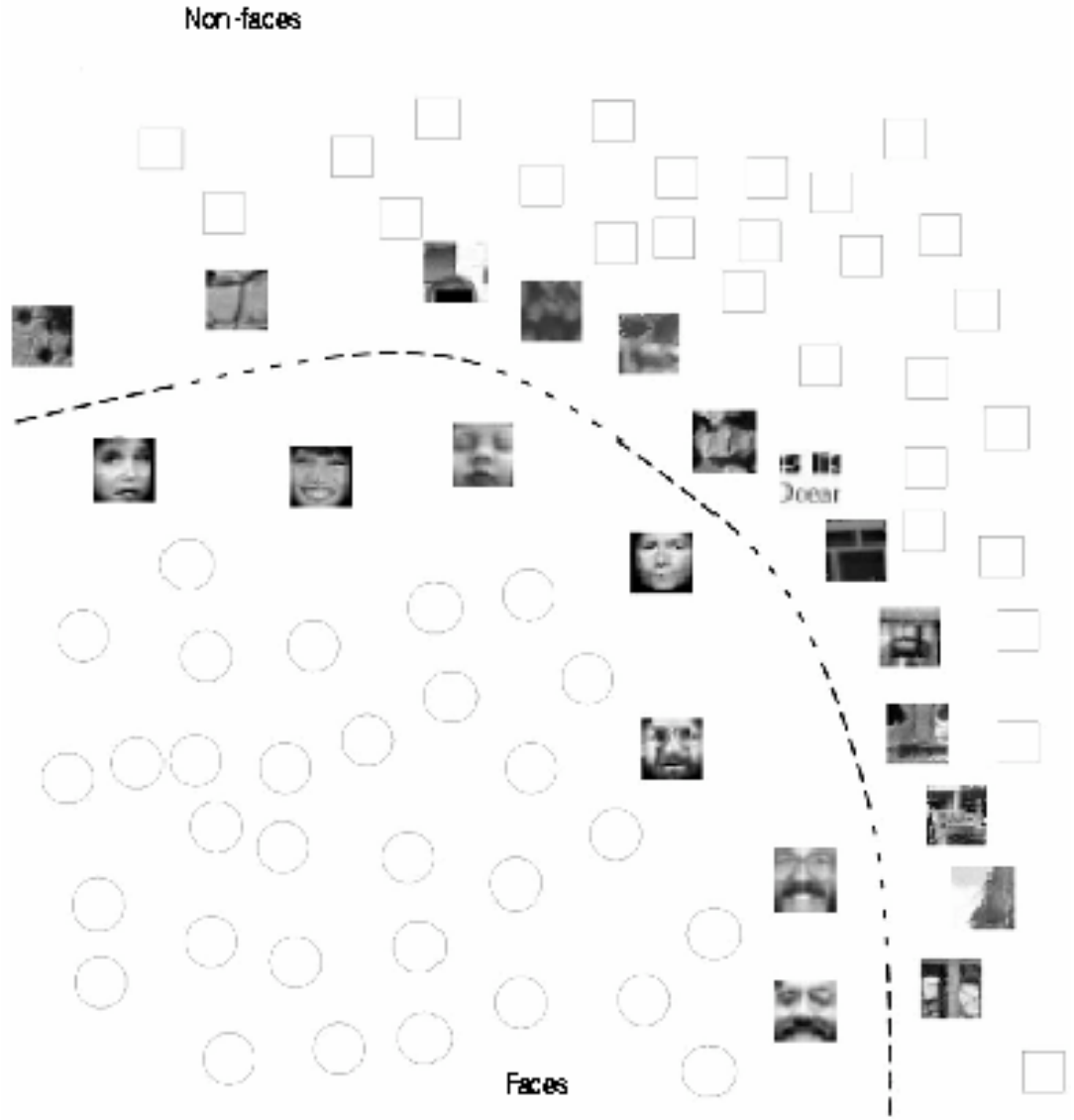
Ziel:



Vorverarbeitung der Daten (Bilder) notwendig

Gesichtserkennung

1. Gleichgroße Fenster (z.B 20x20 Pixel) aus dem Bild ausschneiden
2. Bild skalieren (Verkleinern/Vergrößern) und wieder Fenster ausschneiden
3. Verschiedene Fenster vorverarbeiten (Graustufen, Schärfe...)



Gesichtserkennung

4. SVM klassifiziert Gesicht/Nicht Gesicht

5. Falls Gesicht: Markieren mit Rahmen

6. „Bootstrapping“

Spamfilterung

Parameter z.B.:

- Anzahl bestimmter Wörter: FREE, ...
- Struktur der Absendeadresse
- Bestimmte Zeichen im Header

Klassen:

- Spam
- No Spam

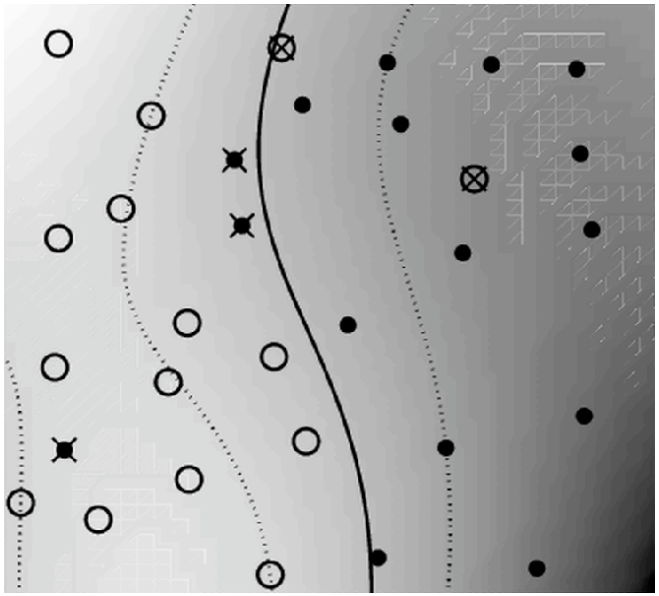
Praktische Anwendungen

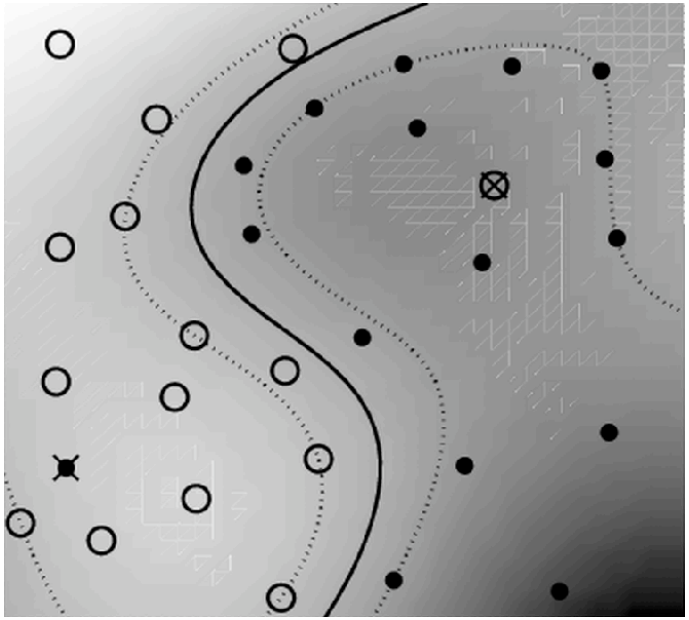
In der **Bioinformatik**

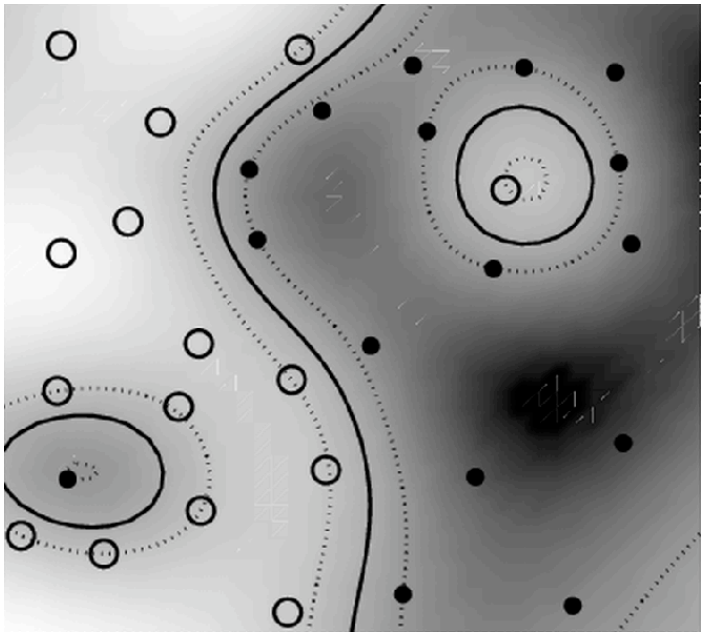
Man arbeitet oft mit einem kleinen Datensatz hochdimensionaler Daten, z.B.:

Klassifizierung eines Proteins:

_ Konvertierung in einen 20-dimensionalen Vektor, indem man seine Aminosäure-Sequenz darstellt





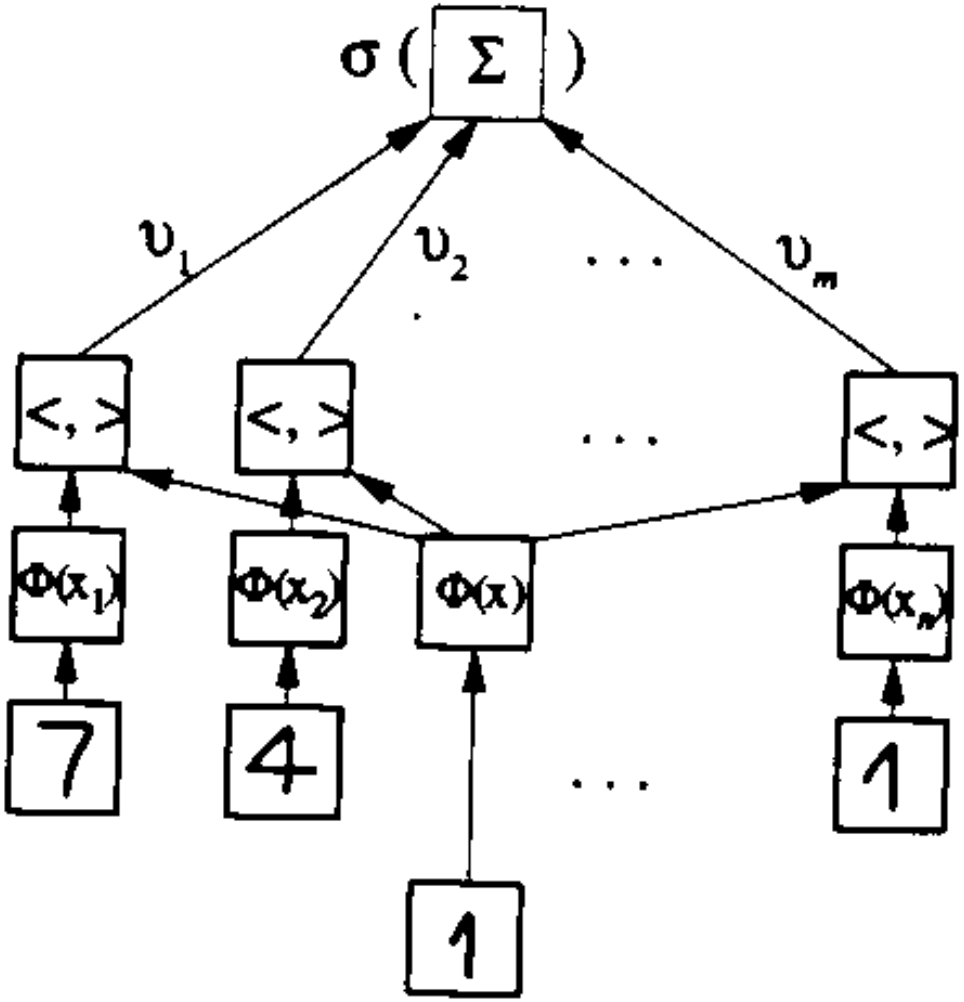


Praktische Anwendungen

Beispiel für Erkennung mehrerere Klassen:

Handschrifterkennung

Verwendeter Kernel:
‚Gaussian-Dynamic-
Time-Warping-Kernel‘



SVM: Vorteile

- Klassifizierung sehr schnell möglich
- Hohe Präzision und geringe Fehlerwahrscheinlichkeit
- Benutzung von relativ einfachen und gut verständlichen Funktionen und Rechentechniken
- Effektiv auf Daten mit vielen Merkmalen
- Gute Performance in ‚Real-World'-Anwendungen

SVM Nachteile

- Benötigt sehr viel Zeit fürs Lernen (Entscheidung für die richtige Hyperebene)
- Es muss empirisch nach der geeigneten Kernelfunktion gesucht werden.
- Neues Training erforderlich, falls neue Eingabedaten verschieden sind.
- Richtige Vorverarbeitung der Daten (z.B. Skalierung, welche Merkmale sind wichtig) ist essentiell

Vielen Dank für die Aufmerksamkeit.

Noch Fragen?