# Softcomputing

Einsatz zum Lösen von Problemen,

- die unvollständig beschrieben sind
- die keine eindeutige Lösung haben
- für die keine effizienten exakten Algorithmen bekannt sind

einige Ansätze:

- Fuzzy-Logik, probabilistische Logik
- Künstliche neuronale Netze
- Evolutionäre Algorithmen

# Lernen

(Schrittweise) Änderung eines Systems (Verfahrens zur Problemlösung), so dass es bei der zukünftigen Anwendung dasselbe oder ähnliche Probleme besser löst.

- Aufgaben (Problem): Menge von Eingaben
- Aufgabeninstanz: Eingabe
- Lösung der Instanz: Ausgabe
- Bewertung der Lösung: Zuordnung Lösung $\rightarrow$ Güte

Schritte bei der Lösung einer Aufgabeninstanz:
Schüler (System)

1. verwendet ein Lösungsverfahren $V$ für diese Aufgabe
2. bestimmt eine Lösung $l$ der gegebenen Aufgabeninstanz
3. erfährt (oder bestimmt) die Bewertung dieser Lösung $l$
4. modifiziert das Lösungsverfahren $V$ zu $V'$, um (in Zukunft) Lösungen mit besseren Bewertungen zu finden
5. wendet im nächsten Schritt zur Lösung dieser Aufgabe das Lösungsverfahren $V'$ an

Lernen: Schritte 3 und 4

# Lernverfahren

Lernen durch

- Auswendiglernen (gegebener Beispiele)
- Anleitung (Anweisungen)
- logische Ableitung neuer Lösungsverfahren
- Analogie (zu gegebenen Beispielen)
  anhand Ähnlichkeit
- Erfahrung (durch gegebene Beispiele)
  Fähigkeit zur Verallgemeinerung
- Probieren und Beobachten
  (Erzeugen eigener Beispiele)

nach Art des Lernenden:

- natürliches Lernen
- künstliches / maschinelles Lernen

# Lernen durch gegebene Beispiele

nach der zum Lernen verwendbaren Information:

überwachtes Lernen (supervised learning)

- korrigierendes Lernen (corrective learning)
- bestärkendes Lernen (reinforcement learning)

unüberwachtes Lernen (unsupervised learning)

gewünschte Eigenschaften des Löseverfahrens:

- Korrektheit
  der Lösungen für die gegebenen Beispiele
- Generalisierung
  „sinnvolle“ Lösungen für ähnliche Aufgaben

# Korrigierendes Lernen

**Trainingsmenge:** Paare (Eingabe, Ausgabe)
(partielle Funktion an Stützstellen)

**Lernziel:** (möglichst einfache) Funktion, die an den Stützstellen mit der Trainingsmenge übereinstimmt

**Rückmeldung:** Trainer sagt nach jedem Lernschritt die korrekte Ausgabe.

**Prinzip:** Lernen durch Nachahmen (mit Korrektur)

- Klassizierung
  (Zuordnung von Objekten zu Klassen, abhängig von den Merkmalen der Objekte)
  z.B. Zuordnung Sensorwerte $\rightarrow$ Alarmklasse
  Trainingsmenge: Menge von Paaren (Objekteigenschaften, Klasse)
- Lernen von Funktionen
  Trainingsmenge: Menge von Paaren (Parameter, Funktionswert)

# Bestärkendes Lernen

**Trainingsmenge:** Eingaben

**Lernziel:** (möglichst einfache) Funktion, die den Stützstellen korrekte Werte zuordnet

**Rückmeldung:** Trainer sagt nach jedem Lernschritt, ob die Ausgabe korrekt war.

**Idee:** Lernen durch Probieren

- Klassifizierung
  Trainingsmenge: Menge von Objekten (mit ihren Eigenschaften)
  Bewertung der Lösung: ja, falls Zuordnung zur korrekten Klasse, sonst nein
- Lernen von Plänen (Anlagestrategien, Bewegungsabläufe usw.)
  z.B. Aufstehen eines humanoiden Roboters
  Trainingsmenge: Menge von Parametern (Motorstellung)
  Bewertung der Lösung: ja, falls Plan zum Erfolg geführt hat (Roboter steht sicher), sonst nein

# Unüberwachtes Lernen

Trainingsmenge: Eingaben

Lernziel:
- Gruppierung ähnliche Muster
- oft auch topologisch sinnvolle Anordnung

Idee: Lernen ohne Trainer (ohne Rückmeldung)

- Entdecken von Strukturen
- Selbstorganisation von Objekten zu Gruppen
  (mit gemeinsamen Merkmalen, typische Vertreter)
- topologieerhaltende Abbildungen
  (z.B. Körperteile $\rightarrow$ Gehirnregionen)
- Assoziation (z.B. in Schrifterkennung)

# Neuronale Netze

Neuron – Nerv (griechisch)

Modellierung und Simulation der Strukturen und Mechanismen im Nervensystem von Lebewesen

| Biologisches Vorbild | Mathematisches Modell |
|---|---|
| Nervenzellen (Neuronen) | künstliche Neuronen |
| Struktur (eines Teiles) eines Nervensystems | künstliche neuronale Netze (KNN) unterschiedlicher Struktur |
| Aktivierung von Neuronen, Reizübertragung | künstlichen Neuronen zugeordnete Funktionen |
| Anpassung (Lernen) | Änderungen verschiedener Parameter des KNN |

# Natürliche Neuronen

ZNS besteht aus miteinander verbundenen Nervenzellen (Neuronen)

Struktur eines Neurons:

- Zellkörper
- Dendriten
- Synapsen (verstärkende, hemmende)
- Axon

# Natürliche Neuronen – Funktionsweise

Informationsübertragung durch elektrochemische Vorgänge:

- aktivierte Zelle setzt an Synapsen Neurotransmitter frei,
- Neurotransmitter ändern die Durchlässigkeit der Zellmembran für Ionen an den Dendriten der empfangenden Zelle,
- Potential innerhalb der empfangenden Zelle ändert sich durch diffundierende Ionen,
- überschreitet die Summe der an allen Synapsen entstandenen Potentiale (Gesamtpotential) der Zelle einen Schwellwert, entsteht ein Aktionsspotential (Zelle feuert),
- Aktionspotential (Spannungsspitze) durchquert das Axon (Nervenfaser) zu den Synapsen zu Nachbarzellen,
- aktivierte Zelle setzt an Synapsen Neurotransmitter frei, usw.

Stärke der Information durch Häufigkeit der Spannungsspitzen (Frequenzmodulation).

# Eigenschaften natürlicher neuronaler Netze

- geringe Taktrate $10^{-}3$ s
- parallele Arbeit sehr vieler ($10^{11}$) Neuronen
- Neuronen sehr stark miteinander vernetzt
  (ca. 10 000 Nachbarn)
- Verarbeitungseinheit = Speicher

Vorteile:

- hohe Arbeitsgeschwindigkeit durch Parallelität,
- Funktionsfähigkeit auch nach Ausfall von Teilen des Netzes,
- Lernfähigkeit,
- Möglichkeit zur Generalisierung

Ziel: Nutzung dieser Vorteile zum Problemlösen
durch Wissensrepräsentation als künstliche neuronale Netze

# Natürliche Neuronen – Lernen

Speicherung von Informationen durch Anpassung der Durchlässigkeit (Leitfähigkeit) der Synapsen

- Regel von Hebb (1949):
  Synapsen zwischen gleichzeitig aktiven Zellen werden immer durchlässiger (Reizschwelle wird verringert),
  Verbindung an dieser Synapse wird stärker
- lange nicht benutzte Synapsen verlieren mit der Zeit ihre Durchlässigkeit
  Verbindung an dieser Synapse wird schwächer.

# Anwendungen künstlicher neuronaler Netze

Anwendungsgebiete:

- Bildverarbeitung, z.B.
  - Objekterkennung
  - Szenenerkennung
  - Schrifterkennung
  - Kantenerkennung
- Medizin, z.B. Auswertung von Bildern, Langzeit-EKGs
- automatische Spracherkennung
- Sicherheit, z.B. Biometrische Identifizierung
- Wirtschaft, z.B. Aktienprognosen, Kreditrisikoabschätzung
- Robotik, z.B. Lernen vom Bewegungsabläufen
- Steuerung autonomer Fahrzeuge

# Geschichte künstlicher neuronaler Netze

- 1943, Warren McCulloch, Walter Pitts:
  A logical calculus of the ideas immanent in nervous activity
- 1949, Donald O. Hebb: Lernmodell
  The organization of behaviour
- 1957 Frank Rosenblatt: Perzeptron (1 Schicht)
  erster Neurocomputer MARK 1
  (Ziffernerkennung in $20 \times 20$-Bildsensor)
- 1969, Marvin Minsky, Seymour Papert: Perceptrons
- 1971 Perzeptron mit 8 Schichten
- 1974 Backpropagation (Erfindung)
- 1982, Teuvo Kohonen: selbstorganisierene Karten
- 1982, John Hopfield: Hopfield-Netze
- 1985, Backpropagation (Anwendung)
- 1997 long short-term memory
- 2000, Begriff Deep Learning für KNN, Faltungsnetze
- 2009 Training mit GPUs
- 2017 AlphaZero, ...

# Künstliche Neuronen: McCulloch-Pitts-Neuron ohne Hemmung

einfaches abstraktes Neuronenmodell von
McCulloch und Pitts, 1943

Aufbau eines künstlichen Neurons $u$ (Tafel)

| | | |
|---|---|---|
| Eingabe: | $x = (x_1, \ldots, x_{m_u}) \in \{0,1\}^{m_u}$ | (ankommende Reize) |
| Schwellwert: | $\theta_u \in \mathbb{R}$ | (Reizschwelle) |
| Ausgabe: | $f(x_1, \ldots, x_{m_u}) \in \{0,1\}$ | (weitergegebener Reiz) |

Parameter eines McCulloch-Pitts-Neurons $u$ ohne Hemmung:

- $m_u$: Anzahl der (erregenden) Eingänge
- $\theta_u$: Schwellwert

# McCulloch-Pitts-Neuron ohne Hemmung: Funktionen

Eingangsfunktion des Neurons $u$: $I_u$: $\{0,1\}^{m_u} \to \mathbb{R}$ mit

$$I_u(x_1, \ldots, x_{m_u}) = \sum_{i=1}^{m_u} x_i$$

(Summe aller erregenden Eingänge des Neurons $u$)
Aktivierungsfunktion des Neurons $u$ (abhängig vom Schwellwert $\theta_u$): $A_u$: $\mathbb{R} \times \mathbb{R} \to \{0,1\}$ mit

$$A_u(\theta_u, v) = \begin{cases} 1 & \text{falls } v \geq \theta_u \\ 0 & \text{sonst} \end{cases}$$

(Stufenfunktion mit Stufe bei $\theta_u$)
Ausgabefunktion des Neurons $u$: $O_u$: $\{0,1\} \to \{0,1\}$ mit

$$O_u(v) = v$$

(Identität)

# McCulloch-Pitts-Neuron ohne Hemmung: Berechnung

vom Neuron $u$ berechnete Funktion: $f_u: \{0,1\}^{m_u} \to \{0,1\}$ mit

$$\begin{aligned} f_u(x_1, \ldots, x_{m_u}) &= O_u(A_u(\theta_u, I_u(x_1, \ldots, x_{m_u}))) \\ &= \begin{cases} 1 & \text{falls } \sum_{i=1}^{m_u} x_i \geq \theta_u \\ 0 & \text{sonst} \end{cases} \end{aligned}$$

$m_u$-stellige Boolesche Funktion

# McCulloch-Pitts-Neuron ohne Hemmung: Beispiele

elementare Boolesche Funktionen $\vee, \wedge$

mehrstellige $\vee, \wedge$

Existiert zu jeder Booleschen Funktion $f : \{0,1\}^n \to \{0,1\}$ ein McCulloch-Pitts-Neuron ohne Hemmung, welches $f$ berechnet?

Nein, nur monotone Boolesche Funktionen,
z.B. $\neg$ nicht

Warum?

# Geometrische Interpretation

Jedes McCulloch-Pitts-Neuron $u$ mit $m_u$ Eingängen teilt die Menge $\{0,1\}^{m_u}$ in zwei Teilmengen:

$$\begin{aligned} f_u^{-1}(1) &= \{(x_1,\ldots,x_{m_u}) \in \{0,1\}^{m_u} \mid f(x_1,\ldots,x_{m_u}) = 1\} \\ &= \{(x_1,\ldots,x_{m_u}) \in \{0,1\}^{m_u} \mid \sum_{i=1}^{m_u} \geq \theta_u\} \end{aligned}$$

und

$$\begin{aligned} f_u^{-1}(0) &= \{(x_1,\ldots,x_{m_u}) \in \{0,1\}^{m_u} \mid f(x_1,\ldots,x_{m_u}) = 1\} \\ &= \{(x_1,\ldots,x_{m_u}) \in \{0,1\}^{m_u} \mid \sum_{i=1}^{m_u} < \theta_u\} \end{aligned}$$

geometrische Interpretation als Teilräume des $R^m$
Grenze zwischen beiden Bereichen:
$(m_u - 1)$-dimensionaler Teilraum $\sum_{i=1}^{m_u} x_i = \theta$
parallele Schnitte (abhängig von $\theta$)

# Geometrische Interpretation: Beispiele

Beispiele:

- Neuron $u$ mit $m_u = 2$ Eingängen und Schwellwert $\theta_u = 1$

$$f_u(x_1, x_2) = \begin{cases} 1 & \text{falls } x_1 + x_2 \geq 1 \\ 0 & \text{sonst} \end{cases}$$

Bereich der $x_1, x_2$-Ebene mit $f_u(x_1, x_2) = 1$ ist die Halbebene mit $x_2 \geq 1 - x_1$.

$x_2 = g(x_1) = 1 - x_1$ ist eine lineare Trennfunktion zwischen den Halbebenen mit $f_u(x_1, x_2) = 0$ und $f_u(x_1, x_2) = 1$.

- Neuron $v$ mit $m_v = 3$ Eingängen und $\theta_v = 1$

# Linear trennbare Funktionen

Zwei Mengen $A, B \subseteq \mathbb{R}^n$ heißen genau dann linear trennbar, wenn eine lineare Funktion $g : \mathbb{R}^n \to \mathbb{R}$ mit $g(x_1, \ldots, x_n) = a_0 + \sum_{i=1}^{n} a_i x_1$ existiert, so dass

- für alle $(x_1, \ldots, x_n) \in A$ gilt $g(x_1, \ldots, x_n) > 0$
- für alle $(x_1, \ldots, x_n) \in B$ gilt $g(x_1, \ldots, x_n) < 0$

(eindeutig beschreiben durch $n+1$-Tupel $(a_0, a_1, \ldots, a_n)$ )

Eine Boolesche Funktion $f : \{0,1\}^n \to \{0,1\}$ heißt genau dann linear trennbar, wenn die Mengen $f^{-1}(0)$ und $f^{-1}(1)$ linear trennbar sind.

Beispiele: $\vee, \wedge, \neg x_1, x_1 \to x_2, x_1 \wedge \neg x_2$

Die Boolesche Funktion `XOR` ist nicht linear trennbar.

# McCulloch-Pitts-Neuron mit Hemmung

McCulloch-Pitts-Neuron $u$ mit Hemmung:

| | | |
|---|---|---|
| Eingabewerte: | $x = (x_1, \ldots, x_{m_u}) \in \{0,1\}^{m_u}$ | erregend |
| | $y = (y_1, \ldots, y_{m'_u}) \in \{0,1\}^{m'_u}$ | hemmend |
| Schwellwert: | $\theta_u \in \mathbb{R}$ | |
| Ausgabe: | $f(x_1, \ldots, x_{m_u}, y_1, \ldots, y_{m'_u}) \in \{0,1\}$ | |

Parameter eines McCulloch-Pitts-Neurons $u$ (mit Hemmung):

- $m_u$: Anzahl der erregenden Eingänge
- $m'_u$: Anzahl der hemmenden Eingänge
- $\theta_u$: Schwellwert

# Funktionen bei hemmenden Eingängen

Eingangsfunktion des Neurons $u$: $I_u : \{0,1\}^{m_u+m'_u} \to \mathbb{R} \times \mathbb{R}$

$$I_u(x_1, \ldots, x_{m_u}, y_1, \ldots, y_{m'_u}) = \left( \sum_{i=1}^{m_u} x_i, \sum_{i=1}^{m'_u} y_i \right)$$

(Summe aller erregenden Eingänge des Neurons $u$,
Summe aller hemmenden Eingänge des Neurons $u$)
Aktivierungsfunktion des Neurons $u$ (abhängig von $\theta_u$):
$A_u : \mathbb{R} \times (\mathbb{R} \times \mathbb{R}) \to \{0,1\}$

$$A_u(\theta_u, (x,y)) = \begin{cases} 1 & \text{falls } x \geq \theta_u \text{ und } y \leq 0 \\ 0 & \text{sonst} \end{cases}$$

(Stufenfunktion)
Ausgabefunktion des Neurons $u$: $O_u$: $\{0,1\} \to \{0,1\}$ mit

$$O_u(v) = v$$

(Identität)

# Berechnung bei hemmenden Eingängen

Gesamtfunktion des Neurons $u$

$$f_u(x_1, \ldots, x_{m_u}, y_1, \ldots, y_{m'_u}) = O_u(A_u(\theta_u, I_u(x_1, \ldots, x_{m_u}, y_1, \ldots, y_{m'_u})))$$

Jedes McCulloch-Pitts-Neuron $u$ mit $m_u$ erregenden Eingängen, $m'_u$ hemmenden Eingängen und Schwellwert $\theta_u$ repräsentiert die Boolesche Funktion $f_u : \{0,1\}^{m_u+m'_u} \to \{0,1\}$:

$$f_u(x_1, \ldots, x_{m_u}, y_1, \ldots, y_{m'_u}) = \begin{cases} 1 & \text{falls } \sum_{i=1}^{m_u} x_i \geq \theta_u \\ & \text{und } \sum_{i=1}^{m'_u} y_i \leq 0 \\ 0 & \text{sonst} \end{cases}$$

Beispiele mit Hemmung:

- elementare Boolesche Funktion: $\neg$
- komplexere Boolesche Funktionen, z.B.
  $x_1 \wedge \neg x_2$
  $\neg x_1 \wedge x_2 \wedge x_3$,
  $\neg(x_1 \vee \neg x_2 \vee \neg x_3)$

# McCulloch-Pitts-Netze

McCulloch-Pitts-Netz:

gerichteter Graph mit

- McCulloch-Pitts-Neuronen als Ecken und
- gerichteten Kanten zwischen Neuronen
  zwei Arten: erregend, hemmend

Berechnung der Neuronen-Funktionen
(entsprechend Struktur des Netzes):

- parallel
- sequentiell
- rekursiv

# McCulloch-Pitts-Netze

## Ein-Schicht-McCulloch-Pitts-Netz

parallele Schaltung mehrerer
Mc-Culloch-Pitts-Neuronen
repräsentiert Boolesche Funktionen mit mehreren Ausgaben
Beispiel: Parallelschaltung von $x_1 \wedge \neg x_2$ und $\neg x_1 \wedge x_2$

## Mehr-Schicht-McCulloch-Pitts-Netz

parallele und sequentielle Schaltung mehrerer
Mc-Culloch-Pitts-Neuronen
Beispiel: XOR

Analogie zu logischen Schaltkreisen

Jede Boolesche Funktion $f : \{0,1\}^n \to \{0,1\}$ lässt sich durch ein McCulloch-Pitts-Netz berechnen.

McCulloch-Pitts-Netz mit zwei Schichten genügt
(analog DNF, CNF in Aussagenlogik)

# Modifikationen von McCulloch-Pitts-Neuronen

- Durch Vervielfachung eines Einganges erhöht sich seine Wirkung (sein Gewicht).
- Vervielfachung (absolut) hemmender Eingänge ändert die berechnete Funktion nicht.
- relative Hemmung:
  hemmende Eingänge verhindern das Feuern der Zelle nicht völlig, sondern erschweren es (erhöhen den Schwellwert, negatives Gewicht).
- Absolute Hemmung lässt sich durch relative Hemmung mit großer Schwellwerterhöhung (auf Anzahl aller erregenden Eingänge $+1$) simulieren.
- Durch Einführung von Gewichten wird Trennung in hemmende und erregende Eingänge überflüssig.

# Parameter künstlicher Neuronen

verschiedene künstliche Neuronenmodelle unterscheiden sich in:

- Anzahl Typen der Ein- und Ausgabewerte,
- zulässige Gewichte an den Eingangskanten,
- Eingabe-, Ausgabe- und Aktivierungsfunktion

Jedes Neuron mit $m$ Eingängen repräsentiert eine Funktion von $m$ Eingabewerten