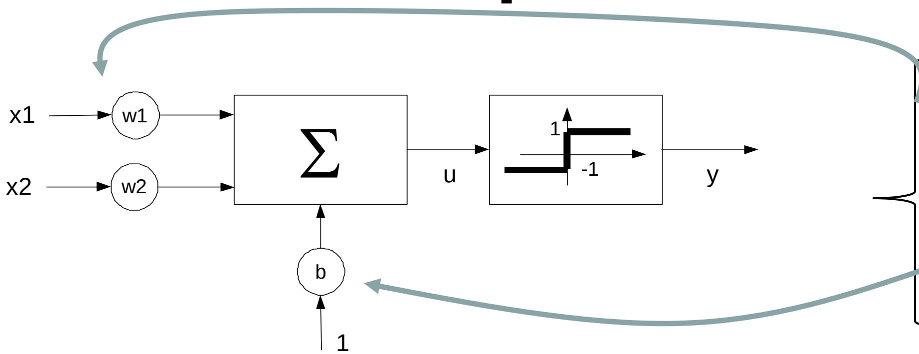
SZTUCZNE SIECI NEURONOWE

Ang. Artifficial neural network

Wykład 2

Dr inż. Piotr Urbanek

Perceptron dwuwejściowy



$$\left.\begin{aligned} W_{nowe} &= W_{stare} + błąd * \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}^T \\ b_{nowe} &= b_{stare} + błąd \end{aligned}\right.$$

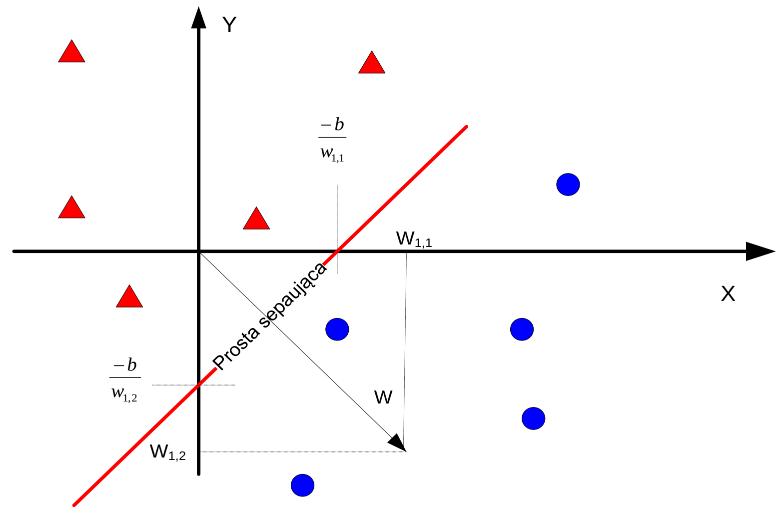
$$y = f(w_1 \cdot x_1 + w_2 \cdot x_2 + b)$$

$$y = f\left(\begin{bmatrix} w_1 & w_2 \end{bmatrix} * \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + b\right)$$

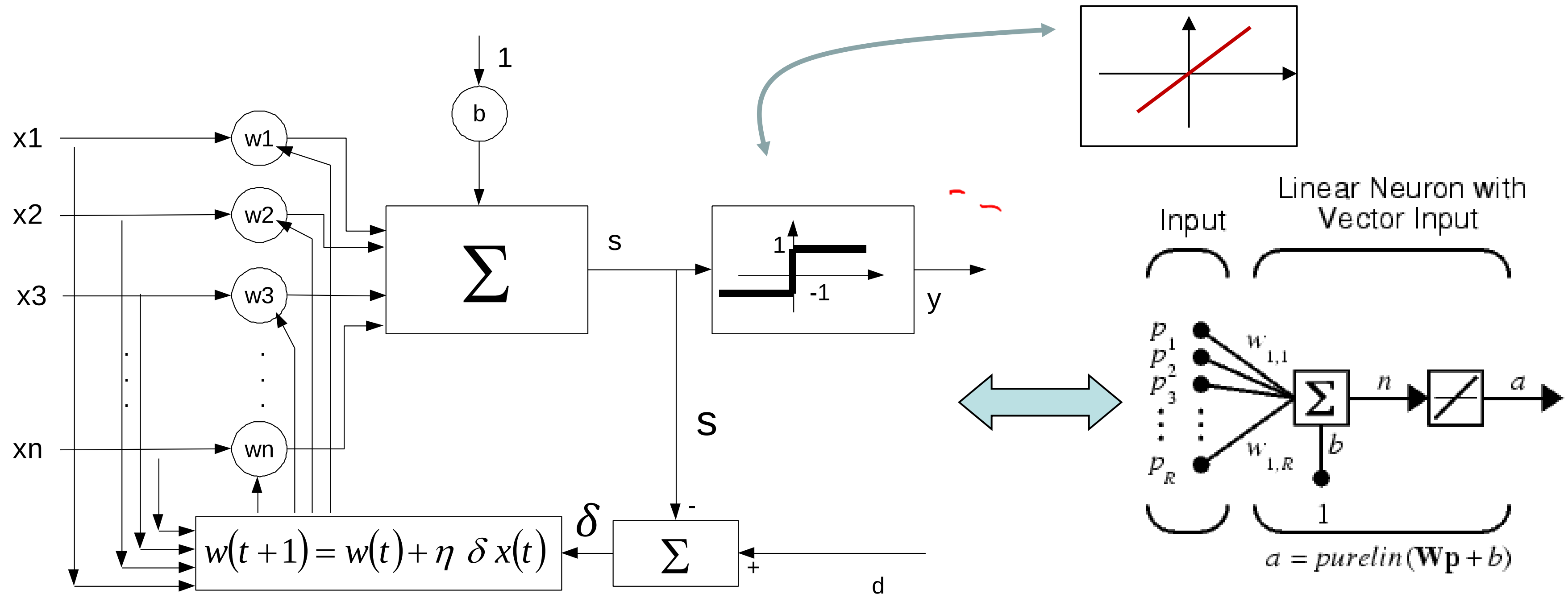
$$y = ax + b$$

$$a \rightarrow \begin{bmatrix} w_1 & w_2 \end{bmatrix}$$

$$x \rightarrow \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$



Neuron Adaline (Adaptive Linear Neuron)



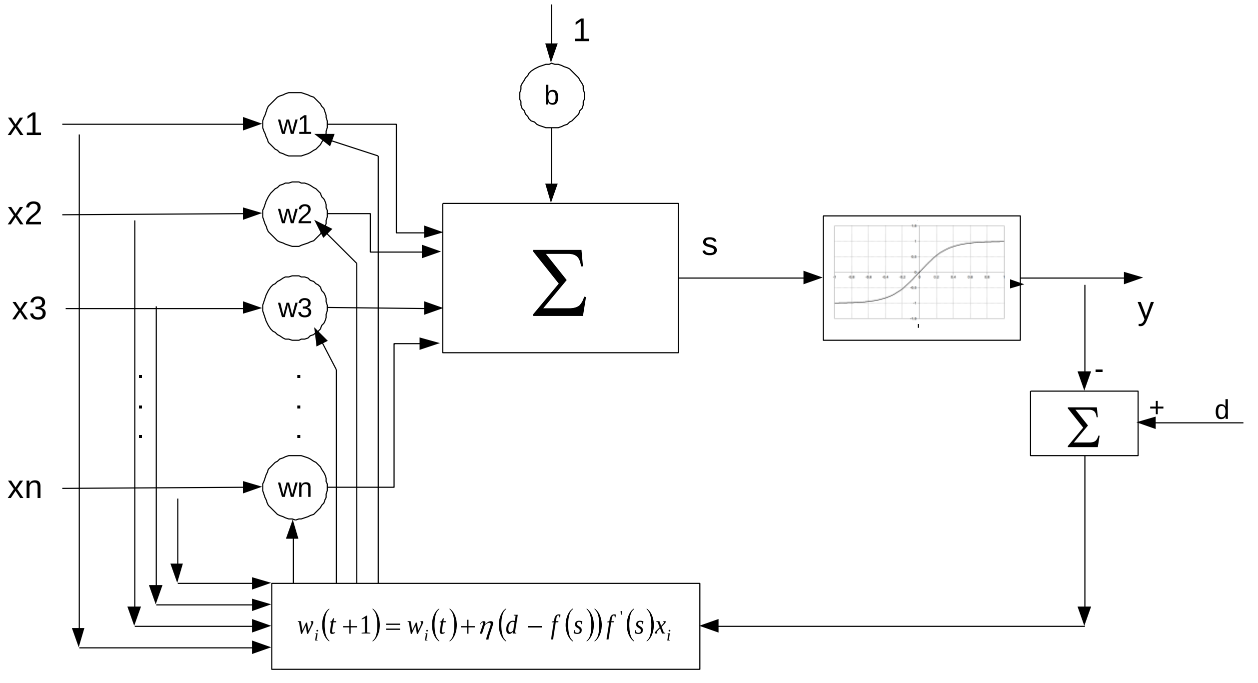
$$\delta = d - s$$

$$\boxed{w_i(t+1) = w_i(t) + \eta\,\delta\, x_i}$$

Minimalizacja funkcji:

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

Neuron sigmoidalny.



Funkcja aktywacji:

unipolarna:

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

$$f^{'}(s)=\beta\, f(s)(1-f(s))$$

bipolarna:

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

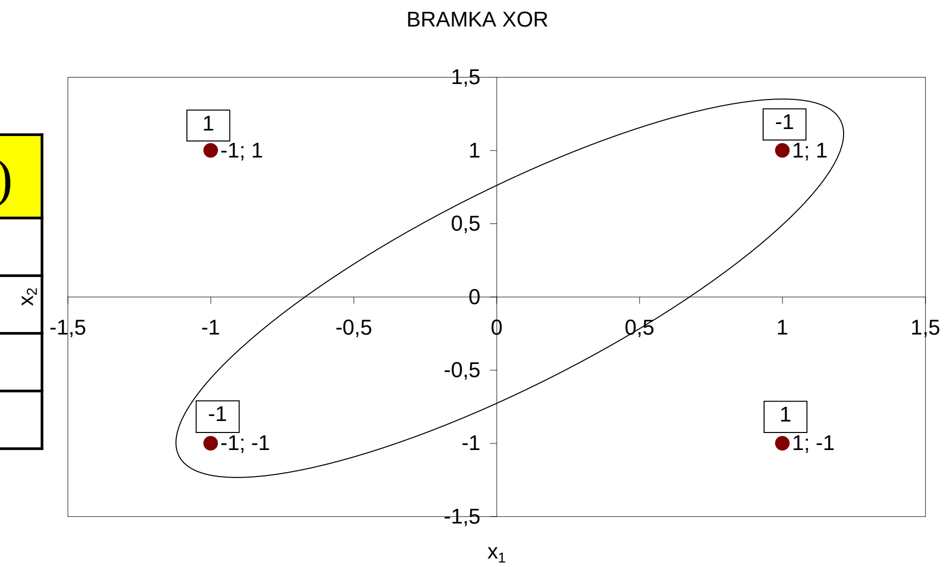
$$f^{'}(s)=\beta(1-f^{2}(s))$$

$$\boxed{w_i(t+1)=w_i(t)+\eta(d-f(s))f^{'}(s)\,x_i}$$

gdzie:

Bramka XOR

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

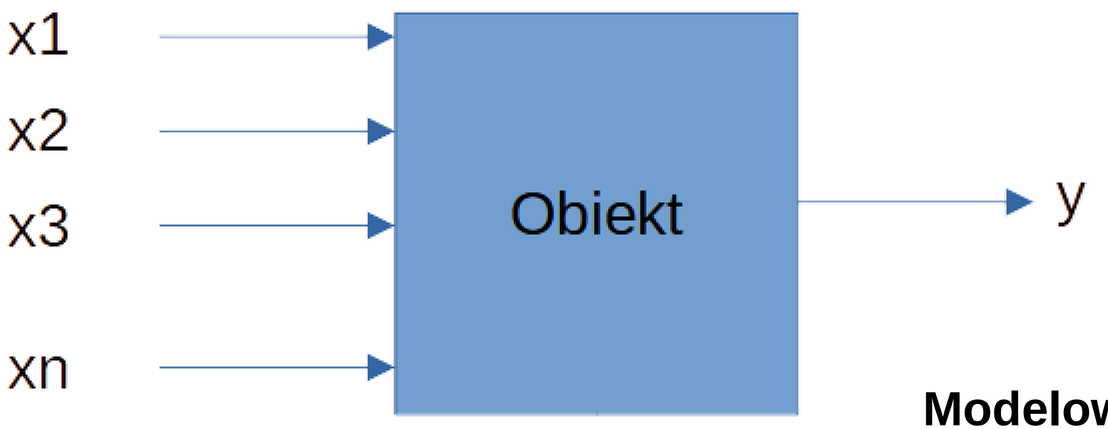


Sieci wielowarstwowe to odpowiednio ze sobą połączone neurony rozmieszczone w dwóch, lub więcej warstwach.

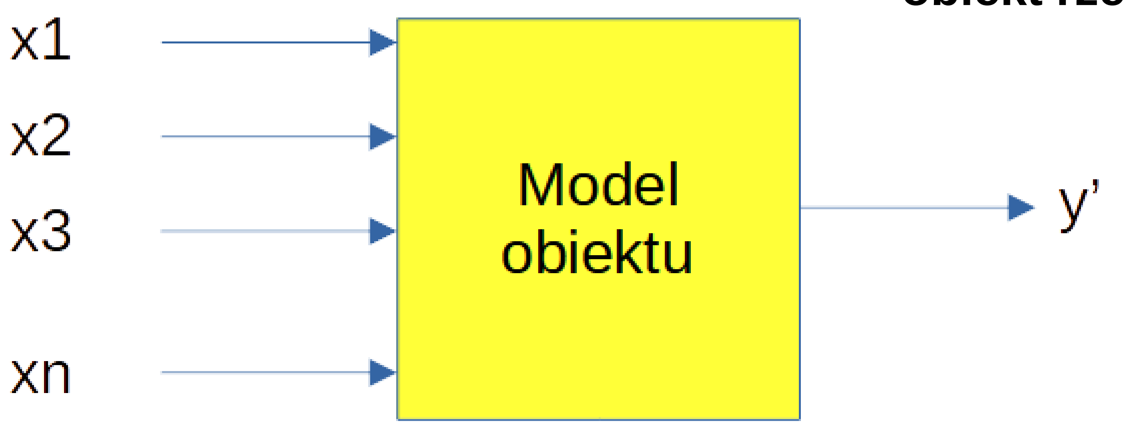
Z reguły są to neurony sigmoidalne, ale używa się również neuronów o liniowych funkcjach aktywacji (są one najczęściej umieszczane w ostatniej, wyjściowej warstwie).

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

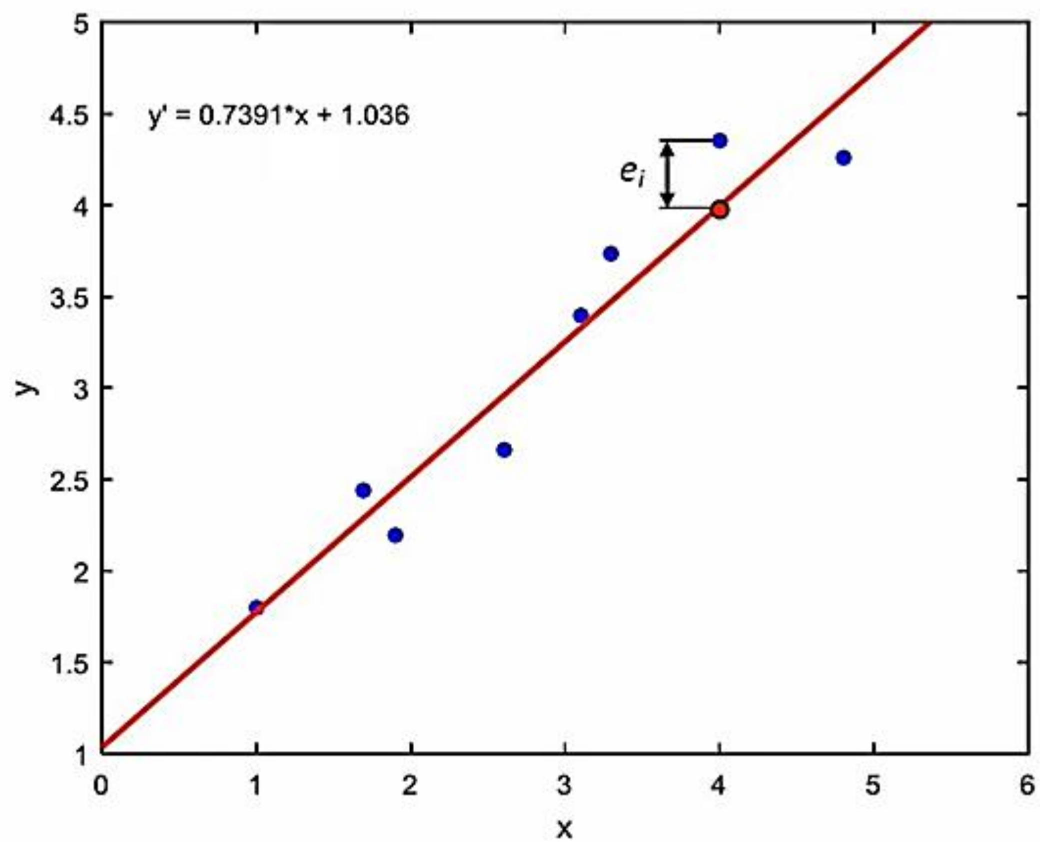
Modelowanie



Modelowanie matematyczne najczęściej polega na aproksymacji zachowania dynamicznego danego obiekt rzeczywistego

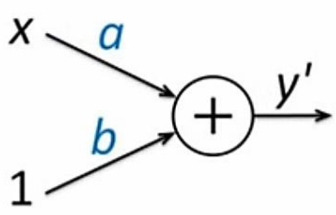


Aproksymacja funkcją liniową



$$E \rightarrow min$$

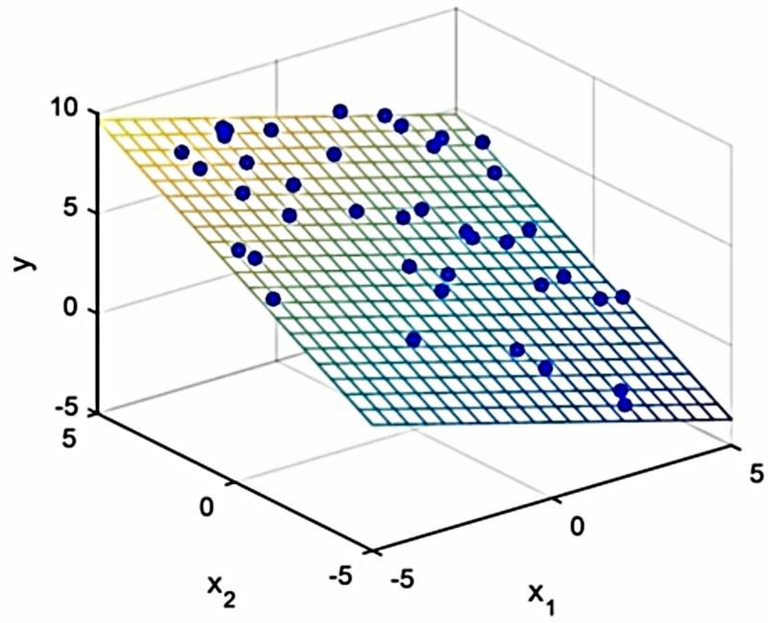
$$h(x) = ax + b$$



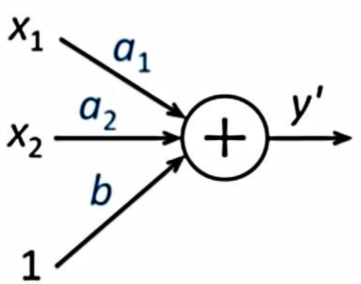
$$E = \frac{1}{M}\sum_{i=1}[y_i - h(x_i)]^2 = \frac{1}{M}\sum_{i=1} e_i^2$$

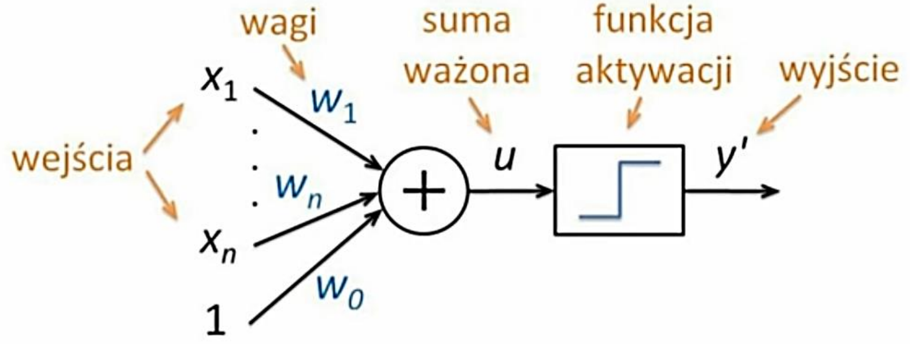
Aproksymacja funkcji dwóch zmiennych

x_1	-1,96	-4,54	-3,05	2,20	2,22	3,78	...	-4,29
x_2	4,44	0,49	2,28	0,77	-4,74	-0,53	...	0,21
y	7,71	5,68	6,44	2,54	-2,09	0,66	...	5,32

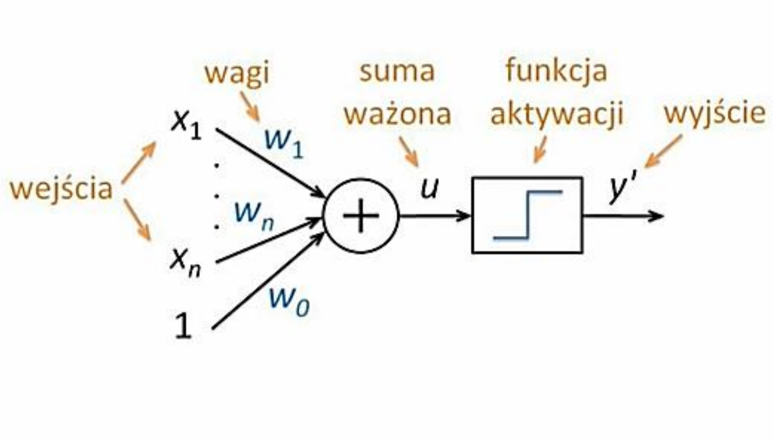


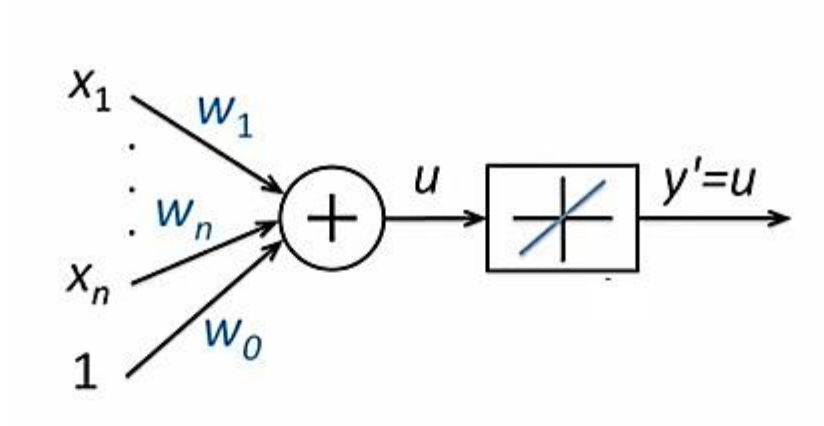
$$h(x) = a_1 x_1 + a_2 x_2 + b$$





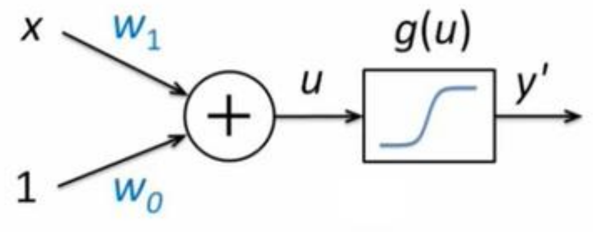
Zamiana funkcji aktywacji





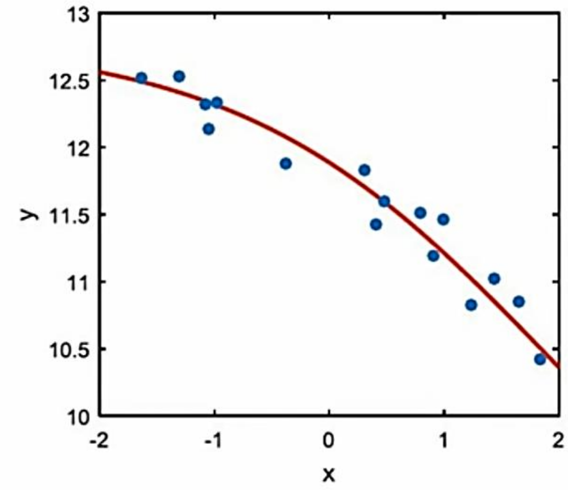
$$h(x) = w_1x_1 + w_2x_2 + \cdots + wnxn + w_0 = \sum_{i=1}^{n} w_ix_i + w_0$$

Rola funkcji aktywacji

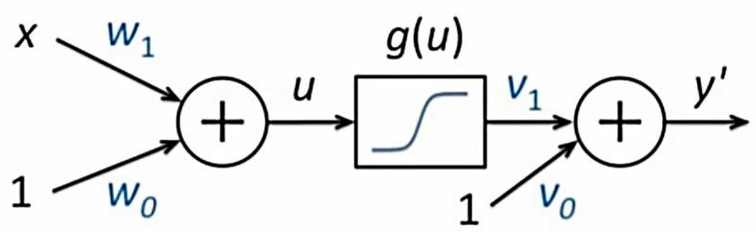


$$g(u) = \frac{1}{1+\exp(-\beta u)} = \frac{1}{1+\exp[-\beta(w1x+w0)]}$$

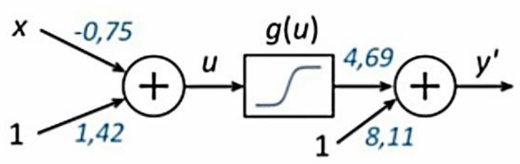
$$\beta = const \neq 0$$



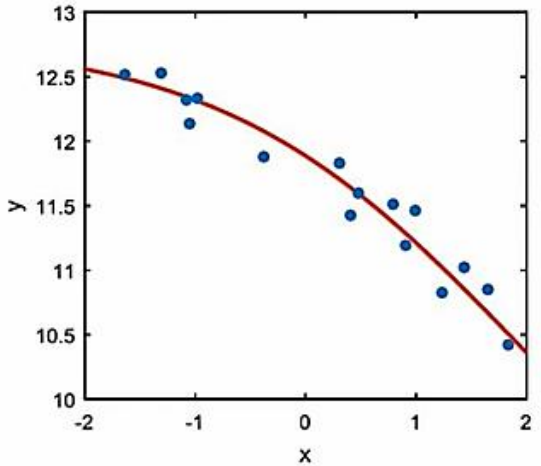
$$h(x) = g(u)v_1 + v_0 = \frac{1}{1+\exp[-\beta(w_1x+w_0)]}v_1 + v_0$$



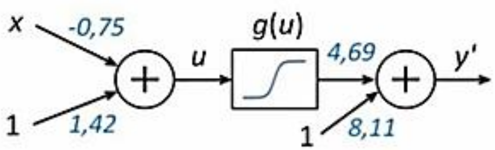
$$h(x) = \frac{1}{1+\exp[-\beta(-0{,}75x+1{,}42)]}\cdot 4{,}69 + 8{,}11$$

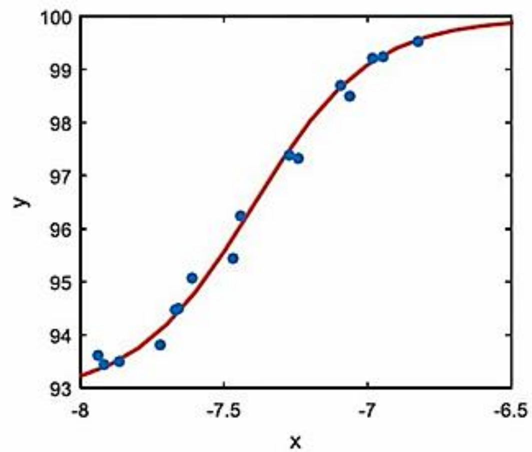


Aproksymacja dowolnych przebiegów funkcji

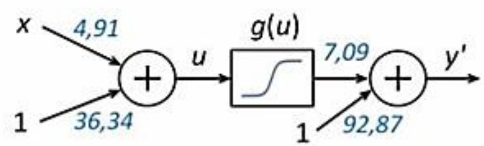


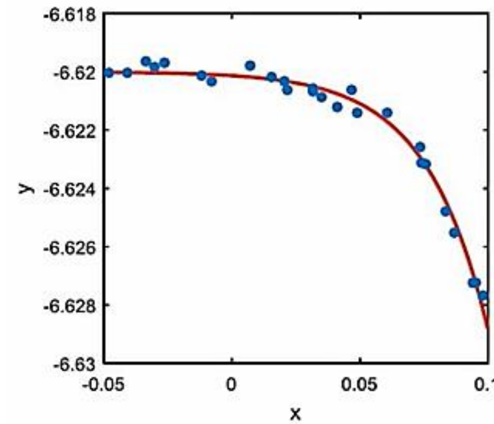
$$h(x) = \frac{1}{1 + \exp[-\beta(-0{,}75x + 1{,}42)]} \cdot 4{,}69 + 8{,}11$$



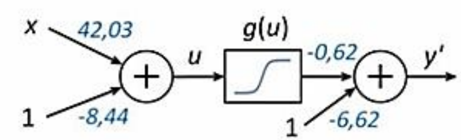


$$h(x) = \frac{1}{1 + \exp[-\beta(4{,}91x + 36{,}34)]} \cdot 7{,}09 + 92{,}87$$

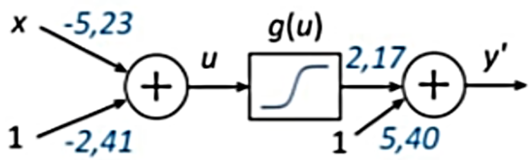


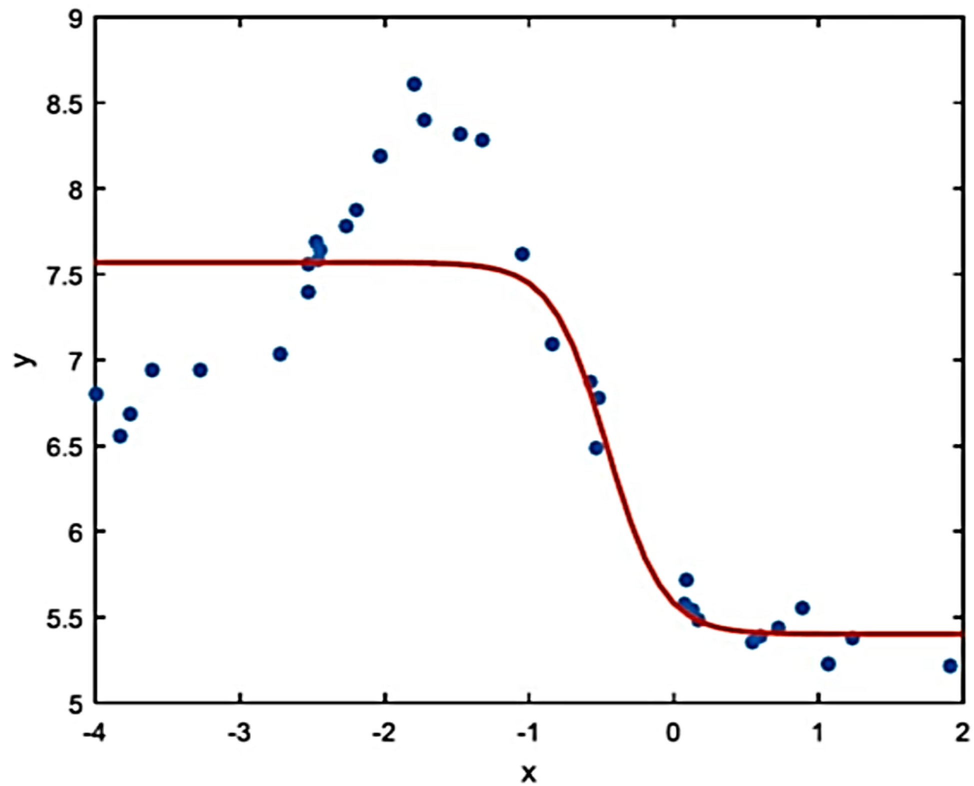


$$h(x) = \frac{1}{1 + \exp[-\beta(42{,}03x - 8{,}44)]} \cdot (-0{,}62) - 6{,}62$$



Problem aproksymacji bardziej złożonych funkcji

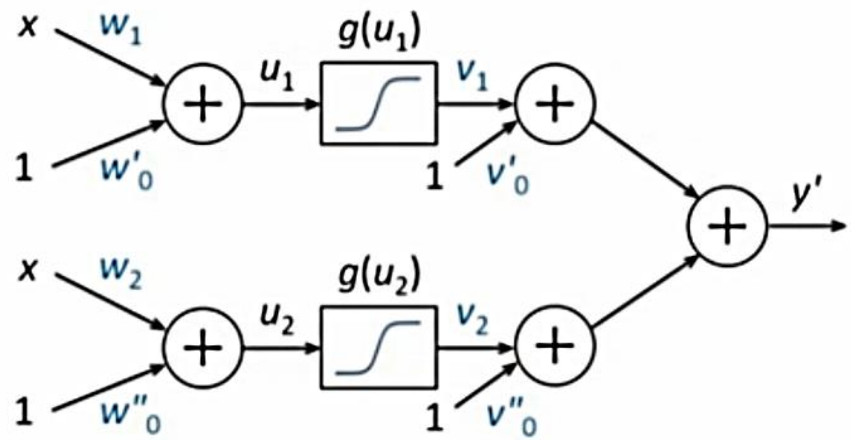




Rozwiązanie:

Dwie struktury połączone równolegle:

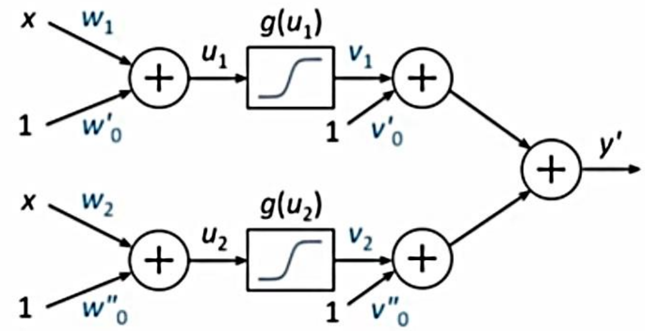
$$h(x) = [g(u_1)v_1 + v'_0] + [g(u_2)v_2 + v''_0]$$



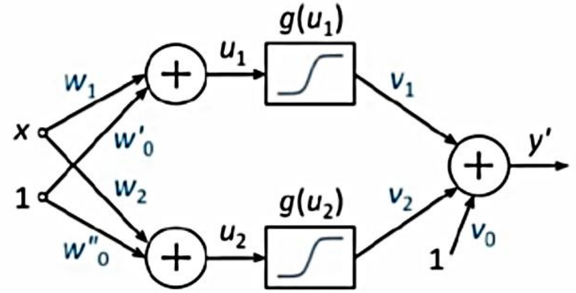
Każdy neuron będzie aproksymował różne fragmenty krzywej

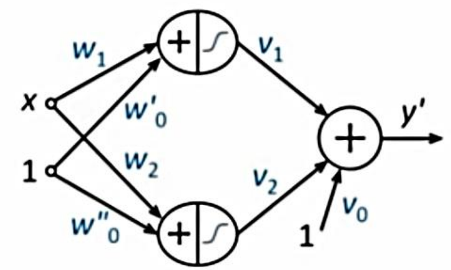
Upraszczanie struktury

$$h(x) = \left[g(u_1)v_1 + v'_0\right] + \left[g(u_2)v_2 + v''_0\right]$$



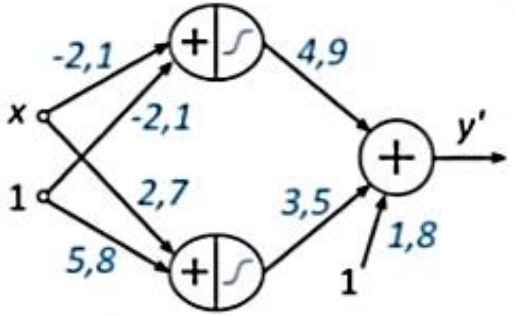
$$h(x) = g(u_1)v_1 + g(u_2)v_2 + v_0$$

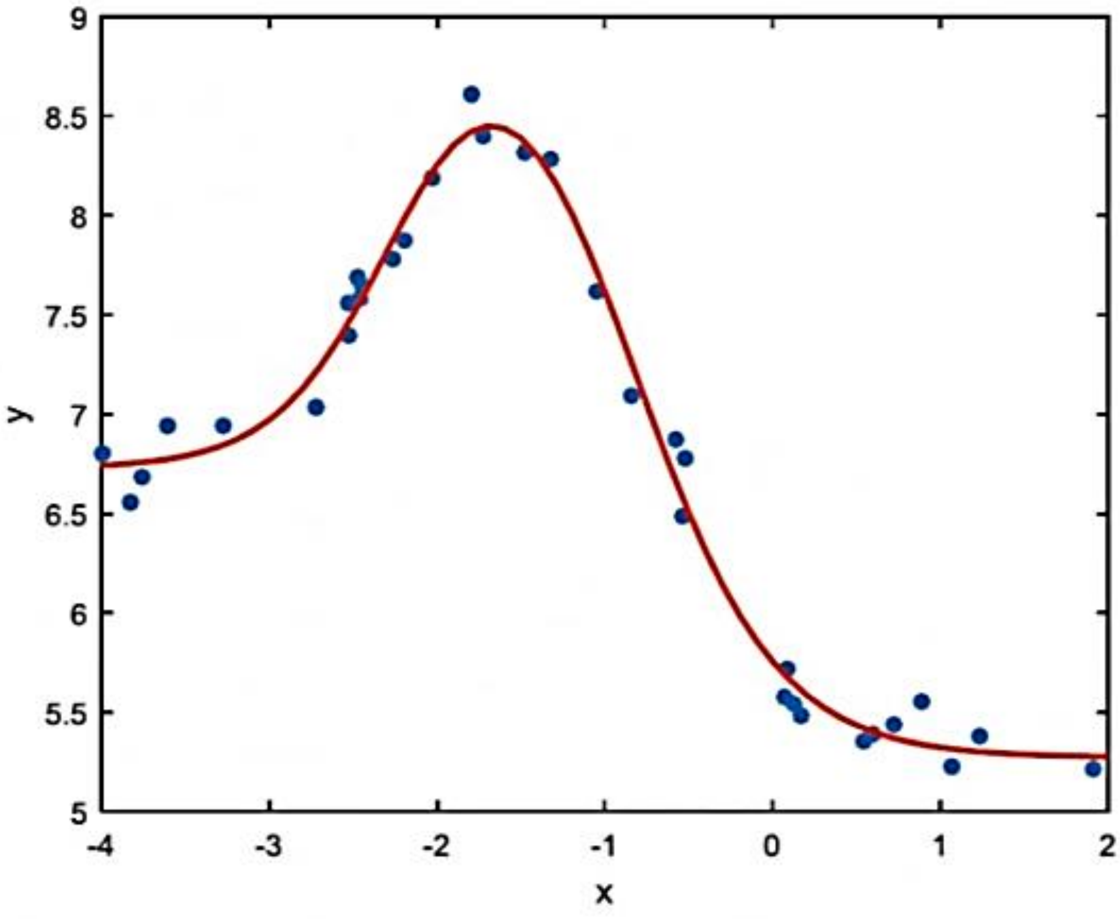


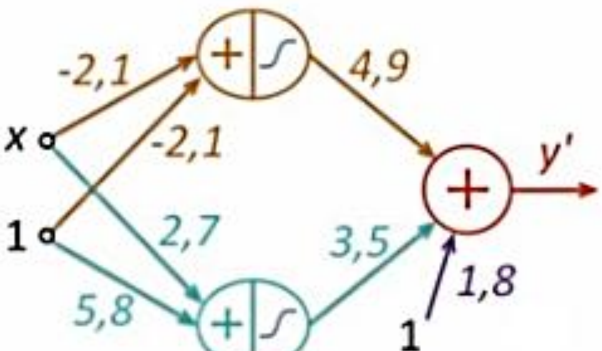


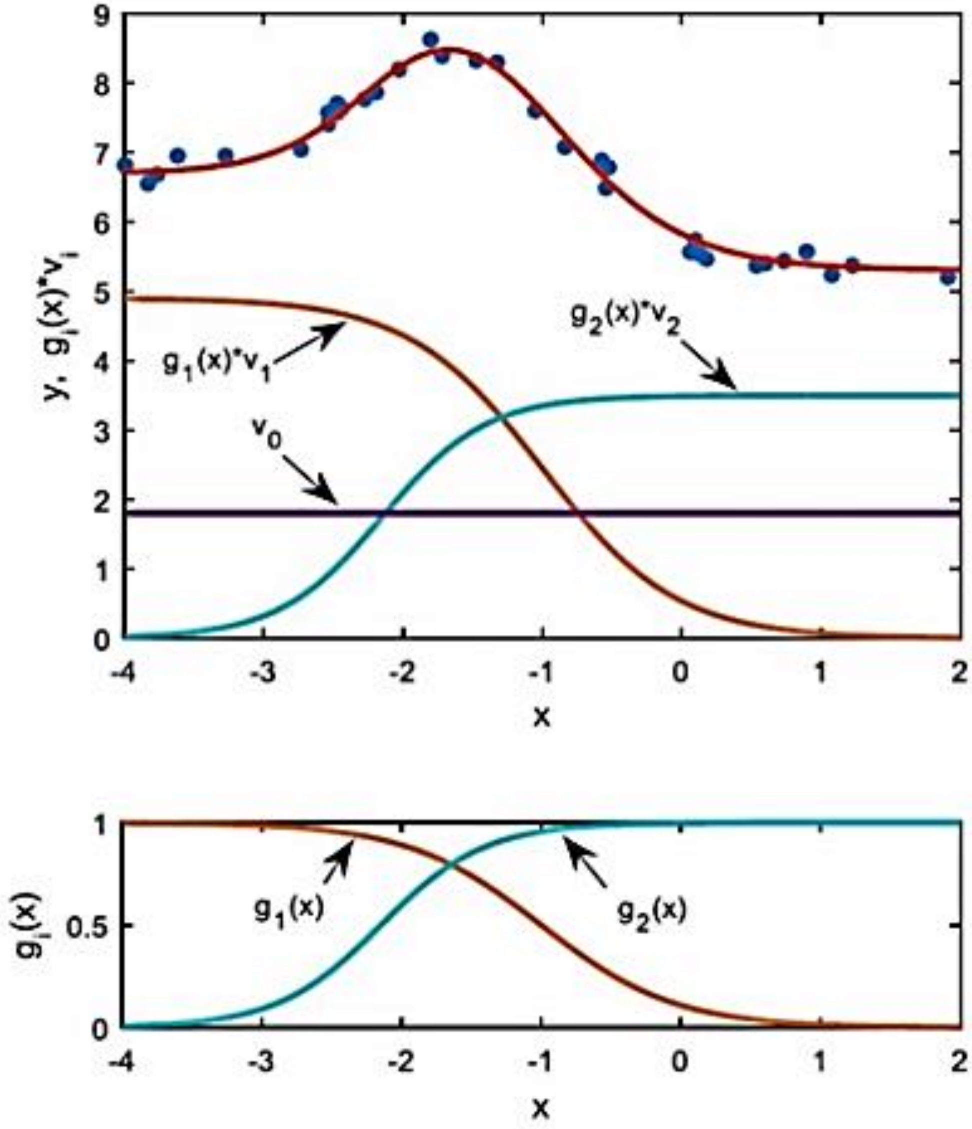
$$h(x) = g_1(x)v_1 + g_2(x)v_2 + v_0 = \frac{1}{1+\exp[-\beta(w_1 x + w'_0)]}v_1 + \frac{1}{1+\exp[-\beta(w_2 x + w''_0)]}v_2 + v_0$$

Aproksymanta utworzona przez model

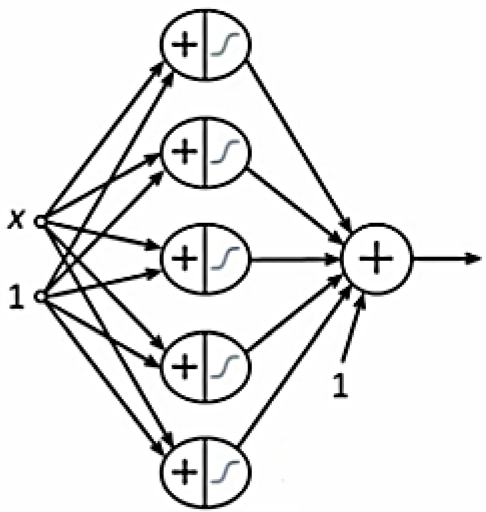


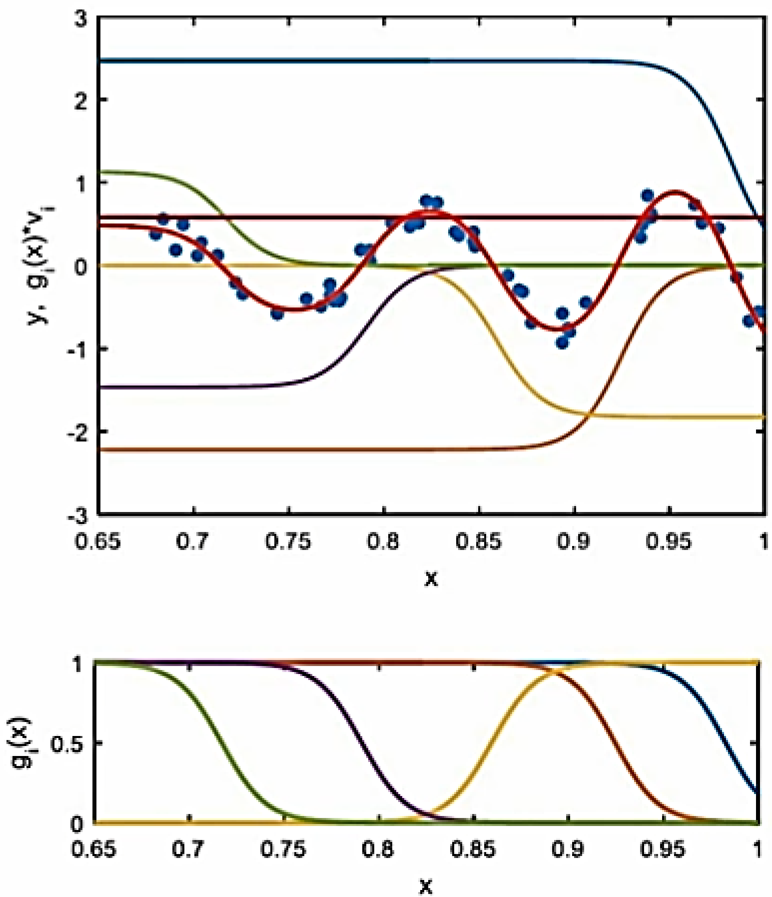



-2,1
4,9
x
-2,1
y'
1
2,7
3,5
5,8
1,8
1

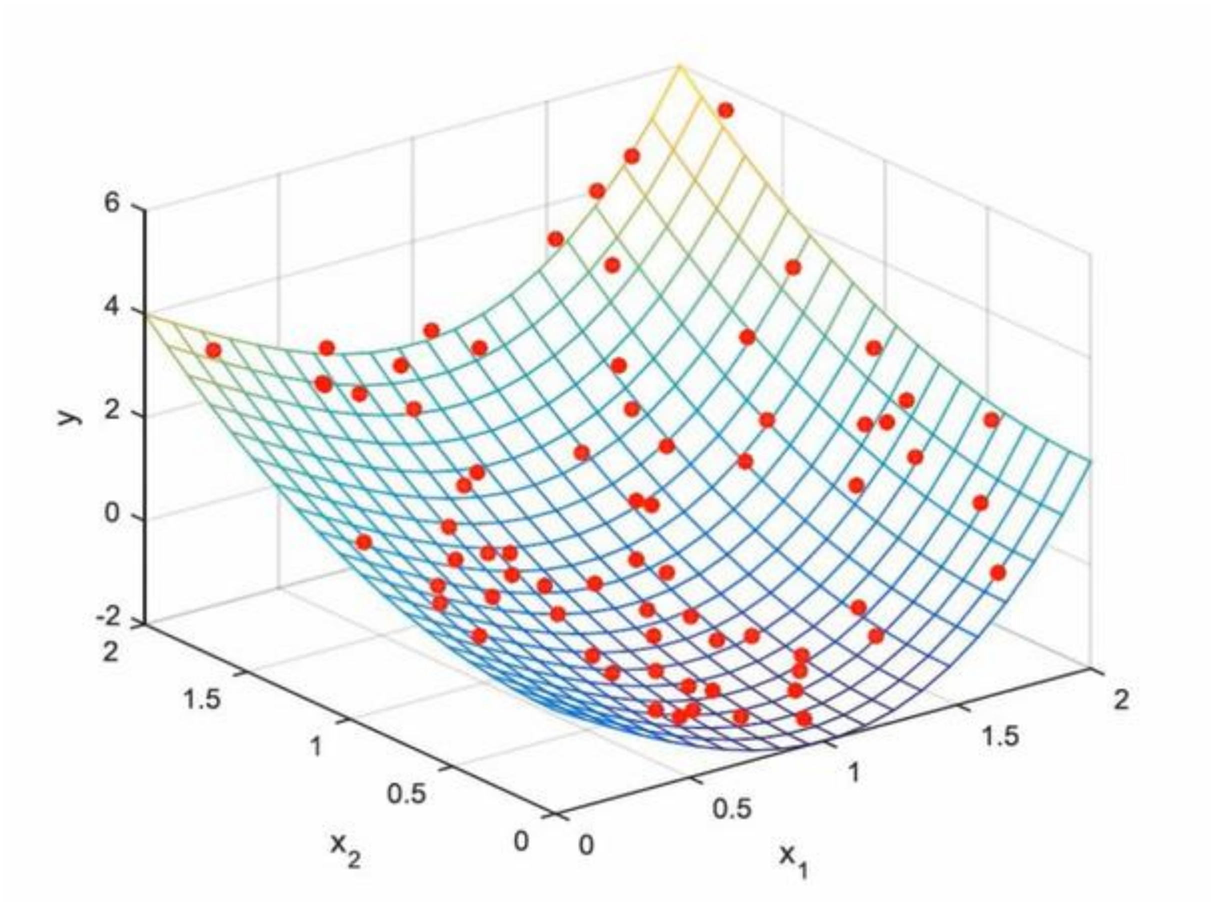

y, g_i(x)*v_i
g_1(x)*v_1
g_2(x)*v_2
v_0
x
g_i(x)
g_1(x)
g_2(x)
x

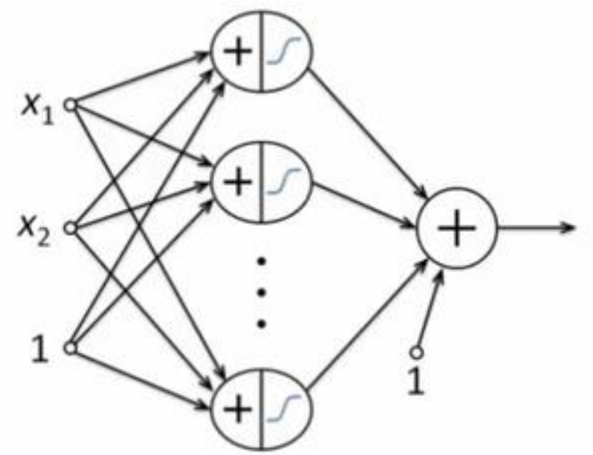
Aproksymanta bardziej złożonej funkcji za pomocą 5 neuronów



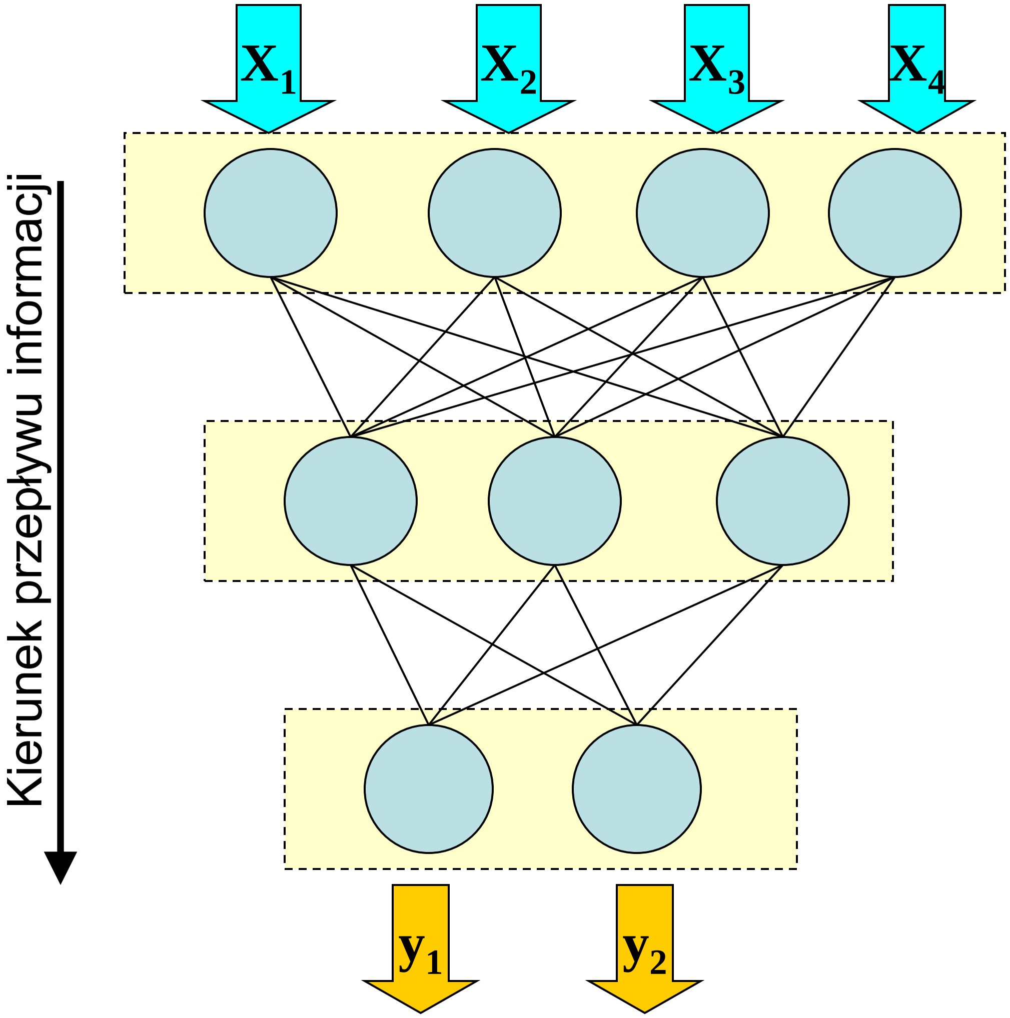


Nieliniowa funkcja dwóch zmiennych





Wielowarstwowe Sztuczne Sieci Neuronowe



Warstwa wejściowa
- Każdy neuron posiada tylko jedno wejście zewnętrzne.

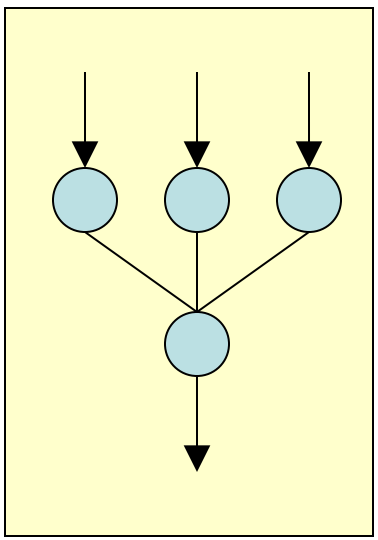
Warstwa ukryta
- Zawiera połączenia warstw

Warstwa wyjściowa
- Każdy neuron posiada jedno wyjście zewnętrzne.

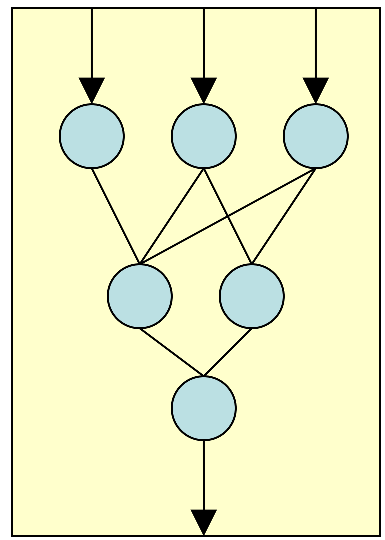
Wielowarstwowe Sztuczne Sieci Neuronowe

Liczba ukrytych warstw może wynosić:

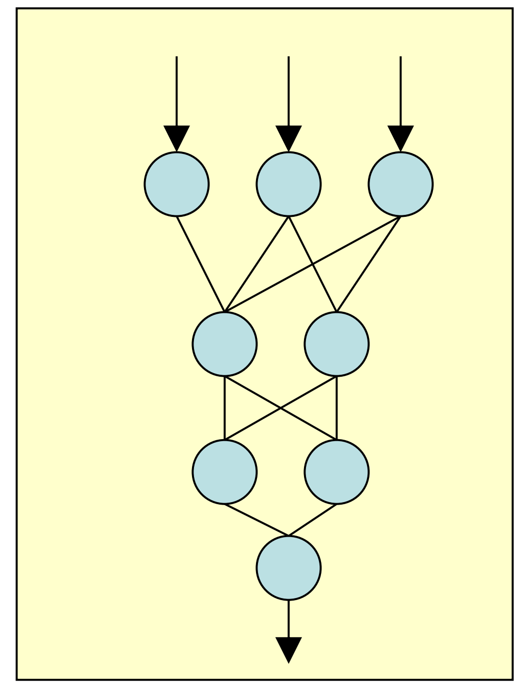
0



1

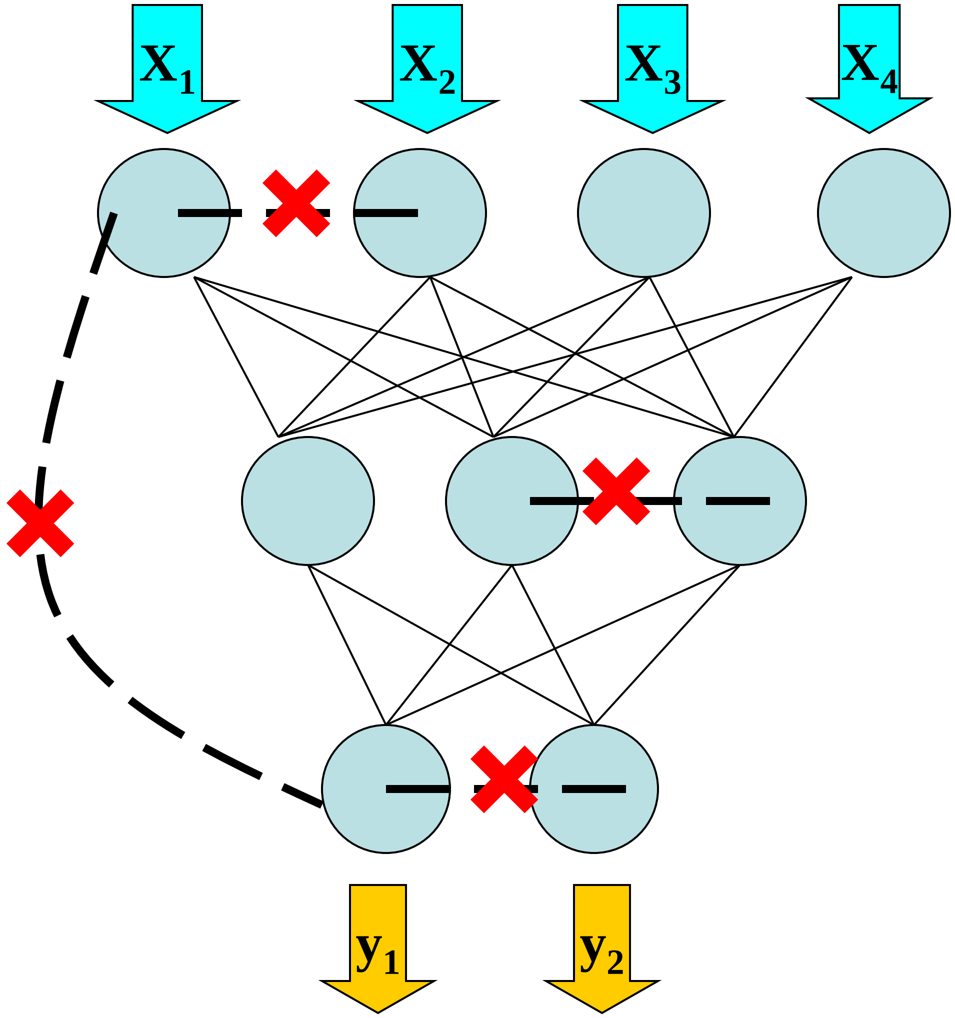


Więcej niż 1



Wielowarstwowe Sztuczne Sieci Neuronowe

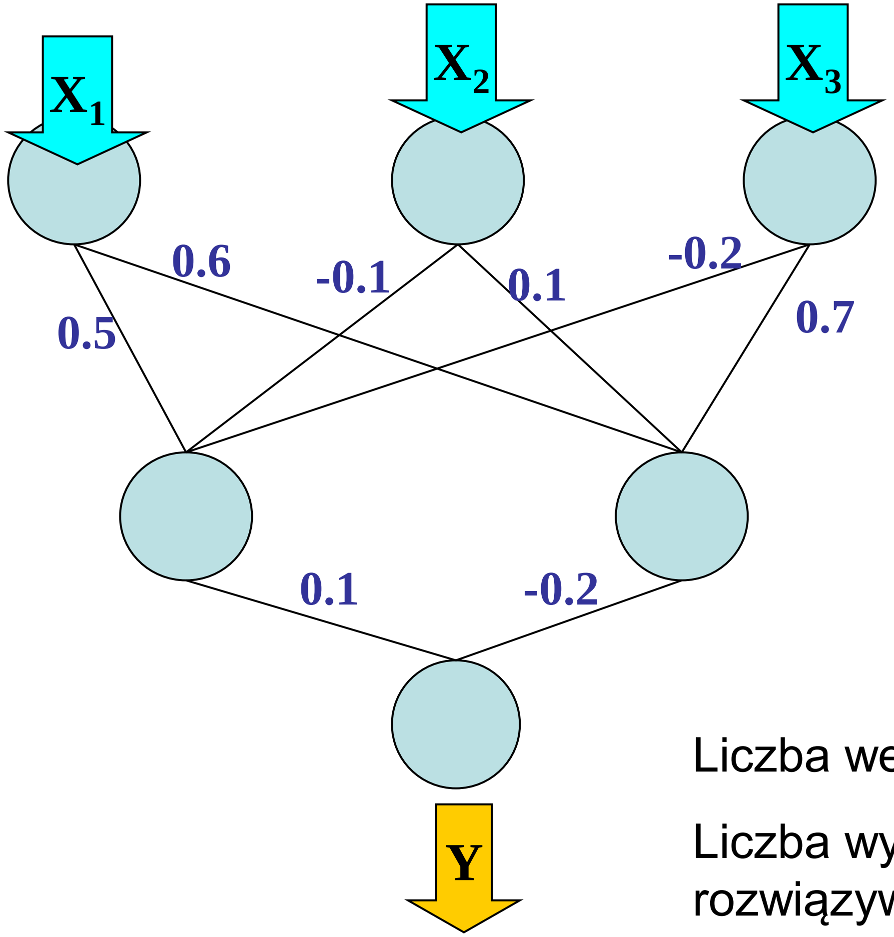
Należy zauważyć, że:



- 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:** $Y = f(X_1\ X_2\ X_3)$



Parametery

Neuronów wejściowych **3**

Warstw ukrytych **1**

Neuronów wyjściowych **1**

Liczba wejść zależy od rozmiaru wektora X

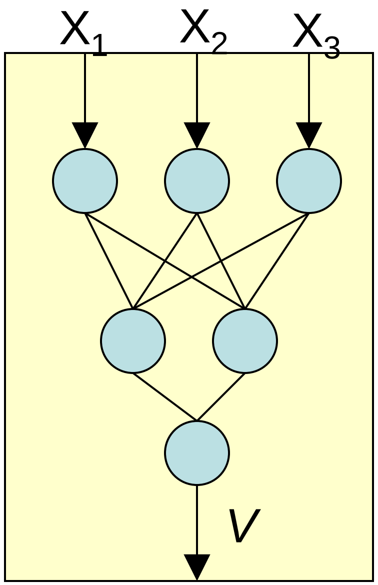
Liczba wyjść zależy od rodzaju rozwiązywanego problemu

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**



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

8 wartości wag do ustalenia.

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

Dane ćwiczeniowe: $(Y_i\ , X_{1i}, X_{2i}, \ \ldots, \ X_{pi}\ )$ i= 1,2,…,n

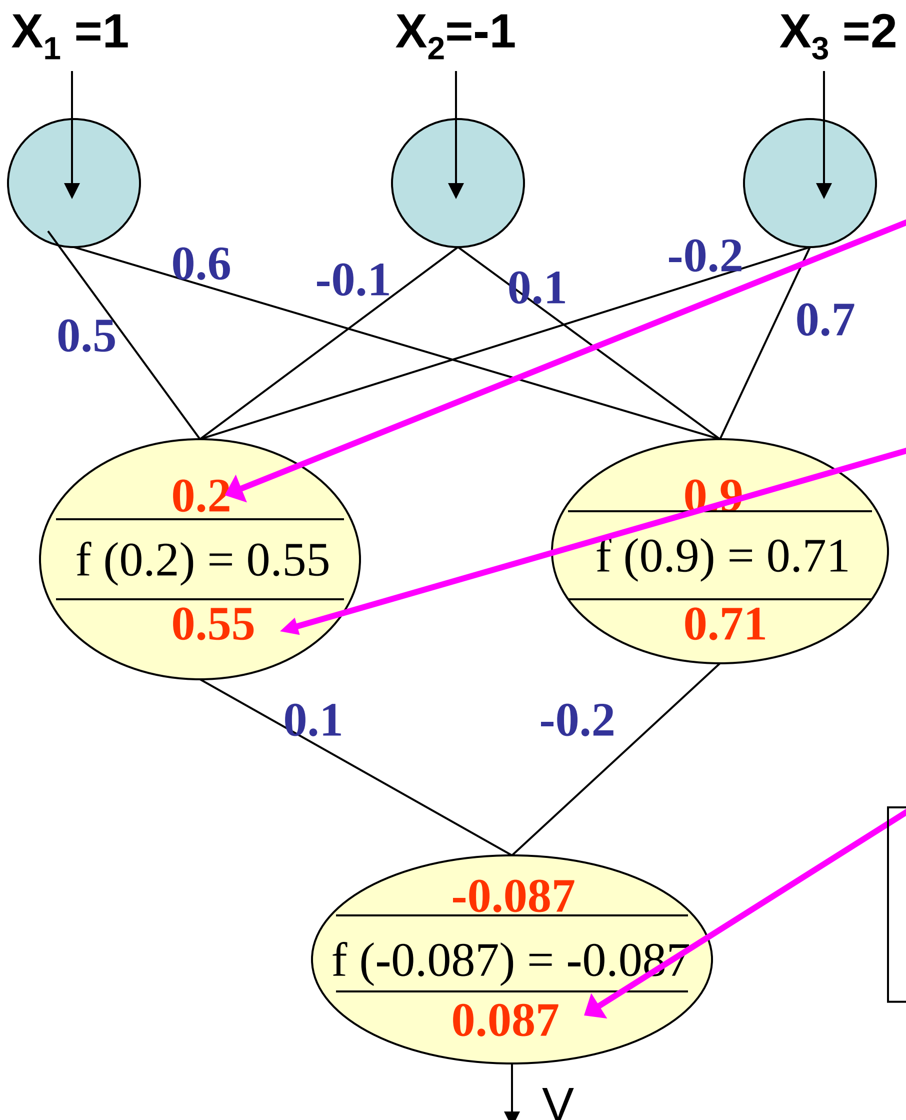
Określając wartości $\underline{\boldsymbol{W}}$, otrzymujemy wartości wyjściowe $(\ V_1, V_2, \ldots, V_n)$

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

Definiujemy całkowity błąd predykcji: $E = \sum_i (Y_i - V_i)^2$

Przykład użycia SSN do predykcji sygnału.

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



$0.2 = 0.5 * 1 – 0.1*(-1) – 0.2 * 2$

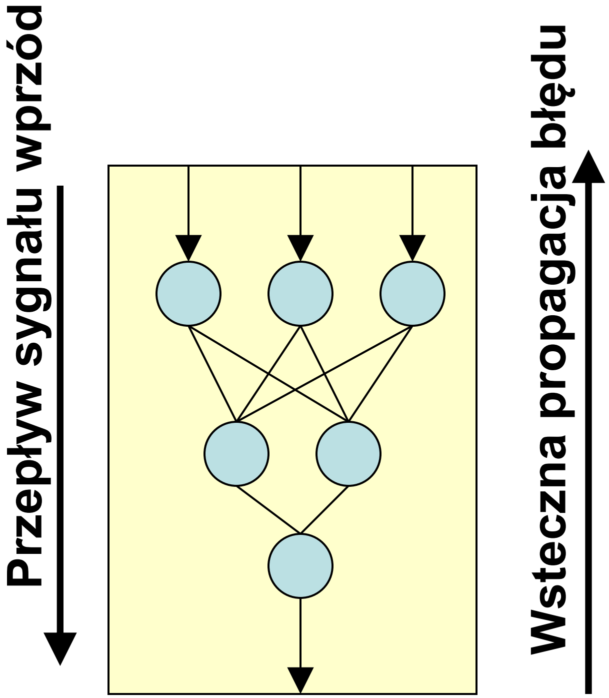
$f(x) = 1 / (1 + e^{-x})$

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

Przewidywane V = 0.478

Spodziewana wartość Y = 2
zatem
Błąd predykcji = (2-0.078) = 1.992

Ćwiczenie modelu SSN



$$e = \sum_i (Y_i - V_i)^2$$

Jak uczyć SSN ?

- Ustalenie losowych wartości wag
- Obliczenie przepływu sygnału odbywa się „w przód”

  X → Sieć **V** → **„Błąd i-tego neuronu”** **= (Y_i – V_i)**

- 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 $\underline{W} = (W_1, W_2, \ldots)$

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

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

Metoda Największego Spadku :

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

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

α = współczynnik uczenia [0;1]

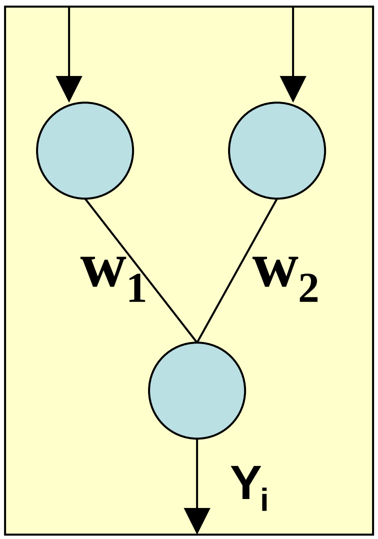
Niekiedy do zmiany wag stosowany jest wzór:

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

β - Współczynnik „Momentum” [0;1]

Geometryczna interpretacja doboru wag

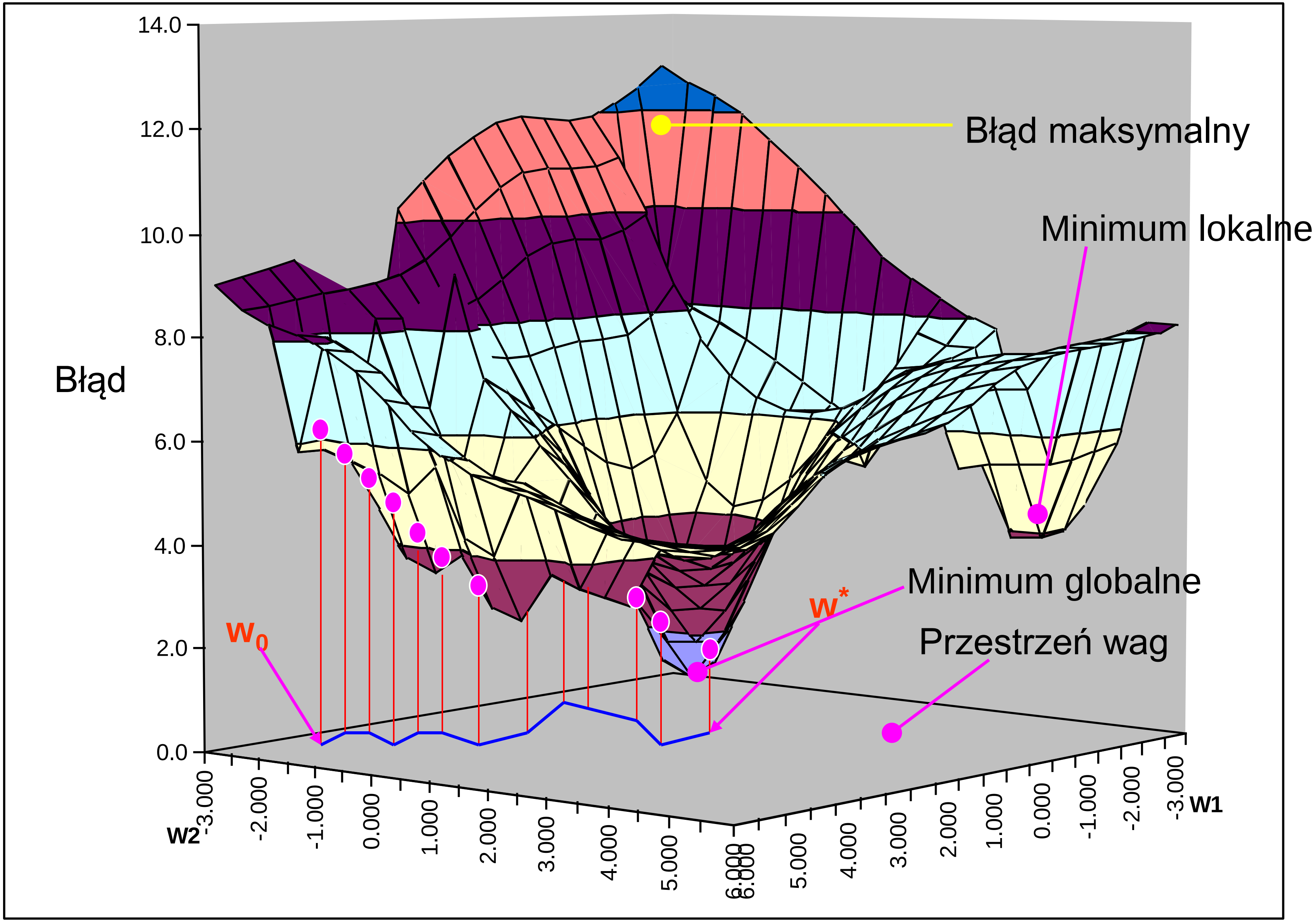
Załóżmy, że mamy SSN złożoną z dwóch wag (w_1 i w_2)



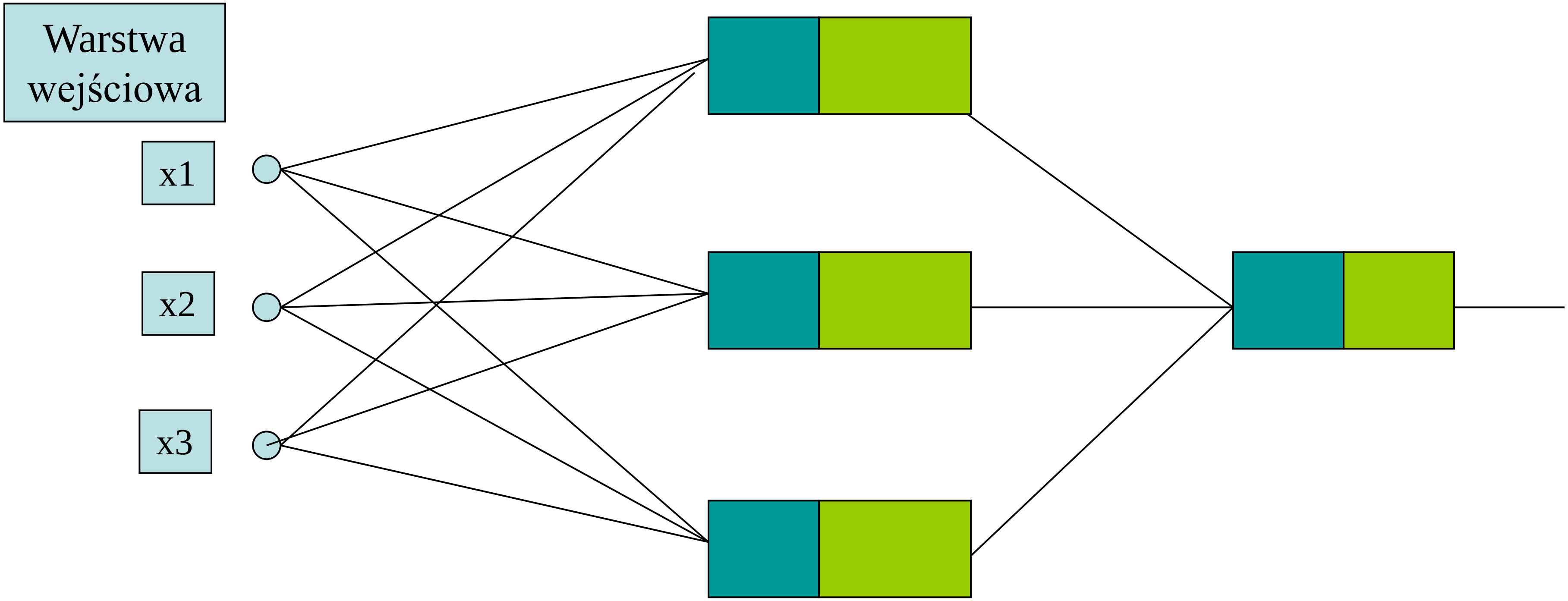
V_i – wartość oczekiwana

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

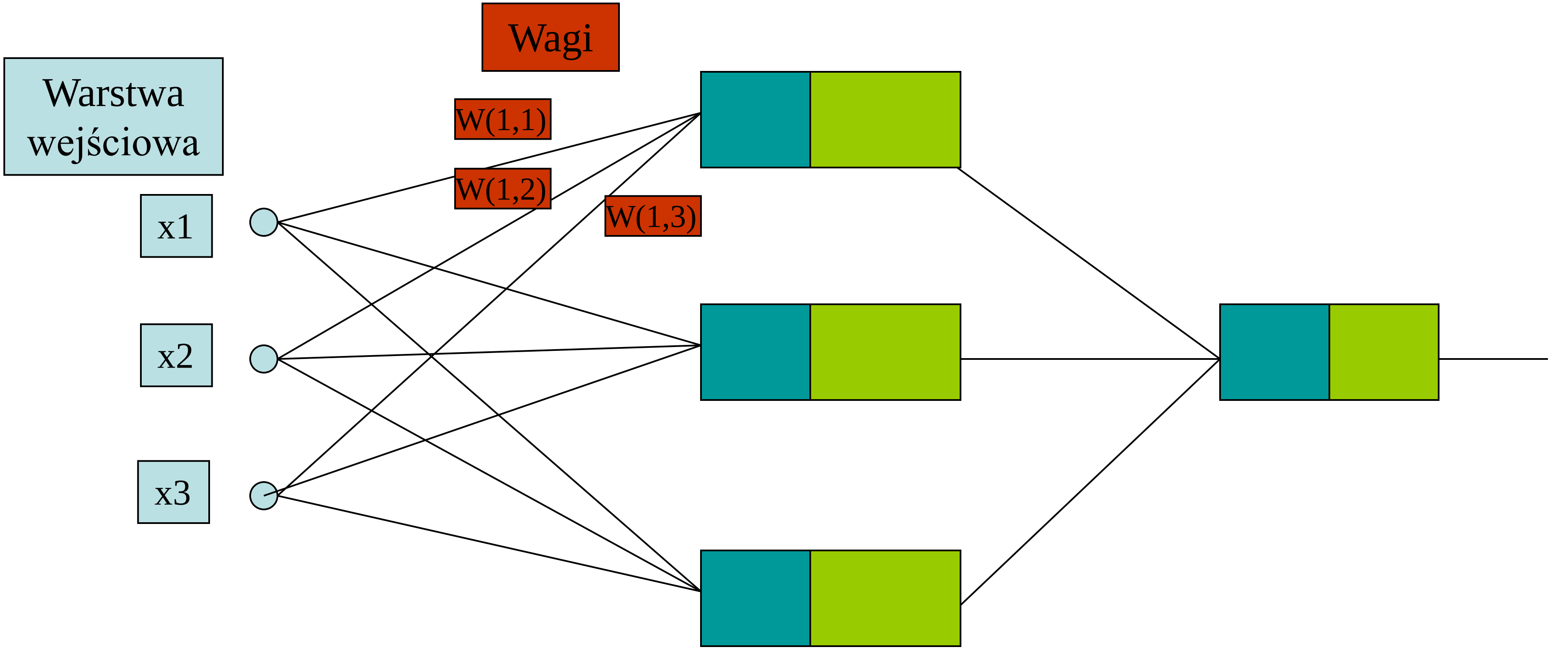
Geometryczna interpretacja doboru wag



Algorytm Wstecznej Propagacji Błędu



Na warstwę wejściową podawane są sygnały wejściowe (x1..xn). 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.

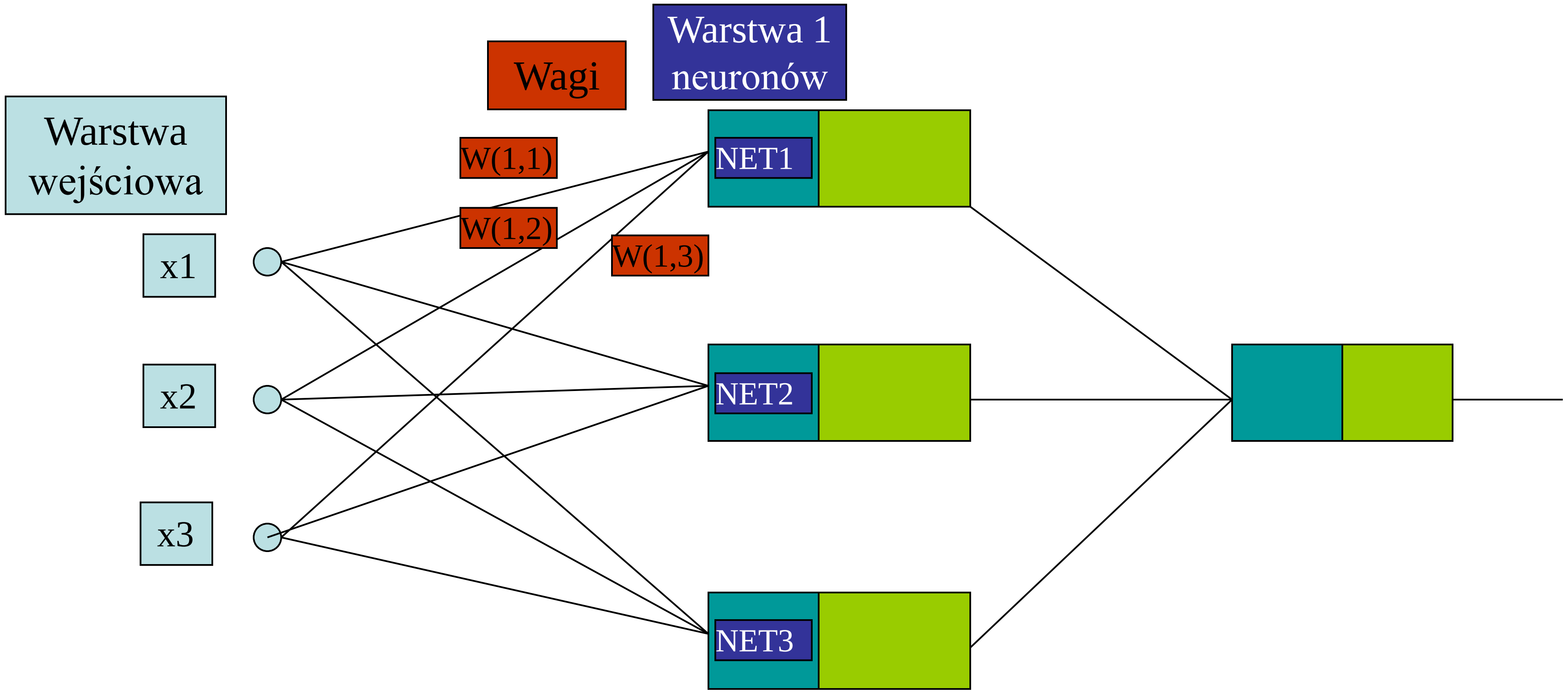


Sygnały wejściowe mnożone są odpowiednio przez wagi każdego z neuronów.

x1*w(1,1), x2*w(1,2), x3*w(1,3)

x1*w(2,1), x2*w(2,2), x3*w(2,3)

x1*w(3,1), x2*w(3,2), x3*w(3,3)



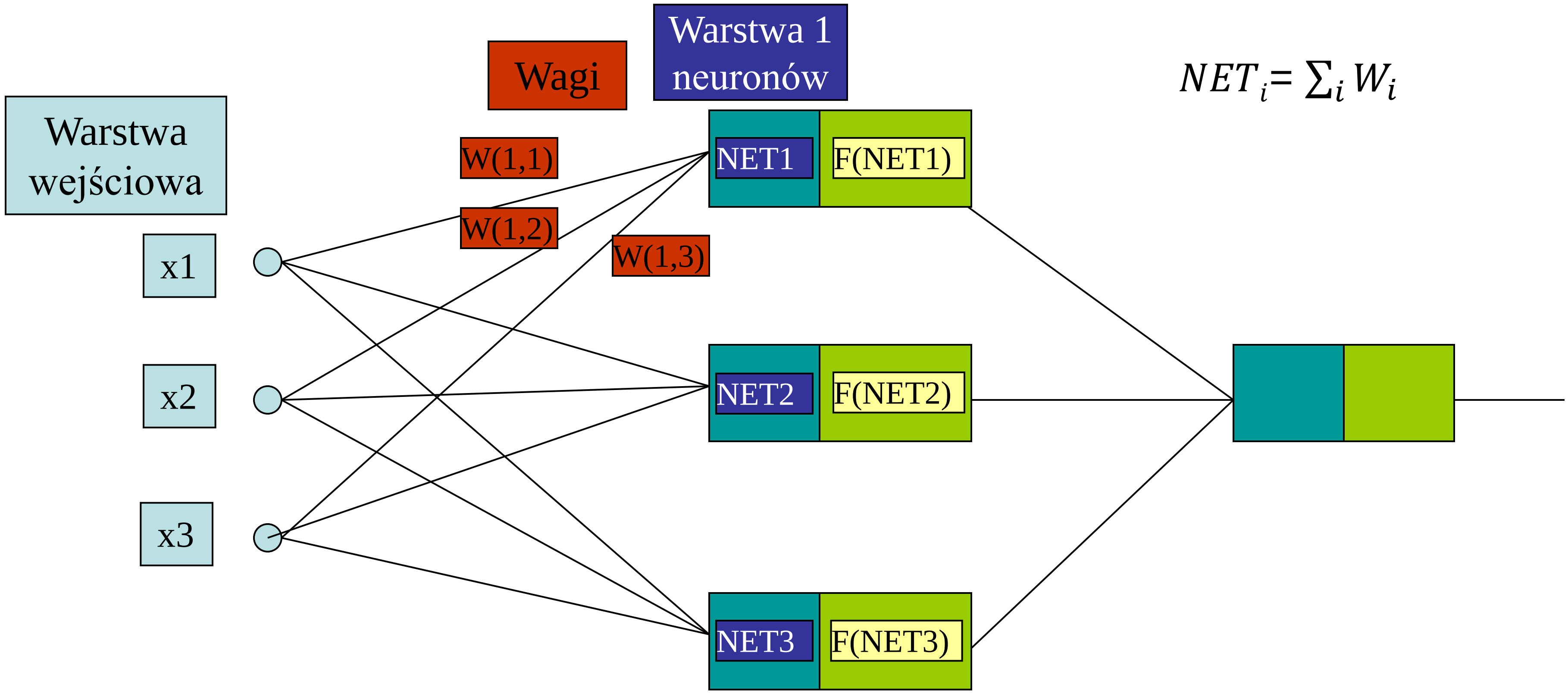
Wartości uzyskane z przemnożenia sygnałów wejściowych oraz odpowiednich wag są sumowane w każdym neuronie dając odpowiednie wartości potencjału neuronu (NET):

NET1=x1*w(1,1)+x2*w(1,2)+x3*w(1,3)

NET2=x1*w(2,1)+x2*w(2,2)+x3*w(2,3)

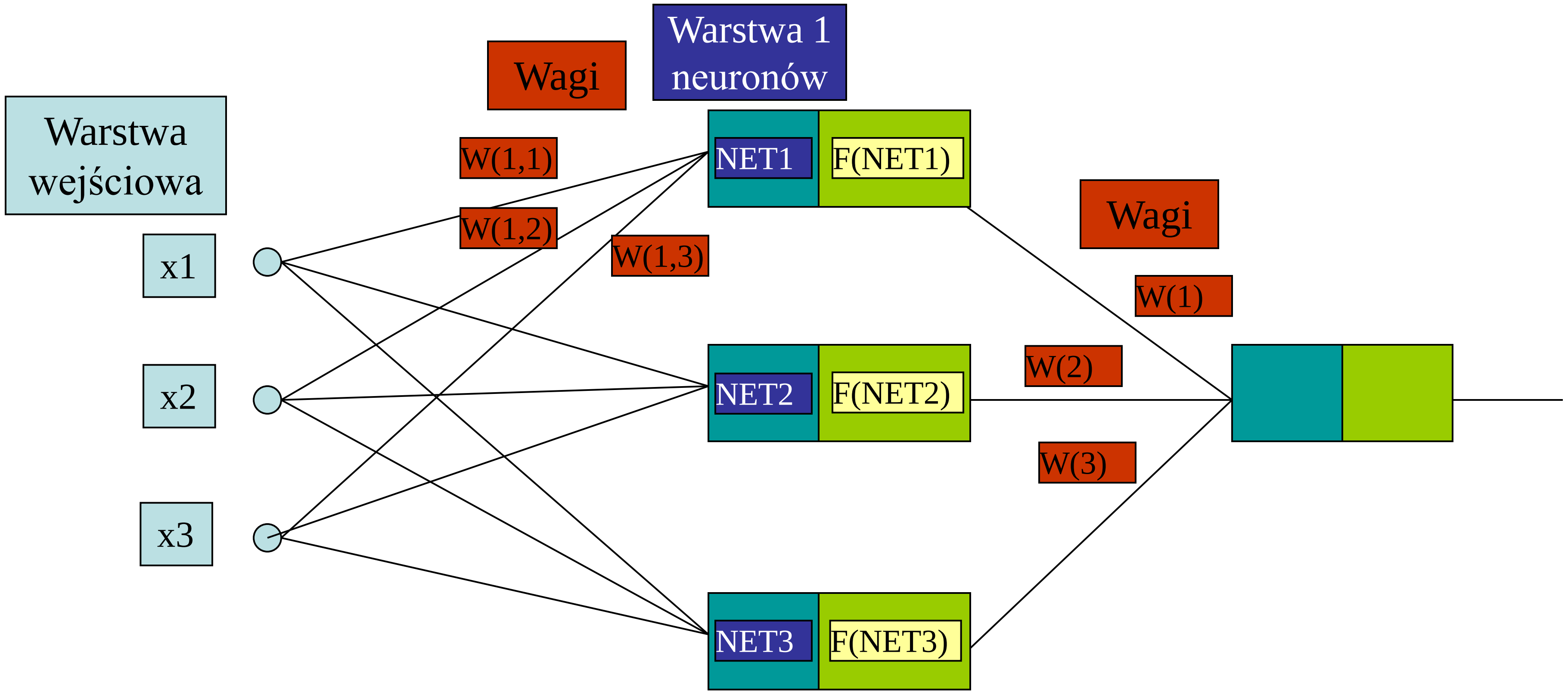
NET3=x1*w(3,1)+x2*w(3,2)+x3*w(3,3)

$NET_i = \sum_i W_i$



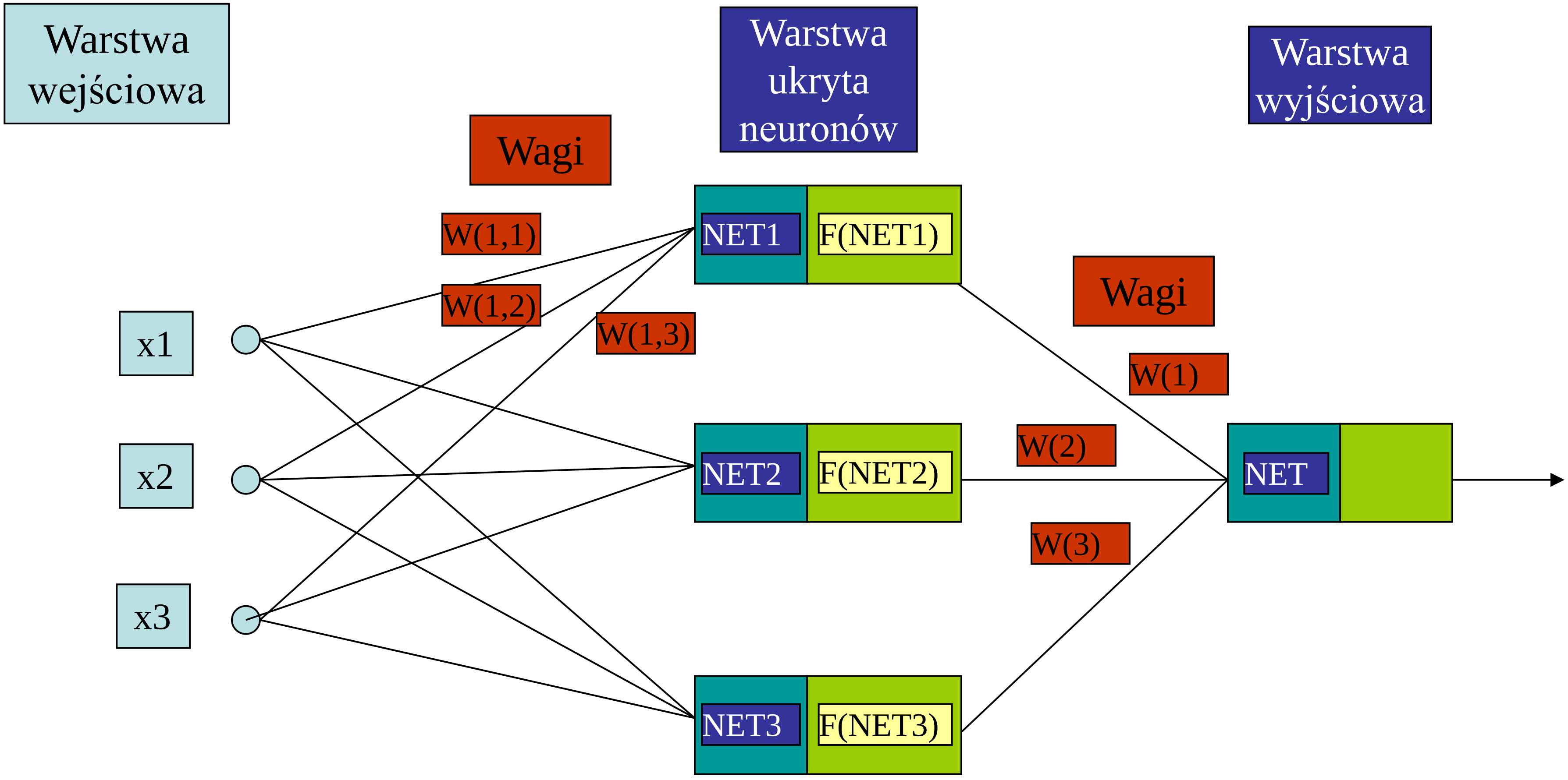
Sygnał potencjału przekazywany jest jako argument do funkcji aktywacji neuronu f(NET). Jest ona nieliniowa oraz koniecznie różniczkowalna. Najczęściej spotykaną funkcją jest tzw. funkcja sigmoidalna postaci:

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



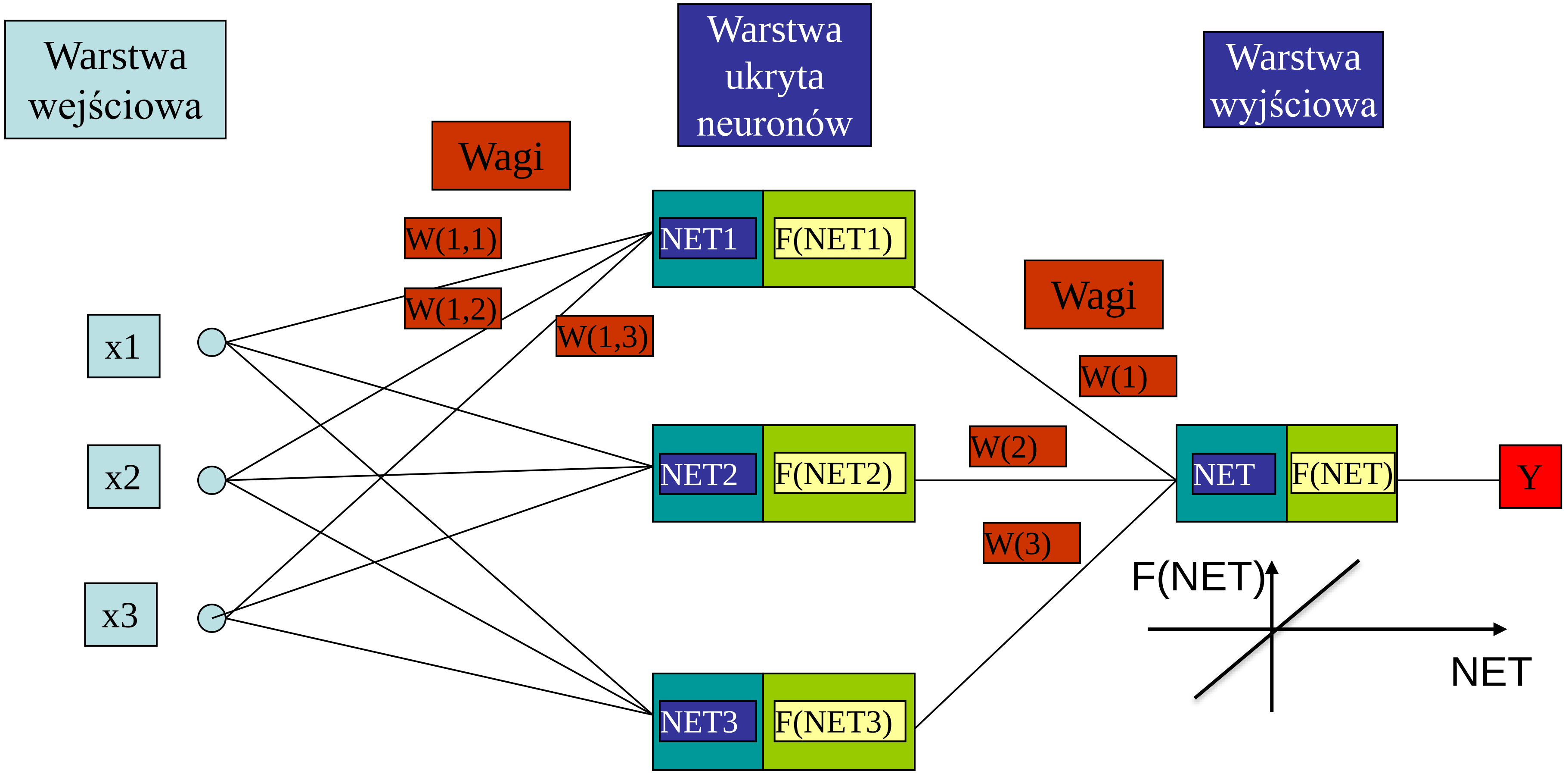
Otrzymane wartości ze wszystkich neuronów zostają przemnożenie przez odpowiednie wagi neuronu warstwy wyjściowej:

F(NET1)*w(1), F(NET2)*w(2), F(NET3)*w(3)



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

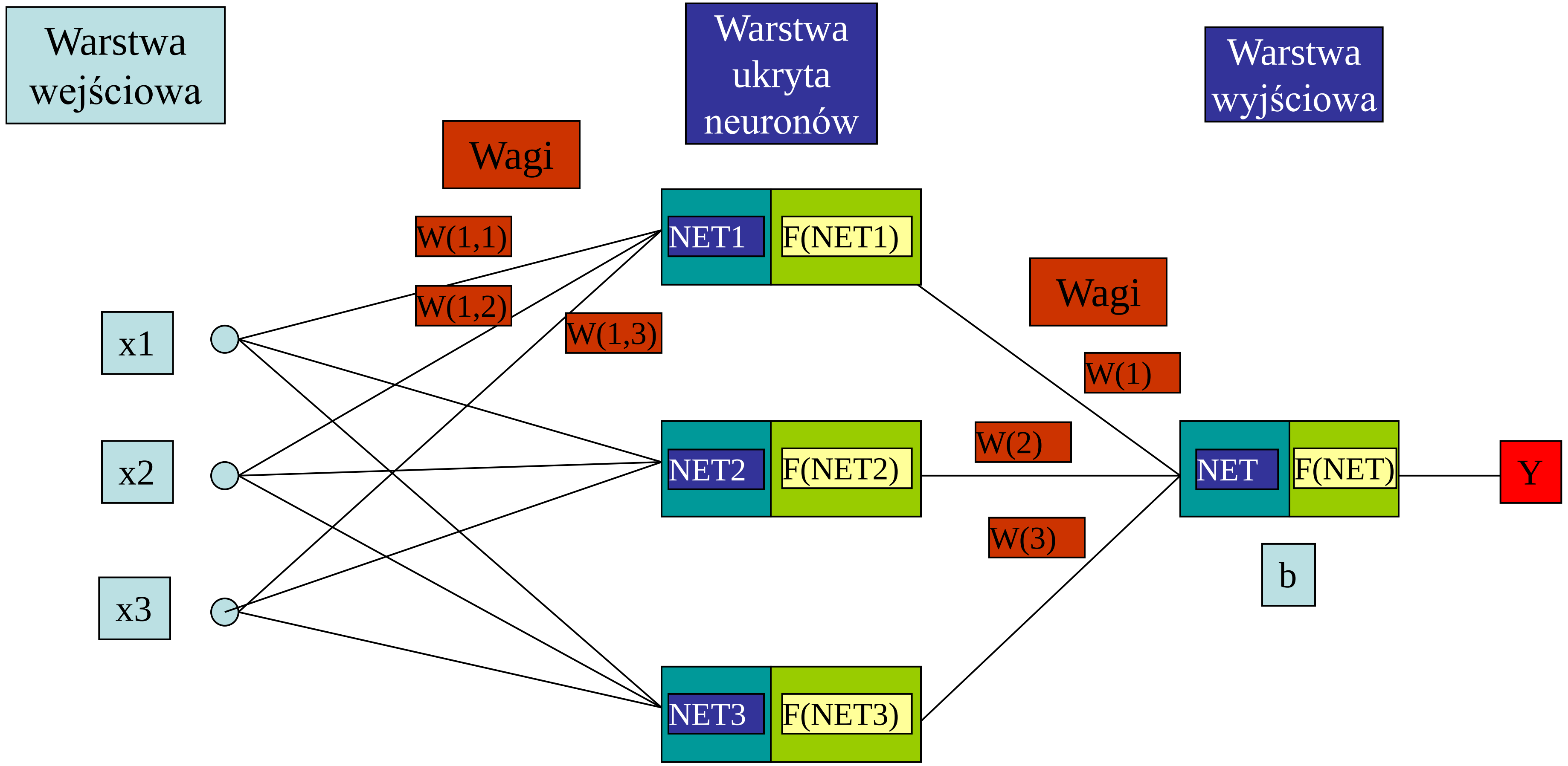
NET = f(NET1)*w(1)+f(NET2)*w(2)+f(NET3)*w(3)



Sygnał NET zostaje podany jako argument liniowej funkcji f(NET) postaci:

$$y = 1*\mathrm{NET}$$

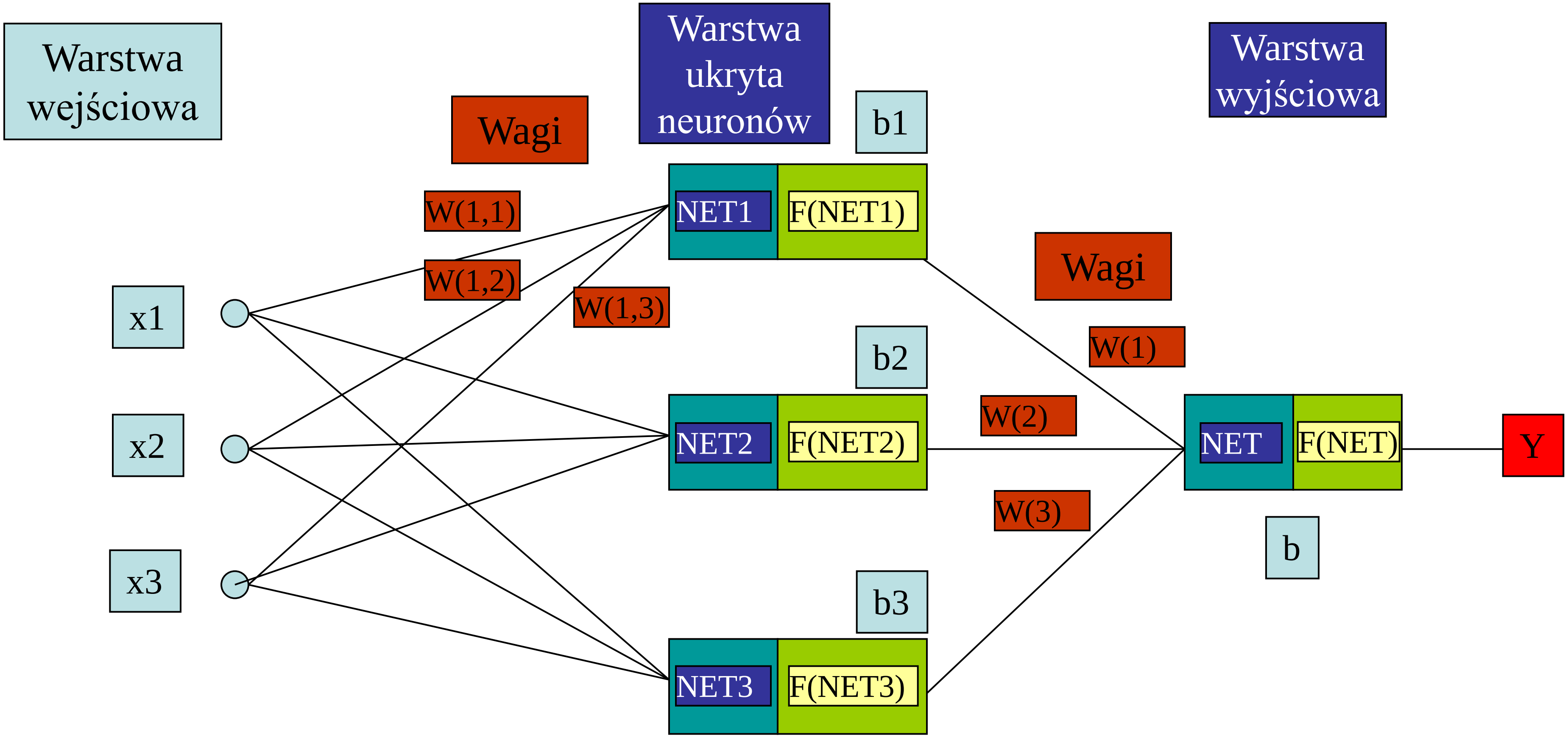
Dając w wyniku wartość wyjściową sieci (Y).



Wyznaczany jest błąd sieci wynikający z porównania odpowiedzi sieci (Y) z odpowiedzią wzorca (Z):

$$b = f'(NET)(Y - Z) \longrightarrow b = 1 \cdot (Y - Z)$$

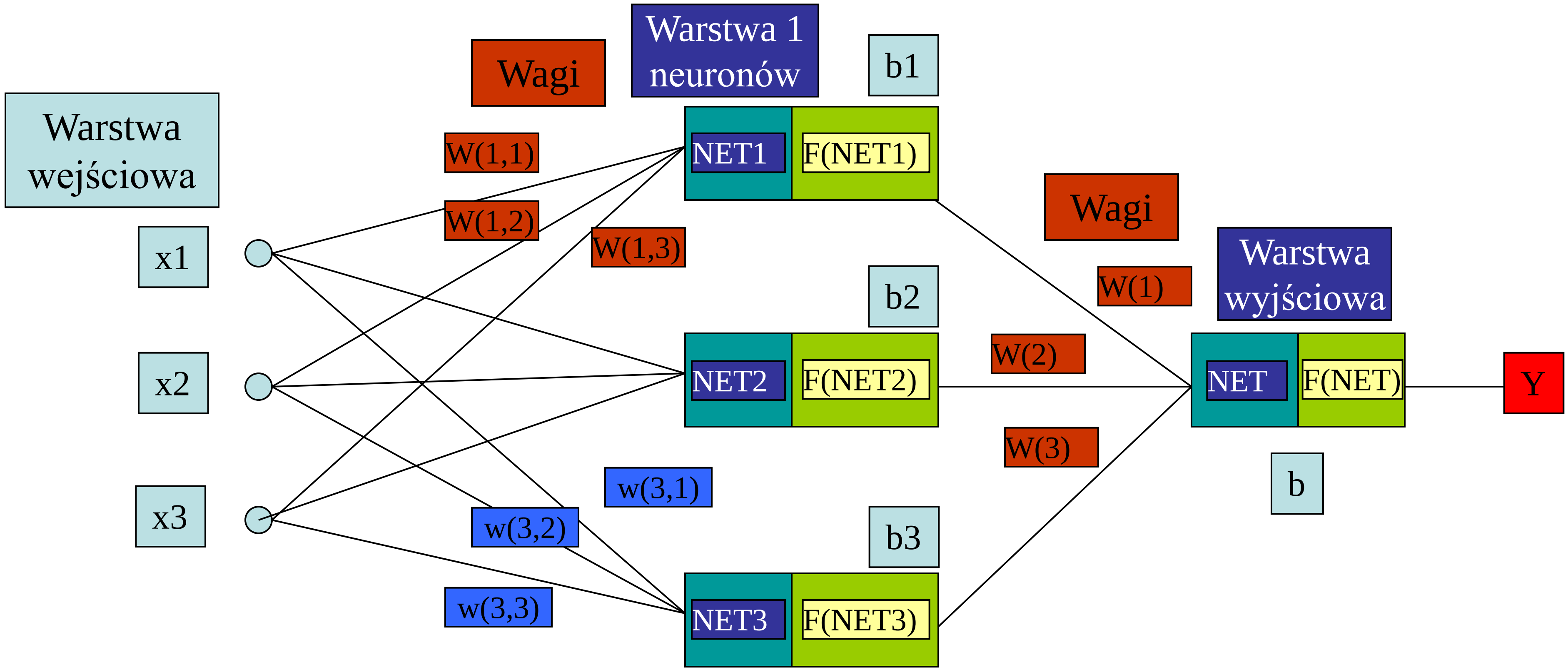
Będący błędem neuronu warstwy wyjściowej



Błąd neuronów warstwy ukrytej wyznaczany jest za pomocą wzoru:

$$b_i = f'(NET_i) \cdot b \cdot w_i$$

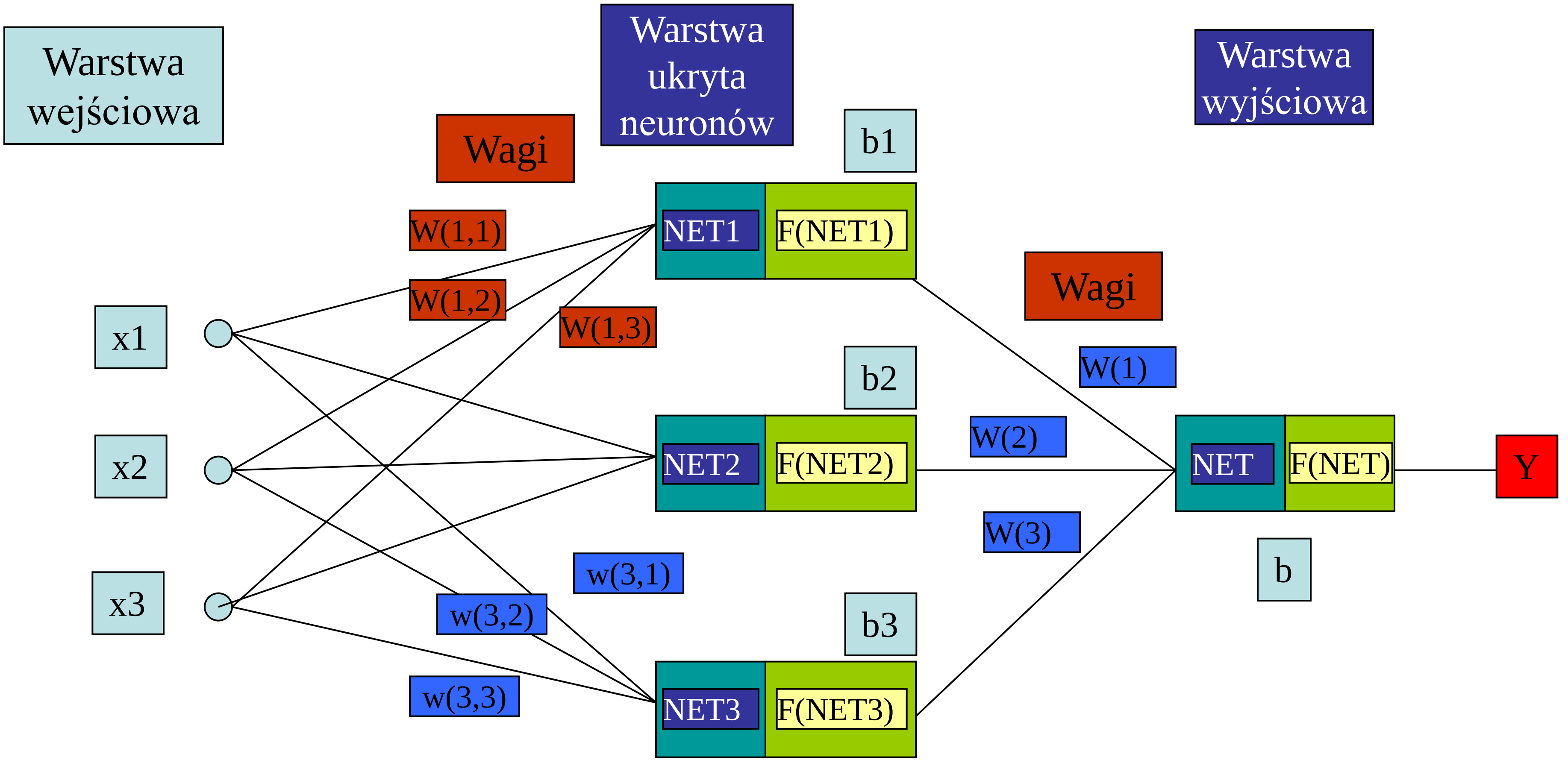
Co oznacza, że błąd neuronu warstwy ukrytej jest proporcjonalny do błędu neuronu z którym jest „mocno połączony”. Im większy jest błąd jego następcy, tym większy powinien być jego błąd. Wyznaczanie błędów odbywa się w kierunku odwrotnym do zwykłej propagacji sygnału i stąd wzięła się nazwa algorytmu.



Aktualizacja wag zachodzi w kierunku przewodzenia sygnałów według wzoru:

$$w_i = w_i^{stare} + \eta b_i x_i$$

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

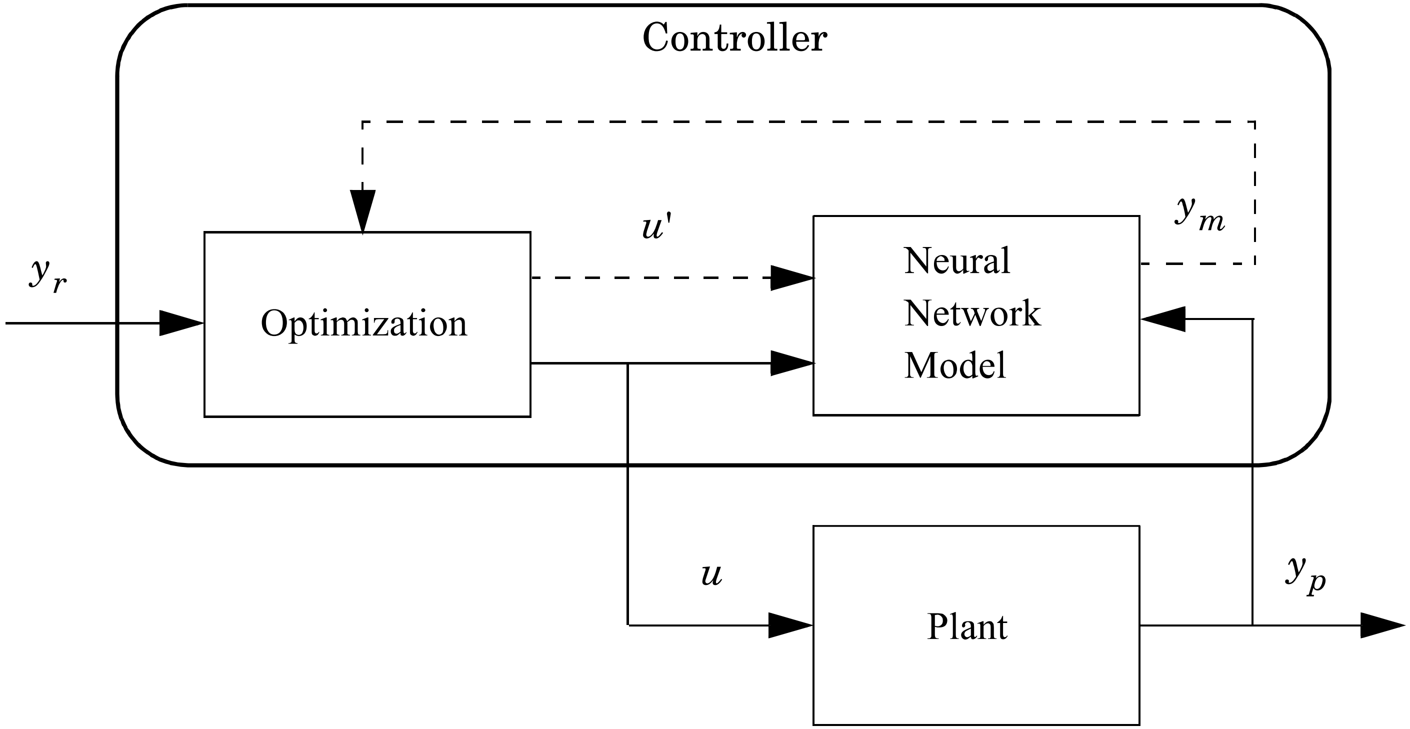


Aktualizacja wag w warstwie wejściowej następuje analogicznie według wzoru:

$$w_{i,j} = w_{i,j}^{stare} + \eta b_i f(NET_i)$$

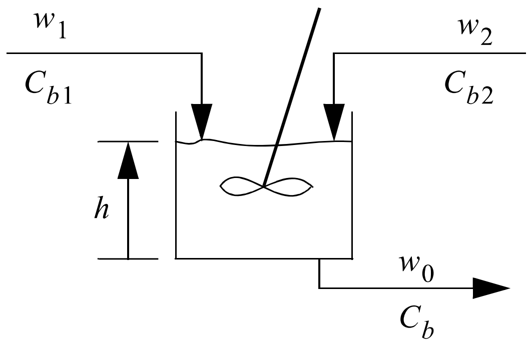
W wyniku algorytmu, nowe wagi powinny być lepiej przystosowane do pokazanego wzorca wejściowego, co oznacza, że przy kolejnym pokazie, wygenerowany błąd będzie mniejszy.

Zastosowanie Sieci typu FF.



$$J = \sum_{j=N_1}^{N_2} (y_r(t+j) - y_m(t+j))^2 + \rho \sum_{j=1}^{N_u} (u'(t+j-1) - u'(t+j-2))^2$$

Model matematyczny urządzenia do utrzymywania stałej koncentracji roztworu.

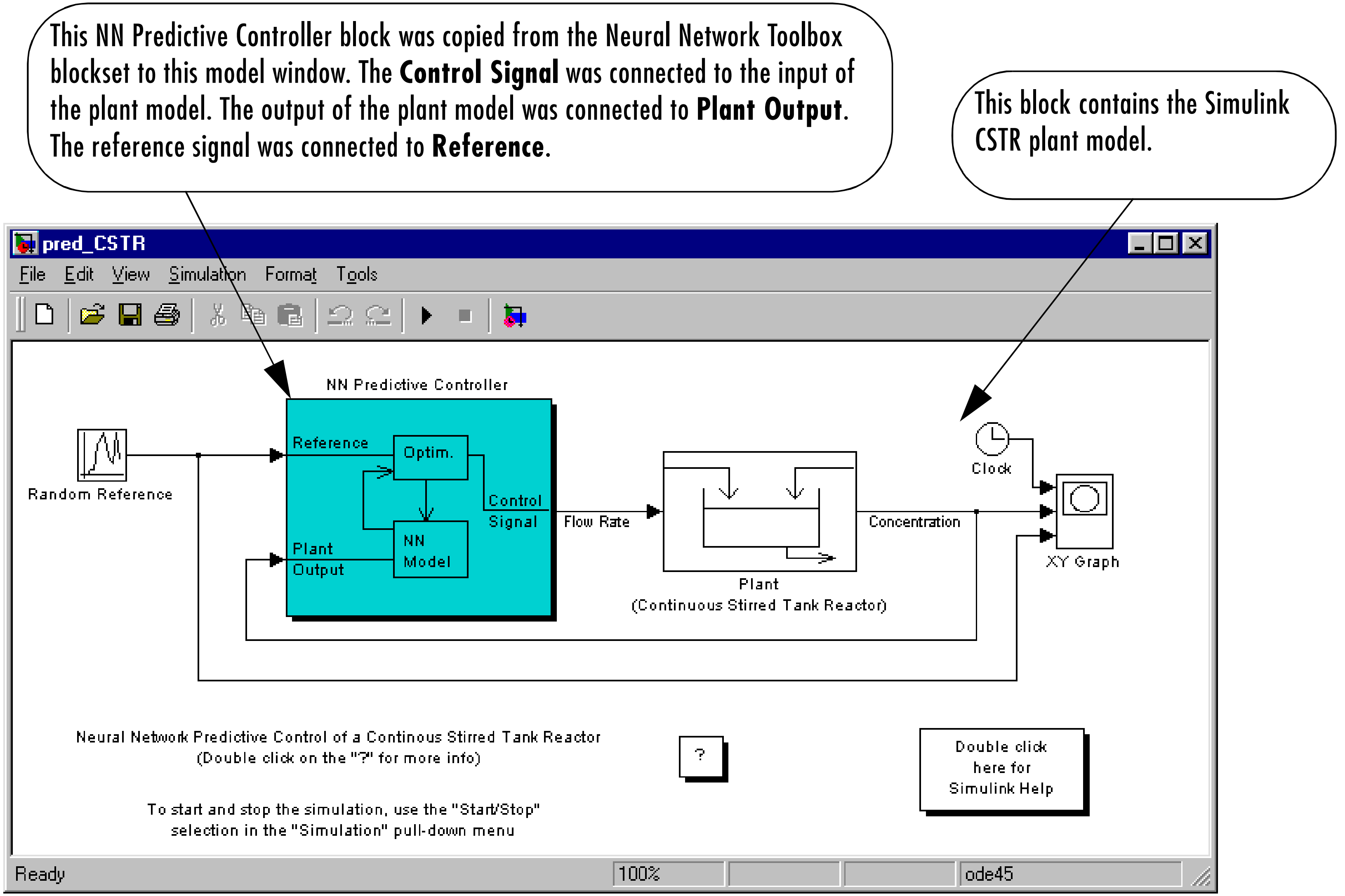


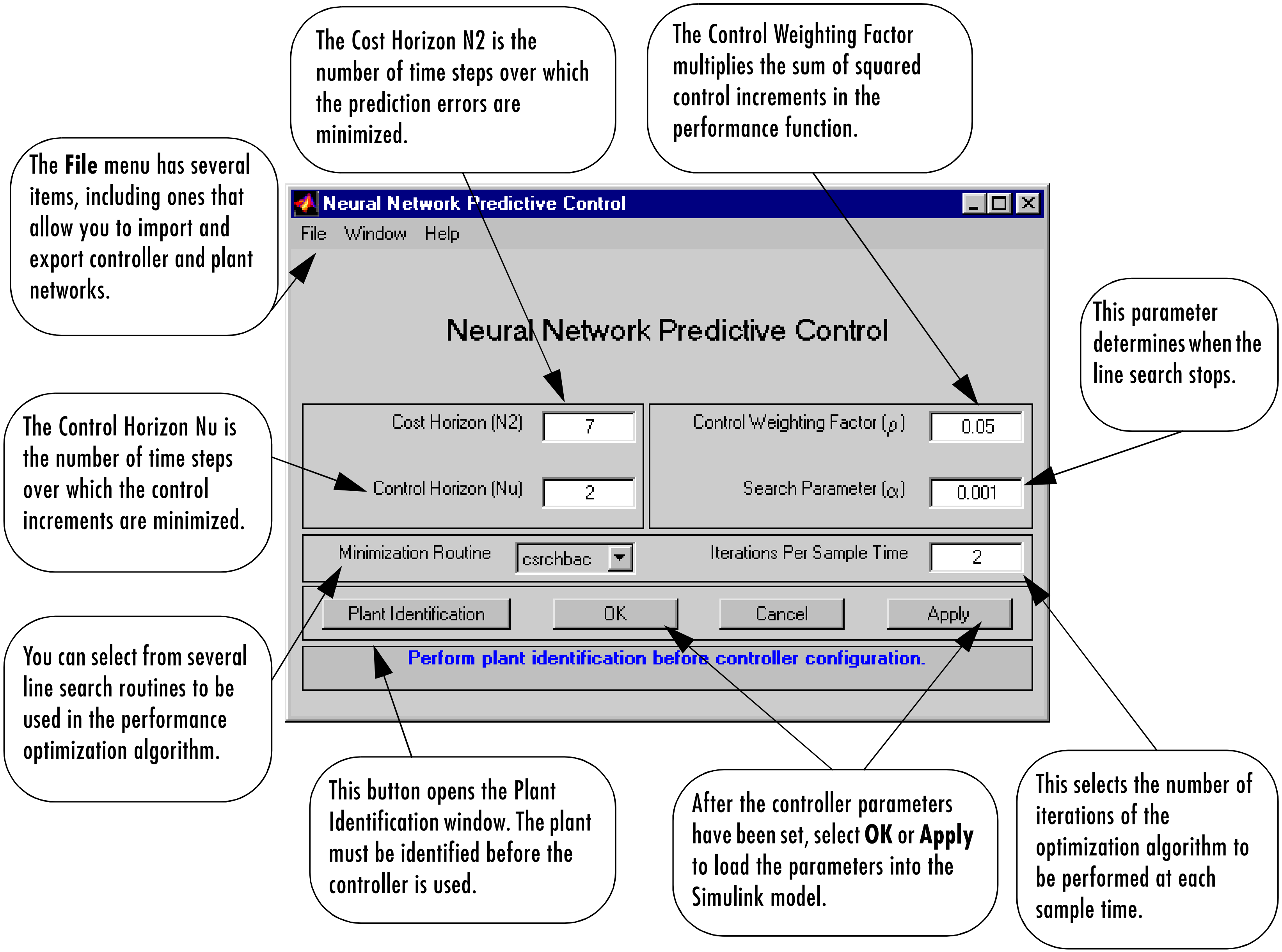
$$\frac{dh(t)}{dt} = w_1(t) + w_2(t) - 0.2\sqrt{h(t)}$$

$$\frac{dC_b(t)}{dt} = (C_{b1} - C_b(t))\frac{w_1(t)}{h(t)} + (C_{b2} - C_b(t))\frac{w_2(t)}{h(t)} - \frac{k_1 C_b(t)}{(1 + k_2 C_b(t))^2}$$

where $h(t)$ is the liquid level, $Cb(t)$ is the product concentration at the output of the process, $w1(t)$ is the flow rate of the concentrated feed $Cb1$, and $w2(t)$ is the flow rate of the diluted feed $Cb2$. The input concentrations are set to $Cb1$ = 24.9 and $Cb2$ = 0.1. The constants associated with the rate of consumption are $k1$ = 1 and $k2$ = 1.

Model matematyczny w Matlabie - Simulinku




The Cost Horizon N2 is the number of time steps over which the prediction errors are minimized.
The Control Weighting Factor multiplies the sum of squared control increments in the performance function.
The File menu has several items, including ones that allow you to import and export controller and plant networks.
Neural Network Predictive Control
File Window Help
Neural Network Predictive Control
This parameter determines when the line search stops.
Cost Horizon (N2) 7
Control Weighting Factor (ρ) 0.05
The Control Horizon Nu is the number of time steps over which the control increments are minimized.
Control Horizon (Nu) 2
Search Parameter (α) 0.001
Minimization Routine csrchbac
Iterations Per Sample Time 2
Plant Identification
OK
Cancel
Apply
You can select from several line search routines to be used in the performance optimization algorithm.
Perform plant identification before controller configuration.
This button opens the Plant Identification window. The plant must be identified before the controller is used.
After the controller parameters have been set, select OK or Apply to load the parameters into the Simulink model.
This selects the number of iterations of the optimization algorithm to be performed at each sample time.