# Grundlagen von Support Vector Maschinen und Anwendungen in der Bildverarbeitung

**Jan Eichhorn**

jan.eichhorn@tuebingen.mpg.de

Max-Planck-Institut für biologische Kybernetik
72076 Tübingen

# Danksagung


- Olivier Bousquet und Bernhard Schölkopf (slides)
- Wolf Kienzle (Demo)

# Übersicht

- **Grundlagen von Support Vector Maschinen**
  - Motivation aus Statistischer Lerntheorie
  - Lineare Klassifizierer – warum großer Margin?
  - Formulierung der Support Vector Maschine
  - Der Kern-Trick
- **Anwendungen in der Bildverarbeitung**
  - Kern-Funktionen für lokale Deskriptoren
  - Separable Filter zur Beschleunigung von SVMs in der Gesichtserkennung

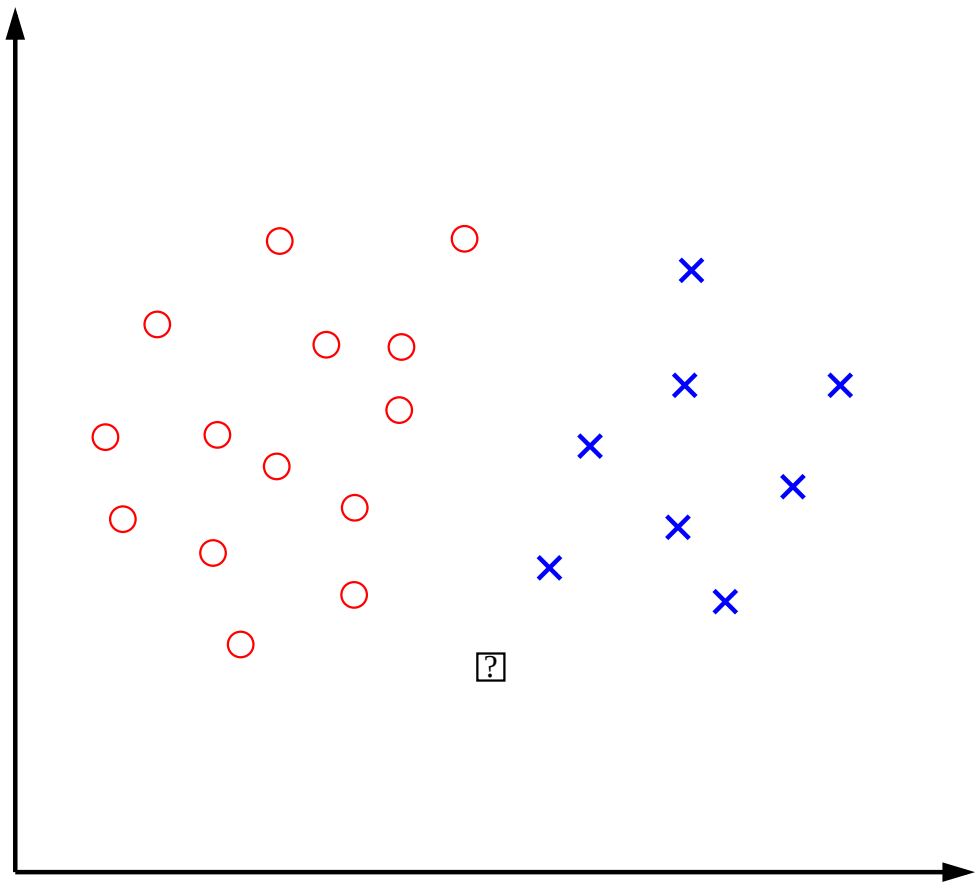
# Das Klassifikationsproblem

**Trainingsdaten:**

$$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1\ldots m} \subset \mathcal{X} \times \mathcal{Y}$$

$$\mathcal{X} = \mathbb{R}^2,$$

$$\mathcal{Y} = \{+1, -1\}$$




**Entscheidungsfunktion:**

$$f(\mathbf{x}) = \mathrm{sgn}(\ldots) = \pm 1$$

# Das Lernproblem

**Gegeben:** Daten $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1\ldots m} \subset \mathcal{X} \times \mathcal{Y}$

**Annahme:** Datenpunkte unabhängig und gleichverteilt nach unbekannter Verteilung: $(\mathbf{x}_i, y_i) \sim \mathrm{P}(\mathbf{x}, y)$

**Ziel:** Schätze **Entscheidungsfunktion** $f \in \mathcal{F}$ so, dass das **Risiko** $R[f]$ minimal ist.

$$R[f] = \int_{\mathcal{X}\times\mathcal{Y}} l(f(\mathbf{x}), y)\, \mathrm{dP}(\mathbf{x}, y)$$

$l(f(\mathbf{x}), y)$ ist die Verlustfunktion für die Vorhersage $f(\mathbf{x})$ wenn der wahre Wert $y$ ist.

# Empirische Risiko-Minimierung



- **Wahres Risiko** $R[f]$ ist nicht berechenbar, da $\mathrm{P}(x, y)$ unbekannt
- Man muss aus den Daten $\mathcal{D}$ lernen $\Rightarrow$ **Induktionsprinzip**
- **Empirisches Risiko** aus den Daten geschätzt:

$$R^m_{\mathrm{emp}}[f] = \frac{1}{m} \sum_{i=1}^{m} l(y_i, f(x_i))$$

$\Rightarrow$ **Empirische Risiko-Minimierung:** finde $f_m$ so dass

$$f_m = \arg\min_{f \in \mathcal{F}} \; R^m_{\mathrm{emp}}[f]$$

- Wie gut approximiert $R^m_{\mathrm{emp}}[f]$ das **Wahre Risiko** $R[f]$?

# No free lunch-Theorem

Betr. zwei Fälle:

1. $\mathcal{F}$ enthält nur **eine einzige Funktion**
   Gesetz der Großen Zahlen $\Rightarrow R^m_{\text{emp}}[f] \overset{m\to\infty}{\longrightarrow} R[f]$
2. $\mathcal{F} = \{\pm 1\}^{\mathcal{X}}$ enthält **alle Funktionen**:
   Betr. zwei Mengen mit $m$ Punkten:
   $\mathcal{D}_{train}, \mathcal{D}_{test} \subset \mathcal{X} \times \{\pm 1\}, \quad \mathcal{D}_{train} \cap \mathcal{D}_{test} = \emptyset$

   Zu jedem $f \in \mathcal{F}$ gibt es ein $f^* \in \mathcal{F}$ so dass:

   $$f^*(\mathbf{x}_i) = f(\mathbf{x}_i),\ \forall \mathbf{x}_i \in \mathcal{D}_{train} \quad \textbf{aber} \quad f^*(\mathbf{x}_j) \neq f(\mathbf{x}_j),\ \forall \mathbf{x}_j \in \mathcal{D}_{test}$$

   Auf $\mathcal{D}_{train}$ sind $f$ und $f^*$ ununterscheidbar, liefern jedoch *entgegengesetzte* Resultate auf $\mathcal{D}_{test}$.

# Einschränkung der Funktionenklasse

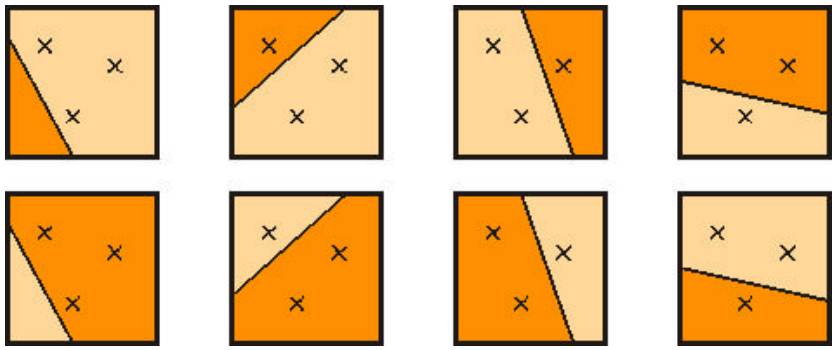
Die Funktionenklasse muss eingeschränkt werden, um erfolgreich lernen zu können.

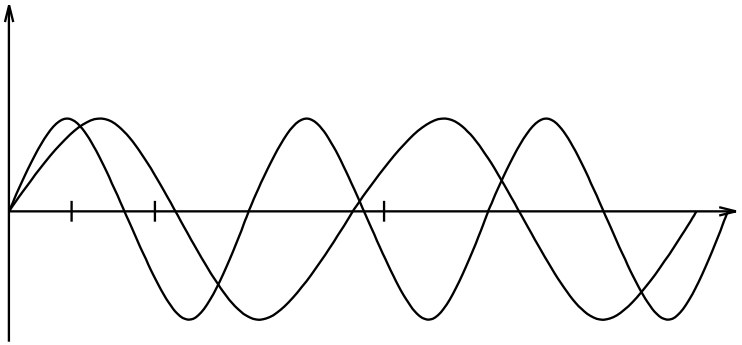
- **Statistische Lerntheorie** beschränkt die *Kapazität* der Funktionenklasse (z.B. VC-Dimension)
- **Bayesianische Lerntheorie** nimmt eine *a priori*-Wahrscheinlichkeitsverteilung der Funktionen im Funktionenraum an
- **Heuristisch:** Der Algorithmus benutzt implizit eine bestimmte Funktionenklasse (Nearest Neighbour) oder es wird eine Regularisierung vorgenommen (early stopping in N.N., Ridge-Regression).

# Die Vapnik-Chervonenkis-Dimension

- Die Menge $\mathcal{X} = \{x_i\}_{i=1...m}$ kann auf max. $2^m$ Arten in zwei Klassen aufgeteilt werden. Wenn Funktionen $f \in \mathcal{F}$ alle Aufteilungen realisieren können, sagt man $\mathcal{F}$ "zertrümmert" $\mathcal{X}$.
- $\mathcal{F}$ hat die VC-Dimension $h$, wenn $\mathcal{F}$ eine Menge von $h$ Punkten "zertrümmern" kann, es aber keine Konfiguration von $h+1$ Punkten gibt, die $\mathcal{F}$ "zertrümmern" kann.
- Gibt es kein solches $h$ so ist $VC(\mathcal{F}) = \infty$

test





Vapnik und Chervonenkis, 1968, 1971

# Gleichmäßige Konvergenz

Das **empirische Risiko** konvergiert gleichmäßig (über alle $f \in \mathcal{F}$) gegen das **wahre Risiko** genau dann wenn VC-Dimension endlich.

$$\lim_{m\to\infty} \mathrm{P}\left(\sup_{f\in\mathcal{F}} \left(|R^m_{\mathrm{emp}}[f] - R[f]|\right) > \epsilon\right) = 0 \quad \Leftrightarrow \quad VC(\mathcal{F}) < \infty$$

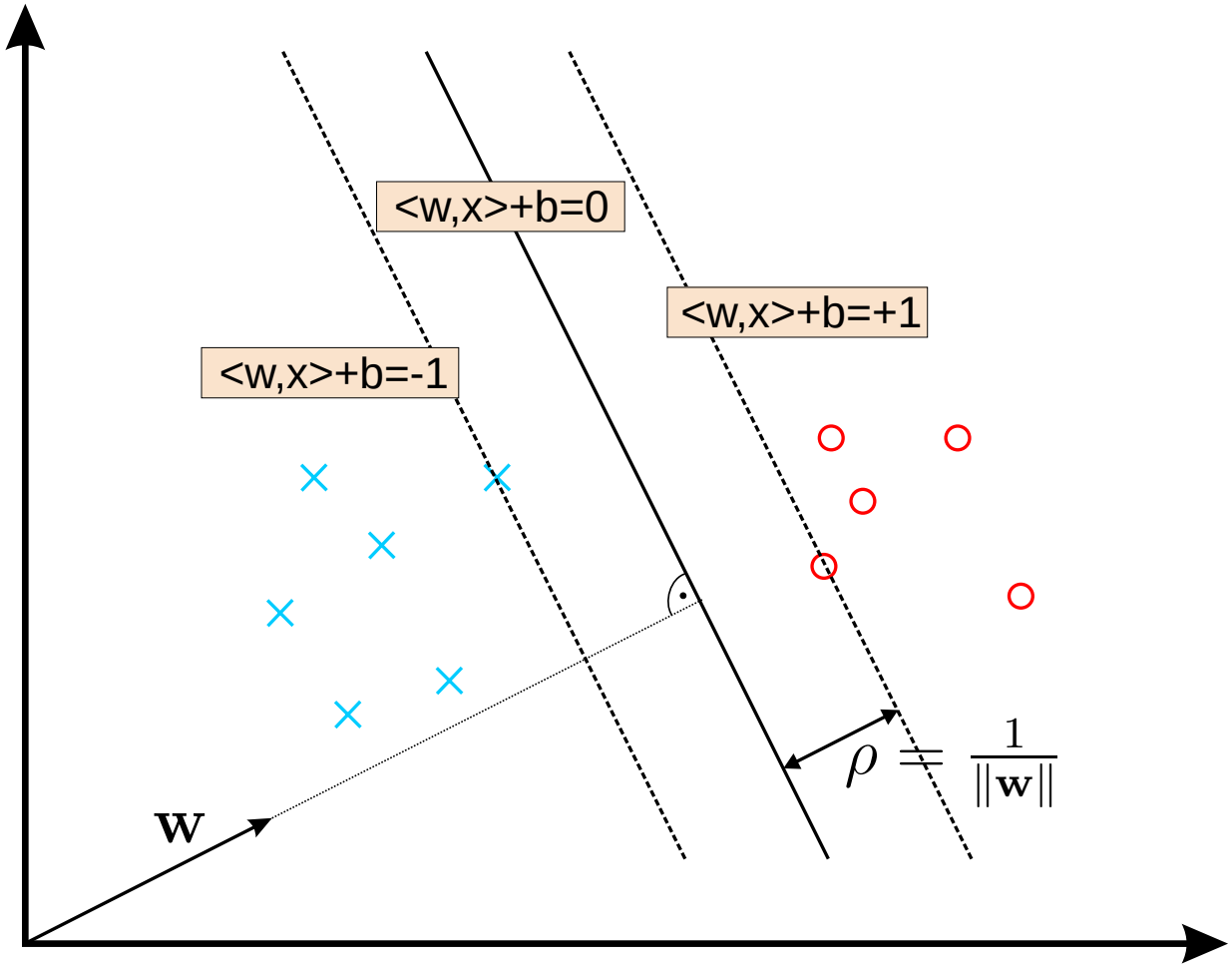
Wenn $h = VC(\mathcal{F}) < m$ dann gilt mit Wahrscheinlichkeit $1 - \delta$:

$$R[f] < R^m_{\mathrm{emp}}[f] + \phi\left(h, m, \delta\right)$$

$$\phi\left(h, m, \delta\right) = \sqrt{\frac{1}{m}\left(h\left(\ln\frac{2m}{h} + 1\right) + \ln\frac{4}{\delta}\right)} \quad \sim \sqrt{\frac{h}{m}}$$

$\Rightarrow$ **Beim Lernen die VC-Dimension kontrollieren!!**

# Hyperebenen und Margin




$\Rightarrow$ **Margin** $\rho$**:** Kleinster Abstand der Punkte zur Hyperebene

# Lineare Klassifizierer

**Entscheidungsfunktion:**

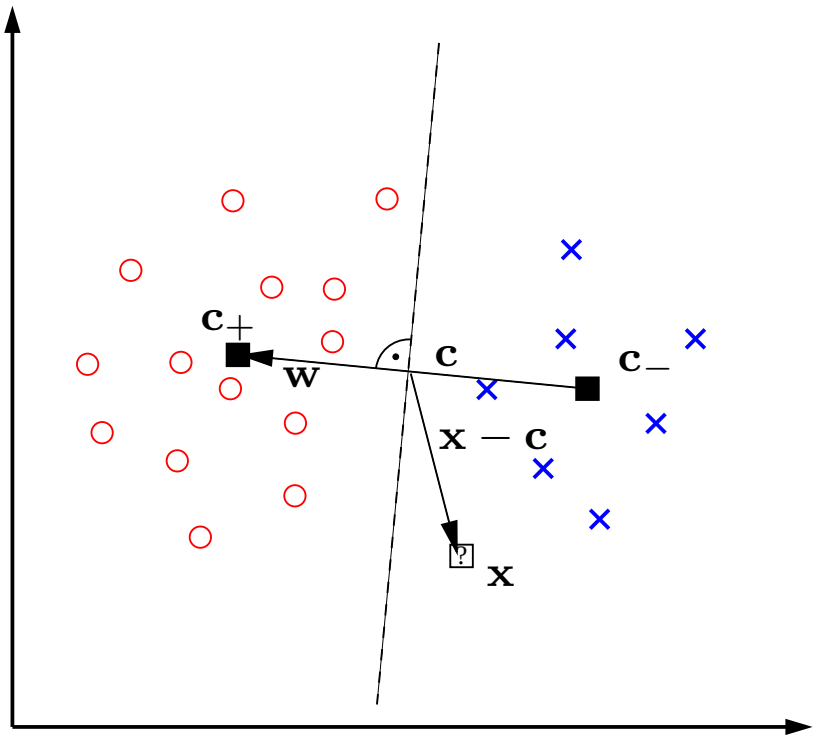
$$f(x) = \operatorname{sgn}\left(\langle \mathbf{w}, \mathbf{x} \rangle + b\right)$$

Beispiel: *Prototype-Klassifizierer*

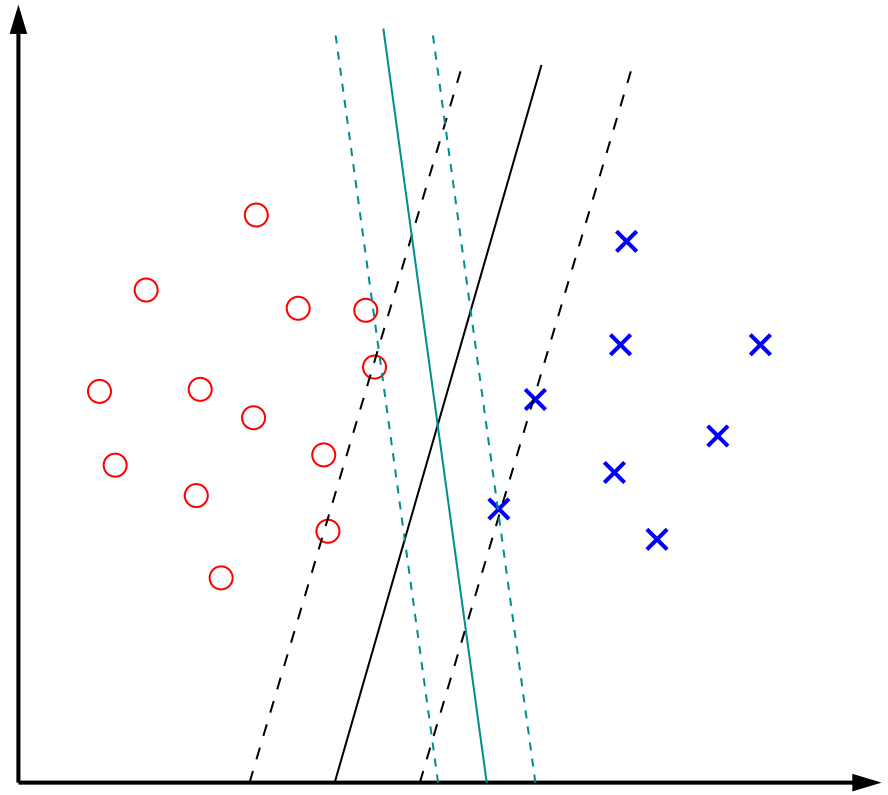
$$f(x) = \operatorname{sgn}\left(\langle \mathbf{c}_+ - \mathbf{c}_-, \mathbf{x} - \mathbf{c} \rangle\right)$$

also

$$\mathbf{w} = \mathbf{c}_+ - \mathbf{c}_-, \quad b = -\langle \mathbf{w}, \mathbf{c} \rangle$$



# Lineare Klassifizierer



Welche der möglichen kanonischen Ebenen hat die besten Verallgemeinerungs-Chancen?

Welche Ebene soll man wählen? $\Rightarrow$ die mit dem **größten Margin**

# Warum Large Margin Klassifizierer?



**Satz [Vapnik, 1979]:** Betrachte **Hyperebenen** $\langle \mathbf{w}, \mathbf{x} \rangle = 0$ wobei $\mathbf{w}$ so normiert ist, dass diese in kanonischer Form bzgl. einer Menge von Punkten $X^* = \{\mathbf{x}_1, \ldots, \mathbf{x}_r\}$ sind, d.h.,

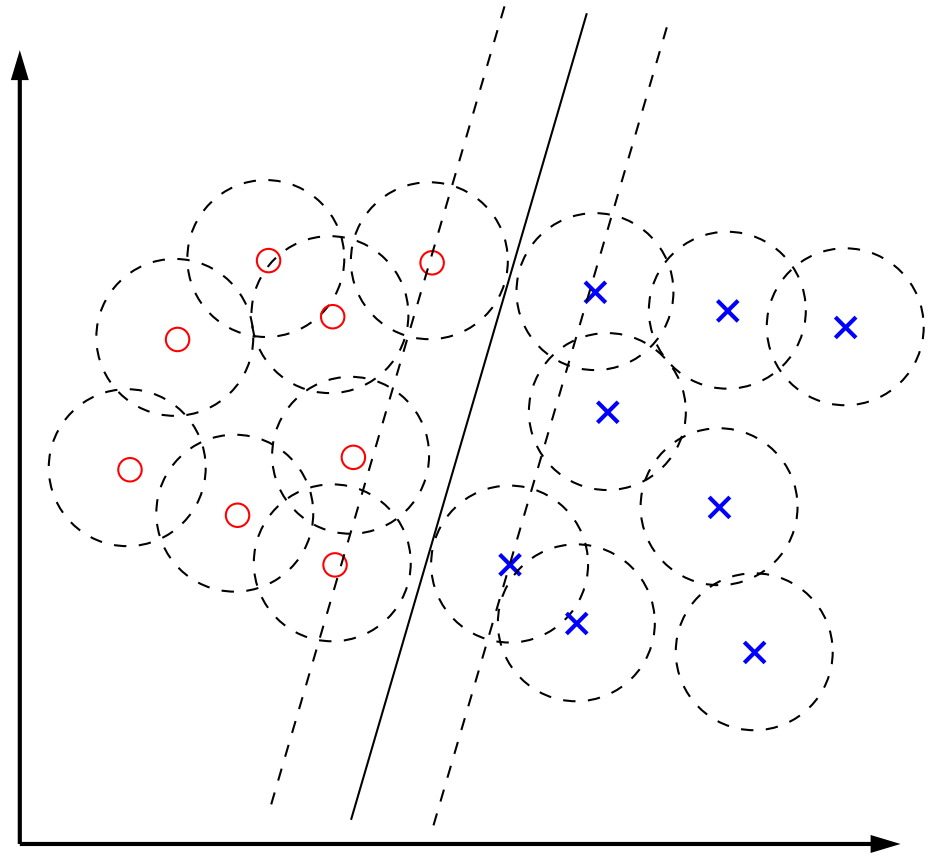
$$\min_{i=1,\ldots,r} |\langle \mathbf{w}, \mathbf{x}_i \rangle| = 1.$$

Die Menge der auf $X^*$ definierten Entscheidungsfunktionen $f_{\mathbf{w}}(\mathbf{x}) = \operatorname{sgn} \langle \mathbf{w}, \mathbf{x} \rangle$ hat eine **VC-Dimension**

$$h \leq \frac{R^2}{\rho^2}$$

Hierbei ist $R$ der Radius der kleinsten Kugel um den Ursprung, die $X^*$ enthält.

# Intuitive Erklärung



- **Rauschen in den Datenpunkten:** Je größer der Margin, desto mehr Rauschen in den Daten verträgt der Algorithmus, ohne dass die Klassifizierung falsch wird.

# Die Support Vector Maschine



Formulierung des Optimierungsproblems für größtmöglichen Margin (bedenke: $\rho \sim 1/\|\mathbf{w}\|$)

$$\min_{\mathbf{w},b} \|\mathbf{w}\|^2$$

unter den Nebenbedingungen

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1, \quad \forall\, i = 1 \ldots m$$

D.h., die Trainingsdaten werden korrekt separiert.

# Die Lagrange-Funktion

Einführung von Lagrange-Multiplikatoren $\alpha_i \geq 0$ liefert die Lagrange-Funktion:

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{m} \alpha_i [\, y_i(\langle \mathbf{w},\, \mathbf{x}_i \rangle + b) - 1]$$

$L(\mathbf{w}, b, \boldsymbol{\alpha})$ wird *minimiert* bzgl. der *primären Variablen* $\mathbf{w}$ und $b$ und *maximiert* bzgl. der *dualen Variablen* $\alpha_i$.

# Die duale Formulierung

Einsetzen der KKT-Bedingungen in $L(\mathbf{w}, b, \boldsymbol{\alpha})$ liefert das duale Problem:

$$\max_{\boldsymbol{\alpha}} \quad W(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \left\langle \mathbf{x}_i, \mathbf{x}_j \right\rangle$$

unter den Nebenbedingungen:

$$\alpha_i \geq 0, \quad \forall i = 1 \ldots m, \quad \text{und} \quad \sum_{i=1}^{m} \alpha_i y_i = 0.$$

$\Rightarrow$ **Konvexes quadratisches Optimierungsproblem**
(d.h. globales Optimum erreichbar)

# Support Vektoren

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$$

wobei $\forall\ i = 1 \ldots m$ entweder

$y_i \cdot [\langle \mathbf{w},\ \mathbf{x}_i \rangle + b] > 1 \qquad \Longrightarrow \alpha_i = 0 \longrightarrow \quad \mathbf{x}_i$ irrelevant

oder

$y_i \cdot [\langle \mathbf{w},\ \mathbf{x}_i \rangle + b] = 1 \quad (\textit{auf}$ dem Margin$) \longrightarrow \quad \mathbf{x}_i$ "Support Vector"

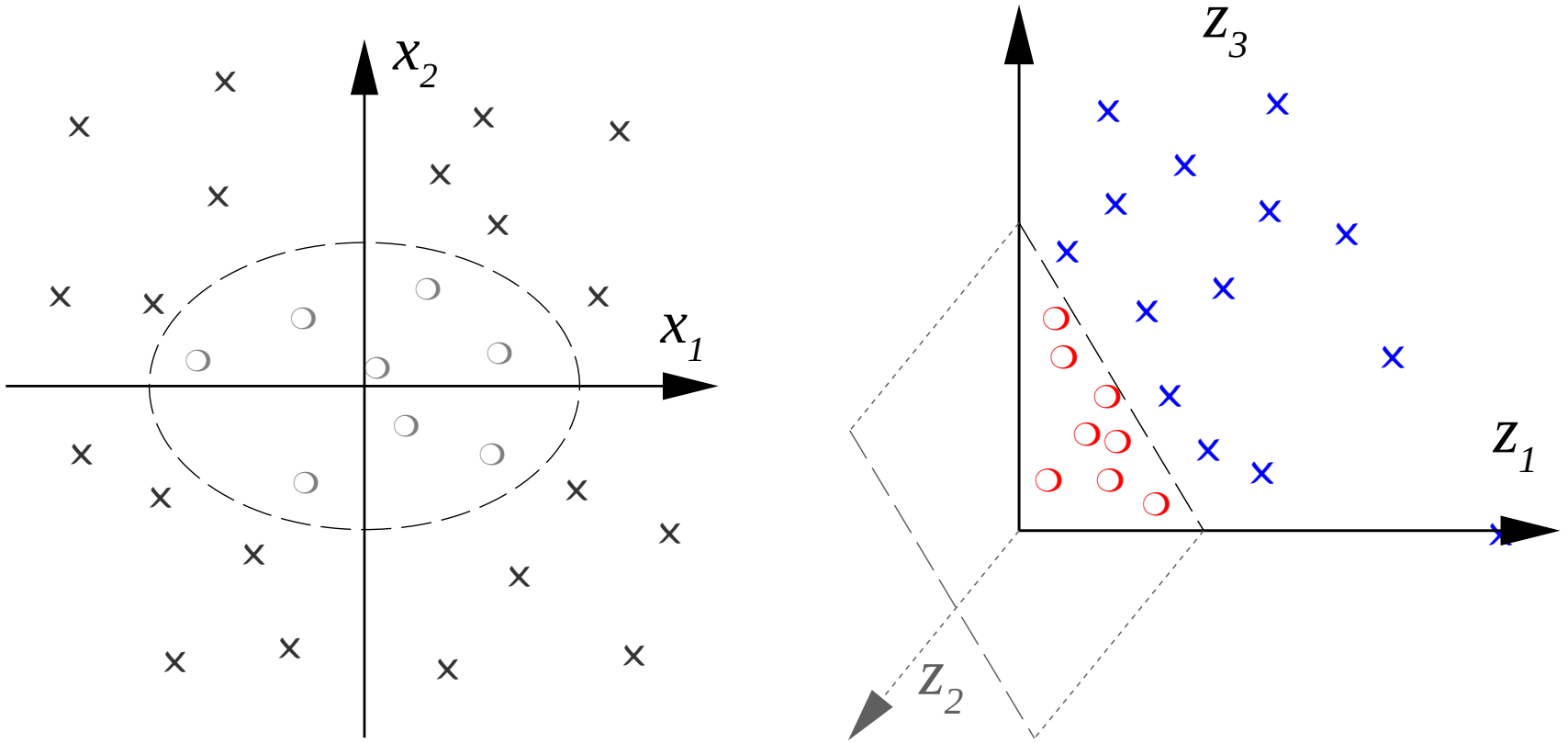
*Die Lösung wird bestimmt durch die Datenpunkte auf dem Margin.*

$$\begin{aligned} f(\mathbf{x}) &= \operatorname{sgn}\left(\langle \mathbf{x},\ \mathbf{w} \rangle + b\right) \\ &= \operatorname{sgn}\left(\sum_{i=1}^{m} \alpha_i y_i \langle \mathbf{x},\ \mathbf{x}_i \rangle + b\right). \end{aligned}$$

# Der Kern-Trick

- Erlaubt lineare Algorithmen auf nicht linear separierbare Probleme anzuwenden durch *effiziente* Berechnung einer (nicht-linearen) **Merkmalstransformation**.

# Merkmalstransformation




$$\Phi : \mathbb{R}^2 \to \mathbb{R}^3$$

$$(x_1, x_2) \mapsto (z_1, z_2, z_3) := (x_1^2, \sqrt{2}\, x_1 x_2, x_2^2)$$

# Merkmalsraum

Für nicht linear separierbare Probleme können in einem Vorverarbeitungsschritt geeignete Merkmale konstruiert werden (z.B. Pixel-Korrelationen in Bildern).
Vorverarbeitung der Daten mit:

$$\begin{aligned} \Phi: \quad & \mathcal{X} \to \mathcal{H} \\ & \mathbf{x} \mapsto \Phi(\mathbf{x}) \end{aligned}$$

Der lineare Klassifizierer kann jetzt auf den neuen Merkmalen arbeiten.

# Der Kern-Trick in der SVM

Die Zielfunktion

$$\max_{\boldsymbol{\alpha}} \quad W(\boldsymbol{\alpha}) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \langle \Phi(\mathbf{x}_i),\, \Phi(\mathbf{x}_j) \rangle$$

und die Entscheidungsfunktion

$$f(\mathbf{x}) = \operatorname{sgn}\left( \sum\nolimits_{i=1}^{m} \alpha_i y_i \langle \Phi(\mathbf{x}),\, \Phi(\mathbf{x}_i) \rangle + b \right).$$

verwenden nur Skalarprodukte der Daten.

Das Skalarprodukt im Merkmalsraum kann durch eine **Kernfunktion**

$$k(\mathbf{x}, \mathbf{x}') = \langle \Phi(\mathbf{x}),\, \Phi(\mathbf{x}') \rangle$$

ausgedrückt werden.

# Welche Kerne sind erlaubt?

Für $\mathcal{X} \neq \emptyset$ sind folgende Aussagen äquivalent:

- $k$ ist *positiv definit (pd)*, d.h.
  - für eine beliebige Menge von Punkten $\mathbf{x}_1, \ldots, \mathbf{x}_m \in \mathcal{X}$ und
  - für alle $a_1, \ldots, a_m \in \mathbb{R}$

  gilt:
  $$\mathbf{a}^\top K \mathbf{a} = \sum_{i,j} a_i a_j K_{ij} \geq 0, \quad \text{wobei } K_{ij} := k(x_i, x_j)$$

- Es gibt eine Funktion $\Phi : \mathcal{X} \to \mathcal{H}$ in einen Hilbertraum $\mathcal{H}$ so dass
  $$k(x, x') = \langle \Phi(x), \Phi(x') \rangle$$

$\mathcal{H}$ ist ein sogenannter *reproducing kernel Hilbert space* (RKHS).

# Kern-Trick — Zusammenfassung

- *Jeder* Algorithmus, der nur Skalarprodukte verwendet, kann den Kern-Trick benutzen
  $\Rightarrow$ Große Vielfalt kernelisierter Standardalgorithmen (z.B. Kernel-Perceptron, Kernel-Fisher-Diskriminante, Kernel-PCA, Kernel-CCA).
- Man kann den Kern als ein nichtlineares *Ähnlichkeitsmaß* interpretieren.
- Daher braucht $\mathcal{X}$ kein Vektorraum zu sein ($\exists$ Kerne auf Graphen, Text etc.). Die Geometrie wird durch die Kernfunktion induziert.

# Beispiel: Gausscher Kern

$$k(x, x') = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right)$$

- $\sigma$ kontrolliert die Lokalität der Daten

demo

# Weiterführende Ansätze

- C-SVM und $\mu$-SVM für nicht separierbare Probleme
- Support Vektor Regression
- One-Class-SVM zur Identifikation von Ausreißern
- Reduced-set-Methoden zur Beschleunigung (s. Anwendung)
- Andere Large-Margin-Klassifizierer (z.B. Boosting) und Kernel-Maschinen

# Anwendungen in der Bildverarbeitung



- SVM für Objekt-Kategorisierung, Repräsentation der Bilder durch **lokale Deskriptoren** $\Rightarrow$ Kernfunktion für Mengen
- SVM in der Gesichtserkennung, **Kaskade** von Klassifizierern, Beschleunigung des ersten Layers durch **separable Filter**

# Objekt-Kategorisierung mit SVM



# Lokale Deskriptoren für Bilder



**Wie?**

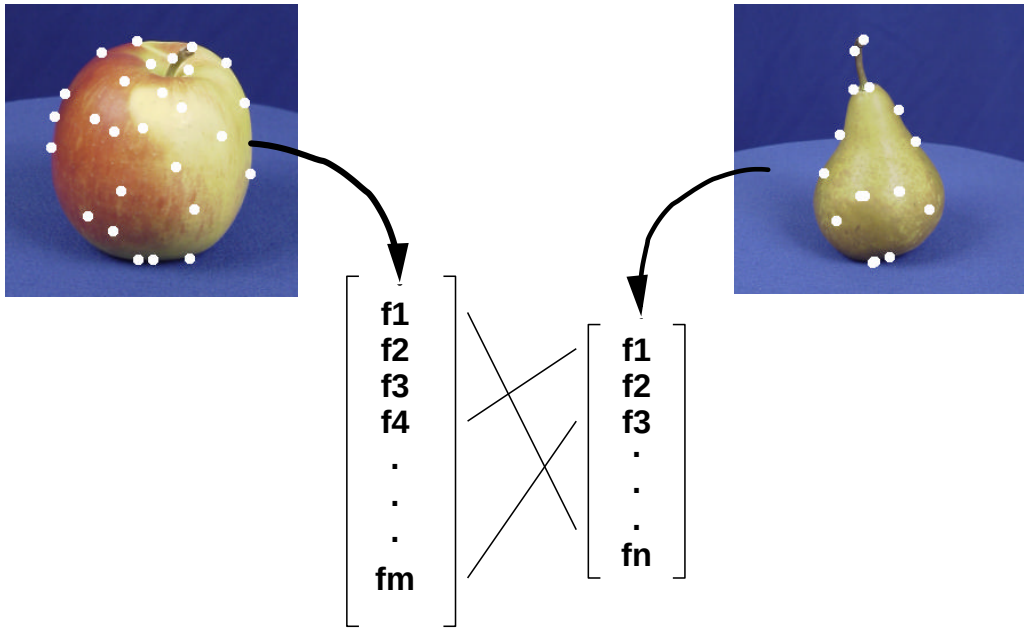
1. Interest Point Detector findet **markante Punkte**
2. In **lokaler** Umgebung wird eine **Bildcharakteristik** berechnet (z.B. Ausschnitt, Gradientenhistogramm etc.)

**Warum?**

- Je nach Konstruktion - **Invariant** unter Translation, Rotation, Skalierung, Änderung von Kontrast, Helligkeit etc.
- Sehr erfolgreich in Objektidentifizierung (**D. Lowe**)

# Geeignete Kernfunktion

SVM braucht eine **Kernfunktion** auf Mengen von lokalen Deskriptoren




Problem:

- Keine natürliche Reihenfolge
- Bilder haben i.A. verschiedene Anzahl von markanten Punkten

# Bisherige Ansätze

- Wallraven et al.[a]: Berechne Mittelwert des Abstandes zum jeweils ähnlichsten Deskriptor
  ⇒ Vergleichen einzelne Deskriptoren

- Kondor und Jebara[b]: Berechnen die Ähnlichkeit zweier Verteilungen
  ⇒ Vergleichen suffiziente Statistiken

[a] C. Wallraven et al, *Recognition with Local Features: the Kernel Recipe*, Proc. of ICCV'03

[b] R. Kondor and T. Jebara, *A Kernel between Sets of Vectors*, Proceedings of the ICML, 2003

# "A Kernel between Sets of Vectors"



Bhattacharyya-Kern für Wahrscheinlichkeitsverteilungen:

$$K(p, p') = \int \sqrt{p(x)}\sqrt{p'(x)}\,\mathrm{d}x$$

- **Geschlossener Ausdruck** für $p \sim \mathcal{N}(\mu, \Sigma)$
- Mehr Variabilität durch **Merkmalstranformation**:

$$\mathbf{x} \sim \mathcal{N}(\mu, \Sigma) \quad \xrightarrow{\Phi} \quad \Phi(\mathbf{x}) \sim \mathcal{N}(\tilde{\mu}, \tilde{\Sigma})$$

$\Rightarrow$ effizient berechenbar durch **Kern-Trick**

# References

[1] B. E. Boser, I. M. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152, Pittsburgh, PA, July 1992. ACM Press.

[2] V. N. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

[3] V. N. Vapnik. *The Nature of Statistical Learning Theory*. Statistics for Engineering and Information Science. Springer Verlag, New York, 2nd edition, 2000.

[4] B. Schölkopf and A. J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

[5] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.

[6] A. J. Smola, P. L. Bartlett, B. Schölkopf, and D. Schuurmans. *Advances in Large Margin Classifiers*. MIT Press, Cambridge, MA, 2000.

[7] E. T. Jaynes. *Probability Theory - The Logic of Science*. Cambridge University Press, Cambridge, U.K., 2003.

[8] D. J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, Cambridge, UK, 2003.