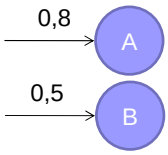
Kapitel 20
Statistische Lernmethoden

Bayes'sche Netze

Bayes'sche Netze
Grundlagen

- Wahrscheinlichkeit, dass ein Ereignis A oder ein Ereignis B eintritt

$$P(A) = 0{,}8$$
$$P(B) = 0{,}5$$



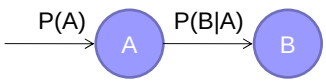
- Wahrscheinlichkeit für das gleichzeitige Eintreten der Ereignisse A und B

$$P(A \wedge B) = P(A) \cdot P(B)$$
$$P(A \wedge B) = 0{,}8 \cdot 0{,}5 = 0{,}4$$

Bayes'sche Netze
Grundlagen

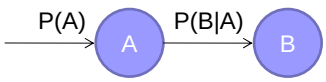
- Bedingte Wahrscheinlichkeit

$$P(B \mid A) = \frac{P(A \wedge B)}{P(A)}$$
$$P(B \mid A) = \frac{P(B) \cdot P(A \mid B)}{P(A)}$$



- Pfadregel
  - Die Wahrscheinlichkeiten längs eines Pfades werden miteinander multipliziert

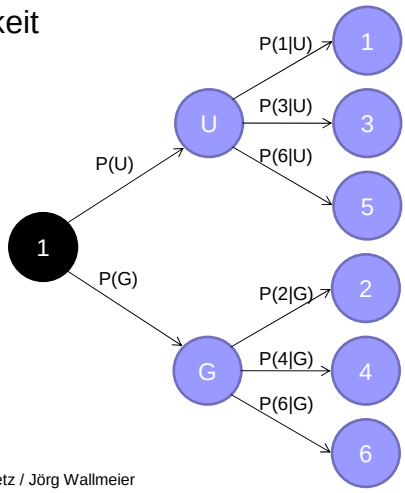
$$P(B) = P(A) \cdot P(B \mid A)$$



Bayes'sche Netze
Grundlagen

- Bedinge Wahrscheinlichkeit

$$P(G) = 1/2$$
$$P(2) = 1/6$$
$$P(2 \mid G) = \frac{P(2 \wedge G)}{P(G)}$$
$$P(2 \mid G) = \frac{P(2) \cdot P(G \mid 2)}{P(G)}$$
$$P(2 \mid G) = \frac{(1/6) \cdot (1/1)}{1/2}$$
$$P(2 \mid G) = \frac{1}{3}$$



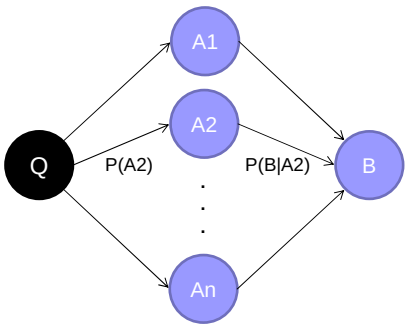
Bayes'sche Netze
Grundlagen

- Totale Wahrscheinlichkeit

$$P(B) = \sum_{i=1}^{n} (P(Ai) \cdot P(B \mid Ai))$$

- Bayes'sche Formel

$$P(A2 \mid B) = \frac{P(Q, A2, B)}{P(B)}$$
$$P(A2 \mid B) = \frac{P(A2) \cdot P(B \mid A2)}{\sum_{i=1}^{n} (P(Ai) \cdot P(B \mid Ai))}$$

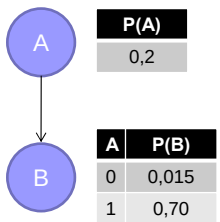


Bayes'sche Netze
Aufbau eines Bayes'sche Netzes

- Bayes'sche Netze kombinieren Graphentheorie und Wahrscheinlichkeitstheorie miteinander
- Bayes'sches Netzwerk ist ein gerichteter azyklischer Graph, der die Wahrscheinlichkeitsverteilung verschiedener voneinander abhängiger Ereignisse modelliert
- Knoten stellen dabei verschiedene Ereignisse (Zufallsgrößen) dar
- Die gerichteten Kanten symbolisieren die Abhängigkeiten der Ereignisse untereinander.

Bayes'sche Netze
Aufbau eines Bayes'sche Netzes

- Mit jedem Knoten ist eine bedingte Wahrscheinlichkeits-Tabelle, die CPT (Conditional Probability Table) assoziiert, die den Effekt der Eltern auf den Knoten quantifiziert



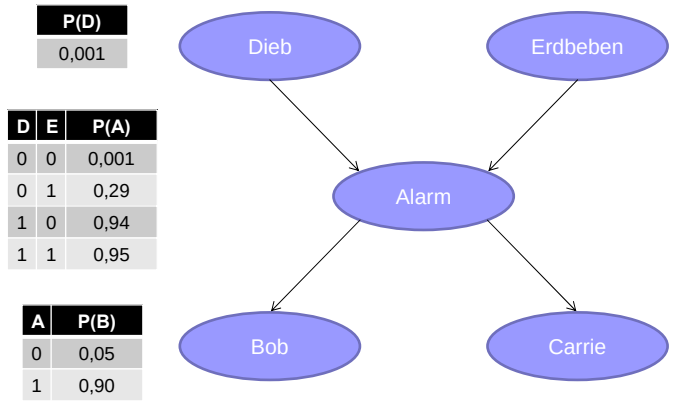
P(A)
0,2

A	P(B)
0	0,015
1	0,70

$$P(Knoten) = P(Knoten \mid Eltern(Knoten))$$
$$P(B) = P(B \mid A)$$

$$P(A=1) = 0{,}2 \qquad P(B \mid A) = 0{,}70$$
$$P(A=0) = 0{,}8 \qquad P(B \mid \overline{A}) = 0{,}015$$

Bayes'sche Netze
Beispiel - Alarmanlage



P(D)
0,001

P(E)
0,002

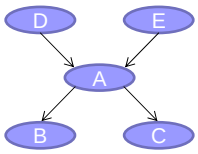
D	E	P(A)
0	0	0,001
0	1	0,29
1	0	0,94
1	1	0,95

A	P(B)
0	0,05
1	0,90

A	P(C)
0	0,01
1	0,70

Bayes'sche Netze
Beispiel - Alarmanlage

$$P(B, C, A, \overline{D}, \overline{E})$$
$$= P(B \mid A) \cdot P(C \mid A) \cdot P(A \mid \overline{B}, \overline{E}) \cdot P(\overline{D}) \cdot P(\overline{E})$$
$$= 0.90 * 0.70 * 0.001 * 0.999 * 0.998$$
$$= 0{,}00062811126$$



P(D)
0,001

P(E)
0,002

A	P(B)
0	0,05
1	0,90

A	P(C)
0	0,01
1	0,70

D	E	P(A)
0	0	0,001
0	1	0,29
1	0	0,94
1	1	0,95

Bayes'sche Netze
Anwendung Bayes’scher Netze

- Anwendungsbereiche
  - Diagnose von Krankheiten
  - Regelung betriebswirtschaftlicher Prozesse
  - Vorhersage von DNA Strukturen
  - Wettervorhersagen
  - Wissensrepräsentation, Fehlerdiagnose
- Oft ist es relativ einfach, die kausalen Beziehungen (d.h. die Topologie) eines Netzes zu bestimmen, dagegen ziemlich schwer, die Wahrscheinlichkeits-tabellen anzugeben

Bayes'sche Netze
Anwendung Bayes’scher Netze

- PATHFINDER-System
  - sehr bekanntes und leistungfähiges Netzwerk zur Diagnose von Lymphknotenerkrankungen
  - umfasst 100 Symptome und 14.000 Wahrschein-lichkeiten, um 100 Krankheiten zu diagnostizieren
  - das Systen ist angeblich mittlerweile besser in der Diagnose als die besten Experten
  - Aber dieses Netz wurde von Experten erstellt und von diesen mit Wahrscheinlichkeitstabellen gefüllt.

Bayes'sche Netze
Lernen von Bayes'schen Netzen

- Beim Lernen von Bayes'schen Netzen muss man im wesentlichen vier Fälle unterscheiden

STRUKTUR	TRAININGSDATEN	VERFAHREN
Bekannt	Vollständig	ML-Verfahren
Bekannt	Unvollständig	EM-Verfahren
Unbekannt	Vollständig	Lokale Suche bzw. Suchen im Modellraum
Unbekannt	Unvollständig	EM + Suchen im Modellraum

Bayes'sche Netze
Lernen von Bayes'schen Netzen

- Lernen der bedingten Wahrscheinlichkeiten
  - Vollständige Trainingsdaten
    - Wahrscheinlichkeiten können lokal für jede der Eltern-Kind-Variablenmenge ermittelt werden
      → ML-Verfahren
  - Unvollständige Trainingsdaten
    - Häufigste Situation in der Praxis
    - Struktur von Experten spezifiziert, aber bedingte Wahrscheinlichkeiten erlernt werden
      → EM-Verfahren

Bayes'sche Netze
Lernen von Bayes'schen Netzen

- Das ML-Parameterlernen (Maximum Likelihood)
  - Daten vollständig
  - ML-Parameterlernen ist denkbar einfach

    → man zählt einfach ab

    $$\theta_{i,j,k} = P(X_i = k | \mathrm{par}(X_i) = j)$$

    (Wahrscheinlichkeit, dass X_i im Zustand k ist, wenn die Eltern in Zustand j sind)
  - ML Schätzung

$$\theta^{ML}_{i,j,k} = \frac{N_{i,j,k}}{\sum_k N_{i,j,k}}$$

Hierbei ist $N_{i,j,k}$ die Anzahl der Trainingsdaten, bei denen $X_i = k, \mathrm{par}(X_i) = j$

Bayes'sche Netze
Lernen von Bayes'schen Netzen

- Das EM-Verfahren
  - Daten unvollständig
  - Iteratives Verfahren zur Optimierung des ML-Verfahrens
  - Am Anfang werden die „beobachtbaren" Parameter des Modells geschätzt (ML-Parameterschätzung), fehlende werden frei geschätzt

Bayes'sche Netze
Lernen von Bayes'schen Netzen

- Das EM-Verfahren (2)
  - Dann werden die folgenden Schritte iteriert bis das Modell vollständig ist:
    - Estimation Step: Berechnet die Verteilung der fehlenden Beobachtungen
    - Maximation Step: Berechnung der ML-Parameter für die beobachtbaren Parameter bei gegebener Verteilung der unbeobachtbaren Parameter

$$\theta_{i,j,k}^{ML} = \frac{E(N_{i,j,k})}{\sum_k E(N_{i,j,k})}$$

Bayes'sche Netze
Fazit

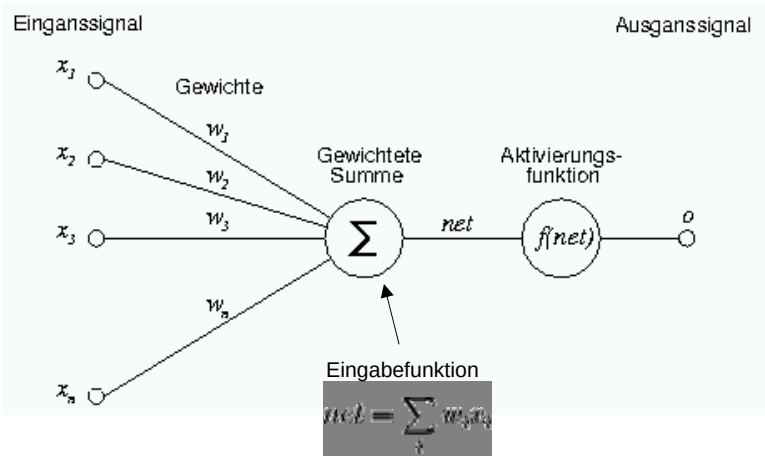
- Bayes'sche Netze kombinieren Graphentheorie und Wahrscheinlichkeitstheorie miteinander
- Knoten stellen dabei verschiedene Ereignisse dar
- gerichteten Kanten symbolisieren die Abhängigkeiten
- zu jedem Knoten eine Wahrscheinlichkeits-Tabelle
- relativ einfach, die Topologie des Netzes zu bestimmen
- schwieriger die Wahrscheinlichkeitstabellen anzugeben
- Lernen der bedingten Wahrscheinlichkeiten
  - ML-Parameterlernen
  - EM-Verfahren

Neuronale Netzte

Neuronale Netzte
Grundlagen

- Künstliche neuronale Netze (KNN) basieren auf der Vernetzung von Neuronen
- Je nach Aufgabe andere Topologie des Netzes
- Nach der Konstruktion folgt die Trainingsphase
- Anwendung
  - Text-, Bild- und Gesichtserkennung, etc.
  - Frühwarnsysteme
  - Prozessoptimierung
  - Robotik und Intelligente Agenten
  - Spiele

Neuronale Netzte
Das künstliche Neuron



Neuronale Netzte
Das künstliche Neuron - Eingangssignale

- Eingangssignale x_i
  - Daten von der Umgebung oder dem Ausgang eines anderen künstlichen Neurons
  - Unterschiedliche Netzwerkmodelle erlauben dabei unterschiedliche Wertebereiche
  - Typische Wertebereiche sind die reellen Zahlen
    - das Intervall `[0,1]`
    - oder die diskreten Werte `{0,1}`

Neuronale Netzte
Das künstliche Neuron - Gewichte

- Gewichte w_i
  - Jeder Verbindung in einem Neuronalen Netz ist eine reelle Zahl als Gewicht zugeordnet
  - Gewicht beschreibt die Stärke der Verbindung
  - In den Verbindungsgewichten ist das Wissen des Neuronalen Netzes gespeichert
  - Abhängig von den Vorzeichen der Gewichte:
    - Negativ: Eingabe hemmend (*inhibitorisch*)
    - Positiv: Eingabe erregend (*exhibitorisch*)

Neuronale Netzte
Das künstliche Neuron - Eingabefunktion

- Eingabefunktion
  - berechnet anhand der Wichtung der Eingaben die Netzeingabe des Neurons (Netto-Input)

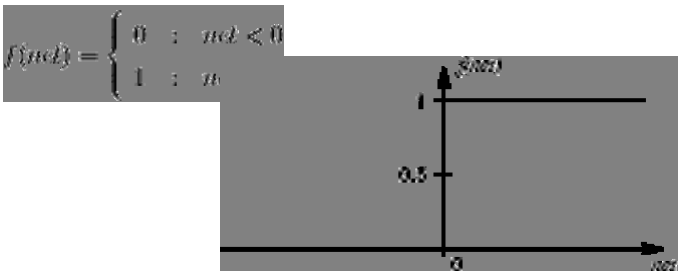
$$net = \sum_i w_i x_i$$

Neuronale Netze
Das künstliche Neuron - Aktivierungsfunktion

- Aktivierungsfunktion `f(net)`
  - bestimmt, abhängig vom Netto-Input `net`, den Output o des Neurons
  - Output kann das Eingangssignal für ein anderes Neuron sein
  - Output kann das Ausgangssignal des Neuronalen Netzes
  - Als Aktivierungsfunktion wird meist eine der nachfolgenden Grundtypen verwendet ⟶

Neuronale Netze
Das künstliche Neuron - Aktivierungsfunktion

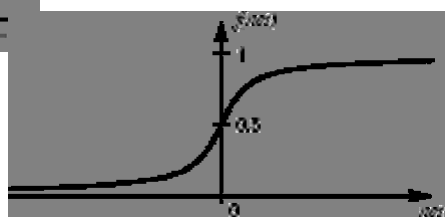
- Schwellwertfunktion
  - Neuronen (Gatter), die diese Aktivierungsfunktion verwenden, nennt man Schwellwertgatter



Neuronale Netze
Das künstliche Neuron - Aktivierungsfunktion

- Sigmoide Funktion
  - Unterschied zur Schwellwertfunktion:
    - kontinuierliches Ausgangssignal u. differenzierbar

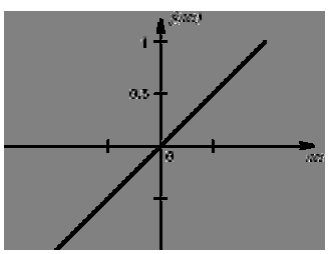
$$f(net) = \frac{1}{1+e^{-}}$$



Neuronale Netze
Das künstliche Neuron - Aktivierungsfunktion

- Lineare Funktion
  - einfache und wenig mächtige Aktivierungsfunktion
  - Hintereinanderschaltung mehrerer linearer Gatter können nur die selbe Funktion berechnen

$$f(net) = net$$



Neuronale Netzte
Netzwerkstrukturen

- Feedforward-Netze
  - azyklische Netze oder Netze ohne Rückkopplung
  - Funktion seiner aktuellen Eingabe
  - außer den Gewichtungen keinen internen Zustand
- Recurrent-Netze
  - zyklische Netze oder Netze mit Rückkopplung
  - gibt seine Ausgaben zurück an eigene Eingaben
  - Netze können somit ein Kurzzeitgedächtnis haben
  - Netze sind wesentlich schwerer zu verstehen

Neuronale Netzte
Einschichtige neuronale Netzwerke (ohne Rückkopplung)

- Perceptron-Netzwerk oder Perzeptron
  - jede Gewichtung bezieht sich nur auf eine einzelne Ausgabe
- Eigenschaften des künstlichen Neurons
  - Eingangssignale sind reellen Zahlen
  - Gewichte sind reellen Zahlen
  - Als Aktivierungsfunktion wird eine Schwellwert-funktion mit variabler Schwelle (Threshold)
  - Ausgangssignal ist somit diskret aus der Menge `{0,1}`

Neuronale Netzte

Einschichtige neuronale Netzwerke (ohne Rückkopplung)

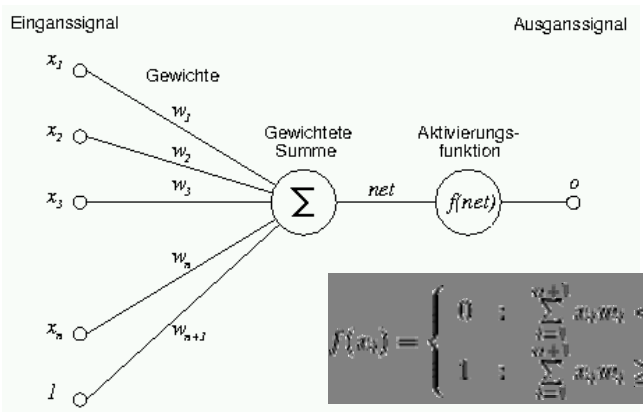
- Vereinfachung der Aktivierungsfunktion

$$f(x_i) = \begin{cases} 0 & : \sum_{i=1}^{n} x_i w_i < \theta \\ 1 & : \sum_{i=1}^{n} x_i w_i \geq \theta \end{cases} \qquad f(x_i) = \begin{cases} 0 & : \sum_{i=1}^{n} x_i w_i - \theta < 0 \\ 1 & : \sum_{i=1}^{n} x_i w_i - \theta \geq 0 \end{cases}$$

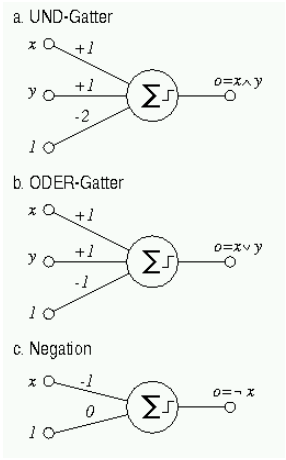
- der Threshold θ kann als zusätzliches Gewicht für einen zusätzlichen Eingang konstant auf -1 liegen (oft auch als $-\theta$ konstant auf 1)
- das zusätzliche Gewicht wird oft auch als Bias-Gewicht bezeichnet

Neuronale Netzte

Einschichtige neuronale Netzwerke (ohne Rückkopplung)



Neuronale Netzte
Einschichtige neuronale Netzwerke (ohne Rückkopplung)



Netzeingabe aus Übertragungsfunktion:
net = 1 * x + 1 * y + (-2) * 1 = x + y - 2

Netzeingabe aus Übertragungsfunktion:
net = 1 * x + 1 * y + (-1) * 1 = x + y - 1

Netzeingabe aus Übertragungsfunktion:
net = -1 * x + 1 * 0 = -x

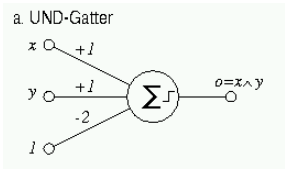
Neuronale Netzte
Einschichtige neuronale Netzwerke (ohne Rückkopplung)

- „Perceptron criterion function“ - Lernalgorithmus
  - Lernregel für Perzeptrons mit binären Ein- und Ausgabewerten
  - Zu Beginn werden die Gewichte zufällig initialisiert
  - Ist die Ausgabe 1 (bzw. 0) und soll auch den Wert 1 (bzw. 0) annehmen, dann tue nichts
  - Ist die Ausgabe 0, soll aber den Wert 1 annehmen, dann werden die Gewichte inkrementiert
  - Ist die Ausgabe 1, soll aber den Wert 0 annehmen, dann werden die Gewichte dekrementiert

Neuronale Netzte

Einschichtige neuronale Netzwerke (ohne Rückkopplung)

- Einfaches Beispiel: UND-Gatter



$w_1 = 1,0$
$w_2 = 1,0$
$w_3 = -2,0$

x	y	T	net	f(net)
0	0	1	-2	0
0	1	1	-1	0
1	0	1	-1	0
1	1	1	0	1

$$net = \sum_i w_i x_i$$

$$f(net) = \begin{cases} 0 & ; \; net < 0 \\ 1 & ; \; net \geq 0 \end{cases}$$

Neuronale Netzte

Einschichtige neuronale Netzwerke (ohne Rückkopplung)

- Schritt 1: Initialisierung mit Zufallswerten

x	y	T	net	f(net)
0	0	1	0,1	1
0	1	1	-0,2	0
1	0	1	0,7	1
1	1	1	0,4	1

$w_1 = 0,6$
$w_2 = -0,3$
$w_3 = 0,1$

- Schritt 2: Anpassung der Gewichte

x	y	T	net	f(net)
0	0	1	-1,9	0
0	1	1	-2,3	0
1	0	1	-2,4	0
1	1	1	-2,7	0

$w_1 = 0,6 - 0 - 1 = -0,4$
$w_2 = -0,3 - 0 - 0 = -0,3$
$w_3 = 0,1 - 1 - 1 = -1,9$

Neuronale Netzte
Einschichtige neuronale Netzwerke (ohne Rückkopplung)

- Schritt 3: Anpassung der Gewichte

x	y	T	net	f(net)
0	0	1	-0,9	0
0	1	1	-0,2	0
1	0	1	-0,3	0
1	1	1	0,4	1

$w_1 = -0,4 + 1 = 0,6$
$w_2 = -0,3 + 1 = 0,7$
$w_3 = -1,9 + 1 = -0,9$

Neuronale Netzte
Einschichtige neuronale Netzwerke (ohne Rückkopplung)

- Grenzen: Das XOR-Problem
  - Soll ein Perzeptron das XOR-Problem lösen, so müssen 3 Gewichte w1,w2,w3 existieren, welche nachfolgende Ungleichungen erfüllt sind

$$x = 0, y = 0: \quad 0 * w_1 + 0 * w_2 + 1 * w_3 < 0 \quad \text{Output } o = 0$$
$$x = 0, y = 1: \quad 0 * w_1 + 1 * w_2 + 1 * w_3 \geq 0 \quad \text{Output } o = 1$$
$$x = 1, y = 0: \quad 1 * w_1 + 0 * w_2 + 1 * w_3 \geq 0 \quad \text{Output } o = 1$$
$$x = 1, y = 1: \quad 1 * w_1 + 1 * w_2 + 1 * w_3 < 0 \quad \text{Output } o = 0$$

Neuronale Netzte

Einschichtige neuronale Netzwerke (ohne Rückkopplung)

- Grenzen: Das XOR-Problem
  - Vereinfacht kann dies angeschrieben werden als:

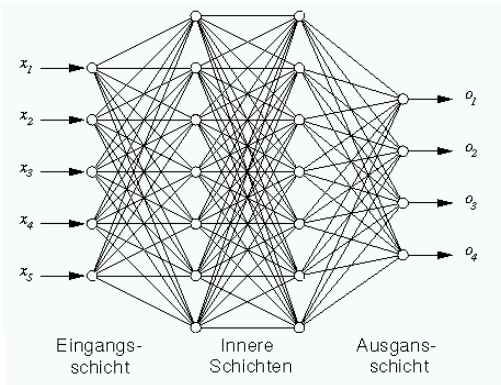
$$
\begin{array}{lll}
x=0, y=0 & ; & w_3 < 0 \quad \text{Output } o=0 \\
x=0, y=1 & ; & w_2 + w_3 \geq 0 \quad \text{Output } o=1 \\
x=1, y=0 & ; & w_1 + w_3 \geq 0 \quad \text{Output } o=1 \\
x=1, y=1 & ; & w_1 + w_2 + w_3 < 0 \quad \text{Output } o=0
\end{array}
$$

  - Wenn die ersten drei Ungleichungen erfüllt sind, kann Ungleichung vier nicht erfüllt werden
  - Es existieren somit keine Gewichte um das XOR-Problem mit Hilfe eines Perzeptrons zu lösen.

Neuronale Netzte

Mehrschichtige neuronale Netzwerke (ohne Rückkopplung)

- Mehrschichtiges Netz mit 5 Eingangsneuronen und 4 Neuronen in der Ausgangsschicht



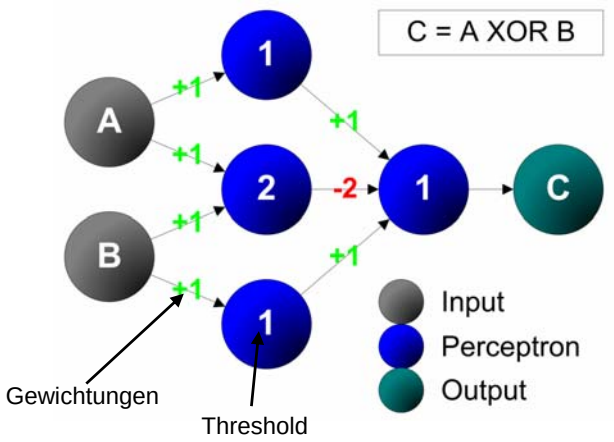
Neuronale Netzte

Mehrschichtige neuronale Netzwerke (ohne Rückkopplung)

- Anzahl der Internen Schichten und wie viele Neuronen pro Schicht vorhanden sind, wird von der Komplexität der Aufgabe bestimmt
- Neben Eingabe- und Ausgabeschicht noch weitere Schichten mit verdeckten Neuronen (hidden layer)
- Alle Neuronen einer Schicht sind vollständig mit den Neuronen der nächsten Schicht vorwärts verknüpft (Feedforward-Netze)

Neuronale Netzte

Mehrschichtige neuronale Netzwerke (ohne Rückkopplung)



Neuronale Netzte

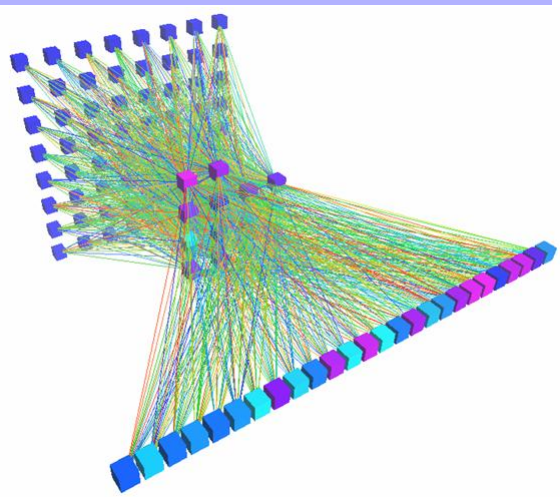
Mehrschichtige neuronale Netzwerke (ohne Rückkopplung)

- Der Backpropagation Lern-Algorithmus
  - Eingabemuster wird angelegt und vorwärts durch das Netz propagiert
  - Ausgabe des Netzes wird mit der gewünschten Ausgabe verglichen
  - Die Differenz wird als Fehler des Netzes erachtet
  - Gewichtungen der Neuronenverbindungen werden abhängig von ihrem Einfluss auf den Fehler geändert
  - Erneutes Anlegen der Eingabe ergibt eine Annäherung an die gewünschte Ausgabe

Neuronale Netzte

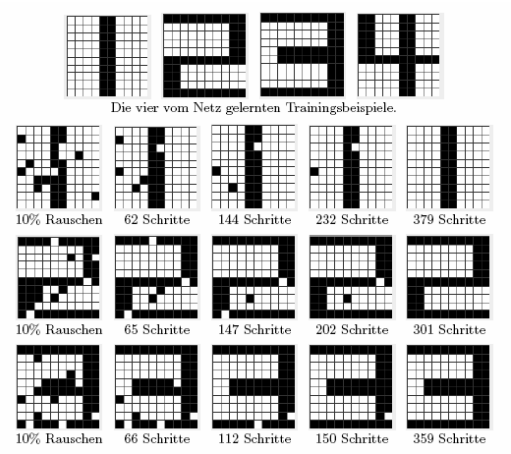
Praxisbeispiele - Mustererkennung

- 64-14-26 Feedforward Netz, das mit Hilfe des Backpropagation-Lern-algorithmus zur Er-kennung von Buchstaben verwendet wurde
- die Farben der Ver-bindungen stellen das Gewicht, die Farben der Neuronen den Schwellenwert dar



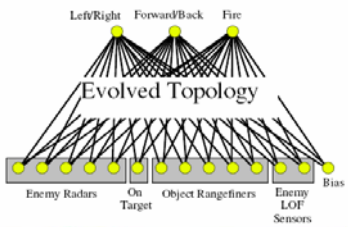
Neuronale Netzte
Praxisbeispiele - Mustererkennung

- Ziffern in einem 10 x10 Pixelfeld
- Hopfield-Netz (Netz mit Rückkopplung) mit 100 Neuronen und 4950 Gewichten
- Bei einem Hopfield-Netz existiert nur eine Schicht, die gleichzeitig als Ein- und Ausgabeschicht fungiert



Neuronale Netzte
Praxisbeispiele - Videospiele

- Verwendung von verschiedenen Netz-Topologien und Lern-Algorithmen für verschiedene Aufgaben
- Tools zur Entwicklung und zum Training
- Beispiel:
  Training für CTF-Modus zweier KI-Teams



(a) Robots approach flag (b) Player attacks on left (c) Robots learn new approach

Neuronale Netzte
Fazit

- Künstliche neuronale Netze (KNN) basieren auf der Vernetzung von Neuronen
- Je nach Aufgabe andere Topologie des Netzes
- Nach der Konstruktion folgt die Trainingsphase
- Netze ohne Rückkopplung und mit Rückkopplung
- Einschichtige und mehrschichtige Netze
- Lernverfahren für einschichtige, mehrschichtige, azyklische und zyklische
- viele Praktische Einsatzgebiete