

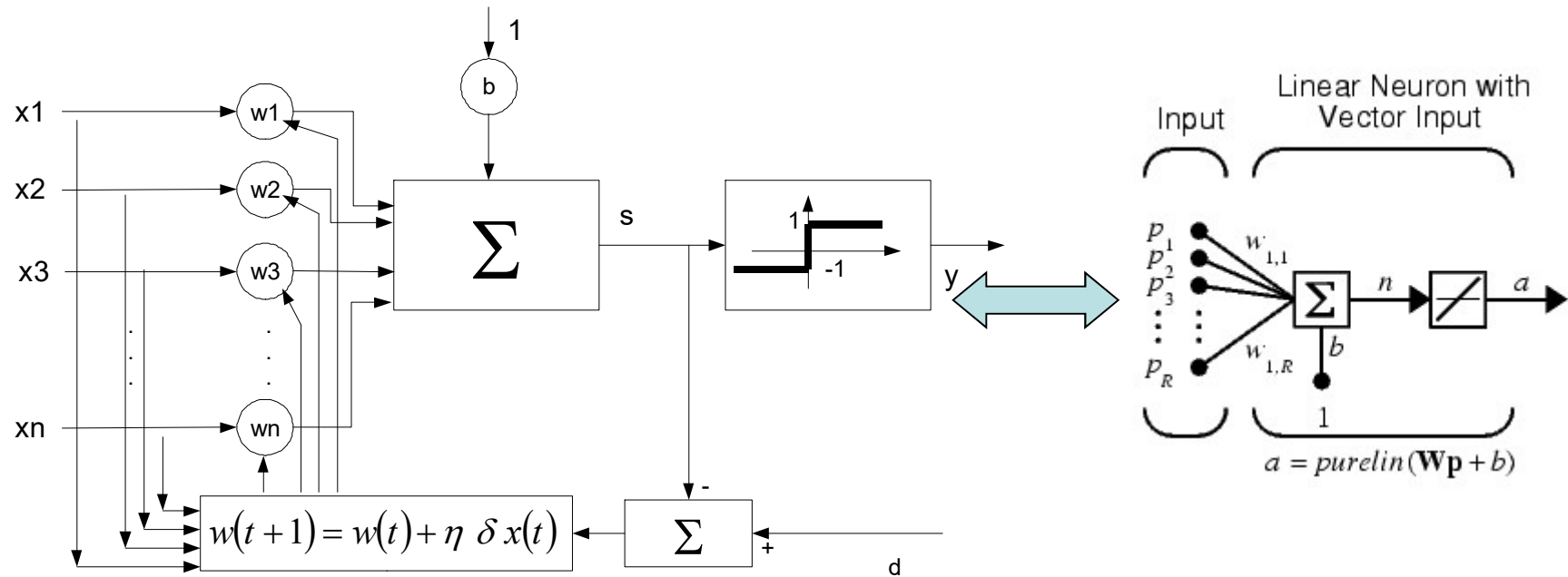
# SZTUCZNE SIECI NEURONOWE

Ang. Artificial neural network

Wykład 2

Dr inż. Piotr Urbanek

## Neuron Adaline (Adaptive Linear Neuron)



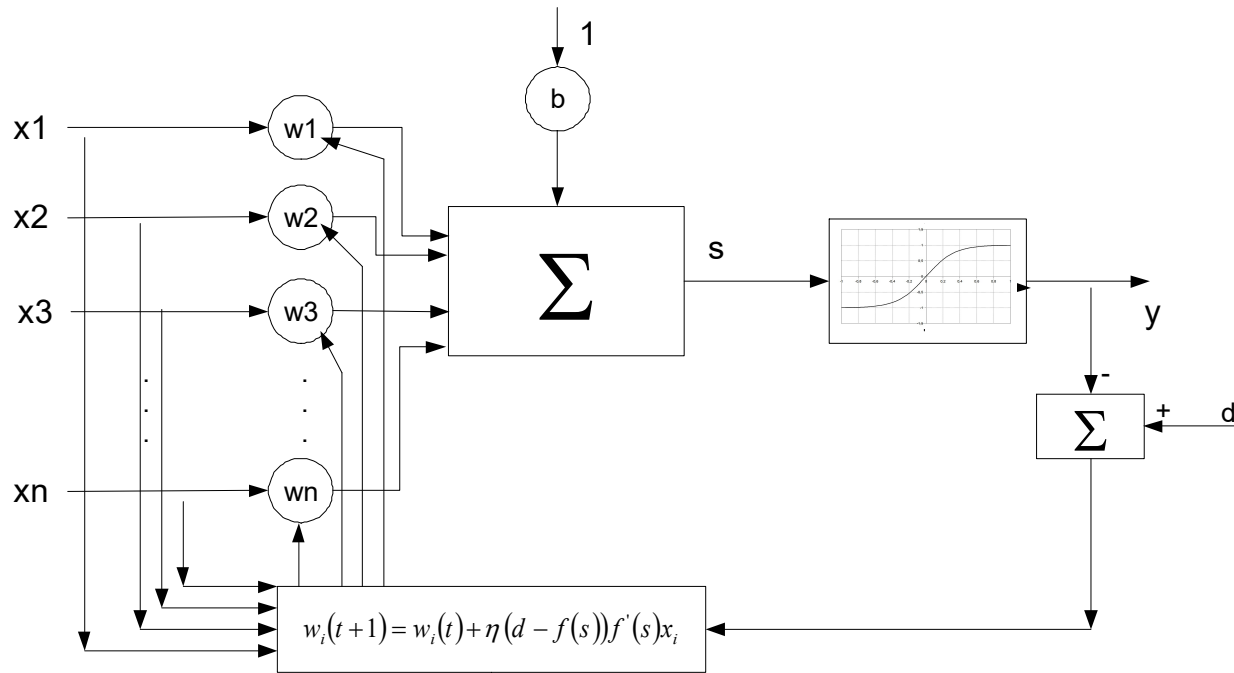
$$\varepsilon = d - s$$

Minimalizacja funkcji: 
$$Q(w) = \frac{1}{2} \varepsilon^2 = \frac{1}{2} \left[ d - \left( \sum_{i=0}^n w_i x_i \right) \right]^2$$

$$w_i(t+1) = w_i(t) + \eta \delta x_i$$

gdzie:  $\delta = d - s$

## Neuron sigmoidalny.



Funkcja aktywacji:

$$f(s) = \frac{1}{1 + e^{-\beta s}}$$

$$f(s) = \tanh(\beta s) = \frac{1 - e^{-\beta s}}{1 + e^{-\beta s}}$$

$$f'(s) = \beta f(s)(1 - f(s))$$

$$f'(s) = \beta (1 - f^2(s))$$

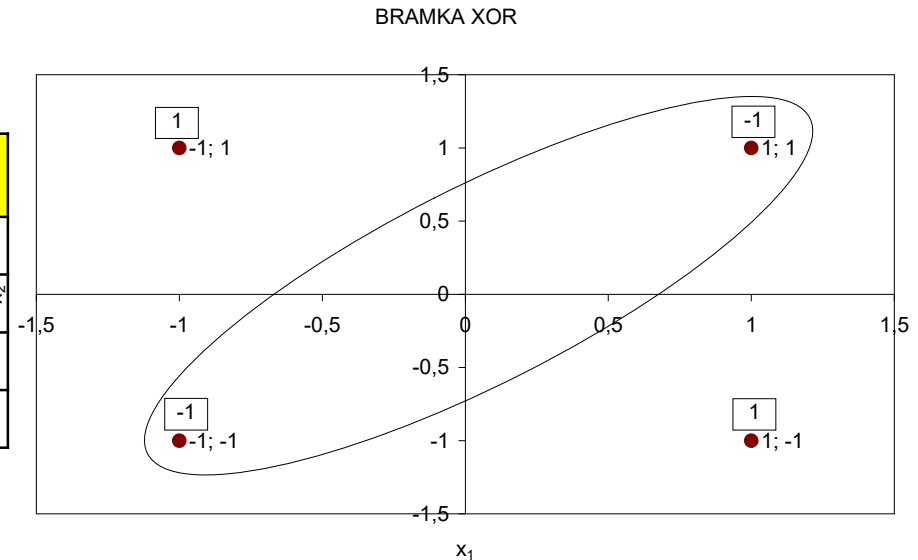
$$w_i(t+1) = w_i(t) + \eta(d - f(s))f'(s)x_i$$

gdzie:

# Sieci jednokierunkowe wielowarstwowe

## Bramka XOR

$x_1$	$x_2$	XOR ( $x_1, x_2$ )
1	1	-1
1	-1	1
-1	1	1
-1	-1	-1

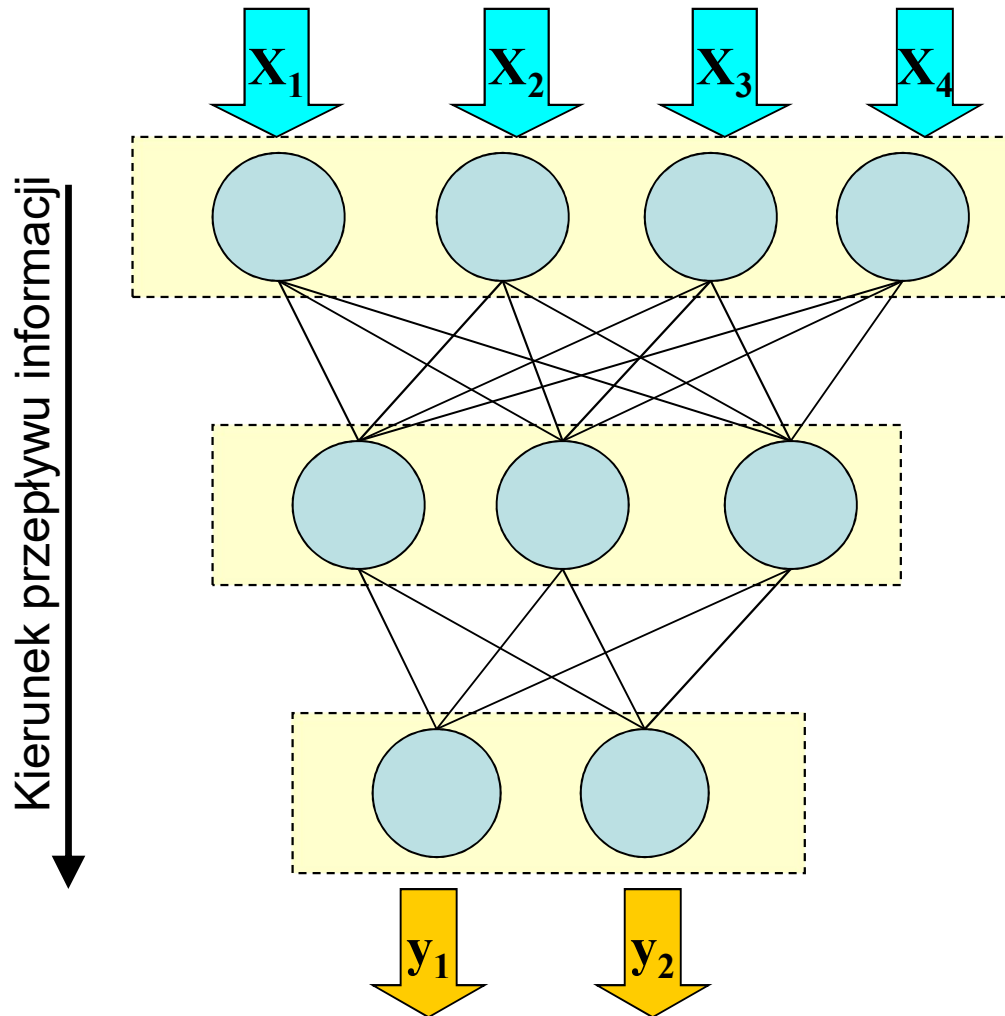


Sieci wielowarstwowe to odpowiednio ze sobą połączone neurony rozmieszczone w dwóch, lub więcej warstwach.

Z reguły są to neurony sigmoidalne, ale używa się również neuronów o liniowych funkcjach aktywacji (są one najczęściej umieszczane w ostatniej, wyjściowej warstwie).

Sieci wielowarstwowe muszą mieć co najmniej dwie warstwy: wejściową i wyjściową.

# Wielowarstwowe Sztuczne Sieci Neuronowe



## Warstwa wejściowa

- Każdy neuron posiada tylko jedno wejście zewnętrzne.

## Warstwa ukryta

- Zawiera połączenia warstw

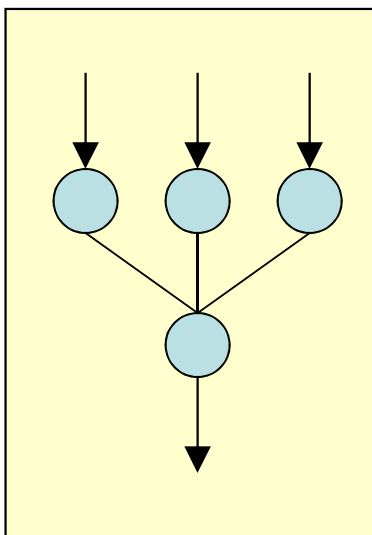
## Warstwa wyjściowa

- Każdy neuron posiada jedno wyjście zewnętrzne.

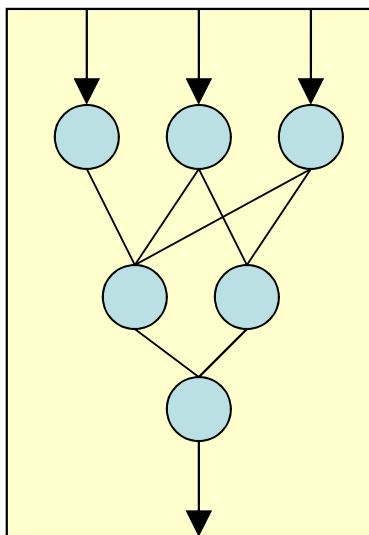
# Wielowarstwowe Sztuczne Sieci Neuronowe

Liczba ukrytych warstw może wynosić:

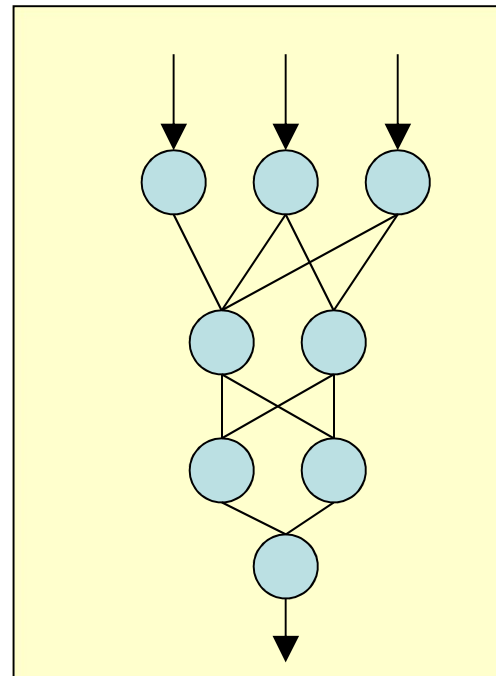
**0**



**1**

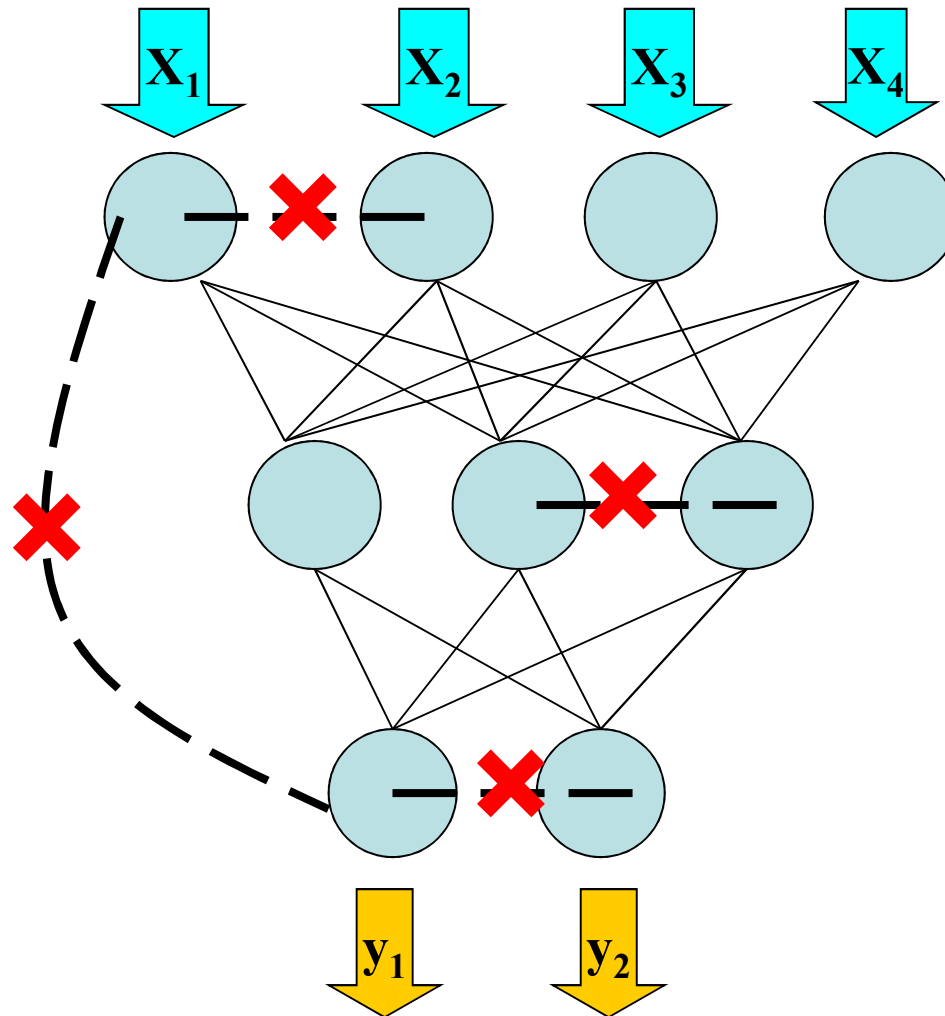


**Więcej niż 1**



## Wielowarstwowe Sztuczne Sieci Neuronowe

Należy zauważyć, że:



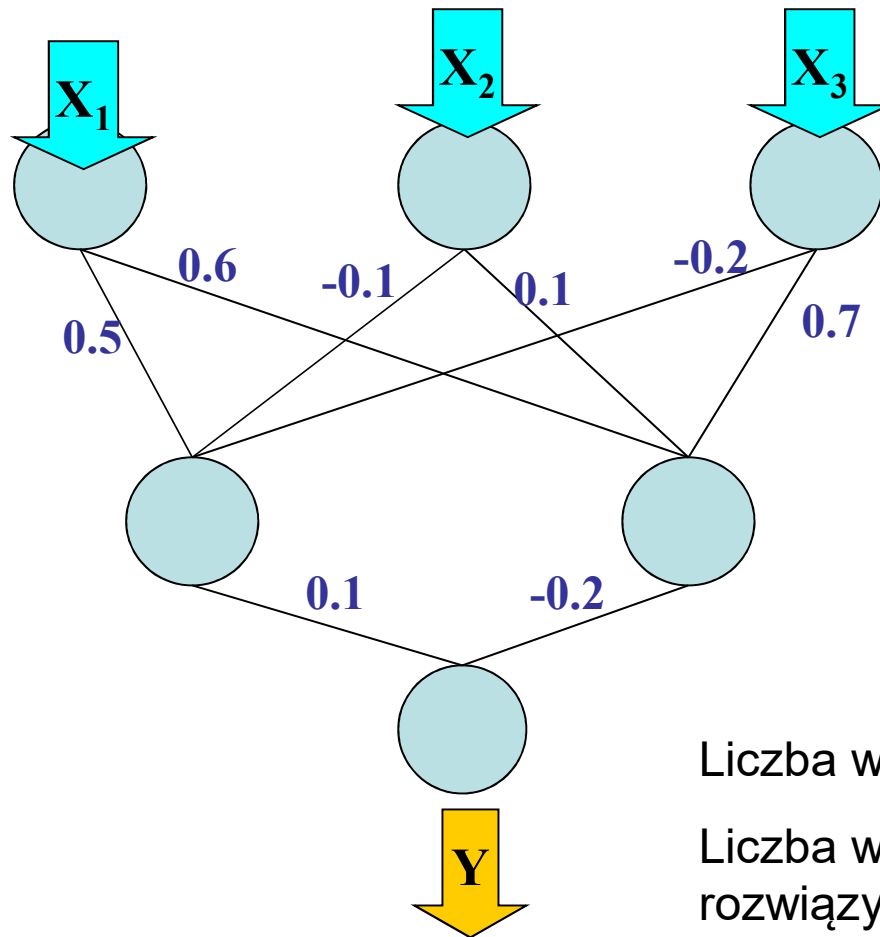
- Nie istnieją połączenia pomiędzy neuronami w danej warstwie.
- Neuron w jednej warstwie jest połączony z innymi neuronami w innych warstwach.
- „Przeskakiwanie” sygnału z ominięciem warstw nie jest dozwolone.

## Przykład wielowarstwowej sieci neuronowej

Wejścia:  $X_1$   $X_2$   $X_3$

Wyjście:  $Y$

Model:  $Y = f(X_1 X_2 X_3)$



### Parametry

# Neuronów wejściowych **3**

# Warstw ukrytych **1**

# Neuronów wyjściowych **1**

Liczba wejść zależy od rozmiaru wektora  $X$

Liczba wyjść zależy od rodzaju rozwiązywanego problemu

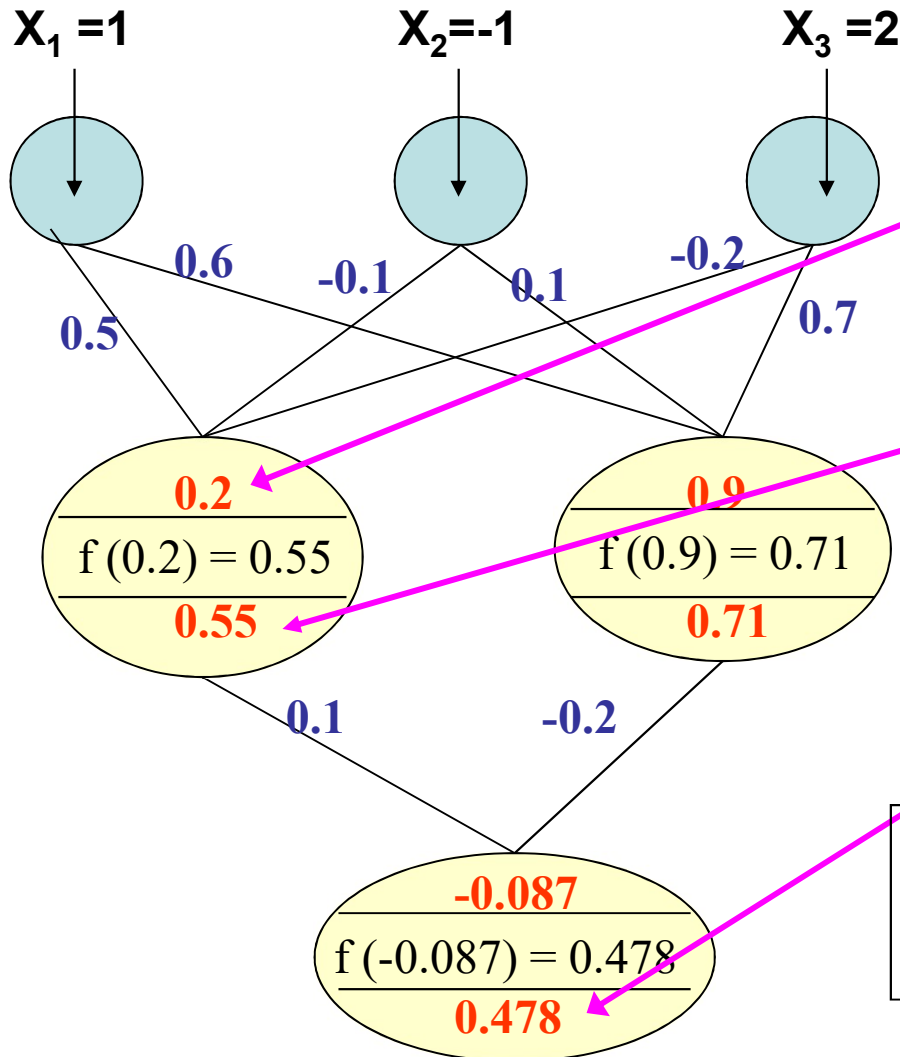


## Przykład użycia SSN do predykcji sygnału.

Wejście:  $X_1 X_2 X_3$

Wyjście:  $Y$

Model:  $Y = f(X_1 X_2 X_3)$



$$0.2 = 0.5 * 1 - 0.1 * (-1) - 0.2 * 2$$

$$f(x) = e^x / (1 + e^x)$$

$$f(0.2) = e^{0.2} / (1 + e^{0.2}) = 0.55$$

Przewidywane  $Y = 0.478$

Spodziewana wartość  $Y = 2$   
zatem  
Błąd predykcji =  $(2 - 0.478) = 1.522$

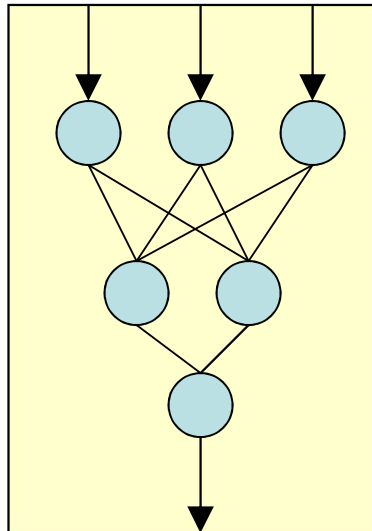
## Budowanie modelu SSN

Jak zbudować model dla konkretnego zagadnienia ?

**Wejście:**  $X_1 X_2 X_3$       **Wyjście:**  $Y$       **Model:**  $Y = f(X_1 X_2 X_3)$

# Wejść = **3**      # Neuronów wyjściowych = # Wyjść = **1**

# Warstw ukrytych = ???      Może **1**      Nie ma jednej strategii.  
# Neuronów warstwie ukrytej = ???      Może **2**      Metoda prób i błędów



Jeśli architektura SSN jest ustalona ... to jak znaleźć wartości wag???

8 wartości wag do ustalenia.

$$\underline{W} = (W_1, W_2, \dots, W_8)$$

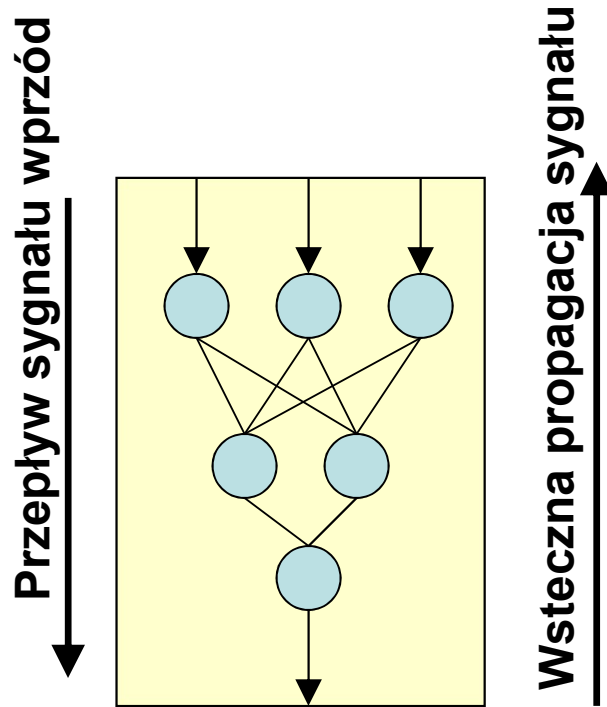
Dane ćwiczeniowe:  $(Y_i, X_{1i}, X_{2i}, \dots, X_{pi})$   $i = 1, 2, \dots, n$   
Określając wartości  $\underline{W}$ , otrzymujemy wartości wyjściowe  $(V_1, V_2, \dots, V_n)$

Zatem  $Y = f(\underline{W})$ .

Definiujemy całkowity błąd predykcji

$$E = \sum (Y_i - V_i)^2$$

## Ćwiczenie modelu SSN



$$E = \sum (Y_i - V_i)^2$$

Jak uczyć SSN ?

- Ustalenie losowych wartości wag
- Obliczenie przepływu sygnału w przód  
 $X \rightarrow$  Sieć  $V \rightarrow$ ; Błąd =  $(Y - V)$
- Dobieramy wagi aby zmniejszyć Błąd SSN
- Kolejne obliczenie przepływu sygnału.  
Dobór wag.
- Powtarzanie procedury, do momentu uzyskania Błędu SSN mniejszego od założonego

## Dobór wag w algorytmie uczenia metodą Wstecznej Propagacji Błędów

$V_i$  – predykcja zwracana przez sieć w  $i$ -tej obserwacji – jest funkcją wektora wag  $\underline{W} = (W_1, W_2, \dots)$

Stąd,  $E$  jest całkowitym błędem predykcji - również jest funkcją  $W$

$$E(\underline{W}) = \sum [Y_i - V_i(\underline{W})]^2$$

### Metoda Największego Spadku :

Dla każdej wagi  $W_i$ , uaktualnianie wag odbywa się wg wzoru:

$$\mathbf{W}_{\text{new}} = \mathbf{W}_{\text{old}} + \alpha * (\partial \mathbf{E} / \partial \mathbf{W}) |_{\mathbf{W}_{\text{old}}}$$

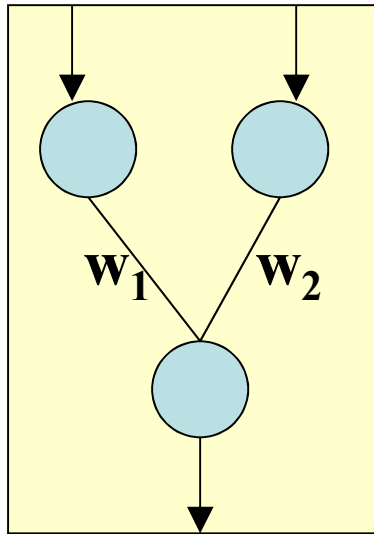
$\alpha$  = współczynnik uczenia [0;1]

Niekiedy do zmiany wag stosowany jest wzór:

$$\mathbf{W}_{(t+1)} = \mathbf{W}_{(t)} + \alpha * (\partial \mathbf{E} / \partial \mathbf{W}) |_{\mathbf{W}_{(t)}} + \beta * (\mathbf{W}_{(t)} - \mathbf{W}_{(t-1)})$$

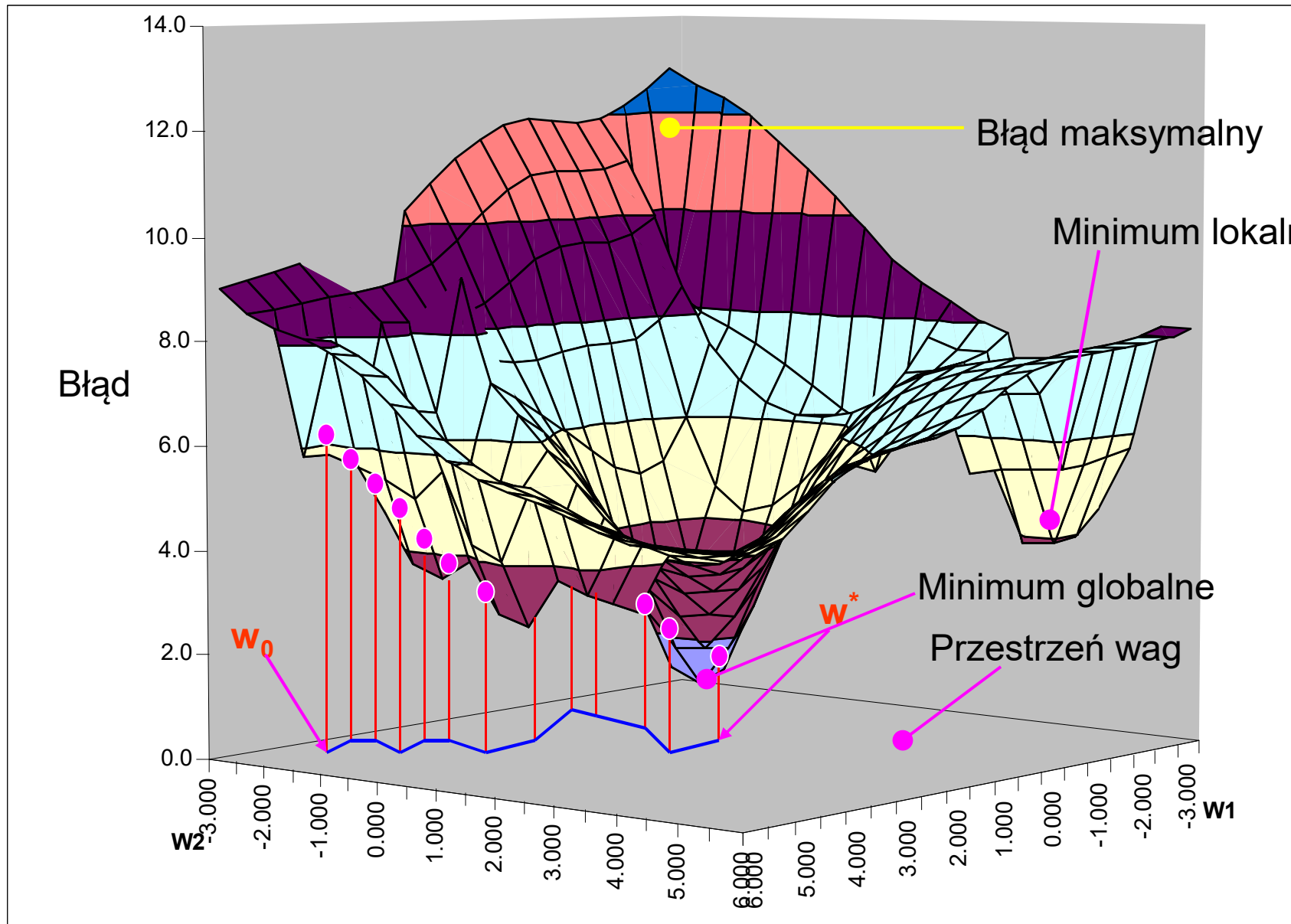
$\beta$  - Współczynnik „Momentum” [0;1]

## Geometryczna interpretacja doboru wag

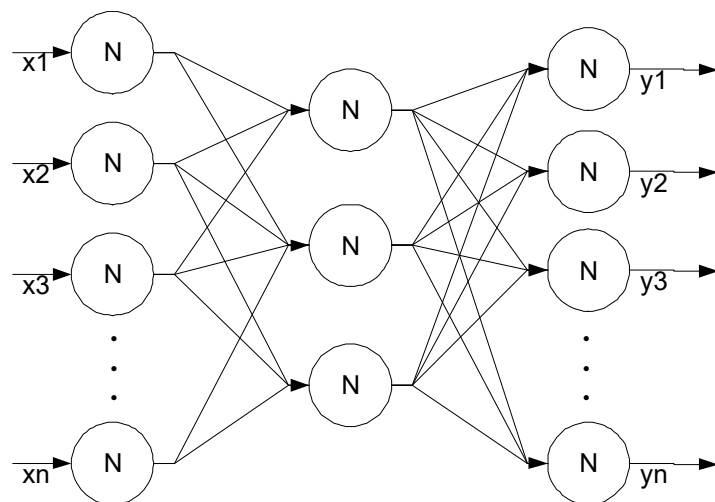


$$E(w_1, w_2) = \sum [ Y_i - V_i(w_1, w_2) ]^2$$

# Geometryczna interpretacja doboru wag



## Sieci jednokierunkowe wielowarstwowe



Warstwa wejściowa    Warstwa ukryta    Warstwa wyjściowa

W sieciach jednokierunkowych, wielowarstwowych sygnały są przekazywane od warstwy wejściowej do warstwy wyjściowej, przy czym nie występują sprzężenia zwrotne do warstw poprzednich.

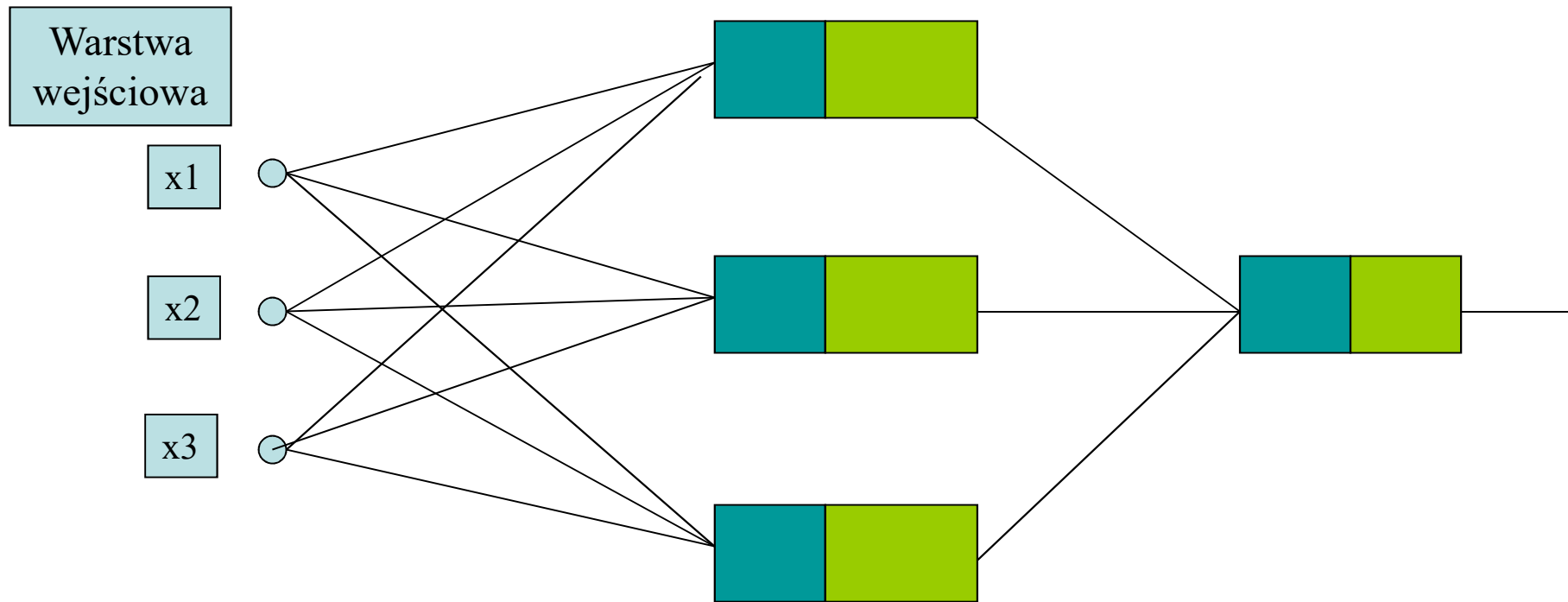
Jednym z najczęściej stosowanych algorytmów uczenia tego typu sieci jest

### **Algorytm wstecznej propagacji błędów.**

Algorytm ten należy do klasy algorytmów uczenia sieci z nauczycielem.

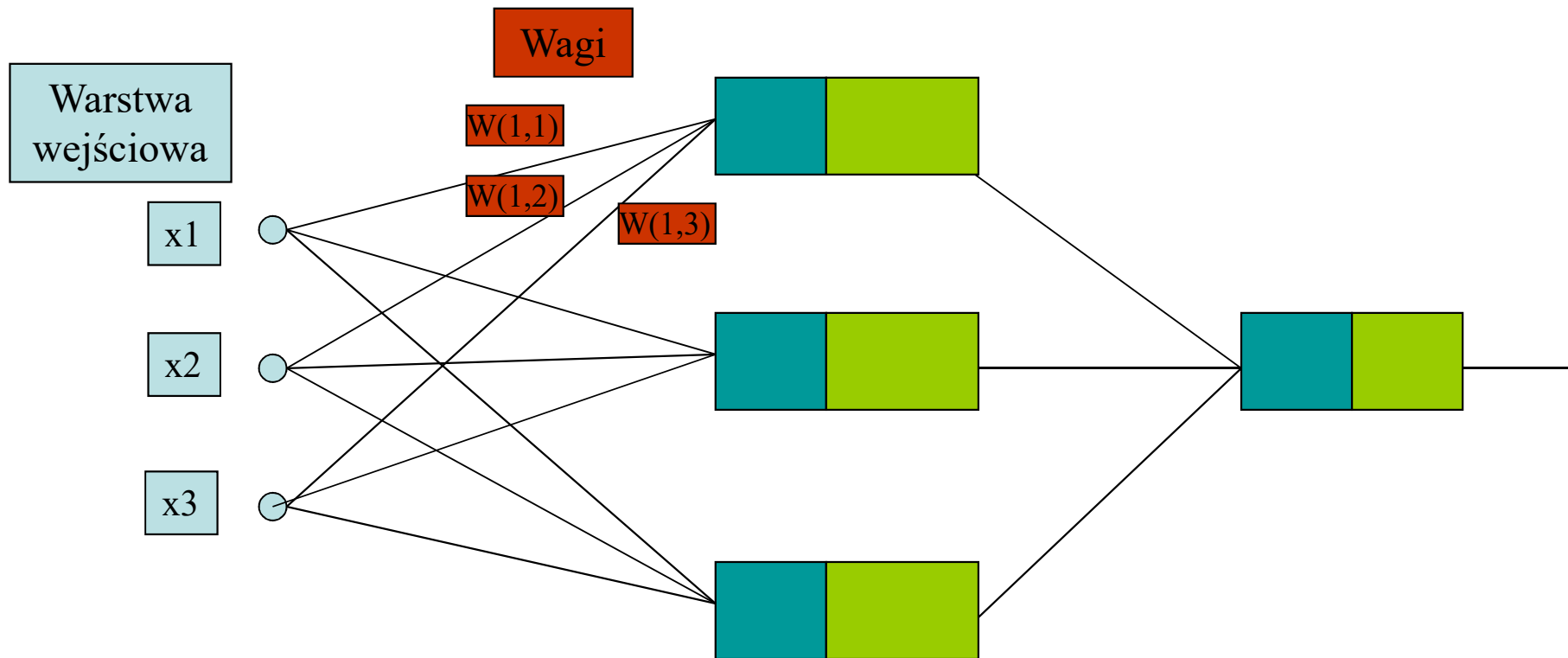
Wykorzystuje on do nauki dwa wektory: uczący i wzorcowy.

# Algorytm Wstecznej Propagacji Błędu



Na warstwę wejściową podawane są sygnały wejściowe ( $x_1..x_n$ ). Mogą to być np. dane próbki sygnału zmieniającego się w czasie ( $y = A \sin(2\pi f \cdot t + \varphi)$ ), próbki głosu do rozpoznania, lub inne.





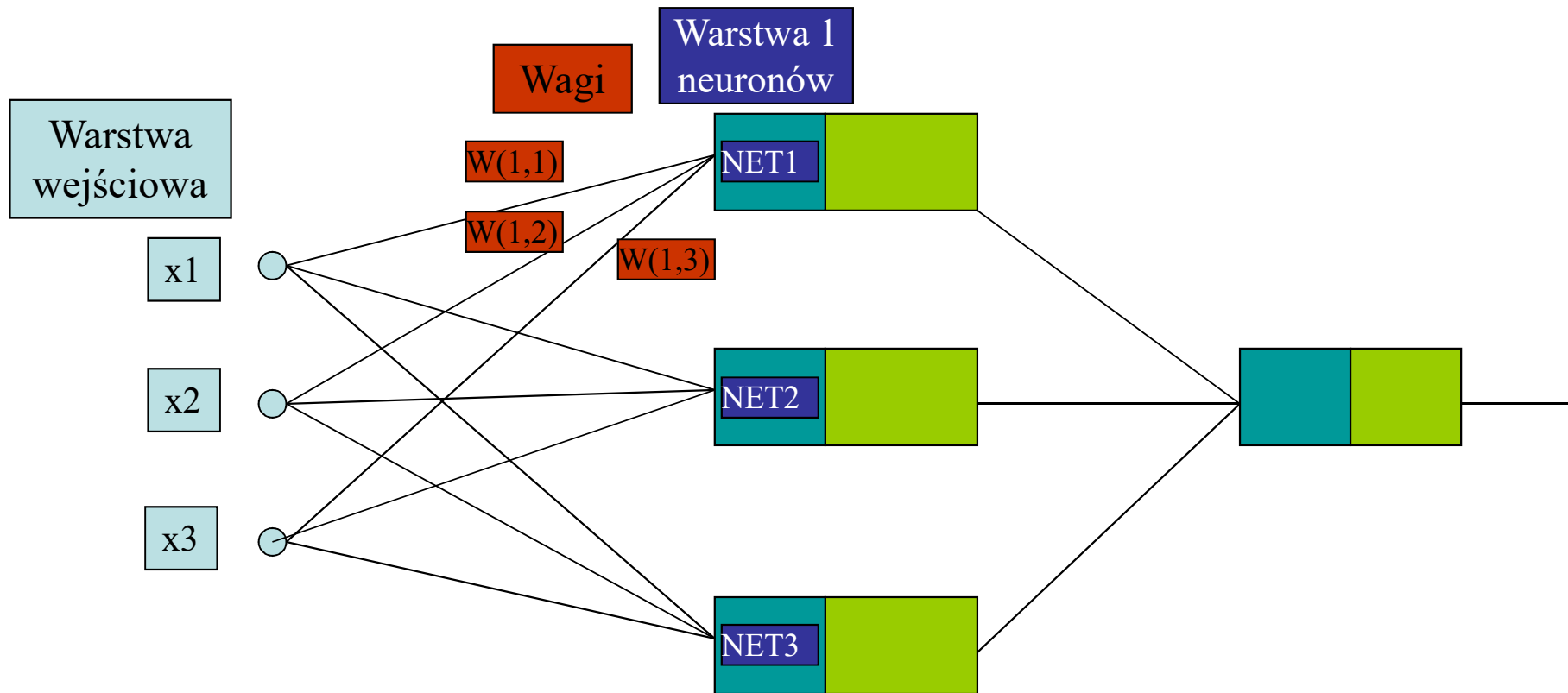
Sygnały wejściowe mnożone są odpowiednio przez wagi każdego z neuronów.

$$x_1 * w(1,1), \quad x_2 * w(1,2), \quad x_3 * w(1,3)$$

$$x_1 * w(2,1), \quad x_2 * w(2,2), \quad x_3 * w(2,3)$$

$$x_1 * w(3,1), \quad x_2 * w(3,2), \quad x_3 * w(3,3)$$

Algorytm Wstecznej Propagacji Błędów



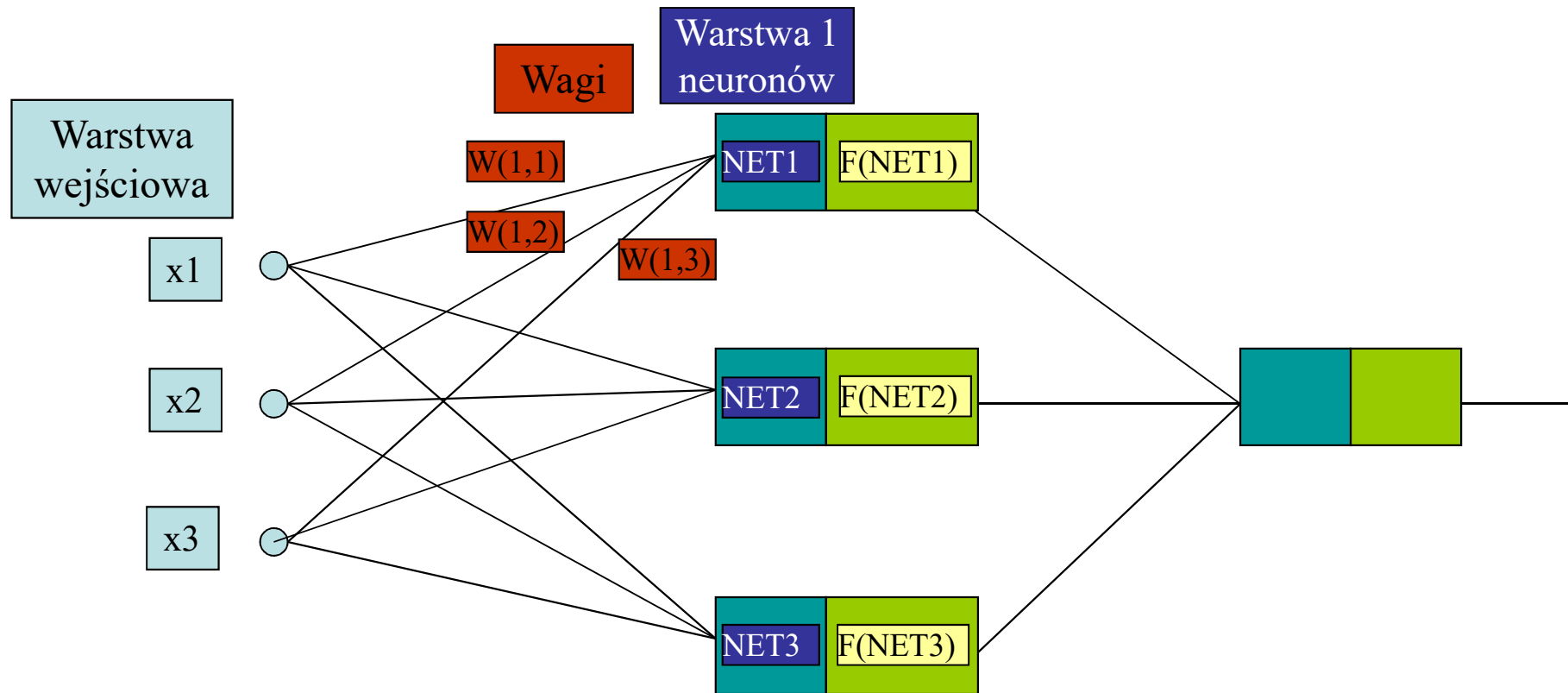
wartości uzyskane z przemnożenia sygnałów wejściowych oraz odpowiednich wag są sumowane w każdym neuronie dając odpowiednie wartości potencjału neuronu (NET):

$$\text{NET1} = x1 * w(1,1) + x2 * w(1,2) + x3 * w(1,3)$$

$$\text{NET2} = x1 * w(2,1) + x2 * w(2,2) + x3 * w(2,3)$$

$$\text{NET3} = x1 * w(3,1) + x2 * w(3,2) + x3 * w(3,3)$$

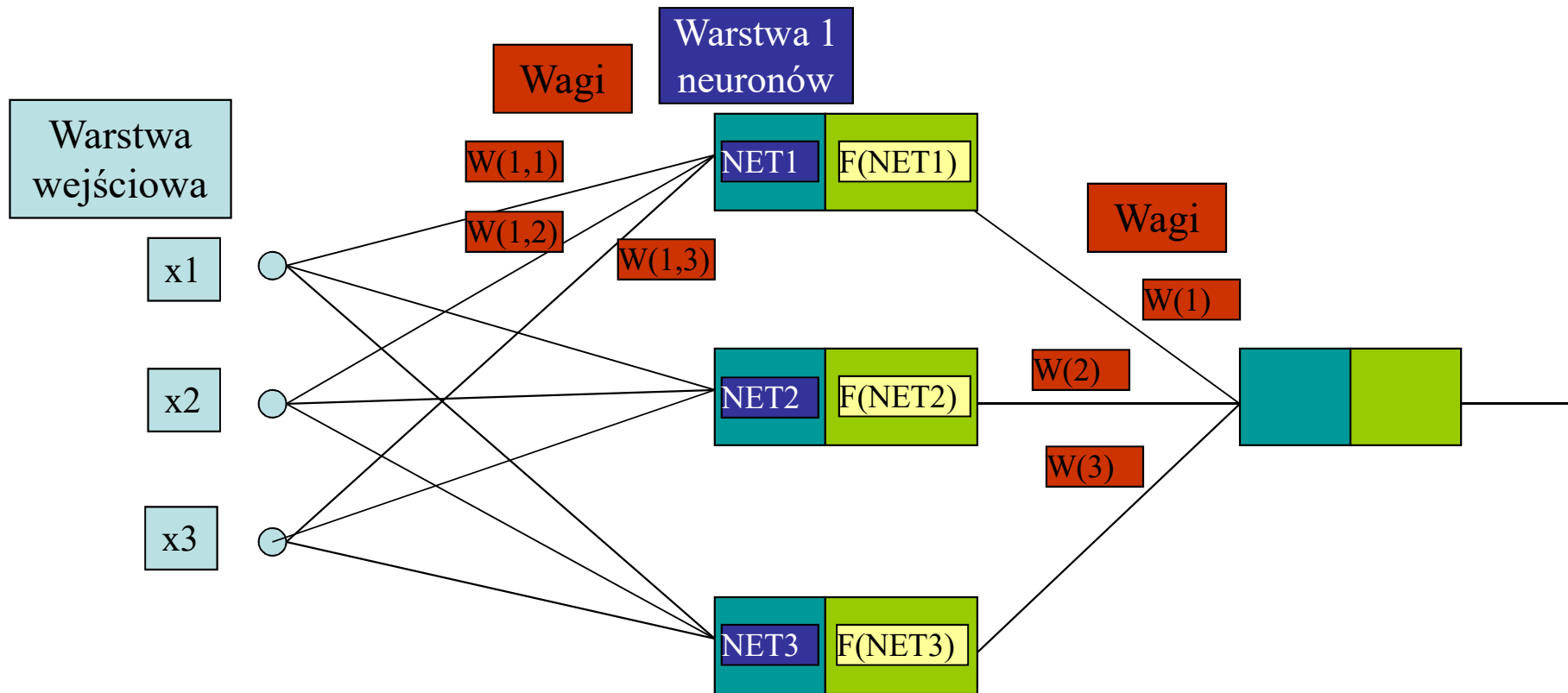
Algorytm Wstecznej Propagacji Błędów



Sygnał potencjału przekazywany jest jako argument do funkcji aktywacji neuronu  $f(\text{NET})$ . Jest ona nieliniowa oraz koniecznie różniczkowalna. Najczęściej spotykaną funkcją jest tzw. funkcja sigmoidalna postaci:

$$f(\text{NET}_i) = \frac{1}{1 + e^{-(\beta x)}}$$

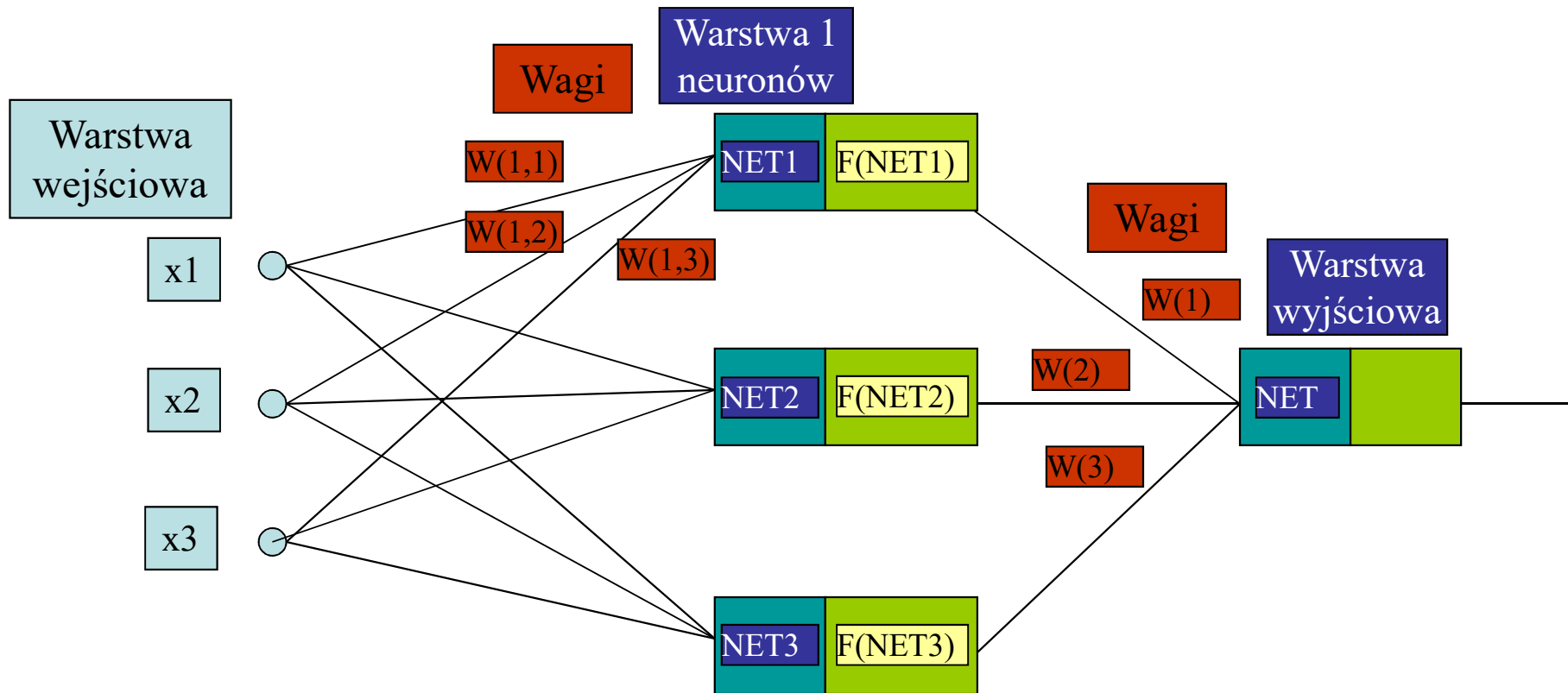
Algorytm Wstecznej Propagacji Błędu



Otrzymane wartości ze wszystkich neuronów zostają przemnożone przez odpowiednie wagi neuronu warstwy wyjściowej:

$$F(\text{NET1}) * w(1), \quad F(\text{NET2}) * w(2), \quad F(\text{NET3}) * w(3)$$

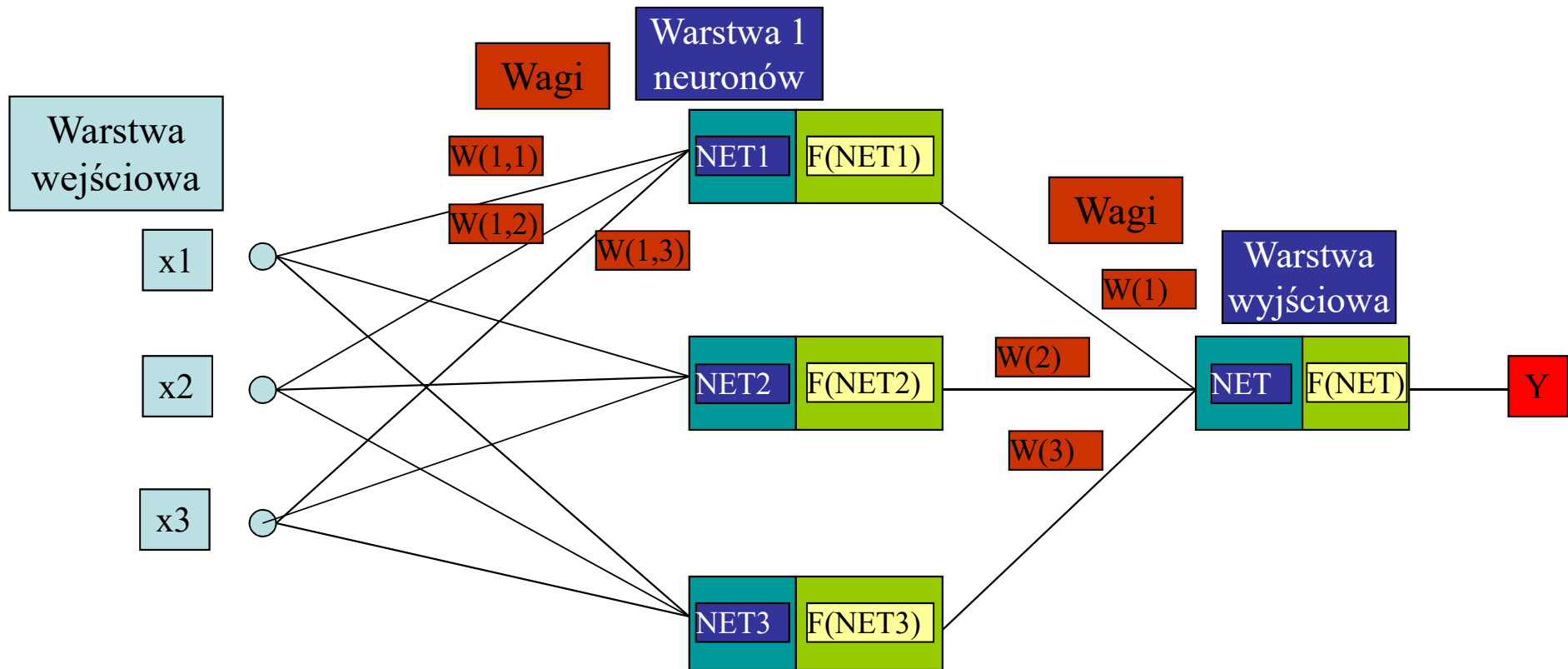
Algorytm Wstecznej Propagacji Błędów



w warstwie wyjściowej ważone wartości wyjściowe z poprzedniej warstwy zostają zsumowane, dając w wyniku potencjał NET

$$NET = f(NET1) * w(1) + f(NET2) * w(2) + f(NET3) * w(3)$$

Algorytm Wstecznej Propagacji Błędu

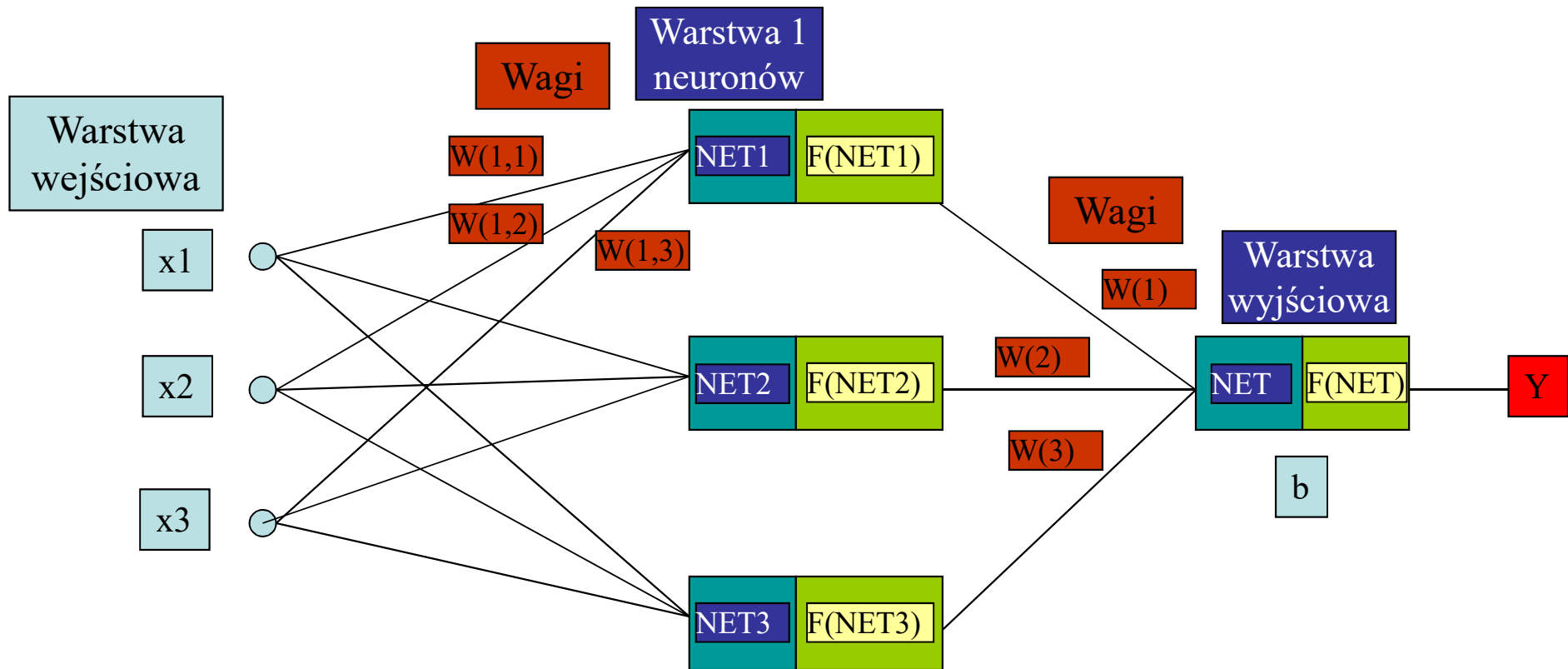


Sygnal NET zostaje podany jako argument nieliniowej funkcji sigmoidalnej  $f(\text{NET})$  postaci:

$$y = \frac{1}{1 + e^{-(\beta x)}}$$

Dając w wyniku wartość wyjściową sieci ( $y$ ).

Algorytm Wstecznej Propagacji Błędów

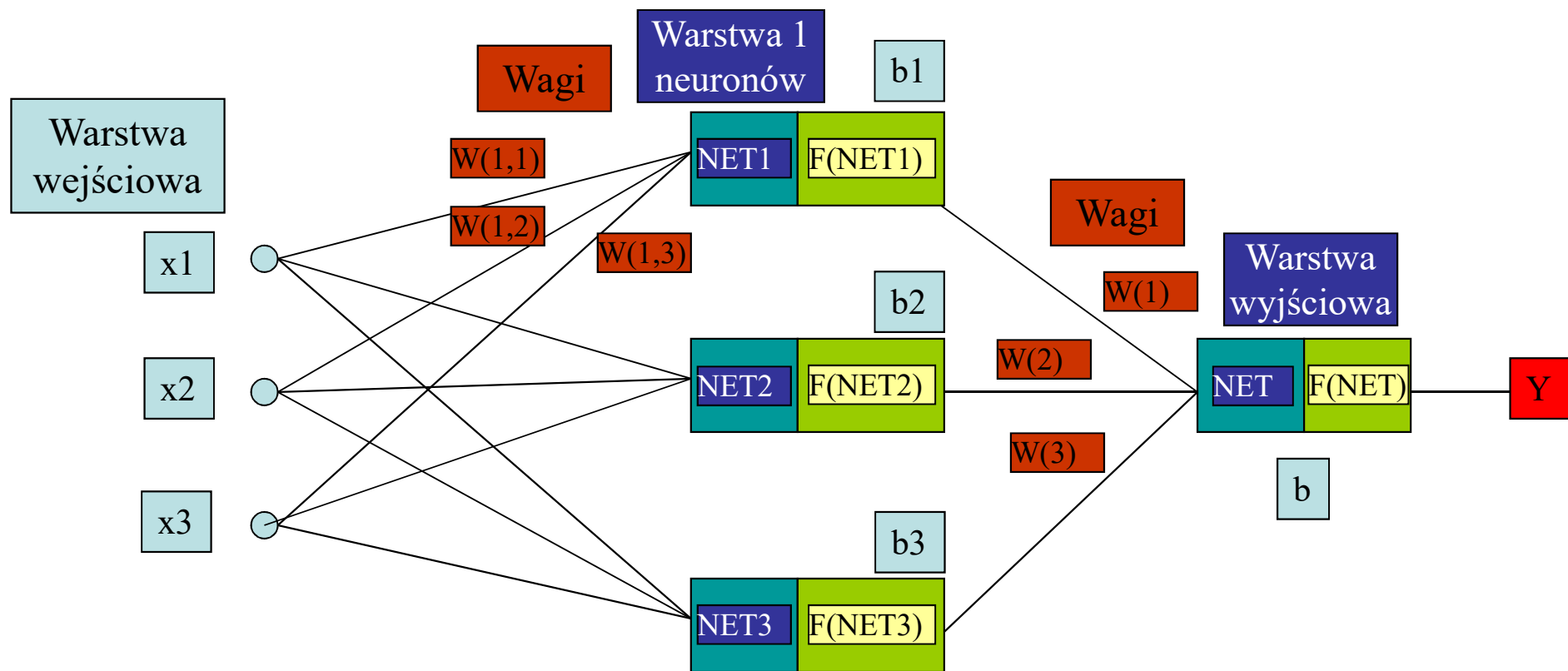


wyznaczony jest błąd sieci porównując odpowiedź sieci ( $Y$ ) z odpowiedzią właściwą ( $Z$ ) otrzymując:

$$b = f'(NET)(Y - Z)$$

Będący błędem neuronu warstwy wyjściowej

Algorytm Wstecznej Propagacji Błędu



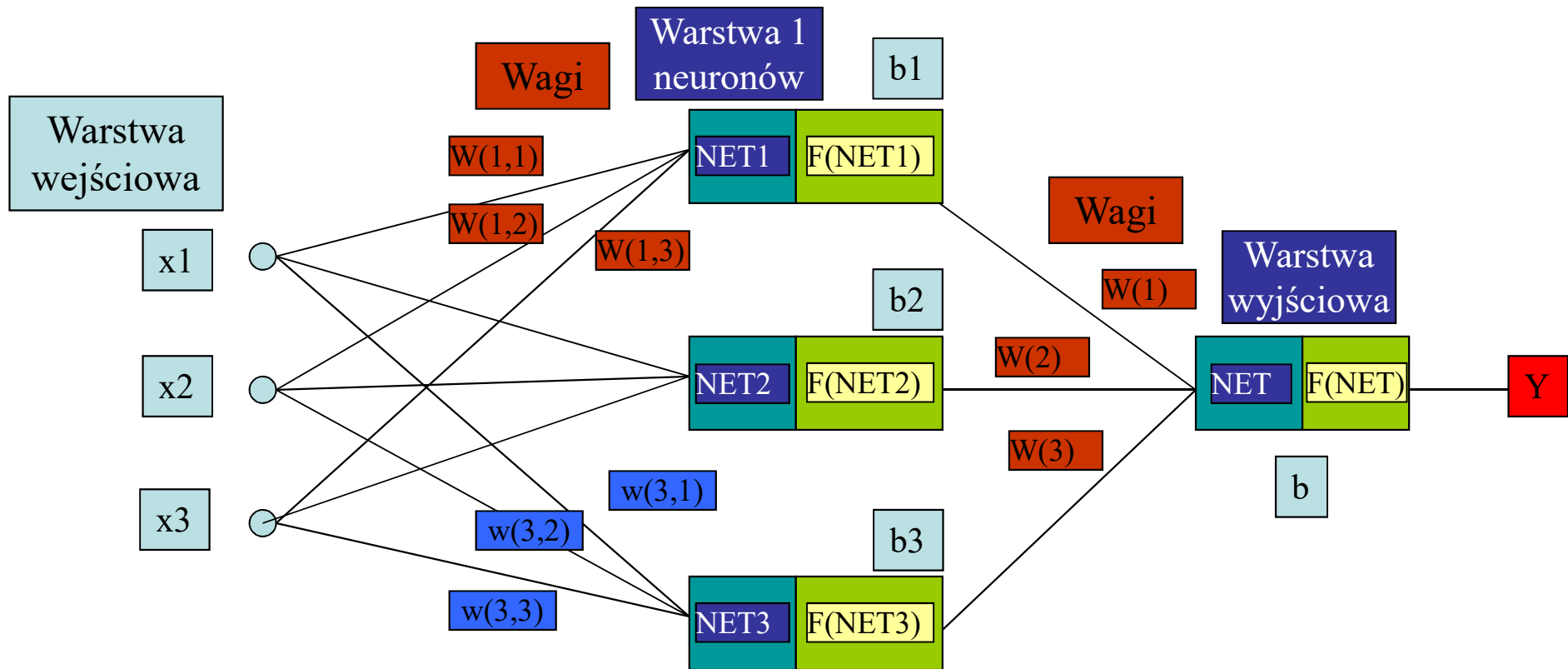
Błąd neuronów warstwy ukrytej wyznaczany jest za pomocą wzoru:

$$b_i = f'(NET)bw_i$$

Co oznacza, że błąd neuronu warstwy ukrytej jest proporcjonalny do błędu neuronu z którym jest „mocno połączony”. Im większy jest błąd jego następcy, tym większy powinien być jego błąd. wyznaczanie błędów odbywa się w kierunku odwrotnym do zwykłej propagacji sygnału i stąd wzięła się nazwa algorytmu.

Algorytm Wstecznej Propagacji Błędu



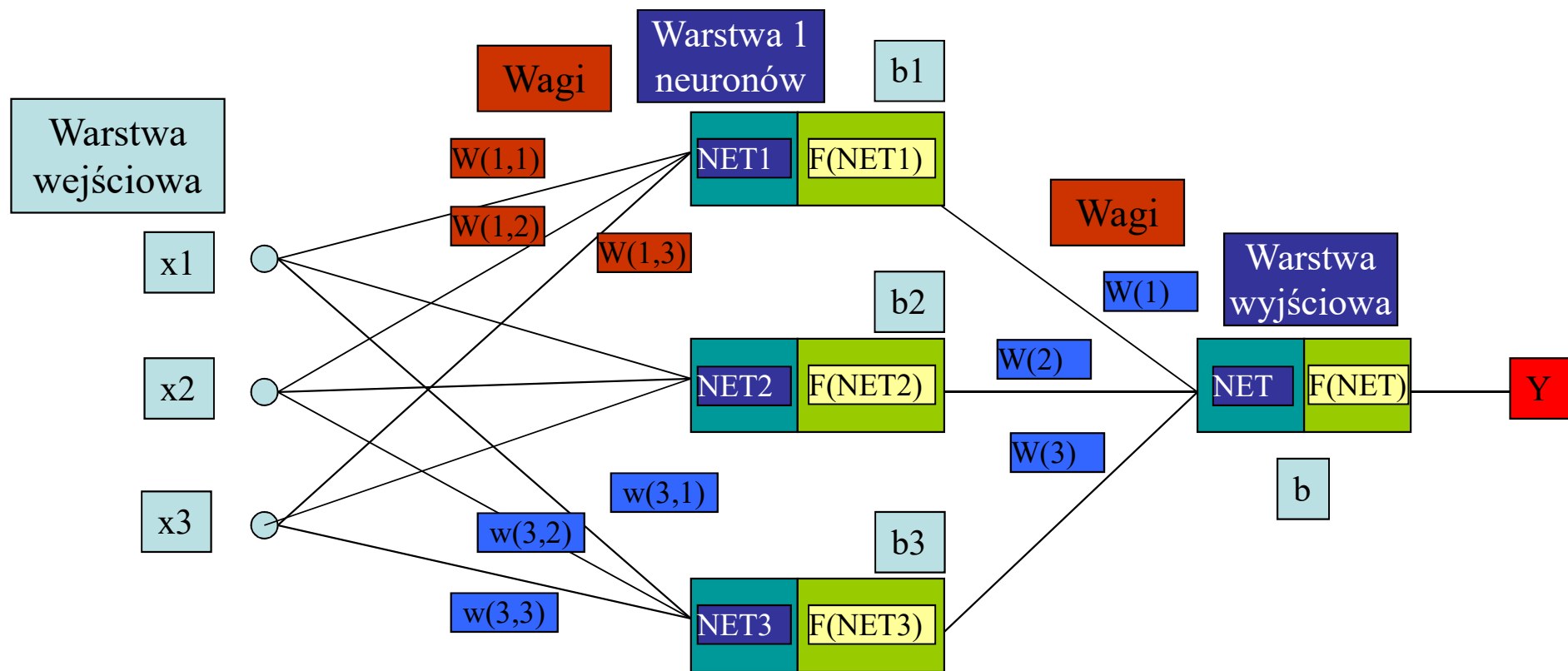


Aktualizacja wag zachodzi w kierunku przewodzenia sygnałów według wzoru:

$$w_{i,j} = w(stare)_{i,j} + \eta b_i x_i$$

Gdzie  $\eta$  tzw. współczynnik uczenia, zwykle 0..0.5 odpowiedzialny jest za szybkość oraz stabilność procesu uczenia. Zmianie podlegają wszystkie wagi neuronów warstwy ukrytej

Algorytm Wstecznej Propagacji Błędów



Aktualizacja w warstwie wyjściowej analogicznie według wzoru:

$$w_i = w(stare)_i + \eta b_i f(NET_i)$$

w wyniku algorytmu, nowe wagi powinny być lepiej przystosowane do pokazanego wzorca wejściowego, co oznacza, że przy kolejnym pokazie, wygenerowany błąd będzie mniejszy.

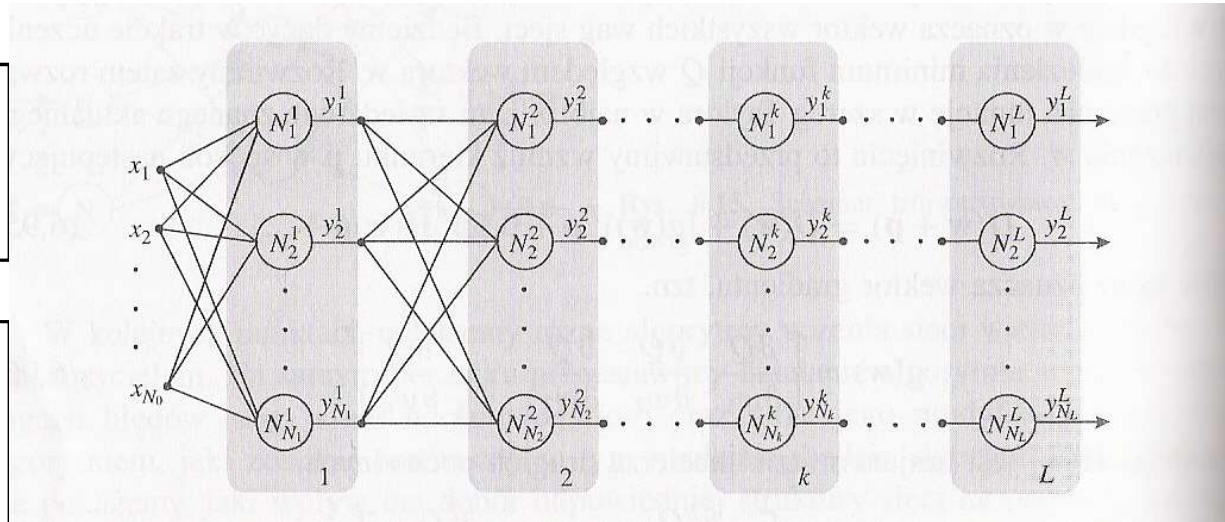
Algorytm Wstecznej Propagacji Błędu

# Uczenie sieci jednokierunkowej, wielowarstwowej metodą wstecznej propagacji błędu (ang. *error backpropagation*)

Podajemy na wejście sieci wartości wejściowe ciągu uczącego

Obliczamy wartości wyjściowe kolejno dla wszystkich neuronów od warstwy wejściowej do warstwy wyjściowej

Modyfikujemy wartości wag w sieci, aby wartość wyjściowa sieci była zbliżona do wartości wzorcowej.



$$N_i^k, i = 1, \dots, N_k$$

$N_i^k$  - neurony sigmoidalne

$$x = [x_1(t), \dots, x_{N_0}(t)]^T \quad t = 1, 2, \dots \quad \text{sygnały wejściowe}$$

$$y_i^{(k)}(t), i = 1, \dots, N_K, k = 1, \dots, L \quad \text{sygnały wyjściowe}$$

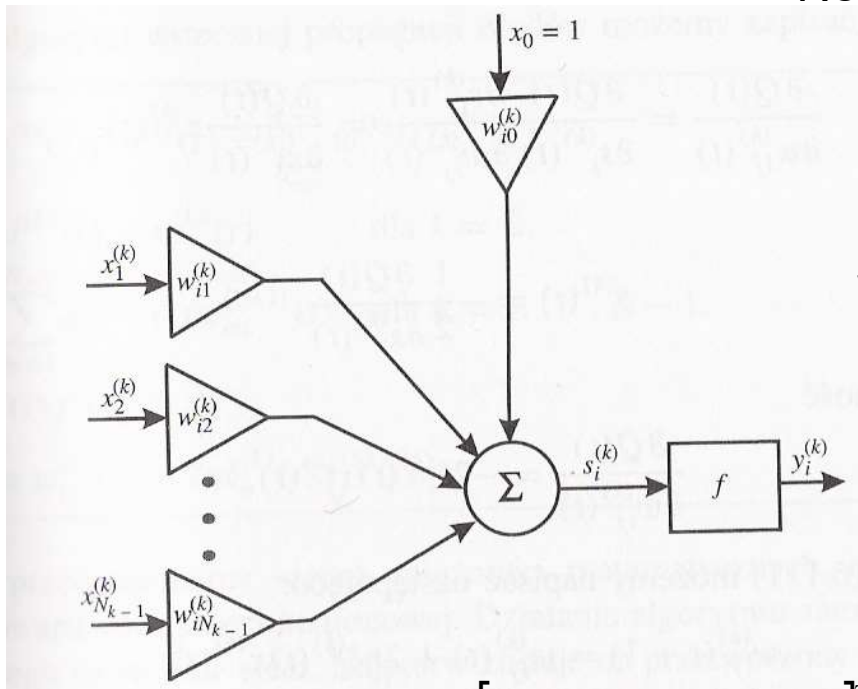
Model neuronu  $N_i^k$

Neuron posiada k-wejść tworzących wektor:

$$x^k(t) = [x_0^k(t), \dots, x_{N_{k-1}}^k(t)]^T$$

Jest on powiązany z sygnałem wyjściowym warstwy (k-1):

$$x_i^k(t) = \begin{cases} x_i(t), & \text{dla } k = 1 \\ y_i^{k-1}(t), & \text{dla } k = 2, \dots, L \\ +1 & \text{dla } i = 0, k = 1, \dots, L \end{cases}$$



$$w_i^k(t) = [w_{i,0}^k(t), \dots, w_{i,N_{k-1}}^k(t)]^T \quad i = 1, \dots, N_K, k = 1, \dots, L$$

Sygnał wyjściowy neuronu w chwili t, t=1,2,... jest określony jako:  $y_i^{(k)}(t) = f(s_i^k(t))$

gdzie:  $s_i^k(t) = \sum_{j=0}^{N_{k-1}} w_{ij}^k(t) x_j^k(t)$   $w_{ij}^k$  - waga i-tego neuronu w warstwie k połączonego z j-tym wejściem.  $x_j^k$  - j-te wejście w warstwie k.

Sygnały wyjściowe neuronu:  $y_1^L(t), y_2^L, \dots, y_{N_L}^L(t)$

Sygnały wzorcowe:  $d_1^L(t), d_2^L, \dots, d_{N_L}^L(t)$

Błąd na wyjściu sieci definiujemy następująco:  $Q(t) = \sum_{i=1}^{N_L} (\varepsilon_i^L)^2 = \sum_{i=1}^{N_L} (d_i^L(t) - y_i^L(t))^2$

Zmiana wag neuronów w sieci będzie następowała zgodnie z regułą największego spadku:

$$w_{ij}^k(t+1) = w_{ij}^k(t) - \eta \frac{\partial Q(t)}{\partial w_{ij}^k(t)} \quad \text{Ponieważ:} \quad \frac{\partial Q(t)}{\partial w_{ij}^k(t)} = \frac{\partial Q(t)}{\partial s_i^k(t)} \cdot \frac{\partial s_i^k(t)}{\partial w_{ij}^k(t)} = \frac{\partial Q(t)}{\partial s_i^k(t)} \cdot x_j^k(t)$$

$$\text{Oznaczając: } \delta_i^k(t) = -\frac{1}{2} \frac{\partial Q(t)}{\partial s_i^k(t)}$$

$$\text{Otrzymujemy: } \frac{\partial Q(t)}{\partial w_{ij}^k(t)} = -2\delta_i^k(t)x_j^k(t)$$

$$w_{ij}^k(t+1) = w_{ij}^k(t) + 2\eta \delta_i^k(t)x_j^k(t)$$

Sposób obliczania wartości  $\delta_i^k$  jest uzależniony od warstwy sieci.

$$\begin{aligned} \text{Dla warstwy ostatniej: } \delta_i^L(t) &= -\frac{1}{2} \frac{\partial Q(t)}{\partial s_i^L(t)} = -\frac{1}{2} \frac{\partial (d_i^L(t) - y_i^L(t))^2}{\partial s_i^L(t)} = \\ &= Q_i^L(t) \frac{\partial y_i^L(t)}{\partial s_i^L(t)} = Q_i^L(t) f'(s_i^L(t)) \end{aligned}$$

dla dowolnej warstwy  $k \neq L$  mamy:

$$\begin{aligned}\delta_i^L(t) &= -\frac{1}{2} \frac{\partial Q(t)}{\partial s_i^k(t)} = -\frac{1}{2} \sum_{m=1}^{N_{k+1}} \delta_m^{(k+1)}(t) w_{mi}^{(k+1)}(t) f'(s_i^k(t)) = \\ &= f'(s_i^k(t)) \sum_{m=1}^{N_{k+1}} \delta_m^{(k+1)}(t) w_{mi}^{(k+1)}(t)\end{aligned}$$

Błąd w k-tej warstwie (z wyjątkiem ostatniej) dla i-tego neuronu:

$$\varepsilon_i^L(t) = \sum_{m=1}^{N_{k+1}} \delta_m^{(k+1)}(t) w_{mi}^{(k+1)}(t), \quad k = 1, \dots, L-1$$

Zatem:  $\delta_i^k(t) = \varepsilon_i^k(t) f'(s_i^k(t))$

## Algorytm wstecznej propagacji błędów dla sieci wielowarstwowej, jednokierunkowej

Sygnał wyjściowy:

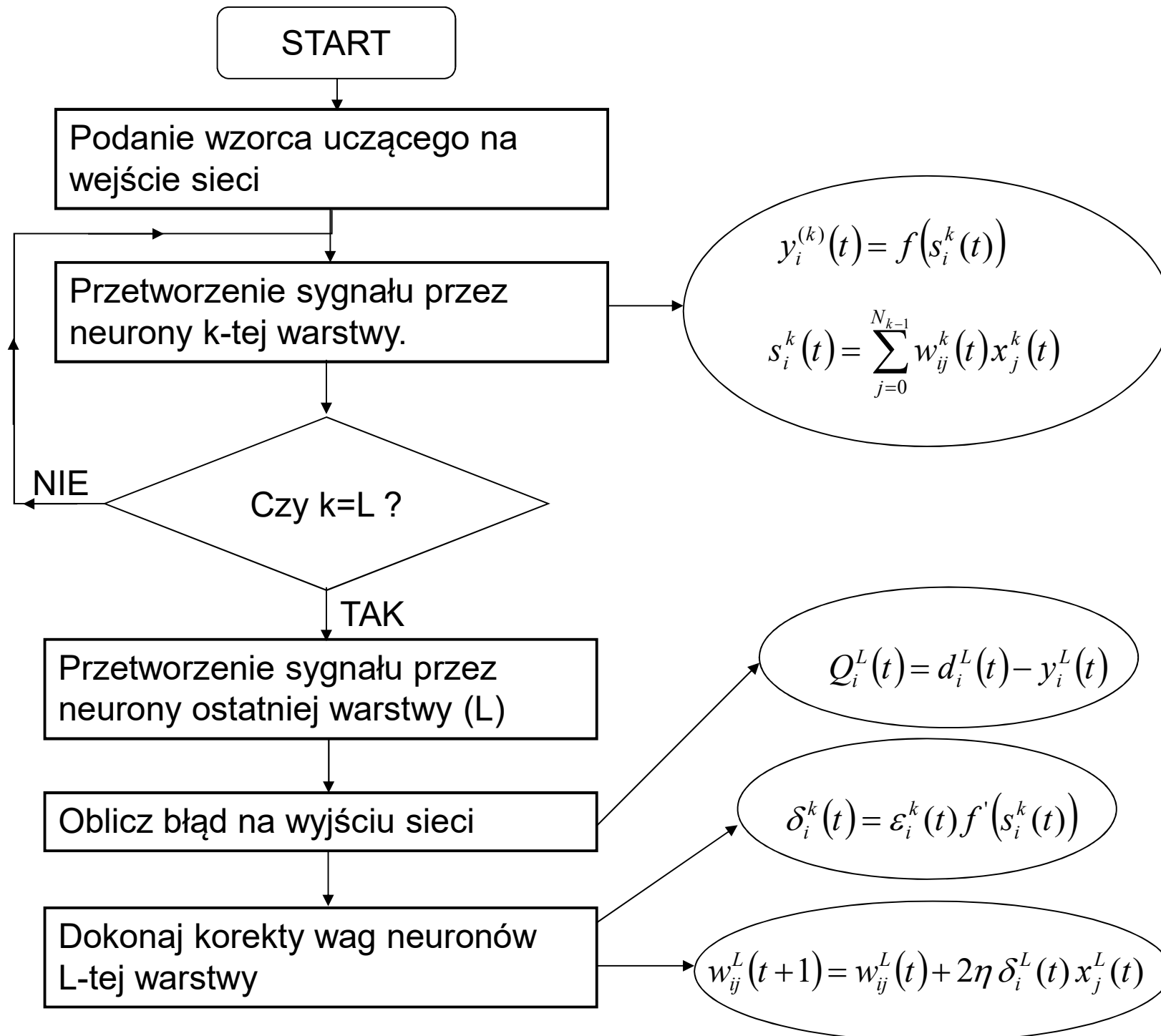
$$y_i^{(k)}(t) = f(s_i^k(t)) \quad \text{gdzie:} \quad s_i^k(t) = \sum_{j=0}^{N_{k-1}} w_{ij}^k(t) x_j^k(t)$$

Miara błędu sieci:

$$Q_i^k(t) = \begin{cases} d_i^L(t) - y_i^L(t) & \text{dla } k = L \\ \sum_{m=1}^{N_{k+1}} \delta_m^{k+1}(t) w_{mi}^{(k+1)}(t) & \text{dla } k = 1, \dots, L-1 \end{cases}$$

$$\delta_i^k(t) = \varepsilon_i^k(t) f'(s_i^k(t))$$

$$w_{ij}^k(t+1) = w_{ij}^k(t) + 2\eta \delta_i^k(t) x_j^k(t)$$





Ponieważ nie znamy wartości  $\delta_i^k$  dla neuronów warstw ukrytych nie możemy zmodyfikować ich wag. Dlatego błąd wyjściowy jest propagowany „do tyłu”, w kierunku wejścia sieci neuronowej zgodnie z połączeniami neuronów i ich funkcjami aktywacji, zgodnie ze wzorem:

$$Q_i^k(t) = \sum_{m=1}^{N_{k+1}} \delta_m^{k+1}(t) w_{mi}^{(k+1)}(t) \quad \text{dla } k = 1, \dots, L-1$$

