Politechnika Rzeszowska

Wydział Elektrotechniki i Informatyki

Katedra Informatyki i Automatyki

Rzeszów, 2019r.

Sztuczna inteligencja

TEMAT:

Zrealizować sieć neuronową uczoną algorytmem wstecznej propagacji błędu z przyśpieszeniem metodą momentum (learnbpm) uczącą się diagnozowania cukrzycy.

Wykonał:

Warchoł Norbert

2EF-DI, P6

Spis treści

1 PRZEDSTAWIENIE PROBLEMU

Ten zbiór danych pochodzi pierwotnie z Narodowego Instytutu Cukrzycy i Chorób Trawiennych i Nerek. Celem zbioru danych jest diagnostyczne przewidywanie czy pacjent cierpi na cukrzycę, na podstawie pewnych pomiarów diagnostycznych zawartych w zbiorze danych. Wybrano te ograniczenia z większej bazy danych. W szczególności, wszyscy pacjenci to kobiety w wieku co najmniej 21 lat z indiańskiego dziedzictwa Pima. Zestawy danych składają się z kilku medycznych zmiennych predykcyjnych i jednej zmiennej docelowej, wyniku. Zmienne predykcyjne obejmują liczbę ciąż, które miał pacjent, ich BMI, poziom insuliny, wiek i tak dalej. Zostaną one wyszczególnione poniżej.

Warto zaznaczyć, że oryginalne źródło bazy danych podane przez prowadzącego zostało usunięte z powodu naruszenia praw autorskich. Poniżej znajduje się oryginalny adres strony: http://archive.ics.uci.edu/ml/datasets/Pima+Indians+Diabetes

Zestaw danych został pobrany na zajęciach z innego źródła i odpowiednio przygotowany(znormalizowany) by bez problemów móc przystąpić do uczenia sieci neuronowej, a materiały potrzebne do opisania danych zostały zaczerpnięte ze strony o adresie: https://www.kaggle.com/uciml/pima-indians-diabetes-database

Zmienna wyjściowa:

1. Wynik (*0=brak choroby, 1=choroba) - 268 zmiennych z 768 ma wartość 1, a reszta to 0.*

Zmienne wejściowe:

1. Ciąża (*Liczba ciąż*)
2. Glukoza - stężenie glukozy w osoczu po 2 godzinach w doustnym teście tolerancji (%)
3. Ciśnienie krwi - ciśnienie rozkurczowe krwi (mm Hg)
4. Grubość skóry - grubość fałdy skórnej trójgłowej (mm)
5. Insulina – 2-godzinna surowica insuliny (mu U/ml)
6. BMI - wskaźnik masy ciała wyrażony w wadze przypadającej na wysokość (kg/m^2)
7. Rodowód cukrzycy – funkcja rodowa cukrzycy
8. Wiek (lata)

2 OPIS SIECI NEURONOWEJ

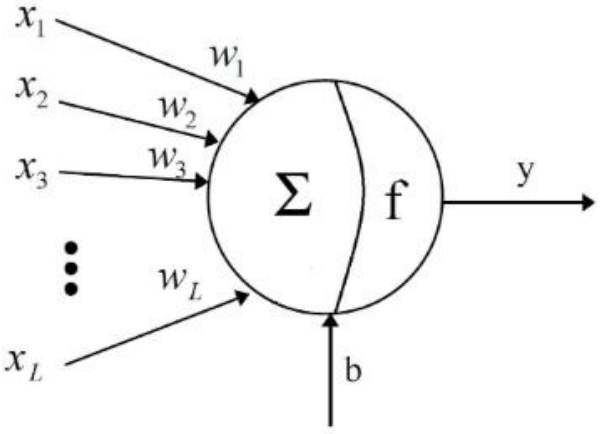
2.1 Wprowadzenie

Sieć neuronowa to model matematyczny, składający się z sieci węzłów obliczeniowych zwanych neuronami i ich połączeń. Jest to pewna technika obliczeniowo-statystyczna, należąca do dziedziny sztucznej inteligencji. Jej działanie symuluje działanie ludzkiego mózgu. Sposób połączenia sztucznych neuronów przypomina wielopołączeniową sieć neuronów w mózgu. Każdy neuron w mózgu jest połączony z około 10 tys. innych, a podczas pracy mózgu pracują wszystkie neurony jednocześnie. W sztucznej sieci neuronowej neurony łączą się tylko z sąsiadami – działają one kolejno a nie jednocześnie.

Sieć neuronowa (o ile jest dobrze zbudowana) jest w stanie nauczyć się aproksymować dowolną funkcję wielu zmiennych. Ponieważ jest to aproksymacja, a nie interpolacja, to sieć jest w stanie uogólniać nabytą wiedzę na nieznane jej, choć podobne problemy. Tę zdolność nazywa się generalizacją. Sieć neuronowa dobrze generalizuje, kiedy odpowiedzi udzielane przez nią dla zestawu danych testowych są prawidłowe lub mieszczą się w granicach ustalonego błędu. Dane testowe muszą pochodzić z tej samej populacji co dane wykorzystane do uczenia sieci.

2.2 Sztuczny neuron

Sztuczny neuron jest systemem przetwarzającym wartości sygnałów wprowadzanych na jego wejścia w pojedynczą wartość wyjściową.



Rysunek 1: Model neuronu.

$$y = f\left(\sum_{j=1}^{L} w_j x_j + b\right)$$

Rysunek 2: Zależność określająca sygnał wyjściowy y neuronu.

W podanym wzorze x_j jest *j*-tym ($j = 1,2,..,L$) sygnałem wejściowym, a w_j - współczynnikiem wagowym (wagą). Ze względu na skrócenie zapisu wygodnie będzie stosować zapis macierzowy do opisu działania neuronu. Niech $x = [x_1,x_2,...,x_L]^T$ będzie wektorem sygnałów wejściowych, $w = [w_1,w_2,...,w_L]$ macierzą wierszową wag, a y i b skalarami. Wówczas:

$$y= f(wx+b)$$

Rysunek 3: Skrócony zapis powyższej zależności.

Ważona suma wejść wraz z przesunięciem często bywa nazywana łącznym pobudzeniem neuronu i oznaczana jako ‘n’.

2.3 Uczenie pod nadzorem pojedynczej warstwy neuronów

Celem procesu uczenia jest taki dobór wag, który pozwoli na właściwe odwzorowanie danych wejściowych w wyjściowe. Zmiana wag odbywa się w kolejnych cyklach zwanych epokami (oznaczonymi literą t). Dla pojedynczej j-tej wagi i -tego neuronu opisujemy to zależnością:

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

Rysunek 4: Zapis zmiany wag w kolejnych cyklach.

Uczenie pod nadzorem (z nauczycielem) zakłada, że każdemu wektorowi wejściowemu x towarzyszy pożądany wektor sygnałów wyjściowych $\hat{y}$ podawany na wyjście warstwy neuronów.

Jeżeli sieć nie jest nauczona, to sygnał ten jest inny niż wymagany, więc sieć popełnia błąd, który można zdefiniować poniższym wzorem:

$$e = y- \hat{y}$$

Rysunek 5: Błąd w przypadku nienauczenia się sieci.

Celem uczenia pod nadzorem jest uzyskanie zgodności odpowiedzi y z wartościami wymaganymi $\hat{y}$, co można sprowadzić do minimalizacji właściwie określonej funkcji celu. Najczęściej przyjmuje ona postać błędu średniokwadratowego wyznaczanego dla wszystkich K neuronów

$$E = \frac{1}{2}\sum_{j=1}^{K} e_j^2$$

Rysunek 6: Funkcja celu w postaci błędu średnio kwadratowego.

Ponieważ E = E(w), zatem poszukiwanie minimum może być dokonane metodą gradientową. Najczęściej stosowaną metodą gradientową zmiany wartości wag jest metoda największego spadku.

$$\Delta w = -\eta \, \Delta E(w)$$

Rysunek 7: Wektor przyrostu wag.

Wyznaczając wzór uczenia gradientowego i ograniczając rozważania tylko do i -tego neuronu i j -tej wagi otrzymamy:

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

Rysunek 8: Wzór przyrostu wagi.

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

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

Rysunek 9: Wzór na pochodną gradientu.

Podstawiając odpowiednie zależności:

$$\frac{\partial E}{\partial y_i} = \frac{\partial}{\partial y_i}\left(\frac{1}{2}(y_1 - \hat{y}_1)^2 + \cdots + \frac{1}{2}(y_i - \hat{y}_i)^2 + \cdots + \frac{1}{2}(y_K - \hat{y}_K)^2\right)$$
$$= y_i - \hat{y}_i = e_i$$

$$\frac{\partial n_i}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}}(w_{i1}x_1 + \cdots + w_{ij}x_j + \cdots + w_{iL}x_L + b_i) = x_j$$

Rysunek 9.1 Wzory do wyznaczenia reguły Widrowa-Hoffa.

otrzymujemy regułę uczenia Widrowa-Hoffa w postaci ogólnej

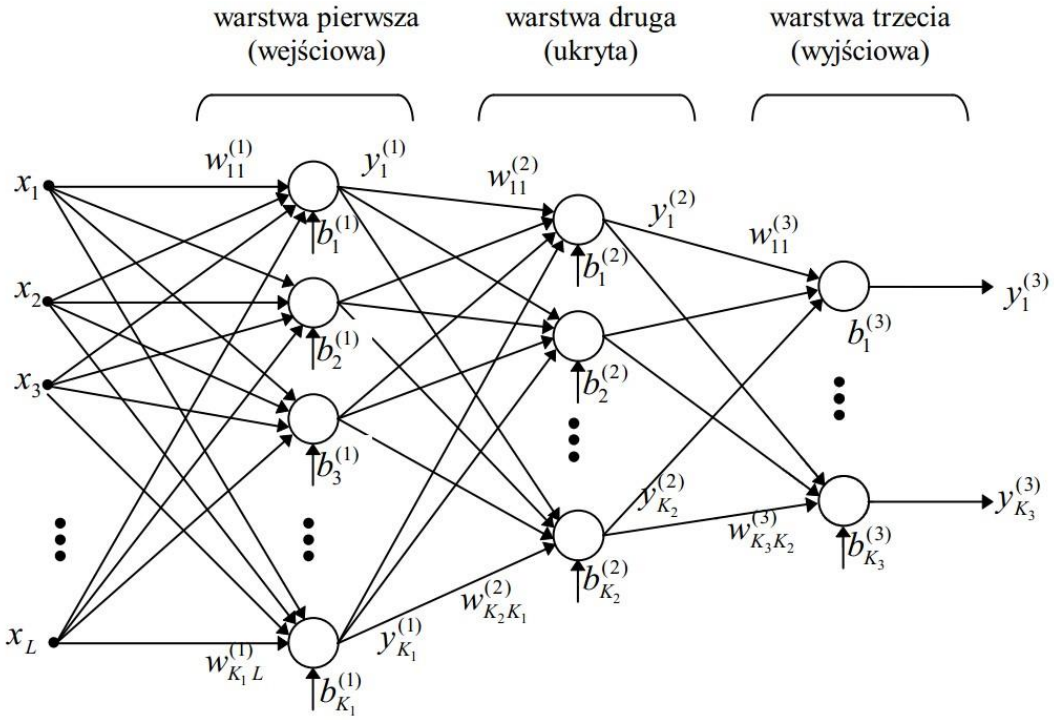
$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} = \eta \cdot (\hat{y} - y_i) \cdot \frac{\partial y_i}{\partial n_i} \cdot x_j$$

Rysunek 10: Postać ogólna reguły uczenia Widrowa-Hoffa.

Dotyczy ona uczenia nadzorowanego sieci jednokierunkowych, gdzie minimalizuje się błąd pomiędzy pożądaną, a aktualną odpowiedzią.

3 ALGORYTM WSTECZNEJ PROPAGACJI BŁĘDU

3.1 Sieć wielowarstwowa



Rysunek 11: Jednokierunkowa sieć wielowarstwowa.

Taką sieć nazywa się wielowarstwową. Występują tu połączenia pomiędzy warstwami neuronów typu każdy z każdym. Sygnały wejściowe podawane są do warstwy wejściowej neuronów, których wyjścia stanowią sygnały źródłowe dla kolejnej warstwy. Każda warstwa neuronów posiada swoją macierz wag w, wektor przesunięć b, funkcje aktywacji f i wektor sygnałów wyjściowych y. Działanie każdej z warstw można rozpatrywać oddzielnie. I tak np. warstwa druga posiada: $L = K_1$ sygnałów wejściowych, $K = K_2$ neuronów i macierz wag $w = w^{(2)}$ o rozmiarach $K_2 \times K_1$. Wejściem warstwy drugiej jest wyjście warstwy pierwszej $x = y^{(1)}$, a wyjściem $y = y^{(2)}$.

Działanie poszczególnych warstw dane jest przez wzory:

$$y^{(1)} = f^{(1)}(w^{(1)} x + b^{(1)})$$
$$y^{(2)} = f^{(2)}(w^{(2)} y^{(1)} + b^{(2)})$$
$$y^{(3)} = f^{(3)}(w^{(3)} y^{(2)} + b^{(3)})$$

Rysunek 12: Wyjścia dla poszczególnych warstw sieci.

Działanie całej sieci można więc opisać jako:

$$y^{(3)} = f^{(3)}(w^{(3)} f^{(2)}(w^{(2)} f^{(1)}(w^{(1)}x + b^{(1)}) + b^{(2)}) + b^{(3)})$$

Rysunek 13: Wyjścia dla poszczególnych warstw sieci.

3.2 Opis algorytmu

Klasycznym algorytmem uczenia jest algorytm wstecznej propagacji błędu (ang. *Back Propagation*). Jego nazwa pochodzi od sposobu obliczania błędów w poszczególnych warstwach sieci. W pierwszej kolejności wyliczany jest błąd ostatniej warstwy, w oparciu o sygnały wyjściowy i wzorcowy. W pozostałych warstwach błąd obliczany jest jako ustalona funkcja błędu neuronów warstwy poprzedzającej. Algorytm ten należy do gradientowych metod optymalizacji, gdyż opiera się o stwierdzenie, że gradient funkcji wskazuje kierunek jej najszybszego wzrostu a zmieniając jego znak na przeciwny, uzyskujemy kierunek najszybszego spadku funkcji.

Możemy w ten sposób minimalizować funkcję celu przez modyfikację jej zmiennych, w przypadku sieci - współczynników wagowych, w kierunku najszybszego spadku funkcji. Zgodnie z algorytmem wstecznej propagacji błędu w każdym cyklu uczącym wyróżnia się następujące etapy uczenia:

- Analiza sieci neuronowej o zwykłym kierunku przepływu sygnałów. W wyniku takiej analizy otrzymuje się wartości sygnałów wyjściowych neuronów warstw ukrytych oraz warstwy wyjściowej, a także odpowiednie pochodne funkcji aktywacji w poszczególnych warstwach.
- Utworzenie sieci propagacji wstecznej przez odwrócenie kierunków przepływu sygnałów, zastąpienie funkcji aktywacji przez ich pochodne, a także podanie do byłego wyjścia (obecnie wejścia) sieci, wymuszenia w postaci odpowiedniej różnicy między wartością aktualną i żądaną. Dla tak utworzonej sieci należy obliczyć wartości odpowiednich różnic wstecznych.
- Adaptacja wag (uczenie sieci) odbywa się na podstawie wyników uzyskanych w punkcie pierwszym i drugim dla sieci zwykłej i sieci o propagacji wstecznej według odpowiednich wzorów.

Cały proces należy powtórzyć dla wszystkich wzorców uczących, kontynuując go do chwili spełnienia warunku zatrzymania algorytmu. Działanie algorytmu kończy się w momencie, gdy norma gradientu spadnie poniżej pewnej wartości określającej dokładność procesu uczenia.

Z praktycznego punktu widzenia istotne znaczenie ma szybkość uczenia. Metoda wstecznej propagacji błędu jest bardzo czasochłonna. Spotyka się szereg metod pozwalających na znaczne przyśpieszenie procesu uczenia. Można wyróżnić dwa podejścia. Pierwsze polega na wprowadzeniu do wzoru na korektę wag dodatkowego składnika będącego miarą „bezwładności" zmiany wag, zwanego „momentum". Drugie podejście polega na odpowiednim doborze współczynnika uczenia η np. metodą adaptacyjnego doboru tego współczynnika.

Metoda momentum jest stosowana wtedy, gdy występuje mały współczynnik uczenia i mały gradient. W takiej sytuacji zmniejszanie błędu jest powolne. W przypadku, kiedy współczynniki te są duże, występują silne oscylacje wokół minimum.

$$w_{ij}(t+1)=w_{ij}(t)+(1-\eta)(\hat{y}_i(t)-y_i(t))\frac{\partial y_j(t)}{\partial z_j}x_i(t)+\eta M_{ij}(t)$$

Rysunek 14: Reguła uczenia z momentum w przypadku uczenia pod nadzorem pojedynczej warstwy sieci.

Momentum $M_{ij}(t)$ jest wyliczane z zależności:

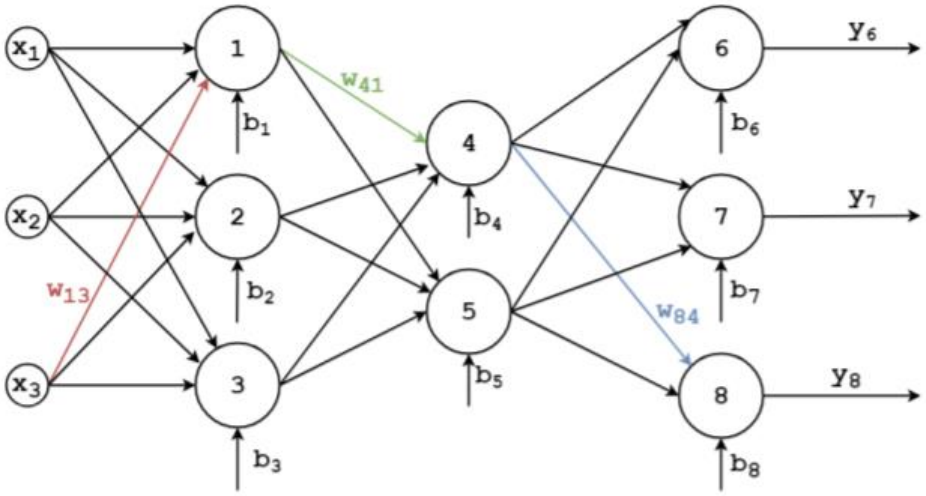
$$M_{ij}(t) = w_{ij}(t) - w_{ij}(t-1)$$

Rysunek 15: Wzór na wyliczanie momentum

η ∈[0, 1] jest współczynnikiem uczenia momentum. Jeżeli η = 1, to nowa waga jest równa ostatniej zmianie wagi. Jeżeli η = 0, to zmiana wagi oparta jest tylko na składowej gradientowej. Najczęściej przyjmuje się η = 0.95. Wprowadzenie składnika momentum zdecydowanie wpływa na zwiększenie szybkości uczenia.

Zasada działania jest następująca: jeżeli w kolejnych krokach gradienty wykazują ten sam kierunek (pochodna cząstkowa dla danej wagi ma ten sam znak) ich działanie się kumuluje i przyrosty są coraz większe. Jeżeli natomiast znaki są przeciwne (gradienty wykazują przeciwny kierunek) działanie jest osłabiane co uniemożliwia powstawanie silnych oscylacji.

Sposób działania algorytmu wstecznej propagacji błędu najłatwiej przedstawić jest na przykładzie. Korzystając z schematu przykładowej sieci jednokierunkowej wielowarstwowej przedstawionego na rysunku 11.1. zaznaczono po jednej przykładowej wadze z każdej warstwy:



Rysunek 11.1: Rozszerzenie schematu przykładowej sieci jednokierunkowej wielowarstwowej z rys. 11.

Obliczając nową wagę warstwy ostatniej, w tym przypadku w_{84}, korzystamy z wzoru na rysunku 9 i 10:

$$\Delta w_{84} = -\eta \frac{\partial E}{\partial w_{84}} = -\eta \frac{\partial E}{\partial y_8} \cdot \frac{\partial y_8}{\partial n_8} \cdot \frac{\partial n_8}{\partial w_{84}}$$

Rysunek 13.1: Przyrost wagi

Pochodna cząstkowa z błędu E po funkcji aktywacji y_8 jest równa $y_8 - \hat{y}_8$, czyli błędowi cząstkowemu e_8 na wyjściu neuronu ósmego (rysunek 9.1). Pochodna cząstkowa z funkcji aktywacji y_8 po łącznym pobudzeniu neuronu n_8 jest zwykłą pochodną funkcji, zależnej od owej funkcji aktywacji. Można ją zapisać jako $\partial y_8/\partial n_8 = f'_8(n_8)$.

Pochodna cząstkowa z łącznego pobudzenia neuronu n_8 po wadze w_{84} jest równy sygnałowi wejściowemu do neuronu 8 pochodzącego od neuronu 4 (rysunek 9.1), a więc jest to sygnał y_4.

Ostatecznie nowa waga w_{84}, oznaczona indeksem górnym z gwiazdką wynosi:

$$w_{84}^* = w_{84} + \eta \cdot e_8 \cdot f_8'(n_8) \cdot y_4 = w_{84} + \delta_8 \cdot y_4$$

Rysunek 13.2: Nowa waga

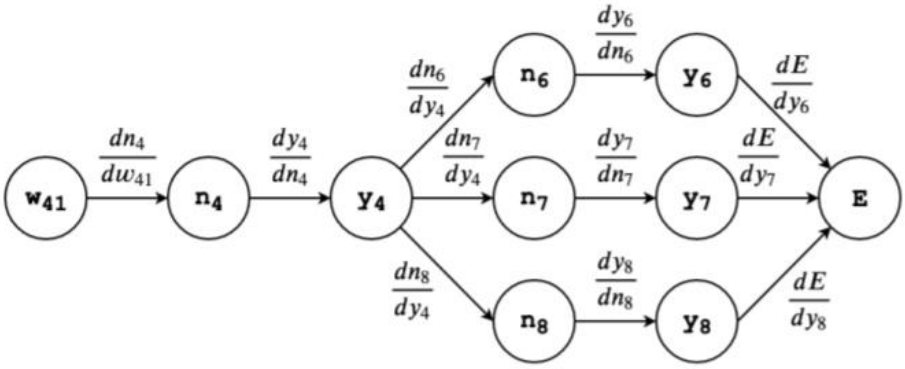
Bardzo często człon $\eta \cdot e_8 \cdot f'_8(n_8)$ jest zapisywany jako δ_8. Bias można obliczyć w bardzo podobny sposób. Dwie pierwsze pochodne cząstkowe liczone są w ten sam sposób, a jedynie ostatnia pochodna przyjmuje postać:

$$\frac{\partial n_i}{\partial b_i} = \frac{\partial}{\partial b_i}(w_{i1}x_1 + \cdots + w_{iL}x_L + b_i) = 1$$

Stąd też wzór na nowy bias przyjmuje postać:

$$b_8^* = b_8 + \eta \cdot e_8 \cdot f_8'(n_8) \cdot 1 = b_8 + \delta_8$$

Dla nowej wagi warstwy drugiej, dla przykładu wagi $w41$ najwygodniej jest skorzystać z pomocniczego schematu zależności (rys. 3.6.)



Rysunek 11.2: Pomocniczy rysunek do wyznaczania przyrostu wag w 2 warstwie.

Na schemacie pokazane są kolejne zależności zmiennych. Błąd E zależy bezpośrednio od sygnałów wyjściowych w ostatniej warstwie (y6−8), które zależą bezpośrednio od łącznego pobudzenia neuronów w odpowiednich neuronach (n6−8), które zależą bezpośrednio od sygnału wyjściowego y_4, który zależy bezpośrednio od łącznego pobudzenia neuronu czwartego (n_4), które zależy bezpośrednio od wagi w_{41}. Pochodne nad strzałkami oznaczają dokładnie te zależności.

Z wcześniejszych rozważań wiadomo, że:

$$\frac{\partial E}{\partial y_i} = e_i\,,\ \frac{\partial y_i}{\partial n_i} = f_i'(n_i)\ \text{oraz}\ \frac{\partial n_i}{\partial w_{ij}} = y_j$$

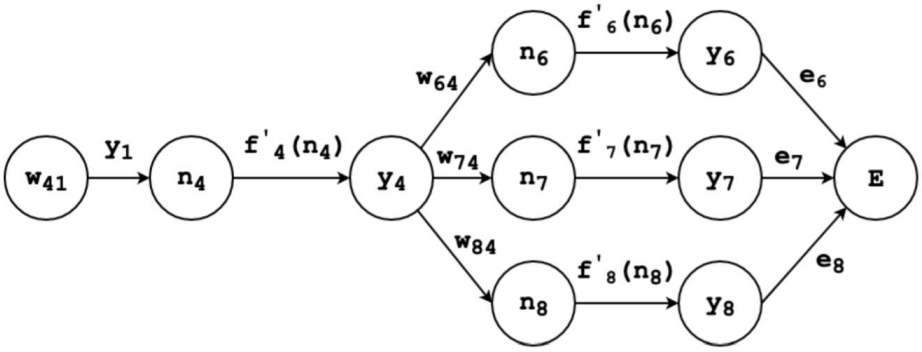
Pozostało obliczyć pochodną cząstkową typu:

$$\frac{\partial n_i}{\partial y_j}$$

Wynosi ona:

$$\frac{\partial n_i}{\partial y_j} = \frac{\partial}{\partial y_j}(w_{i1}y_1 + \cdots + w_{ij}y_j + \cdots + w_{iN}x_N + b_i) = w_{ij}$$

Stosując powyższe zależności, można nanieść je na schemat, który będzie wyglądał jak na kolejnym rysunku rysunku.



Rysunek 11.2: Pomocniczy rysunek do wyznaczania przyrostu wag w 2 warstwie.

Aby otrzymać poszukiwaną pochodną, należy pomnożyć wszystkie pochodne stojące na drodze pomiędzy tymi dwoma punktami. W przypadku rozgałęzienia, poszczególne pochodne w gałęziach należy zsumować. Tak więc zaczynając od końca, otrzymujemy:

$$\frac{\partial E}{\partial w_{41}} = [e_6 \cdot f_6'(n_6) \cdot w_{64} + e_7 \cdot f_7'(n_7) \cdot w_{74} + e_8 \cdot f_8'(n_8) \cdot w_{84}] \cdot f_4'(n_4) \cdot y_1$$

Dla skrócenia zapisu, przyjęto oznaczenie $f'_x(n_x) = f'_x$. Ostatecznie nowa waga w_{41}, oznaczona indeksem górnym z gwiazdką wynosi:

$$w_{41}^* = w_{41} + \eta \cdot [e_6 \cdot f_6' \cdot w_{64} + e_7 \cdot f_7' \cdot w_{74} + e_8 \cdot f_8' \cdot w_{84}] \cdot f_4' \cdot y_1 = \\ = w_{41} + \delta_4 \cdot y_1$$

Człon $\eta \cdot [e_6 \cdot f'_6 \cdot w_{64} + e_7 \cdot f'_7 \cdot w_{74} + e_8 \cdot f'_8 \cdot w_{84}] \cdot f'_4$ jest często zapisywany jako δ_4. Podobnie jak w przypadku warstwy wyjściowej (wzór na pochodną cząstkową łącznego pobudzenia neuronu po biasie), nowy bias jest definiowany jako:

$$b_4^* = b_4 + \eta \cdot [e_6 \cdot f_6' \cdot w_{64} + e_7 \cdot f_7' \cdot w_{74} + e_8 \cdot f_8' \cdot w_{84}] \cdot f_4' = b_4 + \delta_4$$

Dla warstwy 1, modyfikując równanie $\partial n_i / \partial y_i$, pochodna łącznego pobudzenia neuronu po wadze będzie wynosić

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

Z resztą postępujemy analogicznie jak w poprzednich krokach.

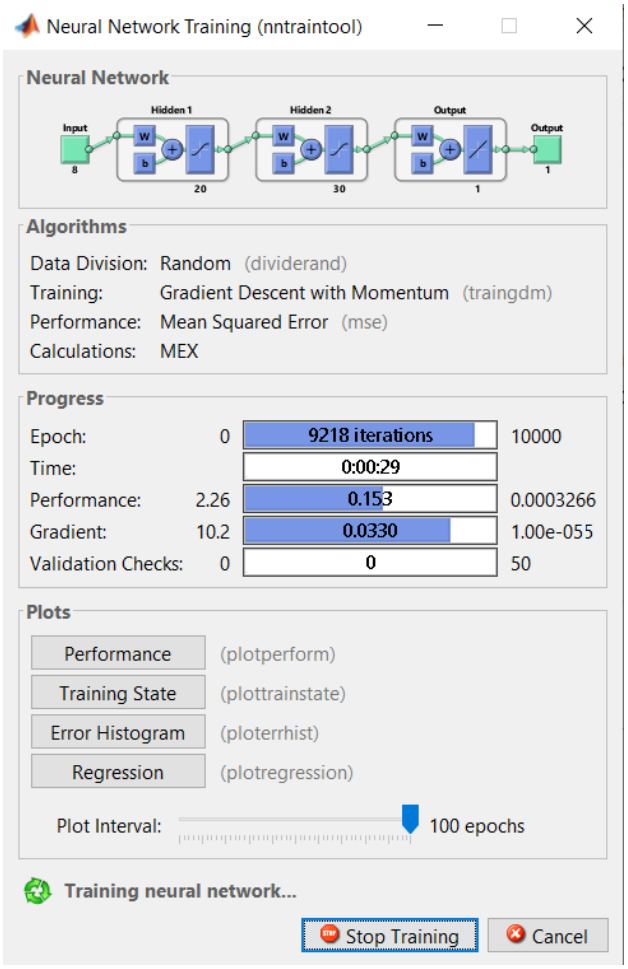
4 SKRYPT PROGRAMU

```
clear; %czysci pamięć i zmienne z workspace'a
format compact; %czyści wyjscia(brak pustych przestrzeni w konsoli)
load pima.mat; %ładowanie danych znormalizowanych
% Przygotowanie zmiennych
P = Pns; % wejście
T = Ts; % pożądane wyjście
% Wyczyszczenie zbednych zmiennych
clearvars -except P T;
% Wektor neuronow warstwy 1 i 2
S1_vec = [10:10:100];
S2_vec = S1_vec;
% Wektory learning ratio, momentum constans i do testów epoki
lr_vec = [0.001 0.01 0.1] ;
mc_vec = [0.1:0.1:0.9 0.99];
%ep_vec = [1000 10000 25000 50000 100000];
%Macierz poprawności klasyfikacji i macierz błędu
PK_v4 = zeros(length(S1_vec),length(S2_vec),length(lr_vec),length(mc_vec));
MSE_v4 = PK_v4;
PK_sr=0;
% licznik ile % do końca nauki, niestety się psuje przy niektórych testach
licznikWC= 0;
work_count = length(S1_vec)*length(S2_vec)*length(lr_vec)*length(mc_vec);
% Zapis nagłówka do pliku
header = 'S1 \t\t S2 \t\t lr \t\t mc \t\t PK[%%] \t\t MSE \t\t epoch \t\t work_progress[%%]\n';
file_var = fopen('epoka.txt', 'wt');    %wt-zezwolenie na zapis
fprintf(file_var, header);
formating = '%2g \t\t %2g \t\t %1.5g \t\t %1.3g \t\t %1.4g \t\t %3.4g \t\t %g \t\t %3.2g\n';
fclose(file_var);

for ind_S1=1:length(S1_vec)
   for ind_S2=1:ind_S1    %:length(S2_vec) ? :ind_S1
      for ind_lr=1:length(lr_vec)
         for ind_mc=1:length(mc_vec)
           %for ind_ep=1:length(ep_vec)
              for ex=1:5
              % Zwiekszenie licznika
              licznikWC=licznikWC+1;
              % Procent wykonania
```

```
            work_progress = licznikWC/work_count *100;
            % Utworzenie sieci
            net = feedforwardnet([S1_vec(ind_S1) S2_vec(ind_S2)],'traingdm');
            % Ustawienie parametrow sieci
            net.trainParam.lr = lr_vec(ind_lr);
            net.trainParam.epochs = 20000;
            %net.trainParam.epochs = ep_vec(ind_ep);
            net.trainParam.goal = 0.25/length(T); %błąd docelowy
            net.trainParam.mc=mc_vec(ind_mc);
            net.trainParam.max_fail=50; %przez ile epok error wzrósł wiec jest przerywane
            % Trenowanie
            [net,tr] = train(net,P,T);
            %Symulacja
            y = net(P);  %wyjscie sieci o rozmiarach jak T
            %Poprawność Klasyfikacji(precyzja) w %
            PK = (1-sum(abs(T-y)>=.5)/length(T))*100; %sum sumuje wartosci w macierzy
            MSE = immse(y,T); %immse liczy blad mse pomiedzy macierzami o takich samych rozmiarach i klasach
            PK_sr=PK_sr+PK;
            end
            PK=PK_sr/ex;
            PK_sr=0;
           % Zapis do macierzy 4D- PK i Błąd średnio kwadratowy
           PK_v4 (ind_S1, ind_S2, ind_lr, ind_mc) = PK; %dolaczenie PK do ... tablicy PK
           MSE_v4(ind_S1, ind_S2, ind_lr, ind_mc) = MSE; %dolaczenie poj. bledu do ... tablicy bledow
          % Zapis do pliku
          file_var = fopen('epoka.txt', 'at'); %at-zezwolenie na odczyt do zapisywania w pliku
            fprintf(file_var, formating, S1_vec(ind_S1), S2_vec(ind_S2), lr_vec(ind_lr), mc_vec(ind_mc), PK, MSE, tr.best_epoch,
            work_progress);
            fclose(file_var);
         %fprintf("'\n")
         %end
        end
      end
    end
end
```

5 OPIS WYKORZYSTANEGO ŚRODOWISKA



Rysunek 16: Przykładowe okno zawierające informacje dotyczące trenowania sieci neuronowej w programie Matlab R2019a

Na samej górze okna przedstawiony jest graficzny schemat naszej sieci neuronowej, z wejściem, wyszczególnionymi warstwami i ilościami neuronów, które w nich się znajdują oraz wyjściem sieci.

Poniżej tego znajduje się zakładka “Algorytmy”, która zawiera informacje o użytych przez Matlaba funkcjach do obsługi sieci. W tej części pozostawiono domyślnie dobrane przez program funkcje do doboru wag i podziału danych do treningu oraz wykonywania obliczeń. Zalecany przez Matlaba błąd MSE również został zaakceptowany i nie został zmieniony w programie. Jedynym elementem, który uległ zmianie to algorytm do nauki sieci jaki jest realizowany w tym projekcie, a nie jest domyślnym algorytmem programu.

Okienko “Postęp” informuje użytkownika m.in. o czasie jaki był potrzebny na przejście danej ilości epok, wydajności sieci, gradiencie oraz sprawdzaniu poprawności nauki sieci.

Ostatnia zakładka zawiera w sobie domyślne wykresy zaproponowane przez Matlaba m.in. do sprawdzania wydajności i stanu nauczenia naszej sieci oraz suwak do zmniejszania częstotliwości odświeżania danych na wykresie.

Ostatnim elementem okienka uczenia sieci są dwa przyciski. Pierwszy z nich przerywa aktualnie wykonywany proces nauki i przechodzi do kolejnego etapu, a drugi anuluje cały proces nauczania i automatycznie zaczyna cały proces od nowa.

6 WYNIKI POMIARÓW I EKSPERYMENTY

6.1 Eksperymenty wstępne

6.1.1 Test pierwszy

W pierwszym eksperymencie dotyczącym bazy danych Indian zbadano jaki wpływ mają poszczególne parametry i tak można było dojść do wniosku, że najlepsze PK uzyskujemy, gdy lr jest równe 0.01 lub 0.001, a mc=0.8.

Oczywiście są to wartości szacunkowe i dopiero dalsze eksperymenty pozwolą nam dokładniej zbadać czy pierwsze spostrzeżenia były słuszne.

Wpływ pozostałych parametrów takich jak S1, S2 zostanie lepiej przebadany w kolejnych eksperymentach, gdyż tutaj nie dało się tego jednoznacznie określić.

Oto przykładowe wyniki z pliku:

S1	S2	lr	mc	MSE	epoch	WP[%]	PK[%]
30	30	0.01	0.6	0.1487	8e+02	0.174	77.08
30	30	0.01	0.7	0.1626	1e+03	0.347	76.95
30	30	0.001	0.6	0.1492	1e+04	0.868	75.26
30	30	0.001	0.7	0.1519	1e+04	1.04	76.82
30	30	0.0001	0.7	0.199	1e+04	1.74	68.88
30	30	0.0001	0.8	0.19	1e+04	1.91	72.4
30	30	0.0001	0.9	0.1863	1e+04	2.08	70.57
50	30	0.01	0.7	0.1425	2e+03	2.43	79.56
50	30	0.001	0.7	0.1793	2e+03	3.13	74.09
50	30	0.001	0.8	0.1482	1e+04	3.3	78.78
50	30	0.001	0.9	0.1479	1e+04	3.47	78.13

6.1.2 Test drugi

W kolejnym eksperymencie postanowiono użyć innego zbioru danych (iris_dataset), ponieważ po przeprowadzeniu kilku testów poziom nauczenia utrzymywał się na niskim poziomie (poniżej 80%) pomimo zmiany wszystkich parametrów sieci.
Przedstawione wyniki PK są procentowymi rezultatami uczenia się jakie uzyskała sieć.
Oto przykładowe wyniki:

S1	S2	lr	mc	MSE	epoch	WP[%]	PK[%]
10	10	0.0001	0.9	0.04608	5e+03	0.333	96.67
10	10	0.001	0.5	0.02664	5e+03	0.4	97.33
10	10	0.001	0.7	0.0369	4e+03	0.533	96.67
10	10	0.001	0.9	0.05548	5e+03	0.667	94
10	10	0.01	0.5	0.0245	5e+03	0.733	97.33
30	10	0.0001	0.5	0.084	5e+03	1.07	94
30	10	0.0001	0.6	0.06229	5e+03	1.13	93.33
30	30	0.001	0.7	0.04242	2e+03	2.53	97.23
30	30	0.001	0.8	0.02426	5e+03	2.6	97.43
50	10	0.001	0.6	0.0408	1e+03	3.47	94.67
50	10	0.001	0.7	0.05094	1e+03	3.53	92.67
50	10	0.001	0.8	0.05658	9e+02	3.6	95.32

Zamieszczonej tabeli widać, sieć uczyła się bardzo dobrze. Może to oznaczać, że zbiór danych używany w projekcie niekorzystnie wpływa na proces uczenia się sieci, przynajmniej w przypadku użycia algorytmu learnBPM.

Dzięki temu można wysnuć wnioski, że stworzony algorytm jest co najmniej poprawny.

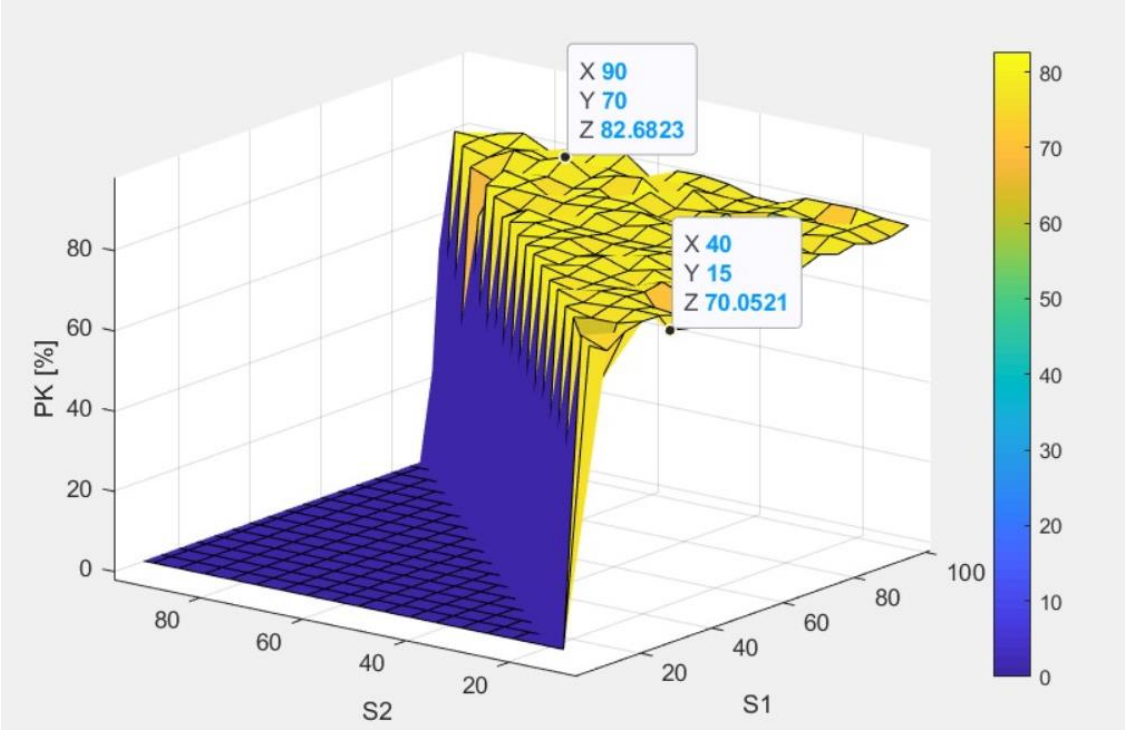
6.2 Eksperymenty właściwe

Nr 1- S1/S2

Pierwszym eksperymentem badano wpływ neuronów w poszczególnych warstwach na nauczenie sieci. Oto część otrzymanych wyników:

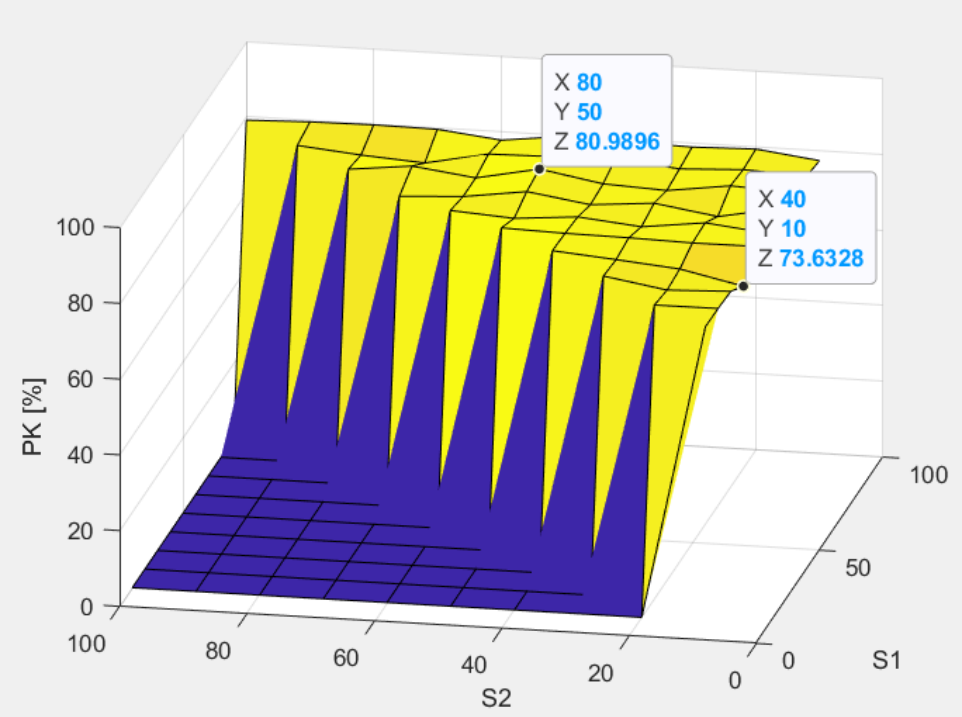
S1	S2	lr	mc	PK [%]	MSE	epoch	work_progress[%]
10	10	0.001	0.8	77.08	0.1608	20000	0.28
15	10	0.001	0.8	74.22	0.1679	20000	0.55
15	15	0.001	0.8	76.69	0.1542	20000	0.83
20	10	0.001	0.8	77.34	0.1562	20000	1.1
20	20	0.001	0.8	78.26	0.1561	20000	1.7
25	10	0.001	0.8	78.39	0.1534	20000	1.9
25	15	0.001	0.8	76.56	0.1629	20000	2.2
25	20	0.001	0.8	76.56	0.1598	20000	2.5
25	25	0.001	0.8	78.13	0.1509	20000	2.8
90	45	0.001	0.8	81.12	0.1404	20000	40
90	70	0.001	0.8	82.68	0.1369	15761	41
90	75	0.001	0.8	80.47	0.1483	9416	42
100	55	0.001	0.8	80.08	0.1426	10932	50
100	65	0.001	0.8	81.51	0.1382	15721	51
100	70	0.001	0.8	81.12	0.1385	20000	51
100	75	0.001	0.8	80.21	0.1445	7959	51
100	80	0.001	0.8	78.26	0.1485	4501	52

Przy większej ilości neuronów, zauważono brak spadku wartości błędu końcowego, a sieć przestawała się uczyć, dlatego 100 neuronów zostało ustawione jako górna granica. Przy większych wartościach należy odpowiednio ustawiać współczynnik *lr,* aby uzyskać satysfakcjonujące i prawidłowe wyniki co zostało lepiej przebadane w dalszych eksperymentach.



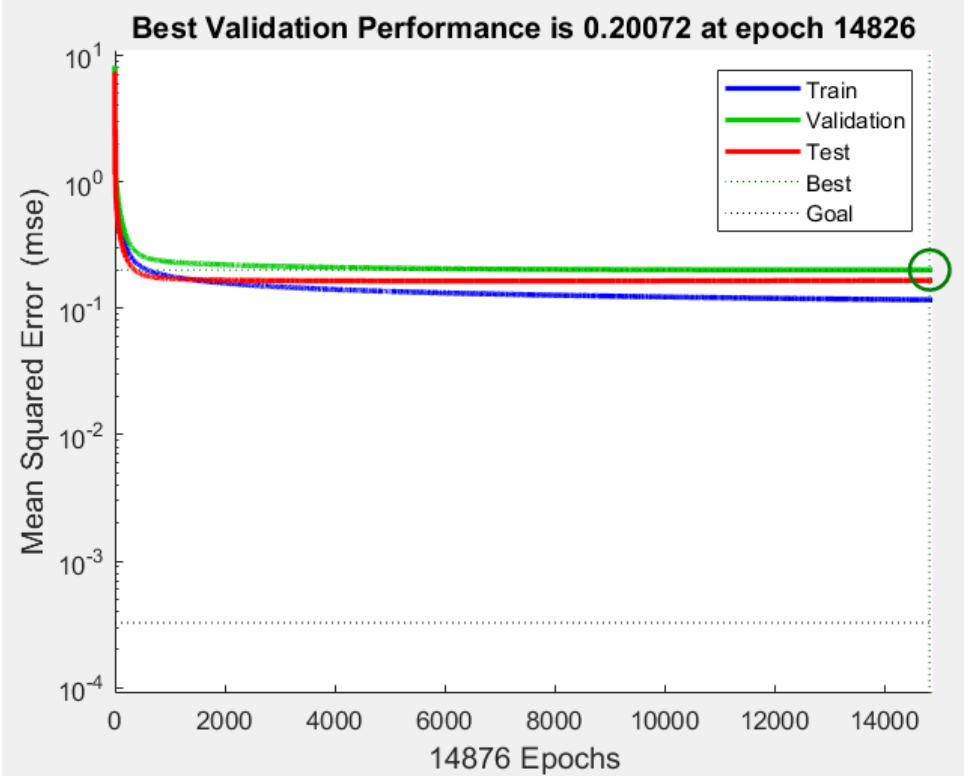
Rysunek 17: Zależność ilości neuronów w warstwie pierwszej od ilości neuronów w warstwie drugiej i ich wpływ na poziom nauczenia się sieci.

Po zauważeniu rozbieżności ponad 12%, w kolejnych eksperymentach postanowiono brać średnią 5 takich samych eksperymentów, co oznaczało wprowadzenie dodatkowej pętli i znacząco wydłużyło proces uczenia, lecz pozwoliło na uzyskanie wyników mniej rozbieżnych i zminimalizowało możliwy problem z losowaniem początkowych wag (domyślny algorytm Matlaba – random).

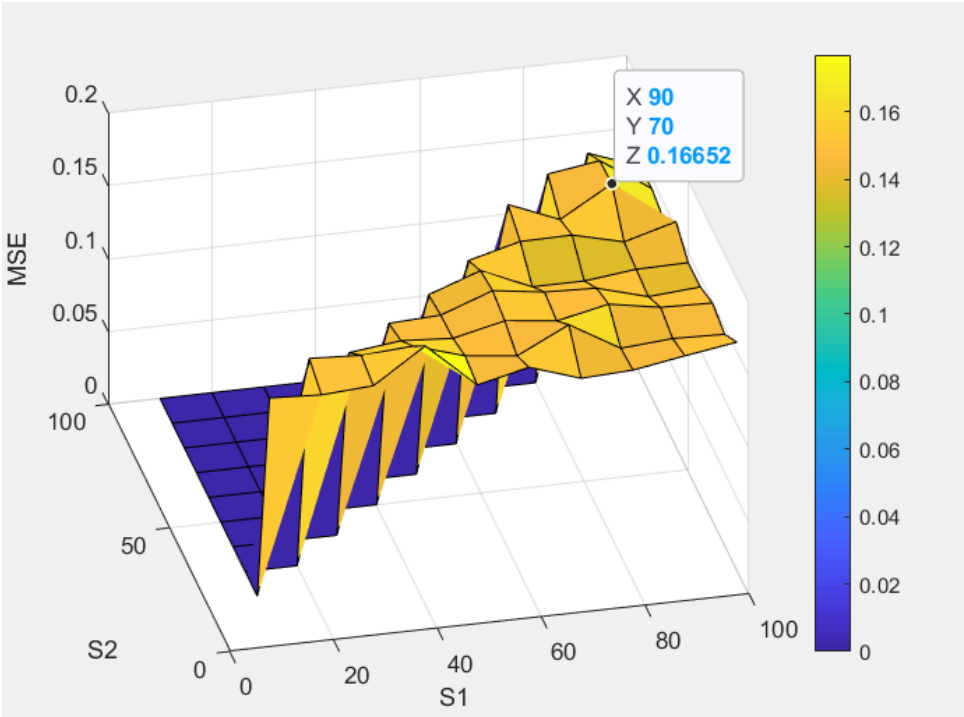


Rysunek 17.1: Zaktualizowany wykres powierzchniowy z rys. 17 przy uwzględnieniu średniej PK

Jak widać z wykresu sieć uczyła się dobrze, choć pomimo brania pod uwagę średniej, nadal występowały spadki. Kolor żółty oznacza skuteczność w okolicy 80% nauczenia się sieci. Zmienna $S1$ pracowała w przedziale od 10 do 100 neuronów, a zmienna $S2$ od 10 do danej wartości $S1$. Najbardziej optymalne wyniki sieć osiągnęła dla wartości zmiennych około $S1 = 90$ i $S2 = 70$, aczkolwiek wpływ ilości neuronów nie był jakoś zauważalny i z pewnością dopiero kolejne testy pomogą wywnioskować, jakie parametry mają największy wpływ na nauczenie sieci.



Rysunek 18: Domyślny wykres z Matlaba pokazujący w której epoce został osiągnięty minimalny błąd w ostatniej konfiguracji parametrów podczas trenowania sieci.



Rysunek 18.1: Wykres powierzchniowy błędu końcowego od neuronów w odpowiednich warstwach.

Minimalny błąd nie przekroczył wartości 0.14, a sam wykres ma mniejsze skoki niż poprzedni bez wyciągania średniej z eksperymentów.

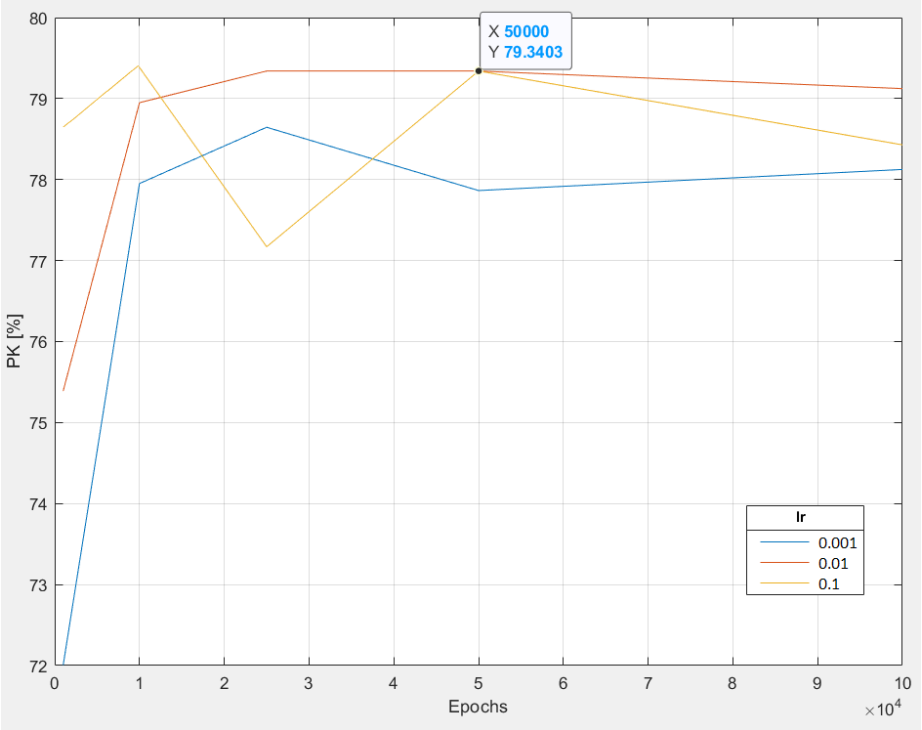
Uwagi dodatkowe:

- przy ustalaniu górnej granicy ilości neuronów, należy zwrócić szczególną uwagę na współczynnik uczenia, ponieważ jak można wyczytać w dokumentacji, w miarę wzrostu ilości neuronów należy zmniejszać lr, gdyż testy mogą wyjść zakłamane i z błędnymi wynikami
- powierzchnie wykresów mają kształt trójkąta, ponieważ zastosowano się do rady prowadzącego; ilość neuronów w warstwie drugiej $S2$ nie może przekroczyć ilości neuronów w warstwie pierwszej $S1$

Nr 2 - maksymalna ilość epok

Drugim eksperymentem była zmiana maksymalnej ilości epok. Przeprowadzono dwa testy, dla dwóch różnych maksymalnych wartości epok.

Pierwszym był test przedstawiający wyniki podczas dążenia do miliona epok ze zmianą jedynie liczby neuronów w warstwach. Wyniki były zbliżone do tych w drugim eksperymencie. Nie było to do końca poprawne podejście, ponieważ należało dodatkowo zmieniać współczynnik uczenia, dlatego eksperyment został przerwany mimo ok. 3h czasu jakie zostały już poświęcone. Lr był korygowany w dalszych testach, co przedstawia wykres poniżej:



Rysunek 20: Wpływ ilości epok na poziom nauczenia się sieci z uwzględnieniem lr.

Jak widać z tabeli oraz wykresów dla większej wartości maksymalnej ilości epok, poziom nauczenia się sieci, nie wzrasta od pewnej ilości epok (ok. 10 000). Po przeprowadzeniu kilku testów wstępnych, należało zwrócić uwagę na współczynnik uczenia, który silnie koreluje z ilością epok i dlatego należało użyć kilku jego wartości by móc stwierdzić jak ilość epok wpływa na uczenie się sieci. Niestety w pierwszej tabeli, nie było zmiany współczynnika uczenia co było błędem, ponieważ dopiero kolejne eksperymenty ukazały jego związek z liczbą epok.

Wyniki osiągane dla 10 tys. i 50 tys. epok były jednymi z najwyższych i dodatkowo były bardzo zbliżone do siebie, dlatego nie było sensu tracić czasu w innych eksperymentach dla 50 tys. epok.

Nr 3 i 4 - współczynnik uczenia i stała momentum

Podczas ostatniego testu badano zmianę nauczenia się sieci poprzez współczynnik uczenia (*lr)* oraz stałą momentum (*mc). Lr* mógł przyjmować wartości z zakresu 0.1 do 0.000001, a stała *mc* wartości od 0.1 do 0.99.

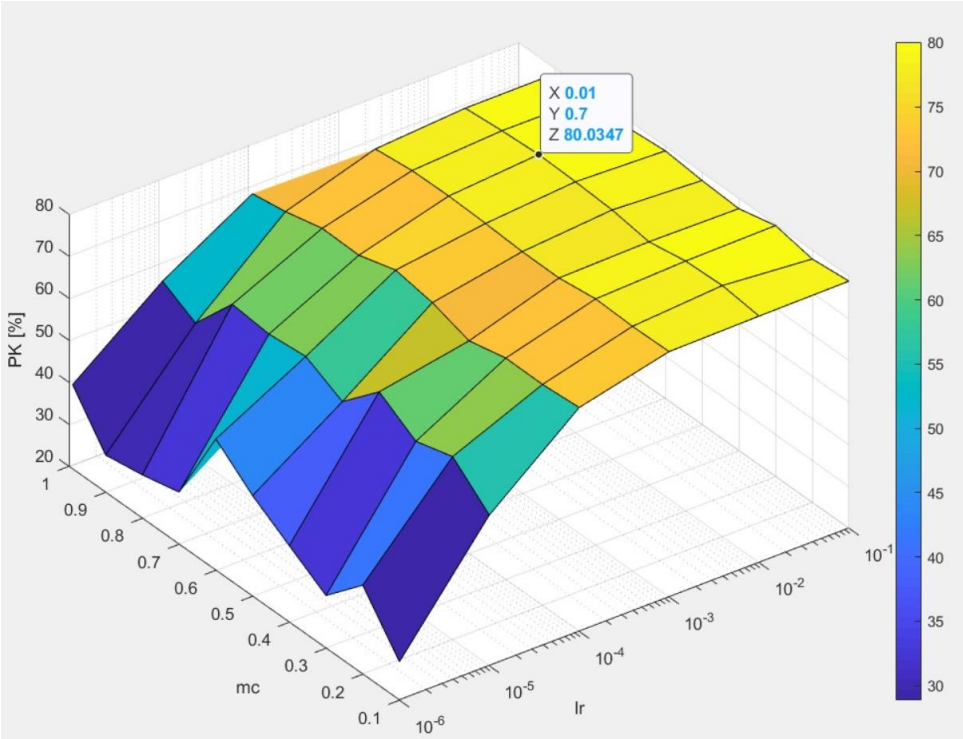
Dla max epoch = 10 000

```
S1 = 50 S2 = 30 lr = 1.000000e-06 mc = 9.90e-01 PK = 4.006076e+01 MSE= 5.897323e-01 Epoka = 10000
S1 = 50 S2 = 30 lr = 1.000000e-05 mc = 1.00e-01 PK = 5.590278e+01 MSE= 2.482027e-01 Epoka = 10000
S1 = 50 S2 = 30 lr = 1.000000e-05 mc = 2.00e-01 PK = 6.362847e+01 MSE= 3.394290e-01 Epoka = 10000
S1 = 50 S2 = 30 lr = 1.000000e-05 mc = 3.00e-01 PK = 6.106771e+01 MSE= 3.445234e-01 Epoka = 10000
S1 = 50 S2 = 30 lr = 1.000000e-05 mc = 4.00e-01 PK = 6.666667e+01 MSE= 2.180738e-01 Epoka = 10000
S1 = 50 S2 = 30 lr = 1.000000e-05 mc = 5.00e-01 PK = 5.815972e+01 MSE= 3.002865e-01 Epoka = 10000
S1 = 50 S2 = 30 lr = 1.000000e-05 mc = 6.00e-01 PK = 6.271701e+01 MSE= 2.591955e-01 Epoka = 10000
S1 = 50 S2 = 30 lr = 1.000000e-05 mc = 7.00e-01 PK = 6.184896e+01 MSE= 2.732705e-01 Epoka = 10000
S1 = 50 S2 = 30 lr = 1.000000e-05 mc = 8.00e-01 PK = 6.276042e+01 MSE= 2.343653e-01 Epoka = 10000
S1 = 50 S2 = 30 lr = 1.000000e-05 mc = 9.00e-01 PK = 5.208333e+01 MSE= 3.573751e-01 Epoka = 10000
S1 = 50 S2 = 30 lr = 1.000000e-05 mc = 9.90e-01 PK = 5.651042e+01 MSE= 3.189783e-01 Epoka = 10000
S1 = 50 S2 = 30 lr = 1.000000e-04 mc = 1.00e-01 PK = 7.330729e+01 MSE= 1.618196e-01 Epoka = 10000
S1 = 50 S2 = 30 lr = 1.000000e-04 mc = 2.00e-01 PK = 7.261285e+01 MSE= 1.757979e-01 Epoka = 10000
S1 = 50 S2 = 30 lr = 1.000000e-04 mc = 3.00e-01 PK = 7.269965e+01 MSE= 1.653818e-01 Epoka = 10000
S1 = 50 S2 = 30 lr = 1.000000e-04 mc = 4.00e-01 PK = 7.061632e+01 MSE= 1.949231e-01 Epoka = 10000
S1 = 50 S2 = 30 lr = 1.000000e-04 mc = 5.00e-01 PK = 7.352431e+01 MSE= 1.754164e-01 Epoka = 10000
S1 = 50 S2 = 30 lr = 1.000000e-04 mc = 6.00e-01 PK = 7.508681e+01 MSE= 1.729210e-01 Epoka = 10000
S1 = 50 S2 = 30 lr = 1.000000e-04 mc = 7.00e-01 PK = 7.243924e+01 MSE= 1.896615e-01 Epoka = 10000
S1 = 50 S2 = 30 lr = 1.000000e-04 mc = 8.00e-01 PK = 7.265625e+01 MSE= 2.117279e-01 Epoka = 10000
S1 = 50 S2 = 30 lr = 1.000000e-04 mc = 9.00e-01 PK = 7.044271e+01 MSE= 1.846790e-01 Epoka = 10000
```

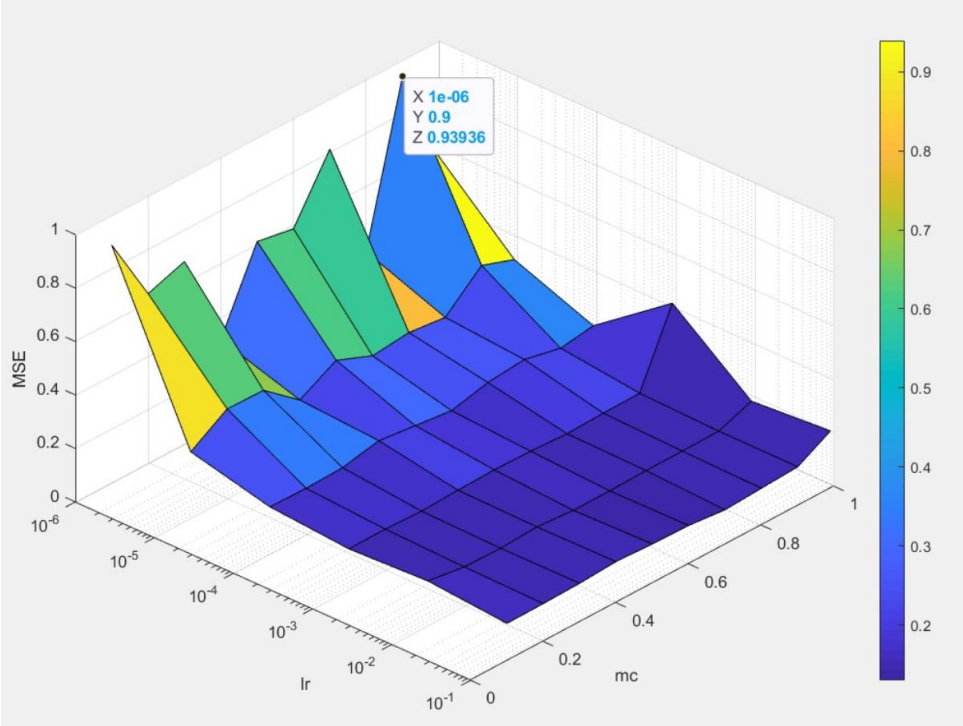
Rysunek 21: Dane bezpośrednio z konsoli Matlaba

Dla max epoch = 20 000

S1	S2	lr	mc	PK [%]	MSE	epoch
40	20	1e-05	0.4	60.59	0.3204	20000
40	20	1e-05	0.5	62.98	0.2141	20000
40	20	1e-05	0.6	65.45	0.2407	20000
40	20	0.0001	0.6	73.31	0.1807	20000
40	20	0.0001	0.7	74.26	0.1753	20000
40	20	0.0001	0.8	74.91	0.1758	20000
40	20	0.001	0.9	78.34	0.1528	20000
40	20	0.001	0.95	78.82	0.1525	20000
40	20	0.001	0.099	78.17	0.1509	20000
40	20	0.01	0.1	79.56	0.1361	11691
40	20	0.01	0.2	78.43	0.1502	2392
40	20	0.01	0.3	79.25	0.1421	3468
40	20	0.01	0.4	78.47	0.1426	7066



Rysunek 22: Wykres powierzchniowy przedstawiający wpływ współczynników mc i lr na nauczenie się sieci



Rysunek 23: Wykres przedstawiający wpływ współczynników mc i lr na błąd końcowy sieci

Zmniejszenie współczynnika uczenia lr ma bardzo duży wpływ na zdolność uczenia się sieci. Gdy zmienna lr przyjmowała zbyt małe wartości, nauczenie sieci było na bardzo niskim poziomie. Podczas testów, należało również zwrócić uwagę na ilość epok, ponieważ im lr był większy i stopniowo od 0.1 zbliżał się do jedynki, tym ilość epok należało drastycznie zmniejszać, ponieważ sieć przestawała się uczyć. Jeśli współczynnik zmniejszano, to sieć uczyła się wolniej i należało zwiększyć ilość epok by osiągnąć cel. Dobrym pomysłem wydaje się ustawienie lr na poziomie 0.01, choć złotym środkiem jest obniżenie go do 0.001 i ustawienie większej ilości epok.

Zmiana stałej momentum również ma wpływ na zdolność uczenia się naszej sieci. Im większa wartość zmiennej mc tym większa szansa nauczenia się sieci, ale należało zwrócić uwagę na fakt, że często powyżej wartości 0.9 nie opłacało się ustawiać tego współczynnika, ponieważ sieć uzyskiwała gorsze wyniki. Najlepszymi wynikami mogła poszczycić się dla mc mieszczącego się w okolicy 0.8, chociaż należy zwrócić uwagę na konieczność zmiany mc dla różnych wartości współczynnika uczenia i czasami jego niska wartość (ok. 0.4), również korzystnie wpływała na naukę.

7 WNIOSKI

Podczas próby nauczenia sieci oszacowania zapadania na chorobę jaką jest cukrzyca u Indian przeprowadzono szereg eksperymentów w celu otrzymania najbardziej optymalnych wyników. Zmianom ulegały następujące zmienne:

- **ilość neuronów w warstwie pierwszej** $S1$ [10 co 10 do 100]
- **ilość neuronów w warstwie drugiej** $S2$ [10 co 10 do S1]
- **współczynnik uczenia** lr [0.000001, 0.00001, 0.0001, 0.001, 0.01, 0.1]
- **stała momentum** mc [0.1 co 0.1 do 0.9 i 0.99]
- **eksperyment** ex [1 - 5]
- **maksymalna ilość epok** $max\ epoch$ [1000, 10 000, 25 000, 50 000, 75 000, 100 000]

Otrzymane wyniki eksperymentów w większości zbliżyły się do wartości 80% lub ją osiągnęły. Z przeprowadzonych prób i testów jasno wynika, iż zmiany, którym ulegały powyższe zmienne wpływały na zdolność uczenia się sieci jak i na czas potrzebny do uzyskania satysfakcjonujących wyników.

Podczas zmiany ilości neuronów w poszczególnych warstwach, dochodziło do zmian otrzymanych wyników, lecz były to zmiany bardzo małe. Nie zauważono żadnej analogi, na podstawie której dałoby się wyprowadzić istotne i przekonywujące wnioski o wpływie ilości neuronów na zdolność uczenia się sieci.

Na koniec przedstawiono zmienne, które w największym stopniu wpływają na zdolność uczenia się sieci. Mowa tutaj o współczynniku uczenia oraz stałej momentum. Zmniejszając wartość zmiennej lr z 0.1 do 0.001 zwiększyliśmy ilość przypadków, w których sieć zdołała się nauczyć w większym stopniu, lecz wydłużało to cały proces. Podobnie podczas zmiany stałej momentum. W moich eksperymentach mogła ona przyjmować wartości: od 0.1 do 0.99, dzięki temu łatwo dostrzec wpływ tej zmiany na ostateczne wyniki. Podsumowując - te dwie zmienne powodują najbardziej widoczne zmiany w otrzymanych rezultatach, a przez to wpływają bezpośrednio na proces uczenia się, ale należy mieć na uwadze, że zmiana tych dwóch parametrów pociąga za sobą konieczność zmian innych (m.in. liczba epok) i nie można było przeprowadzać eksperymentów zmieniając wyłącznie te dwa parametry.

Najlepsze wyniki osiągano dla:

- lr = 0.001
- Epoki = 20 000
- mc = 0.8
- Ilość neuronów nie miała większego wpływu, pod warunkiem, że nie przekraczała “bezpiecznej wartości” w okolicach 90 dla S1 i 70 dla S2, po której należało odpowiednio dostosowywać lr i epoki, aby osiągnąć sensowne wyniki

Najlepsza osiągnięta poprawność klasyfikacji była możliwa do uzyskania, po przeprowadzeniu wielu eksperymentów, a nie wszystkie z nich zostały przedstawione w projekcie, po to by wyróżnić tylko te najistotniejsze. Maksymalne PK wyniosło 82.68%, dla konfiguracji przedstawionej powyżej.

Sieć jak i wszystkie wykresy zostały stworzone w programie **MATLAB R2019a**, co zmusiło do odpowiedniego zmodyfikowania pewnych nieużywanych już w nim poleceń ze starszych wersji programu, w których oryginalny algorytm learnBPM powstał i był bazą, na podstawie której odtworzono jego działanie w najnowszej wersji programu.

8 BIBLIOGRAFIA

Większość danych potrzebnych na zrealizowanie projektu, w tym większa część rysunków, została zaczerpnięta z materiałów stworzonych przez dr hab. inż. Romana Zajdla ze strony:

(1) http://materialy.prz-rzeszow.pl/

Pozostałe źródła:

(2) http://www.rosczak.com/aebp/algorithm.html
(3) https://pl.wikipedia.org/wiki/Neuron
(4) https://ch.mathworks.com/help/matlab/