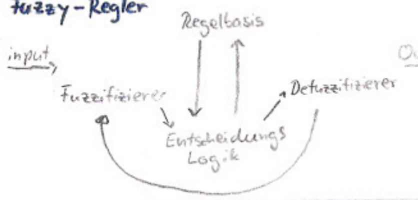
Fuzzy

Fuzzy-Regler



Mengenbeziehungen

$A = B \Rightarrow \mu_A(x) = \mu_B(x)$

$A \subseteq B \Rightarrow \mu_A(x) \leq \mu_B(x)$

$A \cap B \Rightarrow \mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x))$

$A \cup B \Rightarrow \mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x))$

Relationen

$A = B \Rightarrow \mu_A(x) \equiv \mu_B(x)$

$A \subseteq B \Rightarrow \mu_A(x) \leq \mu_B(x)$

$A \subset B \Rightarrow (A \subseteq B)(A \neq B)$

$A \neq B,\ A \subseteq B,\ A \subset B$

Bemerkungen

$\emptyset - \mu_\emptyset(x) \equiv 0$

$\Omega - \mu_\Omega(x) \equiv 1$

$A \cap \bar{A} = \emptyset$

$A \cup \bar{A} = \Omega$

Fuzzy-Menge

x	0	1	2	3	4	5	6	7	8	9	10
Diskreter Fall → $\mu_A(x)$	0.0	0.0	0.0	0.4	0.7	1.0	0.7	0.4	0.0	0.0	0.0

Analytisch: $0.0/0 + 0.0/1 + 0.0/2 + 0.4/3 + 0.7/4 \ldots$

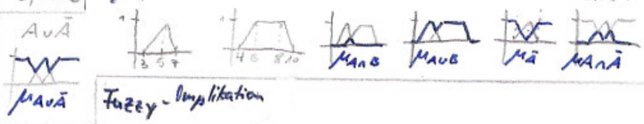
Kontinuierlicher Fall: Allg Notation: $\int_\Omega \mu_A(x)/x$

$A = \int_0^{10} \mu_A(x)/x = \int_0^2 0/x + \int_2^5 (\frac{1}{3}x - \frac{2}{3})/x + \int_5^8 (-\frac{1}{3}x + \frac{8}{3})/x + \int_8^{10} 0/x$

$A = \int_2^5 (\frac{1}{3}x - \frac{2}{3})/x + \int_5^8 (-\frac{1}{3}x + \frac{8}{3})/x$ Analytisch: $L = y(x) = \frac{1}{3}x + b$

$R = y(x) = -\frac{1}{3}x + b$

Bsp: „ca. 5" „[6,8]" $A \cap B$ $A \cup B$ $\bar{A}$ $A \cap \bar{A}$ $A \cup \bar{A}$



Kenngrößen

$A = 0.2/2 + 0.3/3 + 0.4/4 + 0.5/5 + 0.4/6 + 0.3/7 + 0.2/8$

Kardinalzahl $= \text{card}(A) = \sum_{x \in \Omega} \mu_A(x) = 0.2 + 0.3 + 0.4 \ldots = \underline{2.3}$

Höhe $= \text{hgt}(A) = 0.5$ (höchster Pkt) $\neq 1 \Rightarrow$ nicht normalisiert $A_n = 0.4/2 + 0.6/3 + 0.8/4 + 1.0/5 + \ldots$

$\Rightarrow \text{card}(A_n) = \frac{\text{card}(A)}{\text{hgt}(A)} = \frac{2.3}{0.5} = 4.6$ ← normalisierung

Träger (Stützmenge) $= \mu_A(x) > 0 \Rightarrow \text{supp}(A) = \{2,3,4,5,6,7,8\}$ | $\text{supp}(A_n) = \text{supp}(A)$

Kern $= \text{core}(A) = \mu_A(x) = 1 \Rightarrow \text{core}(A) = \{\emptyset\}$ | $\text{core}(A_n) = \{5\}$

α-Schnitt $= A \geq \alpha \Rightarrow \mu_A(x) \geq \alpha \Rightarrow A^{\geq 0.4} = \{4,5,6\}$

Strenger α-Schnitt $= A > \alpha \Rightarrow \mu_A(x) > \alpha \Rightarrow A^{>0.4} = \{5\}$

Fuzzy-Implikation

MIN-Imp: $\min(\mu_A(x), \mu_B(y))$

Zadeh-Imp: $\max(\min(\mu_A(x), \mu_B(y)), 1 - \mu_A(x))$

Lukasiewicz-Implikation: $\min(1, 1 - \mu_A(x) + \mu_B(y))$

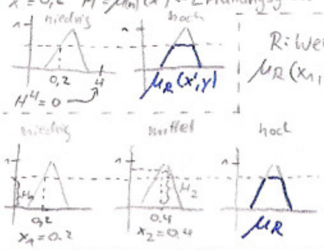
Kleene-Dines-Imp: $\max(1 - \mu_A(x), \mu_B(y))$

Gödel-Imp: 1, wenn $\mu_A(x) < \mu_B(y)$ | $\mu_B(y)$, sonst

Fuzzy-Inferenz

R: Wenn x = niedrig, dann y = hoch | $\mu_R(x,y) = \min(\mu_n(x), \mu_h(y))$ **MIN**

$x' = 0,2$ $H' = \mu_n(x')$ ← Erfüllungsgrad Regel [R] aktiv, wenn $H > 0$

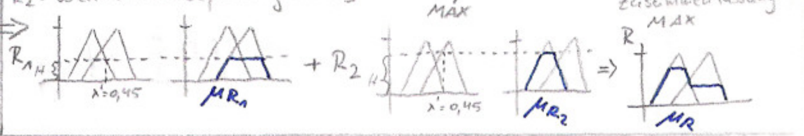


R: Wenn x_1 = niedrig und x_2 = mittel, dann y = hoch

$\mu_R(x_1, x_2, y) = \min(\min(\mu_n(x_1), \mu_m(x_2)), \mu_h(y))$

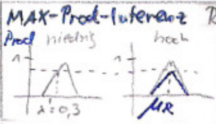
$= \min(H, \mu_h(y))$ mit $H = \min(\mu_n(x_1), \mu_m(x_2))$

R_1: Wenn x = niedrig, dann y = hoch
R_2: Wenn x = mittel, dann y = mittel
} R: R_1 oder R_2 **MAX**

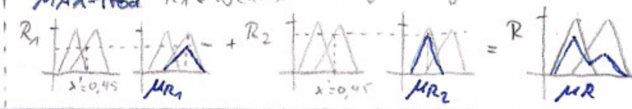


MAX-Prod-Inferenz

R: Wenn x = niedrig, dann y = hoch

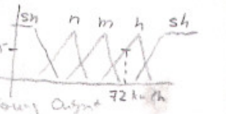


MAX-Prod R_1: Wenn x = niedrig, dann y = hoch R_2: Wenn x = mittel, dann y = mittel



Fuzzifizierung

$v' = 72$ km/h

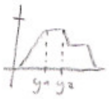


Input v' → Fuzzifizierung → Output $v^{*\prime}$

$(\mu_{sn}(v'), \mu_n(v'), \mu_m(v'), \mu_h(v'), \mu_{sh}(v'))^T$

$v^{*\prime} = (0, 0, 0, 0, 0.6)$

Defuzzifizierung Max-Methode



V_1 = Mittelwert

$y^* = \frac{y_1 + y_2}{2}$

V_2 (linke Max) $= y_1$

V_3 (rechte Max) $= y_2$

neg:

Schwerpunkt-Methode

Vorteil: Robustheit

$y^* = \frac{\int_a^b y \cdot \mu_R(y) \cdot dy}{\int_a^b \mu_R(y) \cdot dy}$

t-Norm (Durchschnitt)

$t: [0,1] \times [0,1] \to t(x,y) \in [0,1]$

1.) neutrales Element
2.) Monotonie
3.) Kommutativität
4.) Assoziativität

t-Conorm (Vereinigung)

$s: [0,1] \times [0,1] \to [0,1]$

Fuzzy-Ähnlich

$\Rightarrow$ kann, wenn kern = kern

Streng-Fuzzy-Ähnlich

Satz: Zwei Fuzzy-Mengen deren Träger + Kern übereinstimmen, sind streng fuzzy-ähnlich + umgekehrt

$M_1 \simeq M_2$; $\mu_1 \simeq \mu_2$ | $\bar{M}_1 \simeq \bar{M}_2$; $1 - \mu_1 \simeq 1 - \mu_2$

KNN

Neuronale Netze...
... bestehen aus Elementen, die untereinander Daten/Informationen austauschen
... sind in der Lage zu lernen und ihre Leistungsfähigkeit zu verbessern

Aktivierungs-Funktion*

$f(x) = \begin{cases} -1 & x \leq -1/a \\ a \cdot x & -1/a < x < 1/a \\ 1 & x \geq 1/a \end{cases}$

mit $x = \text{gin}_i - \Theta$ und $a > 0$

Sigmoidfunktion

$\frac{1}{1 + e^{-g \cdot x}}$

Aktivierung im Bereich [0,1]

Tangenshyperbolikus-Funktion

$\frac{e^{g \cdot x} - e^{-g \cdot x}}{e^{g \cdot x} + e^{-g \cdot x}}$ Aktivierung im Bereich [-1,1]

*Aktivierungsstatus!

Delta Regel

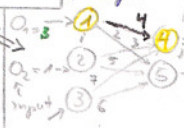
$\Delta W_{ij} = \frac{(\text{Lernrate} \cdot \text{Output} \cdot \text{Fehler})}{\text{Anzahl Inputs}}$

$\Delta W_{14} = \frac{1 \cdot 3 \cdot 0,8}{3} = \underline{\underline{0,8}}$

$W_{14}^{(n)} = 4 + 0,8 = \underline{\underline{4,8}}$

Perceptron von Rosenblatt

Lernrate $\sigma = 1$



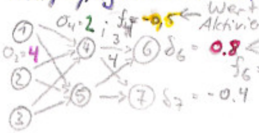
ΔW_{ij} = Lernrate · alter Output · Fehler

Input

$\Delta W_{14} = 1 \cdot 3 \cdot 0,8 = \underline{\underline{2,4}}$

$W_{14}^{(\text{neu})} = W_{14} + \Delta W_{14} = 4 + 2,4 = \underline{\underline{6,4}}$

Backpropagation-Net-Verfahren



$\nabla_6 = 1,6$

$\Rightarrow \delta_6 = f_6' \cdot \nabla_6 = 0,5 \cdot 1,6 = 0,8$

Lernrate $\sigma = 1$

$\Delta W_{46} = 1 \cdot 2 \cdot 0,8 = \underline{\underline{1,6}}$

$\delta_6 = f_6' \cdot (O_6 - O_{6\text{soll}})$ ← physischer Fehler

↖ modifizierte Fehler

$\delta_4 = f_4' \cdot (W_{46} \cdot \delta_6 + W_{47} \cdot \delta_7) = -0,5((3 \cdot 0,8) + (4 \cdot (-0,4)))$

$= -0,5(2,4 + (-1,6))$

$= \underline{\underline{-0,4}}$ ← σ_7

$\Delta W_{24} = 1 \cdot 4 \cdot (-0,4) = \underline{\underline{-1,6}}$

$\hat{=}$ Lernrate · Output · Fehler

Perceptrons: XOR-Funktion nicht lösbar ⇒ Aufbau Perceptrons + dessen (binäre) Neuronen

ja nein

NN:

Inputschicht, innere*Schicht, Outputschicht

Gewicht bewirkt, dass hoher Input viel/klein Einfluss

*hidden

Inputfunktion

1.) MAX $(j = 1 \ldots n) \to \max_j(in_{ij} \cdot W_{ij})$
2.) MIN $\to \min_j(in_{ij} \cdot W_{ij})$
3.) Prod $\to \prod_j (in_{ij} \cdot W_{ij})$
4.) Sum $\to \sum_j in_{ij} \cdot W_{ij}$

1.) Max-Wert aller gewichteten Einzelinputs wird gewählt
2.) Min-Wert [...] wird gewählt
3.) Summiert alle gewichteten Einzelinputs
4.) Multipliziert [...]

① Netzoutput- und Sollontput vergleichen ⇒ Fehler = Gewichtsveränderung
② Kein exakter Fehlerwert (nahe, weit, sehr nahe)
③ Menge durch Kriterien aufteilen

Outputfunktion: Binär oder keine!

Netztypen:

Feedword → Feedforward
Feedback

Entscheidungen

1.) Einzelentscheidung ⇒ ja/nein
2.) Alternativentscheidung ⇒ Auswahl aus Menge
3.) Programmentscheidung ⇒ Auswahl Menge aus Menge

Training:

Netz versucht zur Inputkonstellation den richtigen Output über Einstellen der Gewichte zu finden. Die Gesamtheit aller Versuche bezeichnet man als Trainingsdatensatz.

Lernformen:

① • überwachtes Lernen
② • bestärkendes "
③ • unüberwachtes "

Lernregel:

neues Gewicht = altes Gewicht + Gewichtsänderung

$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t)$

Hebbsche Hypothese:
Nehmen 2 Neuronen zur gleichen Zeit den gleichen Zustand an (aktiv oder inaktiv), dann wird das Gewicht zwischen den Neuronen verstärkt.

Perceptron Regel:
Für jedes Neuron N_i der Ausgabeschicht wird die Solloutputabweichung δ_i berechnet. δ_i wird mit out_j (Output des Vorgängerneurons N_j) und dem Faktor σ (Lernrate) multipliziert.

$\Delta w_{ij} = \sigma \cdot out_j \cdot (a_{gi} - out_i)$ ← Solloutput

$\delta_i = (a_{gi} - out_i)$