Politechnika Rzeszowska
Wydział Elektrotechniki i Informatyki
Katedra Informatyki i Automatyki

Sztuczna Inteligencja

Projekt

Realizacja sieci neuronowej uczonej algorytmem wstecznej propagacji błędu z przyśpieszeniem metodą adaptacyjnego współczynnika uczenia uczącą się rozpoznawania rodzaju naczyń szklanych

Wykonał:
Artur Przystaś
156321
II EF-DI

Rzeszów 2019

Spis treści

1 Opis problemu

1.1 Cel projektu

Celem projektu jest opracowanie i zrealizowanie sieci neuronowej uczonej algorytmem wstecznej propagacji błędu z przyśpieszeniem metodą adaptacyjnego współczynnika uczenia. Sieć neuronowa ma za zadnie zidentyfikować rodzaj naczyń szklanych. Problem został rozwiązany w programie Matlab R2019a, przy użyciu funkcji traingda. Prace obejmowały opracowanie wstępu teoretycznego, przygotowanie skryptu wielowarstwowej sieci neuronowej, przeprowadzenie eksperymentów, opracowanie na ich podstawie danych i wniosków.

1.2 Opis danych

Sieć ma za zadnie identyfikować rodzaj naczyń szklanych za pomocą informacji o:

1. Współczynniku załamania
2. Procentowej zawartości tlenku:
    - sodu,
    - magnezu,
    - glinu,
    - krzemu,
    - potasu,
    - wapnia
    - baru,
    - żelaza

Naczynia mogą należeć do następujących klas:

1. Szyba okienna poddana procesowi float
2. Szyba okienna nie poddana procesowi float
3. Szyba samochodowa poddana procesowi float
4. Szyba samochodowa nie poddana procesowi float
5. Pojemnik
6. Zastawa stołowa
7. Reflektor

Dostarczone dane posiadają 219 przypadków. Ich dystrybucja wygląda następująco:

1. Szyby 163
    (a) Poddane procesowi float
        i. Szyby okienne: 70
        ii. Szyby samochodowe: 17
    (b) Nie poddane procesowi float: 76
        i. Szyby okienne: 76
        ii. Szyby samochodowe: 0

2. Pozostałe
    (a) Pojemniki: 13
    (b) Zastawy stołowe: 9
    (c) Reflektory: 29

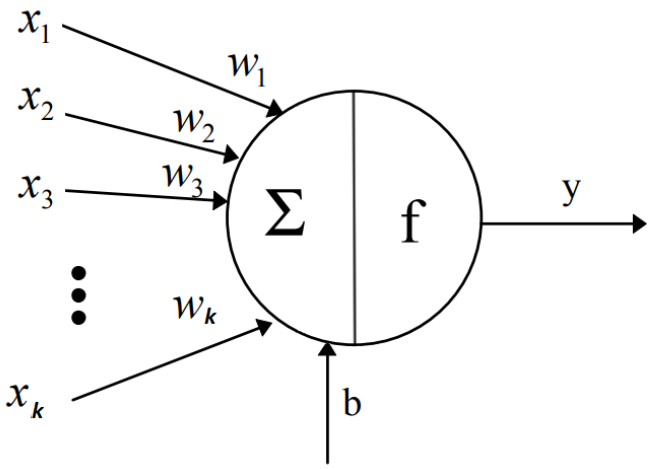
Otrzymane dane zostały, w czasie wykonywania ćwiczenia, poddane normalizacji, tak by wartości znajdowały się w przedziale [-1;1].

2 Zagadnienia teoretyczne

2.1 Sieć neuronowa

Sieć neuronowa jest oparta na strukturze matematycznej i programowym, bądź sprzętowym modelu realizującym obliczenia przez warstwy sztucznych neuronów. Wykonują one obliczenia na wejściu a następnie przekazują te informacje do kolejnego neuronu tworząc sieć. Pierwowzorem sieci neuronowej jest ludzki mózg.

2.2 Model neuronu



Rysunek 1: Model neuronu

Neuron to podstawowy element sieci neuronowej. Odpowiada on za przetwarzanie i przesyłanie danych do kolejnych neuronów. Jego elementy to:

x_k - sygnały wejściowe
w_k - wagi wejść
y - wyjście
b - przesuniecie (bias)
f - funkcja aktywacji
k - liczba wejść

Sygnał wyjściowy neuronu y jest określony zależnością:

$$y = f(\sum_{i=1}^{k} w_i x_i + b) \tag{1}$$

Ponieważ do wyznaczenia wartości wyjścia y służy wzór:

$$y = f(n) \tag{2}$$

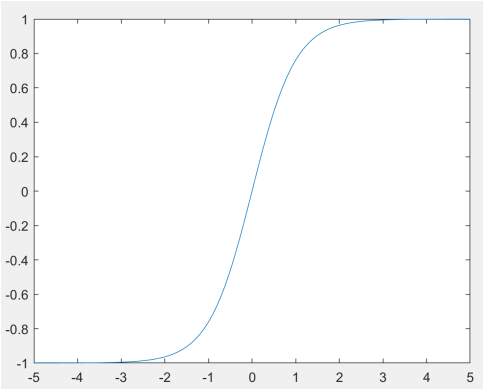
Gdzie

$$n = \sum_{i=1}^{k} w_i x_i + b \tag{3}$$

Oznacza to, że sygnały wejściowe są mnożone przez swoje wagi, później sumowane. Otrzymana suma wraz z wartością biasu jest wejściem funkcji aktywacji, która zwraca wartość wyjściową neuronu.

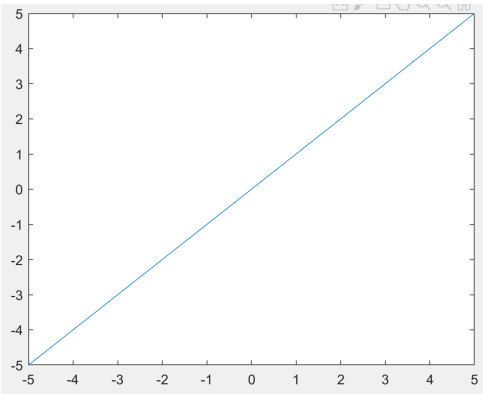
2.3 Funkcja aktywacji

Odpowiedzialna jest za obliczenie sygnału wyjściowego neuronu. Różne funkcje są stosowane jako funkcje aktywacji, w zależności od potrzeb danej sieci neuronowej. W opracowanej sieci neuronowej zastosowane zostały funkcja sigmoidalna dla warstwy pierwszej i drugiej



Rysunek 2: Wykres funkcji sigmoidalnej wykonany przy pomocy Matlab R2019a

A także funkcja liniowa dla warstwy trzeciej:

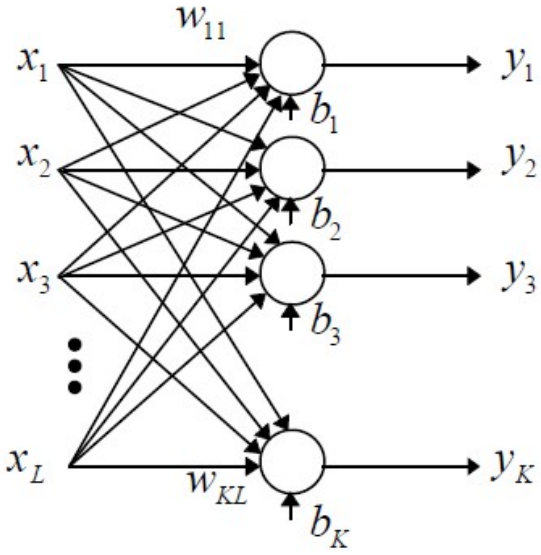


Rysunek 3: Wykres funkcji liniowej wykonany przy pomocy Matlab R2019a

Są to najczęściej wykorzystywane funkcje aktywacji. Są one zaimplementowane w Matlabie R2019a. Funkcja sigmoidalna posiada nazwę tansig, liniowa zaś purelin.

2.4 Sieć neuronowa jednowarstwowa

Projekt zakłada wykorzystanie sieci wielowarstwowej. Omówienie sieci jednowarstwowej pozwoli jednak łatwiej zrozumieć koncepcje warstw uczenia, ponieważ jest ona elementem sieci wielowarstwowej.



Rysunek 4: Schemat sieci jednokierunkowej jednowarstwowej

Działanie sieci jednowarstwowej można opisać następująco

$$y = f\left(wx + b\right) \tag{4}$$

gdzie:

$y = [y_1, y_2, ..., y_K]^T$ - wektor sygnałów wyjściowych

$x = [x_1, x_2, ..., x_L]^T$ - wektor sygnałów wejściowych

$b = [b_1, b_2, ..., b_K]^T$ - wektor przesunięć

$w = \begin{bmatrix} w_{11} & w_{12} & w_{13} & \dots & w_{1L} \\ w_{21} & w_{22} & w_{23} & \dots & w_{2L} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_{K1} & w_{K2} & w_{K3} & \dots & w_{KL} \end{bmatrix}$ - macierz wag

Proces uczenia ma za zadanie tak wybrać wagi, by z zadaną dokładnością odwzorować dane wejściowe w wyjściowe. Zmiana wag odbywa się w cyklach zwanych epokami. Dla j-tej wagi i-tego neuronu można to zapisać zależnością:

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t) \tag{5}$$

gdzie:

t - numer cyklu

Rozważany w projekcie algorytm jest algorytmem uczenia z nauczycielem. Zatem każdemu wektorowi wejściowemu $x = [x_1, x_2, ..., x_L]^T$ towarzyszy pożądany wektor sygnałów wyjściowych

$\hat{y} = [\hat{y}_1, \hat{y}_2, ..., \hat{y}_K]^T$. Para wektorów $(x, \hat{y})$ to para ucząca. Jeżeli sieć nie jest nauczona, to sygnał $\hat{y}$ różni się od sygnału wyjściowego y. Błąd popełniany przez sieć dla każdego wyjścia:

$$e_i = y_i - \hat{y}_i \tag{6}$$

W czasie uczenia dąży się do uzyskania zgodności y z wartościami wymaganymi $\hat{y}$, Problem ten można sprowadzić do minimalizacji określonej funkcji celu. Przyjmując, że jest ona błędem średniokwadratowym wyznaczanym dla wszystkich Kneuronów otrzymujemy:

$$E = \frac{1}{2}\sum_{i=1}^{K} e_i^2 \tag{7}$$

Ponieważ $E = E(w)$ minimum poszukiwać można metodą gradientową. Można to wykonać metodą największego spadku. Wektor wag przyrostu wyraża się następująco:

$$\Delta w = -\eta \nabla E(w) \tag{8}$$

gdzie:

∇ - gradient,

η - współczynnik uczenia.

Znak minus znajduje się we wzorze (8) ponieważ gradient ∇E wskazuje kierunek najszybszego wzrostu funkcji, w minimalizacji chodzi zaś o kierunek odwrotny, czyli jak najszybszy spadek. Dla i-tego neuronu i j-tej wagi orzymujemy:

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} \tag{9}$$

Uwzględniając uwikłanie zależności E od w_{ij} poprzez y_i i n_i, czyli $E = E(y_i(n_i(w_{ij})))$ pochodną możemy zapisać:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial y_i}\frac{\partial y_i}{\partial n_i}\frac{\partial n_i}{\partial w_{ij}} \tag{10}$$

Uwzględniając zależności (6) i (7) otrzymujemy:

$$\frac{\partial E}{\partial y_i} = y_i - \hat{y}_i = e_i \tag{11}$$

Uwzględniając zależność (3) otrzymujemy:

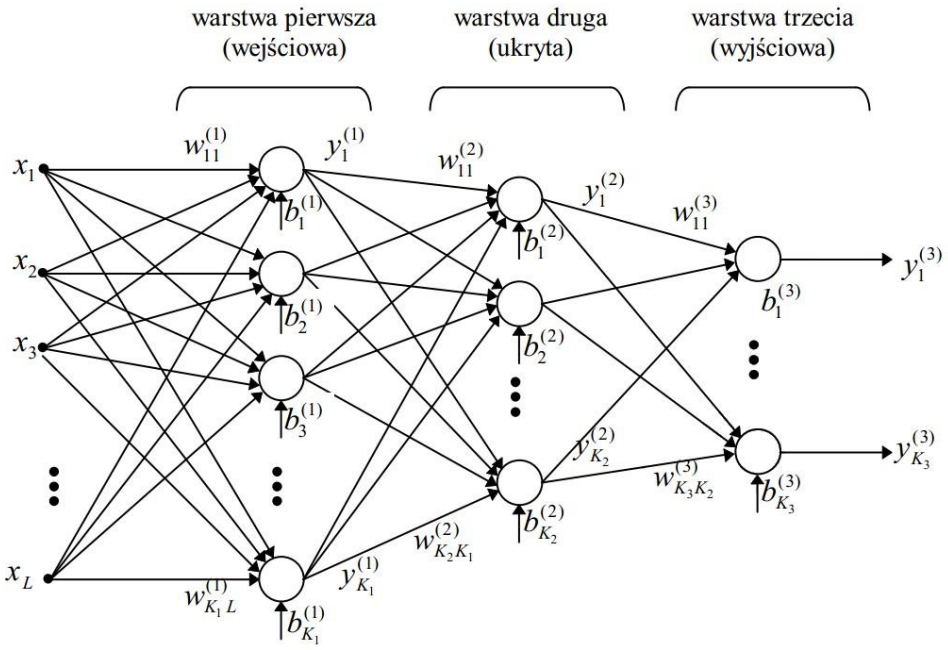
$$\frac{\partial n_i}{\partial w_{ij}} = x_j \tag{12}$$

Wstawiając otrzymane pochodne do równania (10), otrzymane równanie do zależności (9) otrzymujemy:

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} = \eta(\hat{y}_i - y_i)\frac{\partial y_i}{\partial n_i}x_j \tag{13}$$

gdzie $\frac{\partial y_i}{\partial n_i} = f'(n_i)$ równe jest pochodnej cząstkowej i-tej funkcji aktywacji po i-tym łącznym pobudzeniu neuronu.

2.5 Sieć neuronowa wielowarstwowa



Rysunek 5: Schemat sieci jednokierunkowej wielowarstwowej

Sieci jednokierunkowe składają się z neuronów przesyłających sygnały do kolejnych warstw: Z warstwy wejściowej do warstw ukrytych, z warstw ukrytych do warwowy wyjściowej.
Na rysunku (5) przedstawiono sieć trójwarstwową. Sygnały wejściowe warstwy są sygnałami wyjściowymi warstwy poprzedniej. Pomiędzy warstwami neuronów zastosowane są połączenia jeden do wielu.
Warstwy posiadają swoją macierz wag (w), wektor przesunięć (b), funkcję aktywacji (f) a także sygnał wyjściowy (y). Do zapisów dodano numery warstw: np. zapis $w^{(3)}$ oznacza macierz wag trzeciej warstwy.
Sposób działania poszczególnych warstw można opisać następująco:

$$\begin{aligned} y^{(1)} &= f^{(1)}(w^{(1)}x + b^{(1)}) \\ y^{(2)} &= f^{(2)}(w^{(2)}y^{(1)} + b^{(2)}) \\ y^{(3)} &= f^{(3)}(w^{(3)}y^{(2)} + b^{(3)}) \end{aligned} \tag{14}$$

Działanie sieci to więc:

$$y^{(3)} = f^{(3)}\left(w^{(3)}f^{(2)}\left(w^{(2)}f^{(1)}\left(w^{(1)}x + b^{(1)}\right) + b^{(2)}\right) + b^{(3)}\right) \tag{15}$$

2.6 Wsteczna propagacja błędu

Wsteczna propagacja błędu jest mechanizmem służącym uczeniu sieci neuronowej. Polega na zaktualizowaniu wartości wag oraz przesunięć dla każdego neuronu, zaczynając od warstw wyższych, kierując się w stronę niższych.
Dla sieci trójwarstwowej opisywana jest następująco:

$$
\begin{aligned}
E &= \frac{1}{2}\sum_{i_3=1}^{K_3} e_{i_3}^2 = \frac{1}{2}\sum_{i_3=1}^{K_3}\left(y_{i_3}^{(3)} - \hat{y}_{i_3}\right)^2 = \\
&= \frac{1}{2}\sum_{i_3=1}^{K_3}\left(f^{(3)}\left(\sum_{i_2=1}^{K_2} w_{i_3 i_2}^{(3)} y_{i_2} + b_{i_3}^{(3)}\right) - \hat{y}_{i_3}\right)^2 = \\
&= \frac{1}{2}\sum_{i_3=1}^{K_3}\left(f^{(3)}\left(\sum_{i_2=1}^{K_2} w_{i_3 i_2}^{(3)} f^{(2)}\left(\sum_{i_1=1}^{K_1} w_{i_2 i_1}^{(2)} y_{i_1} + b_{i_2}^{(2)}\right) + b_{i_3}^{(3)}\right) - \hat{y}_{i_3}\right)^2 = \\
&= \frac{1}{2}\sum_{i_3=1}^{K_3}\left(f^{(3)}\left(\sum_{i_2=1}^{K_2} w_{i_3 i_2}^{(3)} f^{(2)}\left(\sum_{i_1=1}^{K_1} w_{i_2 i_1}^{(2)} f^{(1)}\left(\sum_{j=1}^{L} w_{i_1 j}^{(1)} x_j + b_{i_1}^{(1)}\right) + b_{i_2}^{(2)}\right) + b_{i_3}^{(3)}\right) - \hat{y}_{i_3}\right)^2
\end{aligned}
\tag{16}
$$

Wyliczanie wag rozpoczyna się od wyliczenia wag neuronów warstwy wyjściowej. Na podstawie tych obliczeń obliczane są wagi warstwy wcześniejszej o jeden itd.

Dla drugiej warstwy sieci z rysunku (5) otrzymujemy:

$$
\frac{\partial E}{\partial w_{i_2 i_1}^{(2)}} = \sum_{i_3=1}^{K_3}\left(e_{i_3} f'^{(3)}(n_{i_3}) w_{i_3 i_2}^{(3)}\right) f'^{(2)}(n_{i_2}) y_{i_1}^{(1)} \tag{17}
$$

Dla warstwy pierwszej otrzymujemy:

$$
\frac{\partial E}{\partial w_{i_1 j}^{(1)}} = f'^{(1)}(n_{i_1}) x_j \sum_{i_2=1}^{K_2}\left(f'^{(2)}(n_{i_2}) w_{i_2 i_1}^{(2)} \sum_{i_3=1}^{K_3}\left(e_{i_3} f'^{(3)}(n_{i_3}) w_{i_3 i_2}^{(3)}\right)\right) \tag{18}
$$

2.7 Adaptacyjny współczynnik uczenia

Jest to metoda pozwalająca na przyspieszenie procesu uczenia. Polega na zmienianiu współczynnika uczenia η w zależności od błędu popełnianego przez sieć.
W metodzie tej błąd najczęściej wyraża się w postaci sumarycznego błędu kwadratowego SSE:

$$SSE(t) = \sum_{i=1}^{M} \Big(y_i(t) - \hat{y}_i(t) \Big)^2 \tag{19}$$

bądź błędu średniokwadratowego MSE:

$$MSE(t) = \frac{1}{M} \sum_{i=1}^{M} \Big(y_i(t) - \hat{y}_i(t) \Big)^2 \tag{20}$$

Zaktualizowanie współczynnika uczenia w czasie t+1 przedstawia się wzorem:

$$\eta(t+1) = \begin{cases} \eta(t) \cdot \xi_d & gdy \quad ERR(t) > er \cdot ERR(t-1) \\ \eta(t) \cdot \xi_i & gdy \quad ERR(t) < ERR(t-1) \\ \eta(t) & gdy \quad ERR(t-1) \leq ERR(t) \leq er \cdot ERR(t-1) \end{cases} \tag{21}$$

gdzie:

η - współczynnik uczenia
ξ_d - współczynnik zmniejszania η
ξ_i - współczynnik zwiększania η
er - dopuszczalny błąd sieci
ERR - wartość błędu SSE bądź MSE

Gdy nowy błąd jest istotnie większy od poprzedniego - współczynnik jest zmniejszany.
W przypadku gdy nowy błąd jest istotnie mniejszy – współczynnik jest zwiększany.
Jeżeli błąd w kroku k zwiększył się w porownaniu do błędu z kroku k-1, ale nie w sposób istotny (4% dla wartości er równej 1.04) - współczynnik nie zmienia się.

3 Skrypt

Skrypt wykorzystany do realizacji projektu został zrealizowany przy pomocy programu Matlab w wersji R2019a. Implementując adaptacyjny współczynnik uczenia posłużono się sumarycznym błędem kwadratowym SSE. Poniżej przedstawiony został kod na którym bazowały przeprowadzone eksperymenty. Był on modyfikowany w zależności od potrzeb danego eksperymentu.

Listing 1: Skrypt użytej sieci neuronowej

```
clear;
close all;
clc;
format compact;

DValue = importdata('glass.data', ',');
Value2=DValue(:,2:10);
y2 = DValue(:,11);


Pn = Value2;
T = y2;

Pn = transpose(Pn);
T = transpose(T);

Pn = mapminmax(Pn);
T=mapminmax(T);

trainFcn = 'traingda';

S1_vec = 1:1:20;
S2_vec = S1_vec;

PK_v=zeros (length(S1_vec),length(S2_vec));
        SSE_v=PK_v;


for S1 = S1_vec
    for S2 = S2_vec:S1
        for ind_lr_inc=1:length(lr_inc_vec)
            for ind_lr_dec=1:length(lr_dec_vec)
                for ind_e=1:length(e_vec)
                    % Create a Pattern Recognition Network
                        hiddenLayerSize = [S1,S2];
                        net = feedforwardnet(hiddenLayerSize, trainFcn);
                        net.performFcn = 'sse';
                        net.layers{1}.transferFcn='tansig';
                        net.layers{2}.transferFcn='tansig';
                        net.layers{3}.transferFcn='purelin';

                        net.trainParam.epochs = 10000;
                        net.trainParam.max_fail = 50;
                        net.trainParam.lr = 0.01;
```

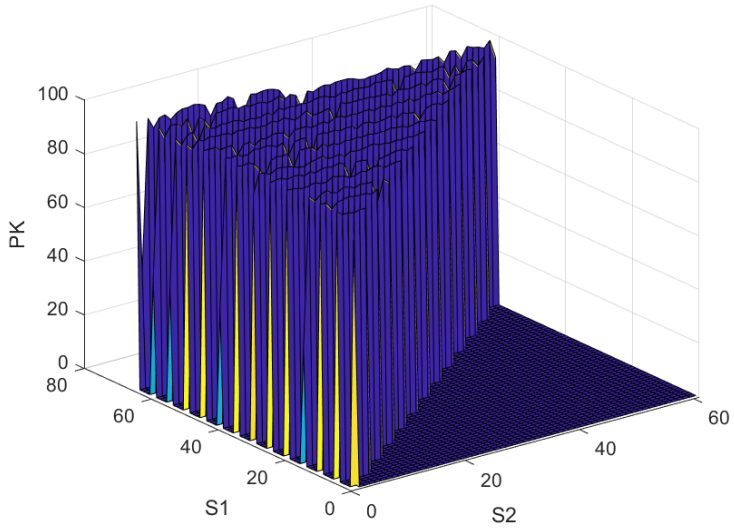
```
45                              net.divideFcn = 'dividetrain';
46                              net.trainParam.goal = 0.25;
47
48                              % Train the Network
49                              [net,tr] = train(net,Pn,T);
50                              % Test the Network
51                              y = net(Pn);
52                              e = gsubtract(T,y);
53                              performance = perform(net,T,y);
54
55
56                              PK = (1-sum(abs(T-y)>=1 )/length(T))*100
57
58                              PK_v(S1, S2) = PK;
59                          SSE_v(S1, S2) = tr.best_perf;
60                      end
61                  end
62              end
63          end
64  end
65  % View the Network
66  view(net)
67
68  save("output8.mat")
```

4 Eksperymenty

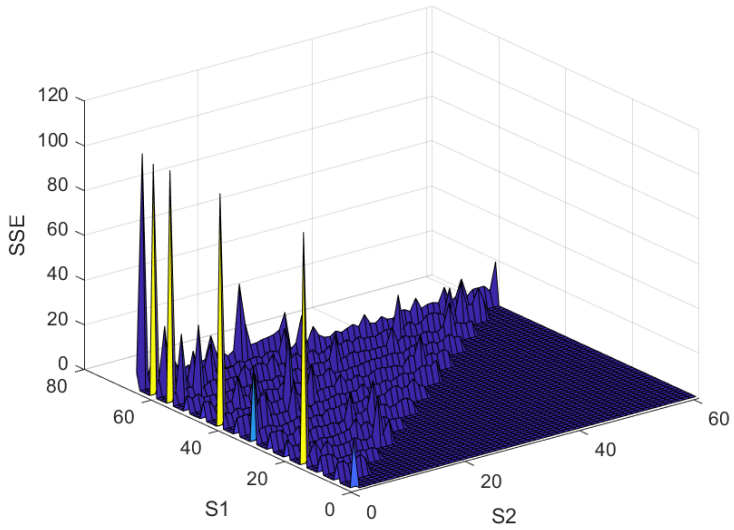
4.1 Eksperyment 1.

Eksperyment miał za zadanie sprawdzić z grubsza ilość neuronów potrzebnych do rozwiązania zadania. Zastosowane parametry:

- Neurony w warstwach: od 1 do 67 z krokiem co 5
- Pozostałe wartości domyślne dla funkcji *traingda*



Rysunek 6: Wykres zależności PK od ilości neuronów w warstwacch dla Eksperymentu 1.



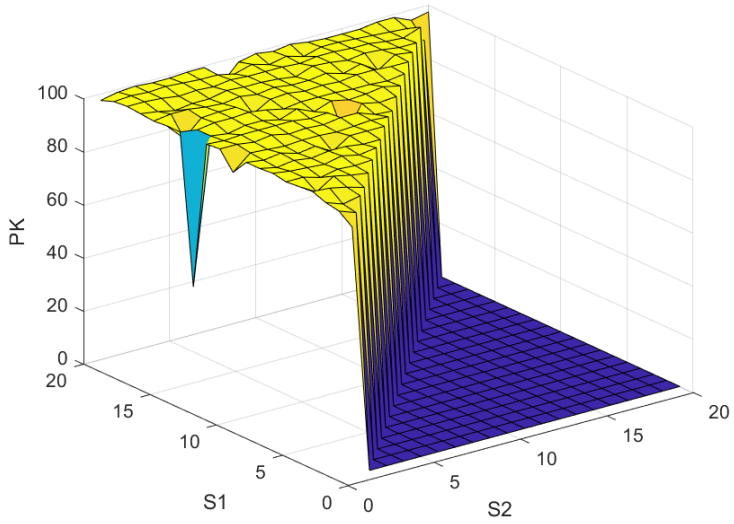
Rysunek 7: Wykres zależności SSE od ilości neuronów w warstwach dla Eksperymentu 1.

Sieć już dla małych ilości neuronów potrafi osiągnąć zadowalające wartości PK oscylujące w granicach 90%. Obserwując wykres błędu możemy zaobserwować pojedyncze ekstremalnie wysokie wartości spowodowane czynnikami losowymi. Pozostałe wartości SSE mieszczą się w granicach zbioru <2;10>.

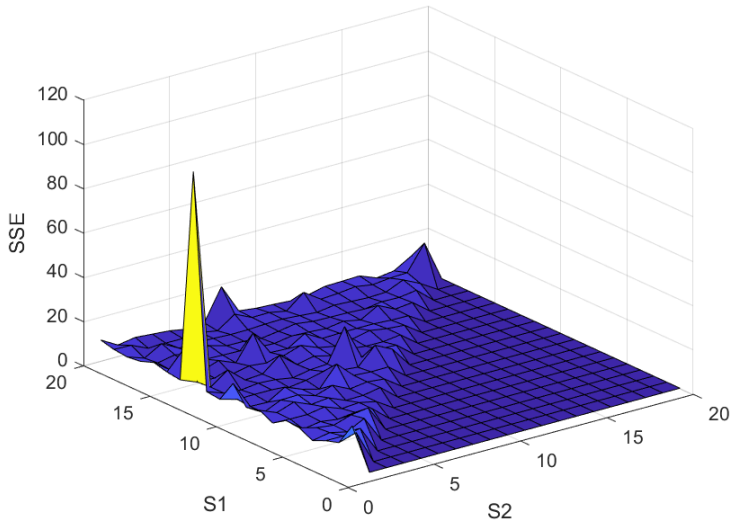
4.2 Eksperyment 2.

Eksperyment ten miał za zadanie dokładnie wyznaczyć najlepszą ilość neuronów w warstwach. Dobierając parametry wyciągnięto wnioski z poprzedniego eksperymentu.

- Neurony w warstwach: od 1 do 20 z krokiem co 1
- Pozostałe wartości domyślne dla funkcji *traingda*



Rysunek 8: Wykres zależności PK od ilości neuronów w warstwach dla Eksperymentu 2.



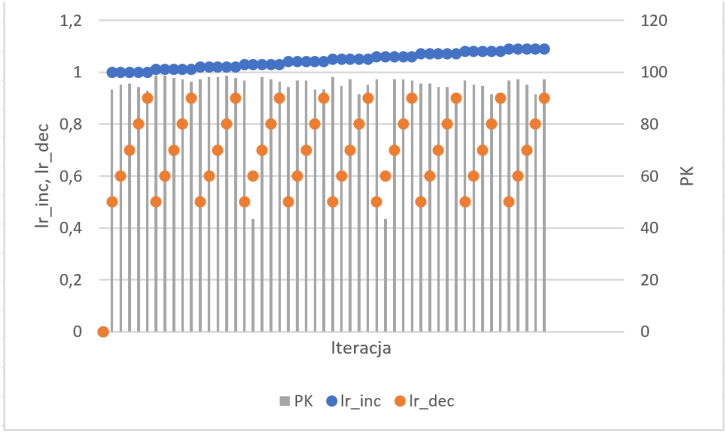
Rysunek 9: Wykres zależności SSE od ilości neuronów w warstwach dla Eksperymentu 2.

Analizując wykresy można dojść do identycznych wniosków z wnioskami Eksperymentu 1. Dla każdej z testowanych konfiguracji wartość PK była zbliżona do 100%. Błąd zawierał się w przedziale o identycznym poziomie wielkości do poprzedniego eksperymentu.

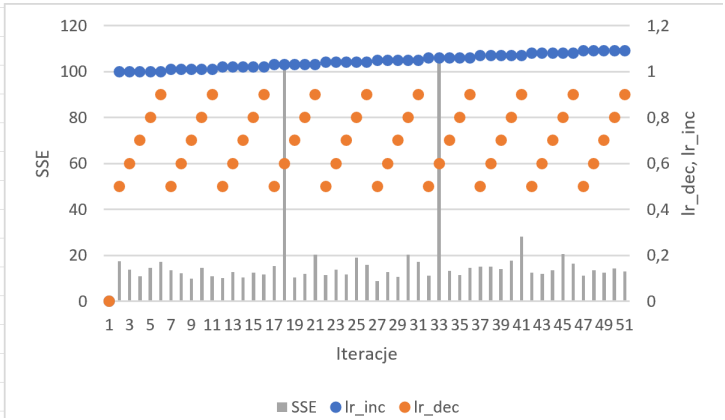
4.3 Eksperyment 3.

Eksperyment miał za zadanie sprawdzić wpływ wartości zwiększania, a także zmniejszania współczynnika uczenia. By tego dokonać wybrano wartość neuronów w warstwach dla którego osiągnięto powyżej 99% wartości PK.

- S1: 7 neuronów
- S2: 2 neurony
- lr_inc: wartości od 1 do 1.09 z krokiem 0.01
- lr_dec: wartości od 0.5 do 0.9 z krokiem 0.1



Rysunek 10: Wykres zależności PK od ilości wartości zmniejszania i zwiększania współczynnika uczenia dla Eksperymentu 3.



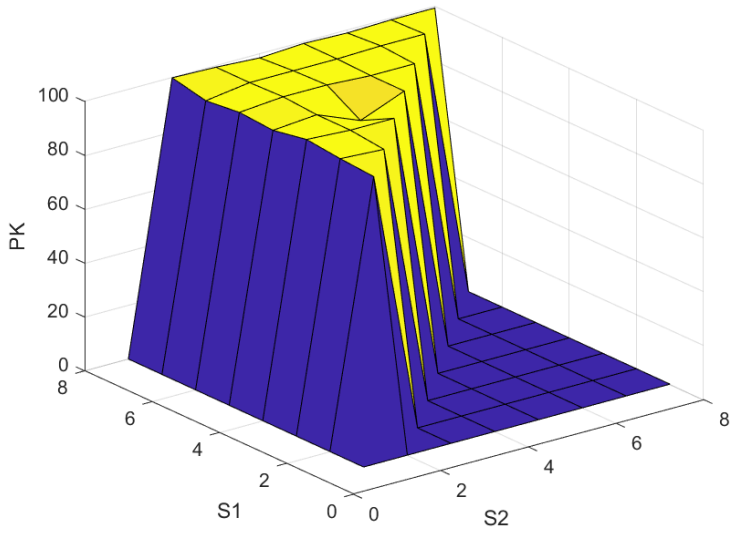
Rysunek 11: Wykres zależności SSE od ilości wartości zmniejszania i zwiększania współczynnika uczenia dla Eksperymentu 3.

Interpretując wykresy możemy możemy zauważyć, że współczynniki inkrementacji i dekrementacji współczynnika uczenia nie mają wielkiego wpływu na poprawność klasyfikacji i błąd SSE. Najlepsza osiągnięta wartość PK wynosi 98,59813084, dla parametrów lr= 1.02, lr= 0.8. Wartość SSE dla tych wartości to 12,35845925.
Najniższy osiągnięty błąd to z kolei 8,915806587 dla pary 1.05, 0.5. Wartość PK dla tych wartości współczynników zmiany wartości współczynnika uczenia jest bliska optymalnemu, wynosi bowiem 98,13084112.

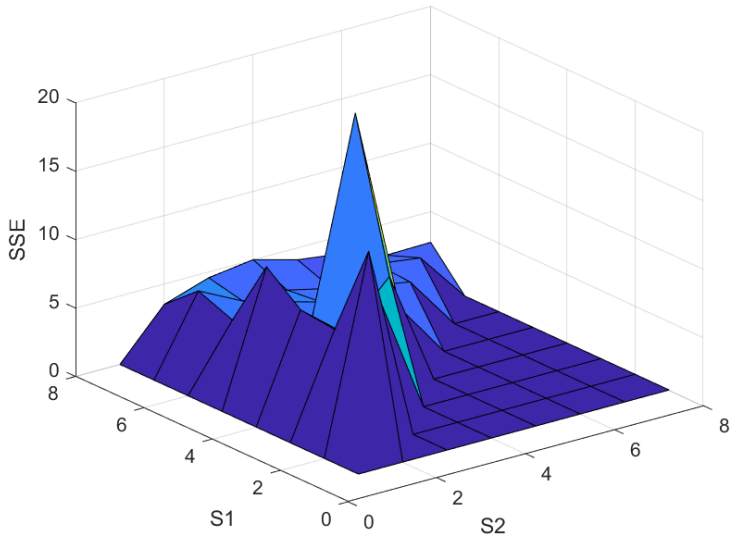
4.4 Eksperyment 4.

Eksperyment miał na celu sprawdzenie, czy zwiększenie maksymalnej liczby epok pozytywnie wpłynie na wartość sumy błędu kwadratowego.

- Maksymalna liczba epok: 50000
- Neurony w warstwach: od 2 do 8 z krokiem co 1
- Pozostałe wartości domyślne dla funkcji *traingda*



Rysunek 12: Wykres zależności PK od ilości neuronów w warstwach dla Eksperymentu 4.



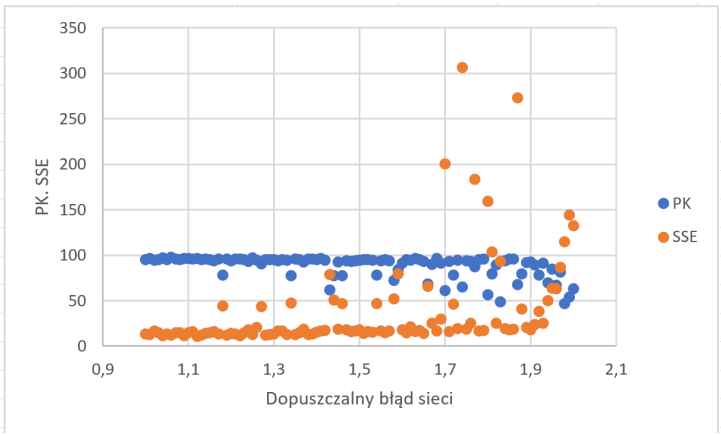
Rysunek 13: Wykres zależności SSE od ilości neuronów w warstwach dla Eksperymentu 4.

Po pięciokrotnym zwiększeniu liczby epok udało się uniknąć ekstremalnych skoków wartości SSE obserwowalnych w poprzednich eksperymentach, spowodowanych czynnikami losowami, jednak statystycznie rząd wielkości wartości SSE pozostał na tym samym poziomie.
Najlepszymi epokami, były epoki zbliżone do 50000 (wartości z przedziału 49944 – 49999).

4.5 Eksperyment 5.

Eksperyment ten miał na celu sprawdzenie wpływu dopuszczalnego błędu sieci na proces uczenia.

- S1: 7 neuronów
- S2: 2 neurony
- Dopuszczalny błąd sieci: wartości od 1 do 2 z krokiem 0.01
- ir_inc: 1.05
- lr_dec: 0.7



Rysunek 14: Wykres zależności PK i SSE od dopuszczalnego błędu sieci

Analizując wykres możemy zaobserwować, że od wartości 1 do 1,7 nie występują poważne odchylenia SSE. Po przekroczeniu wartości 1.7 są one coraz częstsze, a po przekroczeniu 1.9 SSE rośnie liniowo.

5 Wnioski

Celem projektu było rozpoznawanie rodzaju naczyń szklanych skryptem sieci neuronowej uczonej algorytmem wstecznej propagacji błędu z adaptacyjnym współczynnikiem uczenia. Udało się spełnić założenia projektu, a także zapoznać się z tym tematem
Część praktyczna projektu została wykonana w programie Matlab, który posiada zaimplementowane wszystkie niezbędne do wykonania ćwiczenia funkcje. Przede wszystkim warto wymienić funkcję traingda, która zastąpiła znaną z poprzednich wersji funkcję trainbpa. Odpowiednie jej użycie pozwoliło na przeprowadzenie wszystkich eksperymentów. Użyto również funkcji do rysowania wykresów, a także importowania tabel do programu Excel.
Dzięki przeprowadzonym eksperymentom udało się zaobserwować jak wartości poszczególnych parametrów uczenia wpływają na jego rezultat.
Wykonanie projektu dowiodło również, że praca z sieciami neuronowymi wymaga dużej ilości czasu, mocy obliczeniowych komputera, a także wyczucia i doświadczenia w wybieraniu parametrów uczenia.

Bibliografia

[1] `http://archive.ics.uci.edu/ml/datasets/Glass+Identification`

[2] dr hab. inż. Roman Zajdel. *Sztuczna inteligencja, Laboratorium, Ćw6 Model neuronu.*

[3] dr hab. inż. Roman Zajdel. *Sztuczna inteligencja, Laboratorium, Ćw8 Sieć jednokierunkowa jednowarstwowa.*

[4] dr hab. inż. Roman Zajdel. *Sztuczna inteligencja, Laboratorium, Ćw9 Sieć jednokierunkowa wielowarstwowa.*

[5] dr hab. inż. Roman Zajdel. *Sztuczna inteligencja, Laboratorium, Ćw10 Przyśpieszanie procesu uczenia.*

[6] prof. dr hab. inż. Jacek Kluska. *Wykłady ze sztucznej inteligencji.*

[7] L. Rutkowski *Metody i techniki sztucznej inteligencji, Wydawnictwo Naukowe PWN, Warszawa 2019*

[8] `https://www.mathworks.com/help/deeplearning/ref/traingda.html`

[9] `http://pl.wikipedia.org/wiki/Sieć_neuronowa`