Inteligentni SAU

Žarko Zečević
Elektrotehnički fakultet
Univerzitet Crne Gore

Predavanje 1

Uvod u inteligentni SAU

Ishodi učenja:

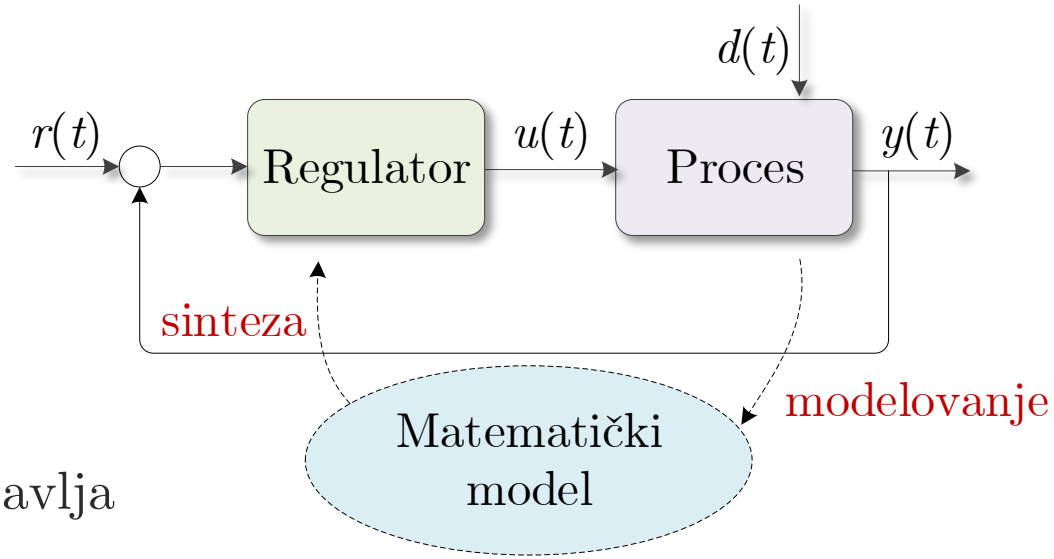
Nakon savladavanja gradiva sa ovog predavanja studenti će moći da:

- Razumiju razliku između klasičnog i inteligentnog upravljanja sistemima
- Matematički modeluju vještački neuron i neuralne mreže
- Naprave razliku između raziličitih topologija neuralnih mreža i njihovih mogućnosti za rješavanje različitih tipova problema

Klasično vs inteligentno upravljanje

Kod klasičnog pristupa upravljanju potrebno je odrediti matematički model procesa, na osnovu kojeg se dizajnira kontroler.

upravljanje zasnovano na poznavanju modela procesa



$y(t)$ – varijabla kojom se upravlja

$u(t)$ – upravljački signal

$d(t)$ – eksterni signali na koje ne možemo da utičemo

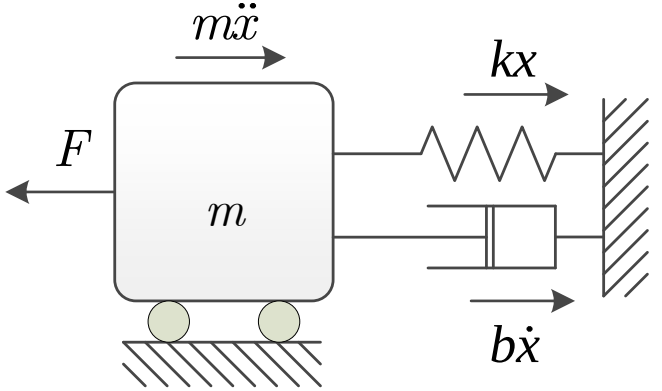
Postoje dva glavna pristupa za određivanje modela sistema:

- fizičko modelovanje,
- identifikacija sistema.

Klasično vs inteligentno upravljanje

Fizičko modelovanje sistema:

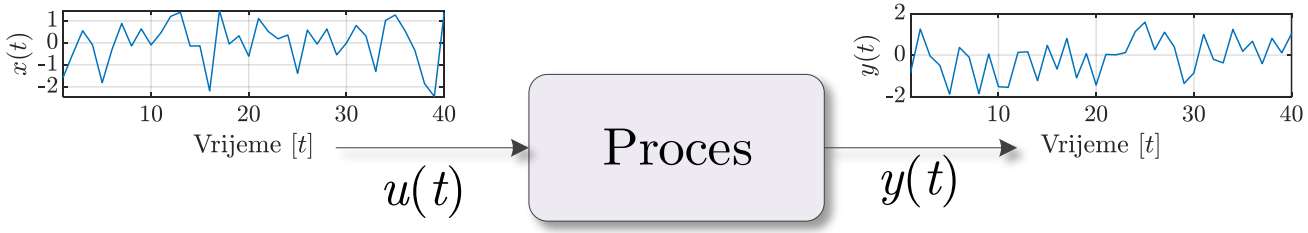
1. Primjenjivanje fizičkih zakona – diferencijalne jednačine (linearne i nelinearne)
2. Linearizacija u okolini radne tačke



$$m\ddot{x} + b\dot{x} + kx = F$$

Identifikacija sistema:

1. Mjerenje ulaza i izlaza
2. Pretpostavljanje strukture modela
3. Estimacija parametara pretpostavljenog modela



$$\hat{m}\ddot{y} + \hat{b}\dot{y} + \hat{k}y = u(t)$$

Na osnovu mjerenja ulaza i izlaza naprednim numeričkim metodama se estimiraju parametri modela $\hat{m}, \hat{b}$ i $\hat{k}$.

Klasično vs inteligentno upravljanje

Sinteza kontrolera se vrši na osnovu dobijenog modela.

Neki od ciljeva/zahtjeva koje kontroler treba da zadovolji su:

- Stabilizacija nestabilnog procesa
- Redukcija uticaja poremećaja
- Poboljašanje performansi (povećanje brzine odziva, smanjenje greške)

Potrebno je odabrati željenu šemu upravljanja i tip kontrolera (lag, lead, PID, state-space, itd.), a zatim podesiti njegove parametre.

Prednosti klasičnog upravljanja:

- Sistemski pristup, matematički elegantan
- Teorijske granice za stabilnost i robustnost

Nedostaci klasičnog upravljanja:

- Potrebno je dosta vremena za savladavanje, apstraktni koncepti
- Ne može se primijeniti na sisteme koji su visoko nelinearni

Klasično vs inteligentno upravljanje

Klasična teorija upravljanja se ne može primijeniti:

- Kada model proces nije dostupan
  - Tada je nemoguće odraditi matematičku sintezu i analizu SAU-a
  - Ekspermintalno podešavanje prametara kontrolera može biti teško
- Dinamika procesa je visoko nelinearna
  - Linearni kontroler ne može da stabilizuje proces
  - Postoje ograničenja u performansama

Primjer: simulirajte u Simulinku sljedeći proces:

$$\frac{d^3y(t)}{dt^3} + \frac{d^2y(t)}{dt^2} + \frac{dy(t)}{dt} = y^2(t)u(t)$$

- Stabilnost i performanse zavise od željenog izlaza
- Ne može se ustabiliti proporcionalnim kontrolerom
- Potrebna je neka nelinearna tehnika upravljanja

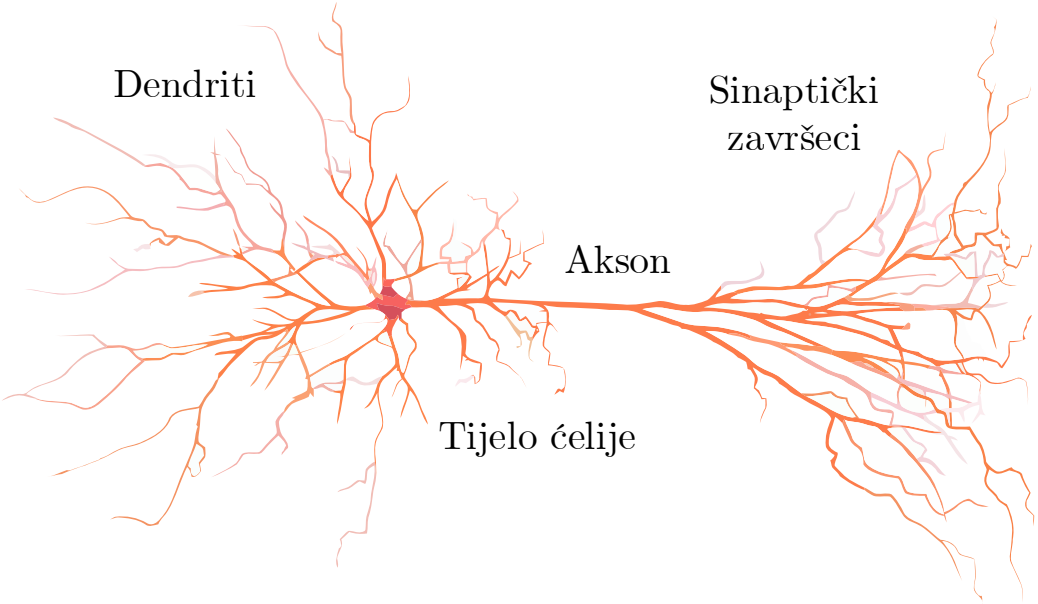
Klasično vs inteligentno upravljanje

Inteligentno upravljanje je klasa upravljanja koja koristi različite tehnike inspirisane ljudskom inteligencijom:

- Neuralne mreže
  - računarski modeli koji imaju sposobnost učenja i generelizacije znanja
  - mogu se primijeniti za rješavanje problema identifikacije sistema i sinteze inteligentnih kontrolera
- Fuzzy logika
  - principi takozvane „zamagljene“ logike koji omogućuvaju ugradnju znanja eksperata u SAU
  - implementacija inženjerskog iskustva u algoritam kontrolera
- Evolucioni algoritmi
  - otimizacija korišćenjem tehnika insprisanih prirodnom evolucijom (genetički algoritmi, Particle swarm optimization (PSO), itd.)
  - Koriste se za optimizaciju parametara kontrolera

Uvod u neuralne mreže

Vještačke neuralne mreže su inteligentni sistemi/računarski modeli koji su nastali po uzoru na način funkcionisanja čovjekovog mozga. Biološki neuron se sastoji od dendritskog stabla koje sakuplja električne impulse (ulaze) sa drugih neurona, i aksona koji preko sinaptičkih završetaka prenosi impulse dalje. Ljudski mozak sadrži oko 10 milijardi neurona i 60 biliona sinaptičkih veza. Neuralne mreže predstavljaju primitivnu imitaciju bioloških mreža. Sastoje od velikog broja međusobno povezanih vještačkih neurona. Njihova glava karakteristika je da mogu da uče, adapitaju se, čuvaju i generalizuju stečeno znanje.



Matematički model neurona

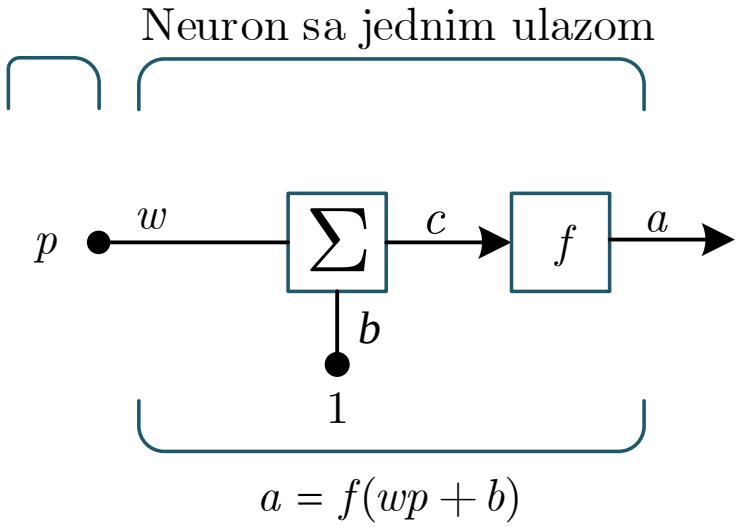
Matematički model neurona sa jednim ulazom je prikazan na slici. Ulazni signal je označen sa p, dok je težinski koeficijent koji množi ulazni signal označen sa w. Sa c je označen izlaz iz *linearnog kombinatora*:

$$c = wp + b,$$

pri čemu b predstavlja bajas. Konačno izlaz iz neurona je jednak:

$$a = f(wp + b),$$

gdje f označava aktivacionu funkciju.
Aktivaciona funkcija određuje kakav će biti izlaz iz neurona u zavisnosti od izlaza iz linearnog kombinatora. Aktivacione funkcije mogu biti linearne i nelinearne, a biraju se u skladu sa problemom koji se rješava neuralnom mrežom. Najšešće se koriste tri aktivacione funkcije: funkcija praga, sigmoidalna funkcija i linearna funkcija.

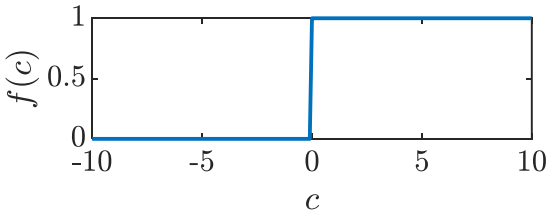


Matematički model neurona

- Funkcija praga

Za ovu vrstu aktivacione funkcije važi:

$$f(c) = \begin{cases} 1 & \text{za } c \geq 0, \\ 0 & \text{za } c < 0. \end{cases}$$



U skladu sa tim, izlaz iz neurona će biti jednak:

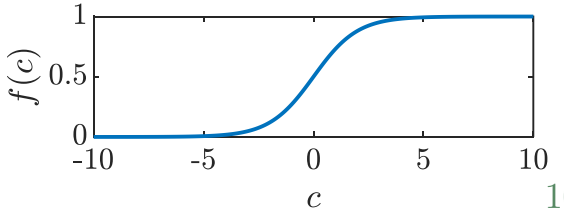
$$a = \begin{cases} 1 & \text{za } c \geq 0, \\ 0 & \text{za } c < 0. \end{cases}$$

Ova funkcija se obično koristi kada neuroni treba da klasifikuju ulazne podatke u neku kategoriju.

Sigmoidalna funkcija:

Ovaj tip aktivacione funkcije predstavlja jednu od najčešće korišćenih funkcija pri projektovanju neuralnih mreža. Jedna vrsta sigmoidalne funkcije je logaritamska-sigmodialna funkcija:

$$f(c) = \frac{1}{1 + e^{-n}}.$$



Matematički model neurona

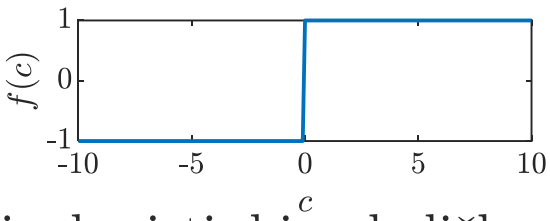
Linearna funkcija:

Linearna funkcija se najčešće koristi u zadnjem sloju višeslojnih mreža. Kod ove funkcije važi: $f(c)=c$.

Za razliku od pragovske funkcije, ova funkcija mapira ulazne podatke na interval od 0 do 1. Također, sigmoidalna funkcija je diferencijabilna, dok funkcija praga nije.

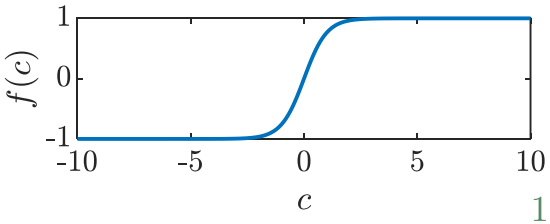
Pomenute aktivacione funkcije preslikavaju ulazne podatke na interval od 0 do 1. Nekada je potrebno proširiti ovaj interval kako bi se kretao od -1 do 1. Tada se funkcija praga definiše jednačinom:

$$\varphi(c)=\begin{cases}1 & \text{za } c>0,\\ 0 & \text{za } c=0,\\ -1 & \text{za } c<0,\end{cases}$$



dok se umjesto logaritamske sigmoidalne funkcije koristi hiperbolička sigmoidalna funkcija:

$$f(c)=\frac{e^{c}+e^{-c}}{e^{c}-e^{-c}}=\tanh(c).$$



Matematički model neurona

Neuron može da ima više ulaza. Neuron sa R ulaza je prikazan na slici. Ulazni signali su označeni sa p_i, dok su odgovarajući težinski koeficijenti označeni sa $w_{1,j}$.

Izlaz iz linearnog kombinatora je jednak:

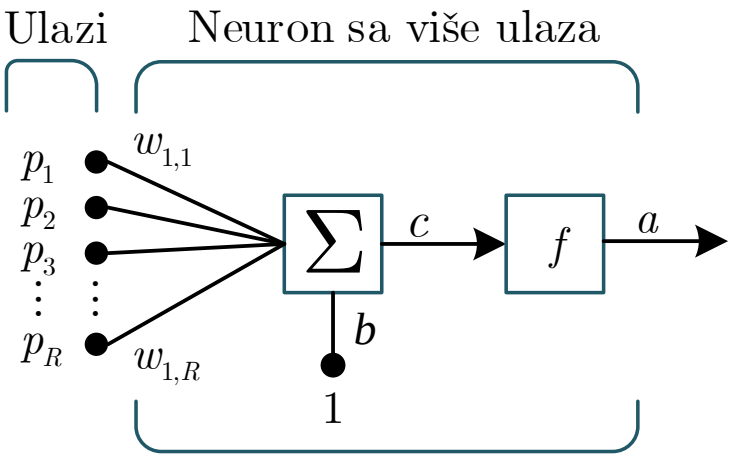
$$c = w_{11}p_1 + w_{12}p_2 + \cdots w_{1R}p_R + b.$$

Prethodna jednačina se može zapisati u vektorskom obliku:

$$c = \mathbf{wp} + b.$$

Konačno, izlaz je jednak:

$$a = f(c) = f(\mathbf{wp} + b).$$



$$\mathbf{p} = \begin{bmatrix} p_1 \\ \vdots \\ p_R \end{bmatrix}$$

$$\mathbf{w} = \begin{bmatrix} w_{1,1} & \cdots & w_{1,R} \end{bmatrix}$$

$$a = f(\mathbf{wp} + b)$$

Primjer – izlaz iz neurona sa dva ulaza

Neuron sa dva ulaza je zadat jednačinom:

$$y = \sigma\left(-4.79x_1 + 5.90x_2 - 0.93\right) = \sigma\left(\mathbf{W}\mathbf{x} + b\right)$$

U Matlab-u kreirati zadati NN model, a zatim nacrtati zavisnost $y(x_1, x_2)$ na opsegu [-2, 2]×[-2 2]. Uporediti dvije aktivacione funkcije: funkciju praga i sigmoidalnu funkciju.

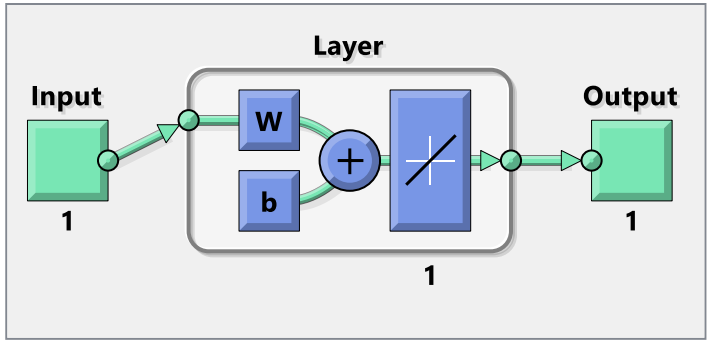
`newlin` *`(P,S)`* – kreira jedan sloj sa linearnom aktivacionom funkcijom
P – matrica dimenzija R×Q, R – broj ulaza, Q broj neurona
S – broj izlaza

```
>> net1=newlin([1;1],1)
>> view(net1)
>> net1.layers{1}.transferFcn = 'hardlim';
```

Linearni sloj sa dva ulaza

Prikaz neuralne mreže

Ovom komandom se mijenja aktivaciona funkcija. Neke od mogućih varijanti su 'logsig' i 'purelin'.



Primjer – izlaz iz neurona sa dva ulaza

U konkretnom primjeru neuron se kreira pomoću sljedeće komande:

```
>> net1=newlin([1;1],1)
```

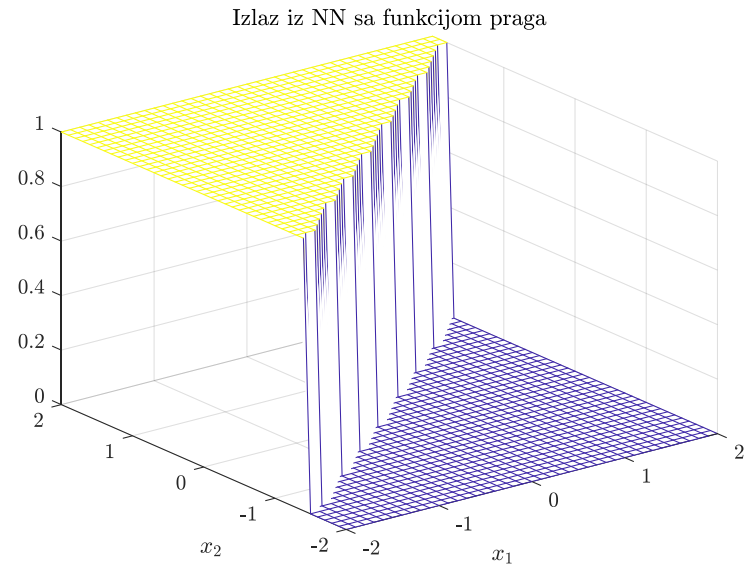
Težinski koeficijenti i bajas se podešavaju na sljedeći način:

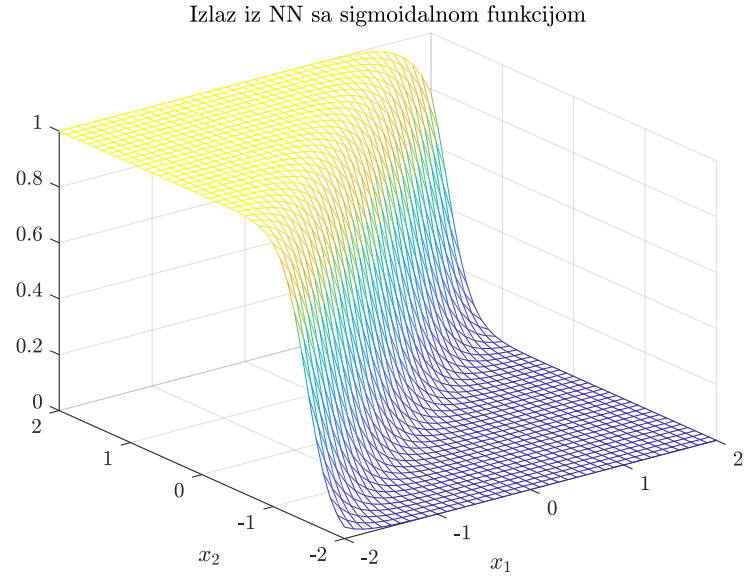
```
>> net1.IW{1,1}=[-4.79 5.9];
>> net1.b{1}=[-0.93];
```

Ako su x_1=[1 2 3], a x_2=[2 3 4], tada se izlaz mreže računa na sljedeći način:

```
>> y1 = sim(net1,[x1;x2]);
```

Ispod su prikazani izlazi neurona za različite vrijednosti x_1 i x_2





Primjer – izlaz iz neurona sa dva ulaza

```
close all
% definisinje tačaka za koje ćemo da nacrtamo izlaz
[x1,x2] = meshgrid(-2:0.1:2);
p1 = x1(:); p2 = x2(:); % pretvaranje matrica u kolone
p = [p1';p2’]; % i
% Podesavanje tezinskih koeficijenata i bijasa
net1 = newff(minmax(p),[1],{'hardlim'});
net1.IW{1,1}=[-4.79 5.9];
net1.b{1}=[-0.93];
net2=net1;
net2.layers{1}.transferFcn = 'logsig';
%Simulacija NN
y1 = sim(net1,p);
y2 = sim(net2,p);
% Crtanje izlaza u zavisonosti od x1 i x2:
a1 = eye(41); a1(:) = y1';
a2 = eye(41); a2(:) = y2';
mesh(x1,x2,a1); xlabel('$x_1$'); ylabel('$x_2$');
title('Izlaz iz NN sa funkcijom praga');
figure(2)
mesh(x1,x2,a2);
xlabel('$x_1$');
ylabel('$x_2$');
title('Izlaz iz NN sa sigmoidalnom funkcijom');
```

Arhitektura neuralnih mreža

Neuralne mreže mogu da sadrže veliki broj neurona. Način na koji su neuroni struktuirani u neralnim mrežema direktno je vezan za algoritme kojima se neuralne mreže obučavaju.

Generalno se, sa aspekta arhitekture (strukture), neuralne mreže se mogu svrstati u jednu od tri kategorije:

- **feedforward mreže sa jednim slojem**: (eng. Single-Layer Feedforward Networks),
- **višeslojne feedforward mreže** (eng. Multilayer Feedforward Networks, FFNN),
- **rekurentne mreže** (eng. Recurrent Networks, RNN).

Prilikom projektovanja neuralnih mreža od velikog značaja je određivanje njihove arhitekture i metoda učenja. Oni se određuju u zavisnosti od zadatka za koji se koristi neuralna meža. To može biti u procesu klasifikacije, aproksimacije funkcija, filtriranja podataka, upravljanja i slično.

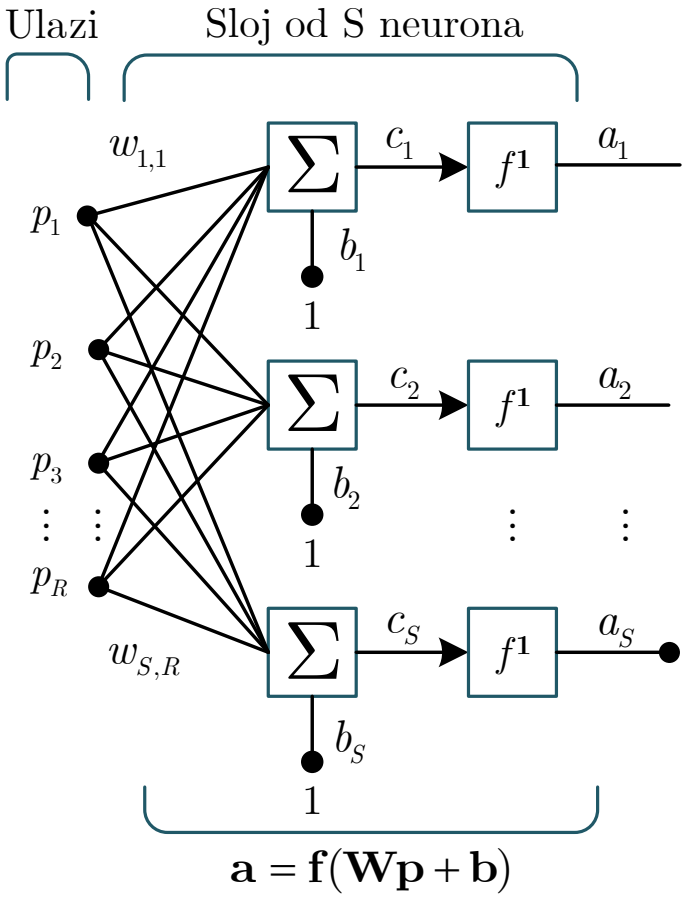
Jednoslojna feedforward mreža

Neuroni su obično organizovani u slojeve na takav način da jedan sloj čini više neurona. Na slici je prikazana jednoslojna mreža koja ima S neurona i R ulaza. Na ulaz mreže dolaze ulazni podaci, dok izlazi iz sloja predstavljaju izlaze mreže. Zavisnost izlaza od ulaznih signala se i u ovom slučaju može zapisati u matričnom obliku:

$$\mathbf{a} = \mathbf{f}(\mathbf{c}) = \mathbf{f}(\mathbf{W}\mathbf{p} + \mathbf{b}).$$

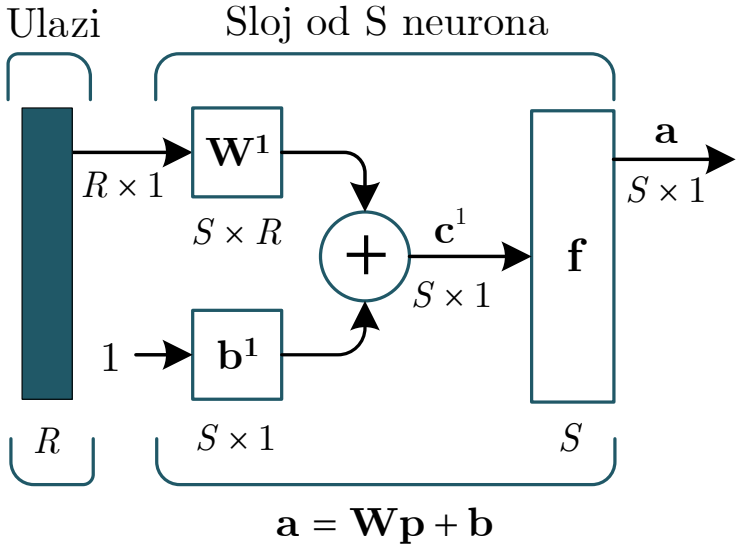
U prethodnoj jednačini $\mathbf{a}$ je vektor izlaznih signala, $\mathbf{p}$ vektor ulaznih signala, $\mathbf{b}$ vektor bijasa, $\mathbf{f}$ vektor aktivacionih funkcija, a $\mathbf{W}$ matrica težinskih koeficijenata:

$$\mathbf{W} = \begin{bmatrix} W_{11} & W_{12} & \cdots & W_{1R} \\ W_{21} & W_{22} & \cdots & W_{2R} \\ \cdots & \cdots & \cdots & \cdots \\ W_{S1} & W_{S2} & \cdots & W_{SR} \end{bmatrix}_{S\times R}.$$



Jednoslojna feedforward mreža

Radi jednostavnosti, sloj od S neurona i R ulaza ćemo prikazivati kao na slici ispod.



Obratiti pažnju da je matrica $\mathbf{W}$ dimenzija $S{\times}R$, odnosno da prva vrsta matrice sadrži koeficijente koji definišu izlaz iz prvog neurona. Vektor p jed dimenzija $R{\times}1$, dok su ostali vektori dimenzija $S{\times}1$. Treba napomenuti aktivacione funkcije neurona mogu da budu različite, mada se najčešće za jedan sloj koristi ista aktivaciona funkcija.

Višeslojna feedforward mreža

Neuralna mreža može da ima više slojeva neurona. Ukoliko samo prvi sloj mreže prima podatke, pri čemu se ti podaci prosljeđuju na ulaz sljedećeg sloja, i tako redom, onda se radi o višeslojnoj feedforward mreži. Termin *feedforward* se koristi da označi da podaci teku u smjeru od ulaza ka izlazu, odnosno da nema povratnih sprega.

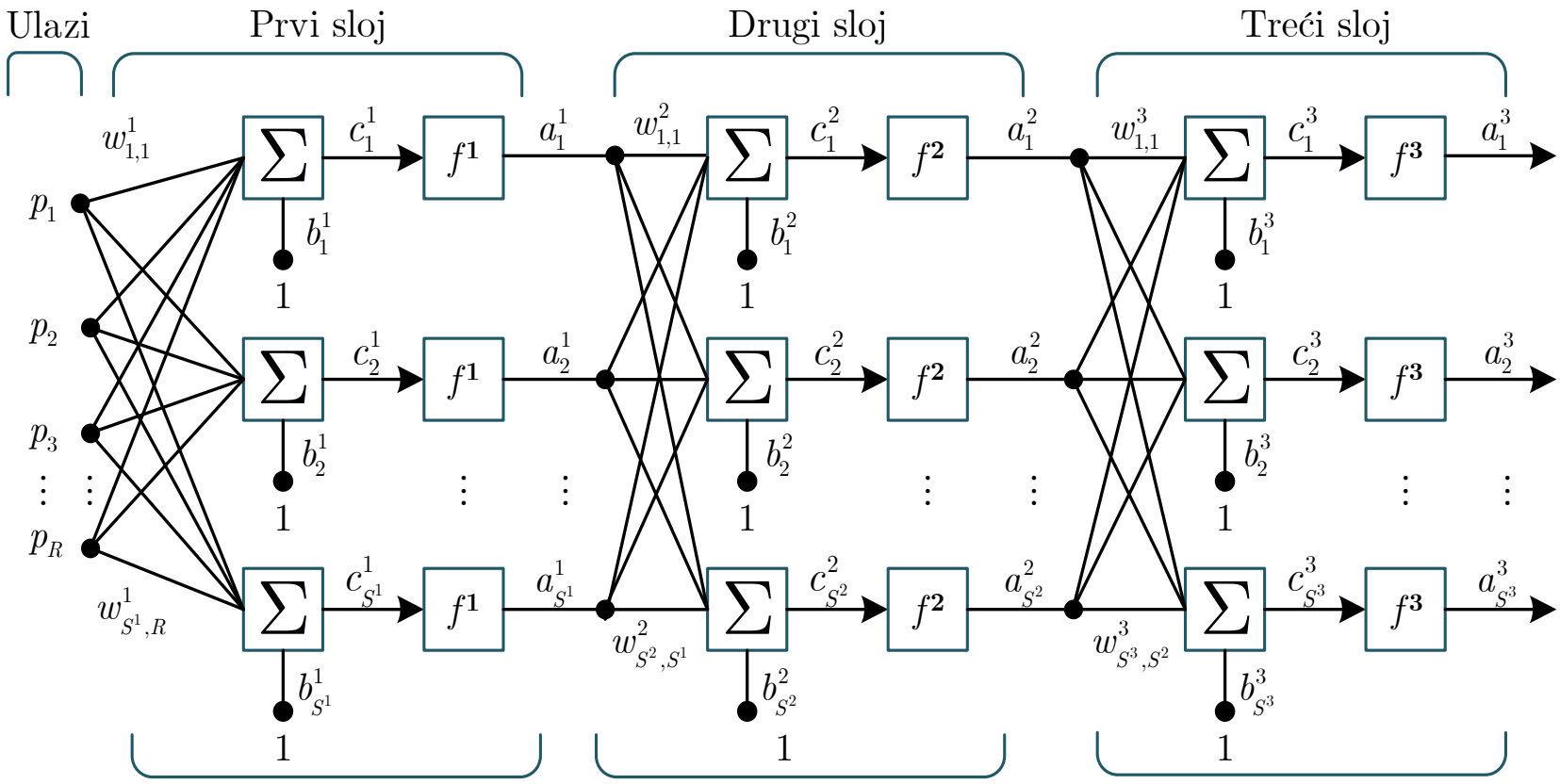
Zadnji sloj mreže se zove izlazni sloj i on najčešće koristi linearnu aktivacionu funkciju. Ostali slojevi se zovu skriveni slojevi i kod njih se najčešće koriste nelinearne aktivacione funkcije. Neki autori pod jednim slojem podrazumijevaju i ulazne podatke.

Dodavanjem jednog ili više skrivenih slojeva povećava se preciznost mreže i njena sposobnost da riješi složenije probleme. Sa druge strane višeslojne mreže su računski kompleksnije i teže su za obučavanje.

Iz literature je poznato da dvoslojna neuralna mreža, ukoliko sadrži dovoljan broj neurona, može da aproksimira bilo koju nelinearnu funkciju. Međutim, ovo ne znači da uvijek možemo odrediti odgovarajuće koeficijente mreže.

Višeslojna feedforward mreža

Na slici je ispod je prikazana mreža koja ima ima više slojeva. Svaki sloj ima svoju težinsku matricu $\mathbf{W}^k$, vektor bajasa $\mathbf{b}^k$, vektor aktivacionih funkcija $\mathbf{f}^k$ i vektor izlaza $\mathbf{a}^k$.

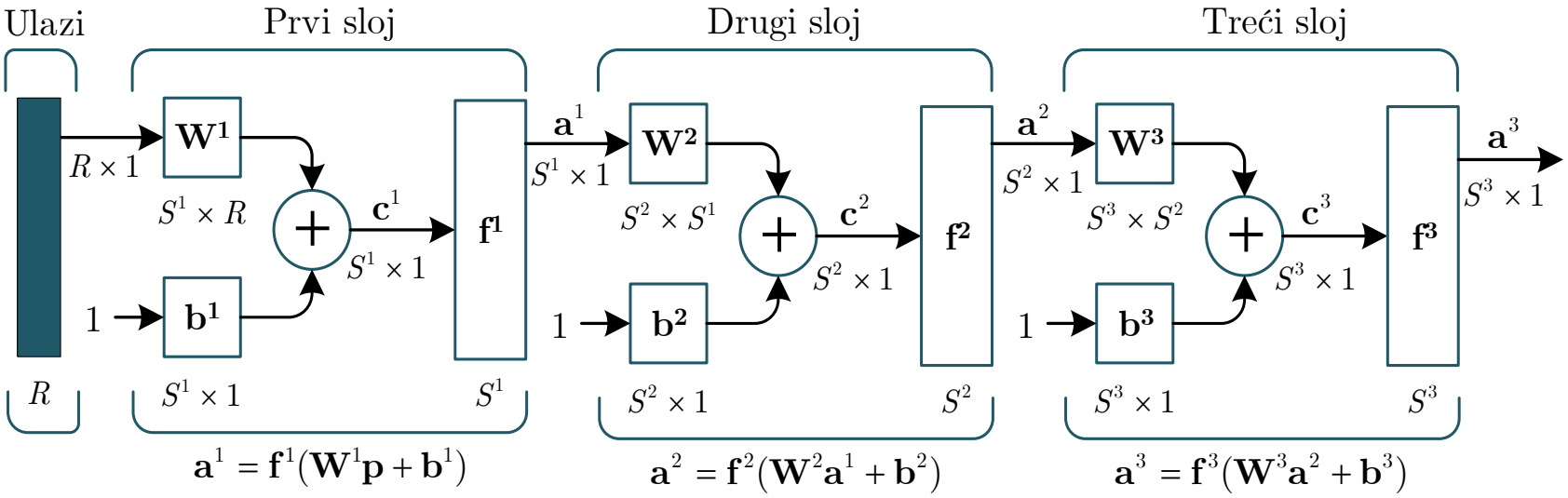


$$\mathbf{a}^3 = \mathbf{f}^3(\mathbf{W}^3(\mathbf{f}^2(\mathbf{W}^2(\mathbf{f}^1(\mathbf{W}^1\mathbf{p} + \mathbf{b}^1)) + \mathbf{b}^2)) + \mathbf{b}^3)$$

Višeslojna feedforward mreža

Uprošćeni dijagram prethodne mreže je dat ispod. U konkretnom primjeru radi se o troslojnoj neuralnoj mreži. Ovaka mreža, kod koje se slojevi nadovezuju jedan drugi se naziva **feedforward neuralna mreža (FFNN)**. Termin feedforward se odnosi na to da je mreža statička, bez memorije, odnosno da nema povratnih sprega. Izlaz u posmatranom trenutku zavisi samo od trenutnih vrijednosti ulaza:

$$\mathbf{a}^3 = \mathbf{f}^3(c) = \mathbf{f}^3(\mathbf{W}^3(\mathbf{f}^2(\mathbf{W}^2(\mathbf{f}^1(\mathbf{W}^1\mathbf{p} + \mathbf{b}^1)) + \mathbf{b}^2)) + \mathbf{b}^3).$$



Primjer – izlaz iz dvoslojne FFNN

Dvoslojna NN sa dva ulaza i jednim izlazom je data jednačinom:

$$y = \mathbf{W}_2\sigma\left(\mathbf{W}_1\mathbf{x} + b_1\right) + b_2,$$

gdje su:

$$\mathbf{W}_1 = \begin{bmatrix} -2.69 & -2.80 \\ -3.39 & -4.56 \end{bmatrix}, \ \mathbf{b}_1 = \begin{bmatrix} 2.21 \\ 4.76 \end{bmatrix},$$

$$\mathbf{W}_2 = \begin{bmatrix} 4.91 & 4.95 \end{bmatrix}, \ \mathbf{b}_2 = -2.28.$$

U Matlab-u kreirati zadati NN model, a zatim nacrtati zavisnost $y(x_1,x_2)$ na opsegu [-2, 2]×[-2 2]. Uporediti dvije aktivacione funkcije: funkciju praga i sigmoidalnu funkciju.

Zadata neuralna mreža jedan skriveni sloj sa dva neurona, i jedan izlazni sloj sa jednim neuronom. U Matlab-u se može definisati preko sljedeće komande:

```
>> net1=feedforwardnet(2)
```

Komanda *feedworward* podrazumijeva izlazni sloj, pa se on ne unosi kao argument. Matlab ne dozvoljava inicijalizaciju koeficijenata, već se mreži moraju proslijediti ulazni i izlazni podaci, na osnovu kojih se konfiguriše broj ulaza i izlaza, i početne vrijednost koeficijenata.

Primjer – izlaz iz dvoslojne FFNN

Pošto NN ima dva ulaza i jedan izlaz, možemo definisati sljedeće vektore slučajnih brojeva:

```
>> p=randn(2,20); % broj ulaza × broj podataka
>> t=randn(1,20); % broj izlaza × broj podataka
```

Mreža se konfiguriše na sljedeći način:

```
>> net1=configure(net1,p,t)
```

Izgled mreže i njene koeficijente možemo prikazati na sljedeći način:

```
>> view(net1)
>> net1.IW{1}; % koeficijenti skrivenog sloja
>> net1.LW{2}; % koeficijenti izlaznog sloja
>> net1.b{1}; net1.b{2}; bajasi skrivenog i izlaznog sloja
```

Podrazumjevana aktivaciona funkcija skrivenih slojeva je sigmoidalni tangens, dok se kod linearnog sloja podrazumijeva linearna funkcija. Ako želimo da promijenimo aktivacione funkcije, to možemo odraditi na sljedeći način:

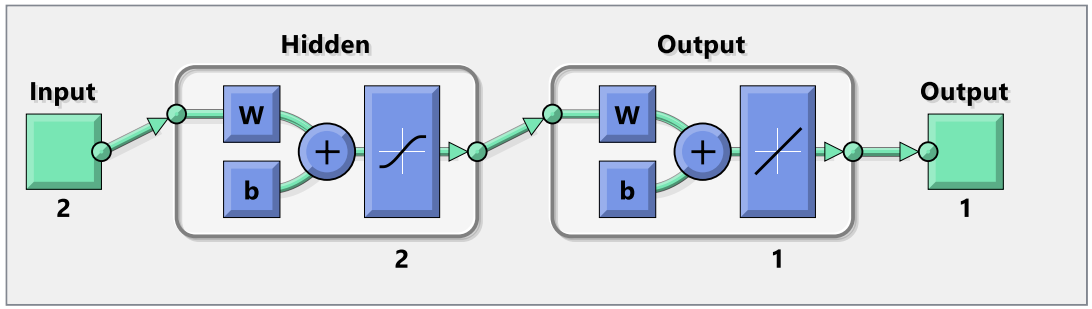
```
>> net1.layers{1}.transferFcn='logsig';
>> net1.layers{1}.transferFcn='linear';
```

Primjer – izlaz iz dvoslojne FFNN

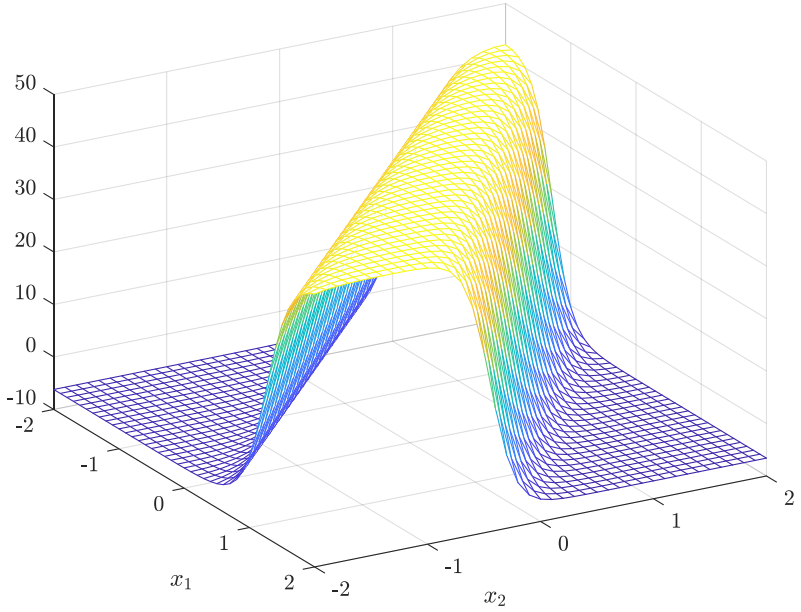
Izlaz iz neuralne mreže za neke proizvoljne ulazne podatke možemo dobiti na sljedeći način:

```
>> p=randn(2,10);
>> sim(net,p)
```

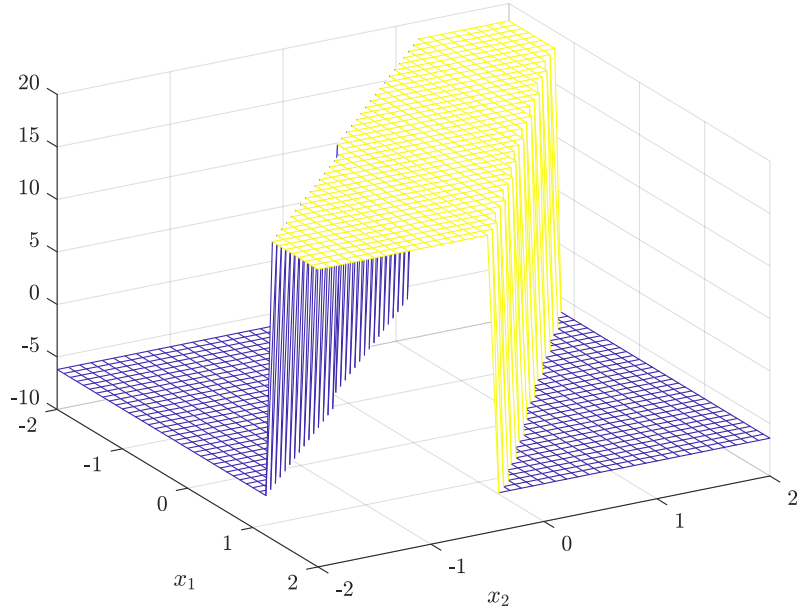
Na slikama je prikazana zadata NN i njeni izlazi za tražene ulazne podatke (x_1, x_2) .



Izlaz iz NN sa sigmoidalnom funkcijom



Izlaz iz NN sa sigmoidalnom funkcijom



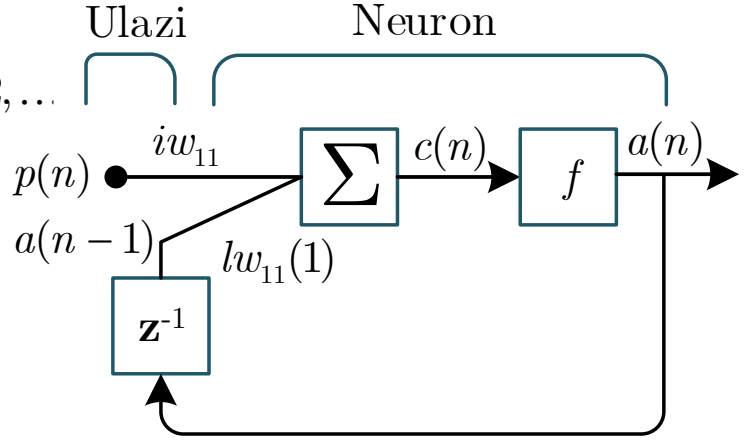
Primjer – izlaz iz dvoslojne FFNN

```
close all
[x1,x2] = meshgrid(-2:0.1:2);
% tacke u kojim zelimo da izracunamo izlaz
p1 = x1(:); p2 = x2(:); p = [p1'; p2'];
net1=feedforwardnet(2)
[x,y] = simplefit_dataset; % dataset za konfiguraciju mreze
W1=[-2.69 -2.80;-3.39 -4.56]; W2=[-4.91 4.95];
b1=[-2.21;4.76]; b2=-2.28;
net1.inputs{1}.processFcns={};
net1=configure(net1,[x;x],t);
net1.IW{1}=W1; net1.b{1}=b1; net1.b{2}=b2; net1.LW{2}=W2;
net2=net1; net2.layers{1}.transferFcn='hardlim';
%Simulacija NN
y1 = sim(net1,p); y2 = sim(net2,p);
% Crtanje izlaza u zavisinosti od x1 i x2:
a1 = eye(41); a1(:) = y1'; a2 = eye(41); a2(:) = y2';
mesh(x1,x2,a1);
AZ = 60; EL = 30; view(AZ,EL);
xlabel('$x_1$'); ylabel('$x_2$');
title('Izlaz iz NN sa funkcijom praga');
figure(2)
mesh(x1,x2,a2);
xlabel('$x_1$'); ylabel('$x_2$'); view(AZ,EL);
title('Izlaz iz NN sa sigmoidalnom funkcijom');
```

Rekurentne neuralne mreže

Dinamičke ili rekurentne neuralne mreže (RNN) se razlikuju od feedforward mreža po tome što imaju najmanje jednu povratnu spregu. Jedan primjer neurona sa povratnom spregom je prikazan na slici ispod.

$$\mathbf{p} = \begin{bmatrix} p(1) & p(2) & \dots \end{bmatrix}, n = 1, 2, \dots$$

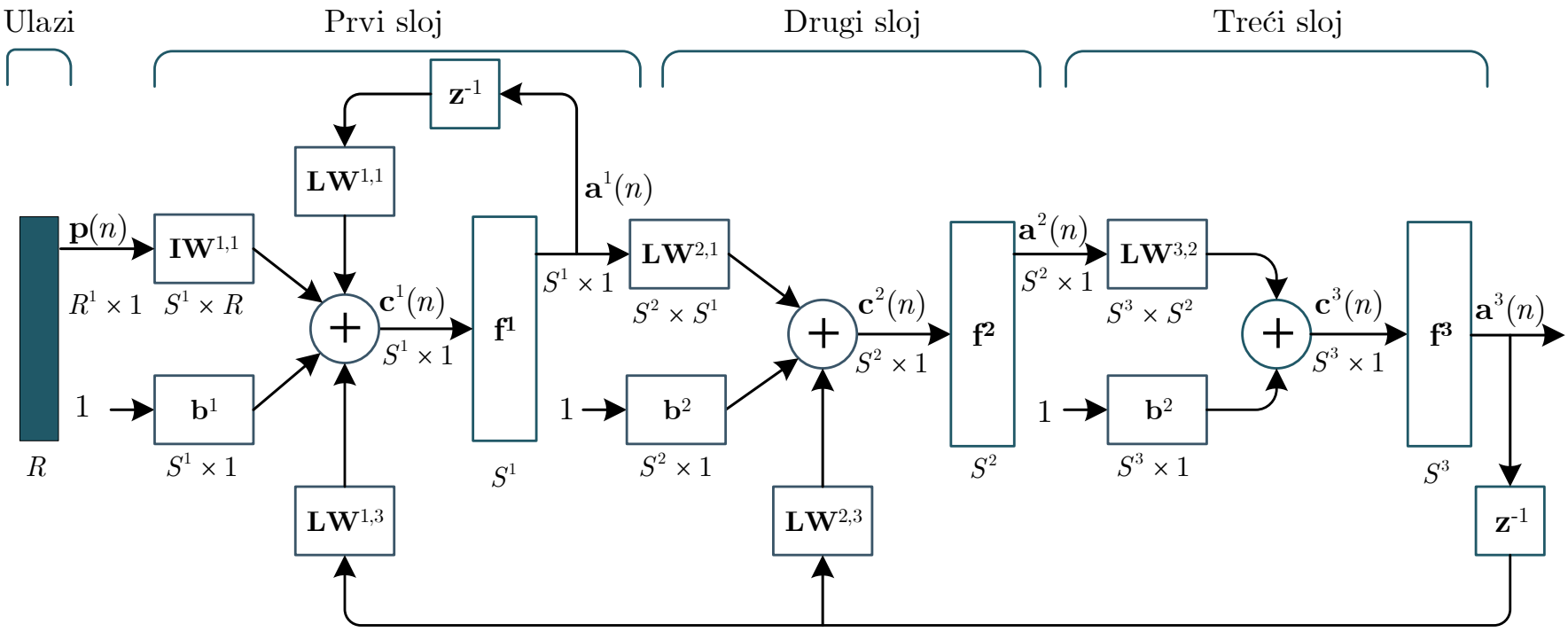


Blok $\mathbf{z}^{-1}$ označava kašnjenje. Ako je ulaz u blok za kašnjenje $a(n)$, tada će njegov izlaz biti $a(n\text{-}1)$, odnosno ovaj blok memoriše izlaz mreže iz prethodne iteracije/vremenskog trenutka. Kako dinamičke mreže imaju memoriju, one mogu da generišu nenulte izlaze i onda kada su ulazi jednaki nuli. Izlaz iz gornjeg neurona se može zapisati na sljedeći način:

$$a(n) = f\left(iw_{11}p(n) + lw_{11}(1)a(n-1)\right).$$

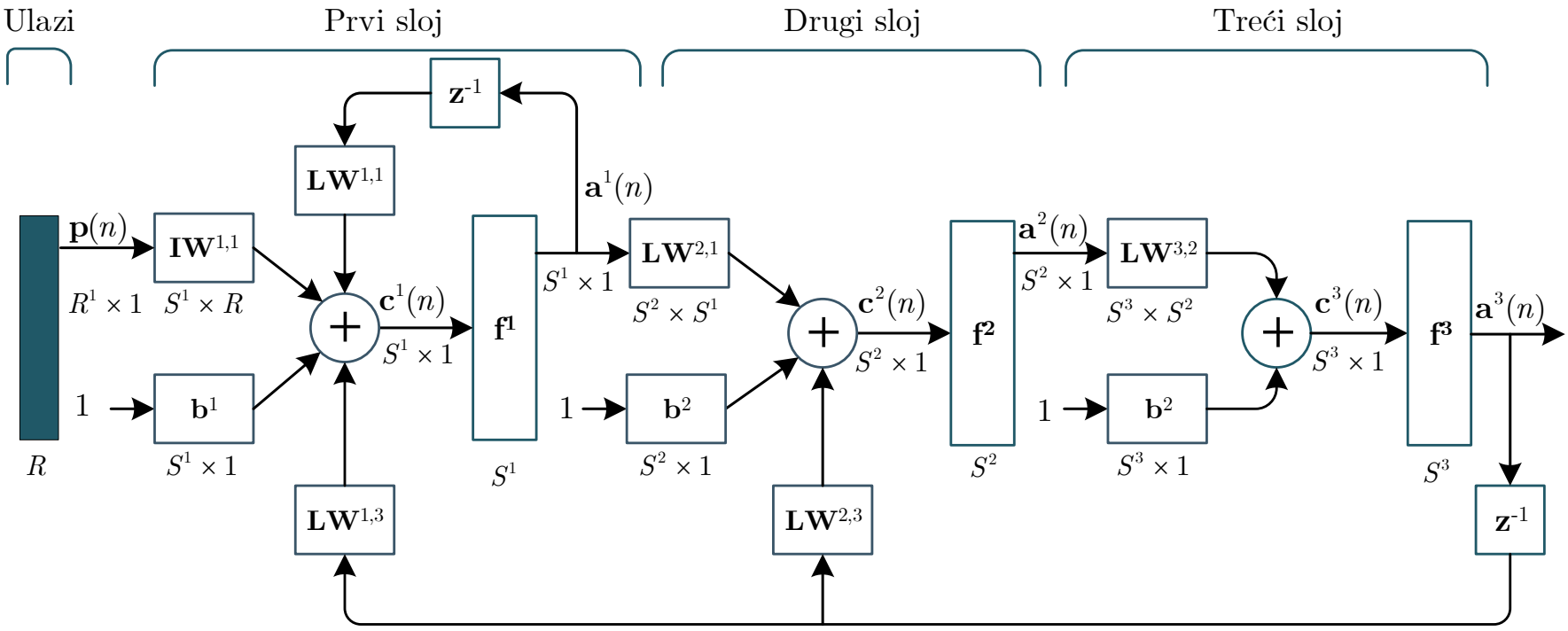
Rekurentne neuralne mreže

Ispod je dat primjer rekurentne mreže sa tri sloja. Radi jednostavnosti je usvojena sljedeća notacija. Koeficijenti koji množe ulaze su označeni sa $\mathbf{IW}^{i,1}$, gdje i označava redni broj sloja na koji je povezan ulaz. Na primjer, ukoliko su ulazni podaci povezani na prvi sloj, tada će koeficijenti koji množe ulaze biti označeni sa $\mathbf{IW}^{1,1}$.



Rekurentne neuralne mreže

Koeficijenti koji množe izlaze nekog sloja biće označeni sa $\mathbf{LW}^{i,j}$, gdje i označava redni broj sloja na koji se dovode podaci, dok je j redni broj sloja sa kojeg se uzimaju podaci. Na primjer, $\mathbf{LW}^{1,1}$ znači da se izlaz iz prve mreže vraća na njen ulaz, dok $\mathbf{LW}^{2,3}$ znači da se izlaz sa trećeg sloja vraća na ulaz drugog sloja. Naravno, povratne petlje sadrže blokove za kašnjenje.



Rekurentne neuralne mreže

Izlaz iz prvog sloja prethodne mreže u iteraciji/trenutku n se može zapisati pomoću sljedeće diferencne jednačine:

$$\mathbf{a}^1(n) = \mathbf{f}^1\left(\mathbf{IW}^{11}\mathbf{p}(n) + \mathbf{LW}^{11}\mathbf{a}^1(n-1) + \mathbf{LW}^{13}\mathbf{a}^3(n-1)\right).$$

Izlaz iz drugog sloja je jednak:

$$\mathbf{a}^2(n) = \mathbf{f}^2\left(\mathbf{LW}^{21}\mathbf{a}^1(n) + \mathbf{LW}^{23}\mathbf{a}^3(n-1)\right).$$

Konačno, izlaz iz trećeg sloja, odnosno izlaz iz mreže je:

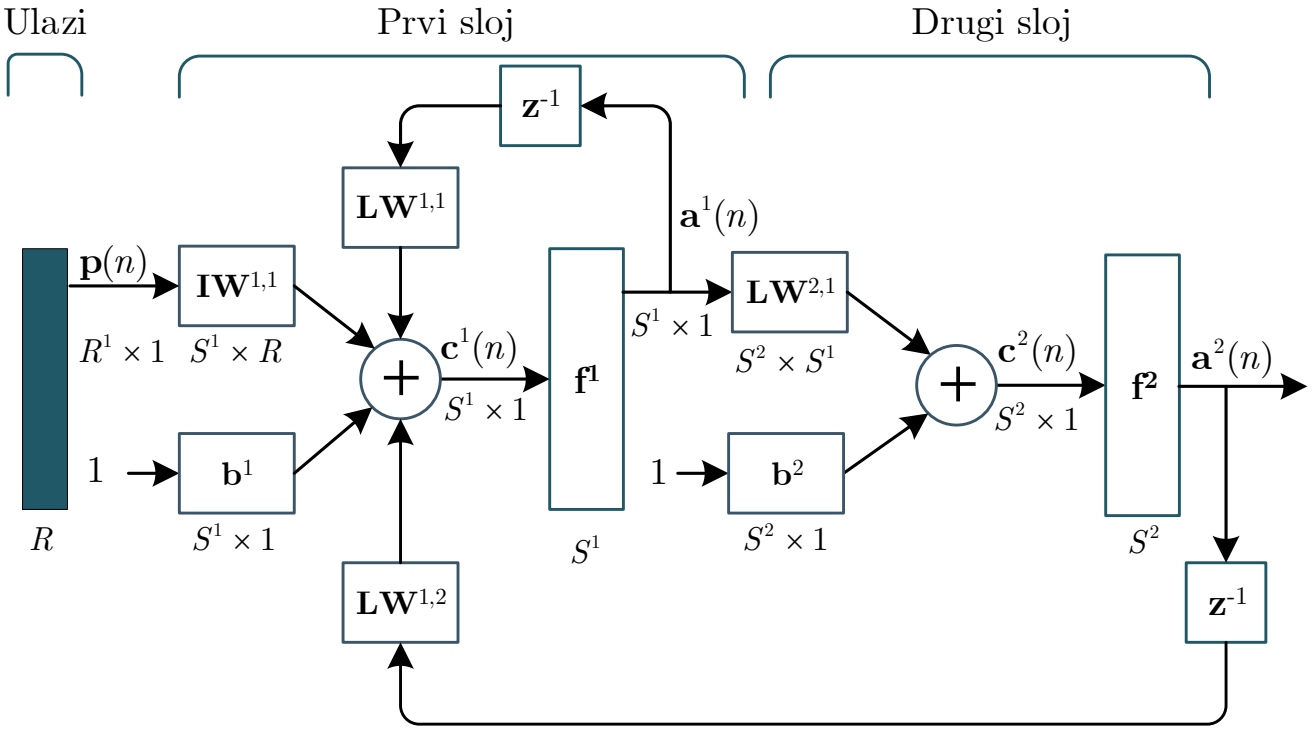
$$\mathbf{a}^3(n) = \mathbf{f}^3\left(\mathbf{LW}^{32}\mathbf{a}^2(n)\right).$$

Postojanje povratne sprege ima veliki uticaj na sposobnost učenja neuralne mreže, kao i na njene performanse. S obizirom da RNN sadrže memoriju, pored aproksimacije funkcija one se mogu koristiti i za aproksimaciju dinamičkih sistema. Zbog ovog svojstva RNN nalaze primjenu u identifikaciji dinamičkih sistema, upravljanju sistemima, detekciji otkaza, obradi prirodnih jezika, prepoznavanju govora, ekvivalizaciji kanala u telekomunikacionim sistemima, itd.

Primjer – rekurentna neuralna mreža

a) U Matlab-u kreirati neuralnu mrežu čija je struktura prikazana na slici ispod. NN ima jedan ulaz i jedan izlaz. Prvi (skriveni) sloj mreže sadrži 2 neurona, dok drugi (izlazni) sloj sadrži 1 neuron. Prva aktivaciona funkcija je *tansin*, dok je druga linearna. Ako su svi koeficijenti i bajasi mreže jednaki 1, nacrtati njen odziv na jedinični dirakov impuls.

b) Napisati izraz za izlaz mreže, a zatim na osnovu njega odrediti odziv mreže na jedinični Dirakov impuls. Rješenje uporediti sa rezultatom iz a).



Primjer – rekurentna neuralna mreža

Za generisanje zadate NN koristićemo komandu *network*, čija je sintaksa:

```
net = network(NI,NL,BC,IC,LC,OC)
NI - broj ulaza; NL - broj slojeva
BC - niz kojim definišemo koji slojevi imaju bajas
IC - niz kojim definišemo na koje sve slojeve je povezan ulaz
LC - matrica kojom definišemo povezanost slojeva
OC - niz kojim definišemo na koje slojeve je povezan izlaz
```

Konkretno, zadatu NN ćemo definisati na sljedeći način:

```
>> net1=network(1,2,[1;1],[1;0],[1 1;1 0],[0 1]);
```

Komanda *network* kreira neuralnu mrežu bez kašnjenja. Kašnjenja se naknadno mogu definisati na sljedeći način:

```
>> net1.layerWeights{1,1}.delays=1; % obratiti paznju na sliku
>> net1.layerWeights{1,2}.delays=1; % i indekse koeficijenata kod
>> net1.layerWeights{2,2}.delays=1; % kojih postoji kašnjenje
```

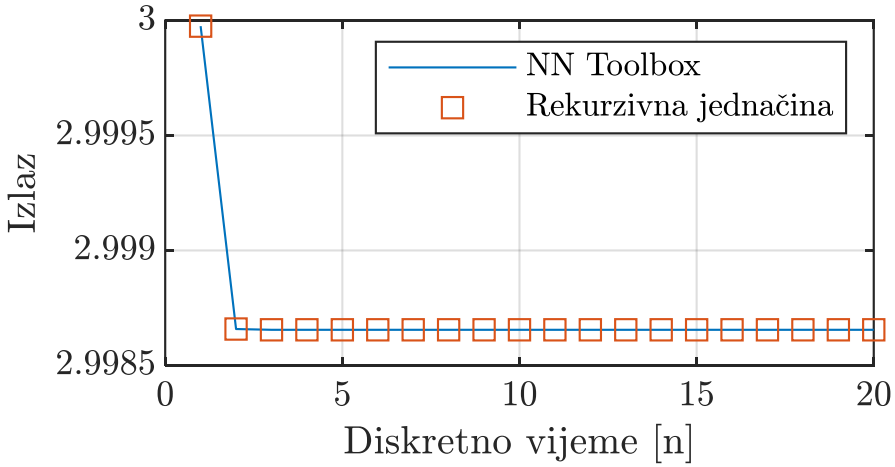
Izlaz iz RNN se računa na sljedeći način:

```
>> t=net1(con2seq(p));
```

Ulazni podaci **p** se prvo moraju pretvoriti u niz ćelija, kako bi mreža razumjela da se da ulazi pristižu u različitim vremenskim trenucima.

Primjer – rekurentna neuralna mreža

Na slikama je prikazana strukura rekurentne neuralne mreže, kao i njen izlaz za zadati ulaz. Izlaz je računat na dva načina: pomoću ugrađenih funckija i pomoću sljedeće diferencne jednačine:



$$\mathbf{a}^2(n) = \mathbf{LW}^{21}\mathbf{a}^1(n) + \mathbf{b}^2$$

$$\mathbf{a}^1(n) = \text{tansig}\left(\mathbf{IW}^{11}\mathbf{p}(n) + \mathbf{LW}^{11}\mathbf{a}^1(n-1) + \mathbf{LW}^{12}\mathbf{a}^2(n-1) + \mathbf{b}^1\right).$$

Kompletan Matlab kod je dostupan na: http://www.tsau.ac.me/ISAU/P1/pr_recnet.m.