Lineare Klassifikatoren

Volker Tresp

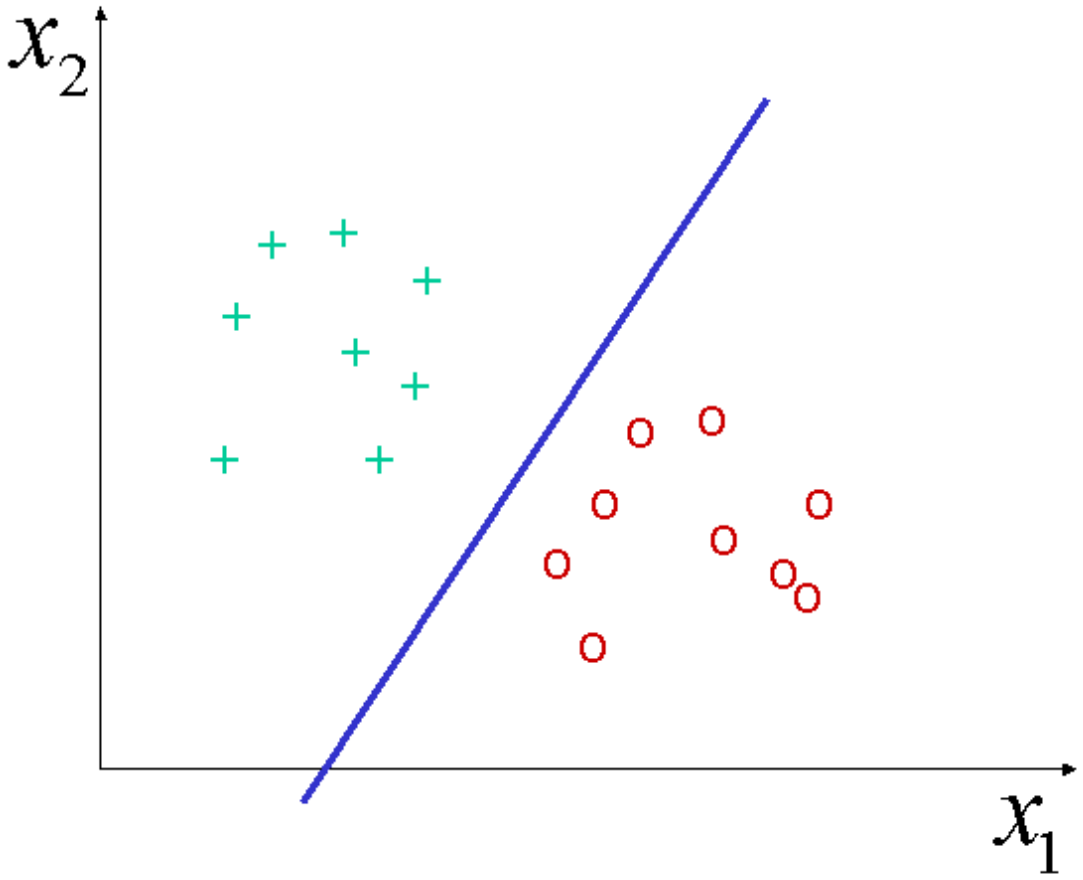
Einführung

- Lineare Klassifikatoren trennen Klassen durch eine lineare Hyperebene (genauer: affine Menge)
- In hoch dimensionalen Problemen trennt schon eine lineare Trennebene die Klassen
- Lineare Klassifikatoren können nicht das *exclusive-or* Problem lösen
- Im Zusammenhang mit Basisfunktionen oder Kernfunktionen können jedoch beliebige Trennflächen modelliert werden

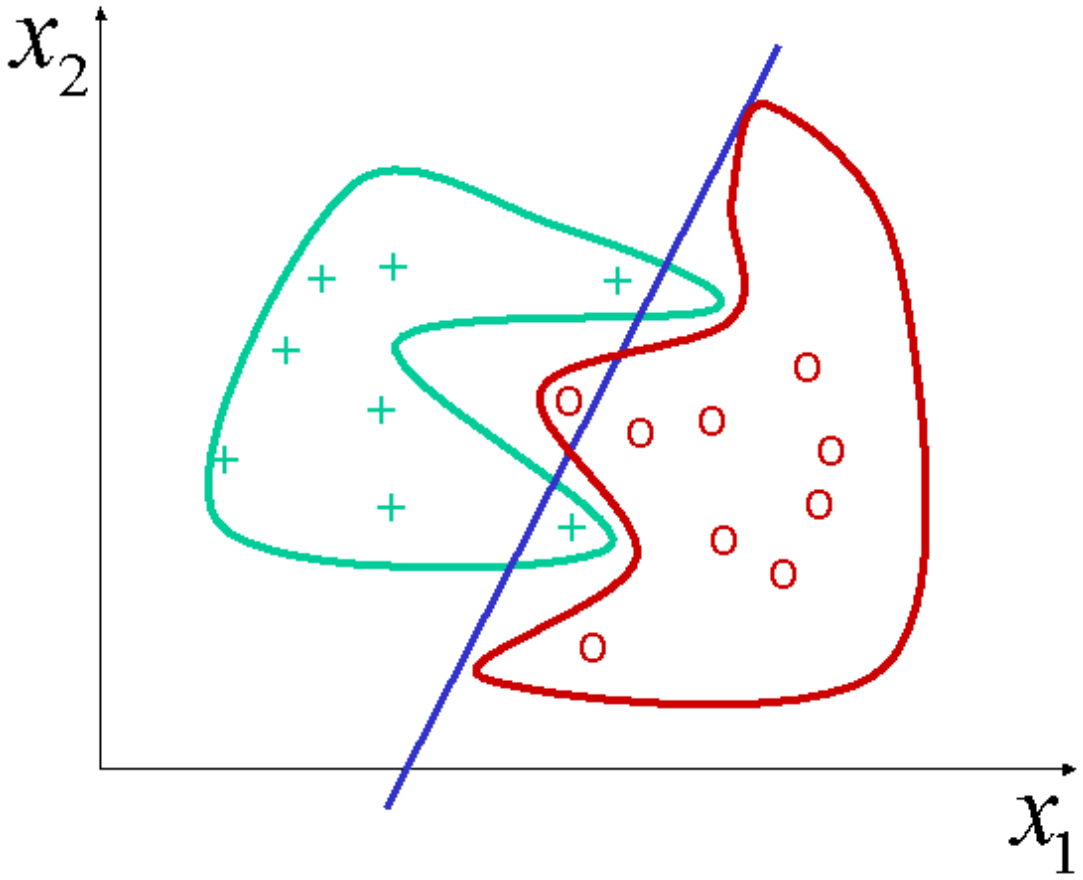
Einführung (2)

- Wir werden uns zunächst auf Klassifikatoren konzentrieren, die zwei Klassen $y_i = 1$ und $y_i = 0$ (oder $y_i = 1$ und $y_i = -1$) voneinander trennen
- Das *Perzeptron* hatten wir bereits kennengelernt. Wir behandeln hier folgende weitere Ansätze
  - I. Generatives Modell zur Klassifikation
  - II. Logistische Regression
  - III. Klassifikation durch Regression

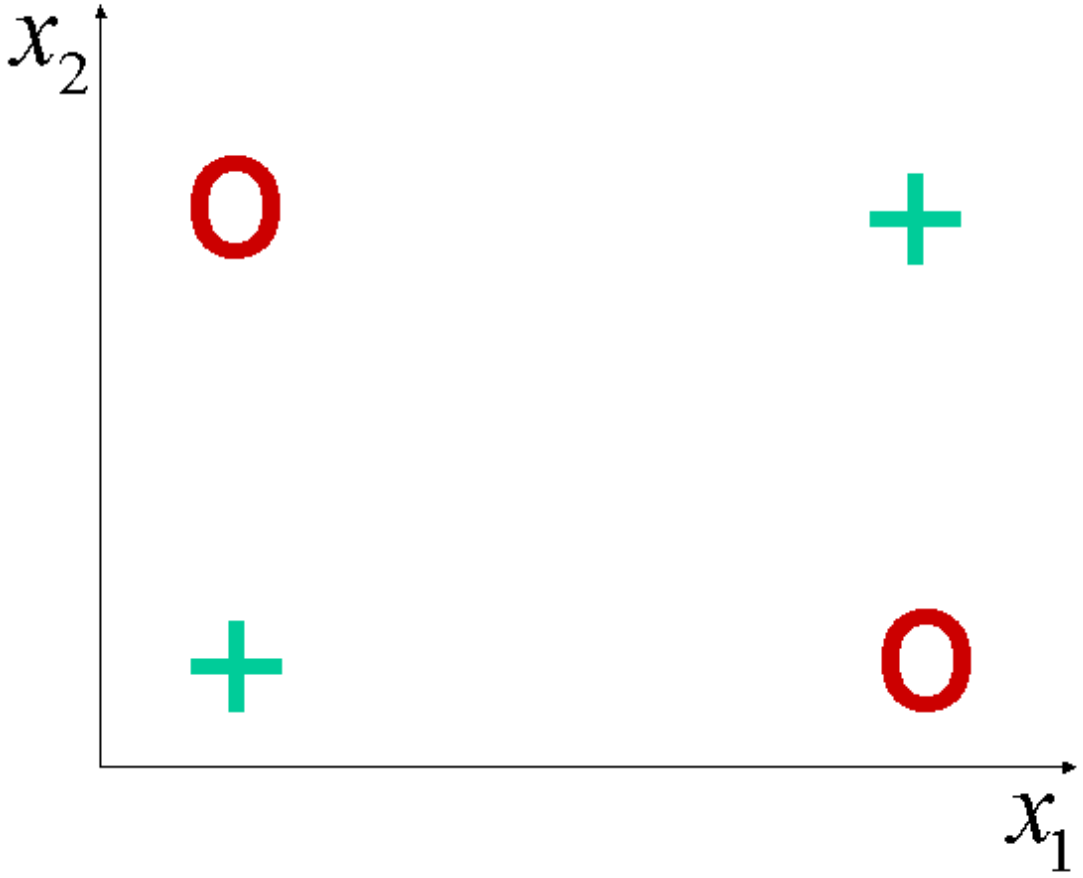
Zwei linear-separierbare Klassen



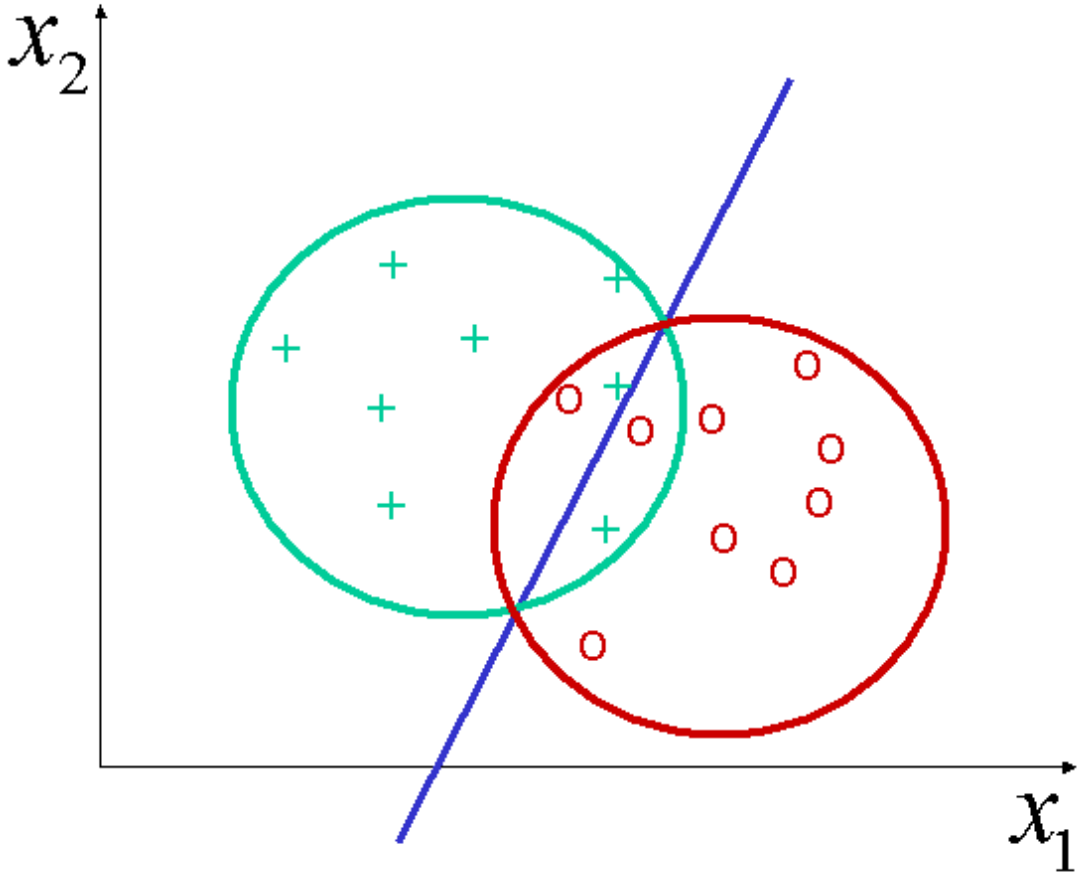
Zwei Klassen, die nicht linear separierbar sind



Das klassische Beispiel nicht-separierbarer Klassen: XOR



Separierbarkeit ist kein Ziel an sich: überlappende Klassen



I. Generatives Modell zur Klassifikation

- In einem generativen Modell modelliert man einen angenommenen datengenerierenden Prozess
- Wir nehmen an, dass die beobachtete Klasse y_i mit Wahrscheinlichkeit

$$P(y_i = 1) = \kappa_1 \qquad P(y_i = 0) = \kappa_0 = 1 - \kappa_1$$

  generiert wurde. Anschließend wurde nach $P(\mathbf{x}_i|y_i)$ ein Datenpunkt $\mathbf{x}_i$ erzeugt
- (Beachte, dass nur hier $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,M})^T$, das heißt $\mathbf{x}_i$ enthält nicht den Bias $x_{i,0}$)

Satz von Bayes

- Wollen wir nun einen Datenpunkt $\mathbf{x}_i$ klassifizieren für den y_i unbekannt ist, so bemühen wir den Satz von Bayes und erhalten

$$P(y_i|\mathbf{x}_i) = \frac{P(\mathbf{x}_i|y_i)P(y_i)}{P(\mathbf{x}_i)}$$

$$P(\mathbf{x}_i) = P(\mathbf{x}_i|y_i = 1)P(y_i = 1) + P(\mathbf{x}_i|y_i = 0)P(y_i = 0)$$

- Maximum-likelihood Schätzer für die Klassenwahrscheinlichkeiten sind

$$\widehat{P}(y_i = 1) = \widehat{\kappa}_1 = N_1/N$$

und

$$\widehat{P}(y_i = 0) = \widehat{\kappa}_0 = N_0/N = 1 - \widehat{\kappa}_1$$

wobei N_1 und N_0 die Anzahl der Trainingsmuster der Klasse 1 b.z.w. der Klasse 0 sind

Klassenspezifische Verteilungen

- Zur Modellierung von $P(\mathbf{x}_i|y_i)$ kann man nun problemangepasste Parametrierungen wählen
- Eine mögliche Wahl sind multivariate Gauß-Verteilungen

$$P(\mathbf{x}_i|y_i = l) = \mathcal{N}(\mathbf{x}_i; \mu^{(l)}, \Sigma)$$

mit

$$\mathcal{N}\left(\mathbf{x}_i; \mu^{(l)}, \Sigma\right) = \frac{1}{(2\pi)^{M/2}\sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}\left(\mathbf{x}_i - \mu^{(l)}\right)^T \Sigma^{-1}\left(\mathbf{x}_i - \mu^{(l)}\right)\right)$$

- Beachte, dass beide Gauss-Verteilungen die gleiche Kovarianzmatrix aber andere Zentren besitzen (diese Wahl ist nicht zwingend, hat sich aber als oft sinnvoll erwiesen)

Maximum-likelihood Schätzer für Zentren und Kovarianzen

- Man erhält als maximum-likelihood Schätzer für die Zentren

$$\widehat{\mu}^{(l)} = \frac{1}{N_l} \sum_{i:y_i=l} \mathbf{x}_i$$

- Man erhält als unverzerrte (*unbiased*) Schätzer für die Kovarianzmatrix

$$\widehat{\Sigma} = \frac{1}{N-M} \sum_{l=0}^{1} \sum_{i:y_i=l} (\mathbf{x}_i - \widehat{\mu}^{(l)})(\mathbf{x}_i - \widehat{\mu}^{(l)})^T$$

Abgeleitete a posteriori Verteilung

- Es folgt

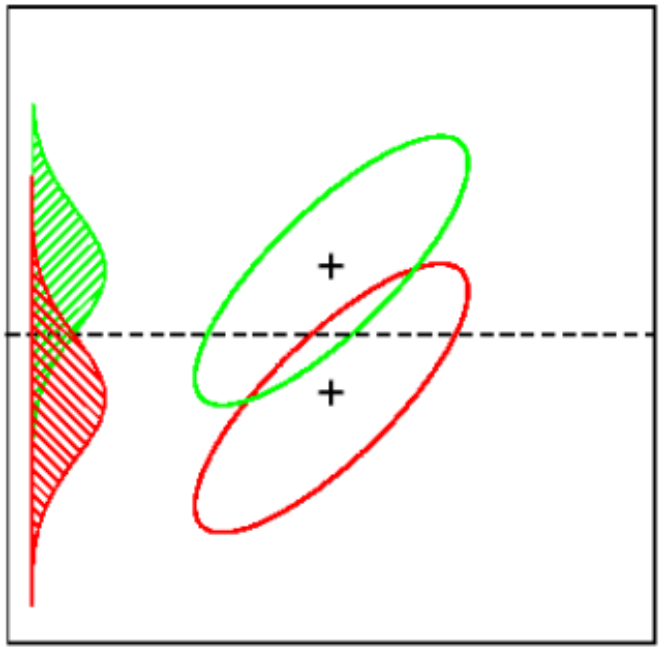
$$P(y_i = 1|\mathbf{x}_i) = \frac{P(\mathbf{x}_i|y_i = 1)P(y_i = 1)}{P(\mathbf{x}_i|y_i = 1)P(y_i = 1) + P(\mathbf{x}_i|y_i = 0)P(y_i = 0)}$$

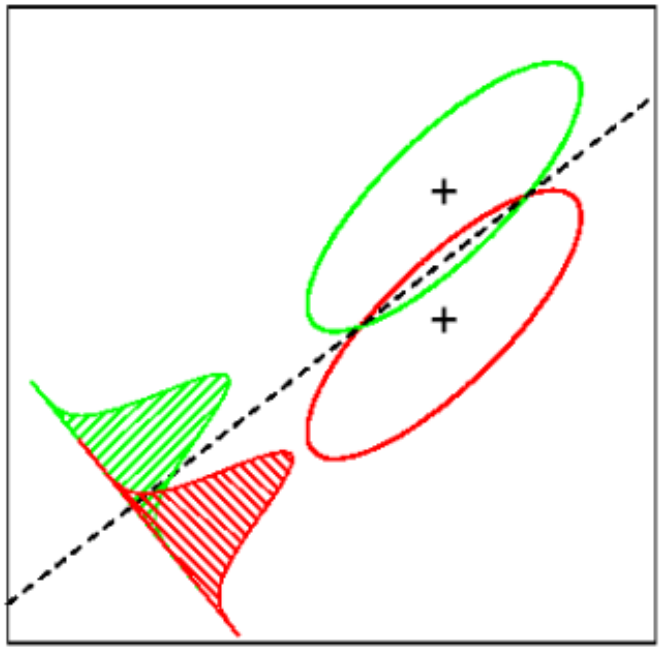
$$= \frac{1}{1 + \frac{\kappa_0}{\kappa_1} \exp\left((\mu^{(0)} - \mu^{(1)})^T \Sigma^{-1} \mathbf{x}_i + \frac{1}{2}\mu^{(0)^T} \Sigma^{-1} \mu^{(0)} - \frac{1}{2}\mu^{(1)^T} \Sigma^{-1} \mu^{(1)}\right)}$$

$$= \text{sig}\left(\begin{pmatrix} 1 \\ \mathbf{x}_i \end{pmatrix}^T \begin{pmatrix} w_0 \\ \mathbf{w} \end{pmatrix}\right)$$

$$\mathbf{w} = \Sigma^{-1}\left(\mu^{(1)} - \mu^{(0)}\right)$$

$$w_0 = -\log \kappa_0/\kappa_1 - \frac{1}{2}\mu^{(0)^T} \Sigma^{-1} \mu^{(0)} + \frac{1}{2}\mu^{(1)^T} \Sigma^{-1} \mu^{(1)}$$





Kommentare

- Der generative Klassifikator führte zu einer linearen Trennebene
- Der vorgestellte Ansatz ist analog zu Fishers linearer Diskriminantenanalyse, bei der eine Projektion der Daten auf eine Dimension gesucht, so dass die Daten der gleichen Klasse eine kleine Varianz besitzen und gleichzeitig der Abstand der Zentren der beiden Klassen in der Projektion maximal ist
- Das heißt man bekommt das gleiche Ergebnis aus einem Optimalitätskriterium ohne Gauss-Verteilungen annehmen zu müssen

II. Logistische Regression

- Das generative Modell motivierte

$$P(y_i = 1|\mathbf{x}_i) = \text{sig}\left(\mathbf{x}_i^T\mathbf{w}\right)$$

(jetzt haben wir wieder $\mathbf{x}_i^T = (x_{i,0} = i, 1, x_{i,1}, \ldots, x_{i,M-1})^T$)

- Es ist nun nahe liegend, dass man die Likelihood des bedingten Modells optimiert

$$L(\mathbf{w}) = \prod_{i:y_i=1} \text{sig}\left(\mathbf{x}_i^T\mathbf{w}\right) \prod_{i:y_i=0} \left(1 - \text{sig}\left(\mathbf{x}_i^T\mathbf{w}\right)\right)$$

Log-Likelihood

- Log-Likelihood-Kostenfunktion

$$l = \sum_{i:y_i=1} \log\left(\frac{1}{1+\exp(-\mathbf{x}_i^T\mathbf{w})}\right) + \sum_{i:y_i=0} \log\left(\frac{1}{1+\exp(\mathbf{x}_i^T\mathbf{w})}\right)$$

$$= -\sum_{i:y_i=1} \log(1+\exp(-\mathbf{x}_i^T\mathbf{w})) - \sum_{i:y_i=0} \log(1+\exp(\mathbf{x}_i^T\mathbf{w}))$$

Adaption

- Die Ableitung der Log-Likelihood nach den Gewichten

$$\frac{\partial l}{\partial \mathbf{w}} = \sum_{i:y_i=1} \frac{\mathbf{x}_i \exp(-\mathbf{x}_i^T \mathbf{w})}{1+\exp(-\mathbf{x}_i^T \mathbf{w})} - \sum_{i:y_i=0} \frac{\mathbf{x}_i \exp(\mathbf{x}_i^T \mathbf{w})}{1+\exp(\mathbf{x}_i^T \mathbf{w})}$$

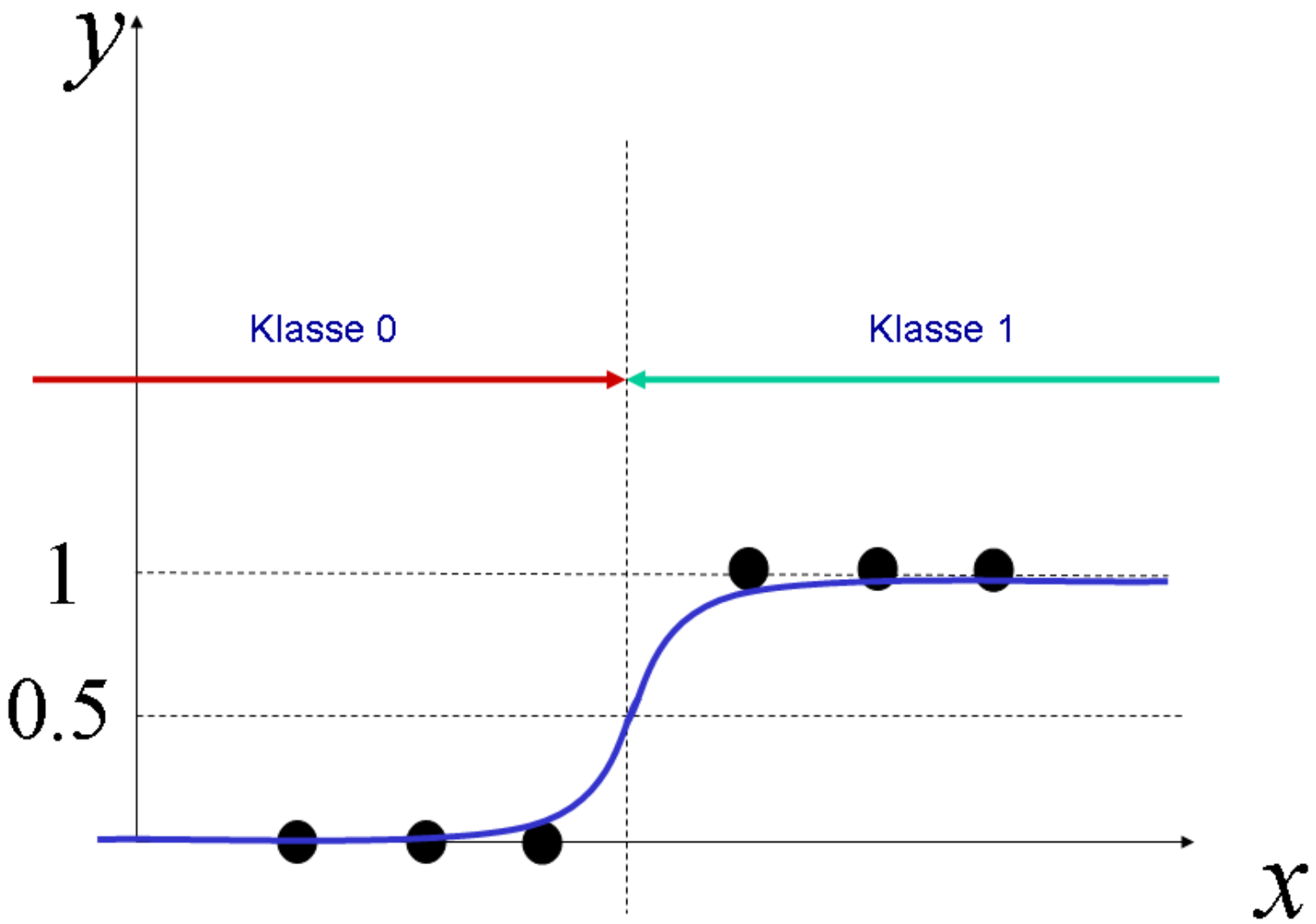
$$= \sum_{i:y_i=1} \mathbf{x}_i(1-\mathrm{sig}(\mathbf{x}_i^T \mathbf{w})) - \sum_{i:y_i=0} \mathbf{x}_i \mathrm{sig}(\mathbf{x}_i^T \mathbf{w})$$

$$= \sum_{i=1}^{N} (y_i - \mathrm{sig}(\mathbf{x}_i^T \mathbf{w}))\mathbf{x}_i$$

- Gradientenbasierte Optimierung der Parameter (zur *Maximierung* der Log-Likelihood)

$$\mathbf{w} \longleftarrow \mathbf{w} + \eta \frac{\partial l}{\partial \mathbf{w}}$$

- Üblicherweise wird jedoch ein Newton-Raphson Verfahren zur Optimierung benutzt


y
Klasse 0
Klasse 1
1
0.5
x

III. Klassifikation durch Regression

- Lineare Regression:

$$f(\mathbf{x}_i, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j x_{i,j}$$

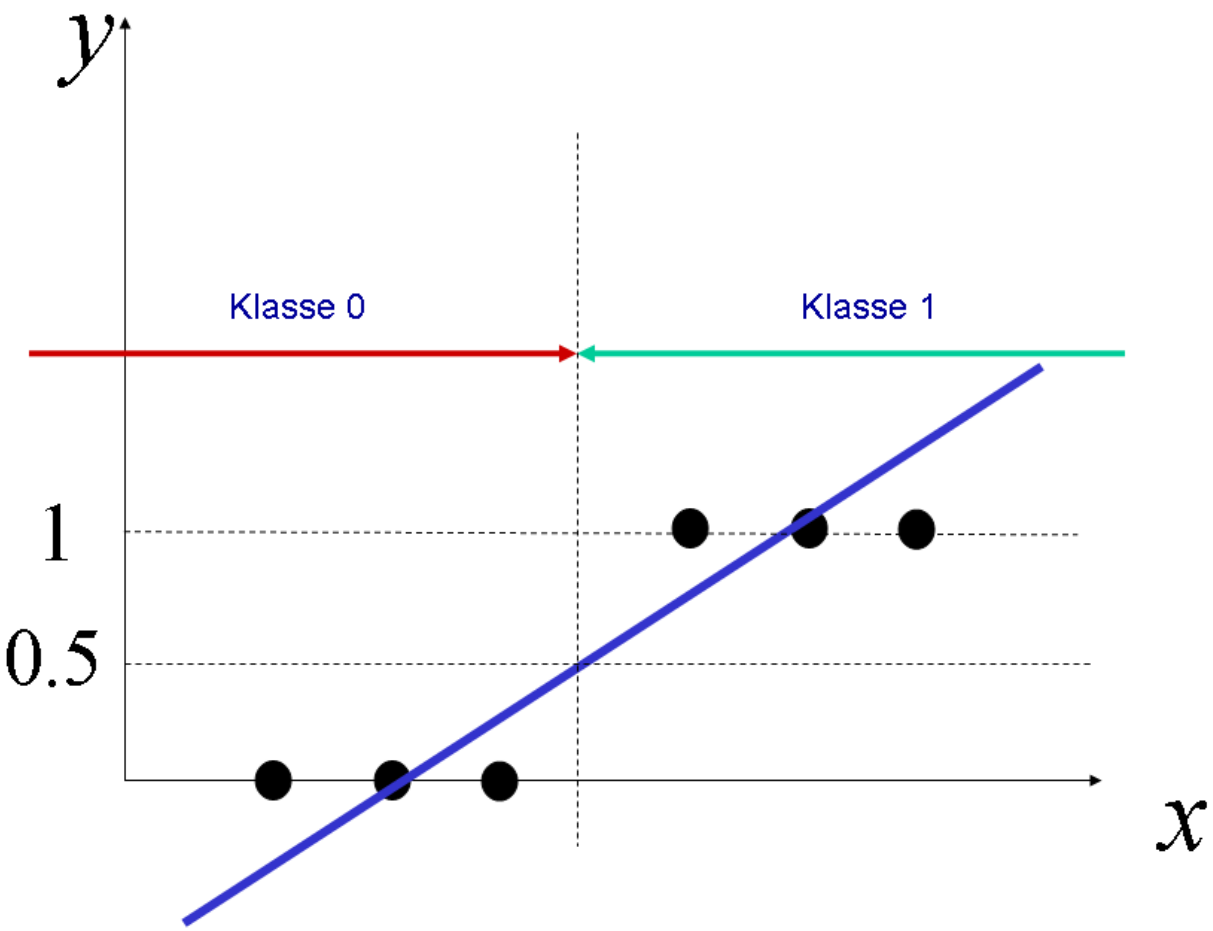
$$= \mathbf{x}_i^T \mathbf{w}$$

- Wir definieren als Zielgröße $y_i = 1$ falls Muster $\mathbf{x}_i$ zu Klasse 1 gehört und $y_i = 0$ falls Muster $\mathbf{x}_i$ zu Klasse 0 gehört
- Wir berechnen Gewichte $\mathbf{w}_{LS} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$ als LS-Lösung, genau wie in der linearen Regression
- Für einen neues Muster $\mathbf{z}$ berechnen wir $f(\mathbf{z}) = \mathbf{z}^T\mathbf{w}_{LS}$ und ordnen das Muster Klasse 1 zu falls $f(\mathbf{z}) > 1/2$; ansonsten ordnen wir das Muster Klasse 0 zu

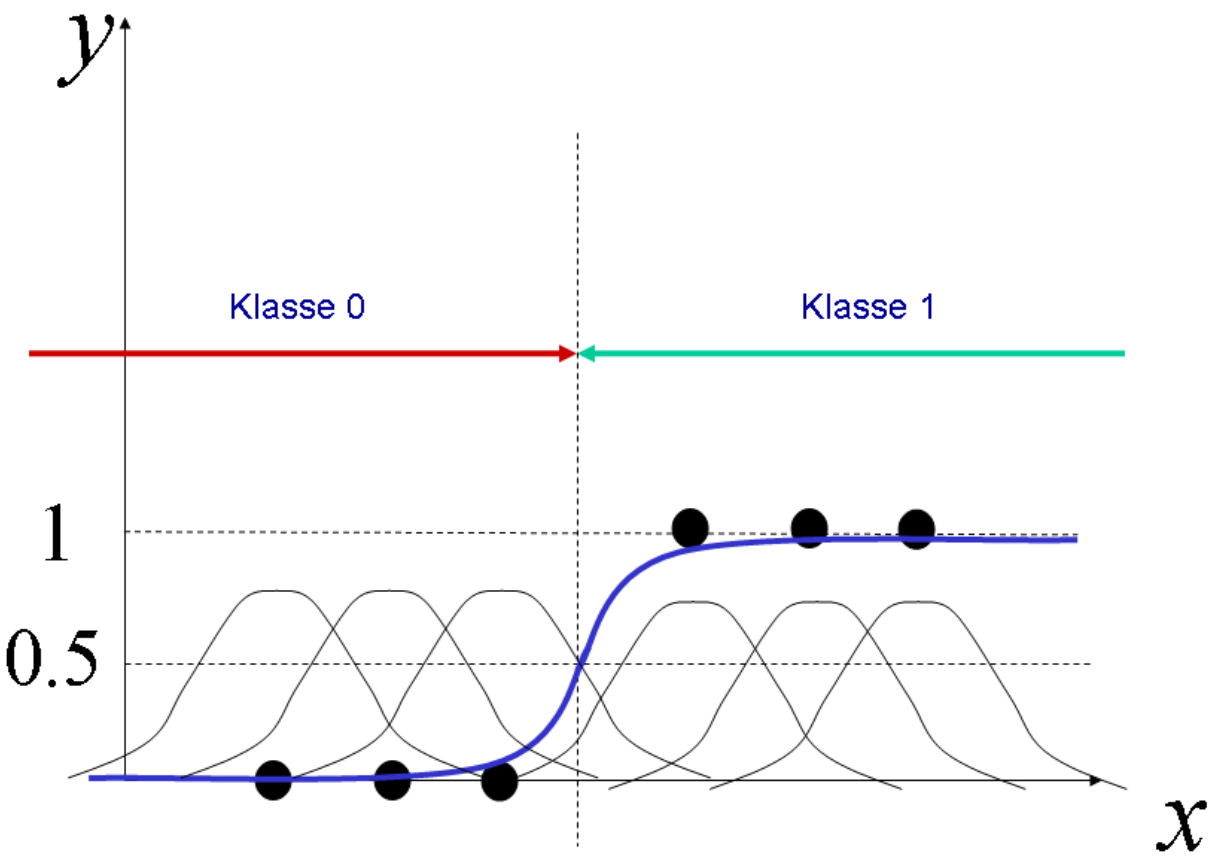
Bias

- Asymptotisch konvergiert eine LS-Lösung zur Klassenwahrscheinlichkeit $P(c = 1|\mathbf{x})$; allerdings ist eine lineare Funktion in der Regel nicht fähig, die Klassenwahrscheinlichkeit zu repräsentieren (Bias!); dennoch kann die resultierende Klassifikationsentscheidung durchaus sinnvoll sein
- Man kann gute Klassifikationsentscheidungen in hohen Dimensionen erwarten und im Zusammenhang mit Basisfunktionen und Kernfunktionen; in manchen Fällen erreicht man dann Konsistenz

Klassifikation durch Regression mit linearen Funktionen



Klassifikation durch Regression mit radialen Basisfunktionen



Performanz

- Obwohl der Ansatz für lineare Klassifikation etwas simplistisch erscheint, ist die Performanz in Kombination mit Basisfunktionen oder Kernfunktionen durchaus sehr gut! Dort wegen der großen Einfachheit und guten Performanz sehr populär

Vergleich: Musterbasiertes Lernen

- Perzeptron

$$w_j \longleftarrow w_j + \eta \left(y(t) - \text{sign} \left(x(t)^T w \right) \right) x_j(t)$$

$y(t) \in \{-1, 1\}$. Beachte, dass die Klammer Null ist, wenn richtig klassifiziert. Ansonsten ist der Term entweder gleich 2 oder gleich -2.

- Logistic regression

$$w_j \longleftarrow w_j + \eta \left(y(t) - \text{sig} \left(x(t)^T w \right) \right) x_j(t)$$

Die "natürliche" kontinuierliche Verallgemeinerung; $y(t) \in \{-1, 1\}$

- Neuronale Netze

$$w_j \longleftarrow w_j + \eta \left(y(t) - \text{sig} \left(x(t)^T w \right) \right) \text{ sig}' \left(x(t)^T w \right) \ x_j(t)$$

Wird ein Muster mit hoher Sicherheit falsch klassifiziert, ist der Gradient nahezu Null!

- Regression (ADALINE)

$$w_j \longleftarrow w_j + \eta \left(y(t) - \left(x(t)^T w \right) \right) x_j(t)$$