

UCZENIE MASZYNOWE

SZTUCZNE SIECI NEURONOWE DO APROKSYMACJI FUNKCJI

Dr hab. inż. Grzegorz Dudek
Wydział Elektryczny
Politechnika Częstochowska

Projekt finansowany w ramach programu Ministra Nauki i Szkolnictwa Wyższego pod nazwą „Regionalna Inicjatywa Doskonałości” w latach 2019 - 2022 nr projektu 020/RID/2018/19 kwota finansowania 12 000 000 PLN

PROBLEM APROKSYMACJI FUNKCJI

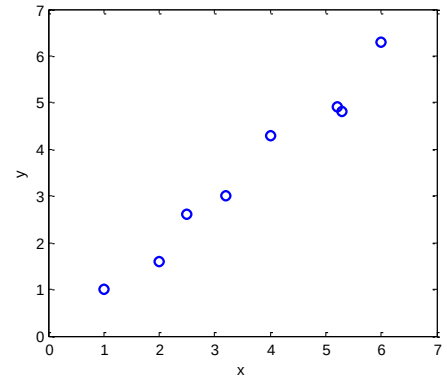
Aproksymacja funkcji – przybliżanie funkcji, polegające na wyznaczeniu dla danej funkcji $f(x)$ takiej funkcji $h(x)$, która w określonym sensie najlepiej ją przybliża. Przybliżenie w tym wypadku rozumiane jest jako minimalizacja pewnej funkcji błędu. Często miarą błędu jest średni błąd kwadratowy.

Typowe zadanie aproksymacji – dany jest zbiór punktów (np. pomiarowych)

Zakładamy postać funkcji aproksymującej, np. funkcję liniową:

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

gdzie a i b to współczynniki, które należy tak dobrać, aby błąd aproksymacji był jak najmniejszy.

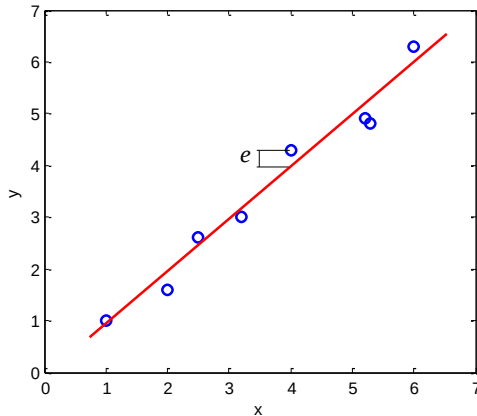


PROBLEM APROKSYMACJI FUNKCJI

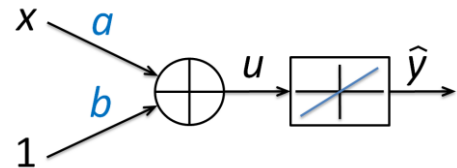
Błąd aproksymacji (M oznacza liczbę punktów):

$$E = \sum_{i=1}^M (y_i - h(x_i))^2 = \sum_{i=1}^M e_i^2 \rightarrow \min$$

W wyniku otrzymujemy aproksymantę:

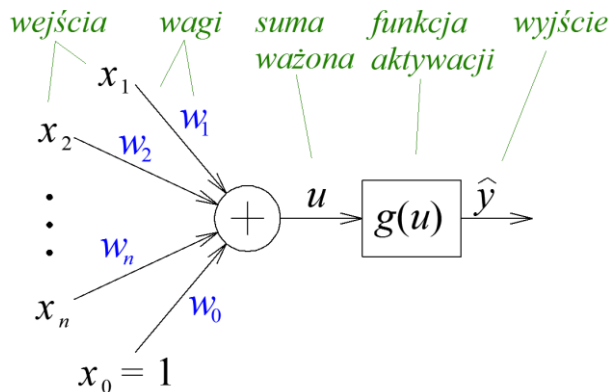


Reprezentacja graficzna funkcji aproksymującej:



MODEL NEURONU

W przypadku aproksymacji funkcji wielowymiarowej, model ma postać:



Jest to model neuronu, który realizuje funkcję:

$$h(\mathbf{x}) = g\left(\sum_{j=0}^n x_j w_j\right)$$

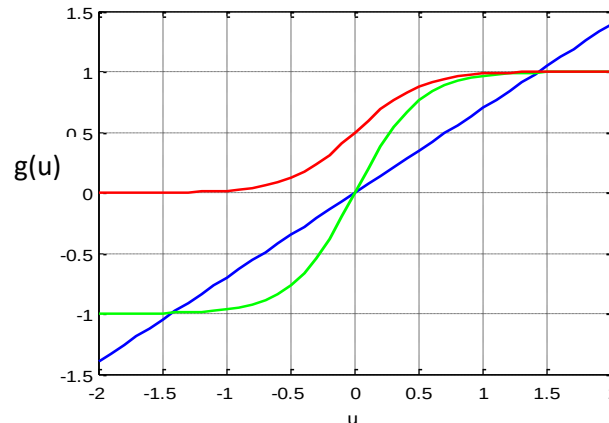
FUNKCJE AKTYWACJI

Funkcje najczęściej używane jako funkcje aktywacji:

1. **sigmoidalna unipolarna**: $g(u) = \frac{1}{1 + \exp(-\beta u)}$

2. **sigmoidalna bipolarna**: $g(u) = \frac{2}{1 + \exp(-\beta u)} - 1$ lub $g(u) = \tanh(\beta u)$

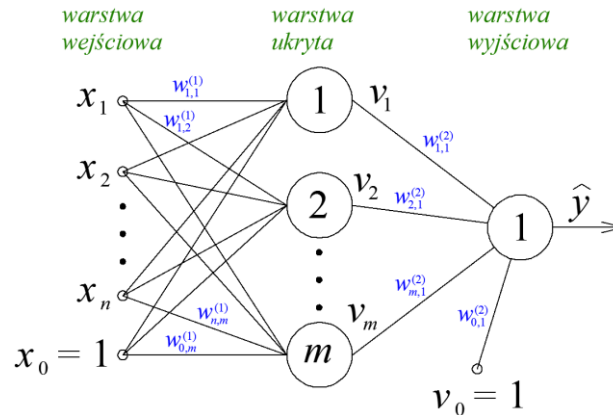
3. **liniowa**: $g(u) = \beta u$



SIEĆ NEURONOWA

Pojedynczy neuron ma ograniczone zdolności aproksymacyjne. Aby aproksymować bardziej złożone funkcje łączy się wiele neuronów w sieć. Funkcje realizowane przez neurony dają po złożeniu dobrą aproksymantę.

Udowodniono, że sieć taka, zwana **wielowarstwowym perceptronem**, może aproksymować dowolną funkcję z dowolnie małym błędem.



Sieć może posiadać więcej niż jedną warstwę ukrytą oraz więcej niż jeden neuron na wyjściu.

Sieć uczymy na zbiorze **treningowym** (uczącym) złożonym z wektorów wejściowych (zmiennych niezależnych) $\mathbf{x}$ i skojarzonych z nimi wartości pożądaných odpowiedzi y (zmienna zależna). Sieć może mieć kilka wyjść, wtedy mamy do czynienia z wektorem pożądanęj odpowiedzi $\mathbf{y}$. Każdą parę $\mathbf{x}$ i y nazywamy **wzorcem uczącym**.

Uczenie przebiega iteracyjnie:

1. Na wstępie losujemy wagi sieci i dobieramy parametry uczenia sieci.
2. Na wejście sieci podajemy (prezentujemy) i -ty wektor wejściowy. Składowe tego wektora są przemnażane przez wagi pierwszej warstwy, następnie sumowane i przetwarzane przez funkcje aktywacji neuronów. Na wyjściu tych neuronów otrzymujemy wektor $\mathbf{v} = [v_1 \ v_2 \ \dots \ v_m]$, którego składowe przemnażane są przez wagi drugiej warstwy, sumowane i przetwarzane przez funkcję aktywacji neuronu w warstwie wyjściowej. Otrzymujemy wyjście $\hat{y}$. Ponieważ wagi były losowe, to co otrzymujemy na wyjściu w pierwszym kroku jest przypadkowe. Możemy wyznaczyć błąd sieci:

$$e_i^2 = \frac{1}{2}(\hat{y}_i - y_i)^2 = \frac{1}{2} \left[g \left(\sum_{j=0}^m w_{j,1}^{(2)} v_j \right) - y_i \right]^2 = \frac{1}{2} \left[g \left(\sum_{j=1}^m w_{j,1}^{(2)} g \left(\sum_{k=0}^n w_{k,j}^{(1)} x_{i,k} \right) + w_{0,1}^{(2)} v_0 \right) - y_i \right]^2$$

Błąd ten służy do korekty wag.

3. W fazie wstecznej propagacji błędu błąd wędruje od wyjścia w kierunku wejść. Najpierw błąd przechodzi na „drugą” stronę neuronu wyjściowego, wymaga to wyznaczenie pochodnej funkcji aktywacji tego neuronu. Następnie oblicza się składowe tego błędu wnoszone przez poszczególne wejścia neuronu wyjściowego i modyfikuje się wagi związane z tymi wejściami. Dalej sygnały błędu wędrują na „drugą” stronę neuronów ukrytych i wykorzystywane są do adaptacji wag związanych z wejściami tych neuronów. Po adaptacji wag błąd dla i -tego wzorca uczącego będzie mniejszy. [Szczegółowy opis metody wstecznej propagacji błędu można znaleźć w \[Oso\].](#)
4. Kroki 2 i 3 powtarzamy dla wszystkich wzorców uczących. Prezentacja wszystkich wzorców ze zbioru uczącego nazywa się **epoką**. Trening sieci wymaga wielu epok uczenia.

W efekcie uczenia minimalizowany jest błąd średniokwadratowy po wszystkich wzorcach uczących:

$$E = \sum_{i=1}^M e_i^2 \rightarrow \min$$

ALGORYTM WSTECZNEJ PROPAGACJI BŁĘDU

Celem jest minimalizacja funkcji błędu sieci:

$$E = e^2 = \frac{1}{2}(\hat{y} - y)^2 = \frac{1}{2} \left[g \left(\sum_{j=1}^m w_j^{(2)} v_j + w_0^{(2)} v_0 \right) - y \right]^2 = \frac{1}{2} \left[g \left(\sum_{j=1}^m w_j^{(2)} g \left(\sum_{k=0}^n w_{k,j}^{(1)} x_k \right) + w_0^{(2)} v_0 \right) - y \right]^2$$

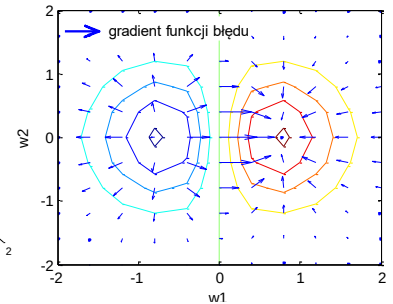
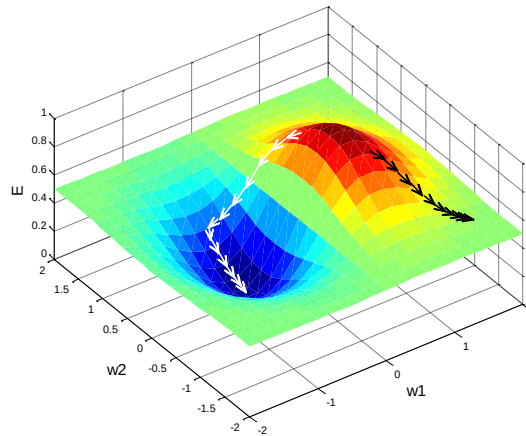
Do znalezienia minimum stosuje się metodę gradientową zwaną metodą największego spadku, zgodnie z którą przyrost wag określa się wg kierunku ujemnego gradientu:

$$\Delta \mathbf{w} = -\eta \nabla E(\mathbf{w})$$

gdzie $\mathbf{w}$ jest wektorem wag,
 $\eta > 0$ współczynnikiem uczenia, a

$$\nabla E(\mathbf{w}) = \left[\frac{\partial E}{\partial w_{1,1}^{(1)}}, \frac{\partial E}{\partial w_{1,2}^{(1)}}, \dots, \frac{\partial E}{\partial w_m^{(2)}} \right]$$

jest gradientem błędu.



ALGORYTM WSTECZNEJ PROPAGACJI BŁĘDU

Pochodne cząstkowe funkcji błędu względem wag warstwy wyjściowej:

$$\frac{\partial E}{\partial w_j^{(2)}} = (\hat{y} - y) \frac{\partial g(u^{(2)})}{\partial u^{(2)}} v_j, \quad j = 0, 1, \dots, m$$

gdzie $u^{(2)} = \sum_{j=1}^m w_j^{(2)} v_j + w_0^{(2)} v_0$.

Wprowadźmy oznaczenie: $\delta^{(2)} = (\hat{y} - y) \frac{\partial g(u^{(2)})}{\partial u^{(2)}}$, stąd regułę aktualizacji wag neuronu wyjściowego można zapisać:

$$\Delta w_j^{(2)} = -\eta \delta^{(2)} v_j.$$

Pochodne cząstkowe błędu względem wag warstwy ukrytej:

$$\frac{\partial E}{\partial w_{k,j}^{(1)}} = (\hat{y} - y) \frac{\partial \hat{y}}{\partial v_j} \frac{\partial v_j}{\partial w_{k,j}^{(1)}} = (\hat{y} - y) \left(\frac{\partial g(u^{(2)})}{\partial u^{(2)}} w_{k,j}^{(2)} \right) \left(\frac{\partial g(u_j^{(1)})}{\partial u_j^{(1)}} x_k \right), \quad j = 1, \dots, m; k = 0, 1, \dots, n$$

gdzie $u_j^{(1)} = \sum_{k=0}^n w_{k,j}^{(1)} x_k$.

ALGORYTM WSTECZNEJ PROPAGACJI BŁĘDU

Przyjmując oznaczenie: $\delta_j^{(1)} = (\hat{y} - y) \frac{\partial g(u^{(2)})}{\partial u^{(2)}} w_{k,j}^{(2)} \frac{\partial g(u_j^{(1)})}{\partial u_j^{(1)}} = \delta^{(2)} w_{k,j}^{(2)} \frac{\partial g(u_j^{(1)})}{\partial u_j^{(1)}}$ otrzymuje się $\frac{\partial E}{\partial w_{k,j}^{(1)}} = \delta_j^{(1)} x_k$

i wzór na aktualizację wag neuronów warstwy ukrytej:

$$\Delta w_{k,j}^{(1)} = -\eta \delta_j^{(1)} x_k$$

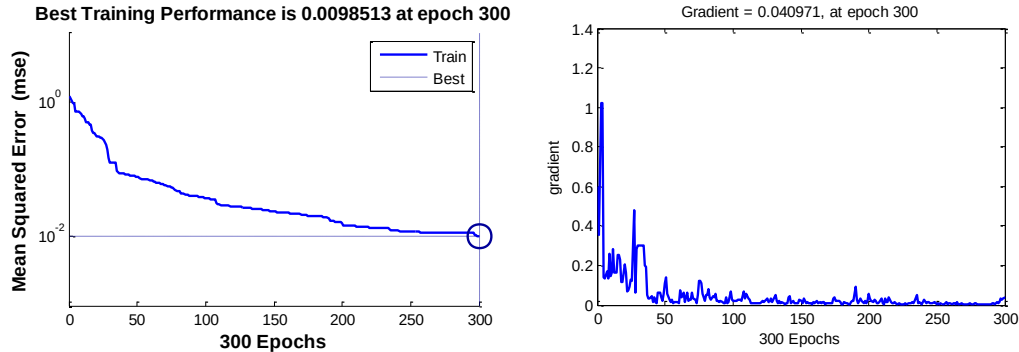
Po wyznaczeniu poprawek dla wag dokonuje się aktualizacji wag:

$$\mathbf{w}(l+1) = \mathbf{w}(l) + \Delta \mathbf{w}$$

- Metoda wstecznej propagacji błędu wymaga, a by funkcje aktywacji były różniczkowalne.
- Skuteczność metody zależy od kształtu funkcji błędu (wielomodalność, płaskie obszary), punktu startowego wag, długości kroku (η).
- Algorytm utyka w minimach lokalnych.
- Istnieją inne metody uczenia, które w uproszczony sposób wyznaczają kierunek przesunięcia wektora wag (algorytmy: zmiennej metryki, Levenberga–Marquardta, gradientów sprzężonych).

ALGORYTM WSTECZNEJ PROPAGACJI BŁĘDU

Przykładowy przebieg błędu i jego gradient w kolejnych epokach



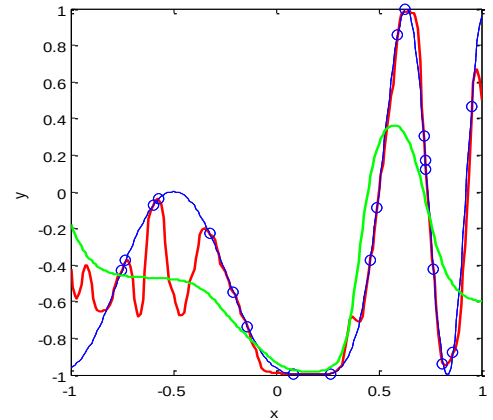
Po nauczaniu sieci sprawdzamy jej działanie na nowym zbiorze danych zwanym **testowym**. Błędy wyznaczone na tym zbiorze świadczą o jakości działania sieci.

PROBLEMY UCZENIA SIECI

W procesie uczenia musimy rozstrzygnąć kilka problemów:

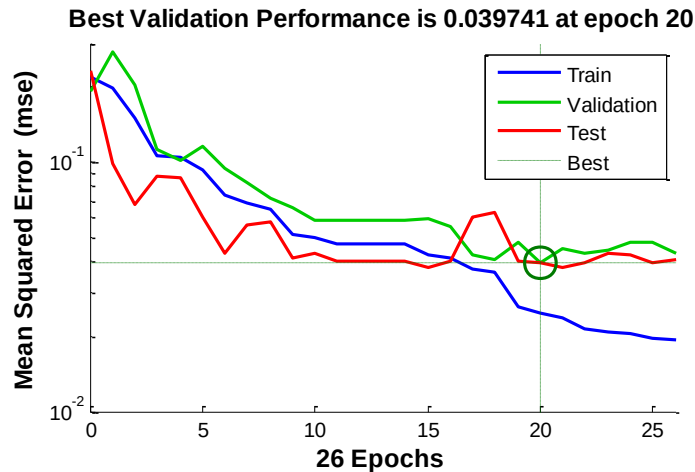
- jak długo sieć ma się uczyć
- ile powinno być neuronów w warstwie ukrytej
- jakie powinny być funkcje aktywacji neuronów
- jaką metodę uczenia wybrać
- czy i w jaki sposób wstępnie przetworzyć dane.

Jeśli trening jest zbyt krótki lub/i liczba neuronów zbyt mała sieć będzie **niedouczona** (duże błędy), zbyt długi trening lub/i zbyt duża liczba neuronów skutkuje **przeuczeniem** – błędy uzyskane na zbiorze uczącym będą bliskie 0, lecz błędy na zbiorze testowym okażą się duże.

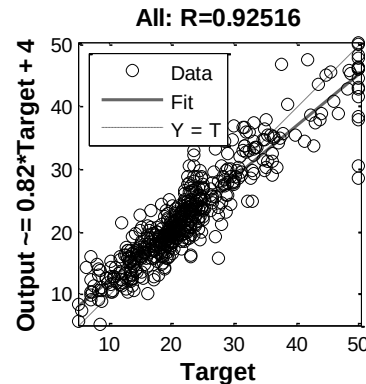
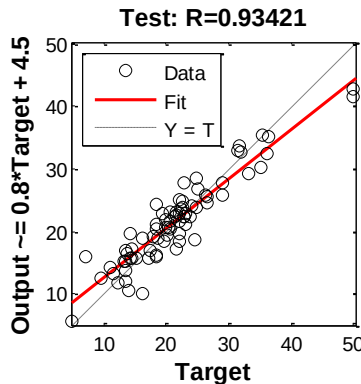
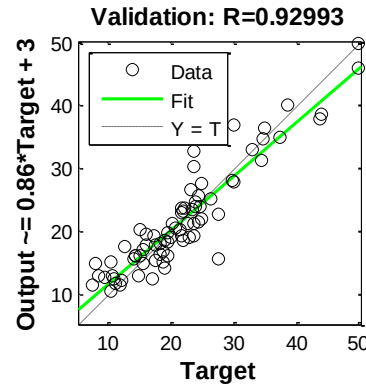
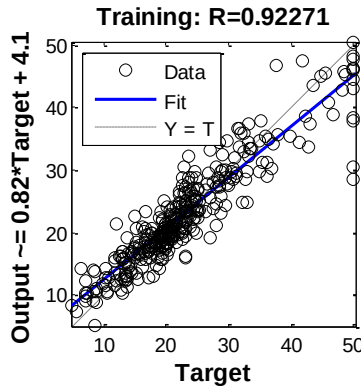


PROBLEMY UCZENIA SIECI

Sieć powinna posiadać zdolność uogólniania (**generalizacji**) zdobytej wiedzy na nowe przykłady, które nie uczestniczyły w procesie uczenia. Aby wzmocnić tę zdolność w trakcie uczenia w każdej epoce testuje się sieć na tzw. zbiorze **walidacyjnym**. Jeśli błąd na tym zbiorze przestaje maleć lub zaczyna wzrastać, co oznacza, że sieć traci zdolność uogólniania, przerywa się trening.



DOPASOWANIE MODELU



Współczynnik determinacji:

$$R^2 = \frac{\sum_{i=1}^M (\hat{y}_i - \bar{y})^2}{\sum_{i=1}^M (y_i - \bar{y})^2}$$

gdzie $\bar{y}$ – wartość średnia pożądaných odpowiedzi (target).

$R^2 =$

0,0 - 0,5 - dopasowanie niezadowalające

0,5 - 0,6 - dopasowanie słabe

0,6 - 0,8 - dopasowanie zadowalające

0,8 - 0,9 - dopasowanie dobre

0,9 - 1,0 - dopasowanie bardzo dobre

