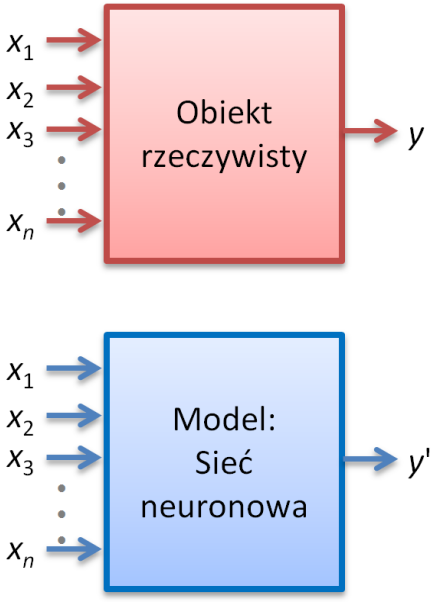
ZAAWANSOWANE METODY EKSPLORACJI

SIECI NEURONOWE W ZADANIACH REGRESJI I KLASYFIKACJI

Prof. dr hab. inż. Grzegorz Dudek
Wydział Matematyki i Informatyki
Uniwersytet Łódzki

Modelowanie



Cechy sztucznych sieci neuronowych

- własność uniwersalnego aproksymatora
- uczenie się, pozyskiwanie wiedzy z danych
- odporność na zakłócenia danych
- równoległa architektura
- adaptacyjność

Zastosowania sztucznych sieci neuronowych

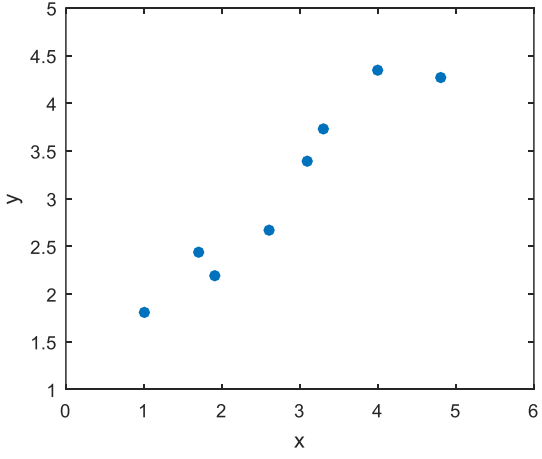
- sterowanie procesów przemysłowych
- identyfikacja systemów
- rozpoznawanie i klasyfikacja wzorców
- predykcja szeregów czasowych
- analiza danych statystycznych
- rozpoznawanie sekwencji
- odszumianie i kompresja obrazu i dźwięku
- diagnostyka techniczna i medyczna
- prognozowanie
- ...

Sieci neuronowe do regresji

APROKSYMACJA FUNKCJI

Aproksymacja funkcji polega na przybliżeniu pewnej funkcji $f(x)$ danej w postaci zbioru punktów (np. pomiarowych) funkcją $h(x)$ zapisaną wzorem.

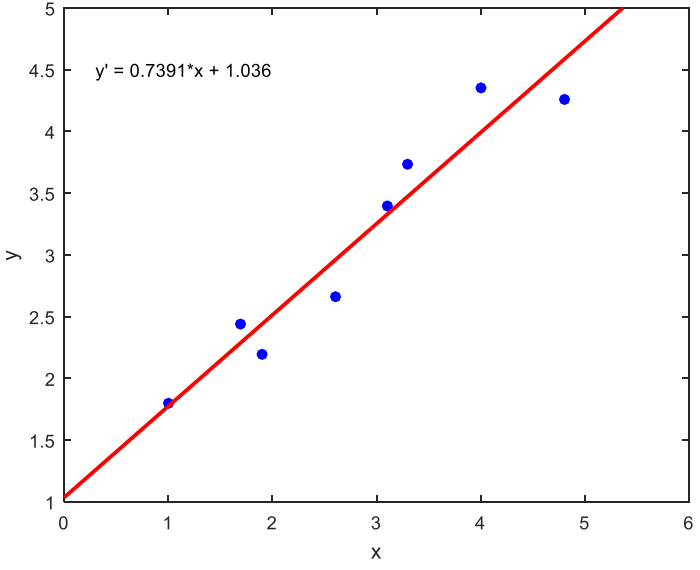
x	1,00	1,70	1,90	2,60	3,10	3,30	4,00	4,80
y	1,80	2,44	2,19	2,66	3,39	3,73	4,34	4,26



APROKSYMACJA FUNKCJI

Liniowa funkcja aproksymująca:

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

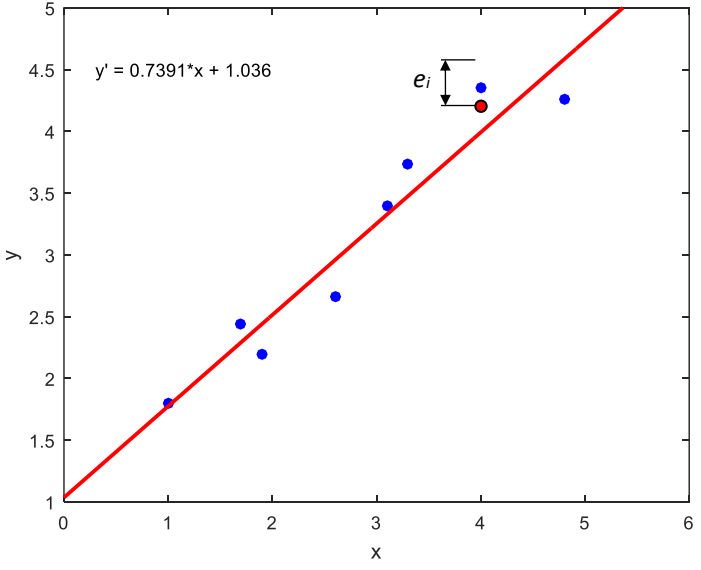


Błąd aproksymacji

Miarą przybliżenia funkcji *f*(*x*) przez funkcję *h*(*x*) jest najczęściej średni błąd kwadratowy (*mean squared error, MSE*) :

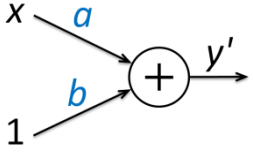
$$E = \frac{1}{M}\sum_{i=1}^{M}[y_i - h(x_i)]^2 = \frac{1}{M}\sum_{i=1}^{M} e_i^2$$

$$E \to \min$$



APROKSYMACJA FUNKCJI

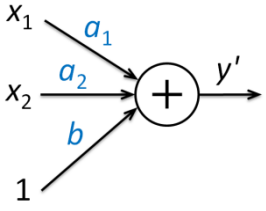
$$h(x) = ax + b$$

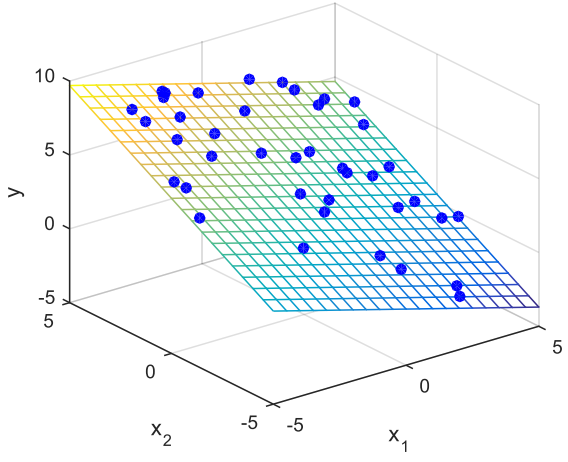


APROKSYMACJA FUNKCJI

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

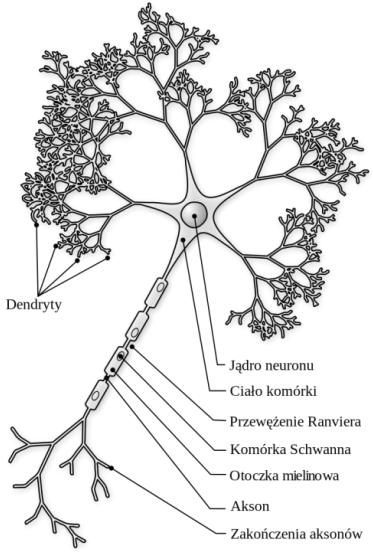
$$h(x) = a_1 x_1 + a_2 x_2 + b$$

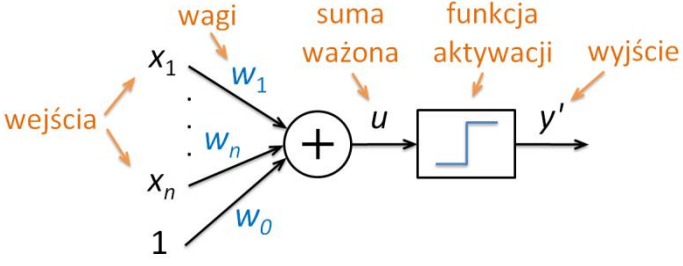




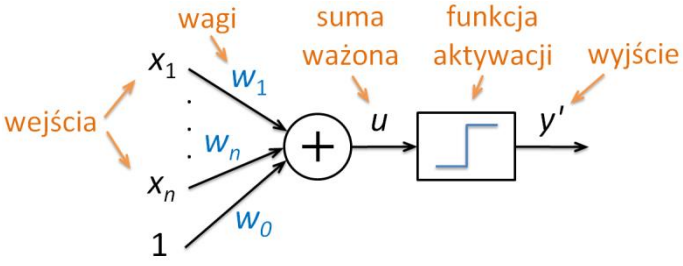
MODEL NEURONU

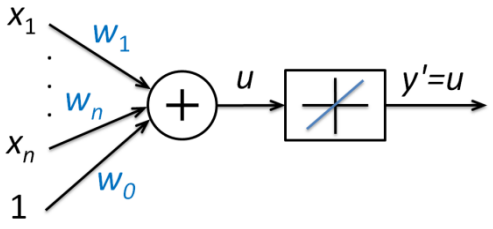
Neuron McCullocha-Pittsa





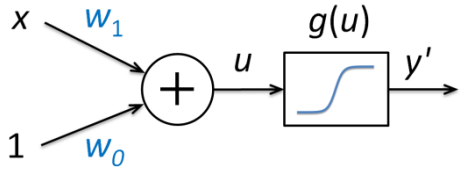
MODEL NEURONU





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

MODEL NEURONU

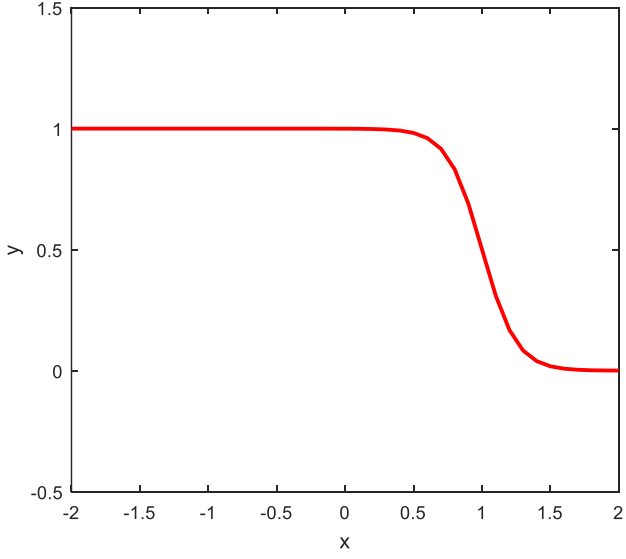


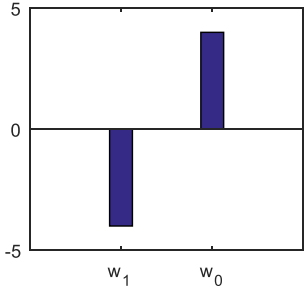
$$g(u) = \frac{1}{1+\exp(-\beta u)} = \frac{1}{1+\exp[-\beta(w_1 x + w_0)]}$$

$$\beta = \text{const} \neq 0$$

MODEL NEURONU

$$g(u) = \frac{1}{1+\exp(-\beta u)} = \frac{1}{1+\exp[-\beta(w_1 x + w_0)]}$$

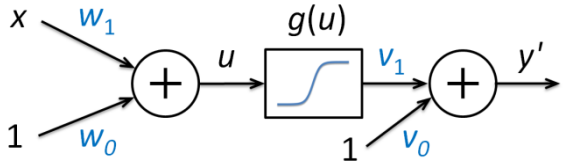




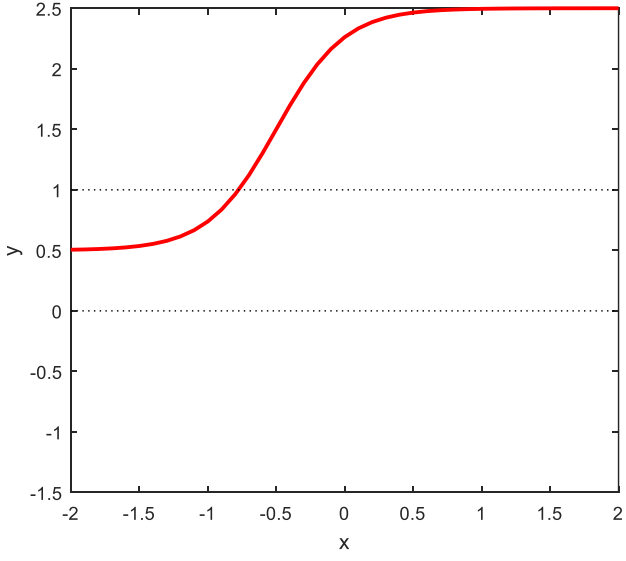
Video

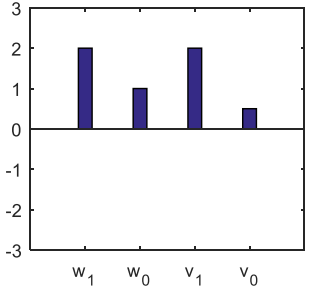
MODEL NEURONU

$$h(x) = g(u)v_1 + v_0 = \frac{1}{1+\exp[-\beta(w_1 x + w_0)]} v_1 + v_0$$



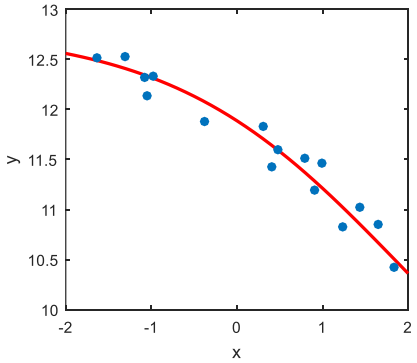
MODEL NEURONU

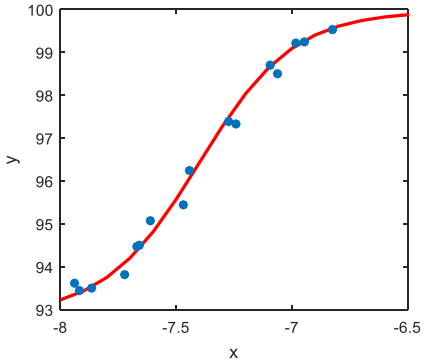


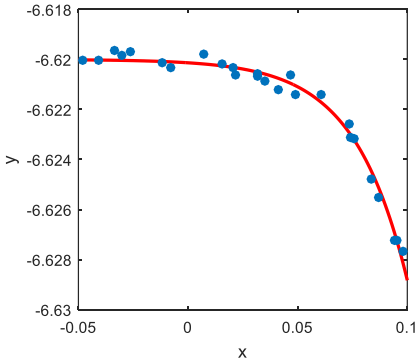


Video

MODEL NEURONU



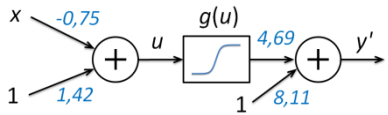


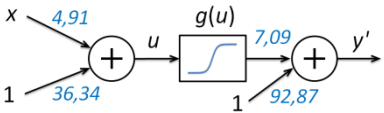


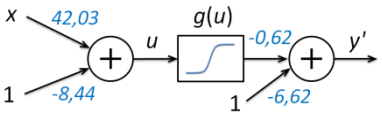
$$h(x) = \frac{1}{1+\exp[-\beta(-0{,}75x+1{,}42)]} \cdot 4{,}69 + 8{,}11$$

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

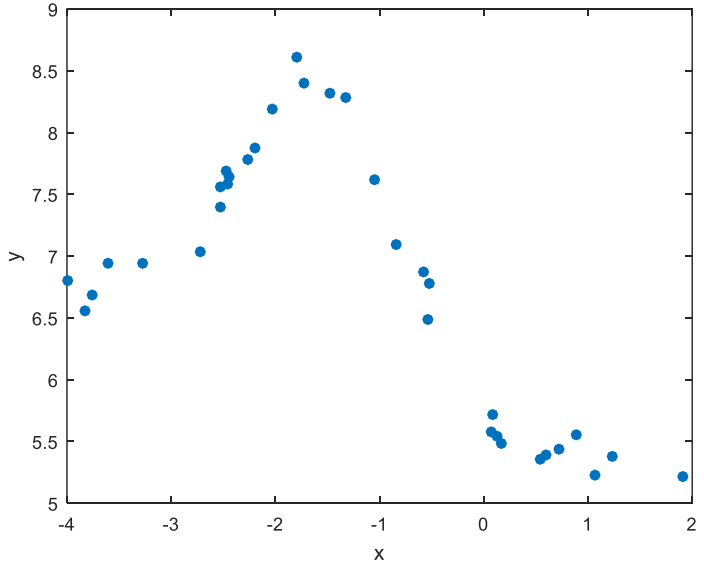
$$h(x) = \frac{1}{1+\exp[-\beta(42{,}03x-8{,}44)]} \cdot (-0{,}62) - 6{,}62$$



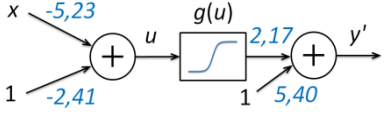


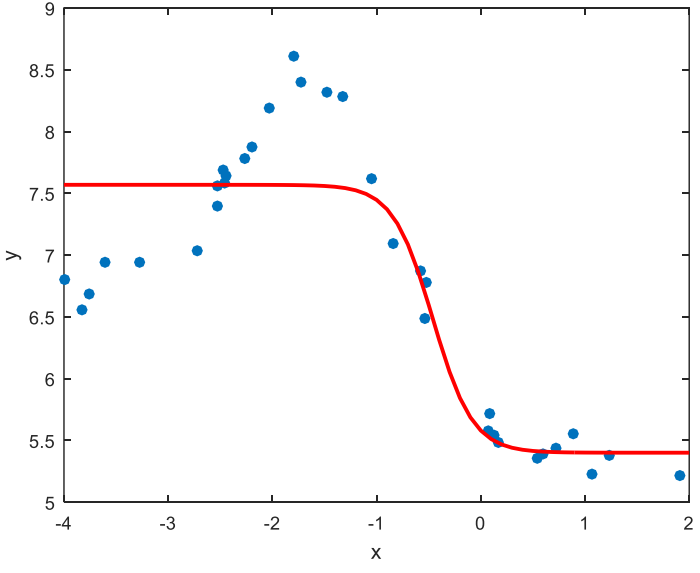


MODEL NEURONU



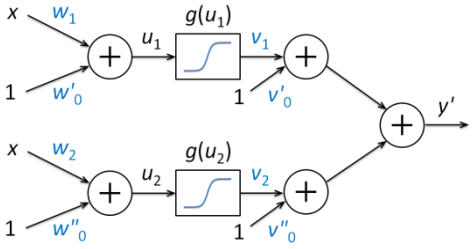
APROKSYMACJA FUNKCJI ZŁOŻONYCH





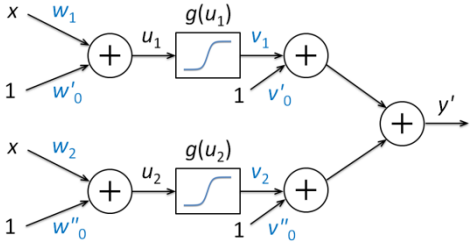
APROKSYMACJA FUNKCJI ZŁOŻONYCH

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

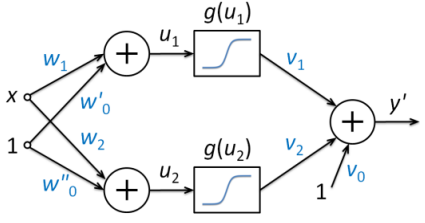


APROKSYMACJA FUNKCJI ZŁOŻONYCH

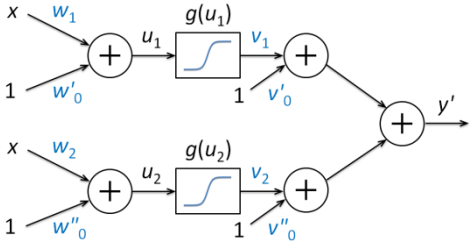
$$h(x) = [g(u_1)v_1 + v'_0] + [g(u_2)v_2 + v''_0]$$



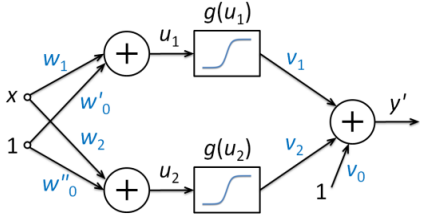
$$h(x) = g(u_1)v_1 + g(u_2)v_2 + v_0$$

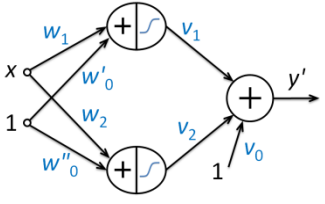


$$h(x) = [g(u_1)v_1 + v'_0] + [g(u_2)v_2 + v''_0]$$



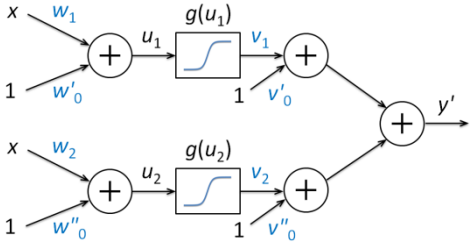
$$h(x) = g(u_1)v_1 + g(u_2)v_2 + v_0$$



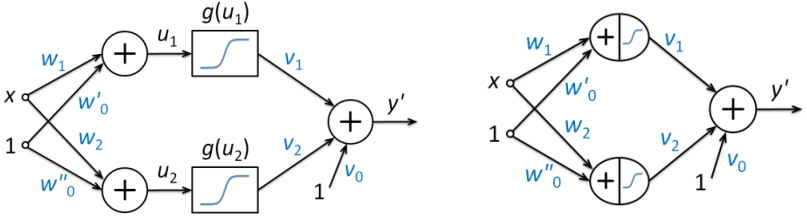


APROKSYMACJA FUNKCJI ZŁOŻONYCH

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

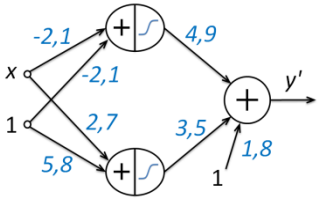


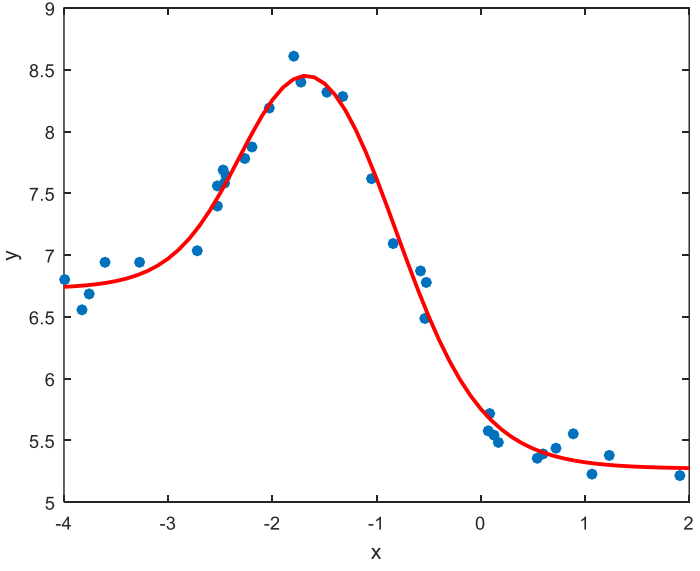
$$h(x) = g(u_1)v_1 + g(u_2)v_2 + v_0$$



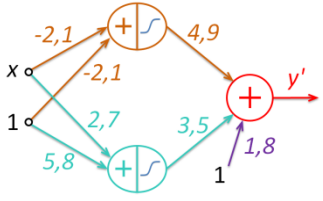
$$h(x) = g_1(x)v_1 + g_2(x)v_2 + v_0 = \frac{1}{1+\exp[-\beta(w_1 x + w'_0)]}v_1 + \frac{1}{1+\exp[-\beta(w_2 x + w''_0)]}v_2 + v_0$$

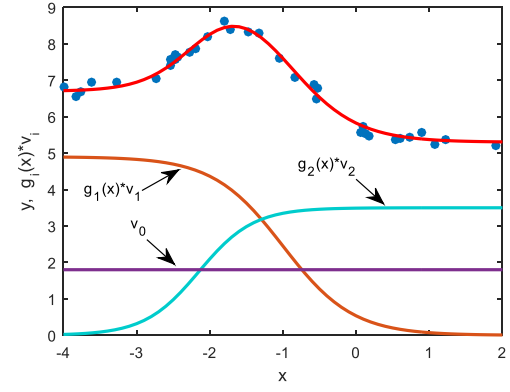
APROKSYMACJA FUNKCJI ZŁOŻONYCH

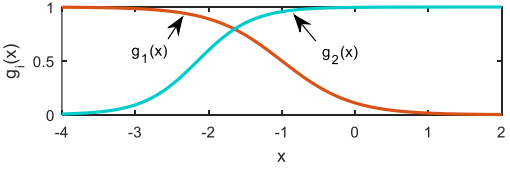




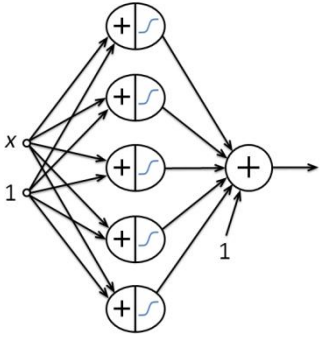
APROKSYMACJA FUNKCJI ZŁOŻONYCH

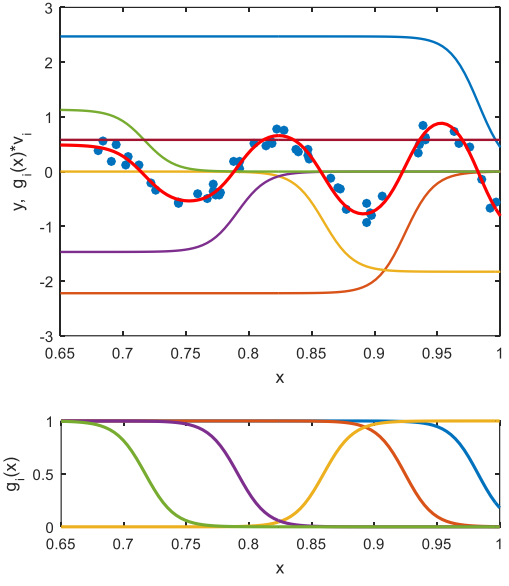




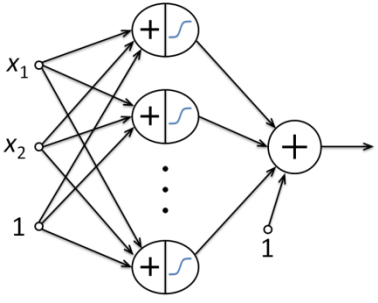


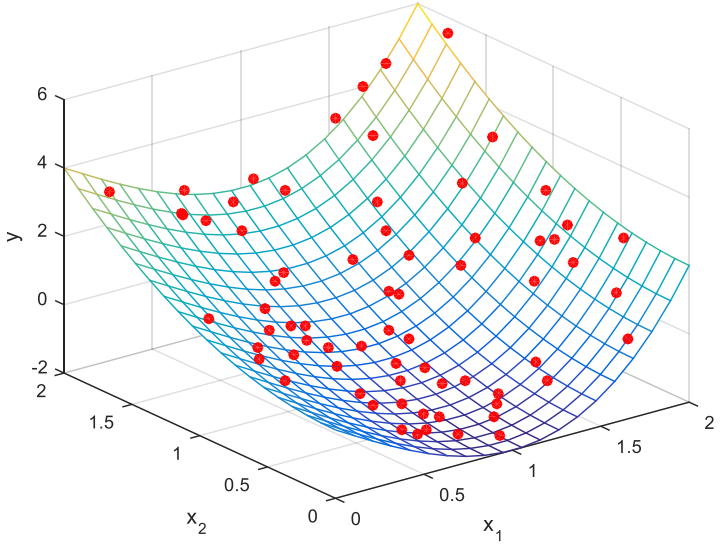
APROKSYMACJA FUNKCJI ZŁOŻONYCH





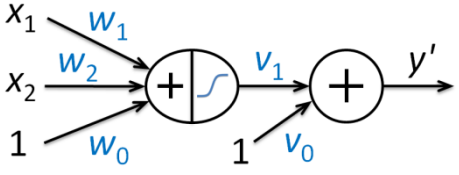
APROKSYMACJA FUNKCJI ZŁOŻONYCH



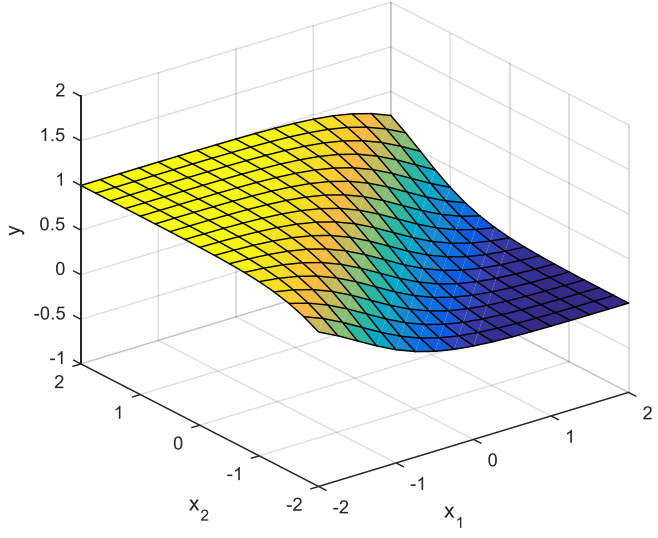


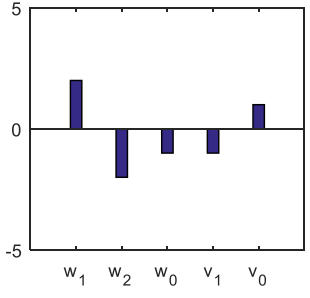
APROKSYMACJA FUNKCJI ZŁOŻONYCH

$$h(x) = \frac{1}{1 + \exp[-\beta(w_1 x_1 + w_2 x_2 + w_0)]} v_1 + v_0$$



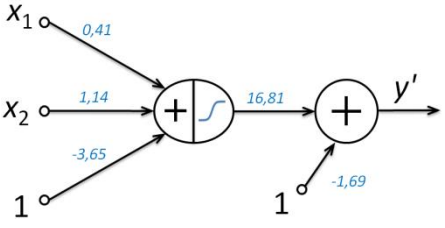
APROKSYMACJA FUNKCJI ZŁOŻONYCH



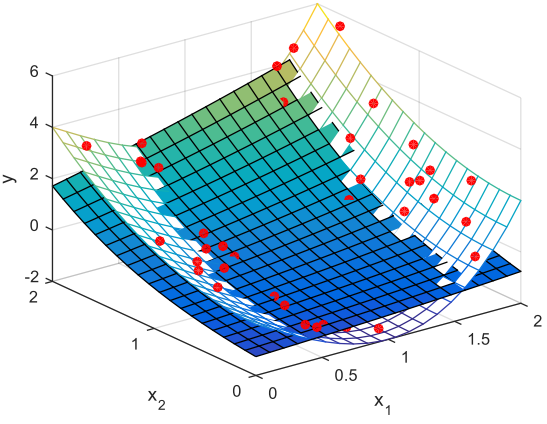


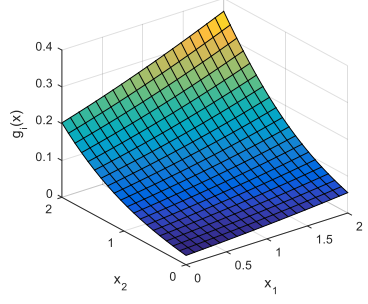
Video

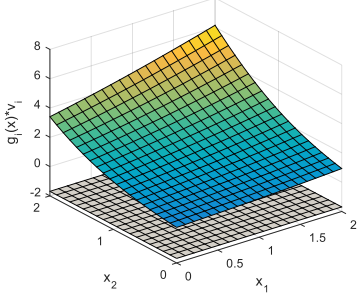
APROKSYMACJA FUNKCJI ZŁOŻONYCH



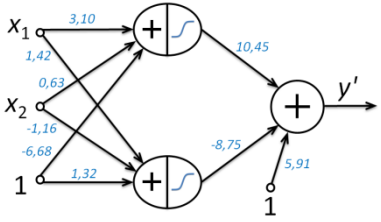
E = 0,9881



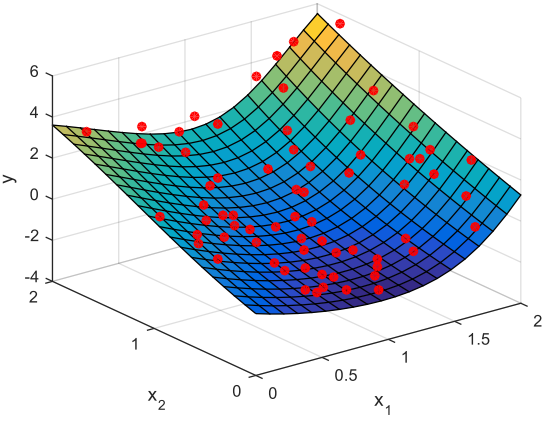


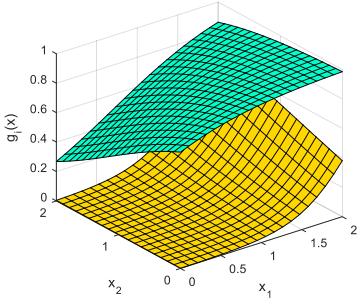


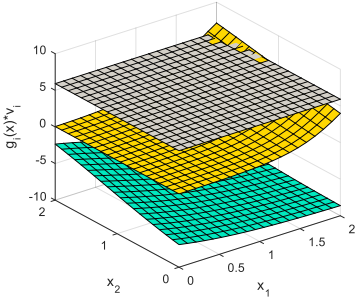
APROKSYMACJA FUNKCJI ZŁOŻONYCH



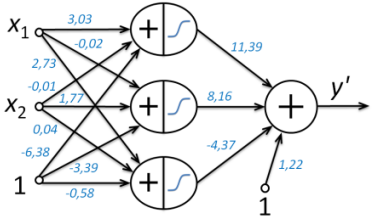
$E = 0{,}0524$



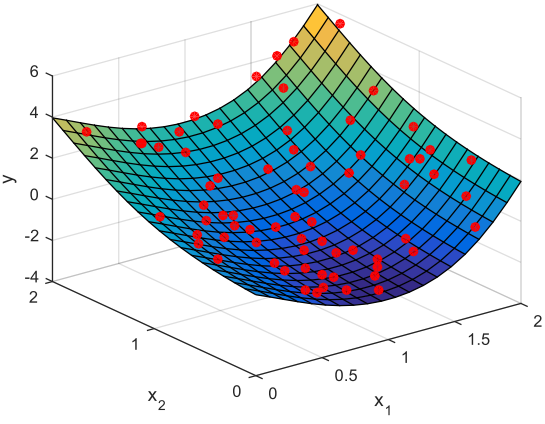


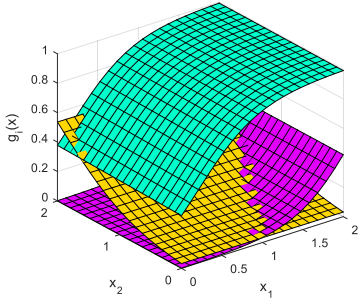


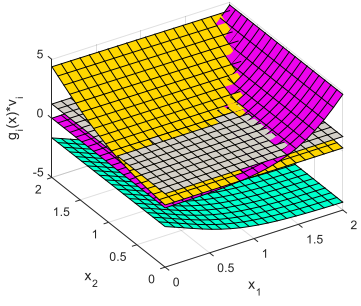
APROKSYMACJA FUNKCJI ZŁOŻONYCH



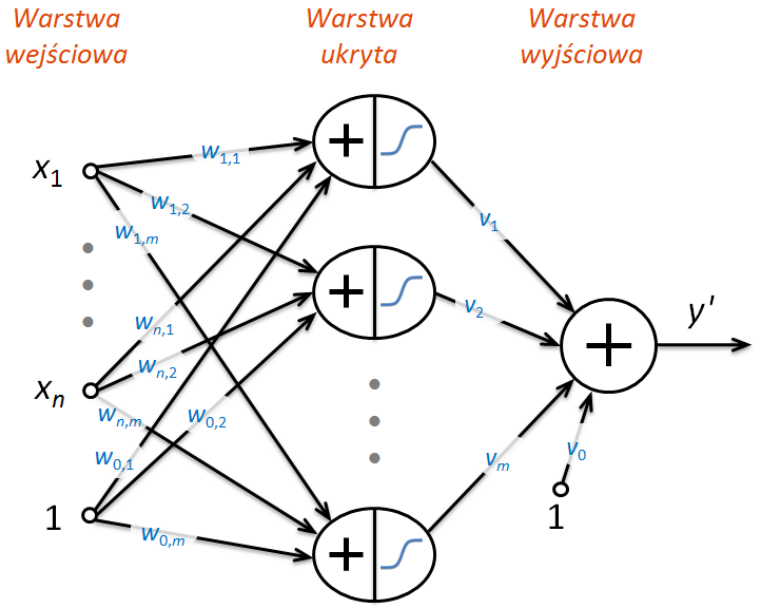
E = 0,0006





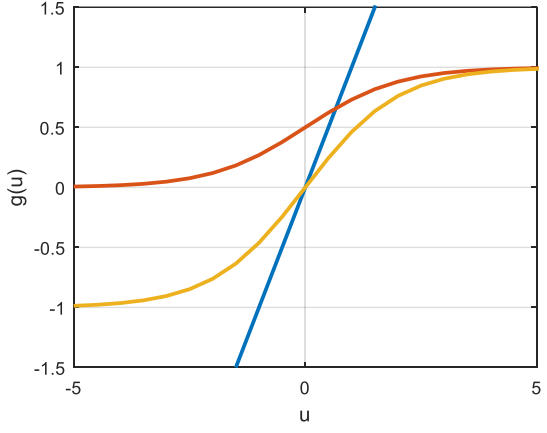


SZTUCZNA SIEĆ NEURONOWA



$$h(\mathbf{x}) = \sum_{i=1}^{m} g_i(\mathbf{x}) v_i + v_0$$

Funkcje aktywacji neuronów



– sigmoidalna unipolarna

$$g(u) = \frac{1}{1+\exp(-\beta u)}$$

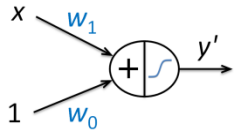
– sigmoidalna bipolarna

$$g(u) = \frac{2}{1+\exp(-\beta u)} - 1$$

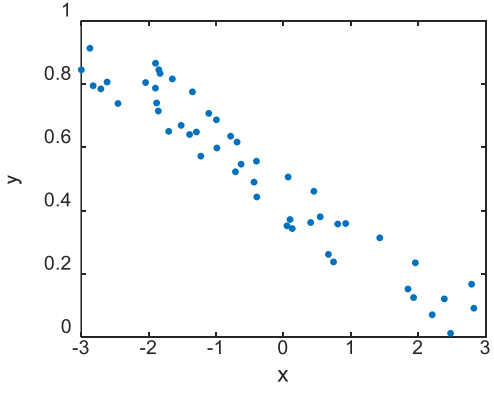
– liniowa

$$g(u) = u$$

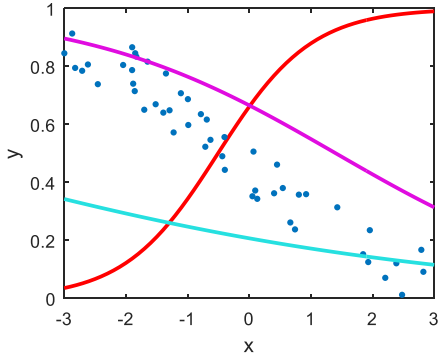
$$u = \sum_{i=1}^{n} w_i x_i + w_0 \qquad \beta = \text{const} \neq 0$$



$$h(x) = \frac{1}{1 + \exp[-\beta(w_1 x_1 + w_0)]}$$



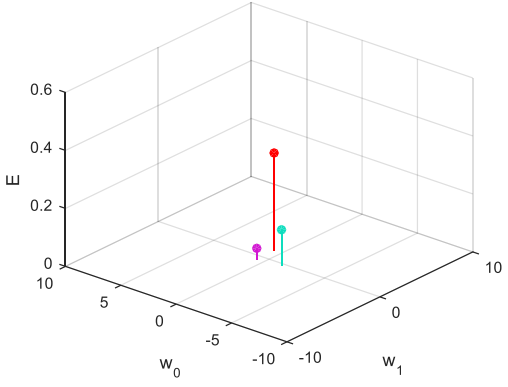
UCZENIE SIECI NEURONOWYCH



$w_1 = 1.32$ $w_0 = 0.67$

$w_1 = -0.49$ $w_0 = 0.69$

$w_1 = -0.23$ $w_0 = -1.34$

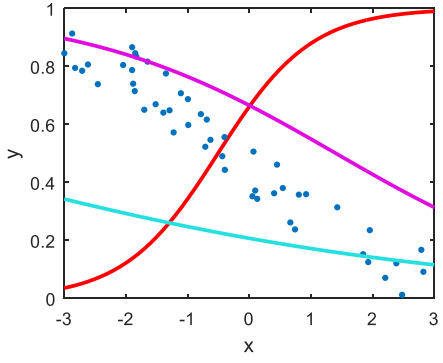


$E = 0.3352$

$E = 0.0378$

$E = 0.1236$

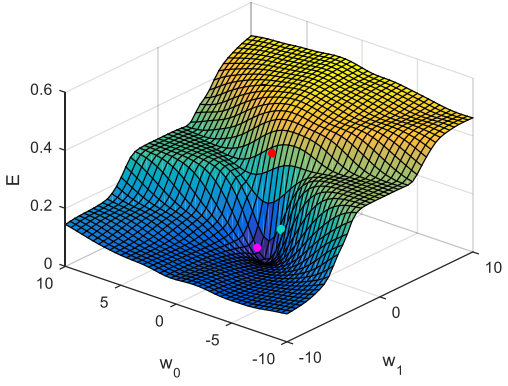
UCZENIE SIECI NEURONOWYCH



$w_1 = 1.32$ $w_0 = 0.67$

$w_1 = -0.49$ $w_0 = 0.69$

$w_1 = -0.23$ $w_0 = -1.34$



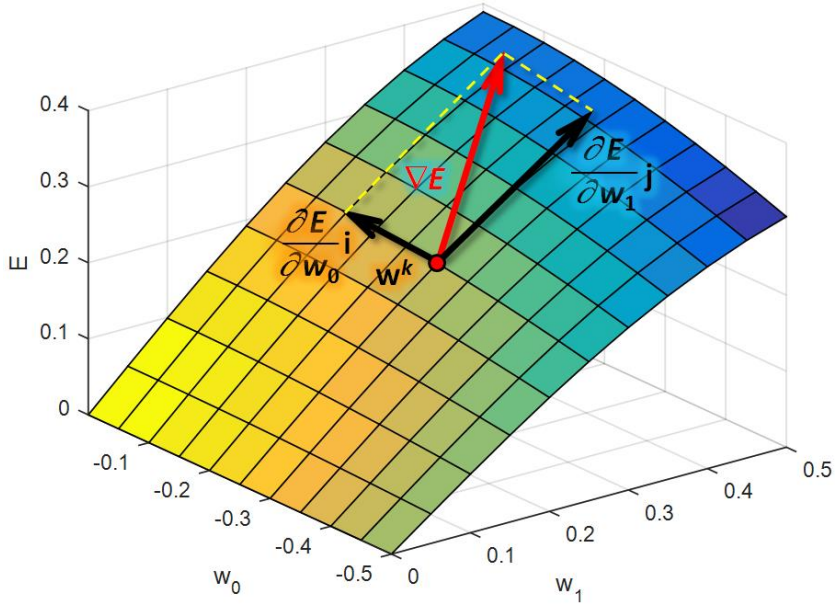
E = 0.3352

E = 0.0378

E = 0.1236

$$\nabla E(\mathbf{w}) = \frac{\partial E(\mathbf{w})}{\partial w_0}\mathbf{i} + \frac{\partial E(\mathbf{w})}{\partial w_1}\mathbf{j}$$

$$\nabla E(\mathbf{w}) = \left[\frac{\partial E(\mathbf{w})}{\partial w_0} \quad \frac{\partial E(\mathbf{w})}{\partial w_1}\right]$$

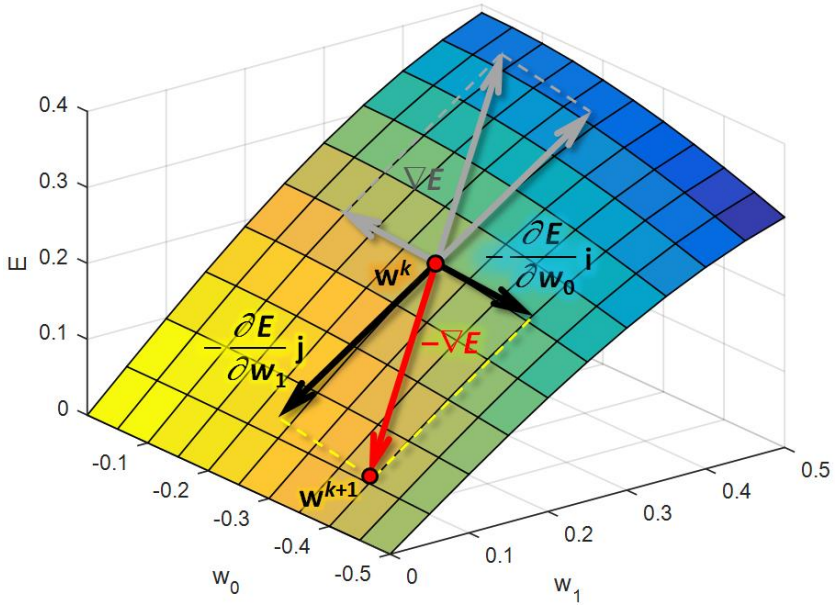


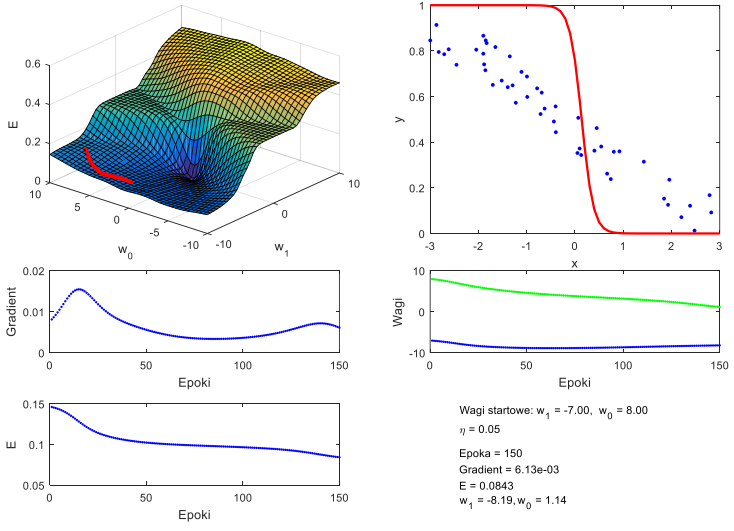
$$\nabla E(\mathbf{w}) = \frac{\partial E(\mathbf{w})}{\partial w_0}\mathbf{i} + \frac{\partial E(\mathbf{w})}{\partial w_1}\mathbf{j}$$

$$\nabla E(\mathbf{w}) = \left[\frac{\partial E(\mathbf{w})}{\partial w_0} \quad \frac{\partial E(\mathbf{w})}{\partial w_1}\right]^T$$

$$\mathbf{w}^{k+1} = \mathbf{w}^k - \eta \nabla E(\mathbf{w}^k)$$

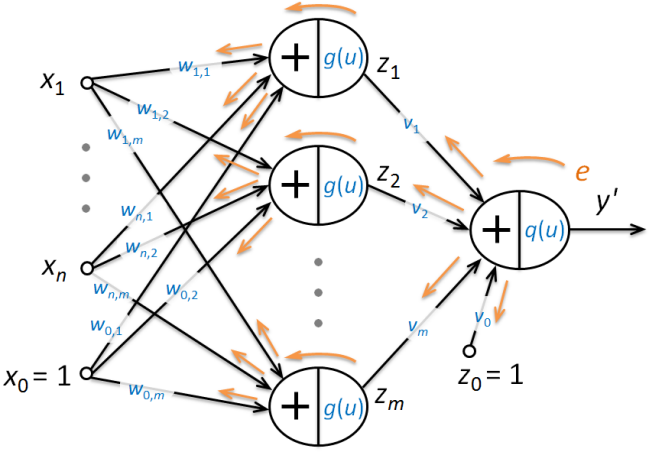
$$\mathbf{w}^{k+1} = \left[w_0^k - \eta \frac{\partial E(\mathbf{w}^k)}{\partial w_0} \quad w_1^k - \eta \frac{\partial E(\mathbf{w}^k)}{\partial w_1}\right]^T$$




E
w0
w1
Gradient
Epoki
E
Epoki
y
x
Wagi
Epoki
Wagi startowe: w1 = -7.00, w0 = 8.00
η = 0.05
Epoka = 150
Gradient = 6.13e-03
E = 0.0843
w1 = -8.19, w0 = 1.14

Video

Wsteczna propagacja błędu (backpropagation)



Wzory

Algorytm uczenia - skumulowana aktualizacja wag

1. Inicjalizacja: topologii sieci (liczba neuronów ukrytych), wag, η, kryterium stopu

2. Wykonuj cyklicznie do momentu spełnienia kryterium stopu (epoki k = 1, 2, ...)

    2.1. Wykonuj M-krotnie

        2.1.1. Wybierz punkt uczący ($\mathbf{x}_m$, y_m)

        2.1.2. Wyznacz odpowiedź sieci na ($\mathbf{x}_m$, y_m) - y'

        2.1.3. Oblicz błąd e = (y' - y)

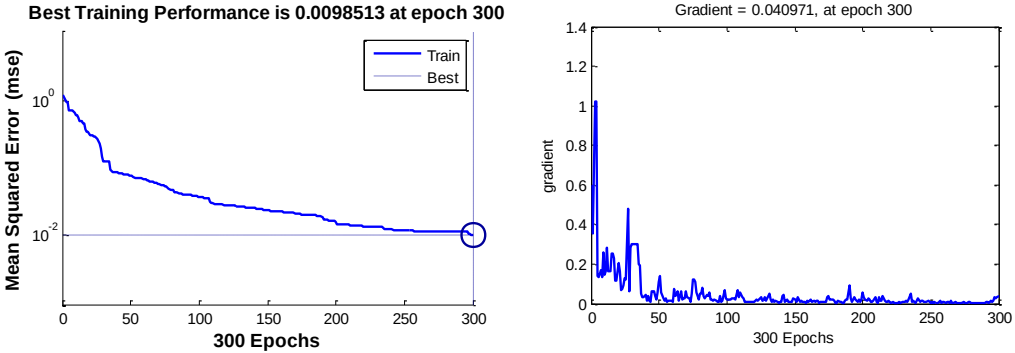
        2.1.4. Wyznacz poprawki wag: $\Delta v_i^k = \Delta v_i^k + \eta\delta'' z_i$, $\quad \Delta w_{j,i}^k = \Delta w_{j,i}^k + \eta\delta_i' x_j$

    2.2. Zmodyfikuj wagi: $v_i^{k+1} = v_i^k - \Delta v_i^k$, $\quad w_{j,i}^{k+1} = w_{j,i}^k - \Delta w_{j,i}^k$

Uwagi

- 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 (współczynnika uczenia).
- 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).

Przykładowy przebieg błędu i jego gradient w kolejnych epokach

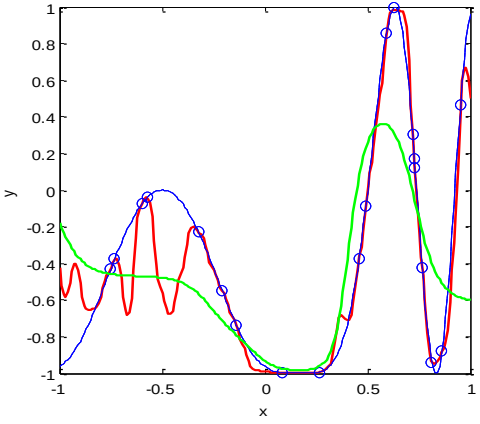


Po nauczeniu sieci sprawdzamy jej działanie na nowym zbiorze danych zwanym testowym. Błędy wyznaczone na tym zbiorze świadczą o jakości działania sieci.

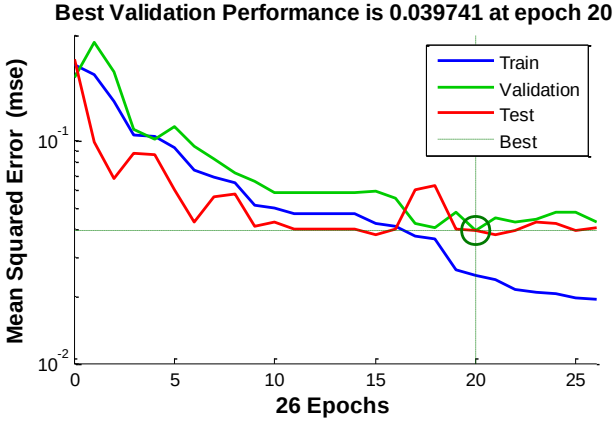
W procesie uczenia musimy rozstrzygnąć kilka problemów:

- 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.

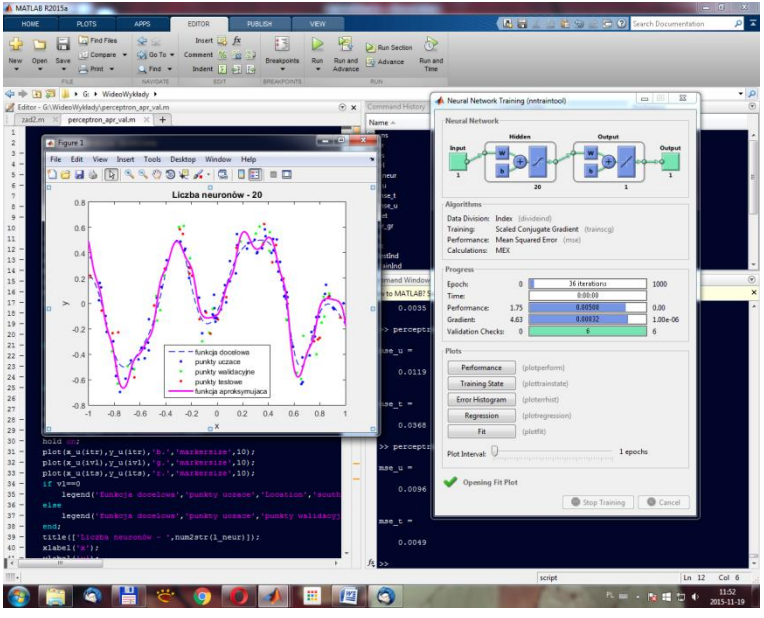
Jeśli trening jest zbyt krótki lub/i liczba neuronów zbyt mała sieć będzie niedouczona (duże błędy), zbyt długi trening lub/i zbyt duża liczba neuronów skutkuje przeuczeniem – błędy uzyskane na zbiorze uczącym będą bliskie 0, lecz błędy na zbiorze testowym okażą się duże.



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, wtedy przerywa się trening.



UCZENIE SIECI NEURONOWYCH

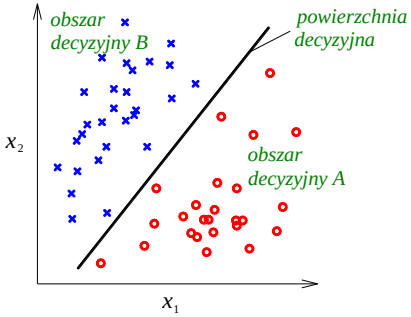


Video

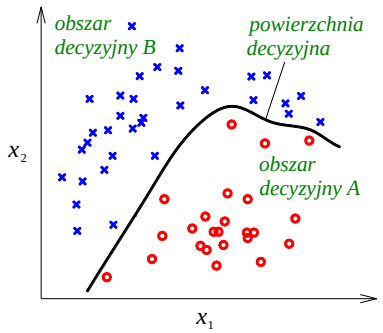
Sieci neuronowe do klasyfikacji

Typowe zadania klasyfikacji:

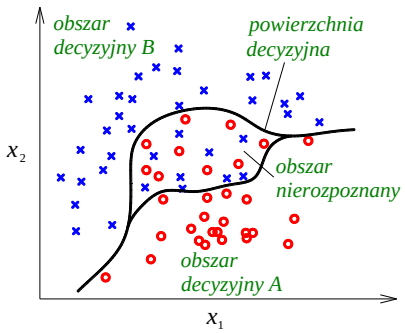
Zbiory separowalne liniowo



Zbiory separowalne nieliniowo

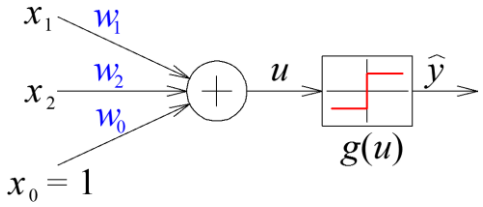


Zbiory nieseparowalne



NEURON JAKO KLASYFIKATOR

Neuron ze skokową funkcją aktywacji pełni funkcję klasyfikatora liniowego



Neuron realizuje funkcję:

$$g(u) = \begin{cases} +1 & \text{jeśli } u = w_1 x_1 + w_2 x_2 + w_0 \geq 0 \\ -1 & \text{jeśli } u = w_1 x_1 + w_2 x_2 + w_0 < 0 \end{cases}$$

+1 na wyjściu neuronu oznacza klasę A, –1 oznacza klasę B.

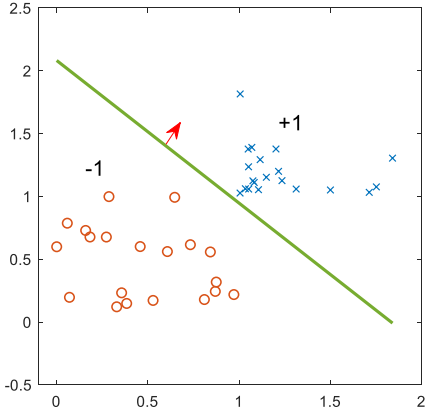
+1 pojawi się, gdy $w_1 x_1 + w_2 x_2 + w_0 \geq 0$, tzn.: $x_2 \geq -\frac{w_1}{w_2} x_1 - \frac{w_0}{w_2}$

–1 pojawi się, gdy: $x_2 < -\frac{w_1}{w_2} x_1 - \frac{w_0}{w_2}$

Powyższe nierówności definiują półpłaszczyzny – obszary decyzyjne obu klas. Linia decyzyjna rozdzielająca te obszary ma postać:

$$x_2 = -\frac{w_1}{w_2} x_1 - \frac{w_0}{w_2}$$

Jak widać wagi sieci określają współczynnik kierunkowy i wyraz wolny linii decyzyjnej. Wagi powinny być tak dobrane, aby prosta separowała obie klasy (jeśli to możliwe).



W regule perceptronowej wartości wag (a tym samym współczynników hiperpłaszczyzny dyskryminacyjnej) uzyskuje się w procesie uczenia z nauczycielem na podstawie zbioru trenującego.

Reguła klasyfikacji w przypadku dwóch klas ma postać (tzw. dychotomizator):

$$\mathbf{w}^T\mathbf{x} > 0 \quad \forall \mathbf{x} \in \text{klasy} +1$$
$$\mathbf{w}^T\mathbf{x} < 0 \quad \forall \mathbf{x} \in \text{klasy} -1$$

gdzie: $\mathbf{x} = [1, x_1, x_{2,\ldots,} x_n]^T$, $\mathbf{w} = [w_0, w_1, w_{2,\ldots,} w_n]^T$

Szukamy takich wag, które minimalizują kryterium:

$$J(\mathbf{w}) = \sum_{\mathbf{x} \in Z} \delta_{\mathbf{x}} \mathbf{w}^T \mathbf{x}$$

gdzie: Z – podzbiór przykładów niepoprawnie klasyfikowanych, $\delta_{\mathbf{x}} = -1$, jeśli $\mathbf{x} \in$ klasy –1 i $\delta_{\mathbf{x}} = +1$, jeśli $\mathbf{x} \in$ klasy +1.

Do znalezienia minimum można zastosować algorytm największego spadku gradientu. W kolejnych iteracjach tego algorytmu modyfikujemy współczynniki, do momentu osiągnięcia minimum kryterium (poprawnej klasyfikacji wszystkich przykładów uczących).

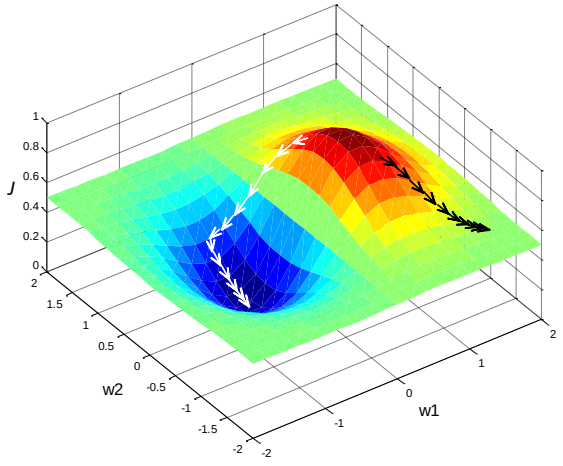
Gradient $\nabla J(\mathbf{w}) = \left[\frac{\partial J}{\partial w_1}, \frac{\partial J}{\partial w_2}, ..., \frac{\partial J}{\partial w_0}\right]$ ze znakiem ujemnym wskazuje kierunek "przesunięcia" wag:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla J(\mathbf{w})$$

gdzie $\eta > 0$ jest współczynnikiem uczenia.

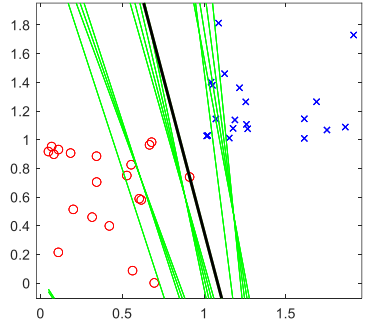
Ponieważ $\nabla J(\mathbf{w}) = \sum_{\mathbf{x} \in Z} \delta_{\mathbf{x}} \mathbf{x}$, perceptronową regułę uczenia możemy zapisać:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \sum_{\mathbf{x} \in Z} \delta_{\mathbf{x}} \mathbf{x}$$



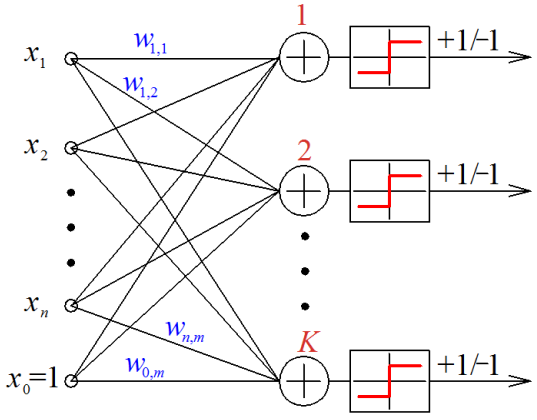
Algorytm

1. Wybierz losowo $\mathbf{w}$, ustal η.
2. Powtarzaj
   - 2.1. $Z = \emptyset$
   - 2.2. Powtarzaj dla $i = 1, 2, \ldots, N$
     - 2.2.1. Jeśli $\delta_x \mathbf{w}^T \mathbf{x}_i \geq 0$, to $Z = Z \cup \{\mathbf{x}_i\}$
   - 2.3. Jeśli $Z = \emptyset$, to zakończ
   - 2.4. $\mathbf{w} \leftarrow \mathbf{w} - \eta \sum_{x \in Z} \delta_x \mathbf{x}_i$



- Algorytm przerywa działanie, gdy znajdzie jakąkolwiek płaszczyznę separującą klasy.
- Jeśli przykłady są liniowo separowalne, algorytm zawsze znajduje rozwiązanie w skończonej liczbie kroków (jest zbieżny).

Gdy liczba klas jest większa od 2, wynosi *K*, stosujemy *K* neuronów. Każdy neuron reprezentuje inną klasę. Klasa *i*-ta sygnalizowana jest wartością +1 na wyjściu *i*-tego neuronu. Pozostałe neurony na wyjściach mają wartość –1.



W tym przypadku etykieta klasy ma postać wektora o *K* składowych: $\mathbf{y} = [y_1, y_2, \ldots, y_K]$, $y_i = \pm 1$.

Symbole klas można zakodować na mniejszej liczbie bitów, np. kl. 1: -1-1, kl. 2: -1+1, kl. 3: +1-1, kl. 4: +1+1.

Wagi sieci adaptuje się w procesie uczenia według wzoru:

$$\mathbf{w}_i \leftarrow \mathbf{w}_i + \frac{1}{2}\eta(y_i - \hat{y}_i)\mathbf{x},$$

gdzie *i* to numer neuronu (klasy).

Algorytm dla *K* klas

W przypadku *K* klas liniowo separowalnych oczekujemy:

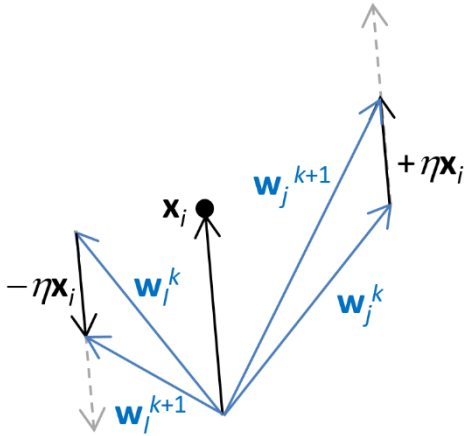
$$\mathbf{w}_j^T\mathbf{x} > \mathbf{w}_l^T\mathbf{x} \text{ dla każdego } \mathbf{x} \in \text{klasy } j, (l = 1,2,\dots,K, l \neq j)$$

1. Wybierz losowo $\mathbf{w}_j$ dla j = 1, 2, ..., K, ustal η.
2. Powtarzaj (k = 1, 2, …)
   - 2.1. Powtarzaj dla i = 1, 2, ..., N
     - 2.1.1. Jeśli $\boldsymbol{x}_i$ ma klasę j i dla pewnych l zachodzi $\mathbf{w}_l^T\mathbf{x} \geq \mathbf{w}_j^T\mathbf{x}$ (błędna klasyfikacja), to:

$$\mathbf{w}_j \leftarrow \mathbf{w}_j + \eta\mathbf{x}_i$$

$$\mathbf{w}_l \leftarrow \mathbf{w}_l - \eta\mathbf{x}_i$$

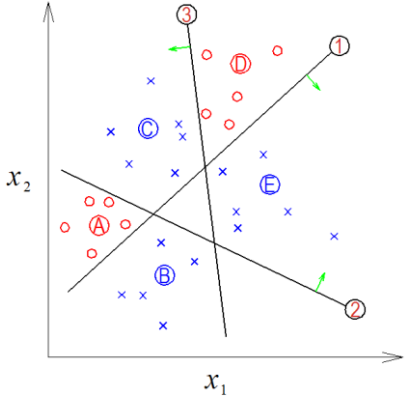
   - 2.2. Jeśli w pętli 2.1 nie nastąpiła modyfikacja żadnych wag $\mathbf{w}$, to zakończ.

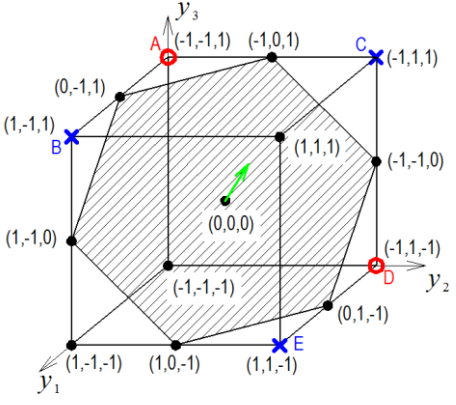


KLASYFIKACJA DANYCH LINIOWO NIESEPAROWALNYCH

Każdy neuron ze skokową funkcją aktywacji dzieli liniowo płaszczyznę na dwie części, tak aby wydzielone obszary decyzyjne zawierały przykłady z jednej klasy.

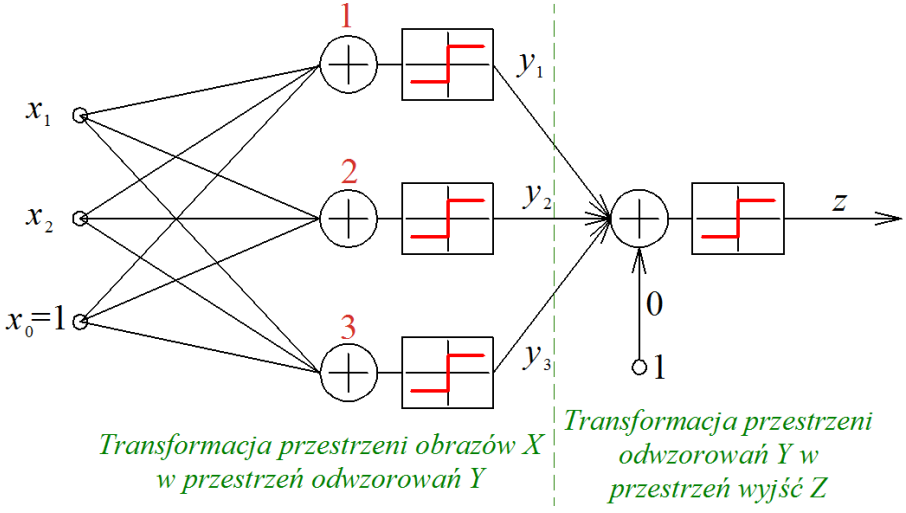
Na wyjściu m neuronów otrzymujemy wektor $\mathbf{y} = [y_1, y_2, \ldots, y_m]$, $y_i = \pm 1$. Zachodzi transformacja n-wymiarowych przykładów **x** (przestrzeń obrazów) w m-wymiarową przestrzeń wektorów wyjściowych **y** (przestrzeń odwzorowań). Przykłady w tej nowej przestrzeni są separowalne za pomocą płaszczyzny realizowanej przez neuron drugiej warstwy.





KLASYFIKACJA DANYCH LINIOWO NIESEPAROWALNYCH

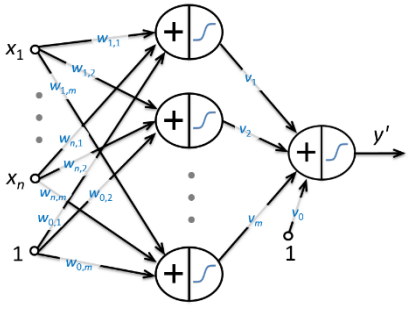
Model sieci dwuwarstwowej do klasyfikacji obrazów liniowo nieseparowalnych.

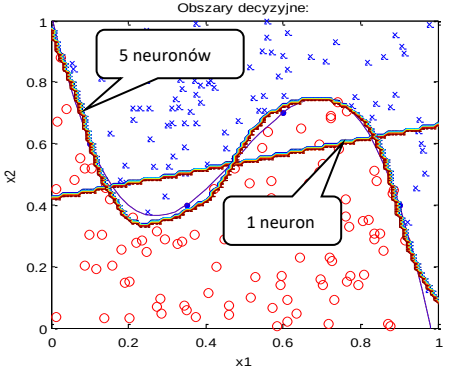


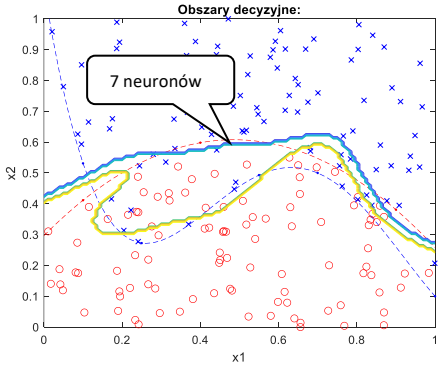
SIECI NEURONOWE DO KLASYFIKACJI DANYCH

Do tworzenia złożonych, nieliniowych powierzchni decyzyjnych stosuje się sieci wielowarstwowe z nieliniowymi funkcjami aktywacji.

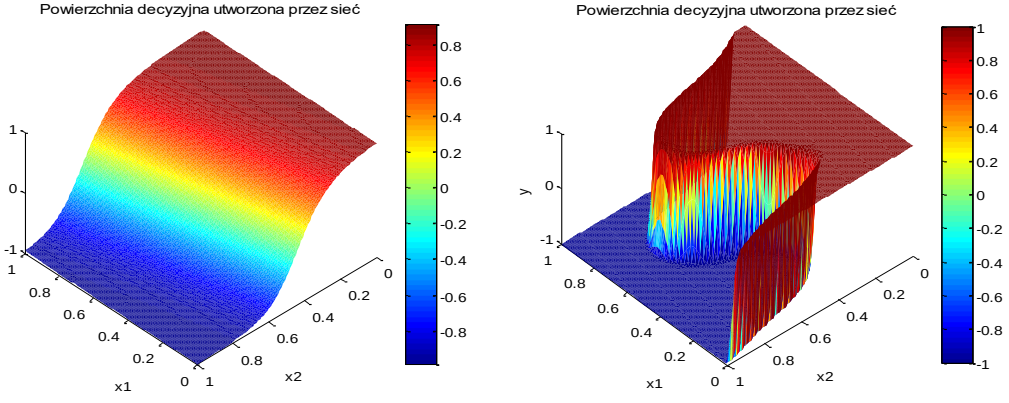
Im więcej neuronów tym powierzchnie decyzyjne mogą być bardziej złożone.







W trakcie treningu minimalizowany jest błąd średniokwadratowy pomiędzy pożądanym numerem klasy (+1 lub –1), a odpowiedzią sieci. Przy sigmoidalnej bipolarnej funkcji aktywacji neuronu wyjściowego odpowiedź sieci jest liczbą rzeczywistą z zakresu od –1 do +1. Dla zamieszczonych powyżej danych uczących powierzchnie odpowiedzi sieci z jednym i pięcioma neuronami wyglądają następująco:



Linie decyzyjne powstają z przekroju powierzchni odpowiedzi sieci płaszczyzną $y = 0$, co oznacza, że jeśli sieć daje odpowiedź dodatnią przyjmuje się klasę +1, a jeśli ujemną – klasę –1.

Można przyjąć inną zasadę – jeśli odpowiedź sieci jest powyżej +0,8 oznacza to klasę +1, a jeśli poniżej –0,8 – oznacza to klasę –1. Odpowiedzi w przedziale [–0,8; +0,8] uznawane są jako brak decyzji (obszar nierozpoznany). W takim przypadku linie decyzyjne utworzone przez sieć z jednym neuronem (wariant 3) mogą wyglądać tak: