Predmet: "Vještačka inteligencija"
Laboratorijska vježba 10: Fuzzy logika
Odgovorna nastavnica: Vanr. prof. dr Amila Akagić

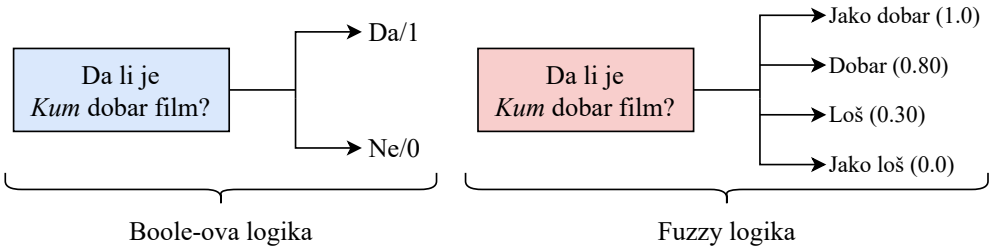
Sadržaj vježbe:

1 Cilj vježbe

U sklopu desete laboratorijske vježbe, studenti se upoznaju sa osnovima fuzzy logike. Kroz vježbu, studenti kroz postupak dizajna jednostavnog fuzzy kontrolera uče osnovne pojmove vezane za fuzzy logiku.

2 Fuzzy logika

Riječ fuzzy u doslovnom prevodu označava nešto što je nejasno. Svaki događaj, proces, ili funkcija koja se kroz vrijeme kontinualno mijenja ne može uvijek biti definisana kao tačna ili netačna, što znači da se takve aktivnosti moraju definisati na jedan drugačiji (fuzzy) način. Upravo to je osnova fuzzy logike. Fuzzy logika nastoji da oponaša proces odlučivanja kod ljudi, te se nosi sa nepreciznim i nejasnim informacijama. U suštini, predstavlja pojednostavljenje realnog svijeta bazirano na stepenu istinitosti (u odnosu na binarnu istinu, kao npr. kod Boole-ove logike). Na slici 1 je prikazana razlika između Boole-ove logike i fuzzy logike.



Slika 1: Razlika između Boole-ove (binarne) logike i fuzzy logike.

Može se reći da fuzzy logika nije logika koja je nejasna, već logka koja se koristi da se opiše nejasnost (nepreciznost) neke pojave.

2.1 Fuzzy skupovi

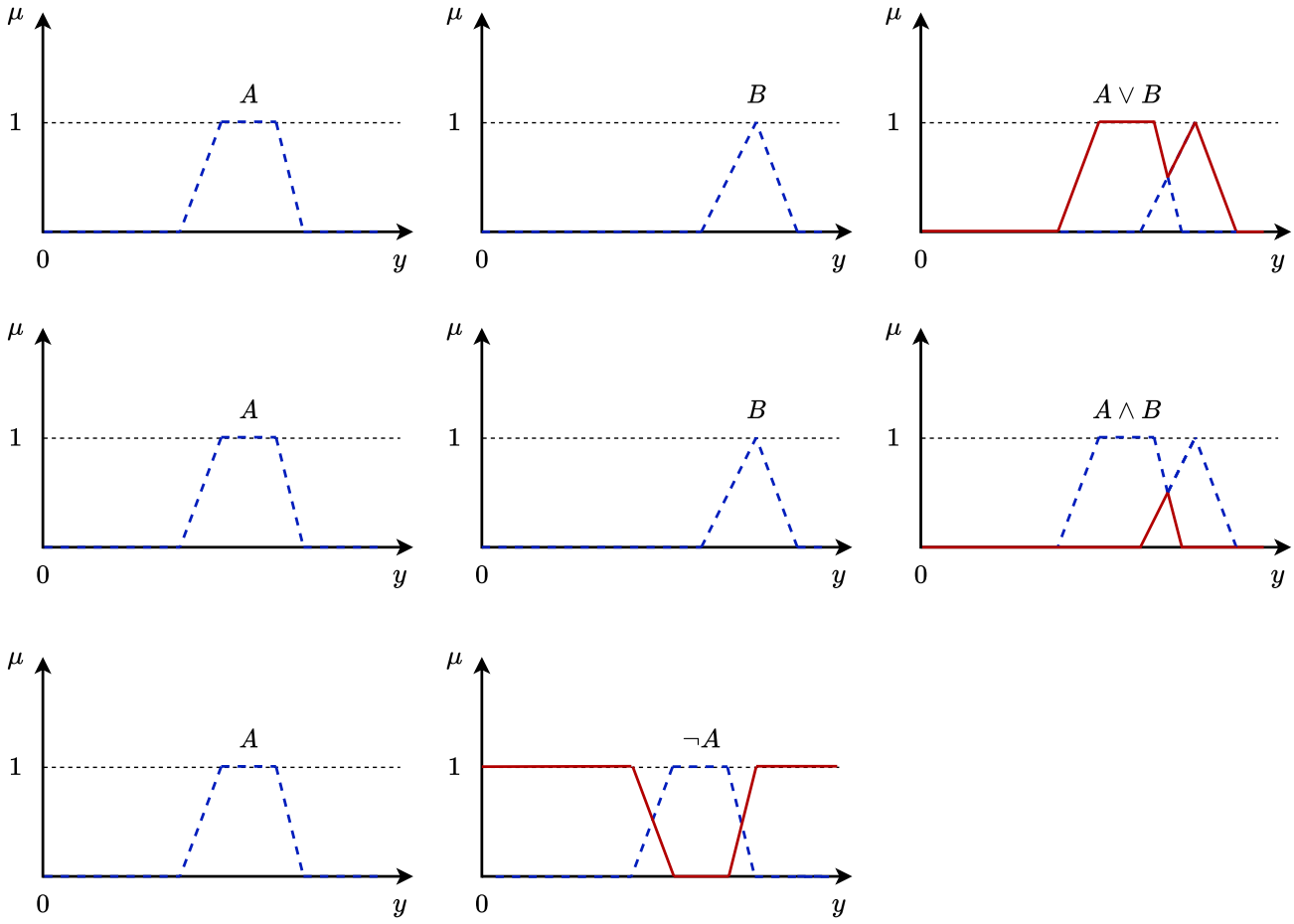
Fuzzy skupovi se mogu smatrati ekstenzijom klasičnih matematskih skupova. Najlakši način da se opiše razlika između fuzzy skupa i klasičnog skupa jeste kroz kontekst pripadnosti skupu. Dok je kod klasičnih skupova pripadnost uvijek totalna, kod fuzzy skupova to ne mora biti slučaj. To znači da, za razliku od klasičnih skupova, kod kojih važi da neki element ili jeste ili nije u skupu, fuzzy skup može u sebi da sadrži neki element sa odgovarajućim stepenom

pripadnosti. Ukoliko se uzme prethodni primjer, onda bi kod klasičnih skupova film bio ili u skupu loših, ili u skupu dobrih filmova, i to u potpunosti (film dakle može biti samo poptuno dobar ili potpuno loš). Međutim, kod fuzzy skupova važi da film može pripadati skupu dobrih filmova sa određenim stepenom pripadnosti. Veći stepen pripadnosti je i veća indikacija da element (film) više odgovara tom skupu. Na primjer, za stepen priadnosti 0.80 je rečeno da je film dobar, jer većinski pripada skupu dobrih filmova, dok je za stepen pripadnosti od 0.30 film više loš nego dobar.

Nakon što je uveden pojam fuzzy skupa, potrebno je vidjeti do kakve interakcije može doći između različitih fuzzy skupova, to jeste potrebno je definisati operacije nad fuzzy skupovima. Tri su osnovne operacije, i to:

1. Fuzzy unija (OR), koja je opisana operacijom maksimum ($\vee$);
2. Fuzzy presjek (AND), opisana operacijom minimum ($\wedge$);
3. Fuzzy komplement (NOT), opisana negacijom ($\neg$).

Na slici 2 su prikazane prethodno nabrojane operacije nad fuzzy skupovima.



Slika 2: Operacije nad fuzzy skupovima. Stepen pripadnosti se obično označava sa μ, i kreće se od 0 do 1.

2.2 Funkcije pripadnosti

Stepen pripadnosti nekom fuzzy skupu se opisuje funkcijom pripadnosti. One zapravo predstavljaju stepen istine u fuzzy logici. Funkcije pripadnosti mogu biti raznih oblika, ali su najčešće trapezoidne i trougaone. Funkcije pripadnosti se sastoje od tri dijela, i to:

1. Korijen (eng. *core*) funkcije - predstavlja region gdje je vrijednost funkcije 1, to jeste gdje postoji potpuna pripadnost fuzzy skupu ($\mu = 1$);
2. Naslon (eng. *support*) funkcije - region gdje funkcija pripadnosti ima vrijednost koja nije nula ($\mu \neq 0$);
3. Granica (eng. *boundary*) funkcije - region funkcije gdje je vrijednost pozitivna, ali nije 1 ($\mu > 0 \wedge \mu < 1$).

2.3 Fuzzy varijable

Fuzzy varijabla ima jasnu (eng. *crisp*) vrijednost, te je jednaka nekom broju unutar unaprijed definisanog domena (univerzuma). Crisp vrijednost je način na koji se gleda na ovu varijablu u "običnoj" matematici. Proces kojim se crisp vrijednost pretvara u fuzzy vrijednost (to jeste povezivanje crisp vrijednosti sa vrijednošću funkcije pripadnosti) naziva se fazifikacija. Proces suprotan tome je defazifikacija, i obično se koristi nakon sprovođenja procesa fuzzy zaključivanja, kako bi se dobio konačni rezultat. Postoji nekoliko načina na koje se može vršiti defazifikacija, a najpoznatiji su MoM (eng. *mean over maxima*) i centroid metode.

U primjeru sa filmom, crisp vrijednost bi mogla biti broj nagrada koje je film dobio, na osnovu koje se može odrediti vrijednost funkcije pripadnosti.

2.4 Fuzzy sistem upravljanja

Ostaje još vidjeti kako zapravo koristiti sve spomenute pojmove kako bi se dolazilo do zaključaka. Fuzzy kontrolni sistem zapravo povezuje fuzzy varijable i funkcije pripadnosti koristeći skup predefinisanih pravila. Ova pravila zapravo opisuju odnose između različitih fuzzy varijabli, te su napisana u `IF-THEN` sintaksi. `IF` dio se naziva uslov (eng. *antedecent*), dok se `THEN` dio naziva posljedica (eng. *consequent*). Primjer jednog fuzzy pravila je dat u nastavku:

```
IF usluga_je_dobra THEN napojnica_je_dobra
```

3 Fuzzy logika u Python-u - skfuzzy

Paket koji će biti korišten za kreiranje jednostavnih fuzzy sistema upravljanja u sklopu jezika Python je **skfuzzy** paket. Kako bi se instalirao ovaj paket, može se koristiti `pip` (odnosno `pip3`), i to pomoću sljedeće naredbe:

```
pip install -U scikit-fuzzy
```

Alternativno, ovaj paket se može direktno preuzeti sa službenog repozitorija koji se nalazi na adresi

https://github.com/scikit-fuzzy/scikit-fuzzy

Nakon preuzimanja, potrebno je izvršiti ekstrakciju datoteka, te unutar glavnog direktorija pokrenuti naredbu

```
python setup.py install
```

Nako ovoga, skfuzzy paket je instaliran. Detalji korištenja ovog paketa će biti opisani kroz zadatak.

4 Zadaci za rad u laboratoriji

Predviđeno je da studenti samostalno rješavaju zadatke.

Zadatak 1 - Fuzzy sistem upravljanja za iznos napojnice

U nekim dijelovima svijeta (prvenstveno u SAD-u), davanje napojnice konobaru za dobru uslugu je normalna pojava, i očekivano ponašanje. Obično se napojnica daje u procentima računa koji se plaća za hranu. Međutim, često je teško odrediti koliki tačno procenat dati kao napojnicu. Kako bismo riješili ovaj problem, dizajnirati ćemo jednostavan fuzzy kontroler koji će na izlazu dati procenat računa koji je potrebno izdvojiti za napojnicu.

Kada je u pitanju kreiranje samog modela, potrebno je voditi se stvarnim svijetom, te čovjekovim procesom odlučivanja. Tako će, za primjer davanja napojnice, postojati dva glavna faktora koja će da odrede veličinu napojnice: kvalitet hrane i kvalitet usluge. Pretpostavimo da se svaki od ovih faktora može ocijeniti ocjenom od 0 do 10. Pomoću ovih ocjena, davat ćemo napojnicu između 0 i 25%. Ovaj problem se može formulisati na sljedeći način:

- Ulazi:
  - Usluga:
    - Univerzum (crisp vrijednost) - kakva je bila usluga konobara na skali od 0 do 10?
    - Fuzzy skup - npr. loša, prihvatljiva, ili odlična (usluga).
  - Hrana:
    - Univerzum - koliko je ukusna bila hrana, na skali od 0 do 10?
    - Fuzzy skup - loša, prihvatljiva, odlična (hrana).
- Izlazi:
  - Napojnica:
    - Univerzum - koliko bismo trebali dati za napojnicu, na skali od 0 do 25%?
    - Fuzzy skup - mala, srednja, velika (napojnica).
- Pravila:
  - `IF usluga IS dobra OR hrana IS dobra THEN napojnica IS velika`
  - `IF usluga IS prihvatljiva THEN napojnica IS srednja`
  - `IF usluga IS loša AND hrana IS loša THEN napojnica IS mala`

a) Prvo je potrebno definisati dva ulaza i izlaz. To se može uraditi pomoću sljedećeg isječka koda:

```
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl

# Definisanje ulaza i izlaza
hrana = ctrl.Antecedent(np.arange(0, 11, 1), 'hrana')
usluga = ctrl.Antecedent(np.arange(0, 11, 1), 'usluga')
napojnica = ctrl.Consequent(np.arange(0, 26, 1), 'napojnica')
```

U ovom isječku, ulazi se definišu pomoću metode `ctrl.Antedecent`, a izlazi metodom `ctrl.Consequent`. Argumenti koje ove metode primaju su opseg vrijednosti (u našem slučaju je to od 0 do 10 za ulaze, i od 0 do 25 za izlaz), kao i naziv ulaza/izlaza);

b) Sada se ulazima i izlazima moraju pridružiti funkcije pripadnosti. To je moguće uraditi automatski ili ručno. Za ulaze ćemo funkcije pripadnosti definisati automatski, koristeći naredbu `automf` sa argumentom 3 (jer želimo po 3 funkcije pripadnosti za uslugu i hranu). Treba napomenuti da u slučaju automatskog definisanja funkcija pripadnosti, njihovi nazivi budu generisani od strane skfuzzy paketa, te će se zvati *poor*, *average*, i *good*. Za izlaz, odnosno napojnicu, definišemo ručno funkcije pripadnosti. Pošto želimo trougaone funkcije pripadnosti, koristit ćemo `trimf` metodu, kojoj je potrebno prosljediti niz od 3 rastućih vrijednosti koje opisuju tačke trougla (u prvoj i trećoj je vrijednost pripadnosti nula, dok je u drugoj jedan). Ovo se sve može postići sljedećim isječkom koda:

```
# Automatsko dodavanje funkcija pripadnosti
# pomocu automf funkcije (3, 5, ili 7 funkcija pripadnosti)
hrana.automf(3)
usluga.automf(3)

# Rucno pravljenje funkcija pripadnosti
napojnica['mala'] = fuzz.trimf(napojnica.universe, [0, 0, 13])
napojnica['srednja'] = fuzz.trimf(napojnica.universe, [0, 13, 25])
napojnica['velika'] = fuzz.trimf(napojnica.universe, [13, 25, 25])
```

c) Sada je potrebno definisati pravila. Ona se definišu pomoću metode `ctrl.Rule` te se u prvi argument šalje `IF` dio, a u drugi `THEN` dio. Konstrukcije kao što su `OR` i `AND` se dobijaju na uobičajeni način, simbolima | i &. Isječak koda koji definiše pravila je dat u nastavku:

```
# Definisanje pravila
p1 = ctrl.Rule(hrana['poor'] | usluga['poor'], napojnica['mala'])
p2 = ctrl.Rule(usluga['average'], napojnica['srednja'])
p3 = ctrl.Rule(usluga['good'] | hrana['good'], napojnica['velika'])
```

d) Nakon što su definisani svi potrebni dijelovi, sve što preostaje je kreiranje sistema upravljanja. To se radi jednostavnom naredbom:

```
# Sada se jednostavno moze kreirati sistem upravljanja
sistem = ctrl.ControlSystem([p1, p2, p3])
```

e) Kako bi se sistem mogao koristiti, potrebno ga je prevesti u simulaciju. Pomoću skfuzzy, to također postaje jednostavno:

```
# Kako bi se sistem koristio, mora se prevesti u simulaciju
sistemSim = ctrl.ControlSystemSimulation(sistem)
```

f) Konačno, moguće je izvršiti simulaciju, tako što se sistemu daju odgovarajući crisp ulazi, te se na kraju može ispisati rezultat:

```
# Sada se moze izvrsiti simulacija:
sistemSim.input['hrana'] = 6.5
sistemSim.input['usluga'] = 9.8

# Sracunaj izlaz iz fuzzy sistema
sistemSim.compute()
print(sistemSim.output['napojnica'])
```

Za vrijednosti iz isječka iznad, sistem bi trebao preporučiti davanje napojnice od oko 20%.

Zadatak 2

Ponoviti zadatak 1, ali na način da je sada maksimalna količina napojnice 50%. Također, ne koristiti automatski generisane funkcije pripadnosti za ulaze, već za prihvatljivu uslugu i hranu koristiti trapezoidnu funkciju pripadnosti koja će korijen imati na intervalu [4, 6] (pomoć: pogledati funkciju `trapmf` u službenoj dokumentaciji skfuzzy paketa[1]). Ostale funkcije pripadnosti ostaviti onakve kakve jesu, to jeste koristiti trougaone funkcije.

[1] https://scikit-fuzzy.readthedocs.io/en/latest/api/skfuzzy.html