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

Inhalt

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: Erlernen eines *Klassifikators*
  aus *m* Trainingsbeispielen $(\boldsymbol{x}_1,y_1),\ldots,(\boldsymbol{x}_m,y_m)$

- Einfachster Fall der binären Klassifikation: jedes Beispiel besteht aus
  - *n*-elementigem Datenvektor $\boldsymbol{x}_i = (x_{i1},\ldots,x_{in})$
  - dessen binärer Klassifikation $y_i \in \{+1,-1\}$

- Bspl.: Klassifikation aller deutschen Internetseiten in die beiden Klassen *„mit Informatikbezug“* und „*ohne Informatikbezug*“:
  - Datenvektoren $\boldsymbol{x}_i$ 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 $\boldsymbol{W} \cdot \boldsymbol{I} = 0$ durch die Perzeptron-Lernregel:

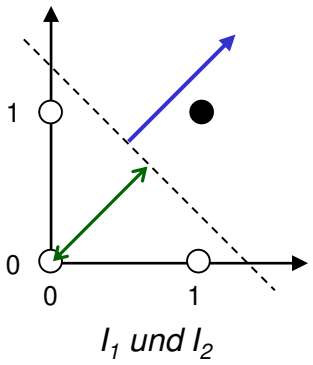
  $w_j \leftarrow w_j + \alpha \cdot I_j \cdot (T\text{-}O)$ *für Eingabe* ***I****, wahre Ausgabe* ***T*** *und errechnete Ausgabe* ***O***

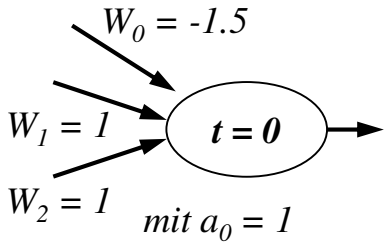
- Bspl. für die Boolesche Funktion ***and***(I_1,I_2) mit $g = step_0$, $a_0 = 1$ und $W_0 = -1.5$

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

  bzw. mit ***x*** *für* ***I***:

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





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

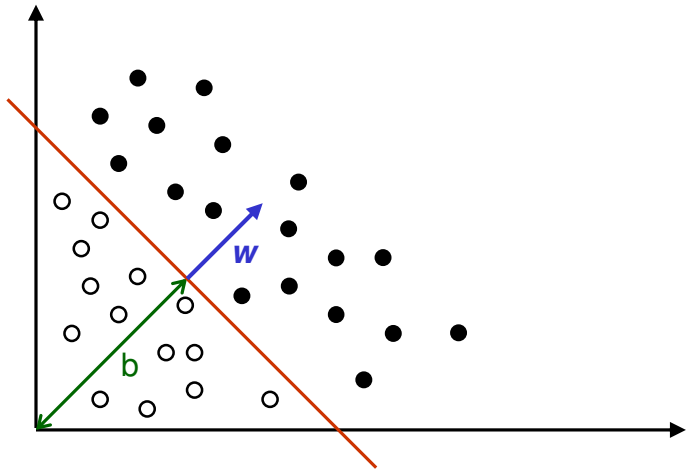
Lineare Klassifikation durch Perzeptrons (2)

Perzeptron bislang:

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

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

- Jetzt allgemein für die binäre Klassifikation $y_i \in \{+1,-1\}$ mit $g = sign$:
  - Trennende Hyperebene $f(\mathbf{x}) = \boldsymbol{w} \cdot \boldsymbol{x} - b = 0$ *mit* $||\boldsymbol{w}|| = 1$
  - Ausgabefunktion $h(\mathbf{x}) = sign(f(\mathbf{x}))$



Trennhyperebene für 2-dim. Trainingsmenge

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

Lineare Klassifikation durch Perzeptrons (3)

Perzeptron:

- Perzeptron-Lernregel neu formuliert:

Bei negat. Produkt unterschiedliche Vorzeichen von y_i und $sign(\langle \boldsymbol{w}_t, \boldsymbol{x}_i \rangle) \rightsquigarrow$ **Fehler**

Bisher:

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

$$\textbf{if } y_i \cdot sign(\langle \boldsymbol{w}_t, \boldsymbol{x}_i \rangle) < 0 \textbf{ then}$$

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

$$t \leftarrow t+1$$

Hier: Schwellwert b über zusätzliche Eingabekante mit $w_0 = b$ und $a_0 = -1$ kodiert, also $f(\boldsymbol{x}_i) = \langle \boldsymbol{w}_t, \boldsymbol{x}_i \rangle$ und Ausgabe $h(\boldsymbol{x}_i) = sign(f(\boldsymbol{x}_i))$

y_i entspricht jetzt T, $\boldsymbol{x}_i$ entspricht I_j und $sign(\langle \boldsymbol{w}_t, \boldsymbol{x}_i \rangle)$ entspricht O

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

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

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

$\rightsquigarrow$ es werden nur informative Punkte (Fehlklassifikationen) benutzt

$\rightsquigarrow$ die Koeffizienten der Punkte reflektieren deren Bedeutung

Perzeptrons: Duale Repräsentation

Perzeptron:

- Neue duale Formulierung für trennende Hyperebene

$$f(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} = \langle \boldsymbol{w}, \boldsymbol{x} \rangle = \sum_j \alpha_j \cdot y_j \cdot \langle \boldsymbol{x}_j, \boldsymbol{x} \rangle = 0$$

w als Linearkombination der Trainingsbeispiele

  mit ***w*** als Linearkombination aus den Trainingsbeispielen: $\boldsymbol{w} = \sum_j \alpha_j \cdot y_j \cdot \boldsymbol{x}_j$

- Damit ist auch die Lernregel neu formulierbar:

$$\textbf{\textit{if}}\ \ y_i \cdot sign\left(\sum_j \alpha_j \cdot y_j \cdot \langle \boldsymbol{x}_j, \boldsymbol{x}_i \rangle\right) < 0\ \ \textbf{\textit{then}}\ \ \alpha_i \leftarrow \alpha_i + \eta$$

  Anmerkung: hier treten die Daten nur in Skalarprodukten auf

- Wozu das Ganze?

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

- Wahl der Trennebene?

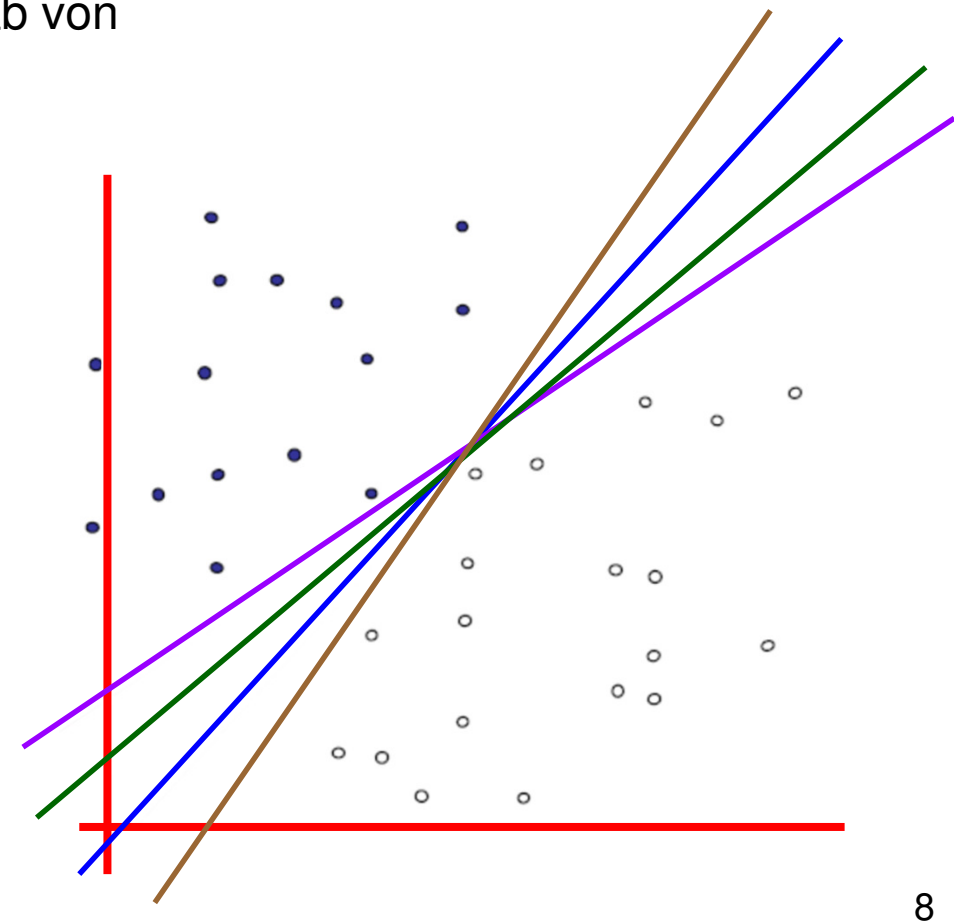
Auswahl der Trennungsebene

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?*

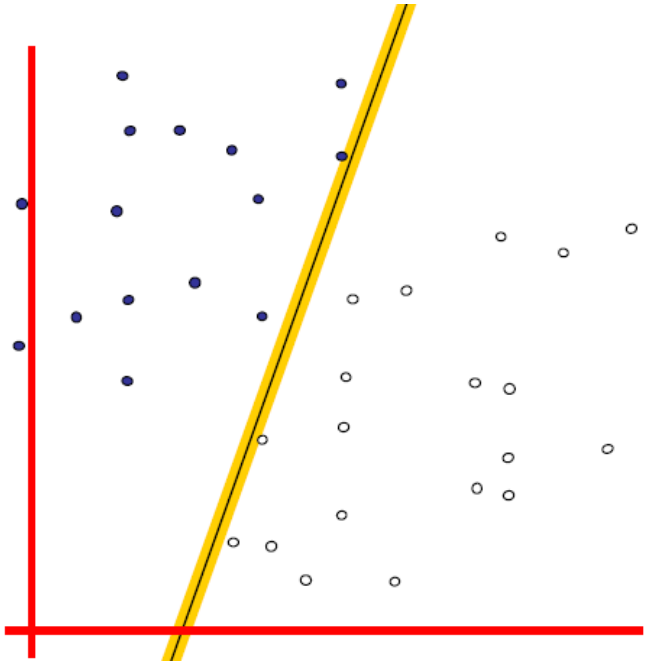


Maximum Margin

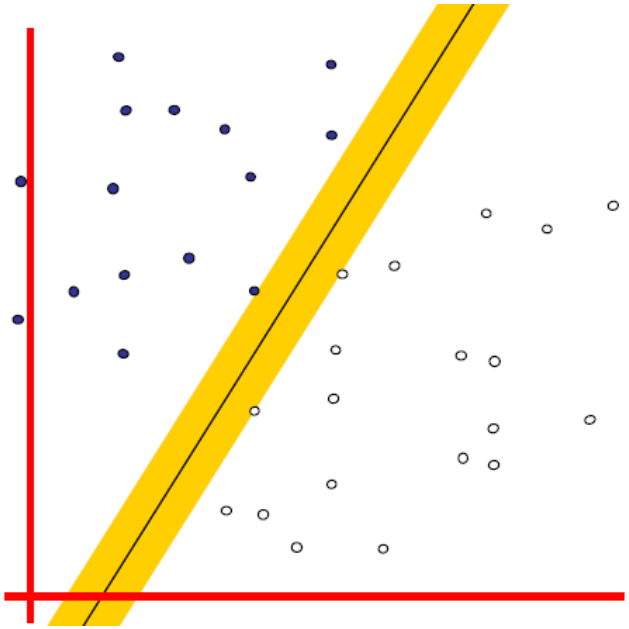
Die *Support Vectore Machine* (*SVM*) wählt die Trennebene,
welche den kleinsten Abstand der Trainingsbeispiele zur Trennebene,
die sog. *Trennspanne* (engl. *Margin*), maximiert

$\rightsquigarrow$ *Maximum Margin Classification*

Kleiner Margin

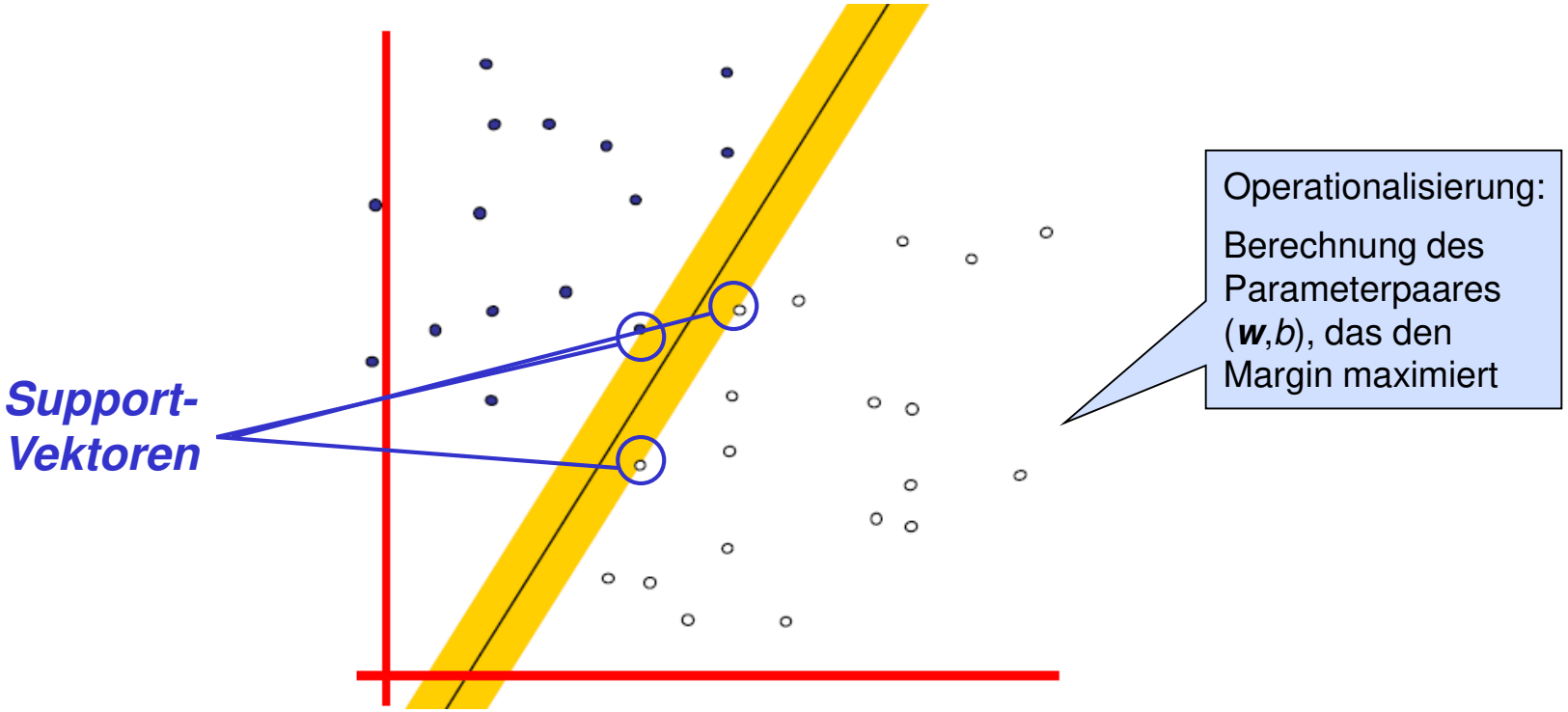


Großer Margin



Support-Vektoren

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



Maximum Margin = Minimal Norm

- Vor.: Wir normieren $f(\mathbf{x}) = \mathbf{w}\cdot\mathbf{x} - b$ so, 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 $\langle \mathbf{w}, \mathbf{x}^- \rangle - b = -1$

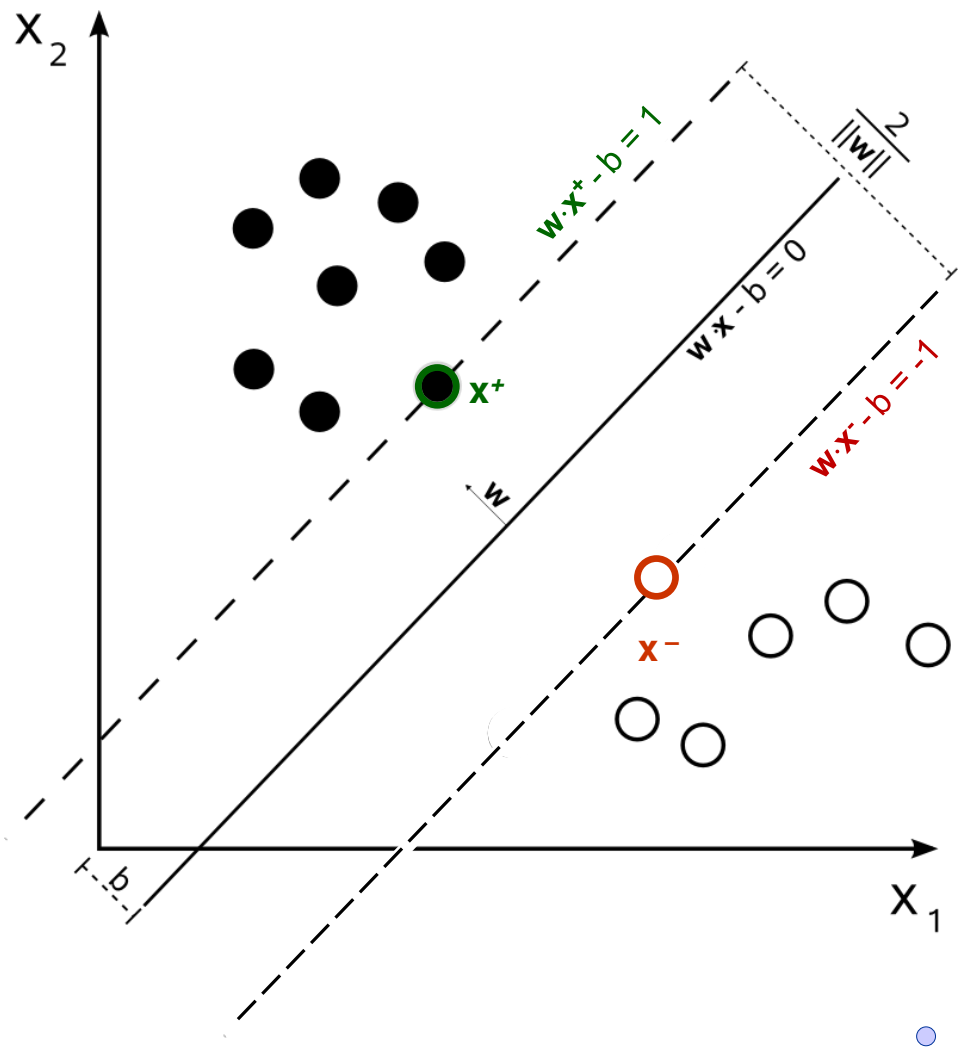
$\rightsquigarrow \langle \mathbf{w}, (\mathbf{x}^+ - \mathbf{x}^-) \rangle = 2$

$\rightsquigarrow M = \langle (\mathbf{w}/\|\mathbf{w}\|), (\mathbf{x}^+ - \mathbf{x}^-) \rangle = 2/\|\mathbf{w}\|$

für Stützvektoren $\mathbf{x}^+$ und $\mathbf{x}^-$

- Maximieren von $M = 2/\|\mathbf{w}\|$ heißt $\|\boldsymbol{w}\|/2$ minimieren …

  … unter der Nebenbedingung, dass alle Trainingsbeispiele korrekt klassifiziert sein müssen



Das Optimierungsproblem (1)

Minimieren von $||\boldsymbol{w}||/2$ ist analog zu:

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

unter den Nebenbedingungen

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

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

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

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

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

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

$$L(\mathbf{w},b,\boldsymbol{\alpha}) = ½\cdot\langle\mathbf{w},\mathbf{w}\rangle - \sum_i \alpha_i \,[\, y_i\,(\langle\mathbf{w},\mathbf{x}_i\rangle - b) - 1]$$

- Wir setzen dazu

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

und erhalten

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

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

↝ Folgefolie

Das Optimierungsproblem (3)

- Wir setzen in

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

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

$$L(\mathbf{w},b,\boldsymbol{\alpha}) = \tfrac{1}{2}\cdot\langle \mathbf{w},\mathbf{w}\rangle - \sum_i \alpha_i y_i \langle \mathbf{w}\ ,\mathbf{x}_i\rangle + b\cdot\sum_i \alpha_i y_i + \sum_i \alpha_i$$

$$= -\tfrac{1}{2}\cdot \sum_{ij} \alpha_i y_i \alpha_j y_j \langle \boldsymbol{x}_i\ ,\boldsymbol{x}_j\rangle + \sum_i \alpha_i\ .$$

Summanden ausmultipliziert

- Dieses duale Problem ist nun bzgl. $\boldsymbol{\alpha}$ zu maximieren:

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

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

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 des dualen Problems und Herleitung der $\alpha_i \geq 0$, die $W(\boldsymbol{\alpha})$ maximieren
   - Punkte mit $\alpha_i > 0$ liegen direkt auf dem Margin ⇝ 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: $b = 1/N_{sup} \cdot \Sigma_{i=1:N_{sup}} \langle \mathbf{w}, \mathbf{x}_i \rangle - y_i$

3) Die Entscheidungsfunktion für ungesehene Beispiele $\mathbf{x}_{neu}$ ist nun:

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

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

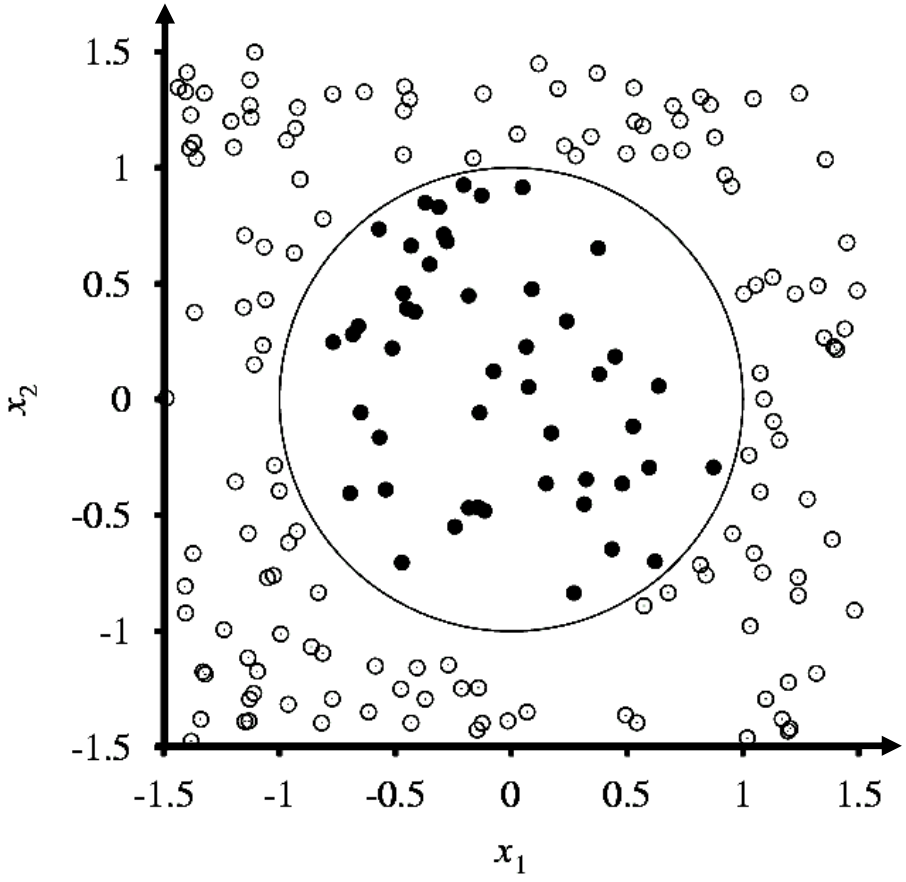
Eigenschaften des Optimierungsproblems

- Das Optimierungsproblem für W($\boldsymbol{\alpha}$) ist konvex, hat also keine lokalen Optima!
- Die Lösung der quadrat. Optimierung für W($\boldsymbol{\alpha}$) 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:

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

- 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: $x_1^2 + x_2^2 \leq 1$

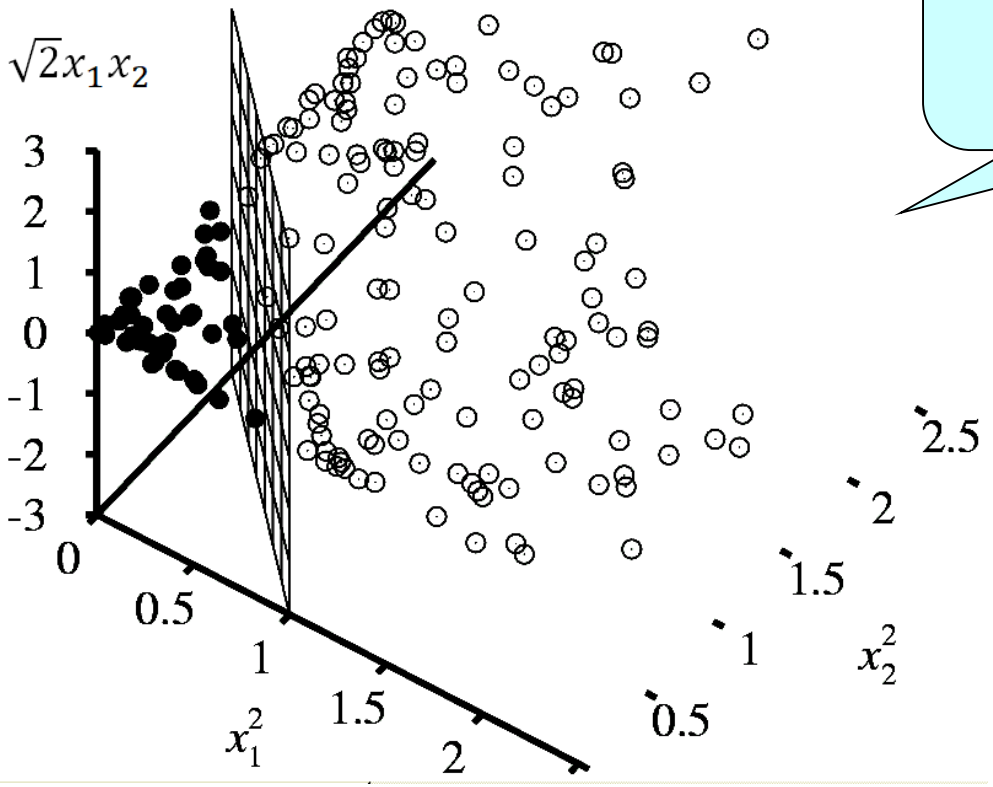
→ Daten sind nicht linear separierbar

Höherdimensionale Merkmalsräume

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

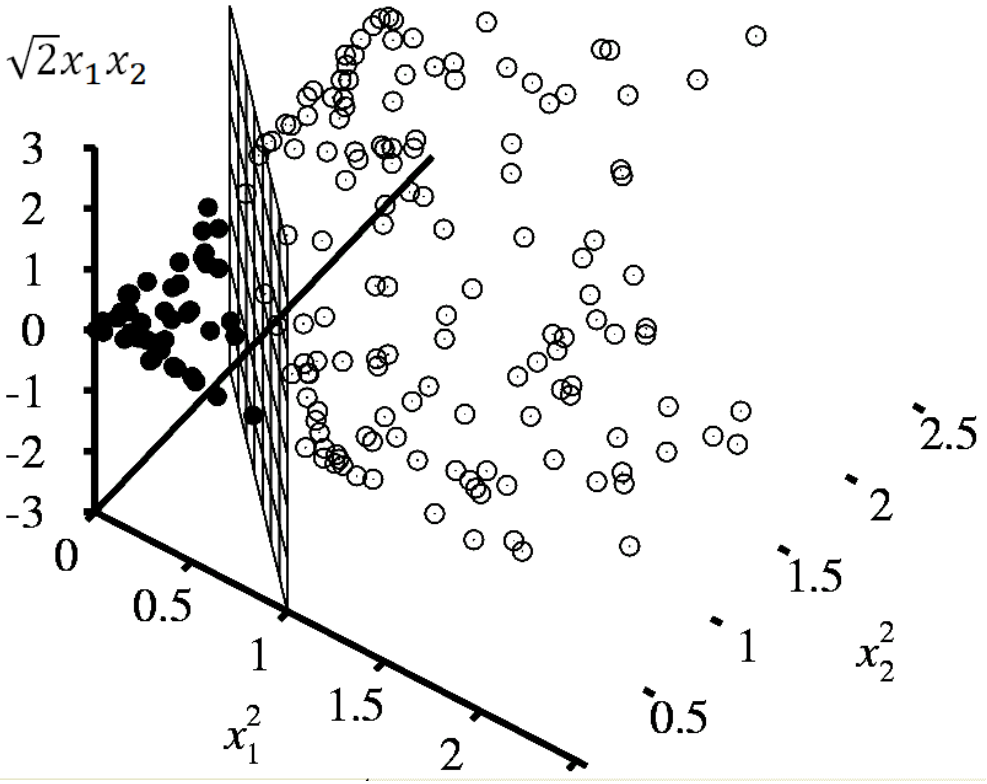
→ Daten sind im Merkmalsraum linear separierbar!

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



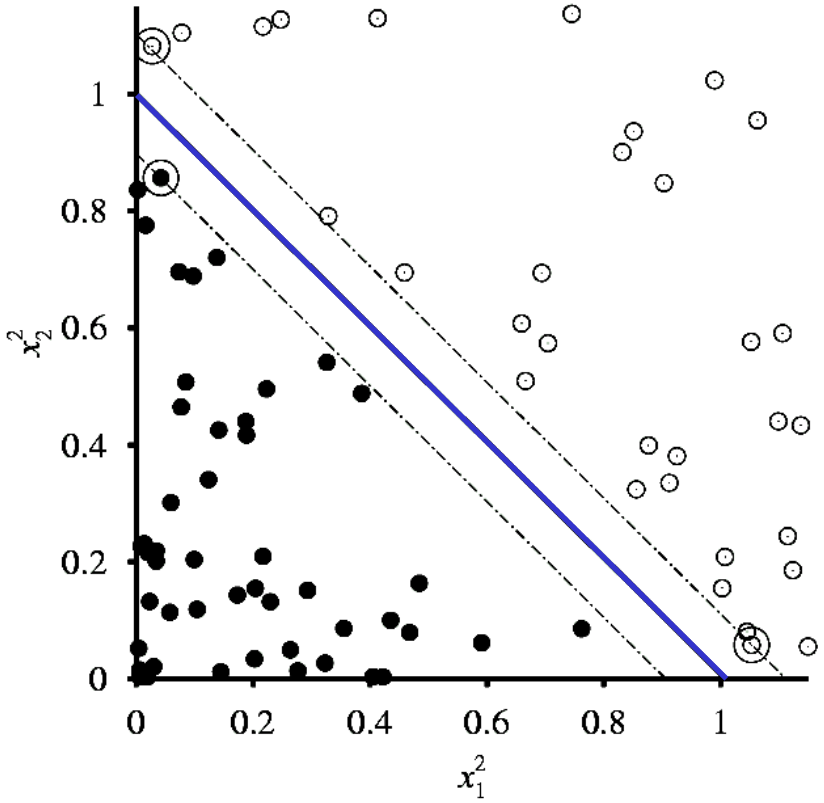
Lineare Separierbarkeit im Merkmalsraum

- Generell gilt: Sofern Datenvektoren $\boldsymbol{x}_i$ 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:

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

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 kann teuer sein
- Berechnung der Skalarprodukte im Merkmalsraum ist auch teuer

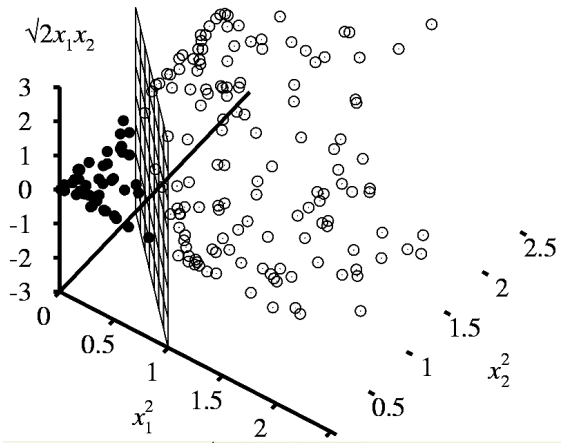
Lösung ist der Kernel-Trick:

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

Anwendung auf Kreisbeispiel

Im Beispiel wurde gewählt Abbildung $F(\boldsymbol{x}) = (f_1, f_2, f_3)$

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



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

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

$$= x_{i1}^2 x_{j1}^2 + 2 x_{i1} x_{i2} x_{j1} x_{j2} + x_{i2}^2 x_{j2}^2$$

$$= \left(x_{i1} x_{j1} + x_{i2} x_{j2} \right)^2 = \left\langle \boldsymbol{x}_i, \boldsymbol{x}_j \right\rangle^2 .$$

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

Verallgemeinert

- Allg. wird also eine Kernfunktion $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ auf Paare von Datenvektoren angewandt, wenn für diese Skalarprodukte $\langle F(\boldsymbol{x}_i), F(\boldsymbol{x}_j) \rangle$ 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

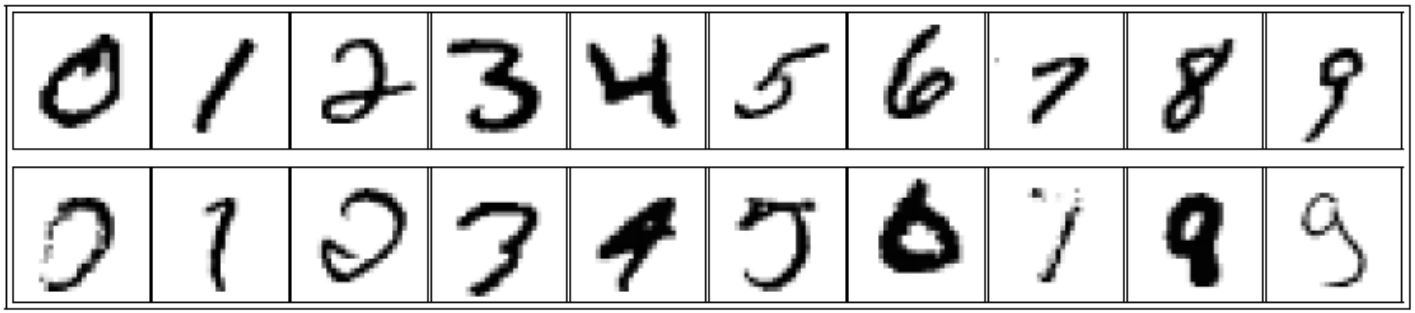
- Lernansätze, die Kernfunktionen einsetzen, werden auch als Kernel-Maschinen bezeichnet

Satz von Mercer

Generalisierung:

- $\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle^2$ ist natürlich nicht die einzige Kernfunktion $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$
- Andere Kernfunktionen entsprechen anderen hochdim. Merkmalsräumen
- Satz von Mercer (1909): jede Kernfunktion mit positiv definiter Kernel-Matrix $\boldsymbol{K}_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ hat einen korrespondierenden Merkmalsraum
- Die Merkmalsräume können selbst für einfache Kernel sehr groß sein:
  - z.B. entspricht der polynomiale Kernel $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (1+\boldsymbol{x}_i \cdot \boldsymbol{x}_j)^d$ 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 radiale Basisfunktionen $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = exp(||\boldsymbol{x}_i - \boldsymbol{x}_j||^2/c)$

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 (Stand 2003):

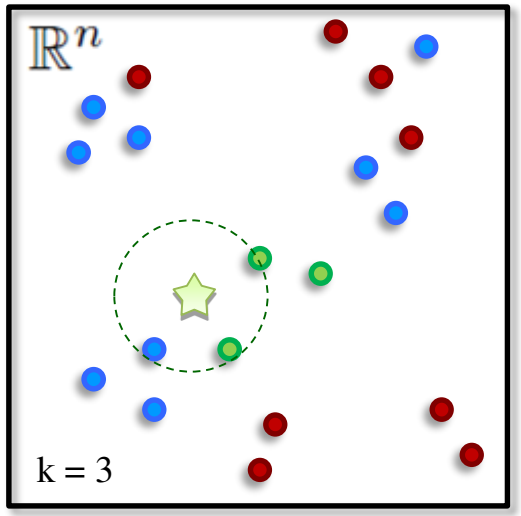
- ***3NN***: einfacher 3-nächste-Nachbarn-Klassifizierer
- ***300 Hidden***: Backprop-Neural-Network mit 400 Eingabeeinheiten (Pixel), 10 Ausgabeeinheiten und 1 verborgenen Schicht mit 300 Einheiten
- ***LeNet***: ein Konvolutionsnetzwerk (Stand 1998), das die Bildstruktur aufgreift mit 32x32 Eingaben, 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 (http://yann.lecun.com/exdb/lenet/ (19.06.20))
- ***Boosted LeNet*** kombiniert 3 Kopien von LeNet in einem Boosting-Ansatz
- ***Virtual SVM***: optimierte SVM mit Kerneln, 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: Erkennung handgeschriebener Ziffern (3)

Zu *k*-NN:

- *k*-NN = *k*-nächste-Nachbarn-Algorithmus ist ein Klassifikationsverfahren, das die Klassenzuordnung einer unbekannten Stichprobe durch Vergleich mit den *k* nächsten Nachbarn vorgenommen wird
- Das Lernen bei *k*-NN besteht also nur aus dem Abspeichern der Trainingsbeispiele



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.

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, ...

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.