Neuronske mreže

Matko Bošnjak, 2011

Uvod

- Automatizirana obrada podataka pogodna za izvršavanje na računalu
- Neautomatizirane obrade podataka izvršavaju živčani sustavi
  - procesiranje prirodnoga jezika
  - rješavanje problema
  - prepoznavanje lica
    - čovjek obavlja u 100-200 ms (100ms potrebno za raspoznavanje lica majke!)
    - računalu treba više vremena, a točnost je upitna
  - itd.
- Tražimo koncept obrade podataka sličan funkcioniranju mozga

- …da li je moguće “kopirati” rad mozga? …krenimo s jednostavnijim
- 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

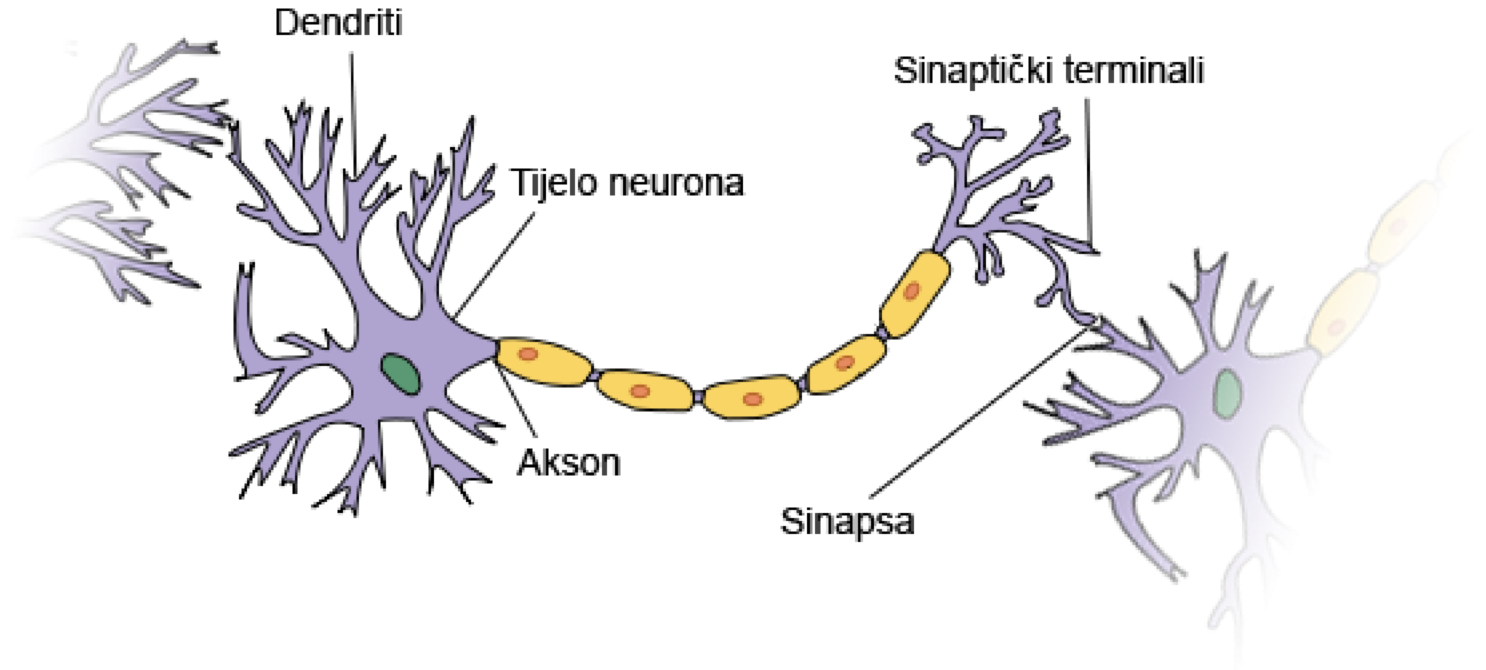
Usporedba mozak - računalo

- Mozak izvršava obradu podataka na potpuno drugačiji način od konvencionalnih digitalnih računala (von Neumannova računala)

	Mozak	Računalo
gradbeni element	**10^{11} neurona**	1.17 B tranzistora (6c 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	**u većini slučajeva ☺**	ne (još?)

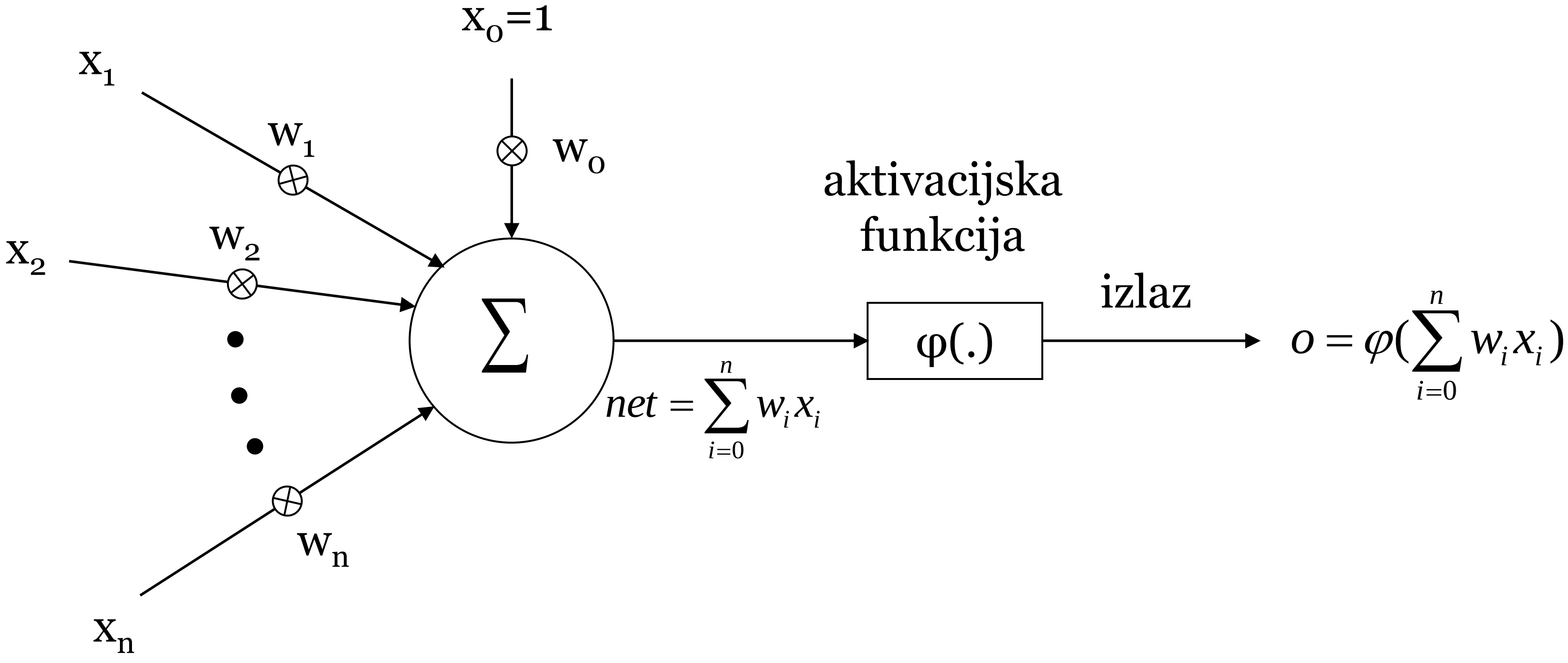
Biološka pozadina

- Dendriti primaju signale drugih neurona
- Akson prenosi impulse do sinaptičkih terminala
- Oni zatim prenose signale na dendrite drugih neurona



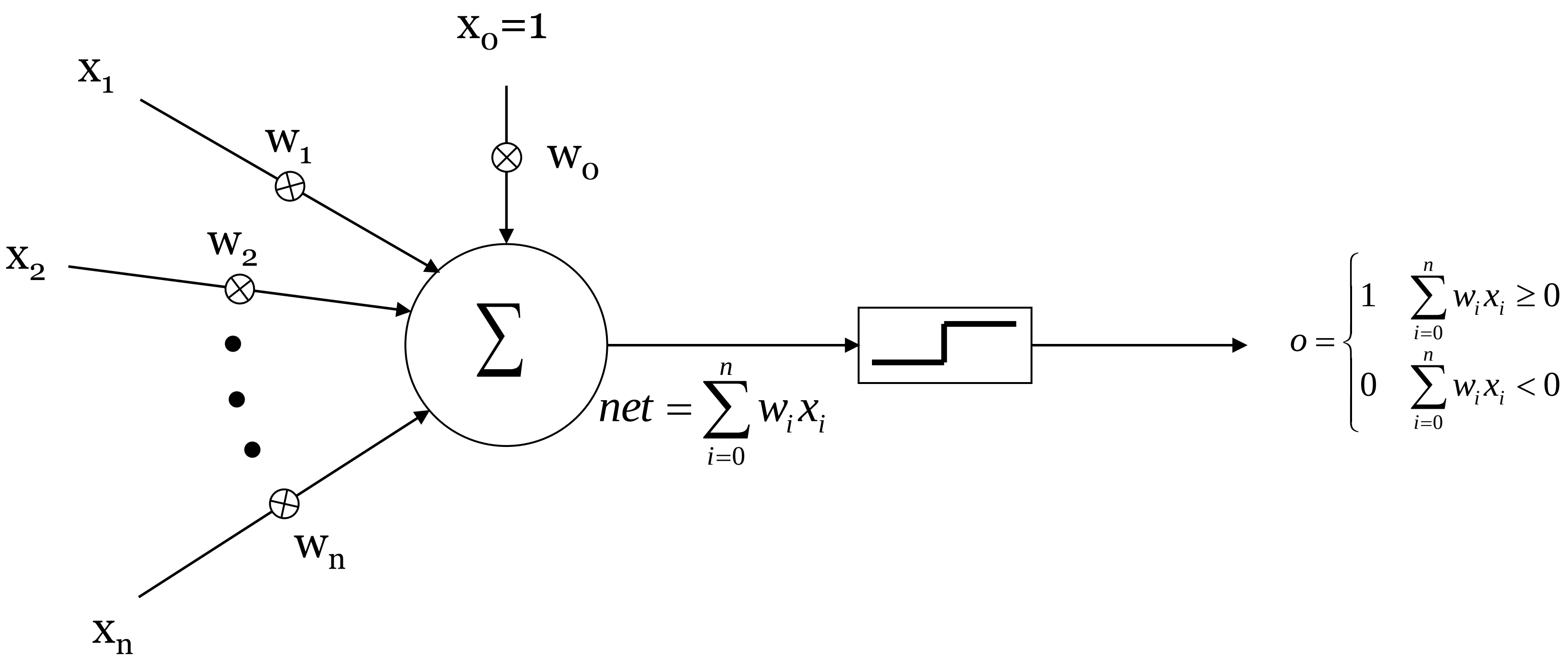
Umjetni "ekvivalent"

- 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 (Σ), a akson aktivacijska funkcija (φ)



Perceptron

- Najjednostavniji oblik neuronske mreže
- Umjetni neuron s funkcijom praga kao aktivacijskom funkcijom
- 1957, Rosenblatt



Reprezentacijska moć perceptrona

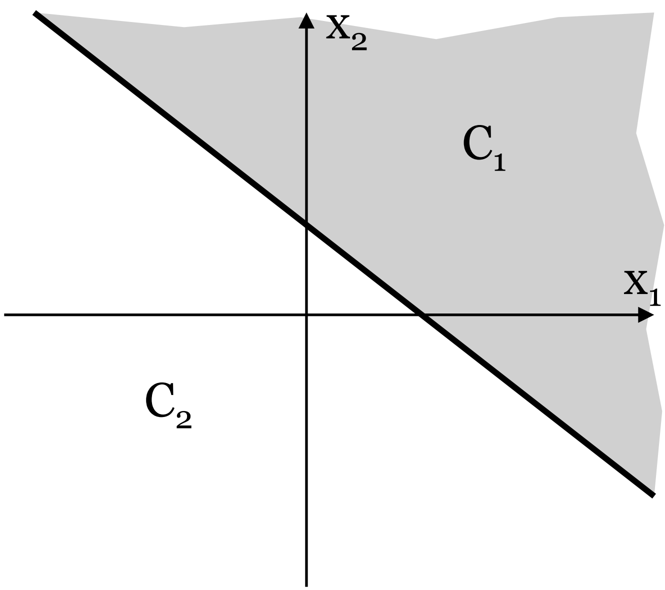
- Pogledajmo malo pobliže izlaz perceptrona

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

- To je jednadžba decizijske hiperravnine u n-dimenzionalnom prostoru

$$\sum_{i=0}^{n} w_i x_i \geq 0 \Rightarrow x \in C_1$$

$$\sum_{i=0}^{n} w_i x_i < 0 \Rightarrow x \in C_2$$



- Perceptron omogućava klasifikaciju!
  - SAMO linearno odvojivih razreda
- U dvije dimenzije
  - decizijski pravac: $w_0 + w_1 x_2 + w_2 x_2 = 0$

Perceptron?

- I što ćemo s perceptronom?
- **Naučiti ga! ☺**
- Čovjek uči tj. usvaja znanja, vještine…navike

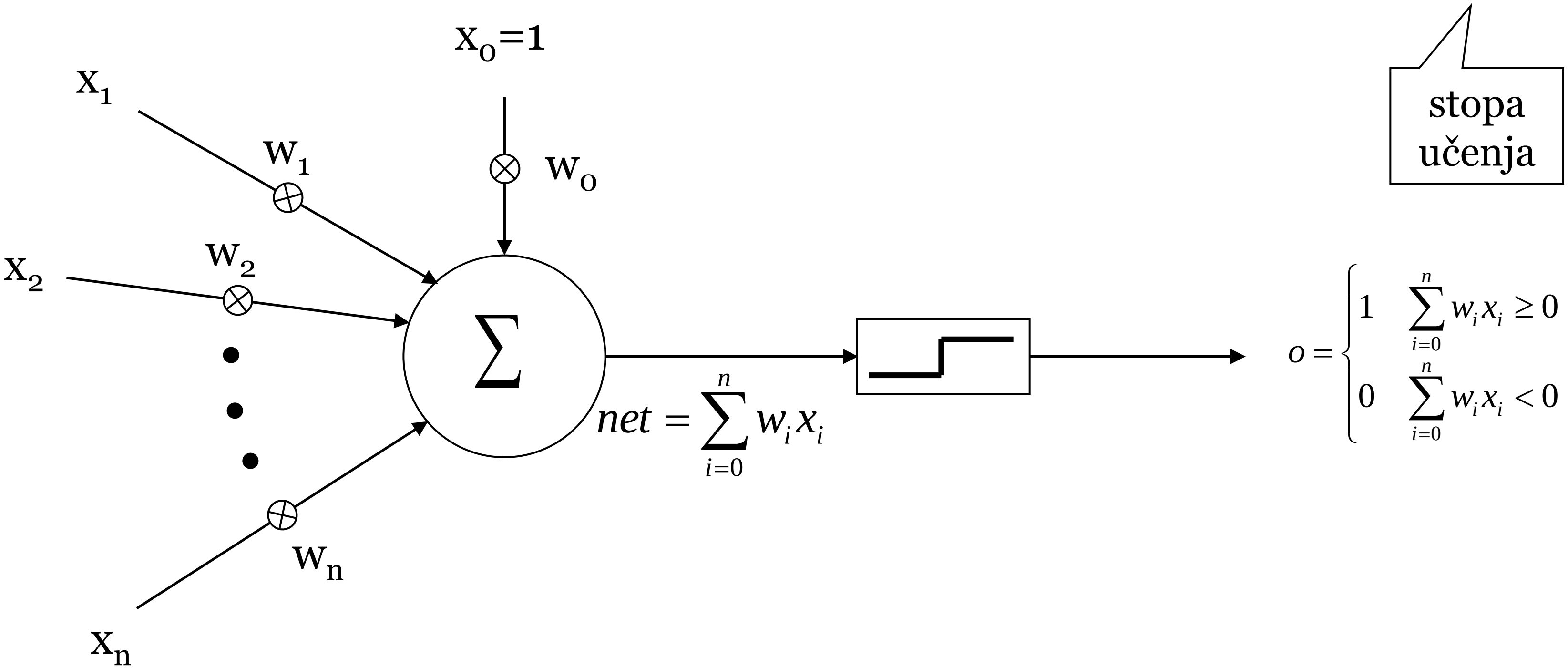
- A neuronska mreža (za početak perceptron)?
  - Hebbova teorija – Učiti znači mijenjati jakost veza
  - adaptacija slobodnih parametara mreže (težina) kroz kontinuiranu stimulaciju okoline

- Paradigme učenja:
  - učenje pod nadzorom ← trenutno nas ovo zanima
  - učenje bez nadzora
  - učenje podrškom

- Što uopće možemo mijenjati (učiti) kod perceptrona?
  - težinske faktore

Učenje perceptrona

- Primjeri za učenje: *($\boldsymbol{x}$, t)* – $\boldsymbol{x}$ 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 $w_i \leftarrow w_i + \Delta w_i \qquad \Delta w_i = \eta(t-o)x_i$



Učenje perceptrona - algoritam

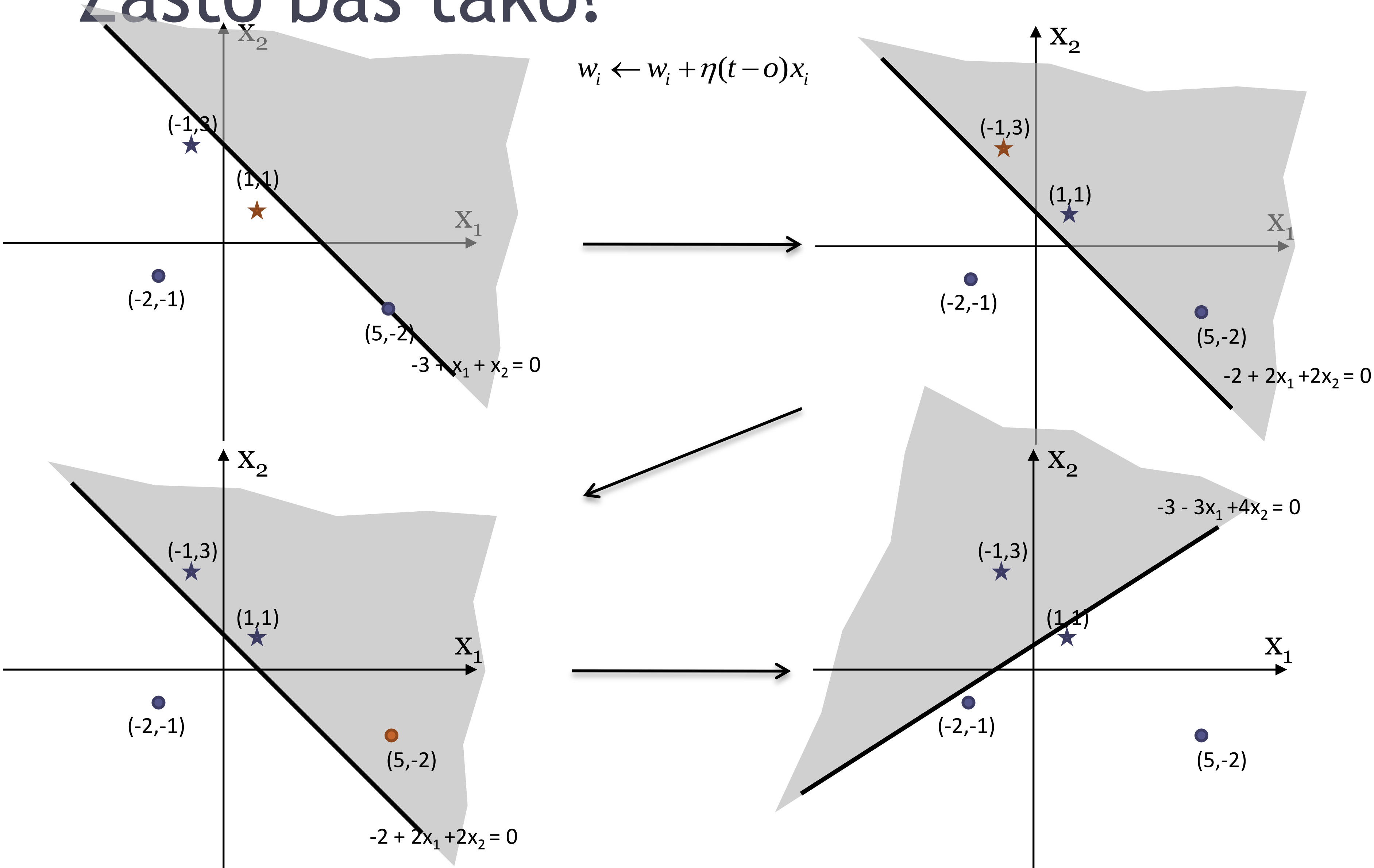
PERCEPTRON(skup za učenje, η)
    postavi težine na nasumično odabrane vrijednosti
    dok nisu svi uzorci ispravno klasificirani
        za svaki uzorak iz skupa učenja
            klasificiraj uzorak perceptronom
            ako je uzorak ispravno klasificiran nastavi sa sljed. uzorkom
            inače primjeni korekciju:

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

- Algoritam konvergira u konačnom broju koraka ako su primjeri za učenje linearno odvojivi i ako je stopa učenja, η dovoljno malena (Minsky i Papert, 1969.)

Zašto baš tako?

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



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 – linearna jedinica (tzv. Adaline)

$$o = \sum_{i=0}^{n} w_i x_i$$

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

$$E(\vec{w}) = \frac{1}{2}\sum_{d \in D}(t_d - o_d)^2 \longrightarrow \Delta w_i = -\eta \frac{\partial E}{\partial w_i} \longrightarrow \Delta w_i = \eta \sum_{d \in D}(t_d - o_d) x_{id}$$

- 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

GRADIJENTNI-SPUST(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 w_i izračunaj

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

za svaku težinu w_i izačunaj

$$w_i \leftarrow w_i + \Delta w_i$$

- 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

Stoh. Gradijentni spust – algoritam

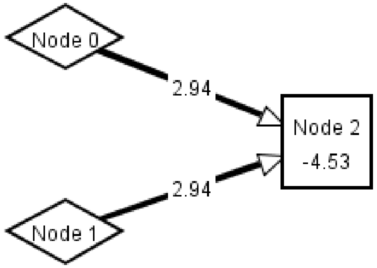
STOHASTIČKI-GRADIJENTNI-SPUST(skup za učenje, η)

- postavi težine na nasumično odabrane vrijednosti
- **dok** nije zadovoljen kriterij zaustavljanja
  - **za svaki** uzorak iz skupa učenja
    - klasificiraj uzorak perceptronom
    - **za svaku** težinu w_i izračunaj
      - $$w_i \leftarrow w_i + \eta(t - o)x_i$$

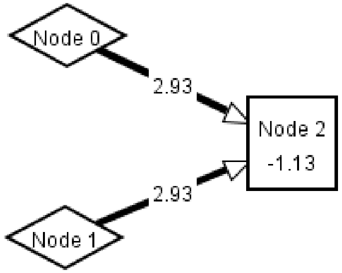
- Delta pravilo još je poznato kao i LMS (least-mean-square), Adaline pravilo ili Widrow-Hoff pravilo

Primjeri

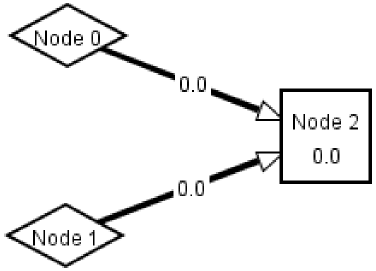
- Neural Networks applet @ www.aispace.org
- AND



OR



XOR



Winnow algoritam

- Algoritam perceptrona je aditivan algoritam
  - težine se mijenjaju tako da se dodaje ili oduzima neki pomak
- Međutim, postoji i multiplikativan algoritam
  - umjesto da dodajemo vrijednost težini, samu težinu množimo nekim faktorom

- Winnow algoritam
  - Linearni klasifikator
  - Bolji je u slučajevima s mnogo beskorisnih dimenzija
  - Može se koristiti za online učenje

- Promatramo ga na prostoru instanci $X = \{0,1\}^n$

Winnow algoritam

WINNOW(skup za učenje, α, θ)
  postavi težine na 1
  dok nisu svi uzorci ispravno klasificirani
    za svaki uzorak iz skupa učenja
      klasificiraj uzorak

$$o = \begin{cases} 1 & \sum w_i x_i \geq \theta \\ 0 & \sum w_i x_i < \theta \end{cases}$$

      ako je uzorak ispravno klasificiran **nastavi** sa sljedećim uzorkom
      inače primjeni korekciju:

$$w_i \leftarrow \begin{cases} \alpha^{x_i} w_i & t = 1 \\ \frac{1}{\alpha^{x_i}} wi & t = 0 \end{cases}$$

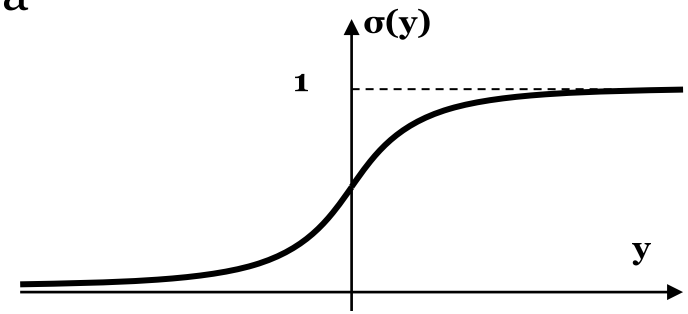
Neuronske mreže

- Definirali smo umjetni neuron, vidjeli smo perceptron sa step funkcijom i linearnu jedinicu (Adaline) bez aktivacijske funkcije
- Sve što smo do sada radili je bilo vezano uz jedan procesni element
- Nameće nam se mogućnost povezivanja više procesnih elemenata
- No, koji procesni element ćemo koristiti kao gradbeni element takve mreže?
  - linearnu jedinicu?
    - linearna kombinacija linearnih funkcija je opet linearna funkcija – ništa od veće ekspresivnosti
  - perceptron?
    - aktivacijska funkcija perceptrona (step funkcija) je nediferencijabilna pa ne možemo koristiti gradijentni spust
- Treba nam element s nelinearnom diferencijabilnom aktivacijskom funkcijom

Sigmoidna funkcija

- Sigmoidna (logistička) funkcija

$$\sigma(y) = \frac{1}{1+e^{-ky}}$$



  - glatka funkcija praga
  - nelinearna
  - diferencijabilna
  - posjeduje zgodno svojstvo da je

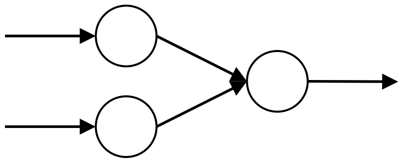
$$\frac{d\sigma(y)}{dy} = \sigma(y)(1-\sigma(y))$$

  - monotono raste s ulazom
  - konstanta k utječe na strminu

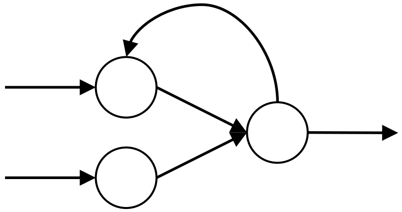
- No možemo koristiti i druge funkcije: tanh, linearna po odsječcima, itd.

Topologija neuronskih mreža

- Ok, imamo “novi tip” umjetnog neurona, ali kako ćemo ga povezati s ostalim neuronima?
- Izlaz jednog neurona predstavlja (može predstavljati) ulaz sljedećemu
- Podjela neuronskih mreža po topologiji (arhitekturi mreže)
- Osnovna podjela
  - aciklične – ne sadrže povratne veze

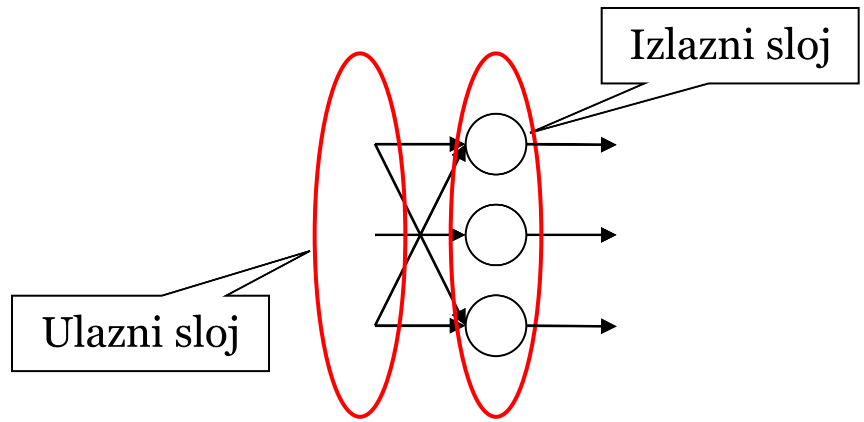


  - cikličke – sadrže povratne veze (tema za sebe)

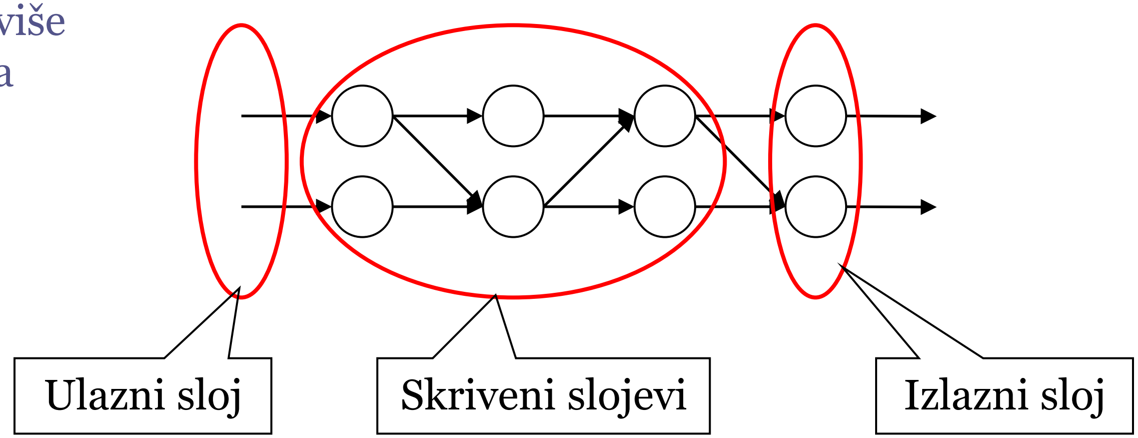


Topologija neuronskih mreža

- Po broju slojeva
  - jednoslojne
    - ulazni sloj se ne računa jer nije procesni sloj

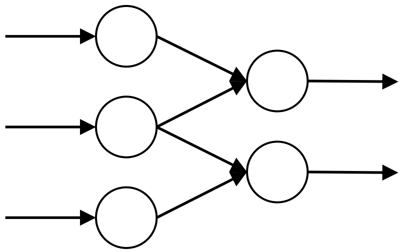


  - višeslojne
    - sadrže jedan ili više skrivenih slojeva

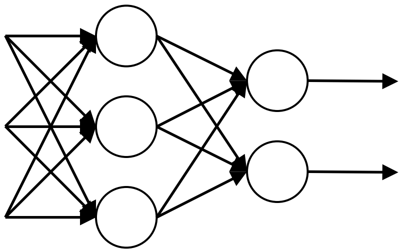


Topologija neuronskih mreža

- Po povezanosti
  - djelomično povezane

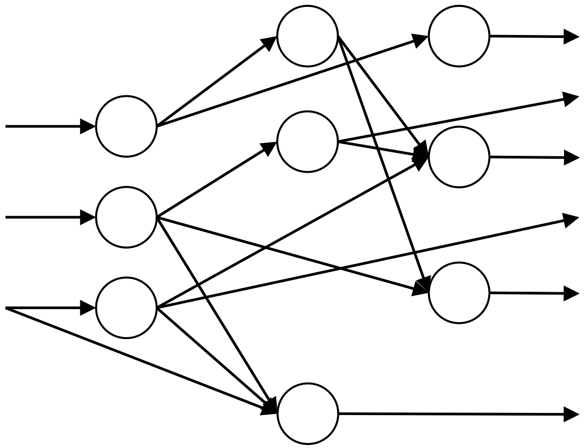


  - potpuno povezane
    - svaki neuron prethodnog sloja povezan je sa svakim neuronom sljedećeg sloja



Topologija neuronskih mreža

- Proizvoljna aciklička mreža



- Topologiju moguće naučiti!!
  - recimo genetskim algoritmom

- Savladali smo gradbeni element, savladali smo topologiju…
- Kako ćemo sada učiti mreže?

Učenje neuronskih mreža

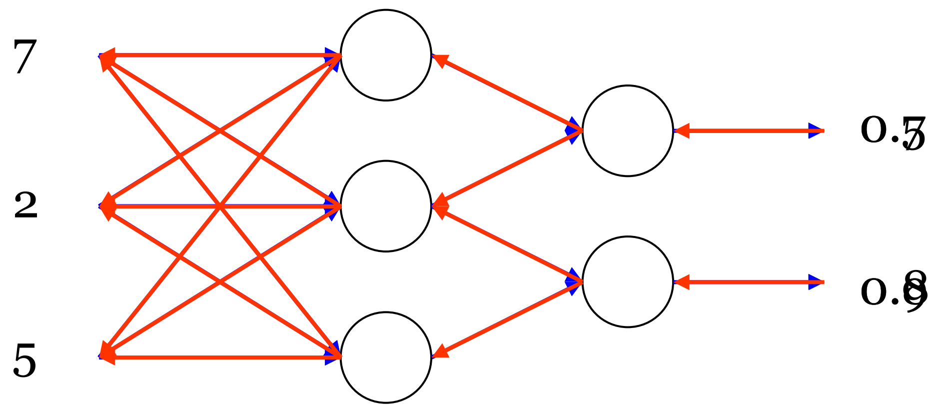
- U slučaju jednoslojne neuronske mreže prilično jednostavno – gradijentni spust
- Ali kako ćemo učiti višeslojnu neuronsku mrežu?
  - isto gradijentnim spustom?
- Tu nailazimo na problem: znamo korigirati težine izlaznih neurona jer kod njih točno znamo iznos pogreške, međutim kolika je pogreška za skrivene neurone? Iz čega nju računati?
- Formula pogreške učenja hipoteza (težina)

$$E(\vec{w}) = \frac{1}{2} \sum_{d \in D} \sum_{k \in output} (t_{kd} - o_{kd})^2 \qquad \Delta w_{ji} = -\eta \frac{\partial E_d}{\partial w_{ji}}$$

  - problem višestrukih lokalnih minimuma – zato ćemo koristiti stohastički gradijentni spust
  - ovisi o izlaznom sloju...a skriveni?

Backpropagation

- Problem:
  - nepoznat željeni odziv neurona u skrivenom sloju
- Rješenje:
  - pogreška se može procijeniti na temelju pogrešaka neurona u sljedećem sloju sa kojima je skriveni neuron spojen



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 o_u 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

$$\delta_h \leftarrow o_h(1-o_h) \sum_{s \in Downstream(h)} w_{sh}\delta_s$$

ugodi svaki težinski faktor w_{ji}

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

Konvergencija i lokalni minimumi

- Backpropagation garantira konvergenciju SAMO u lokalni minimum i ne mora nužno doseći globalni minimum
- Unatoč tome, backpropagation je u praksi izuzetno efektivna metoda aproksimacije funkcija
- Tu se možemo osvrnuti i na korak algoritma

  Inicijaliziraj težinske faktore na male slučajne vrijednosti
- Zašto na male slučajne vrijednosti?
  - na početku gradijentnog spusta, mreža predstavlja vrlo glatku funkciju
  - pogledati sigmoidnu funkciju (18)

- Heuristike za nadjačavanje problema lokalnih minimuma
  - dodavanje momenta u pravilo korekcije težina
  - uporaba stohastičkog gradijentnog spusta
  - učenje višestrukih mreža pa odabir najbolje (ili upotreba ansambla)

Reprezentacijska moć

- Kakve sve funkcije mogu neuronske mreže reprezentirati?
  - odgovor dakako ovisi o topologiji mreže
  - generalno:
    - booleove funkcije
    - kontinuirane funkcije
    - proizvoljne funkcije

- Neuronske mreže mogu aproksimirati proizvoljnu funkciju proizvoljnom preciznošću (Kolmogorovljev egzistencijalni teorem)

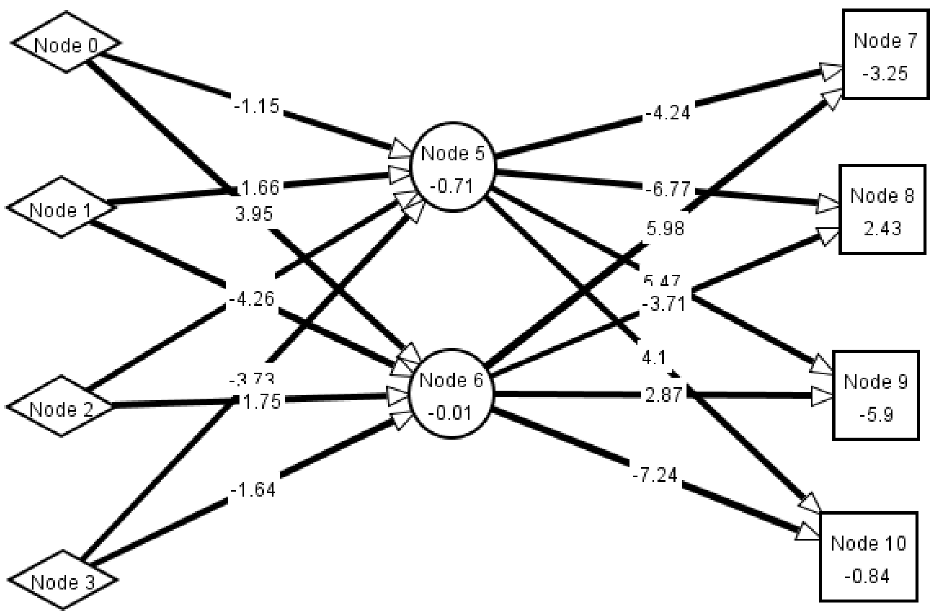
Reprezentacija skrivenog sloja

- Interesantno je zagledati se u skriveni sloj
  - znamo što su izlazni i ulazni sloj, ali što je skriveni?
- 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

Primjer

- Dekoder



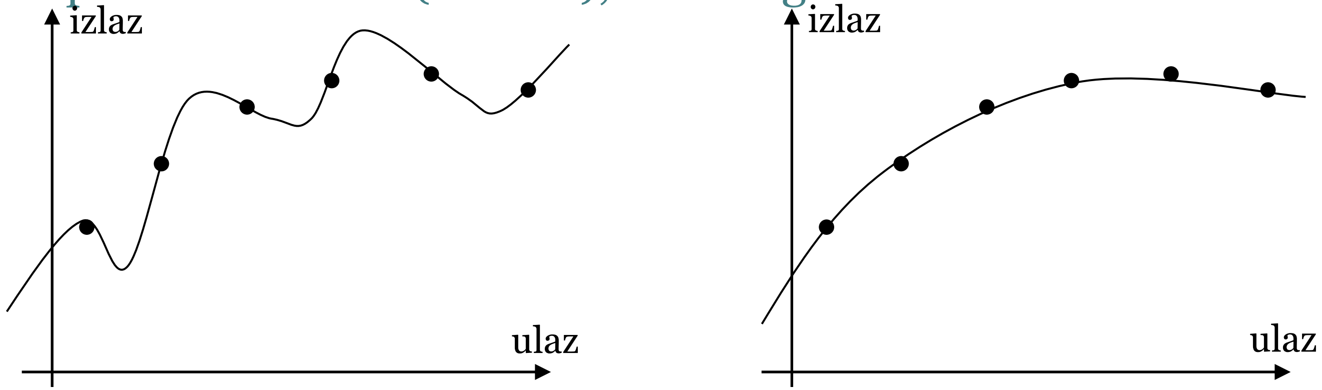
Kriteriji zaustavljanja

- Fiksni broj iteracija petlje
  - ok, ali 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 dostan važan jer
  - premalo iteracija može neznatno smanjiti pogrešku
    - time dobivamo "lošiju" mrežu nego bismo mogli
  - previše može odvesti u overfitting

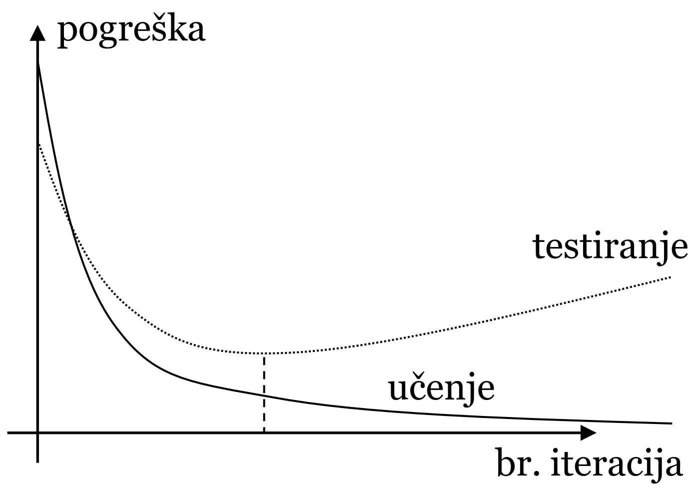
Generalizacija i overfitting

- Svojstvo dobre klasifikacije za nepoznate ulaze zove se generalizacija
- Cilj generalizacije: interpolirati raspoložive podatke najjednostavnijom krivuljom
- Prenaučenost (eng. overfitting)
    - prevelik broj primjera za učenje (eng. overfitting) uzrokuje gubitak svojstva generalizacije, UNM postaje stručnjak za podatke iz skupa za učenje
    - forsiranje što manje pogrešne na skupu za učenje – mreža postaje ekspert za naučeno (štreber), ali loše generalizira



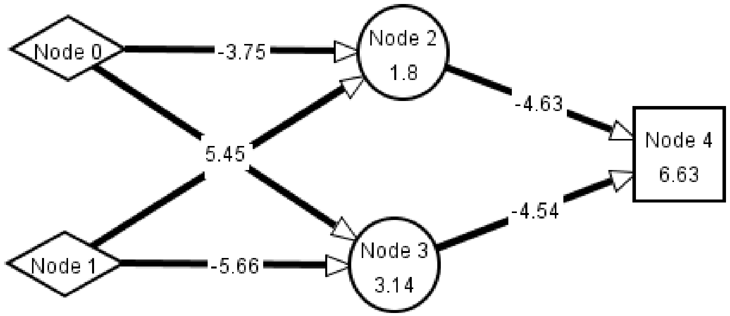
Overfitting

- Kako se riješiti overfittinga?
  - weight decay
    - smanjivanje težina u svakoj iteraciji za neki maleni faktor (sprječavamo kompleksne aproksimacije)
  - uvođenje seta za validaciju
    - daje nam mjeru pogreške za vrijeme učenja
    - počinje rasti kada mreža uči posebnosti skupa za učenje
  - u slučaju malih skupova za učenje
    - k-fold cross-validation

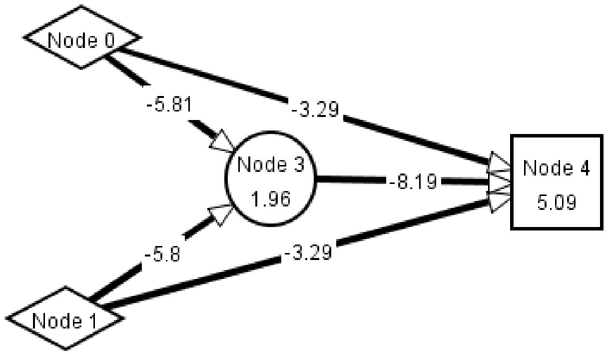


Primjeri

- XOR

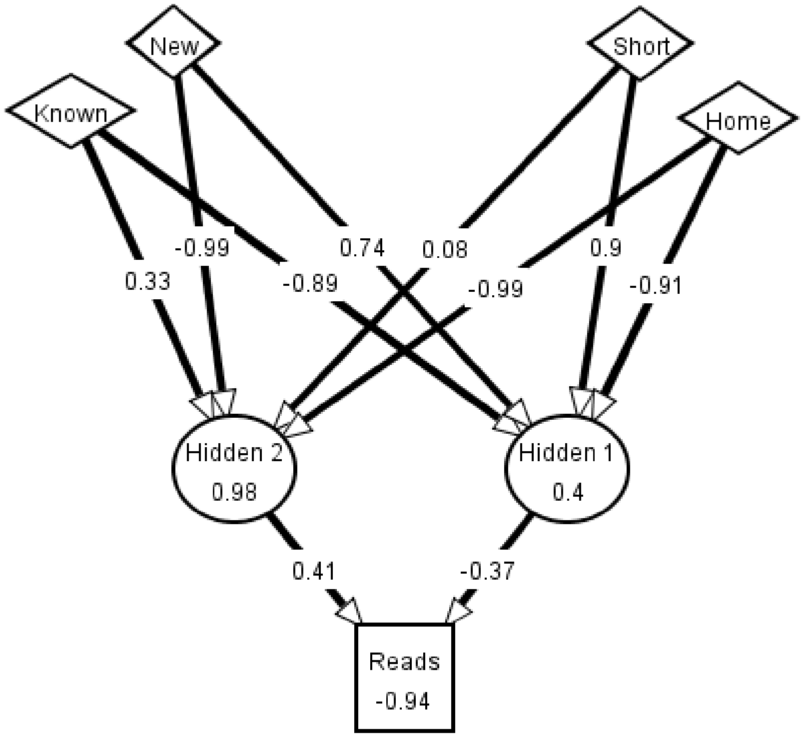


- XOR



Primjeri

- Mail reading



Malo detaljniji primjer

- ZmajMathOCR
  - prepoznavanje i evaluacija rukom pisanih matematičkih izraza i iscrtavanje funkcija jedne varijable