Алгоритам учења перцептрона

Корак 1: Иницијализација. Нека је $t=0$ и нека вектор иницијалних тежинских односа $\mathbf{w}(t) \in \Re^{a}$ дефинише иницијално пресликавање од скривених неурона до излазног неурона.

Корак 2: Изабрати стимулацију (подстреке) перцептрона примењујући узорке из обучавајућег скупа x^k на случајан начин, што се може исказати помоћу $x^k(t)=(s^k(t),\mathbf{o}^k(t)) \in \{0,1\}$, где је $\mathbf{o}^k$ циљни вектор, који представља захтевани одговор на стимулацију.

Корак 3: Израчунати активацију неурона у скривеном слоју h^k, на основу активације улазних неурона, $h^k(t)= q(s^k(t))$.

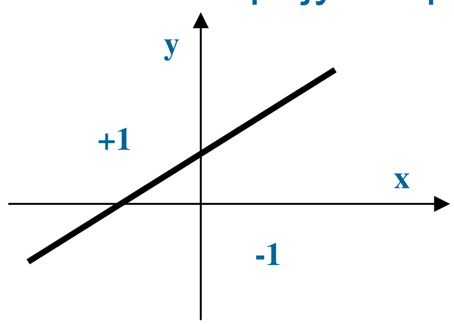
Корак 4: Израчунати активацију излазног неурона мреже r^k, на основу активације неурона из скривеног слоја, $r^k(t)= f(net)= f(\mathbf{w}(t)^T, h^k(t))$, при чему је $f(net)=1$ ако је $net \geq 0$ и $f(net)=0$ ако је $net < 0$.

Корак 5: Модификовати тежинске односе између скривених неурона и излазног неурона, $\mathbf{w}(t+1)= \mathbf{w}(t)+[\mathbf{o}^k(t)- r^k(t)]h^k(t)$.

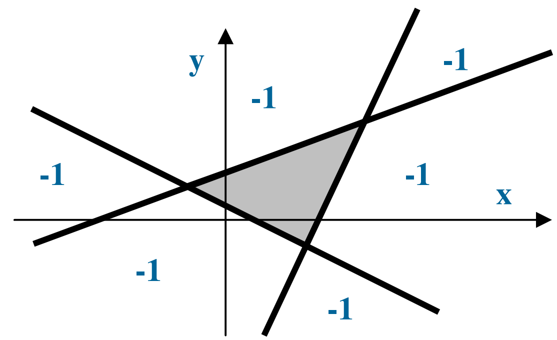
Корак 6: Зауставити учење ако је $t > t_{max}$ или $\mathbf{o}^k(t)= r^k(t)$, за $k=1,...,n$, иначе нека је $t=t+1$ и тада је неопходно вратити се на корак 1.

Детаљна анализа перцептрона у погледу могућности и ограничења примене – *Marvin Minsky* и *Seymour Papert* (1969)

- Главна карактеристика: линеарна дискриминациона функција у простору узорака у коначном броју итерација;



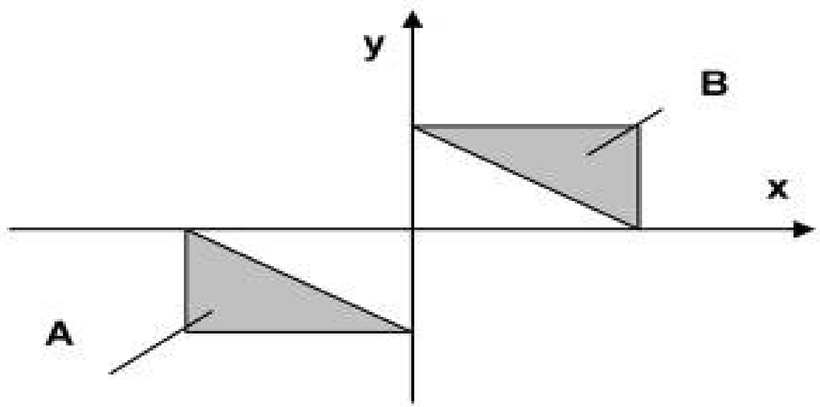
a)



b)

- Коректна класификација: само за линеарно раздвајајуће класе узорака;
- а), само један процесирајући елемент перцептрона је потребан, с обзиром да је цела површина раздељена једном правом линијом на два одвојена региона, који су идентификовани са **+1** и **-1**;
- б), регион формиран од много правих линија => неопходно развити више перцептрона за овај проблем.

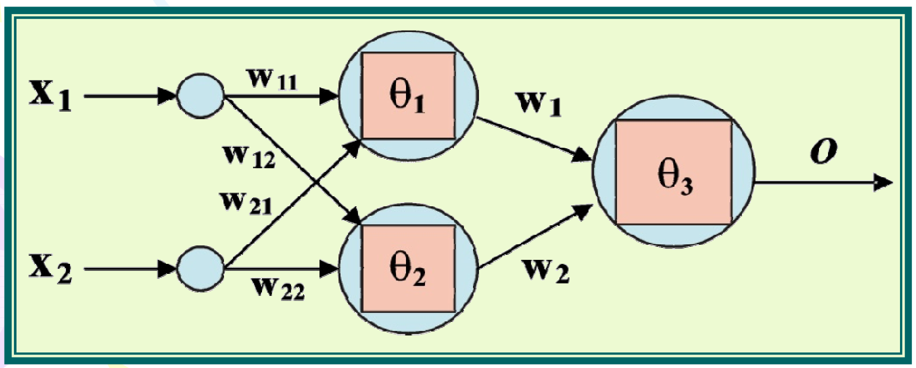
Реализација логичког *ИЛИ*



$w_{11}=1,\ w_{12}=1,\ w_{21}=1,\ w_{22}=1,$
$w_1=1,\ w_2=-1,$
$\theta_1=0.5,\ \theta_2=1.5,\ \theta_3=0.5.$

Logičko ***ILI***

Tačka 1(x_1)	Tačka 2(x_2)	Izlaz
U reg. A (0)	*U reg. B (0)*	*0*
U reg. A (0)	*Van reg. B (1)*	*1*
Van reg. A (1)	*U reg. B (0)*	*1*
Van reg. A (1)	*Van reg. B (1)*	*0*



$w_{11}=1,\ w_{12}=1,\ w_{21}=1,\ w_{22}=1,$

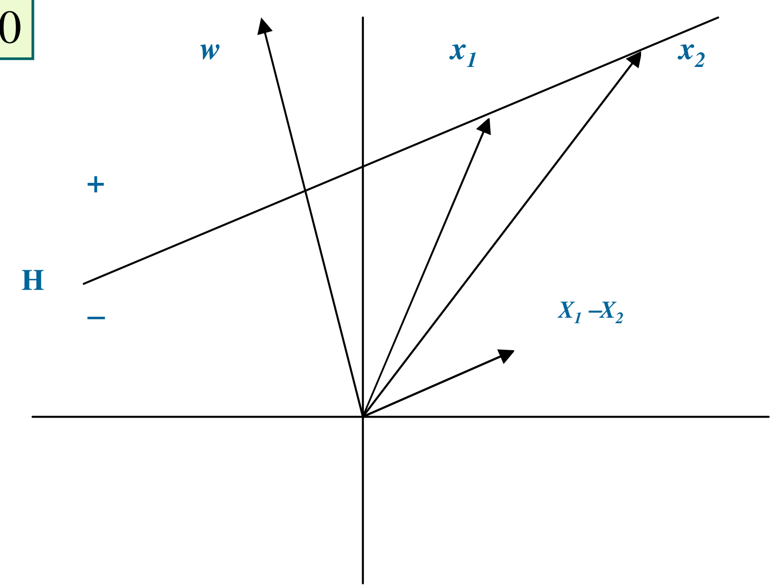
$w_1=1,\ w_2=-1,$

$\theta_1=0.5,\ \theta_2=1.5,\ \theta_3=0.5.$

Вишеслојни перцептрони

- Проширују могућности класификације на шире скупове;
- Уводе се појмови *хиперраван* и *хиперпростор*, тако да су *хиперравни*, за *n*-димензионални простор (*хиперпростор*), одређене *n*-1 димензијом;
- У *n*-димензионалном улазном простору *X*, хиперраван *H* која врши сепарацију, дефинисана је на следећи начин:

$$\boxed{w_1 x_1 + w_2 x_2 + \cdots + w_n x_n = \theta = -w_0}$$



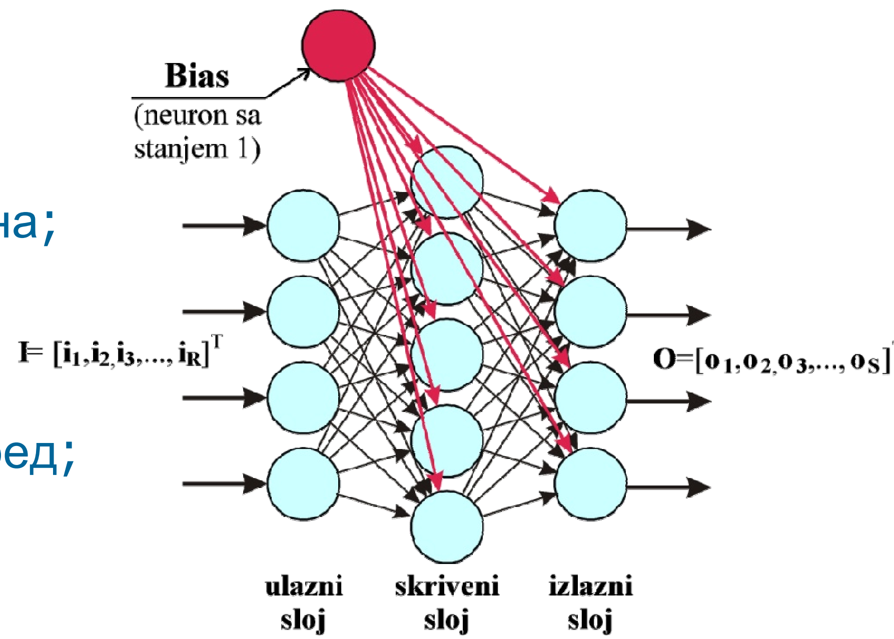
- Вектор тежинских односа

w = $[w_1, w_2, \cdots, w_n]^T$ управан на хиперраван;

- Као линеарни сепаратор, перцептрон има велика ограничења у погледу нелинеарних пресликавања (бинарна активациона функција).

„Backpropagation" (BP) неуронска мрежа

- Развијена са циљем да се реши проблем нелинеарног пресликавања из улазног простора у излазни простор;
- Остварује се и модификација тежинских односа и између улазног и скривеног слоја неурона;



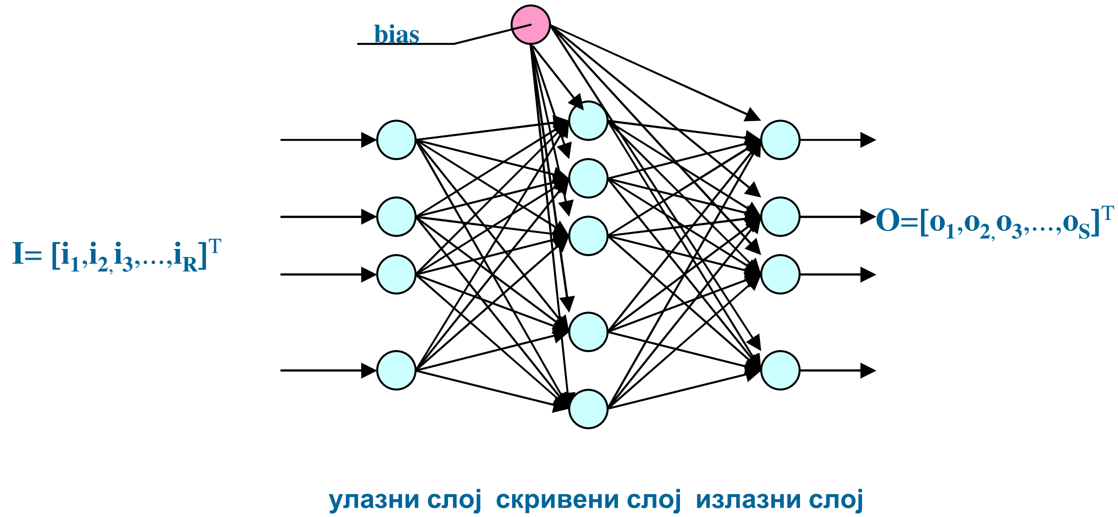
- Као и перцептрон, BP неуронска мрежа је мрежа са простирањем сигнала унапред;
- Реализује супервизорски вид учења;
- Користи *градијентни поступак* при обучавању;
- Учи пресликавања из улазног простора узорка у излазни простор, кроз *процес минимизације грешке* између актуелног излаза и захтеваног излаза;

„Backpropagation" (BP) неуронска мрежа

- Процес учења почиње са презентацијом улазног облика узорка BP мрежи;
- Простирањем кроз мрежу улазног облика узорка остварује се излазни облик;
- Затим се примењује *генерализано делта правило* да би се утврдила грешка на излазу, коју простирањем уназад преко скривеног слоја, користи за „лагано" модификовање сваког тежинског односа између неурона;
- Поступак се понавља за сваки нови узорак;
- Систем BP неуронске мреже користи сигмоидну активациону функцију;
- Применом *генерализаног делта правила* утврђена грешка се коригује, тако што се користи први извод сигмоидне функције $f'(net) = f'(S)$. Уколико сигмоидна функција $f(net)$ брже расте, онда треба извршити већу корекцију грешке и обрнуто.

Архитектура BP мреже

- Елементарна архитектура BP мреже има три слоја;
- Слојеви потпуно повезани;
- Користи се и структура са више од једног скривеног слоја.



- Број неурона у сваком слоју зависи од области примене, као и број скривених слојева;
- Најчешће постоји само један скривени слој, јер је BP мрежа са таквом структуром у стању да обезбеди репродуковање скупа захтеваних излазних облика за све обучавајуће парове;
- Екстра улаз - неурон је увек активан и има излазно стање 1 „*bias*” неурон;
- Понаша се у мрежи као и сваки други неурон, и има задатак да партиципира у процесу учења побољшава конвергентне карактеристике BP мреже.

Генералисано делта правило

- Применом ***генералисаног делта правило*** остварују се основне карактеристике мреже **генерализација** и **нелинеарно раздвајање;**

- *M* број обучавајућих парова, *R* број неурона у улазном слоју и *C* број неурона у излазном слоју;

- Обучавајући парови $(x_1,y_1)\ldots(x_M,y_M)$, који су доведени на улаз мреже, обезбеђују мрежи могућност да реализује нелинеарно пресликавање између x_i и y_i вектора, за $i=1,\ldots,M$.

- *L* број слојева мреже и нека је са *l* означен сваки слој:

  $l = 0$ улазни слој,

  $l = 1,\ldots,L-1$ скривени слој(еви),

  $l = L$ излазни слој.

Генералисано делта правило

- Неуронска мрежа има K_l неурона у сваком слоју (где је $l = 1,\ldots,L$),
- Улази у k-ти неурон l-тог слоја су дати преко, за $j=1,\ldots,K_{l-1}$ (истовремено и излази неурона претходног слоја), који се могу означити са O_{ij}^{l-1} ;
- Сумирана улазна вредност за k-ти неурон у слоју l је:

$$\boxed{S_{ik}^{l} = net_{ik}^{l} = \sum_{j=1}^{K_{l-1}} w_{kj}^{l} I_{ij}^{l} + \theta_{k}^{l}}$$

- Излаз из k-тог неурона у l-*том* слоју је остварен као резултат активације неурона;
- Преко функције преноса пропушта се улаз дат преко израза:

$$\boxed{O_{ik}^{l} = f_{ik}^{l}\left(S_{ik}^{l}\right)}$$

- Грешка на излазном слоју L, BP неуронске мреже, је представљена разликом између израчунатих и задатих излаза:

$$\boxed{E_{pi} = \frac{1}{2}\sum_{k=1}^{K_L}\left(y_{ik} - O_{ik}^{l}\right)^2}$$

Генералисано делта правило

- Минимизација грешке:

$$E_{pi} = \frac{1}{2}\sum_{k=1}^{K_3}\left(y_{ik} - O_{ik}^3 \right)^2 = \quad \frac{1}{2}\sum_{k=1}^{K_3}\left(y_{ik} - f_k^3\left(S_{ik}^3\right)\right)^2 = \ldots =$$

$$= \frac{1}{2}\sum_{k=1}^{K_3}\left(y_{ik} - f_k^3\left(\sum_{j=1}^{K_2} w_{kj}^3 f_j^2\left(\sum_{u=1}^{K} w_{ju}^2 f_u^1\left(\sum_{v=1}^{K_0} w_{jv}^1 O_{iv}^0 + \boldsymbol{\theta}_u^1\right) + \boldsymbol{\theta}_j^2\right) + \boldsymbol{\theta}_k^3\right)\right)^2$$

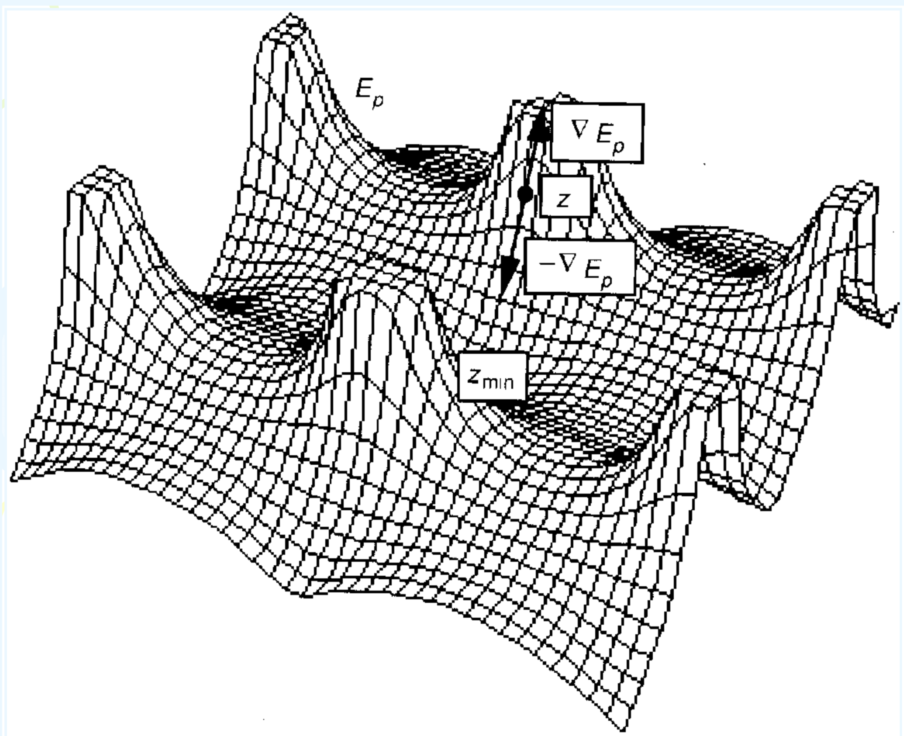
- Градијент грешке E_p, у односу на тежинске односе (за излазни слој (l=3)):

$$\frac{\partial E_{pi}}{\partial w_{kj}^3} = -\left(y_{ik} - O_{ik}^3\right)\frac{\partial O_{ik}^3}{\partial S_{ik}^3}\frac{\partial S_{Iik}^3}{\partial w_{kj}^3} = -\left(y_{ik} - O_{ik}^3\right) f_k^{3\prime}\left(S_{ik}^3\right) I_{ij}^3$$

- Негативни предзнак указује на то да је правац промене тежинских односа са негативним градијентом ∇Ep.

Генералисано делта правило

Правац простирања градијента грешке E_p.



Тежински односи се итеративно мењају док грешка *Ep* не оствари минумум у тачки z_{min}.

- Инкремент промене тежинских односа, за задати параметар учења η, је:

$$\Delta_i w_{kj}^3 = \boldsymbol{\eta}\left(y_{ik} - O_{ik}^3\right) f_k^{3'}\left(S_{ik}^3\right) I_{ij}^3 = \boldsymbol{\eta}\, \boldsymbol{\delta}_{ik}^3 I_{ij}^3$$

$$\boldsymbol{\delta}_{ik}^3 = \left(y_{ik} - O_{ik}^3\right) f_k^{3'}\left(S_{ik}^3\right)$$

- *Параметар учења* η се најчешће бира у опсегу од 0.25 (спорије обучавање) до 0.75 (брже, али нестабилније);

- Метода *променљивог параметра*:у почетку обучавања параметар η већи, током обучавања η *се* смањује, омогућавајући тако мрежи боље карактеристике учења.

Генералисано делта правило

- Други скривени слој (*l*=2):

$$\frac{\partial E_{pi}}{\partial w_{ju}^{2}} = -\sum_{k=1}^{K_3}\left(y_{ik} - O_{ik}^{3}\right)\frac{\partial O_{ik}^{3}}{\partial S_{ik}^{3}}\frac{\partial S_{Iik}^{3}}{\partial O_{ij}^{2}}\frac{\partial O_{ij}^{2}}{\partial S_{ij}^{2}}\frac{\partial S_{Iij}^{2}}{\partial w_{ju}^{2}} = -\sum_{k=1}^{K_3}\left(y_{ik} - O_{ik}^{3}\right)f_k^{3\,\prime}\left(S_{ik}^{3}\right)w_{kj}^{3} f_j^{2\,\prime}\left(S_{ij}^{2}\right)I_{iu}^{2}$$

$$\Delta_i w_{ju}^{2} = \eta\,\delta_{ij}^{2} I_{ij}^{2} \qquad \delta_{ij}^{2} = f_j^{2\,\prime}\left(S_{ij}^{2}\right)\sum_{k=1}^{K_3}\delta_{ik}^{3} w_{kj}^{3}$$

- Први скривени слој (*l*=1):

$$\frac{\partial E_{pi}}{\partial w_{uv}} = -\sum_{k=1}^{K_3}\left(y_{ik} - O_{ik}^{3}\right)\frac{\partial O_{ik}^{3}}{\partial S_{ik}^{3}}\frac{\partial S_{Iik}^{3}}{\partial O_{ij}^{2}}\frac{\partial O_{ij}^{2}}{\partial S_{ij}^{2}}\frac{\partial S_{ij}^{2}}{\partial O_{iu}}\frac{\partial O_{iu}}{\partial S_{iu}}\frac{\partial S_{iu}}{\partial w_{uv}}$$

$$= -\sum_{k=1}^{K_3}\left(y_{ik} - O_{ik}^{3}\right)f_k^{3\,\prime}\left(S_{ik}^{3}\right)w_{kj}^{3}\sum_{j=1}^{K_2} f_j^{2\,\prime}\left(S_{ij}^{2}\right) w_{ju}^{2}\, f_u^{1\,\prime}\left(S_{iu}^{1}\right)x_{iv}$$

$$\Delta_i w_{uv}^{1} = \eta\,\delta_{iu}^{1} x_{iv} \qquad \delta_{iu}^{1} = f_u^{1\,\prime}\left(S_{iu}^{1}\right)\sum_{j=1}^{K_2}\delta_{ij}^{2} w_{ju}^{2}$$

- На основу описаног генералисаног делта правила, алгоритам учења BP мреже може се представити преко седам корака, датих у наставку.

Алгоритам учења BP неуронске мреже

Algoritam učenja BP neuronske mreže

Korak 1: Inicijalizacija i filtriranje. Dovesti ulazni vektor $\mathbf{I}_{ij} = (x_{i1}, x_{i2}, \ldots, x_{iR})$ do neurona u ulaznom sloju.

Korak 2: Izračunati izlazne vrednosti za svaki neuron u mreži korišćenjem izraza:

$$O_{ik}^{l} = f_{ik}^{l}\left(S_{ik}^{l}\right), \quad \text{gde je} \quad S_{ik}^{l} = net_{ik}^{l} = \sum_{j=1}^{K_l} w_{kj}^{l} I_{ij}^{l} + \boldsymbol{\theta}_{k}^{l} \;.$$

Korak 3: Izračunati grešku koju je mreža na izlazu generisala $E_{pi} = \frac{1}{2}\sum_{k=1}^{K_L}\left(y_{ik} - O_{ik}^{L}\right)^2$.

Korak 4: Za izlazni sloj, $l = L$, izračunati vrednosti promene težinskih odnosa pomoću izraza: $\Delta_i w_{kj}^{L} = \boldsymbol{\eta}\left(y_{ik} - O_{ik}^{L}\right) f_{k}^{L'}\left(S_{ik}^{L}\right) I_{ij}^{L} = \boldsymbol{\eta}\, \boldsymbol{\delta}_{ik}^{L} I_{ij}^{L}$

Korak 5: Za skrivene slojeve, $l = 1,\ldots,L-1$, izračunati vrednosti promene težinskih odnosa

$$\Delta_i w_{ju}^{l} = \boldsymbol{\eta}\, \boldsymbol{\delta}_{ij}^{l} I_{ij}^{l}, \quad \text{gde je} \quad \boldsymbol{\delta}_{ij}^{l} = f_{j}^{l'}\left(S_{ij}^{l}\right) \sum_{k=1}^{K_{l+1}} \boldsymbol{\delta}_{ik}^{l+1} w_{kj}^{l+1}$$

Korak 6: Ažurirati (modifikovati) sve vrednosti inkrementalne promene težinskih odnosa $w_{ju}^{l} = w_{ju}^{l} + \Delta_i w_{ju}^{l}$, za $l = 1,\ldots,L$.

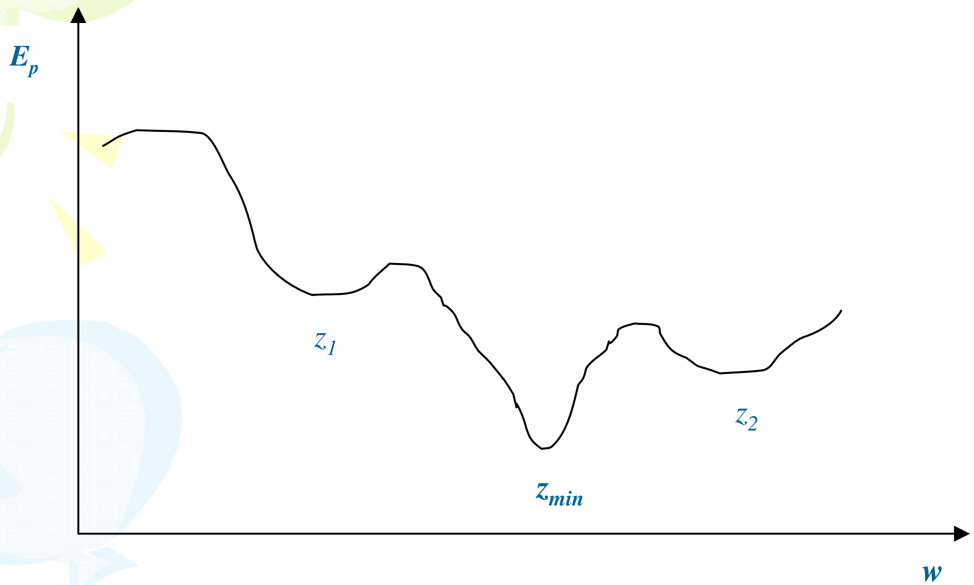
Korak7: Ponoviti sve prethodne korake dok greška koju generiše mreža ne bude bliska nuli ili dovoljno mala.

Napomena: Treba naglasiti da je greška, vezana za neurone u skrivenim slojevima, izračunata pre nego što su njihovi težinski odnosi sa neuronima u izlaznom sloju ažurirani (modifikovani).

Коришћене сигмоидне функције

Neuron	Sigmoidna aktivaciona funkcija	Prvi izvod aktivacione funkcije
5	$f(net) = f(S) = f(x) = \left(1 + e^{-3x}\right)^{-1}$	$f'(net) = f'(S) = f'(x) = \dfrac{3e^{-3x}}{\left(1 + e^{-3x}\right)^2}$
6	$f(net) = \left(1 + e^{-4x}\right)^{-1}$	$f'(net) = \dfrac{4e^{-4x}}{\left(1 + e^{-4x}\right)^2}$
7	$f(net) = \left(1 + e^{-5x}\right)^{-1}$	$f'(net) = \dfrac{5e^{-5x}}{\left(1 + e^{-5x}\right)^2}$
8	$f(net) = \left(1 + e^{-6x}\right)^{-1}$	$f'(net) = \dfrac{6e^{-6x}}{\left(1 + e^{-6x}\right)^2}$
9	$f(net) = \left(1 + e^{-7x}\right)^{-1}$	$f'(net) = \dfrac{7e^{-7x}}{\left(1 + e^{-7x}\right)^2}$
10	$f(net) = \left(1 + e^{-8x}\right)^{-1}$	$f'(net) = \dfrac{8e^{-8x}}{\left(1 + e^{-8x}\right)^2}$

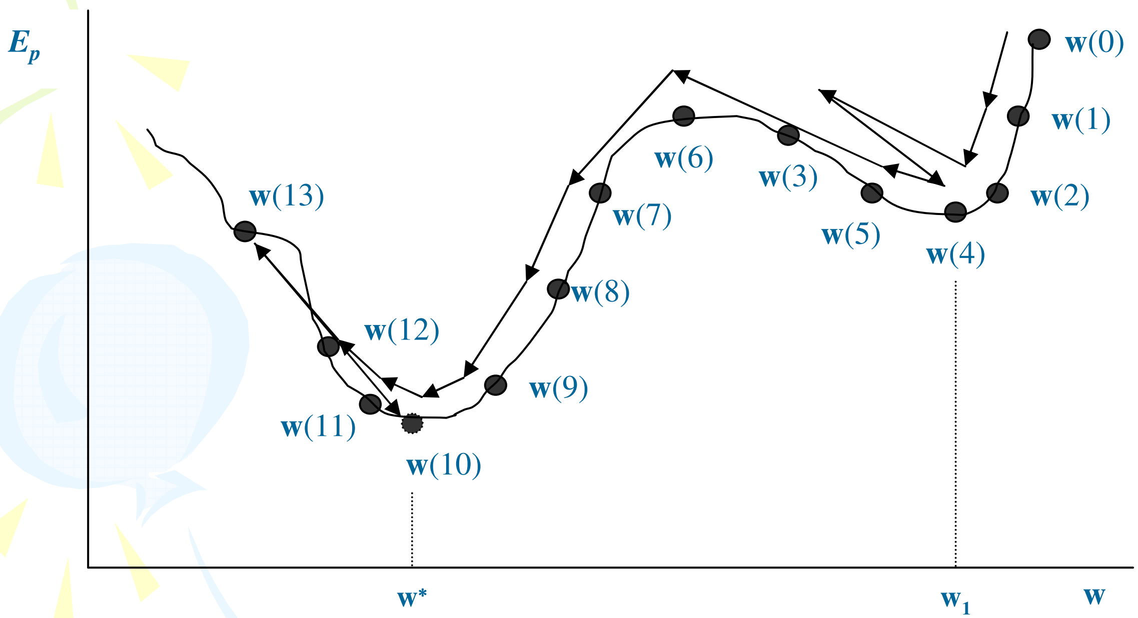
Глобални и локални минимуми



- Попречни пресек хипотетичке комплексне површи грешке E_p у простору тежинских односа;
- Тачка z_{min} је **глобални минимум**. Као што се види, на слици има и других тачака, z_1 и z_2, које представљају **локалне минимуме**;
- Градијентни поступак минимизације грешке E_p има за циљ да се уместо ових локалних минимума пронађе глобални минимум.

• Понекад се обучавање BP неуронске мреже зауставља у тачки која се налази негде на правцу простирања градијента ∇Ep, пре него што је стварни минимум остварен, јер је та вредност грешке *Ep* задовољавајућа. То управо говори о комплексности проблема конвергенције.

Стохастички алгоритам тражења глобалног минимума



- Корисна хеуристичка техника напуштања тренутног локалног минимума и конвергирања ка „дубљим” локалним минимумима;
- Иницијализација алгоритма је у тачки **w**(0);

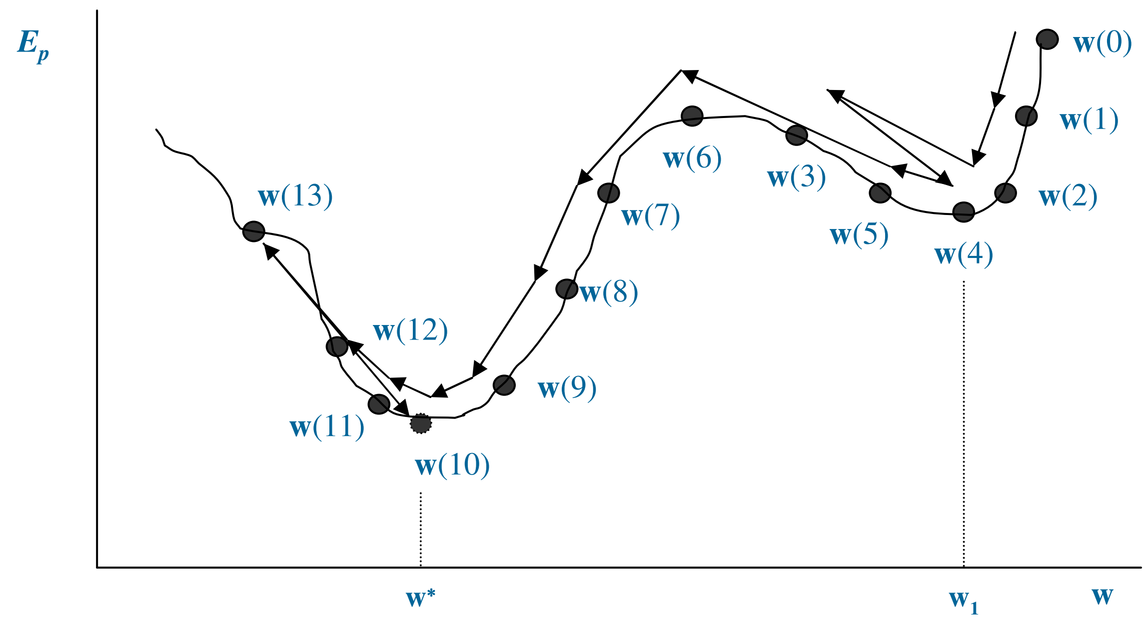
- Трајекторија конвергенције се креће ка стриктном локалном минимуму који се налази у $\mathbf{w}_1$;

Након дивергенције до тачке **w**(3), алгоритам враћа трајекторију конвергенције у **w**(4) уз тренутно задржавање у близини стриктног локалног минимума **w**1;

- Следећа итерација: кретање ка тачки **w**(6);
- Обезбеђује секвенцијално конвергирање ка глобалном минимуму у **w**(10);

Стохастички алгоритам тражења глобалног минимума

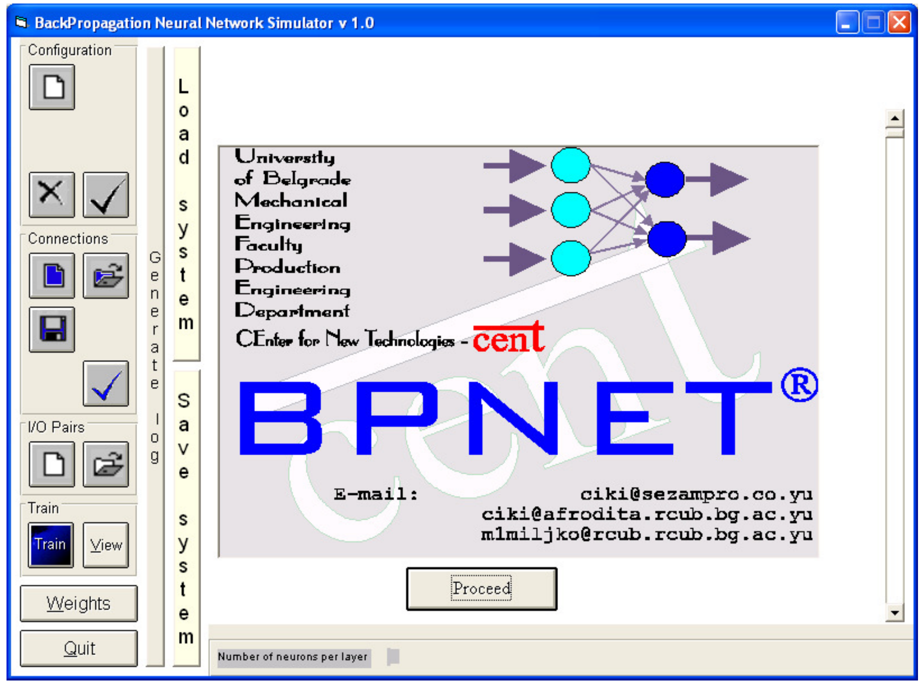
- „Бегом” из стриктног глобалног минимума, преко тачака **w**(11) и **w**(12) (уз евентуални одлазак и у тачку **w**(13)), алгоритам остварује конвергенцију у стриктном глобалном минимуму **w**∗;



- У практичној примени алгоритам спорије конвергира ка глобалном минимуму;
- Сваки следећи пут процес је знатно бржи!
- „Наградно” питање: Да ли је брзина конвергенције важна за учење неуронске мреже? → Одговор: Да, за већину проблема везаних за процес одлучивања, посебно када су роботи у питању!

Софтвер за симулирање вештачких неуронских мрежа - *BPnet*

- ***У BPnet-у*** је спроведена стохастичка конвергенција;
- Реализовано је рескалирање вредности улазних и излазних неурона као и њихово нормирање, при чему су те вредности сведене на распон од 0.1 до 0.9, због успешне и брже стохастичке конвергенције;
- Када је конвергенција успешно завршена, вредности се конвертују, рескалирањем у супротном смеру у стварне вредности (нпр. углова ротације за зглобове робота, З.Миљковић-књига);
- Конвергенција грешке *Ep* се прати и преко неколико контрола софтвера ***BPnet*** („*min. saved error*”);
- Важну улогу у процесу конвергенције грешке *Ep* има и параметар учења η;



- Мора да буде довољно мали како у току тражења глобалног минимума или „дубљег” локалног минимума не би дошло до тога да мрежа веома „тешко” и споро пронађе уопште било какав прихватљив минимум грешке *Ep*;

- За конкретан проблем оптимална вредност параметра учења η је била 0.2.

Хвала на пажњи!

Питања?