

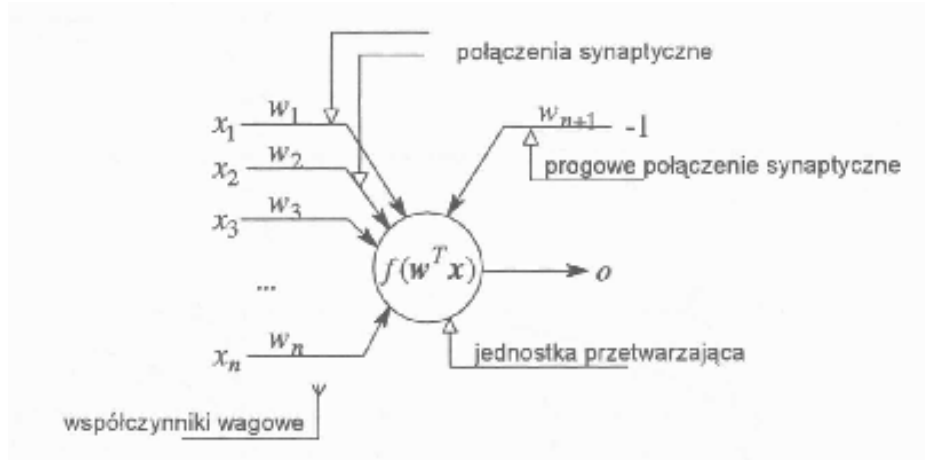
# **Inteligentne systemy decyzyjne: Uczenie maszynowe – sztuczne sieci neuronowe**

## **wykład 2.**

Trening jednokierunkowych sieci  
neuronowych

dr inż. Paweł Żwan  
Katedra Systemów Multimedialnych  
Politechnika Gdańska

# Model matematyczny sztucznego neuronu



$$\text{net} = \mathbf{w}^T \mathbf{x}$$

Wartość wyjściowa neuronu  $\mathbf{o}$  jest określana w oparciu o wzór:  $\mathbf{o} = \mathbf{f}(\text{net})$ ,

gdzie:  $\mathbf{w}$  – wektor wag połączeń wejściowych

$\mathbf{x}$  – wektor wartości sygnałów wejściowych

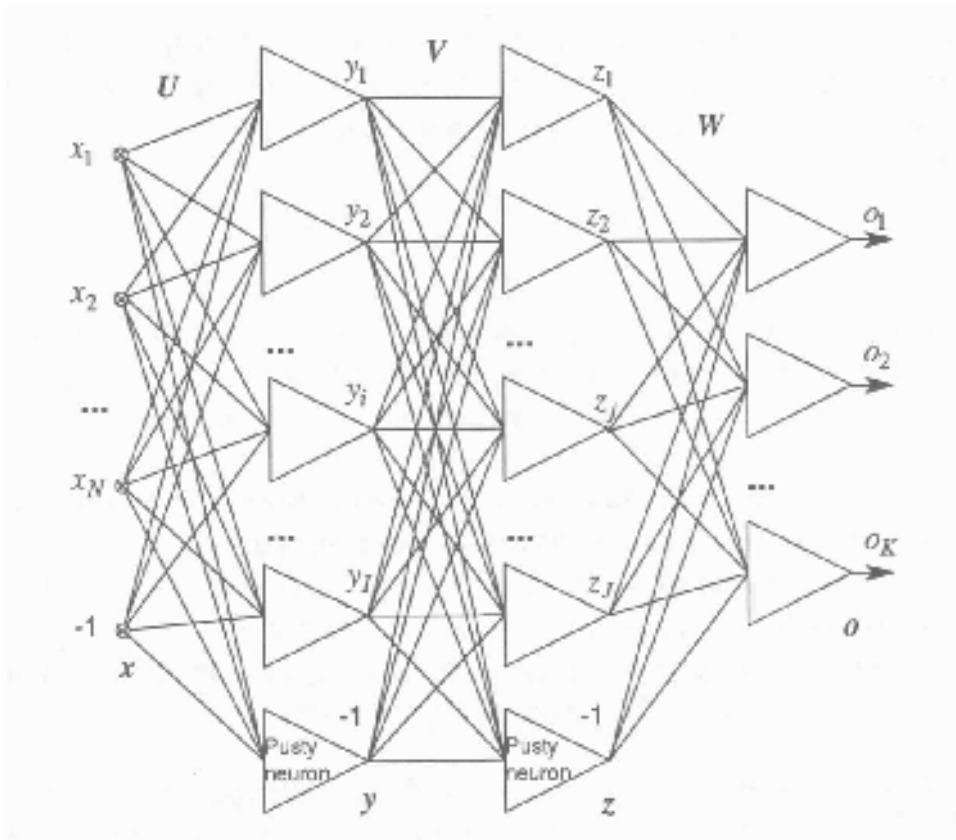
$\mathbf{f}$  – funkcja aktywacji

Dodatkową wagą jest waga progowa, dlatego wektory  $\mathbf{w}$  i  $\mathbf{x}$  określone są jako:

$$\mathbf{w} = [w_1, w_2, \dots, w_n, w_{n+1}], \quad \mathbf{x} = [x_1, x_2, \dots, x_n, -1]$$

# Topologia sieci jednokierunkowej

## - sieci jednokierunkowe



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

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

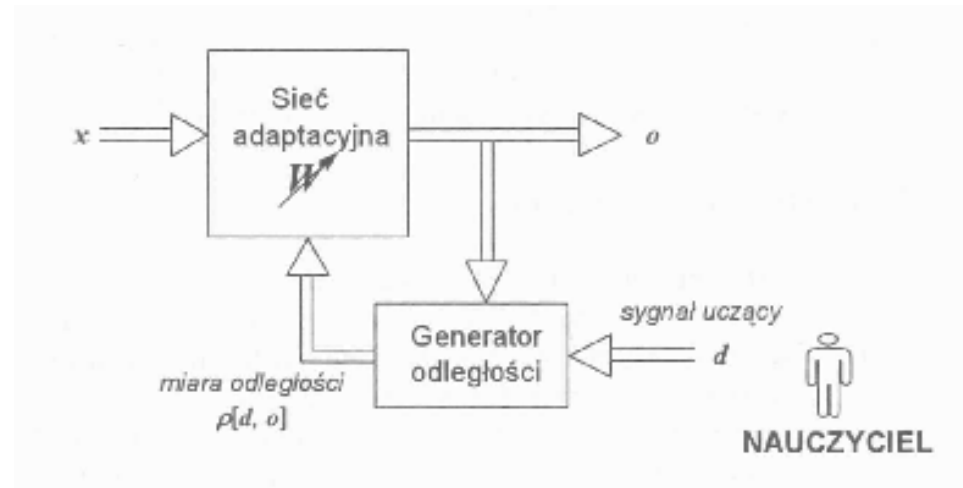
$\mathbf{z}=[z_1, z_2, \dots, z_J, -1]$  – wektor wyjściowy drugiej warstwy ukrytej

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

Macierze  $\mathbf{U}, \mathbf{V}, \mathbf{W}$  zawierają  
Współczynniki wagowe dla wszystkich  
Połączeń synaptycznych

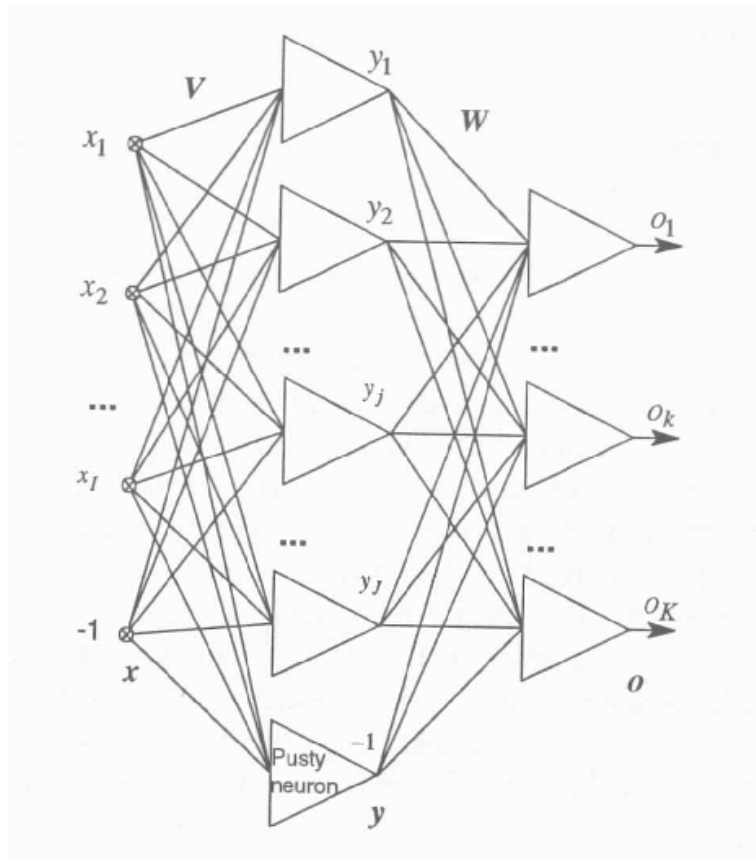
# Metoda treningu sieci jednokierunkowej

## trening z nadzorem



# Reguła delta

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



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

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

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

macierze wag warstw:

$$\mathbf{V} = \begin{bmatrix} v_{11} & v_{12} & \dots & v_{1I} \\ v_{21} & v_{22} & \dots & v_{2I} \\ \dots & \dots & \dots & \dots \\ v_{J1} & v_{J2} & \dots & v_{JI} \end{bmatrix},$$

$$\mathbf{W} = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1J} \\ w_{21} & w_{22} & \dots & w_{2J} \\ \dots & \dots & \dots & \dots \\ w_{K1} & w_{K2} & \dots & w_{KJ} \end{bmatrix}$$

wektory pochodnych funkcji aktywacji:

$$\mathbf{f}'_y = [f'_1(\text{net}_1), f'_2(\text{net}_2), \dots, f'_j(\text{net}_j)]^T$$

$$\mathbf{f}'_o = [f'_1(\text{net}_1), f'_2(\text{net}_2), \dots, f'_K(\text{net}_j)]^T$$

# Reguła delta – cd.

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

Najczęściej stosuje się odł. średniokwadratową:

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

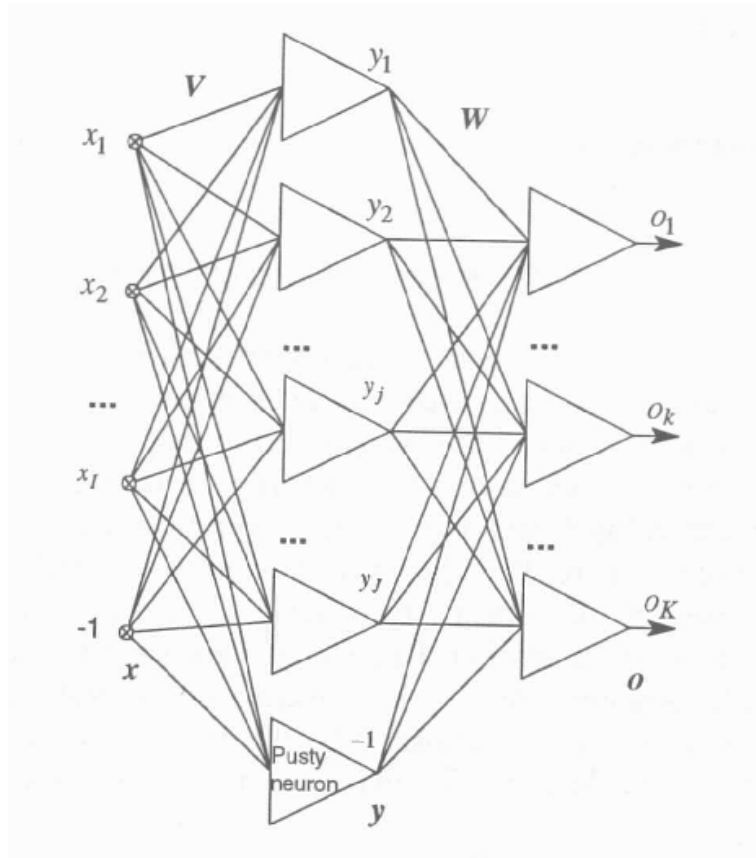
$$E^n = \frac{1}{2} \cdot \sum_{p=1}^P \|\mathbf{d}^{(p)} - \mathbf{o}^{(p)}\|^2$$

gdzie  $\mathbf{p}$  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  $\mathbf{p}$  – tego wektora przyjmuje postać:

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



# 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:

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

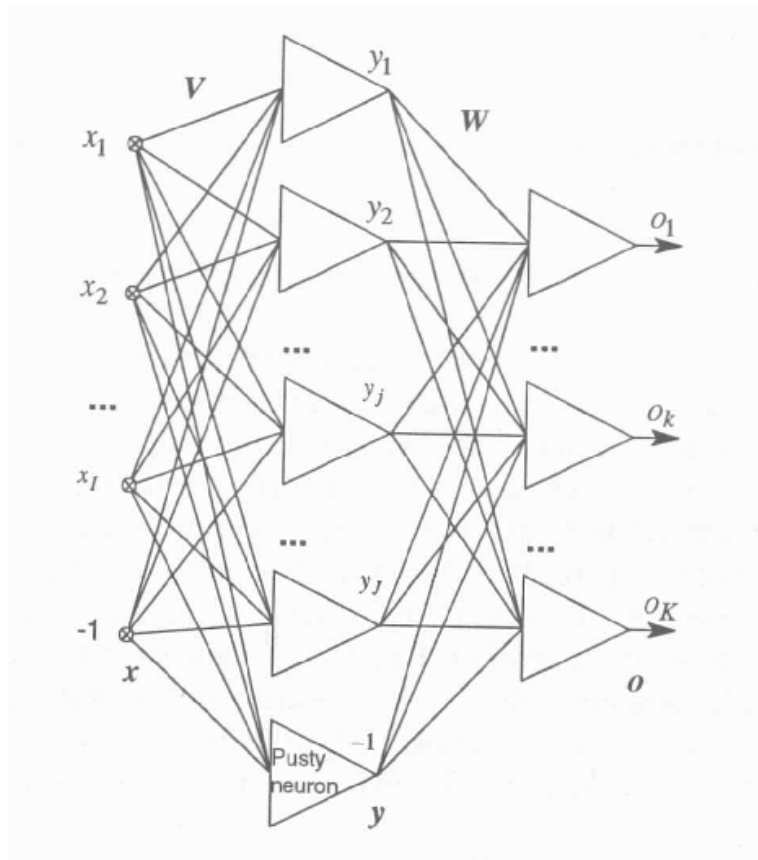
Funkcja błędu:

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

przyjmuje postać:

$$\delta_{\mathbf{o}} = -\nabla E(\mathbf{o}) \text{ lub } \delta_{o_k} = -\frac{\partial E}{\partial \text{net}_k} \text{ - warstwa wyjściowa}$$

$$\delta_{\mathbf{y}} = -\nabla E(\mathbf{y}) \text{ lub } \delta_{y_j} = -\frac{\partial E}{\partial \text{net}_j} \text{ - warstwa ukryta}$$



# Reguła delta – cd.

wtedy:

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

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

gdzie wprowadzono pomocniczy operator:

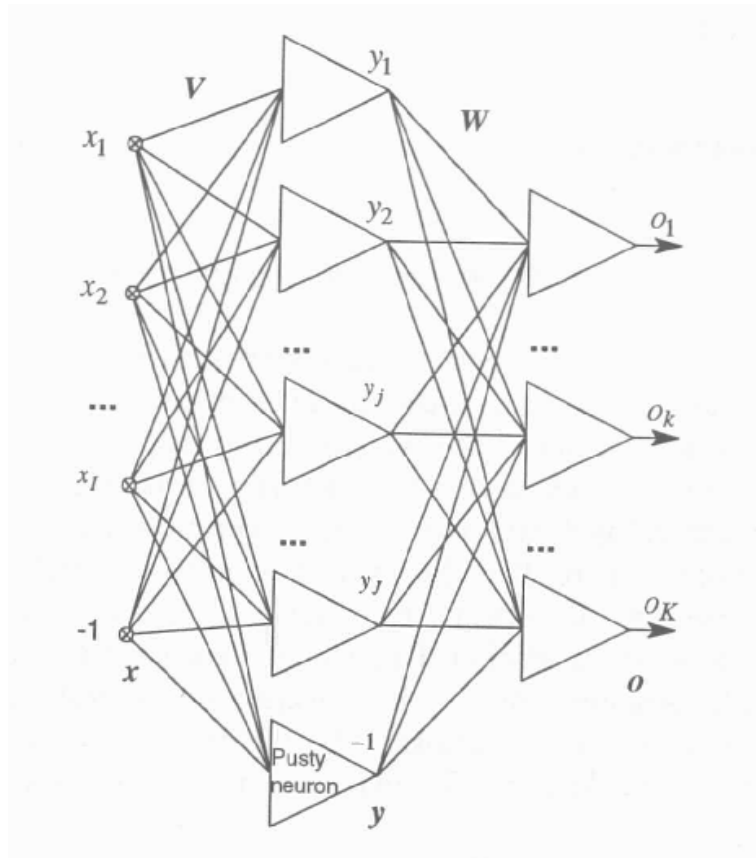
$$\Phi[\mathbf{q}] = \begin{bmatrix} q_1 & 0 & \dots & 0 \\ 0 & q_2 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & q_Q \end{bmatrix},$$

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  $\eta$  jest współczynnikiem szybkości treningu.

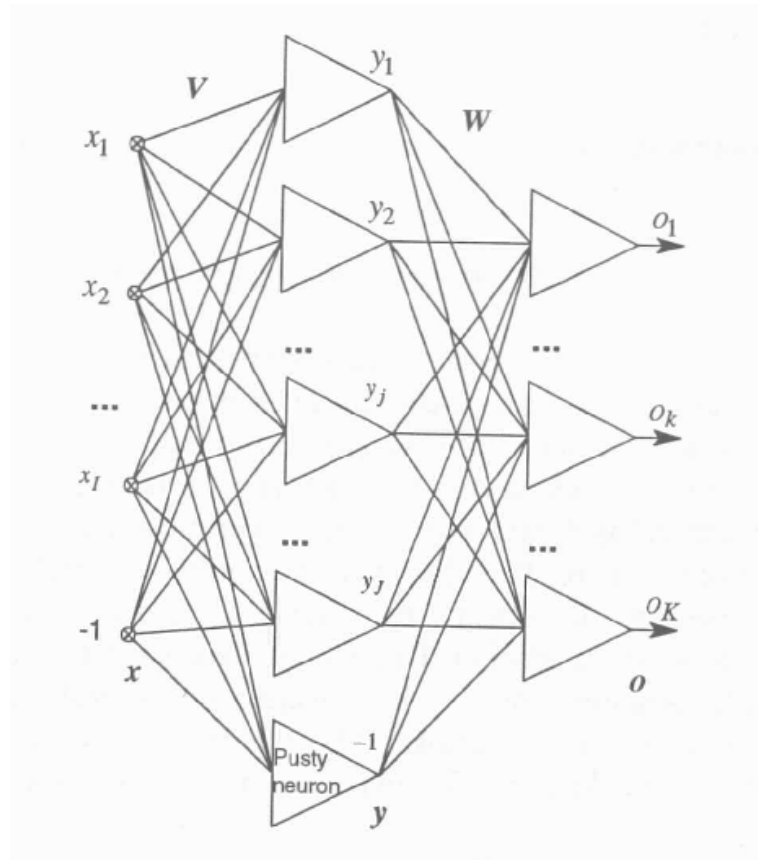




# Reguła delta – cd.

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:

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

składnik momentu

# Dodatkowe właściwości momentu

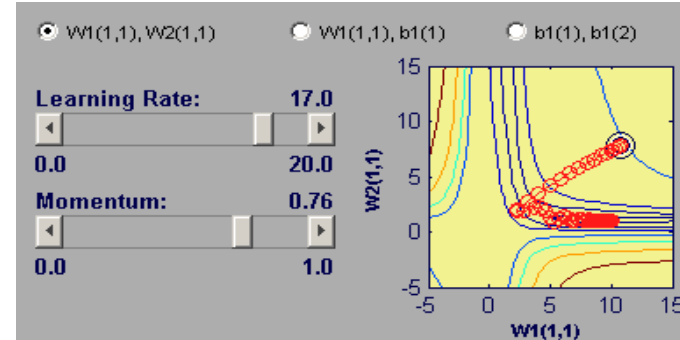
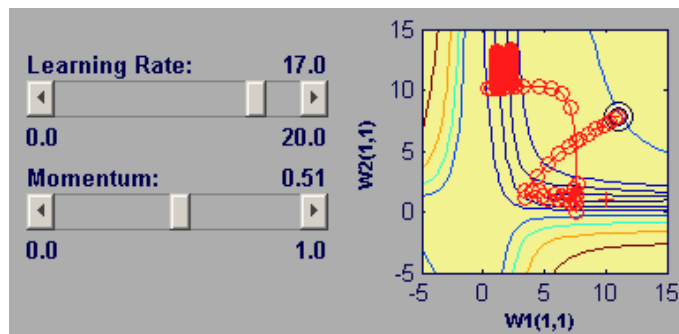
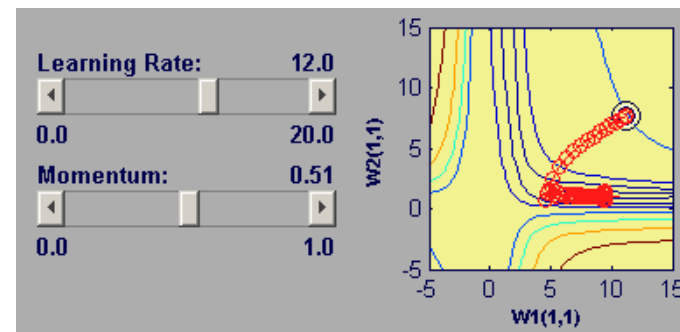
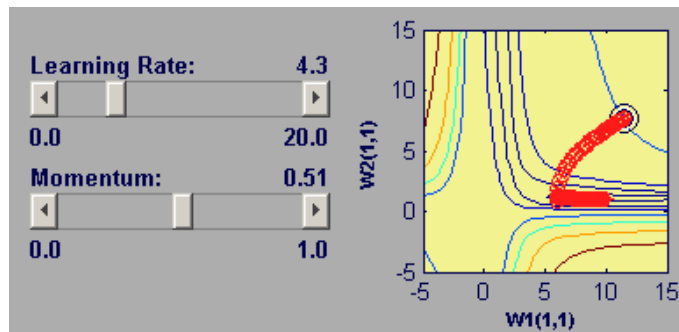
$$\begin{cases} \mathbf{V}^{n+1} = \mathbf{V}^n + \eta \delta_y \mathbf{x}^T + \alpha \cdot \Delta \mathbf{V}^n \\ \mathbf{W}^{n+1} = \mathbf{W}^n + \eta \delta_o \mathbf{y}^T + \alpha \cdot \Delta \mathbf{W}^n \end{cases}$$

składnik momentu

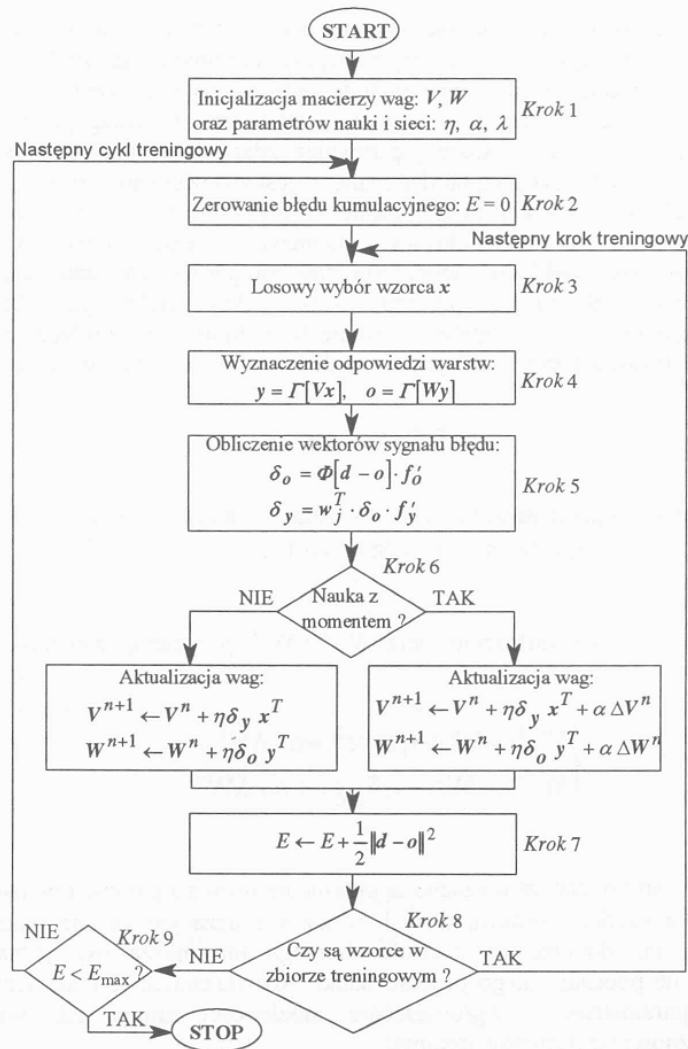
- Moment wprowadza do algorytmu element bezwładności, który zmniejsza chwilowe i gwałtowne zmiany kierunku wskazywanego przez gradient funkcji błędy
- Uczenie nie wchodzi w płytkie minima lokalne
- Zdolność do znacznego przyspieszania nauki dla płaskich obszarów funkcji błędu

# 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 losowymi wartościami z zakresu  $(-1,1)$
- Ustawianie parametrów nauki sieci:
  - funkcji aktywacji neuronów (razem z  $\lambda$ )
  - parametrów nauki -  $\eta$  i  $\alpha$

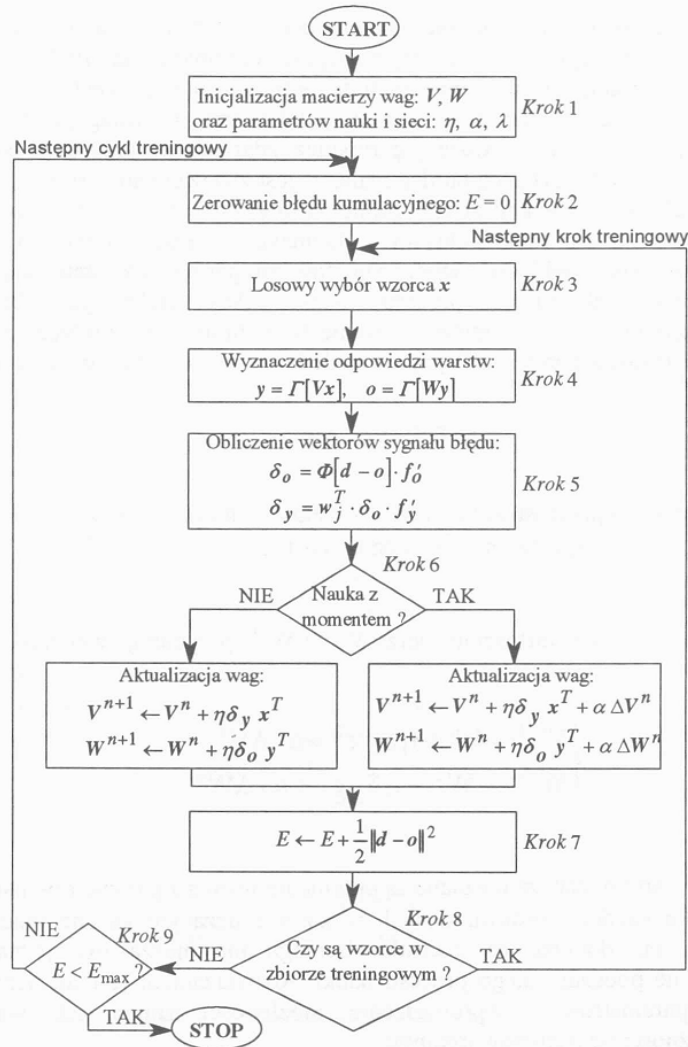
## KROK 2:

- 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

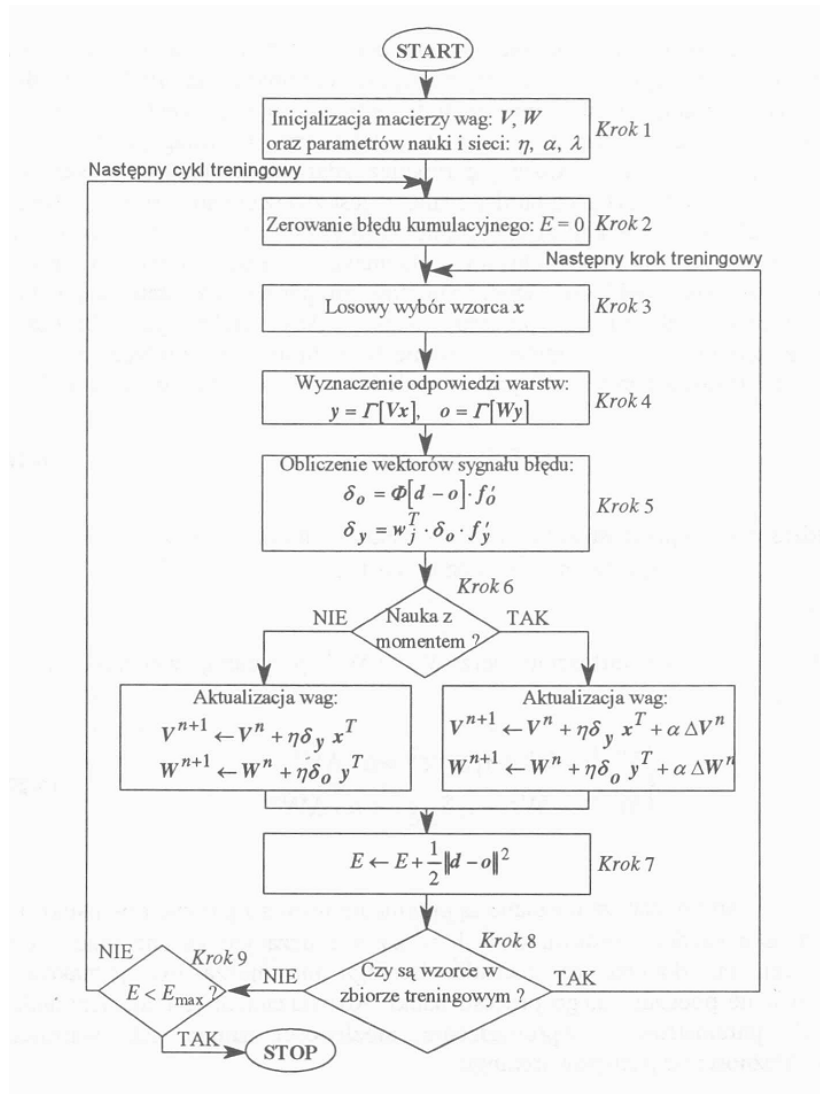
## KROK 6:

- 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:

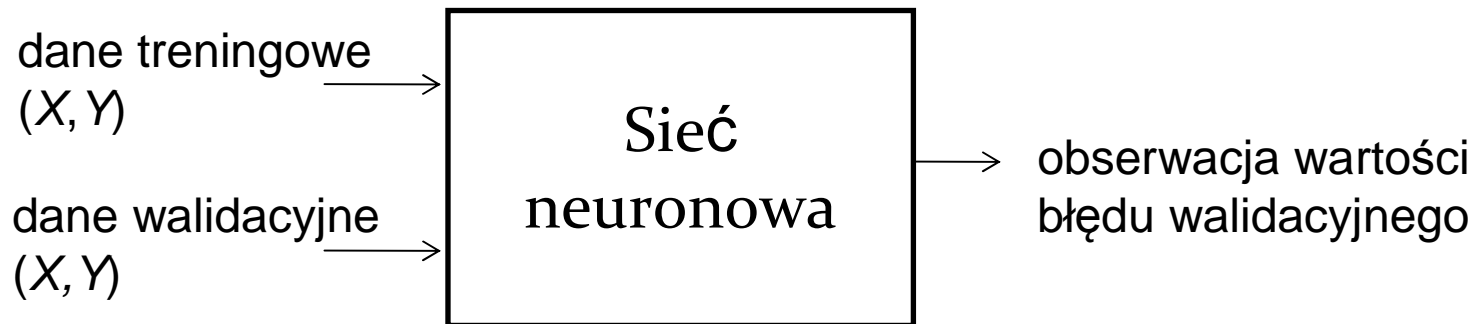
- Jeśli wektor  $x$  nie jest ostatnim wykorzystywanym wzorcem, to sterowanie wraca do kroku 3. Jeśli wektor  $x$  jest ostatnim wektorem uczącym to sterowanie przechodzi kroku 9.

## 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.

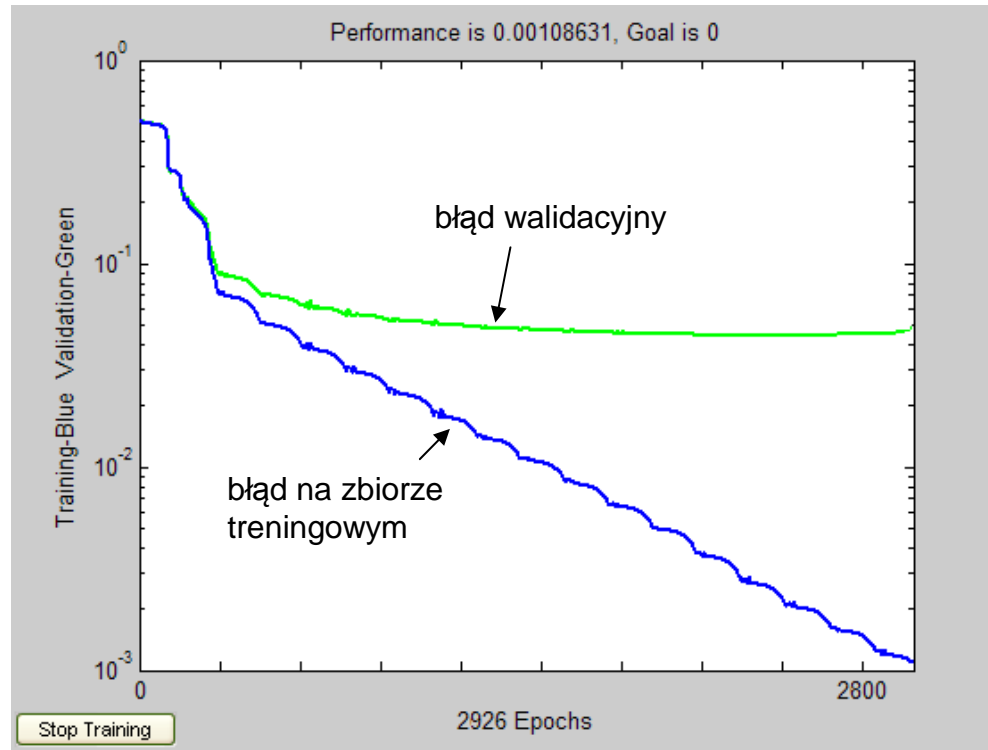
# Właściwości generalizacyjne

- 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



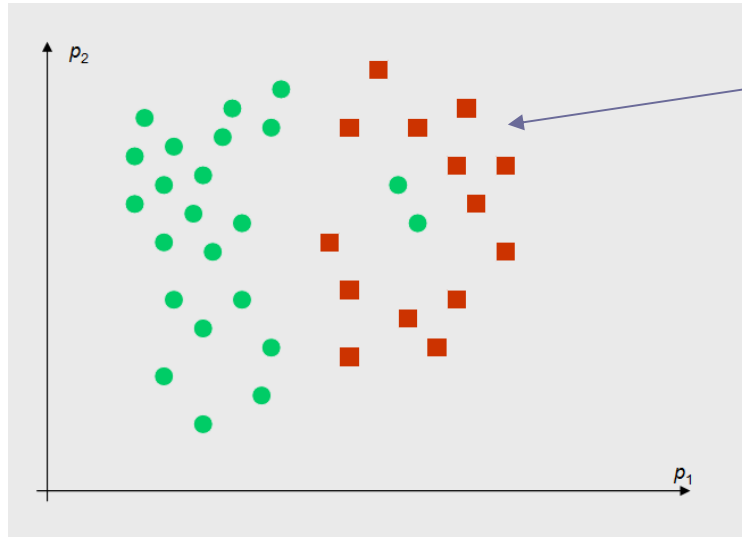


# Właściwości generalizacyjne



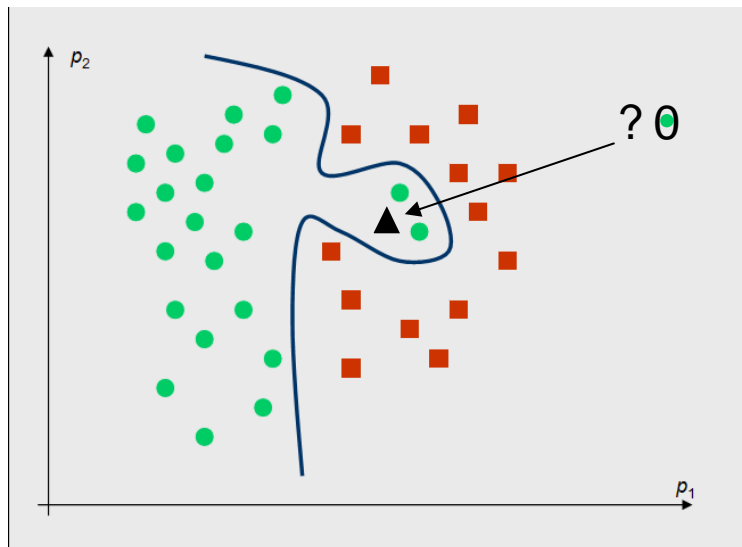
- Kontrola błędu walidacyjnego w każdym kroku nauki uniemożliwia przetrenowanie sieci – utrzymuje zdolności generalizacyjne
- Gdy błąd walidacyjny rośnie przez pewną założoną liczbę cykli (Epochs) nauka sieci powinna być przerywana

# Właściwości generalizacyjne

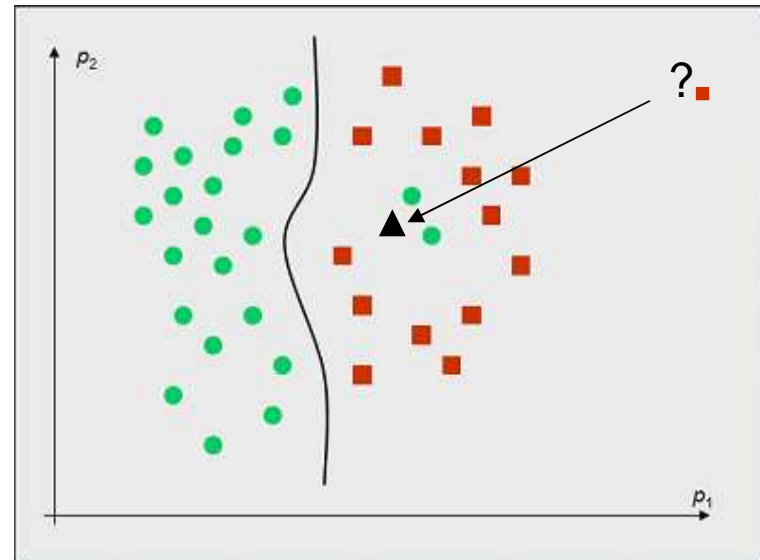


dane treningowe należące do dwóch kategorii

zagadnienie klasyfikacyjne



sieć przetrenowana



sieć z zachowanymi właściwościami general.