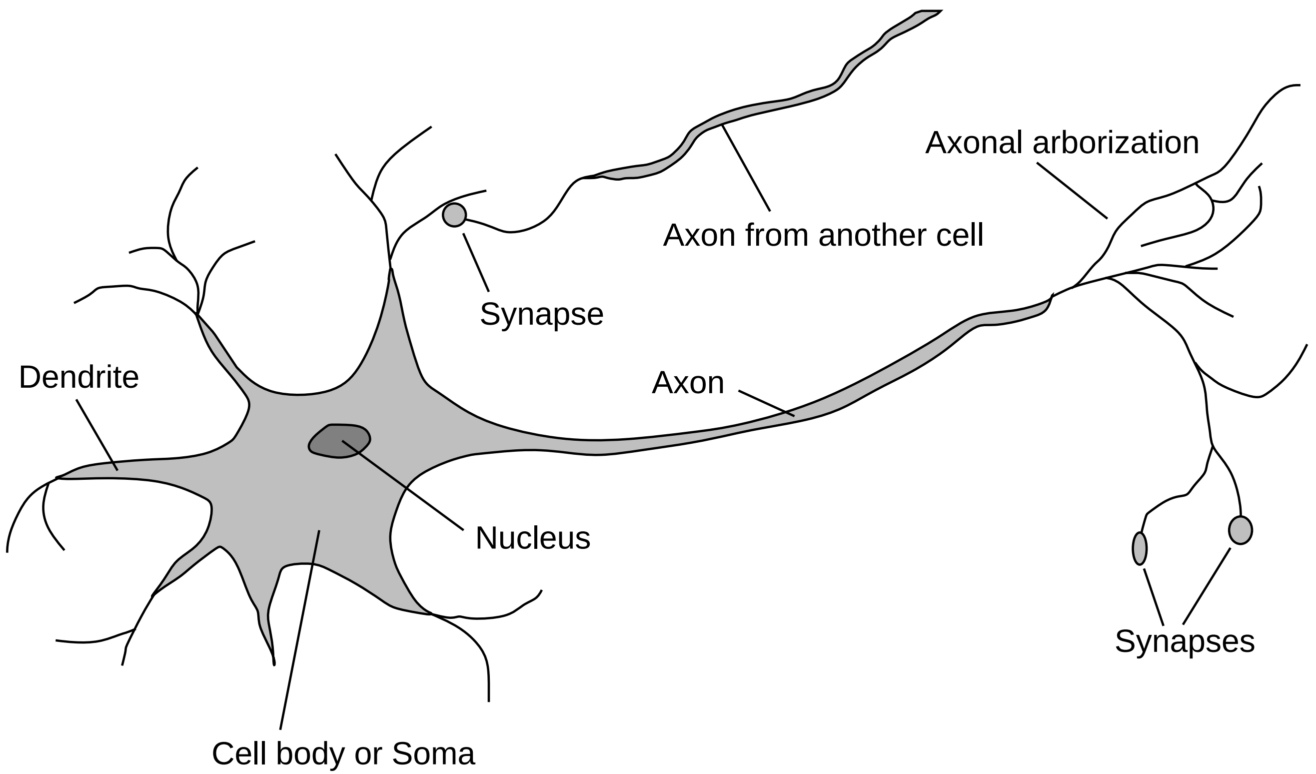
Sztuczna Inteligencja i Systemy Doradcze

Sieci neuronowe

Sieci neuronowe: pomysl

Naśladowanie mózgu działającego jako sieć komórek neuronowych



Sygnały to zaszumione "bodźce trenujące" poziom potencjału elektrycznego komórek

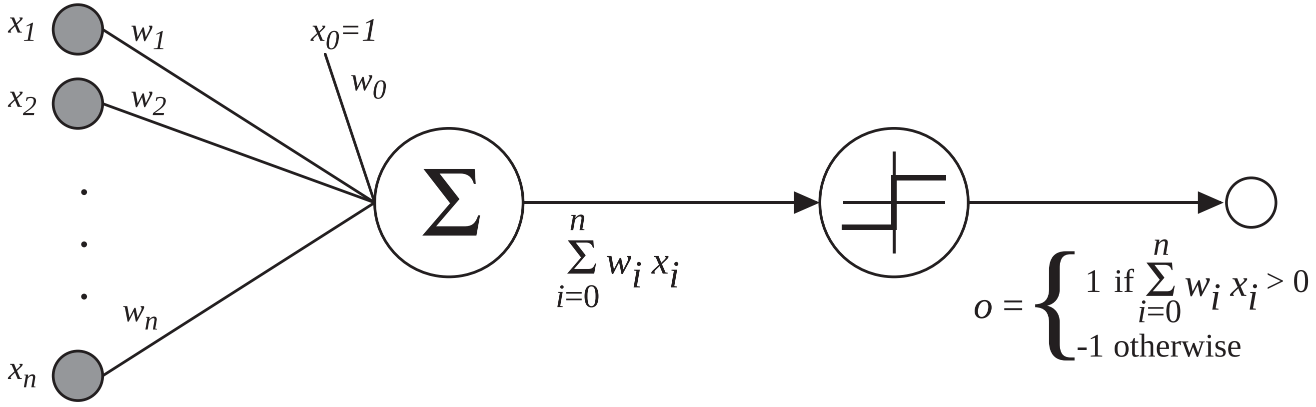
Sieci neuronowe: sztuczne i naturalne

	Komputer	Mózg
Jednostki obliczeniowe	1 CPU 10^8 bramek logicznych	10^{11} neuronów > 20 typów
Jednostki pamięciowe	10^{10} bitów RAM 10^{11} bitów dysku	10^{11} neuronów 10^{14} synaps
Czas cyklu	1 ns (10^{-9} sek.)	1–10 ms (10^{-3} sek.)
Szerokość pasma	10^{10} bitów/sek	10^{14} bitów/sek
Zmiany stanów/sek	10^9	10^{14}

Równoległość daje wciąż umysłowi ludzkiemu ogromną przewagę nad komputerem pomimo dużo wolniejszego czasu przetwarzania informacji

Perceptron

$\vec{x} = (x_1, x_2, \ldots, x_n)$ — wektor wejściowy (obiekt danych)



$\vec{w} = (w_0, w_1, \ldots, w_n)$ — wektor wag perceptronu
w_0 – waga przesunięcia ("przesuwa" próg funkcji aktywacji)

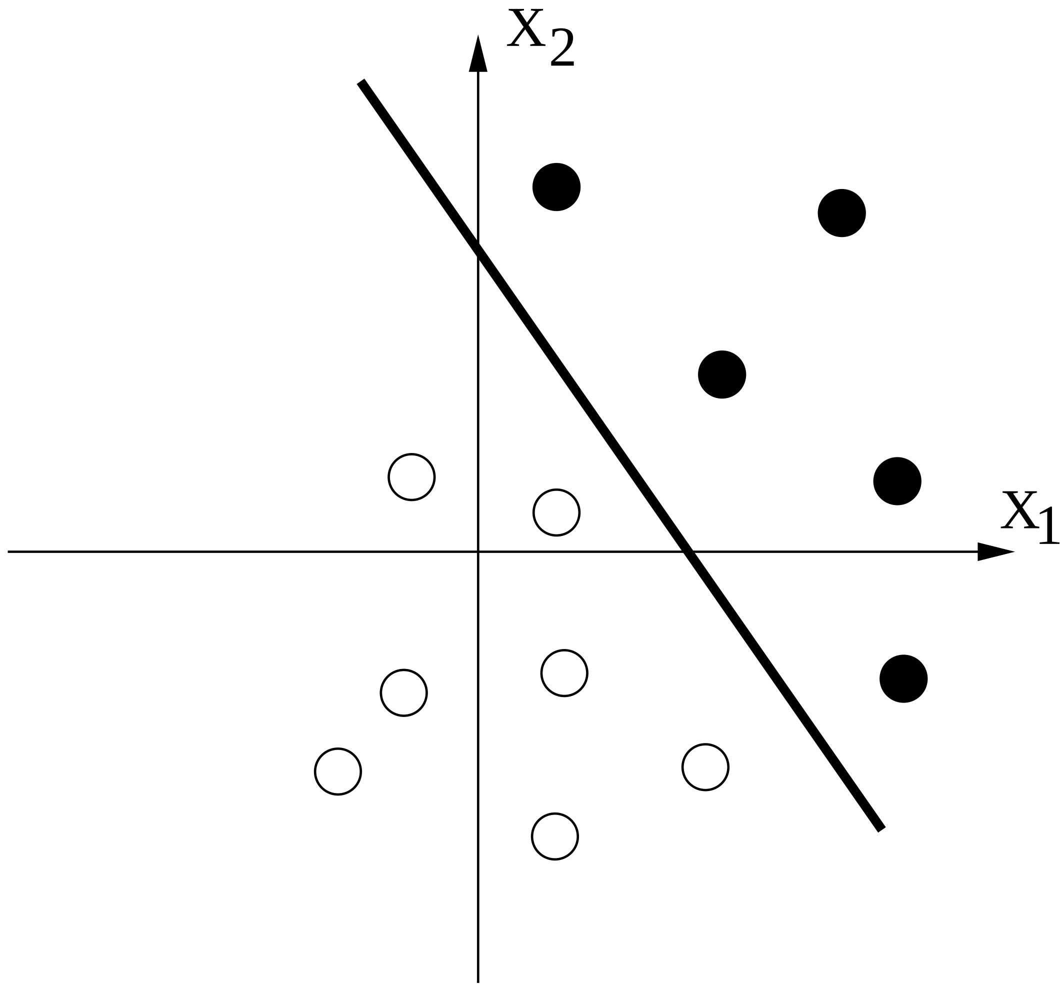
σ – progowa (skokowa) funkcja aktywacji perceptronu

$o(\vec{x})$ – wartość wyjścia perceptronu dla wektora $\vec{x}$

$$o(\vec{x}) = \sigma(\vec{w} \cdot \vec{x}) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + \cdots + w_n x_n > 0 \\ -1 & \text{otherwise.} \end{cases}$$

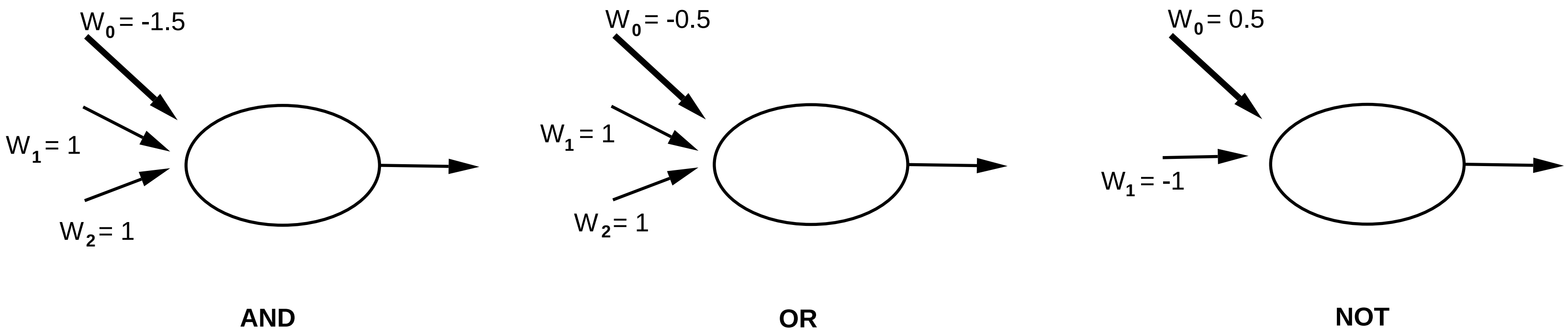
Perceptron: wyrazalnosc

Perceptron reprezentuje liniowe cięcie w przestrzeni wejść



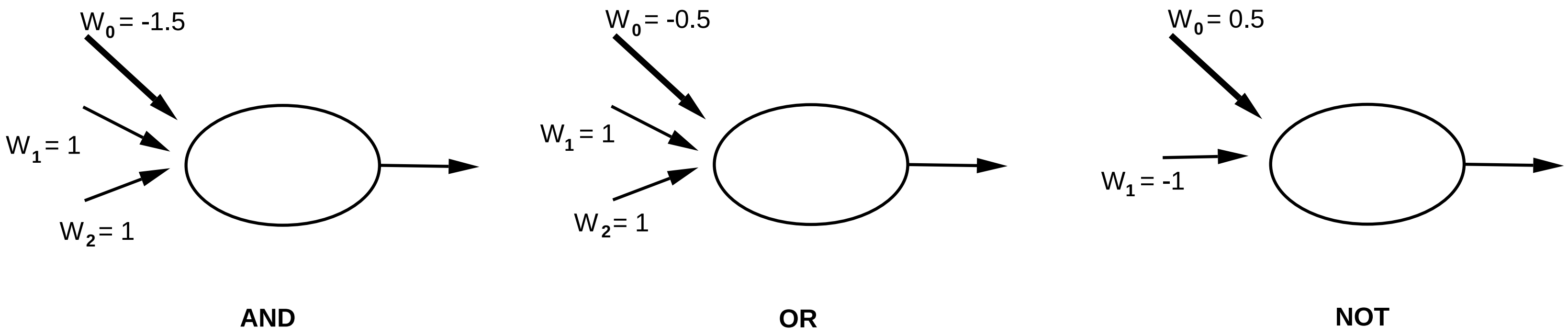
Perceptron: wyrazalnosc

Można wyrazić funkcje logiczne AND, OR, NOT

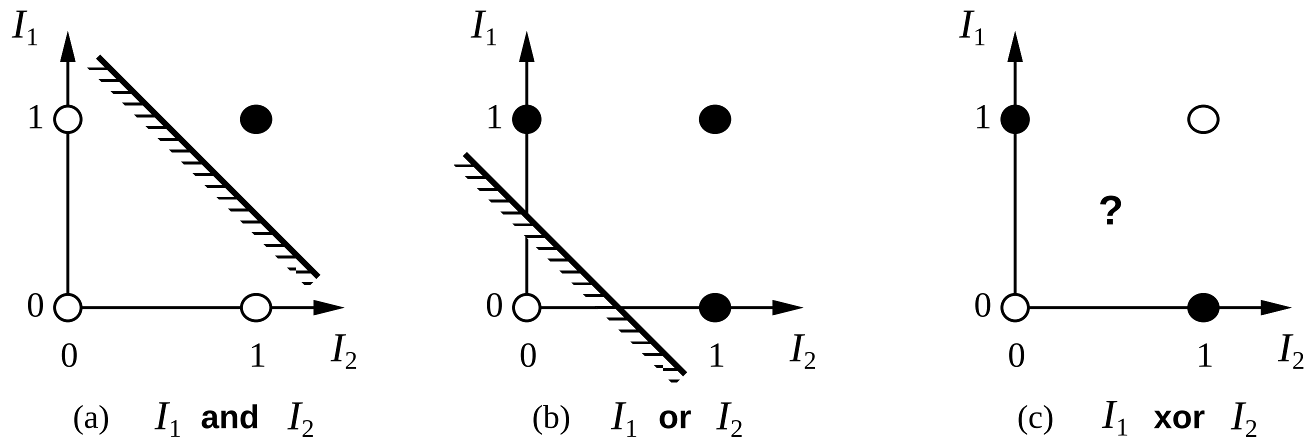


Perceptron: wyrazalnosc

Można wyrazić funkcje logiczne AND, OR, NOT



ale nie da się dobrać wag do funkcji XOR



Uczenie perceptronu: algorytm

function PERCEPTRON-LEARN(*perceptron*, *examples*, α) **returns** a perceptron
  inputs: *examples*, a set of examples, each with input $\vec{x}$ and output $y(\vec{x})$
    perceptron, a perceptron with weights $\vec{w} = (w_0, \ldots, w_n)$
    α, the learning rate

  repeat
    for each $\vec{x} = (x_1, \ldots, x_n)$ **in** *examples* **do**
      $\Delta \vec{w} \leftarrow \alpha \, (y(\vec{x}) - \vec{w} \cdot \vec{x})\vec{x}$
      $\vec{w} \leftarrow \vec{w} + \Delta \vec{w}$
    end
  until some stopping criterion is satisfied
  return *perceptron*

Uczenie perceptronu: wlasnosci

Twierdzenie 1

Jeśli zbiór danych jest liniowo separowalny
a współczynnik szybkości uczenia α wystarczająco mały
$\Rightarrow$ algorytm uczenia perceptronu jest zbieżny

Uczenie perceptronu: wlasnosci

Twierdzenie 1

Jeśli zbiór danych jest liniowo separowalny
a współczynnik szybkości uczenia α wystarczająco mały
$\Rightarrow$ algorytm uczenia perceptronu jest zbieżny

Twierdzenie 2

Jeśli zbiór danych nie jest liniowo separowalny
$\Rightarrow$ algorytm zbiega lokalnie do minimalnego błędu średniokwadratowego

Zbieznosc uczenia perceptronu

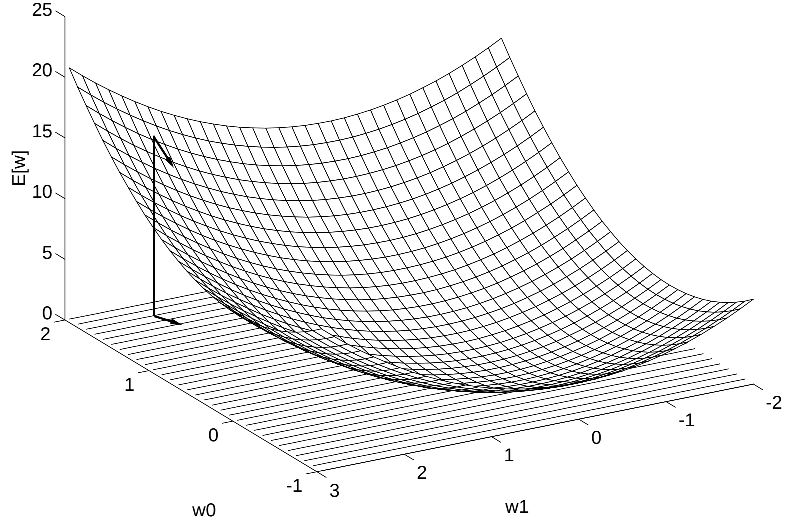
Błąd średniokwadratowy dla zbioru treningowego U

$$E[\vec{w}] = \frac{1}{2}\Sigma_{\vec{x}\in U}(y(\vec{x}) - \vec{w}\cdot\vec{x})^2$$

Zbieznosc uczenia perceptronu

Błąd średniokwadratowy dla zbioru treningowego U

$$E[\vec{w}] = \frac{1}{2}\Sigma_{\vec{x}\in U}(y(\vec{x}) - \vec{w}\cdot\vec{x})^2$$



Gradient błędu średniokwadratowego

$$\nabla E[\vec{w}] = \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \cdots \frac{\partial E}{\partial w_n}\right]$$

$\Rightarrow$ wskazuje kierunek, w którym błąd $E[\vec{w}]$ rośnie

Zbieznosc uczenia perceptronu

Błąd średniokwadratowy $E[\vec{w}] = \frac{1}{2}\sum_{\vec{x}\in U}(y(\vec{x}) - \vec{w}\cdot\vec{x})^2$

$$\frac{\partial E}{\partial w_i} =$$

Zbieznosc uczenia perceptronu

Błąd średniokwadratowy $E[\vec{w}] = \frac{1}{2}\sum_{\vec{x} \in U}(y(\vec{x}) - \vec{w} \cdot \vec{x})^2$

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i}\frac{1}{2}\sum_{\vec{x} \in U}(y(\vec{x}) - \vec{w} \cdot \vec{x})^2$$

Zbieznosc uczenia perceptronu

Błąd średniokwadratowy $E[\vec{w}] = \frac{1}{2}\sum_{\vec{x}\in U}(y(\vec{x}) - \vec{w}\cdot\vec{x})^2$

$$\begin{aligned}\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i}\frac{1}{2}\sum_{\vec{x}\in U}(y(\vec{x}) - \vec{w}\cdot\vec{x})^2 \\ &= \frac{1}{2}\sum_{\vec{x}\in U}\frac{\partial}{\partial w_i}(y(\vec{x}) - \vec{w}\cdot\vec{x})^2\end{aligned}$$

Zbieznosc uczenia perceptronu

Błąd średniokwadratowy $E[\vec{w}] = \frac{1}{2}\sum_{\vec{x}\in U}(y(\vec{x}) - \vec{w}\cdot\vec{x})^2$

$$
\begin{aligned}
\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i}\frac{1}{2}\sum_{\vec{x}\in U}(y(\vec{x}) - \vec{w}\cdot\vec{x})^2 \\
&= \frac{1}{2}\sum_{\vec{x}\in U}\frac{\partial}{\partial w_i}(y(\vec{x}) - \vec{w}\cdot\vec{x})^2 \\
&= \frac{1}{2}\sum_{\vec{x}\in U}2(y(\vec{x}) - \vec{w}\cdot\vec{x})\frac{\partial}{\partial w_i}(y(\vec{x}) - \vec{w}\cdot\vec{x})
\end{aligned}
$$

Zbieznosc uczenia perceptronu

Błąd średniokwadratowy $E[\vec{w}] = \frac{1}{2}\sum_{\vec{x}\in U}(y(\vec{x}) - \vec{w}\cdot\vec{x})^2$

$$
\begin{aligned}
\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i}\frac{1}{2}\sum_{\vec{x}\in U}(y(\vec{x}) - \vec{w}\cdot\vec{x})^2 \\
&= \frac{1}{2}\sum_{\vec{x}\in U}\frac{\partial}{\partial w_i}(y(\vec{x}) - \vec{w}\cdot\vec{x})^2 \\
&= \frac{1}{2}\sum_{\vec{x}\in U}2(y(\vec{x}) - \vec{w}\cdot\vec{x})\frac{\partial}{\partial w_i}(y(\vec{x}) - \vec{w}\cdot\vec{x}) \\
&= \sum_{\vec{x}\in U}(y(\vec{x}) - \vec{w}\cdot\vec{x})\frac{\partial}{\partial w_i}(y(\vec{x}) - \vec{w}\cdot\vec{x})
\end{aligned}
$$

Zbieznosc uczenia perceptronu

Błąd średniokwadratowy $E[\vec{w}] = \frac{1}{2}\sum_{\vec{x}\in U}(y(\vec{x}) - \vec{w}\cdot\vec{x})^2$

$$\begin{aligned}
\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i}\frac{1}{2}\sum_{\vec{x}\in U}(y(\vec{x}) - \vec{w}\cdot\vec{x})^2 \\
&= \frac{1}{2}\sum_{\vec{x}\in U}\frac{\partial}{\partial w_i}(y(\vec{x}) - \vec{w}\cdot\vec{x})^2 \\
&= \frac{1}{2}\sum_{\vec{x}\in U}2(y(\vec{x}) - \vec{w}\cdot\vec{x})\frac{\partial}{\partial w_i}(y(\vec{x}) - \vec{w}\cdot\vec{x}) \\
&= \sum_{\vec{x}\in U}(y(\vec{x}) - \vec{w}\cdot\vec{x})\frac{\partial}{\partial w_i}(y(\vec{x}) - \vec{w}\cdot\vec{x}) \\
&= \sum_{\vec{x}\in U}(y(\vec{x}) - \vec{w}\cdot\vec{x})(-x_i)
\end{aligned}$$

Zbieznosc uczenia perceptronu

Błąd średniokwadratowy $E[\vec{w}] = \frac{1}{2}\sum_{\vec{x}\in U}(y(\vec{x}) - \vec{w}\cdot\vec{x})^2$

$$\begin{aligned}
\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i}\frac{1}{2}\sum_{\vec{x}\in U}(y(\vec{x}) - \vec{w}\cdot\vec{x})^2 \\
&= \frac{1}{2}\sum_{\vec{x}\in U}\frac{\partial}{\partial w_i}(y(\vec{x}) - \vec{w}\cdot\vec{x})^2 \\
&= \frac{1}{2}\sum_{\vec{x}\in U}2(y(\vec{x}) - \vec{w}\cdot\vec{x})\frac{\partial}{\partial w_i}(y(\vec{x}) - \vec{w}\cdot\vec{x}) \\
&= \sum_{\vec{x}\in U}(y(\vec{x}) - \vec{w}\cdot\vec{x})\frac{\partial}{\partial w_i}(y(\vec{x}) - \vec{w}\cdot\vec{x}) \\
&= \sum_{\vec{x}\in U}(y(\vec{x}) - \vec{w}\cdot\vec{x})(-x_i) \\
&= \sum_{\vec{x}\in U} - (y(\vec{x}) - \vec{w}\cdot\vec{x})x_i
\end{aligned}$$

Zbieznosc uczenia perceptronu

Błąd średniokwadratowy $E[\vec{w}] = \frac{1}{2}\sum_{\vec{x}\in U}(y(\vec{x}) - \vec{w}\cdot\vec{x})^2$

$$\begin{aligned}
\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i}\frac{1}{2}\sum_{\vec{x}\in U}(y(\vec{x}) - \vec{w}\cdot\vec{x})^2 \\
&= \frac{1}{2}\sum_{\vec{x}\in U}\frac{\partial}{\partial w_i}(y(\vec{x}) - \vec{w}\cdot\vec{x})^2 \\
&= \frac{1}{2}\sum_{\vec{x}\in U}2(y(\vec{x}) - \vec{w}\cdot\vec{x})\frac{\partial}{\partial w_i}(y(\vec{x}) - \vec{w}\cdot\vec{x}) \\
&= \sum_{\vec{x}\in U}(y(\vec{x}) - \vec{w}\cdot\vec{x})\frac{\partial}{\partial w_i}(y(\vec{x}) - \vec{w}\cdot\vec{x}) \\
&= \sum_{\vec{x}\in U}(y(\vec{x}) - \vec{w}\cdot\vec{x})(-x_i) \\
&= \sum_{\vec{x}\in U} - (y(\vec{x}) - \vec{w}\cdot\vec{x})x_i
\end{aligned}$$

Stąd $\nabla E[\vec{w}] = \sum_{\vec{x}\in U} - (y(\vec{x}) - \vec{w}\cdot\vec{x})\vec{x}$

Zbieznosc uczenia perceptronu

Gradient $\nabla E[\vec{w}] = \sum_{\vec{x} \in U} -(y(\vec{x}) - \vec{w} \cdot \vec{x})\vec{x}$ wskazuje kierunek, w którym błąd średniokwadratowy $E[\vec{w}]$ rośnie

Zbieznosc uczenia perceptronu

Gradient $\nabla E[\vec{w}] = \sum_{\vec{x} \in U} -(y(\vec{x}) - \vec{w} \cdot \vec{x})\vec{x}$ wskazuje kierunek, w którym błąd średniokwadratowy $E[\vec{w}]$ rośnie

$\Rightarrow$ wagi perceptronu są poprawiane w kierunku
dokładnie przeciwnym do gradientu $-\nabla E[\vec{w}]$

Zbieznosc uczenia perceptronu

Gradient $\nabla E[\vec{w}] = \sum_{\vec{x} \in U} -(y(\vec{x}) - \vec{w} \cdot \vec{x})\vec{x}$ wskazuje kierunek, w którym błąd średniokwadratowy $E[\vec{w}]$ rośnie

$\Rightarrow$ wagi perceptronu są poprawiane w kierunku
dokładnie przeciwnym do gradientu $-\nabla E[\vec{w}]$

Stąd w algorytmie uczenia perceptronu:

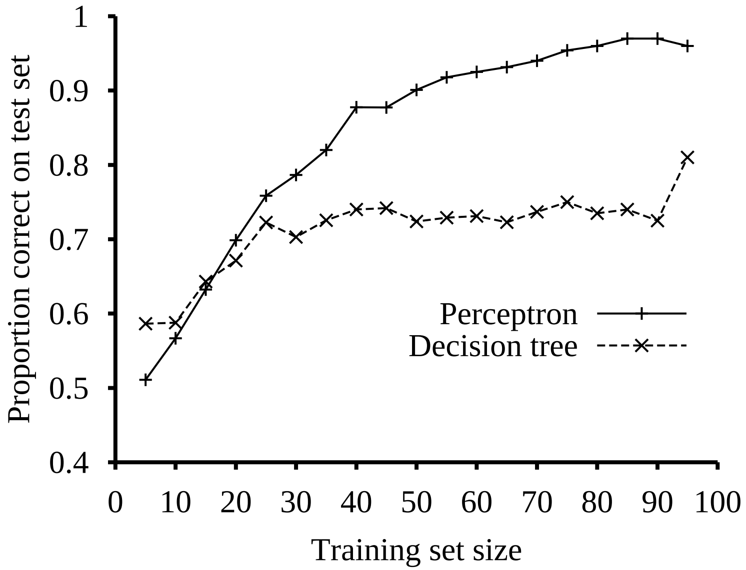
$$\Delta\vec{w} \leftarrow \alpha(y(\vec{x}) - \vec{w} \cdot \vec{x})\vec{x}$$

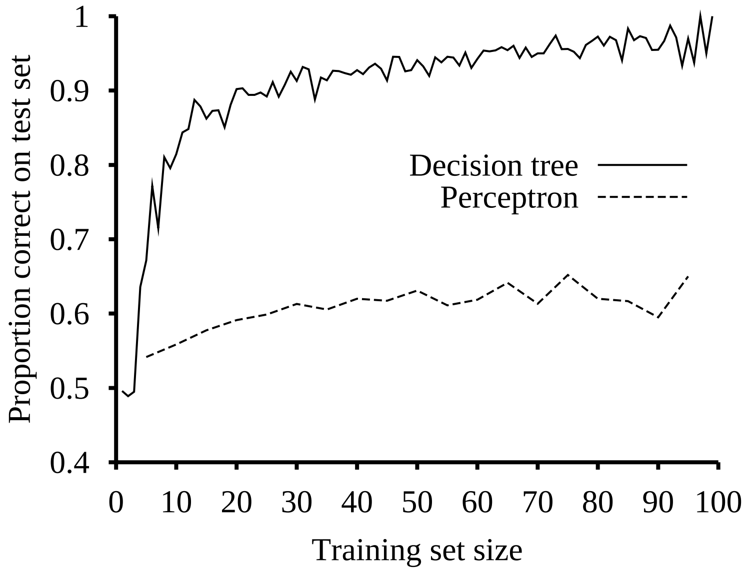
$$\vec{w} \leftarrow \vec{w} + \Delta\vec{w}$$

Porownanie perceptronu i drzewa decyzyjnego

Funkcji większości (> połowa bitów = 1)
lepiej wyuczalna przez perceptron

Decyzja o wstąpieniu do restauracji
lepiej wyuczalna przez drzewo decyzyjne



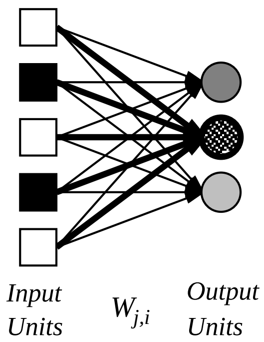


Plaska siec perceptronow

Reprezentuje funkcję wektorową

Poszczególne perceptrony są niezależne

$\Rightarrow$ trzeba trenować każdy perceptron oddzielnie



Wielowarstwowa siec neuronowa

Jednostki:

Jednostki podzielone są na warstwy, każda jednostka przyporządkowana jest do dokładnie jednej warstwy

Wejścia:

Wejścia podłączone są wyłącznie do jednostek znajdujących się w najniższej warstwie

Połączenia:

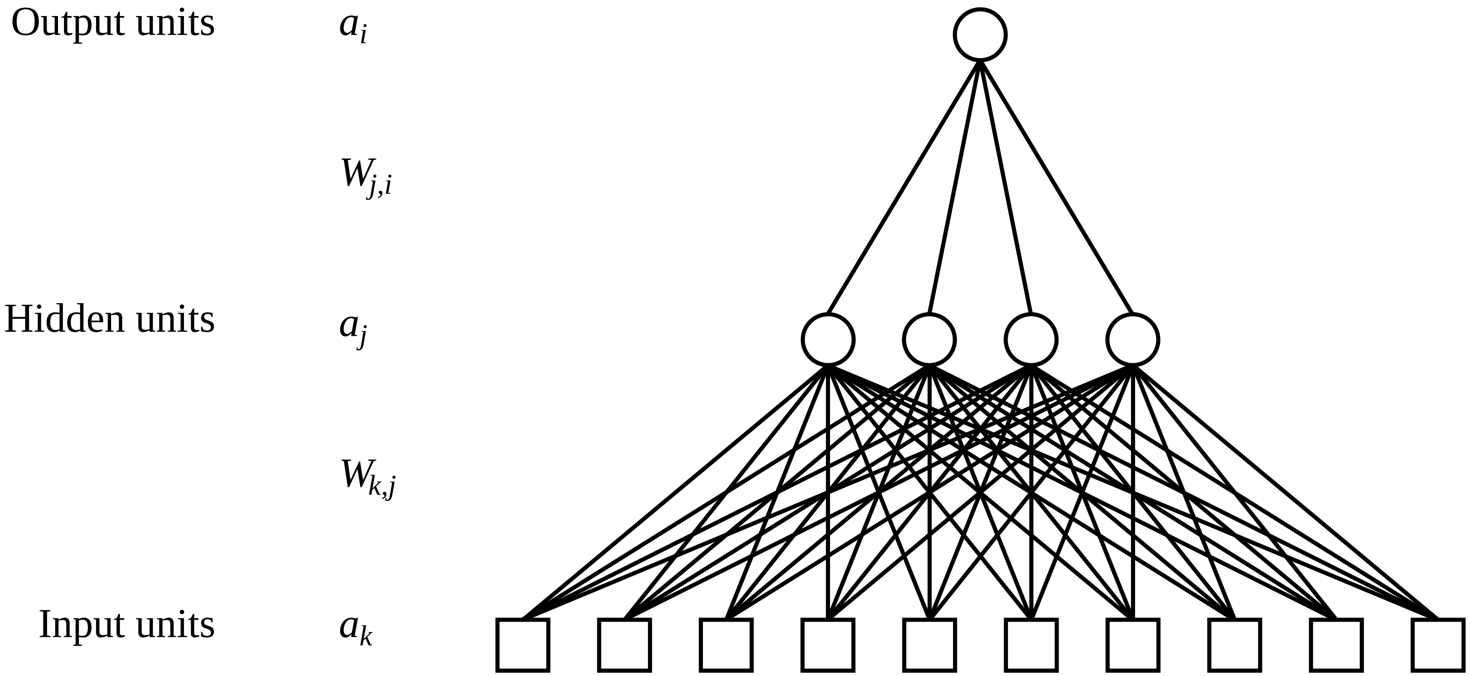
Połączenia występują wyłącznie pomiędzy jednostkami z sąsiednich warstw, łączą zawsze wyjścia jednostek z warstwy niższej z wejścami do jednostek w warstwie wyższej

Wyjście:

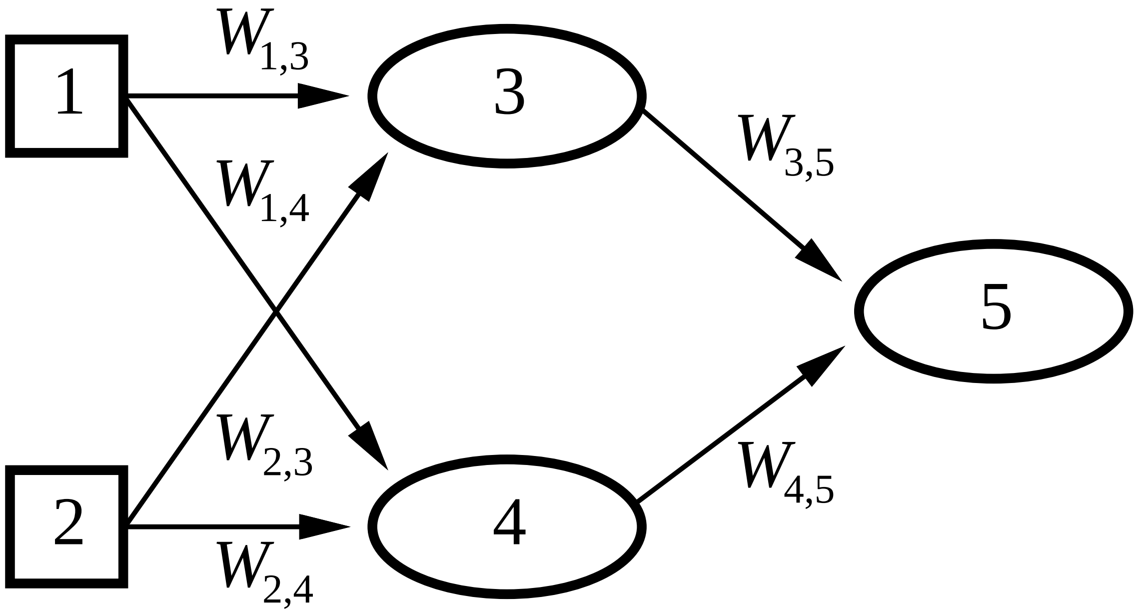
Typowa sieć z jedną wartością funkcji ma tylko jedną jednostkę w najwyższej warstwie, wyjście z tej jednostki jest wyjściem całej sieci

Wielowarstwowa siec neuronowa: przyklad

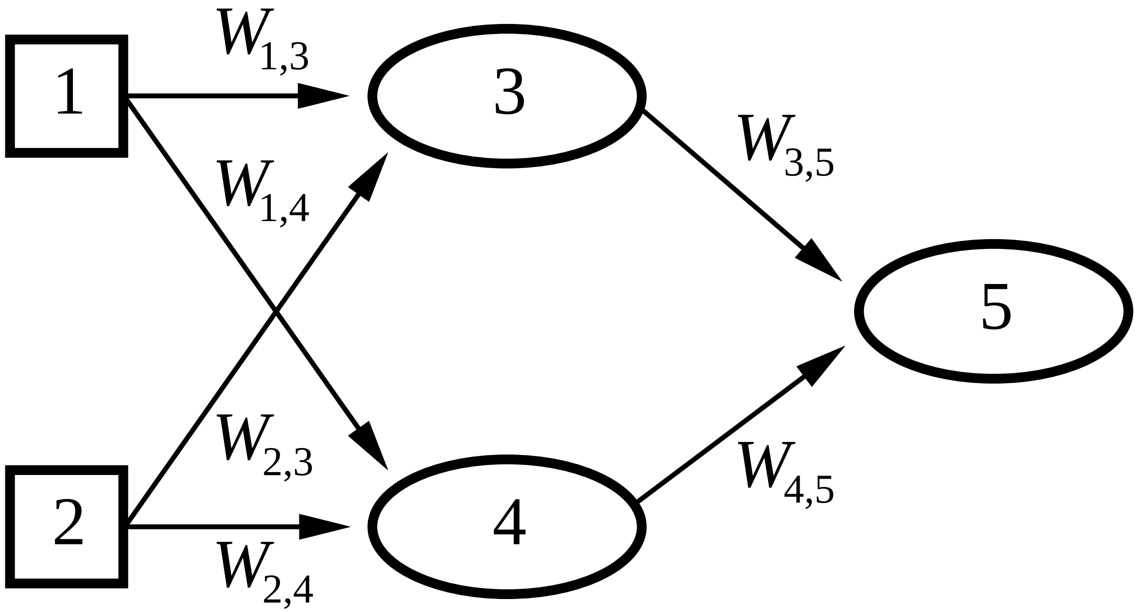
2 warstwy, 10 wejść, 4 neurony ukryte (w warstie wewnętrznej)



Wielowarstwowa siec neuronowa: ewaluacja



Wielowarstwowa siec neuronowa: ewaluacja



$$\begin{aligned} x_5 &= \sigma(w_{3,5} \cdot x_3 + w_{4,5} \cdot x_4) \\ &= \sigma(w_{3,5} \cdot \sigma(w_{1,3} \cdot x_1 + w_{2,3} \cdot x_2) + w_{4,5} \cdot \sigma(w_{1,4} \cdot x_1 + w_{2,4} \cdot x_2)) \end{aligned}$$

Wielowarstwowa siec neuronowa: uczenie

Problem

Progowa funkcja aktywacji jest nieciągła i nieróżniczkowalna
$\Rightarrow$ dla jednostek wewnętrznych nie można wyprowadzić
gradientowej reguły poprawiania wag gwarantującej zbieżność uczenia

Wielowarstwowa siec neuronowa: uczenie

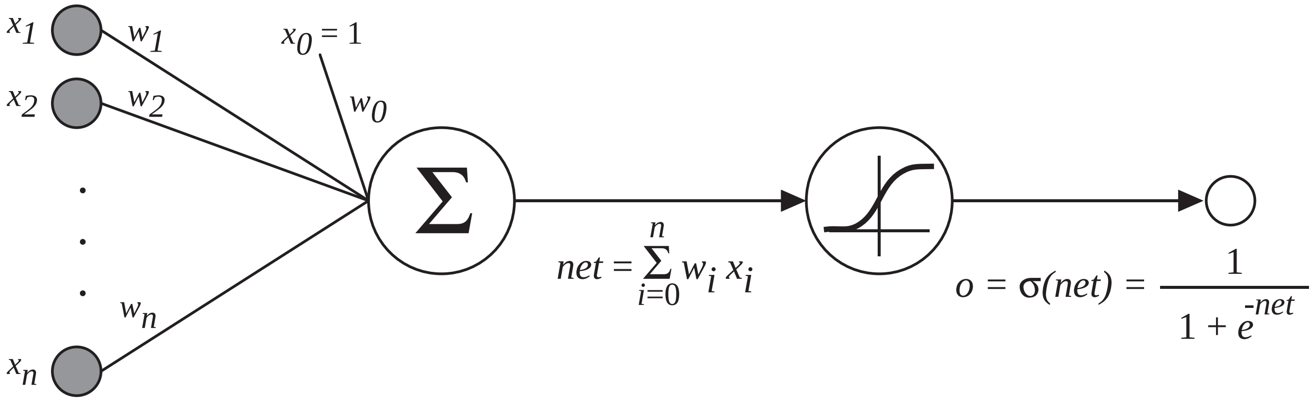
Problem

Progowa funkcja aktywacji jest nieciągła i nieróżniczkowalna
$\Rightarrow$ dla jednostek wewnętrznych nie można wyprowadzić
gradientowej reguły poprawiania wag gwarantującej zbieżność uczenia

Rozwiązanie

Zastosowanie różniczkowalnej funkcji aktywacji

Perceptron z sigmoidalna funkcja aktywacji

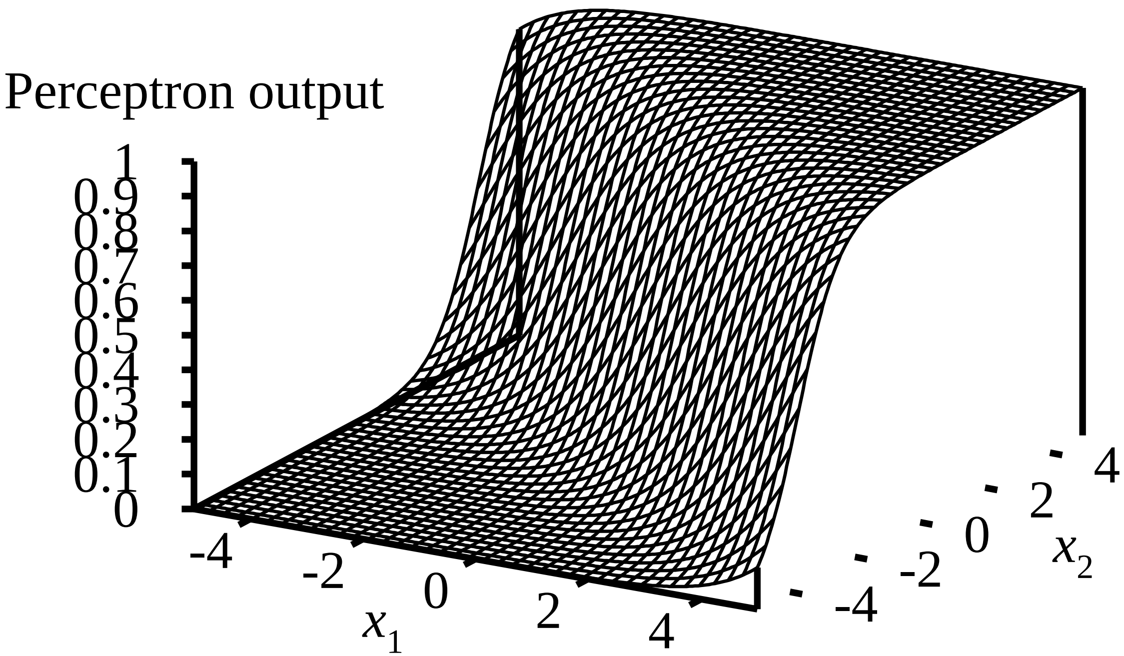


Sigmoidalna funkcja aktywacji perceptronu $\sigma(z) = \frac{1}{1+e^{-z}}$

$$o(\vec{x}) = \sigma(\vec{w} \cdot \vec{x}) = \frac{1}{1 + e^{-\vec{w} \cdot \vec{x}}}$$

Perceptron z sigmoidalna funkcja aktywacji

Funkcja wyjścia dla pojedycznego perceptronu z sigmoidalną funkcją aktywacji i 2 wejściami:

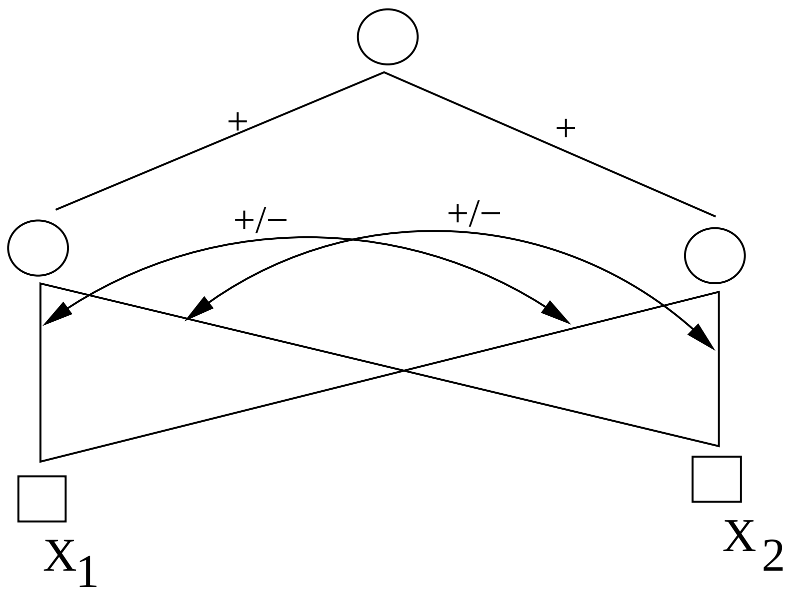


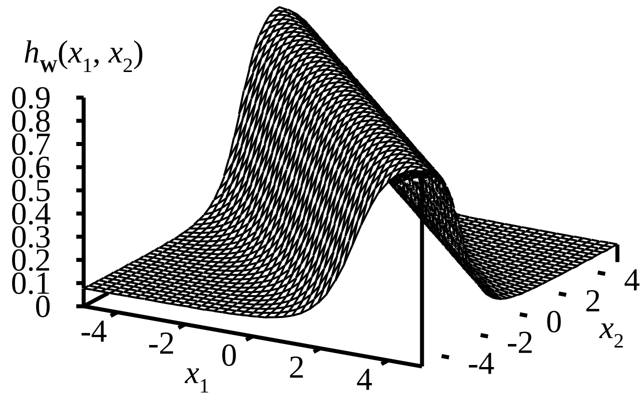
Wielowarstwowa siec neuronowa: przyklad

Liczba wejść: 2, liczba warstw: 2

Warstwa ukryta: 2 perceptrony skierowane przeciwnie do siebie

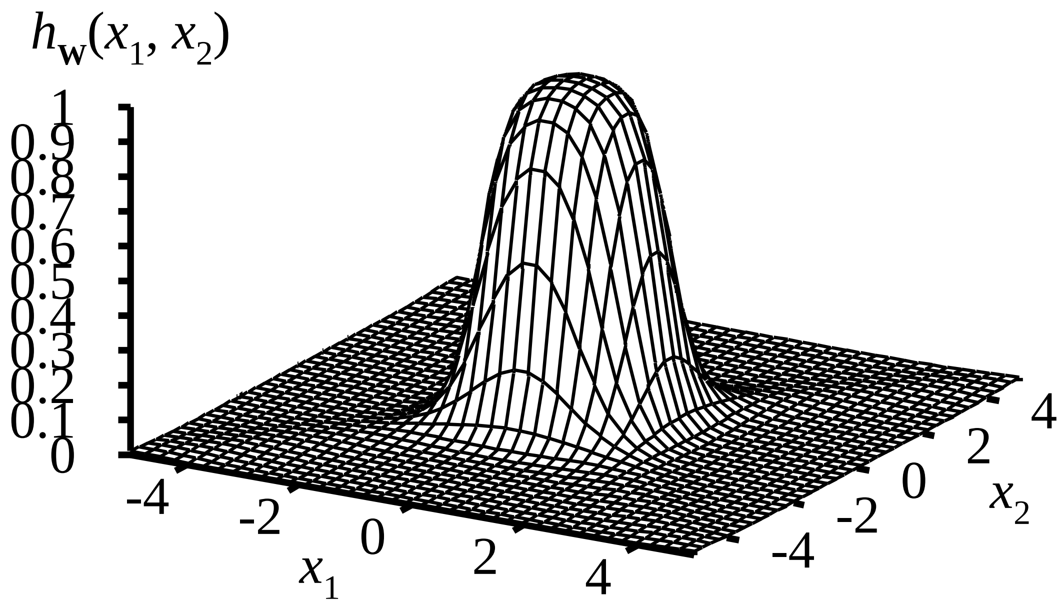
$\Rightarrow$ definiuje krawędź





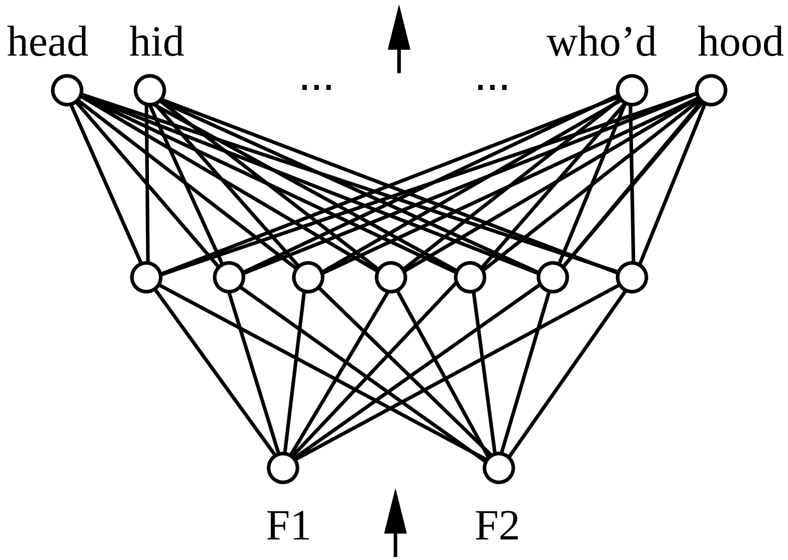
Wielowarstwowa siec neuronowa: przyklad

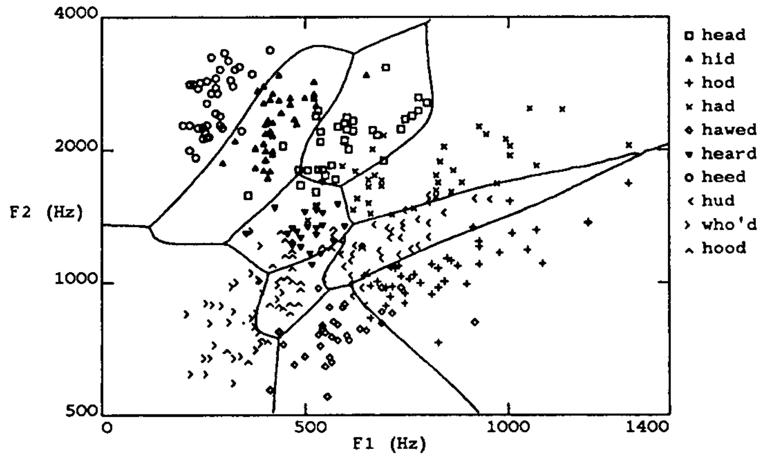
Liczba wejść: 2, liczba warstw: 3
Warstwa ukryta: 2 sieci tworzące krawędzie ustawione prostopadle do siebie
$\Rightarrow$ definiuje ograniczone wzniesienie



Wielowarstwowa siec neuronowa: przyklad

Rozpoznawanie słów: 2 warstwy, 2 wejścia, wiele wyjść





Wielowarstwowa siec neuronowa: wlasnosci

Funkcje boolowskie:

Każda funkcja boolowska może być reprezentowana przez sieć z jedną warstwą ukrytą, ale może wymagać wykładniczej liczby jednostek ukrytych

Funkcje ciągłe:

Każda ograniczona funkcja ciągła może być aproksymowana z dowolnie małym błedem przez sieć z jedną warstwą ukrytą [Cybenko 1989; Hornik et al. 1989]

Dowolna funkcja może być aproksymowana z dowolną dokładnością przez sieć z dwoma warstwami ukytymi [Cybenko 1988]

Propagacja wsteczna: algorytm

function BACK-PROP-UPDATE($examples, layers, \alpha$) **returns** a network
  inputs: $examples$, a set of examples, each with input $\vec{x}$ and output $\vec{y}(\vec{x})$
    $layer_0, layer_1, \ldots, layer_n$, neuron layers sorted from the bottom to the top
    α, the learning rate

  repeat
    for each $\vec{x} = (x_1, \ldots, x_n)$ **in** $examples$ **do**
      for each unit $j \in layer_0$ **do** $o_j \leftarrow x_j$
      for each unit $j \in layer_p$ **in order from** $layer_1$ **up to** $layer_n$ **do**
        $z_j \leftarrow \sum_{i \in\ layer_{p-1}} w_{i,j} o_i$
        $o_j \leftarrow \sigma(z_j)$
      for each unit $j \in layer_n$ **do** $\delta_j \leftarrow \sigma'(z_j)(y_j(\vec{x}) - o_j)$
      for each unit $j \in layer_p$ **in order from** $layer_{n-1}$ **down to** $layer_0$ **do**
        $\delta_j \leftarrow \sigma'(z_j) \sum_{k \in\ layer_{p+1}} w_{j,k} \delta_k$
        $\Delta\, w_{j,k} \leftarrow \alpha \delta_k\, o_j$
        $w_{j,k} \leftarrow w_{j,k} + \Delta\, w_{j,k}$
  until some stopping criterion is satisfied
  return $layers$ with modified weights

Propagacja wsteczna: wlasnosci

Fakt

Algorytm propagacji wstecznej działa dla dowolnego grafu skierowanego bez cykli

Propagacja wsteczna: wlasnosci

Fakt

Algorytm propagacji wstecznej działa dla dowolnego grafu skierowanego bez cykli

Twierdzenie

Algorytm propagacji wstecznej zbiega lokalnie do minimalnego błędu średnio-kwadratowego

Propagacja wsteczna: zbieznosc uczenia

$$E[\vec{w}] = \frac{1}{2}\sum_{\vec{x}\in U}(y(\vec{x}) - o(\vec{x}))^2 = \frac{1}{2}\sum_{\vec{x}\in U}\left(y(\vec{x}) - \sigma(\vec{w}\cdot\vec{x})\right)^2$$

Propagacja wsteczna: zbieznosc uczenia

$$E[\vec{w}] = \frac{1}{2} \Sigma_{\vec{x} \in U} (y(\vec{x}) - o(\vec{x}))^2 = \frac{1}{2} \Sigma_{\vec{x} \in U} \left(y(\vec{x}) - \sigma(\vec{w} \cdot \vec{x})\right)^2$$

$$\frac{\partial E}{\partial w_i} = \frac{\partial}{\partial w_i} \frac{1}{2} \Sigma_{\vec{x} \in U} (y(\vec{x}) - o(\vec{x}))^2$$

Propagacja wsteczna: zbieznosc uczenia

$$E[\vec{w}] = \frac{1}{2}\Sigma_{\vec{x}\in U}(y(\vec{x}) - o(\vec{x}))^2 = \frac{1}{2}\Sigma_{\vec{x}\in U}\left(y(\vec{x}) - \sigma(\vec{w}\cdot\vec{x})\right)^2$$

$$\begin{aligned}
\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i}\frac{1}{2}\Sigma_{\vec{x}\in U}(y(\vec{x}) - o(\vec{x}))^2 \\
&= \frac{1}{2}\Sigma_{\vec{x}\in U}\frac{\partial}{\partial w_i}(y(\vec{x}) - o(\vec{x}))^2
\end{aligned}$$

Propagacja wsteczna: zbieznosc uczenia

$$E[\vec{w}] = \frac{1}{2}\Sigma_{\vec{x}\in U}(y(\vec{x}) - o(\vec{x}))^2 = \frac{1}{2}\Sigma_{\vec{x}\in U}\left(y(\vec{x}) - \sigma(\vec{w}\cdot\vec{x})\right)^2$$

$$\begin{aligned}
\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i}\frac{1}{2}\Sigma_{\vec{x}\in U}(y(\vec{x}) - o(\vec{x}))^2 \\
&= \frac{1}{2}\Sigma_{\vec{x}\in U}\frac{\partial}{\partial w_i}(y(\vec{x}) - o(\vec{x}))^2 \\
&= \frac{1}{2}\Sigma_{\vec{x}\in U}2(y(\vec{x}) - o(\vec{x}))\frac{\partial}{\partial w_i}(y(\vec{x}) - o(\vec{x}))
\end{aligned}$$

Propagacja wsteczna: zbieznosc uczenia

$$E[\vec{w}] = \frac{1}{2}\Sigma_{\vec{x}\in U}(y(\vec{x}) - o(\vec{x}))^2 = \frac{1}{2}\Sigma_{\vec{x}\in U}\left(y(\vec{x}) - \sigma(\vec{w}\cdot\vec{x})\right)^2$$

$$\begin{aligned}
\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i}\frac{1}{2}\Sigma_{\vec{x}\in U}(y(\vec{x}) - o(\vec{x}))^2 \\
&= \frac{1}{2}\Sigma_{\vec{x}\in U}\frac{\partial}{\partial w_i}(y(\vec{x}) - o(\vec{x}))^2 \\
&= \frac{1}{2}\Sigma_{\vec{x}\in U}2(y(\vec{x}) - o(\vec{x}))\frac{\partial}{\partial w_i}(y(\vec{x}) - o(\vec{x})) \\
&= -\Sigma_{\vec{x}\in U}(y(\vec{x}) - o(\vec{x}))\frac{\partial}{\partial w_i}o(\vec{x})
\end{aligned}$$

Propagacja wsteczna: zbieznosc uczenia

$$E[\vec{w}] = \frac{1}{2}\Sigma_{\vec{x}\in U}(y(\vec{x}) - o(\vec{x}))^2 = \frac{1}{2}\Sigma_{\vec{x}\in U}\left(y(\vec{x}) - \sigma(\vec{w}\cdot\vec{x})\right)^2$$

$$\begin{aligned}
\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i}\frac{1}{2}\Sigma_{\vec{x}\in U}(y(\vec{x}) - o(\vec{x}))^2 \\
&= \frac{1}{2}\Sigma_{\vec{x}\in U}\frac{\partial}{\partial w_i}(y(\vec{x}) - o(\vec{x}))^2 \\
&= \frac{1}{2}\Sigma_{\vec{x}\in U}2(y(\vec{x}) - o(\vec{x}))\frac{\partial}{\partial w_i}(y(\vec{x}) - o(\vec{x})) \\
&= -\Sigma_{\vec{x}\in U}(y(\vec{x}) - o(\vec{x}))\frac{\partial}{\partial w_i}o(\vec{x}) \\
&= -\Sigma_{\vec{x}\in U}(y(\vec{x}) - o(\vec{x}))\frac{\partial\sigma}{\partial z}[z = \vec{w}\cdot\vec{x}]\frac{\partial}{\partial w_i}(\vec{w}\cdot\vec{x})
\end{aligned}$$

Propagacja wsteczna: zbieznosc uczenia

$$E[\vec{w}] = \frac{1}{2}\Sigma_{\vec{x}\in U}(y(\vec{x}) - o(\vec{x}))^2 = \frac{1}{2}\Sigma_{\vec{x}\in U}\left(y(\vec{x}) - \sigma(\vec{w}\cdot\vec{x})\right)^2$$

$$\begin{aligned}
\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i}\frac{1}{2}\Sigma_{\vec{x}\in U}(y(\vec{x}) - o(\vec{x}))^2 \\
&= \frac{1}{2}\Sigma_{\vec{x}\in U}\frac{\partial}{\partial w_i}(y(\vec{x}) - o(\vec{x}))^2 \\
&= \frac{1}{2}\Sigma_{\vec{x}\in U}2(y(\vec{x}) - o(\vec{x}))\frac{\partial}{\partial w_i}(y(\vec{x}) - o(\vec{x})) \\
&= -\Sigma_{\vec{x}\in U}(y(\vec{x}) - o(\vec{x}))\frac{\partial}{\partial w_i}o(\vec{x}) \\
&= -\Sigma_{\vec{x}\in U}(y(\vec{x}) - o(\vec{x}))\frac{\partial\sigma}{\partial z}[z = \vec{w}\cdot\vec{x}]\frac{\partial}{\partial w_i}(\vec{w}\cdot\vec{x})
\end{aligned}$$

$$\frac{\partial}{\partial w_i}(\vec{w}\cdot\vec{x}) = x_i$$

Propagacja wsteczna: zbieznosc uczenia

$$E[\vec{w}] = \frac{1}{2}\Sigma_{\vec{x}\in U}(y(\vec{x}) - o(\vec{x}))^2 = \frac{1}{2}\Sigma_{\vec{x}\in U}\left(y(\vec{x}) - \sigma(\vec{w}\cdot\vec{x})\right)^2$$

$$\begin{aligned}
\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i}\frac{1}{2}\Sigma_{\vec{x}\in U}(y(\vec{x}) - o(\vec{x}))^2 \\
&= \frac{1}{2}\Sigma_{\vec{x}\in U}\frac{\partial}{\partial w_i}(y(\vec{x}) - o(\vec{x}))^2 \\
&= \frac{1}{2}\Sigma_{\vec{x}\in U}2(y(\vec{x}) - o(\vec{x}))\frac{\partial}{\partial w_i}(y(\vec{x}) - o(\vec{x})) \\
&= -\Sigma_{\vec{x}\in U}(y(\vec{x}) - o(\vec{x}))\frac{\partial}{\partial w_i}o(\vec{x}) \\
&= -\Sigma_{\vec{x}\in U}(y(\vec{x}) - o(\vec{x}))\frac{\partial\sigma}{\partial z}[z = \vec{w}\cdot\vec{x}]\frac{\partial}{\partial w_i}(\vec{w}\cdot\vec{x}) \\
&= -\Sigma_{\vec{x}\in U}(y(\vec{x}) - o(\vec{x}))\frac{\partial\sigma}{\partial z}[z = \vec{w}\cdot\vec{x}]x_i
\end{aligned}$$

Propagacja wsteczna: zbieznosc uczenia

$\nabla E[\vec{w}] = -\sum_{\vec{x} \in U} \frac{\partial \sigma}{\partial z}[z = \vec{w} \cdot \vec{x}](y(\vec{x}) - o(\vec{x}))\vec{x}$

Propagacja wsteczna: zbieznosc uczenia

$\nabla E[\vec{w}] = -\sum_{\vec{x} \in U} \frac{\partial \sigma}{\partial z}[z = \vec{w} \cdot \vec{x}](y(\vec{x}) - o(\vec{x}))\vec{x}$

Stąd dla neuronów j z najwyższej warstwy stosujemy zmiany wag:

$\delta_j \leftarrow \frac{\partial \sigma}{\partial z}[z = \vec{w} \cdot \vec{x}](y_j(\vec{x}) - o_j(\vec{x}))$

$\Delta w_{i,j} \leftarrow \alpha \delta_j x_{i,j}$

Propagacja wsteczna: zbieznosc uczenia

$\nabla E[\vec{w}] = -\sum_{\vec{x} \in U} \frac{\partial \sigma}{\partial z}[z = \vec{w} \cdot \vec{x}](y(\vec{x}) - o(\vec{x}))\vec{x}$

Stąd dla neuronów j z najwyższej warstwy stosujemy zmiany wag:

$\delta_j \leftarrow \frac{\partial \sigma}{\partial z}[z = \vec{w} \cdot \vec{x}](y_j(\vec{x}) - o_j(\vec{x}))$

$\Delta w_{i,j} \leftarrow \alpha \delta_j x_{i,j}$

Neurony z niższych warstw muszą mieć odpowiednik błędu $y_j(\vec{x}) - o_j(\vec{x})$

Propagacja wsteczna: zbieznosc uczenia

$\nabla E[\vec{w}] = -\sum_{\vec{x} \in U} \frac{\partial \sigma}{\partial z}[z = \vec{w} \cdot \vec{x}](y(\vec{x}) - o(\vec{x}))\vec{x}$

Stąd dla neuronów j z najwyższej warstwy stosujemy zmiany wag:

$\delta_j \leftarrow \frac{\partial \sigma}{\partial z}[z = \vec{w} \cdot \vec{x}](y_j(\vec{x}) - o_j(\vec{x}))$

$\Delta w_{i,j} \leftarrow \alpha \delta_j x_{i,j}$

Neurony z niższych warstw muszą mieć odpowiednik błędu $y_j(\vec{x}) - o_j(\vec{x})$
$\Rightarrow$ dla każdego neuronu $j \in layer_p$ liczona jest ważona "suma błędów" na wyjściach tego neuronu $\sum_{k \in layer_{p+1}} w_{j,k} \delta_k$

Propagacja wsteczna: zbieznosc uczenia

$\nabla E[\vec{w}] = -\sum_{\vec{x} \in U} \frac{\partial \sigma}{\partial z}[z = \vec{w} \cdot \vec{x}](y(\vec{x}) - o(\vec{x}))\vec{x}$

Stąd dla neuronów j z najwyższej warstwy stosujemy zmiany wag:

$\delta_j \leftarrow \frac{\partial \sigma}{\partial z}[z = \vec{w} \cdot \vec{x}](y_j(\vec{x}) - o_j(\vec{x}))$

$\Delta w_{i,j} \leftarrow \alpha \delta_j x_{i,j}$

Neurony z niższych warstw muszą mieć odpowiednik błędu $y_j(\vec{x}) - o_j(\vec{x})$
$\Rightarrow$ dla każdego neuronu $j \in layer_p$ liczona jest ważona "suma błędów" na wyjściach tego neuronu $\sum_{k \in layer_{p+1}} w_{j,k} \delta_k$

Zmiana wag definiowana jest wtedy jako:

$\delta_j \leftarrow \frac{\partial \sigma}{\partial z}[z = \vec{w} \cdot \vec{x}] \sum_{k \in layer_{p+1}} w_{j,k} \delta_k$

$\Delta w_{i,j} \leftarrow \alpha \delta_j x_{i,j}$

Prop. wsteczna z sigmoidalna funkcja aktywacji

Sigmoidalna funkcja aktywacji $\sigma(z) = \frac{1}{1+e^{-z}}$ we wszystkich neuronach

$$o(\vec{x}) = \sigma(\vec{w} \cdot \vec{x}) = \frac{1}{1 + e^{-\vec{w}\cdot\vec{x}}}$$

$$\frac{\partial \sigma}{\partial z} = \left(\frac{1}{1 + e^{-z}}\right)\left(1 - \frac{1}{1 + e^{-z}}\right)$$

$$\frac{\partial \sigma}{\partial z}[z = \vec{w} \cdot \vec{x}] = o(\vec{x})(1 - o(\vec{x}))$$

Wartości współczynników zmiany wag δ_j

— dla neuronów j z warstwy najwyższej:

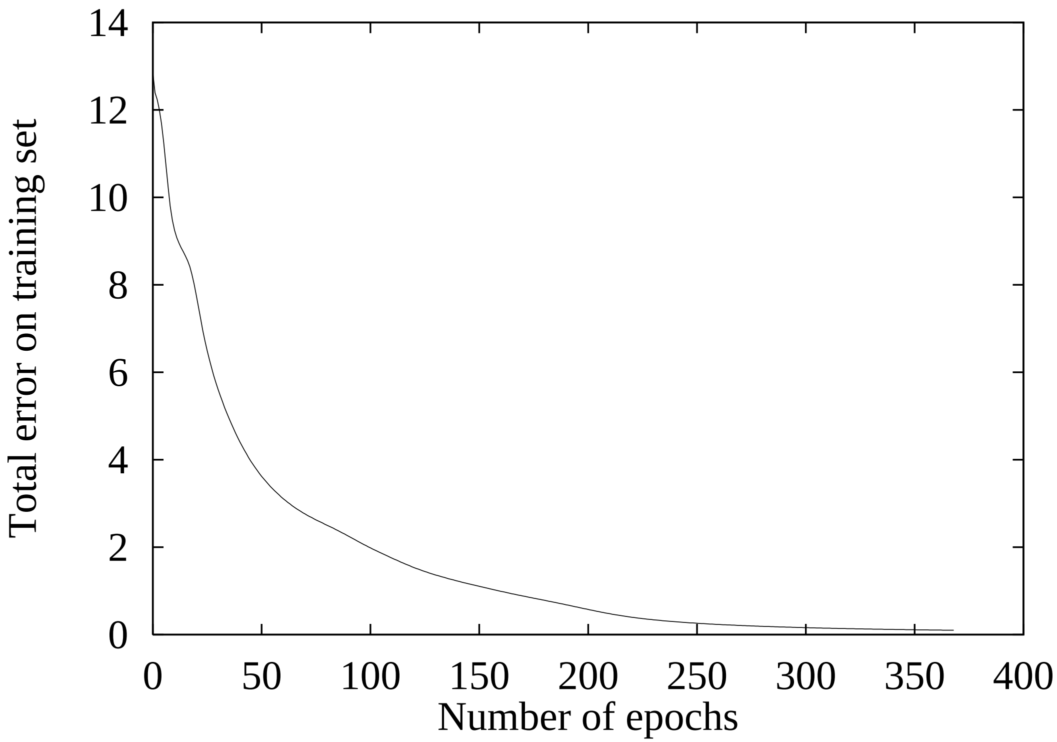
$$\delta_j \leftarrow o_j(1 - o_j)(y_j(\vec{x}) - o_j)$$

— dla neuronów j z każdej niższej warstwy p:

$$\delta_j \leftarrow o_j(1 - o_j) \textstyle\sum_{k \in layer_{p+1}} w_{j,k} \delta_k$$

Propagacja wsteczna: przyklad zbieznosci

Epoka: przebiega jednokrotnie wszystkie obiekty treningowe poprawiając wagi,
na koniec wylicza bład sumaryczny dla całego zbioru treningowego
$\Rightarrow$ algorytm uczenia zatrzymuje się, kiedy błąd przestaje maleć



Dobor wspolczynnika szybkosci uczenia α

Zazwyczaj:

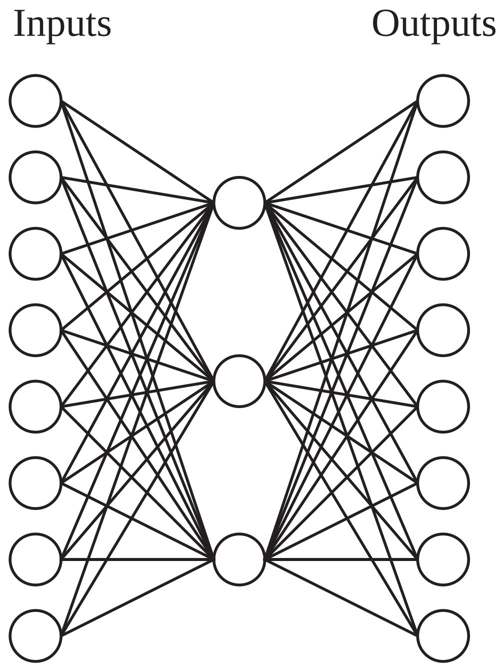
$$\alpha \in [0.01; 0.5]$$

Po każdej ustalonej liczbie epok można redukować geometrycznie:

$$\alpha := \alpha \cdot c \qquad\qquad c \in [0.9; 0.99]$$

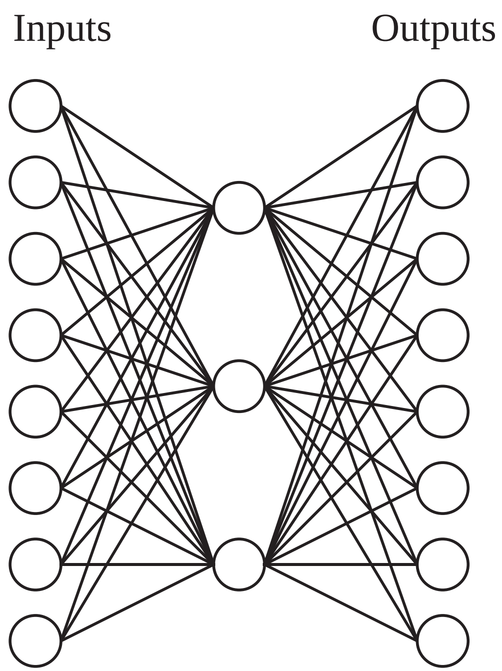
$\Rightarrow$ Pozwala na szybką zbieżność na początku (np. $\alpha \approx 0.5$)
i precyzyjną zbieżność do lokalnego maksimum w końcowej fazie ($\alpha \approx 0$)

Uczenie neuronow ukrytych (wewnetrznych)



Input		Output
10000000	$\rightarrow$	10000000
01000000	$\rightarrow$	01000000
00100000	$\rightarrow$	00100000
00010000	$\rightarrow$	00010000
00001000	$\rightarrow$	00001000
00000100	$\rightarrow$	00000100
00000010	$\rightarrow$	00000010
00000001	$\rightarrow$	00000001

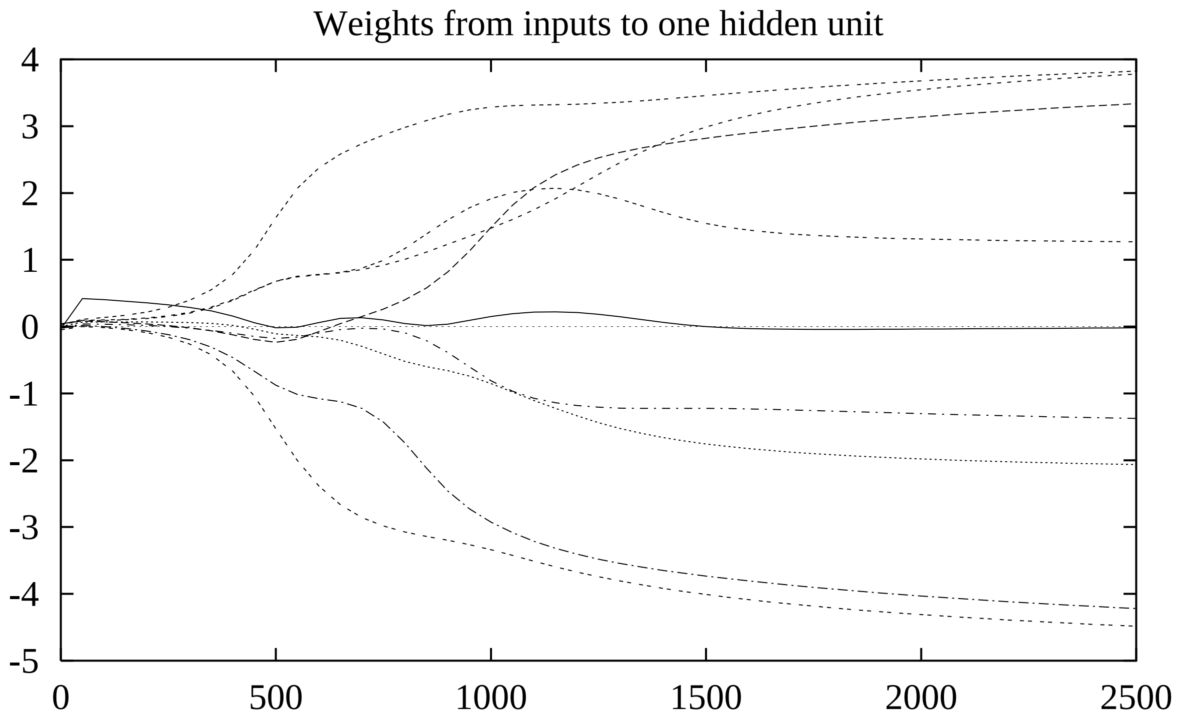
Uczenie neuronow ukrytych (wewnetrznych)



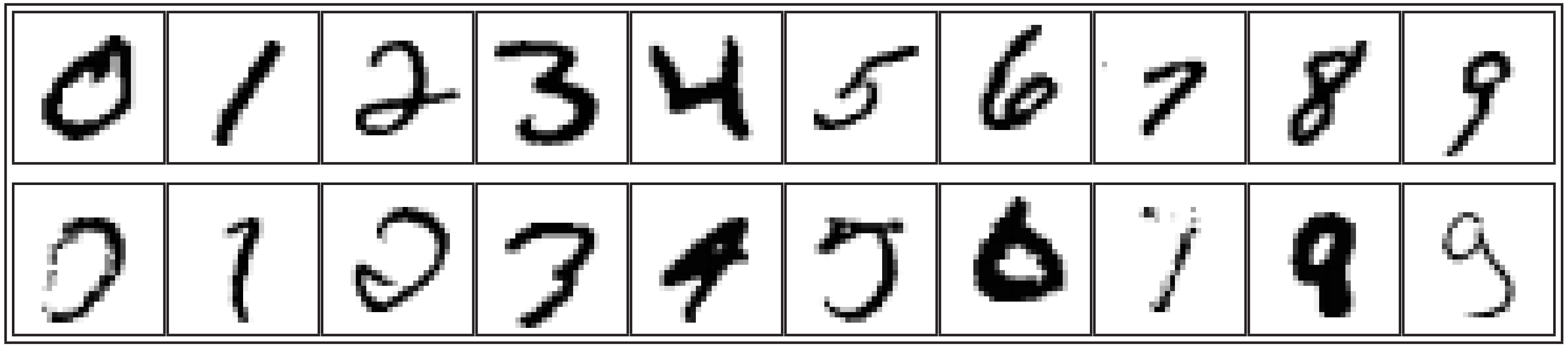
Input		Hidden Values				Output
10000000	→	.89	.04	.08	→	10000000
01000000	→	.01	.11	.88	→	01000000
00100000	→	.01	.97	.27	→	00100000
00010000	→	.99	.97	.71	→	00010000
00001000	→	.03	.05	.02	→	00001000
00000100	→	.22	.99	.99	→	00000100
00000010	→	.80	.01	.98	→	00000010
00000001	→	.60	.94	.01	→	00000001

Uczenie neuronow ukrytych (wewnetrznych)

Trenowanie wag dla jednego z neuronów wewnętrznych:



Rozpoznawanie cyfr recznie pisanych



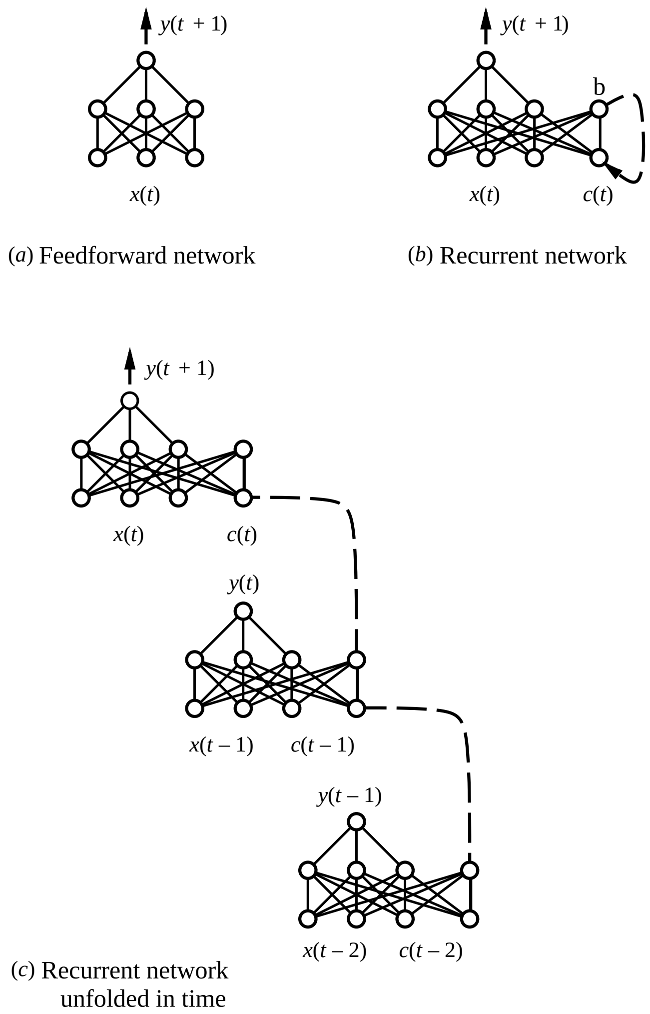
3-nn = 2.4% błędów

Sieć 3-warstwowa (400–300–10) = 1.6% błędów

LeNet (sieć 4-warstwowa, 768–192–30–10) = 0.9% błędów

Sieci rekurencyjne

Zawierają cykle skierowane, zmieniają wagi w kolejnych taktach zegara



Sieci rekurencyjne

◇ Sieci Hopfielda (holograficzne pamięci asocjacyjne)
— symetryczne wagi
— progowa funkcja aktywacji $\sigma(z) = sign(z)$

◇ Maszyny Bolztmanna
— używają stochastycznych funkcji aktywacji