Künstliche Intelligenz

llm18

WS18/19

Georg Ringwelski

Was ist KI?

Zettelwand

- Braune Karte: Welche KI-Systeme haben Ihr Leben bisher verbessert?
- Rote Karte: Welche KI-Systeme würden Sie sich noch wünschen?

Was ist Künstliche Intelligenz?

Man weiß es nicht!

Warum nicht?

Weil es keine Definition
für „Intelligenz“ gibt

Naja, es ist eine Teildisziplin
der Informatik…

Was ist Künstliche Intelligenz?

Systeme, die wie Menschen denken Automatisierung von Verstand, Emotion, Entscheidungsfindung, Problemlösung, Lernen	**Systeme, die rational denken** Studie mentaler Fähigkeiten und Nutzung in maschinell verarbeitbaren Modellen
Systeme, die wie Menschen agieren Maschinen, die Funktionen erfüllen, bei denen der Mensch ihnen noch überlegen ist	**Systeme, die rational agieren** Entwurf intelligenter Agenten, Umsetzung „intelligenten Verhaltens“ in Maschinen

Quelle: Russel, Norvig

Was erwarten wir von der KI?

(1950-2010:)

Eine intelligente Maschine besteht den ***Turing Test***

- Maschine antwortet auf Fragen
- bestanden, wenn die Testperson nicht erkennt, ob die Antworten von einem Menschen stammen

Anforderungen zum Bestehen des Turing Test

- Verarbeitung natürlicher Sprache (Analyse und Synthese)
- Wissensrepräsentation
- Logisches Schließen und Problemlösen
- Maschinelles Lernen

→ Loebner Preis: 3T$, 25T$, 100T$ noch nicht vergeben,

→ chatbots.org: Liste einiger Systeme für den Turing Test und andere KI-Fertigkeiten

Ab ca 2010: „Beyond the Turing Test“

- Ziel: Leistungsfähige Mensch-Maschine-Systeme
  - Stärken des Nutzers integrieren
  - Ergonomische Softwarewerkzeuge erstellen
  - Einsatz von KI-Methoden zur **Ergänzung der menschlichen Leistung**
- Herausforderungen
  - Mensch-Maschine-Schnittstelle → LV: EUS
  - Festlegung der „Arbeitsteilung“
  - Anwendung von KI-Methoden für die Aufgaben des Computers → LV: KI

Wo wird KI heute angewendet?

- Planung und Disposition → Demo
- Spiele
- Autonome Kontrolle (zB Steuerungs- und Messtechnik)
- Diagnose und Wartung
- Steuerung in Robotik und eingebettete Systeme
- Spracherkennung und –synthese
- etc

Themen KI

1. Einführung KI

   Organisatorisches

2. Maschinelles Lernen
   1. Lernende Klassifizierungssysteme
   2. Evaluierung lernender Systeme
   3. Lernverfahren
3. Problemlösen
   1. Vollständige Suche
   2. Unvollständige Suche

Ziele der LV

- Sie haben eine genaue Vorstellung von einigen Begrifflichkeiten aus dem KI Umfeld: Maschinelles Lernen, Optimierung…
- Sie haben Erfahrung mit der Realisierung und Anwendung von Lern- und Suchverfahren.
- Sie können nichtdeterministische Systeme wissenschaftlich fundiert evaluieren.

Kontext der LV

Voraussetzung

- Programmierung, etwas Software-Engineering, ~~statistisches Testen~~

Fortsetzung/Ergänzung:

- Klassifikation/Lernen auch in Data Mining
- Problemlösung, Nutzerinteraktion auch in „Optimierung und Entscheidungsunterstützung“

Literatur

KI allgemein:

- Russel, Norvig: „Künstliche Intelligenz“

Maschinelles Lernen:

- Ethem Alpaydin: „Maschinelles Lernen“

Suche, Problemlösen:

- Burke, Edmund: „Search Methodologies“

Organisatorisches

- Integrierte LV: VL und ÜB „vermischt“
- Prüfungsleistung: PL
  - 3 komplexe Aufgaben: Lernen, Problemlösen
  - Details auf → Aufgabenblatt
- Lernen von den anderen: Jeder stellt seine Lösung vor
- Infos, Folien und code auf meiner website
- Noch Fragen?

Themen KI

1. Einführung KI
2. Maschinelles Lernen
   1. Lernende Klassifizierungssysteme
   2. Evaluierung lernender Systeme
   3. Lernverfahren
3. Problemlösen
   1. Vollständige Suche
   2. Unvollständige Suche

Was ist Maschinelles Lernen?

ML stellt Konzepte bereit, die

Wahrnehmungen nicht nur für das aktuelle Handeln verwenden, sondern auch künftige Handlungen verbessern.

[Russel, Norvig]

Was ist Maschinelles Lernen?

Def .

A computer program is said to *learn* from Experience E wrt. some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

[Mitchell]

→ Lernprobleme sind gegeben durch (T,P,E)

Was ist Maschinelles Lernen?

Beispiel: Schrifterkennung (Handschrift)

Automatisches „Abschreiben“ handschriftlicher Texte von Bildern in Textdateien.

Was ist hier Erfahrung, Fertigkeit und Leistungsmessung?

Was ist Maschinelles Lernen?

Erarbeiten Sie in Kleingruppen T,P und E für folgende Aufgaben

- Automatisches Spurhalten von PKW
- Schach/Dame/Go… spielen
- Erreichen aller Ecken in beliebigen Wohnungen mit einem Staubsaugerroboter

Entwicklung lernender Systeme

1. *Erfahrung (E)* modellieren und Verarbeiten
2. Repräsentation der *Fertigkeit (T)*
3. *Zielfunktion* (P) festlegen
4. Annäherungsalgorithmus für die Fertigkeit T festlegen (das ist der Lernalgorithmus, zB KNN)

---------- und dann…

1. Trainieren mit E mit Bewertung P
2. Einsatz des trainierten Systems

Entwurf lernender Systeme

1. Erfahrung modellieren und Verarbeiten

Auswahl

- Systeme lernen ausschließlich aus der gegebenen Erfahrung E
- Ziel: Repräsentation „aller“ Aufgaben T die mit P bewertet werden können

und Reihenfolge

- Zu jedem Zeitpunkt ist nur gelernt, was vorher erfahren wurde
- Viele Systeme lernen „inkrementell“, so dass die Ergebnisse von der Reihenfolge abhängen können

Entwurf lernender Systeme

2. Fertigkeit festlegen

Was genau soll das System leisten?

Eine Abbildung von (aktuellen und lokalen) Beobachtetem auf eine intelligente Reaktion

z.B.

- Aktueller Zustand → Aktion
- Bild → Klasse des dargestellten Objekts

Erfahrung muss also immer aus Paaren bestehen:

(<Beobachtung>,

<intelligente Reaktion>)

Entwurf lernender Systeme

2. Fertigkeit festlegen

Übung in Murmelgruppe (10 min)

Geben Sie formale Definitionen für E,T und P zur Klassifikation von Fahrzeugen an. Als Erfahrung dienen folgende Beispiele

1. Auto: 4 Räder, 100 PS
2. Motorrad: 2 Räder, 20 PS
3. Fahrrad: 2 Räder, 0 PS
4. Auto: 4 Räder, 120 PS
5. Motorrad: 2 Räder: 100 PS

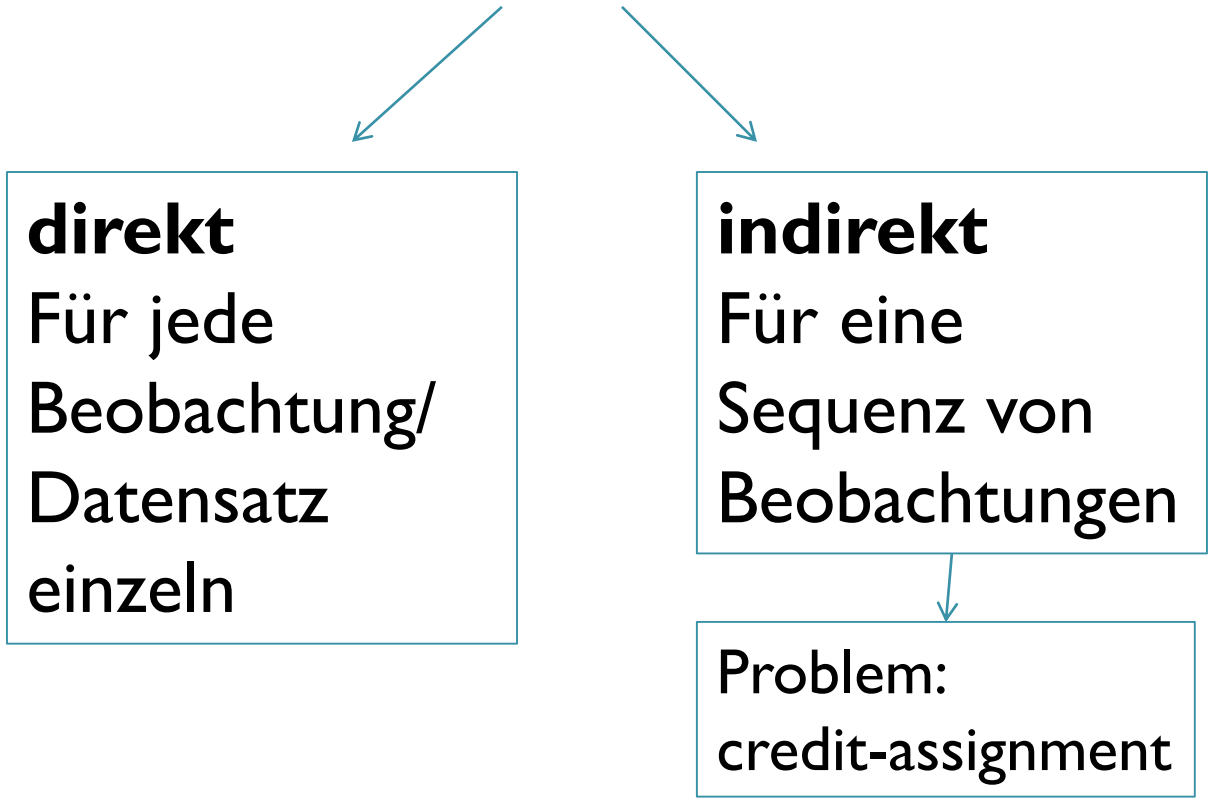
Präsentieren Sie Ihre Lösung

Entwurf lernender Systeme

2. Fertigkeit festlegen

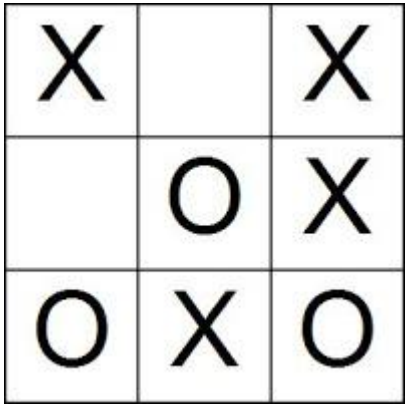
Erfahrung ist:

Beobachtung + feedback



Erfahrung sammeln

Beispiel: Der Computer lernt TicTacToe



Modellieren Sie T, P und E

Erfahrung sammeln

Übung Murmelgruppe (10 min):
Der Computer lernt das Spiel Mühle

Modellieren Sie T, P und E

Präsentieren Sie Ihre Lösung

Erfahrung sammeln

Arten des maschinellen Lernens

- *Überwachtes Lernen*: Lernen einer Fertigkeit aus Beispielen und Gegenbeispielen :

  E = Beispiel x Bewertung

- *Nicht überwachtes Lernen*: Lernen eines Konzepts: E = positive Beispiele

- *Verstärkendes Lernen*: Lernen durch Ausprobieren und Belohnungen:

  E = getätigtes Verhalten (ggf. mehrere Aktionen) x Belohnung

Entwurf lernender Systeme
3. Zielfunktion festlegen

- Die Zielfunktion gibt an, wie gut das System seine Fertigkeit durchgeführt hat
- Beispiele
  - Wurde das Bild richtig erkannt?
  - Hat der Computer das Spiel gewonnen?
- Damit kann die Qualität des lernenden Systems durch statische Tests gemessen werden.

Entwicklung lernender Systeme

1. *Erfahrung (E)* modellieren und Verarbeiten
2. Repräsentation der *Fertigkeit (T)*
3. *Zielfunktion* (P) festlegen
4. Annäherungsalgorithmus für die Fertigkeit T festlegen (das ist der Lernalgorithmus, zB KNN)

---------- und dann…

1. Trainieren mit E und Bewertung P
2. Einsatz des trainierten Systems

Entwurf lernender Systeme

4. Repräsentation der Fertigkeit

Maschinell effizient berechenbar

Diverse Methoden werden in der KI entwickelt

- Regeln, Formeln, Funktionen („Expertensysteme“)
- Künstliche Neuronale Netze
- Entscheidungsbäume
- Und viele mehr (vgl Übung 2)

zur Approximation der Fertigkeit als effiziente Funktion (i.d.R. O(1))

Die Erzeugung darf länger dauern

Fragestellungen beim ML

- Welche Algorithmen lernen am besten Fertigkeiten aus Trainingsmengen?
- Wie groß muss die Trainingsmenge sein und welche Beispiele sollten enthalten sein?
- Was ist die beste Reihenfolge für Trainings-Erfahrungen?
- U.v.m.

Themen KI

1. Einführung KI
2. Maschinelles Lernen
   1. Lernende Klassifizierungssysteme
   2. Evaluierung lernender Systeme
   3. Lernverfahren
3. Problemlösen
   1. Vollständige Suche
   2. Unvollständige Suche

2.1 lernende Klassifizierungssysteme

Was ist Klassifikation?

- Voraussetzung: Ein Universum unterscheidbarer Objekte/Aktionen
- Klassifikation der Objekte/ Aktionen in (meist disjunkte) *Konzepte*
- Jedes Objekt/Aktion wird durch einen *Merkmalsvektor* spezifiziert

2.1 lernende Klassifizierungssysteme

Bsp:

- **3 Konzepte**: Auto, Motorrad, Fahrrad
- **2 Merkmele**: PS, Anz_Räder
- Trainingsmenge mit 3 Datensätzen:

Bsp Nr	Anz-Räder	PS	Konzept
1	4	100	Auto
2	2	90	Motorrad
3	2	2	E-Fahrrad

2.1 lernende Klassifizierungssysteme

Übungsaufgabe 1

Generierung von Merkmalsvektoren aus Bildern

→Diskussion Ihrer Ideen

→Brainstorming

2.1 lernende Klassifizierungssysteme

Übung Merkmalsvektoren

Definieren Sie in Einzelarbeit *Merkmalsvektoren* mit *Konzepten* für das Konzeptlernen von

- Pflanzen
- Autos
- Lebensmitteln
- Lebensversicherungen
- Laptop Computer
- Smartphones

Und präsentieren Sie ihr Entscheidung mit Beispielen

2.1 lernende Klassifizierungssysteme

Konzeptlernen/Klassifikationslernen

1. *Training* des Systems mit Paaren: <Merkmalsvektor, Konzept>

   (Bei nur einem Konzept: positive und negative Beispiele)

2. *Testen* des Systems durch Klassifizierung anderer Beispiele
3. *Betrieb* des Systems: Klassifikation

2.1 lernende Klassifizierungssysteme

Beispiel Konzeptlernen

Konzipieren Sie ein System zum Erkennung handgeschriebener Ziffern.

Erarbeiten Sie in Kleingruppen:

Was ist im Einzelnen zu tun?

2.1 lernende Klassifizierungssysteme

Induktives Lernen

- **Ziel**: Hypothese so zu definieren, dass sie das Universum richtig in die gewünschten Konzepte partitioniert
- **Grenze**: Ausschließlich das Wissen aus der Erfahrung (Trainingsmenge) steht zur Verfügung
- **Idee**: Wenn die Erfahrung groß genug ist wird die Hypothese „allgemeingültig“ für künftige Aufgaben

2.1 lernende Klassifizierungssysteme

Bsp:

- **Konzepte**: Auto, Motorrad, Fahrrad
- **Merkmele**: PS, Anz_Räder
- Trainingsdaten:

Bsp Nr	Anz-Räder	PS	Konzept
1	4	100	Auto
2	2	90	Motorrad
3	2	2	E-Fahrrad

→Definieren Sie einen Klassifikator in Form eines Entscheidungsbaumes (Einzelarbeit)

2.1 lernende Klassifizierungssysteme

Ordnung auf Hypothesen

- **Ziel**: Suche nach der besten Hypothese h in einer implizit gegebenen Menge
- **Idee**: partielle Ordnung auf Hypothesen „ist allgemeiner als“

Def: Let h_j and h_k be hypothesis (boolean-valued functions) defined over object data X. Then h_j is more general or equal to h_k ($h_j \geq h_k$) iff

$\forall x \in X \,.\, h_k(x) = \text{true} \rightarrow h_j(x) = \text{true}$

- **Auswahl**: nicht zu allgemein (*underfitting*) und nicht zu speziell (*overfitting*)

2.1 lernende Klassifizierungssysteme

Bsp:

- **Konzepte**: Auto, Motorrad, Fahrrad
- **Merkmele**: PS, Anz_Räder
- Trainingsdaten:

Bsp Nr	Anz-Räder	PS	Konzept
1	4	100	Auto
2	2	90	Motorrad
3	2	2	E-Fahrrad

→Ordnen Sie die gefundenen Hypothesen

2.1 lernende Klassifizierungssysteme

Fallstudie: Entscheidungsbaum alg. cal2

- Objekte werden durch Merkmale $x_1 \ldots x_n$ beschrieben
- Die Objekte sind in paarweise disjunkte Klassen/Konzepte eingeteilt:

  τ: Objekt→Konzept
- Das System soll die Konzepte lernen, d.h. künftige Objekte dem richtigen Konzept zuordnen können
- Hypothesen werden durch einen ***Entscheidungsbaum*** α dargestellt

2.1 lernende Klassifizierungssysteme

Fallstudie: cal2, der Algorithmus

α = ? // allgemeinste Hypothese

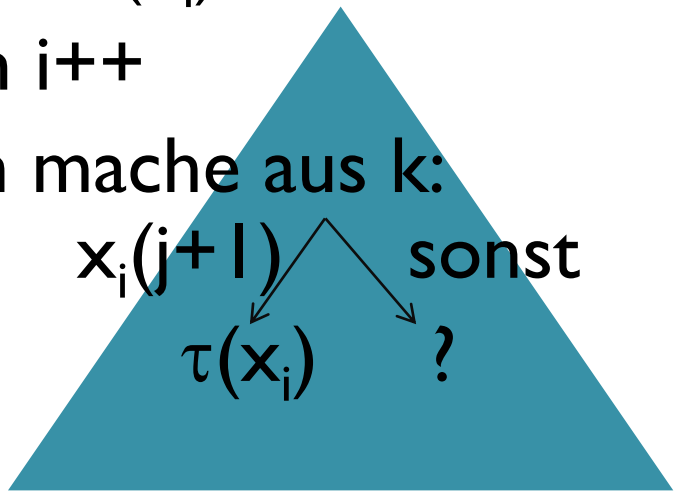
Solange ! abbruch(i)

k_j := klassifiziere(α, x_i) //Konzept Ebene j

wenn k_j = ? dann k_j := $\tau(x_i)$; i++

wenn k_j = $\tau(x_i)$ dann i++

wenn $k_j \neq \tau(x_i)$ dann mache aus k:



Legende

α : Entscheidungsbaum

k_x : Konzept in Ebene x

x_i : Trainingsbeispiel

$\tau(x)$: Konzept von x

2.1 lernende Klassifizierungssysteme

Bsp:

Bsp Nr	Gewicht	Garpunkt	Konzept
1	300	medium	Steak
2	250	durch	Schnitzel
3	250	englisch	Steak

→ Trainieren Sie einen Entscheidungsbaum mit cal2!

2.1 lernende Klassifizierungssysteme

Fallstudie: cal2, Abbruch

Konzepte (bei cal2) sind lt. Vor. disjunkt

⇒ $\exists n_0 \ \forall\ n \geq n_0$. klassifiziere $(\alpha, x_n) = \tau(x_n)$

⇒ Wenn n_0 erreicht ist terminiert der Algorithmus

- Aber was ist n_0?
- So können sehr große Bäume entstehen (overfitting)

Bei nicht-disjunkten Konzepten (zB Messfehler) fehlen dann Merkmale→ Abbruch

2.1 lernende Klassifizierungssysteme

Wie lange muss man wirklich trainieren?

- Bis die vorhandene Zeit ausgeschöpft ist

  (i.d.R. steht beim ML viel offline-Zeit Verfügung)
- Bis eine gewünschte Sicherheit erreicht wurde, z.B.: Unter den letzten n Objekten wurden m richtig klassifiziert (i.e. die Hypothese traf zu (m/n)% zu)

Also:

- Wähle möglichst große Trainingsmengen
- Setze einzelne Trainingsobjekte nicht zu oft im selben Lernprozess ein

2.1 lernende Klassifizierungssysteme

Weitere Herausforderungen bei Entscheidungsbäumen a)

Invarianz gegen Reihenfolge der Merkmale im Vektor (zB im Algorithmus ID3)

- → wähle immer als nächstes das Merkmal mit der geringsten Entropie bzgl. allen Testbeispielen
- → Bevorzugung von Merkmalen mit vielen Ausprägungen
- → so entstehen eher breite Bäume

2.1 lernende Klassifizierungssysteme

Weitere Herausforderungen bei Entscheidungsbäumen b)

Invarianz gegen diverse Ausprägungen eines Attributes

- → An den Zweigen im Entscheidungsbaum stehen nicht Werte, sondern Grenzen
- → Diese Schwellwerte werden dynamisch mit gelernt
- → Mit binären Entscheidungen (CART-Alg.) oder mehreren Schwellen (C4.5-Alg.) möglich

2.1 lernende Klassifizierungssysteme

Wie werden die Systeme dann benutzt?

- Eingabe: ein Objekt (bzw. sein Merkmalsvektor)
- Ausgabe das Konzept dieses Objekts

2.1 lernende Klassifizierungssysteme

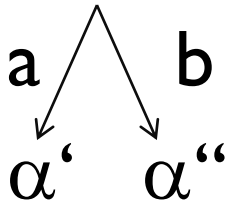
Fallstudie: cal2, Anwendung zur Klassifikation

klassifiziere(α,y)

wenn $\alpha \in \{?\} \cup$ Konzept dann α

sonst

wenn α = j und y(j) =a



dann klassifziere(α‘,y)

sonst klassifiziere(α“,y)

2.1 lernende Klassifizierungssysteme

Zusammenfassung

- Was sind lernende Klassifizierungssysteme?
- Was ist (in diesem Zusammenhang) eine Hypothese?
- Wie werden Hypothesen geordnet?
- Wie kann man Hypothesen kompakt darstellen?

Übungsaufgabe 1

Fragestellungen abstimmen

Themen KI

1. Einführung KI
2. Maschinelles Lernen
   1. Lernende Klassifizierungssysteme
   2. Evaluierung lernender Systeme
   3. Lernverfahren
3. Problemlösen
   1. Vollständige Suche
   2. Unvollständige Suche

2.2 Evaluierung lernender Systeme

Leitfragen

- Wie gut ist ein Klassifikator für eine Problemklasse?

- Wenn ein Klassifikator bei einer Problemklasse gut ist ist, was ist dann für andere Problemklassen zu erwarten?

2.2 Evaluierung lernender Systeme

Wie gut ist ein Klassifikator?

Ausgangssituation: eine unbekannte Verteilung der Konzepte unter den Beispielen B

Wir suchen:

a) Repräsentative Trainingsmenge $T \subset B$
b) Repräsentative Menge zur Evaluierung $E \subset B$

Es gilt immer: $T \cap E = \emptyset$

2.2 Evaluierung lernender Systeme

Def: der **Beispiel-Fehler** $err_E(h)$ einer Hypothese h bzgl einer Zielfunktion f und Evaluationsmenge E ist

ersetzen

$$err_E(h) := \frac{1}{n} \sum_{x \in E} \delta(f(x), h(x))$$

wobei n =|E| und

$$\delta(f(x),h(x)) = \begin{cases} 1, \text{ falls } f(x) \neq h(x), \\ 0 \text{ sonst.} \end{cases}$$

Ggf: weitere Klasse für „weiß nicht“

2.2 Evaluierung lernender Systeme

Beispiel: Fehler bei einer Testmenge T

- Bei einem t = (fv ,concept) $\in$ T vergleichen wir hypothese.classify(fv) mit concept
- z:.B. bei |T| = 100 wurden 30 richtig, 20 falsch und 50 gar nicht klassifiziert
- Dann ist err_T(hypothese) = 1/100 * 70 (wenn nicht klassifiziert als falsch gilt)
- Oder eben 1/50 * 20

  (wenn nicht klassifiziert nicht zählt)

2.2 Evaluierung lernender Systeme

Wie gut ist ein Klassifikator wirklich, also nicht nur in einem Beispiel T/E ?

→

Wie groß ist der *durchschnittliche* Fehler bei unterschiedlichen T/E?

→

Wir machen n Experimente mit unterschiedlichen Testmengen $E \subseteq EE$ und berechnen den Durchschnitt

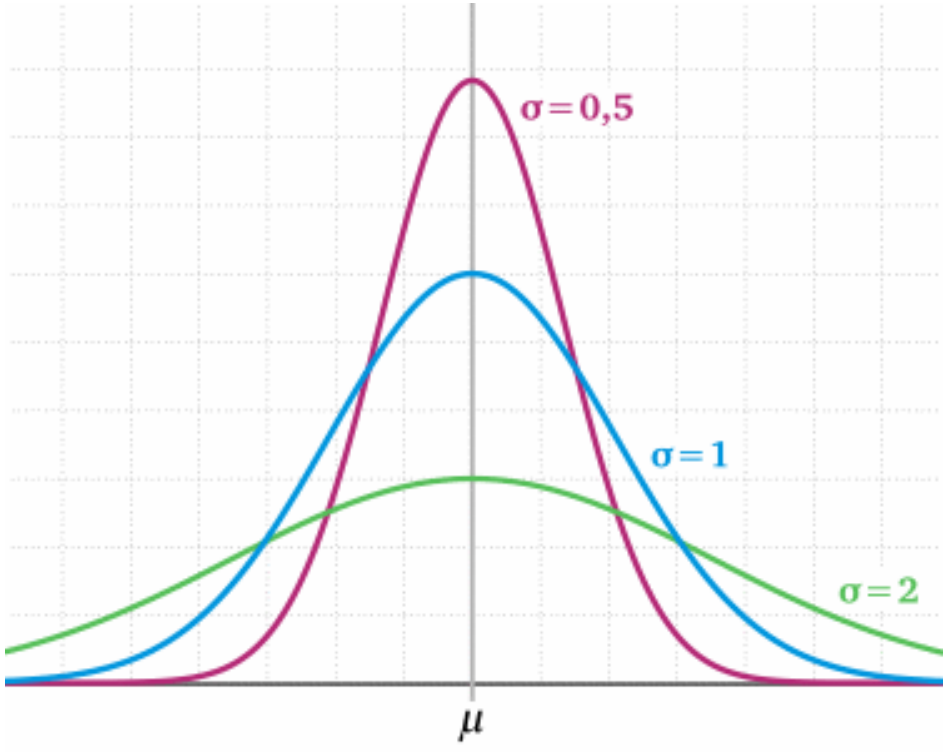
$$err_{EE}(h) = 1/n * \Sigma\ err_{E}(h)$$

→

Und erhalten normalverteilte Messwerte

2.2 Evaluierung lernender Systeme

Welche Kurve ist am aussagekräftigsten?



Wir wollen messen, wie viele Ergebnisse in der Nähe des Mittelwertes liegen

2.2 Evaluierung lernender Systeme

Konfidenzintervall

Mit der Wahrscheinlichkeit N% liegt der wahre Fehler im Intervall

$$err_{EE}(h) \pm z_N \frac{\sigma}{\sqrt{n}}$$

mit

N%	50%	68%	80%	90%	95%	98%	99%
z_N	0.67	1.00	1.28	1.64	1.96	2.33	2.58

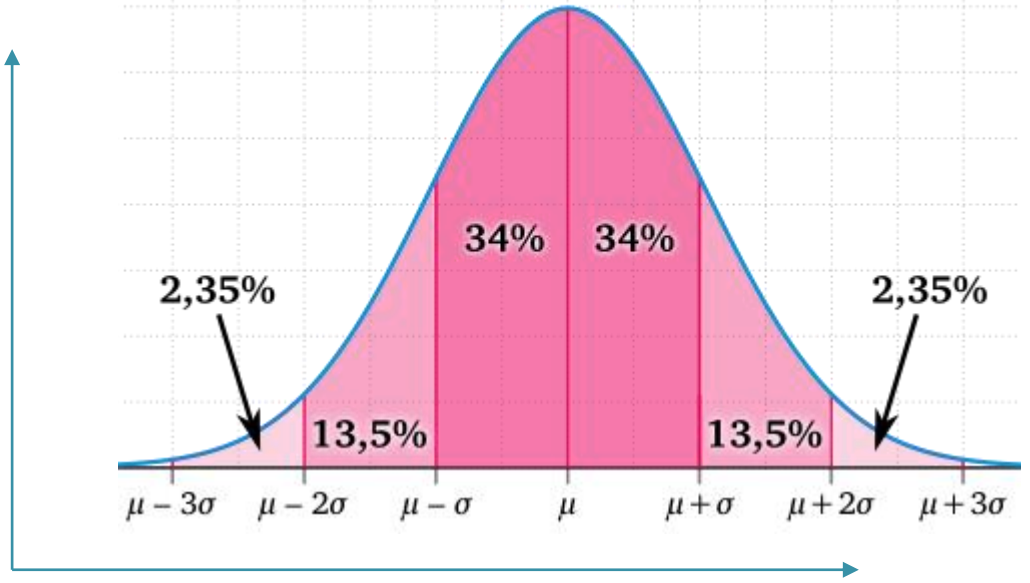
Standardabweichung:

$$\sigma = \sqrt{\frac{\sum_{E \subset EE} (err_E(h) - err_{EE}(h))^2}{n}}$$

2.2 Evaluierung lernender Systeme

Wiederholung von Experimenten

Experimente mit gemessenem Testwert



Messwert

Mittellwert ist μ

Konfidenzintervall mit 68% : $\mu \pm \sigma$

Kionfidenzintervall mit 95%: $\mu \pm 2\sigma$

2.2 Evaluierung lernender Systeme

Beispiel: Konfidenzintervall eines Experiments

- Wir definieren 100 verschiedene Testmengen T = {T1…T100}
- Wir berechnen jeweils den Fehler: $err_{Ti}(h)$
- Wir berechnen den durchschnittlichen Fehler: $1/100 * \Sigma\ err_{Ti}(h)$
- Und das Konfidenzintervall (zB 95%):

  $1.96 * \sigma$ /sqrt(100)

2.2 Evaluierung lernender Systeme

HOWTO: Evaluierung eines Klassifikators

Wiederhole für verschiedene Testmengen und Merkmals-Reihenfolgen{

1. Verwende repräsentative Testbeispiele (die keine Trainingsbeispiele sind)
2. Klassifiziere sie und zähle, wie oft richtig/falsch/gar nicht klassifiziert wurde
3. Berechne den Fehler: $err_E(h)$, seinen Durchschnitt mit vorigen Durchläufen und sein Konfidenzintervall, wenn dieses zu groß ist gehe zu 1.

}

Berechne Durchschnitt und Konfidenzintervall über diesen Tests als Ergebnis

Themen KI

1. Einführung KI
2. Maschinelles Lernen
   1. Lernende Klassifizierungssysteme
   2. Evaluierung lernender Systeme
   3. Lernverfahren
3. Problemlösen
   1. Vollständige Suche
   2. Unvollständige Suche

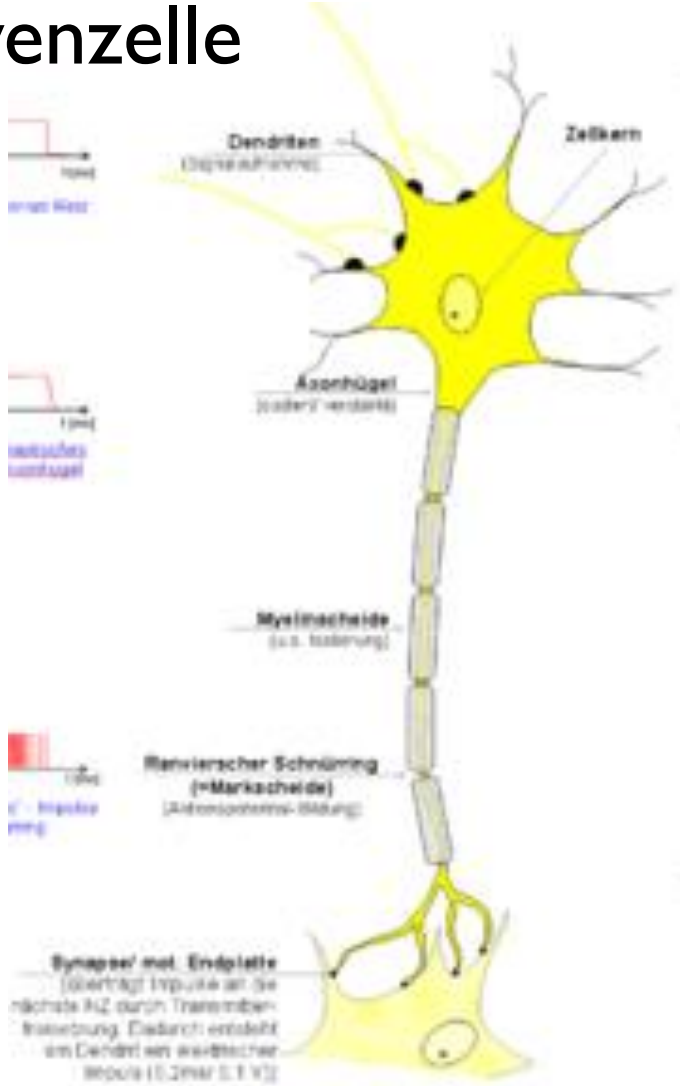
2.3 Künstliche neuronale Netze

- Erstes Modell [McCulloch,Pitts, 1943] als Modell einer menschlichen Nervenzelle
- Heute: Massive Fortschritte der kognitiven Psychologie und Neurophysiologie → Das Gehirn funktioniert eben nicht so einfach!
- KNN dennoch erfolgreiche Methode des maschinellen Lernens

2.3 Künstliche neuronale Netze

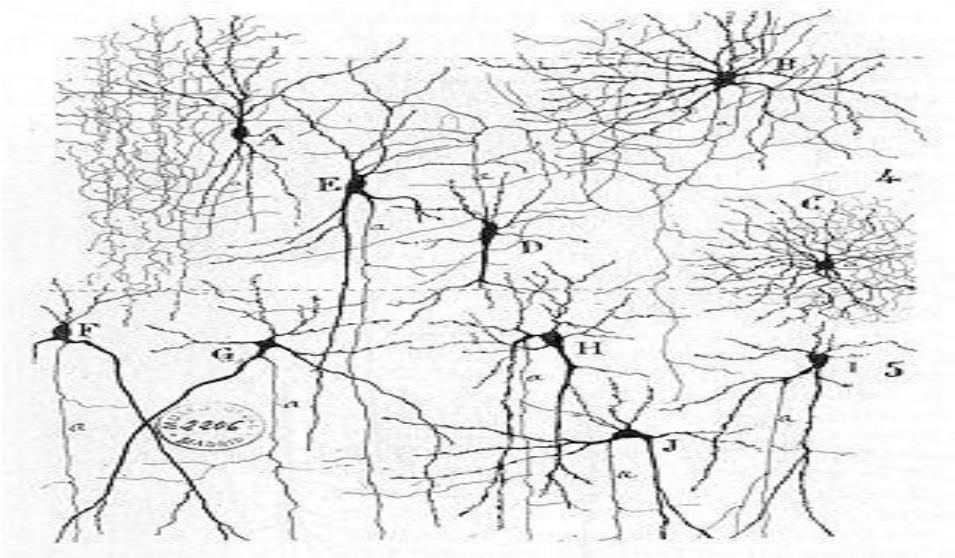
Eine biologische Nervenzelle

- Eingabe: Dendriten, Ausgabe: Axon,
- Im Soma werden solange impulse gesammelt, bis das Axon feuert



2.3 Künstliche neuronale Netze

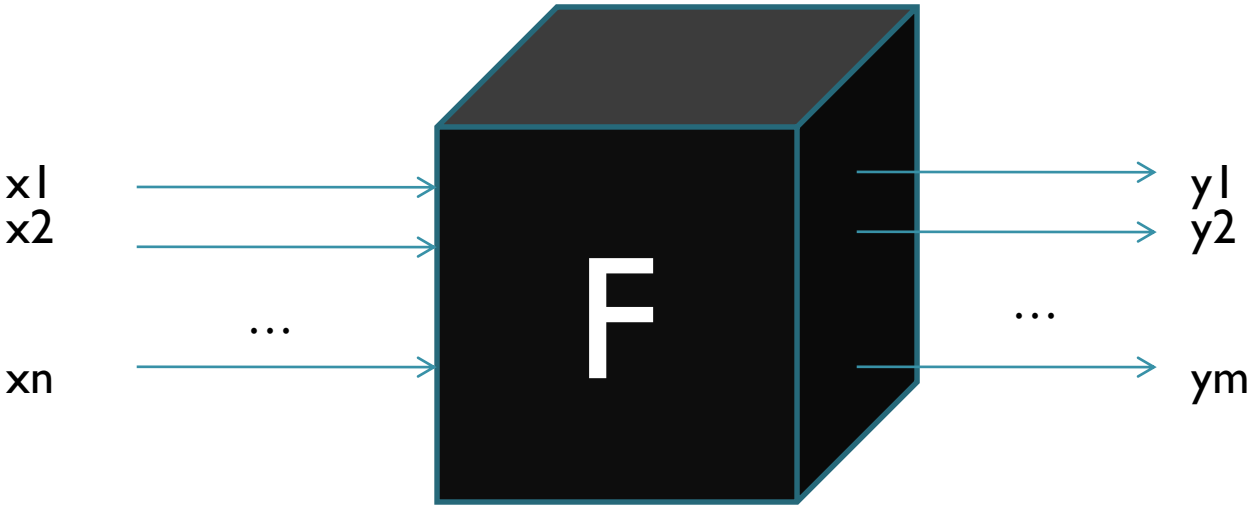
Biologische neuronale Netze



- Netze aus Millionen Neuronen verbunden durch noch mehr Synapsen
- Lernen: Veränderung der Reizschwellen, Synapsenbildung, neue Zellen, Hormone u.v.m.

2.3 Künstliche neuronale Netze

Das Modell von McCulloch und Pitts

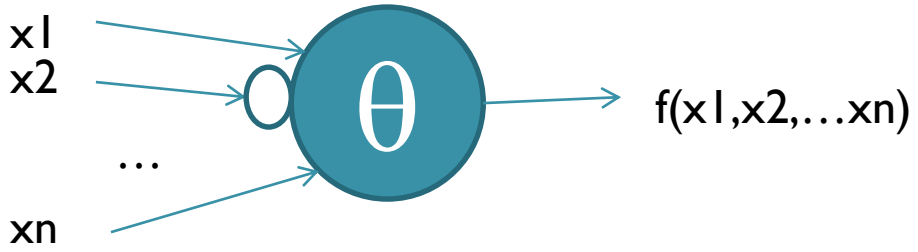


- Das KNN ist eine Abbildung: F: $\Re^n \rightarrow \Re^m$
- Vorbereich: Merkmalsvektoren, Nachbereich: Kodierung von Konzepten
- Es besteht aus „Neuronen“

2.3 Künstliche neuronale Netze

Das Modell von McCulloch und Pitts

Ein Neuron



- θ ist dabei ein Schwellenwert
- Es gibt hemmende (○) und erregende Leitungen

2.3 Künstliche neuronale Netze

Das Modell von McCulloch und Pitts

Berechnung der Ausgabe

Seien x1…xn erregende Leitungen und y1…ym hemmende Leitungen, dann implementiert das Neuron die Funktion $f: N^n \rightarrow \{0,1\}$ mit

$$f(x1\ldots xn, y1\ldots ym) = \begin{cases} 0, \text{ falls } y1 \vee \ldots \vee ym = 1 \\ 0, \text{ falls } x1+\ldots+xn < \theta \\ 1, \text{ sonst} \end{cases}$$

Dabei ist θ ein variabler Schwellenwert

2.3 Künstliche neuronale Netze

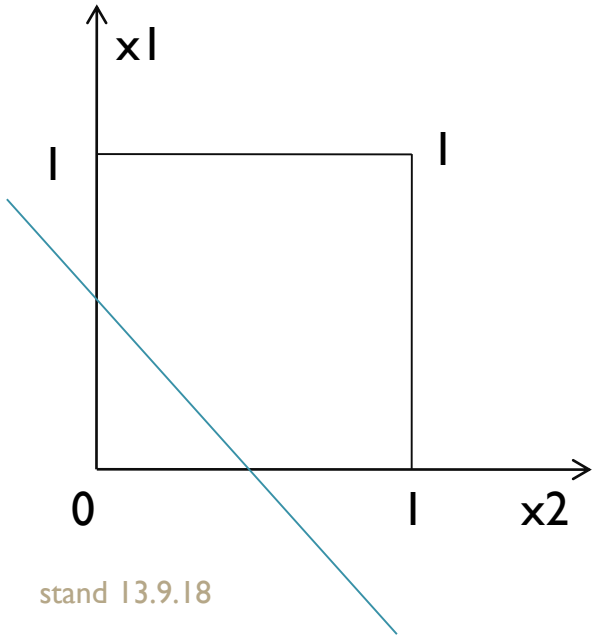
Das Modell von McCulloch und Pitts

Gemoetrische Interpretation

Bei n Eingabeneuronen definiert eine McCullochPitts-zelle eine Hyperbene im n-dimensionalen Raum

$f: \{0,1\}^2 \rightarrow \{0,1\}$

$f(x1,x2) = 0$, falls $x1=x2=0$
$= 1$, sonst



2.3 Künstliche neuronale Netze

Das Modell von McCulloch und Pitts

Eigenschaften

- Mit einem Netz aus McCulloch-Pitts-Zellen kann man jede Hypothese zur Klassifizierung realisieren
- Die Klassifikation ist effizient berechenbar

(Lernverfahren aber unklar: Topologie und Schwellenwerte wären einzustellen)

2.3 Künstliche neuronale Netze

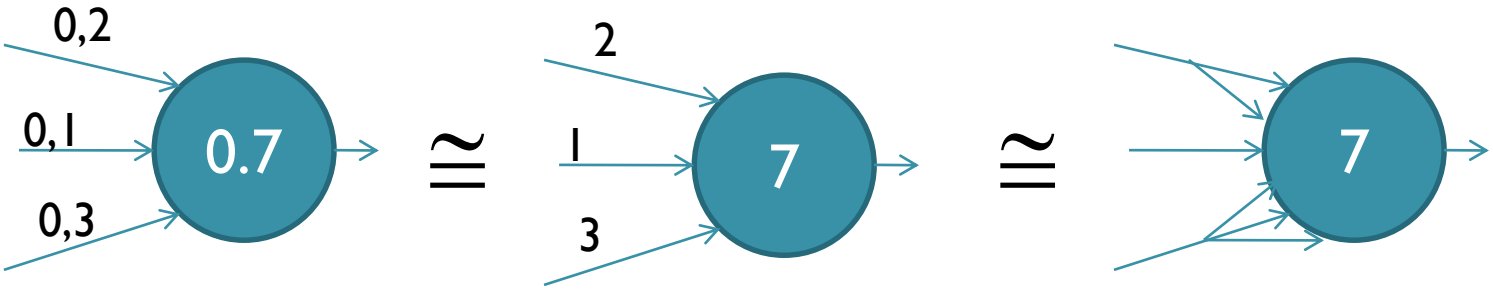
Neuronen mit Relativer Hemmung

$$f(x1 \ldots xn, y1 \ldots ym) = \begin{cases} 0, \text{ falls } \Sigma\, x < \theta + \Sigma\, y \\ 1, \text{ sonst} \end{cases}$$

- McCulloch-Pitts-Neuronen verwenden *absolute* Hemmung (s.o.)
- Relativ hemmende Verbindungen schalten nicht auf null, sondern erhöhen den Schwellenwert

2.3 Künstliche neuronale Netze

Neuronen in gewichteten Netzen



Jedes gewichtete Netz kann als äquivalentes ungewichtetes Netz dargestellt werden

2.3 Künstliche neuronale Netze

Das Modell von McCulloch und Pitts

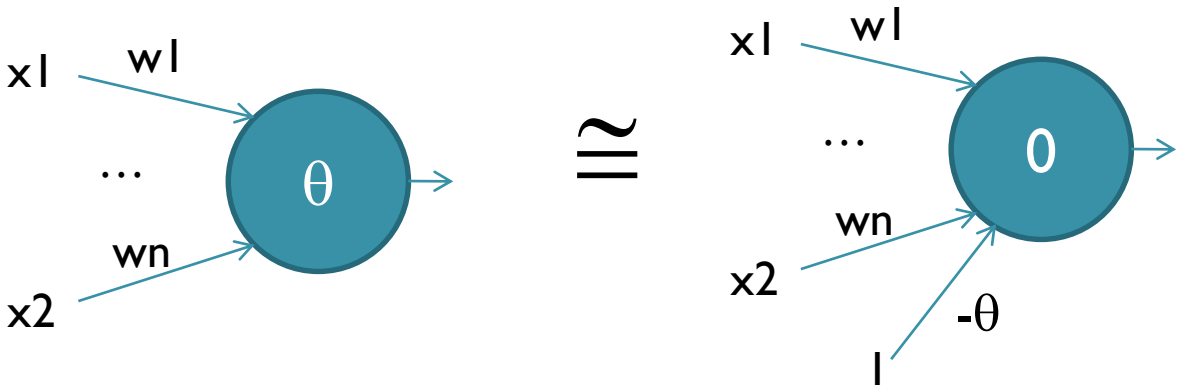
Beispiel

Definieren Sie ein einschichtiges Netz gewichteter Neuronen mit relativer Hemmung, das folgende Beispiele richtig klassifiziert

Bsp Nr	Anz-Räder	PS	Konzept
1	4	100	Auto
2	2	90	Motorrad
3	2	2	E-Fahrrad

2.3 Künstliche neuronale Netze

Normierung des Schwellwerts



Zur besseren Handhabung beim Lernen wird der Schwellenwert als ein „normales“ Gewicht betrachtet

2.3 Künstliche neuronale Netze

Das Modell von McCulloch und Pitts

Beispiel

Modifizieren Sie das Netz aus dem Letzten Beispiel, so dass als Schwellenwert immer 0 verwendet wird

Bsp Nr	Anz-Räder	PS	Konzept
1	4	100	Auto
2	2	90	Motorrad
3	2	2	E-Fahrrad

2.3 Künstliche neuronale Netze

Das Modell von McCulloch und Pitts

Mächtigkeit

Satz: Gewichtete Netze mit relativ hemmenden Verbindungen und normiertem Schwellwert können in äquivalente ungewichtete McCulloch-Pitts-Netze transformiert werden und umgekehrt.

(und hat damit die selben positiven Eigenschaften)

2.3 Künstliche neuronale Netze

Perzeptron vs. McCulloch-Pitts

- Ein Perzeptron ist eine gewichtete, normierte McCulloch-Pitts-Zelle
- McCulloch-Pitts-Netze sind schwer trainierbar
  - Netz-Topologie ???
  - Schwellenwerte ???
- Perzeptron:
  - Topologie ???
  - Gewichte an den Eingängen (inkl. Schwellenwert) werden gelernt

2.3 Künstliche neuronale Netze

Das Perzeptron

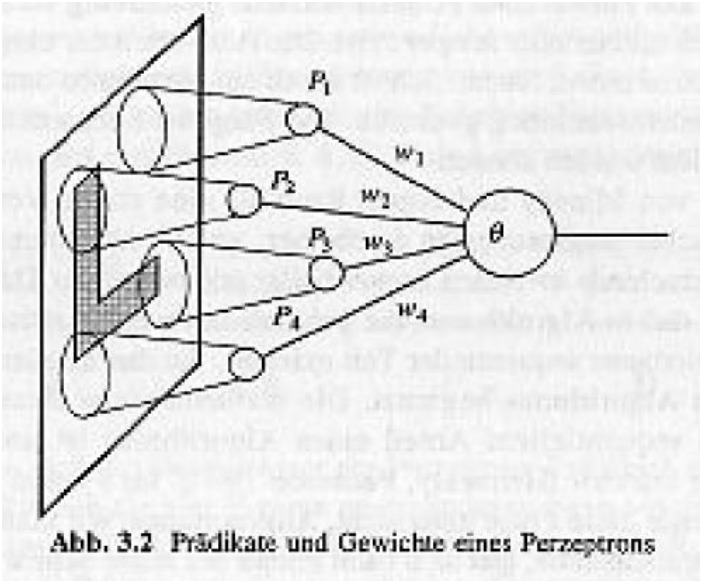


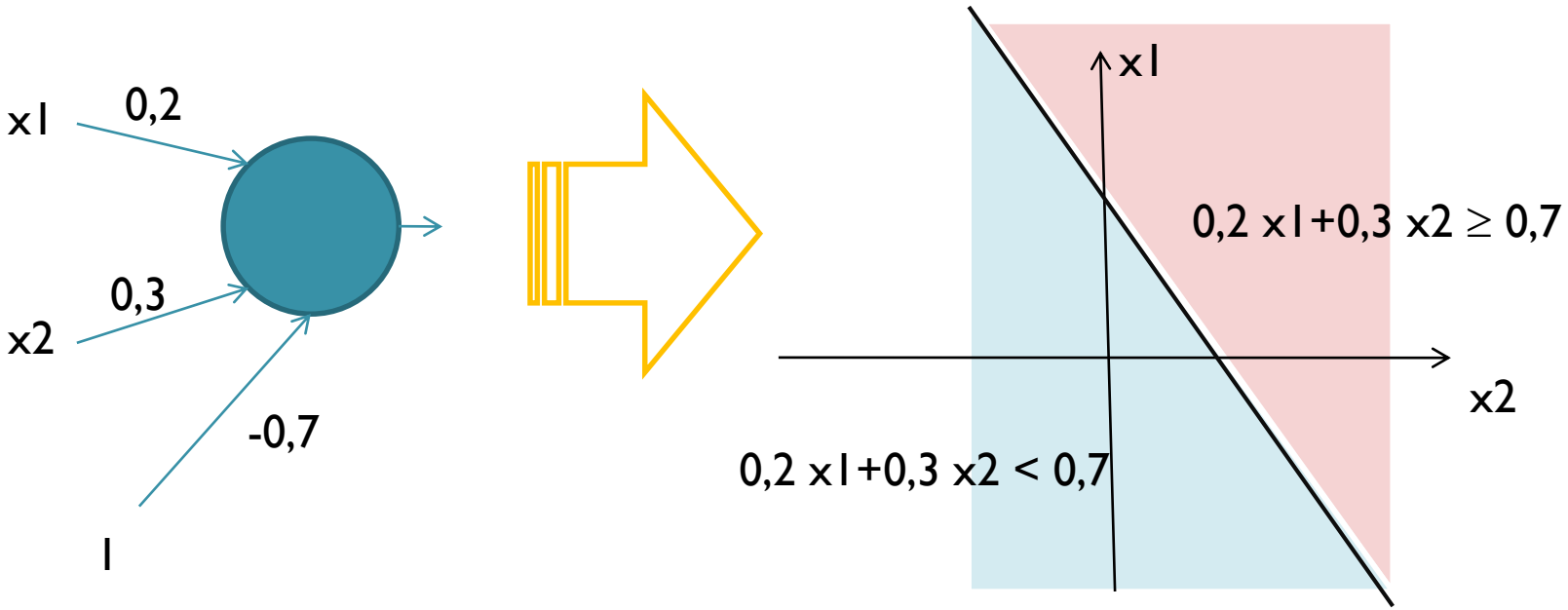
Abb. 3.2 Prädikate und Gewichte eines Perzeptrons

- Prädikate P_i sind beliebig komplex und liefern 0 oder 1
- Die Zelle feuert gdw. $\Sigma\, w_i P_i + \theta \geq 0$

2.3 Künstliche neuronale Netze

Das Perzeptron

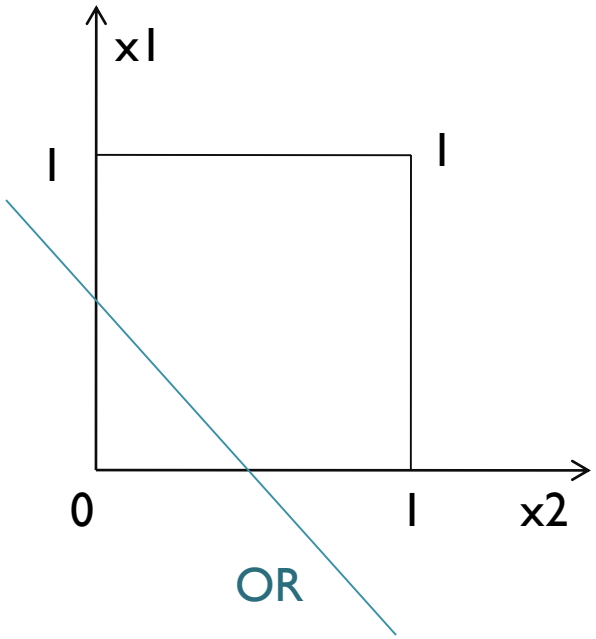
Geometrische Interpretation

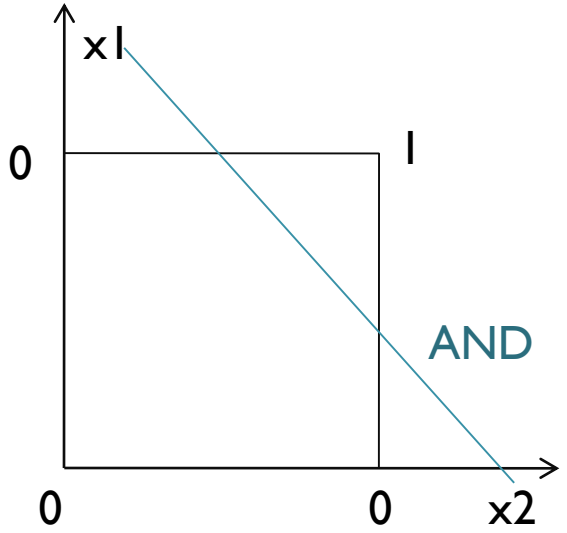


2.3 Künstliche neuronale Netze

Das Perzeptron

Fallstudie: logische Funktionen

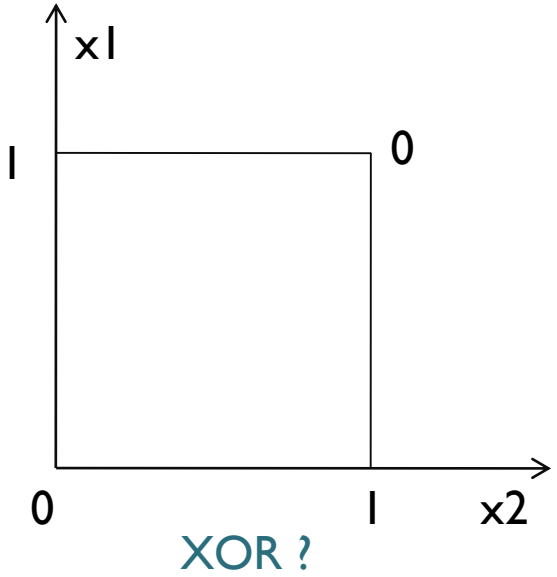




2.3 Künstliche neuronale Netze

Das Perzeptron

XOR-Problem



Satz: Die XOR-Funktion kann nicht mit einem einzelnen Perzeptron berechnet werden

2.3 Künstliche neuronale Netze

Das Perzeptron

Lineare Entscheidungsfunktionen

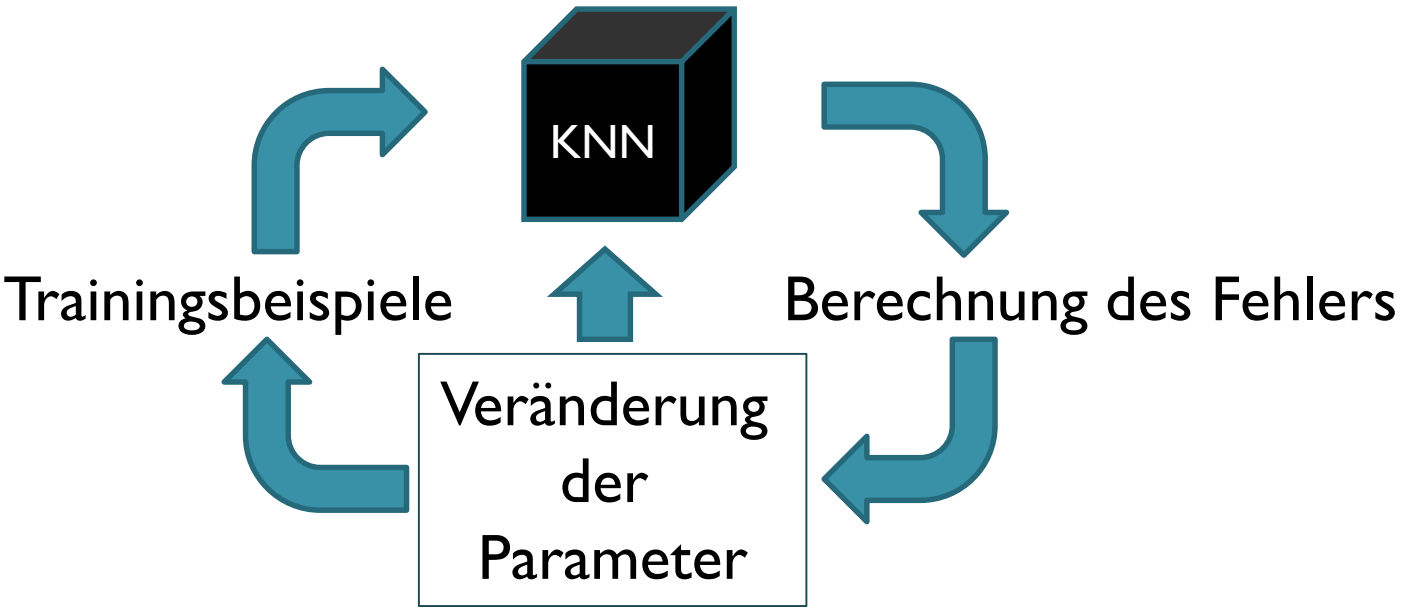
Def: Zwei Mengen A und B von Punkten im n-dimensionalen Raum sind linear trennbar, falls $n+1$ reelle Zahlen $w_1 \ldots w_{n+1}$ existieren, so dass

- $\forall (x_1 \ldots x_n) \in A \,.\, \Sigma\, w_i x_i \geq w_{n+1}$
- $\forall (x_1 \ldots x_n) \in B \,.\, \Sigma\, w_i x_i < w_{n+1}$

2.3 Künstliche neuronale Netze

Das Perzeptron

Lernalgorithmus [Rosenblatt 60]



Überwachtes Lernen

2.3 Künstliche neuronale Netze

Das Perzeptron

Lernalgorithmus [MinskyPapert69]

Eingabe:

P: Menge positiver Beispiele

N: Menge negativer Beispiele

im n-dimensionalen Raum

Ausgabe:

Gewichtsvektor $w = (w_1, w_2, \ldots, w_n)$, der P und N linear trennt

2.3 Künstliche neuronale Netze

Das Perzeptron

Lernalgorithmus [MinskyPapert69]

Satz: Wenn P und N endlich und linear trennbar sind verändert der Algorithmus w nur in einer endlichen Zahl von Iterationen.

d.h. Man kann (spätestens) abbrechen, wenn kein Punkt mehr falsch klassifiziert wurde

2.3 Künstliche neuronale Netze

Das Perzeptron

Lernalgorithmus [MinskyPapert69]

Setze w zufällig

solange ! Abbruch()

wähle $x \in P \cup N$ zufällig

wenn $x \in P \wedge w * x \leq 0$ dann w := w+x

wenn $x \in N \wedge w * x > 0$ dann w := w-x

// sonst war es richtig →mache nichts

2.3 Künstliche neuronale Netze

Das Perzeptron

δ-Regel

- Ziel: schnellere Konvergenz
- Problem: Beim Perzeptron werden oft zu kleine oder zu große Schritte gemacht
- Grund: einfache Verrechnung der Merkmale mit Gewichten
- Die δ-Regel passt den Schritt an den beobachteten Fehler an

2.3 Künstliche neuronale Netze

Das Perzeptron

δ-Regel

Wenn $x \in P$ falsch klassifiziert wurde, dann

galt: $w * x \leq 0$

der Fehler ist also $\delta = -w * x$

die Änderung mit δ-Regel ist dann

$$w := w+(\delta*\varepsilon)x$$

wobei ε eine positive Zahl ist

Analog bei $x \in N$: $w := w-(\delta*\varepsilon)x$

2.3 Künstliche neuronale Netze

Das Perzeptron

Mächtigkeit

- Ein einzelnes Perzeptron kann linear trennbare Funktionen lernen
- zB von 16 binären boolschen Funktionen sind 14 linear trennbar (XOR nicht)

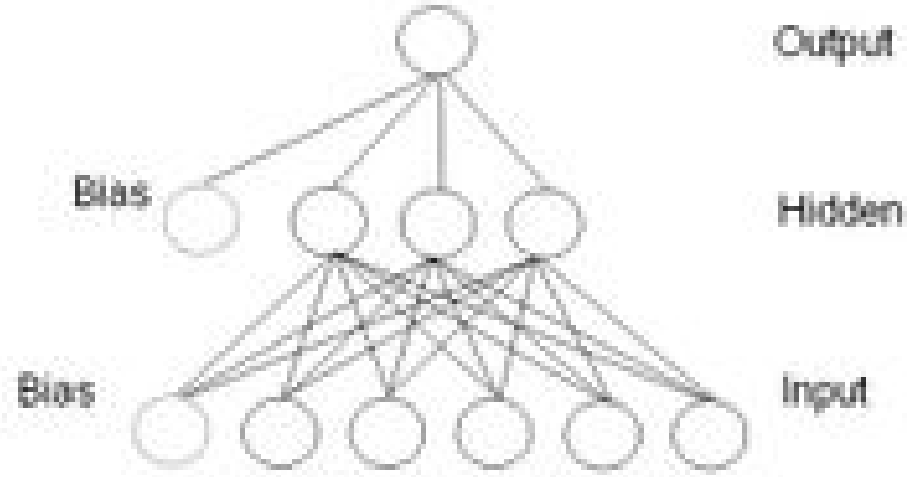
ABER:

Wenn n → ∞ dann geht die Zahl der linear trennbaren Funktionen gegen 0

2.3 Künstliche neuronale Netze

Mehrschichtige Netze

- Ziel: mehr Probleme lösen (zB XOR)
- Idee: Mehrere Neuronen bilden ein Netz
- Ausgaben der n-ten Schicht sind Eingaben der n+1-ten Schicht



2.3 Künstliche neuronale Netze

Netzarchitekturen

Def: Eine **Netzarchitektur** (E,N,A,T) ist gegeben durch die Mengen

- E von Eingabestellen,
- N von Neuronen,
- A von Ausgabestellen
- T von gewichteten Kanten

Eine gewichtete Kante (k_i,k_j,w_{ij}) ist gegeben durch $k_i \in E \cup N$, $k_j \in N \cup A$, und das Gewicht $w_{ij} \in \mathfrak{R}$

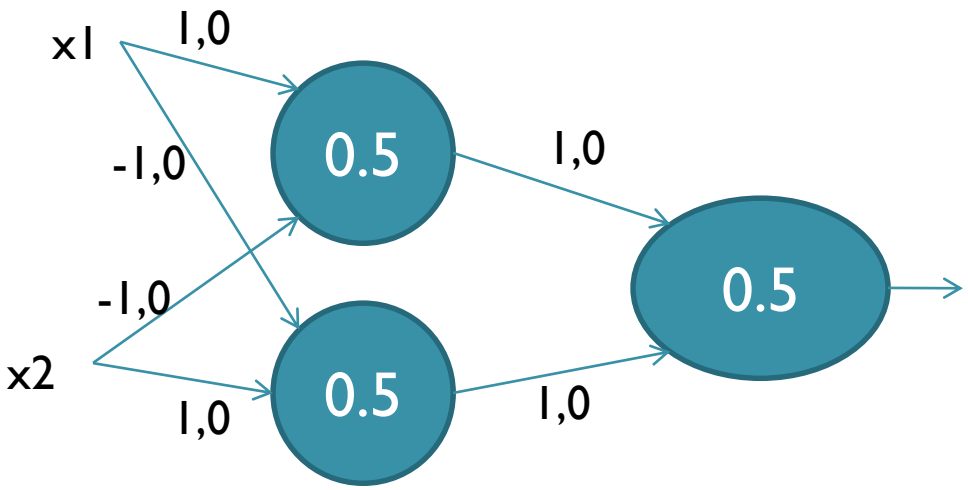
2.3 Künstliche neuronale Netze

Schichtenarchitektur

Def: Eine Netzarchitektur ist eine **Schichtenarchitektur** wenn die Menge N in m $N_1, N_2, \ldots, N_m$ Gruppen aufgeteilt werden kann, wobei Eingabestellen ausschließlich an N_1 und Ausgabestellen ausschließlich an N_m angeschlossen werden.

2.3 Künstliche neuronale Netze

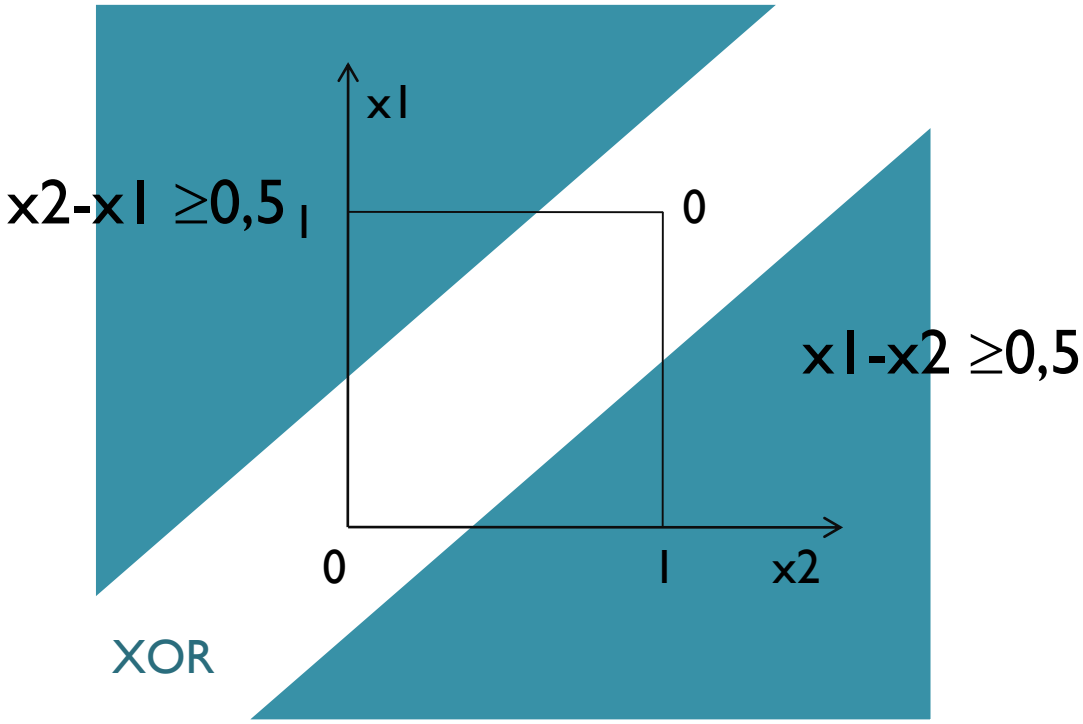
Nochmals: das XOR-Problem



Lösung in einem zweischichtigen Netz aus Perzeptrons

2.3 Künstliche neuronale Netze

Nochmals: das XOR-Problem→ gelöst



Übungsaufgabe 1

1. Topologie des Netzes
   1. Einschichtig!
   2. Merkmale aus Vektoren an Eingabeschicht
   3. Kodierung der Konzepte an Ausgabeschicht

2.3 Künstliche neuronale Netze

Lernen im mehrschichtigen Netzen

- Diverse Algorithmen
  - Backpropagation (machen wir) und Varianten
  - Competitive Learning, Kohonennetze (Methoden unüberwachten Lernens)
- Herausforderungen
  - Problem der Zuordnung von Fehlern zu einzelnen Neuronen
  - Problem der Einstellung der Gewichte nach diesem Fehler

2.3 Künstliche neuronale Netze

Backpropagation

- Berechnung des Fehlers (δ-Regel)
- Propagierung des Fehlers auf die einzelnen Neuronen
  - Suche nach Minimum der Aktivierungsfunktion für den gemessenen Fehler δ **an jedem Neuron**
  - Abstieg am Gradienten
- Die Kombination der Gewichte, die den kleinsten Fehler verursacht ist anzunähern

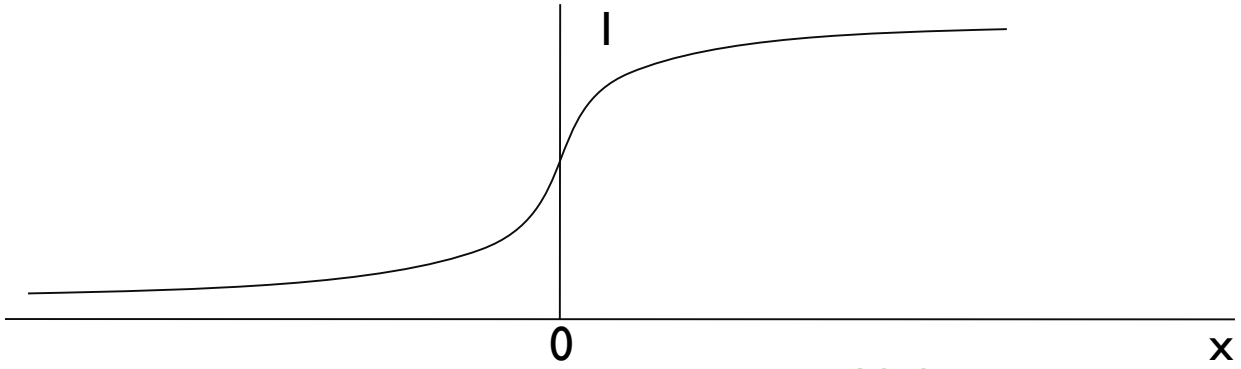
2.3 Künstliche neuronale Netze

Eine Differenzierbare Aktivierungsfunktion

Die Sigmoide: $s_c: \mathcal{R} \rightarrow]0,1[$

$$s_c(x) = \frac{1}{1+e^{-cx}}$$

Dabei ist c die „Temperaturkonstante“



2.3 Künstliche neuronale Netze

Bedeutung der Temperaturkonstante

- Bei $c=1$ schreiben wir nur „s“ statt „s_c“
- Bei $c \rightarrow \infty$ nähert sich s_c der Treppenfunktion
- Bei kleinen c erhält man eine kontinuierliche „Zerlegung“ anstatt Trennebenen → unscharfe Klassifizierung

2.3 Künstliche neuronale Netze

Ableitung der Sigmoide (mit c=1)nach x

$$\frac{ds(x)}{dx} = \frac{e^{-x}}{(1+e^{-x})^2} = s(x)(1-s(x))$$

2.3 Künstliche neuronale Netze

Lernen mit Backpropagation

1. Initialisiere die Gewichte zufällig
2. Berechne den Fehler (feedforward)
3. Propagiere Fehler zurück bis zur Eingabeschicht (backpropagation)
4. Korrigiere Gewichte um den Fehler zu minimieren, beginne bei Eingabeschicht

2.3 Künstliche neuronale Netze

Lernen mit Backpropagation

1. initialisierung

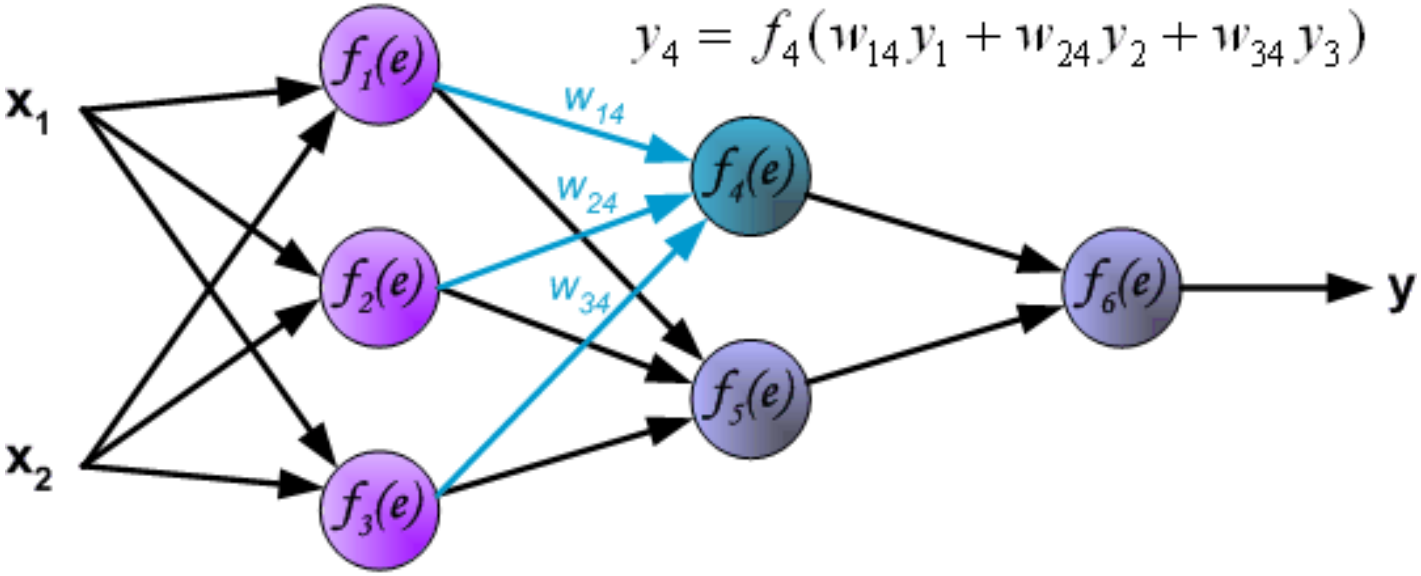
…

2.3 Künstliche neuronale Netze

Lernen mit Backpropagation

2. feedforward

Klassifizierung des Beispiels



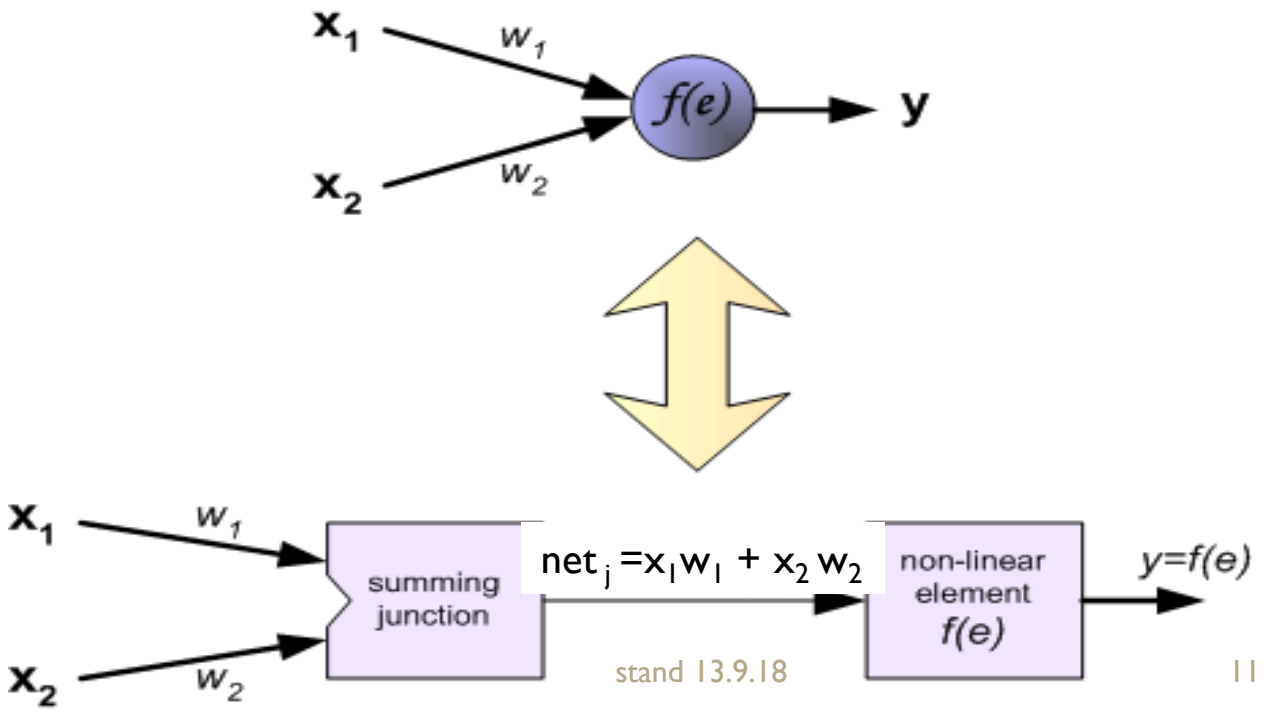
2.3 Künstliche neuronale Netze

Lernen mit Backpropagation

2. feedforward

Abspeichern der Werte f‘(net $_j$) in den Neuronen

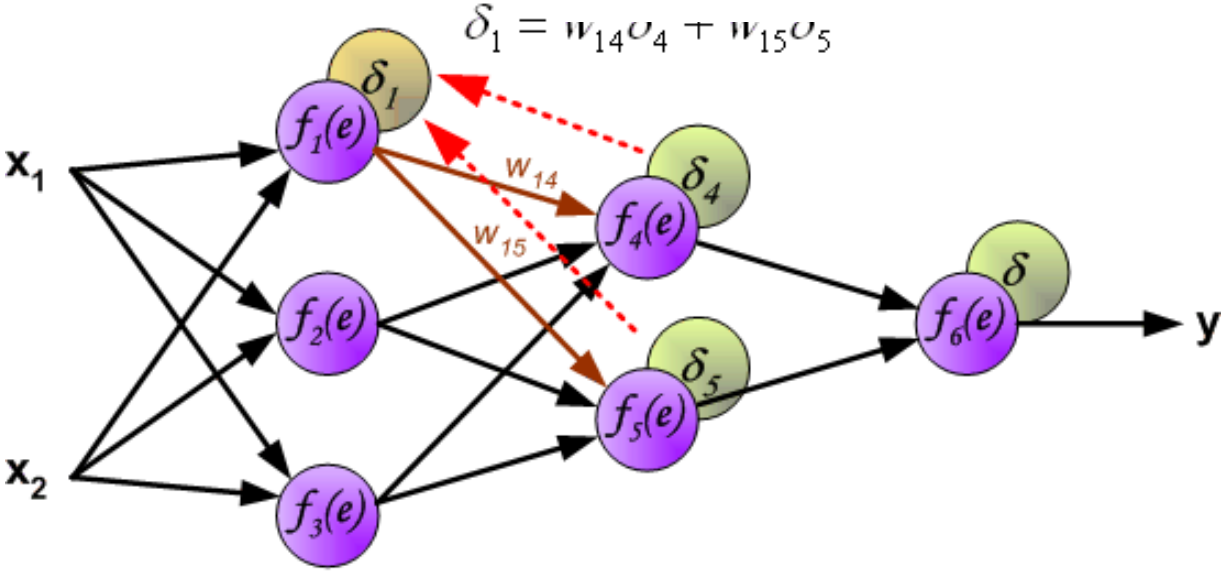
f‘ ist die Ableitung der Aktivierungs-funktion, net $_j$ die gewichtete Summe der Eingaben



2.3 Künstliche neuronale Netze

Lernen mit Backpropagation

3. Propagierung des Fehlers

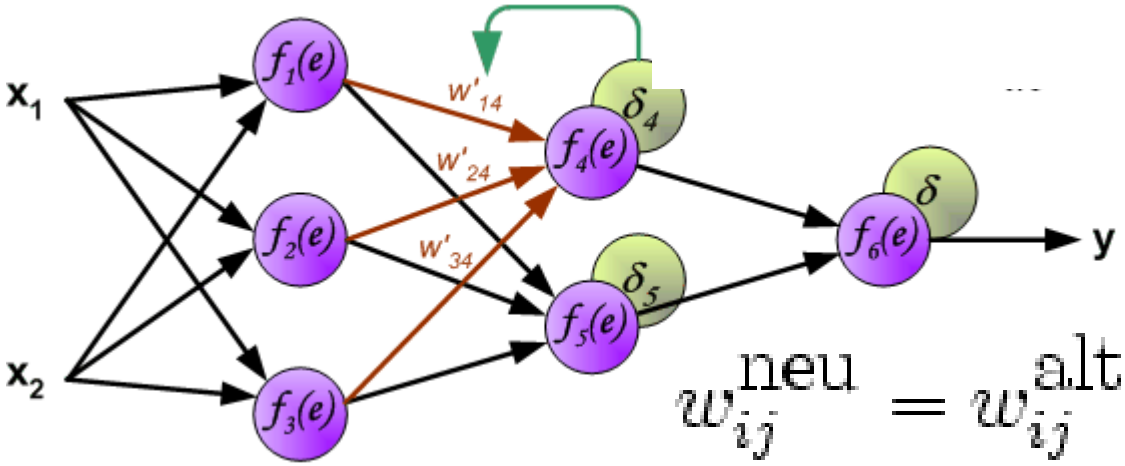


$$\delta_j = \begin{cases} f'(\text{net}_j)(t_j - o_j) & \text{falls } j \text{ Ausgabeneuron ist,} \\ f'(\text{net}_j) \sum_k \delta_k w_{jk} & \text{falls } j \text{ verdecktes Neuron ist.} \end{cases}$$

2.3 Künstliche neuronale Netze

Lernen mit Backpropagation

4. Korrektur der Gewichte



$$w_{ij}^{\text{neu}} = w_{ij}^{\text{alt}} + \Delta w_{ij}$$

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} = \eta \delta_j x_i$$

Wobei η die Lernrate beschreibt

Themen KI

1. Einführung KI
2. Maschinelles Lernen
   1. Lernende Klassifizierungssysteme
   2. Evaluierung lernender Systeme
   3. Lernverfahren
3. Problemlösen
   1. Vollständige Suche
   2. Unvollständige Suche