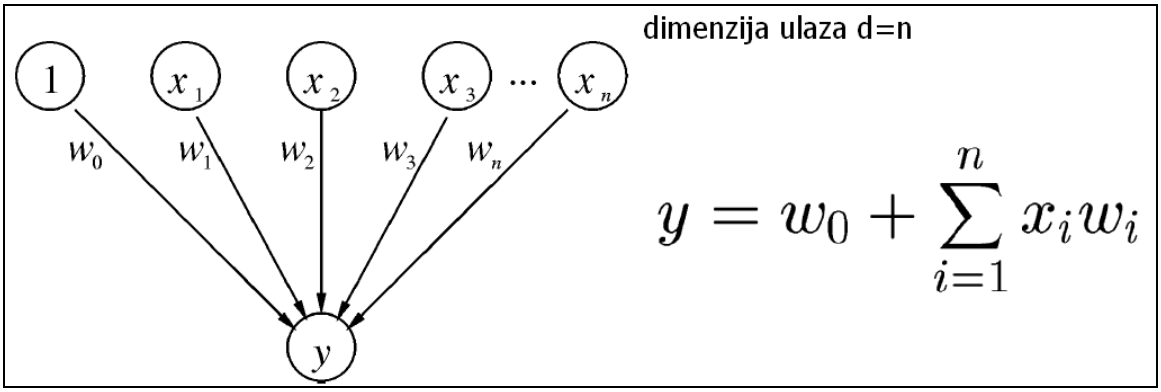
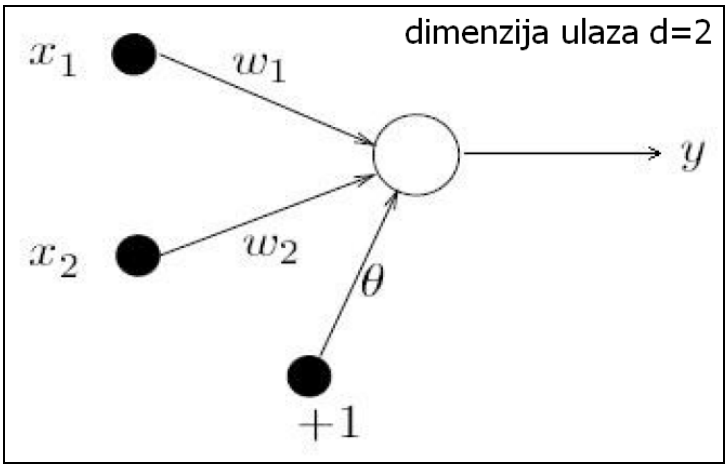
Perceptron

Perceptron je feed-forward neuronska mreža sa jednim slojem:



Slika 1 – Perceptron sa n ulaza

Ponekad se ulaz koji odgovara grani označenoj sa w_0 označava sa -1 umjesto sa 1. Brojevi $x_1, x_2, \ldots, x_n$ su ulazi mreže a brojevi $w_1, w_2, \ldots, w_n$ su težinski koeficijenti ili težine.



Slika 2 – perceptron sa 2 ulaza

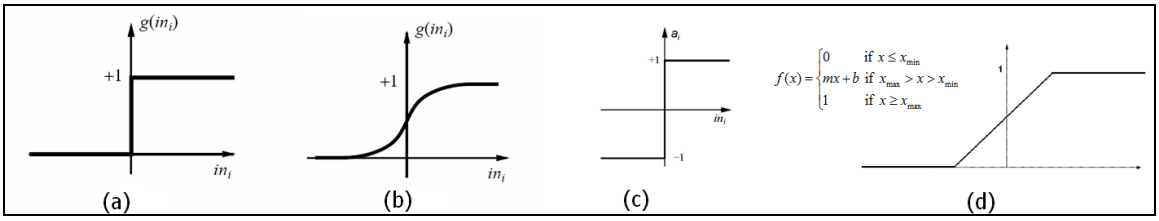
Ako sa w označimo matricu-kolonu $w = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{pmatrix}$ a sa x vektor-vrstu $x = (x_1, x_2, \ldots, x_n)$,

tada se izraz $y = w_0 + \sum_{i=1}^{n} w_i x_i$ može zapisati u vektorskom obliku kao $y = w_0 + w^T \cdot x$

ili $y = w_0 + < w^T, x >$, gdje simbol T označava transponovanje, a simboli <> ili $\cdot$ označavaju skalarni proizvod. Vektor *w* je vektor težina a *x* je ulazni vektor.

Ponekad se umjesto datih vektora *w* i *x* posmatraju sljedeći vektori: $w = \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{pmatrix}$ i $x = (1, x_1, x_2, \ldots, x_n)$, pa se tada dobija formula $y = < w^T, x >$. Ova formula jeste jednačina hiperravni u linearnom prostoru R^n (dimenzija prostora je d=n).

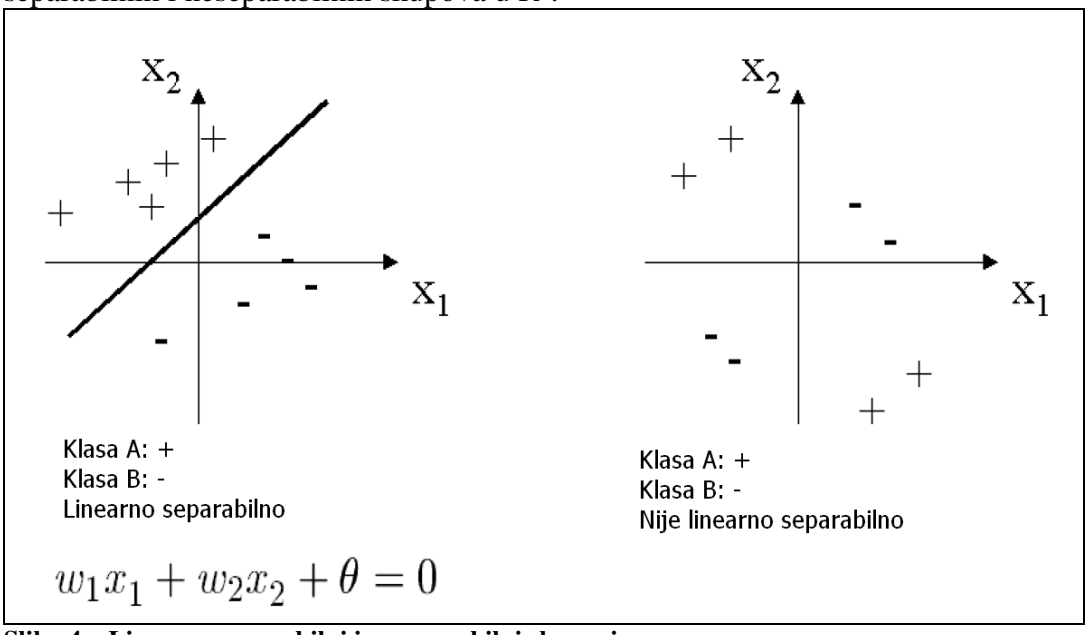
Često se umjesto direktnog računanja vrijednosti y prvo izračuna vrijednost $s = w_0 + \sum_{i=1}^{n} w_i x_i = w_0 + < w^T, x >$, pa se zatim izlaz neurona dobija formulom $y = f(s)$, gdje je *f* aktivaciona funkcija neurona, koja je obično ili threshold funkcija (slika 3 (a)) ili sigmoidna funkcija (slika 3 (b)) ili signum (slika 3 (c)) ili dio po dio linearna (slika 3 (d)).



Slika 3 – Primjeri aktivacionih funkcija

U nekim udžbenicima pod perceptronom se podrazumijeva samo ona mreža čija je aktivaciona funkcija signum.

Perceptron može klasifikovati linearno separabilne skupove, Na slici 4 dat je primjer separabilnih i neseparabilnih skupova u R^2.

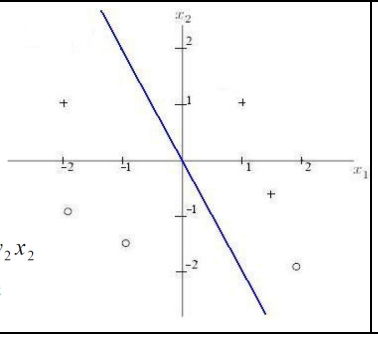
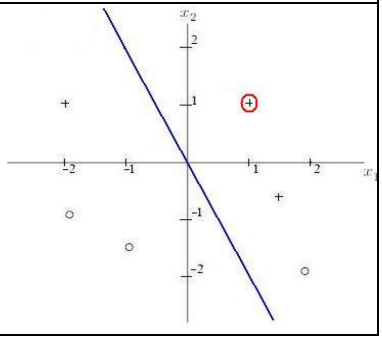
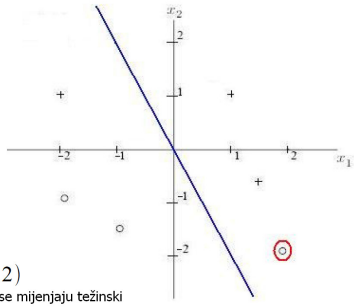
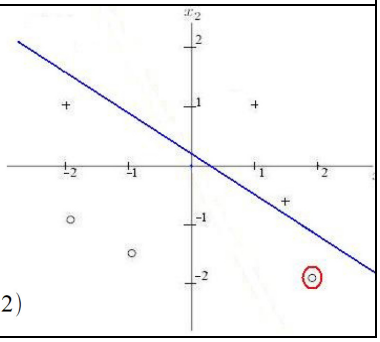
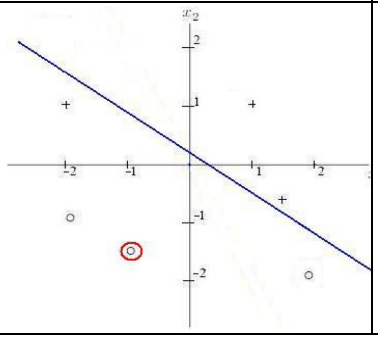
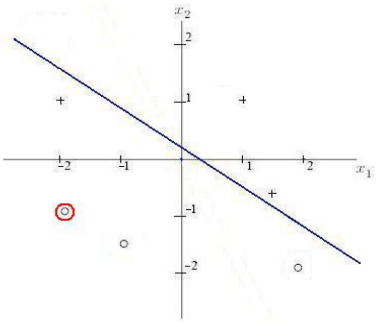


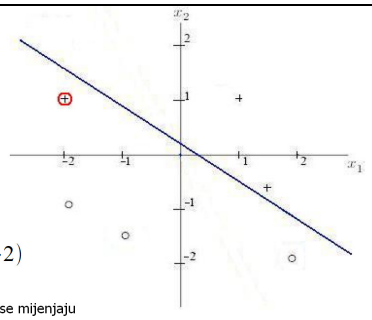
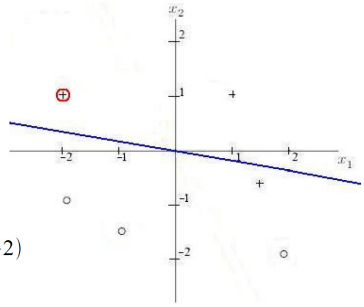
Slika 4 – Linearno separabilni i neseparabilni skupovi

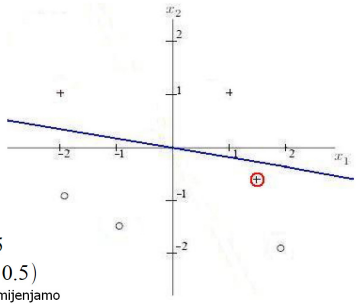
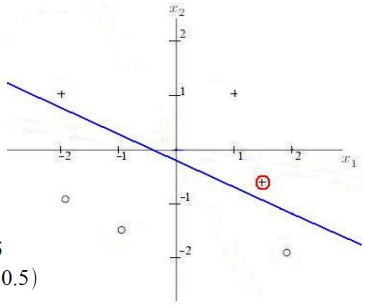
Algoritam obučavanja perceptrona:

1. Date su dvije klase A i B. Označimo sa η mali pozitivni realni broj – koeficijent brzine obučavanja (learning rate). Postavimo slučajne vrijednosti za težinske koeficijente. Uvedimo funkciju $d(x) = \begin{cases} +1, \text{ako je x u klasi A} \\ -1, \text{ako je x u klasi B} \end{cases}$.
2. Slučajno izaberimo jedan primjer $x = (x_1, x_2, \ldots, x_n)$ iz ulaznog skupa podataka
3. Ako je klasifikacija dobra, ne radimo ništa.
4. Ako je klasifikacija pogrešna, modificiramo vektor težina w:
   $w_i = w_i + \eta \cdot d(x) \cdot x_i$
5. Ponavljamo korake 2-4 sve dok ima pogrešno klasifikovanih primjera.

Primjer: Klasa A predstavljena je simbolom +, dok je klasa B predstavljena simbolom -. Na sljedećim slikama prikazani su koraci u radu algoritma:

<table>
<tr><th>Početne vrijednosti</th><th>Prvi primjer</th></tr>
<tr><td>Početne vrijednosti
$\eta = 0.2$
$w = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0.5 \end{pmatrix}$
$0 = w_0 + w_1 x_1 + w_2 x_2$
$= 0 + x_1 + 0.5 x_2$
$\Rightarrow \; x_2 = -2x_1$
  </td><td>$\eta = 0.2$
$w = \begin{pmatrix} 0 \\ 1 \\ 0.5 \end{pmatrix}$
$x_1 = 1, x_2 = 1$
$w^T x > 0$
Klasifikacija dobra
ne radimo ništa
  </td></tr>
<tr><th>Drugi primjer</th><th>Poslije promjene težina</th></tr>
<tr><td>$\eta = 0.2$
$w = \begin{pmatrix} 0 \\ 1 \\ 0.5 \end{pmatrix}$
$x_1 = 2, x_2 = -2$
$w_0 = w_0 - 0.2 * 1$
$w_1 = w_1 - 0.2 * 2$
$w_2 = w_2 - 0.2 * (-2)$
Klasifikacija nije dobra, pa se mijenjaju težinski koeficijenti
  </td><td>$\eta = 0.2$
$w = \begin{pmatrix} -0.2 \\ 0.6 \\ 0.9 \end{pmatrix}$
$x_1 = 2, x_2 = -2$
$w_0 = w_0 - 0.2 * 1$
$w_1 = w_1 - 0.2 * 2$
$w_2 = w_2 - 0.2 * (-2)$
  </td></tr>
<tr><th>Treći primjer</th><th>Četvrti primjer</th></tr>
<tr><td>$\eta = 0.2$
$w = \begin{pmatrix} -0.2 \\ 0.6 \\ 0.9 \end{pmatrix}$
$x_1 = -1, x_2 = -1.5$
$w^T x < 0$
Klasifikacija dobra
ne radimo ništa
  </td><td>$\eta = 0.2$
$w = \begin{pmatrix} -0.2 \\ 0.6 \\ 0.9 \end{pmatrix}$
$x_1 = -2, x_2 = -1$
$w^T x < 0$
Klasifikacija dobra
ne radimo ništa
  </td></tr>
</table>

Peti primjer	Poslije promjene težina
$\eta = 0.2$ $w = \begin{pmatrix} -0.2 \\ 0.6 \\ 0.9 \end{pmatrix}$ $x_1 = -2,\ x_2 = 1$ $w_0 = w_0 + 0.2 * 1$ $w_1 = w_1 + 0.2 * (-2)$ $w_2 = w_2 + 0.2 * 1$ Klasifikacija nije dobra, pa se mijenjaju težinski koeficijenti 	$\eta = 0.2$ $w = \begin{pmatrix} 0 \\ 0.2 \\ 1.1 \end{pmatrix}$ $x_1 = -2,\ x_2 = 1$ $w_0 = w_0 + 0.2 * 1$ $w_1 = w_1 + 0.2 * (-2)$ $w_2 = w_2 + 0.2 * 1$ 

Šesti primjer	Poslije promjene težina
$\eta = 0.2$ $w = \begin{pmatrix} 0 \\ 0.2 \\ 1.1 \end{pmatrix}$ $x_1 = 1.5,\ x_2 = -0.5$ $w_0 = w_0 + 0.2 * 1$ $w_1 = w_1 + 0.2 * 1.5$ $w_2 = w_2 + 0.2 * (-0.5)$ Klasifikacija nije dobra pa mijenjamo težinske koeficijente 	$\eta = 0.2$ $w = \begin{pmatrix} 0.2 \\ 0.5 \\ 1 \end{pmatrix}$ $x_1 = 1.5,\ x_2 = -0.5$ $w_0 = w_0 + 0.2 * 1$ $w_1 = w_1 + 0.2 * 1.5$ $w_2 = w_2 + 0.2 * (-0.5)$ 

Svi primjeri su dobro klasifikovani, a granica klasifikacije je prava
$0.2 + 0.5 \cdot x_1 + 1 \cdot x_2 = 0$

Teorema o konvergenciji perceptrona: Za linearno separabilni skup, dati alogoritam u konačnom broju koraka pronalazi granicu klasifikacije.

Perceptron sa sigmoidnom aktivacionom funkcijom

Izlaz ovakvog perceptrona je $o = \sigma(s) = \frac{1}{1+e^{-s}}, s = \sum_{i=0}^{d} w_i x_i$. Obučavanje je složenije jer nije jasno kako da se mijenjaju težinski koeficjenti. Označimo sa x_e ulazni vektor, sa x_{ie} njegovu i-tu koordinatu, sa (x_e, y_e) ulazni par a sa o_e izlaz mreže za ulaz x_e. Koristi se metod gradijentnog spusta (gradient descent, pogledajte lekciju "Lokalno traženje"): $w_i = w_i - \eta \cdot \frac{\partial E}{\partial w_i}$, gdje je E funkcija greške $E = \frac{1}{2}\sum_e (y_e - o_e)^2$. Tada je:

$$\frac{\partial E}{\partial w_i} = \frac{\partial\left(\frac{1}{2}\sum_e (y_e - o_e)^2\right)}{\partial w_i} = \frac{1}{2}\sum_e \frac{\partial (y_e - o_e)^2}{\partial w_i} = \frac{1}{2}\cdot 2 \cdot \sum_e (y_e - o_e)\frac{\partial (y_e - o_e)}{\partial w_i}$$

$$= \sum_e (y_e - o_e)\frac{\partial (y_e - \sigma(s))}{\partial w_i} = \sum_e (y_e - o_e)\cdot(-1)\cdot\sigma'(s)\cdot\frac{\partial s}{\partial w_i} = -\sum_e (y_e - o_e)\cdot\sigma'(s)\cdot x_{ie}$$

Lako se vidi da je $\sigma'(s) = \sigma(s)\cdot(1-\sigma(s))$, pa je konačna formula za gradijentni spust:
$w_i = w_i + \eta \cdot \sum_e (y_e - o_e)\cdot\sigma(s)\cdot(1-\sigma(s))\cdot x_{ie}$.

Gradient Descent Algoritam za *Sigmoidni perceptron* (Batch learning)

Initialization: Examples $\{(\mathbf{x}_e, y_e)\}_{e=1}^{N}$, initial weights w_i set to small random values, learning rate parameter η small real value (i.e. 0.1)
Repeat
for each training example ($\mathbf{x}_e$, y_e)

- *calculate* the output: $o = \sigma(s) = 1 / (1 + e^{-s})$, where: $s = \sum_{i=0}^{d} w_i x_i$

- if the Perceptron does not respond correctly compute weight *corrections*:

$\Delta w_i = \Delta w_i + \eta (y_e - o_e) \sigma(s)(1 - \sigma(s)) x_{ie}$

update the weights with the accumulated error from all examples

$w_i = w_i + \Delta w_i$ // `Gradient Descent Rule`

until termination condition is satisfied.

Primjer:
Perceptron sa dva ulaza x_1 i x_2, početne težine $w_1 = 0.5$, $w_2 = 0.3$ i $w_0 = -1$. $\eta = 0.1$
Prvi ulazni primjer: $x_1 = 2$, $x_2 = 1$, $y = 0$
Izlaz perceptrona je :

$$o = \sigma(-1 + 2 * 0.5 + 1 * 0.3) = \sigma(0.3) = 0.5744$$

Promjene težina:

$$\Delta w_0 = (0 - 0.5744) * 0.5744 * (1 -0.5744) * 1 = - 0.1404$$
$$\Delta w_1 = (0 - 0.5744) * 0.5744 * (1 -0.5744) * 2 = - 0.2808$$
$$\Delta w_2 = (0 - 0.5744) * 0.5744 * (1 -0.5744) * 1 = - 0.1404$$

Drugi ulazni primjer: $x_1 = 1$, $x_2 = 2$, $y = 1$
Izlaz perceptrona je:

$$o = \sigma(-1 + 1 * 0.5 + 2 * 0.3) = \sigma(0.1) = 0.525$$

Promjene težina:

$$\Delta w_0 = - 0.1404 + (1 - 0.525) * 0.525 * (1 - 0.525) * 1 = -0.0219$$
$$\Delta w_1 = - 0.2808 + (1 - 0.525) * 0.525 * (1 - 0.525) * 1 = -0.1623$$
$$\Delta w_2 = - 0.1404 + (1 - 0.525) * 0.525 * (1 - 0.525) * 2 = 0.0966$$

Ako nema drugih ulaznih primjera, u skladu sa algoritmom dobijamo sljedeće težine:

$$w_0 = - 1 + (-0.0219) = -1.0219$$
$$w_1 = 0.5 + (-0.1623) = 0.3966$$
$$w_2 = 0.3 + 0.0966 = 0.3966$$

Inkrementalni Gradient Descent algoritam za *Sigmoidni perceptron*

Prethodna verzija algoritma je primjer tzv. "batch obučavanja", gdje se promjene koeficijenata obavljaju poslije obrade čitavog ulaznog skupa podataka. Postoji i inkrementalna (ili online) verzija algoritma, gdje se promjene težina obavljaju poslije svakog ulaznog primjera.

Initialization: Examples $\{(\mathbf{x}_e, y_e)\}_{e=1}^{N}$, initial weights w_i set to small random values, learning rate parameter $\eta = 0.1$
Repeat
for each training example ($\mathbf{x}_e$, y_e)

- *calculate* the output: $o = \sigma(s) = 1 / (1 + e^{-s})$, where: $s = \sum_{i=0}^{d} w_i x_i$

- if the Perceptron does not respond correctly *update the weights*:
$w_i = w_i + \eta (y_e - o_e) \sigma(s)(1 - \sigma(s)) x_{ie}$ // `Incremental Gradient Descent Rule`

until termination condition is satisfied.

Za primjere višeslojnog perceptrona, pogledati lekciju Perceptron 2013.