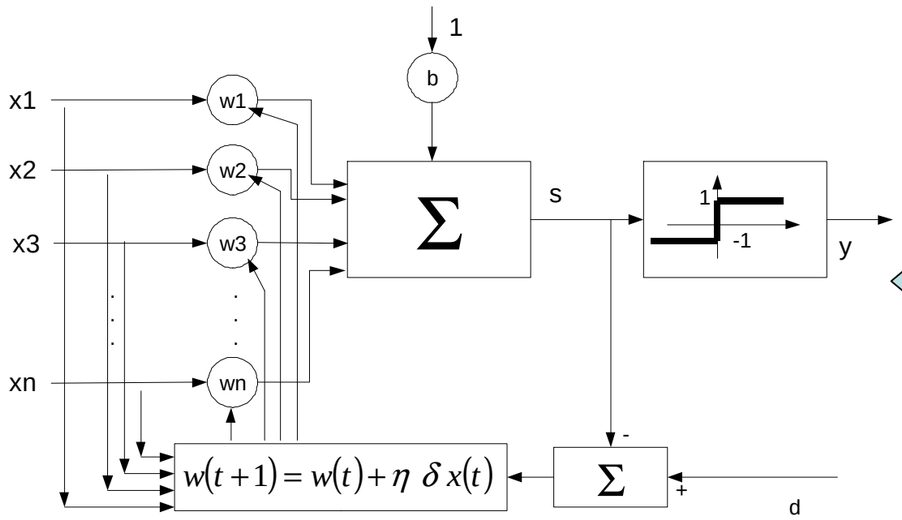
SZTUCZNE SIECI NEURONOWE

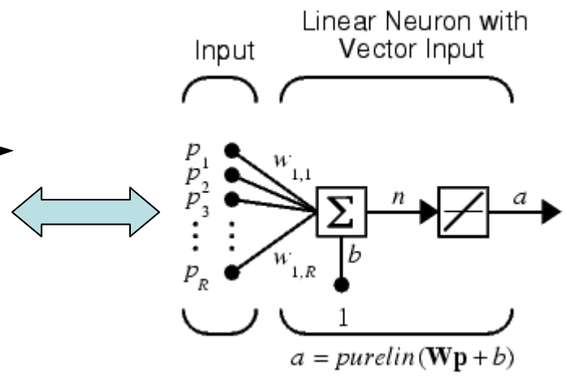
Ang. Artifficial neural network

Wykład 3

Dr inż. Piotr Urbanek

Neuron Adaline (Adaptive Linear Neuron)





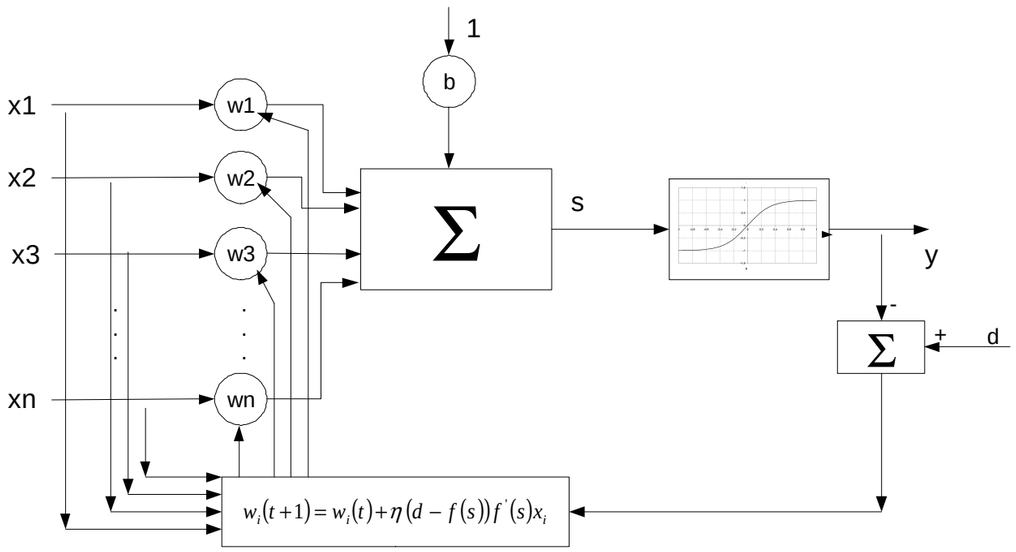
$\varepsilon = d - s$

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.



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

gdzie:

Funkcja aktywacji:

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

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

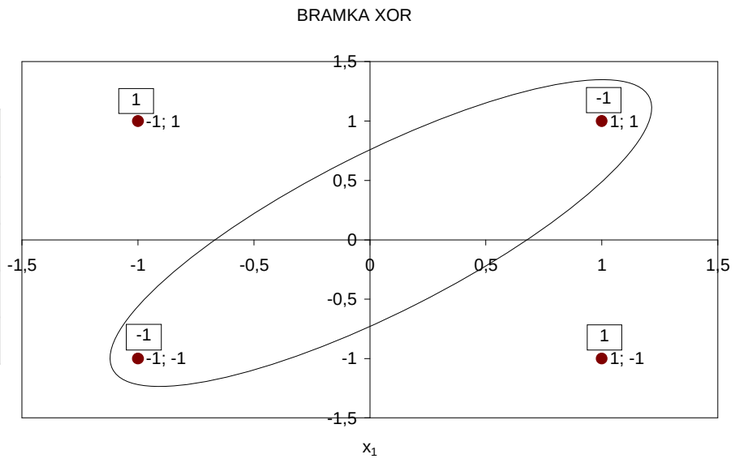
$$f'(s)=\beta f(s)(1-f(s))$$

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

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

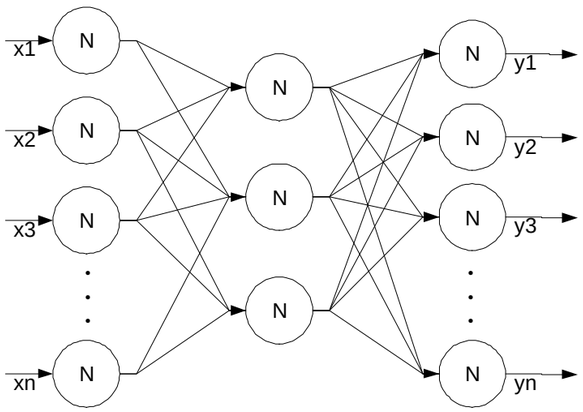


Sieci wielowarstwowe to odpowiednio ze sobą połączone neurony rozmieszczone w dwóch, lub więcej warstwach.

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.

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.

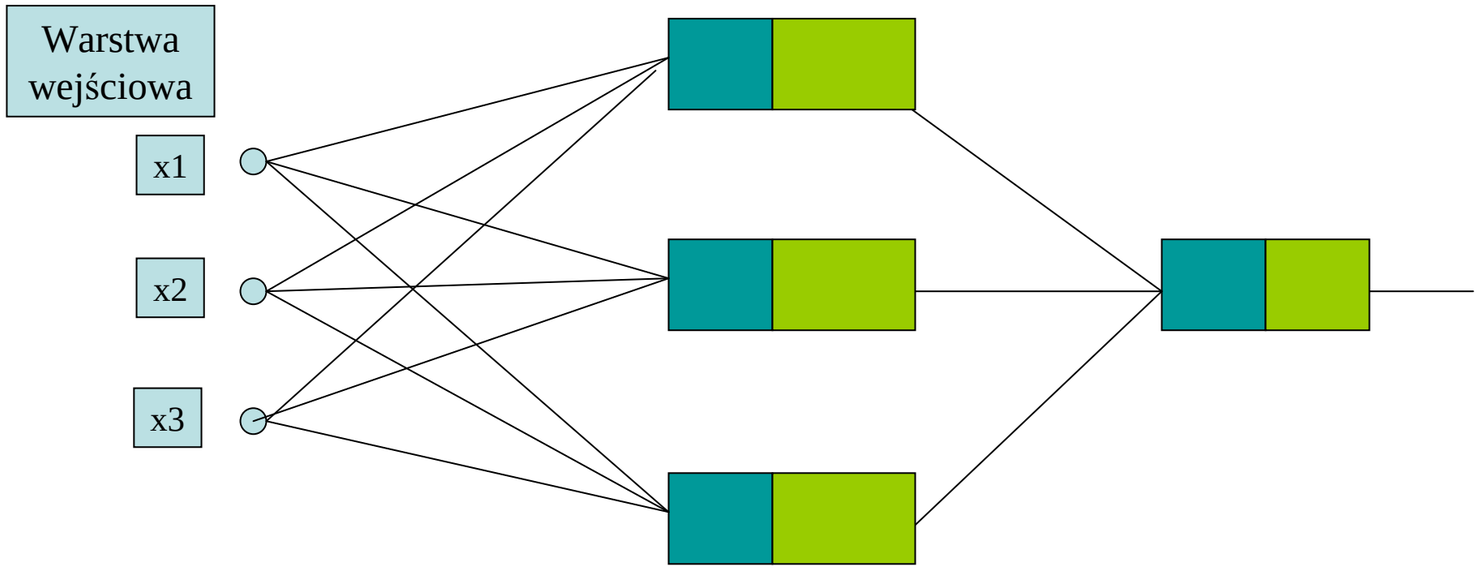
Jednym z najczęściej stosowanych algorytmów uczenia tego typu sieci jest

Algorytm wstecznej propagacji błędów.

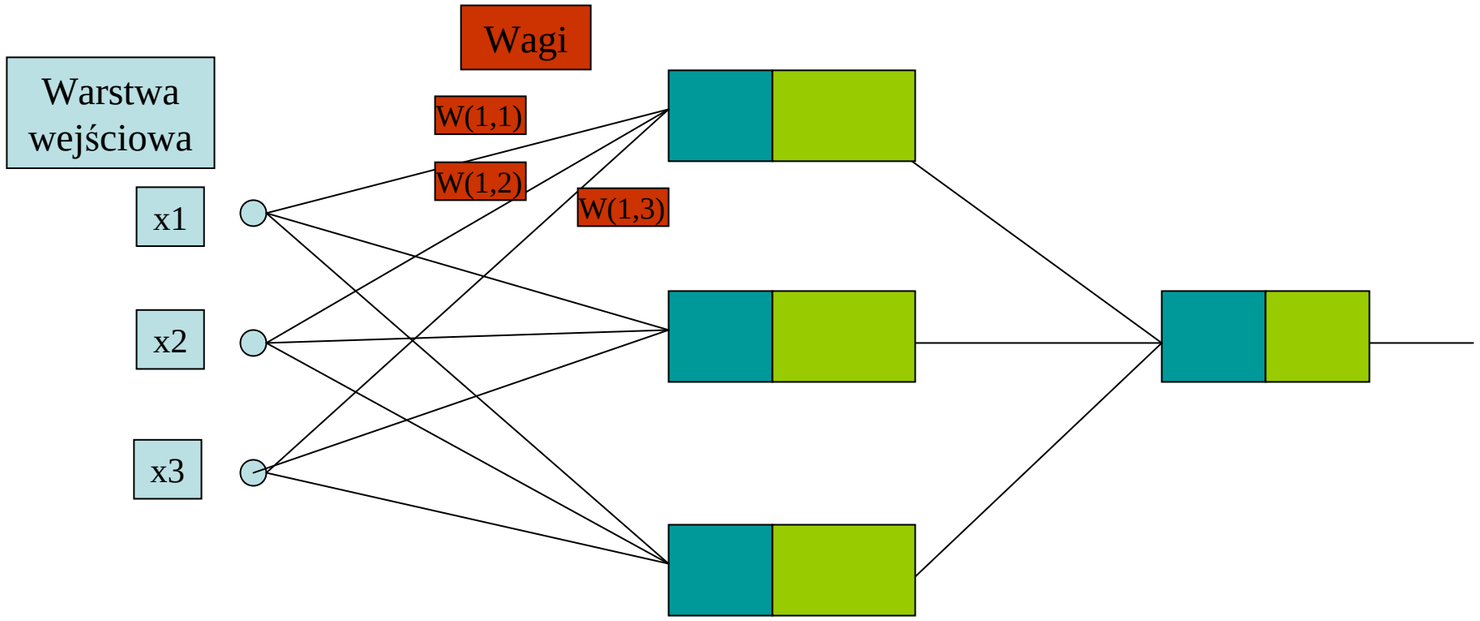
Algorytm ten należy do klasy algorytmów uczenia sieci z nauczycielem.

Wykorzystuje on do nauki dwa wektory: uczący i wzorcowy.

Algorytm Wstecznej Propagacji Błędu



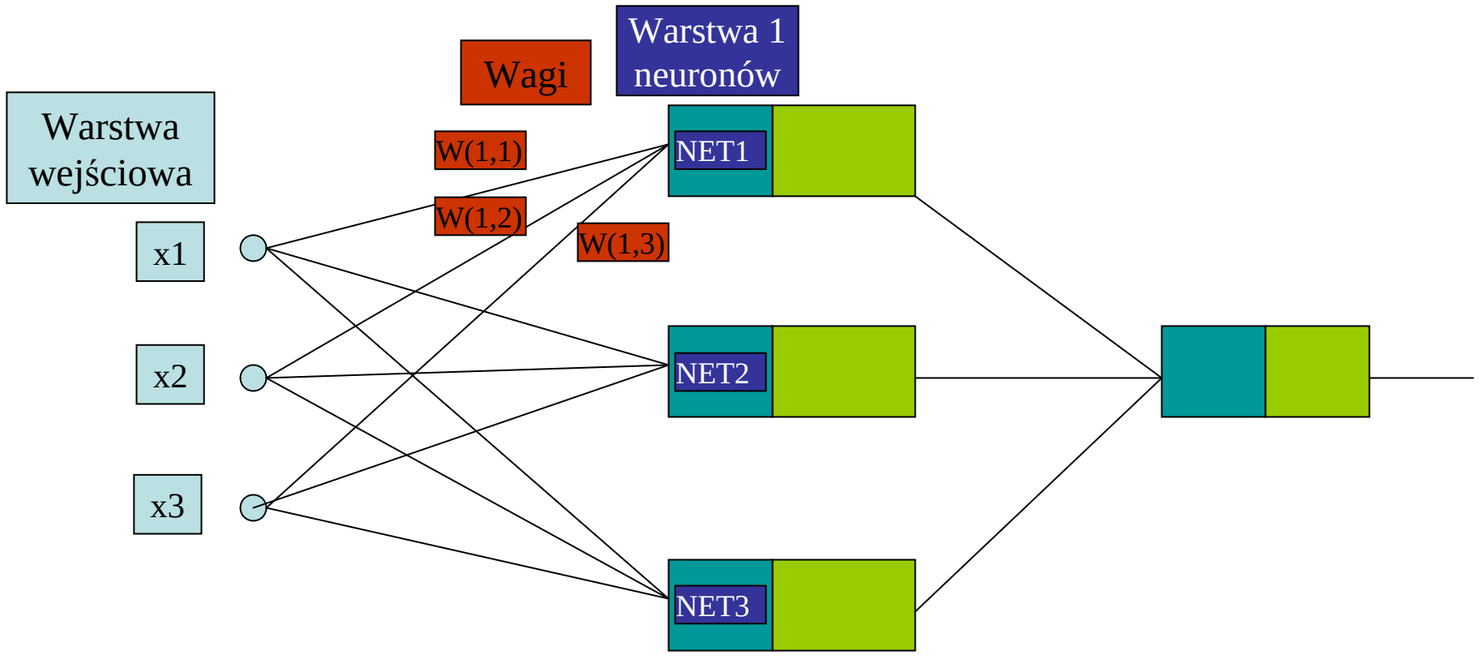
Na warstwę wejściową podawane są sygnały wejściowe (x1..xn). Mogą to być np. Dane giełdowe, zdigitalizowane próbki głosu do rozpoznania, lub inne. Celem warstwy wejściowej jest odpowiednie rozprowadzenie tych sygnałów do neruronów warstwy pierwszej sieci neuronowej.



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)

Algorytm Wstecznej Propagacji Błędu



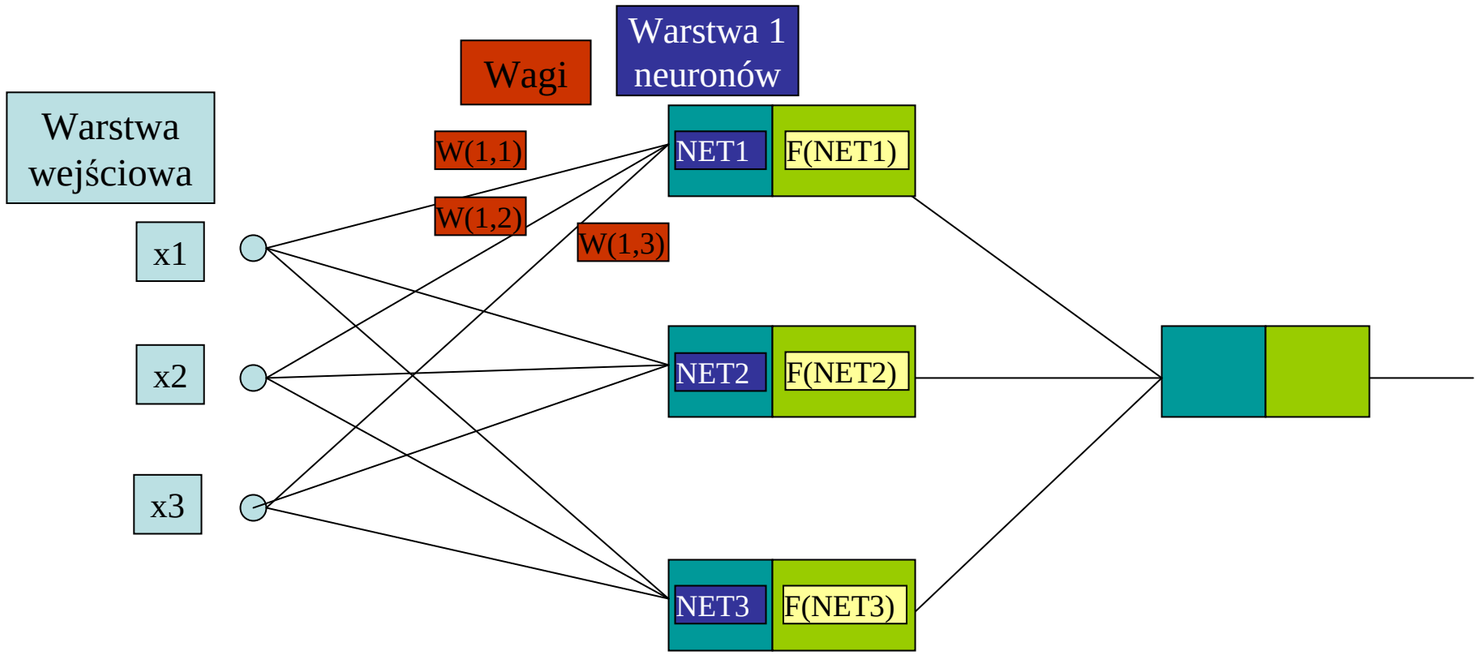
Wartości uzyskane z przemnożenia sygnałów wejściowych oraz odpowiednich wag są sumowane w każdym neuronie dając odpowiednie wartości potencjału neuronu (NET):

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)

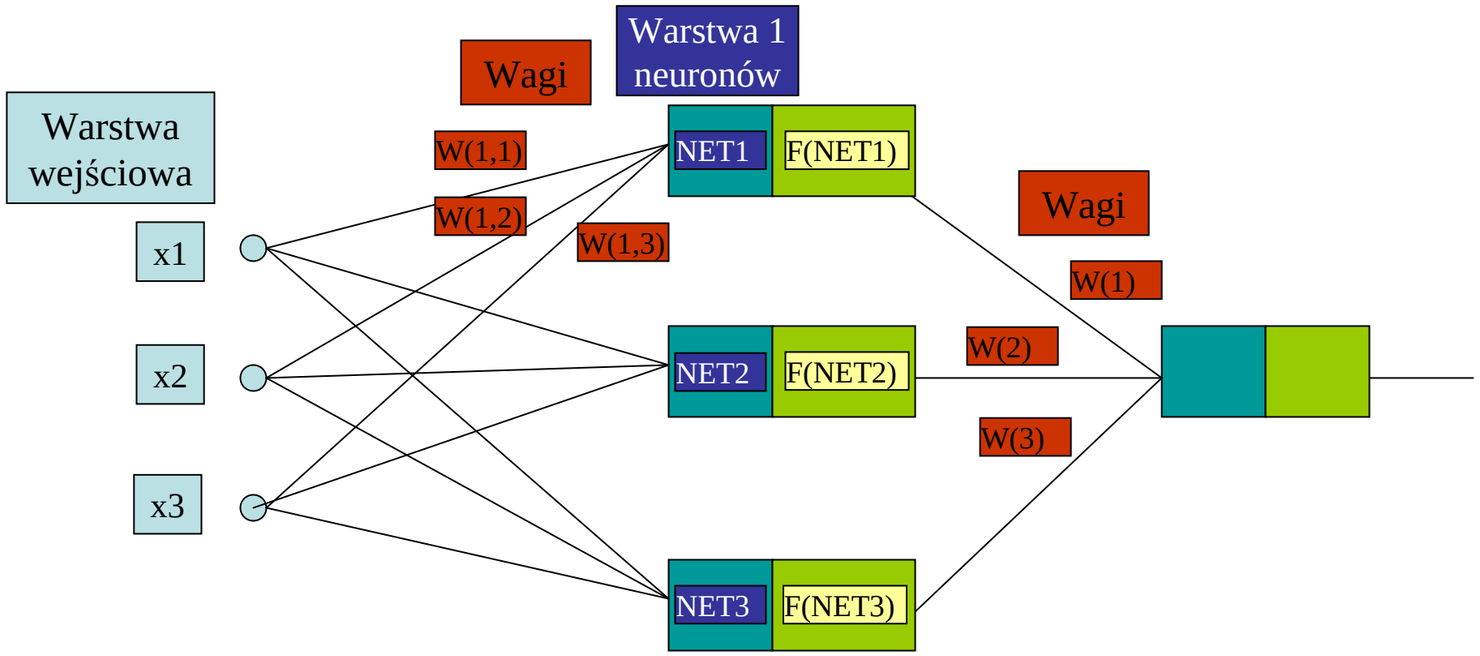
Algorytm Wstecznej Propagacji Błędu



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:

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

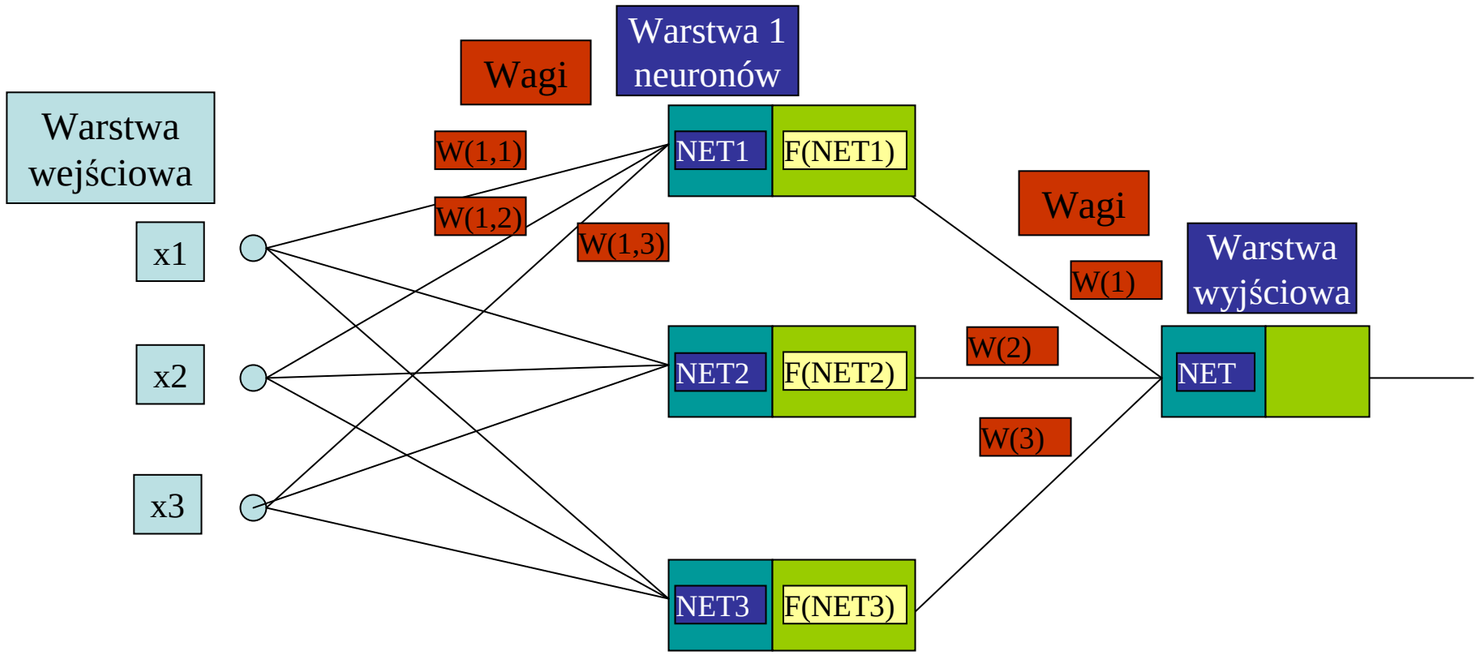
Algorytm Wstecznej Propagacji Błędu



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

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

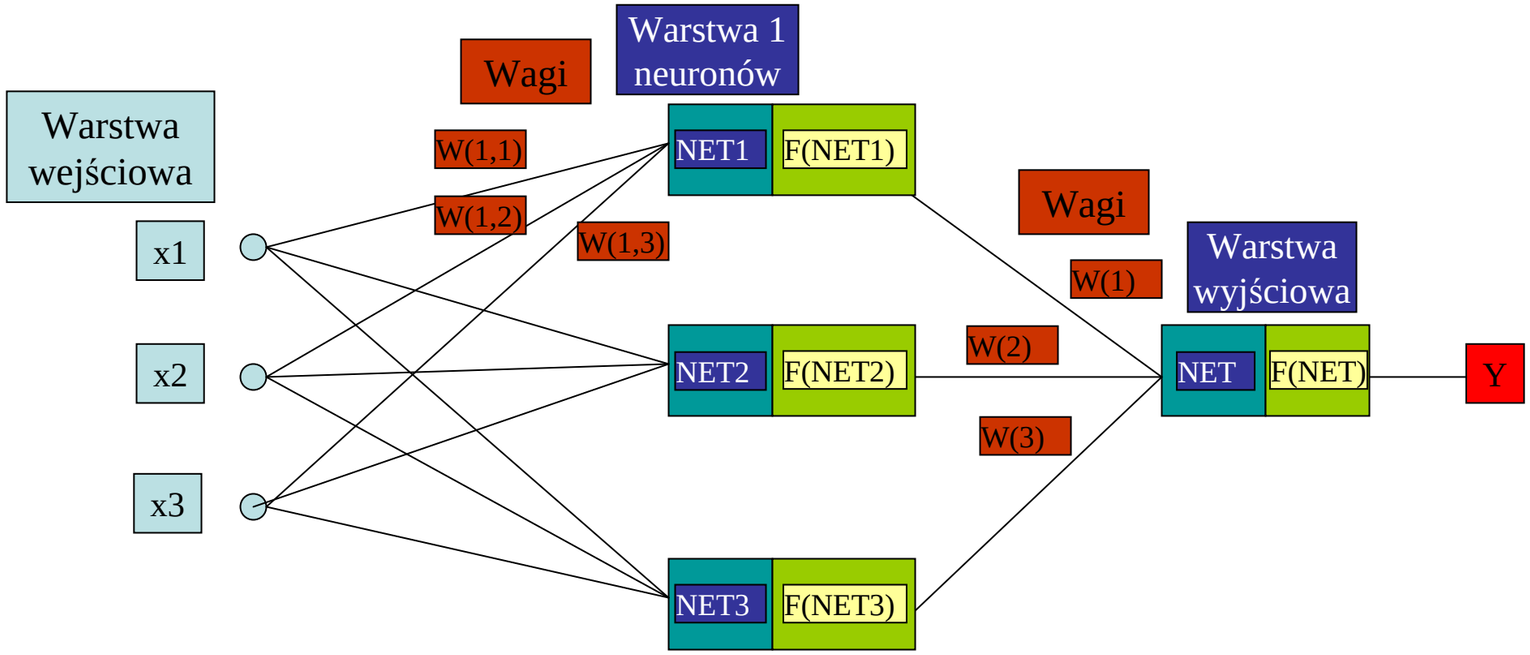
Algorytm Wstecznej Propagacji Błędu



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

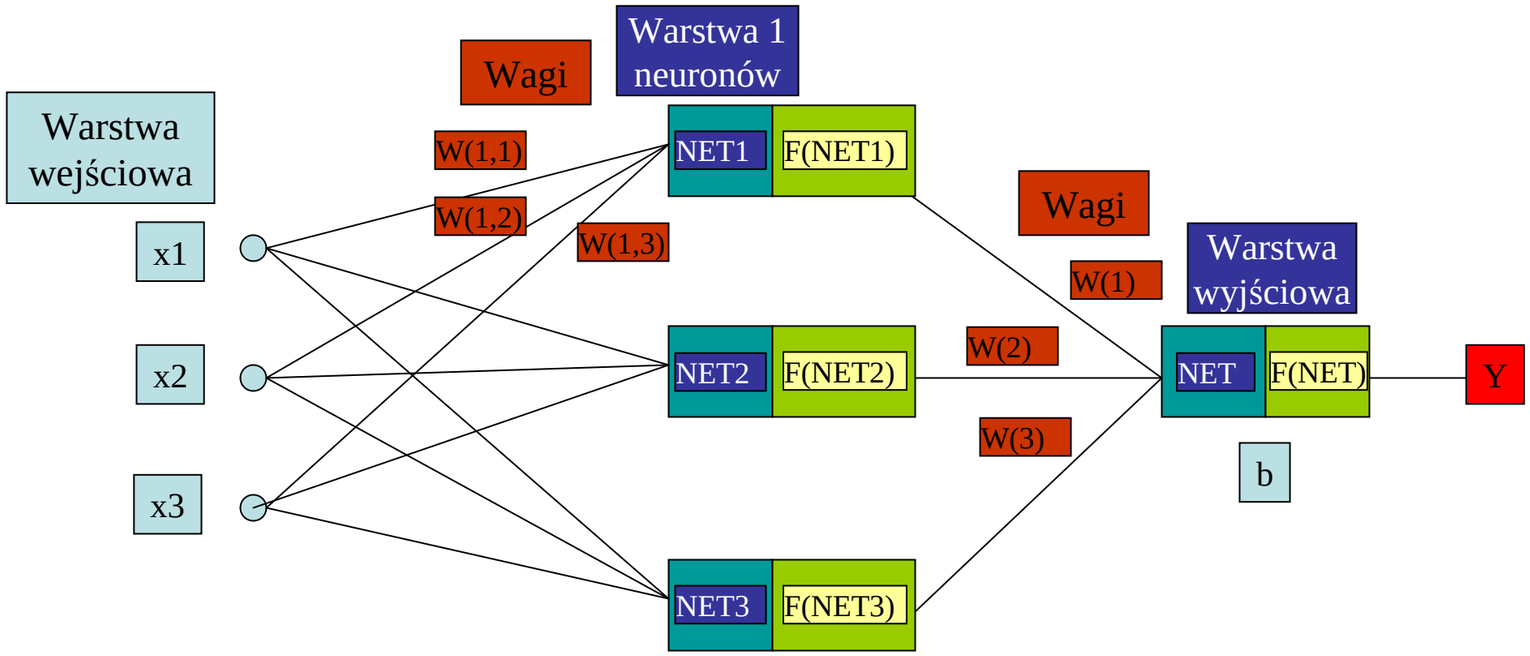


Sygnał NET zostaje podany jako argument nieliniowej funkcji sigmoidalnej f(NET) postaci:

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

Dając w wyniku wartość wyjściową sieci (y).

Algorytm Wstecznej Propagacji Błędu

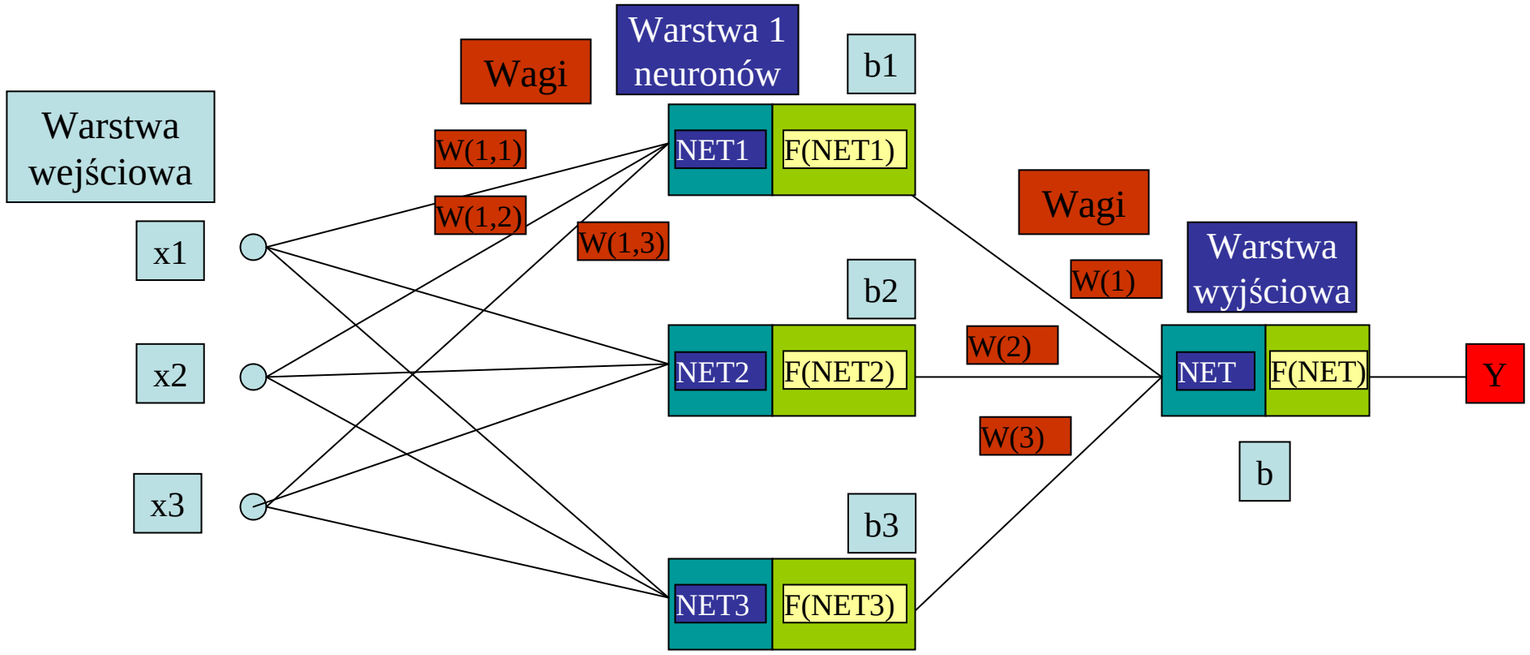


Wyznaczony jest błąd sieci porównując odpowiedź sieci (Y) z odpowiedzią właściwą (Z) otrzymując:

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

Będący błędem neuronu warstwy wyjściowej

Algorytm Wstecznej Propagacji Błędu

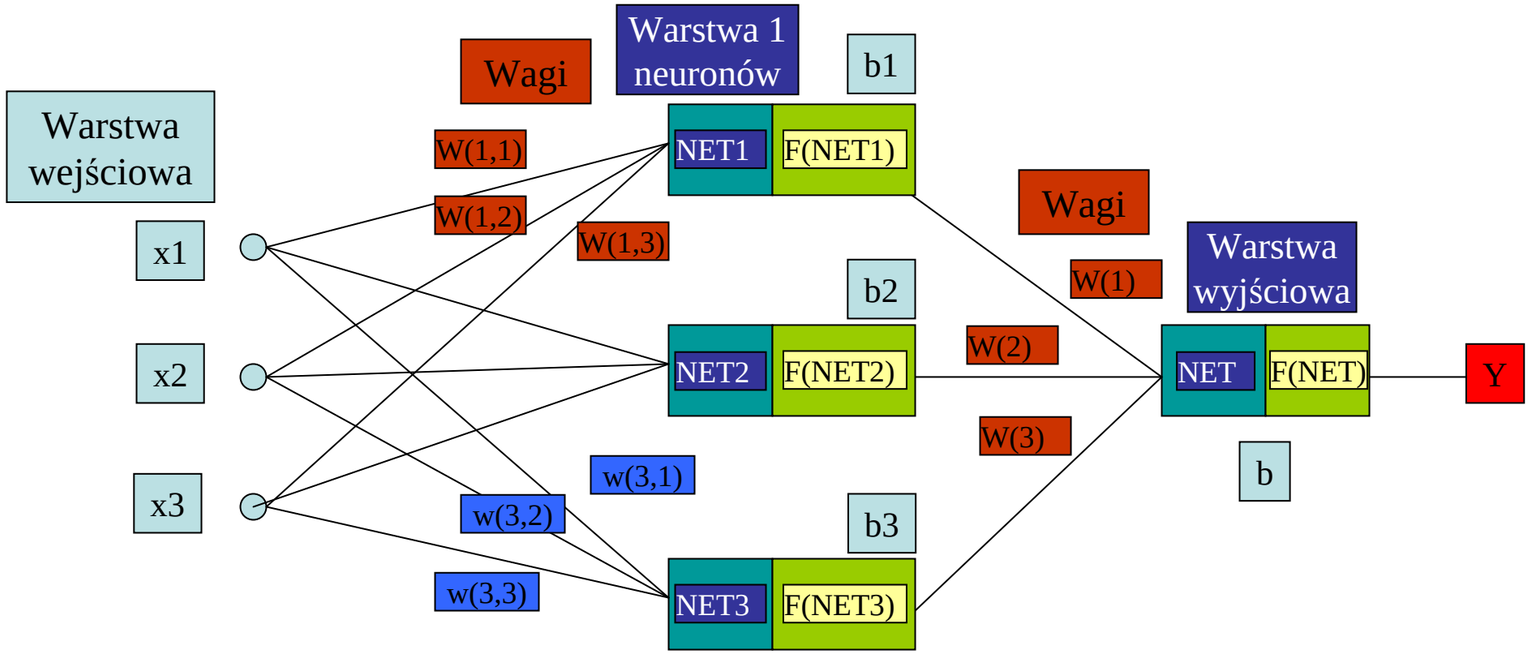


Wyznaczony jest neuronów warstwy ukrytej za pomocą wzoru:

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

Oznacza to, 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. Przetwarzane są w ten sposób wszystkie neurony (w naszym przypadku są tylko dwie warstwy, ale w sytuacji ogólniejszej przed warstwą ukrytą może istnieć jeszcze jedna warstwa).

Algorytm Wstecznej Propagacji Błędu

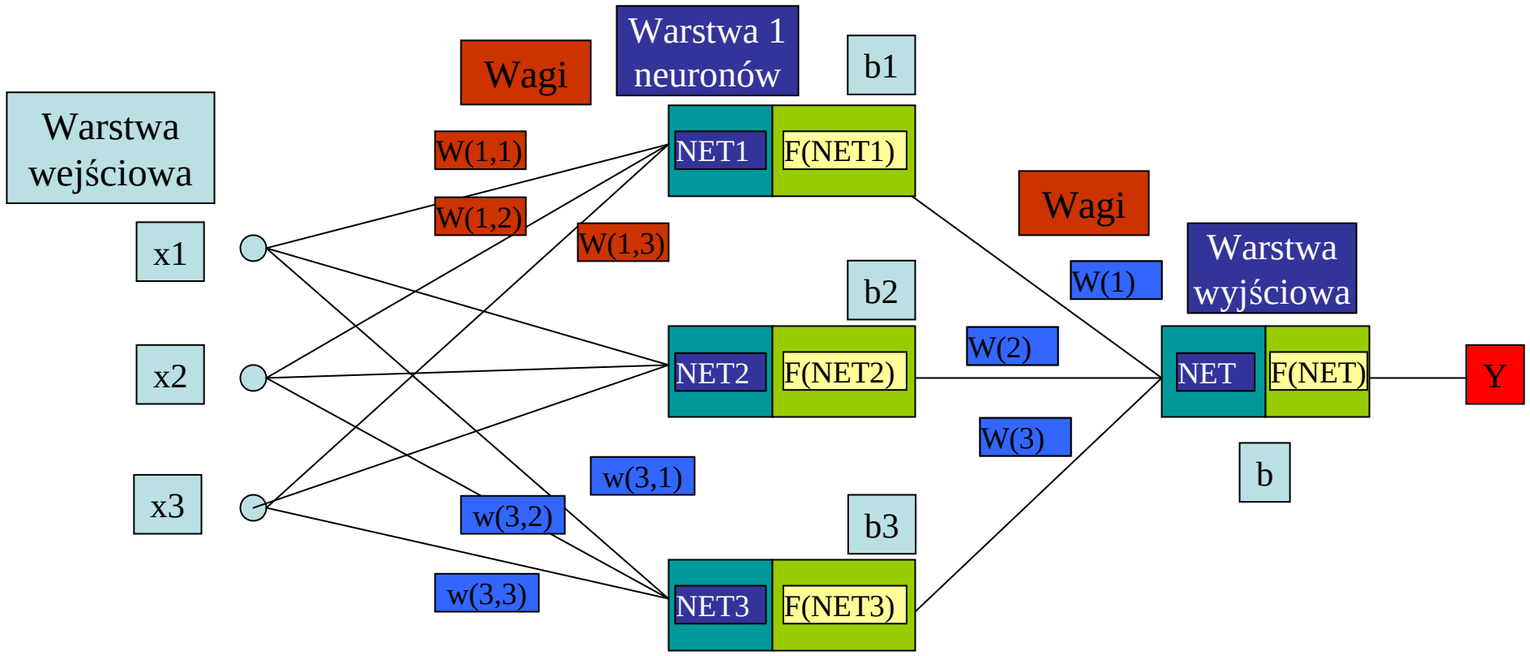


Aktualizacja wag zachodzi w kierunku przewodzenia sygnałów według wzoru:

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

Gdzie η 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 Błędu



Aktualizacja w warstwie wyjściowej analogicznie według wzoru:

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

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*) – ujęcie matematyczne

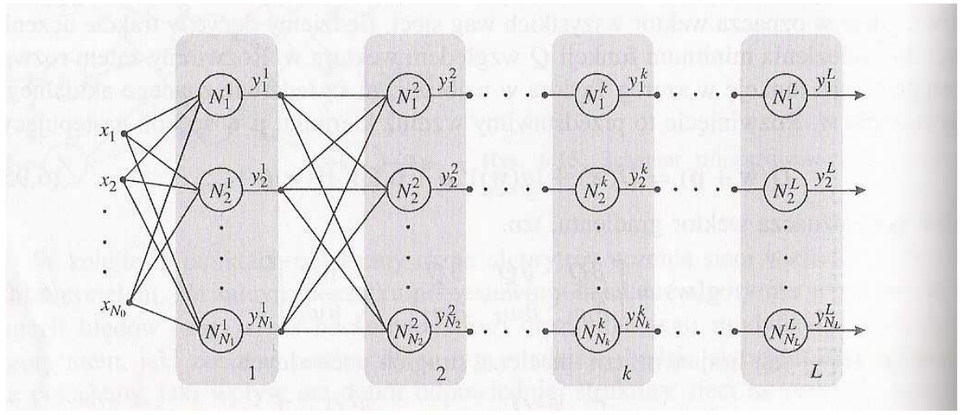
Podajemy na wejście sieci wartości wejściowe ciągu uczącego

↓

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.



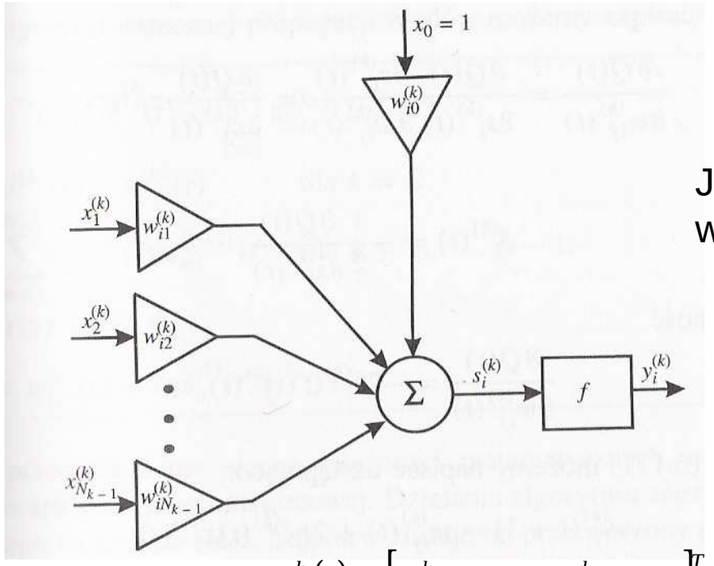
$N_i^k, i = 1, \ldots, N_k$

N_i^k - *neurony sigmoidalne*

$x = \left[x_1(t), \ldots, x_{N_0}(t)\right]^T \qquad t = 1, 2, \ldots$ *sygnały wejściowe*

$y_i^{(k)}(t),\ i = 1, \ldots, N_K, k = 1, \ldots, L$ *sygnały wyjściowe*

Model neuronu N_i^k



Neuron posiada k-wejść tworzących wektor:

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

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

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

$$w_i^k(t) = \left[w_{i,0}^k(t), \ldots, w_{i,N_{k-1}}^k(t)\right]^T \qquad i = 1, \ldots, N_K, k = 1, \ldots, L$$

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

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

w^k_{ij} - waga i-tego neuronu w warstwie k połączonego z j-tym wejściem. x^k_j – j-te wejście w warstwie k.

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

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

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

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

$$w_{ij}^k(t+1)=w_{ij}^k(t)-\eta\frac{\partial Q(t)}{\partial w_{ij}^k(t)}$$

Ponieważ: $$\frac{\partial Q(t)}{\partial w_{ij}^k(t)}=\frac{\partial Q(t)}{\partial s_i^k(t)}\cdot\frac{\partial s_i^k(t)}{\partial w_{ij}^k(t)}=\frac{\partial Q(t)}{\partial s_i^k(t)}\cdot x_j^k(t)$$

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

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

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

Sposób obliczania wartości δ_i^k jest uzależniony od warstwy sieci.

Dla warstwy ostatniej: $$\delta_i^L(t)=-\frac{1}{2}\frac{\partial Q(t)}{\partial s_i^L(t)}=-\frac{1}{2}\frac{\partial\left(d_i^L(t)-y_i^L(t)\right)^2}{\partial s_i^L(t)}=$$

$$=Q_i^L(t)\frac{\partial y_i^L(t)}{\partial s_i^L(t)}=Q_i^L(t)\,f'\left(s_i^L(t)\right)$$

dla dowolnej warstwy k ≠ L mamy:

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

$$= f'\left(s_i^k(t)\right)\sum_{m=1}^{N_{k+1}} \delta_m^{(k+1)}(t) w_{mi}^{(k+1)}(t)$$

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

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

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

Algorytm wstecznej propagacji błędów dla sieci wielowarstwowej, jednokierunkowej

Sygnał wyjściowy:

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

Miara błędu sieci:

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

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

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

START

Podanie wzorca uczącego na wejście sieci

Przetworzenie sygnału przez neurony k-tej warstwy.

$$y_i^{(k)}(t) = f\left(s_i^k(t)\right)$$

$$s_i^k(t) = \sum_{j=0}^{N_{k-1}} w_{ij}^k(t)\, x_j^k(t)$$

Czy k=L ?

NIE

TAK

Przetworzenie sygnału przez neurony ostatniej warstwy (L)

Oblicz błąd na wyjściu sieci

$$Q_i^L(t) = d_i^L(t) - y_i^L(t)$$

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

Dokonaj korekty wag neuronów L-tej warstwy

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

Ponieważ nie znamy wartości δ_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:

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