Neuronové sítě

doc. RNDr. PaedDr. Eva Volná, PhD.
eva.volna@osu.cz

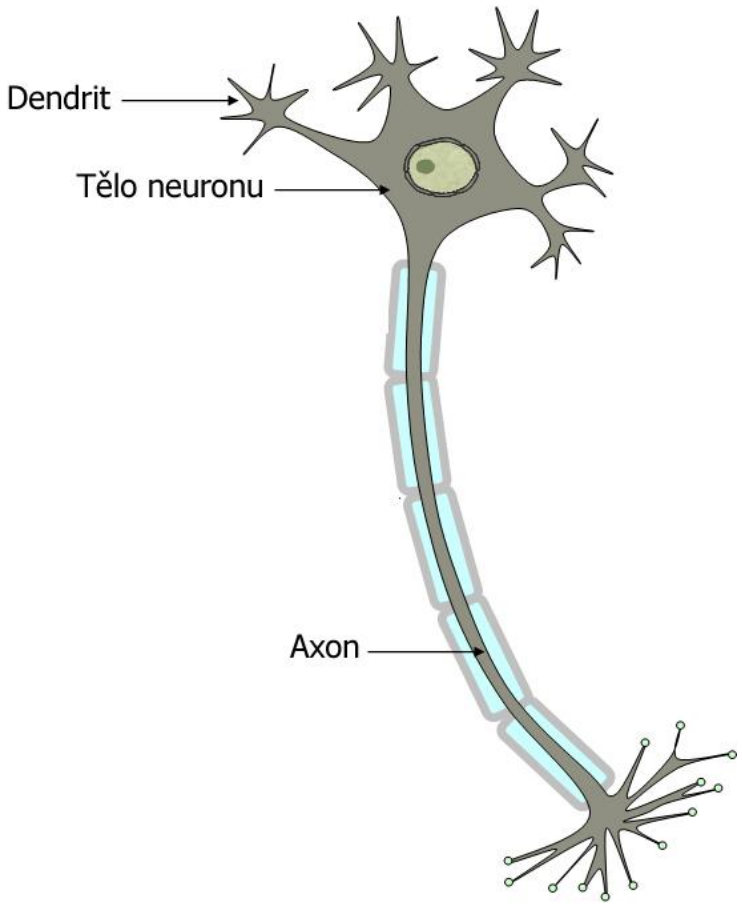
Matematický model neuronu

Neuronová síť

Biologický neuron

- Základní jednotkou mozkové tkáně je neuron.
- Neuronů je v mozku asi kolem 100 miliard.

Struktura neuronu



Každý neuron se skládá ze dvou částí – z **buněčného těla** a **axonu**.

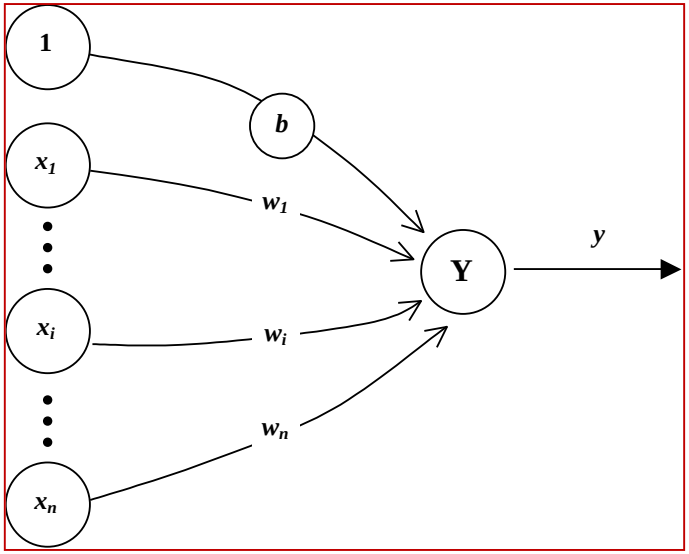
Axon slouží k předávání signálu dalšímu neuronu.

Na těle se nachází speciální výběžky, označované jako **dendrity**, které naopak slouží k příjmu signálů od jiných neuronů.

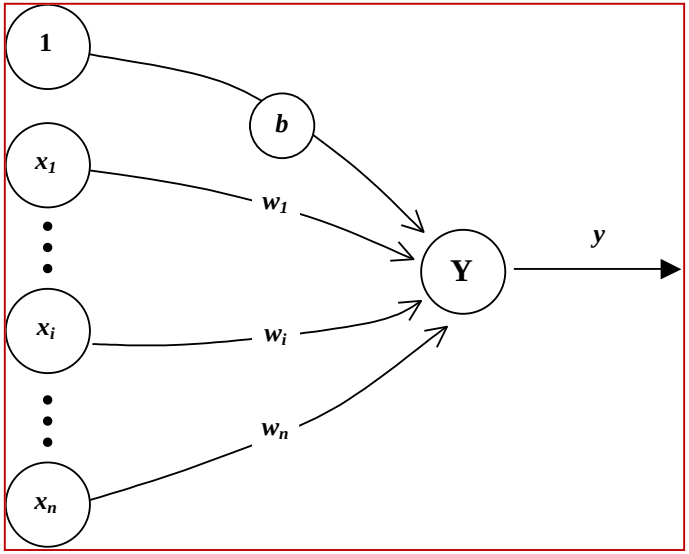
- Neuron přijímá signály prostřednictvím **dendritů**, zpracuje je a pomocí **axonu** je předává dál.
- **Šíření signálu** prostřednictvím neuronů se děje na základě **změn elektrického napětí** na jejich membránách spojených s přečerpáváním elektricky nabitých částic.
- Tyto procesy probíhají takřka bleskovou rychlostí.

Formální neuron

Základem matematického modelu neuronové sítě je formální neuron.



Formální neuron Y má obecně n reálných vstupů, které modelují **dendrity** a určují vstupní vektor $\boldsymbol{x} = (x_1, \ldots, x_n)$.



Tyto vstupy jsou ohodnoceny reálnými **synaptickými váhami** tvořícími vektor $\boldsymbol{w} = (w_1, \ldots , w_n)$.

Ve shodě s neurofyziologickou motivací mohou být synaptické váhy i **záporné**, čímž se vyjadřuje jejich inhibiční charakter.

Vnitřní potenciál neuronu

Vážená suma vstupních hodnot y_in představuje **vnitřní potenciál neuronu** Y:

$$y\_in = \sum_{i=1}^{n} w_i x_i$$

Bias může být do vztahu včleněn přidáním komponent $x_0 = 1$ k vektoru **x**, tj. **x** = (1, x_1, x_2, ..., x_n).

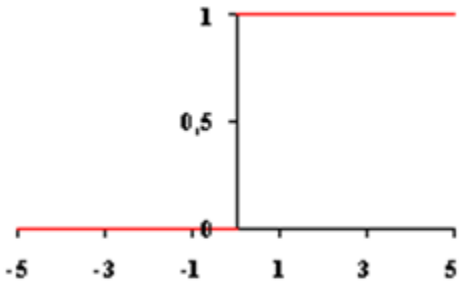
Bias je dále zpracováván jako jakákoliv jiná váha, tj. $w_0 = b$.

Vstup do neuronu Y je pak dán následujícím vztahem:

$$y\_in = \sum_{i=0}^{n} w_i x_i = w_0 + \sum_{i=1}^{n} w_i x_i = b + \sum_{i=1}^{n} w_i x_i$$

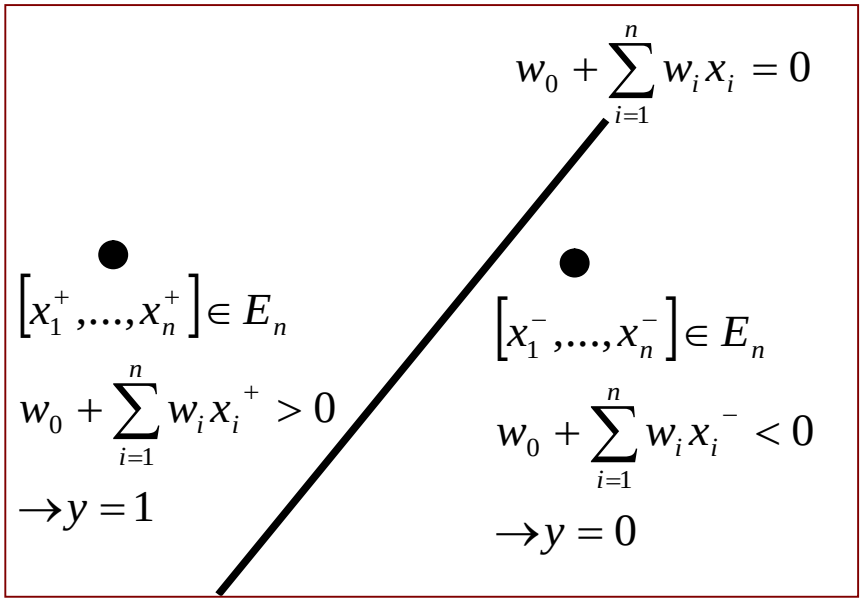
- Hodnota vnitřního potenciálu *y_in* **po dosažení hodnoty** b indukuje výstup (**stav**) y neuronu Y, který **modeluje elektrický impuls axonu**.
- Nelineární nárůst výstupní hodnoty *y* = f(*y_in*) při dosažení hodnoty potenciálu b je dán **aktivační (přenosovou) funkcí** f. Nejjednodušším typem přenosové funkce je ostrá nelinearita, která má pro neuron Y tvar:

$$f(y\_in) = \begin{cases} 1 & \text{pokud} \quad y\_in \geq 0 \\ 0 & \text{pokud} \quad y\_in < 0 \end{cases}$$



Příklad

K lepšímu pochopení funkce jednoho neuronu nám pomůže **geometrická interpretace funkce neuronu.**



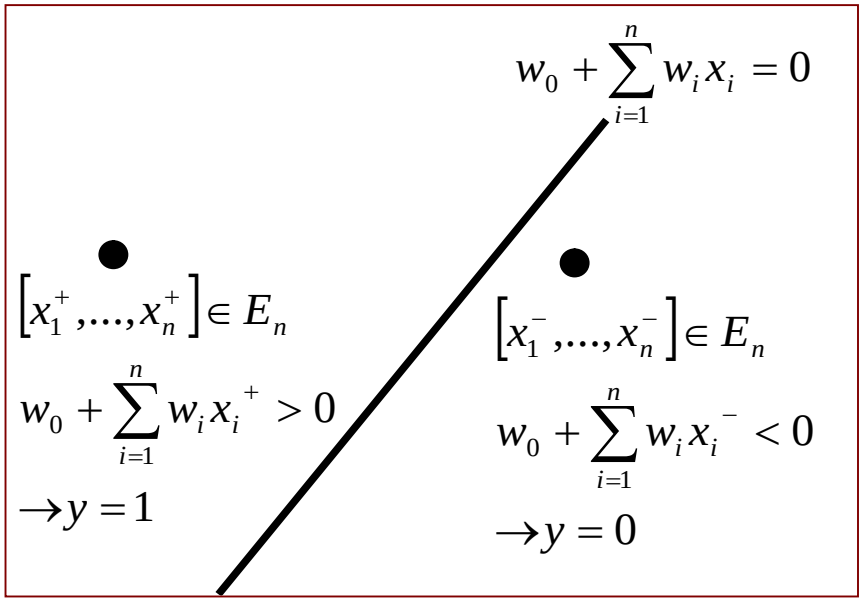
Vstupy neuronu chápeme jako **souřadnice bodu** v n-rozměrném Euklidovském vstupním prostoru E_n. **V tomto prostoru má rovnice nadroviny** tvar:

$$w_0 + \sum_{i=1}^{n} w_i x_i = 0.$$

Tato nadrovina dělí vstupní prostor na dva poloprostory.

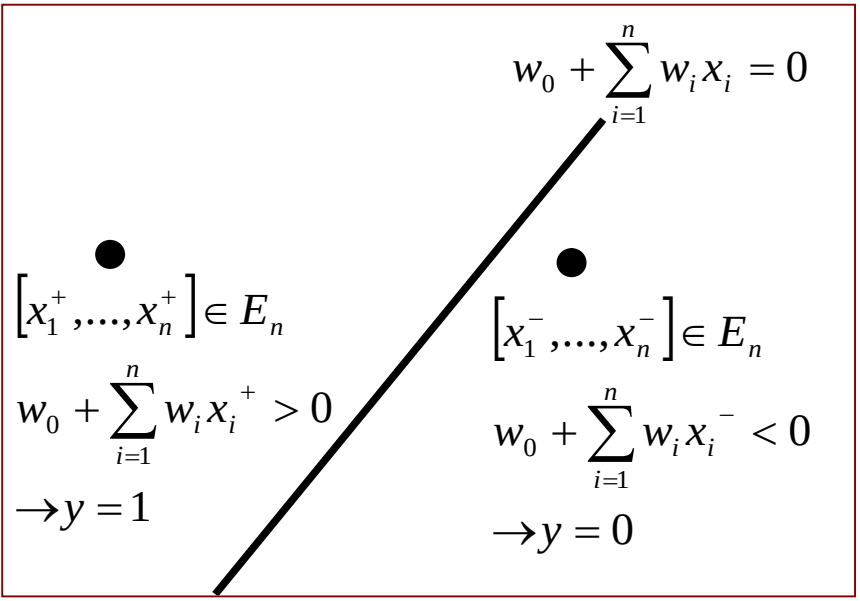
Souřadnice bodů $[x_1^+,...,x_n^+]$, které leží v jednom poloprostoru, splňují nerovnost:

$$w_0 + \sum_{i=1}^{n} w_i x_i^+ > 0$$



Body $[x_1^-,...,x_n^-]$ z druhého poloprostoru pak , splňují nerovnost:

$$w_0 + \sum_{i=1}^{n} w_i x_i^- < 0.$$



Synaptické váhy $\boldsymbol{w} = (w_0, \ldots, w_n)$ chápeme jako **koeficienty této nadroviny**.

Neuron Y tedy *klasifikuje, ve kterém z obou poloprostorů určených nadrovinou leží bod, jehož souřadnice jsou na vstupu*, tj. **realizuje dichotomii vstupního prostoru**.

Neuron Y je **aktivní**, je-li jeho stav $y = 1$ a **pasivní**, pokud je jeho stav $y = 0$.

Neuronová síť

- Každá neuronová síť je složena z formálních neuronů, které jsou vzájemně propojeny tak, že výstup jednoho neuronu je vstupem do (obecně i více) neuronů.
- Počet neuronů a jejich vzájemné propojení v síti určuje *architekturu* (*topologii*) neuronové sítě.
- Z hlediska využití rozlišujeme v síti *vstupní*, *pracovní* (*skryté, mezilehlé, vnitřní*) a *výstupní neurony*.
- Šíření a zpracování informace v síti je umožněno změnou stavů neuronů ležících na cestě mezi vstupními a výstupními neurony.
- ***Stavy*** všech neuronů v síti určují *stav neuronové sítě* a *synaptické váhy* všech spojů představují ***konfiguraci*** *neuronové sítě*.

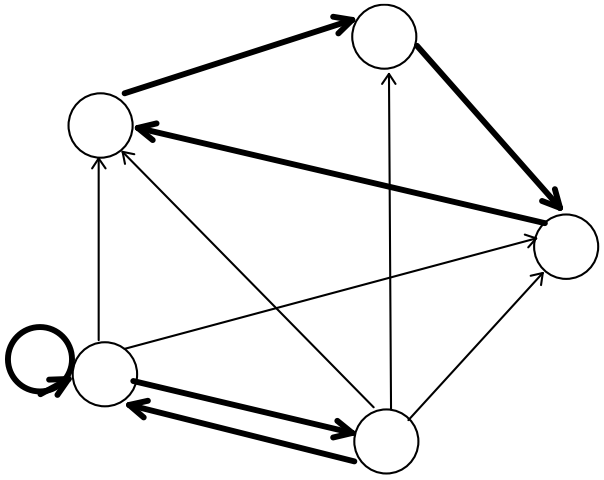
- Neuronová síť se v čase vyvíjí, mění se stav neuronů, adaptují se váhy.
- Celkovou dynamiku neuronové sítě dělíme do tří dynamik, tj. máme tři režimy práce sítě: ***organizační*** (změna topologie), ***aktivní*** (změna stavu) a ***adaptivní*** (změna konfigurace).
- Uvedené dynamiky neuronové sítě jsou obvykle zadány *počátečním stavem* a *matematickou rovnicí*, resp. pravidlem, které určuje vývoj příslušné charakteristiky sítě (topologie, stav, konfigurace) v čase.
- Změny, které se řídí těmito zákonitostmi probíhají v odpovídajících režimech práce neuronové sítě.
- Konkretizací jednotlivých dynamik obdržíme různé modely neuronových sítí vhodné pro řešení různých tříd úloh.

Organizační dynamika

- specifikuje architekturu neuronové sítě.
- Organizační dynamika převážně předpokládá pevnou architekturu neuronové sítě (tj. takovou architekturu, která se již v čase nemění).
- Rozlišujeme dva typy architektury: *cyklická* (*rekurentní*) a *acyklická* (*dopředná*) síť.

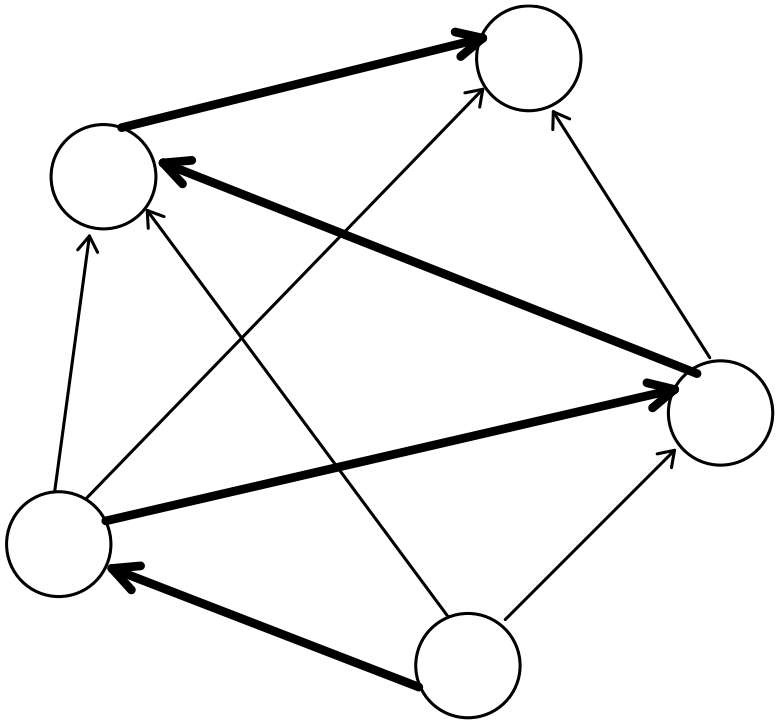
Cyklická topologie neuronové sítě

- V případě ***cyklické*** topologie existuje v síti skupina neuronů, která je spojena v kruhu (tzv. *cyklus*).
- Nejjednodušším příkladem cyklu je *zpětná vazba* neuronu, jehož výstup je zároveň jeho vstupem.
- Nejvíce cyklů je v *úplné topologii* cyklické neuronové sítě, kde výstup libovolného neuronu je vstupem každého neuronu.



Acyklická topologie neuronové sítě

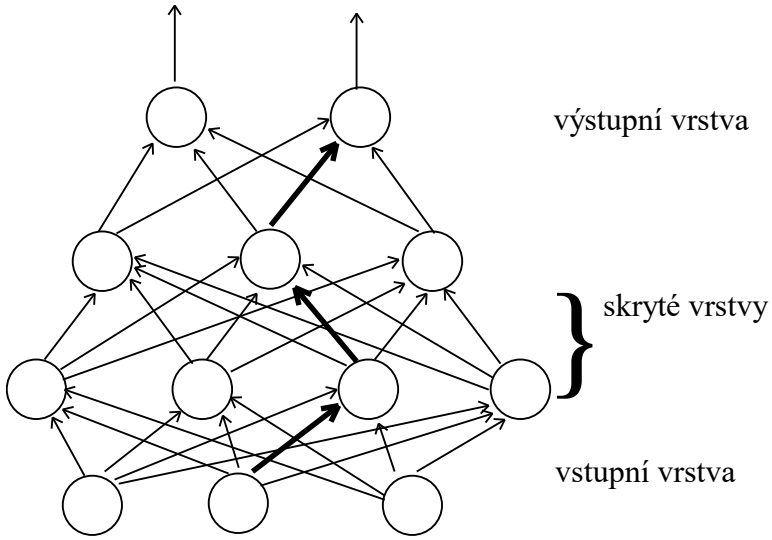
V ***acyklických*** sítích cyklus neexistuje a všechny cesty vedou jedním směrem.



Vícevrstvá neuronová síť

- Speciálním případem topologie acyklické neuronové sítě je ***vícevrstvá neuronová síť***.
- V této síti jsou následující vrstvy neuronů: *vstupní a výstupní* vrstva a *skryté* (*mezilehlé, vnitřní*) vrstvy.
- Neurony jedné vrstvy spojeny se všemi neurony bezprostředně následující vrstvy.

- Topologii sítě lze zadat jen počty neuronů v jednotlivých vrstvách: 3-4-3-2.
- Cesta v síti vede směrem od vstupní vrstvy k výstupní, přičemž obsahuje po jednom neuronu z každé vrstvy.



Aktivní dynamika

- Aktivní dynamika specifikuje počáteční stav sítě a způsob jeho změny v čase při pevné topologii a konfiguraci.
- V aktivním režimu se na začátku nastaví stavy vstupních neuronů a pak probíhá vlastní výpočet.
- Většinou se předpokládá diskrétní čas, tj. na počátku se síť nachází v čase 0 a stav sítě se mění jen v čase 1, 2, 3, ...
- V každém takové časové kroku je podle daného pravidla aktivní dynamiky jsou vybrány neurony, které aktualizují svůj stav na základě svých vstupů.
- Stav výstupních neuronů je výstupem neuronové sítě (tj. výsledkem výpočtu).
- Neuronová síť tak v aktivním režimu realizuje nějakou funkci na vstupním prostoru - neuronová síť se využívá k výpočtům.

Přenosové (aktivační) funkce

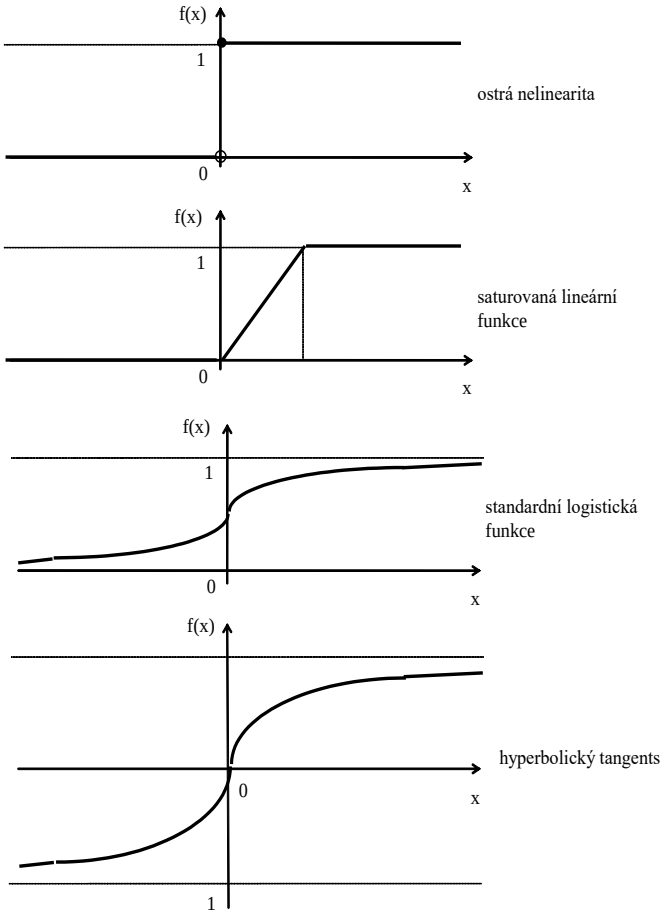
- Aktivní dynamika neuronové sítě také určuje funkci jednoho neuronu, jejíž předpis je většinou pro všechny (nevstupní) neurony v síti stejný (tzv. *homogenní* neuronová síť).
- Příklady *aktivačních funkcí*:

$$f(x) = \begin{cases} 1 & \text{pokud } x \geq 1 \\ 0 & \text{pokud } x < 0 \end{cases} \quad \text{ostrá nelinearita}$$

$$f(x) = \begin{cases} 1 & x \geq 1 \\ x & 0 \leq x \leq 1 \\ 0 & x < 0 \end{cases} \quad \text{saturovaná lineární funkce}$$

$$f(x) = \frac{1}{1+e^{-x}} \quad \text{standardní (logistická) sigmoida}$$

$$f(x) = \frac{1-e^{-x}}{1+e^{-x}} \quad \text{hyperbolický tangens}$$



Adaptivní dynamika

- Adaptivní dynamika neuronové sítě specifikuje *konfiguraci* sítě a způsob, jakým se mění váhové hodnoty v čase.
- V adaptivním režimu se na začátku nastaví váhy v síti na počáteční konfiguraci (např. náhodně).
- Po inicializaci konfigurace sítě probíhá vlastní adaptace.
- Cílem adaptace je nalézt takovou konfiguraci sítě ve váhovém prostoru, která by v aktivním režimu realizovala předepsanou funkci.
- *Jestliže aktivní režim sítě se využívá k vlastnímu výpočtu funkce sítě pro daný vstup, pak adaptivní režim slouží k učení („programování“) této funkce*.

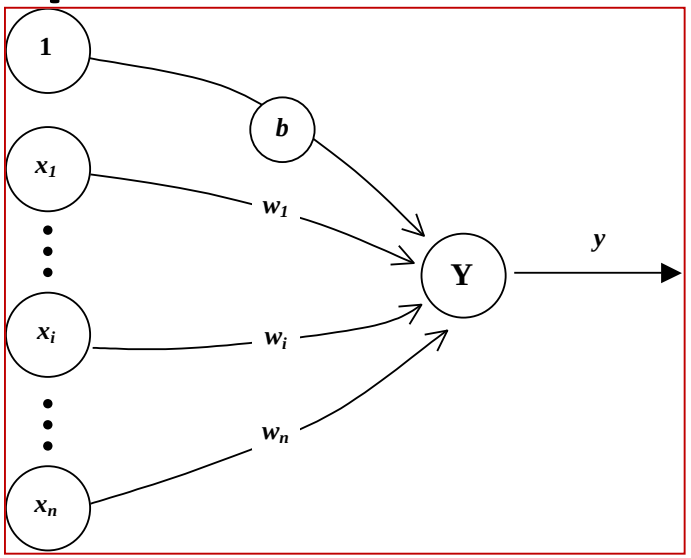
Adaptivní dynamika

- Požadovaná funkce sítě je zadána tzv. *tréninkovou množinou* dvojic vstup/výstup sítě.
- Způsobu adaptace, kdy požadované chování sítě modeluje učitel, který pro vzorové vstupy sítě informuje adaptivní mechanismus o správném výstupu sítě, se nazývá *učení s učitelem* (supervised learning).
- Někdy učitel hodnotí kvalitu momentální skutečné odpovědi (výstupu) sítě pro daný vzorový vstup pomocí známky, která je zadána místo požadované hodnoty výstupu sítě (tzv. *klasifikované učení*).
- Jiným typem adaptace je tzv. *samoorganizace*. V tomto případě tréninková množina obsahuje jen vstupy sítě. To modeluje situaci, kdy není k dispozici učitel, proto se tomuto způsobu adaptace také říká *učení bez učitele*.

Perceptron

Architektura perceptronu

Za typický **perceptron** je považována jednoduchá neuronová síť s *n vstupy* ($x_1, x_2, ..., x_n$) a **jedním pracovním neuronem** spojeným se všemi svými vstupy.



Každému takovému spojení je přiřazena váhová hodnota ($w_1, w_2, ..., w_n$).

Signál přenášený vstupními neurony je buď **binární** (hodnoty 0 nebo 1), nebo **bipolární** (hodnoty -1, 0 nebo 1).

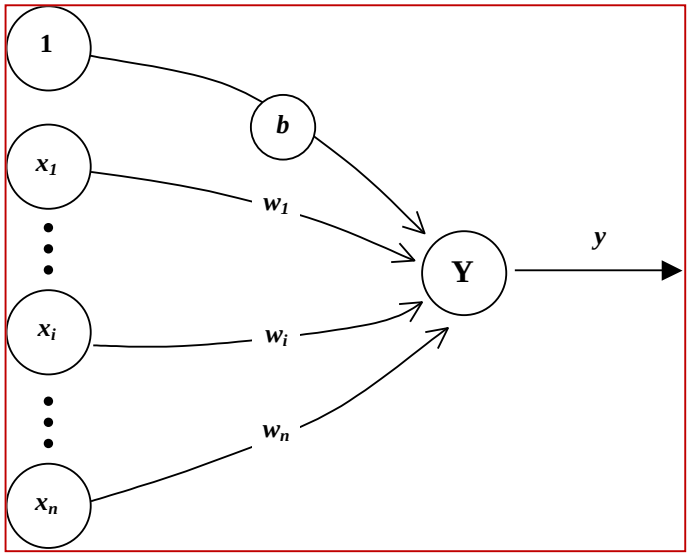
Architektura perceptronu

Vstupem do perceptronu je

$$y\_in = b + \sum_{i=1}^{n} x_i w_i$$

Výstupem z perceptronu je pak

$$y = f(y\_in) = \begin{cases} 1 \; pokud \; y\_in > \theta \\ 0 \; pokud - \theta \le y\_in \le \theta \\ -1 \; pokud \; y\_in < -\theta \end{cases}$$



kde θ je libovolný, ale **pevný práh** aktivační funkce *f*.

Princip adaptačního pravidla perceptronu

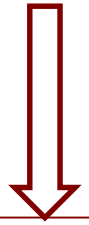
- Váhové hodnoty jsou adaptovány podle adaptačního pravidla perceptronu tak, aby **diference mezi skutečným a požadovaným výstupem byla co nejmenší**.
- To je princip **učení s učitelem**.
- Adaptační pravidlo perceptronu je mnohem silnější než Hebbovo adaptační pravidlo.

Adaptační algoritmus perceptronu

Krok 0. Inicializace vah w_i *(i = 1* až *n)* a biasu *b malými náhodnými čísly.*

Přiřazení inicializační hodnoty koeficientu učení $\alpha\ (0 < \alpha \leq 1)$.

Krok 1. Dokud není splněna **podmínka ukončení výpočtu**, opakovat kroky (2 až 6).



Podmínka ukončení:
jestliže ve *2. kroku* již nenastává žádná změna váhových hodnot, stop; jinak, pokračovat.

Adaptační algoritmus perceptronu

Krok 2. Pro každý tréninkový pár **s:**t

(tj. vstupní vektor **s** a příslušný výstup t**)**,

provádět kroky (3 až 5).

Krok 3. Aktivuj vstupní neurony:

$x_i = s_i$.

Krok 4 Vypočítej skutečnou hodnotu na vý

$$y\_in = b + \sum_i x_i w_i ;.$$

$$y = \begin{cases} 1 & pokud\ y\_in > \theta \\ 0 & pokud - \theta \le y\_in \le \theta \\ -1 & pokud\ y\_in < -\theta \end{cases}$$

Adaptační algoritmus perceptronu

Krok 5 Aktualizuj váhové hodnoty a bias pro daný vzor

jestliže $y \neq t$,

$$w_i(new) = w_i(old) + \alpha\, t\, x_i \quad (i = 1 \text{ až } n).$$

$$b(new) = b(old) + \alpha\, t.$$

jinak

$$w_i(new) = w_i(old)$$

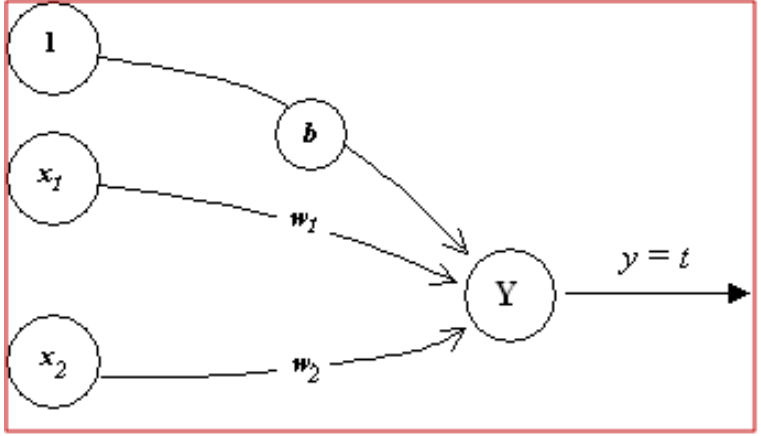
$$b(new) = b(old)$$

Krok 6. Podmínka ukončení.

Adaptační algoritmus perceptronu
SHRNUTÍ

- **Aktualizaci** podléhají pouze ty váhové hodnoty, které **neprodukují požadovaný výstup** *y*.
- To znamená, že čím více tréninkových vzorů má korektní výstupy, tím méně je potřeba času k jejich tréninku.
- Práh aktivační funkce θ je pevná nezáporná hodnota.
- Tvar aktivační funkce je takový, že umožňuje vznik **pásu pevné šířky** (určené hodnotou prahu θ) oddělujícího oblast **pozitivní** odezvy od oblasti **negativní** odezvy na vstupní signál.

Příklad:



Adaptační algoritmus perceptronu pro logickou funkci **AND**: **binární vstupní** hodnoty, **bipolární výstupní** hodnoty. Pro jednoduchost předpokládejme, že práh ***θ = 0,2*** a koeficient učení ***α = 1.***

$$y\_in = b + \sum_i x_i w_i ;$$

$$y = \begin{cases} 1 & pokud\ y\_in > \theta \\ 0 & pokud\ -\theta \le y\_in \le \theta \\ -1 & pokud\ y\_in < -\theta \end{cases}$$

jestliže $y \neq t$**:**
$w_i(new) = w_i(old) + \alpha\, t\, x_i$,
$b(new) = b(old) + \alpha\, t.$

čas	VSTUP		VÝSTUP			PŘÍRUSTKY VAH			VÁHOVÉ HODNOTY		
	x_1	x_2	y_in	y	t	Δw_1	Δw_2	Δb	w_1	w_2	b
0									0	0	0
1	1	1	0	0	1	1	1	1	1	1	1
2	1	0	2	1	-1	-1	0	-1	0	1	0
3	0	1	1	1	-1	0	-1	-1	0	0	-1
4	0	0	-1	-1	-1	1	0	0	0	0	-1

Postup řešení:

čas	VSTUP		VÝSTUP			PŘÍRUSTKY VAH			VÁHOVÉ HODNOTY		
	x_1	x_2	y_in	y	t	Δw_1	Δw_2	Δb	w_1	w_2	b
0									0	0	0
1	1	1	0	0	1	1	1	1	1	1	1
2	1	0	2	1	-1	-1	0	-1	0	1	0
3	0	1	1	1	-1	0	-1	-1	0	0	-1
4	0	0	-1	-1	-1	1	0	0	0	0	-1

$$y\_in = b + \sum_i x_i w_i; \quad y\_in = 0 + 1*0 + 1*0 = 0$$

$$y = \begin{cases} 1 & pokud y\_in > 0.2 \\ 0 & pokud -0.2 \le y\_in \le 0.2 \\ -1 & pokud y\_in < -0.2 \end{cases}$$

$$y = 0$$

jestliže $y \neq t$:

$w_i(new) = w_i(old) + \alpha\, t\, x_i$,

$b(new) = b(old) + \alpha\, t.$

$w_1(new) = 0 + 1*1*1 = 1$

$w_2(new) = 0 + 1*1*1 = 1$

$b(new) = 0 + 1*1 = 1$

čas	VSTUP		VÝSTUP			PŘÍRUSTKY VAH			VÁHOVÉ HODNOTY		
	x_1	x_2	y_in	y	t	Δw_1	Δw_2	Δb	w_1	w_2	b
0									0	0	0
1	1	1	0	0	1	1	1	1	1	1	1
2	1	0	2	1	-1	-1	0	-1	0	1	0
3	0	1	1	1	-1	0	-1	-1	0	0	-1
4	0	0	-1	-1	-1	1	0	0	0	0	-1

$$y\_in = b + \sum_i x_i w_i; \quad y\_in = 1 + 1*1 + 0*1 = 2$$

$$y = \begin{cases} 1 & pokud\, y\_in > 0.2 \\ 0 & pokud -0.2 \le y\_in \le 0.2 \\ -1 & pokud\, y\_in < -0.2 \end{cases}$$

$$y = 1$$

jestliže $y \neq t$:

$w_i(new) = w_i(old) + \alpha\, t\, x_i$,

$b(new) = b(old) + \alpha\, t$.

$w_1(new) = 1 - 1*1*1 = 0$

$w_2(new) = 1 - 1*1*0 = 1$

$b(new) = 1 - 1*1 = 0$

čas	VSTUP		VÝSTUP			PŘÍRUSTKY VAH			VÁHOVÉ HODNOTY		
	x_1	x_2	y_in	y	t	Δw_1	Δw_2	Δb	w_1	w_2	b
0									0	0	0
1	1	1	0	0	1	1	1	1	1	1	1
2	1	0	2	1	-1	-1	0	-1	0	1	0
3	0	1	1	1	-1	0	-1	-1	0	0	-1
4	0	0	-1	-1	-1	1	0	0	0	0	-1

$$y\_in = b + \sum_i x_i w_i; \quad y\_in = 0 + 0*0 + 1*1 = 1$$

$$y = \begin{cases} 1 & pokud\, y\_in > 0.2 \\ 0 & pokud -0.2 \le y\_in \le 0.2 \\ -1 & pokud\, y\_in < -0.2 \end{cases}$$

$$y = 1$$

jestliže $y \neq t$**:**

$w_i(new) = w_i(old) + \alpha\, t\, x_i$,

$b(new) = b(old) + \alpha\, t.$

$w_1(new) = 0 - 1*1*0 = 0$

$w_2(new) = 1 - 1*1*1 = 0$

$b(new) = 0 - 1*1 = -1$

čas	VSTUP		VÝSTUP			PŘÍRUSTKY VAH			VÁHOVÉ HODNOTY		
	x_1	x_2	y_in	y	t	Δw_1	Δw_2	Δb	w_1	w_2	b
0									0	0	0
1	1	1	0	0	1	1	1	1	1	1	1
2	1	0	2	1	-1	-1	0	-1	0	1	0
3	0	1	1	1	-1	0	-1	-1	0	0	-1
4	0	0	-1	-1	-1	1	0	0	0	0	-1

$$y\_in = b + \sum_i x_i w_i; \quad y\_in = -1 + 0*0 + 0*0 = -1$$

$$y = \begin{cases} 1 & pokud\, y\_in > 0.2 \\ 0 & pokud -0.2 \le y\_in \le 0.2 \\ -1 & pokud\, y\_in < -0.2 \end{cases}$$

$$y = -1$$

jestliže $y \neq t$**:**

$w_i(new) = w_i(old) + \alpha\, t\, x_i$,

$b(new) = b(old) + \alpha\, t.$

$w_1(new) = 0$

$w_2(new) = 0$

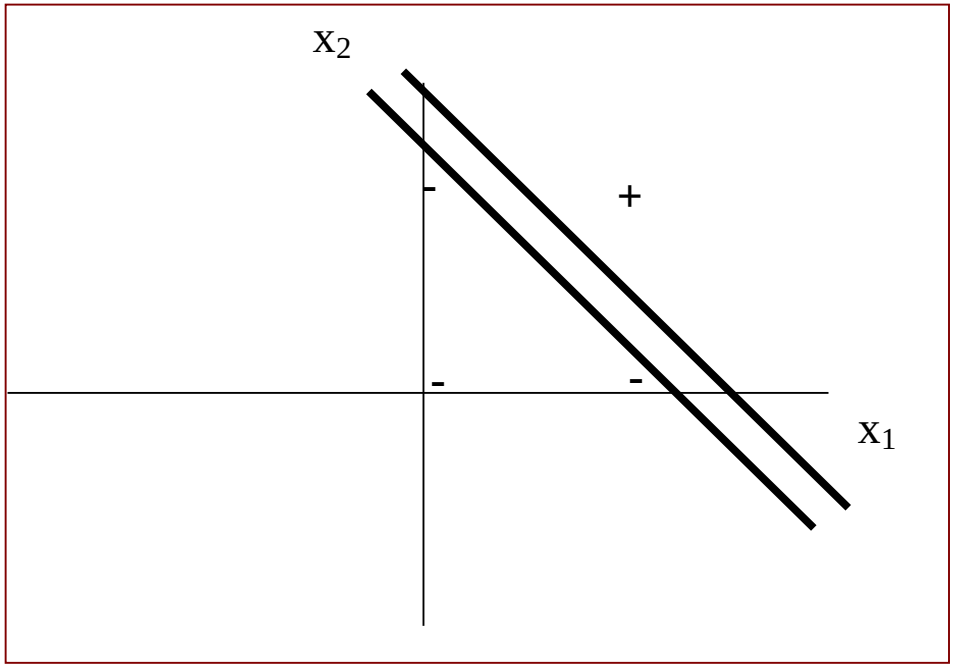
$b(new) = -1$

Postup řešení:

čas	**VSTUP**		**VÝSTUP**			**PŘÍRUSTKY VAH**			**VÁHOVÉ HODNOTY**		
	x_1	x_2	y_in	y	t	Δw_1	Δw_2	Δb	w_1	w_2	b
0									0	0	0
1	1	1	0	0	1	1	1	1	1	1	1
2	1	0	2	1	-1	-1	0	-1	0	1	0
3	0	1	1	1	-1	0	-1	-1	0	0	-1
4	0	0	-1	-1	-1	1	0	0	0	0	-1
...											
37	1	1	1	1	1	0	0	0	2	3	-4
38	1	0	-2	-1	-1	0	0	0	2	3	-4
39	0	1	-1	-1	-1	0	0	0	2	3	-4
40	0	0	-4	-1	-1	0	0	0	2	3	-4

Hraniční pás pro logickou funkci AND po adaptaci algoritmem perceptronu.

čas	VSTUP		VÝSTUP			PŘÍRUSTKY VAH			VÁHOVÉ HODNOTY		
	x_1	x_2	y_in	y	t	Δw_1	Δw_2	Δb	w_1	w_2	b
					...						
40	0	0	-4	-1	-1	0	0	0	2	3	-4

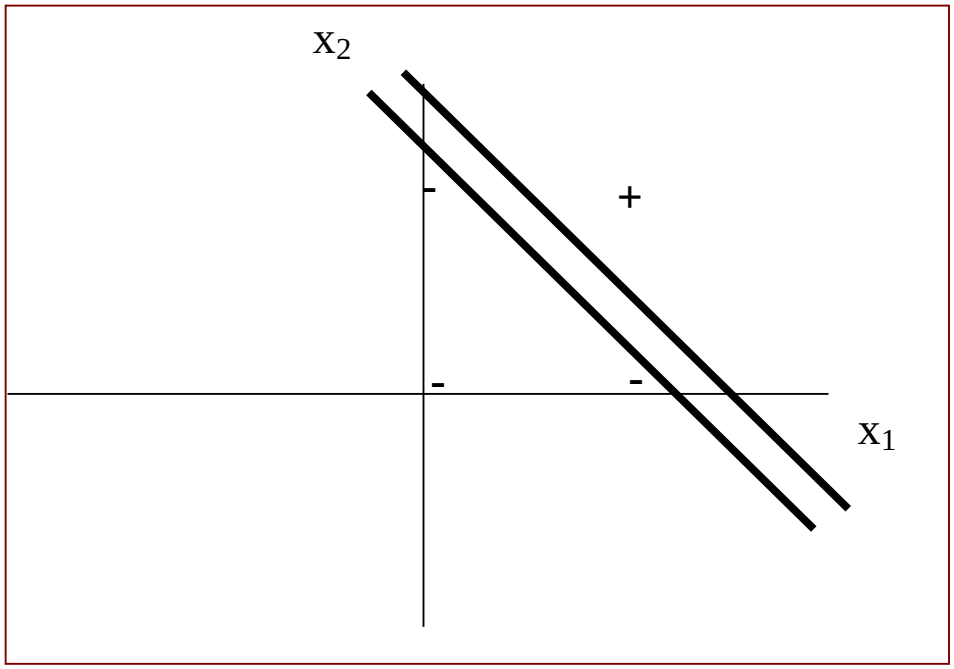


*Oblast **kladné** odezvy* je dána body, pro které platí: $2x_1 + 3x_2 - 4 > 0{,}2$
a hraniční přímka této oblasti má tvar:

$$x_2 = -\frac{2}{3}x_1 + \frac{7}{5}$$

Hraniční pás pro logickou funkci AND po adaptaci algoritmem perceptronu.

čas	**VSTUP**		**VÝSTUP**			**PŘÍRUSTKY VAH**			**VÁHOVÉ HODNOTY**		
	x_1	x_2	y_in	y	t	Δw_1	Δw_2	Δb	w_1	w_2	b
					...						
40	0	0	-4	-1	-1	0	0	0	2	3	-4



*Oblast **záporné** odezvy* je dána body, pro které platí: $2x_1 + 3x_2 - 4 < -0{,}2$
a hraniční přímka této oblasti má tvar:

$$x_2 = -\frac{2}{3}x_1 + \frac{19}{15}$$

Adaline

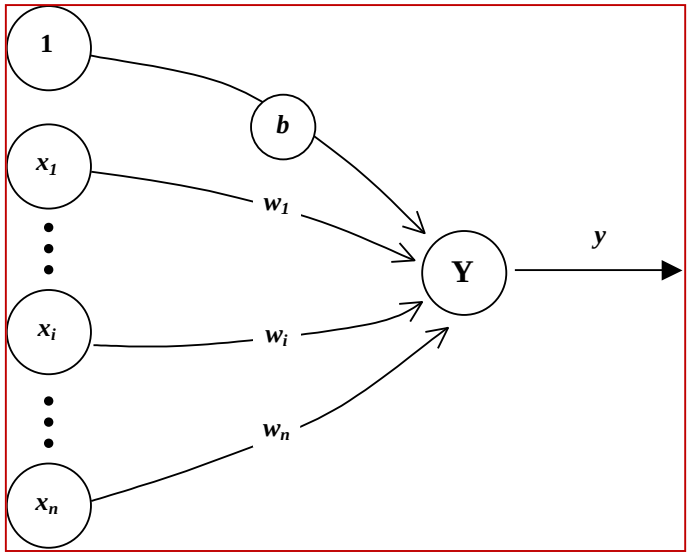
Architektura Adaline

Adaline, tj. **Adaptive Linear Neuron**.

Adaline je jednoduchá neuronová síť s *n vstupy* ($x_1, x_2, ..., x_n$) a **jedním pracovním neuronem** spojeným se všemi svými vstupy. Každému takovému spojení je přiřazena váhová hodnota ($w_1, w_2, ..., w_n$).

Výstupem z Adaline je

$$y = y\_in = b + \sum_{i=1}^{n} x_i w_i$$

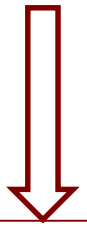


Adaptační algoritmus Adaline

Krok 0. Inicializace vah malými náhodnými hodnotami.

Přiřazení inicializační hodnoty koeficientu učení α

Krok 1. Dokud není splněna **podmínka ukončení výpočtu**, opakovat kroky (2 až 6).



Podmínka ukončení:
jestliže největší změna váhových hodnot je menší než maximální povolená chyba, stop;
jinak, pokračovat.

Adaptační algoritmus Adaline

Krok 2. Pro každý bipolární tréninkový pár **s:**t (tj. vstupní vektor **s** a příslušný výstup t**)**, provádět kroky (3 až 5).

Krok 3. Aktivovat vstupní neurony:

$x_i = s_i$.

Krok 4 Vypočítat skutečnou hodnotu na výstupu:

$$y\_in = b + \sum_i x_i w_i \;;.$$

$y = y\_in$.

Adaptační algoritmus Adaline

Krok 5 Aktualizovat váhové hodnoty a

i = 1, ..., n:

$w_i(new) = w_i(old) + \alpha\,(t - y\_in)\,x_i.$

$b(new) = b(old) + \alpha\,(t - y\_in).$

Krok 6. Podmínka ukončení:

jestliže největší změna váhových hodnot je menší než maximální povolená chyba, stop;

jinak, pokračovat.

Geometrický význam funkce Adaline

Funkce **Adaline** se nepatrně liší od perceptronu.

Uvažujme vstup $\boldsymbol{x}=(x_1, \ldots, x_n)$, tj. bod $[x_1, \ldots, x_n]$ v n-rozměrném vstupním prostoru. Nadrovina s koeficienty $\boldsymbol{w}$ pro daný neuron Adaline určená rovnicí

$$w_0 + \sum_{i=1}^{n} w_i x_i = 0$$

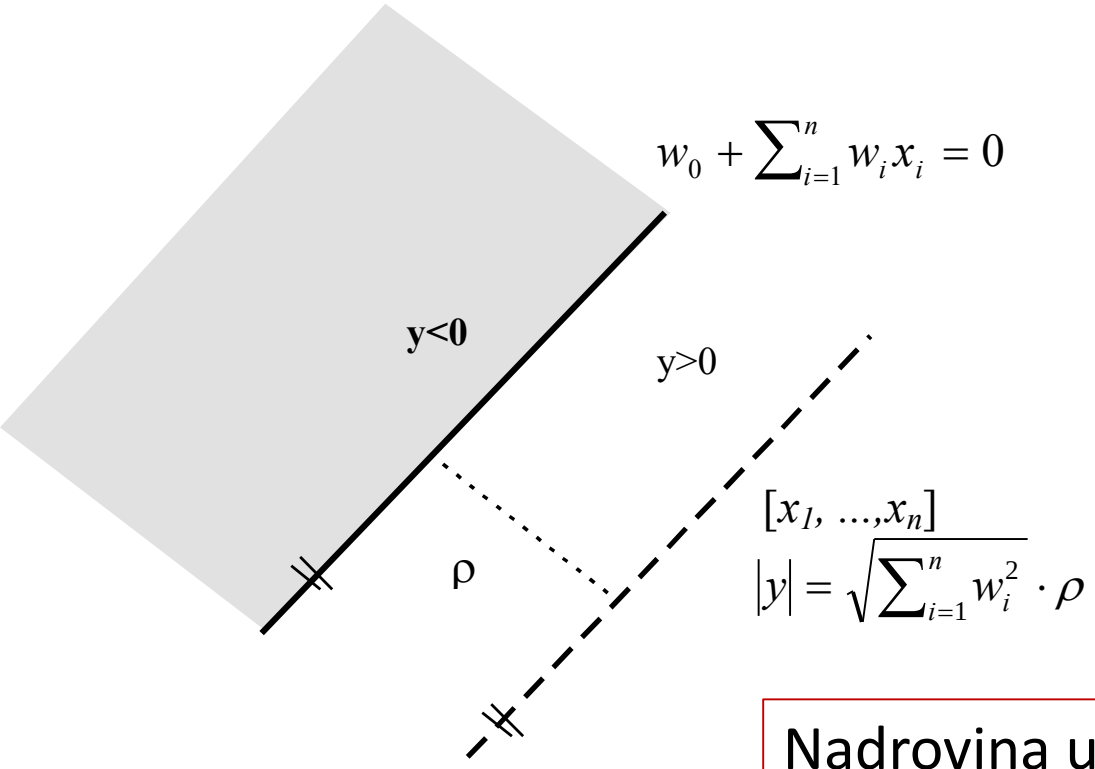
rozděluje tento prostor na dva poloprostory, ve kterých má hodnota výstupu y zapsaného rovnicí

$$y = \sum_{i=1}^{n} w_i x_i$$

odlišné znaménko (tj. je buď kladná, nebo záporná).

Geometrický význam funkce Adaline

Absolutní hodnota výstupu z neuronu Adaline závisí lineárně na **vzdálenosti vstupu *x* od nadroviny** ve vstupním prostoru.



Body ze vstupního prostoru, které mají stejný výstup, leží na jedné nadrovině rovnoběžné se separující nadrovinou.

Nadrovina určená stejným výstupem je znázorněna přerušovanou čarou.

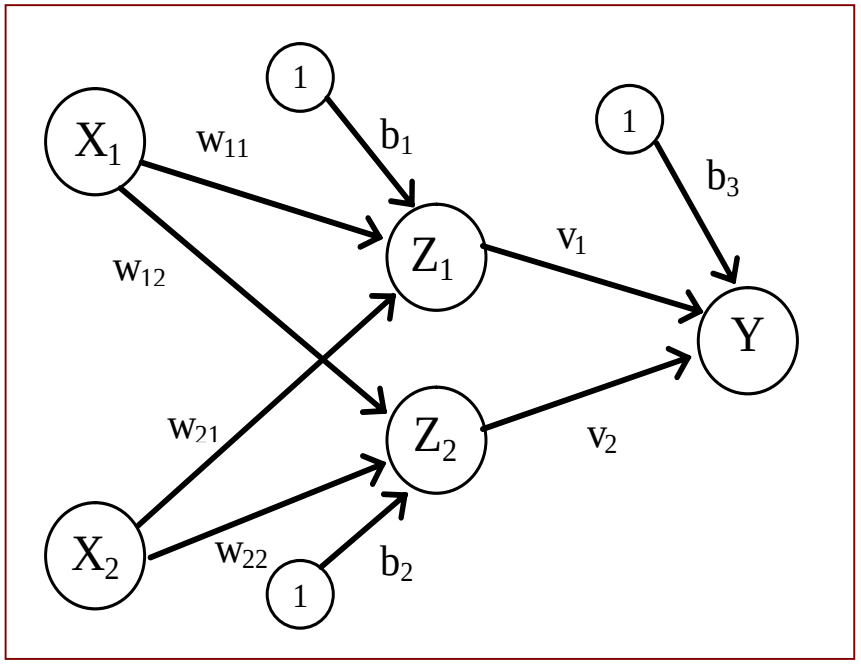
Madaline

Architektura Madaline

Madaline, tj. *Many Adaptive Linear Neurons*.
Stavebním kamenem v tomto modelu je neuron **Adaline**.

Výstupy (z_1 a z_2) z obou skrytých neuronů typu Adaline (Z_1 a Z_2), jsou určeny signály (x_1 a x_2) vycházejícími z neuronů X_1 a X_2, které závisí na příslušné prahové funkci.

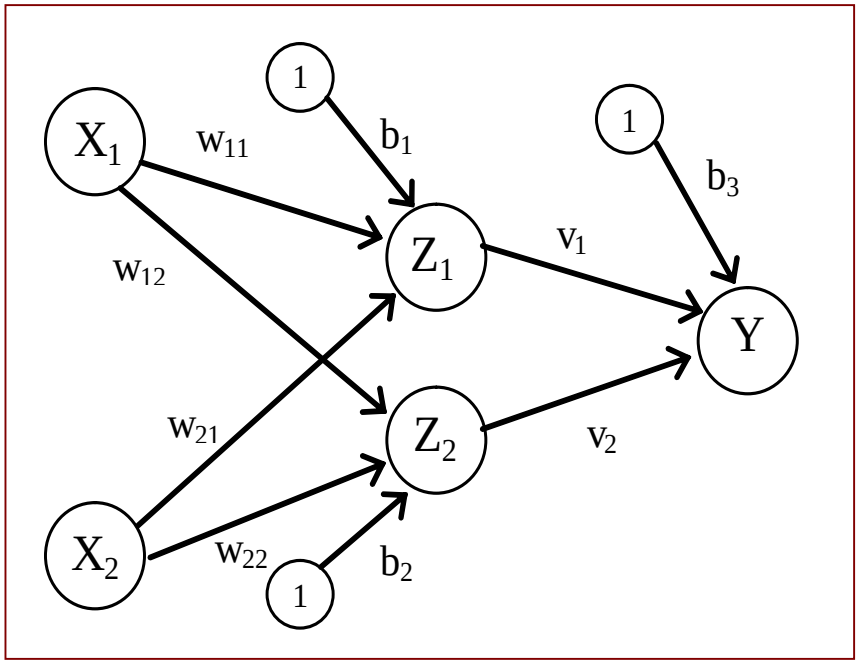
Pak i **skutečný výstup** y je nelineární funkcí vstupního vektoru (x_1, x_2) a příslušné prahové funkce.



Použití skrytých neuronů Z_1 a Z_2 sice dává síti větší výpočtové možnosti, ale naproti tomu komplikuje adaptační proces.

Adaptační algoritmy MRI a MRII (Madaline rules)

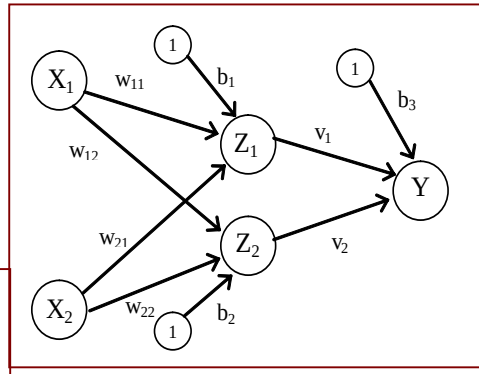
- Adaptační algoritmus MRI **adaptuje** pouze váhové hodnoty příslušející oběma **skrytým neuronům**, zatímco váhové hodnoty příslušející **výstupnímu neuronu** jsou **fixní**.
- Adaptační algoritmus MRII upravuje **všechny váhové hodnoty**.



Princip algoritmu MRI

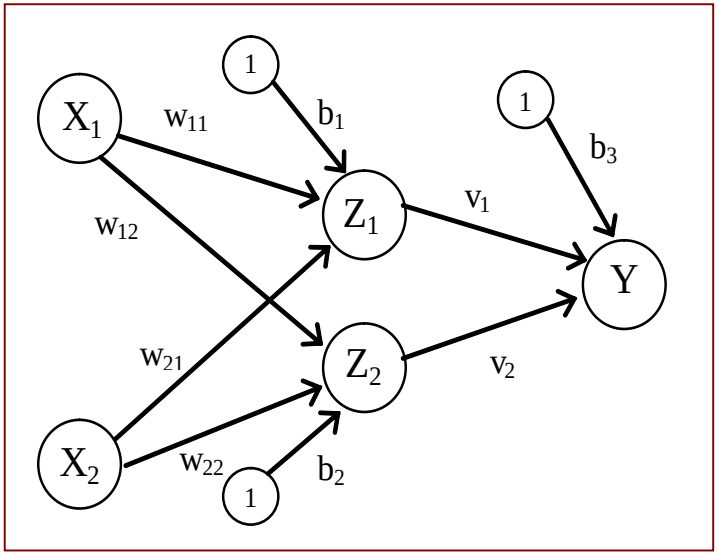
- Váhové hodnoty v_1 a v_2 a bias b_3, příslušející **výstupnímu neuronu** *Y*, jsou určeny tak, že **výstupní signál** z *Y* je roven *1*, **pokud je alespoň jedna** hodnota signálu vycházejícího ze **skrytých neuronů** (tj. Z_1 a Z_2 nebo obou z nich) **rovna jedné**.
- Pokud jsou **oba signály** vysílané ze Z_1 i Z_2 rovny -1, má **výstupní signál** z *Y* hodnotu *-1*.
- Jinými slovy, **výstupní neuron** *Y* **provádí logickou funkci „OR“** na signálech vysílaných z neuronů Z_1 a Z_2. Můžeme tedy přiřadit

$$v_1 = \frac{1}{2}, \quad v_2 = \frac{1}{2}, \quad b_3 = \frac{1}{2}.$$



Princip algoritmu MRI

- Váhové hodnoty příslušející **prvnímu skrytému neuronu** Adaline (w_{11} a w_{21}) a váhové hodnoty příslušející **druhému skrytému neuronu** Adaline (w_{12} a w_{22}) jsou adaptovány podle algoritmu MRI následovně:
- **Aktivační funkce** pro Z_1, Z_2 a Y je dána vztahem:



$$f(x) = \begin{cases} 1 & \text{pokud } x \geq 0; \\ -1 & \text{pokud } x < 0. \end{cases}$$

Adaptační algoritmus MRI

Krok 0. Inicializace váhových hodnot v_1 a v_2 a biasu b_3 .

$v_1 = 0.5$

$v_2 = 0.5$

$b_3 = 0.5$

Inicializace zbývajících vah malými náhodnými hodnotami.

Přiřazení inicializační hodnoty koeficientu učení α..

Krok 1. Dokud není splněna **podmínka ukončení výpočtu**, opakovat kroky (2 až 8).

Podmínka ukončení:

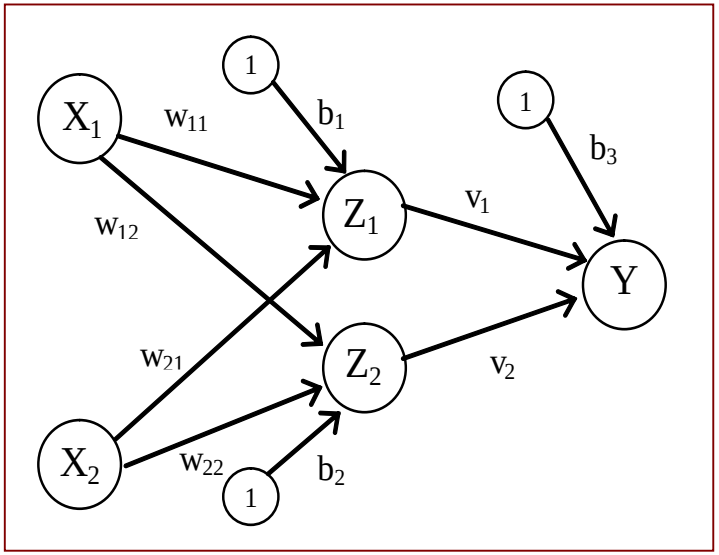
pokud již *nenastávají žádné změny* váhových hodnot nebo pokud již bylo vykonáno *maximálně definované množství* váhových změn, *stop*; jinak, *pokračovat.*

Adaptační algoritmus MRI

Krok 2. Pro každý tréninkový pár **s:t** provádět kroky (3 až 7).

Krok 3. Aktivovat vstupní neurony:

$x_i = s_i$.



Krok 4 Vypočítat vstupní hodnoty skrytých neuronů:

$$z\_in_1 = b_1 + x_1 w_{11} + x_2 w_{21},$$
$$z\_in_2 = b_2 + x_1 w_{12} + x_2 w_{22}.$$

Krok 5 Stanovení výstupních hodnot skrytých neuronů:

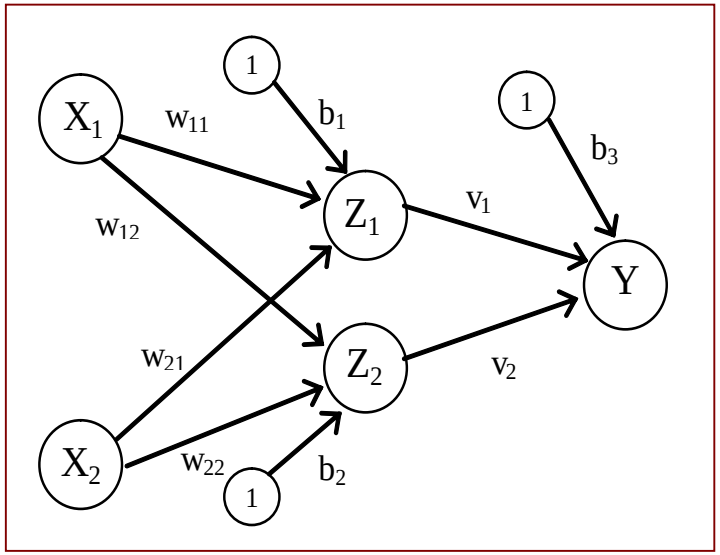
$$z_1 = f\left(z\_in_1\right),$$
$$z_2 = f\left(z\_in_2\right).$$

Adaptační algoritmus MRI

Krok 6 Stanovení skutečné výstupní hodnoty signálu neuronové sítě Madaline:

$$y\_in = b_3 + z_1 v_1 + z_2 v_2;$$

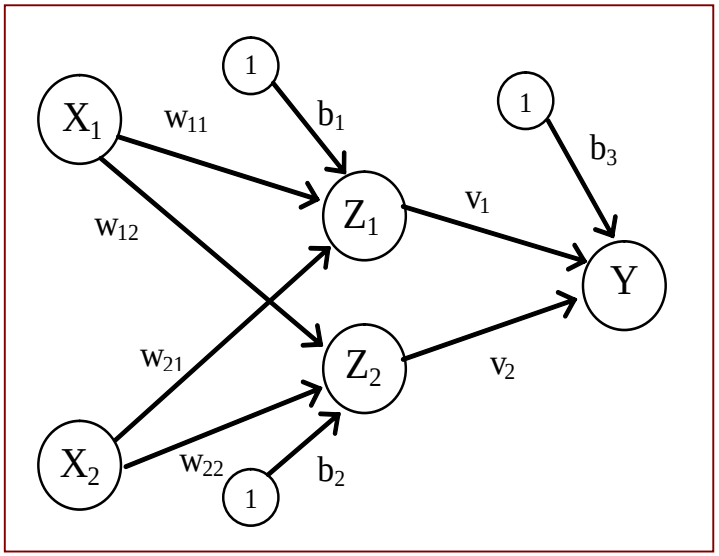
$$y = f\left(y\_in\right).$$



$$v_1 = \frac{1}{2}, \quad v_2 = \frac{1}{2}, \quad b_3 = \frac{1}{2}.$$

Adaptační algoritmus MRI

Krok 7 Aktualizovat váhové hodnoty:



Pokud je $y = t$, nenastávají žádné změny.

Jinak (pro $y \neq t$):

Je-li $t = 1$, potom pro váhové hodnoty na spojeních vedoucích k Z_J ($J=1,2$) platí:

$$w_{iJ}(new) = w_{iJ}(old) + \alpha (1 - z\_in_J)\, x_i.$$

$$b_J(new) = b_J(old) + \alpha (1 - z\_in_J).$$

Je-li $t = -1$, potom pro váhové hodnoty na spojeních vedoucích k Z_K ($K=1,2$) platí:

$$w_{iK}(new) = w_{iK}(old) + \alpha (-1 - z\_in_K)\, x_i.$$

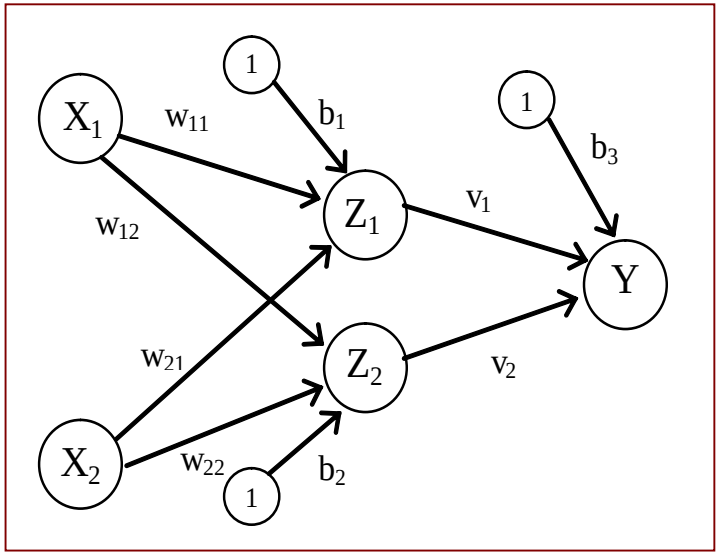
$$b_K(new) = b_K(old) + \alpha (-1 - z\_in_K).$$

Krok 8. Podmínka ukončení

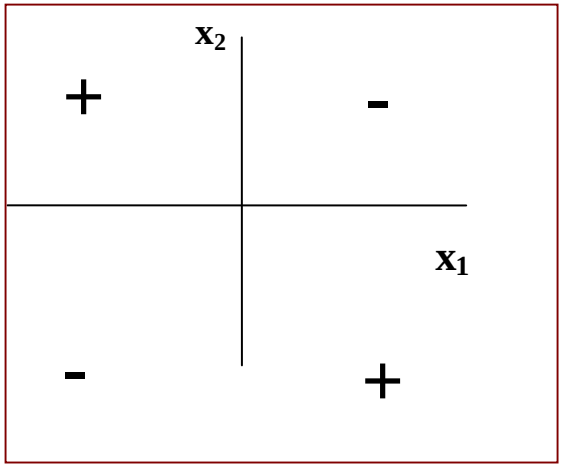
Příklad:

Adaptační algoritmus MRI pro logickou funkci XOR pro **bipolární vstupní i výstupní hodnoty**.

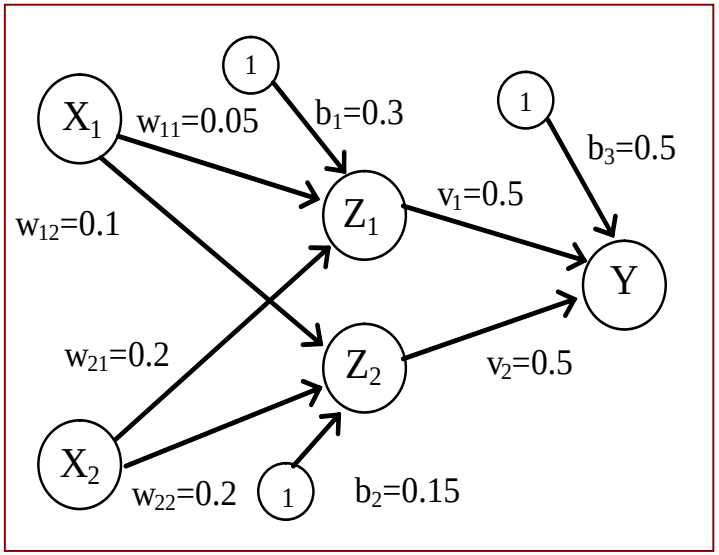
V tabulce je uvedena **trénovací množina.**



VSTUP		POŽADOVANÝ
x_1	x_2	VÝSTUP
1	1	-1
1	-1	1
-1	1	1
-1	-1	-1



Postup řešení:

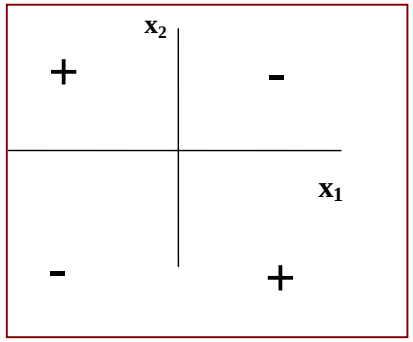


Krok 0.

α. = 0.5;

Inicializace váhových hodnot

váhy vedoucí do Z_1			**váhy vedoucí do Z_2**			**váhy vedoucí do Y**		
w_{11}	w_{21}	b_1	w_{12}	w_{22}	b_2	v_1	v_2	b_3
0,05	0,2	0,3	0,1	0,2	0,15	0,5	0,5	0,5



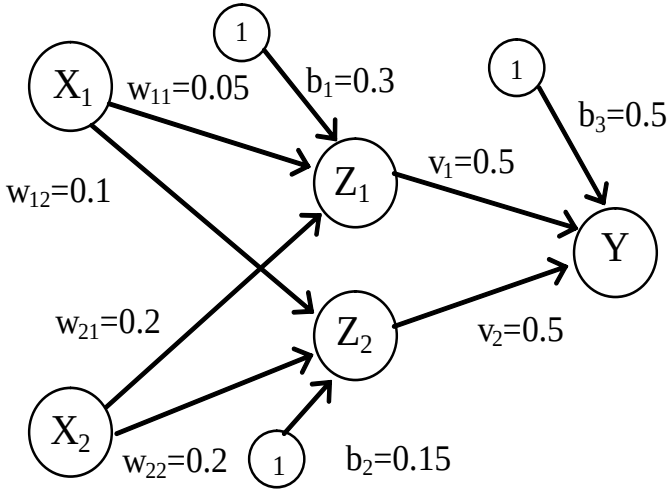
VSTUP		POŽADOVANÝ
x_1	x_2	VÝSTUP
1	1	-1
1	-1	1
-1	1	1
-1	-1	-1

$x_1 = 1, x_2 = 1, t = -1.$

$z\_in_1 = 0.3 + 0.05*1 + 0.2*1 = 0.55$
$z\_in_2 = 0.15 + 0.1*1 + 0.2*1 = 0.45$

$z_1 = f(0.55)=1$
$z_2 = f(0.45)=1$

$y\_in = 0.5 + 0.5*1 + 0.5*1=1.5$
$y = f(1.5)=1$



$$f(x) = \begin{cases} 1 & \text{pokud } x \geq 0; \\ -1 & \text{pokud } x < 0. \end{cases}$$

Aktualizovat váhové hodnoty na spojeních vedoucích k Z_1:

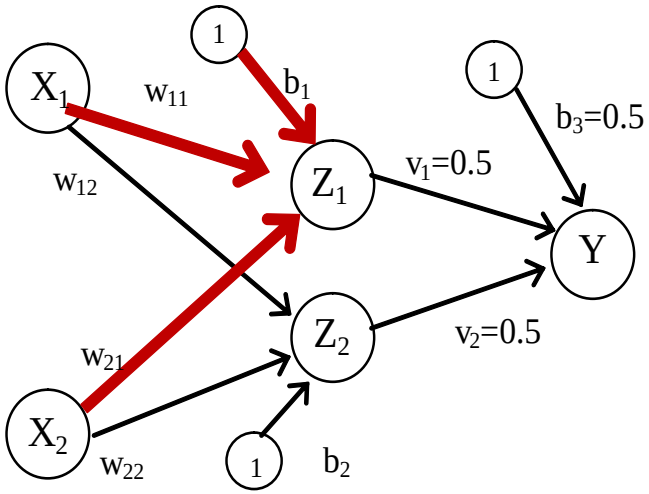
VSTUP		POŽADOVANÝ VÝSTUP
x_1	x_2	
1	1	-1
1	-1	1
-1	1	1
-1	-1	-1

$y = 1,\ \ t = -1$

$$w_{11}(new) = w_{11}(old) + \alpha(-1 - z\_in_1)x_1$$
$$= 0.05 + (0.5)(-1.55)$$
$$= -0.725$$

$$w_{21}(new) = w_{21}(old) + \alpha(-1 - z\_in_1)x_2$$
$$= 0.2 + (0.5)(-1.55)$$
$$= -0.575$$

$$b_1(new) = b_1(old) + \alpha(-1 - z\_in_1)$$
$$= 0.3 + (0.5)(-1.55)$$
$$= -0.475$$



Aktualizovat váhové hodnoty na spojeních vedoucích k Z_2:

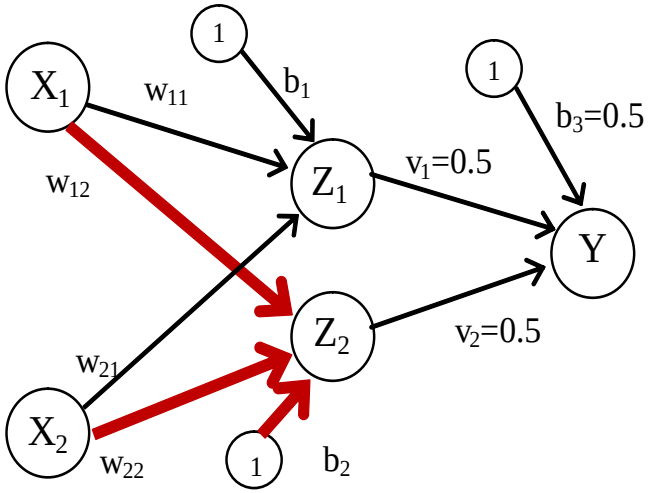
$y = 1,\ t = -1$

$$w_{12}(new) = w_{12}(old) + \alpha(-1 - z\_in_2)x_1$$
$$= 0.1 + (0.5)(-1{,}45)$$
$$= -0.625$$

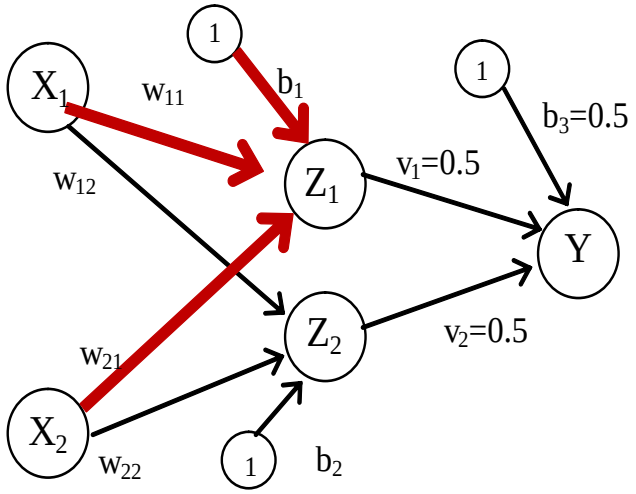
$$w_{22}(new) = w_{22}(old) + \alpha(-1 - z\_in_2)x_2$$
$$= 0.2 + (0.5)(-1.45)$$
$$= -0.525$$

$$b_2(new) = b_2(old) + \alpha(-1 - z\_in_2)$$
$$= 0.15 + (0.5)(-1.45)$$
$$= -0.575$$

VSTUP		POŽADOVANÝ VÝSTUP
x_1	x_2	
1	1	-1
1	-1	1
-1	1	1
-1	-1	-1



Geometrická interpretace nalezených váhových hodnot :

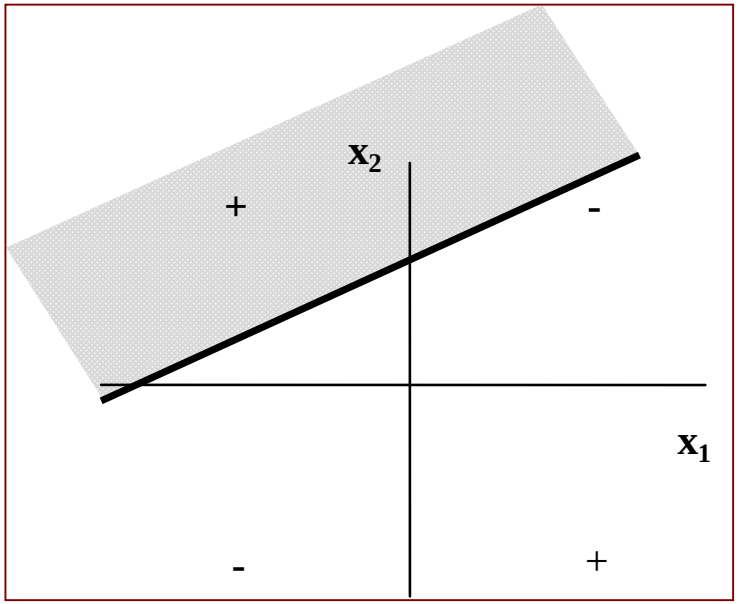


Po čtyřech tréninkových cyklech, byly nalezeny tyto váhové hodnoty pro neuron Z_1 :

$w_{11} = -0.73$

$w_{21} = 1.53$

$b_1 = -0.99$



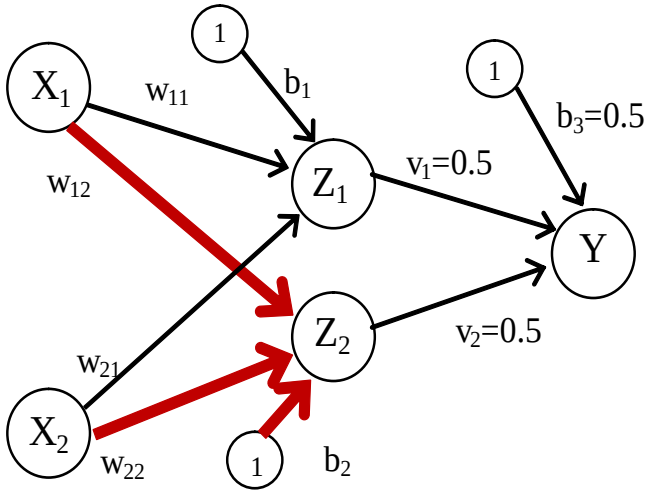
Pro skrytý neuron Z_1 má hraniční přímka pro oblast kladné odezvy tvar:

$$x_2 = -\frac{w_{11}}{w_{21}} x_1 - \frac{b_1}{w_{21}}$$

$$= \frac{0.73}{1.53} x_1 + \frac{0.99}{1.53}$$

$$= 0.48 x_1 + 0.65$$

Geometrická interpretace nalezených váhových hodnot :



Po čtyřech tréninkových cyklech, byly nalezeny tyto váhové hodnoty pro neuron Z_2 :

$w_{12} = 1.27$

$w_{22} = -1.33$

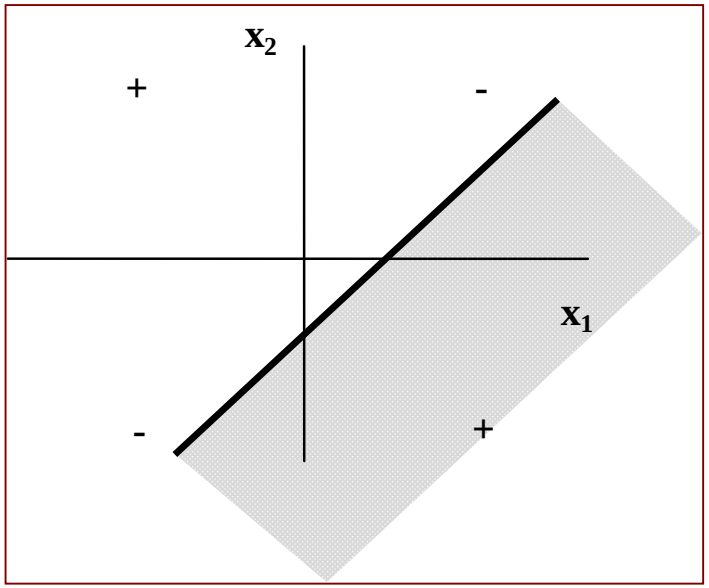
$b_2 = -1.09$

Pro skrytý neuron Z_2 má hraniční přímka pro oblast kladné odezvy tvar:

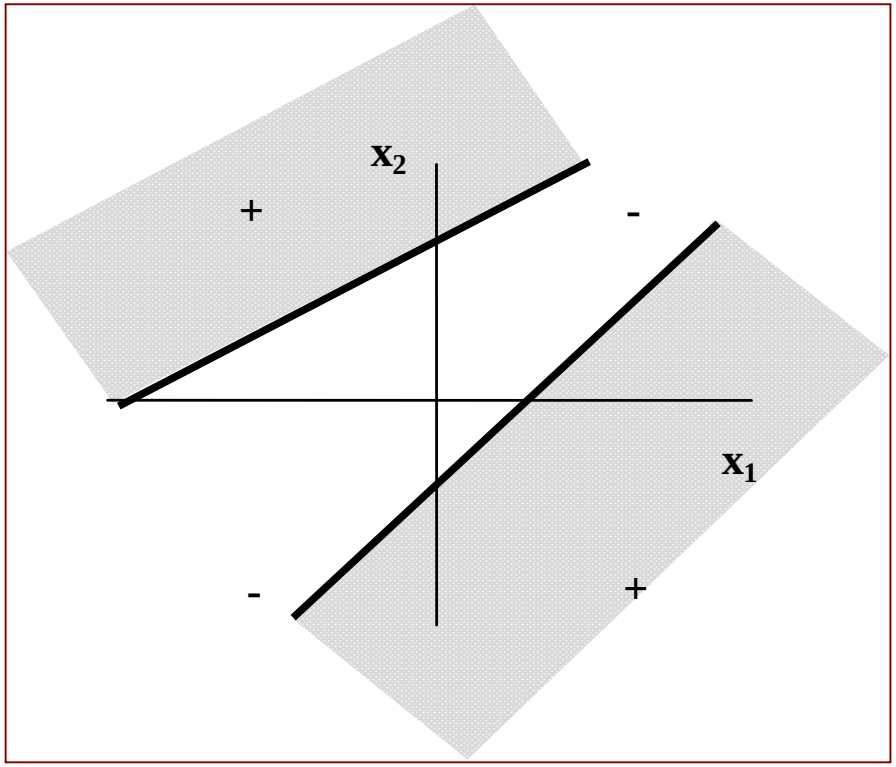
$$x_2 = -\frac{w_{12}}{w_{21}} x_1 - \frac{b_2}{w_{22}}$$

$$= \frac{1.27}{1.33} x_1 + \frac{1.09}{1.33}$$

$$= 0.96 x_1 - 0.82$$



Geometrická interpretace nalezených váhových hodnot :



Oblast kladné odezvy pro Madaline pro XOR funkci.

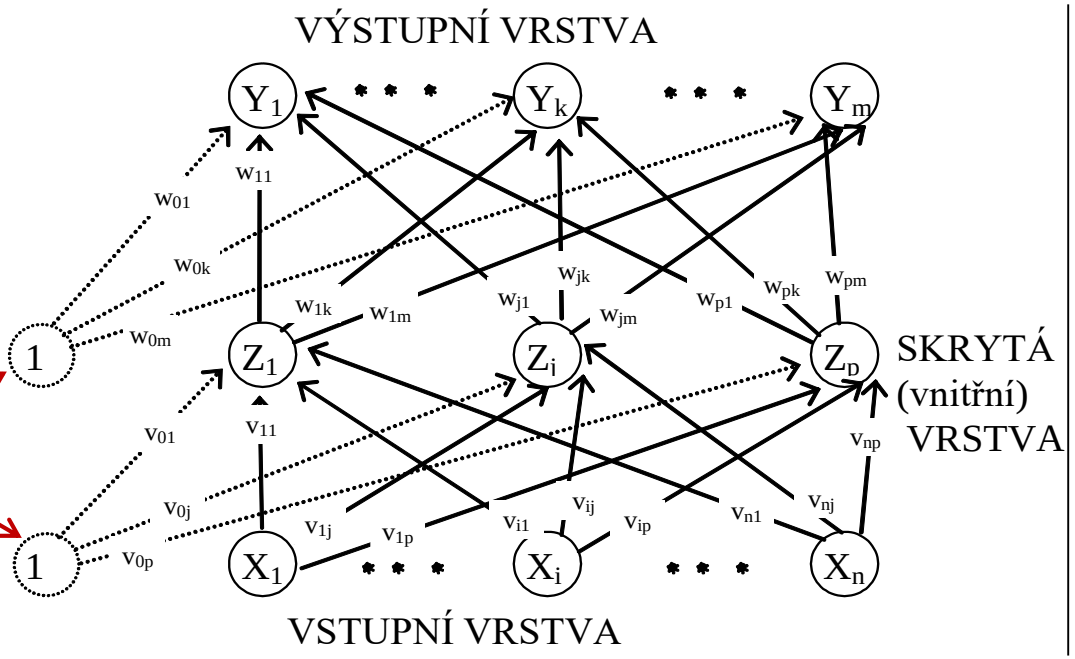
Oblast kladné odezvy vznikne sjednocením obou oblastí kladné odezvy skrytých neuronů Z_1 *a* Z_2.

Backpropagation

Topologie vícevrstvé sítě

Vícevrstvá neuronová síť je tvořena minimálně třemi vrstvami neuronů: **vstupní**, **výstupní** a alespoň jednou **vnitřní** vrstvou. Vždy mezi dvěma sousedními vrstvami se pak nachází tzv. *úplné propojení neuronů*, tedy každý **neuron nižší vrstvy** je spojen se **všemi neurony vrstvy vyšší**.

Bias odpovídá váhové hodnotě přiřazené spojení mezi daným neuronem a fiktivním neuronem, jehož aktivace je vždy *1*.



Standardní metoda backpropagation

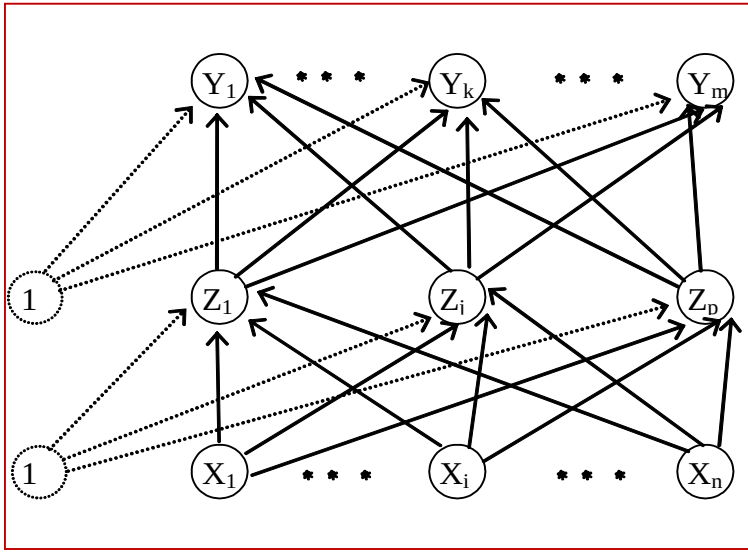
Adaptační algoritmus **zpětného šíření chyby** (*backpropagation*) je používán v přibližně 80% všech aplikací neuronových sítí.

Samotný algoritmus obsahuje tři etapy:

- dopředné (*feedforward*) šíření vstupního signálu tréninkového vzoru
- zpětné šíření chyby
- aktualizace váhových hodnot na spojeních

Dopředného (*feedforward)* šíření signálu

- Během dopředného šíření signálu obdrží každý **neuron ve vstupní vrstvě** (X_i, $i = 1,..., n$) **vstupní signál** (x_i) a zprostředkuje jeho **přenos** ke všem **neuronům vnitřní vrstvy** ($Z_1, ..., Z_p$).
- Každý **neuron ve vnitřní vrstvě** vypočítá svou **aktivaci** (z_j) a **pošle** tento signál všem **neuronům ve výstupní vrstvě**.
- Každý **neuron ve výstupní vrstvě** vypočítá svou **aktivaci** (y_k), která odpovídá jeho **skutečnému výstupu** (*k.* neuronu) **po předložení vstupního vzoru**.
- V podstatě tímto způsobem **získáme odezvu neuronové sítě** na **vstupní podnět** daný excitací neuronů vstupní vrstvy.



Dopředného (*feedforward)* šíření signálu

Takovým způsobem probíhá **šíření signálů i v biologickém systému**, kde *vstupní vrstva* může být tvořena např. *zrakovými buňkami* a ve **výstupní vrstvě mozku** jsou pak **identifikovány jednotlivé objekty** sledování.

Otázkou zůstává to nejdůležitější, **jakým způsobem jsou stanoveny synaptické váhy** vedoucí ke korektní odezvě na vstupní signál?
Proces stanovení synaptických vah je spjat s pojmem **učení (adaptace)** neuronové sítě.

Generalizace (zobecnění)

Neuronová síť je schopna **na základě naučeného usuzovat na jevy**, které **nebyly součástí učení**, které však **lze** nějakým způsobem z naučeného **odvodit**.

tj. je schopna *generalizace* (zobecnění*)* nad naučeným materiálem.

Trénovací množina

- je nutná k naučení neuronové sítě.
- *Trénovací množina* obsahuje prvky popisující řešenou problematiku
- Každý vzor trénovací množiny popisuje jakým způsobem jsou excitovány neurony vstupní a výstupní vrstvy.
- Formálně můžeme za trénovací množinu ***T*** považovat množinu *q* prvků (vzorů), které jsou definovány uspořádanými dvojicemi následujícím způsobem :

$$T = \left\{ \left(\mathbf{x}_{k,} \mathbf{t}_k \right) \middle| \, \mathbf{x}_k \in \{0,1\}^n, \mathbf{t}_k \in \{0,1\}^m, k = 1, \cdots q \right\}$$

kde

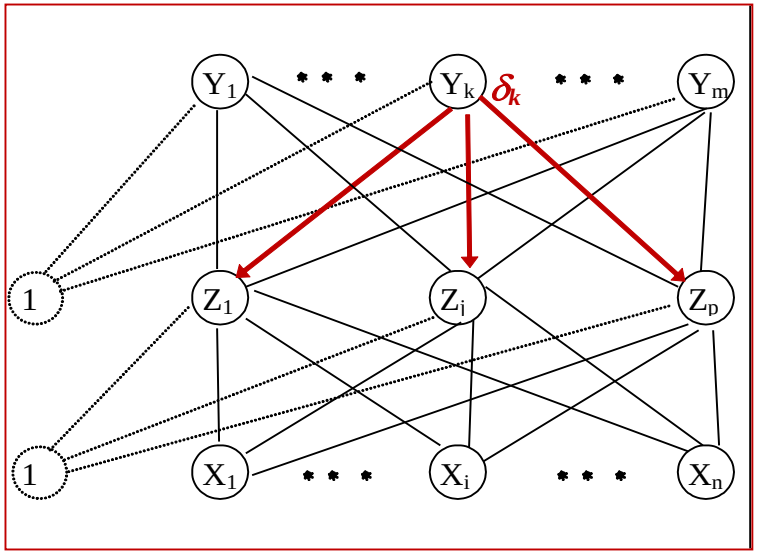
q	počet **vzorů** trénovací množiny
$\boldsymbol{x}_k$	vektor excitací **vstupní vrstvy** tvořené *n* neurony
$\boldsymbol{t}_k$	vektor excitací **výstupní vrstvy** tvořené *m* neurony

Backpropagation

- je metoda, která umožňuje **adaptaci** neuronové sítě nad danou **trénovací množinou**.
- *Backpropagation* v překladu znamená **metodu zpětného šíření**.
- Adaptace spočívá v **opačném šíření informace** směrem **od vrstev vyšších k vrstvám nižším**.
- Během adaptace neuronové sítě metodou jsou **srovnávány vypočítané aktivace** y_k s **definovanými výstupními hodnotami** t_k pro **každý neuron** ve výstupní vrstvě a pro **každý tréninkový vzor**.

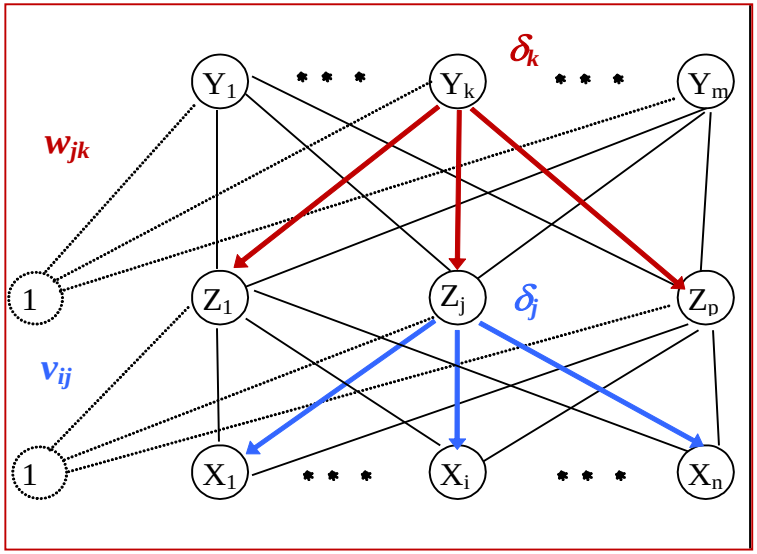
Backpropagation

- Na základě tohoto srovnání je **definována chyba neuronové sítě**,
- pro kterou je vypočítán **faktor** δ_k ($k = 1, \ldots, m$). δ_k, jež **odpovídá části chyby**, která se **šíří zpětně** z neuronu Y_k **ke všem neuronům předcházející vrstvy** majícím s tímto neuronem **definované spojení**.



Backpropagation

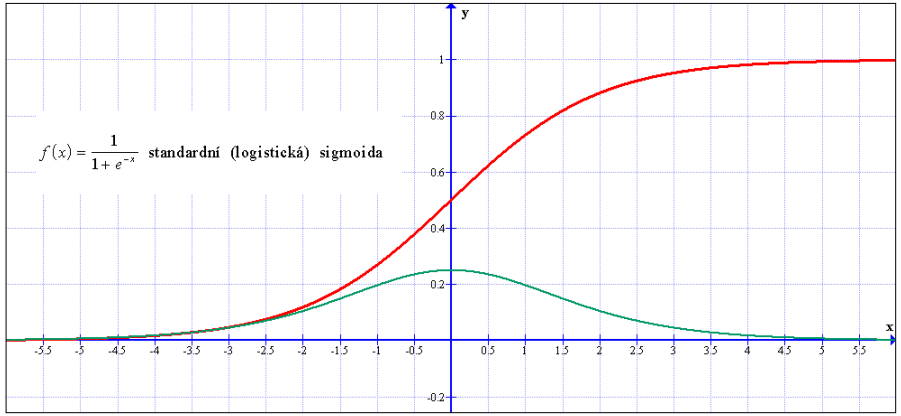
- Podobně lze definovat i **faktor** δ_j (j = *1, …, p*), který je **části chyby šířené zpětně z neur**onu Z_j ke **všem neuronům vstupní vrstvy**, jež mají s tímto neuronem **definované spojení**.

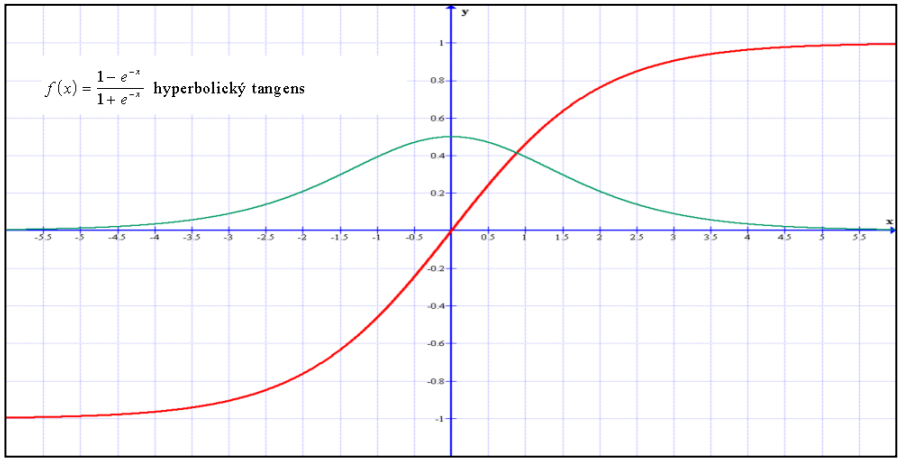


- Úprava váhových hodnot w_{jk} na spojeních mezi neurony **vnitřní** a **výstupní vrstvy** závisí na **faktoru** δ_k a **aktivacích** z_j neuronů Z_j ve **vnitřní vrstvě**.
- Úprava váhových hodnot v_{ij} na spojeních mezi neurony **vstupní** a **vnitřní vrstvy** závisí na **faktoru** δ_j a **aktivacích** x_i neuronů X_i ve **vstupní vrstvě**.

Aktivační funkce

Aktivační funkce pro neuronové sítě s adaptační metodou **backpropagation** musí mít následující **vlastnosti**: musí být **spojitá**, **diferencovatelná** a **monotónně neklesající**. Nejčastěji používanou aktivační funkcí je proto **standardní** (**logická**) **sigmoida** a **hyperbolický tangens**.





Zeleně jsou zobrazeny jejich první derivace.

Chyba sítě

- **Chyba sítě** *E(**w**)* je vzhledem k tréninkové množině definována jako **součet parciálních chyb** sítě $E_l(\mathbf{w})$ vzhledem k **jednotlivým tréninkovým vzorům** (*q*) a závisí na **konfiguraci** sítě ***w***:

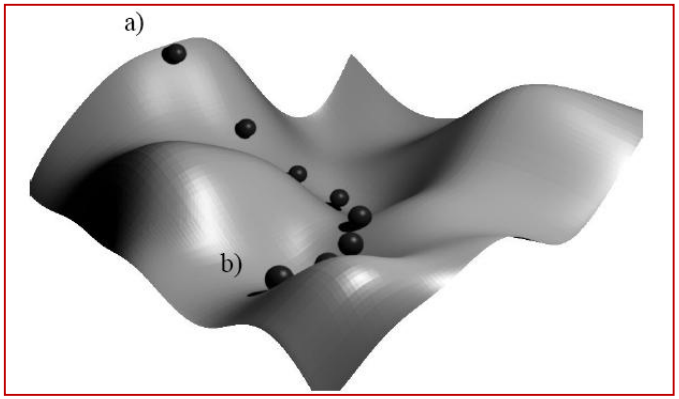
$$E(\mathbf{w}) = \sum_{l=1}^{q} E_l(\mathbf{w}).$$

- Parciální chyba $E_l(\mathbf{w})$ sítě pro *l.* tréninkový vzor ($l = 1, \ldots, q$) je úměrná součtu mocnin odchylek **skutečných hodnot výstupu** sítě pro vstup *l.* tréninkového vzoru od **požadovaných hodnot výstupů** u tohoto vzoru:

$$E_l(\mathbf{w}) = \frac{1}{2}\sum_{k \in Y}(y_k - t_k)^2.$$

Cíl adaptace

- Cílem adaptace je **minimalizace chyby** sítě ve **váhovém prostoru**.
- Vzhledem k tomu, že **chyba** sítě **přímo závisí** na **nelineární složené funkci vícevrstvé sítě**, představuje tento cíl netriviální optimalizační problém.
- Pro jeho řešení se v základním modelu používá nejjednodušší varianta **gradientní metody**, která vyžaduje **diferencovatelnost chybové funkce**.



Hlavním **problémem** gradientní metody je, že pokud již nalezne **lokální minimum**, pak toto minimum nemusí být **globální**.

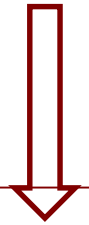
Gradientní pohyb váhového vektoru z bodu na povrchu (a) paraboloidu směrem k nejnižšímu bodu (b).

Adaptační algoritmus backpropagation

Krok 0. Váhové hodnoty a bias jsou inicializovány malými náhodnými čísly.

Přiřazení inicializační hodnoty koeficientu učení α.

Krok 1. Dokud není splněna **podmínka ukončení výpočtu**, opakovat kroky (2 až 9).



Podmínka ukončení:
pokud již nenastávají žádné změny váhových hodnot nebo pokud již bylo vykonáno maximálně definované množství váhových změn, stop; jinak, pokračovat.

Feedforward:

Krok 3. Aktivovat vstupní neurony (X_i, $i=1, ...n$)

$x_i = s_i$..

Krok 4 Vypočítat vstupní hodnoty vnitřních neuronů:

(Z_j, $j=1,..., p$):

$$z\_in_j = v_{0j} + \sum_{i=1}^{n} x_i v_{ij}.$$

Stanovení výstupních hodnot vnitřních neuronů

$$z_j = f\left(z\_in_j\right).$$

Krok 5 Stanovení skutečných výstupních hodnoty signálu neuronové sítě (Y_k, $k=1, ..., m$):

$$y\_in_k = w_{0k} + \sum_{j=1}^{p} z_j \, w_{jk},$$

$$y_k = f\left(y\_in_k\right).$$

Backpropagation:

Krok 6 Ke každému neuronu ve výstupní vrstvě (Y_k, *k=1, ..., m)* je přiřazena hodnota očekávaného výstupu pro vstupní tréninkový vzor. Dále je vypočteno $\delta_k = (t_k - y_k) f'(y\_in_k)$, které je součástí váhové korekce $\Delta w_{jk} = \alpha \delta_k z_j$ i korekce biasu $\Delta w_{0k} = \alpha \delta_k$.

Krok 7 Ke každému neuronu ve vnitřní vrstvě (*Zj, j=1, ..., p)* je přiřazena sumace jeho delta vstupů (tj. z neuronů, které se nacházejí v následující vrstvě), $\delta\_in_j = \sum_{k=1}^{m} \delta_k w_{jk}$. Vynásobením získaných hodnot derivací jejich aktivační funkce obdržíme $\delta_j = \delta\_in_j f'(z\_in_j)$, které je součástí váhové korekce $\Delta v_{ij} = \alpha \delta_j x_i$ i korekce biasu $\Delta v_{0j} = \alpha \delta_j$.

Aktualizace vah a prahů:

Krok 8 Každý neuron ve výstupní vrstvě (Y_k, $k=1, ..., m$) aktualizuje na svých spojeních váhové hodnoty včetně svého biasu ($j=0, ..., p$):

$$w_{j\,k}(new) = w_{j\,k}(old) + \Delta w_{j\,k}\,.$$

Každý neuron ve vnitřní vrstvě (Z_j, $j=1, ..., p$) aktualizuje na svých spojeních váhové hodnoty včetně svého biasu ($i=0, ..., n$):

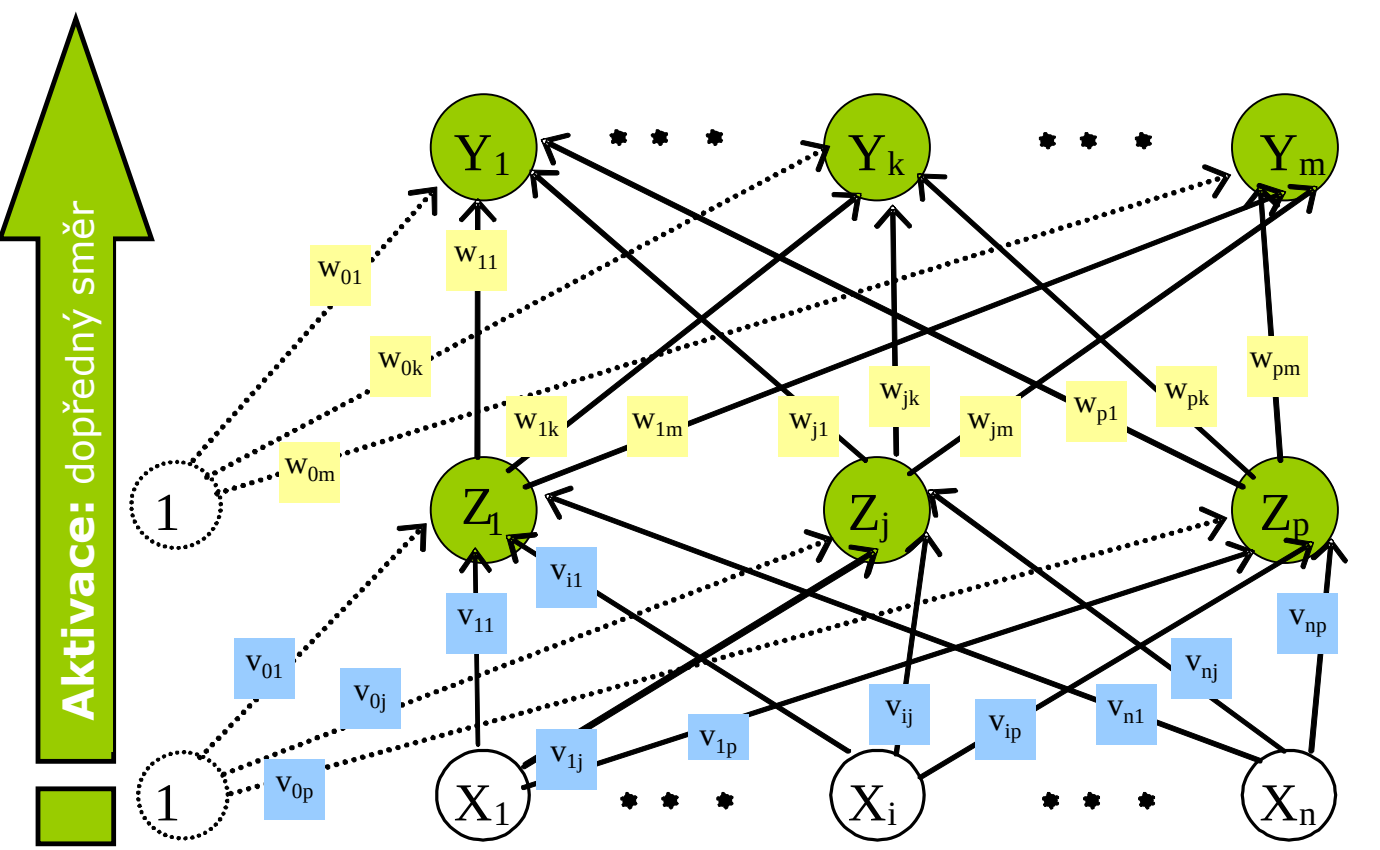
$$v_{i\,j}(new) = v_{i\,j}(old) + \Delta v_{i\,j}\,.$$

Krok 9. Podmínka ukončení:

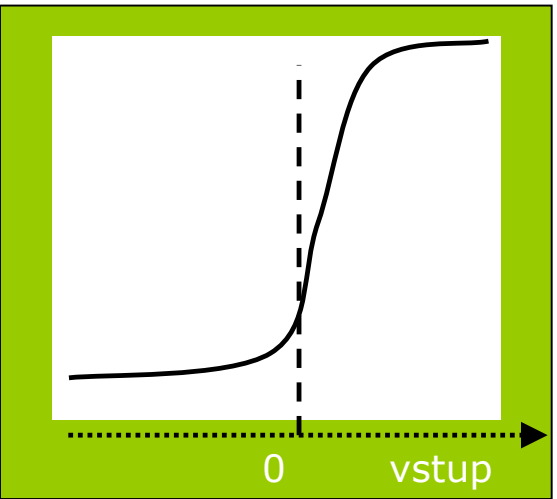
pokud již nenastávají žádné změny váhových hodnot nebo pokud již bylo vykonáno maximálně definované množství váhových změn, stop; jinak, pokračovat.

Backpropagation

- Ačkoliv algoritmus backpropagation je formulován pro **klasický von Neumannovský model počítače**, lze jej **implementovat distribuovaně**.
- Pro každý tréninkový vzor probíhá **nejprve aktivní režim** pro jeho vstup tak, že informace se v neuronové síti šíří **od vstupu k jejímu výstupu**.
- Výpočet sítě při zpětném chodu probíhá **sekvenčně po vrstvách**, přitom **v rámci jedné vrstvy může probíhat paralelně**.



$$E(\mathbf{w}) = \sum_{l=1}^{q} E_l(\mathbf{w}).$$



$$\begin{aligned}\Delta w_{jk} &= -\alpha \frac{\partial E}{\partial w_{jk}} \\ &= \alpha[t_k - y_k] f'(y\_in_k) z_j \\ &= \alpha \delta_k z_j;\end{aligned}$$

$$\begin{aligned}\Delta v_{ij} &= -\alpha \frac{\partial E}{\partial v_{ij}} \\ &= \alpha\ f'(z\_in_j) x_i \sum_k \delta_k w_{jk}, \\ &= \alpha \delta_j x_i.\end{aligned}$$

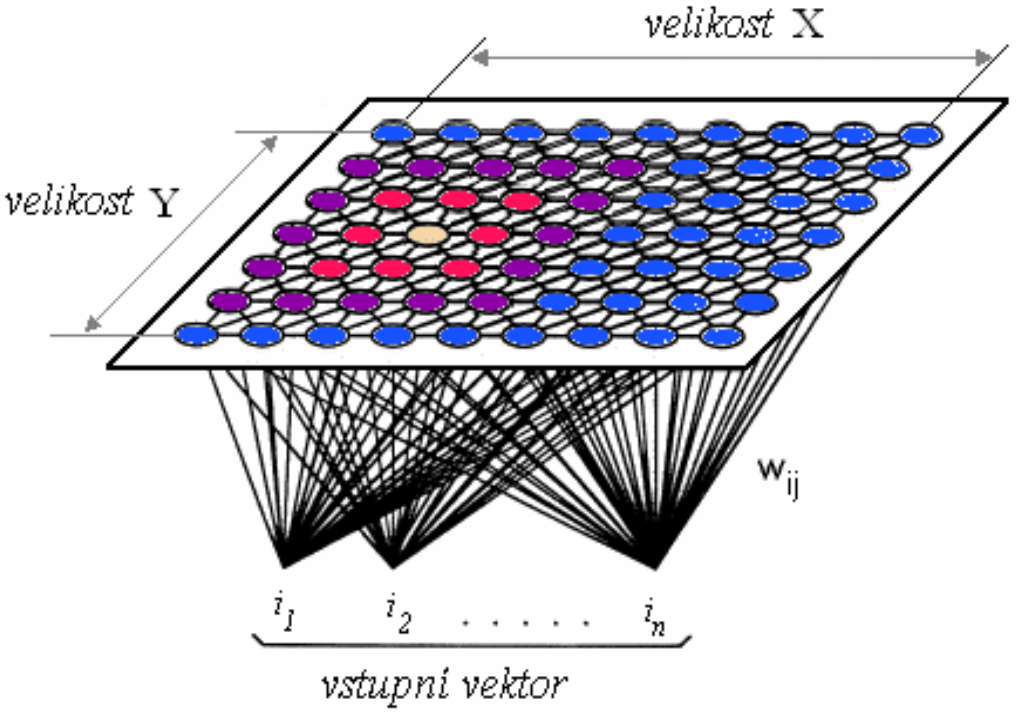
Kohonenovy samoorganizační mapy

SOM (*Self -Organizing Map*)

- Tato neuronová síť je nejdůležitější architekturou vycházející ze **strategie soutěžního učení** (tj. *učení bez učitele*).
- Základním principem učícího procesu je **vytvoření množiny reprezentantů** mající **stejné pravděpodobnosti výběru**.
- Algoritmus **nemá informace** o požadovaných aktivitách výstupních neuronů v **průběhu adaptace**, ale adaptace vah **odráží statistické vlastnosti trénovací množiny**.
- Jsou-li si tedy dva libovolné vzory **blízké ve vstupním prostoru** způsobují v síti **odezvu na neuronech**, které jsou si **fyzicky blízké ve výstupním prostoru**.
- Hlavní ideou těchto neuronových sítí je **nalézt prostorovou reprezentaci složitých datových struktur**. Mnohodimenzionální data se tímto způsobem zobrazují v daleko jednodušším prostoru.

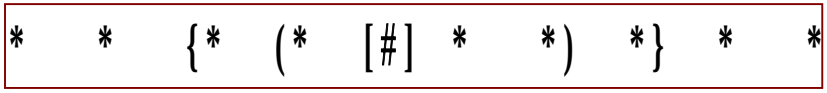
Topologie SOM

- Jedná se o **dvouvrstvou síť** s **úplným propojením** neuronů mezi vrstvami.
- **Výstupní neurony** jsou navíc **uspořádány** do nějaké **topologické struktury**, nejčastěji to bývá **dvojrozměrná mřížka** nebo **jednorozměrná řada** jednotek.
- Tato **topologická struktura** určuje, které neurony spolu v síti **sousedí**.

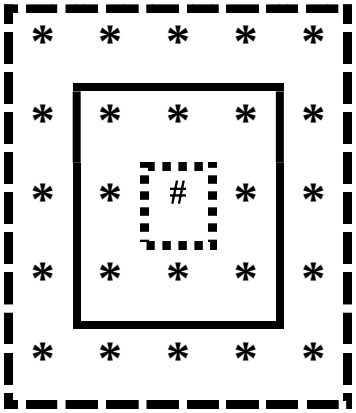


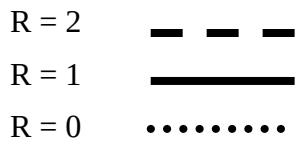
Sousedství (okolí) výstupního neuronu

Pro **adaptační proces** je rovněž důležité zavést pojem ***okolí J* výstupního neuronu (*j**) o *poloměru* (velikosti) *R*,** což je **množina** všech **neuronů** ($j \in J$), jejichž **vzdálenost** v síti je od daného neuronu (*j**) **menší** nebo **rovna** R: $J = \{j; d(j,j^*) \leq R\}$. To, jak měříme vzdálenost $d(j,j^*)$, **je závislé na topologické struktuře** výstupních neuronů



Sousedství definovaná v lineární výstupní oblasti pro různé hodnoty parametru R

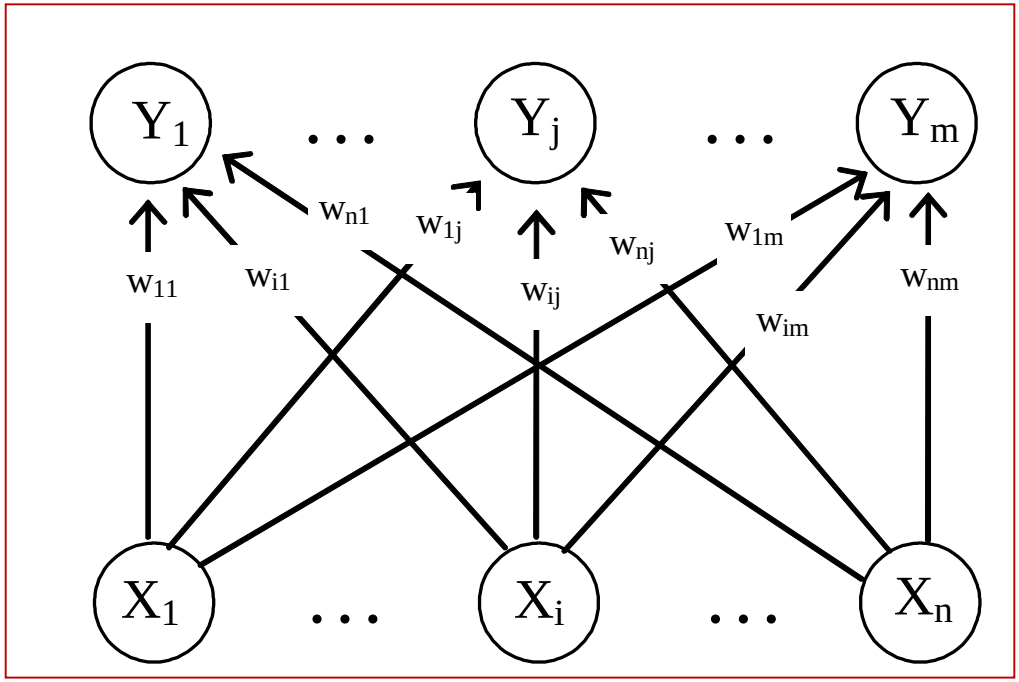




Sousedství v pravoúhlé dvojrozměrné oblasti různé hodnoty parametru R

Kohonenova samoorganizační mapa

Obecná architektura Kohonenovy samoorganizační mapy obsahující *m* neuronů ve **výstupní vrstvě** (tj. $Y_1,\ldots, Y_m$) a *n* neuronů ve **vstupní vrstvě** (tj. $X_1,\ldots, X_n$).



Kohonenův algoritmus

Princip adaptivní dynamiky je jednoduchý:

- Procházíme celou **tréninkovou množinu** a po předložení jednoho tréninkového **vzoru** proběhne mezi neurony sítě **kompetice**.
- Její **vítěz** pak spolu s neurony, které jsou v jeho **okolí**, změní své **váhové hodnoty**.
- Reálný **parametr učení** $0 < \alpha \leq 1$ určuje **míru změny** vah.
- Na počátku učení je obvykle **blízký jedné** a postupně se **zmenšuje** až na **nulovou hodnotu**, což zabezpečuje **ukončení** procesu **adaptace**.
- Rovněž i **velikost okolí** R není konstantní: na **začátku** adaptace je okolí **velké** (např. polovina velikosti sítě) a na **konci** učení potom zahrnuje jen **jeden samotný vítězný neuron** (tj. $R = 0$).

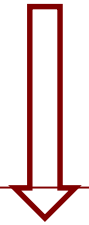
Kohonenův algoritmus

Krok 0. Inicializace všech váhových hodnot w_{ij}:

Inicializace poloměru sousedství; tj okolí (R).

Inicializace parametru učení (α).

Krok 1. Pokud není **splněna podmínka ukončení**, provádět kroky (2-8).



Podmínka ukončení:
Parametr učení α je blízký nule.

Krok 2. Pro každý vstupní vektor $\boldsymbol{x} = (x_1,...,x_n)$ opakovat kroky 3 až 5.

Krok 3. Pro každé j $(j = 1,..., m)$ vypočítat:

$$D(j) = \sum_i \left(w_{ij} - x_i \right)^2 .$$

Krok 4. Najít index J takový, že $D(J)$ je minimum.

Krok 5. Aktualizace váhových hodnot všech neuronů $(j \in J)$ tvořících topologické sousedství charakterizované indexem J, tj. pro všechna i $(i = 1,..., n)$ platí:

$$w_{ij}(new) = w_{ij}(old) + \alpha\left[x_i - w_{ij}(old)\right].$$

Krok 6. Aktualizace parametru učení.

Krok 7. Zmenšení poloměru R topologického sousedství.

Krok 8. Test podmínky ukončení.

Geometrický význam algoritmu adaptace

- **Vítězný neuron** a všichni jeho **sousedé** v síti **posunou** svůj **váhový vektor** o určitou poměrnou vzdálenost **směrem k aktuálnímu vstupu**.
- **Vítězný neuron**, který nejlépe reprezentuje předložený vstup (je mu nejblíže), ještě více takto **zlepší svou relativní pozici** vůči němu.
- **Problémem** vzniklým při adaptaci může být **nevhodná** náhodná **inicializace vah**, která vede k **blízkým počátečním neuronům** ve výstupní vrstvě a tudíž pouze **jeden** z nich vyhrává kompetici zatímco **ostatní** zůstávají nevyužity.

Aktivní režim

- V aktivním režimu se **sousedství** neuronů **neprojevuje**.
- Předložíme-li síti vstupní vektor, výstupní neuron, který je mu **nejblíže** se **excituje** na hodnotu rovnu **jedné**, zatímco **výstupy ostatních** neuronů jsou rovny **nule**.
- Každý neuron tak reprezentuje **třídu objektů** ze vstupního prostoru: tj. pouze **jeden** výstupní neuron, jehož **potenciál** ($w.x$) je **maximální** odpovídá **vstupnímu** vektoru $\boldsymbol{x}$.
- Princip „**vítěz bere vše**“ se realizuje tzv. *laterální inhibicí*; všechny **výstupní** neurony jsou navzájem **propojeny** vazbami, které mezi nimi přenášejí **inhibiční signály**.
- Každý **výstupní** neuron se pak snaží v **kompetici zeslabit** ostatní neurony silou **úměrnou** jeho **potenciálu**, který je **tím větší, čím je neuron blíže vstupu**.
- Výstupní neuron s největším potenciálem **utlumí** ostatní výstupní neurony a sám **zůstane aktivním**.

Příklad

Mějme 4 **vektory**:
(1, 1, 0, 0); (0, 0, 0, 1); (1, 0, 0, 0); (0, 0, 1, 1).

Maximální počet **shluků** je: $m = 2$.

Předpokládejme, že **parametr učení** α je definován vztahy:

$$\alpha(0) = 0.6$$

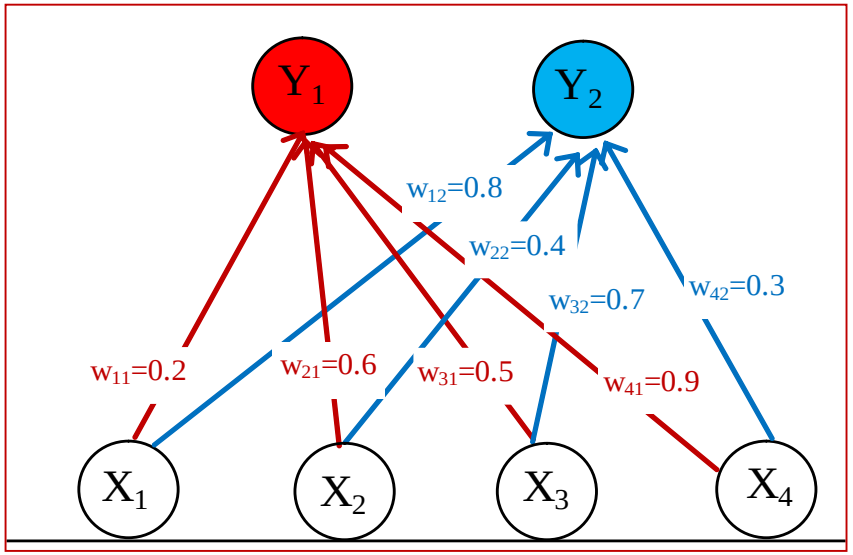
$$\alpha(t+1) = 0.5\ \alpha(t)$$

Protože jsou k dispozici pouze **dva shluky**, **okolí** bodu J je nastaveno tak, že v každém kroku **aktualizuje** své **váhové** hodnoty pouze **jeden** neuron **výstupní** vrstvy, tj. $R = 0$.

Inicializace:

Inicializace **váhové matice**:

$$\begin{bmatrix} 0.2 & 0.8 \\ 0.6 & 0.4 \\ 0.5 & 0.7 \\ 0.9 & 0.3 \end{bmatrix}$$



Inicializace poloměru **sousedství**:

$$R = 0$$

Inicializace **parametru učení**:

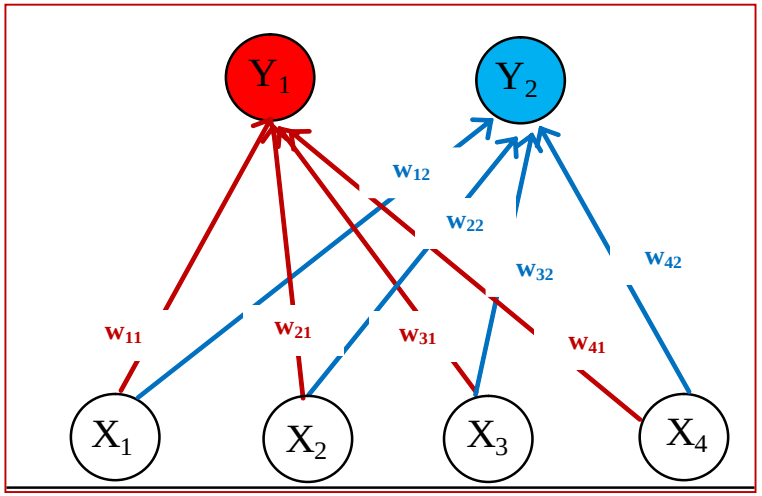
$$\alpha(0) = 0.6$$

Pro **první** vektor (1, 1, 0, 0) :

$$D(1) = (0.2-1)^2 + (0.6-1)^2 + (0.5-0)^2 + (0.9-0)^2 = 1.86$$

$$D(2) = (0.8-1)^2 + (0.4-1)^2 + (0.7-0)^2 + (0.3-0)^2 = 0.98$$

Vstupní vektor je blíže uzlu 2 (*J = 2).*



Aktualizace váhových **hodnot vítězného neuronu p**odle vztahu:

$$w_{i2}(new) = w_{i2}(old) + 0.6\left[x_i - w_{i2}(old)\right]$$

$$= 0.4 w_{i2}(old) + 0.6 x_i$$

Aktualizace **druhého sloupce** váhové matice:

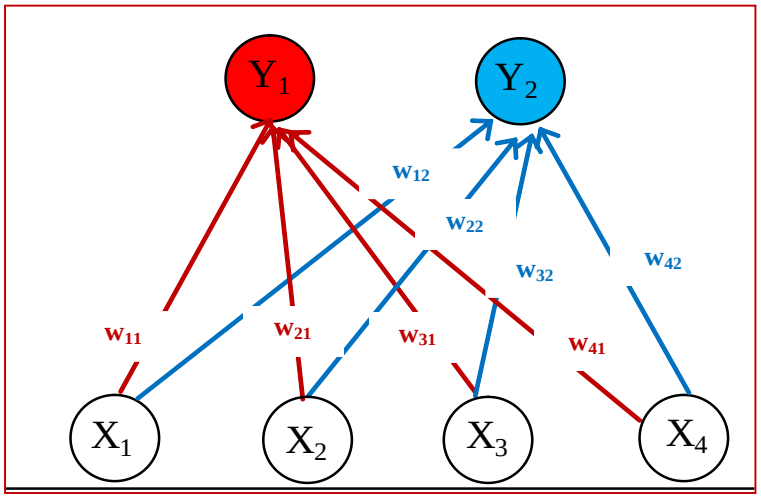
$$\begin{bmatrix} 0.2 & 0.92 \\ 0.6 & 0.76 \\ 0.5 & 0.28 \\ 0.9 & 0.12 \end{bmatrix}$$

Pro **druhý** vektor (0, 0, 0, 1) :

$$D(1) = (0.2-0)^2 + (0.6-0)^2 + (0.5-0)^2 + (0.9-1)^2 = 0.66$$

$$D(2) = (0.92-0)^2 + (0.76-0)^2 + (0.28-0)^2 + (0.12-1)^2 = 2.27$$

Vstupní vektor je blíže uzlu 1 (*J = 1).*



Aktualizace váhových **hodnot vítězného neuronu p**odle vztahu:

$$w_{i1}(new) = w_{i1}(old) + 0.6\left[x_i - w_{i1}(old)\right]$$

$$= 0.4\,w_{i1}(old) + 0.6\,x_i$$

Aktualizace **prvního sloupce** váhové matice:

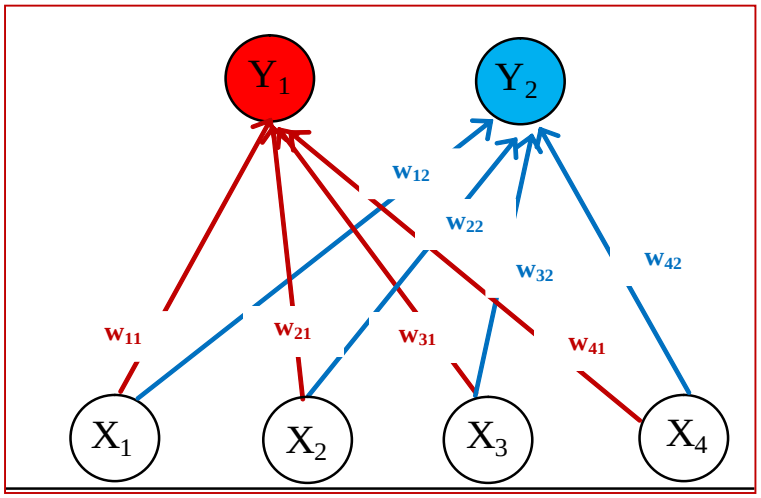
$$\begin{bmatrix} 0.08 & 0.92 \\ 0.24 & 0.76 \\ 0.20 & 0.28 \\ 0.96 & 0.12 \end{bmatrix}$$

Pro **třetí** vektor (1, 0, 0, 0) :

$$D(1) = (0.08-1)^2 + (0.24-0)^2 + (0.2-0)^2 + (0.96-0)^2 = 1.86$$

$$D(2) = (0.92-1)^2 + (0.76-0)^2 + (0.28-0)^2 + (0.12-0)^2 = 0.67$$

Vstupní vektor je blíže uzlu 2 (*J = 2).*



Aktualizace váhových **hodnot vítězného neuronu p**odle vztahu:

$$w_{i2}(new) = w_{i2}(old) + 0.6\left[x_i - w_{i2}(old)\right]$$

$$= 0.4 w_{i2}(old) + 0.6 x_i$$

Aktualizace **druhého sloupce** váhové matice:

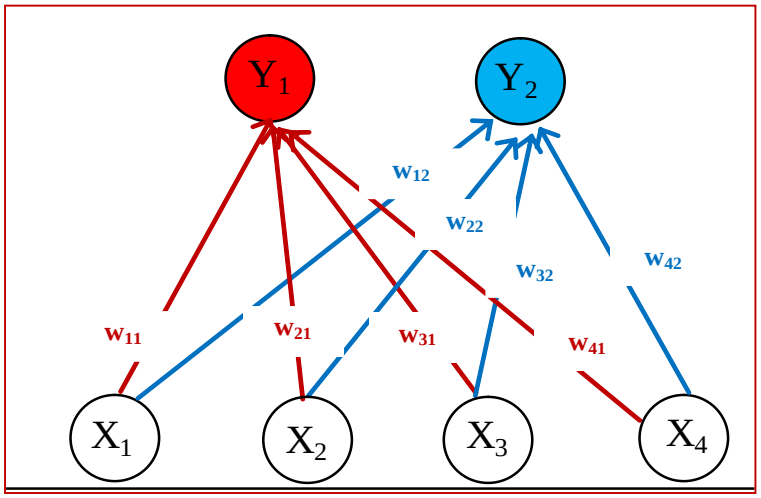
$$\begin{bmatrix} 0.08 & 0.968 \\ 0.24 & 0.304 \\ 0.20 & 0.112 \\ 0.96 & 0.048 \end{bmatrix}$$

Pro **čtvrtý** vektor (0, 0, 1, 1) :

$$D(1) = (0.08-0)^2 + (0.24-0)^2 + (0.2-1)^2 + (0.96-1)^2 = 0.705$$

$$D(2) = (0.968-0)^2 + (0.304-0)^2 + (0.112-1)^2 + (0.048-1)^2 = 2.72$$

Vstupní vektor je blíže uzlu 1 (*J = 1).*



Aktualizace váhových **hodnot vítězného neuronu p**odle vztahu:

$$w_{i1}(new) = w_{i1}(old) + 0.6\left[x_i - w_{i1}(old)\right]$$

$$= 0.4 w_{i1}(old) + 0.6 x_i$$

Aktualizace **prvního sloupce** váhové matice:

$$\begin{bmatrix} 0{,}032 & 0{,}968 \\ 0{,}096 & 0{,}304 \\ 0{,}680 & 0{,}112 \\ 0{,}984 & 0{,}048 \end{bmatrix}$$

Zmenšení parametru učení:
α = 0.5 (0.6) = 0.3

Aktualizace váhových hodnot vítězného neuronu *j* (*j* = *1, 2*)
ve druhém cyklu bude prováděna podle vztahu:

$$w_{ij}(new) = w_{ij}(old) + 0.3\left[x_i - w_{ij}(old)\right]$$
$$= 0.7\,w_{ij}(old) + 0.3x_i.$$

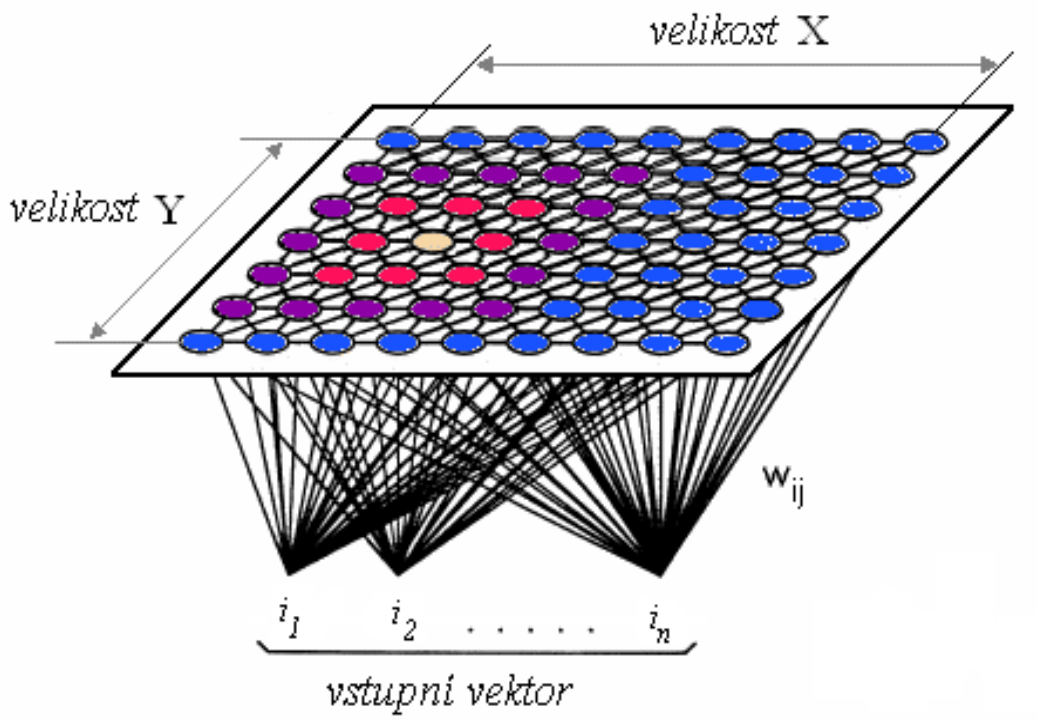
Parametr učení zmenšil svou hodnotu během adaptace na 0.01 a výsledná váhová matice nabývala konverguje k matici:

$$\begin{bmatrix} 0.0 & 1.0 \\ 0.0 & 0.5 \\ 0.5 & 0.0 \\ 1.0 & 0.0 \end{bmatrix}$$

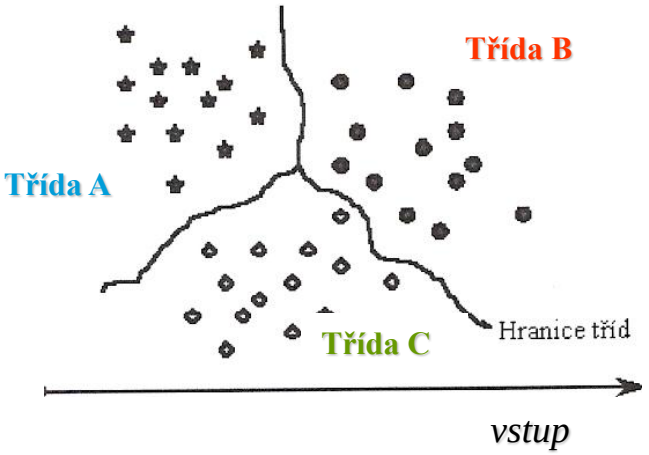
Vstupní vektory:
(1, 1, 0, 0); (0, 0, 0, 1); (1, 0, 0, ,0); (0, 0, 1, 1)

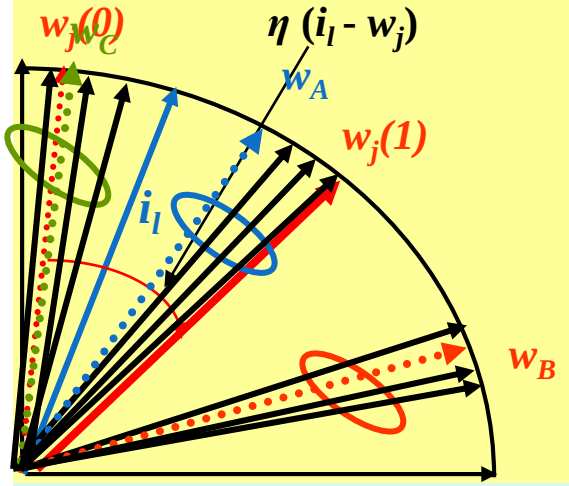
První sloupec nabývá hodnot, které odpovídají **průměrným hodnotám složek** obou **vektorů** přiřazeným **prvnímu** neuronu **výstupní vrstvy**, tj. vektoru 2: (0, 0, 0, 1) a vektoru 4: (0, 0, 1, 1).

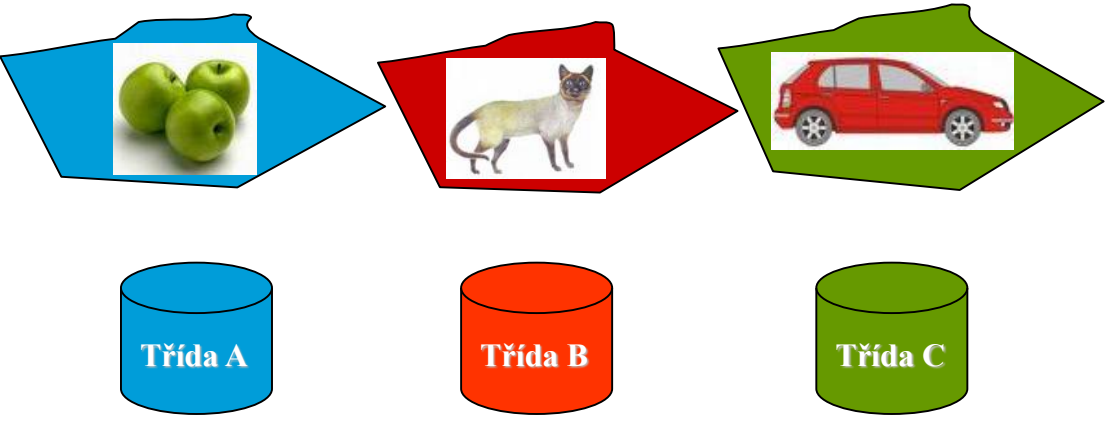
Druhý sloupec nabývá hodnot, které odpovídají **průměrným hodnotám složek** obou vektorů přiřazeným **druhému** neuronu výstupní vrstvy, tj. vektoru 1: (1, 1, 0, 0) a vektoru 3: (1, 0, 0, ,0).



TŘÍDY







$$w_j = w_j + \Delta w_j$$

$$\Delta w_j = \eta\,(i_l - w_j)$$

$$i_l - w_j = \sqrt{\sum_k (i_{lk} - w_{jk})^2}$$

Proces shlukování

- vysvětlíme prostřednictvím funkce **hustoty pravděpodobnosti**.
- Tato funkce reprezentuje **statistický nástroj popisující rozložení dat v prostoru**.
- Pro daný bod prostoru lze stanovit **pravděpodobnost**, že **vektor** bude v daném bodu **nalezen**.
- Je-li dán **vstupní prostor** a **funkce hustoty pravděpodobnosti**, pak je možné dosáhnout **takové organizace mapy**, která se této funkci **přibližuje**.
- Jinými slovy řečeno**, pokud jsou vzory ve vstupním prostoru rozloženy podle nějaké distribuční funkce, budou váhové vektory rozloženy analogicky**.

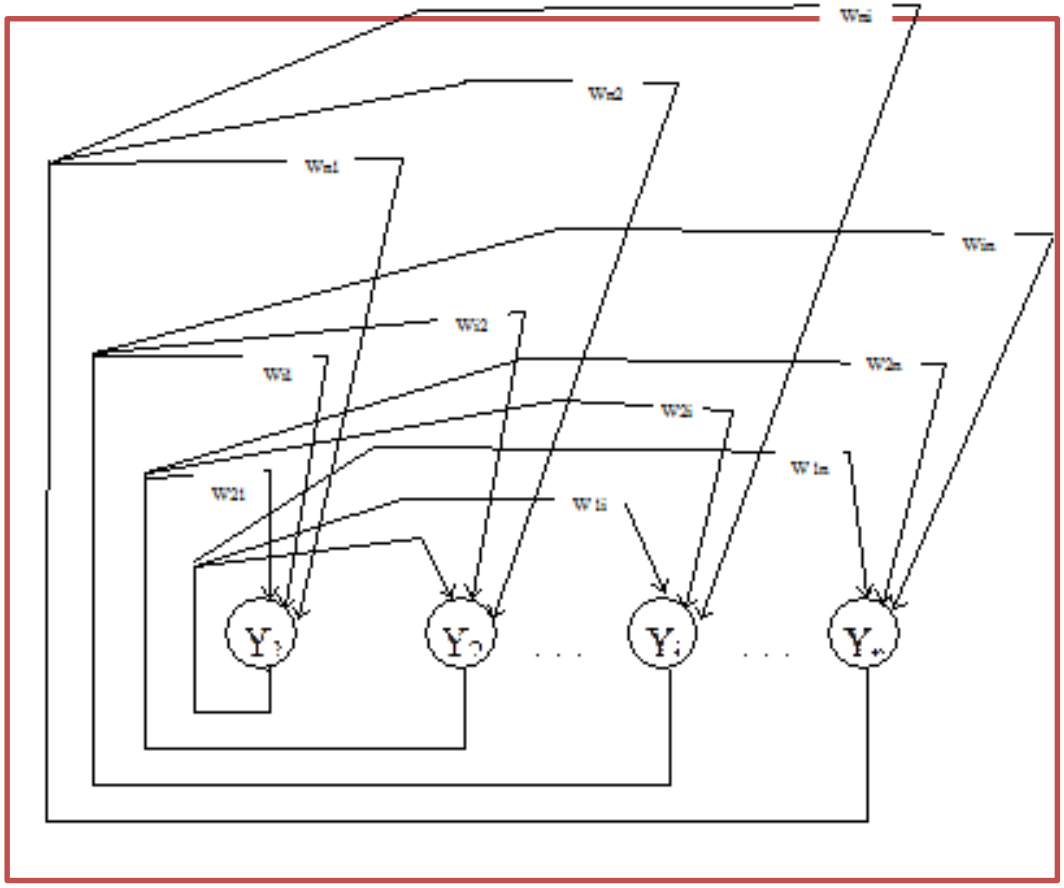
Diskrétní Hopfieldova síť

Hopfieldova síť

- **Model** Hopfieldovy neuronové sítě je **založen** na **využití** *energetické funkce* **svázané** s *neuronovou sítí tak*, jak je to běžné u *fyzikálních systémů*.
- **Organizační dynamika** diskrétní Hopfieldovy sítě specifikuje **úplnou topologii cyklické** neuronové sítě s *n* neurony, kde **každý neuron** v síti je spojen **se všemi ostatními neurony** sítě, tj. má všechny neurony za své vstupy.
- Všechny neurony v síti jsou tedy **zároveň vstupní** i **výstupní**.

Topologie Hopfieldovy sítě

- **Každý spoj** v síti mezi neuronem i ($i = 1, ..., n$) a neuronem j ($j = 1, ..., n$) je ohodnocen **celočíselnými synaptickými vahami** w_{ij} a w_{ji}, které jsou **symetrické**, tj. $w_{ij} = w_{ji}$.
- V **základním** modelu platí, že **žádný neuron není spojen sám se sebou**, tj. odpovídající váhy $w_{jj} = 0$ ($j = 1, ..., n$) jsou nulové.



Hebbovo učení

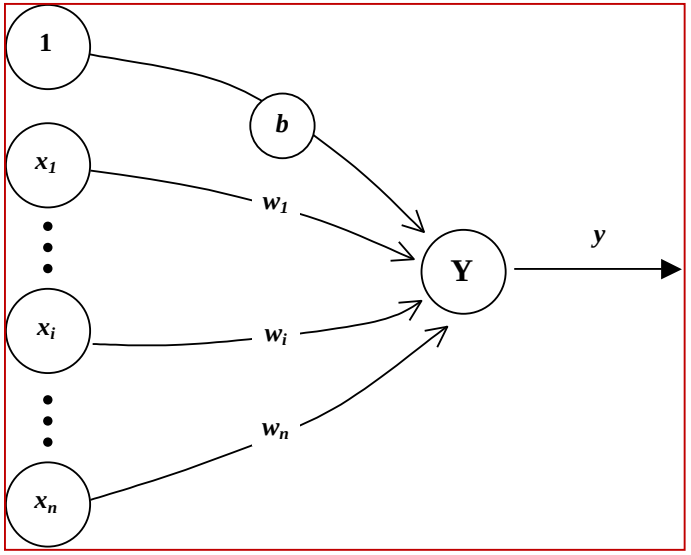
- Hebbovo učení je založeno na myšlence, že *váhové hodnoty* na spojení **mezi dvěma neurony, které jsou současně ve stejném stavu**, budou **narůstat** a naopak.
- *Změna synaptické váhy* spoje mezi dvěma neurony **je úměrná jejich souhlasné aktivitě**, tj. součinu jejich stavů.
- Donald Hebb tímto způsobem vysvětloval vznik podmíněných reflexů.

Princip Hebbova učení

Uvažujme **jednovrstvou neuronovou síť**, ve které jsou *všechny vstupní neurony propojeny s jediným výstupní neuronem Y*, ale ne již navzájem mezi sebou.

Pokud jsou složku vstupního vektoru ***x*** = ($x_1, \ldots, x_n$) reprezentovány v **bipolární formě**, lze složky příslušného váhového vektoru ***w***=($w_1, \ldots, w_n$) aktualizovat následovně: $w_i(new) = w_i(old) + x_i y$,

kde *y* je výstup z neuronu *Y*.



Hebbovo adaptační pravidlo

Krok 0. Inicializace všech vah:

$w_i = 0 \quad (i = 1$ až $n)$

Krok 1. Pro každý vzor - tréninkový pár, tj. vstupní vektor (**s**) a příslušný výstup (**t**), opakovat následující kroky (2 až 4).

Krok 2. Aktivovat vstupní neurony:

$x_i = s_i \quad (i = 1$ až $n)$.

Krok 3. Aktivovat výstupní neuron:

$y = t$.

Krok 4. Aktualizovat váhy podle

$w_i(new) = w_i(old) + x_i y \quad (i = 1$ až $n)$.

Aktualizovat biasy podle

$b(new) = b(old) + y$.

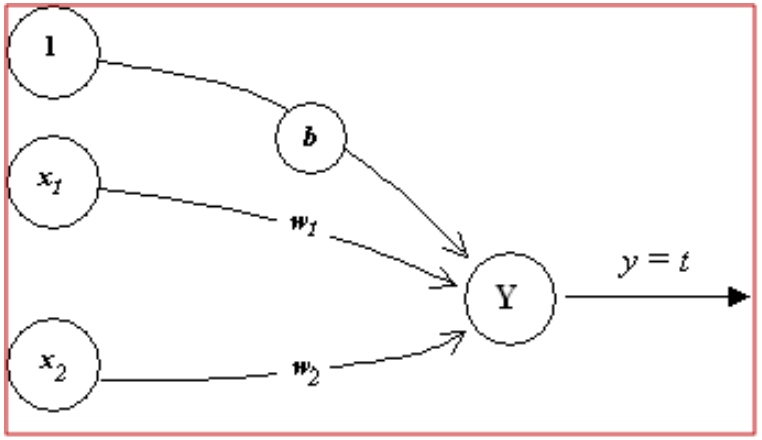
- ***Bias*** lze zapsat také jako *váhovou hodnotu* přiřazenou výstupu z neuronu**, jehož aktivace má vždy hodnotu** 1.
- Aktualizace váhových hodnot může být také vyjádřena ve **vektorové formě** jako:

$$\boldsymbol{w}(new) = \boldsymbol{w}(old) + \boldsymbol{x}y.$$

- Tento algoritmus je pouze *jedním z mnoha způsobu implementace* Hebbova pravidla učení.
- Tento algoritmus **vyžaduje jen jeden průchod tréninkovou množino**u.

Příklad:

Hebbovovo pravidlo učení pro **logickou funkci AND** v **bipolární reprezentaci** můžeme zapsat následovně**.**



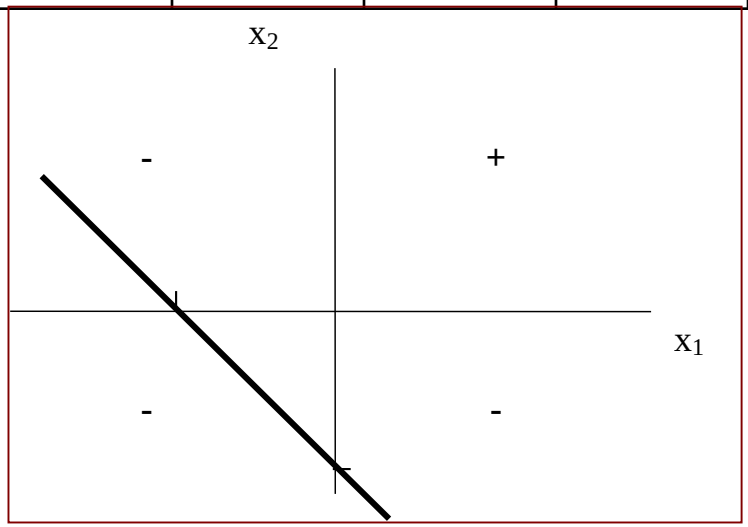
čas	**VSTUP**		**POŽADOVANÝ VÝSTUP**	**PŘÍRUSTKY VAH**			**VÁHOVÉ HODNOTY**		
	x_1	x_2	t	Δw_1	Δw_2	Δb	w_1	w_2	b
0							0	0	0
1	1	1	1	1	1	1	1	1	1
2	1	-1	-1	-1	1	-1	0	2	0
3	-1	1	-1	1	-1	-1	1	1	-1
4	-1	-1	-1	1	1	-1	2	2	-2

Grafický postup řešení:

čas	VSTUP		POŽADOVANÝ VÝSTUP	PŘÍRUSTKY VAH			VÁHOVÉ HODNOTY		
	x_1	x_2	t	Δw_1	Δw_2	Δb	w_1	w_2	b
0							0	0	0
1	1	1	1	1	1	1	1	1	1
2	1	-1	-1	-1	1	-1	0	2	0
3	-1	1	-1	1	-1	-1	1	1	-1
4	-1	-1	-1	1	1	-1	2	2	-2

Rovnice přímky pro **první** tréninkový vzor:

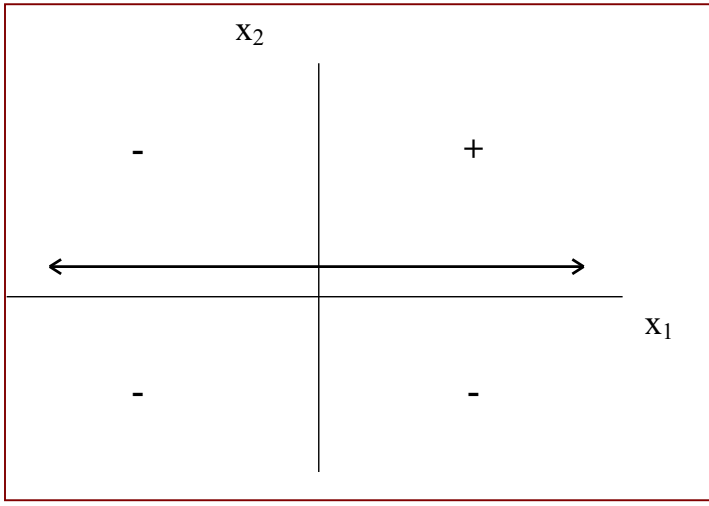
$$x_2 = -x_1 - 1$$



čas	VSTUP		POŽADOVANÝ VÝSTUP	PŘÍRUSTKY VAH			VÁHOVÉ HODNOTY		
	x_1	x_2	t	Δw_1	Δw_2	Δb	w_1	w_2	b
0							0	0	0
1	1	1	1	1	1	1	1	1	1
2	1	-1	-1	-1	1	-1	0	2	0
3	-1	1	-1	1	-1	-1	1	1	-1
4	-1	-1	-1	1	1	-1	2	2	-2

Rovnice přímky pro **druhý** tréninkový vzor:

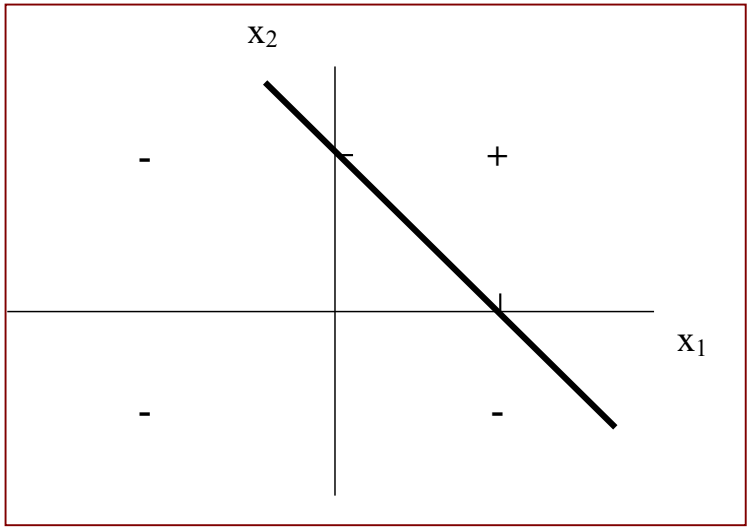
$$x_2 = 0$$



čas	**VSTUP**		**POŽADOVANÝ VÝSTUP**	**PŘÍRUSTKY VAH**			**VÁHOVÉ HODNOTY**		
	x_1	x_2	t	Δw_1	Δw_2	Δb	w_1	w_2	b
0							0	0	0
1	1	1	1	1	1	1	1	1	1
2	1	-1	-1	-1	1	-1	0	2	0
3	-1	1	-1	1	-1	-1	1	1	-1
4	-1	-1	-1	1	1	-1	2	2	-2

Rovnice přímky pro **třetí** tréninkový vzor:

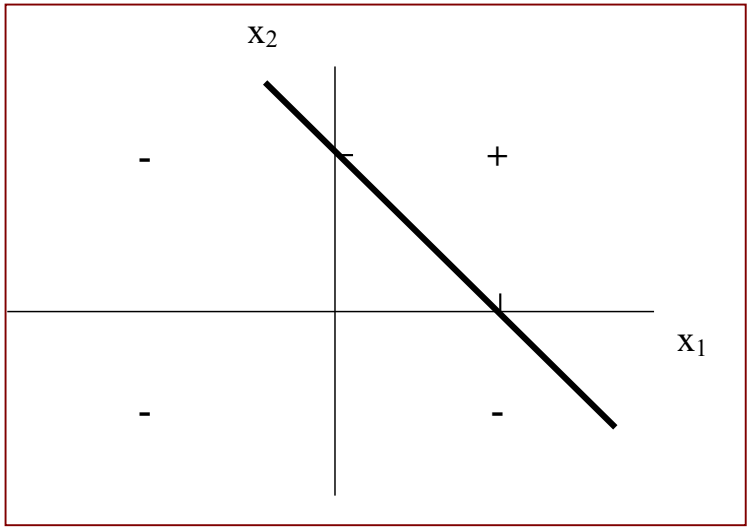
$$x_2 = -x_1 + 1$$



čas	VSTUP		POŽADOVANÝ VÝSTUP	PŘÍRUSTKY VAH			VÁHOVÉ HODNOTY		
	x_1	x_2	t	Δw_1	Δw_2	Δb	w_1	w_2	b
0							0	0	0
1	1	1	1	1	1	1	1	1	1
2	1	-1	-1	-1	1	-1	0	2	0
3	-1	1	-1	1	-1	-1	1	1	-1
4	-1	-1	-1	1	1	-1	2	2	-2

Rovnice přímky pro **čtvrtý** tréninkový vzor:

$$x_2 = -x_1 + 1$$



Adaptace Hopfieldovy sítě

- Hlavní myšlenka adaptace Hopfieldova modelu spočívá v tom, že jsou nejprve **inicializovány všechny neurony** sítě buď binárními hodnotami 0, 1 nebo **bipolárními** hodnotami -1,+1.
- Vzhledem k tomu, že **jsou** všechny neurony navzájem **propojeny**, **začnou se ovlivňovat**.
- To znamená, že **jeden neuron se snaží ostatní neurony excitovat** na rozdíl od jiného, který se **snaží o opačné**.
- Probíhá **cyklus postupných změn excitací neuronů** až do okamžiku **nalezení kompromisu** - síť **relaxovala** do **stabilního stavu**.

- Jinými slovy **výstupy předchozího kroku** se staly novými **vstupy současného kroku**.
- Tento proces je vysvětlitelný následujícím algoritmem: **tréninkové vzory nejsou v Hopfieldově síti uloženy přímo**, ale jsou **reprezentovány pomocí vztahů mezi stavy neuronů**.

Adaptační algoritmus Hopfieldovy sítě

- **První** popis adaptačního algoritmu Hopfieldovy sítě pochází z roku 1982 a používá **binární hodnoty** pro excitace neuronů.
- Požadovaná **funkce sítě** je specifikována tréninkovou množinou *P* vzorů **s***(p)*, *p = 1, ..., P*, z nichž každý je zadán vektorem *n* **binárních** stavů **vstupních** resp. **výstupních** neuronů, které v případě **autoasociativní paměti splývají**:

$$\mathbf{s}(p) = \left(s_1(p), \ldots, s_i(p), \ldots, s_n(p)\right),$$

potom je váhová matice **W** = {w_{ij}} dána následujícím vztahem:

$$w_{ij} = \sum_p \left[2s_i(p) - 1\right]\left[2s_j(p) - 1\right] \qquad \text{pro } i \neq j$$

a

$$w_{i\,i} = 0.$$

Adaptační algoritmus Hopfieldovy sítě

- Jiný popis adaptačního algoritmu Hopfieldovy sítě pochází z roku 1984 a pracuje s **bipolárními hodnotami pro excitace neuronů**.
- Požadovaná funkce sítě je rovněž specifikována tréninkovou množinou *P* vzorů **s***(p)*, *p = 1, ..., P*, z nichž každý je zadán vektorem *n* **bipolárních** stavů **vstupních** resp. **výstupních neuronů**, které v případě **autoasociativní paměti splývají**:

$$\mathbf{s}(p) = \left(s_1(p), \ldots, s_i(p), \ldots, s_n(p)\right),$$

potom je váhová matice **W** = {w_{ij}} dána následujícím vztahem:

$$w_{ij} = \sum_p s_i(p)\, s_j(p) \qquad \text{pro } i \neq j$$

a

$$w_{i\,i} = 0.$$

Adaptační algoritmus Hopfieldovy sítě

Krok 0. **ADAPTAČNÍ FÁZE**

Inicializace vah, tj. **zapamatování vzorů**.

Použitím Hebbova adaptačního pravidla.

AKTIVAČNÍ FÁZE - RELAXACE

Dokud síť nezrelaxovala do stabilního stavu,

opakovat kroky (1 až 7).

Krok 1. Pro každý vstupní vektor ***x***, opakovat kroky (2 až 6).

Krok 2. Inicializace sítě vnějším vstupním vektorem $\boldsymbol{x}$:

$$y_i = x_i, \quad (i = 1, \ldots, n)$$

Krok 3. Pro každý neuron Y_i opakovat kroky (4 až 6).

(Neurony jsou uspořádány náhodně)

Krok 4 Vypočítat vnitřní potenciál neuronu:

$$y\_in_i = x_i + \sum_j y_j w_{ji}.$$

Krok 5 Stanovení výstupu neuronu lze chápat jako aplikaci aktivační funkce:

$$y_i = \begin{cases} 1 & \text{pokud } y\_in_i > \theta_i \\ y_i & \text{pokud } y\_in_i = \theta_i \\ 0 & \text{pokud } y\_in_i < \theta_i \end{cases}.$$

Prahová hodnota θ_i je obvykle nulová. Aktualizace neuronů probíhají sice v **náhodném pořadí**, ale musí být prováděny **stejnou průměrnou rychlostí**.

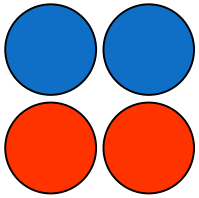
Krok 6 Transport hodnoty y_i ostatním neuronům.

(Takto budeme aktualizovat hodnoty aktivačního vektoru.)

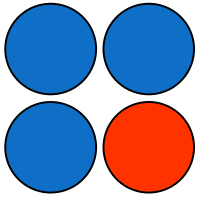
Krok 7. Test konvergence.

Učení

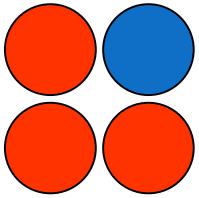
$$w_{ij} = \begin{cases} \sum_{k=1}^{s} x_{ki} x_{kj} & pro \quad i \neq j \\ 0 & pro \quad i = j, \end{cases}$$



$x_1=(1,1,-1,-1)$



$x_2=(1,1,1,-1)$

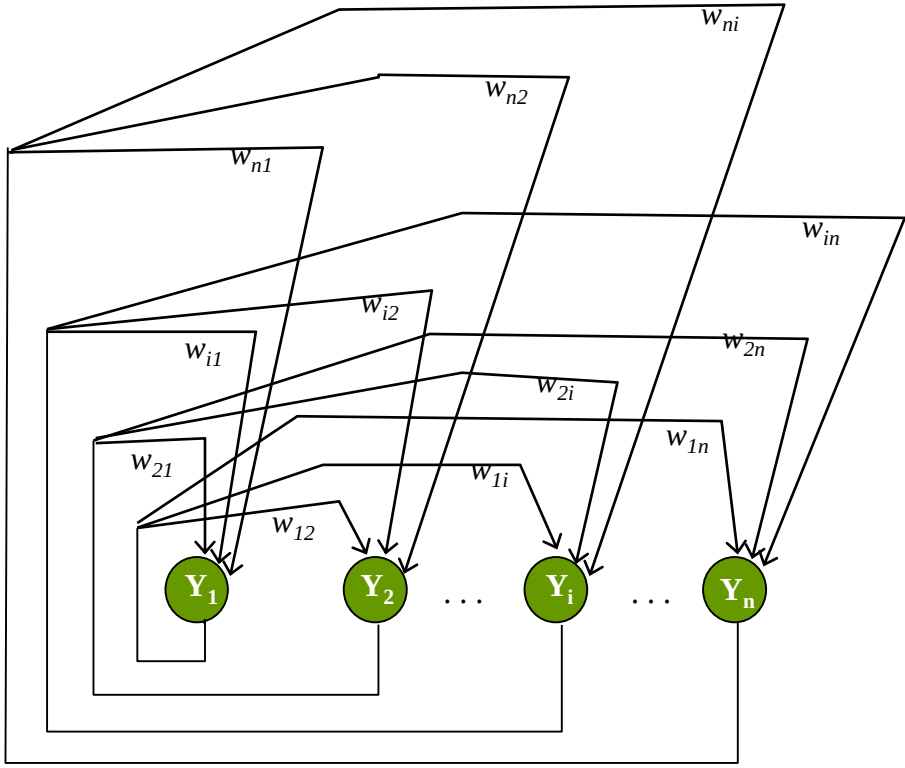


$x_3=(1,-1,-1,-1)$

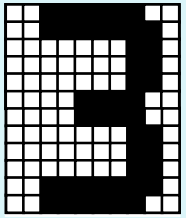
$$E(x) = -\frac{1}{2} \boldsymbol{x}^T \boldsymbol{W} \boldsymbol{x}$$

$$= -\frac{1}{2} \sum_{i,j} w_{ji} x_i x_j$$

$$\begin{pmatrix} 0 & 1 & 1 & -1 \\ 1 & 0 & -1 & -3 \\ 1 & -1 & 0 & 1 \\ -1 & -3 & 1 & 0 \end{pmatrix}$$



Vybavování



$$y_i(0) = x_i$$

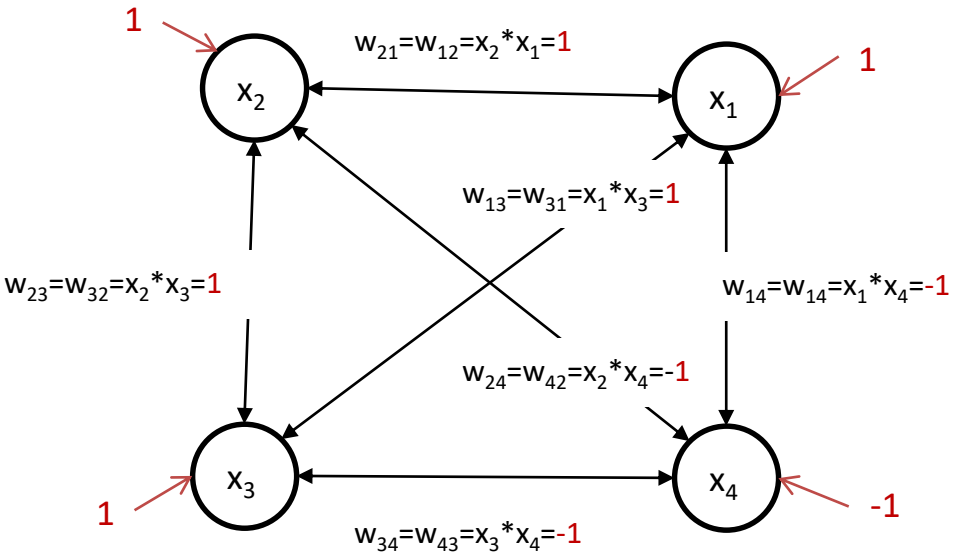
$$y_j(t+1) = f\left(\sum_{i=1}^{n} w_{ij} y_i(t) \right)$$

Příklad

Pomocí diskrétního Hopfieldova modelu určete, zda je vstupní vektor (0, 0, 1, 0**) „naučeným“ vzorem (tj. zda byl součástí trénovací množiny).**

Váhová matice pro zapamatování vektoru **(1, 1, 1, 0)** (v bipolární reprezentaci **(1, 1, 1, -1)**) má tvar:

$$\mathbf{W} = \begin{bmatrix} 0 & 1 & 1 & -1 \\ 1 & 0 & 1 & -1 \\ 1 & 1 & 0 & -1 \\ -1 & -1 & -1 & 0 \end{bmatrix}$$

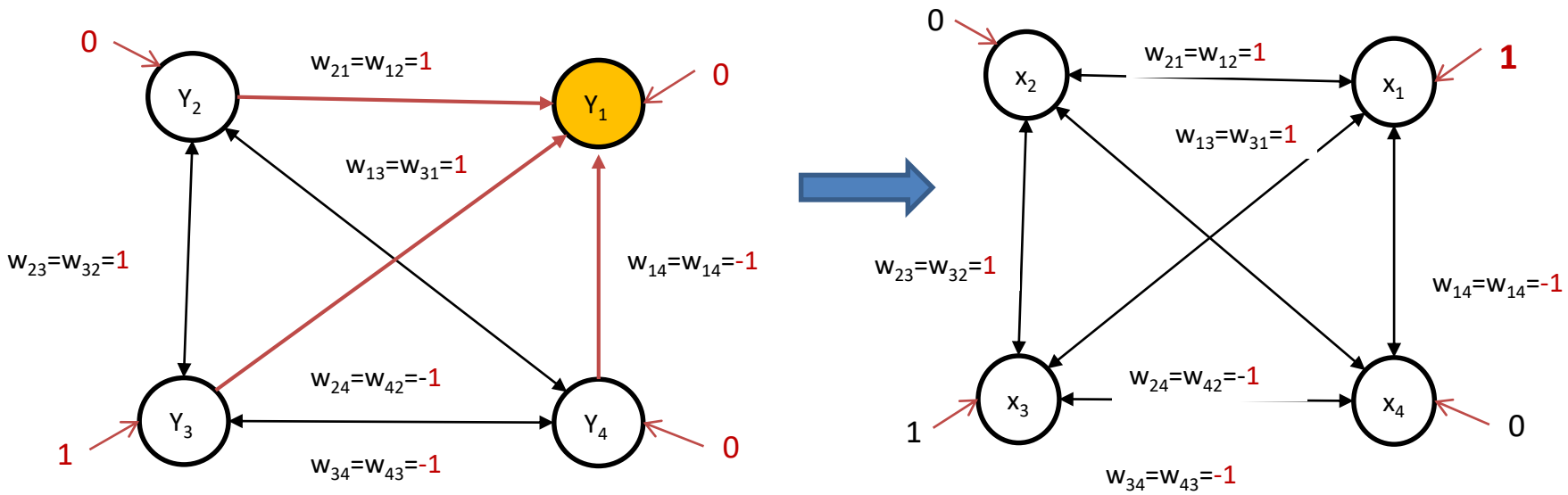


Vstupní vektor **(0, 0, 1, 0)** :

Vybrat $\mathbf{Y_1}$ a aktualizovat jeho aktivaci:

$$y\_in_1 = x_1 + \sum_j y_j w_{j1} = 0+0+1+0 = 1$$

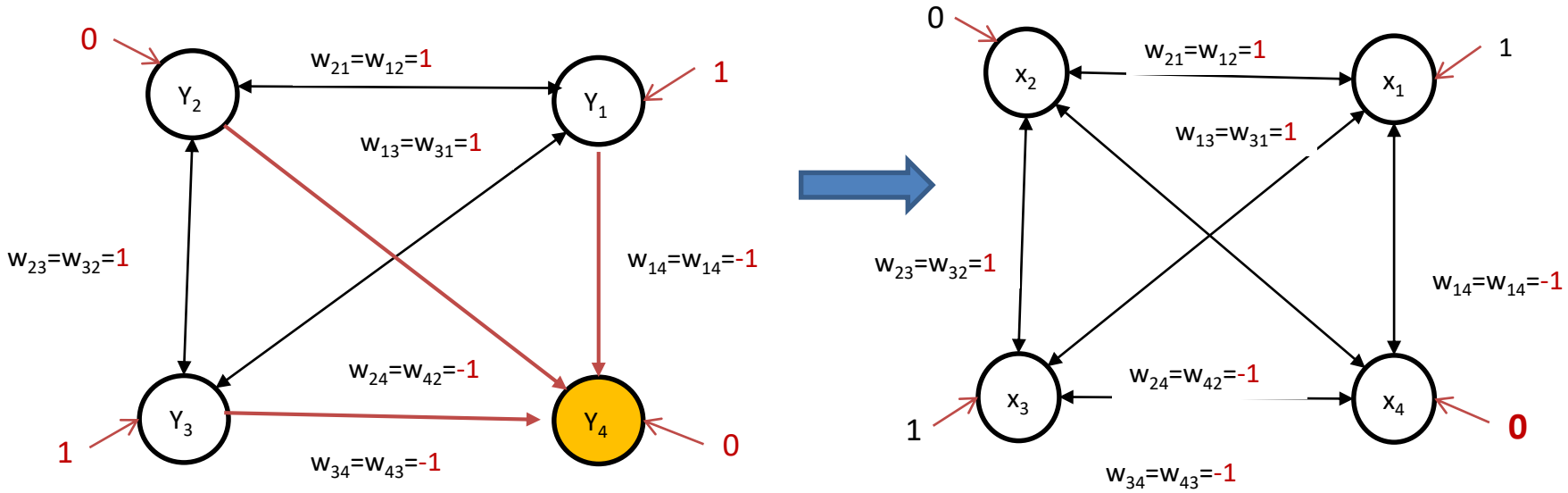
$y\_in_1 > 0 \rightarrow y_1 = 1$



Vybrat $\mathbf{Y_4}$ a aktualizovat jeho aktivaci:

$$y\_in_4 = x_4 + \sum_j y_j w_{j4} = 0 - 1 + 0 - 1 = -2$$

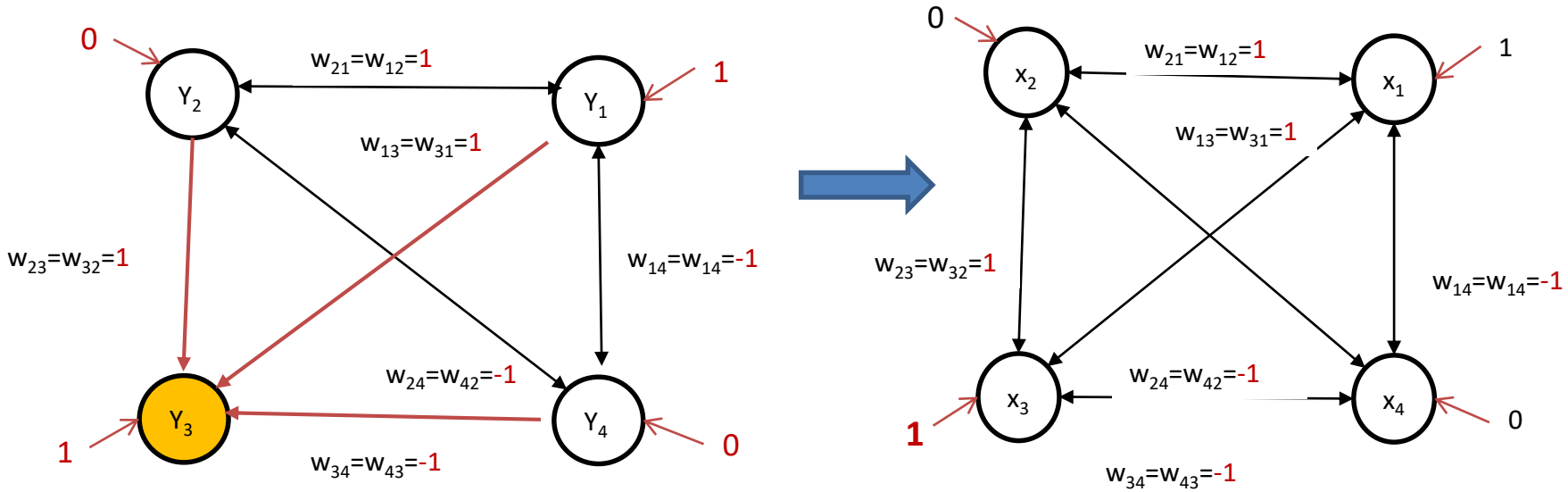
$y\_in_4 < 0 \rightarrow y_4 = 0$



Vybrat $\mathbf{Y_3}$ a aktualizovat jeho aktivaci:

$$y\_in_3 = x_3 + \sum_j y_j w_{j3} = 1+1+0+0 = 2$$

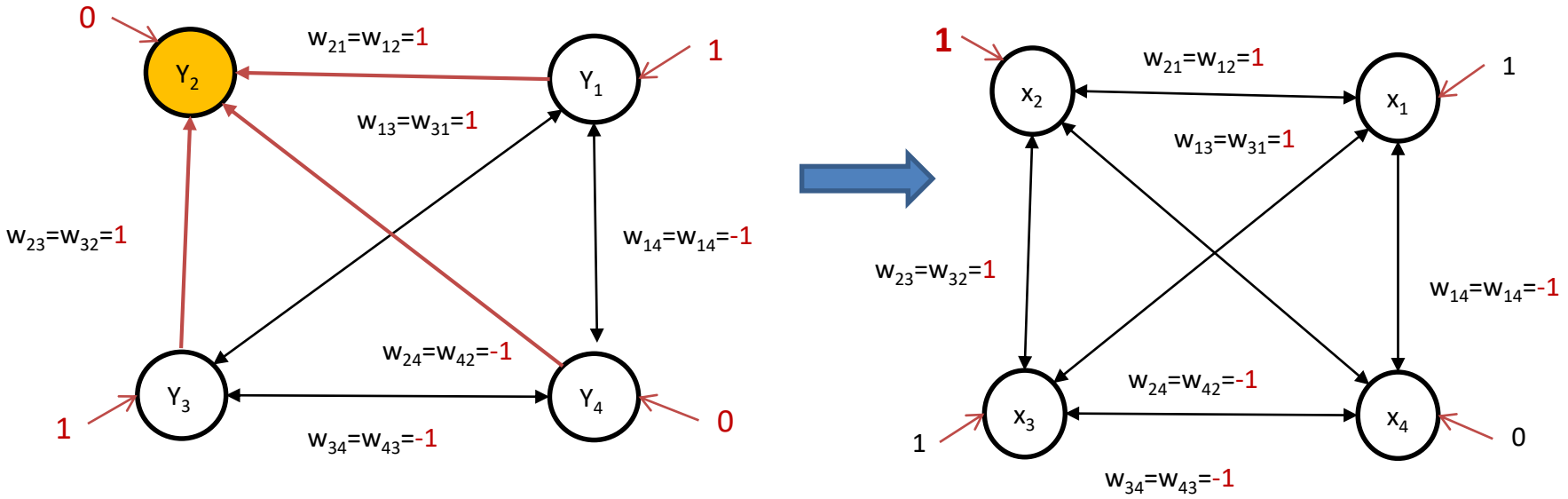
$y\_in_3 > 0 \rightarrow y_3 = 1$



Vybrat $\mathbf{Y_2}$ a aktualizovat jeho aktivaci:

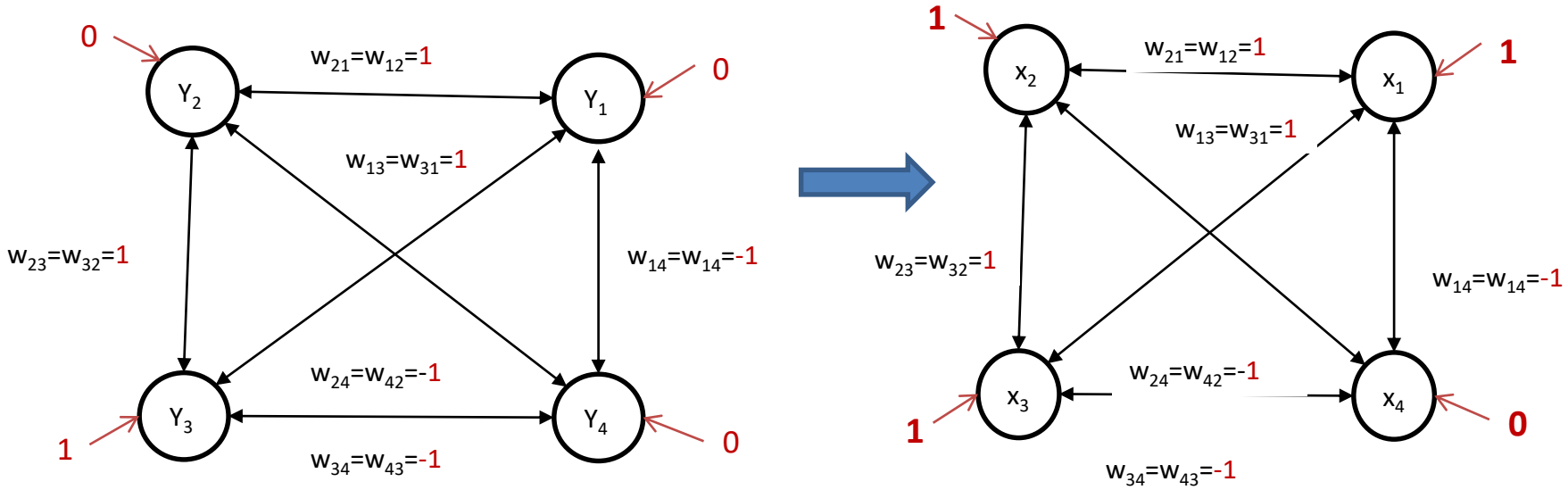
$$y\_in_2 = x_2 + \sum_j y_j w_{j2} = 0+1+1+0 = 2$$

$y\_in_2 > 0 \rightarrow y_2 = 1$



Shrnutí:

Aktivace každého neuronu byla aktualizována alespoň jednou během celého výpočtu. *Vstupní vektor* **(**0, 0, 1, 0) *konverguje k uloženému vzoru* **(1, 1, 1, 0)** (v bipolární reprezentaci **(1, 1, 1, -1).**



Energetická funkce Hopfieldovy sítě

- K lepšímu pochopení **aktivní dynamiky** Hopfieldovy sítě byla Hopfieldem, **v analogii s fyzikálními ději** definována tzv. *energetická funkce E* sítě, která **každému stavu sítě přiřazuje jeho potenciální energii**.
- Energetická funkce je **zdola ohraničená** a pro daný stav systému je **nerostoucí**.
- V teorii neuronových sítí se *stavem systému* rozumí množina aktivací všech neuronů.
- Pokud je již tato energetická funkce **nalezena**, bude síť konvergovat **ke stabilní množině aktivací neuronů** v daném časovém okamžiku.

- Energetická funkce pro diskrétní Hopfieldovu síť je dána následovně: $$E = -0{,}5\sum_{i \neq j}\sum_{j} y_i y_j w_{ij} - \sum_{i} x_i y_i + \sum_{i} \theta_i y_i .$$
- Z definice **energetické funkce** vyplývá, že **stavy sítě s nízkou energií mají největší stabilitu**.
- **Hopfieldova síť** má ve srovnání s vícevrstvou sítí adaptovanou učícím algoritmem **backpropagation** *opačný charakter aktivní a adaptivní dynamiky*.
- Zatímco **adaptace Hopfieldovy sítě** podle Hebbova zákona je **jednorázovou záležitostí**, jejíž trvání **závisí jen na počtu tréninkových vzorů**, učící algoritmus **backpropagation** realizuje **iterativní proces** minimalizující chybu sítě **gradientní metodou** bez záruky konvergence.

- V jeho **blízkém okolí**, v tzv. *oblasti atrakce*, se nachází všechny **vstupy blízké** tomuto **vzoru**.
- Ty představují **počáteční stavy sítě**, ze kterých se při **minimalizaci energetické funkce** v **aktivním režimu** síť dostane do příslušného **minima**, tj. **stabilního stavu** odpovídajícího uvažovanému **tréninkovému vzoru**.
- Geometricky se tedy **energetická plocha rozpadá** na oblasti **atrakce lokálních minim** a příslušná **funkce Hopfieldovy sítě** přiřadí **v aktivním režimu** ke každému **vstupu** náležejícímu do **oblasti atrakce** nějakého lokálního minima **právě toto minimum**.

- Na druhou stranu **délka trvání aktivní fáze** vícevrstvé sítě **je dána pouze počtem vrstev**, zatímco **aktivní režim Hopfieldovy** sítě **představuje iterativní proces minimalizující energii sítě** diskrétní variantou gradientní metody s nejistou konvergencí.
- **Cílem adaptace Hopfieldovy sítě** podle Hebbova zákona je nalezení takové konfigurace, aby funkce sítě v aktivním režimu **realizovala autoasociativní paměť**.
- To znamená: bude-li **vstup sítě blízký** nějakému **tréninkovému vzoru**, **výstup sítě** by měl potom **odpovídat tomuto vzoru**.
- Z hlediska energie by každý tréninkový **vzor** měl být lokálním **minimem energetické funkce**, tj. **stabilním stavem sítě**.

- Při **učení** Hopfieldovy sítě podle Hebbova zákona pro asociativní sítě **samovolně vznikají** na energetické ploše **lokální minima**, tzv. *nepravé vzory* (*fantomy*), které **neodpovídají žádným tréninkovým vzorům**.
- **Výstup sítě pro vstup** dostatečně **blízký** takovému **fantomu neodpovídá žádnému vzoru**, a tudíž nedává žádný smysl.
- Existují varianty adaptivní dynamiky Hopfieldovy sítě, při nichž se takto vzniklé fantomy mohou dodatečně odučit.

Kapacita Hopfieldovy paměti

- Hopfield experimentálně nalezl, že počet binárních vzorů, který může být zapamatován a opětovně vyvolán s požadovanou přesností, je dán přibližně **P ≈ 0,15n**, kde **n** je počet neuronů v síti.
- Pro sítě pracující s bipolárními vzory byl odvozen obdobný vztah:

$$P = \frac{n}{2 \log_2 n}$$

- I když se v praxi ukazuje, že uvedené teoretické odhady jsou poněkud nadhodnocené, přesto základní model Hopfieldovy autoasociativní paměti má díky své malé kapacitě spíše teoretický význam.

Aplikace neuronových sítí

Aplikační oblasti neuronových sítí

- Neuronové sítě v současnosti patří mezi významnou část **počítačově orientované umělé inteligenci**.
- Dále mají neuronové sítě uplatnění v **kognitivní vědě**, l**ingvistice**, **neurovědě**, **řízení procesů**, **přírodních a společenských vědách** apod.
- Původním cílem výzkumu neuronových sítí byla snaha **pochopit a modelovat** způsob, jakým **myslíme** a způsob, jak **funguje lidský mozek**.
- Nejvýznamnější oblasti použití umělých neuronových sítí **kromě úloh z oblasti klasifikace** a rozpoznávání vzorů jsou následující:

Klasifikace

- Klasifikace je činnost, při které se posuzované objekty zařazují do příslušných tříd. *Z matematického hlediska lze tyto činnosti nazývat funkční aproximací*.
- Klasifikace je jednou z oblastí, kde se **neuronové sítě** velmi dobře uplatňují.
- Klasifikace znamená v podstatě **ohodnocení** daného problému a jeho **zařazení** do příslušné třídy.

Příklady

Klasifikace je aktivita, kterou člověk provádí, aniž by si to uvědomoval.

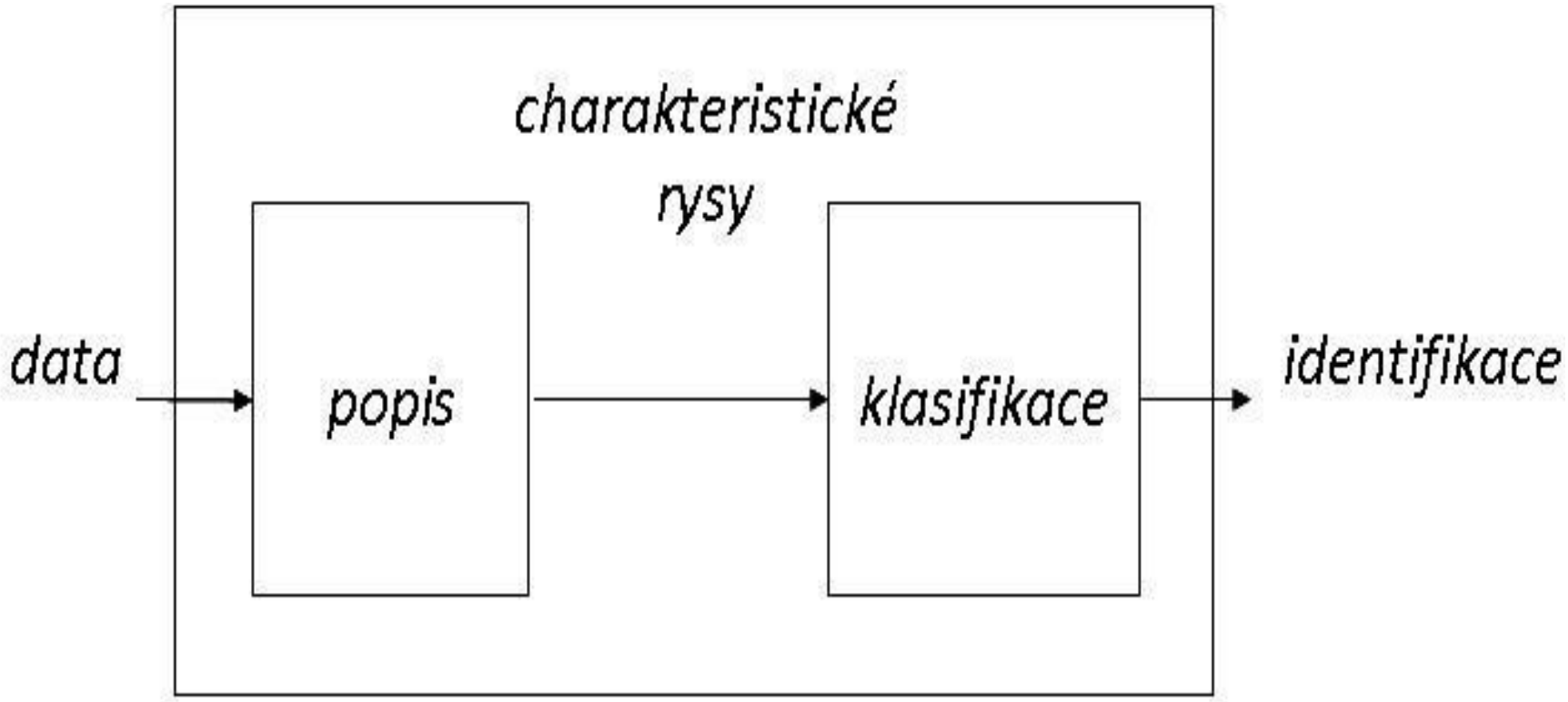
- Pokud **v obchodě vybíráme** mezi dvěma výrobky, musíme rozhodnout, který je horší a který lepší (nyní není podstatné z jakého hlediska).
- Pokud **chceme jít do kina**, obvykle předem oklasifikujeme film a na základě tohoto výsledku se rozhodneme.
- Pokud se nad otázkou klasifikace v našem životě zamyslíte, jistě sami najdete mnoho **podobných příkladů**.

Předzpracování dat a vlastní klasifikace

Obecně je klasifikace prováděna ve dvou dílčích krocích:

1. **Výběr klíčových vlastností** - vstupní data jsou nejprve předzpracována algoritmem, který extrahuje klíčové vlastnosti ze vstupních objektů, např. eliminaci vlivu zašumění, posunutí, otočení nebo poškození ve vstupních datech na klasifikaci.
   Pro **extrakci charakteristických rysů** neexistuje žádné obecné pravidlo. Jejich výběr souvisí s **danou aplikací** a závisí na **typu dat**, ale velkou roli zde hrají **zkušenosti a intuice experta**.
2. **Vlastní klasifikace** - vlastnosti extrahované z objektů jsou předloženy **klasifikátoru** k roztřídění. Klasifikátor tedy nepracuje přímo s *objekty*, ale s jejich **obrazy** připravenými během předzpracování.

Obecné schéma klasifikátoru



Čím je kvalitnější proces předzpracování vstupních dat, tím jednodušší klasifikační algoritmus lze použít. Přitom mohou nastat dvě extrémní situace:

- **Dokonalé předzpracování.** Výstupem dokonalého předzpracování je **číslo třídy**, do níž vstupní objekt patří. Úloha je tedy **vyřešena** hned **v prvním kroku** a použití klasifikátoru není nutné.
- **Dokonalý klasifikátor**. V tomto případě je použit klasifikátor, který je dostatečně "inteligentní" na to, aby všechny vstupní objekty správně **zařadil přímo**, **bez nutnosti** jakéhokoliv **předzpracování**, tj. dokáže **samostatně** vydedukovat všechny klíčové vlastnosti ze vstupních objektů.
- Klasifikátory zpravidla vždy tíhnou k **jednomu** z obou dílčích kroků více, tj. **leží mezi** právě popsanými oběma extrémy

Reprezentace dat

Vstupní data mohou být systému předkládána v různé podobě. V zásadě můžeme rozlišit dvě **základní možnosti**:

- **Číselné vyjádření** sledovaných parametrů.
- **Obrazová data** (využití metod tzv. strojového vidění), i když v konečném důsledku lze výslednou bitmapu rovněž reprezentovat čísly.

Zobrazení vícerozměrných dat tabulkou

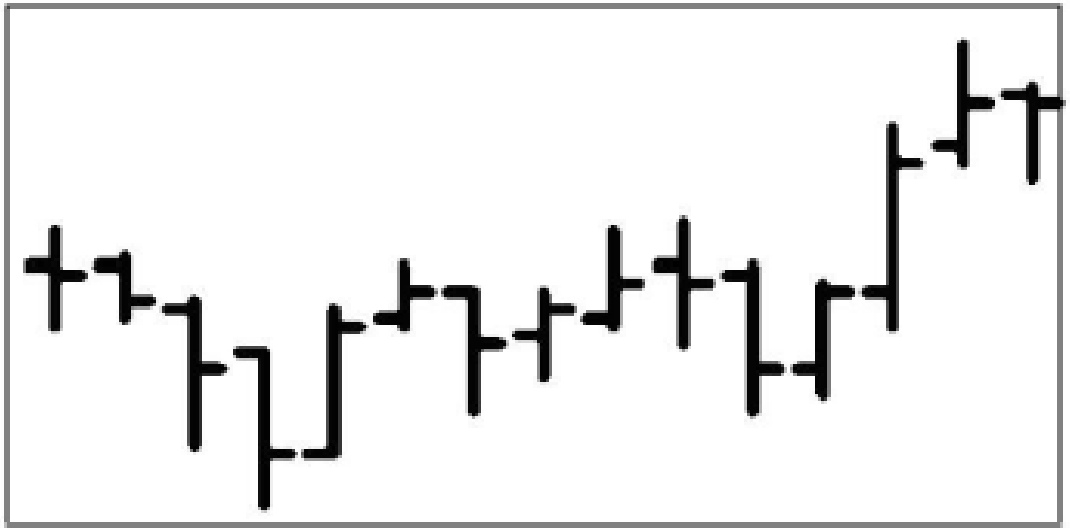
Číselné vyjádření jednoho konkrétního úseku OHLC dat (Open, High, Low, Close)

DATE	TIME	OPEN	HIGH	LOW	CLOCE	VALUE
2010.06.02	8:15	1.22220	1.22260	1.22140	1.22210	107
2010.06.02	8:20	1.22220	1.22230	1.22150	1.22170	76
2010.06.02	8:25	1.22160	1.22170	1.27990	1.22090	71
2010.06.02	8:30	1.22110	1.22110	1.22910	1.21970	85
2010.06.02	8:35	1.21980	1.22160	1.22970	1.22140	78
2010.06.02	8:40	1.22150	1.22220	1.22140	1.22190	61
2010.06.02	8:45	1.22180	1.22190	1.22030	1.22120	84
2010.06.02	8:50	1.22130	1.22180	1.22070	1.22160	71
2010.06.02	8:55	1.22150	1.22260	1.22140	1.22200	61
2010.06.02	9:00	1.22220	1.22270	1.22120	1.22200	91
2010.06.02	9:05	1.22210	1.22220	1.22030	1.22090	53
2010.06.02	9:10	1.22080	1.22200	1.22050	1.22190	87
2010.06.02	9:15	1.22180	1.22390	1.22140	1.22350	90
2010.06.02	9:20	1.22340	1.22500	1.22350	1.22430	87
2010.06.02	9:25	1.22440	1.22450	1.22330	1.22430	77

Obrazová reprezentace vzoru

Obrazová reprezentace vzoru obsahuje pouze informace z 3. -6. sloupce tabulky. Přesto je velikost vzoru (počet pixelů) rovna 7440.

Tabulkové vyjádření celého vzoru vystačí s 1680 bity (15 řádků a 7 sloupců při rozlišení 16 bitů na číslo).

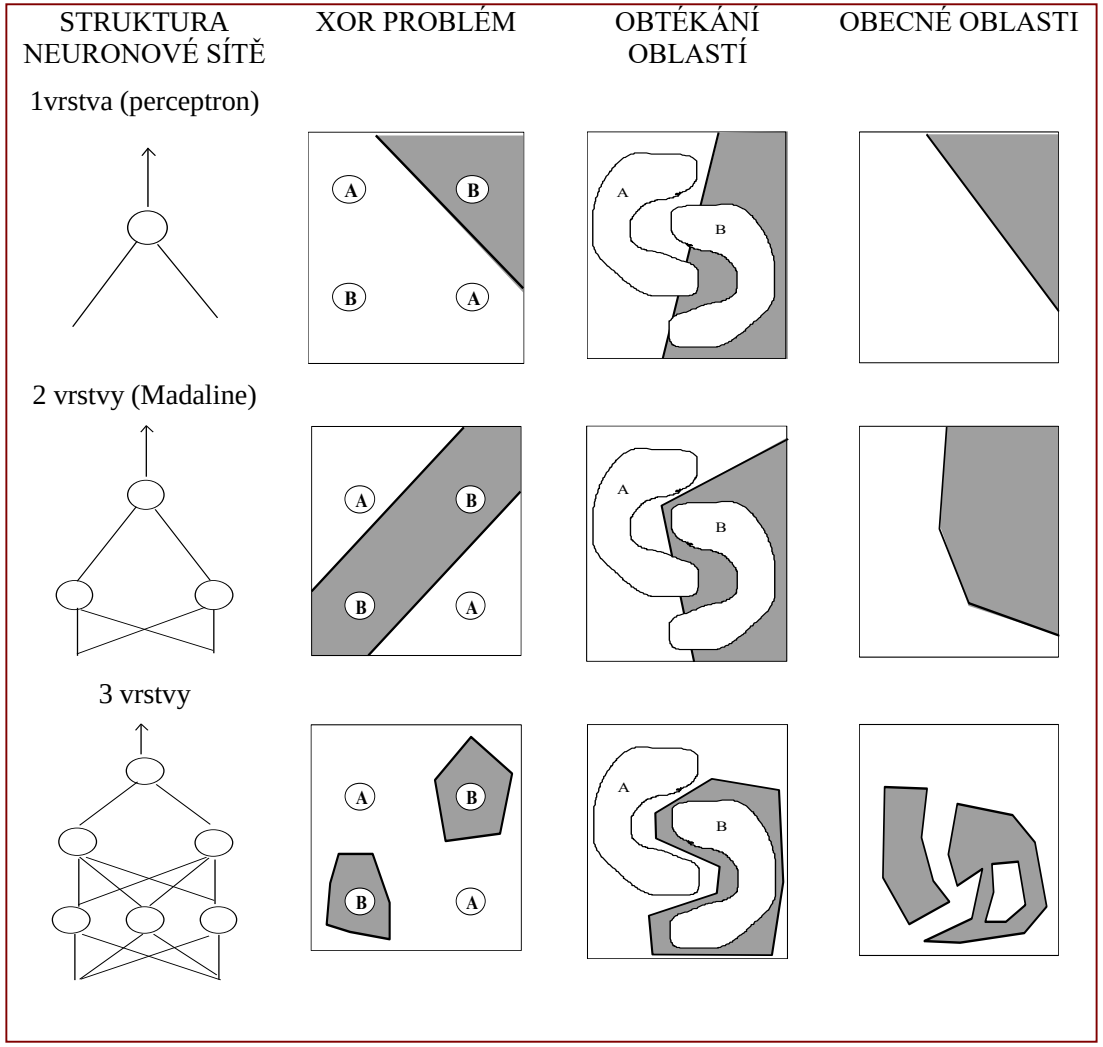


Výhodou **obrazových dat** je, že lépe korespondují s intuitivní lidskou představou o rozpoznávání vzorů.

Neuronové sítě a možnosti klasifikace

Klasifikace je jednou z **hlavních aplikačních oblastí** pro umělé neuronové sítě.

Na obrázku jsou zobrazeny různé typy neuronových sítí (tj. neuronové sítě s různým počtem vnitřních vrstev) a jejich možnosti klasifikace.



Použití klasifikátorů na bázi umělých neuronových sítí

V mnoha případech se jedná o systémy, kde klasifikace reprezentuje pouze součást řešení a celkové řešení využívá řadu dalších metod.

- OCR programy, které musí na digitalizovaném dokumentu rozpoznat jednotlivá písmena (OCR software se dodává k většině scannerů).
- kontrola dopravy, kde se klasifikace používá při detekci a rozpoznávání poznávacích značek aut
- průmyslová kontrola kvality výrobků
- predikce trendu časové řady (resp. vývoje ekonomických ukazatelů poskytuje podnikům informace pro strategické plánování)
- automatické řízení systému v reálném čase (adaptace systému na změny během jeho provozu v reálném čase)

Řízení v dynamicky se měnících podmínkách

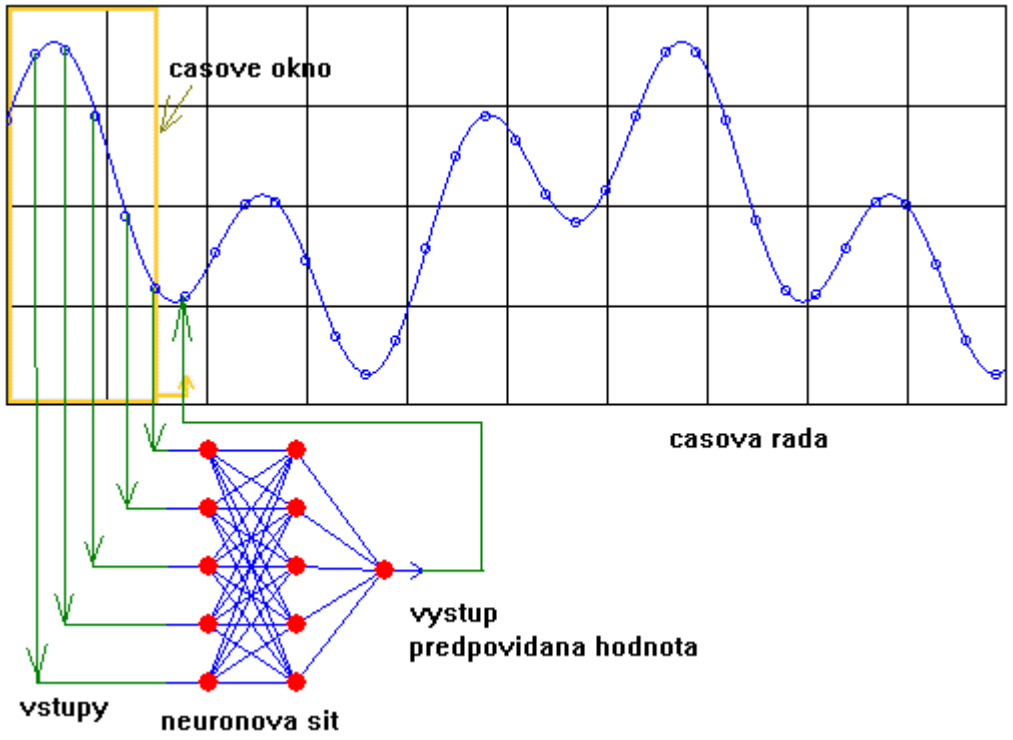
Demonstračním příkladem řídícího systému může být **autopilot automobilu**. Auto řízené neuronovou sítí určovalo na základě **vzdálenosti** a **rychlosti** nejbližších aut v obou pruzích svou **vlastní rychlost** a **změnu pruhu**. Dále neuronová síť **ovládala volant podle zakřivení dálnice**, **polohy auta v pruhu** a **aktuálního úhlu volantu**.

Je zajímavé, že neuronová síť se kromě úspěšného řízení vozidla včetně předjíždění naučila i **různé zvyky** a **styl jízdy** (např. riskantní rychlá jízda a časté předjíždění nebo naopak opatrná pomalá jízda) podle **řidičů - trenérů**, od kterých byly **získány tréninkové vzory**.

Predikce

Jinou důležitou aplikační oblastí neuronových sítí je ***predikce*** a příp. následné ***rozhodování***. Typickými příklady z této oblasti jsou **předpověď počasí**, vývoj cen akcií na burze, **spotřeba elektrické energie** apod.

Např. při meteorologické předpovědi jsou *vstupem* neuronové sítě **odečty základních parametrů** (např. teplota, tlak apod.) v čase a *učitelem* je skutečný **vývoj počasí** v následujícím období.



Analýza signálů

Jiným příkladem uplatnění neuronových sítí je **analýza signálů** jako **EKG, EEG** apod.

Spojitý signál je **vzorkován** ve **stejných časových intervalech** a několik posledních diskrétních hodnot úrovně signálu slouží jako **vstup** do neuronové sítě.

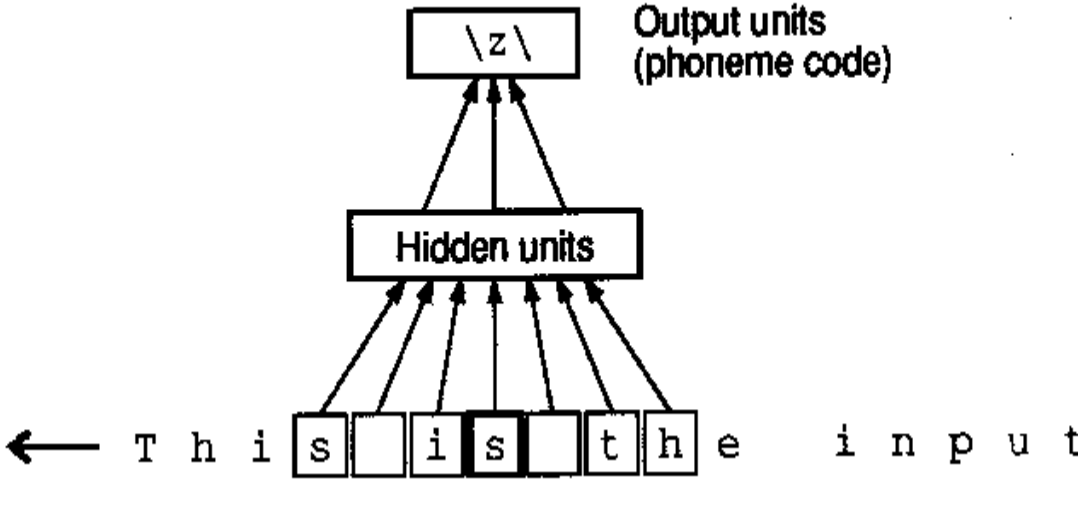
Naučená neuronová síť je schopna **identifikovat specifický tvar signálu**, který je důležitý pro **diagnostiku**.

Např. neuronová síť s topologií 40 - 17 - 1 byla použita pro klasifikaci EEG signálů se specifickými α-rytmy.

Transformace signálů

Systém NETtalk je určený pro **převod** anglicky psaného **textu** na **mluvený signál**. Funkce sítě je následující: vstupní text se postupně přesouvá u vstupních neuronů po jednom písmenu zprava doleva a přitom je aktivní právě ten **výstupní neuron**, který reprezentuje **fonén** odpovídající **prostřednímu** ze 7 písmen vstupního textu. V našem příkladě se čte prostřední písmeno „*S*“, kterému odpovídá fonén [z].

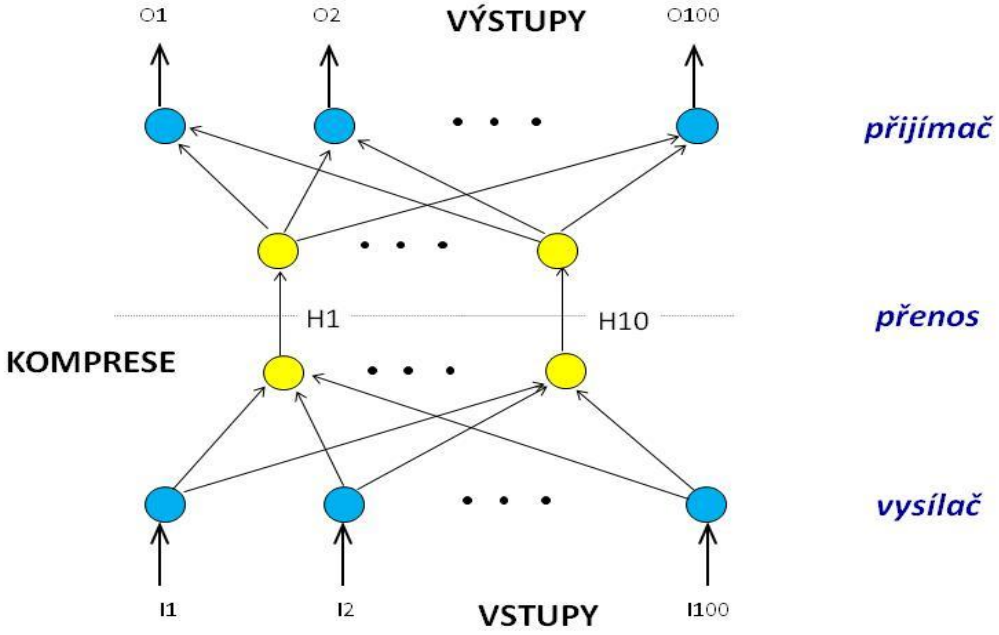
Úspěšná implementace systému NETtalk vedla ke snaze vytvořit systém založený na neuronové síti s obrácenou funkcí, která by převáděla mluvený jazyk do psané formy (tzv. **fonetický psací stroj**).



Komprese dat

Další možností využití neuronových sítí je **komprese dat** např. pro přenos **televizního signálu**, **telekomunikaci** apod. Počet neuronů ve **vnitřních vrstvách** je výrazně **menší** než je počet neuronů ve vstupní a výstupní vrstvě. Počet neuronů ve **vstupní** i **výstupní** vrstvě je **stejný**, protože obě vrstvy reprezentují **stejný signál**.

Stav **skrytých neuronů**, tj. **komprimovaný obraz** je přenášen k příjemci, který jej **dekóduje** výpočtem stavů **výstupních** neuronů. Tímto způsobem je získán téměř původní obraz.



Expertní systémy

Velkým problémem **klasických expertních systémů** založených na **pravidlech** je vytvoření **báze znalostí**, která bývá časově velmi náročnou záležitostí s nejistým výsledkem.

Neuronové sítě představují alternativní řešení, kde **reprezentace znalostí v bázi vzniká učením z příkladových inferencí**. V tomto případě **aktivní režim** neuronové sítě zastupuje funkci **inferenčního stroje**.

Univerzální neuronový expertní systém EXPSYS obohacuje vícevrstvou neuronovou síť o intervalovou aritmetiku **pro práci s nepřesnou informací** a o heuristiku analyzující síť, která umožňuje jednoduché **vysvětlení závěrů**.

Systém EXPSYS byl úspěšně aplikován v **energetice** a **medicíně**.

Aproximace funkcí

Dopředná třívrstvá neuronová síť je **univerzální aproximátor**, tj. je schopna **aproximovat** s požadovanou přesností **libovolnou spojitou funkci**.

Neuronové sítě mohou být chápané jako **univerzální prostředek** pro **regresní analýzu**, kde je **tvar funkce** určený **konfigurací neuronové sítě**.

Pod pojmem **konfigurace** neuronové sítě máme na mysli nejen **topologii** vzájemného propojení neuronů, ale také nastavení **váhových** a **prahových** koeficientů na těchto spojeních na určité hodnoty.

Použití NN

(1) Rozpoznávání vzorů. (2) Shlukování. Klasifikace. (3) Aproximace funkcí. (4) Predikce. (5) Optimalizace - TSP. (6) Filtrace šumu. (7) Řízení.