Neuronske mreže: Višeslojni perceptron

Prof. dr. sc. Sven Lončarić

Fakultet elektrotehnike i računarstva
https://www.fer.unizg.hr/predmet/neumre_c

Pregled predavanja

- Uvod
- Višeslojni perceptron
- Učenje s povratnom propagacijom pogreške
- Diskusija
- Zadaci

Uvod

- U ovom poglavlju govorit ćemo o višeslojnim neuronskim mrežama bez povratnih veza
- engl. multilayer feed-forward network, multilayer perceptron (MLP)
- MLP mreže imaju jedan ulazni sloj, jedan izlazni sloj i jedan ili više skrivenih slojeva neurona
- MLP mreže se koriste za rješavanje širokog spektra problema gdje se učenje pod nadzorom odvija pomoću algoritma s povratnom propagacijom pogreške
- engl. error back-propagation algorithm

Uvod

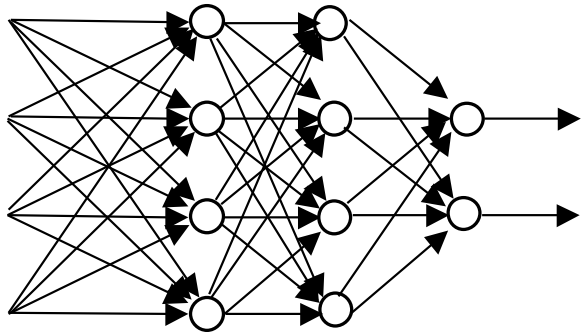
- Višeslojni perceptron ima tri glavna svojstva:
- Model neurona ima nelinearnu izlaznu karakteristiku koja je glatka (za razliku od Rosenblattovog perceptrona)
- Često se koristi sigmoidna aktivacijska funkcija oblika:

$$y_i = \frac{1}{1 + \exp(-v_i)}$$

- Mreža sadrži jedan ili više skrivenih slojeva neurona
- Mreža je dobro povezana (postoji puno sinapsi)

Višeslojni perceptron

- Mreža na slici sadrži jedan ulazni sloj, dva skrivena sloja i jedan izlazni sloj neurona
- Broj neurona u pojedinom sloju može biti različit
- Izlazi neurona jednog sloja su ulazi u sljedeći sloj
- Nema povratnih veza



Uvod

- Navedene karakteristike temelj su dobrih svojstava višeslojnih perceptrona no isto tako i slijedećih mana:
  - Višestruke nelinearnosti i visoka povezanost otežava teoretsku analizu ove mreže
  - Korištenje skrivenih neurona otežava vizualizaciju procesa učenja
  - Proces učenja je teži zbog velikog broja neurona i zbog potrebe da mreža sama odluči što trebaju naučiti skriveni neuroni (za izlazne neurone se zna)

BP algoritam učenja

- engl. error back-propagation
- Signal pogreške na izlazu neurona *j* u koraku *n* jednak je razlici između željenog i dobivenog odziva:

  $e_j(n) = d_j(n) - y_j(n)$

  gdje je neuron *j* izlazni neuron
- Trenutna (u koraku *n)* kvadratna pogreška na izlazu mreže jednaka je:

$$E(n) = \frac{1}{2}\sum_{j \in C} e_j^2(n)$$

  gdje je C skup izlaznih neurona

BP algoritam

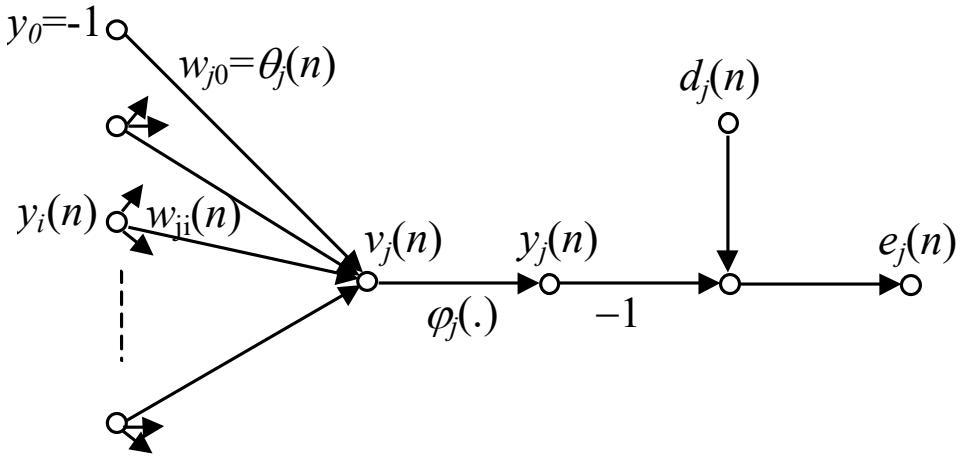
- Prosječna kvadratna pogreška za sve uzorke jednaka je:

$$E_{sr} = \frac{1}{N} \sum_{n=1}^{N} E(n)$$

  gdje je *N* broj uzoraka u skupu za učenje
- E_{sr} predstavlja ocjenu kvalitete učenja mreže
- Cilj učenja je odrediti težine tako da se minimizira srednja pogreška E_{sr}
- Ovaj cilj se postiže algoritmom sličnim LMS učenju gdje se težine podešavaju za svaki dani uzorak za učenje

BP algoritam

- Pretpostavimo da imamo neuron na slici:



BP algoritam

- Aktivacija na ulazu u nelinearni blok neurona *j* jednaka je:

$$v_j(n) = \sum_{i=0}^{p} w_{ji}(n) y_i(n)$$

  gdje je *p* ukupni broj ulaza u neuron *j*

- Izlaz neurona *j* jednak je:

$$y_j(n) = \varphi_j(v_j(n))$$

BP algoritam

- Korekcija težine izvodi se slično kao kod LMS algoritma u smjeru negativnog gradijenta
- Gradijent se može izračunati na sljedeći način:

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = \frac{\partial E(n)}{\partial e_j(n)} \frac{\partial e_j(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} \frac{\partial v_j(n)}{\partial w_{ji}(n)}$$

$$\frac{\partial E(n)}{\partial e_j(n)} = e_j(n) \qquad \frac{\partial y_j(n)}{\partial v_j(n)} = \varphi_j'(v_j(n))$$

$$\frac{\partial e_j(n)}{\partial y_j(n)} = -1 \qquad \frac{\partial v_j(n)}{\partial w_{ji}(n)} = y_i(n)$$

BP algoritam

- Na temelju prethodnih izraza dobivamo:

$$\frac{\partial E(n)}{\partial w_{ji}(n)} = -e_j(n)\varphi_j^{'}(v_j(n))y_i(n)$$

- Korekcija težina definirana je delta pravilom (metoda najbržeg spusta):

$$\Delta w_{ji}(n) = -\eta \frac{\partial E(n)}{\partial w_{ji}(n)}$$

BP algoritam

- Dakle dobiveni izraz može se pisati kao:

$$\Delta w_{ji}(n) = \eta e_j(n)\varphi_j^{'}(v_j(n))y_i(n) = \eta\delta_j(n)y_i(n)$$

gdje je $\delta_j(n)$ tzv. lokalni gradijent:

$$\delta_j(n) = e_j(n)\varphi_j^{'}(v_j(n))$$

- Korekcija težine ovisi o pogrešci $e_j(n)$ na izlazu neurona *j*
- Za računanje pogreške imamo dva slučaja:
  - Neuron *j* je izlazni neuron
  - Neuron *j* je skriveni neuron

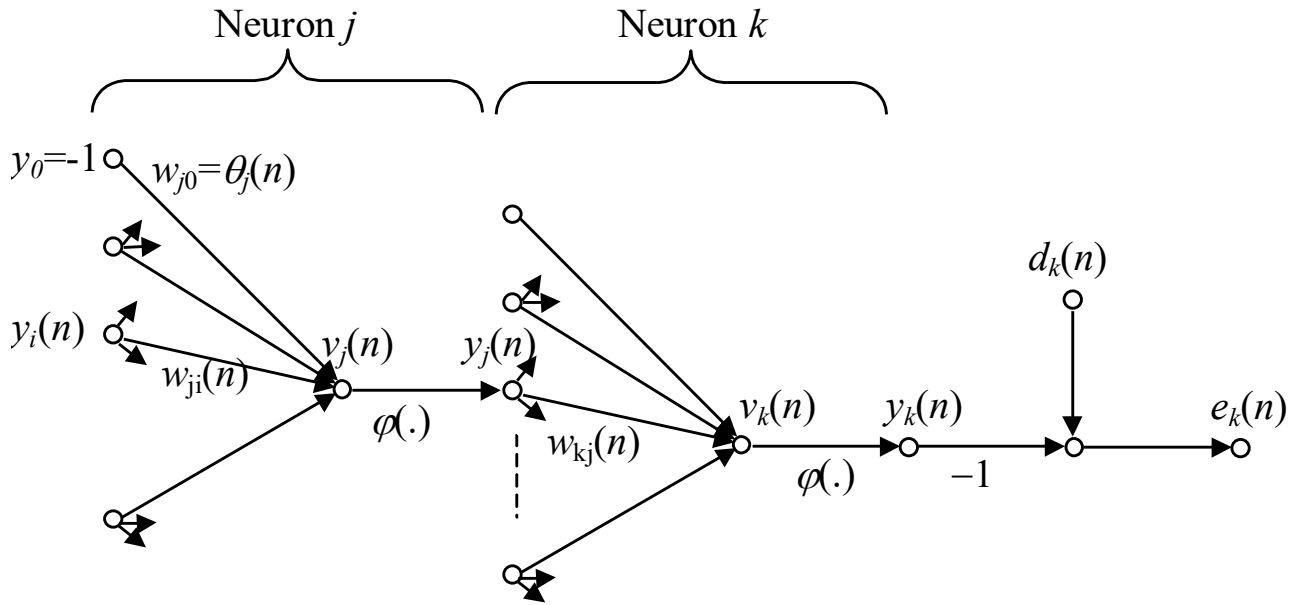
Neuron *j* je izlazni neuron

- Ako je neuron *j* u izlaznom sloju onda je pogrešku $e_j(n)$ lako izračunati budući da je poznat željeni odziv $d_j(n)$
- Lokalni gradijent $\delta_j(n)$ računa se pomoću prethodno izvedenog izraza:

$$\delta_j(n) = e_j(n)\varphi_j'(v_j(n))$$

Neuron *j* je skriveni neuron

- Pretpostavimo da se neuron *j* nalazi u skrivenom sloju spojenom na izlazni sloj (vidi sliku):



Neuron *j* je skriveni neuron

- Ako je neuron *j* u skrivenom sloju onda nije poznat željeni odziv tog neurona pa ni pogreška
- U tom slučaju pogreška skrivenog neurona može se procjeniti na temelju pogrešaka neurona u slijedećem sloju na koje je skriveni neuron spojen
- Prema prethodno izvedenim izrazima može se pisati:

$$\delta_j(n) = -\frac{\partial E(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial v_j(n)} = -\frac{\partial E(n)}{\partial y_j(n)} \varphi_j'(v_j(n))$$

- Parcijalnu derivaciju pogreške po ulazu možemo izračunati na slijedeći način:

Neuron *j* je skriveni neuron

- Krenimo od izraza:

$$E(n) = \frac{1}{2} \sum_{k \in C} e_k^2(n)$$

- Tražena parcijalna derivacija je onda jednaka:

$$\frac{\partial E(n)}{\partial y_j(n)} = \sum_k e_k \frac{\partial e_k(n)}{\partial y_j(n)} = \sum_k e_k \frac{\partial e_k(n)}{\partial v_k(n)} \frac{\partial v_k(n)}{\partial y_j(n)}$$

- Potrebne dvije parcijalne derivacije s desne strane možemo dobiti kako je prikazano u nastavku:

Neuron *j* je skriveni neuron

- Budući da je:

$$e_k(n) = d_k(n) - y_k(n) = d_k(n) - \varphi_k(v_k(n))$$

onda vrijedi da:

$$\frac{\partial e_k(n)}{\partial v_k(n)} = -\varphi_k'(v_k(n))$$

Neuron *j* je skriveni neuron

- Interna aktivnost *k*-tog neurona jednaka je:

$$v_k(n) = \sum_{j=0}^{q} w_{kj}(n) y_j(n)$$

  gdje je *q* ukupni broj ulaza u neuron *k*

- Dakle vrijedi da je:

$$\frac{\partial v_k(n)}{\partial y_j(n)} = w_{kj}(n)$$

Neuron *j* je skriveni neuron

- Na temelju izvedenoga možemo pisati da je tražena parcijalna derivacija jednaka:

$$\frac{\partial E(n)}{\partial y_j(n)} = -\sum_k e_k(n)\varphi_k'(v_k(n))w_{kj}(n) = -\sum_k \delta_k(n)w_{kj}(n)$$

- Konačno možemo pisati da je za skriveni neuron *j* lokalni gradijent jednak:

$$\delta_j(n) = -\varphi_j'(v_j(n))\sum_k \delta_k(n)w_{kj}(n)$$

i na temelju lokalnog gradijenta može se izračunati korekcija težine za skriveni neuron *j*

Sažetak BP algoritma

- Korekcija $\Delta w_{ji}(n)$ za težinu koja ide od neurona *i* na neuron *j* računa se delta pravilom:

(korekcija) = (parametar) × (lokalni gradijent neurona *j*) × (*i*-ti ulaz u neuron *j*)

$$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n)$$

- Ako je neuron *j* izlazni neuron onda:

$$\delta_j(n) = \varphi_j'(v_j(n)) e_j(n)$$

- Ako je neuron *j* skriveni neuron onda:

$$\delta_j(n) = \varphi_j'(v_j(n)) \sum_k \delta_k(n) w_{kj}(n)$$

Dva prolaza u računanju

- U praktičnoj izvedbi BP algoritma mogu se prepoznati dva prolaza:
    1. prolaz unaprijed (engl. forward pass)
    2. povratni prolaz (engl. backward pass)

Prolaz unaprijed

- U prolazu prema naprijed težine se ne mijenjaju a izlazi neurona se računaju počevši od ulaznog sloja prema izlaznom sloju
- Signali se računaju prema ranije spomenutim izrazima za internu aktivnost i izlaz pojedinog neurona:

$$v_j(n) = \sum_{i=0}^{p} w_{ji}(n) y_i(n)$$

$$y_j(n) = \varphi_j\left(v_j(n)\right)$$

Povratni prolaz

- Povratni prolaz započinje računanje od izlaznog sloja
- Signali pogreške se propagiraju od izlaznog sloja prema ulazu, sloj po sloj, rekurzivno računajući lokalni gradijent δ za svaki neuron
- Na temelju delta pravila za svaki neuron računa se korekcija težina kako je ranije prikazano u sažetku BP algoritma

Sigmoidna aktivacijska funkcija

- Kao aktivacijska funkcija najčešće se koristi funkcija:

$$y_j(n) = \varphi_j\left(v_j(n)\right) = \frac{1}{1+\exp(-v_j(n))}$$

- U ranije izvedenim izrazima za računanje lokalnog gradijenta pojavljuje se derivacija ove funkcije:

$$\varphi_j'\left(v_j(n)\right) = \frac{\exp(-v_j(n))}{\left[1+\exp(-v_j(n))\right]^2} = y_j(n)\left[1-y_j(n)\right]$$

Sigmoidna aktivacijska funkcija

- Za neuron *j* koji je u izlaznom sloju vrijedi $y_j(n) = o_j(n)$
- Dakle za izlazni neuron *j* možemo pisati:

$$\begin{aligned}\delta_j(n) &= e_j(n)\varphi_j'(v_j(n)) \\ &= \left[d_j(n) - o_j(n)\right] o_j(n)\left[1 - o_j(n)\right]\end{aligned}$$

- Za skriveni neuron možemo pisati:

$$\begin{aligned}\delta_j(n) &= \varphi_j'(v_j(n))\sum_k \delta_k(n) w_{kj}(n) \\ &= y_j(n)\left[1 - y_j(n)\right]\sum_k \delta_k(n) w_{kj}(n)\end{aligned}$$

Sigmoidna aktivacijska funkcija

- Iz izraza za derivaciju sigmoidne funkcije vidi se da je derivacija najveća za $y_j(n) = 0.5$
- Za izlazne vrijednosti blizu 0 ili blizu 1 vrijednost derivacije je mala
- To znači da BP algoritam najviše korigira težine onih neurona čiji su izlazi srednje veličine (oko 0.5)
- To doprinosi stabilnosti BP algoritma učenja

Učenje s inercijom

- Iznos korekcije težine je proporcionalan parametru η koji određuje brzinu učenja η
- Premali parametar η uzrokuje sporo učenje
- Preveliki parametar η može biti uzrok nestabilnosti
- Za rješenje ovog problema delta pravilo može se modificirati dodatkom člana za inerciju:

$$\Delta w_{ji}(n) = \alpha \Delta w_{ji}(n-1) + \eta \delta_j(n) y_i(n)$$

- Član za inerciju usporava brzinu promjene korekcije težine (djeluje kao niskopropusni filtar)

Načini treniranja

- U izvedbi BP algoritma učenje se odvija uzastopnim prezentiranjem parova za treniranje
- Jedna prezentacija svih parova za treniranje naziva se epoha
- Učenje se odvija epoha za epohom dok se srednja kvadratna pogreška dovoljno ne smanji i težine i pragovi mreže ne stabiliziraju
- Redoslijed prezentacije parova treba slučajno mijenjati između epoha
- Na taj način se dobiva stohastičko pretraživanje u prostoru težina

Načini treniranja

- Postoje dva glavna pristupa treniranju
- U individualnom načinu treniranja (prezentiranom u prethodnom materijalu) korekcija težina se izvodi nakon svakog prezentiranog uzorka za učenje
  - prednost: jednostavnija izvedba
- U grupnom načinu treniranja mreži se prezentiraju svi parovi uzoraka iz skupa za treniranje i izračuna se pogreška za sve parove uzoraka te se korekcija težina izvodi jednom za sve uzorke
  - prednost: dobiva se točnija procjena gradijentnog vektora
  - mana: treba više memorije

Kriterij zaustavljanja učenja

- Ne može se dokazati da BP algoritam konvergira niti postoje točno definirani uvjeti za prestanak učenja
- Neki od uvjeta za prestanak iteracija su:

  1. gradijent na plohi pogreške je dovoljno mali (manji od nekog praga)
  2. promjena E_{sr} između dvije epohe je dovoljno mala
  3. mreža ima dobra svojstva generalizacije (provjeriti testiranjem)

Inicijalizacija mreže

- Prije početka učenja potrebno je iznose težina inicijalizirati na neke vrijednosti
- Ako postoji neko a priori znanje o mreži to znanje se može iskoristiti da se težine postave na neke određene vrijednosti
- Ako ne postoji a priori znanje (najčešći slučaj) onda se težine inicijaliziraju na neke slučajne vrijednosti uniformno distribuirane u nekom intervalu
- Pogrešan izbor može dovesti do preuranjenog zasićenja (pogreška E_{sr} se ne smanjuje kroz nekoliko epoha ali onda nakon toga se počne smanjivati - uzrok: sedlo na plohi pogreške)

Problem zasićenja neurona

- Ako za neki ulazni uzorak interna aktivnost nekog neurona ima jako pozitivnu ili jako negativnu vrijednost onda će izlaz tog neurona biti +1 ili -1 i derivacija u toj točki sigmoidne krivulje bit će jako mala
- Tada se kaže se da je neuron u zasićenju
- Ako je željena vrijednost -1, a stvarna +1, ili obratno, kaže se da je neuron pogrešno zasićen
- Kad se ovo desi treba puno iteracija da se izlaz korigira zbog male derivacije i prema tome male korekcije težina

Problem preuranjenog zasićenja

- Na početku procesa učenja u mreži imamo korektno i pogrešno zasićene neurone
- Nakon nekog vremena težine nezasićenih neurona brzo se mijenjaju jer je iznos derivacije veći i dolazi do trenutnog smanjenja iznosa pogreške E_{sr}
- Ako u tom momentu zasićeni neuroni i dalje ostaju zasićeni dolazi do pojave preuranjenog zasićenja iznosa pogreške E_{sr}
- Preuranjeno zasićenje pogreške E_{sr} je kad se pogreška kroz veći broj iteracija ne smanjuje nego ostaje konstantna i izgleda kao da je mreža završila učenje a zapravo nije

Problem preuranjenog zasićenja

- Načini smanjenja vjerojatnosti preuranjenog zasićenja su:

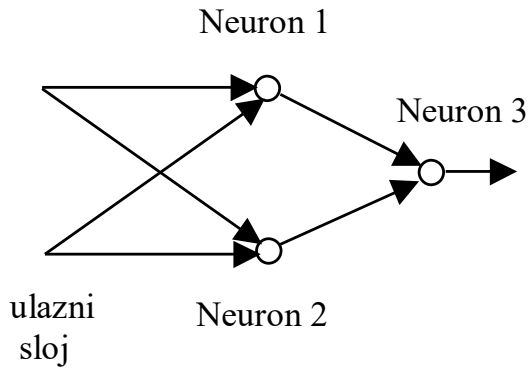
  1. Odabrati početne vrijednosti težina i pragova kao uniformno distribuirane vrijednosti unutar *uskog* intervala

  2. Vjerojatnost pogrešnog zasićenja je manja kad ima manje skrivenih neurona u mreži - koristiti minimalan neophodni broj skrivenih neurona

  3. Vjerojatnost pogrešnog zasićenja je manja kad neuroni rade u linearnom dijelu karakteristike

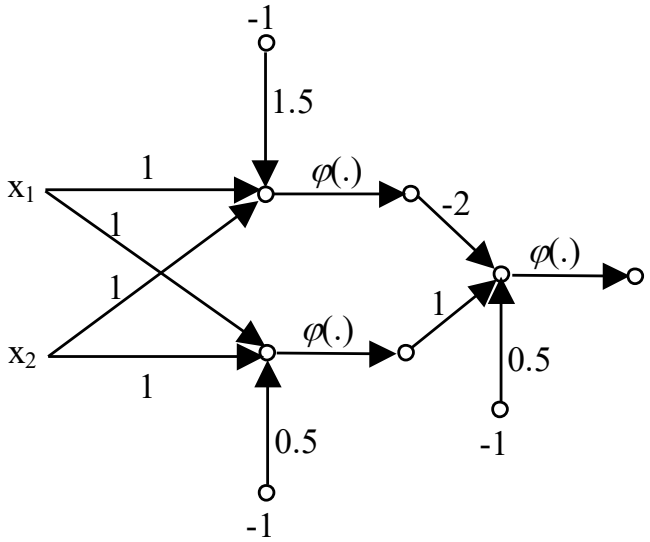
Primjer: XOR problem

- 0 XOR 0 = 0 0 XOR 1 = 1
- 1 XOR 1 = 0 1 XOR 0 = 1
- XOR problem može se rješiti npr. s dva skrivena neurona i jednim izlaznim neuronom:



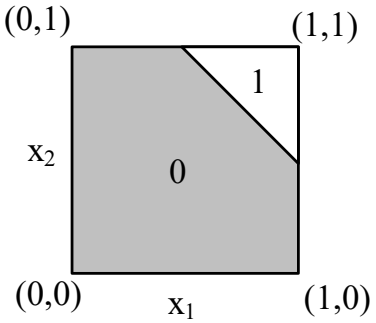
Primjer: XOR problem

- Dijagram toka neuronske mreže za realizaciju XOR funkcije



Primjer: XOR problem

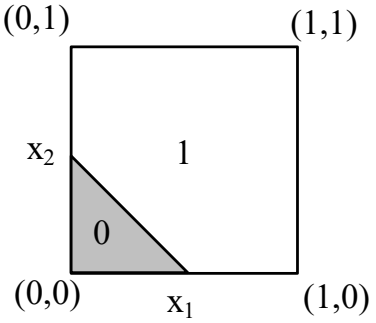
- Slika prikazuje područja klasifikacije pojedinih neurona



Neuron 1

$w_{11} = w_{12} = 1$

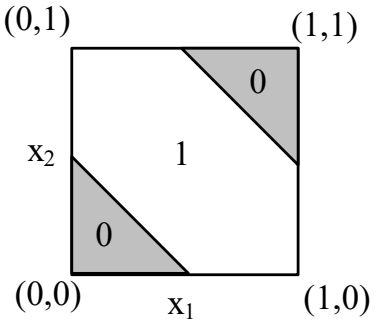
$\Theta_1 = 1.5$



Neuron 2

$w_{21} = w_{22} = 1$

$\Theta_2 = 0.5$



Neuron 3

$w_{31} = -2 \quad w_{32} = 1$

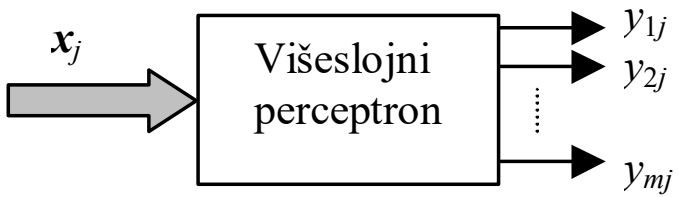
$\Theta_2 = 0.5$

Primjena u raspoznavanju uzoraka

- engl. pattern recognition
- Višeslojni perceptron vrlo se često koristi u raspoznavanju uzoraka
- U raspoznavanju uzoraka problem je klasificirati nepoznati uzorak u jednu od N klasa
- Nepoznati uzorak je vektor koji opisuje objekt ili pojavu koju treba klasificirati (npr. oblik objekta, izgovorenu riječ)

Problem klasifikacije uzoraka

- Pretpostavimo da treba rješiti problem klasifikacije vektora $\boldsymbol{x}_j$ u jednu od m klasa C_k , $k = 1, ..., m$
- Ovaj problem možemo rješiti mrežom koja ima m izlaza



- Mrežu treba trenirati tako da za ulazni vektor $\boldsymbol{x}_j$ koji pripada klasi C_k k-ti izlaz ima vrijednost 1, a ostali izlazi 0

Problem klasifikacije uzoraka

- Nakon što je proces učenja završen mreža radi tako da se na ulaz postavi nepoznati vektor **x** koji nije iz skupa vektora za učenje
- Mreža za ulazni vektor **x** na izlazu daje vektor **y** = $[y_1 \ldots y_m]^T$ na temelju kojeg treba odlučiti kojoj klasi pripada nepoznati vektor **x**
- Pravilo odlučivanja: Klasificiraj vektor **x** u klasu C_k ako vrijedi:

$$y_k > y_i \quad \forall i \neq k$$

Problem odabira parametara

- Pri projektiranju mreže potrebno je odrediti parametre mreže: broj skrivenih neurona, konstantu brzine učenja i konstantu inercije (ako se koristi učenje s inercijom)
- U praksi se najčešće eksperimentalno odrede optimalni parametri
- Kao kriterij za ocjenu koristi se točnost klasifikacije

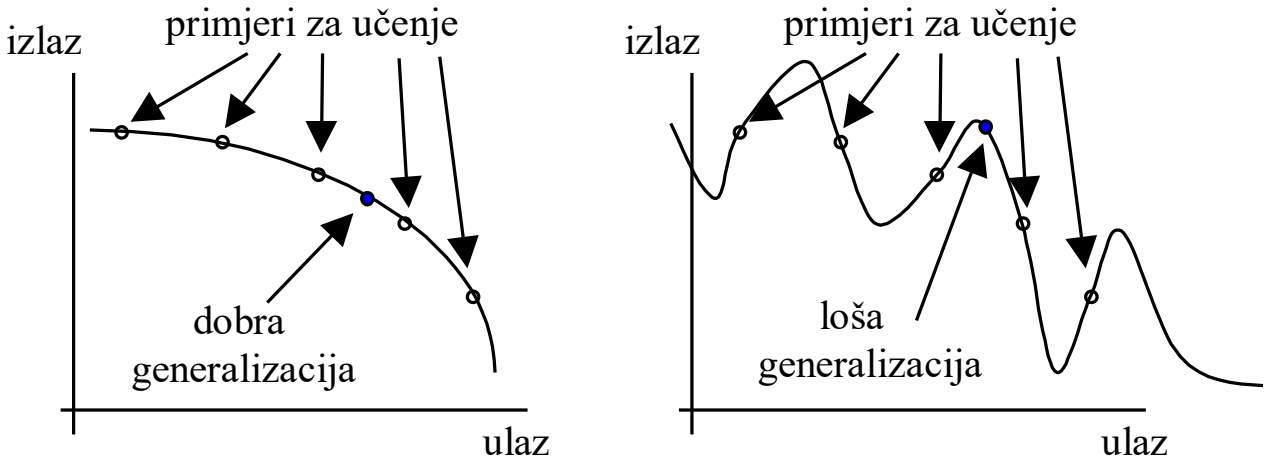
Generalizacija

- Kod učenja s BP algoritmom koristimo skup primjera za učenje kojim treniramo mrežu
- Koristimo maksimalni broj primjera koji su nam na raspolaganju
- Nakon što je proces učenja završen nadamo se da će mreža “dobro” raditi i za vektore koji nisu sadržani u skupu primjera s kojima je mreža trenirana
- To svojstvo se zove generalizacija i za mrežu se kaže da dobro generalizira kad radi dobru klasifikaciju i za nepoznate ulaze

Generalizacija

- Mreža se može promatrati kao nelinearno ulazno-izlazno preslikavanje
- Proces učenja može se interpretirati kao proces aproksimacije željenog ulazno-izlaznog preslikavanja
- U tom slučaju svojstvo generalizacije se može promatrati ne kao neko mistično svojstvo mreže nego kao svojstvo dobre nelinearne interpolacije
- Neuroni s kontinuiranim aktivacijskim funkcijama daju i sveukupno kontinuirano preslikavanje ulaz-izlaz

Generalizacija



Generalizacija

- Dobra generalizacija je interpolacija raspoloživih podataka (znanja) s najjednostavnijom krivuljom
- Najjednostavnija krivulja je ona koja je najviše glatka tj. nema velikih varijacija
- Loša generalizacija može biti posljedica prevelikog broja primjera za učenje (engl. overfitting)
- Preveliki broj primjera za učenje je kad ima više primjera nego što je broj stupnjeva slobode samog procesa iz kojeg se generiraju primjeri
- U tom slučaju potreban je i veći broj skrivenih neurona nego što je neophodno

Kros-validacija

- engl. cross-validation
- Kros-validacija je metoda koja se koristi i u statistici
- Raspoloživi skup podataka podijeli se u skup za učenje i skup za testiranje
- Skup za učenje podijeli se dalje u skup za estimaciju modela (tj. treniranje mreže) i skup za evaluaciju modela (validaciju)
- Skup za validaciju iznosi obično 10-20 % od skupa za treniranje

Kros-validacija

- Cilj je ove procedure da se validacija modela radi na podacima različitim od onih s kojim se mreža trenirala
- Nakon što se izabere najbolji model za treniranje se koristi cijeli skup za treniranje
- Mreža se konačno testira pomoću skupa primjera za testiranje

Svojstva BP algoritma

- BP algoritam je najpopularniji algoritam za učenje s nadzorom kod višeslojnih perceptrona
- BP algoritam je gradijentna tehnika koja ima slijedeća dva glavna svojstva:

  1. Jednostavan je i radi lokalno

  2. Izvodi stohastički najbrži spust u prostoru težina

Primjene

- Višeslojni perceptron je korišten u mnogim primjenama:
  - NETtalk: NN koje uče izgovarati engleski tekst
  - Prepoznavanje govora
  - OCR (prepoznavanje slova)
  - Identifikacija sustava
  - Automatsko upravljanje
  - Upravljanje autonomnim vozilom
  - Detekcija i klasifikacija radarskih i sonarskih signala
  - Medicinska dijagnostika srčanih bolesti
  - Analiza signala i slika

Zadaci

- Problem 6.1:
- Pokazati da mreža na slici rješava XOR problem

  a) nacrtati granice odlučivanja (decision regions)

  b) napisati tablicu istinitosti za mrežu