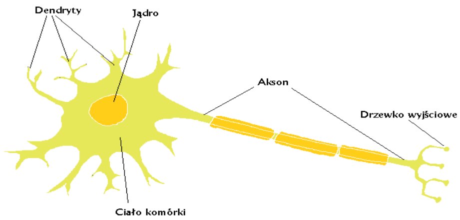
## SIECI NEURONOWE

- Mózg:
  - ok. 100 mld neuronów połączonych ze sobą
  - liczba połączeń między neuronami: $10^{15}$
  - impulsy z częstotliwością 1-100 Hz, czas trwania 1-2 ms
  - szybkość pracy mózgu: $10^{18}$ operacji/s
- neurony – podobne do siebie, proste
- funkcjonowanie sieci: przetwarzanie sygnałów elektrochemicznych
- przetwarzanie – równoległe, wiele neuronów pobudzanych jest równocześnie



## SIECI NEURONOWE

- Model neuronu biologicznego

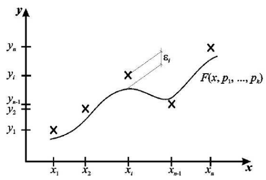



Dendryty – odbierają impulsy z innych neuronów
Akson – przekazuje impuls do neuronu kolejnego

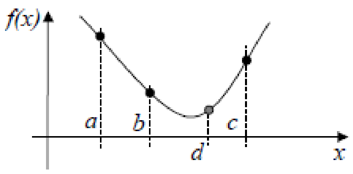


## SIECI NEURONOWE - zastosowania

- Aproksymacja funkcji



- Predykcja, prognozowanie
- Optymalizacja






## SIECI NEURONOWE - zastosowania

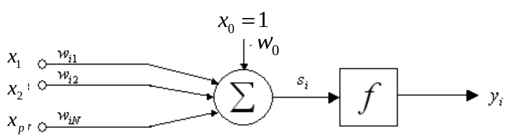
Uwaga

Sieć neuronowa pełni w każdym z powyższych zastosowań rolę ***uniwersalnego aproksymatora funkcji wielu zmiennych***, realizując funkcję nieliniową o postaci $y = f(x)$, gdzie $x$ jest wektorem wejściowym, a $y$ realizowaną funkcją wektorową wielu zmiennych.



## MODEL NEURONU

- Sieć neuronowa jest uproszczonym modelem mózgu ludzkiego
- Matematyczny model sztucznego neuronu




$x = \left[x_1, \ldots, x_p\right]^T$ -wektor wejściowy, $w_i = \left[w_{i1}, \ldots, w_{ip}\right]^T$ wagi $i$-tego neuronu ($w_0$ - bias, próg), $f$ – funkcja aktywacji neuronu, $s_i$ - sygnał wyjściowy $s_i = \sum_{j=0}^{p} x_j w_{ij} = w_i^T x + w_0$, $y_i = f(s_i)$ - odpowiedź neuronu



## SIECI NEURONOWE

**Charakterystyka**

- Neurony - proste jednostki przetwarzające
- Sieć neuronowa – połączony układ neuronów
- Wiedza neuronu – w całości zapisana jest w jego wagach
- Wiedza sieci – zawarta w wagach neuronów oraz niejawnie w strukturze połączeń między neuronami
- Uczenie sieci neuronowej – dobór wag neuronów

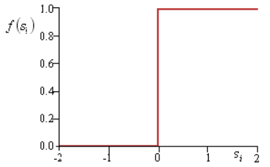
## FUNKCJE AKTYWACJI NEURONU

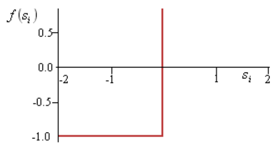
- Funkcja tożsamościowa: $f(s_i) = s_i$
- Funkcje skokowe:
  - Unipolarna

$$f(s_i) = \begin{cases} 1 & dla \quad s_i \geq 0 \\ 0 & dla \quad s_i < 0 \end{cases}$$

  - Bipolarna

$$f(s_i) = \begin{cases} 1 & dla \quad s_i \geq 0 \\ -1 & dla \quad s_i < 0 \end{cases}$$






## FUNKCJE AKTYWACJI NEURONU

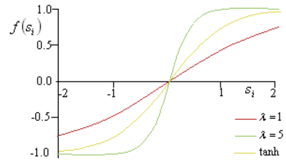
- Funkcje ciągłe nieliniowe
  - Unipolarna

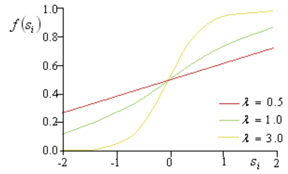
$$f_\lambda(s_i) = \frac{1}{1+\exp(-\lambda s_i)}$$

  - Bipolarna

$$f_\lambda(s_i) = \frac{2}{1+\exp(-\lambda s_i)} \text{-1}$$

$$f(s_i) = \tanh(\lambda s_i) = \frac{1-\exp(\lambda s)}{1+\exp(-\lambda s)}$$



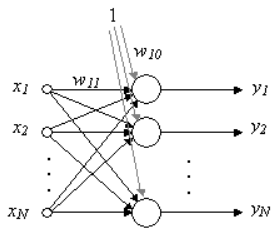

## TYPY ARCHITEKTURY

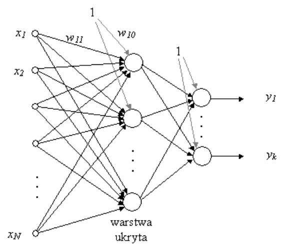


## SIECI

- **Jednokierunkowe**

Jednowarstwowe Wielowarstwowe



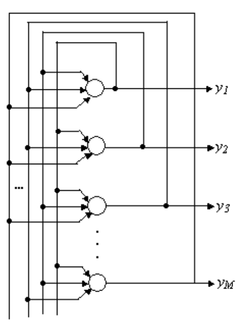



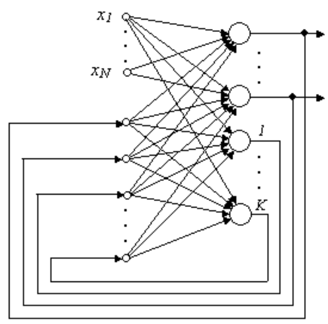



## TYPY ARCHITEKTURY SIECI

- **Rekurencyjne**

Jednowarstwowe Wielowarstwowe






## PERCEPTRON ROSENBLATTA

- Perceptron Rosenblatta / perceptron dyskretny
  – neuron ze skokową bipolarną funkcją aktywacji
- Perceptron Rosenblatta jako klasyfikator:

Rozważamy zadanie klasyfikacji dla dwóch klas

Zbiór uczący:

$$(x_1, d_1), \ldots, (x_n, d_n) \in R^n \times D, \quad D = \{-1, +1\}$$

Reguła decyzyjna:

$$f(s_i) = \begin{cases} 1 & dla \quad s_i \geq 0 \\ -1 & dla \quad s_i < 0 \end{cases}$$



## PERCEPTRON ROSENBLATTA

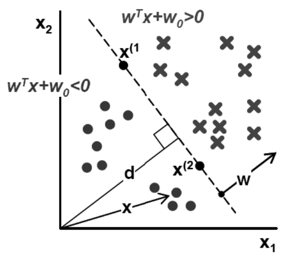
W przypadku liniowo separowanym istnieje hiperpłaszczyzna taka, że:

$$f(s) = w^T x + w_0 \geq 0 \text{ dla } d_i = +1$$

$$f(s) = w^T x + w_0 < 0 \text{ dla } d_i = -1$$

Reguła decyzyjna:

$$f(s_i) = \begin{cases} 1 & dla \quad s_i \geq 0 \\ -1 & dla \quad s_i < 0 \end{cases}$$



## UCZENIE PERCEPTRONU

- Uczenie perceptronu polega na znalezieniu wartości jego wag
- Różnica pomiędzy oczekiwaną wartością wyjścia neuronu a wartością otrzymaną stanowi błąd popełniony przez neuron przy prezentacji *j*-tego przykładu: $\delta_j = d_j - y_j$
- Poszukujemy rozwiązania minimalizującego perceptronową funkcję kryterialną: $J(\tilde{w}) = \sum_{\{x_i : d_i(w^T x + w_0) \le 0\}} \left(-\tilde{w}^T x - \tilde{w}_0\right)$
- Sumowanie odbywa się po obiektach niepoprawnie zaklasyfikowanych – minimalizacja liczby niepoprawnych zaklasyfikowań



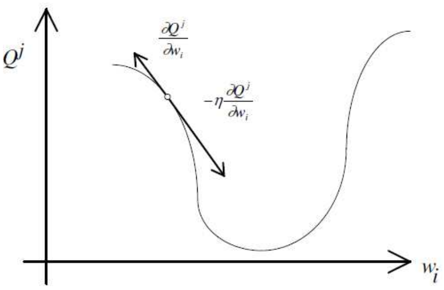
## UCZENIE PERCEPTRONU

- W praktyce wygodnie jest stosować rozszerzony zapis $x = \left[1, x_1, ..., x_p\right]^T$, wtedy $J(\tilde{w}) = \sum_{\{x_i : d_i(w^T x + w_0) \le 0\}} \left(-\tilde{w}^T x\right)$
- Poszukiwanie minimum perceptronowej funkcji kryterialnej jest równoważne minimalizacji błędu średniokwadratowego $Q = \frac{1}{2}\sum_{j=1}^{N}\left(d_j - y_j\right)^2 = \sum_{j=1}^{N} Q_j^{\,2}$, $Q_j = \frac{1}{2}\delta_j^2$
- Minimum powyższej funkcji kryterialnej poszukujemy metodą największego spadku gradientu



## UCZENIE PERCEPTRONU

Metoda największego spadku gradientu (ang. *gradient descent*):
Posługując się tą metodą dostajemy zależność pomiędzy modyfikacją *i*-tej wagi neuronu a zmianą wartości błędu przezeń popełnianego przy prezentacji *j*-tego przykładu






## UCZENIE PERCEPTRONU

- Metoda największego spadku gradientu:
  Rozpoczynając od rozwiązania początkowego (losowego) w każdym kroku uczenia do aktualnych wartości wag dodaje się poprawkę wprost proporcjonalną do gradientu funkcji kryterialnej w punkcie *w* (aktualna wartość wag) według wzoru:

$$w_{k+1} = w_k - \eta\nabla\left(J\left(w_k\right)\right)$$

- Gradient funkcji $J(\tilde{w}) = \sum_{\{x_i : d_i(w^T x + w_0) \le 0\}} \left(-\tilde{w}^T x\right)$ jest równy

$$\nabla J = \sum_{\{x_i : d_i(w^T x + w_0) \le 0\}} (-x)$$



## UCZENIE PERCEPTRONU

Reguła uczenia delta

1. Inicjuj wagi $w_0$ perceptronu wartościami losowymi
2. Podaj kolejny *j*-ty wektor zbioru uczącego i oblicz odpowiedź perceptronu
3. Jeżeli $d_j \ne y_j$ zmodyfikuj wagi neuronu, według wzoru:
   $w_{k+1} = w_k - \eta x_j$
   (w przeciwnym przypadku wagi się nie zmieniają)
4. Jeżeli *j=N* oblicz błąd całej epoki i w przypadku gdy jest większy od założonej wartości podstaw *j=1* i idź do 2.



## UCZENIE PERCEPTRONU

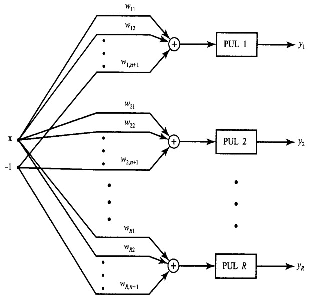
**Twierdzenie:**

Jeżeli istnieje wektor wag *w* oraz stała $w_0$ takie, że dla każdego obiektu zbioru uczącego zachodzi $d_i\left(w^T x + w_0\right) > 0$ (zbiór liniowo separowalny), to algorytm uczenia perceptronu pozwala na odnajdzie wag *w* oraz $w_0$ w skończonej liczbie kroków dla dowolnych wartości wag początkowych.

## Liniowy klasyfikator neuronowy

Klasyfikator perceptronowy dla przypadku wielu klas

- W przypadku wieloklasowym najprostszy klasyfikator neuronowy przyjmuje postać pojedynczej warstwy neuronów (perceptronów)
- Konieczne jest ustalenie reprezentacji etykiet zbioru uczącego zapewniające rozróżnienie klas






## Liniowy klasyfikator neuronowy

- Najczęściej spotykana reprezentacja, tzw. reprezentacja lokalna:
  Załóżmy, że dany jest zbiór uczący *V* wektorów cech obiektów należących do *K* klas.
  $V = \{(x_1, d_1), ..., (x_N, d_N)\}$, gdzie $x_i \in C_k$
  $d_i \in R^K$ takie, że $d_{ij} = \begin{cases} 0 & j \neq k \\ 1 & j = k \end{cases}$
- Powyższa reprezentacja dotyczy sieci neuronowych z unipolarną funkcją aktywacji, w przypadku stosowania bipolarnej funkcji aktywacji w miejsce 0 wpisujemy -1



## Liniowy klasyfikator neuronowy

Obserwacja:

- o Liczba neuronów jest równa ilości klas.

- Funkcje klasyfikujące liniowego klasyfikatora neuronowego:
  $g_i(x) = w_i^T x$, gdzie $w_i$ jest wektorem wag *i*-tego neuronu
- Wartością *i*-tej funkcji klasyfikującej jest odpowiedź *i*-tego neuronu
- Przypomnienie: Odpowiedź sieci obliczamy stosując wzór macierzowy: $f(net) = f(Wx)$, $W \in R^K \times R^{m+1}$, gdzie *m* jest wymiarem przestrzeni cech obiektów uczących



## Liniowy klasyfikator neuronowy

Uczenie klasyfikatora w przypadku użycia dyskretnej funkcji aktywacji

- Przebiega podobnie do uczenia pojedynczego neuronu
- W przypadku poprawnej klasyfikacji wagi nie są modyfikowane
- W przypadku niepoprawnej klasyfikacji dla obiektu *x* należącego do klasy *k* modyfikowane są zgodnie ze wzorem:

$$w_i = \begin{cases} w_i + cx & \text{dla} \quad i = k \\ w_i - cx & \text{dla} \quad i : i \neq k \wedge w_i^T x \geq w_k^T x \\ w_i & \text{dla} \quad i : i \neq k \wedge w_i^T x < w_k^T x \end{cases}$$



## Liniowy klasyfikator neuronowy

Uczenie klasyfikatora w przypadku użycia dyskretnej funkcji aktywacji

- można podać ogólniej wzór korekcji wag, wg wzoru:

$$w_i^{(k+1)} = w_i^{(k)} + \frac{1}{2}\eta(d_i - y_i)x_i ,$$

  gdzie indeks dolny oznacza składową odpowiednich wektorów
- Korekcji wag wg powyższego wzoru dokonujemy dla każdego podawanego wektora
- Uczenie sieci kończymy po odpowiedniej liczbie epok, bądź poprzez kontrolę błędu wyrażonego ilością niepoprawnych zaklasyfikowań.



## UCZENIE SIECI PERCEPTRONOWEJ

**Twierdzenie:**

Jeżeli istnieje macierz wag *W* oraz stała taka, że dla każdego obiektu zbioru uczącego zachodzi $d = Wx$ (co oznacza, że macierz wag *W* gwarantuje poprawną klasyfikację wszystkich wektorów zbioru uczącego – przypadek liniowo separowalny), to algorytm uczenia liniowe sieci perceptronowej pozwala na odnalezienie wag *W* w skończonej liczbie kroków dla dowolnych wartości wag początkowych.

## Uczenie sieci jednowarstwowej – reguła delta

Przypadek z ciągłymi funkcjami aktywacji

Niech $x = [x_1, ..., x_{m-1}, -1]^T$ będzie podawanym na wejście wektorem, $W = \begin{bmatrix} w_{11} & \cdots & w_m \\ \vdots & \ddots & \vdots \\ w_{K1} & \cdots & w_{Km} \end{bmatrix}$ macierzą wag,

oraz $y = f(Wx)$ odpowiedzią sieci.

Ponadto niech *d* oznacza pożądany sygnał wyjściowy.

Błąd klasyfikacji podanego wektora *x* wyrażamy wzorem:

$$E = \frac{1}{2}(d-y)^T (d-y)$$



## Uczenie sieci jednowarstwowej – reguła delta

Błąd *E* minimalizujemy metodą gradientową,

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}},$$

Ponieważ błąd *E* zależy od wagi $w_{ij}$ tylko poprzez funkcję $net_i = w_i^T x$, zgodnie więc ze wzorem na pochodne cząstkowe funkcji złożonej:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial net_i} \frac{\partial net_i}{\partial w_{ij}}$$

Pochodną $\delta_i = \frac{\partial E}{\partial net_i}$ nazywamy sygnałem delta generowanym przez neuron *i*.



## Uczenie sieci jednowarstwowej – reguła delta

Ponieważ $\frac{\partial net_i}{\partial w_{ij}} = x_j$, wyrażenie na korekcję wag przyjmuje postać $w_{ij} := w_{ij} + \eta \delta_i x_j$. Przy zastosowaniu notacji macierzowej:

$W := W + \eta \delta x^T$, gdzie $\delta = \begin{bmatrix} \delta_1 \\ \vdots \\ \delta_K \end{bmatrix}$, natomiast

$$\delta_i = \frac{\partial E}{\partial net_i} = -\frac{1}{2} \frac{\partial}{\partial net_i} (d_i - y_i)^2 = (d_i - y_i) \frac{\partial y_i}{\partial net_i} = (d_i - y_i) f'(net_i)$$



## Uczenie sieci jednowarstwowej – reguła delta

Ostatecznie otrzymujemy wzór na korekcję wag, nazywany regułą uczenia delta dla sieci jednowarstwowych z dowolną różniczkowalną funkcją aktywacji $w_{ij} := w_{ij} + \eta (d_i - y_i) f'(net_i) x_j$, co przy przyjętych oznaczeniach daje:

$$w_{ij} := w_{ij} + \eta \delta_i x_j, \text{ gdzie } \delta_i = (d_i - y_i) f'(net_i)$$

- W przypadku unipolarnej funkcji aktywacji:
$$\delta_i = (d_i - y_i) y_i (1 - y_i)$$
- W przypadku bipolarnej funkcji aktywacji:
$$\delta_i = \frac{1}{2}(d_i - y_i)(1 - y_i^2)$$
- Wartości $\delta$ zależą tylko od wyjścia *y* oraz oczekiwanej odp. *d*



## Uczenie sieci jednowarstwowej – reguła delta

Uwagi

- Uczenie sieci odbywa się w cyklach (epokach) uwzględniających wszystkie elementy zbioru uczącego
- Warunek stopu procedury uczenia określa się w postaci ilości cykli uczących, bądź progu **całkowitego błędu klasyfikacji** $E = \sum_{k=1}^{N} E_k$, gdzie $E_k$ jest błędem klasyfikacji *k*-tego wektora zbioru uczącego.



## Uczenie sieci jednowarstwowej – reguła delta

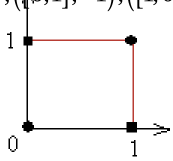
Uwagi

- Dla każdego wiersza macierzy wag *W* korygujemy wagi $w_i$ odpowiadające *i*-temu neuronowi, $i = 1, ..., K$.
- Korekcja wag dla neuronu unipolarnego odbywa się zgodnie ze wzorem:
$$w_i := w_i + \eta (d_i - y_i) y_i (1 - y_i) x$$
- Korekcja wag dla neuronu bipolarnego odbywa się zgodnie ze wzorem:
$$w_i := w_i + \frac{1}{2} \eta (d_i - y_i)(1 - y_i^2) x$$

# PROBLEM XOR

- Przykład problemu klasyfikacji nie dającego się rozwiązać za pomocą pojedynczego perceptronu, czy warstwy perceptronów (klasyfikator liniowy):

  $V = \{([0,0],1),([1,1],1),([0,1],-1),([1,0],-1)\}$

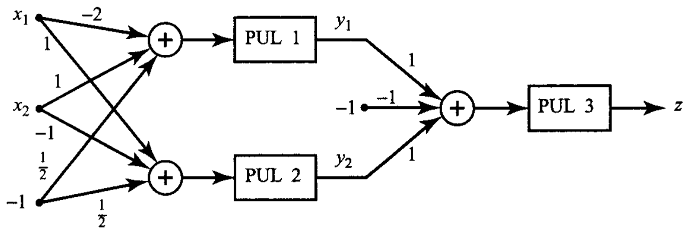



- Rozwiązanie – zastosowanie sieci wielowarstwowej – dodanie warstwy ukrytej



# Klasyfikatory liniowe – problem XOR

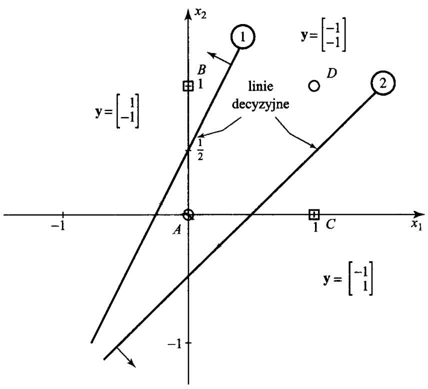
Przykład. Rozważmy sieć neuronową daną poniższym rysunkiem:




Sieć zapewnia poprawną klasyfikację wszystkich obiektów zbioru uczącego *V*.



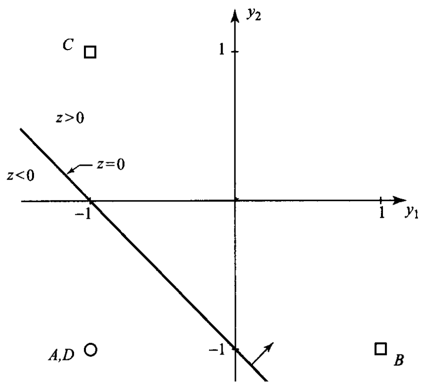
# Klasyfikatory liniowe – problem XOR




Na rysunku zaznaczono podział realizowany przez pierwszą warstwę



# Klasyfikatory liniowe – problem XOR




na rysunku zaznaczono podział realizowany przez drugą warstwę



# BUDOWA SIECI WIELOWARSTWOWEJ

- Rozpatrzmy jednokierunkową sieć neuronową, złożoną z warstw pojedynczych neuronów, w ten sposób, że wyjścia neuronów warstwy poprzedniej tworzą wektor podawany na wejście każdego z neuronów warstwy następnej
- Neurony każdej warstwy mają zawsze tę samą liczbę wejść równą liczbie neuronów warstwy poprzedniej.
- W ramach jednej warstwy neurony nie mają połączeń między sobą.



# BUDOWA SIECI WIELOWARSTWOWEJ

- Niech $x = [x_1, \ldots, x_p]^T$ będzie klasyfikowanym obiektem. Wektor ten rozszerzony o dodatkową składową równą 1 jest podawany na wejście sieci, w związku z tym warstwa wejściowa klasyfikatora neuronowego liczy *p*+1 neuronów
- Warstwa wyjściowa liczy *K* neuronów gdzie *K* jest liczbą klas. Odpowiedzą sieci jest zatem pewien wektor $y = [y_1, \ldots, y_K]^T$.
- Funkcje klasyfikujące klasyfikatora neuronowego są postaci

  $$g_j(x) = y_j, \quad j = 1, \ldots, K,$$

  interpretujemy je jako miarę przynależności do klas

## BUDOWA SIECI WIELOWARSTWOWEJ

- Uczenie sieci odbywa się na podstawie zbioru uczącego $V=\{(x_1,d_1),...,(x_N,d_N)\}$, podczas uczenia modyfikowane są wagi pojedynczych neuronów
- Najczęściej wykorzystywaną funkcją aktywacji neuronu jest unipolarna funkcja sigmoidalna:

$$f_\lambda(s_i)=\frac{1}{1+\exp(-\lambda s_i)}$$



## UCZENIE SIECI WIELOWARSTWOWEJ

Modyfikowane są wagi pojedynczych neuronów sieci tak, że nowy wektor wag neuronu wyznaczany jest ze wzoru:

$$w_{k+1}=w_k+\eta\delta X\ ,$$

gdzie $X$ -jest wektorem podanym na wejście neuronu rozszerzonym o dodatkową składową równą 1

$\eta$ -jest współczynnikiem liczbowym decydującym o szybkości uczenia

$\delta$ -różnica między oczekiwaną wartością odpowiedzi sieci $z$, a wartością rzeczywistą $y$ pomnożoną przez wartość pochodnej funkcji aktywacji.



## UCZENIE SIECI WIELOWARSTWOWEJ

W pochodna funkcji aktywacji $f(x)=\dfrac{1}{1+\exp(-x)}$ wyraża się wzorem:

$$f'(x)=(1-\exp(-x))^{-2}\exp(-x)=f(x)(1-f(x))$$

Czyli:

$$\delta=(z-y)y(1-y)\,.$$

Aby zmodyfikować wagi neuronu trzeba znać oczekiwaną wartość odpowiedzi dla neuronu, która to wartość jest znana tylko w przypadku neuronów warstwy wyjściowej.



## UCZENIE SIECI WIELOWARSTWOWEJ

Algorytm wstecznej propagacji błędu pozwala na wykorzystanie wektora wartości $\delta$ wyznaczonych dla ostatniej warstwy do wyznaczenia wektora wartości $\delta$ warstwy przedostatniej, itd. przechodząc od ostatniej warstwy do pierwszej. Otrzymuje się zatem wzór opisujący składowe wektora $\delta$:

$$\delta_{(i,n)}=\left(\sum_k \delta_{(i+1,k)}w_{(k,n)}\right)y_n(1-y_n)\,,$$

gdzie indeks $i$ oznacza numer warstwy ($i$ – aktualna, $i$+1 - następna), $k$ numer neuronu w warstwie następnej , $n$ numer aktualnie rozpatrywanego neuronu ($n$-ta składowa wektora wejściowego dla następnej warstwy).

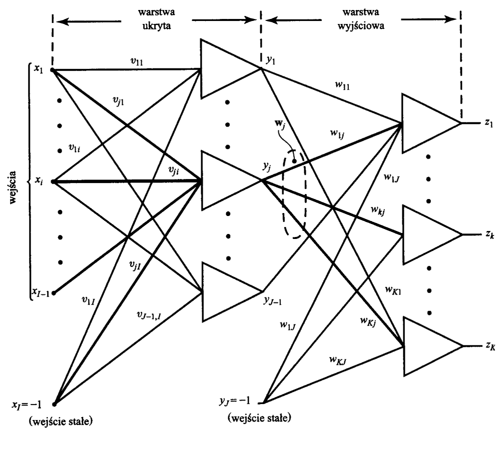


## Uczenie sieci wielowarstwowej

- Zgodnie z regułą delta dla sieci jednowarstwowej korekcja wag odbywa się w oparciu o oczekiwaną odpowiedź sieci oraz wartość uzyskaną.
- W przypadku sieci wielowarstwowych nie znamy oczekiwanej odpowiedzi dla żadnej z warstw z wyjątkiem ostatniej
- Uogólniając regułę delta na przypadek dwuwarstwowy można pokazać, że korekcja wag dochodzących do $j$-tego neuronu w warstwie ukrytej jest proporcjonalna do sumy ważonej wszystkich wartości $\delta$ w warstwie następnej



## Uczenie sieci wielowarstwowej



## Uczenie sieci wielowarstwowej

**Reguła uczenia wstecznej propagacji błędu**

1. Podanie na wejście wektora uczącego $x$ i obliczenie aktualnego wyjścia $y$
2. Obliczenie błędów w warstwie ostatniej na podstawie różnicy otrzymanych wartości wektora $y$ oraz wzorcowych $d$
3. Adaptacja wag od warstwy wejściowej do wyjściowej
4. Obliczenie błędów dla neuronów we wszystkich warstwach wcześniejszych po kolei (jako funkcji błędu warstwy następnej, który jest już znany)
5. Powtarzamy procedurę do momentu kiedy sygnały wyjściowe sieci będą dostatecznie bliskie oczekiwanym



## METODA WSTECZNEJ PROPOAGACJI BŁĘDU

Rozważmy problem klasyfikacji dla $K$ klas.

Niech $x = [x_1, ..., x_{m-1}, -1]^T$ będzie podawanym na wejście wektorem, $W^k$ macierzą wag $k$-tej warstwy neuronów, $y^k$ odpowiedzią $k$-tej warstwy, oznaczmy dodatkowo odpowiedź warstwy ostatniej symbolem $y$.

1. Podaj na wejście wektor $x$
2. Oblicz wektor $\delta \in R^K$ błędów dla ostatniej warstwy

   $\delta_i = (d_i - y_i)\, y_i (1 - y_i)$ (dla neuronu unipolarnego)

   $\delta_i := \frac{1}{2}(d_i - y_i)(1 - y_i^2)$ (dla neuronu bipolarnego)

   dla $i = 1, ..., K$



## METODA WSTECZNEJ PROPOAGACJI BŁĘDU

3. Oblicz wektory $\delta^k$ błędów dla każdej z warstw. (indeks $k$ oznacza $k$-tą warstwę)

   $\delta_i^k = y_i (1 - y_i) \sum_j \delta_j^{k+1} w_{ji}^{k+1}$ (dla neuronu unipolarnego)

   $\delta_i^k := \frac{1}{2}(1 - y_i^2) \sum_j \delta_j^{k+1} w_{ji}^{k+1}$ (dla neuronu bipolarnego)

4. Uaktualnij wartości wag wszystkich warstw zgodnie ze wzorem

   $w_{ij}^k := w_{ij}^k + \eta \delta_i^k y_j^{k-1}$, przy czym $y^0 = x$



## Sieć wielowarstwowa jako nieliniowy aproksymator

- Wielowarstwowa sieć neuronowa z ciągłymi funkcjami aktywacji może być wykorzystana do aproksymacji dowolnej funkcji wielu zmiennych
- Uczenie sieci odbywa się na podstawie zbioru uczącego znanych wartości funkcji (uczenie nadzorowane)
- Zakładamy, że zadanie polega na aproksymacji funkcji $p$ zmiennych $h(x)$. Dany jest zbiór uczący

  $V = \{(x_1, h(x_1)), ..., (x_N, h(x_N))\}$, $x_i \in R^p$, $h(x_i) \in R$

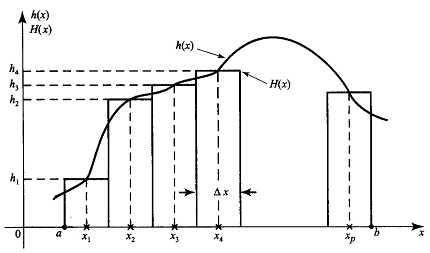


## Sieć wielowarstwowa jako nieliniowy aproksymator

- Idea aproksymacji za pomocą funkcji schodkowej

  $$H(x) = \sum_{i=1}^{N} \left[ \zeta\left( x - x_i + \frac{1}{2}\Delta x \right) - \zeta\left( x - x_i - \frac{1}{2}\Delta x \right) \right] h(x_i),$$

  gdzie $\xi(x) = \begin{cases} 0 & x < 0 \\ 1 & x > 0 \end{cases}$

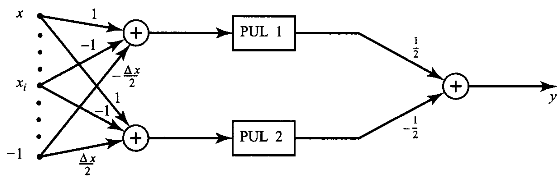





## Sieć wielowarstwowa jako nieliniowy aproksymator

Korzystając z zależności $\xi(x) = \frac{1}{2}\mathrm{sgn}(x) - \frac{1}{2}$ pojedynczy składnik sumy wyrazić można za pomocą wzoru

$\frac{1}{2}\mathrm{sgn}\left( x - x_i + \frac{1}{2}\Delta x \right) - \frac{1}{2}\mathrm{sgn}\left( x - x_i - \frac{1}{2}\Delta x \right)$, przez co może być realizowana za pomocą następującej sieci poniżej



## Sieć wielowarstwowa jako nieliniowy aproksymator

- Sieć realizująca pojedynczą bramkę składa się z dwóch neuronów dyskretnych i węzła sumującego
- $N$ takich układów z węzłem sumującym ich wyjścia pomnożone przez wartości funkcji w odpowiednich punktach $x_i$ jest wystarczająca do realizacji aproksymacji za pomocą funkcji schodkowej
- Zamiast neuronów dyskretnych użyć można neuronów ciągłych z bipolarną funkcją aktywacji, co daje lepsze przybliżenie aproksymowanej funkcji



## Sieć wielowarstwowa jako nieliniowy aproksymator

- Przy aproksymacji funkcji dwu lub więcej zmiennych dziedzina funkcji dzielona jest na odpowiednie kwadraty, kostki itd.
- Pojedyncza bramka w przypadku $p$-wymiarowym realizowana może być za pomocą 2$p$ neuronów i węzła sumującego
- Sieć o strukturze dwuwarstwowej składająca się z 2$Np$ neuronów na pierwszej warstwie oraz $N$+1 oddzielnych węzłów sumujących może modelować dowolną funkcję p zmiennych
- <u>Uwaga:</u> wartości aproksymowanej funkcji muszą być w przedziale [-1,1] dla neuronów bipolarnych na warstwie wyjściowej (normalizacja), bądź gdy normalizacja nie jest możliwa użyć funkcji aktywacji $f(net) = net$



## Sieć wielowarstwowa jako nieliniowy aproksymator

- Przedstawiona idea konstrukcji sieci neuronowej dla zadania aproksymacji jest teoretyczna i nie gwarantuje optymalności uzyskiwanych wyników
- Najczęściej do niepowodzeń uczenia sieci prowadzi
  - Zbyt duża lub zbyt mała liczba neuronów w warstwie ukrytej
  - Brak deterministycznych związków pomiędzy wartościami wektorów wejściowych a oczekiwanymi wyjściami sieci
- Konieczna zatem jest odpowiednia weryfikacja i walidacja uzyskiwanych wyników



## Uczenie sieci - uwagi

- Brak jest ogólnych metod doboru parametrów uczenia sieci neuronowej
- Parametry sieci:
  - Funkcja aktywacji neuronu / parametr funkcji aktywacji
  - Liczba neuronów / liczba warstw – architektura sieci
  - Stała uczenia $\eta$
- Uaktualniania wag można dokonywać również raz na cykl uczący (nie po podaniu każdej próbki) zgodnie ze wzorem

  $\Delta w = \sum_{i=1}^{N} \Delta w_i$ , gdzie $\Delta w_i$ jest poprawką obliczoną po podaniu *i*-tego wektora uczącego



## Uczenie sieci - uwagi

- Gradientowe metody uczenia sieci, takie jak metoda propagacji wstecznej błędu zbieżne są do minimum lokalnego
- Osiągnięcie minimum lokalnego zależy od wyboru początkowych wartości wag oraz stałej uczenia $\eta$

  Rozwiązanie problemu minimum lokalnego
  1. Powtórzyć proces uczenia kilkakrotnie porównując otrzymywane wyniki dla różnych wartości wag początkowych i stałej uczenia $\eta$
  2. Dodanie do wag w każdym kroku uczącym losowego wektora małych wartości
  3. Dodawanie zakłóceń do wektorów uczących
  4. Losowe podawanie wektorów uczących



## Uczenie sieci - uwagi

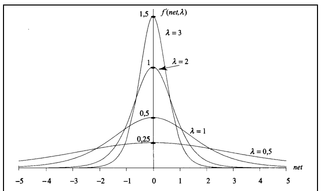
Ad2.
Metoda jest pewnym wariantem algorytmu **szukania przypadkowego**

- Dodawanie do aktualnego wektora wag wektora losowego tak długo aż do osiągnięcie błędu mniejszego niż pierwotnie

Ad4.

- Losowe podawanie wektorów uczących powoduje wyeliminowanie cykliczności pojawiania się wzorców, która może prowadzić do znoszenia się kolejnych korekt wag i zahamowania zbieżności

## Uczenie sieci - uwagi



**Pochodna bipolarnej funkcji aktywacji**

Współczynnik korekcji wag $\eta$

- Nie istnieje żadna metoda optymalnego doboru wartości współczynnika $\eta$. Literaturze na ogół stosowane są wartości w przedziale od 0,001 do 10.
- Można również stosować zmienne wartości współczynników korekcji wag – zwiększać o stałą gdy błąd systematycznie spada oraz zmniejszać geometrycznie gdy błąd rośnie.



## Uczenie sieci - uwagi

Ilość neuronów w warstwie ukrytej

- Nie istnieje żadna ogólna metoda optymalnego doboru ilości neuronów w warstwie ukrytej
- Sieci z pojedynczą warstwą ukrytą są w stanie wygenerować dowolne obszary decyzyjne w *p*-wymiarowej przestrzeni wektorów wejściowych
- Liczba neuronów w warstwie ukrytej zależy od wymiaru wejściowej przestrzeni cech i liniowo separowanych obszarów decyzyjnych



## Uczenie sieci - uwagi

Niech dana jest przestrzeń cech *p* wymiarowa (nierozszerzona o dodatkową składową -1) liniowo separowalna na *M* obszarów decyzyjnych, gdzie $K \le M$, *K* – liczba klas, *J* – ilość neuronów w warstwie ukrytej

1. Wybór $J = N$ gwarantuje poprawną klasyfikację wszystkich wektorów zbioru uczącego
2. Maksymalna liczba obszarów na które *J* hiperpłaszczyzn może podzielić przestrzeń $R^p$ (Mirchandini, Cao, 1989):

$$M(J,p) = \sum_{k=0}^{p} \binom{J}{k}, \quad \binom{J}{k} = 0 \text{ dla } J < k$$



## Uczenie sieci - uwagi

<u>Przykład XOR.</u>

- Do rozwiązania problemu XOR wystarczy podział na $M = 3$ obszary decyzyjne, dlatego dla $p = 2$ dostajemy $J = 2$ neurony w warstwie ukrytej, bo $M(2,2) = \sum_{k=0}^{p} \binom{2}{k} = 3$.



## Sieci neuronowe z połączeniami funkcjonalnymi

- Sieci wielowarstwowe pozwalają na klasyfikację w przypadkach liniowo nieseparowalnych
  - Pierwsza warstwa dokonuje podziału przestrzeni cech na obszary decyzyjne za hiperpłaszczyzn
  - Druga warstwa dokonuje klasyfikacji
- Innym sposobem realizacji klasyfikacji w przypadku liniowo nieseparowalnym jest użycie sieci z połączeniami funkcjonalnymi (Pao 1989)
  - Sieć składa się tylko z pojedynczej warstwy neuronów
  - Rozszerzona zostaje wejściowa przestrzeń cech
  - Do uczenia wystarcza prosta reguła delta



## Sieci neuronowe z połączeniami funkcjonalnymi

Model tensorowy

- Dodatkowe składowe tworzone są w postaci iloczynów $x_i x_j$, gdzie $i, j \in \{1,...,p\}$, $i < j$
- Można również dołożyć kolejne składowe $x_i x_j x_k$, $i, j, k \in \{1,...,p\}$, $i < j < k$

## Sieci neuronowe z połączeniami funkcjonalnymi

Model funkcjonalny

- Dodatkowe składowe tworzone są na podstawie ciągu funkcji ortogonalnych
- Do rozszerzenia pojedynczego wejścia $x_i$ zastosować można ciąg funkcji $\sin \pi x,\ \cos \pi x,\ \sin 2\pi x,\ \cos 2\pi x, \dots$, co pozwala na aproksymację funkcji jednej zmiennej
- Przy rozszerzeniu za pomocą ciągu funkcji trygonometrycznych aproksymacja za pomocą SSN jest analogiczna do aproksymacji z wykorzystaniem szeregu Fouriera ze skończoną liczbą wyrazów
- Dobre rezultaty można uzyskiwac już przy rozszerzeniu o kilka składowych (Pao 1989)



## Sieci neuronowe z połączeniami funkcjonalnymi

- Uczenie sieci z połączeniami funkcjonalnymi jest znacznie prostsze od stosowania metody propagacji wstecznej błędu dla sieci wielowarstwowych
- Zbieżność uczenie może być znacznie szybsza niż w przypadku sieci wielowarstwowych
- Brak jest ogólnych porównań działania sieci nauczonych metodą propagacji wstecznej błędu do sieci z połączeniami funkcjonalnymi