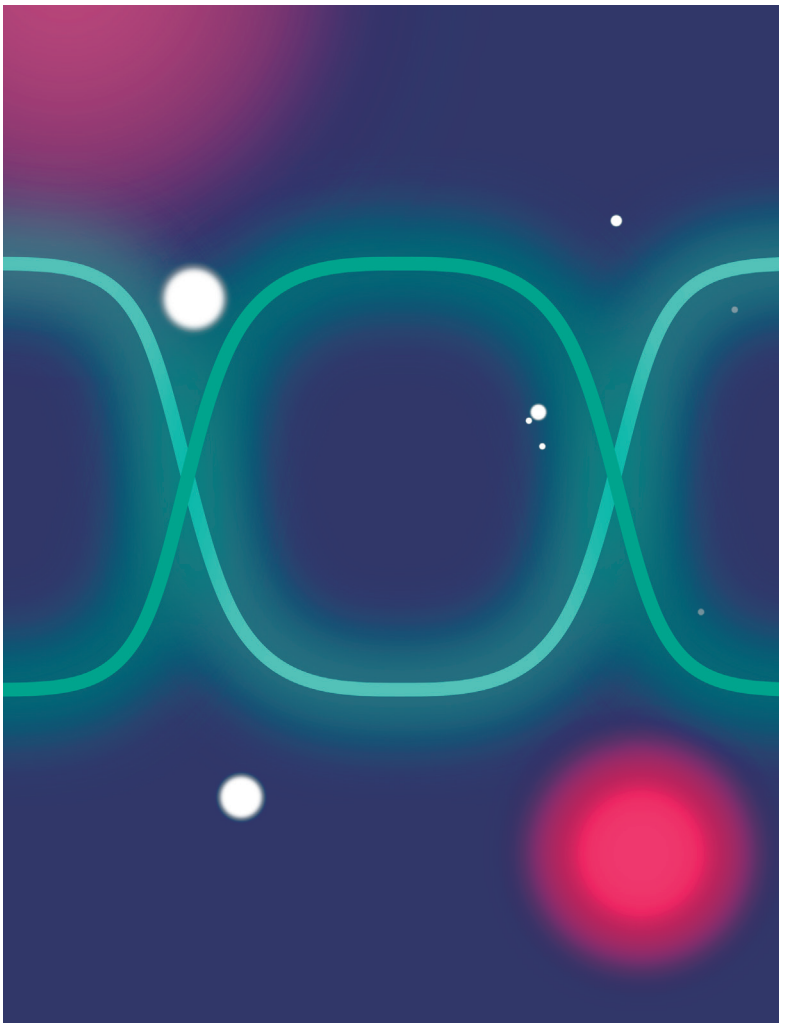
FUZZYLOGIK, TEIL 1

Unscharf zur ruckelfreien U-Bahn



Unscharfe Lösungen lassen sich mit Fuzzylogik modellieren. dotnetpro erklärt die Grundlagen.

Fuzzylogik oder „unscharfe" Logik: Viele haben davon gehört, wenige wissen, wie es funktioniert, und die wenigsten haben es in der Praxis eingesetzt.

In den letzten Jahren haben maschinelles Lernen und neuronale Netze dank Cloud-Anwendungen ein Revival erlebt, aber Fuzzylogik scheint auf der Strecke geblieben zu sein. Das ist schade, denn Fuzzylogik ist ein leistungsfähiges Mittel, um pragmatisch zu modulieren, Aufgaben des maschinellen Lernens zu lösen und nichtlineare Regler auf einfache Weise zu implementieren.

Dieser erste Teil einer Artikelfolge zum Thema erklärt die grundlegenden Prinzipien der Fuzzylogik. Der zweite Teil wird dann zeigen, wie mit der Technologie auf elegante Weise ein Regler implementiert werden kann, der das dynamische Verhalten eines klassischen PID-Reglers stark verbessert.

Fuzzylogik wurde 1965 von Lotfi A. Zadeh erfunden, einem Professor der Computerwissenschaften an der Universität von California in Berkeley [1]. Popularität erlangte die Fuzzy-Technologie in den 80er Jahren, insbesondere in Japan während der sogenannten japanischen Fuzzy-Welle. Das Paradebeispiel ist die Steuerung der vollautomatischen U-Bahn in Sendai [2]. Mitte der 90er Jahre brachte Europa die Technologie mit Fördermitteln und Artikeln zu sich.

Dann, nach einigen Jahren des Hypes, wurde es still um die Fuzzy-Theorie. Aber warum? War es schlechtes Marketing? War es der dubiose Name „Fuzzylogik", der in sich einen Widerspruch enthält? Denn wie kann etwas unscharf und doch gleichzeitig logisch sein?

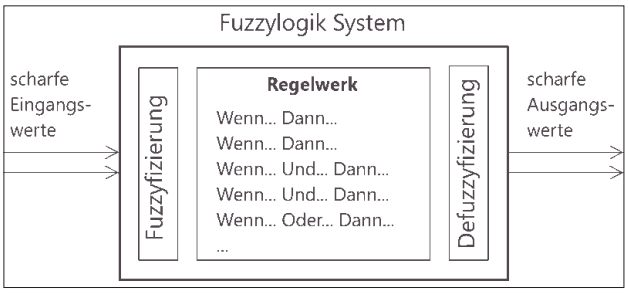
Oder waren es vielleicht die vielen Freiheitsgrade, mit denen der Anwender klarkommen muss, wenn er sich eine Lösung erarbeitet? Im Internet ist kein eindeutiger Grund für das Schweigen zu finden, allerdings auch keine negativen Artikel.

Tatsache ist, dass in den 90er Jahren ein Projekt nach dem anderen erfolgreich mit der Technologie umgesetzt wurde. Besonders erfolgreich war der Autor dieses Artikels mit dem PIFuzzy-Regler, der auf elegante Weise die Unzulänglichkeiten des linearen PID-Reglers beseitigt. Dieser PIFuzzy-Regler wird in einem zweiten Artikel als Praxisbeispiel dienen.

Unscharfe Behauptungen

Fuzzylogik ist eine Erweiterung der booleschen Logik, bei der eine Behauptung nicht nur wahr oder falsch ist, sondern einen Wahrheitsgrad irgendwo zwischen null und eins hat. Deshalb ist hier auch von „unscharfen Behauptungen" die Rede.

Zum Beispiel würde die Aussage „Heute beträgt die Chance auf Regen 30 Prozent" in Fuzzylogik eine Behauptung wie „Heute wird es regnen" mit dem Wahrheitsgrad 0,3 sein.



Ein Überblick über ein Fuzzylogik-System (Bild 1)

Ein anderes Beispiel ist Googles reCaptcha v3, der Dienst, der abzuschätzen versucht, ob er es mit einem Menschen oder mit einem Bot zu tun hat. Er bietet Spamschutz für Kontaktformulare, indem der reCaptcha-Server dem Nutzer des Formulars einen Wert zwischen 0,1 und 1 liefert. Je niedriger der Wert, desto größer die Chance, dass der Nutzer ein Spam-Roboter ist. In Fuzzylogik entspricht dies der Behauptung „Der Nutzer ist ein Mensch“ mit einem Wahrheitsgrad zwischen 0,1 und 1.

Aus den Behauptungen oben folgen zum Beispiel boolesche Entscheidungen wie „ab 20 Prozent Chance auf Regen nehme ich den Regenschirm mit“ oder „die Mail wird nur versendet, wenn der reCaptcha-v3-Wert größer als 0,6 ist“. Diese booleschen Resultate werden auch „crisp values“ genannt (0 oder 1). Aber für einen Regler werden Fließkommawerte zwischen 0 und 100 (Prozent) benötigt, die zum Beispiel als Stellwert für eine Heizung dienen; oder zwischen -100 und 100 (Prozent), wenn das System neben der Heizung auch eine regelbare Kühlung hat. Wie das zu erreichen ist, wird das Beispiel eines vereinfachten Raumtemperaturreglers zeigen.

Fuzzylogik-System im Überblick

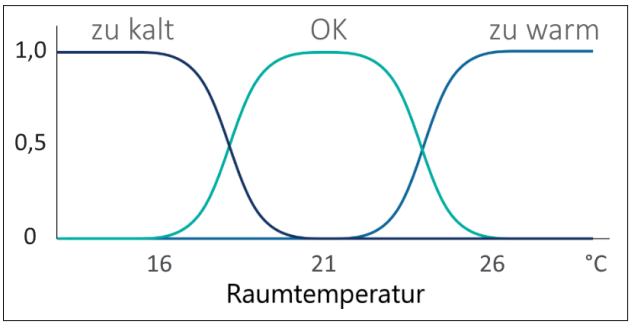
Mit Fuzzylogik lässt sich auf natürlichem Weg menschliches Wissen in ein Computerprogramm umwandeln. Dazu werden zunächst scharfe Eingangswerte in Behauptungen mit Wahrheitsgraden umgewandelt (Fuzzyfizierung).

In einem Regelwerk werden die Behauptungen zu Schlussfolgerungen oder Aktionen kombiniert (ebenfalls im linguistischen Bereich). Durch Defuzzyfizierung werden aus der Summe der Schlussfolgerungen oder Aktionen ein oder mehrere scharfe Ausgangswerte berechnet. Der Prozess ist in vereinfachter Form in Bild 1 dargestellt.

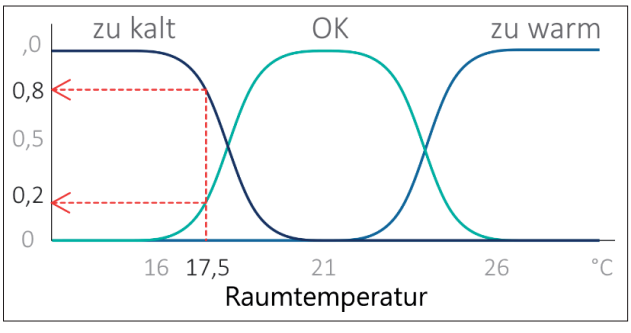
Fuzzysets und Zugehörigkeitsfunktionen

Die Raumtemperatur wird gemäß Bild 2 in drei Behauptungen unterteilt: *zu kalt*, *OK* und *zu warm*. Die einzelnen Behauptungen heißen Zugehörigkeitsfunktionen (membership functions) und bilden zusammen ein Fuzzy-Set. Die Raumtemperatur wird in der Fuzzy-Welt auch linguistische Variable genannt, da sie auf sprachlichen Formulierungen (die drei Zugehörigkeitsfunktionen) basiert.

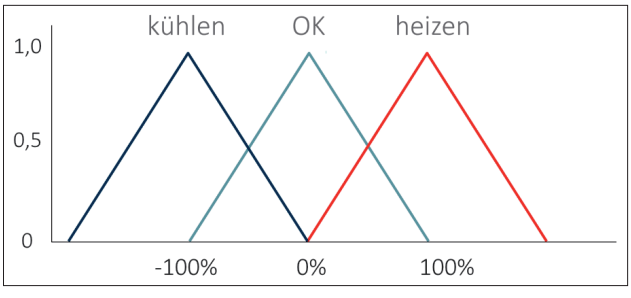
Jede Behauptung hat einen Wahrheitsgrad, der bestimmt wird durch die Raumtemperatur. Bei 21 °C ist die Wahrheit der Behauptung *OK 1* und die anderen Behauptungen sind *0*. Wenn die Raumtemperatur 17,5 °C beträgt, hat die Behauptung *OK* nur noch einen Wahrheitsgrad von *0,2* und die Behauptung *zu kalt* einen Wahrheitsgrad von *0,8* (Bild 3).



Die linguistische Variable Raumtemperatur (Bild 2)



Fuzzyfizierung der Raumtemperatur (Bild 3)



Das Fuzzy-Set Regelung mit den zugehörigen Funktionen heizen und kühlen (Bild 4)

Wie schon erwähnt, muss sich der Urheber eines Fuzzylogik-Systems mit vielen Freiheiten herumschlagen. Wie so oft gilt auch hier: So viel wie nötig, so wenig wie möglich.

Ein paar Faustregeln aus der Praxis helfen dabei, diese Freiheiten in den Griff zu bekommen:

- Der Wahrheitsgrad einer Zugehörigkeitsfunktion sollte sich immer zwischen 0 und 1 bewegen (Grenzwerte inklusive).
- Es sollen sich nicht mehr als zwei Zugehörigkeitsfunktionen überlappen, wobei die Summe der Wahrheitsgrade immer 1 beträgt (komplementär).
- Die linke und rechte Zugehörigkeitsfunktion sollen „open end“ sein, um den gesamten Wertebereich abzudecken.
- Vorzugsweise sollten dreieckige Zugehörigkeitsfunktionen verwendet werden, da trapezförmige Funktionen zu toten Zonen führen und gaußsche Funktionen kompliziert zu implementieren sind. Das Beispiel der Raumtemperatur lässt sich also noch vereinfachen.
- Beginne mit drei Zugehörigkeitsfunktionen pro Fuzzy-Set. Wenn drei Funktionen nicht ausreichen, kann man auf fünf Funktionen ausweichen.

Regelwerk und Defuzzyfizierung

Die unscharfen Behauptungen kann man mit Aktionen verbinden, so entsteht ein Regelwerk. Die Regeln können in- ▶

dividuell gewichtet werden, sodass sie mehr Einfluss auf den Stellwert haben, dazu später mehr.

Ein einfaches Regelwerk für den Raumtemperaturregler könnte folgendermaßen aussehen, wobei alle Regeln gleich gewichtet (1x) sind:

```
1x WENN zu kalt DANN heizen
1x WENN OK DANN OK
1x WENN zu warm DANN kühlen
```

Als Ausgangsvariable wird eine linguistische Variable *Regelung* definiert, die aus symmetrisch gleichmäßig verteilten Dreiecken besteht. Das Fuzzy-Set *Regelung* besteht aus den Zugehörigkeitsfunktionen *heizen* (Maximum bei 100 Prozent), *OK* (Maximum bei 0 Prozent) und *kühlen* (Maximum bei -100 Prozent), siehe Bild 4.

Mit der sogenannten Flächen-Schwerpunkt-Methode lässt sich aus dem Regelwerk und den Fuzzy-Sets der Ausgangsvariable *Regelung* ein Stellwert gewinnen. Diese Berechnung wird auch Defuzzyfizierung genannt.

Bei 17,5 °C gilt, siehe auch Bild 5:

```
1x WENN zu kalt (= 0.8) DANN heizen (100%)
1x WENN OK (= 0.2) DANN OK (0%)
1x WENN zu warm (= 0.0) DANN kühlen (-100%)
Ausgang = (0.8 x 100% + 0.2 x 0%) / (0.8 + 0.2) = 80%
```

Bei 18,5 °C gilt:

```
1x WENN zu kalt (= 0.5) DANN heizen (100%)
1x WENN OK (= 0.5) DANN OK (0%)
1x WENN zu warm (= 0.0) DANN kühlen (-100%)
Ausgang = (0.5 x 100% + 0.5 x 0%) / (0.5 + 0.5) = 50%
```

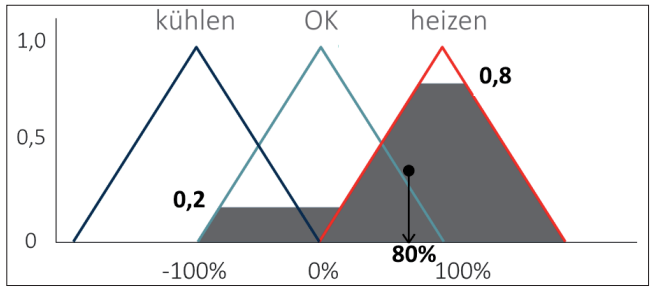
Bei 21 °C gilt:

```
1x WENN zu kalt (= 0.0) DANN heizen (100%)
1x WENN OK (= 1.0) DANN OK (0%)
1x WENN zu warm (= 0.0) DANN kühlen (-100%)
Ausgang = (1.0 x 0%) / (1.0) = 0%
```

Um die Definition der Ausgangsvariablen zu vereinfachen, gibt es ebenfalls ein paar Faustregeln:

- Verteile symmetrische Dreiecke über den Bereich der Ausgangsvariable, verwende keine Trapeze, weil diese zu toten Regelzonen führen.
- Es sollen sich immer zwei Zugehörigkeitsfunktionen überlappen, wobei die Summe der Wahrheitsgrade immer 1 ist (komplementär).
- Fange an mit drei Zugehörigkeitsfunktionen pro Fuzzy-Set. Wenn drei Funktionen nicht ausreichen, kann man auf fünf Funktionen ausweichen.

Bis jetzt wurde mit viel Aufwand ein einfacher proportionaler Regler modelliert. Interessant wird es, wenn auch die Änderung der Raumtemperatur mit einbezogen wird. Im Regelwerk lassen sich dann beide linguistischen Variablen mit einem fuzzylogischen UND verknüpfen. Bei einem fuzzylogischen UND wird der tiefste Wahrheitsgrad der beiden Variablen genommen, bei einem fuzzylogischen ODER der höchste Wahrheitsgrad.



Defuzzyfizierung basiert auf der Flächen-Schwerpunkt-Methode (Bild 5)

Jetzt kann das Regelwerk erweitert werden mit dynamischen Regeln wie dieser:

```
1x WENN (Raumtemperatur=) OK UND (Änderung=) tiefer
  DANN heizen
1x WENN (Raumtemperatur=) zu warm UND (Änderung=) höher
  DANN kühlen
```

Diese Regeln werden nur aktiv, wenn sich die Temperatur in der definierten Richtung ändert. Das dynamische Verhalten des Reglers lässt sich durch die individuelle Gewichtung der einzelnen Regeln gezielt beeinflussen.

Zusammenfassung

Dieser Artikel hat die Grundprinzipien von Fuzzylogik erklärt, die nötig sind, um eine unscharfe Lösung zu modellieren. Im zweiten Artikel in einer der nächsten dotnetpro-Ausgaben wird mit Fuzzylogik ein Regler modelliert, der auf dem klassischen PID-Regler basiert. Dieser klassischen PID-Regler wird die unterschiedlichen dynamischen Regleranforderungen eines Prozesses nur mit einem Parameter, dem D-Anteil, abdecken. Der zu modellierende PIFuzzy-Regler ermöglicht es dann, das dynamische Regelverhalten gezielt auf den Prozess abzustimmen. ■

[1] Lofti Zadeh bei Wikipedia,
www.dotnetpro.de/SL2204Fuzzy1
[2] U-Bahn Sendai bei Wikipedia,
www.dotnetpro.de/SL2204Fuzzy2

Erik Stroeken
arbeitet als Expert Software Engineer bei Noser Engineering AG in der Schweiz. Der gebürtige Niederländer ist fast 35 Jahre mit Herz und Seele in der Softwareentwicklung im industriellen Umfeld tätig.
erik.stroeken@noser.com

dnpCode A2204Fuzzy