UNIVERZITET U BIHAĆU
TEHNIČKI FAKULTET
BIHAĆ

Auditorne vježbe iz predmeta

VJEŠTAČKA INTELIGENCIJA I EKSPERTNI SISTEMI

Una Drakulić, MA elektrotehnike
Viši asistent

AGENTI ~ u vještačkoj inteligenciji AI Agents

Agent za vještačku inteligenciju je autonomni sistem koji prikuplja informacije iz okruženja putem senzora, obrađuje podatke pomoću internog modela, donosi odluke pomoću algoritama učenja ili optimizacije i/ili djeluje nad okruženjem putem aktuatora. Opisuje se funkcijom:

$$f: S \rightarrow A$$

Gdje je S skup stanja okruženja, a A skup svih mogućih rješenja.

Razlikujemo nekoliko vrsta AI agenata:

1. **Reaktivni agenti (Simple Reflex Agents)**

Reaktivni agenti donose odluke isključivo na osnovu trenutnog stanja okruženja, bez korištenja memorije prošlih događaja. Reakcije se baziraju na pravilo–akcija (if–then) strukturama.

$$a = f(s)$$

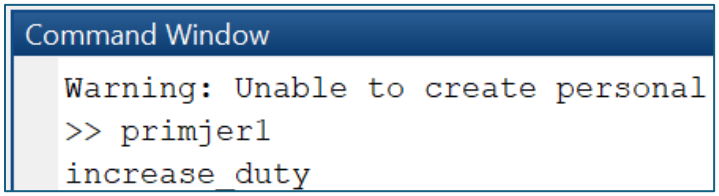
Gdje je s trenutno stanje, a a akcija.

Karakterisitike ovog agenta su da ne pamte prethodna stanja, jednostavni su i brzi, pa su zbog toga pogodni za deterministička okruženja.

Primjer1

Reaktivni agent - primjer regulacije napona

```
Vmeas = 10.5;    % izmjereni napon
Vref  = 12;

if Vmeas < Vref
    action = 'increase_duty';
else
    action = 'decrease_duty';
end

disp(action)
```



2. Model-bazirani agenti

Model-bazirani agenti posjeduju memoriju stanja i interni matematički ili logički model okruženja. Oni ne donose odluke samo na osnovu trenutnog stanja, već i koriste historiju sistema. Opisuje se funkcijom:

$$s_t^{int} = f(s_t, s_{t-1}, \dots, s_0)$$

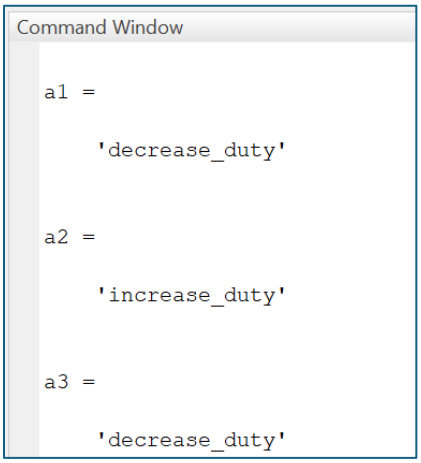
Model-bazirani agenti imaju bolje ponašanje u dinamičkim sistemima i bolju sposobnost predviđanja.

Primjer2

model_based_agent.m

```
function action = model_based_agent(Vmeas)
    % Model-bazirani agent sa memorijom

    persistent prevV

    if isempty(prevV)
        prevV = 0;
    end

    if Vmeas < prevV
        action = 'increase_duty';
    else
        action = 'decrease_duty';
    end

    prevV = Vmeas;
end
```

```
clc; clear;

a1 = model_based_agent(10.2)
a2 = model_based_agent(10.0)
a3 = model_based_agent(11.5)
```



3. Ciljno-orijentisani agenti

Ciljno orijentisani agenti djeluju na osnovu jasno definisanog cilja i koriste algoritme planiranja da bi odredili niz koraka koji vode ka cilju. Za razliku od reaktivnih agenata ovi ne reaguju samo na stanje već razmatraju buduća stanja.

Opisuje se funkcijom:

$$A = \{a_1, a_2, \dots, a_n\}$$

Primjer 3

```
% na primjer cilj je dostizanje referentnog napona

Vtarget = 12;
V = 9;

while abs(V - Vtarget) > 0.1
    if V < Vtarget
        V = V + 0.5;
    else
        V = V - 0.5;
    end
end

disp("Cilj postignut: " + V)
```

```
>> primjer3
Cilj postignut: 12
```

4. Utility-based agenti

Utility-based agenti ne biraju akcije samo da bi postigli cilj, već da bi maksimizirali funkciju korisnosti (utility function). Ovi agenti su pogodni za optimizaciju.

Funkcija korisnosti:

$$U(s) = \omega_1 x_1 + \omega_2 x_2 + \cdots + \omega_n x_n$$

Primjer4

```
V = 11.7;
I = 3.2;

w1 = 0.7;
w2 = 0.3;

U = w1*(1/abs(12-V)) + w2*I;

disp("Utility vrijednost: " + U)
```

```
>> primjer4
Utility vrijednost: 3.2933
```

5. Učeći agenti (Learning Agents)

Učeći agenti predstavljaju najnapredniju klasu agenata u vještačkoj inteligenciji. Njihova osnovna karakteristika je sposobnost da kontinuirano unapređuju svoje ponašanje na osnovu iskustva i podataka prikupljenih iz okruženja. Za razliku od klasičnih agenata koji rade po unaprijed definisanim pravilima, učeći agenti adaptivno prilagođavaju svoje odluke promjenama u sistemu.

Struktura učećeg agenta

Tipična arhitektura učećeg agenta sastoji se od četiri osnovne komponente:

- **Learning element (element učenja)** – odgovoran za modifikaciju internog modela i poboljšanje strategije odlučivanja na osnovu novih podataka.
- **Performance element (element izvršavanja)** – donosi konkretne odluke i bira akcije na osnovu trenutne politike ponašanja.
- **Critic (kritičar)** – analizira rezultat izvršenih akcija i daje povratnu informaciju u obliku nagrade ili kazne.
- **Problem generator (generator problema)** – podstiče istraživanje novih strategija ponašanja radi pronalaženja boljih rješenja.

Algoritmi koje koriste učeći agenti

Učeći agenti se oslanjaju na napredne algoritme mašinskog učenja, među kojima se najčešće koriste:

- **Neuronske mreže** – omogućavaju modeliranje složenih nelinearnih odnosa između ulaznih i izlaznih veličina sistema.
- **Reinforcement Learning (učenje poticanjem)** – omogućava agentu da uči optimalne strategije kroz interakciju sa okruženjem na osnovu sistema nagrada i kazni.
- **Genetski algoritmi** – koriste principe prirodne selekcije za optimizaciju parametara modela i ponašanja agenta.

Zadaci za vježbanje

Zadatak1

Razviti učećeg agenta zasnovanog na neuronskoj mreži koji na osnovu izmjerenih ulaznih veličina sunčeve radijacije (G), temperature (T) i stanja napunjenosti baterije (SOC) donosi odluku o upravljačkom signalu duty cycle za DC-DC pretvarač u fotonaponskom sistemu. Cilj agenta je da održi stabilan izlazni napon sistema na vrijednosti od 12 V. Agent mora naučiti vezu između ulaznih parametara i potrebnog upravljačkog signala na osnovu dostupnih podataka.

```
clc; clear; close all;

%% 1. Trening podaci (simulirani)
% Ulazi: [G; T; SOC]
G   = linspace(200,1000,100);
T   = linspace(10,40,100);
SOC = linspace(0.3,1,100);

X = [G; T; SOC];

% Željeni izlaz - duty cycle (simulacija)
Y = 0.4 + 0.0005*G - 0.003*T + 0.2*SOC;

%% 2. Kreiranje učećeg agenta (neuronske mreže)
net = feedforwardnet(10);
net.trainParam.epochs = 300;

%% 3. Treniranje agenta
net = train(net, X, Y);

%% 4. Testiranje agenta
testInput = [800; 25; 0.7];
duty = net(testInput);

disp("Izlaz agenta - duty cycle:");
disp(duty);
```

```
Command Window
  Izlaz agenta - duty cycle:
      0.8812
```

Zadatak2

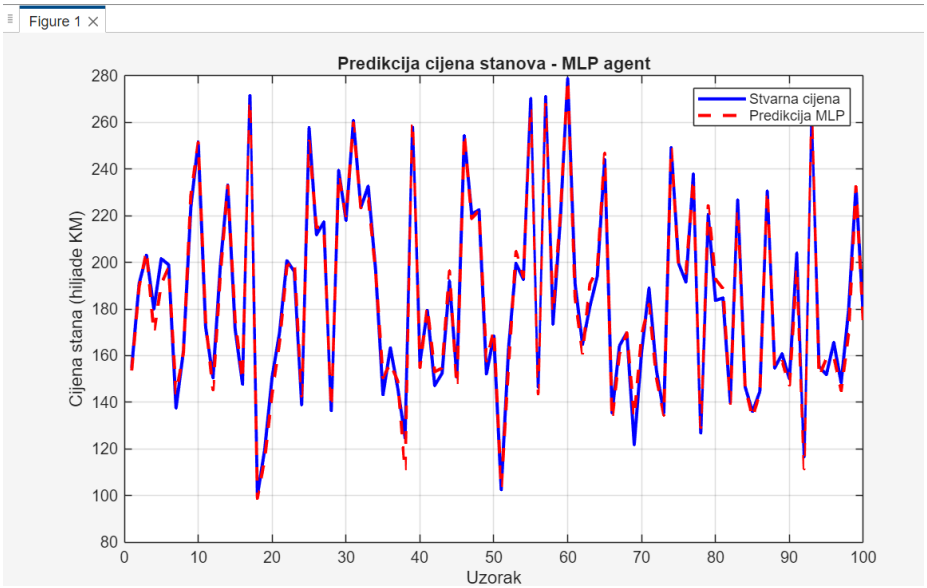
Kreirati i naučiti MLP neuronsku mrežu da predviđa cijenu stana na osnovu njegovih karakteristika. Napraviti ulaznu matricu X i ciljnu vrijednost Y (simulirani podaci), pa testirati mrežu na novim podacima i prikazati predikcije i stvarne vrijednosti.

Ulazi (features):

1. Površina stana (m^2)
2. Broj soba
3. Sprat
4. Starost zgrade (godine)

Izlaz (target): Cijena stana (u hiljadama KM)

```
clc; clear; close all;

%% Simulirani podaci
% Broj uzoraka
N = 100;

% Ulazi: Površina, Broj soba, Sprat, Starost zgrade
Povrsina = randi([40,150],1,N);     % m2
Sobe     = randi([1,5],1,N);        % broj soba
Sprat    = randi([1,10],1,N);       % sprat
Starost  = randi([0,50],1,N);       % godine

X = [Povrsina; Sobe; Sprat; Starost];

% Ciljna cijena (simulirana funkcija + šum)
Y = 50 + 1.2*Povrsina + 10*Sobe + 2*Sprat - 0.5*Starost + 5*randn(1,N);

%% Kreiranje i treniranje MLP mreže
net = feedforwardnet(10);   % 1 skriveni sloj sa 10 neurona
net.trainParam.epochs = 500;
net = train(net, X, Y);

%% Testiranje na novim podacima
testX = [80; 3; 2; 10];   % primjer stana
predPrice = net(testX);

disp("Predviđena cijena stana (hiljade KM):");
disp(predPrice);

%% Prikaz rezultata: Predikcije vs stvarne vrijednosti
Y_pred = net(X);

figure;
plot(Y,'b','LineWidth',2); hold on;
plot(Y_pred,'r--','LineWidth',2);
grid on;
xlabel('Uzorak');
ylabel('Cijena stana (hiljade KM)');
title('Predikcija cijena stanova - MLP agent');
legend('Stvarna cijena','Predikcija MLP');
```



<u>Zadatak3</u> Predikcija uključivanja ventilatora pomoću MLP mreže

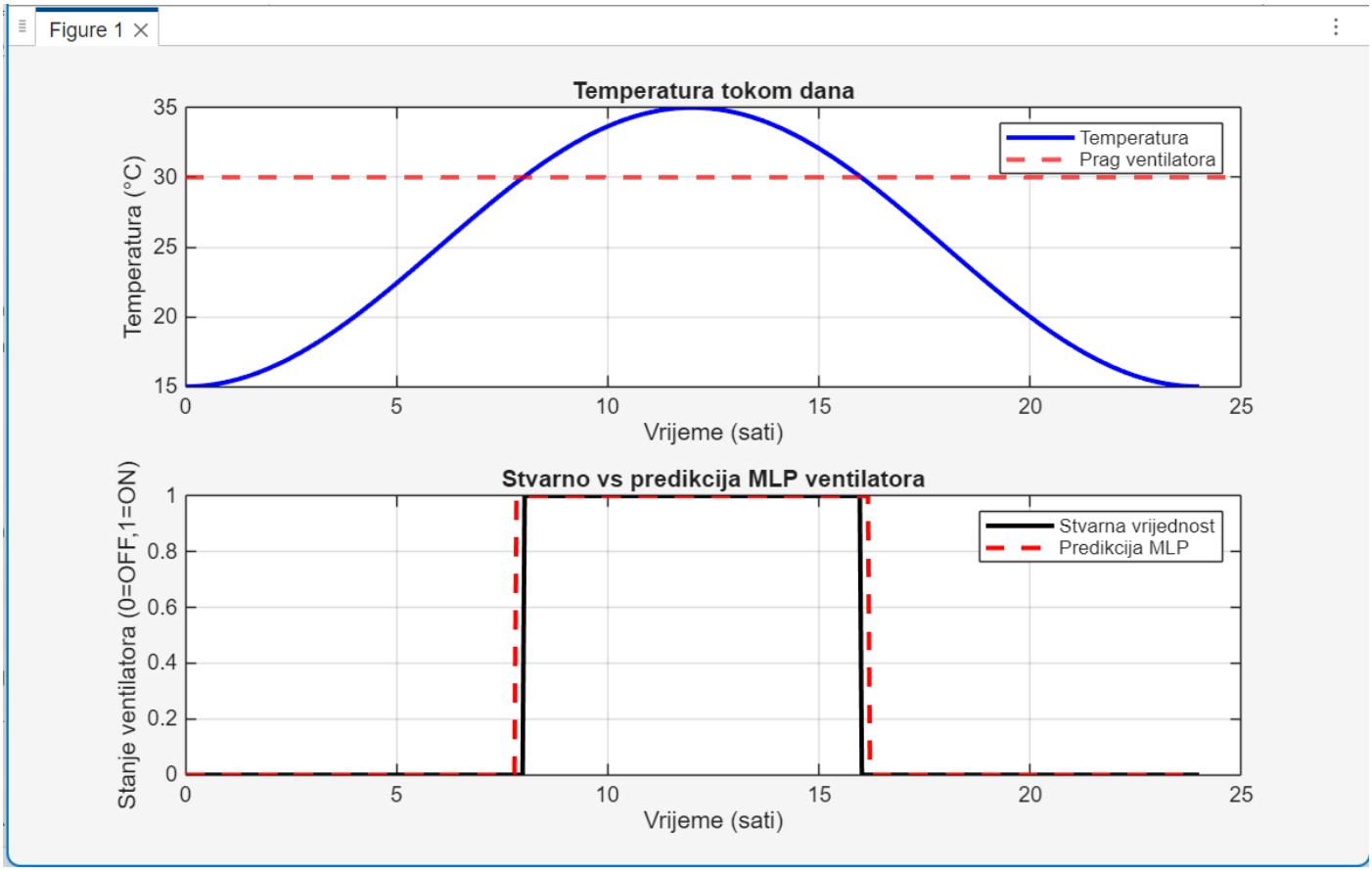
Potrebno je automatski upravljati ventilatorom na osnovu temperature. Ventilator se uključuje kada temperatura prelazi prag od 30°C i isključuje kada temperatura padne ispod praga. Potrebno je kreirati MLP (multilayer perceptron) neuronsku mrežu koja na osnovu ulaznog signala temperature tokom dana predviđa stanje ventilatora (0 = OFF, 1 = ON). Mreža treba naučiti vezu između temperature i uključivanja ventilatora i pravilno predviđati kada ventilator treba da se uključi ili isključi.

Potrebno je:

1. Generisati deterministički niz temperatura koji simulira promjene tokom dana: od minimalne temperature od 15°C do maksimalne od 35°C.
2. Definisati stvarno stanje ventilatora:
   - 1 (ON) ako je temperatura ≥ 30°C
   - 0 (OFF) ako je temperatura < 30°C
3. Kreirati MLP sa dva skrivena sloja (10 i 5 neurona) i ReLU aktivacijom u skrivenim slojevima. Izlazni sloj koristi logsig aktivaciju.
4. Trenirati mrežu na svim uzorcima temperature.
5. Predikcija mreže treba da bude binarna (0 ili 1) i da što preciznije prati stvarno stanje ventilatora.
6. Prikažite rezultate kroz graf:
   - Subplot 1: temperatura tokom dana sa crvenom isprekidanom linijom koja označava prag od 30°C.
   - Subplot 2: stvarno stanje ventilatora i predikcija MLP-a.

Predikcija mreže treba da bude logična i precizna (tačnost > 95%). Može se koristiti prag 0.5 za konverziju izlaza MLP mreže u binarni signal.

```
clc; clear; close all;

%% 1. Generisanje podataka
N = 500;                      % veći broj uzoraka
t = linspace(0,24,N);         % vrijeme u satima

% Temperatura tokom dana (deterministički sinus)
T = 25 + 10*sin(pi*(t-6)/12); % od 15 do 35°C

% Stvarno stanje ventilatora (1=ON, 0=OFF)
Fan = double(T >= 30);

% Ulazna matrica (samo temperatura za jednostavnost)
X = T;

%% 2. Kreiranje MLP
hiddenLayerSize = [10 5];    % dva sloja: 10 neurona + 5 neurona
net = feedforwardnet(hiddenLayerSize);

% Aktivacija slojeva: ReLU (poslin) za bolje ucenje naglih promjena
for i = 1:length(net.layers)-1
    net.layers{i}.transferFcn = 'poslin';
end
% Izlazni sloj: logsig za 0-1
net.layers{end}.transferFcn = 'logsig';

% Trening
net.trainFcn = 'trainlm';
net.trainParam.epochs = 1000;

%% 3. Treniranje mreže
net = train(net, X, Fan);

%% 4. Predikcija
Y_pred = net(X);
Y_class = double(Y_pred > 0.5);   % prag 0.5 daje binarni izlaz

%% 5. Tačnost
accuracy = sum(Y_class == Fan)/N * 100;
fprintf('Tačnost MLP: %.2f %%\n', accuracy);

%% 6. Grafički prikaz
figure;

% Subplot 1: temperatura
subplot(2,1,1);
plot(t, T,'b-','LineWidth',2); hold on;
yline(30,'r--','LineWidth',2);   % prag ventilatora
grid on;
xlabel('Vrijeme (sati)');
ylabel('Temperatura (°C)');
title('Temperatura tokom dana');
legend('Temperatura','Prag ventilatora');

% Subplot 2: ventilator
subplot(2,1,2);
plot(t, Fan,'k-','LineWidth',2); hold on;
plot(t, Y_class,'r--','LineWidth',2);
grid on;
xlabel('Vrijeme (sati)');
ylabel('Stanje ventilatora (0=OFF,1=ON)');
title('Stvarno vs predikcija MLP ventilatora');
legend('Stvarna vrijednost','Predikcija MLP');
```



Zadatak 4

Razviti MLP neuronsku mrežu koja predviđa sentiment:

- 1 = pozitivna recenzija
- 0 = negativna recenzija

Tekst datoteka **reviews.txt** (20 recenzija)

I absolutely love this product, it works perfectly and exceeded my expectations
The item was terrible, I am very disappointed and would not recommend it
Fantastic quality, I am very happy with my purchase and everything works great
This is the worst experience I have ever had, very poor design and functionality
I am impressed, excellent performance and perfect for my needs
Awful, it broke after two uses and I hate it
Good value for money, really satisfied with the product
Terrible service, the item arrived damaged and support was awful
I highly recommend this, fantastic and awesome product
I dislike this product, very bad quality and poor performance
Perfect, just what I needed and works as described
I am very disappointed, it did not meet my expectations
Great item, excellent design and I love using it
Worst purchase ever, terrible quality and I hate it
Amazing product, very happy with the results and recommend it
Poor build, bad materials and disappointing overall
I am satisfied, good quality and works perfectly

Awful experience, would never buy again
Fantastic, excellent, and perfect, very happy
Terrible, worst item I have bought, do not recommend

```
clc; clear; close all;

%% 1. Ucitavanje fajla
fid = fopen('reviews.txt','r');
C = textscan(fid,'%s','Delimiter','\n');
fclose(fid);
reviews = C{1};
N = length(reviews);

%% 2. Lista pozitivnih i negativnih rijeci
positiveWords =
{'good','excellent','fantastic','love','great','happy','reco
mmend','awesome','perfect'};
negativeWords =
{'bad','poor','terrible','hate','awful','worst','disappointe
d'};

%% 3. Kreiranje ulaznih podataka
X = zeros(2,N); % 2 ulaza: broj pozitivnih i negativnih
rijeci
Y = zeros(1,N); % ciljni izlaz: 0=negativno, 1=pozitivno

for i = 1:N
    text = lower(reviews{i});
    words = split(text);
    posCount = sum(ismember(words,positiveWords));
    negCount = sum(ismember(words,negativeWords));
    X(:,i) = [posCount; negCount];

    % Pravilo za cilj
    if posCount > negCount
        Y(i) = 1;
    else
        Y(i) = 0;
    end
end

%% 4. Kreiranje i trening MLP
net = feedforwardnet(5); % skriveni sloj sa 5 neurona
net.trainParam.epochs = 500;
net = train(net,X,Y);

%% 5. Predikcija
Y_pred = net(X);
Y_class = double(Y_pred > 0.5);
```

```
%% 6. Tacnost
accuracy = sum(Y_class == Y)/N * 100;
fprintf('Tacnost MLP: %.2f %%\n', accuracy);

%% 7. Graf predikcija vs stvarno stanje
figure;
stem(1:N,Y,'k','LineWidth',2); hold on;
stem(1:N,Y_class,'r--','LineWidth',1.5);
xlabel('Recenzija'); ylabel('Klasa (0=neg,1=poz)');
title('Stvarno stanje vs predikcija MLP');
legend('Stvarna klasa','Predikcija MLP');
grid on;
```

Od ukupnog broja 20 recenzija: