Das Perzeptron

Volker Tresp

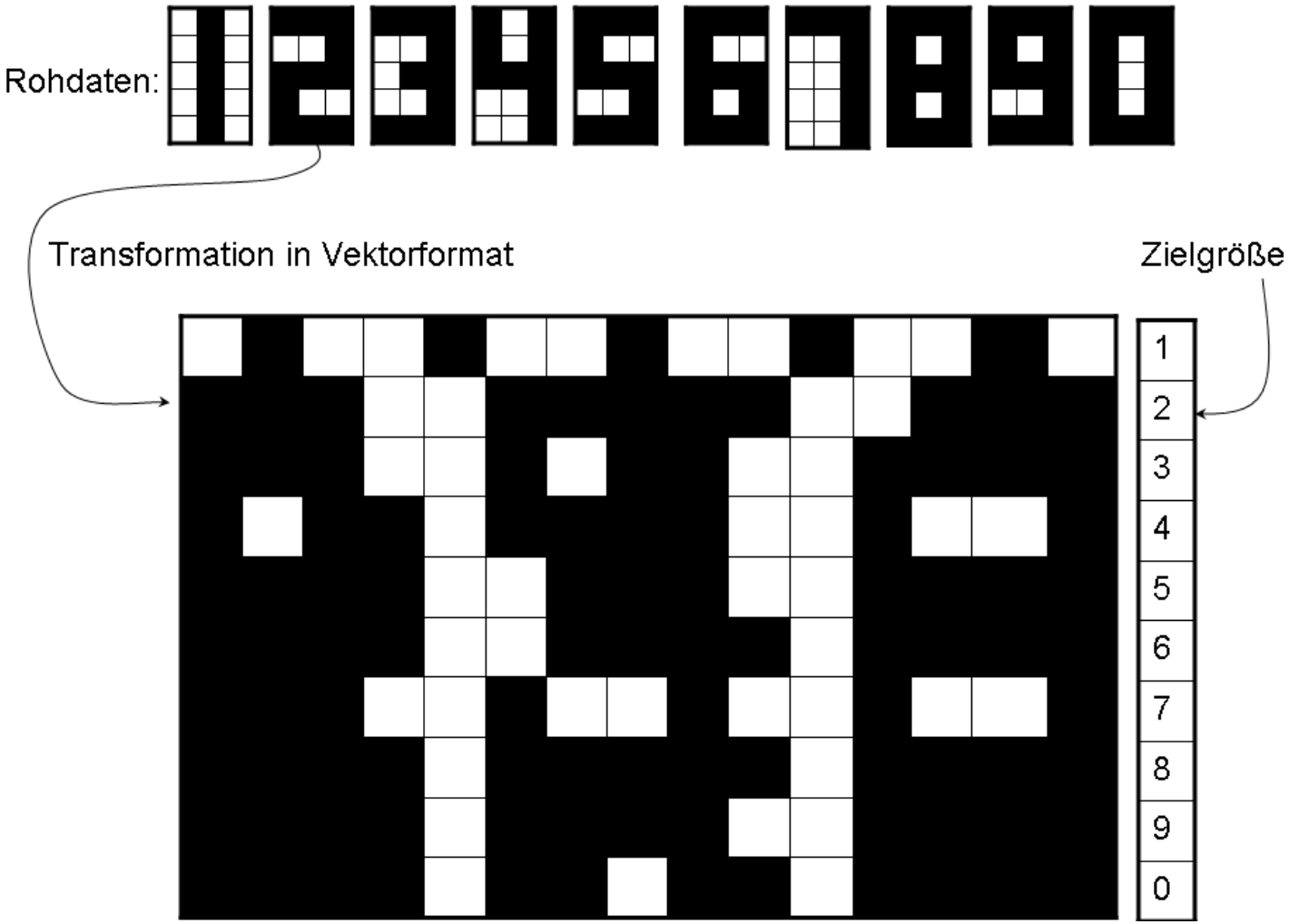
Einführung

- Das Perzeptron war eines der ersten ernstzunehmenden Lernmaschinen
- Die wichtigsten Elemente
  - Sammlung und Vorverarbeitung der Trainingsdaten
  - Wahl einer Klasse von Lernmodellen: oft definiert durch die freien Parameter in einer Lernstruktur
  - Wahl einer Kostenfunktion
  - Ableitung einer Lernregel zur Selektion des optimalen Lernmodells: oft, Lernen des optimalen Parametervektors

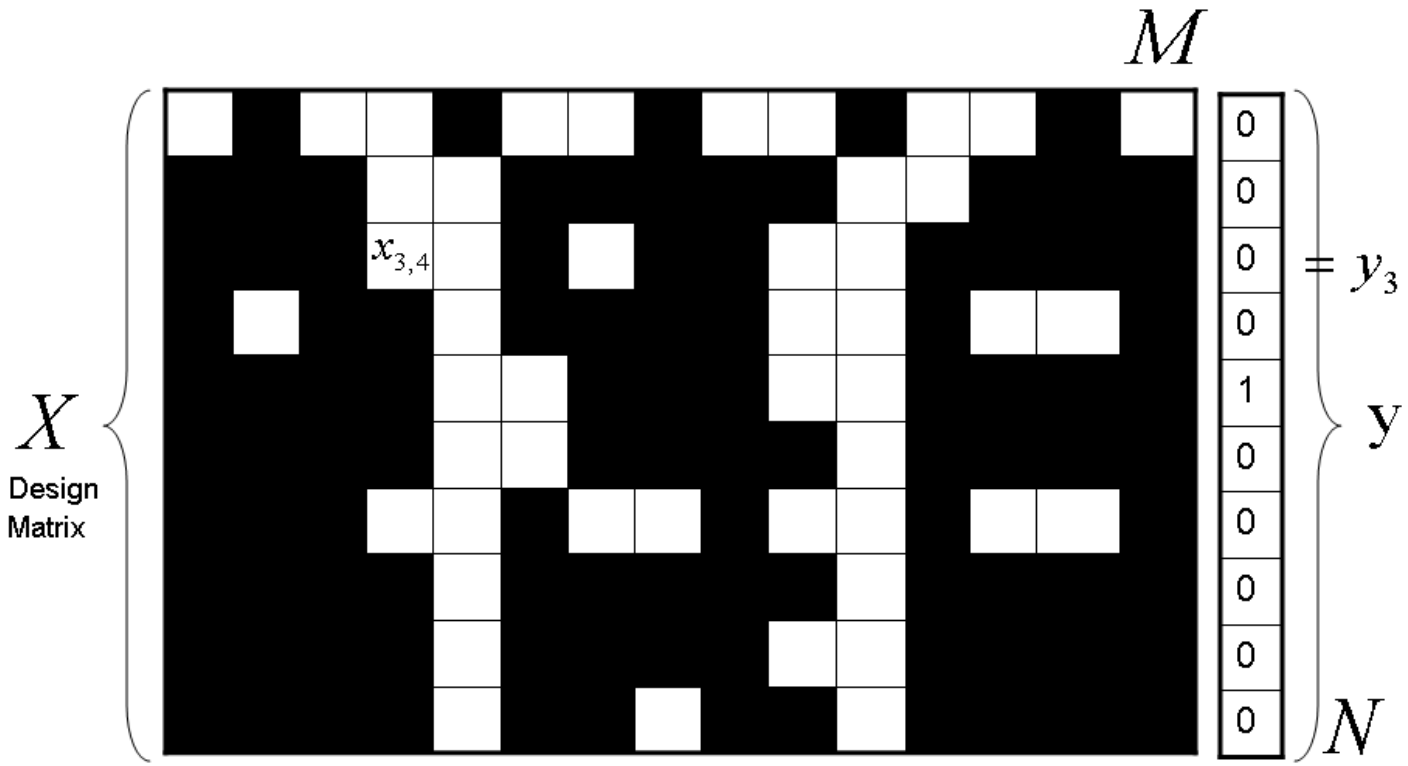
Ein prototypisches Lernproblem

- Klassifikation von gedruckten/geschriebenen Ziffern
- Anwendung: Lesen von Postleitzahlen
- Allgemeiner: OCR (*optical character recognition*)

Transformation der Rohdaten (2-D) in Mustervektoren (1-D) einer Lernmatrix



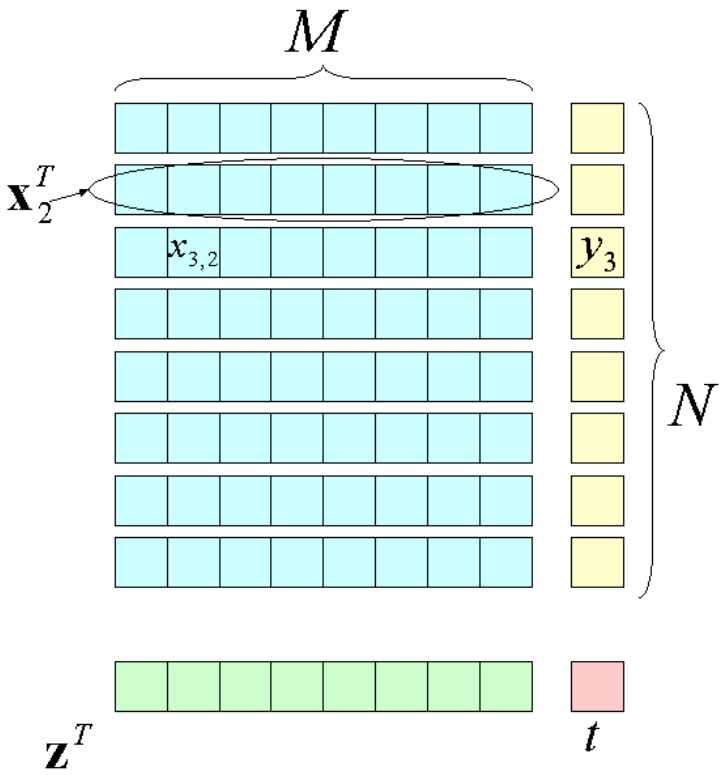
Binäres Klassifikationsproblem



Zielgröße - 1: das Muster gehört zur Klasse 5

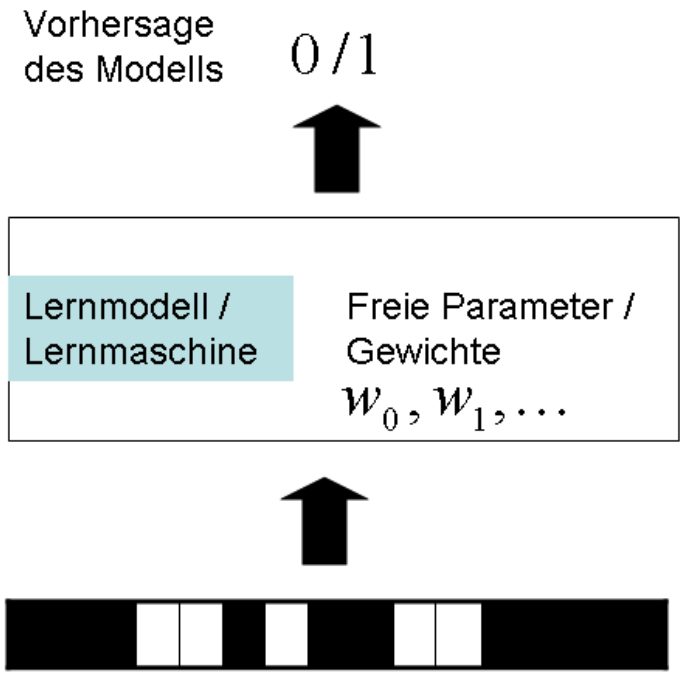
Zielgröße - 0: das Muster gehört nicht zur Klasse 5

Die Datenmatrix für überwachtes Lernen



M	Anzahl der Eingangsvariablen
N	Anzahl der (Trainings-) Datenpunkte
$\mathbf{x}_i$	$= (x_{i,0}, \ldots, x_{i,M-1})^T$ i-ter Eingangsvektor
$x_{i,j}$	j-te Komponente von $\mathbf{x}_i$
$\mathbf{X}$	$= (x_1, \ldots, x_N)^T$ Designmatrix
y_i	i-te Zielgröße zu $\mathbf{x}_i$
$\mathbf{y}$	$= (y_1, \ldots, y_N)^T$ Vektor der Zielgrößen
$\widehat{y}_i$	Vorhersage zu $\mathbf{x}_i$
$\mathbf{d}_i$	$= (x_{i,0}, \ldots, x_{i,M-1}, y_i)^T$ i-tes Muster
D	$= \{\mathbf{d}_1, \ldots, \mathbf{d}_N\}$ (Trainings-) Datensatz
$\mathbf{z}$	Testeingangsvektor
t	Unbekannte Testzielgröße zu $\mathbf{z}$

Lernmodell



Die freien Parameter / Gewichte $w_0, w_1, \ldots$

werden so eingestellt, dass das Modell gute Ergebnisse auf den Trainingsdaten

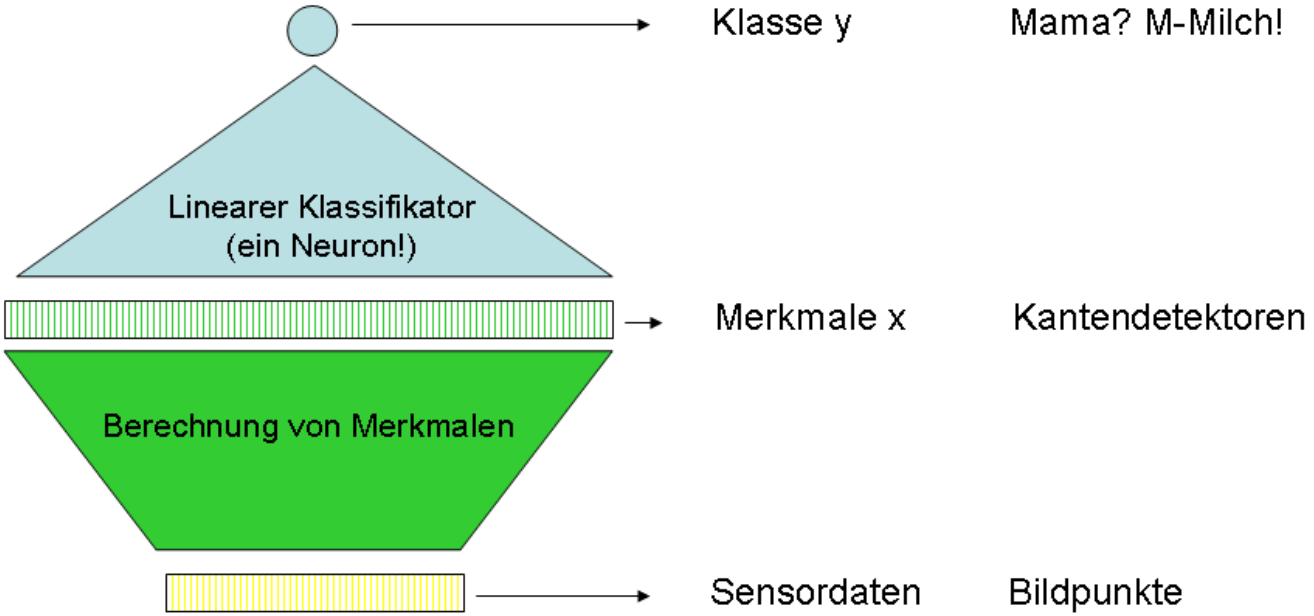
$$d_i = (x_i, y_i)$$

liefert

Eigentliches Ziel ist die gute Performanz auf neuen Testdaten

$$(z, t_i)$$

Ein biologisch motiviertes Lernmodell



Input-Output Modelle

- Ein biologisches System muss basierende auf Sensordaten eine Entscheidung treffen
- Ein OCR System klassifiziert eine handgeschriebene Ziffer
- Ein Prognosesystem sagt den morgigen Energieverbrauch voraus

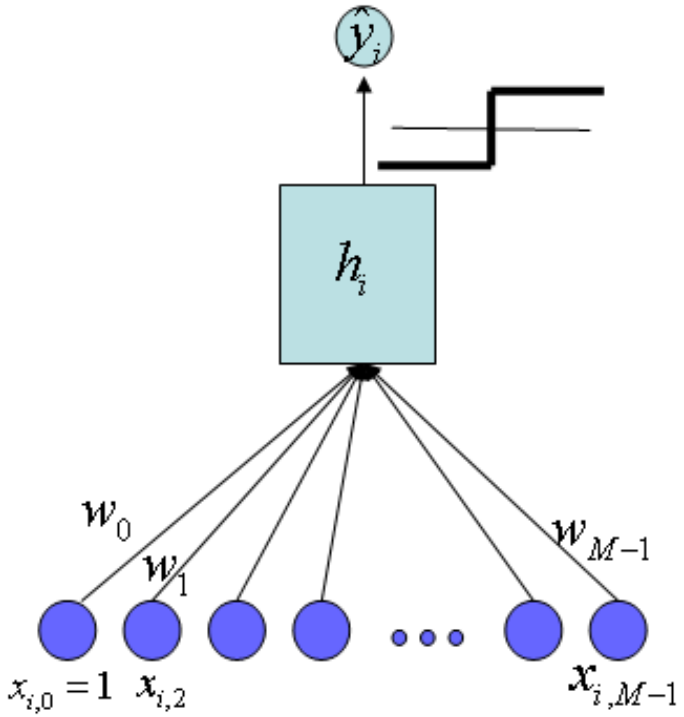
Überwachtes Lernen

- Im überwachten Lernen geht man davon aus, dass in den Trainingsdaten sowohl die Eingangsmuster als auch die Zielgrößen bekannt sind
- Zum Beispiel kann das Eingangsmuster Attribute eines Objektes repräsentieren und die Zielgröße steht für die Klassenzugehörigkeit des Objektes
- Ziel ist die korrekte Klassifikation neuer Muster
- Der vielleicht einfachste aber eigentlich auch schon erstaunlich mächtige Klassifikator ist ein linearer Klassifikator
- Ein linearer Klassifikator eignet sich im Besonderen, wenn die Eingangsdimension M hoch ist; wenn dies nicht der Fall ist, kann man die Daten in einen hoch dimensionalen Raum transformieren, so dass auch dann ein linearer Klassifikator wieder anwendbar wird (Teil der Vorlesung)
- Ein linearer Klassifikator kann durch ein Perzeptron realisiert werden, also einem einzigen formalisierten Neuron!

Überwachtes Lernen und Lernen von Entscheidungen

- Man kann mit einigem Recht argumentieren, dass Maschinelles Lernen nur interessant ist, inwieweit es zu Entscheidungen führt
- Allerdings lassen sich viele Entscheidungen auf ein überwachtes Lernproblem (Vorhersage/Klassifikationsproblem) zurückführen: Wenn ich die Postleitzahl automatisch lese und erkenne, dann weiß ich wohin ich den Brief schicken muss
- Dies bedeutet: Entscheidung und Aktion kann man häufig auf Klassifikation reduzieren

Die Lernmaschine: Das Perzeptron



- Zunächst wird die Aktivierungsfunktion des Perzeptrons als gewichtete Summe der Eingangsgrößen x_i berechnet zu

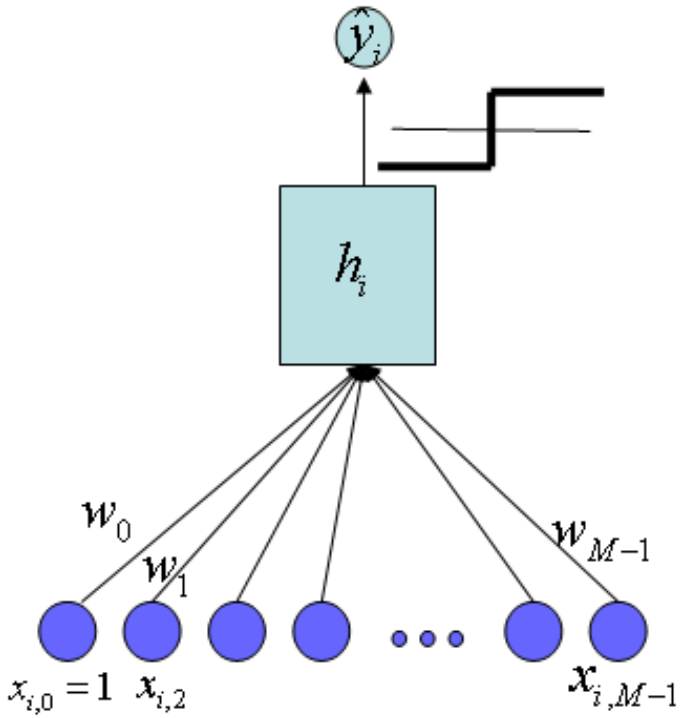
$$h_i = \sum_{j=0}^{M-1} w_{i,j} x_{i,j}$$

  (beachte: $x_{i,0} = 1$ ist ein konstanter Eingang, so dass w_0 dem Bias entspricht)
- Die binäre Klassifikation $y_i \in \{1, -1\}$ wird berechnet zu

$$\hat{y}_i = \text{sign}(h_i)$$

- Die lineare Klassifikationsgrenze (*separating hyperplane*) ist definiert durch $h_i = 0$

Gewichtete Abstimmung im Perzeptron



- Das Perzeptron wird oft als graphisches Modell dargestellt mit Eingangsknoten und einem Ausgangsknoten
- Der Bias w_0 stellt ein, wie sich das Perzeptron ohne Eingang verhält
- Ist $x_{i,j} = 1$ stimmt der j-te Eingang im i-ten Muster mit dem Gewicht $|w_j|$ für die binäre Klasse $\mathsf{sign}(w_j)$
- *Das heißt, man kann die Antwort eines Perzeptrons als gewichtete Abstimmung der Eingangsgrößen betrachten*

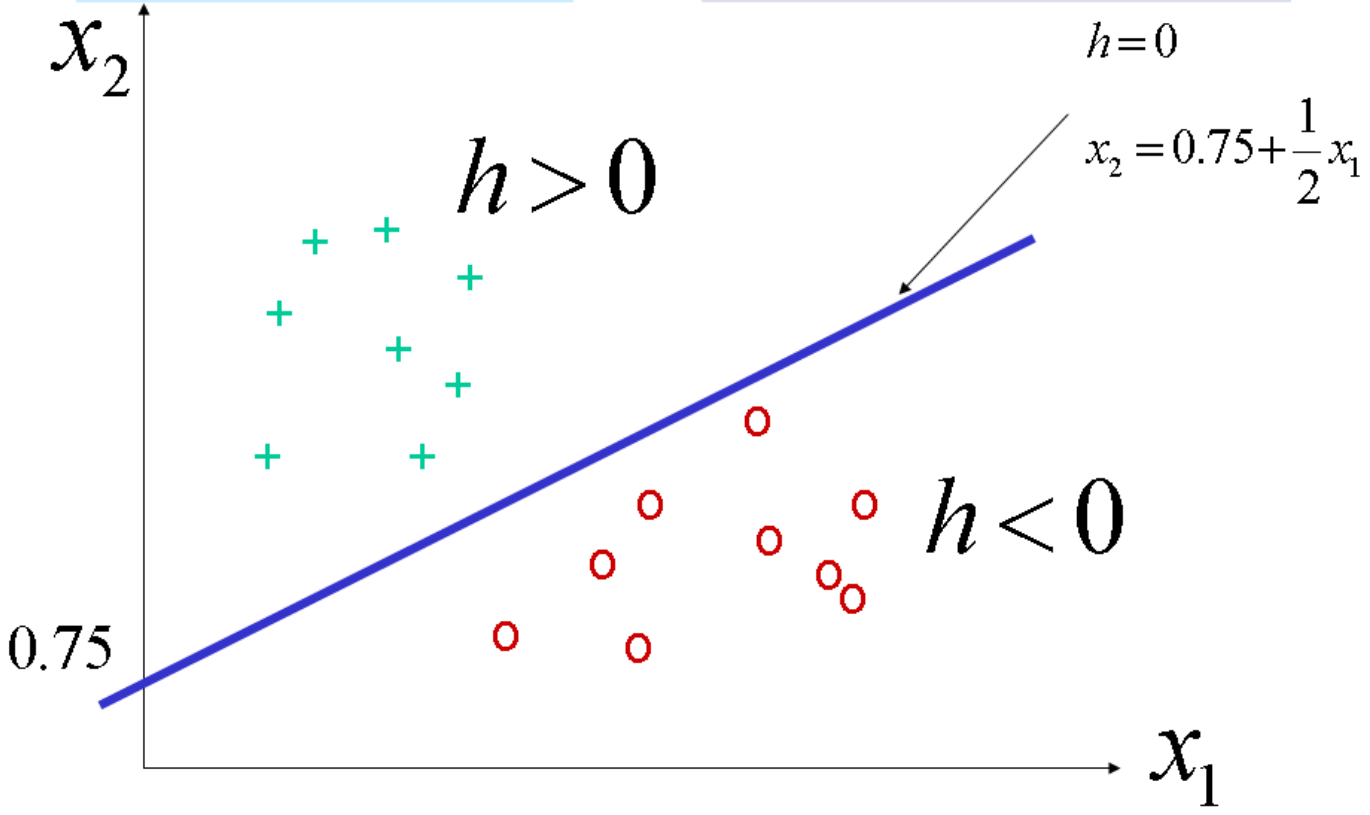
2-D Darstellung der Perzeptron-Entscheidungsebene

- Die Klassengrenzen eines Perzeptrons werden gerne in zwei Dimensionen graphisch dargestellt ($M = 3$) (nächste Folie)
- Der Vorteil ist, dass man eine gewisse Intuition gewinnen kann
- Der Nachteil ist, dass die Intuition auch irreführend ist, da das Perzeptron typischerweise in hoch dimensionalen Problemen ($M >> 1$) Anwendung findet

Zwei linear-separierbare Klassen

$$h = -0.75 - \frac{1}{2}x_1 + x_2$$

$$h = 0 \Rightarrow x_2 = 0.75 + \frac{1}{2}x_1$$



Herleitung der Perzeptron-Lernregel

- Nachdem wir das Modell (Perzeptron) ausgewählt haben, benötigen wir einen Lernalgorithmus (Lernregel), die dazu führt, dass die Parameter/Gewichte sinnvoll eingestellt werden
- Dazu definiert man eine geeignete Kostenfunktion, die von den Trainingsdaten und den Parametern abhängt. Da die Trainingsdaten gegeben sind, ist nur die Abhängigkeit von den Parametern entscheidend
- Im Lernvorgang (Training) sucht man nach den Parametern, die die Kostenfunktion minimieren
- Eine sinnvolle Kostenfunktion ist die Summe der quadratischen Abstände zwischen den Zielgrößen und der Vorhersage des Perzeptrons

$$\text{cost}_Q(\mathbf{w}) = \sum_{i=1}^{N} \left(y_i - \text{sign} \left(\sum_{j=0}^{M-1} w_j x_{i,j} \right) \right)^2$$

- Beim Perzeptron wurde eine andere Kostenfunktion gewählt

Die Perzeptron-Kostenfunktion

- Die (synaptischen) Gewichte w_i sollten so eingestellt werden, dass das Perzeptron die N Trainingsdaten $\{y_1, \ldots, y_N\}$ richtig klassifiziert
- Dazu definieren wir als Kostenfunktion des Perzeptrons

$$\text{cost} = -\sum_{i \in \mathcal{M}} y_i h_i = -\sum_{i \in \mathcal{M}} \left(y_i \sum_{j=0}^{M-1} w_j x_{i,j} \right)$$

  wobei $\mathcal{M} \subseteq \{1, \ldots, N\}$ die Indexmenge der (gegenwärtig) falsch klassifizierten Muster ist und $x_{i,j}$ der Wert des j-ten Eingangs im i-ten Musters ist
- Offensichtlich ist cost $= 0$ nur dann, wenn alle Muster richtig klassifiziert werden (dann ist $\mathcal{M} \subseteq \emptyset$), ansonsten cost > 0, da in der Indexmenge y_i und h_i unterschiedliche Vorzeichen haben

Gradientenabstieg

- Man initialisiert die Parameter (typischerweise durch kleine Zufallszahlen)
- In jedem Lernschritt verändert man die Parameter, so dass die Kostenfunktion verringert wird
- Gradientenabstieg: Man verändert den Parametervektor in Richtung des negativen Gradienten
- Die Ableitung der Kostenfunktion nach den Gewichten ist (Beispiel w_j)

$$\frac{\partial \text{cost}}{\partial w_j} = -\sum_{i \in \mathcal{M}} y_i x_{i,j}$$

- Eine sinnvolle Lernregel ist somit

$$w_j \longleftarrow w_j + \eta \sum_{i \in \mathcal{M}} y_i \mathbf{x}_{i,j}$$

Die Perzeptron-Lernregel

- Im tatsächlichen Algorithmus werden in zufälliger Reihenfolge dem Perzeptron falsch klassifizierte Muster angeboten (stochastischer Gradientenabstieg). Einmal ist dies biologisch plausibler, und zweitens ist die Konvergenz schneller. Seien $\mathbf{x}_t$ und y_t die angebotenen Muster im t-ten Iterationsschritt. Dann wird adaptiert für $t = 1, 2, \ldots$

$$w_j \longleftarrow w_j + \eta y_t x_{t,j} \quad j = 1, \ldots, M$$

- Eine Gewicht wächst an, wenn (postsynaptisches) $y(t)$ und (präsynaptisches) $x_j(t)$ gleiches Vorzeichen besitzen; bei unterschiedlichen Vorzeichen verringert sich das Gewicht (vergleiche: **Hebb'sches Lernen**)

- $\eta > 0$ ist die Lernrate, typischerweise $0 < \eta < 0.1$

Kommentare

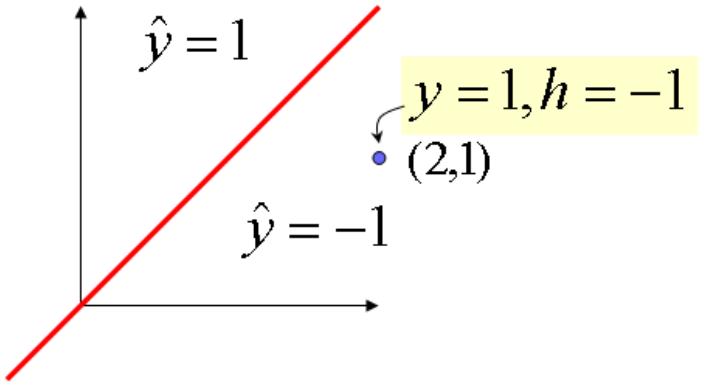
- Konvergenzbeweis: Bei genügend kleiner Lernrate η und linear trennbaren Problemen konvergiert der Algorithmus nach endlich vielen Schritten
- Alternative Schreibweise der Kostenfunktion

$$\text{cost} = \sum_{i=1}^{N} \left| -y_i \left(\sum_{j=0}^{M-1} w_j x_{i,j} \right) \right|_+$$

wobei $|arg|_+ = \max(arg, 0)$.

Beispiel: Perzeptron Lernen, $\eta = 0.1$

$$h = 0 \times 1 - 1 \times x_1 + 1 \times x_2$$



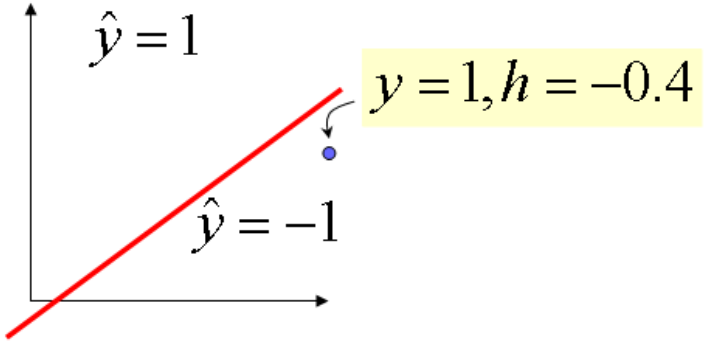
Trennebene vor Adaption: $x_2 = x_1$

Adaption:

$$w_0 \leftarrow 0 + 0.1 \times (1 \times 1) = 0.1$$
$$w_1 \leftarrow -1 + 0.1 \times (1 \times 2) = -0.8$$
$$w_2 \leftarrow 1 + 0.1 \times (1 \times 1) = 1.1$$

Entsprechend Hebb: alle Parameter wachsen

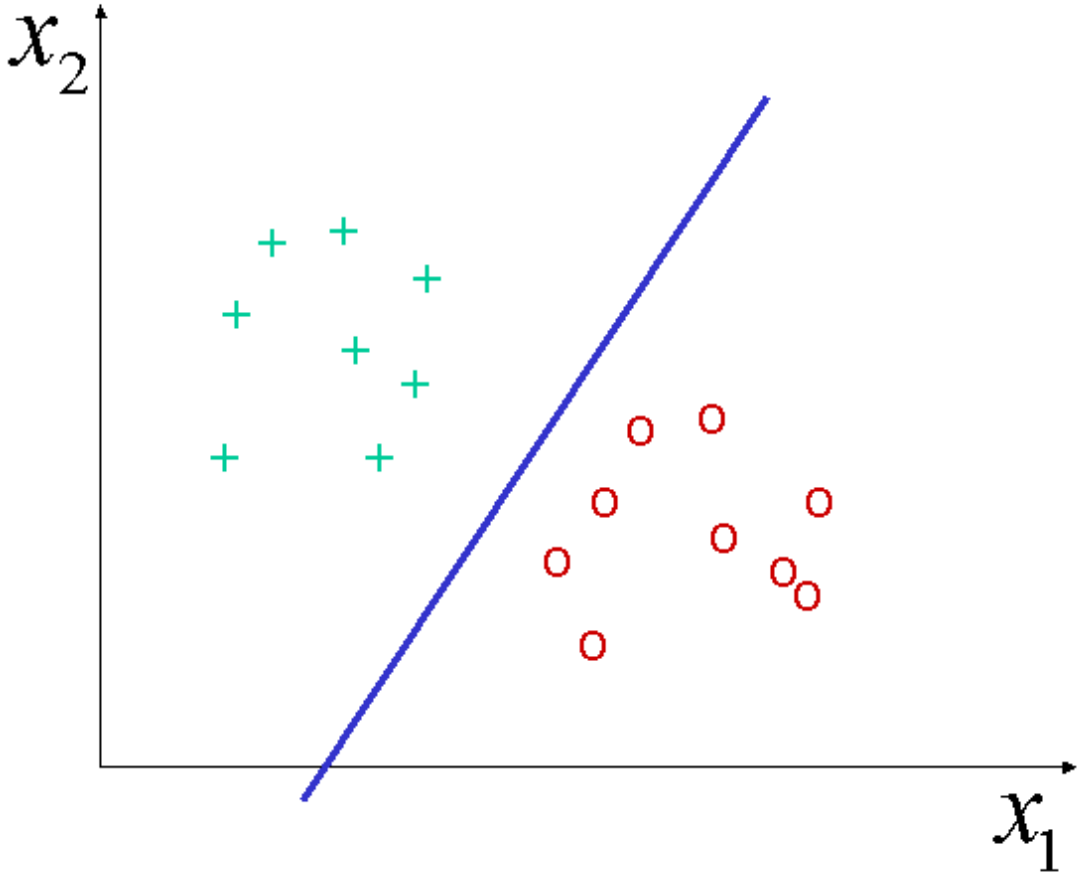
$$h = 0.1 \times 1 - 0.8 \times x_1 + 1.1 \times x_2$$



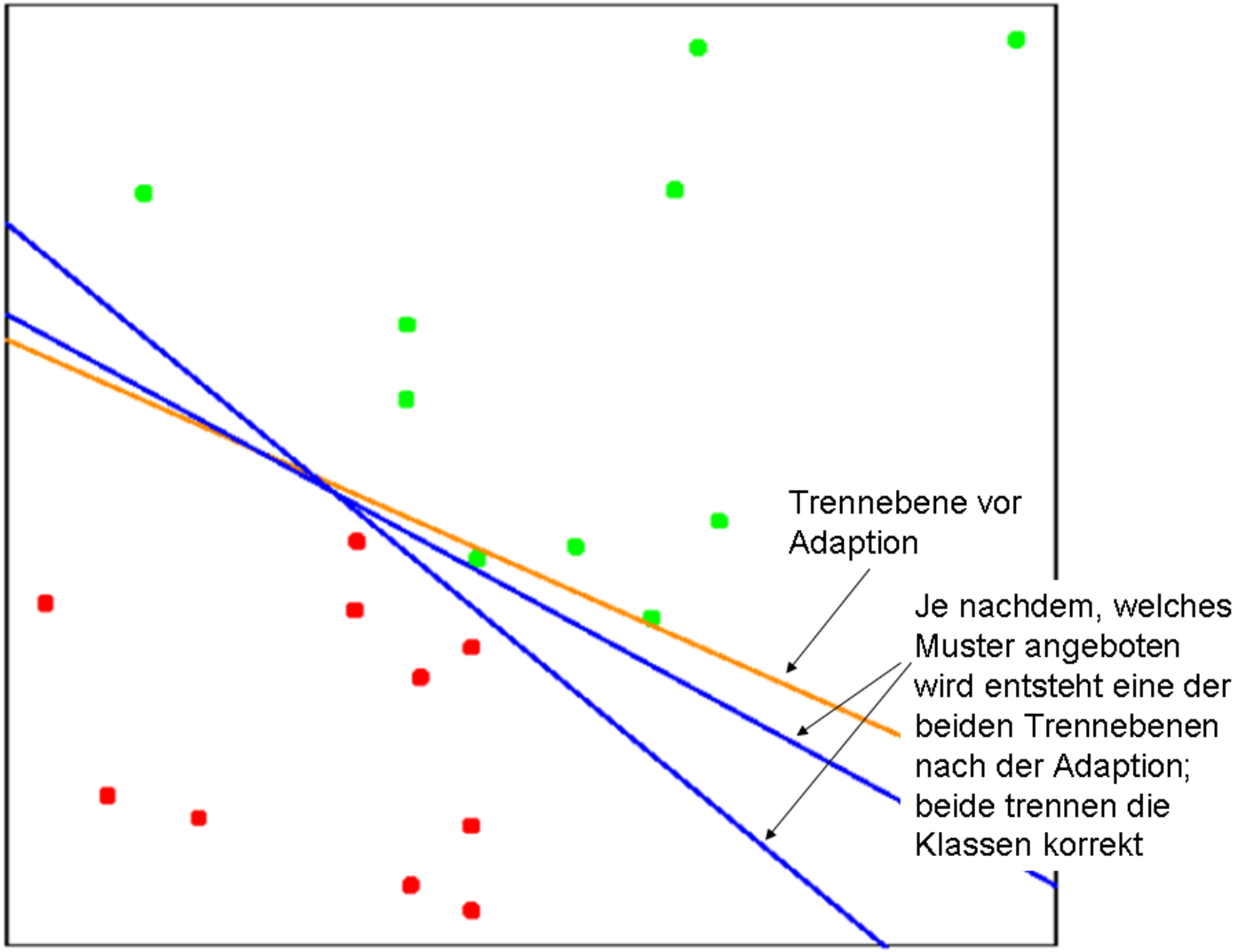
Trennebene nach Adaption:

$$x_2 = -0.09 + 0.72 \times x_1$$

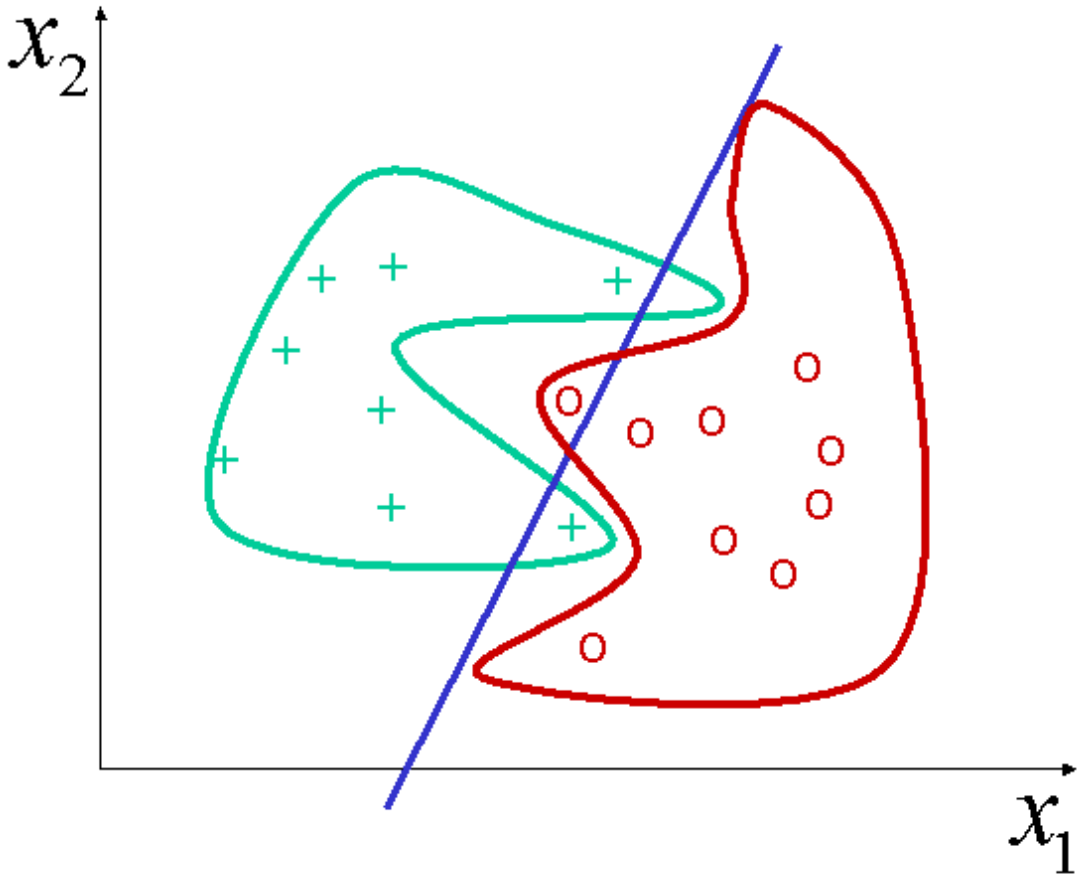
Zwei linear-separierbare Klassen



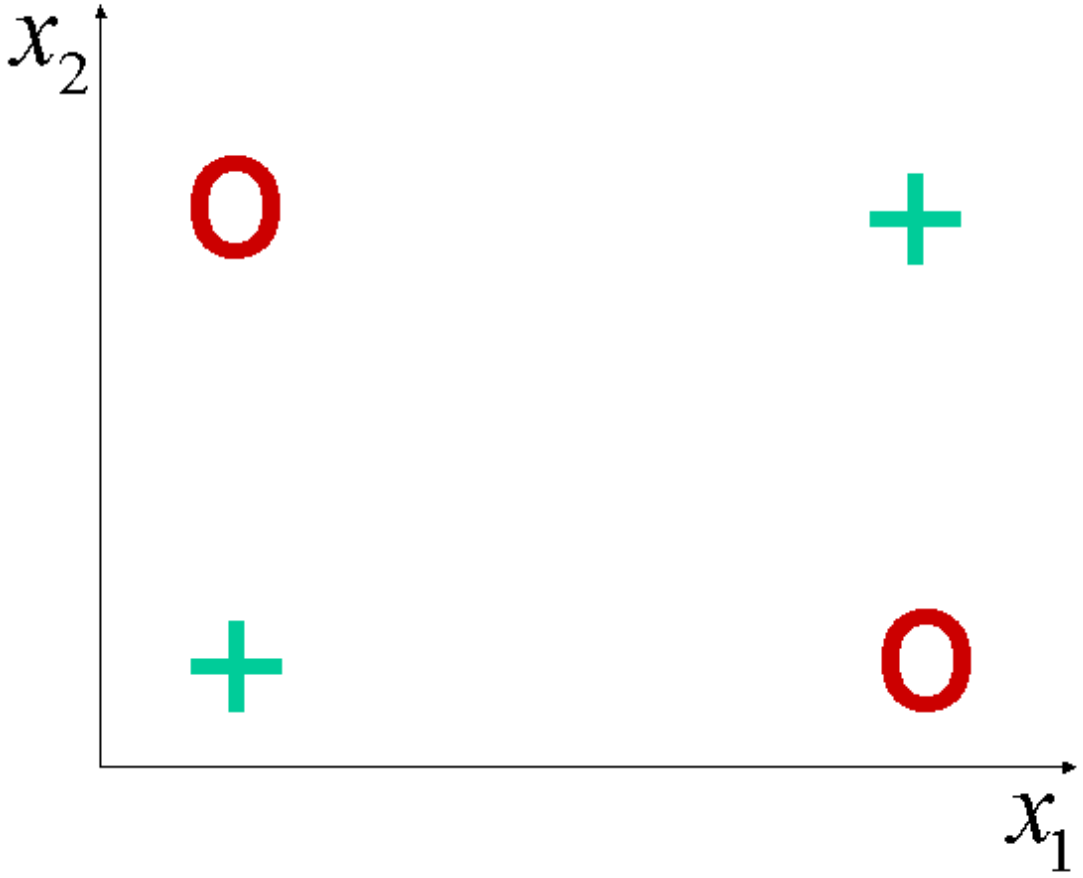
Konvergenz des Perzeptrons und Mehrdeutigkeit der Lösung



Zwei Klassen, die nicht linear separierbar sind



Das klassische Beispiel nicht-separierbarer Klassen: XOR



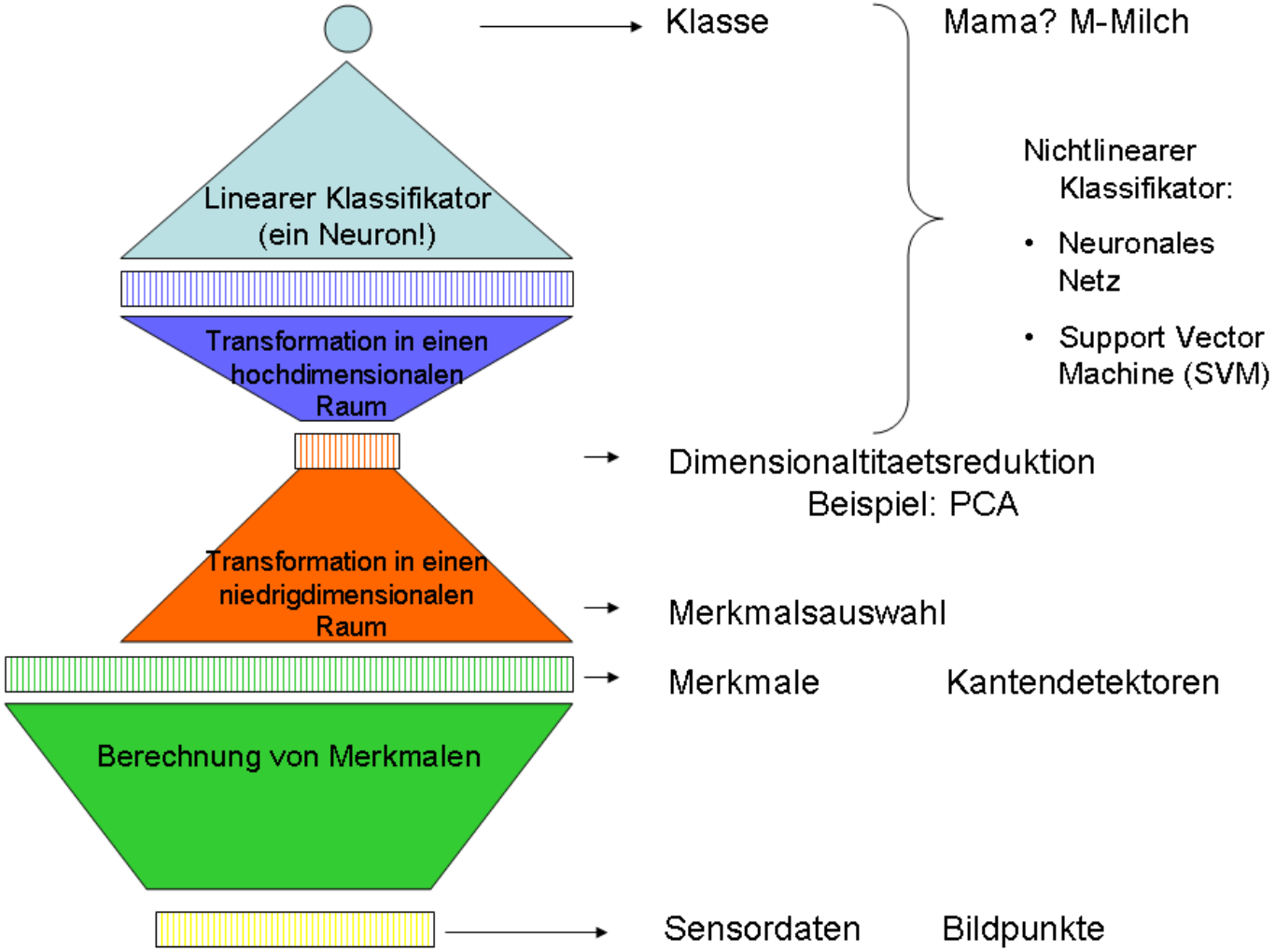
Kommentare zum Perzeptron

- Konvergenz kann sehr schnell sein
- Lineare Klassifikatoren sind auch heute von zentraler Bedeutung: mit $M \to \infty$ werden alle Probleme linear separierbar!
- In manchen Fällen sind die Rohdaten schon hoch dimensional: Texte haben Merkmalsvektoren > 10000 (Anzahl der möglichen Worte)
- In anderen Fällen transformiert man zunächst die Eingangsdaten in einen hoch-dimensionalen (∞-dimensional) Raum, und wendet dann einen linearen Klassifikator an (Kernel-trick in der *Support Vector Machine*, Neuronale Netze)
- Betrachtet man die Mächtigkeit eines einzigen Neurons, wie viel Rechenleistung kann man von 100 Milliarden Neuronen erwarten?
- Gibt es solche "Großmutterzellen" (*grandmother cells*?) oder Großmutterareale im Gehirn?

Kommentare zum Perzeptron (2)

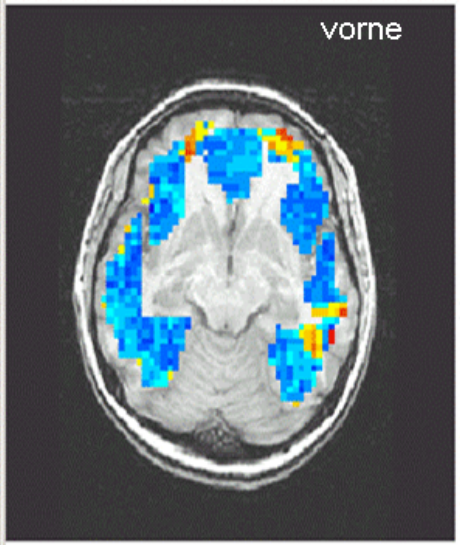
- Die Perzeptron Lernregel wird heutzutage nicht mehr viel benutzt
  - keine Konvergenz bei nicht trennbaren Klassen
  - Trennebene nicht eindeutig
- Alternative lineare Klassifikatoren:
  - linear Support Vector Maschine
  - Fisher linear Discriminant
  - Logistic Regression
- Die folgende Graphik zeigt ein mächtigeres Lernmodell; gleichzeitig das Programm eines Großteils der Vorlesung!

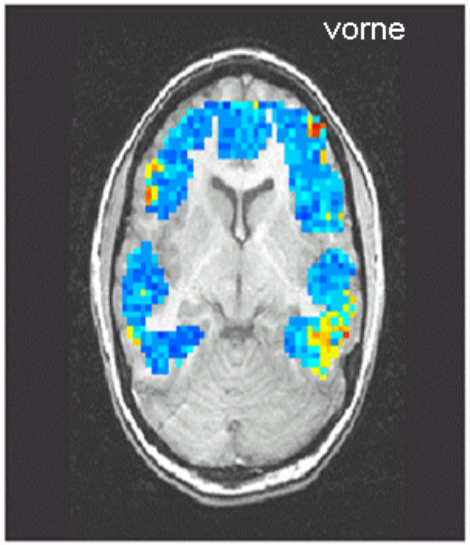
Ein mächtigeres Lernmodell

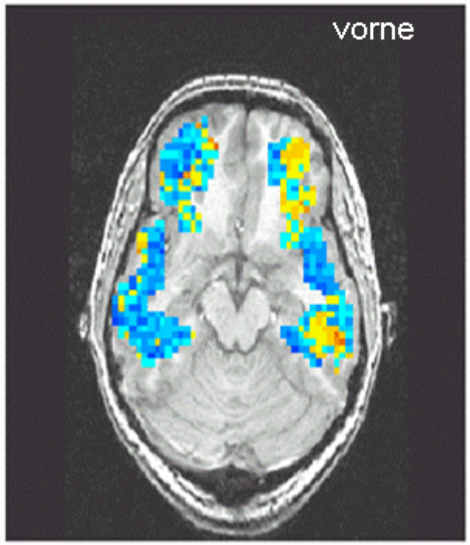


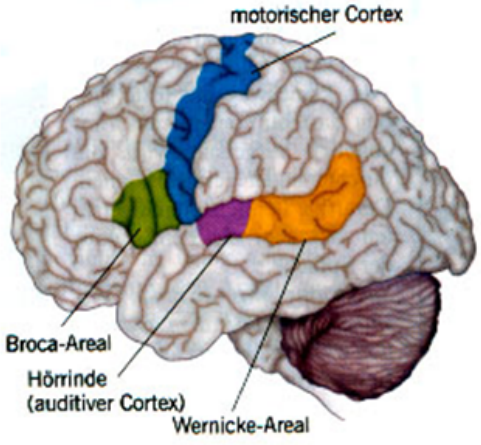
Analyse von fMRI Gehirn-Schnittbildern (Tom Mitchel et al., CMU)

- Ziel: Anhand der Schnittbilder zu klassifizieren, ob jemand an Werkzeuge, Gebäude, Essen, oder eine Reihe anderer semantischer Kategorien denkt
- Der trainierte lineare Klassifikator ist zu 90% genau und kann zum Beispiel trennen, ob jemand Worte über Werkzeuge oder über Gebäude liest
- In der Abbildung sieht man farblich markiert die wichtigsten Voxels, die für die Klassifikation ausschlaggebend sind für 3 Probanden: die wichtigen Voxels sind bei allen drei in ähnlichen Regionen


vorne


vorne


vorne


motorischer Cortex
Broca-Areal
Hörrinde
(auditiver Cortex)
Wernicke-Areal

Paradigma der Mustererkennung

- von Neumann: *... the brain uses a peculiar statistical language unlike that employed in the operation of man-made computers...*
- Eine Klassifikationsentscheidung wird parallel anhand des gesamten Musters getroffen und nicht als logische Verknüpfung einer kleinen Anzahl von Größen oder als mehrschichtiges logisches Programm
- Die linear gewichtete Summe entspricht mehr einem Abstimmen als einer logischen Entscheidungsfunktion; jeder Eingang hat entweder immer einen positiven oder immer einen negativen (gewichteten) Einfluss
- Robustheit: in hohen Dimensionen kommt es nicht auf einen einzigen Eingangswert an; jeder Eingang macht seinen "kleinen" Beitrag

Nachbemerkung

Warum Mustererkennung?

- Eines der großen Rätsel des Maschinellen Lernens ist der mangelnde Erfolg im Versuch, einfache deterministische logische Regeln zu erlernen
- Probleme: Die erlernten Regeln sind oft trivial, bekannt, extrem komplex oder unmöglich zu interpretieren; die Performanz eines Klassifikators basierend auf einfachen deterministischen logischen Regeln ist in der Regel nicht sehr gut!
- Dies steht scheinbar im Gegensatz zum eigenen Eindruck, dass die Welt wohldefinierten einfachen Regeln gehorcht
- Ebenso folgen (fast) alle Computer Programme, Maschinen (Autos), deterministischen Regeln

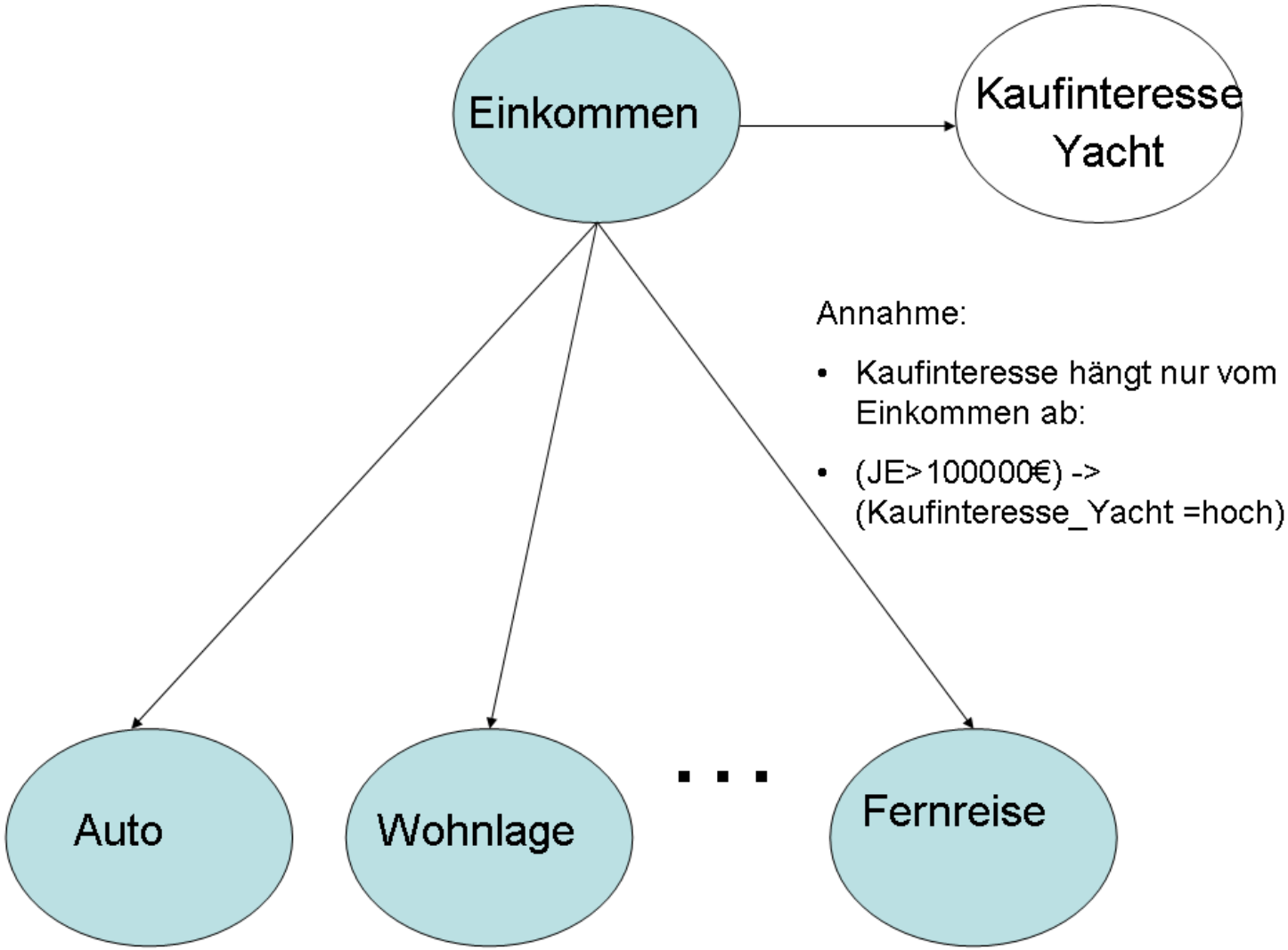
Beispiel: Alle Vögel fliegen hoch

- Definiere fliegen: aus eigener Kraft, mindestens 20 m weit, mindestens 1 m hoch, mindestens einmal am Tag seines erwachsenen Lebens, ...
- Ein Objekt ist als Vogel klassifiziert; der Vogel kann fliegen,
  - Außer er ist ein Pinguin, oder ein, ...
  - Außer er ist schwer verletzt oder tot
  - Außer er ist zu alt
  - Außer er ist gestutzt worden
  - Außer er hat eine Reihe von Krankheiten
  - Außer er lebt ausschließlich in einem Stall
  - Außer mat hat ihm ein schweres Gewicht an seine Körperteile befestigt
  - Außer er lebt als Papagei in einem zu kleinen Käfig, ...

Mustererkennung

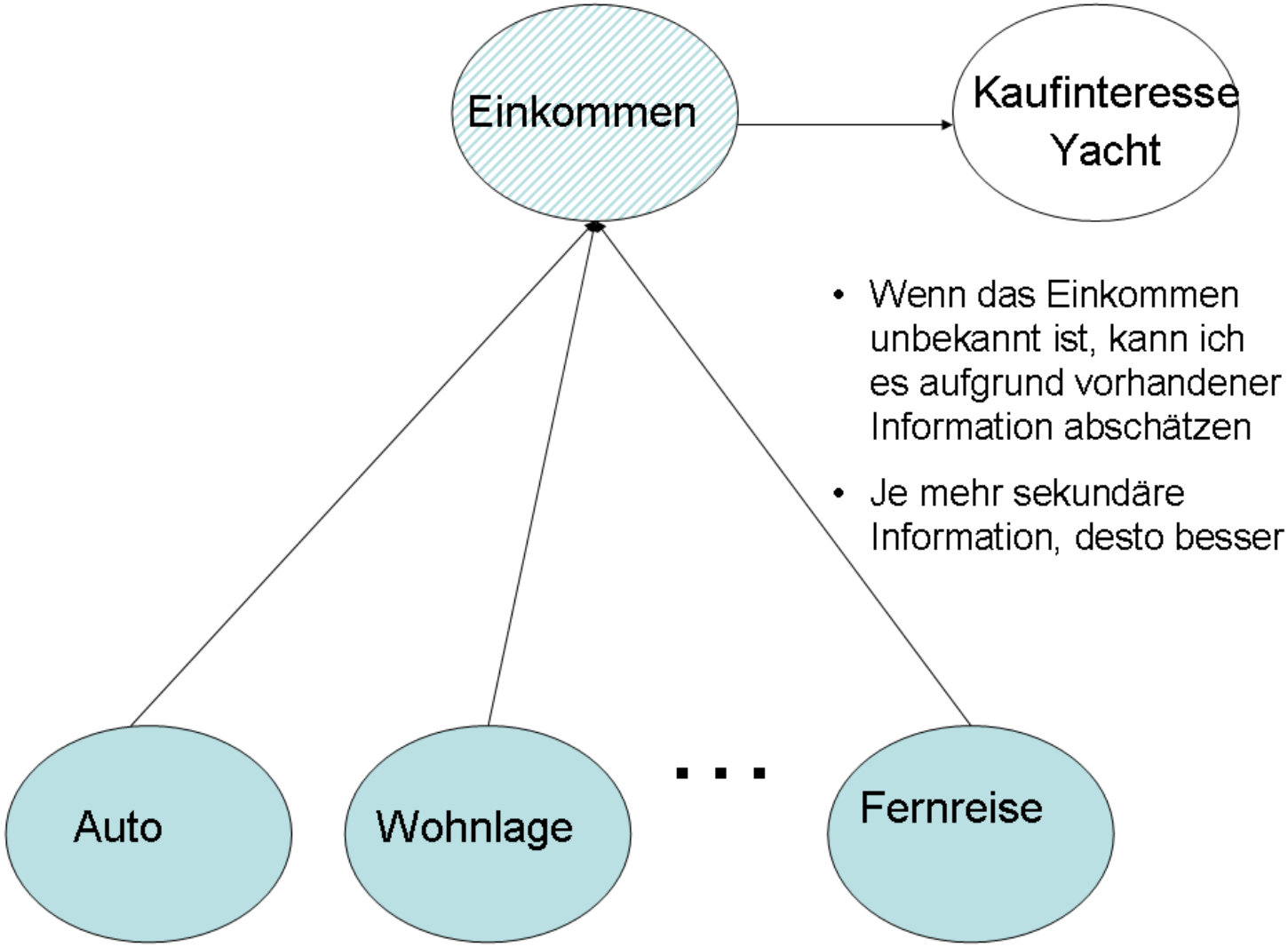
- Von allen Vögeln auf der Welt fliegen 90%
- Von allen Vögeln auf der Welt, die nicht zur Klasse flugunfähiger Arten gehören, fliegen 94%
- Von allen Vögeln auf der Welt, die nicht zur Klasse flugunfähiger Arten gehören, und die nicht von Menschen gehalten werden, fliegen 96% ...
- Grundsätzliches Probleme:
  - Komplexität des unterliegenden (deterministischen) Systems
  - Unvollständige Information
- Hieraus begründet sich der Erfolg statistisch-basierter Ansätze

Beispiel: Kaufentscheidung



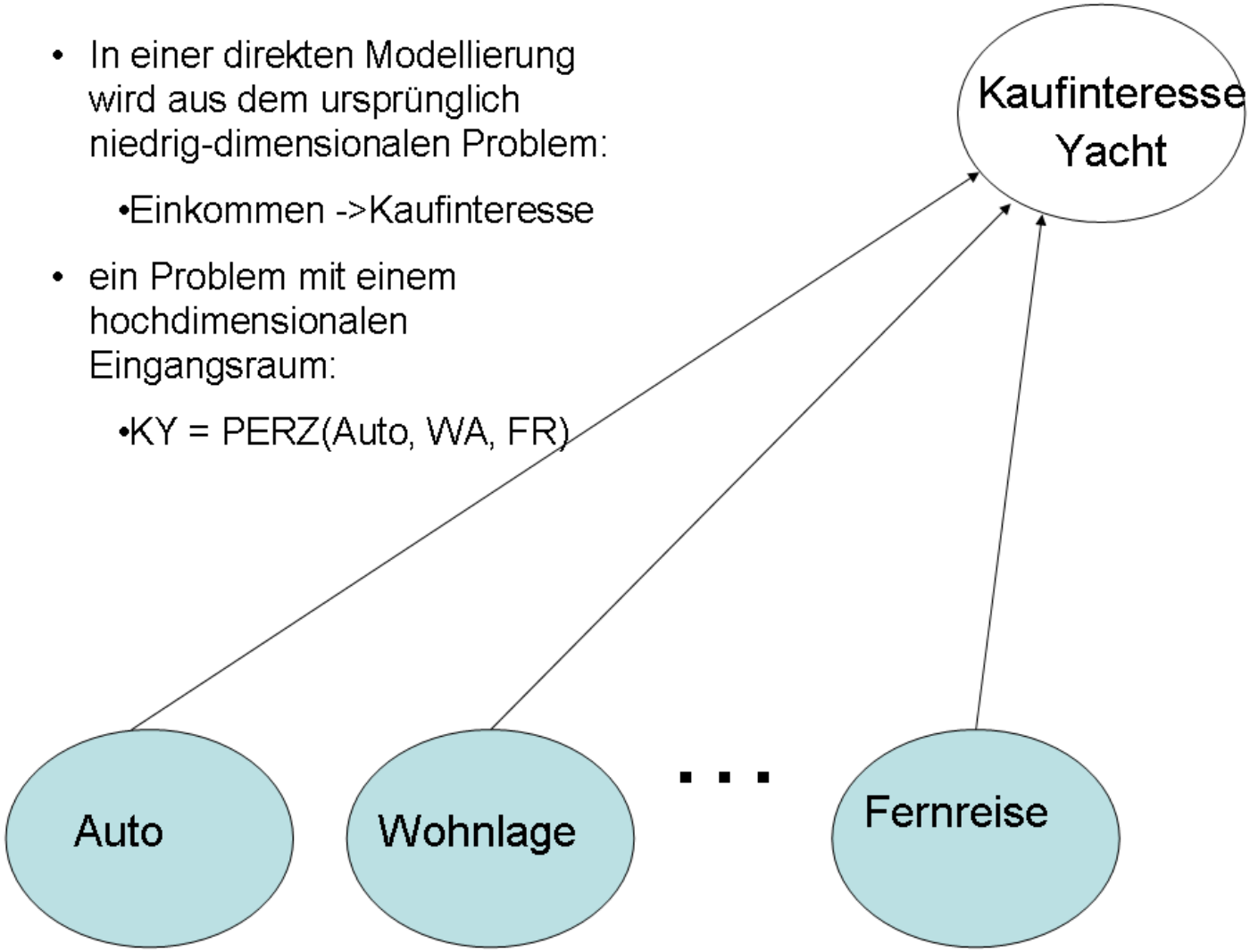
Annahme:

- Kaufinteresse hängt nur vom Einkommen ab:
- (JE>100000€) -> (Kaufinteresse_Yacht =hoch)



- Wenn das Einkommen unbekannt ist, kann ich es aufgrund vorhandener Information abschätzen
- Je mehr sekundäre Information, desto besser

- In einer direkten Modellierung wird aus dem ursprünglich niedrig-dimensionalen Problem:
  - •Einkommen ->Kaufinteresse
- ein Problem mit einem hochdimensionalen Eingangsraum:
  - •KY = PERZ(Auto, WA, FR)



Lernen versus Aktionen

- Wenn die Ableitung von Informationen oder Wissen im Vordergund steht, sind Lernansätze mächtig, da diese mit der Komplexität der Welt umgehen können.
- Die Entscheidungsregel selber kann man hingegen oft als Regel oder Funktion definieren. Ein Grund warum mein Smart Phone (vermultich) ohne Maschinelles Lernen auskommt.