SPIE09 Digitalna obrada zvučnih signala

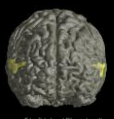
05. Uvod u obradu zvučnih signala umjetnim neuralnim mrežama

Ozren Bilan, viši predavač

Uvod u obradu zvučnih signala umjetnim neuralnim mrežama

Ljudski mozak je sofisticirano računalo velike složenosti s frekvencijom takta 1kHz.

Hitoshi Kiyonobu



u pdf inačici nisu vidljive animacije

Uvod u umjetne neuralne mreže

Umjetne neuralne mreže predstavljaju **paralelno raspodijeljenu mrežu** koja se sastoji od procesnih jedinica koje nazivamo **neuroni**. Motivirana je humanim kognitivnim procesima: ljudski mozak je vrlo **kompleksno nelinearno paralelno računalo** s frekvencijom takta od 1 kHz.

Mreža se sastoji od **nizova vanjskih ulaza i izlaza** koji primaju informacije iz okoline i nakon obrade prosljeđuju obrađene informacije okolini.

Neuroni su povezani **sinapsama**, kojima su pridruženi **sinaptički koeficijenti**.

Koeficijenti pohranjuju znanje koje se pribavlja iz okoline.

Učenje se postiže **prilagođavanjem koeficijenata** u skladu prema **algoritmu učenja**. Neuroni u ljudskom mozgu odumiru pri čemu se razvijaju nove sinapse.

Zbog toga **neuroni mogu evoluirati modifikacijom vlastite topologije**.

Jedan od primarnih ciljeva umjetne neuralne mreže je generalizacija tj. **uopćavanje stečenog znanja na slične ali i nove nepoznate ulazne obrasce** – to je **umjetna inteligencija**.

Oštećenje bilo kojeg dijela umjetne neuralne **mreže** najčešće **ima vrlo mali utjecaj na sposobnosti proračuna** cjeline. To znači da

uspješno nadilaze štetne utjecaje na ulazne signale (npr. šum).

Prednosti bioloških neuralnih sustava su:

- **relativna brzina** kojom izvode proračune
- **robusnost** prema štetnim utjecajima okoline i unutarnjim degradacijama signala

Ljudski mozak

Neurološki sustav čovjeka možemo sagledati kao **sustav od tri stupnja**:

Centralni dio sustava je **mozak** prikazan kao **neuralna mreža**.

Strelice usmjerene s lijeva na desno ukazuju na **prijenos informacija unaprijed kroz sustav**

Strelice usmjerene s desna na lijevo ukazuju na **povratnu vezu sustava**.

Receptori vrše pretvorbu **stimulusa vanjske okoline** u **električne impulse** koji mrežom **prenose informacije**

Efektori pretvaraju **električne impulse** neuralne mreže na **odziv sustava prema okolini**.

Automatizirana obrada podataka pogodna za izvršavanje na računalu

Neautomatizirane obrade podataka izvršavaju živčani sustavi

Automatizirana obrada podataka pogodna za izvršavanje na računalu

Program za obradu teksta: Osnovna zadaća mu je pohrana informacija, organiziranje informacija (odreži i zalijepi, provjera pravopisa, izgled stranica, te dobava informacija (kao što su pohrana dokumenta u memoriju ili ispis na pisaču). **Zadaće se izvršavaju pomakom podataka s jednog mjesta na drugo i ispitivanjem nejednakosti (*A=B, A<B*, itd.)**.

Zamislimo sortiranje liste riječi abecednim redom: Svako slovo predstavlja 8 bitna znamenka, ASCII vrijednost prvog slova riječi. Poredak abecednim redom uključuje raspoređivanje riječi po kojemu ASCII vrijednosti jednoliko rastu od početka do kraja liste. To se radi ponavljanjem dva koraka sve dok se ne završi sortiranje po abecedi.

1) Prvo se ispituju dva susjedna unosa da li su u abecednom redoslijedu (*IF A>B THEN* ...).
2) Drugi korak, ako nisu, zamijeni ih tako da postanu (*A↔B*).
3) Kada se ova dva koraka ponove dovoljno puta na svim zadanim susjednim riječima, lista će postati poredana po abecedi.

	MANIPULACIJA PODACIMA	MATEMATIČKI PRORAČUN
Tipična primjena	Obrada teksta, Baze podataka, Tablični kalkulatori Operacijski sustavi	DSP, dinamičko upravljanje, znanstvene simulacije
Glavna operacija	pomak podataka (*A→B*) ispitivanje (*Ako je A=B, onda je...*)	zbrajanje (*A + B = C*) množenje (*A x B = C*)

Automatizirana obrada podataka pogodna je za izvršavanje na računalu

Neautomatizirane obrade podataka izvršavaju živčani sustavi

npr. procesiranje prirodnoga jezika

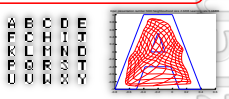
ili rješavanje problema prepoznavanje lica

- čovjek obavlja u 100-200 ms (100ms potrebno za raspoznavanje lica)
- računalu treba više vremena, a točnost je upitna

Tražimo koncept obrade podataka sličan funkcioniranju mozga
je li moguće *kopirati* rad mozga?

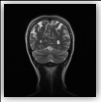
Biološke neuronske mreže

- biološki organizmi, živčani sustavi



Umjetne neuronske mreže

- primitivne imitacije bioloških neuronskih mreža
- pokušavaju približiti računala mogućnostima mozga imitacijom njegovih procesnih elemenata na jako pojednostavljen načina



Mozak protiv računala

Ljudski mozak i *von Neumannova računala* **različito obrađuju podatke**

- neuroni su **pet-šest redova veličine sporiji** od digitalne logike *(ms i ns)*
- mozak **nadoknađuje brzinu ogromnim brojem neurona**
  (mozak ima oko 10 milijardi neurona i oko 60 000 milijardi međuspojeva)
- mozak je **energetski enormno efikasan**
  (10^{-16} J po operaciji u sekundi prema 10^{-6} J po operaciji u sekundi)

	mozak 	računalo 
elementi	**10^{11} neurona**	780M tranzistora (*Core i7*)
broj veza	**10^{14} sinapsi** (10^3 po neuronu)	≤ 32
energetska potrošnja	**10^{-16} J po operaciji**	10^{-6} J po operaciji
brzina prijenosa	ms ciklus	**ns ciklus**
način rada	**serijski i paralelno**	uglavnom serijski
tolerancija na pogreške	**da**	ne
signali	analogni	digitalni
sposoban učiti	**da**	malo
svjestan/inteligentan	**kako tko**	još ne

Nakon rođenja mozak se izgrađuje pomoću ***iskustva*** koje se gradi godinama:

- najbrži razvoj dešava se tijekom prve dvije godine života kada se formira ***10^6 sinapsi u sekundi***
- razvoj mozga nastavlja se i nakon početne faze
- **Internet i društvene mreže koče razvoj mozga**

Podjela neuronskih mreža

Biološke ili prirodne neuronske mreže

- biološki organizmi
- mozak ljudi i životinja
- visoka složenost i paralelizam

Umjetne neuronske mreže

- motivirane biološkim neuronskim mrežama
- za sada su primitivne imitacije bioloških mreža
- implementacija na računalima ili pomoću specijaliziranih sklopova (analognih, digitalnih, hibridnih)

Definicija umjetne neuronske mreže

Alexander i Morton (1990): umjetna neuronska mreža je:

masivni paralelni distribuirani procesor koji dobro pamti iskustveno znanje

Slična je mozgu u dva aspekta jer:

- **Znanje stiče procesom učenja**
- **Koriste međusobne veze između neurona za spremanje znanja**

Biološko neuronsko stablo

Neuronsko stablo je funkcionalna i strukturalna jedinica nervnog sustava. On može primiti i proslijediti impulsnu informaciju. U tu svrhu postoje izdanci, *dendriti* i *axoni, a* informaciju između njih prenose *sinapse*. Tijelo neurona je **soma**

Neuroni su kod sisavaca uvijek u snopovima, te ih se naziva **živčanim snopovima.**

Dendriti primaju signale drugih neurona

Akson prenosi impulse do sinaptičkih terminala

- Oni zatim prenose signale na dendrite drugih neurona

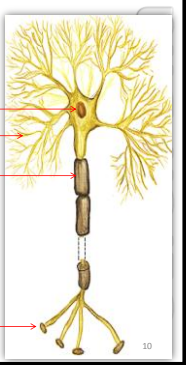
Sinapse

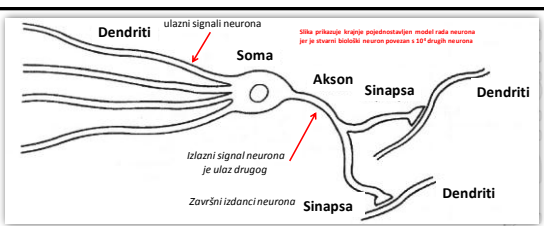
- omogućuju interakciju između neurona
- *Presinaptički* elektro kemijski proces oslobađa *neurotransmiter* koji difundira kroz sinaptičku pukotinu. Nakon difuzije izaziva se elektro kemijski *postsinaptički* proces
- Sinapsu možemo prikazati kao nereciprocni četveropol

Piramidalni neuron

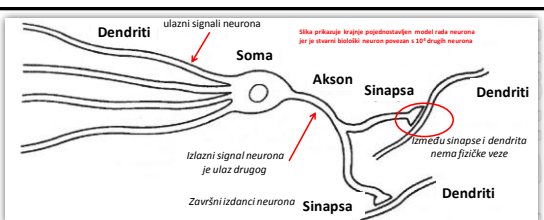
- može imati 10000 ili više ulaznih sinapsi
- njegov izlaz može se prenositi na desetke tisuća drugih neurona

Prikažimo neuron jednostavnije…





- **Biološki neuron** se sastoji od **tijela** (***soma***)
- Prima signale od ostalih neurona preko izdanaka koji se zovu ***dendriti***
- Signali koje neuron prima mogu biti **pozitivni** (*afirmativni*) ili **negativni** (*negacijski*)
- **Signali** koji stižu u neuron **se zbrajaju**
- Ako je rezultat veći od **granične vrijednosti** (*PRAG*) neuron **generira signal** (*OKIDA*)



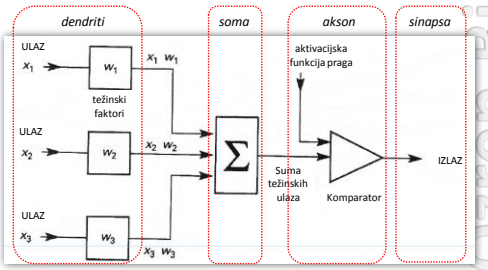
- **Izlazni signal** s neurona javlja se na dijelu koji se zove ***akson***
- **Aksoni završavaju u izdancima koji se zovu *sinapse***
- **Sinapse** povezuju izlazni signal jednog neurona s *dendritima* drugog neurona (gdje taj izlazni signal postaje ulazni signal drugog neurona)
- **Nema fizičke veze između *sinapse* jednog neurona i *dendrita* drugog** neurona, već se veza ostvaruje pomoću izmjene kemijskog spoja ***neurotransmitera*** (neuroprijenosnika). *Presinaptički* proces oslobađa tvar koja difundira kroz sinaptičku pukotinu i izaziva *postsinaptički* proces.

Sinapsu možemo zamisliti kao nereciprocni četveropol

Vrlo pojednostavljen model rada neurona jer je biološki neuron povezan s 10000 drugih

Koeficijenti predstavljaju ***ZNANJE***

- **Ulazni signali** *x1, x2, x3* **množe se s težinskim koeficijentima** *w1, w2, w3* **i zbrajaju** se
- **Ako je zbroj veći od granične vrijednosti**, neuron **generira izlazni signal** prema drugome



Matematički model neurona

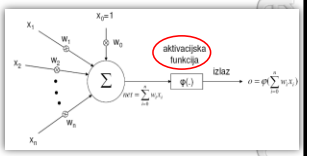
Elementi modela neurona:

Skup sinapsi tj. ulaza od kojih svaki ima svoju težinu što pišemo

signal *xj* na ulazu *j* neurona *k* ima težinu *wkj*

Sumator za zbrajanje otežanih ulaza. Ove operacije računaju linearnu kombinaciju ulaza.

Nelinearna aktivacijska funkcija koja ograničava izlaz neurona na interval [0,1]



Umjetni *ekvivalent*

McCulloch-Pitts model neurona (1943.)
Threshold Logic Unit (TLU)

Analogija:

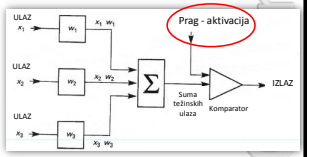
signali su numeričke vrijednosti (x_n),

jakost sinapse opisuje težinski faktor (w_n),

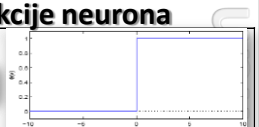
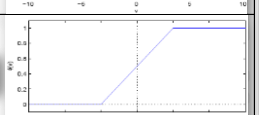
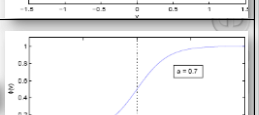
tijelo neurona je zbrajalo (*S*), a

akson aktivacijska funkcija (*j*)

Interesira nas **AKTIVACIJSKA FUNKCIJA…**

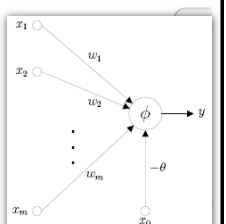


Aktivacijske funkcije neurona

Funkcija praga ili Heaviside funkcija Neuron s ovakvim tipom aktivacije naziva se *McCulloch-Pitts* model. Svojstvo modela je ***sve ili ništa.***	$\phi(v) = \begin{cases} 1 & \text{ako} \quad v \geq 0 \\ 0 & \text{ako} \quad v < 0 \end{cases}$	
Funkcija linearna po odsječcima Možemo je shvatiti kao aproksimaciju nelinearnog pojačala . Definicija podrazumijeva jedinični faktor pojačanja unutar linearnog dijela	$\phi(v) = \begin{cases} 1 & \text{ako} \quad v \geq \frac{1}{2} \\ v & \text{ako} \quad \frac{1}{2} < v < \frac{1}{2} \\ 0 & \text{ako} \quad v \leq -\frac{1}{2} \end{cases}$	
Sigmoidna funkcija najčešći oblik aktivacijske funkcije neuralne mreže. Primjer sigmoida je funkcija, određena izrazom $\Phi(v)$, gdje je $a > 0$ parametar strmine. U graničnom slučaju kad $a \rightarrow \infty$, sigmoidna funkcija postaje **funkcija praga**. Sigmoid je diferencijabilan što je vrlo važno svojstvo pri učenju neuralne mreže.	$\phi(v) = \frac{1}{1 + \exp(-av)}$ Matlab koristi ***tansig***	

Perceptron – neuronska mreža od jednog neurona

- Najjednostavnija neuronske mreže je **perceptron.**
- Sastoji se od **samo jednog neurona**.
- Perceptron se formira nelinearnim neuronom, modelom neurona *McCulloch-Pitts*.
- Taj umjetni neuron za aktivacijsku funkciju ima **funkciju praga za** izlaz perceptrona (*1957, Rosenblatt)*



- Aktivacijska funkcija ϕ perceptrona je *funkcija praga* - **Heaviside step funkcija**
- Izlaz perceptrona je :

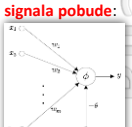
$$y = \phi(\tilde{w} \cdot \tilde{x}) \qquad \begin{cases} \tilde{w} = (-\theta, w_1, \ldots, w_m) \\ \tilde{x} = (1, x_1, \ldots, x_m) \end{cases}$$

$$o = \begin{cases} 1 & \sum_{i=0}^{n} w_i x_i \geq 0 \\ 0 & \sum_{i=0}^{n} w_i x_i < 0 \end{cases} \qquad \sum_{i=0}^{n} w_i x_i = 0$$

Cilj perceptrona

Perceptrona mora **točno klasificirati skup vanjskih signala pobude**:

x1, x2, x3, . . . , xm



u jednu od dvije klase *C1* ili *C2*. (istina ili laž; 0 ili 1)

Točka x_i se pridružuje klasi ***C1*** ako je izlaz perceptrona *y* jednak 1, a
Točka x_i se pridružuje klasi ***C2*** ako je izlaz perceptrona *y* jednak 0.

Perceptron ima **ogromnu logičku snagu** jer može izračunati skoro sve binarne Boolove logičke funkcije, dok izlazne klase *C1* i *C2* predstavljaju istinu ili laž.

Poznavanjem binarnih Boolovih logičkih funkcija proizlazi da **perceptron sa samo 2 ulaza može izračunati 14 od 16 mogućih funkcija.**

Nadzirano učenje perceptrona

Ako je po volji zadan vektor početne sinaptičke težine ***w***, perceptron možemo naučiti da izračuna specifičnu funkciju cilja ***t***. Težine se prilagođavaju u skladu s algoritmom učenja prema klasificiranim primjerima učenja pri čemu stanje mreže konvergira prema točnoj vrijednosti. To znači:

perceptron uči iz vlastitog iskustva.

Za primjer odredimo da je funkciju cilja ***t*** jednaka *(x, t(x))*, gdje je ***x*** ulazni vektor. Perceptronu se onda zadaje obrazac za učenje ***s***, koji **predstavlja sekvencu primjera koji tvore iskustvo**, npr:

s = (x1, t(x1)), (x2, t(x2)), . . . , (xm, t(xm))

Težina vektora ***w*** mijenja se u skladu s algoritmom učenja, nakon što se izvrše svi primjeri.

Algoritam učenja Perceptrona

Za svaku konstantu učenja ***v > 0***, težinski vektor ***w*** se ažurira pri svakom stupnju učenja ne slijedeći način:

w' = w + v(t(x) – hw(x))x

Gdje je:

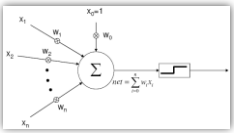
- ***hw*** izlaz kojeg računa perceptron primjenom tekućih težina vektora ***w***.
- ***t(x)*** funkcija cilja, očekivani izlaz perceptrona.

Učenje perceptrona

- Primjeri za učenje: (x, t) – x je ulazni vektor, t je željeni izlaz
- Perceptron za određeni primjer daje izlaz (o), a mi znamo što želimo dobiti
- na izlazu (t) – razlika nam govori o potrebi korigiranja težina
  - ukoliko nema razlike, sve je u redu
  - ukoliko ima, moramo raditi korekciju

H – stope učenja

$$w_i \leftarrow w_i + \Delta w_i \qquad \Delta w_i = \eta(t-o)x_i$$



Učenje perceptrona - algoritam

PERCEPTRON (skup za učenje, η)

- Postavi težine na slučajno odabrane vrijednosti
- dok nisu svi uzorci ispravno klasificirani
- za svaki uzorak iz skupa učenja
- klasificiraj uzorak perceptronom
- ako je uzorak ispravno klasificiran nastavi sa sljedećim uzorkom
- inače primjeni korekciju

Algoritam konvergira u konačnom broju koraka ako su primjeri za učenje linearno separabilni i ako je *η* dovoljno malen

(*Minsky i Papert, 1969.*)

Gradijentni spust i delta pravilo

• Prethodni algoritam ne konvergira za linearno neseparabilne primjere
• Zato uvodimo **gradijentni spust**
– pretraga prostora hipoteza (težina) za pronalazak težina koje najbolje odgovaraju podacima za učenje
• **U ovom slučaju koristimo perceptron BEZ aktivacijske funkcije**
To je linearna jedinica (tzv. *Adaline*)

• Pogreška učenja hipoteza (težina), gdje je D skup za učenje:

• Gradijentni spust minimizira E iterativnim modificiranjem težina u malim koracima.
• U svakom koraku, vektor težina se mijenja u smjeru najvećeg spusta niz plohu pogreške.
• Proces se nastavlja sve dok se ne dostigne globalni minimum pogreške

GRADIJENTNI-SPUST– algoritam (skup za učenje, η)
Postavi težine na nasumično odabrane vrijednosti
dok nije zadovoljen kriterij zaustavljanja
za svaki uzorak iz skupa učenja
Klasificiraj uzorak perceptronom
za svaku težinu wi izračunaj
i za svaku težinu wi izračunaj
• Gradijentni spust (delta pravilo) asimptotski konvergira prema minimumu
pogreške (bez obzira da li su primjeri za učenje linearno separabilni ili ne)

Stohastički gradijentni spust
• Problemi gradijentnog spusta
– spora konvergencija u minimum
– ne garantira pronalazak globalnog minimuma u slučaju više lokalnih minimuma
• Stohastički gradijentni spust
– gradijentni spust korigira težine nakon izračuna nad svim primjerima
– stohastički gradijentni spust aproksimira gradijentni spust inkrementalnom korekcijom težina
• Razlike
– gradijentni spust je sporiji jer zahtjeva više računanja (sve težine odjednom), međutim kako koristi pravi gradijent, radi i veće korake
– stohastički gradijentni spust može ponekad izbjeći lokalne minimume jer koristi više manjih gradijenata pogreške umjesto globalnog

STOHASTIČKI-GRADIJENTNI-SPUST algoritam (skup za učenje, η)
Postavi težine na nasumično odabrane vrijednosti
dok nije zadovoljen kriterij zaustavljanja
za svaki uzorak iz skupa učenja
Klasificiraj uzorak perceptronom
za svaku težinu wi izračunaj
Delta pravilo još je poznato kao i LMS (*least-mean-square*), *Adaline* pravilo ili *Widrow-Hoff* pravilo

Backpropagation – algoritam

BACKPROPAGATION (skup za učenje, η)

- Inicijaliziraj težinske faktore na male slučajne vrijednosti
- dok nije ispunjen uvjet zaustavljanja
- za svaki *(x, t)* iz skupa za učenje
- Izračunaj izlaz za svaku jedinicu u mreže
- za svaku izlaznu jedinicu k izračunaj pogrešku $\delta_k \leftarrow o_k(1-o_k)(t_k-o_k)$
- za svaku skrivenu jedinicu h izračunaj pogrešku $\delta_h \leftarrow o_h(1-o_h)\sum_{\text{sve Downstream(h)}} w_{sh}\delta_s$
- Ugodi svaki težinski faktor w_{ji}

$w_{ji} \leftarrow w_{ji} + \Delta w_{ji}$ gdje je $\Delta w_{ji} = \eta \delta_j x_{ji}$

Reprezentacijska moć i zaustavljanje

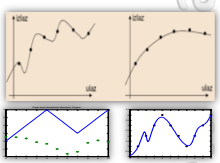
Kakve sve funkcije mogu neuronske mreže reprezentirati? odgovor ovisi o topologiji mreže:
• **bool funkcije**
• **kontinuirane funkcije**
• **proizvoljne funkcije**
• **Neuronske mreže mogu aproksimirati proizvoljnu funkciju proizvoljnom preciznošću**
(*Kolmogorovljev egzistencijalni teorem*)

Reprezentacija skrivenog sloja
što predstavlja skriveni sloj?
• Backpropagation korigira težine tako da definira neurone skrivenog sloja što efektivnije u svrhu minimizacije kvadratne pogreške E
• To ga vodi prema definiciji skrivenog sloja koja nije eksplicitno zadana ulazom, ali je izgrađena tako da na najbolji način predstavi značajke primjera za učenje koji su najvažniji za učenje ciljne funkcije
• Automatsko otkrivanje korisne reprezentacije skrivenog sloja je ključna sposobnost višeslojnih mreža

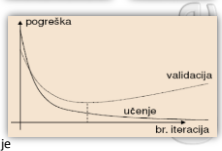
Kriteriji zaustavljanja
• **Fiksni broj iteracija petlje**
– što možemo očekivati od takve mreže ako o grešci ne znamo ništa unaprijed?
• **Unaprijed određena pogreška skupa za učenje**
– zaustavljamo učenje kada pogreška padne ispod praga
• **Unaprijed određena pogreška skupa za testiranje**
– također zaustavljamo učenje kada pogreška padne ispod zadanog praga
Kriterij zaustavljanja je važan jer
- premalo iteracija može neznatno smanjiti pogrešku
- time dobivamo *lošiju* mrežu nego bismo mogli
- previše može odvesti u prenaučenost ili *overfitting*

Generalizacija i prenaučenost

• Svojstvo dobre klasifikacije za nepoznate ulaze nazivamo generalizacija
• Cilj generalizacije: interpolirati raspoložive podatke najjednostavnijom krivuljom
• Prenaučenost ili *overfitting*
– prevelik broj primjera za učenje uzrokuje gubitak svojstva generalizacije, mreža postaje stručnjak za podatke iz skupa za učenje
– forsiranje **što manje pogreške** na skupu za učenje – mreža postaje ekspert za naučeno, ali loše generalizira



Kako se riješiti **prenaučenosti** ?
– postupkom **smanjivanje težina** *weight decay*
• smanjivanje težina u svakoj iteraciji za mali faktor **sprječava kompleksne aproksimacije**
– uvođenjem **parametra validacije**
• daje **mjeru pogreške** za vrijeme učenja
• **počinje rasti** kada mreža uči posebnosti skupa za učenje



Neuronske mreže

- Definirali smo umjetni neuron i perceptron sa step funkcijom
- definirali smo linearnu jedinicu (*Adaline*) bez aktivacijske funkcije
- Sve do sada smo radili s jednim procesnim elementom
- Postoji li mogućnost ***povezivanja više procesnih elemenata*** **?**
- ***Koji procesni element koristiti*** kao gradbeni element takve mreže?

– linearnu jedinicu?

- linearna kombinacija linearnih funkcija je opet linearna funkcija

– perceptron?

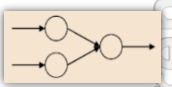
- aktivacijska funkcija perceptrona (*step funkcija*) je nediferencijabilna pa ne možemo koristiti gradijentni spust
- Treba nam element s ***nelinearnom diferencijabilnom*** **aktivacijskom funkcijom** – to je **sigmoid** (Matlab koristi ***tansig***)

Topologija neuronskih mreža

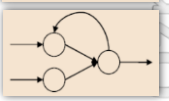
- imamo *novi tip* umjetnog neurona
- kako ćemo ga povezati s ostalim neuronima?
- Izlaz jednog neurona predstavlja ili može predstavljati ulaz sljedećem
- Podjela neuronskih mreža po topologiji (arhitekturi mreže)

- Osnovna podjela

acikličke – ne sadrže povratne veze



cikličke – sadrže povratne veze (*tema za sebe*)



Grafovi i arhitektura neuralne mreže

Arhitektura ili topologija mreže **određuje način na koji su neuroni međusobno povezani.** Neuralne mreže postoje u četiri temeljno različite klase arhitekture mreže:

1. **Jednoslojne** mreže *bez povratnih veza* (*single-layer feedforward networks*)
2. **Višeslojne** mreže *bez povratnih veza* (*multi-layer feedforward networks*)
3. Mreže **s povratnim vezama** (*recurrent networks*)
4. **Ljestvičaste** mreže (*lattice structures*)

Grafovi

Neuronske mreže mogu se prikazati pomoću orijentiranih grafova slično grafu toka signala. U grafu imamo dvije vrste grana:

1. **Sinaptička grana** koja označava **linearnu** ulazno-izlaznu **relaciju** ***množenja s težinom***
2. **Aktivacijska grana** koja predstavlja **nelinearnu** ***ulazno izlaznu karakteristiku aktivacijske funkcije***

Topologija neuronskih mreža

Po broju slojeva

– **jednoslojne**

(*ulazni sloj se ne računa jer nije procesni sloj*)

– **višeslojne**

(*sadrže jedan ili više skrivenih slojeva*)

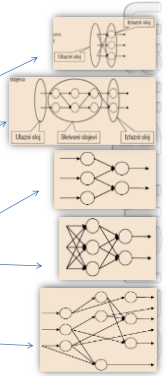
Po povezanosti

– **djelomično povezane**

– **potpuno povezane**

(*svaki neuron prethodnog sloja povezan je sa svakim neuronom sljedećeg sloja*)

Proizvoljna aciklička mreža



Primjene

- **Klasifikacija uzoraka**

– optičko prepoznavanje znakova

– **raspoznavanje govora/govornika**

– detekcija QRS kompleksa

- Klasteriranje (*clustering*)

– **Kompresija podataka**

- **Aproksimacija funkcija**
- Predviđanja

– **Adaptivna linearna predikcija**

– kretanje cijena dionica

– vremenske prognoze

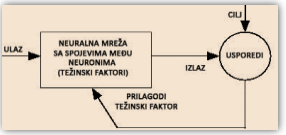
- **Optimizacije**
- Asocijativne memorije
- Upravljanje

Neuralna mreža u Matlabu

Neuralne mreže sastavljene su od **paralelnog spoja vrlo jednostavnih elemenata**. Ti elementi inspirirani su biološkim nervnim sustavom. Kao i u prirodi, **način spajanja elemenata određuje funkciju mreže**. Neuralna mreža **može se naučiti izvođenju posebnih funkcija jednostavnom prilagodbom vrijednosti težine** među spojevima elemenata.

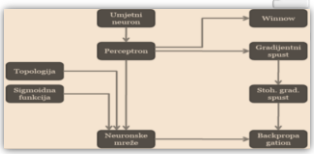
Neuralna mreža se **uči tako da specifični ulaz rezultira izlaznim ciljem**. Slika ilustrira takvu situaciju. **Mreža se podešava temeljem usporedbe izlaza i cilja, sve dok izlaz mreže ne postane jednak cilju.**

U tipičnim situacijama za učenje mreže potreban je veliki broj parova ulaz/cilj. Neuralne mreže mogu se naučiti da izvode vrlo složene funkcije sa različitim primjenama što uključuje prepoznavanje uzorka i oblika, identifikaciju, klasifikaciju, **obradu zvučnih i video signala** te sustave regulacije.



Neuralne mreže primjenjujemo u tri koraka: *projektiranje, učenje i ispitivanje*

Projektiranje neuralne mreže:



– Određujemo **broj** čvorova u svakom od tri sloja (ulaznom, skrivenom, izlaznom).
– Određujemo što je **_(x)** za svaki čvor (uobičajeno svi čvorovi mreže sloja koriste isti **_(x)**). Nazivamo ih prijenosne funkcije
– Moramo specificirati koju **optimizacijsku rutinu** ili **učenje koristimo**. Najčešće se koristi ***traingdx***, gradijentni spust ili ***trainlm*** (*Levenburg-Marquardt*, kombinacija gradijentnog spusta i *Newtonovog postupka*).
– Pri projektiranju su opcije: **funkcija greške** (tvornički je podešen srednji kvadrat greške *MSE*), dijagram napretka učenja, itd.

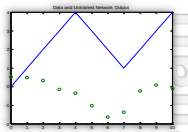
Učenje:
nakon projektiranja mrežu učimo tako da optimiziramo funkciju greške. Postupak određuje najbolji skup koeficijenata za naš skup ulaznih podataka.

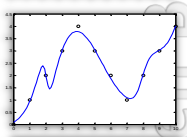
Ispitivanje:
Mrežu ispitujemo kako bi vidjeli da li je postignuta dobra ravnoteža **memorizacije** (točnosti) i **uopćavanja** (generalizacije).

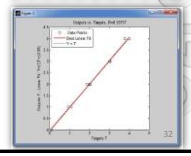
Primjer učenja mreže

```
P = [0 1 2 3 4 5 6 7 8 9 10]; %uzorak za vježbanje - učenje (vrijednosti domene)
T = [0 1 2 3 4 3 2 1 2 3 4]; %cilj vježbanja - učenja  (vrijednosti područja)
net = newff([0 10],[5 1],{'tansig' 'purelin'});
%Grafički prikazujemo polazne-izvorne točke i neuvježbani-nenaučeni izlaz
Y = sim(net,P);
figure(1)
plot(P,T,P,Y,'o')
title('Podaci i neuvježbani izlaz mreže')
%Uvježbavamo mrežu i grafički prikazujemo rezultat
net.trainParam.goal=0.01; %0 je podešena vrijednost – premala je!
net.trainParam.epochs = 50;
%broj epoha
%za naš primjer ne treba duga vježba (ne treba mnogo učiti)
net = train(net,P,T);
X = linspace(0,10); %nove točke domene
Y = sim(net,X); %izlaz mreže
figure(2)
plot(P,T,'ko',X,Y)
%alternativni drugačiji način testiranja naučenog funkcijom postreg
figure(3)
Tout=sim(net,P);  %dobivamo izlaz mreže iz područja uvježbavanja
[m,b,r]=postreg(T,Tout); %Izvodi se linearna regresija
```

```
Warning: NEWFF used in an obsolete way.
> In obs_use at 18
  In newff>create_network  at 127
  In newff at 102
    See  help for NEWFF to update calls to the new argument list.
```







Komentar vježbe

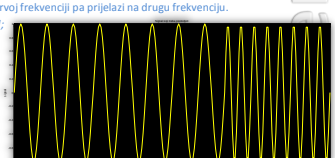
- Vrijednosti domene uzoraka za učenju su između minimuma 0 i maksimuma 10. To su znamenke koje se pojavljuju u prvom argumentu ***newff***. Veličina domene i skup područja zadani su kao ***m × n***, gdje je *m*=veličina, a *n*=broj točaka podataka.
- Potrebni su argumenti prikazani u ***newff***. Pri tome prvi argument određuje minimalnu i maksimalnu vrijednost skupa domene. Moguće je koristiti funkciju ***minmax(P)*** umjesto upisivanja stvarnih vrijednosti. Tako se implicitno određuje broj čvorova ulaznog sloja što određuje dimenziju domene. Drugi argument u ***newff***, [5,1] definira broj čvorova skrivenog sloja i izlaznog sloja. Taj vektor implicitno definira koliko slojeva postoji jer je jednak veličini vektora. Dakle, možemo imati slojeva koliko god želimo.
- Posljednji argument određuje tip prijenosne funkcije , ***_(x)***, koja se koristi.
- Funkcija **tansig** je **inverzna tangens** funkcija koja ima iste karakteristike kao standardna **sigmoidalna** funkcija.
- U izlazu uvijek koristimo linearni sloj pa je ovaj argument uvijek ***purelin***.
- Podešen postupak učenja (**optimizacije**) je postupak ***Levenburg-Marquardt***. Predstavlja kombinaciju *gradijentnog spusta* i *Newtonovog postupka*).
- U idućim vježbama koristit ćemo različite postupke.

Adaptivna linearna predikcija zvučnog signala neuralnom mrežom

Ilustrirat ćemo kako naučiti adaptivni linearni sloj da predvidi slijedeću vrijednost zvučnog signala ako je zadana tekuća i posljednje četiri vrijednosti

Dva segmenta signala određena su od 0 do 6 sekundi u koracima od 1/40 sekunde.

```
% generiramo vremenski promjenjivi zvučni signal
time1 = 0:0.025:4;     % od 0 do 4 s
time2 = 4.025:0.025:6;  % od 4 do 6 s
time = [time1 time2];  % od 0 to 6 s
%Signal se mijenja tako da započinje na prvoj frekvenciji pa prijelazi na drugu frekvenciju.
signal = [sin(time1*4*pi) sin(time2*8*pi)];
plot(time,signal)
xlabel('vrijeme');
ylabel('signal');
title('Signal koji treba predvidjeti');
```



Prilagodba problema neuralnoj mreži

Signal se konvertira u ***cell array***. Neuralna mreža predstavlja vremensku oznaku u obliku stupaca *cell array*. Treba ih razlikovati od različitih vremenskih uzoraka, koji su predstavljeni stupcima matrica.

signal = ***con2seq****(signal);* *(upiši u Matlabu >>help* ***con2seq****)*

Kako bi postavili problem koristi ćemo prvih pet vrijednosti signala kao početna ulazna stanja kašnjenja. Preostale signale (ostatak) ćemo koristiti kao ulaz.

```
Xi = signal(1:5);
X = signal(6:end);
timex = time(6:end);
```

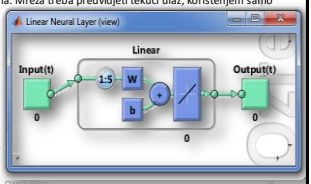
Ciljeve smo odredili tako da su prilagođeni ulazima. Mreža treba predvidjeti tekući ulaz, korištenjem samo posljednjih pet vrijednosti.

```
T = signal(6:end);
```

Kreiranje *Linear Layer*
Funkcija ***linearlayer*** kreira linearni sloj sa jednim Neuronom s *tap delay* posljednjih pet ulaza.

```
net = linearlayer(1:5,0.1);
view(net)
```

Naredba view(net) prikazat će linearnu neuralnu mrežu



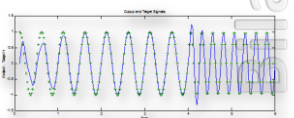
Prilagodba linearnog sloja i signal pogreške

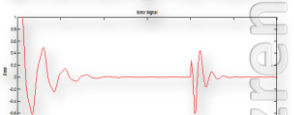
Funkcija ***adapt*** simulira mrežu na ulazu i prilagođava težinske faktore prateći vremensku promjenu odziva. Primjećujemo kako neuralna mreža blisko prati signal cilja. Funkcija vraća ažuriranu mrežu, izlaz i prikazuje nastalu grešku.

```
[net,Y] = adapt(net,X,T,Xi);

%Prikazat ćemo izlazni signal i cilj

plot(timex,cell2mat(Y),timex,cell2mat(T),'+')
xlabel('Time');
ylabel('Izlazni signal -      signal cilja +');
title('Izlazni signal i signal cilja');
%prikazat ćemo grafički i pogrešku E
E = cell2mat(T)-cell2mat(Y);
plot(timex,E,'r')
hold off
xlabel('vrijeme');
ylabel('pogreška');
title('Signal pogreške');
```





Možemo primijetiti da je **pogreška**, osim za početak signala, **izuzetno mala** jer **neuralna mreža uči** ponašanje sustava na početku i nakon prijelaza. Pokazali smo kako simulirati adaptivnu linearnu mrežu koja predviđa slijedeću vrijednost signala iz tekuće i proteklih vrijednosti usprkos promjenama ponašanja signala.