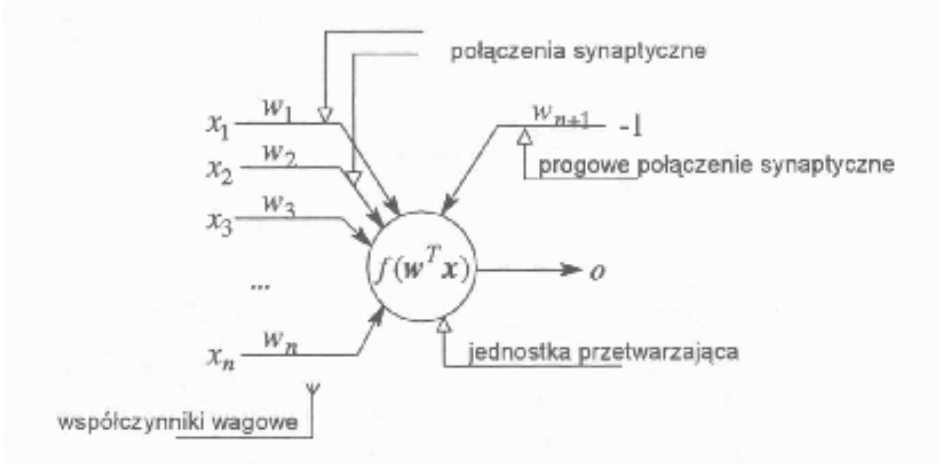
Inteligentne systemy decyzyjne: Uczenie maszynowe – sztuczne sieci neuronowe

wykład 2.

Trening jednokierunkowych sieci neuronowych

dr inż. Paweł Żwan
Katedra Systemów Multimedialnych
Politechnika Gdańska

Model matematyczny sztucznego neuronu



net=$w^T x$

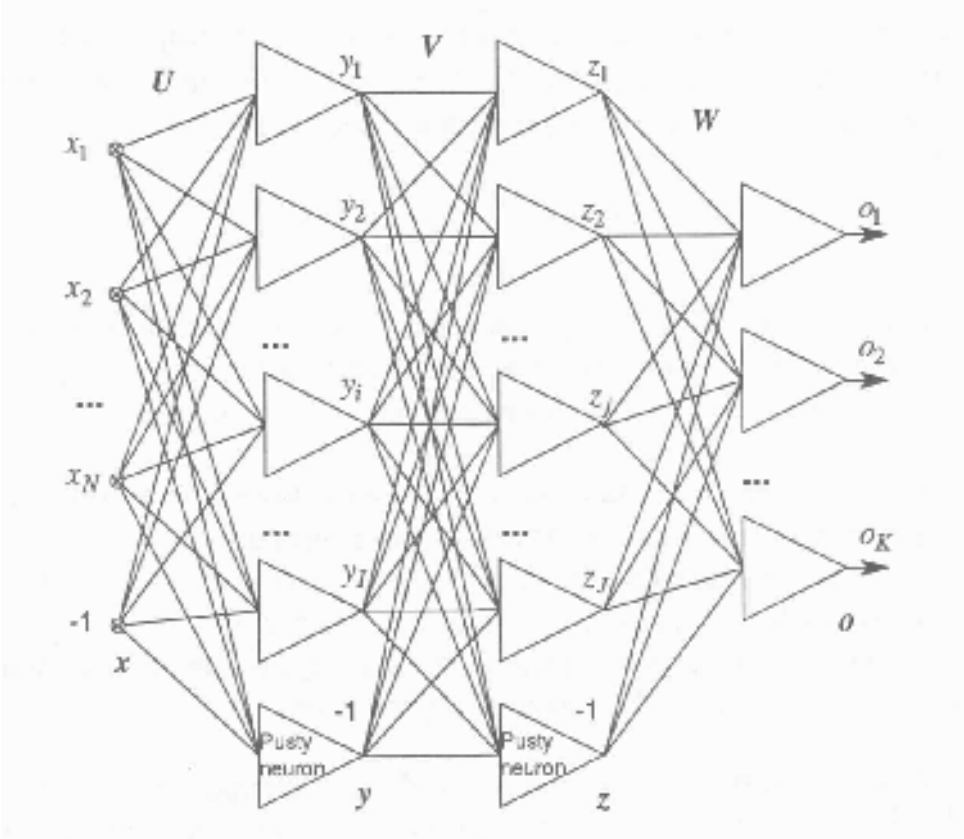
Wartość wyjściowa neuronu ***o*** jest określana w oparciu o wzór: o=***f***(**net**),

gdzie: ***w*** – wektor wag połączeń wejściowych
x – wektor wartości sygnałów wejściowych
f – funkcja aktywacji

Dodatkową wagą jest waga progowa, dlatego wektory w i x określone są jako:
$w=[w_1,w_2,\ldots,w_n,w_{n+1}]$, $x=[x_1,x_2,\ldots,x_n,-1]$

Topologia sieci jednokierunkowej

- **sieci jednokierunkowe**



$\boldsymbol{x}=[x_1,x_2,...,x_N,-1]$ – wektor wejściowy

$\boldsymbol{y}=[y_1,y_2,...,y_I,-1]$ – wektor wyjściowy pierwszej warstwy ukrytej

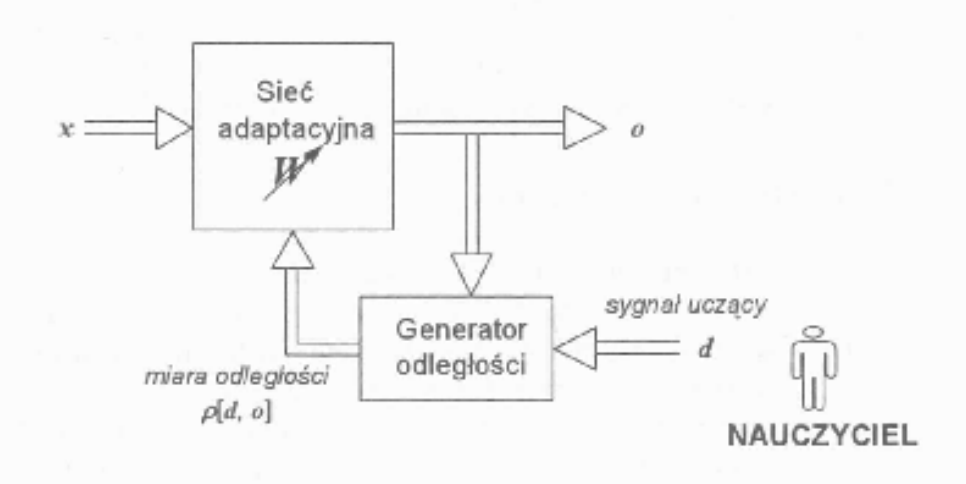
$\boldsymbol{z}=[z_1,z_2,...,z_J,-1]$ – wektor wyjściowy drugiej warstwy ukrytej

$\boldsymbol{o}=[o_1,o_2,...,o_K,-1]$ – wektor wyjściowy

Macierze **U**,**V**,**W** zawierają Współczynniki wagowe dla wszystkich Połączeń synaptycznych

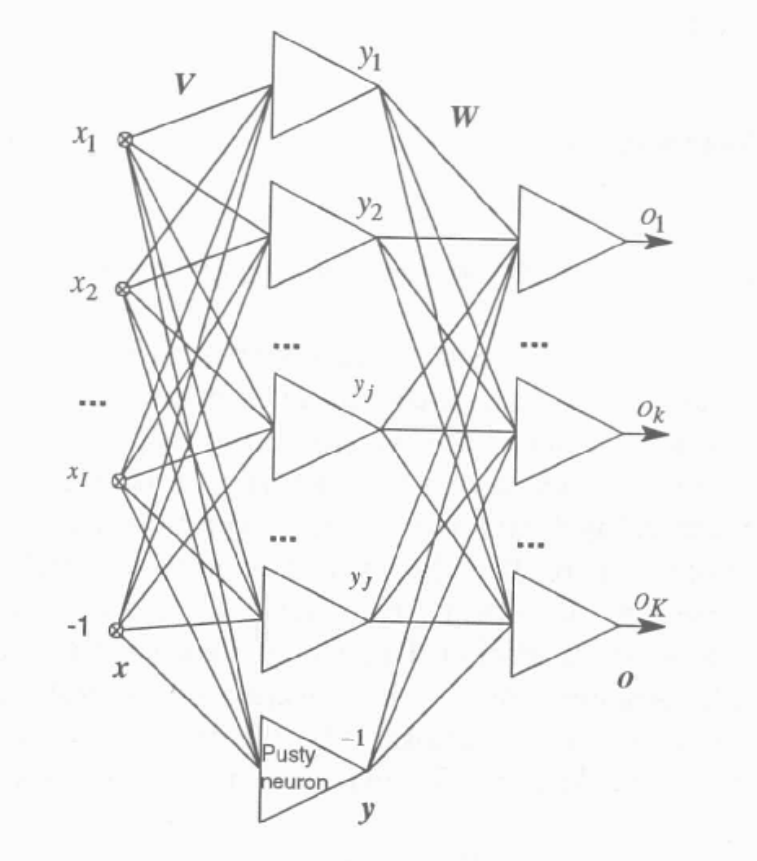
Metoda treningu sieci jednokierunkowej

trening z nadzorem



Reguła delta

- Rozważmy jednokierunkową sieć neuronową składającą się z dwóch warstw: ukrytej, zawierającej *J* neuronów oraz wyjściowej, zawierającej *K* neuronów



$\mathbf{x} = [x_1, \ldots, x_{I-1}, -1]^T$

$\mathbf{y} = [y_1, \ldots, y_{J-1}, -1]^T$

$\mathbf{o} = [o_1, \ldots, o_K]^T$

macierze wag warstw:

$$\mathbf{V} = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1I} \\ v_{21} & v_{22} & \cdots & v_{2I} \\ \cdots & \cdots & \cdots & \cdots \\ v_{J1} & v_{J2} & \cdots & v_{JI} \end{bmatrix}, \quad \mathbf{W} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1J} \\ w_{21} & w_{22} & \cdots & w_{2J} \\ \cdots & \cdots & \cdots & \cdots \\ w_{K1} & w_{K2} & \cdots & w_{KJ} \end{bmatrix}$$

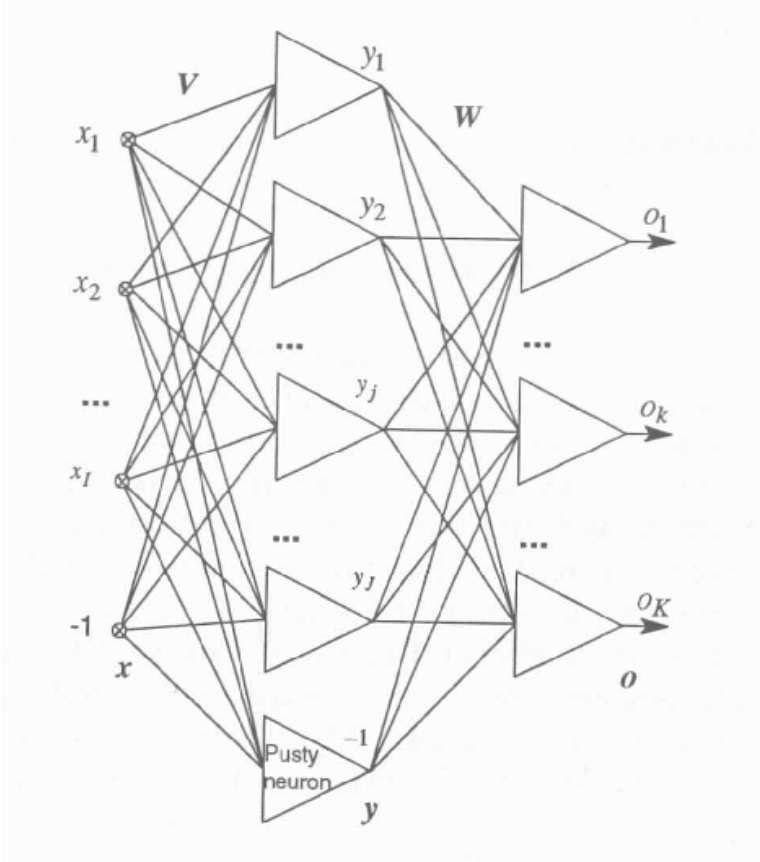
wektory pochodnych funkcji aktywacji:

$\mathbf{f_y'} = [f_1'(net_1), f_2'(net_2), \ldots, f_J'(net_J)]^T$

$\mathbf{f_o'} = [f_1'(net_1), f_2'(net_2), \ldots, f_K'(net_J)]^T$

Reguła delta – cd.

- Metoda uczenia jest metodą nadzorowaną, dlatego można określić miarę między wskazaniem sieci ***o***, a oczekiwaną odpowiedzią sieci ***d***, czyli tzw. funkcję błędu.



Najczęściej stosuje się odl.średniokwadratową:

$$E^n = \sum_{p=1}^{P}\sum_{k=1}^{K}\frac{1}{2}\cdot\left(d_k^{(p)} - o_k^{(p)}\right)^2 = \frac{1}{2}\cdot\sum_{p=1}^{P}\sum_{k=1}^{K}\left(d_k^{(p)} - o_k^{(p)}\right)^2$$

$$E^n = \frac{1}{2}\cdot\sum_{p=1}^{P}\left\|\mathbf{d}^{(p)} - \mathbf{o}^{(p)}\right\|^2$$

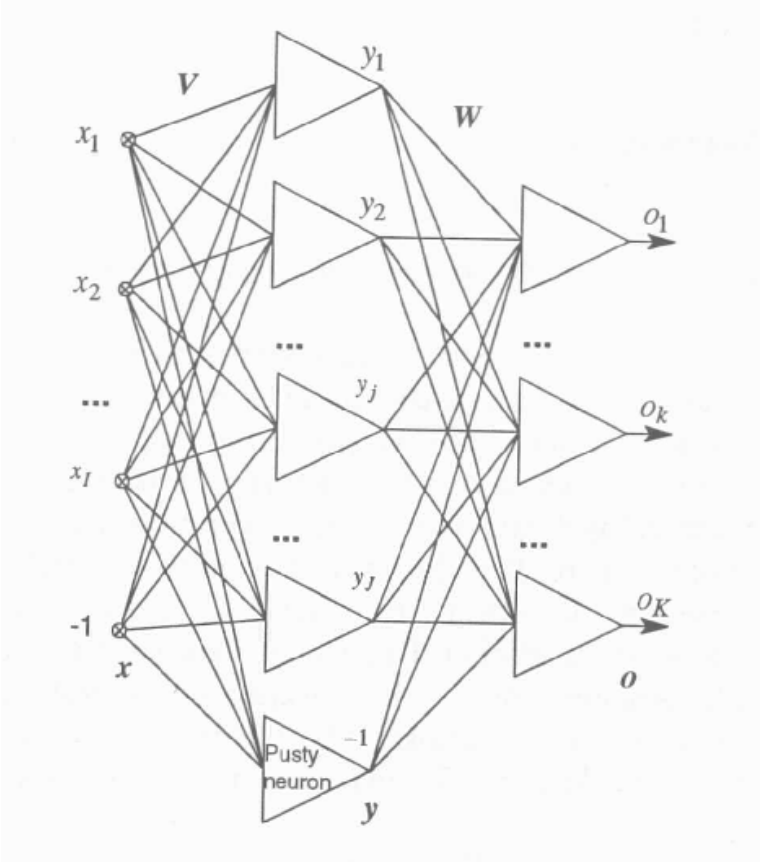
gdzie ***p*** jest liczbą wektorów w zb. uczącym

Błąd ten liczony jest dla wszystkich wektorów ze zbioru uczącego – **błąd skumulowany**. Podczas treningu prezentowane są kolejno wektory uczące, wówczas funkcja błędu dla ***p*** – tego wektora przyjmuje postać:

$$E(p)^n = \rho\left(\mathbf{d}^{(p)}, \mathbf{o}^{(p)}\right) = \frac{1}{2}\cdot\left\|\mathbf{d}^{(p)} - \mathbf{o}^{(p)}\right\|^2$$

Reguła delta – cd.

- W dalszej części przyjęto, że rozważania dotyczą pojedynczego ***p***-tego wektora ze zbioru uczącego.



Niech zdefiniowany zostanie operator:

$$\Gamma[q]=\begin{bmatrix} f_1(q_1) & 0 & \dots & 0 \\ 0 & f_2(q_2) & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & f_Q(q_Q) \end{bmatrix}$$

wtedy:

$$\mathbf{o}=\Gamma[\mathbf{W}\mathbf{y}]=\Gamma[\mathbf{W}\cdot\Gamma[\mathbf{V}\mathbf{x}]]$$

Funkcja błędu:

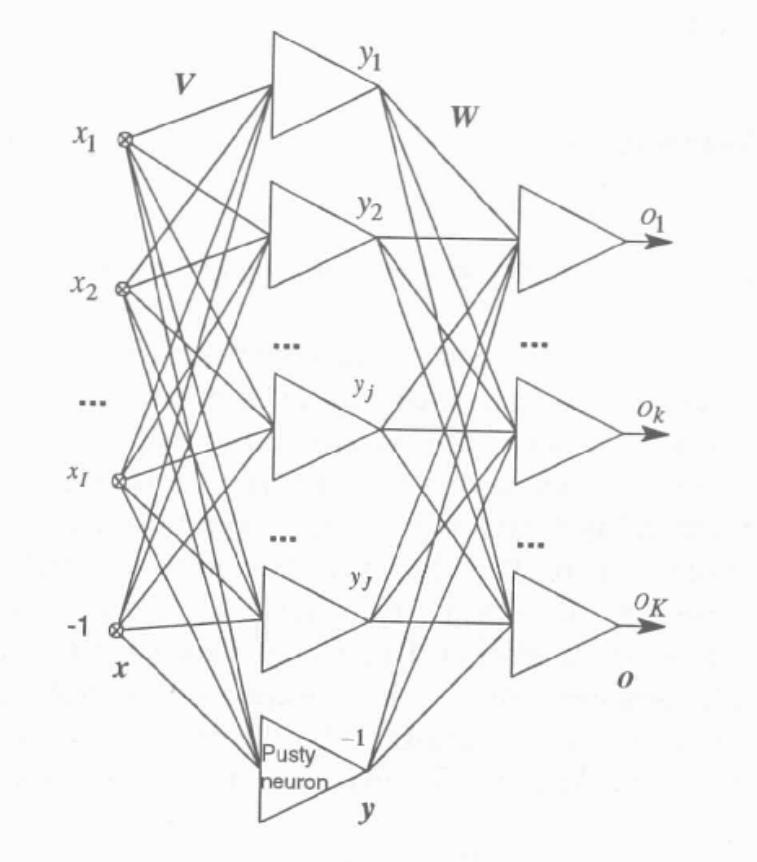
$$E(p)^n=\rho\left(\mathbf{d}^{(p)},\mathbf{o}^{(p)}\right)=\frac{1}{2}\cdot\left\|\mathbf{d}^{(p)}-\mathbf{o}^{(p)}\right\|^2$$

przyjmuje postać:

$$\delta_o=-\nabla E(\mathbf{o}) \text{ lub } \delta_{ok}=-\frac{\partial E}{\partial net_k}$$ - warstwa wyjściowa

$$\delta_y=-\nabla E(\mathbf{y}) \text{ lub } \delta_{yj}=-\frac{\partial E}{\partial net_j}$$ - warstwa ukryta

Reguła delta – cd.



wtedy:

$$\delta_o = \Phi[\mathbf{d}-\mathbf{o}]\cdot \mathbf{f}'_o \quad \text{lub} \quad \delta_{ok} = (d_k - o_k)\cdot f'(net_k)$$

$$\delta_y = \mathbf{w}_j^T \cdot \delta_o \cdot \mathbf{f}'_y \quad \text{lub} \quad \delta_{yj} = f'_j(net_j)\cdot \sum_{k=1}^{K}(d_k - o_k)\cdot f'_k(net_k)\cdot w_{kj}$$

gdzie wprowadzono pomocniczy operator:

$$\Phi[q] = \begin{bmatrix} q_1 & 0 & \dots & 0 \\ 0 & q_2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & q_Q \end{bmatrix},$$

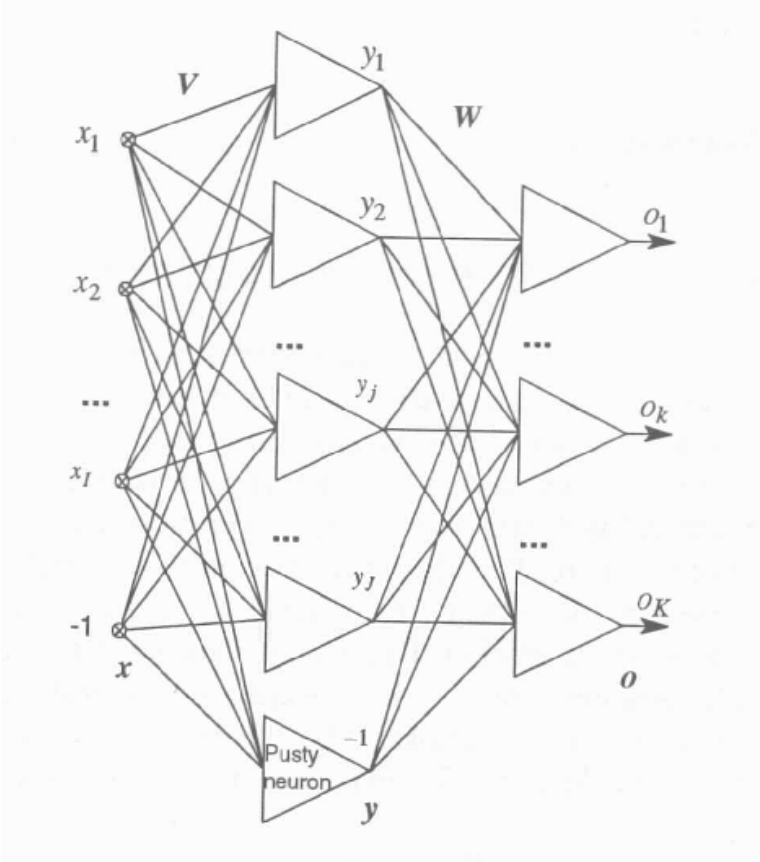
Takie określenie błędu nakłada ograniczenia na postać funkcji aktywacji, która musi być **ciągła** i **różniczkowalna**.

Reguła DELTA mówi, że aktualizacja macierzy wag V i W następuje wg zależności:

$$\begin{cases} \Delta \mathbf{V}^{n+1} = -\eta \cdot \nabla E(\mathbf{V}^n) \\ \Delta \mathbf{W}^{n+1} = -\eta \cdot \nabla E(\mathbf{W}^n) \end{cases}$$

gdzie $\boldsymbol{\eta}$ jest współczynnikiem szybkości treningu.

Reguła delta – cd.



W wyniku podstawienia wcześniejszych wyników otrzymujemy:

$$\begin{cases} \mathbf{V}^{n+1} = \mathbf{V}^{n} + \eta\,\delta_{y}\,\mathbf{x}^{T} \\ \mathbf{W}^{n+1} = \mathbf{W}^{n} + \eta\,\delta_{o}\,\mathbf{y}^{T} \end{cases}$$

Wagi w kolejnym kroku nauki są równe wagom w poprzednim kroku powiększonym proporcjonalnie do iloczynów wektorów wejściowych warstw i wektorów błędu działania sieci.

Minimalizacja funkcji błędu jest oparta o metody gradientowe i
nie gwarantuje zbieżności nauki.

Zbieżność procesu nauki

- Możliwość zatrzymania w minimum lokalnym funkcji błędu
- Możliwość zatrzymania nauki w płaskim obszarze funkcji błędu przy zbyt małej wartości współczynnika nauki
- Aby poprawić właściwości nauki stosuje się dodatkowy składnik **MOMENTU**, uzależniający przyrost wag od przyrostu wag w poprzednim kroku nauki
- Po uwzględnieniu składnika momentu, wzory wartości macierzy wag w kroku *n*+1 wyglądają następująco:

$$\begin{cases} \mathbf{V}^{n+1} = \mathbf{V}^{n} + \eta\,\delta_{y}\,x^{T} + \alpha\cdot\Delta\mathbf{V}^{n} \\ \mathbf{W}^{n+1} = \mathbf{W}^{n} + \eta\,\delta_{o}\,y^{T} + \alpha\cdot\Delta\mathbf{W}^{n} \end{cases}$$

składnik momentu

Dodatkowe właściwości momentu

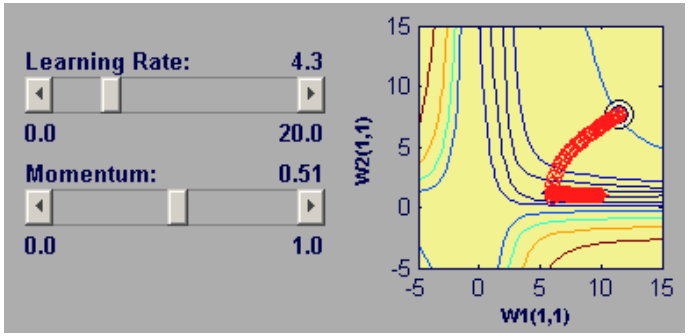
$$\begin{cases} \mathbf{V}^{n+1} = \mathbf{V}^{n} + \eta\,\delta_{y}\,x^{T} + \alpha \cdot \Delta\mathbf{V}^{n} \\ \mathbf{W}^{n+1} = \mathbf{W}^{n} + \eta\,\delta_{o}\,y^{T} + \alpha \cdot \Delta\mathbf{W}^{n} \end{cases}$$

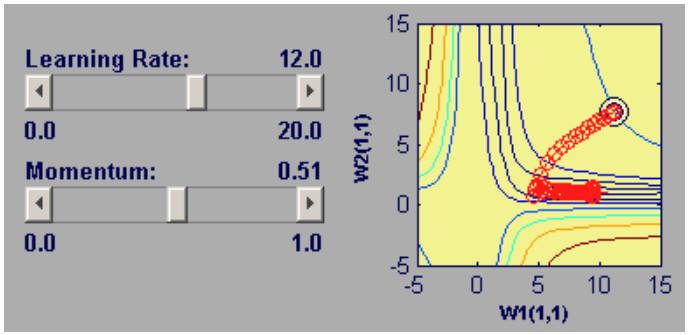
składnik momentu

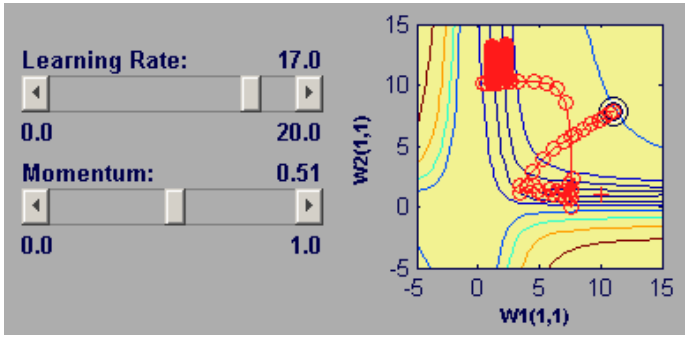
- Moment wprowadza do algorytmu element bezwładności, który zmniejsza chwilowe i gwałtowne zmiany kierunku wskazywanego przez gradient funkcji błędy
- Uczenie nie wchodzi w płytkie minima lokalne
- Zdolność do znacznego przyspieszania nauki dla płaskich obszarów funkcji błędu

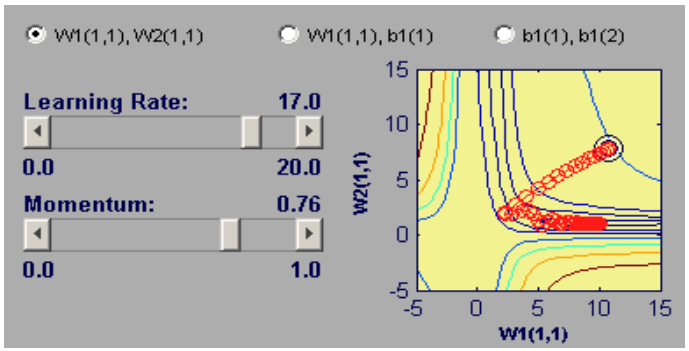
Właściwy dobór współczynników nauki

- Właściwy dobór współczynników nauki umożliwia wyjście z minimów lokalnych funkcji błędu i szybkie osiągnięcie wartości bliskich jej minimum globalnego.

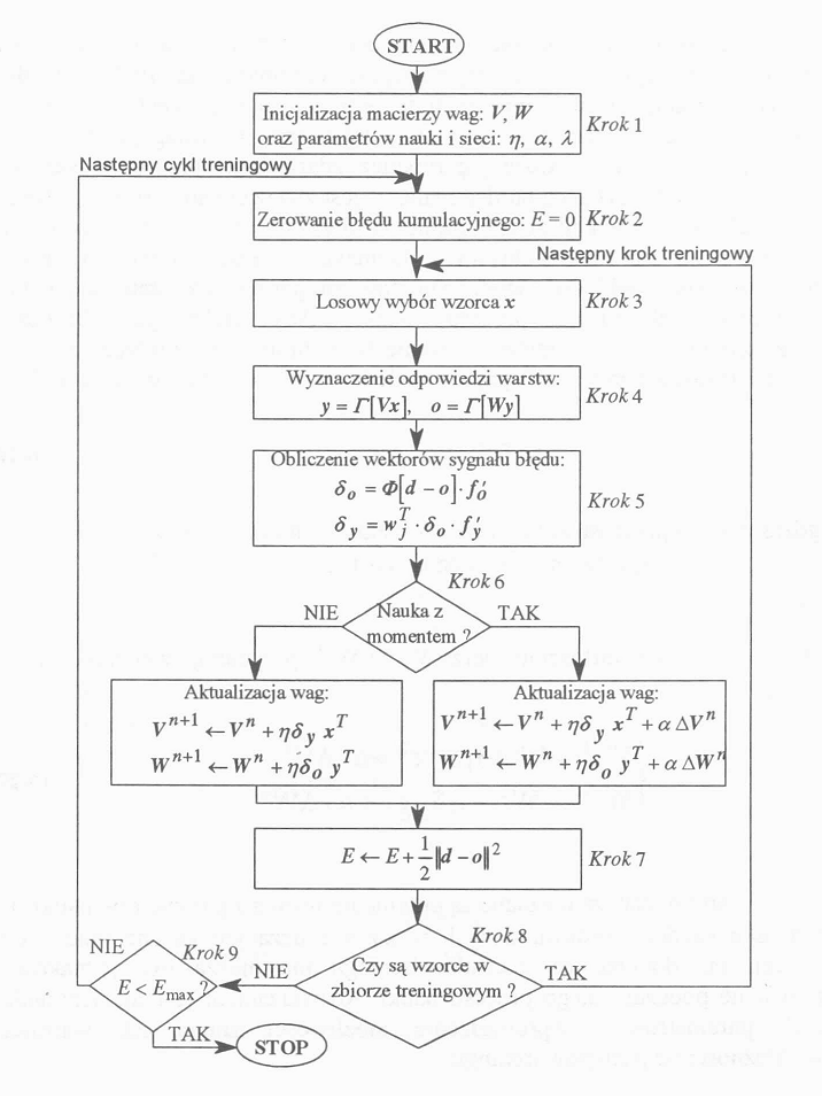








Algorytm wstecznej propagacji błędu



KROK 1:

- Inicjalizacja macierzy wag V i W małymi losowyim wartościami z zakresu (-1,1)
- Ustawianie parametrów nauki sieci:
  - funkcji aktywacji neuronów (razem z **λ**)
  - parametrów nauki - ***η*** *i* ***α***

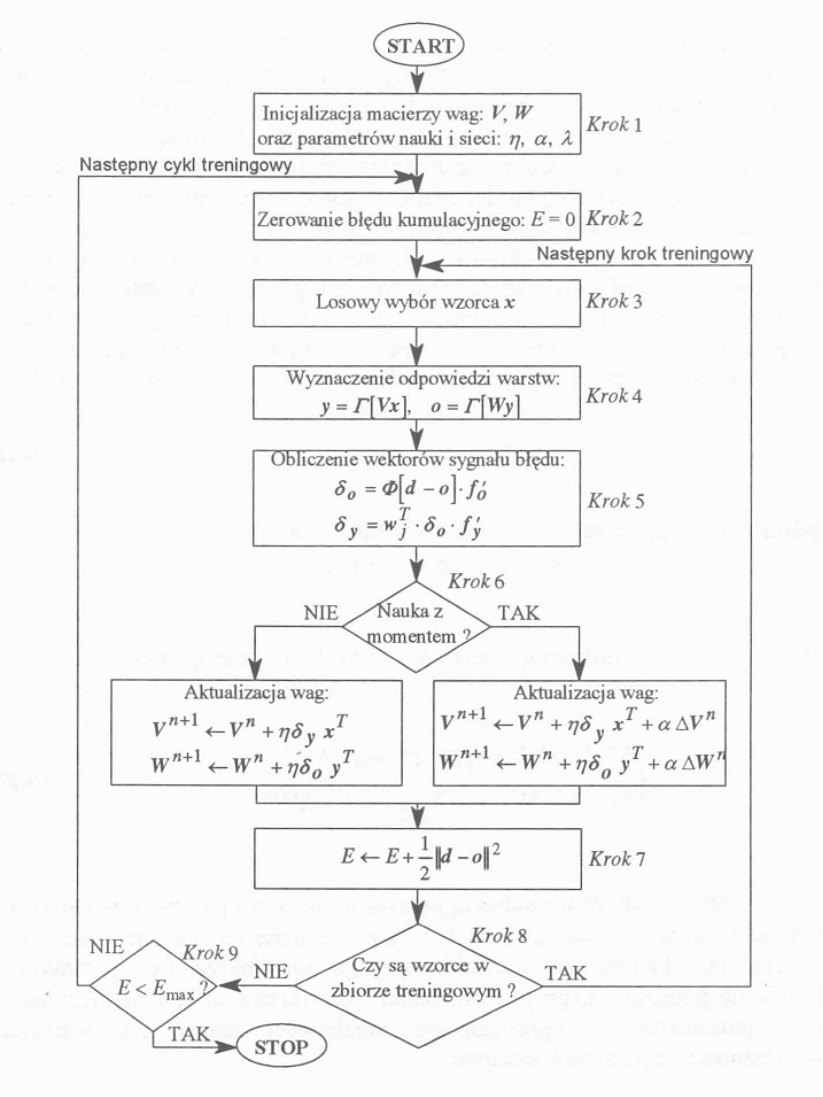
KROK 2:

- Ustawienie wartości błędu skumulowanego na ***E***=0 dla każdego nowego cyklu treningowego

KROK 3:

- Wybór dowolnego wektora ze zbioru uczącego, najlepiej wybór losowy. Ustawienie oczekiwanej odpowiedzi sieci ***d***.

Algorytm wstecznej propagacji błędu



KROK 4:

- Wyznaczanie odpowiedzi warstw sieci **y**, **o** (przetwarzanie wektora wejściowego przez sieć)

KROK 5:

- Obliczanie sygnałów błędów dla kolejnych warstw

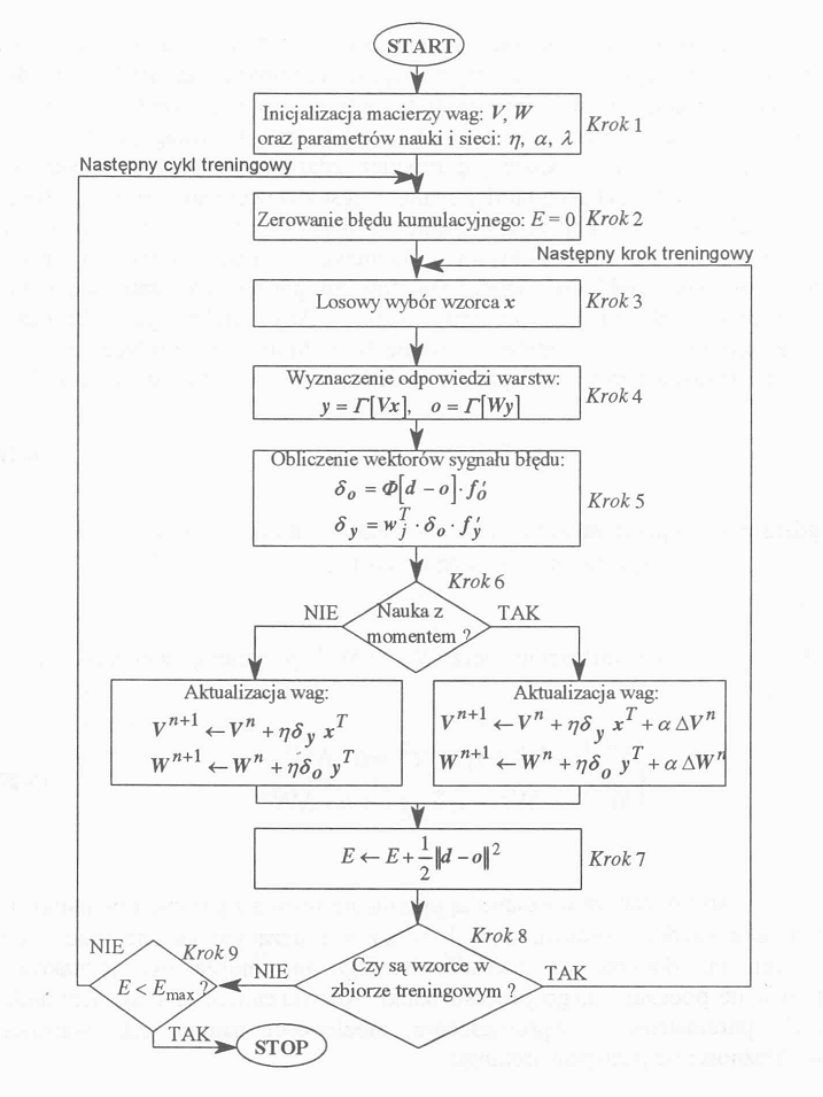
KROK 6:

- Obliczanie nowych wartości wag, z uwzględnieniem momentu lub bez (w zależności od wyboru)

KROK 7:

- Obliczana jest wartość funkcji błędu dla wektora x. Wartość ta dodawana jest do wartości błędu skumulowanego ***E***.

Algorytm wstecznej propagacji błędu



KROK 8:

- Jeśli wektor x nie jest ostatnim wykorzystywanym wzorcem, to sterowanie wraca do kroku 3. Jeśli wektor x jest ostatnim wektorem uczącym to sterowanie przechodzi kroku 9.

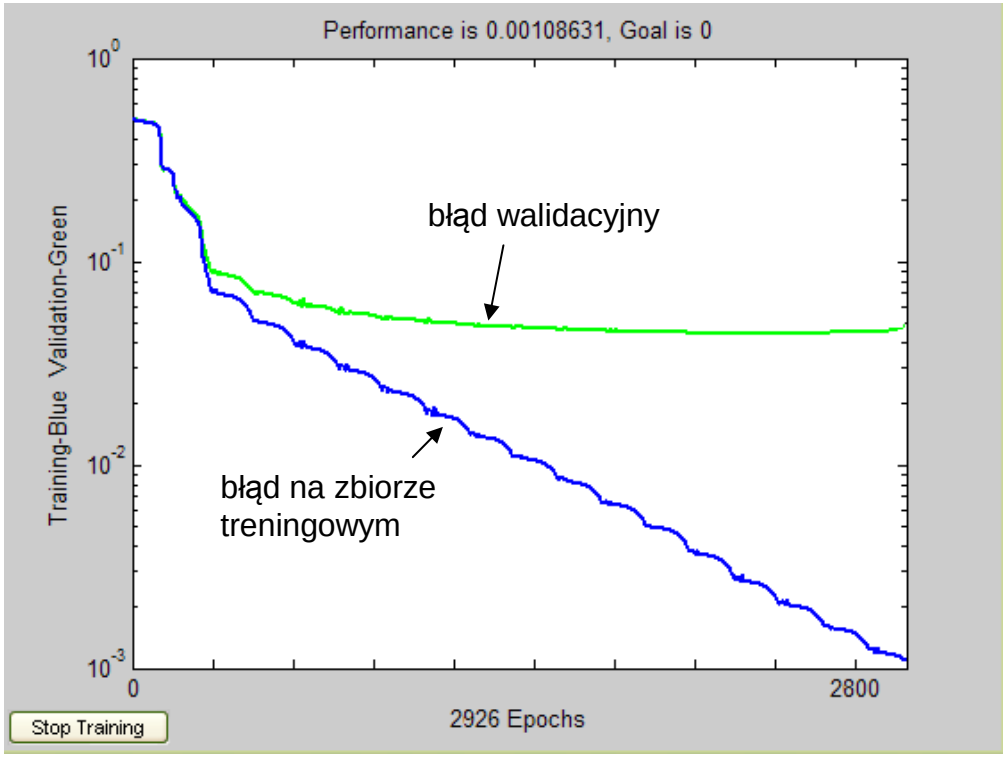
KROK 9:

- Sprawdzany jest warunek czy wartość błędu skumulowanego jest większa od zadanej progowej wartości E_{MAX}. Jeśli tak to trening się zatrzymuje.
  Jeśli warunek ten nie jest spełniony to następuje kolejny cykl treningowy i powrót do kroku 2.

Właściwości generalizacyjne

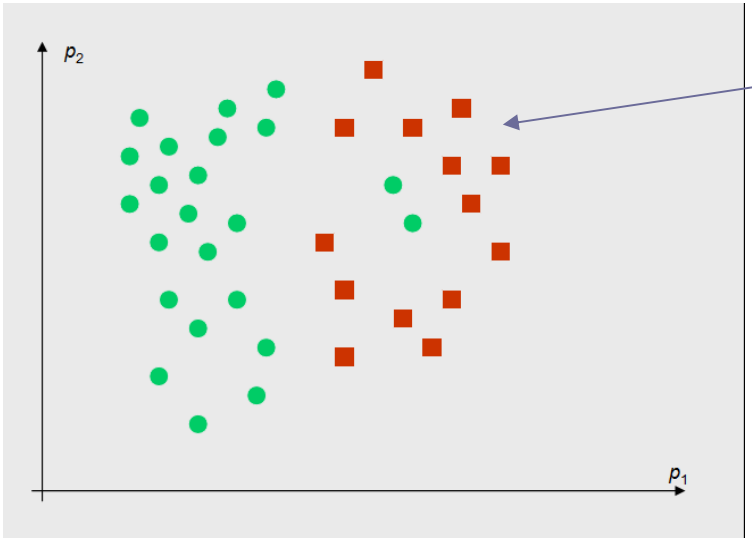
- Sieć można przeuczyć, gdy algorytm powtarza się w zbyt wielu krokach
- Sieć przeuczona posiada bardzo dobre właściwości reagowania na obiekty ze zbioru treningowego ale nie umie sobie dobrze radzić z przykładami spoza zbioru uczącego
- Metoda przeciwdziałania – sprawdzanie działania sieci dla danych walidacyjnych

Właściwości generalizacyjne



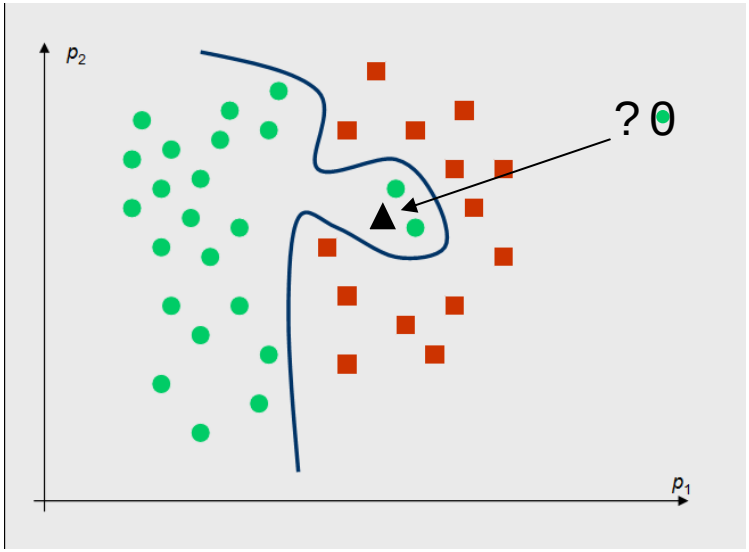
- Kontrola błędu walidacyjnego w każdym kroku nauki uniemożliwia przetrenowanie sieci – utrzymuje zdolności generalizacyjne
- Gdy błąd walidacyjny rośnie przez pewna założoną liczbę cyklów (Epochs) nauka sieci powinna być przerywana

Właściwości generalizacyjne

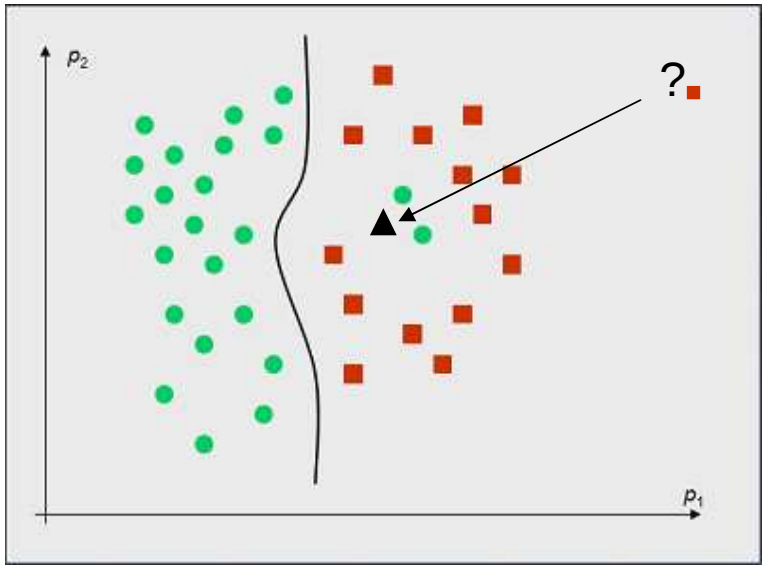


dane treningowe należące
do dwóch kategorii

zagadnienie klasyfikacyjne



sieć przetrenowana



sieć z zachowanymi właściwościami general.