

Grundlagen der Künstlichen Intelligenz

16 Support-Vektor-Maschinen

Perceptron revisited, Lernen von linearen Max-Margin
Klassifikatoren, Nichtlinear separierbare Daten,
Datenraum & Merkmalsraum,
Kernel-Funktionen, Kernel-Maschinen

Volker Steinhage

Linear separierbare Daten

- Perceptron und duale Darstellung
- Der Margin und seine Maximierung

Nichtlinear separierbare Daten

- Transformation in Merkmalsräume
- Der Kernel-Trick
- Mercer-Kernels
- Kernel-Machines

Lernen von Klassifikatoren

- Aufgabe: aus m Trainingsbeispielen $(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)$ soll eine *Klassifikationsregel* gelernt werden.
-
- Einfachster Fall der *binären Klassifikation*: jedes Beispiel besteht aus
 - n -elementigem Datenvektor $\mathbf{x}_i = (x_{i1}, \dots, x_{in})$ und
 - seiner *binären Klassifikation* $y_i \in \{+1, -1\}$.
-
- Bspl.: Klassifikation aller deutschen Internetseiten in die beiden Klassen „mit Informatikbezug“ und „ohne Informatikbezug“:
 - Datenvektoren \mathbf{x}_i mit binären Elementen x_{ij} für das Auftreten bzw. Nichtauftreten relevanter Schlagworte.
 - Ziel: Klassifikation neuer Internetseiten mit geringer Fehlerquote.

Lineare Klassifikation durch Perzeptrons (1)

Perzeptron bislang:

- Lernen einer Trennungsebene $\mathbf{W} \cdot \mathbf{I} = 0$ durch die Perzeptron-Lernregel:

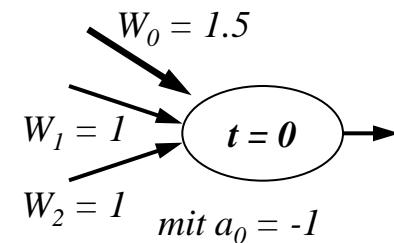
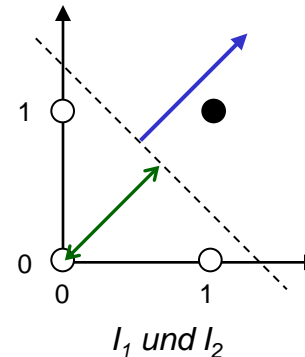
$$w_j \leftarrow w_j + \alpha \cdot I_j \cdot (T - O) \quad \text{für Eingabe } \mathbf{I}, \text{ wahre Ausgabe } \mathbf{T} \text{ und errechnete Ausgabe } \mathbf{O}$$

- Beispiel für die Boolesche Funktion **and** (I_1, I_2) mit $g = \text{step}_0$, $a_0 = -1$ und $W_0 = 1.5$

$$\mathbf{W} \cdot \mathbf{I} = (1.5, 1, 1)^T \cdot (-1, I_1, I_2)^T = 0 \quad \text{mit } t = 0$$

bzw. mit \mathbf{x} für \mathbf{I} :

$$\mathbf{W} \cdot \mathbf{x} - b = (1, 1)^T \cdot (x_1, x_2)^T - b = 0 \quad \text{mit } b = 1.5$$



Zur Erinnerung: $\text{step}_t(x) = \begin{cases} 1 & \text{if } x \geq t \\ 0 & \text{if } x < t \end{cases}$

Lineare Klassifikation durch Perzeptrons (2)

Perzeptron bislang:

- Lernen einer Trennungsebene $\mathbf{W} \cdot \mathbf{I} = 0$ durch die Perzeptron-Lernregel:

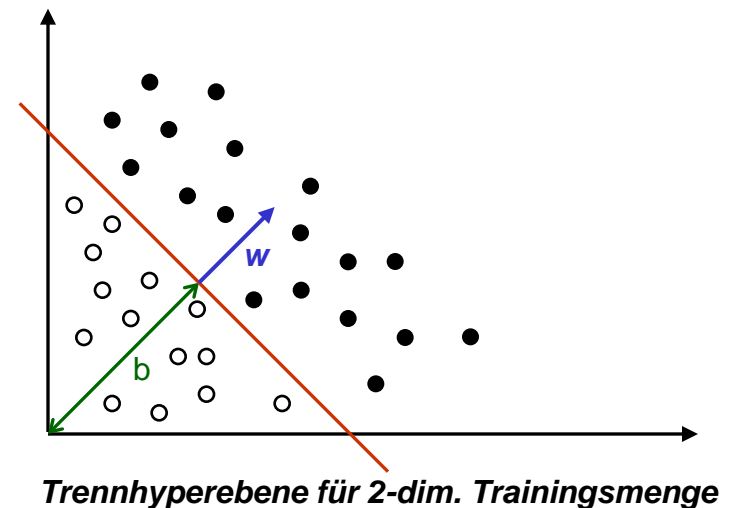
$$w_j \leftarrow w_j + \alpha \cdot I_j \cdot (T - O) \quad - \text{für Eingabe } \mathbf{I}, \text{ wahre Ausgabe } \mathbf{T} \text{ und errechnete Ausgabe } \mathbf{O}$$

- Jetzt allgemein für die binäre Klassifikation $y_i \in \{+1, -1\}$ mit $g = \text{sign}$:

- Trennende Hyperebene $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} - b = 0$ mit $\|\mathbf{w}\| = 1$
- Ausgabefunktion $h(\mathbf{x}) = \text{sign}(f(\mathbf{x}))$

Zur Erinnerung:

$$\text{sign}(x) = \begin{cases} +1 & \text{if } x \geq t \\ -1 & \text{if } x < t \end{cases}$$



Lineare Klassifikation durch Perzeptrons (3)

Perzeptron

- Perzeptron-Lernregel neu formuliert:

Bei negat. Produkt
unterschiedliche
Vorzeichen von y_i
und $\langle \mathbf{w}_t, \mathbf{x}_i \rangle \leadsto$ **Fehler**

Bisher:

$$w_j \leftarrow w_j + \alpha \cdot I_j \cdot (T - O)$$

if $y_i \cdot \langle \mathbf{w}_t, \mathbf{x}_i \rangle < 0$ **then**

$$\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \eta \cdot \mathbf{x}_i \cdot y_i$$

$$t \leftarrow t+1$$

Hier: **Schwellwert** b über zusätzliche Eingabekante mit $w_0 = b$ und $a_0 = -1$ kodiert

mit **Gewichtsvektor** \mathbf{w}_t in **Epoche** t , Trainingsbeispielen (\mathbf{x}_i, y_i) und **Lernrate** η .

- D.h. \mathbf{w}_t ergibt sich als eine Linearkombination der Trainingsbeispiele (\mathbf{x}_i, y_i) mit Koeffizienten $\alpha_i \geq 0$:

$$\mathbf{w}_t = \sum_i \alpha_i \cdot y_i \cdot \mathbf{x}_i$$

die α_i entspr.
 η oder 0

- \leadsto es werden nur informative Punkte (**Fehlklassifikationen**) benutzt
- \leadsto die Koeffizienten der Punkte reflektieren deren Bedeutung.

Perzeptrons: Duale Repräsentation

Perzeptron

- Neue duale Formulierung für trennende Hyperebene

\mathbf{w} als Linearkombination
der Trainingsbeispiele

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} = \langle \mathbf{w}, \mathbf{x} \rangle = \sum_i \alpha_i \cdot y_i \cdot \langle \mathbf{x}_i, \mathbf{x} \rangle = 0$$

mit der Darstellung des Vektors \mathbf{w} als Linearkombination aus Trainingsbeispielen: $\mathbf{w} = \sum_i \alpha_i \cdot y_i \cdot \mathbf{x}_i$.

- Damit ist auch die Lernregel neu formulierbar:

$$\text{if } y_i \cdot \sum_j \alpha_j \cdot y_j \cdot \langle \mathbf{x}_j, \mathbf{x}_i \rangle < 0 \text{ then } \alpha_i \leftarrow \alpha_i + \eta$$

Anmerkung: hier treten die Daten nur in Skalarprodukten auf!

- Wozu das Ganze?

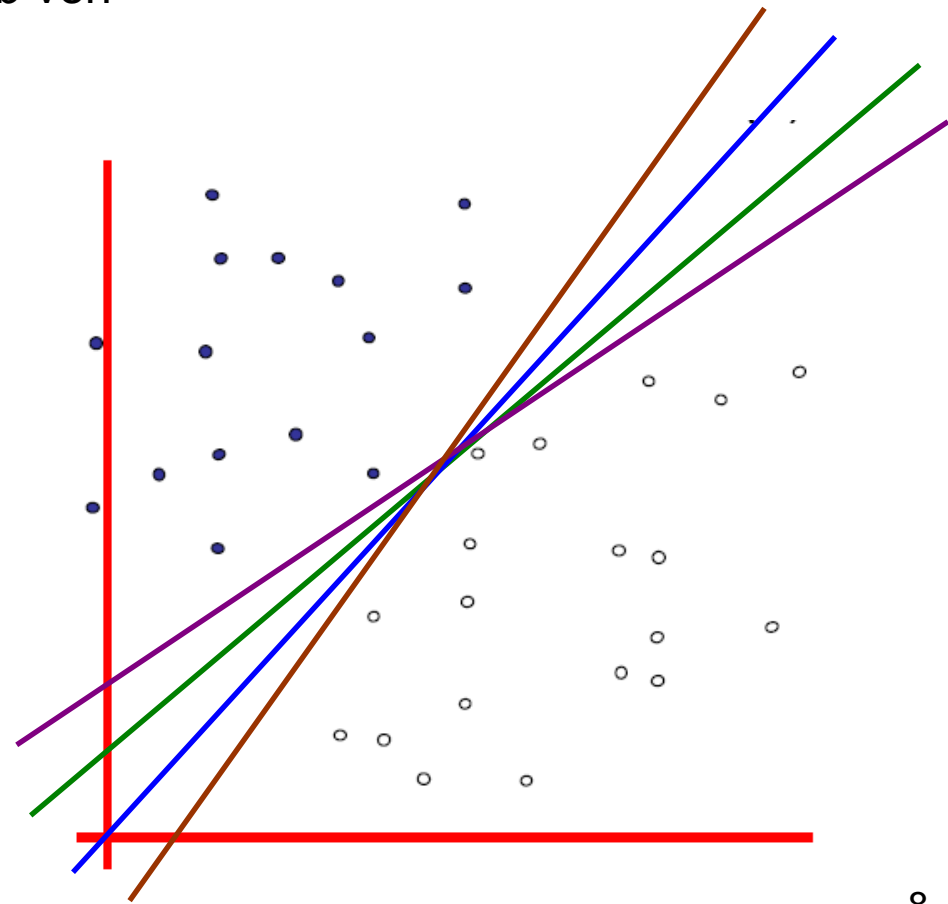
→ Support-Vektor-Maschinen sind lineare Klassifikatoren, die die duale Darstellung verwenden, um die optimale Trennungsebene zu wählen.

Auswahl der Trennungsebene

Es gibt i. A. nicht nur eine Trennungsebene

- Die Perceptron-Lernregel findet **irgendeine** Trennungsebene.
- Die Wahl der Trennungsebene hängt ab von
 - Lernrate und
 - Reihenfolge der verarbeiteten Trainingsdaten.

→ *Gibt es eine beste Trennebene?*



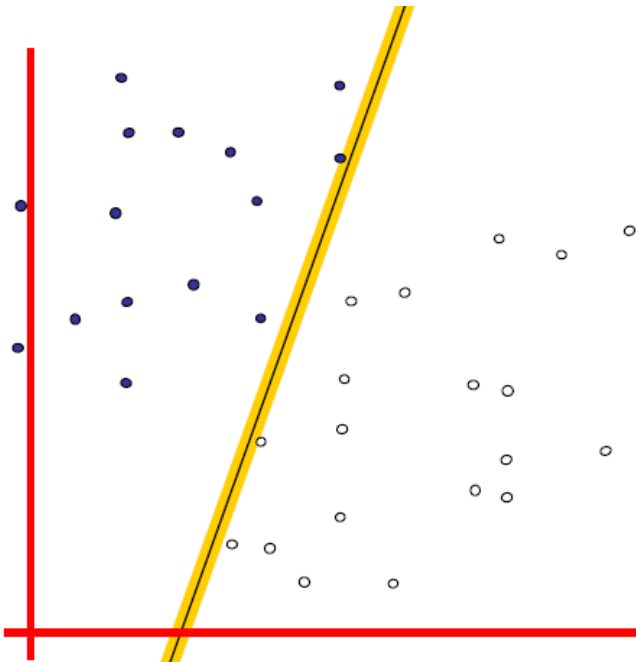
Der Margin

Die *Support Vector Machine (SVM)* wählt die Trennebene so,

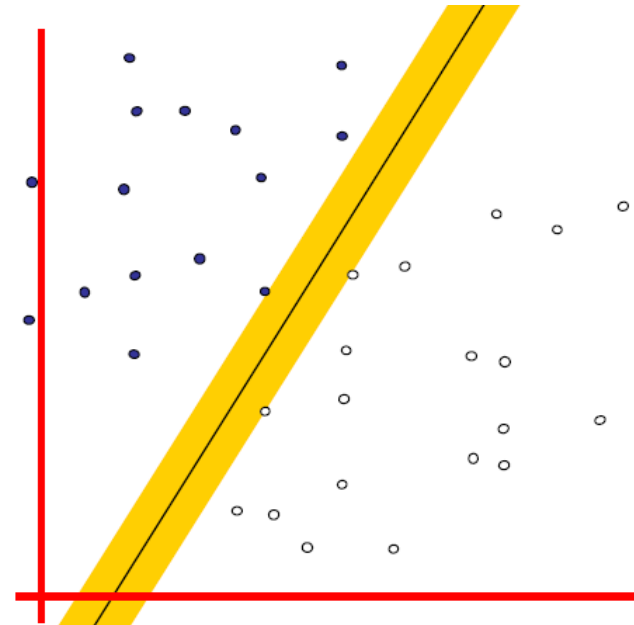
- dass der kleinste Abstand der Trainingsbeispiele zur Trennebene, die sog. *Trennschranke* (engl. *Margin*), maximiert wird.

→ man spricht von *Maximum-Margin-Klassifikation*.

Kleiner Margin

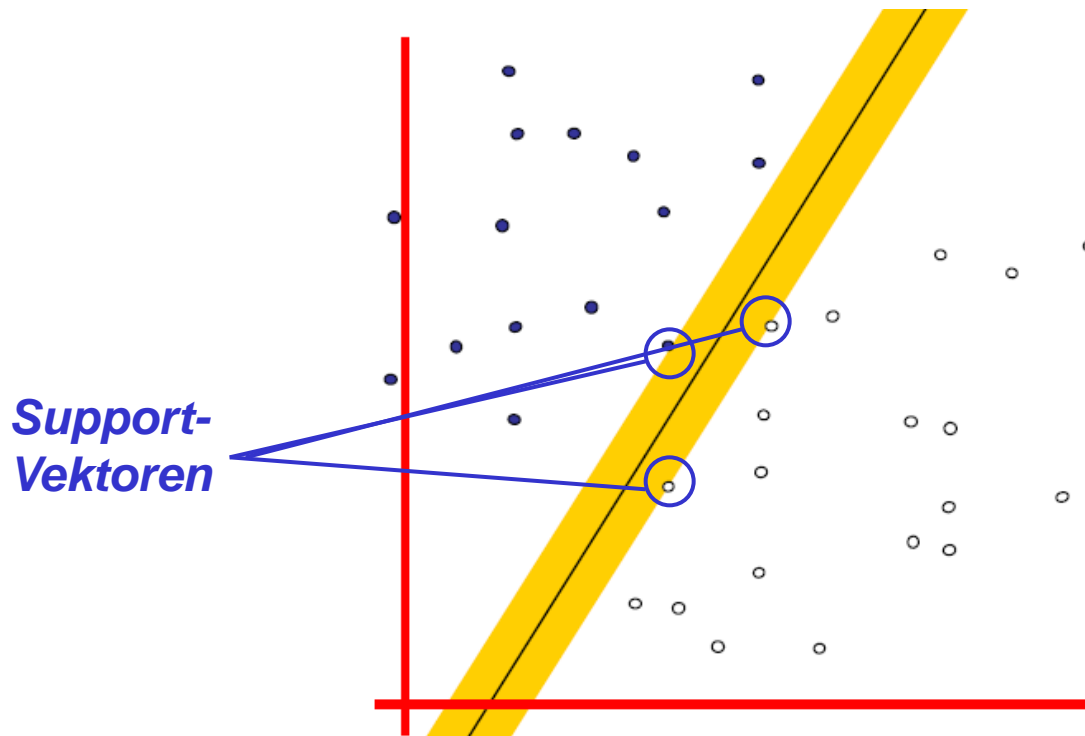


Großer Margin



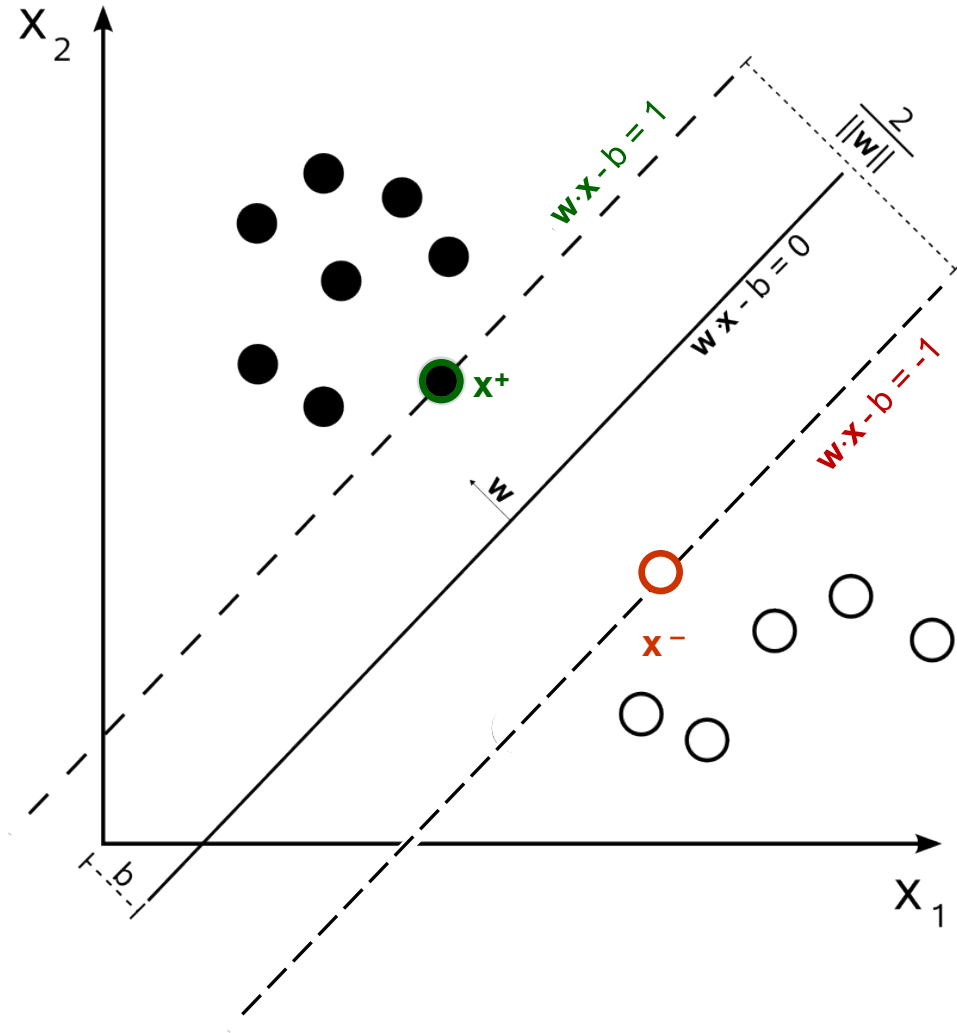
Support-Vektoren

- Die Trainingspunkte, welche den Margin berühren, heißen **Stützvektoren** (*Support Vectors*), weil sie die Trennungszone „stützen“.
- Der Klassifizierer heißt entsprechend: *lineare Support-Vektor-Maschine (SVM)*.
- Das SVM-Lernverfahren maximiert die Breite des Margins



Maximum Margin = Minimal Norm

- Wir **normieren** $f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} - b$ derart, dass die **Funktionswerte auf den Margin-Grenzen** gerade **+1** bzw. **-1** betragen.
- Die Breite M des Margin ist dann eine Funktion von \mathbf{w} :
 $\langle \mathbf{w}, \mathbf{x}^+ \rangle - b = +1$ und $\langle \mathbf{w}, \mathbf{x}^- \rangle - b = -1$
 $\leadsto \langle \mathbf{w}, (\mathbf{x}^+ - \mathbf{x}^-) \rangle = 2$
 $\leadsto M = \langle (\mathbf{w}/\|\mathbf{w}\|), (\mathbf{x}^+ - \mathbf{x}^-) \rangle = 2/\|\mathbf{w}\|$
- Maximieren von $M = 2/\|\mathbf{w}\|$ heißt $\|\mathbf{w}\|/2$ minimieren!
- Unter der Nebenbedingung, dass alle Trainingsbeispiele korrekt klassifiziert sein müssen!



Das Optimierungsproblem (1)

Analog zu:

$$\arg \min_{(\mathbf{w}, b)} \frac{1}{2} \cdot \|\mathbf{w}\|^2 = \frac{1}{2} \cdot \langle \mathbf{w}, \mathbf{w} \rangle$$

unter den Nebenbedingungen

$$y_i (\langle \mathbf{w}, \mathbf{x}_i \rangle - b) \geq 1.$$

Korrekte Klassifikation
der Trainingsbeispiele
mit minimalem Abstand
1 nach Normierung

Lösung:

- Einführen der Lagrange-Multiplikatoren $\alpha_i \geq 0$ und Zusammenfassung des Optimierungsproblems in der sog. Lagrange-Funktion

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2} \cdot \langle \mathbf{w}, \mathbf{w} \rangle - \sum_i \alpha_i [y_i (\langle \mathbf{w}, \mathbf{x}_i \rangle - b) - 1].$$

- Also wird $L(\mathbf{w}, b, \alpha)$ minimiert für \mathbf{w} und b und maximiert für die $\alpha_i \geq 0$:

$$\arg \min_{(\mathbf{w}, b)} \max_{\alpha \geq 0} \left\{ \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^n \alpha_i [y_i (\mathbf{w} \cdot \mathbf{x}_i - b) - 1] \right\}.$$

Das Verfahren der Lagrange-Multiplikatoren (nach Joseph-Louis Lagrange) formuliert Optimierungsprobleme mit Nebenbedingungen (*constrained optimization problems*) derart um, dass für jede Nebenbedingung eine neue unbekannte skalare Variable, ein sog. Lagrange-Multiplikator eingeführt wird, und definiert dann eine Linearkombination, welche die Multiplikatoren als Koeffizienten einbindet.

Das Optimierungsproblem (2)

- Wir minimieren

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2} \cdot \langle \mathbf{w}, \mathbf{w} \rangle - \sum_i \alpha_i [y_i (\langle \mathbf{w}, \mathbf{x}_i \rangle - b) - 1].$$

- Wir setzen dazu

$$\frac{\partial}{\partial b} L(\mathbf{w}, b, \alpha) = 0 \quad \text{und} \quad \frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}, b, \alpha) = \mathbf{w} - \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i = 0.$$

und erhalten

$$\sum_{i=1}^n \alpha_i y_i = 0 \quad \text{und} \quad \mathbf{w} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i.$$

- Diese Ergebnisse setzen wir wieder in $L(\mathbf{w}, b, \alpha)$ ein.

↪ Folgefolie



Das Optimierungsproblem (3)

- Wir setzen in

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2} \cdot \langle \mathbf{w}, \mathbf{w} \rangle - \sum_i \alpha_i [y_i (\langle \mathbf{w}, \mathbf{x}_i \rangle - b) - 1].$$

$\sum_i \alpha_i y_i = 0$ und $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$ ein und erhalten mit einigen Umformungen

$$\begin{aligned} L(\mathbf{w}, b, \alpha) &= \frac{1}{2} \cdot \langle \mathbf{w}, \mathbf{w} \rangle - \sum_i \alpha_i y_i \langle \mathbf{w}, \mathbf{x}_i \rangle + b \cdot \sum_i \alpha_i y_i + \sum_i \alpha_i \\ &= -\frac{1}{2} \cdot \sum_{ij} \alpha_i y_i \alpha_j y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle + \sum_i \alpha_i. \end{aligned}$$

Summanden
ausmultipliziert

- Dieses duale Problem ist nun bzgl. α zu maximieren:

$$W(\alpha) = \sum_i \alpha_i - \frac{1}{2} \cdot \sum_{ij} \alpha_i y_i \alpha_j y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle.$$

Summe der α_i ist positiv.
Abgezogen werden die
 $\alpha_i y_i \langle \mathbf{w}, \mathbf{x} \rangle$ aller Trainings-
beispiele.

unter den Nebenbedingungen

$$\alpha_i \geq 0 \text{ und } \sum_i \alpha_i y_i = 0 \text{ (aus der Minimierung für } b)$$

Das Optimierungsproblem (4)

Lösung und Vorgehensweise einer SVM:

- 1) Lösen das dualen Problems und Ableitung der $\alpha_i \geq 0$, die $W(\alpha)$ maximieren
 - Punkte mit $\alpha_i > 0$ liegen direkt auf dem Margin \leadsto Stützvektoren
 - Die restlichen Trainingspunkte haben keinen Einfluss $\alpha_i = 0$
- 2) Ermittlung der Trennebene mit maximaler Trennspanne (Margin):
 - Normalenvektor \mathbf{w} nach: $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$,
 - Offset b über alle N_{sup} Stützvektoren nach: $b = 1/N_{\text{sup}} \cdot \sum_{i=1:N_{\text{sup}}} \langle \mathbf{w}, \mathbf{x}_i \rangle - y_i$
- 3) Die Entscheidungsfunktion ist nun:

$$y = h(\mathbf{x}_{\text{neu}}) = \text{sign}(\langle \mathbf{w}, \mathbf{x}_{\text{neu}} \rangle - b) = \text{sign}(\sum_i \alpha_i y_i \langle \mathbf{x}_i, \mathbf{x}_{\text{neu}} \rangle - b).$$

-
- Wichtige Beobachtung: in der Zielfunktion $W(\alpha)$ sowie in der Entscheidungsfunktion $y = h(\mathbf{x}_{\text{neu}})$ treten die Trainingsdaten \mathbf{x}_i nur in Skalarprodukten auf.

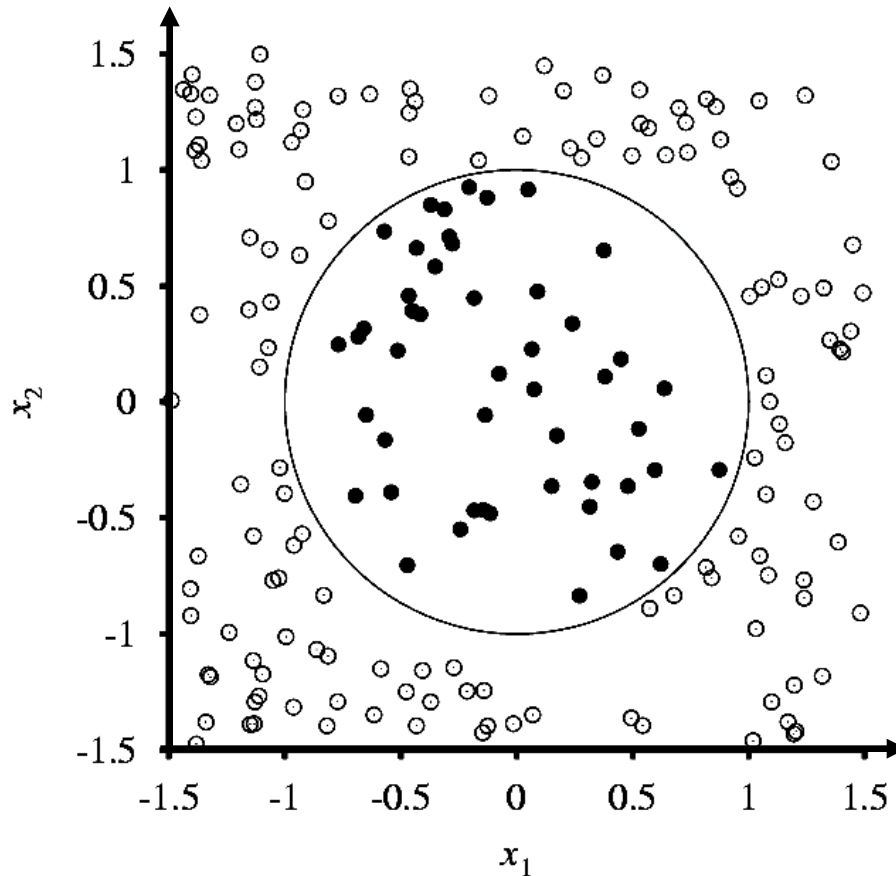
Eigenschaften des Optimierungsproblems

- Das Optimierungsproblem für $W(\alpha)$ ist konvex, hat also keine lokalen Optima!
- Die Lösung der quadrat. Optimierung für $W(\alpha)$ ist effizient implementierbar.
- Die Datenvektoren gehen nur als Skalarprodukte ein! Dies gilt auch für den nach der Bestimmung der Gewichte α_i berechneten Klassifikator selbst:

$$h(\mathbf{x}_{neu}) = \text{sign}\left(\sum_i \alpha_i y_i \langle \mathbf{x}_{neu}, \mathbf{x}_i \rangle - b\right).$$

- Die α_i sind nur für Support-Vektoren größer als Null.
- Die Anzahl der Support-Vektoren ist in der Regel deutlich kleiner als die Anzahl der Trainingsbeispiele.
- Weiterer Vorteil der SVM: die SVM macht keine Annahmen bzgl. der Verteilung der Klassen (z.B. Normalverteilung o. Ä.)

Nichtlinear separierbare Daten



Beispiel:

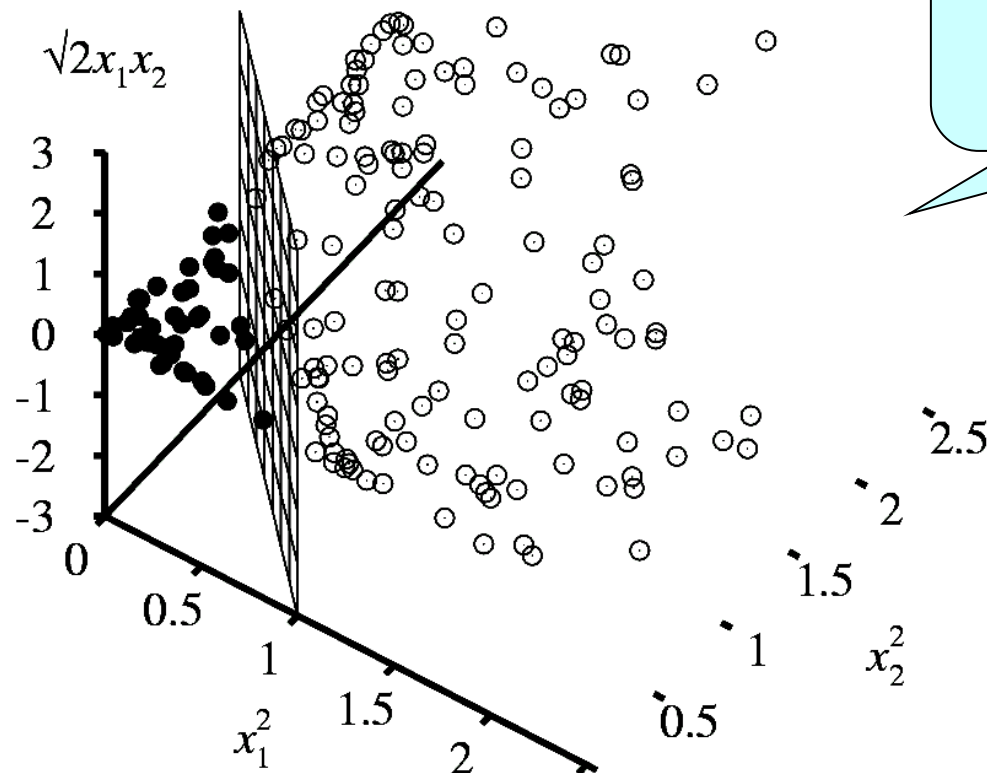
- Geg.: 2-dim. **Datenraum**
- alle positiv klassifizierten Trainingsbeispiele liegen innerhalb einer Kreisregion
- alle negativ klassifizierten Trainingsbeispiele liegen außerhalb der Kreisregion
- Sep.-Grenze: $x_1^2 + x_2^2 \leq 1$.

⇒ Daten sind nicht linear separierbar!

Höher dimensionale Merkmalsräume

- Idee: Transformiere jeden **Datenvektor** $\mathbf{x} = (x_1, x_2)$ des **originalen Datenraums** in einen neuen **Merkmalsvektor** $F(\mathbf{x})$ eines **höherdimensionalen Merkmalsraumes**.

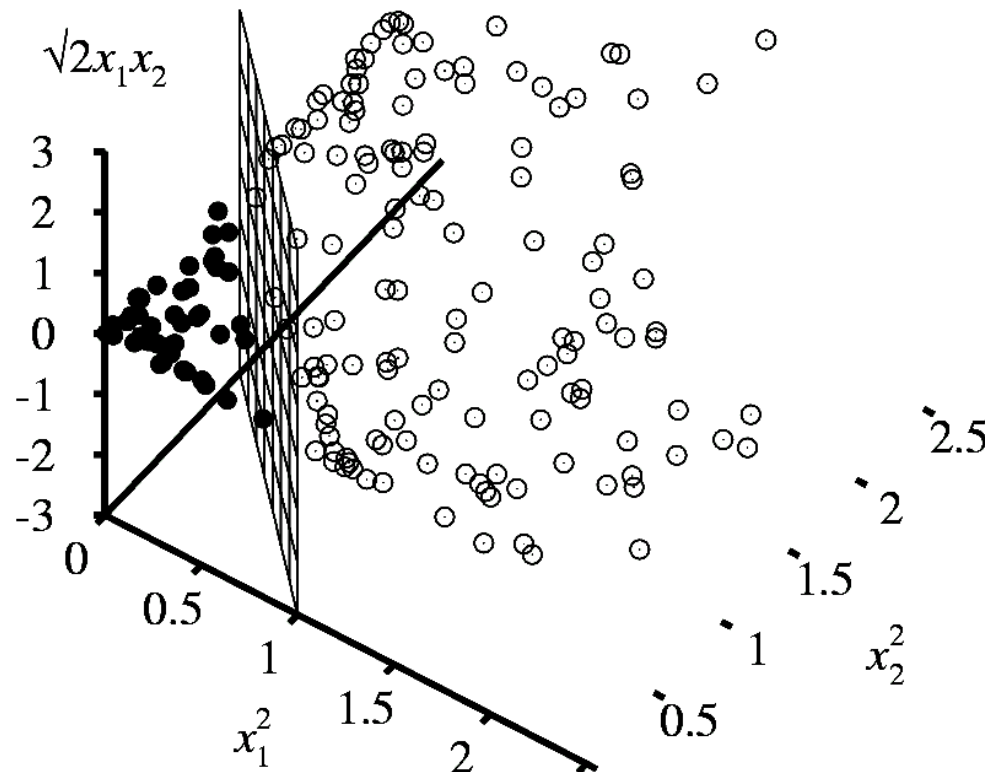
⇒ Daten sind im Merkmalsraum linear separierbar!



Bspl.: $F(\mathbf{x}) = (f_1, f_2, f_3)$
mit $f_1 = x_1^2$,
 $f_2 = x_2^2$,
 $f_3 = (\sqrt{2} \cdot x_1 \cdot x_2)$

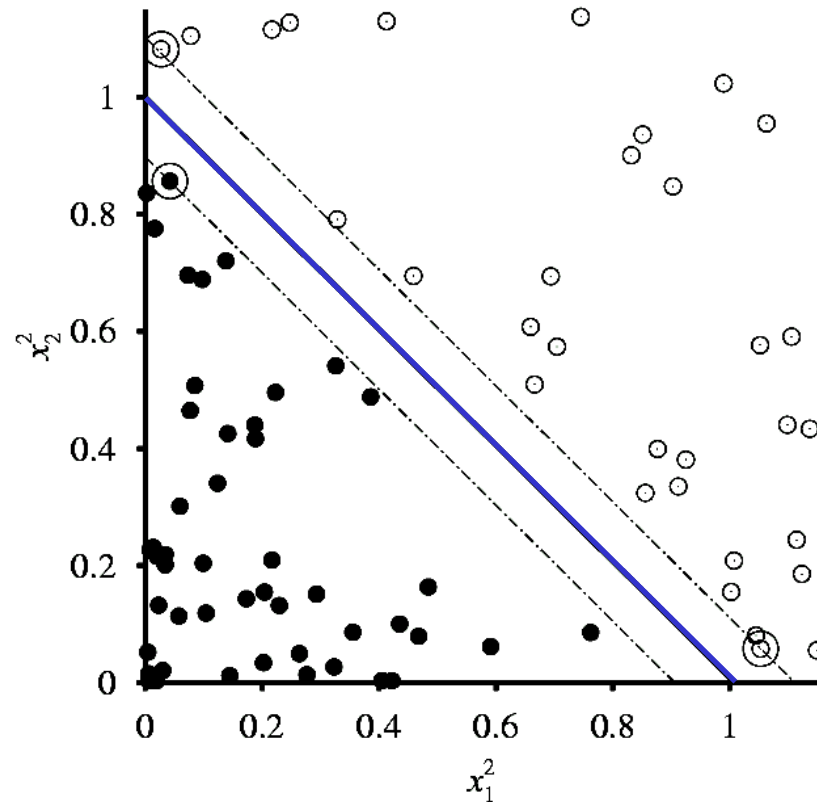
Lineare Separierbarkeit im Merkmalsraum

- Generell gilt: Sofern **Datenvektoren \mathbf{x}_i** in einen neuen **Merkmalsraum (Feature Space)** ausreichend hoher Dimension abgebildet werden, sind sie dort linear separierbar!
- Ausreichende Dimension: sofern n Datenvektoren vorliegen, ist die lineare Separierbarkeit i.A. bei Merkmalsräumen der Dimension $\geq n$ gegeben.



SVM auf nichtlinear separierbaren Daten

- Gefahr: Overfitting!
- Lösung: bestimme den Maximum-Margin-Klassifikator im Merkmalsraum.



SVM auf nichtlinear separierbaren Daten (2)

- Finden des optimalen linearen Klassifikators erweist sich als Instanz folg. Optimierung:

$$\max \left(\sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \right) \quad (*)$$

Zur Erinnerung: der lineare Klassifikator soll aber nicht im **originalen Datenraum**, sondern im **hochdimensionalen Merkmalsraum** gefunden werden!

Also ersetze in (*): $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$ durch $\langle F(\mathbf{x}_i), F(\mathbf{x}_j) \rangle$.

Der Kernel-Trick

Ein hochdimensionaler Merkmalsraum macht also eigentlich nichtlinear separierbare Probleme linear separierbar.

Aber:

- Berechnung der Abbildung in den Merkmalsraum ist evtl. teuer,
- Berechnung der Skalarprodukte im Merkmalsraum ist auch teuer.

Lösung ist der **Kernel-Trick**:

Bei geeigneter Wahl von F kann $\langle F(\mathbf{x}_i), F(\mathbf{x}_j) \rangle$ *effizient* ohne vorherige Abbildung der einzelnen Datenvektoren in den Merkmalsraum berechnet werden.

Anwendung auf Kreisbeispiel

Für das Beispiel mit Abbildung $F(\mathbf{x}) = (f_1, f_2, f_3)$

mit $f_1 = x_1^2, f_2 = x_2^2, f_3 = (\sqrt{2} x_1 x_2)$

entspricht $\langle F(\mathbf{x}_i), F(\mathbf{x}_j) \rangle = \langle \mathbf{x}_i, \mathbf{x}_j \rangle^2$:

$$\begin{aligned}\langle F(\mathbf{x}_i), F(\mathbf{x}_j) \rangle &= \left\langle \begin{pmatrix} x_{i1}^2 \\ x_{i2}^2 \\ \sqrt{2} x_{i1} x_{i2} \end{pmatrix}, \begin{pmatrix} x_{j1}^2 \\ x_{j2}^2 \\ \sqrt{2} x_{j1} x_{j2} \end{pmatrix} \right\rangle \\ &= x_{i1}^2 x_{j1}^2 + 2x_{i1} x_{i2} x_{j1} x_{j2} + x_{i2}^2 x_{j2}^2 \\ &= (x_{i1} x_{j1} + x_{i2} x_{j2})^2 = \langle \mathbf{x}_i, \mathbf{x}_j \rangle^2.\end{aligned}$$

$\langle \mathbf{x}_i, \mathbf{x}_j \rangle^2$ heißt in diesem Kontext **Kernfunktion** oder **Kernel-Funktion** und wird mit $K(\mathbf{x}_i, \mathbf{x}_j)$ bezeichnet.

Verallgemeinert

Allg. wird also eine **Kernfunktion** $K(\mathbf{x}_i, \mathbf{x}_j)$ auf Paare von Datenvektoren angewandt, wenn für diese **Skalarprodukte** $\langle F(\mathbf{x}_i), F(\mathbf{x}_j) \rangle$ in einem entspr. **Merkmalsraum** auszuwerten sind.

Konsequenz: wir können in hochdimensionalen Merkmalsräumen lernen, wobei wir aber lediglich Kernfunktionen berechnen und nicht die vollständige Transformation in den Merkmalsraum.

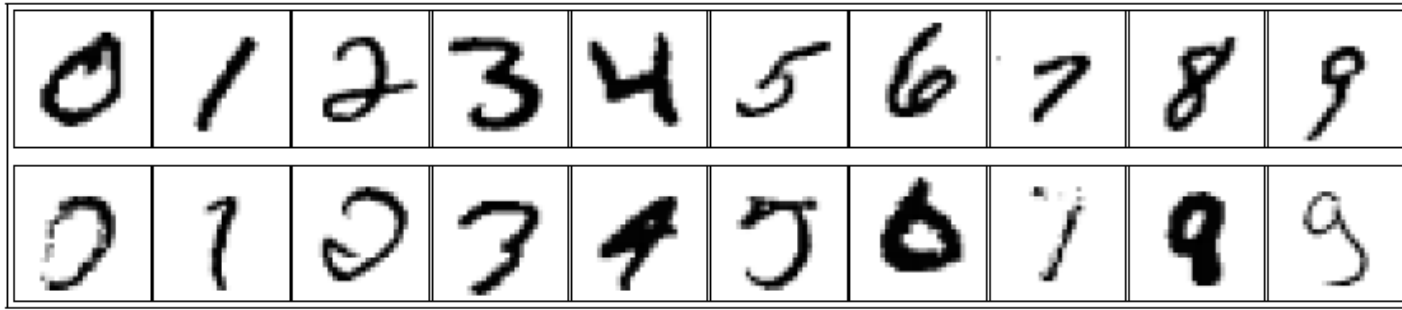
Lernansätze, die Kernfunktionen einsetzen, werden auch als Kernel-Maschinen bezeichnet.

Satz von Mercer

Generalisierung:

- $\langle \mathbf{x}_i, \mathbf{x}_j \rangle^2$ ist natürlich nicht die einzige Kernfunktion $K(\mathbf{x}_i, \mathbf{x}_j)$.
- Andere Kernfunktionen entsprechen anderen Merkmalsräumen.
- **Satz von Mercer** (1909): jede Kernfunktion mit positiv definiter Kernel-Matrix $\mathbf{K}_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$ hat einen korrespondierenden Merkmalsraum.
- Die Merkmalsräume können selbst für einfache Kernel sehr groß sein:
 - z.B. entspricht der polynomiale Kernel $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i \cdot \mathbf{x}_j)^d$ einem Merkmalsraum, dessen Dimension exponentiell in d ist,
 - bei Verwendung solcher Kernel findet man dann optimale lineare Trennungen effizient in Merkmalsräumen mit Milliarden Dimensionen.
- Häufig eingesetzt werden z.B. gaußsche radiale Basisfunktionen $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/c)$.

SVM: Erkennung handgeschriebener Ziffern (1)



Beispiele aus dem NIST*-Datensatz

- mit 60.000 klassifizierten 8-Bit-Grauwertbildern,
- jedes Bild á 20×20 Pixel.

* U.S. Nat. Institute of Science & Technology

SVM: Erkennung handgeschriebener Ziffern (2)

Verfahren im Vergleich:

- **3NN**: einfacher 3-nächste-Nachbarn-Klassifizierer
- **300 Hidden**: Backprop-Neural-Network mit 400 Eingaben (Pixel), 10 Ausgaben und 1 verborgenen Schicht mit 300 Einheiten
- **LeNet**: spezialisiertes neur. Netz, das die Bildstruktur aufgreift mit 32x32 Eingaben, die über 3 verborgene Schichten mit ca. 800, 200 bzw. 30 Einheiten zu 10 Ausgabeeinheiten führt. Die verborgenen Schichten führen schrittweise eine Klassifikation von immer größeren Bildteilbereichen durch. Siehe auch: <http://yann.lecun.com/exdb/lenet/> (16.06.14).
- **Boosted LeNet** kombiniert 3 Kopien von LeNet
- **Virtual SVM**: optimierte SVM mit Kernen, die i.W. auf Produkten benachbarter Pixelpaare basieren anstatt auf Produkten über allen Pixelpaaren.
- **Shape Matching**: Methoden des Computersehens ermitteln korrespondierende Bildbereiche. Die resultierenden Transformationen werden als Distanzmaß für eine 3NN genutzt.

	3 NN	300 Hidden	LeNet	Boosted LeNet	SVM	Virtual SVM	Shape Match
Error rate (pct.)	2.4	1.6	0.9	0.7	1.1	0.56	0.63
Runtime (millisec/digit)	1000	10	30	50	2000	200	
Memory requirements (Mbyte)	12	.49	.012	.21	11		
Training time (days)	0	7	14	30	10		
% rejected to reach 0.5% error	8.1	3.2	1.8	0.5	1.8		

SVM: Zusammenfassung

Die SVM kombiniert im wesentlichen drei Techniken:

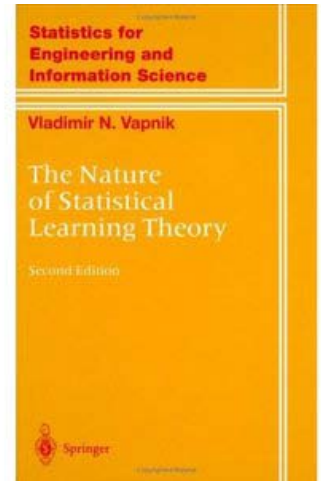
1. „Optimale“ lineare Klassifikation mit Hilfe der Maximum-Margin-Berechnung.
Dies ist ein konvexes Optimierungsproblem.
 2. Repräsentation des Problems in der dualen Darstellung, dadurch Verwendung der Daten nur in Skalarprodukten.
 3. Ersetzung der Skalarprodukte durch Kernfunktionen, die Abstände in höherdimensionalen Merkmalsräumen berechnen.
- Die Kombination der drei Techniken ermöglicht eine effiziente und gut generalisierende Klassifikation für nichtlinear separierbare Daten.
- Public Domain SVM: *mySVM* über <http://www-ai.cs.uni-dortmund.de/SOFTWARE/MYSVM/index.html> (17.06.2013)

Kernel Machines

- Die SVM ist im Prinzip ein linearer Klassifikator, der durch den Kernel-Trick zu einem nichtlinearen erweitert wird.
- Viele lange bekannte lineare Verfahren können auch so erweitert werden.
- Dazu immer erforderlich: Umformung des Verfahrens so, dass Daten nur in Skalarprodukten auftreten.
- In den 90er Jahren Boom der Kernel-Verfahren: SVM, Kernel-LDA, Kernel-PCA, Kernel-ridge-regression, ...
- www.kernel-machines.org (17.06.2013)

Literatur

- Y. LeCun, L. Bottou, Y. Bengio and P. Haffner: Gradient-Based Learning Applied to Document Recognition, Proceedings of the IEEE, 86(11):2278-2324, November 1998.
- B. Schölkopf, A. J. Smola: *Learning with Kernels*, MIT-Press, Cambridge (Mass.), 2002.
- R. O. Duda, P. E. Hart & D. G. Storck: Pattern Classification. 2nd Ed. Wiley Interscience, 2000.
- V. N. Vapnik: *Statistical Learning Theory*, Wiley, N.Y., 1998.



- mySVM - a support vector machine:

<http://www-ai.cs.uni-dortmund.de/SOFTWARE/MYSVM/index.html> (17.06.2013)