Невронни Мрежи

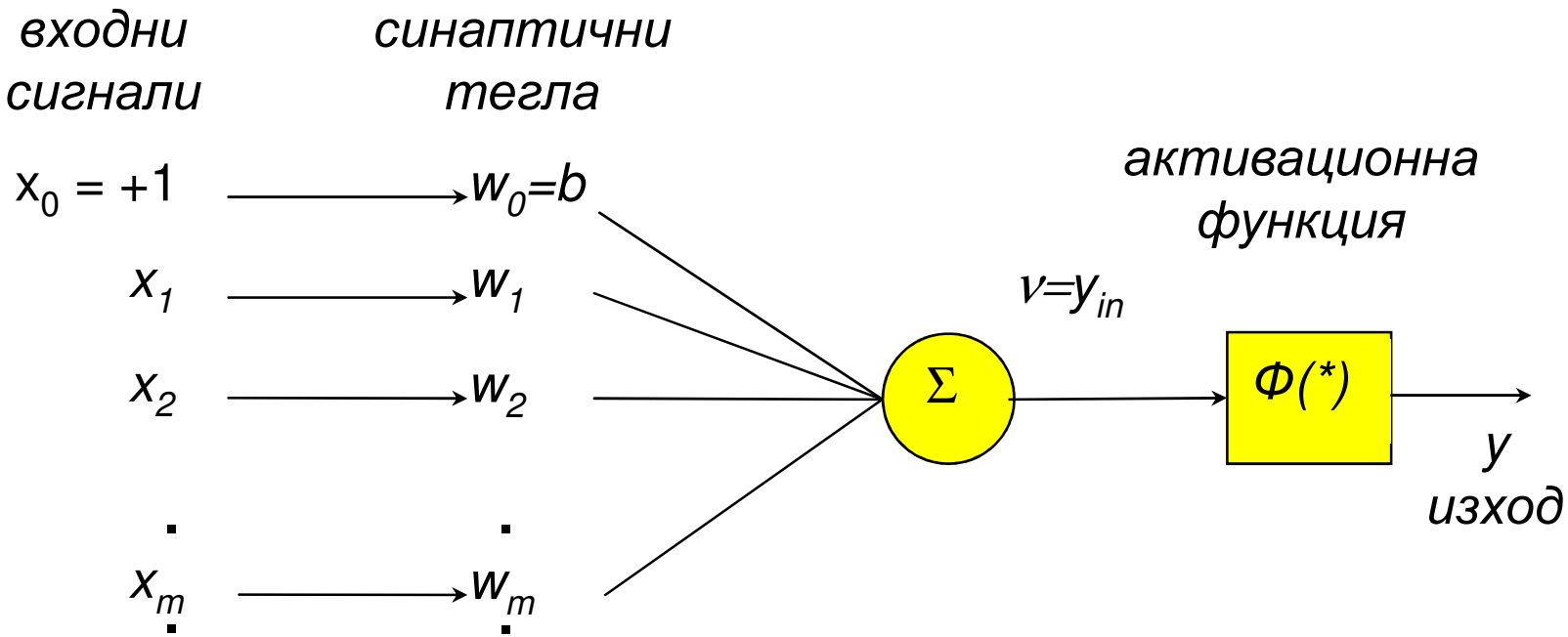
проф. В. Младенов

2. Адаптивен линеен елемент (ADALINE) и делта метод (най-малки квадрати). Перцептрон и правило за обучение на перцептрона. Сходимост на правилото за обучение на перцептрона. Ограничения при използването на перцептрона.

Адаптивен линеен елемент **ADALINE**

- *Адаптивният линеен елемент* – ADAptive LINear Element (ADALINE) е въведен през 1962 г. от Bernard Widrow и представлява единичен неврон, теглата на който се настройват с делта правилото.
- Алгоритъмът на Widrow-Hoff за настройка на теглата се базира на метода на най-малките квадрти и е пример за използване на обучение при настройката.
- Основа за различни видове адаптивна филтрация използвана при обработката на сигнали.

Адаптивен линеен елемент **ADALINE**



Адаптивен линеен елемент ADALINE

$$v = \sum_{j=0}^{m} w_j x_j \Rightarrow v = \vec{x}^T \cdot \vec{w} \qquad y = \phi_{lin}(v) = v$$

Адаптивен линеен елемент **ADALINE**

.

Последователен подход. Нека на стъпка *t* моментната грешка е

$$e(t) = d(t) - y(t) \Rightarrow e(t) = d(t) - \mathbf{x}^T(t) \cdot \mathbf{w}(t)$$

където:

w*(t)* е текущата стойност на вектора с теглата на мрежата и тези тегла трябва да се донастройват.

d*(t) е желаната стойност на изхода*

e*(t) е грешката*

Целта на обучението на неврона при последователния подход е чрез подходящ избор на теглата да бъде минимизирана моментната грешка. В тази връзка се дефинира целева функция или функция на грешката, като функция от теглата свързани с неврона:

$$\varepsilon(t) = \varepsilon(\mathbf{w}) = \frac{1}{2}e^2(t) = \frac{1}{2}\left(d(t) - \mathbf{x}^T(t)\mathbf{w}(t)\right)^2$$

От горната формула се вижда, че *ε* е функция от теглата *ε= ε(***w***)* и за намиране на нейния минимум трябва да се правят стъпки в посоката, обратна на градиента на функцията (т.к. градиентът определя посоката на най-бързото нарастване на функцията). Това е добре известният градиентен подход за оптимизация. В случая градиентът на целевата функция по отношение на вектора с неизвестните тегла (векторът от частните производни на *ε(***w***)* по отношение на вектора **w**) е

$$\frac{\partial \varepsilon(\mathbf{w})}{\partial \mathbf{w}} = e(t)\frac{\partial e(t)}{\partial \mathbf{w}} \qquad \frac{\partial \varepsilon(\mathbf{w})}{\partial w_j(t)} = e(t)\frac{\partial e(t)}{\partial w_j(t)}, \quad j = 0,1,\ldots,m$$

Адаптивен линеен елемент **ADALINE**

- Векторът с частните производни на грешката по отношение на теглата се намира след диференциране

$$\frac{\partial e(t)}{\partial \mathbf{w}(t)} = \frac{\partial (d(t) - \mathbf{x}^T(t) \cdot \mathbf{w}(t))}{\partial \mathbf{w}(t)} = -\mathbf{x}(t)$$

- Компонентите на вектора с частните производни на грешката по отношение на теглата са

$$\frac{\partial e(t)}{\partial w_j(t)} = \frac{\partial (d(t) - \mathbf{x}^T(t) \cdot \mathbf{w}(t))}{\partial w_j(t)} = -x_j(t), \quad j = 0,1,...,m$$

- След заместване на предходните уравнения, за вектора на градиента се получава

$$\frac{\partial \varepsilon(t)}{\partial \mathbf{w}(t)} = -\mathbf{x}(t) \cdot e(t) \qquad \frac{\partial \varepsilon(t)}{\partial w_j(t)} = -x_j(t) \cdot e(t), \quad j = 0,1,...,m$$

- Обучението се свежда до настройка на вектора с теглата чрез промяна на текущата му стойност с $\Delta\mathbf{w}(t)$ в посока, обратна на градиената на функцията на грешката, т.е.

$$\Delta\mathbf{w}(t) \propto -\frac{\partial \varepsilon(t)}{\partial \mathbf{w}(t)} = \mathbf{x}(t) \cdot e(t) \qquad \Delta w_j(t) \propto -\frac{\partial \varepsilon(t)}{\partial w_j(t)} = x_j(t) \cdot e(t), \quad j = 0,1,...,m$$

Адаптивен линеен елемент **ADALINE**

В резултат на разглежданията за донастройката на теглата на стъпка *t* се получава т.н. LMS delta rule (**МНК**)

стъпка обратна на посока на градиента

$$\mathbf{w}(t+1) = \mathbf{w}(t) + \Delta\mathbf{w}(t) = \mathbf{w}(t) + \eta \cdot \mathbf{x}(t) \cdot e(t)$$

а отделните компоненти на вектора **w***(t+1)* са

$$w_j(t+1) = w_j(t) + \Delta w_j(t) = w_j(t) + \eta \cdot x_j(t) \cdot e(t)$$

където η е параметър, наричан скорост на обучение и определя големината на стъпката, която се прави в посока обратна на градиента на целевата функция. Трябва да се има предвид, че векторът на градиента определя посоката на промяна на вектора с теглата, а η определя големината на стъпката при промяната. Ако η е с голяма стойност, процедурата по настройка на теглата може да стане разходяща, а ако η има малка стойност, процедурата ще е много бавна.

Практическото правило за избор на параметъра η е

$$0.1 \le m\eta < 1,$$

където m е броят на входовете на неврона.

Алгоритъм за прилагане на МНК

Алгоритъмът за прилагане на разглеждания метод включва:

- Стъпка 1: Инициализиране (задаване на начални стойности) на теглата (**w***(0)*=***0*** например), избор на стойност на скоростта на обучение η, дефиниране на критериий за спиране (обикновено това е желана стойност на целевата функция, която трябва да бъде достигната при минимизацията) и инициализиране на брояча на стъпките *t=1*.

- Стъпка 2: Изчисляване на грешката $e(t) = d(t) - \mathbf{x}^T(t) \cdot \mathbf{w}(t)$

- Стъпка 3: Донастройка на теглата $\mathbf{w}(t+1) = \mathbf{w}(t) + \eta \cdot \mathbf{x}(t) \cdot e(t)$

- Стъпка 4: Проверка на критерия за спиране. Ако не е удовлетворен, броячът на стъпките се увеличава с 1, т.е. *t=t+1* и се преминава отново към стъпка 2 от алгоритъма. Ако критерият за спиране е удовлетворен, алгоритъмът спира и векторът с теглата взема последната изчислена стойност.

Сходимост на МНК

Сходимостта на алгоритъма на най-малките квадрати зависи от статистическите свойства на данните от обучаващата извадка, както и от скоростта на сходимост η. Ако данните от обучаващата извадка са независими и са избрани на база на Гаусово разпределение, то може да бъде показано, че алгоритъмът е сходящ ако

$$0<\eta<2/\lambda_{max},$$

където λ_{max} е стойността на най-голямата собствена стойност на корелационната матрица на данните от обучаващата извадка

$$\mathbf{R}_{\mathbf{x}} = E\left[x(j)x^T(j)\right] = \lim_{P\to\infty}\frac{1}{P}\sum_{j=1}^{P}x(j)x^T(j)$$

Предимства и недостатъци на МНК

- **Предимства**

  -алгоритъмът на най-малките квадрати е много прост

  - той не зависи от модела

  -работи добре както със стационарни (постоянни стойности на статистическите им параметри), така и с нестационарни данни (статистическите им свойства са променливи).

- **Недостатъци на алгоритъма**

  -бавната сходимост

  -зависимост на скоростта на сходимост от диапазона на собствените стойности на корелационната матрица на данните от обучаващата извадка.

Пакетен и последователен подход

- Градиентният подход при решаването на оптимизационната задача (най-малки квадрати) за настройка на теглата на линейния неврон е свързан с намаляване на стойността на целевата функция на всяка стъпка, поради промяната на теглата в посока обратна на градиента на функцията (посоката на най-бързото нарастване на функцията).
- При последователния подход целевата функция е свързана с моментната грешка
- При пакетния подход целевата фунция се базира на усреднената грешка от всички входно-изходни данни в рамките на един епох.
- Поради това пакетният подход води до по-добри резултати, въпреки че е по-бавен.
- Последователният подход е по-бърз, но при него на всяка стъпка промяната на теглата зависи от реда на следване на елементите от обучаващата извадка.
- И при двата подхода за използването на делта правилото критерий за спиране е достигането на дадена желана стойност на целевата функция. И в двата случая като допълнителен критерий за спиране може да се използва зададен максимален брой епохи. След достигане на този брой, независимо от стойността на целевата функция, настройката на целевата функция спира.

Методи за ускоряване сходимостта на алгоритъма на най-малките квадрати

Адаптивна промяна на скоростта на обучение η

- Първоначално η може да се увеличава докато процедурата стане разходяща. После η може бързо да бъде намалявана. Известни са няколко начина за избор на скоростта на обучение:
- - $\eta(t) = \eta_0$; $t=1,2,3,\ldots$ - постоянна стойност на η;
- - $\eta(t) = \eta_0/t$; $t=1,2,3,\ldots$ - постоянно намаляване с промяна на стъпката;
- - $\eta(t) = \eta_0/(1+(t/\tau))$; $t=1,2,3,\ldots$ - закаляване с промяна на стъпката; (iterative annealing)

Методи за ускоряване сходимостта на алгоритъма на най-малките квадрати

Реализации с използване на т.нар “моменти” при донастройка на теглата

Тук на всяка стъпка вместо използване само на информация за градиента се използват и стойности за промяната на теглата от предишните стъпки.

$$\Delta \mathbf{w}(t) = -\eta \frac{\partial \varepsilon}{\partial \mathbf{w}}(t) + \alpha\, \mathbf{v}(t-1) = -\eta \frac{\partial \varepsilon}{\partial \mathbf{w}}(t) + \alpha\, \Delta \mathbf{w}(t-1)$$

Методи за ускоряване сходимостта на алгоритъма на най-малките квадрати

Посоката, обратна на градиента на целевата функция обикновено не сочи нейния минимум.

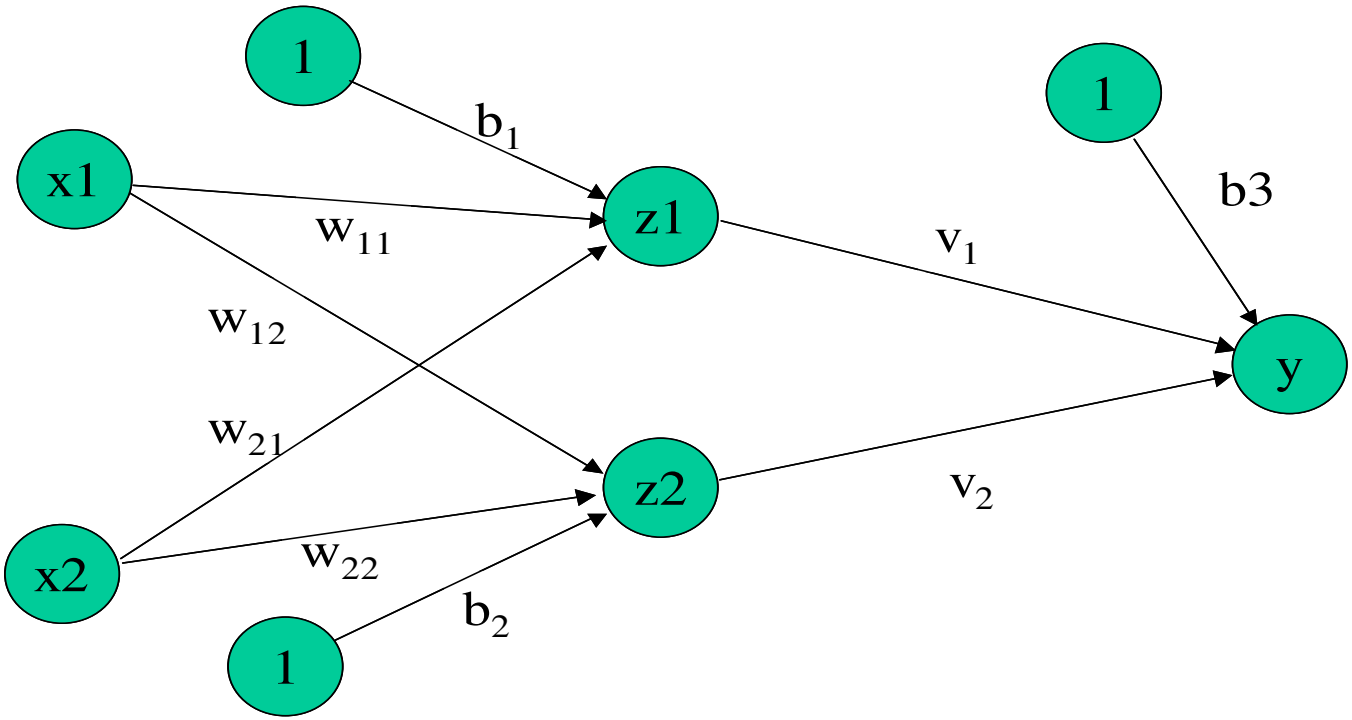
С модификации на данните, така че от всяка точка минимумът на целевата функция да е в посоката обратна на градиента минимумът може да се достигне само за една стъпка.

Може да се покаже, че ако $\mathbf{X}$ е $P \times m$ матрица, редовете на която са входните вектори, а $\mathbf{d}$ е P-мерен вектор от желаните изходи на неврона, то $\mathbf{w} = \mathbf{X}^{+}\mathbf{d}$,

където $\mathbf{X}^{+}=(\mathbf{X}^{T}\mathbf{X})^{-1}\mathbf{X}^{T}$ е т.нар псевдообратна матрица на данните от обучаващата извадка.

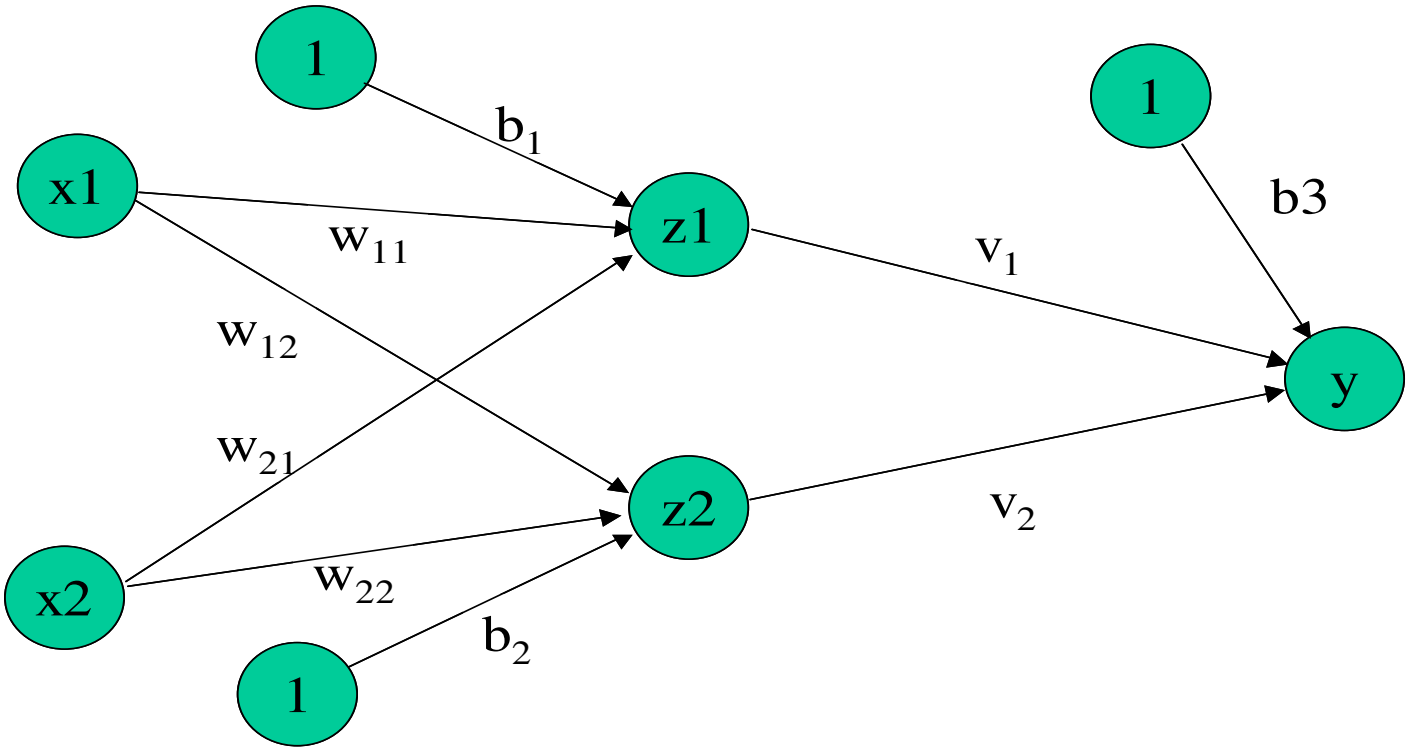
Many ADAptive LINear Elements (**MADALINE**)

В много приложения за повишаване на изчислителните възможности, се използват мрежи съставени от много линейни адаптивни елементи – Many ADAptive LINear Elements (MADALINE). В оригиналния си вариант MADALINE се състои от два слоя неврони – скрит слой от ADALINE (неврони *z1* и *z2*) и изходен слой (неврон *y*).



Many ADAptive LINear Elements (**MADALINE**)

В случая теглата на двата линейни адаптивни елементи *z1* и *z2* от скрития слой, които ползват едни и същи входни сигнали, се настройват с обучение, а теглата на изходния неврон се избират въз основа на анализ на операцията, която трябва да се реализира.



Many ADAptive LINear Elements (**MADALINE**)

Ако невронът *y* трябва да изпълнява функцията логическо или “OR”, изходният му сигнал трябва да е –1, ако сигналите от *z1* и *z2* са –1, а в останалите случаи изходният му сигнал трябва да е 1.

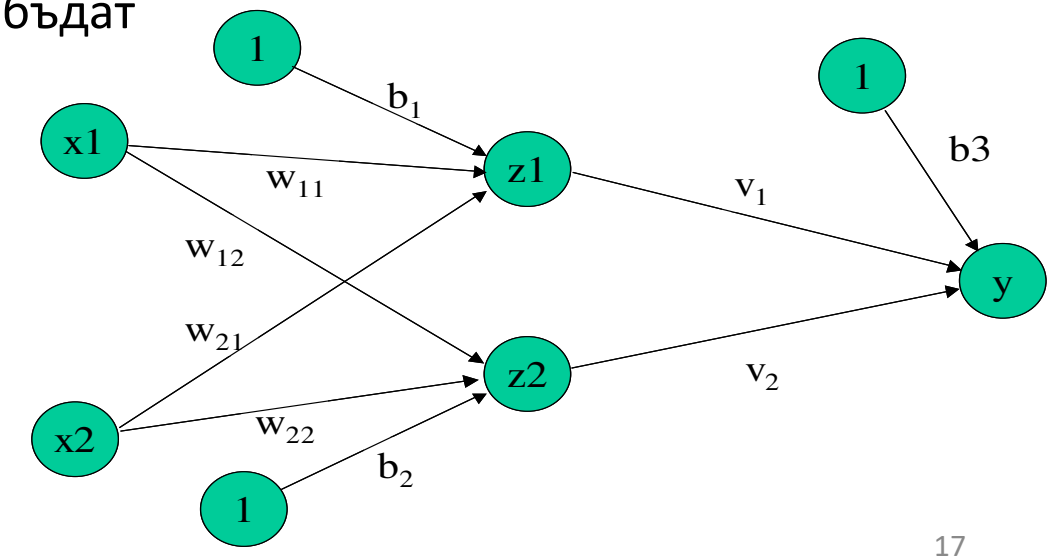
Тогава активационните функции на трите неврона трябва да са от вида

$$f(v)=\begin{cases}1, & ако \;\; v \geq 0\\ -1, & ако \;\; v<0\end{cases}$$

а теглата v_1 и v_2 и отместването *b*,

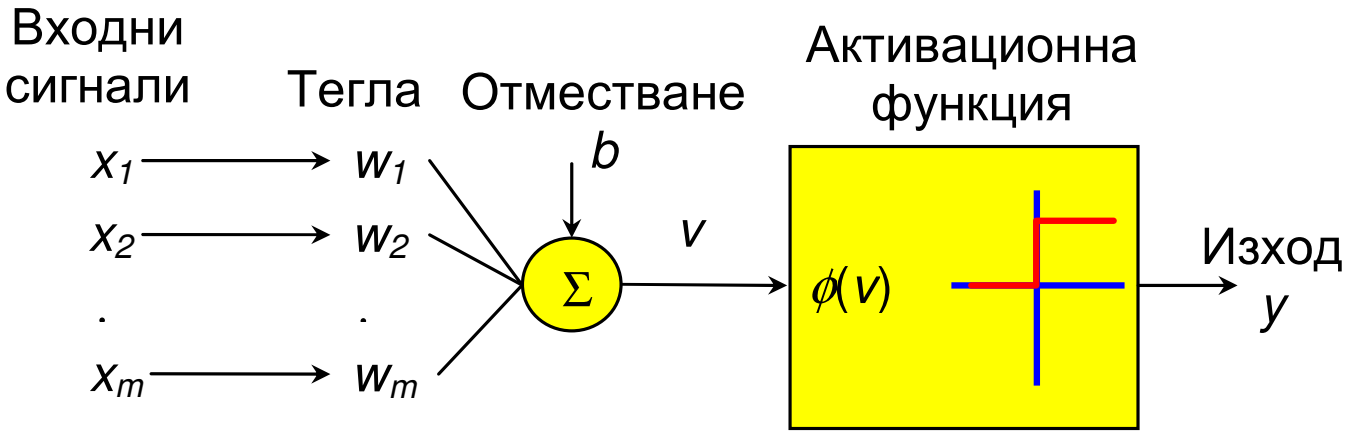
свързани с неврона *y* могат да бъдат избрани по следния начин:

v_1=0.5, v_2=0.5 и *b*=0.5.



Перцептрон

• За разлика от до сега разглеждания адаптивен линеен елемент и МНК перцептронът е нелинеен неврон.
•Перцептронът е базиран на неврона на McCulloch-Pitt, но за разлика от него е свързан с правило за обучение, наречено
правило за обучение на перцептрона.
• Перцептронът използва релейна активационна функция ф(v) даваща 1 или 0 на изхода на мрежата.



$$v = \sum_{j=1}^{m} w_j x_j + b \qquad y = \phi(v)$$

Единичен перцептрон

Входът на мрежата е $$v=\sum_{j=1}^{m} w_j x_j + b = \sum_{j=0}^{m} w_j x_j$$

$w_0=b$, а x_0 е сигнал зададен с постоянна стойност 1. Изходният сигнал y на перцептрона се определя с формулата

Изходът на мрежата е $$y=\phi(v)=\begin{cases}1, & \text{ако } v \geq \theta \\ 0, & \text{ако } v < \theta\end{cases}$$

- θ е прагова стойност свързана с перцептрона (неврона)
- ако стойността му не е спомената специално се счита, че е нула $\theta=0$.

Единичен перцептрон

- С перцептрона се дефинира решаващо правило - хиперравнина на решенията в m-мерното пространство.
- Хиперравнината за вземане на решение (decision boundary) за дадена мрежа се определя от вход, който дава 0 на изхода на мрежата. Едното подпространство, което се дефинира с уравнението $w_1x_1+w_2x_2+...+ w_mx_m+b\geq 0$ съответства на изходен сигнал на перцептрона y=1, докато другото подпространство, което се дефинира с $w_1x_1+w_2x_2+...+ w_mx_m+b<0$ съответства на изходен сигнал на перцептрона y=0. В този контекст даден перцептрон дефинира хиперравнина за вземане на решение (разделяща хиперравнина).
- Хиперравнината за вземане на решение се задава с:

$$\sum_{j=1}^{m} w_j x_j + b = w_1 x_1 + w_2 x_2 + \ldots + w_m x_m + b = 0$$

Разделяща хиперравнина - пример

- Да приемем, че имаме $w_1 = 1$, $w_2 = 2$, and $b = -1$ за перцептрон с два входа (m=2) x_1 и x_2
- Хиперравнината на решенията е права(линия) с уравнение :

$$w_1x_1 + w_2x_2 + b = 1x_1 + 2x_2 - 1 = 0$$

- Изходният сигнал на перцептрона за точките от горната дясна полуравнина е 1, а за долната лява полуравнина е 0
- За да се намерят x_1 и x_2 пресечните точки при $x_2=0$ и $x_1=0$:
  - Задава се $x_1 = 0$ и се решава за x_2, $2x_2 - 1 = 0$ или $x_2 = 0.5$
  - Задава се $x_2 = 0$ и се решава за x_1, $1x_1 - 1 = 0$ или $x_1 = 1$

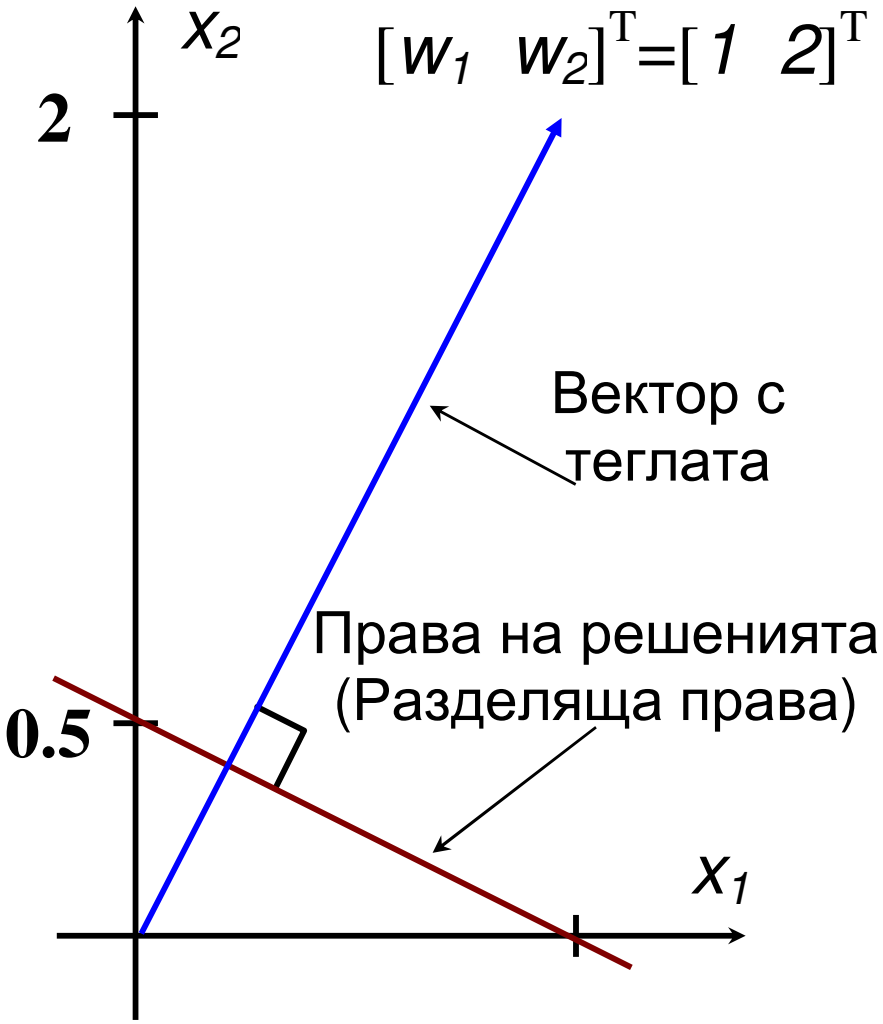
Разделяща хиперравнина - пример

ГЕОМЕТРИЧНА ИНТЕРПРЕТАЦИЯ

- Хиперравнината за вземане решение е перпендикулярна на вектора на теглата

- Всяка точка от едната страна на граничната хиперравнина дава изходен сигнал на перцептрона 1, а другата полуравнина дава изходен сигнал на перцептрона 0

Коя страна е ‘1’ и коя ‘0’?



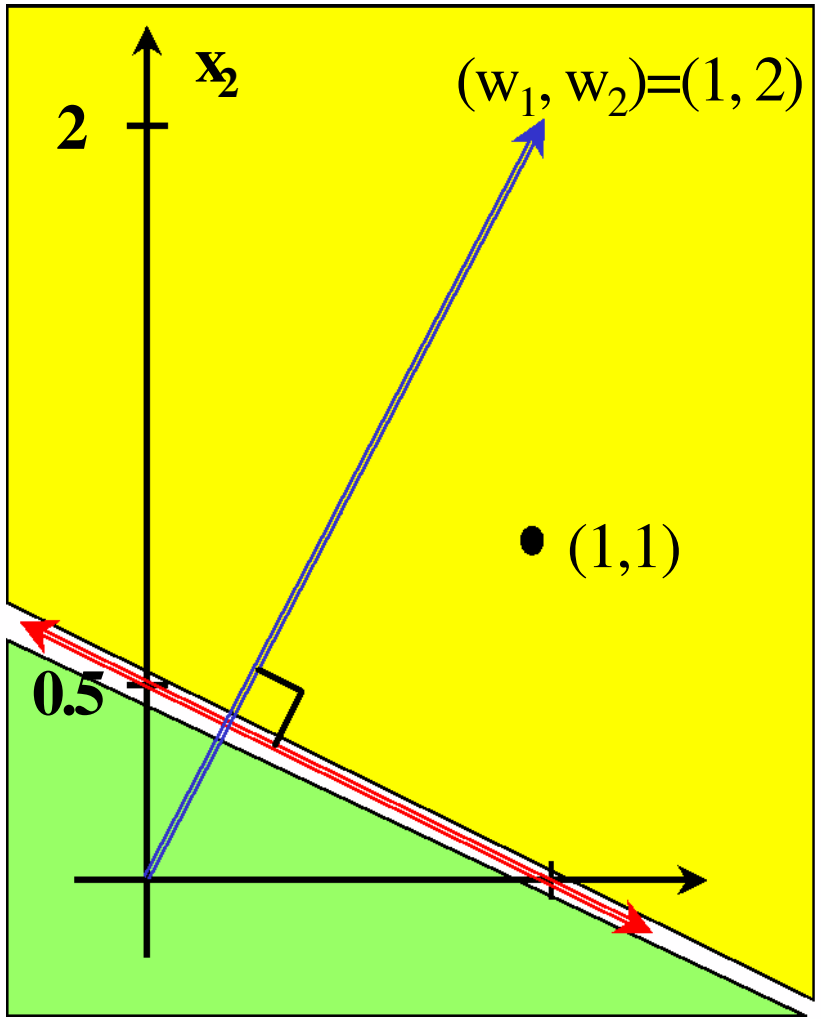
Разделяща хиперравнина - пример

- Нека вземем пробна точка от едната страна на полуравнината:

$(x_1, x_2)=(1, 1)$

$\phi (1 . 1 + 2 . 1 . 1) = \phi (2) = 1$

- Всички точки нагоре и надясно от линията на решенията (the decision line) ще дават изход 1 и всички точки под и наляво от линията ще дават изход 0



Разделяща хиперравнина

Всички точки над и надясно от хиперравнината за вземане на решение са с: $\vec{w} \cdot \vec{x} > -b$

Всички точки надолу и наляво са с: $\vec{w} \cdot \vec{x} < -b$

- Векторът с теглата $[w_1 \;\; w_2]^T = [1 \;\; 2]^T$ (от примера) е перпендикулярен на правата на решенията и е насочен към полуравнината, за която стойността на изходния сигнал е *1*.

- Векторът на теглата е насочен с посоката към хиперравнината за вземане на решение даваща 1.

Разделяща хиперравнина

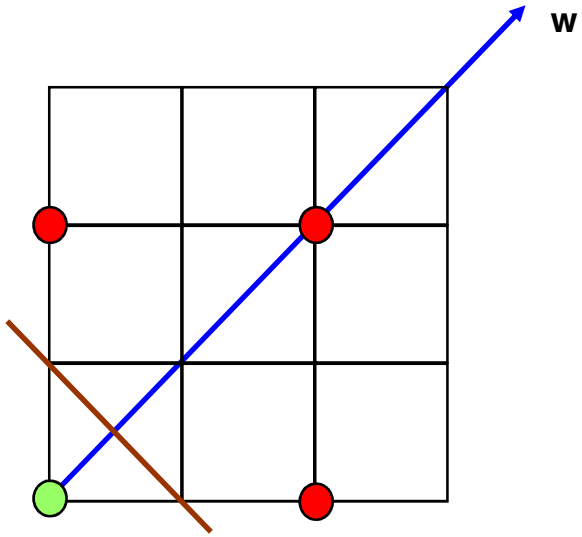
Нека разгледаме следния елементарен пример

$$\left\langle \mathbf{x}(1)=\begin{bmatrix}0\\0\end{bmatrix}, d(1)=0 \right\rangle \quad \left\langle \mathbf{x}(2)=\begin{bmatrix}0\\1\end{bmatrix}, d(2)=1 \right\rangle \quad \left\langle \mathbf{x}(3)=\begin{bmatrix}1\\0\end{bmatrix}, d(3)=1 \right\rangle \quad \left\langle \mathbf{x}(4)=\begin{bmatrix}1\\1\end{bmatrix}, d(4)=1 \right\rangle$$

•Входните вектори **x**(1), **x**(2), **x**(3) и **x**(4) са изобразени като точки в двумерна координатна система.

• Те могат да бъдат разгледани като образи (обекти), които принадлежат на два класа, в зависимост от стойността на желания изходен сигнал на неврона.

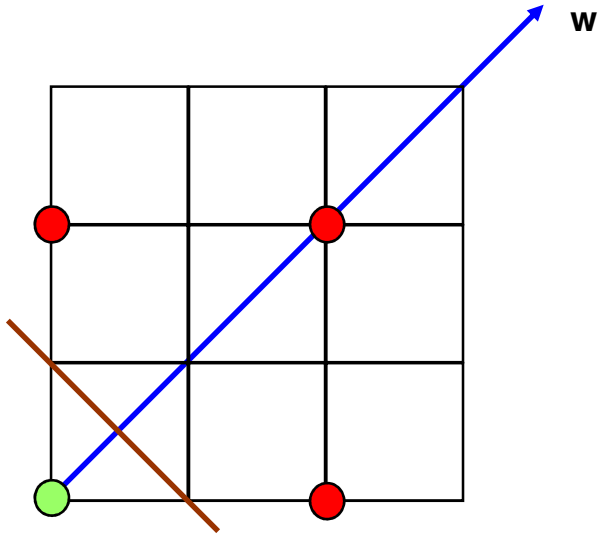
•Точките, които съответстват на изходен сигнал 1, са означени на фигурата със защриховано кръгче, докато тази точка, която съответства на изходен сигнал 0 е означена с незащриховано кръгче.



Пример за разделяне на образи на два класа

Разделяща хиперравнина

• Вектора на теглата определя разположението на разделящата хиперравнина. В случая ще изберем вектора да е с ‘подходяща’ посока. Съществува безкрайно голям брой от варианти за избор. Един от тях е $\mathbf{w}=(2,2)^T$



Пример за разделяне на образи на два класа

• Отместването определя конкретната позиция на правата на решенията по отношение на зададената от вектора **w** ориентация. Нека точката **x=(0,0.5) е на избраната** разделяща хиперравнина.

Стойността на b **е** $(2,2)\cdot(0,0.5)^T+b=0$ **и** $b=-1$

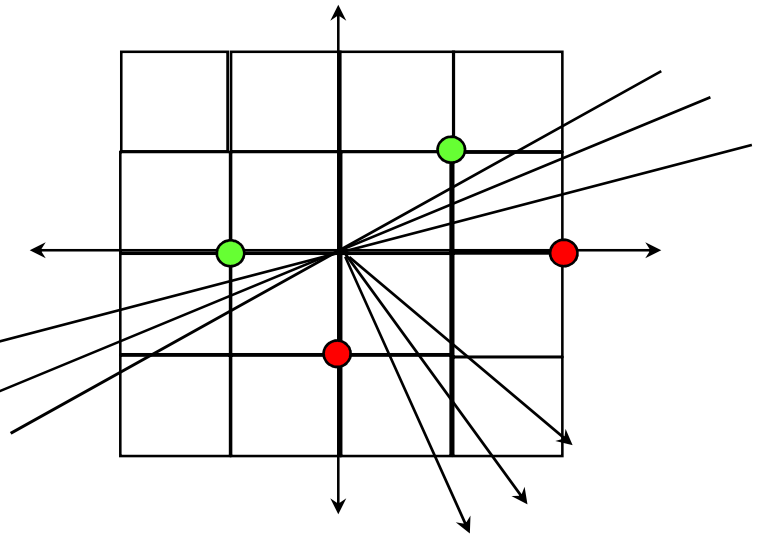
Изходният сигнал на съответния прцептрон е: $$y=\phi\left\{\vec{w}^T\cdot\vec{x}+b\right\}=\phi\left\{[2,2]\cdot\begin{bmatrix}x_1\\x_2\end{bmatrix}-1\right\}$$

Обучение на перцептрона

Нека да разгледаме друг пример

$$\left\langle \bar{x}_1 = \begin{bmatrix} 2 \\ 0 \end{bmatrix}, d_1 = 1 \right\rangle \left\langle \bar{x}_2 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, d_2 = 1 \right\rangle \left\langle \bar{x}_3 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, d_3 = 0 \right\rangle \left\langle \bar{x}_4 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}, d_4 = 0 \right\rangle$$

- За разгледания двумерен пример съществуват безкрайно много прави на решенията, които могат да разделят образите на два класа.
- На фигурата са показани само три прави, минаващи през центъра (в този случай отместванията са нула) и съответните им вектори с теглата (перпендикулярни на правите). За този пример фиксираме отместването да е 0.



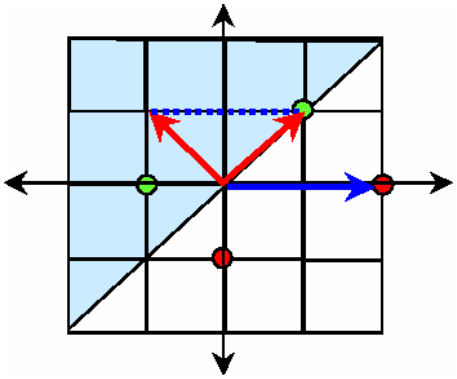
Пример за разделяне на 4 образи в 2 два класа с по 2 образа

Обучение на перцептрона

Първата стъпка е да инициализираме вектора с теглата $w^T=(-1,1)^T$. Можем да изберем каквито и да са стойности, но обикновено се избират случайно генерирани стойности за теглата.

- Прилагаме първия образец (**training sample**) от обучаващата извадка $x_1=(2,0)$, с който се получава:

$$\phi(\mathbf{w}^T\mathbf{x}) = \phi((-1,1).(2,0)^T)= \phi(-2) = 0$$



Това е грешно (d=1) и поради тази причина **е необходимо да се премести вектора на теглата към образеца (sample-a)**

- Настройваме вектора на теглата по следния начин:

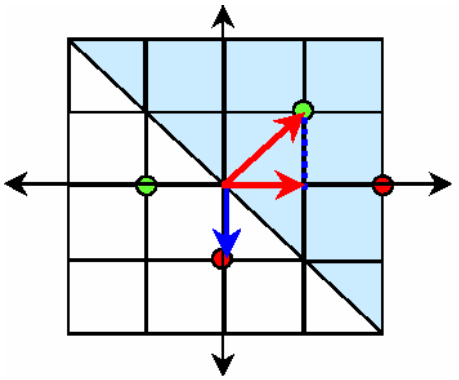
w= w+ x така, че:

$\mathbf{w} = (-1,1)^T + (2,0)^T = (1,1)^T$

Обучение на перцептрона

- Прилагаме втория образец (training sample) от обучаващата извадка $x_2=(0,-1)$, с който се получава: $\phi(\mathbf{w}^T\mathbf{x}) = \phi((1,1)\cdot(0,-1)^T)= \phi(-1) = 0$

Това е грешно (d=1) и поради тази причина **е необходимо да се премести вектора на теглата към образеца (sample-a)**



- Настройваме вектора на теглата по следния начин:

w= w+ x така, че:

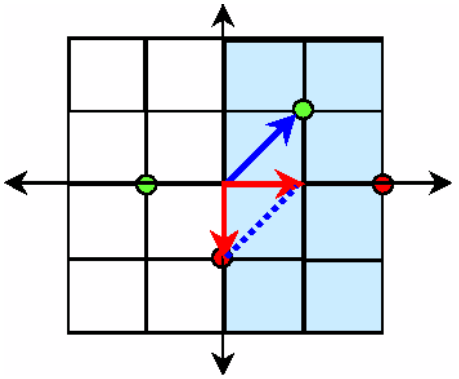
$\mathbf{w} = (1,1)^T + (0,-1)^T = (1,0)^T$

Обучение на перцептрона

- Прилагаме третия образец (training sample) от обучаващата извадка $x_3=(1,1)$, с който се получава:

$$\phi(\mathbf{w}^T\mathbf{x}) = \phi((1,0)\cdot(1,1)^T)= \phi(1) = 1$$

Това е грешно (d=0) и поради тази причина **е необходимо да се премести вектора на теглата в посока далечна на образеца (sample-а)**



- Настройваме вектора на теглата по следния начин:

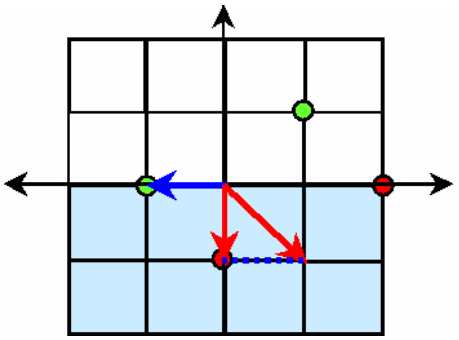
w= w-x така, че:

$$\mathbf{w} = (1,0)^T - (1,1)^T = (0,-1)^T$$

Обучение на перцептрона

- Прилагаме четвъртия образец (training sample) от обучавашата извадка $x_4=(-1,0)$, с който се получава: $\phi(\mathbf{w}^T\mathbf{x}) = \phi((0,-1)\cdot(-1,0)^T) = \phi(0) = 1$

Това е грешно (d=0) и поради тази причина **е необходимо да се премести вектора на теглата в посока далечна на образеца (sample-a)**



- Настройваме вектора на теглата по следния начин:

w= w—x така, че:

$\mathbf{w} = (0,-1)^T - (-1,0)^T = (1,-1)^T$

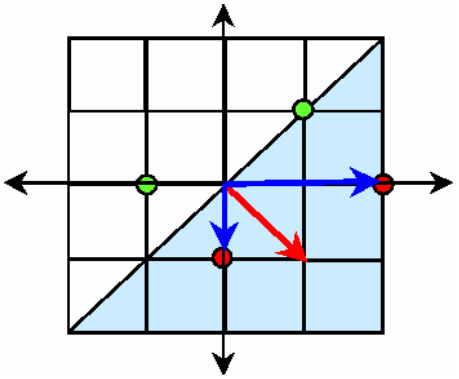
Обучение на перцептрона

По време на обучението остават грешки!!!
Тогава можем да приложим данните за обучение отново.

- Прилагаме първия семпъл (образец) от обучаващата извадка $x_1=(2,0)$, който дава:

$$\phi(\mathbf{w}^T\mathbf{x}) = \phi((1,-1)\cdot(2,0)^T)= \phi(2)= 1$$

Това е правилно → **не променяме теглата**



- Прилагаме втория семпъл (образец) от обучаващата извадка $x_2=(0,-1)$, който дава:

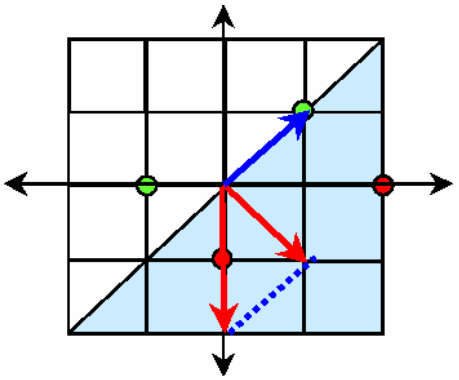
$$\phi(\mathbf{w}^T\mathbf{x}) = \phi((1,-1)\cdot(0,-1)^T = \phi(1) = 1$$

Това е правилно → **не променяме теглата**

Обучение на перцептрона

- Прилагаме третия образец (training sample) от обучаващата извадка $x_3=(1,1)$, с който се получава: $\phi(\mathbf{w}^T\mathbf{x}) = \phi((1,-1)\cdot(1,1)^T)= \phi(0) = 1$

Това е грешно (d=0) и поради тази причина
ние променяме теглата



- Настройваме вектора на теглата по следния начин:

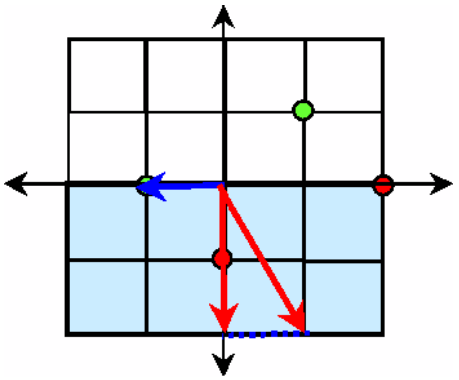
w= w—x така, че:

$\mathbf{w} = (1,-1)^T - (1,1)^T = (0,-2)^T$

Обучение на перцептрона

- Прилагаме четвъртия образец (training sample) от обучаващата извадка $x_4=(-1,0)$, с който се получава: $\phi(\mathbf{w}^T\mathbf{x}) = \phi((-0,-2)\cdot(-1,0)^T)= \phi(0) = 1$

Това е грешно (d=0) и поради тази причина **ние променяме теглата**



- Настройваме вектора на теглата по следния начин:

w= w—x така, че:

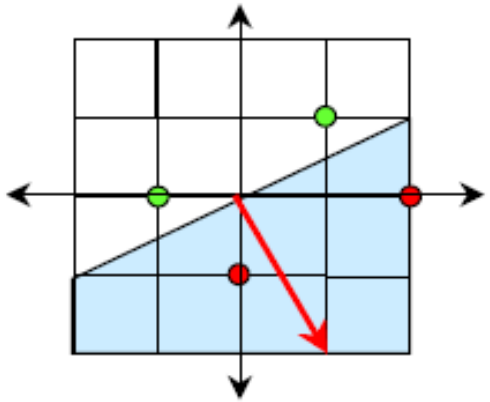
$\mathbf{w} = (0,-2)^T - (-1,0)^T = (1,-2)^T$

Обучение на перцептрона

Решение:

$x_1=(2,0)$ $\phi((1,-2)\cdot(2,0)^T = 2\ ) = 1$
$x_2=(0,-1)$ $\phi((1,-2)\cdot(0,-1)^T = 2\ ) = 1$
$x_3=(1,1)$ $\phi((1,-2)\cdot(1,1)^T = -1\ ) = 0$
$x_2=(-1,0)$ $\phi((1,-2)\cdot(-1,0)^T = -1\ ) = 0$

$$\phi\left\{\begin{bmatrix}1 & -2\end{bmatrix}\cdot\begin{bmatrix}x_1\\ x_2\end{bmatrix}\right\}$$



Правило за обучение на перцептрона

- Ако $d = 0$ и $\phi(\mathbf{w}^T\mathbf{x}) = 1$, тогава $\mathbf{w} = \mathbf{w} - \mathbf{x}$
- Ако $d = 1$ и $\phi(\mathbf{w}^T\mathbf{x}) = 0$, тогава $\mathbf{w} = \mathbf{w} + \mathbf{x}$
- Ако $d = \phi(\mathbf{w}^T\mathbf{x})$, тогава $\mathbf{w} = \mathbf{w}$

Или

- Нека $e = d - \phi(\mathbf{w}^T\mathbf{x})$
- Така правилото се обобщава като:

$$\mathbf{w} = \mathbf{w} + e\mathbf{x}$$

Обучение на множество перцептрони

- Правилото за обучение на перцептрона може да бъде ализирано за два или повече перцептрона

- За k-ти перцептрон:
  Теглата се настройват по $\mathbf{w}_k := \mathbf{w}_k + e_k\mathbf{x}$
  Отместването се настройва по $b_k := b_k + e_k$

Много перцептрони

Матричната векторна форма на правилото за обучение на много перцептрони

Векторна функция $\phi(\mathbf{W}.\mathbf{x} + \mathbf{b})$

$$\phi(\mathbf{W}\cdot\mathbf{x}+\mathbf{b}) = \phi\left\{\begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2m} \\ \vdots & \vdots & \cdots & \vdots \\ w_{k1} & w_{k2} & \cdots & w_{km} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_k \end{bmatrix}\right\}$$

Вектор на грешката при много перцептрони

Вектора на грешката **e** е **d** - ϕ(**W.x** + **b**)

$$\mathbf{e} = \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_k \end{bmatrix} = \left\{ \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_k \end{bmatrix} - \phi \left\{ \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2m} \\ \vdots & \vdots & \cdots & \vdots \\ w_{k1} & w_{k2} & \cdots & w_{km} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_k \end{bmatrix} \right\} \right\}$$

Правило за обучение на мрежа от много перцептрони

Настройка на теглата: $\mathbf{W := W + e.x^T}$

$$\begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2m} \\ \vdots & \vdots & \cdots & \vdots \\ w_{k1} & w_{k2} & \cdots & w_{km} \end{bmatrix} := \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2m} \\ \vdots & \vdots & \cdots & \vdots \\ w_{k1} & w_{k2} & \cdots & w_{km} \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_k \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}^T$$

$$\begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2m} \\ \vdots & \vdots & \cdots & \vdots \\ w_{k1} & w_{k2} & \cdots & w_{km} \end{bmatrix} := \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2m} \\ \vdots & \vdots & \cdots & \vdots \\ w_{k1} & w_{k2} & \cdots & w_{km} \end{bmatrix} + \begin{bmatrix} e_1x_1 & e_1x_2 & \cdots & e_1x_m \\ e_2x_1 & e_2x_2 & \cdots & e_2x_m \\ \vdots & \vdots & \cdots & \vdots \\ e_kx_1 & e_kx_2 & \cdots & e_kx_m \end{bmatrix}$$

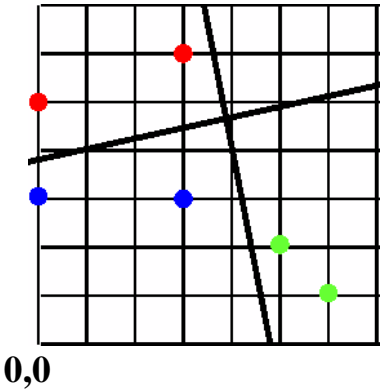
Правило за обучение на мрежа от много перцептрони

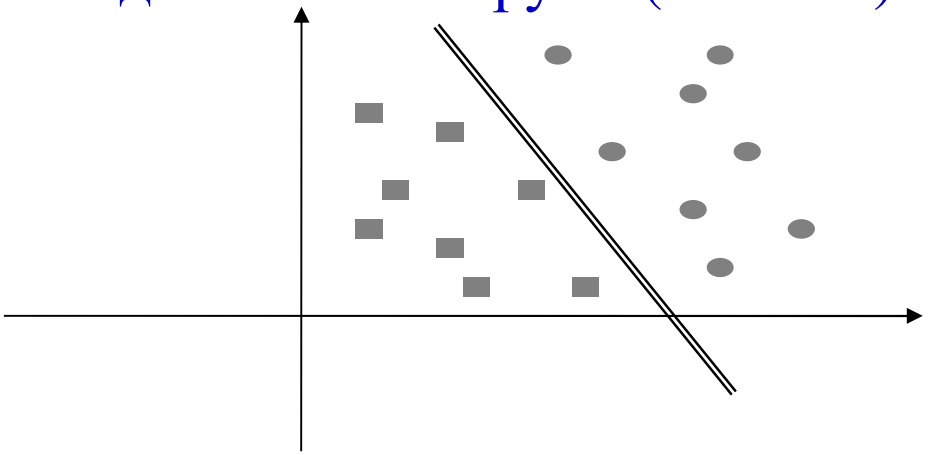
Настройка на отместванията: **b := b + e**

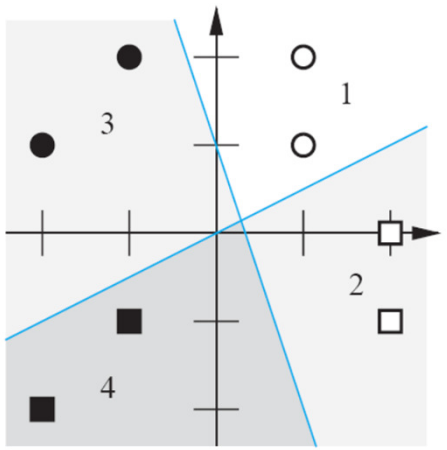
$$\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_k \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_k \end{bmatrix} + \begin{bmatrix} e_1 \\ e_2 \\ \vdots \\ e_k \end{bmatrix}$$

Какво може да се направи с повече перцептрони?

- Един перцептрон може да раздели входните данни на две изходни групи (класове).
- Два перцептрона могат да разделят входните данни на четири групи (класове).
- K перцептрона могат да разделят входните данни на 2^K групи (класове) ?

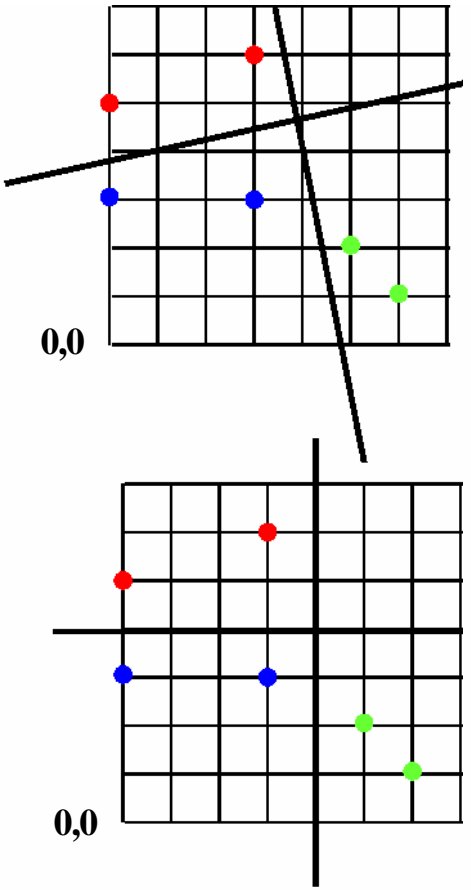






Задача за класификация на данни в 3 класа

Вход (x_1,x_2)	Клас (d_1,d_2)
(6,1)	(0,1)
(5,2)	(0,1)
(0,3)	(0,0)
(3,3)	(0,0)
(0,5)	(1,0)
(3,6)	(1,0)



Доказателство за сходимост

Разгледаната процедура на *правилото за обучение на перцептрона* е сходяща при произволни начални стойности на теглата и отместването, ако образите от обучаващата извадка са линейно разделими. Това е и същността на теоремата за сходимост на правилото за обучение на перцептрона, която може да бъде формулирана по следния начин:

Теорема: Нека са дадени векторите от обучаващата извадка **x**(1), **x**(2), ..., **x**(P), които са образи от два класа C1 и C2. Ако класовете са линейно разделими, то независимо от началната стойност на вектора **w**=$[w_0\ w_1\ \ldots\ w_m]^T$, правилото за обучение на перцептрона е сходящо за краен брой стъпки към вектор с тегла, гарантиращ правилна класификация на образите.

Доказателство за сходимост

Доказателство:

- В случая ще бъде използван модела на перцептрона с посоянен сигнал $x_0=1$ и тегло равно на отместването $w_0=b$, свързано с него
- Ако имаме пакет от входни вектори: $\mathbf{x}(1), \mathbf{x}(2), \mathbf{x}(3),...$ които са представителни образци на две линейно разделими класа C 1 и C2, то приемаме, че съществува вектор на теглата $\mathbf{w}^*$ при който:

$\mathbf{w}^{*T}\mathbf{x} > 0$ ако $\mathbf{x}$ принадлежи към C1

$\mathbf{w}^{*T}\mathbf{x} <= 0$ ако $\mathbf{x}$ принадлежи към C2

или нека $\mathbf{w}^T\mathbf{x}(j)\geq 0$ ако $\mathbf{x}(j)\in C1$ и $\mathbf{w}^T\mathbf{x}(j)<0$ ако $\mathbf{x}(j)\in C2$.

- Нека класът -C2 е образуван от всички елементи на C2, но с обратен знак, т.е. $-x(j)\in -C2$, ако $x(j)\in C2$. Тогава множеството от образите на обучаващата извадка $C1\cup C2$ може да бъде модифицирано като бъде формирано множеството $C=C1\cup -C2$, което се състои от всички елементи на C1 и взетите със знак минус елементи на C2. Тогава $\forall x(j)$, $x(j)\in C$, $\mathbf{w}^T\mathbf{x}(j)\geq 0$, т.е. всички елементи на C са от един и същи клас, за който $\mathbf{w}^T\mathbf{x}(j)\geq 0$, $j=1,2,...,P$.
- Тогава всичките желани стойности на изхода за C са само +1

Доказателство за сходимост

- Ще приложим правилото за обучение на перцептрона, като в скоби ще поставяме номера на образеца (sample) от обучаващата извадка
- Приемаме, че инициализираната стойност на теглата е **w**(0)=**0**

Нека приемем, че **x**(t) е елемент от C и $\mathbf{w}^{*T}\mathbf{x} <= 0$ за всички t=1,2,3,..P
Тези образи са некоректно класифицирани и съгласно правилото за обучение на перцептрона **w(нов) = w (стар)+ x**. Следователно ако началната стойност на вектора **w** е **0**, то на стъпка *t*, векторът с теглата е

w(t) = **x**(1) + **x**(2)+ **x**(3) + …+ **x**(t)

Ще бъде поазано, че ако образите от двата класа C1 и C2 са линейно разделими и се използва правилото за обучение на перцептрона за краен брой стъпки *t* ще бъде определен вектор с тегла w, с който се класифицират правилно всички входни вектори (образи)!

Доказателство за сходимост

Корекцията на вектора на теглата при прилагане на образците е:

$\mathbf{w}(t) = \mathbf{x}(1) + \mathbf{x}(2) + \ldots + \mathbf{x}(t)$ (1)

- т.к. сме приели, че C1 и C2 са линейно разделими, то трябва да има решение $\mathbf{w}^*$, при което $\mathbf{w}^{*T}\mathbf{x}(t) > 0$ за всички образци от клас C 1

- Нека $\alpha = \min \mathbf{w}^{*T}\mathbf{x}(t)$ (2)

за всяко **x** от клас C

- Умножавайки (1) по $\mathbf{w}^{*T}$ ни дава

$\mathbf{w}^{*T}\mathbf{w}(t) = \mathbf{w}^{*T}\mathbf{x}(1) + \mathbf{w}^{*T}\mathbf{x}(2) + \ldots \mathbf{w}^{*T}\mathbf{x}(t)$

Вземайки впредвид (2)

$\mathbf{w}^{*T}\mathbf{w}(t) >= t\alpha$

Доказателство за сходимост

- Използвайки неравенството на Коши-Шварц (Cauchy-Schwartz) можем да запишем, че:

$$\|\mathbf{w}^*\|^2\|\mathbf{w}(t)\|^2 >= [\mathbf{w}^{*T}\mathbf{w}(t)]^2 >= t^2\,\alpha^2$$

или

$$\|\mathbf{w}(t)\|^2 >= t^2\,\alpha^2 / \|\mathbf{w}^*\|^2 \qquad (3)$$

- отбелязваме, че $\mathbf{w}(i) = \mathbf{w}(i-1) + \mathbf{x}(i)$ за $\mathbf{x}(i)$ бидейки образец от C,където $i = 1, 2, 3, ..., t$. Вземайки модула повдигнат на квадрат на $\mathbf{w}(i) = \mathbf{w}(i-1) + \mathbf{x}(i)$ получаваме

$$\|\mathbf{w}(i)\|^2 = \|\mathbf{w}(i-1)\|^2 + \|x(i)\|^2 + 2\mathbf{w}^T(i-1)\mathbf{x}(i)$$

- т.к. сме направили допускане, че образците са погрешно класифицирани, $\mathbf{w}^T(i-1)\mathbf{x}(i) < 0$ то ние можем да запишем неравенството:

$$\|\mathbf{w}(i)\|^2 <= \|\mathbf{w}(i-1)\|^2 + \|x(i)\|^2$$

Доказателство за сходимост

От $\|\mathbf{w}(i+1)\|^2 - \|\mathbf{w}(i)\|^2 <= \|x(i)\|^2$ за i = 1, 2, 3, . . .,t

- $(\|\mathbf{w}(1)\|^2 - \|\mathbf{w}(0)\|^2 <= \|\mathbf{x}(1)\|^2) +$
  $(\|\mathbf{w}(2)\|^2 - \|\mathbf{w}(1)\|^2 <= \|\mathbf{x}(2)\|^2) +$
  $(\|\mathbf{w}(3)\|^2 - \|\mathbf{w}(2)\|^2 <= \|\mathbf{x}(3)\|^2) +$
  .....
  $(\|\mathbf{w}(t)\|^2 - \|\mathbf{w}(t-1)\|^2 <= \|\mathbf{x}(t)\|^2),$

$$\|\mathbf{w}(t)\|^2 + \|\mathbf{w}(0)\|^2 <= \|\mathbf{x}(1)\|^2 + \|\mathbf{x}(2)\|^2 + \ldots + \|\mathbf{x}(t)\|^2$$

и защото $w(0)=0$

$$\|\mathbf{w}(t)\|^2 <= \|\mathbf{x}(1)\|^2 + \|\mathbf{x}(2)\|^2 + \ldots + \|\mathbf{x}(t)\|^2 <= t\beta \quad (4)$$

където $\beta=\max(\|\mathbf{x}(i)\|^2), \mathbf{x}(i) \in C, \beta>=0$

Доказателство за сходимост

От (3) и (4) следва: $\|\mathbf{w}(t)\|^2 \geq \frac{t^2\alpha^2}{\|\mathbf{w}^*\|^2}$ и $\|\mathbf{w}(t)\|^2 \leq t\beta$ $\boxed{\frac{\alpha^2 t^2}{\|\mathbf{w}^*\|^2} \leq \|\mathbf{w}(t)\|^2 \leq \beta t}$

От тук следва, че трябва да има някое t удовлетворяващо и двете уравнения, което ще наречем t_{max} . Тогава:

$$\frac{t_{max}^2\alpha^2}{\|\mathbf{w}^*\|^2} = t_{max}\beta$$

(функцията ограничаваща $\|\mathbf{w}(t)\|^2$ отдолу, нараства квадратично с увеличаване на стъпката *t*, а функцията която ограничава $\|\mathbf{w}(t)\|^2$ отгоре нараства линейно с увеличаване на *t, m.e.* първия член ограничаващ функцията отдолу нараства с увеличаване на t по-бързо отколкото втория)

Решавайки уравнението по t_{max} ние намираме, че:

$$t_{max} = \frac{\beta\|\mathbf{w}^*\|^2}{\alpha^2}$$

• Поради тази причина, за всички семпли t, с $\mathbf{w}(0)=\mathbf{0}$, приемайки наличието на решение, процесът на обучение ще приключи за най-много за t_{max} стъпки

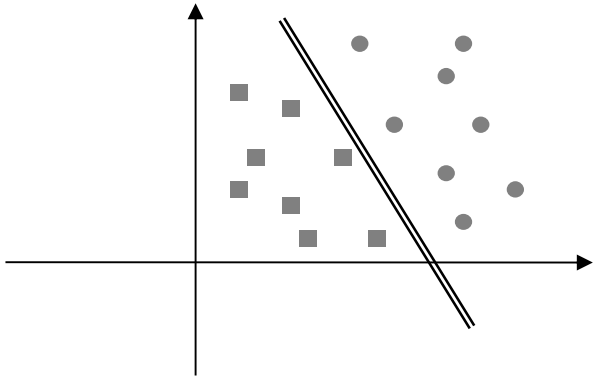
Или за някое $t = t_{max}$ горната и долната граници на най-горното уравнение ще се изравнят, поради което *t* не може да става по-голямо от t_{max} , т.е. ще дойде момент, в който настройката на теглата спира т.к. всички образи са класифицирани правилно, с което теоремата е доказана.

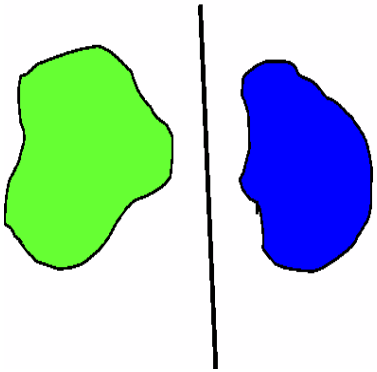
Алгоритъм за обучение на перцептрона

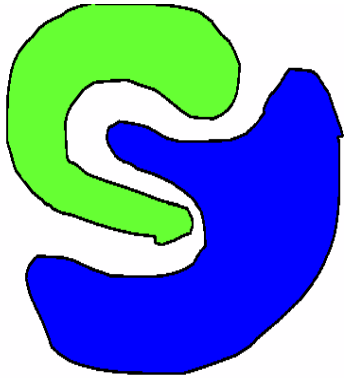
- Задаваме $\mathbf{w}(0) = \mathbf{0}$
- за всеки образец от обучаващата извадка
  x(t), t=1 ,2,3 ……..
- се изчислява $e = d - \phi(\mathbf{w}^T\mathbf{x}(t))$, където e грешката, d е жалания изход на мрежата, **w** е вектор на теглата
- Прилагаме правилото за обучение: $\mathbf{w} = \mathbf{w} + e\mathbf{x}(t)$ за да обновим коефициентите на теглата и $b = b + e$ за да обновим стойностите на отместванията
- Прекратяваме процедурата ако **няма обноване на теглата** при подаване на данните от обучаващата извадка или до достигане на някакъв друг критерии за спиране.

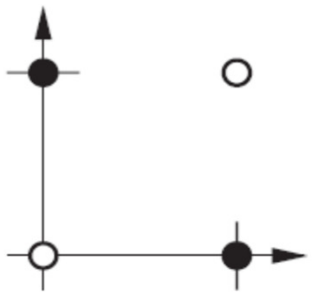
Ограничения на перцептрона

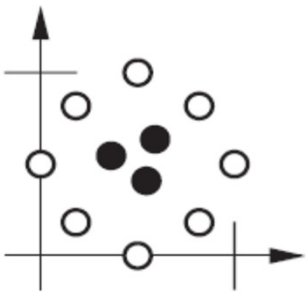
- От геометрична гледна точка един перцептрон може да раздели пространството на входните вектори на два класа.
- С един перцептрон могат да бъдат разделяни (правилно класифицирани) само линейно разделими образи

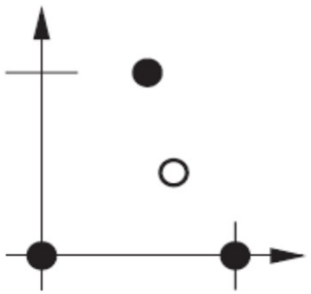








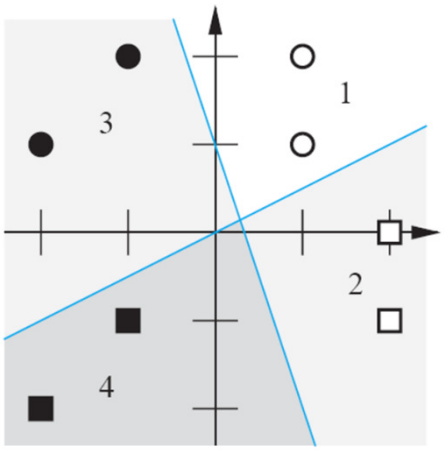


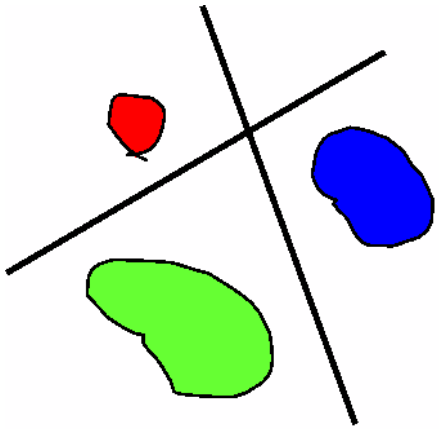


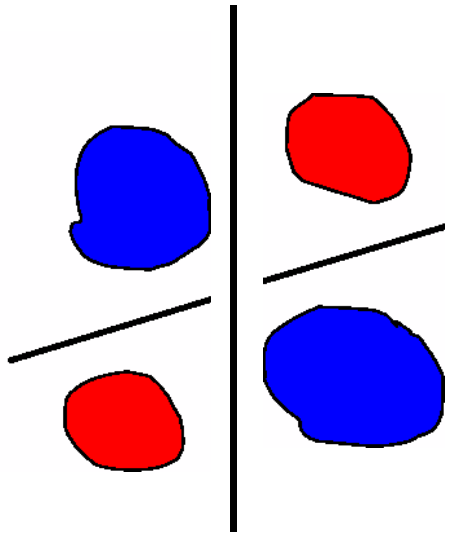
Примери за линейно неразделими класове

Ограничения на перцептрона

- Един перцепртон задава една разделяща хипер-равнина
- Два перцептрона могат да зададат две разделящи хипер-равнини разделящи пространството на четири региона





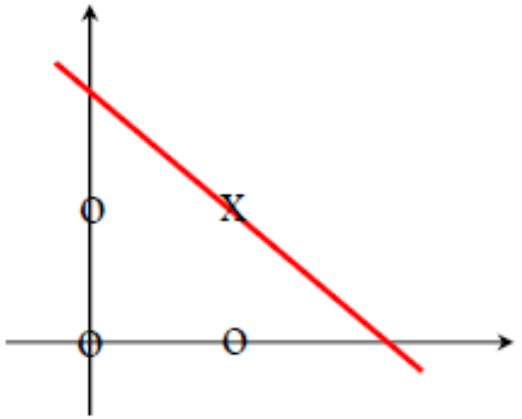


Ограничения на перцептрона

Логическа AND функция (И)

Вх./Изх. образци разделяща равнина

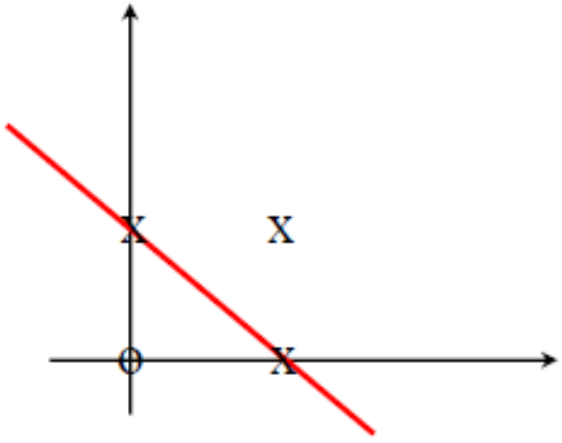
x_1	x_2	y	$w_1 = 1$
0	0	0	$w_2 = 1$
0	1	0	$b = -2$
1	0	0	$\theta = 0$
1	1	1	$-2 + x_1 + x_2 = 0$



Логическа OR функция (ИЛИ)

Вх./Изх. образци разделяща равнина

x_1	x_2	y	$w_1 = 1$
0	0	0	$w_2 = 1$
0	1	1	$b = 1$
1	0	1	$\theta = 0$
1	1	1	$-1 + x_1 + x_2 = 0$



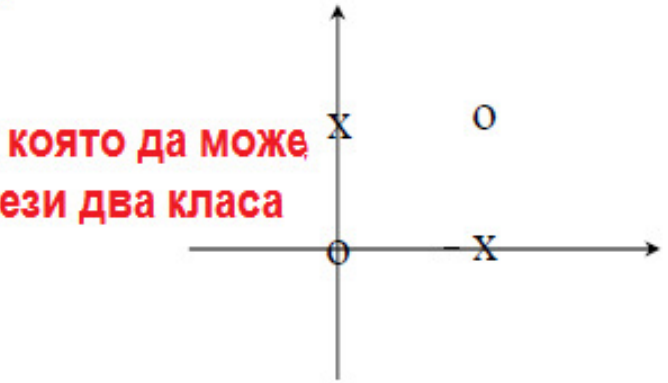
Ограничения на перцептрона

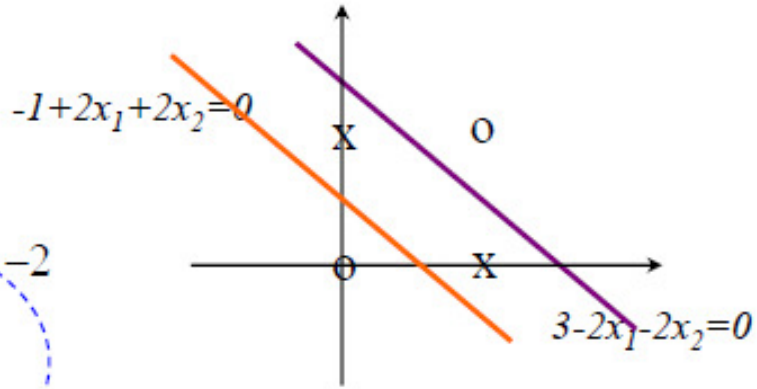
Логическа XOR (изключващо ИЛИ) функция

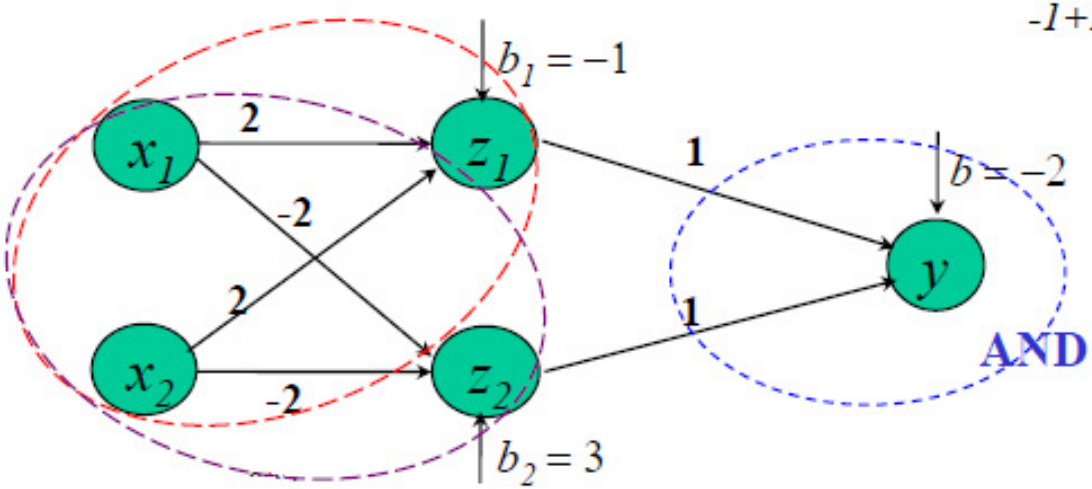
Вх./Изх. образци разделяща равнина

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

Няма линия, която да може да раздели тези два класа







AND neuron