



**WYDZIAŁ  
ELEKTROTECHNIKI  
I INFORMATYKI**  
POLITECHNIKI RZESZOWSKIEJ

## Sztuczna inteligencja - projekt

Realizacja sieci neuronowej uczonej algorytmem  
wstecznej propagacji błędu (learnbp) uczącej się  
rozpoznawania kwiatów irysa

Autor: Brajan Miśkowicz  
Nr albumu: 163976  
Grupa: EF-DI, P4

Rzeszów, 2021



# Spis treści

<b>1. Wstęp</b>	<b>4</b>
1.1. Opis projektu	4
1.2. Przedstawienie danych uczących	4
<b>2. Teoria</b>	<b>5</b>
2.1. Neuron biologiczny	5
2.2. Neuron sztuczny	6
2.3. Sieć jednokierunkowa jednowarstwowa	7
2.4. Sieć neuronowa wielowarstwowa	9
2.5. Propagacja wsteczna	10
2.5.1. Uczenie pod nadzorem pojedynczej warstwy neuronów	10
2.5.2. Uczenie pod nadzorem sieci wielowarstwowej	13
<b>3. Eksperymenty</b>	<b>15</b>
3.1. Eksperyment 1 - podział danych	15
3.2. Eksperyment 2 - ilość neuronów na poszczególnych warstwach	18
3.2.1. Dla współczynnika uczenia $lr = 0.1$	20
3.2.2. Dla współczynnika uczenia $lr = 0.01$	22
3.2.3. Dla współczynnika uczenia $lr = 0.001$	24
3.2.4. Dla współczynnika uczenia $lr = 0.0001$	26
3.2.5. Dla współczynnika uczenia $lr = 0.00001$	28
3.2.6. Zagęszczone poszukiwania	30
3.3. Eksperyment 3 - wielkość parametru współczynnika uczenia	32
3.3.1. Współczynnik uczenia przy średniej ze wszystkich kombinacji ilości neuronów	32
3.3.2. Współczynnik uczenia dla optymalnej ilości neuronów	35
<b>4. Wnioski</b>	<b>38</b>
<b>Literatura</b>	<b>40</b>



# 1. Wstęp

## 1.1. Opis projektu

Celem jest realizacja sieci neuronowej uczonej algorytmem wstecznej propagacji błędu (learnbp) uczącej się rozpoznawania kwiatów irysa. Projekt polega na napisaniu odpowiednich skryptów pozwalających na wykorzystanie sieci neuronowej - nauczanie jej rozpoznawania kwiatów irysa częścią danych - danymi uczącymi, oraz wypróbowanie działania sieci na pozostałych danych - testujących. Ponadto dzięki analizie otrzymanych z danych testowych wyników kolejnych eksperymentów należy wyznaczyć takie wartości parametrów występujących w algorytmie wstecznej propagacji, aby sieć działała możliwie najlepiej, co zostanie opisane szerzej podczas eksperymentów.

## 1.2. Przedstawienie danych uczących

Używanym zbiorem podczas projektu jest zbiór danych kwiatów irysa dostępny na stronie: <http://archive.ics.uci.edu/ml/datasets/Iris>. W zbiorze znajdują się cztery parametry opisujące trzy gatunki kwiatów irysa: Iris Setosa, Iris Versicolour, oraz Iris Virginica. Wśród opisu zbioru można znaleźć informacje między innymi o tym, że znajduje się w nim 150 instancji, po 50 z każdego gatunku.

Wspomniane wcześniej parametry opisujące kwiaty to:

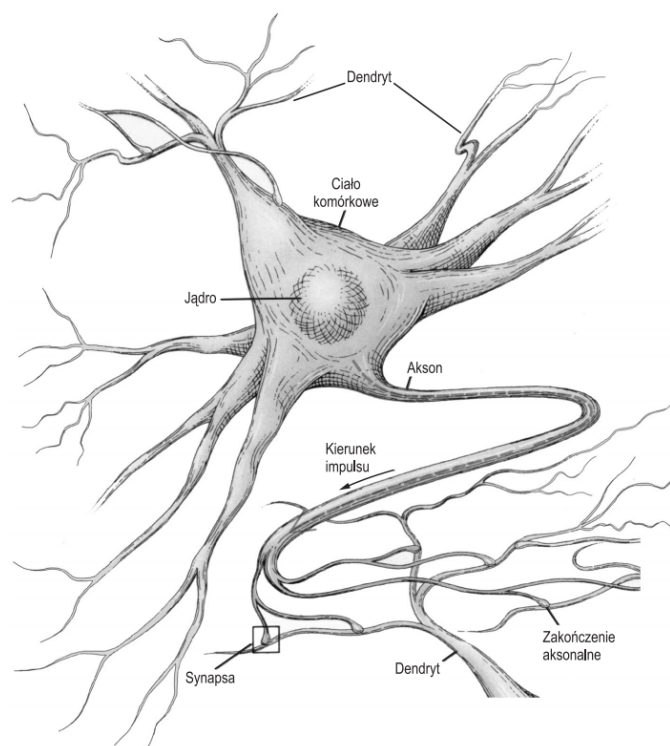
- długość kielicha [cm],
- szerokość kielicha [cm],
- długość płatków [cm],
- szerokość płatków [cm].

W zbiorze nie pojawiają się braki danych co ułatwia ich normalizację. Na potrzeby projektu gatunki Iris Setosa, Iris Versicolour, oraz Iris Virginica zostały oznaczone podczas testów odpowiednio jako 1, 2 i 3. Z racji rozbieżności zakresów wartości parametrów danych zostały znormalizowane do zakresu wartości od -1 do 1.

## 2. Teoria

### 2.1. Neuron biologiczny

Najważniejszym organem układu nerwowego człowieka jest mózg, który kontroluje ciało i umysł człowieka. Na podstawie wiedzy, którą dzisiaj dysponujemy można stwierdzić, że mózg człowieka jest najbardziej złożonym obiektem we wszechświecie, jaki znamy. Komórki, z których składa się mózg, są nazywane neuronami. Każdy z neuronów za pomocą wypustek – dendrytów (gr. dendron – drzewo) może być połączony z wieloma innymi neuronami, istnieją neurony połączone z kilkoma tylko sąsiadami, ale są i takie, które mają dziesiątki tysięcy sąsiadów. Sygnały wejściowe doprowadzane są do komórki za pośrednictwem synaps, zaś sygnał wyjściowy odprowadzany jest za pomocą aksonu i jego odgałęzień, które docierają do dendrytów innych neuronów tworząc kolejne synapsy. W ten sposób każdy neuron może komunikować się z wieloma innymi, nawet odległymi neuronami. Ogólnie mówiąc, aktywność mózgu polega na generowaniu przez neurony i przesyłaniu między nimi impulsów.

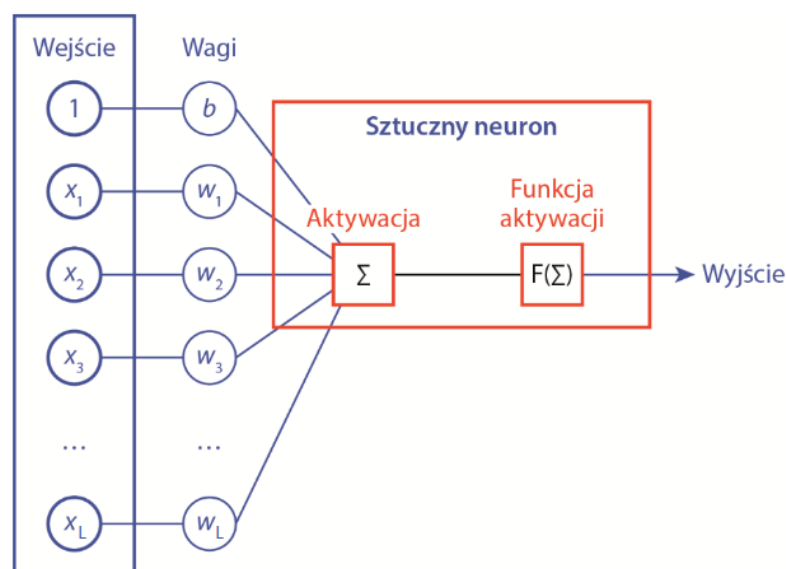


Rysunek 2.1: Wygląd typowego neuronu i jego części składowych[1]

Przekazywanie impulsu nerwowego od jednej komórki do drugiej opiera się na wydzielaniu pod wpływem nadchodzących od synaps bodźców substancji chemicznych nazywanych neuromediatorami, które oddziałują na błonę komórki, powodując zmianę jej potencjału, gdzie ta zmiana jest proporcjonalna do ilości neuromediatora. Synapsy różnią się od siebie wielkością oraz możliwościami gromadzenia tegoż neuromediatora, dlatego też impulsy docierające do neuronów za pomocą konkretnych synaps mogą powodować pobudzenia o różnej sile. Wynika z tego, że można wejściom komórki przypisać wagi, odpowiadające ilości wydzielanego jednorazowo na poszczególnych synapsach neuromediatora.

## 2.2. Neuron sztuczny

Podstawowym elementem sieci neuronowej jest sztuczny neuron, którego działanie wzorowane jest na funkcjonowaniu biologicznej komórki nerwowej, ale w uproszczonej formie. Pojedynczy neuron odpowiada za przetwarzanie danych wejściowych na wartość wyjściową, która jest zależna od tego, jak dany neuron reaguje na te dane. Neuron reprezentuje jednostkę przetwarzania, która przyjmuje wartość wejściową i zgodnie z określonymi regułami przekazuje inną wartość na wyjściu. Potraktować można więc każdy neuron jak swego rodzaju procesor, który sumuje z odpowiednimi wagami sygnały wejściowe pochodzące z innych neuronów, następnie tworzy nieliniową funkcję i przekazuje tę wartość do kolejnych neuronów z nim powiązanych.



Rysunek 2.2: Budowa sztucznego neuronu[7]

Powyższy rysunek pokazuje model neuronu posiadający  $L$  wejść  $x$  i jedno wyjście. Najpierw sumowane są wszystkie składowe przemnożone przez ich wagi  $w$ , a później wynik sumowania poddawany jest działaniu funkcji aktywacji, która generuje wartość na wyjściu. Nad wejściami można zauważyć tak zwane biasy, które pojawiają się na dodatkowym wejściu neuronu, mają stałą wartość - równą 1 - oznaczoną jako  $b$  i mają za zadanie zmieniać wartość zwracaną przez neurony tak, by sieć lepiej reprezentowała zadanie. Z tą wiedzą o neuronie można zapisać wzór opisujący sztuczny neuron:

$$y = f\left(\sum_{j=1}^L w_j x_j + b\right) \quad (2.1)$$

$y$  jest tutaj wyjściem neuronu, natomiast  $f$  oznacza funkcję aktywacji, ale można zapisać ją w prostrzy sposób traktując  $x$  jako wektor sygnałów wejściowych oraz  $w$  jako macierz wierszową wag[4], tak więc:

$$x = [x_1, x_2, \dots, x_L]^T \quad (2.2)$$

$$w = [w_1, w_2, \dots, w_L] \quad (2.3)$$

oraz

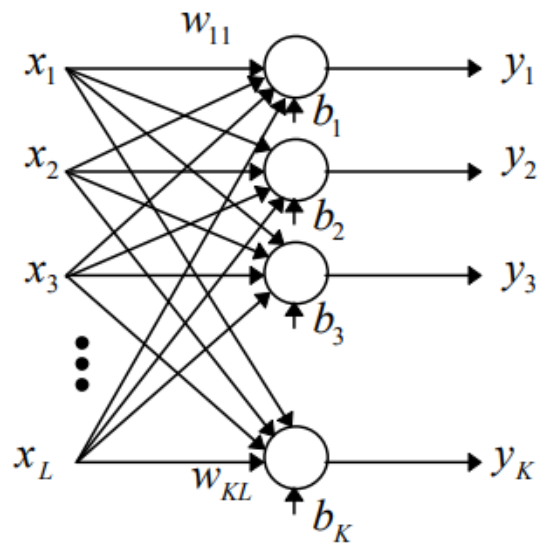
$$y = f(wx + b) \quad (2.4)$$

### 2.3. Sieć jednokierunkowa jednowarstwowa

Neurony połączone między sobą tworzą układ nazywany sztuczną siecią neuronową (w skrócie siecią neuronową). W zależności od sposobu połączenia neuronów można wyróżnić sieci jednokierunkowe lub rekurencyjne (ze sprzężeniem zwrotnym). Stosowana jest w tym projekcie jest jedna z najpowszechniejszych sieci - sieć jednokierunkowa. W sieciach jednokierunkowych przepływ sygnałów odbywa się w nich wyłącznie w kierunku od wejścia (poprzez ewentualne warstwy ukryte) do wyjścia. Wykluczony jest przepływ sygnałów w drugą stronę, co powoduje, że sieci tego typu są przeciwstawiane sieciom rekurencyjnym. Typowa sieć jednokierunkowa posiada pełne połączenia między warstwami, tj. wyjście neuronu z jednej warstwy przekazywane jest na wejście wszystkich na warstwie kolejnej. Połączenia międzywarstwowe występują jedynie między sąsiednimi warstwami.



Najprostszą siecią neuronową jednokierunkową jest sieć jednowarstwowa widoczna poniżej, tworzą ją neurony ułożone w jednej warstwie.



Rysunek 2.3: Sieć neuronowa jednowarstwowa[5]

Mając uzyskany wcześniej obraz neuronu i jego działania:

$$y = f(wx + b) \quad (2.5)$$

można w sposób podobny zapisać jak funkcjonuje cała warstwa neuronów za pomocą tego samego wzoru, w tym przypadku jednak  $b$  będzie wektorem przesunięć, a  $y$  - wektorem sygnałów wyjściowych, a nie skalarami. Dodatkowo w tutaj nie będzie macierzą wierszową, a dwuwymiarową o wymiarach  $K \times L$ , gdzie  $K$  to liczba neuronów, a  $L$  - liczba sygnałów wejściowych [5]:

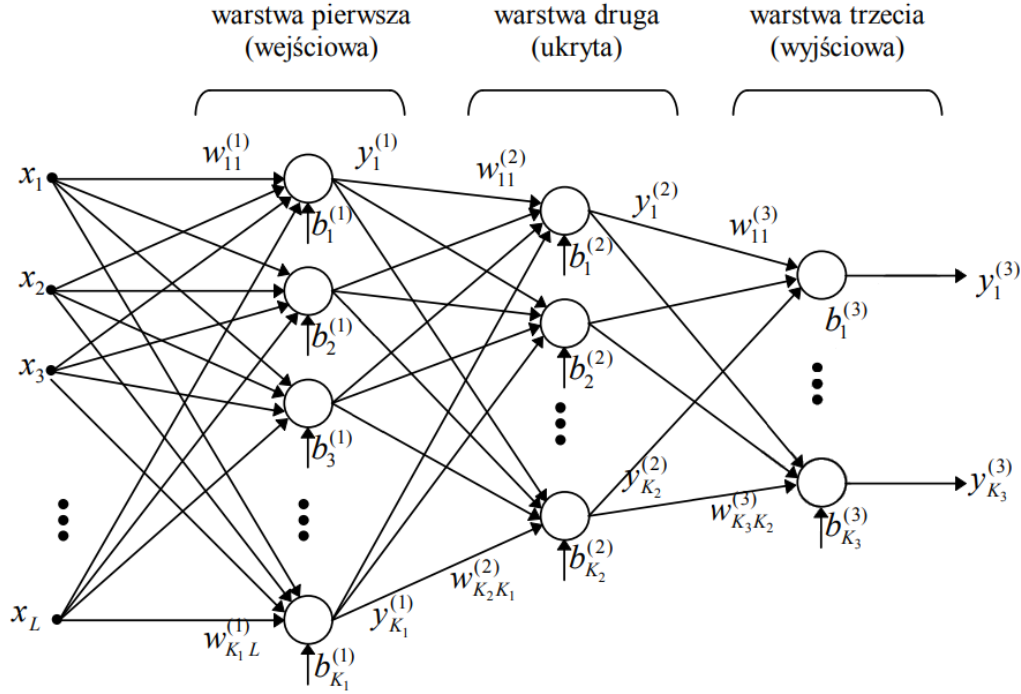
$$b = [b_1, b_2, \dots, b_K]^T \quad (2.6)$$

$$y = [y_1, y_2, \dots, y_K]^T \quad (2.7)$$

$$w = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1L} \\ w_{21} & w_{22} & \cdots & w_{2L} \\ \cdots & \cdots & \ddots & \cdots \\ w_{K1} & w_{K2} & \cdots & w_{KL} \end{bmatrix} \quad (2.8)$$

## 2.4. Sieć neuronowa wielowarstwowa

Sieć wielowarstwowa różni się od jednowarstwowej tym, że zawiera ona co najmniej jedną warstwę ukrytą.



Rysunek 2.4: Sieć jednokierunkowa wielowarstwowa[6]

Przedstawiona została powyżej sieć neuronowa wielowarstwowa, z warstwą ukrytą. Widać tutaj dokładnie to, co zostało wspomniane wcześniej - tzn. wejściem warstwy drugiej jest wyjście warstwy pierwszej, a wejściem trzeciej wyjście drugiej.

Mając uzyskany wcześniej wzór opisujący funkcjonowanie całej warstwy neuronów oraz wiedzę na temat budowy sieci można wyznaczyć wzory dla poszczególnych warstw - przykładowo, jak w przypadku dla trzech warstw:

$$y^{(1)} = f^{(1)}(w^{(1)}x + b^{(1)}) \quad (2.9)$$

$$y^{(2)} = f^{(2)}(w^{(2)}y^{(1)} + b^{(2)}) \quad (2.10)$$

$$y^{(3)} = f^{(3)}(w^{(3)}y^{(2)} + b^{(3)}) \quad (2.11)$$

Oznaczenia tutaj są identyczne, jak wcześniej. Górne indeksy mówią natomiast, z której

warstwy brane są wartości. Po wyznaczeniu wzoru na poszczególne warstwy możliwe jest opisanie całej sieci za pomocą jednego wzoru:

$$y^{(3)} = f^{(3)} \left( w^{(3)} f^{(2)} \left( w^{(2)} f^{(1)} \left( w^{(1)} x + b^{(1)} \right) + b^{(2)} \right) + b^{(3)} \right) \quad (2.12)$$

## 2.5. Propagacja wsteczna

W algorytmie wstecznej propagacji najpierw obliczany jest sygnał wyjściowy, a na jego podstawie błąd warstwy wyjściowej. Następnie odwórcony zostaje kierunek przepływu sygnałów, a funkcje aktywacji zostają zastąpione przez ich pochodne, a wagi są odpowiednio aktualizowane. Propagacja wsteczna jest metodą uczenia nadzorowanego - tzn. że dla każdego sygnału wejściowego dana jest pożądana odpowiedź.

### 2.5.1. Uczenie pod nadzorem pojedynczej warstwy neuronów

Każda sieć neuronowa jest przybliżeniem funkcji, więc różni się od niej o pewną wartość. Ta wartość to błąd, a naszym celem powinno być dążenie do zminimalizowania go. Ponieważ w sieci neuronowej błąd jest funkcją wag, to chcemy więc zminimalizować błędy w stosunku do wag. Wiedząc, że podczas uczenia każdy wektor wejściowy  $x$  posiada swój odpowiednik w postaci pożądanego wektora sygnałów wyjściowych:

$$x = [x_1, x_2, \dots, x_L]^T \quad (2.13)$$

$$\hat{y} = [\hat{y}_1, \hat{y}_2, \dots, \hat{y}_K]^T \quad (2.14)$$

To błąd sieci można zdefiniować następująco:

$$e = y - \hat{y} \quad (2.15)$$

Z racji faktu, iż poszukuje się zgodności odpowiedzi  $y$  z pożądanym wyjściem  $\hat{y}$ , to cel uczenia można określić jako minimalizację pewnej funkcji, którą najczęściej jest błąd średniokwadratowy wyznaczany dla wszystkich  $K$  neuronów:

$$E = \frac{1}{2} \sum_{j=1}^K e_j^2 \quad (2.16)$$

Wzór opisujący wartość nowej wagi dla  $i$ -tego neuronu i  $j$ -tej wagi jest postaci:

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}(t) \quad (2.17)$$

gdzie  $t$  oznacza epokę, czyli numer cyklu uczenia. Kierunek minimalizacji tworzony jest na podstawie przestrzeni wielowymiarowej w taki sposób, aby wagi podążały w kierunku punktu o najmniejszej wartości błędu. Stosowany do wyznaczenia kierunku jest wektor gradientu względem wag wszystkich warstw sieci. Zmiana wartości wag jest równa:

$$\Delta w = -\eta \nabla E(w) \quad (2.18)$$

gdzie  $\eta$  jest współczynnikiem uczenia, a  $\nabla$  gradientem wskazującym najszybszy kierunek wzrastania funkcji - z jego powodu wynika znak minusa, ponieważ zmiana  $w$  powinna prowadzić do kierunku przeciwnego - najszybszego malenia błędu. Z poprzedniego wzoru dla tego samego jednego neuronu wynika:

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} \quad (2.19)$$

Z racji tego, że  $E$  zależne jest od wagi poprzez pobudzenie, oraz wyjście neuronu, to:

$$E = E(y_i(z_i(w_{ij}))) \quad (2.20)$$

A pochodna będzie równa:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial y_i} \frac{\partial y_i}{\partial z_i} \frac{\partial z_i}{\partial w_{ij}} \quad (2.21)$$

Korzystając z otrzymanego wcześniej wzoru na popełniany błąd i błąd średniokwadratowy można obliczyć, że:

$$\frac{\partial E}{\partial y_i} = \frac{\partial}{\partial y_i} \left( \frac{1}{2}(y_1 - \hat{y}_1)^2 + \dots + \frac{1}{2}(y_i - \hat{y}_i)^2 + \dots + \frac{1}{2}(y_K - \hat{y}_K)^2 \right) = y_i - \hat{y}_i = e_i \quad (2.22)$$

Na podstawie zależności opisującej działanie neuronu natomiast:

$$\frac{\partial z_i}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \left( w_{i1}x_1 + \dots + w_{ij}x_j + \dots + w_{iL}x_L + b_i \right) = x_j \quad (2.23)$$

Podłożyć można otrzymane pochodne cząstkowe do otrzymanej wcześniej pochodnej:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial y_i} \frac{\partial y_i}{\partial z_i} \frac{\partial z_i}{\partial w_{ij}} = e_i \cdot \frac{\partial y_i}{\partial z_i} \cdot x_j \quad (2.24)$$

Natomiast pochodną do wzoru na zmianę pojedynczej wagi:

$$\Delta w_{ij} = \eta(\hat{y}_1 - y_1) \cdot \frac{\partial y_i}{\partial z_i} \cdot x_j \quad (2.25)$$

Przyjmując, że zależność  $y = y(z)$  jest sigmoidalna, to ostatecznie zmiana wagi będzie równa:

$$\Delta w_{ij} = \eta(\hat{y}_1 - y_1)(1 - y_i)y_i x_j \quad (2.26)$$

### 2.5.2. Uczenie pod nadzorem sieci wielowarstwowej

Jak już wcześniej wspomniano każdy neuron posiada funkcję aktywacji, te funkcje są takie same dla neuronów konkretnych warstw. Sieci jednokierunkowe wielowarstwowe w warstwach wejściowej i ukrytej najczęściej wykorzystują funkcje aktywacji typu sigmoidalnego – tansig lub logsig, ale ich minusem jest wąski zakres sygnału wyjściowego, tzn. od 0 do 1, lub -1 do 1. Z tego też powodu w warstwie wyjściowej wykorzystywana jest nie posiadająca takich ograniczeń funkcja liniowa purelin. Funkcja błędu dla sieci wielowarstwowej przyjmuje postać:

$$\begin{aligned}
 E &= \frac{1}{2} \sum_{i_3=1}^{K_3} e_{i_3}^2 = \frac{1}{2} \sum_{i_3=1}^{K_3} \left( y_{i_3}^{(3)} - \hat{y}_{i_3} \right)^2 = \\
 &= \frac{1}{2} \sum_{i_3=1}^{K_3} \left( f^{(3)} \left( \sum_{i_2=1}^{K_2} w_{i_3 i_2}^{(3)} y_{i_2} + b_{i_3}^{(3)} \right) - \hat{y}_{i_3} \right)^2 = \\
 &= \frac{1}{2} \sum_{i_3=1}^{K_3} \left( f^{(3)} \left( \sum_{i_2=1}^{K_2} w_{i_3 i_2}^{(3)} f^{(2)} \left( \sum_{i_1=1}^{K_1} w_{i_2 i_1}^{(2)} y_{i_1} + b_{i_2}^{(2)} \right) + b_{i_3}^{(3)} \right) - \hat{y}_{i_3} \right)^2 = \\
 &= \frac{1}{2} \sum_{i_3=1}^{K_3} \left( f^{(3)} \left( \sum_{i_2=1}^{K_2} w_{i_3 i_2}^{(3)} f^{(2)} \left( \sum_{i_1=1}^{K_1} w_{i_2 i_1}^{(2)} f^{(1)} \left( \sum_{j=1}^L w_{i_1 j}^{(1)} x_j + b_{i_1}^{(1)} \right) + b_{i_2}^{(2)} \right) + b_{i_3}^{(3)} \right) - \hat{y}_{i_3} \right)^2
 \end{aligned} \tag{2.27}$$

$j = 1, \dots, L$  to numery wejść warstwy pierwszej, a  $j_1 = 1, \dots, K_1; j_2 = 1, \dots, K_2; j_3 = 1, \dots, K_3$ ; to numery wejść warstw pierwszej, drugiej i trzeciej.

Rozpoczynając od warstwy wyjściowej składniki gradientu funkcji celu względem wag neuronów poszczególnych warstw otrzymuje się przez różniczkowanie powyższej zależności:

$$\frac{\partial E}{\partial w_{i_3 i_2}^{(3)}} = \frac{\partial E}{\partial f^{(3)}} \frac{\partial f^{(3)} \left( z_{i_3}^{(3)} \right)}{\partial z_{i_3}^{(3)}} \frac{\partial z_{i_3}^{(3)}}{\partial w_{i_3 i_2}^{(3)}} = (y_{i_3} - \hat{y}_{i_3}) \frac{\partial f^{(3)} \left( z_{i_3}^{(3)} \right)}{\partial z_{i_3}^{(3)}} y_{i_2}^{(2)} \tag{2.28}$$

gdzie łączne pobudzenie  $i_3$ -tego neuronu warstwy wyjściowej jest równe:

$$z_{i_3}^{(3)} = \sum_{i_2=1}^{K_2} w_{i_3 i_2}^{(3)} y_{i_2} + b_{i_3}^{(3)} \tag{2.29}$$

Podkładając do poprzedniego równania zależność:

$$\delta_{i_3}^{(3)} = (y_{i_3} - \hat{y}_{i_3}) \frac{\partial f^{(3)} \left( z_{i_3}^{(3)} \right)}{\partial z_{i_3}^{(3)}} \tag{2.30}$$

Otrzymane zostaje ostatecznie:

$$\frac{\partial E}{\partial w_{i_3 i_2}^{(3)}} = \delta_{i_3}^{(3)} y_{i_2}^{(2)} \quad (2.31)$$

Wyznaczyć można w podobny sposób elementy gradientu względem wag warstwy ukrytej:

$$\begin{aligned} \frac{\partial E}{\partial w_{i_2 i_1}^{(2)}} &= \frac{\partial E}{\partial f^{(3)}} \frac{\partial f^{(3)}(z_{i_3}^{(3)})}{\partial z_{i_3}^{(3)}} \frac{\partial z_{i_3}^{(3)}}{\partial f^{(2)}} \frac{\partial f^{(2)}(z_{i_2}^{(2)})}{\partial z_{i_2}^{(2)}} \frac{\partial z_{i_2}^{(2)}}{\partial w_{i_2 i_1}^{(2)}} = \\ &= \sum_{i_3=1}^{K_3} (y_{i_3} - \hat{y}_{i_3}) \frac{\partial f^{(3)}(z_{i_3}^{(3)})}{\partial z_{i_3}^{(3)}} w_{i_3 i_2}^{(3)} \frac{\partial f^{(2)}(z_{i_2}^{(2)})}{\partial z_{i_2}^{(2)}} y_{i_1}^{(1)} \end{aligned} \quad (2.32)$$

oraz wejściowej:

$$\begin{aligned} \frac{\partial E}{\partial w_{i_1 j}^{(1)}} &= \frac{\partial E}{\partial f^{(3)}} \frac{\partial f^{(3)}(z_{i_3}^{(3)})}{\partial z_{i_3}^{(3)}} \frac{\partial z_{i_3}^{(3)}}{\partial f^{(2)}} \frac{\partial f^{(2)}(z_{i_2}^{(2)})}{\partial z_{i_2}^{(2)}} \frac{\partial z_{i_2}^{(2)}}{\partial f^{(1)}} \frac{\partial f^{(1)}(z_{i_1}^{(1)})}{\partial z_{i_1}^{(1)}} \frac{\partial z_{i_1}^{(1)}}{\partial w_{i_1 j}^{(1)}} = \\ &= \sum_{i_3=1}^{K_3} (y_{i_3} - \hat{y}_{i_3}) \frac{\partial f^{(3)}(z_{i_3}^{(3)})}{\partial z_{i_3}^{(3)}} \sum_{i_2=1}^{K_2} \frac{\partial f^{(2)}(z_{i_2}^{(2)})}{\partial z_{i_2}^{(2)}} w_{i_2 i_1}^{(2)} \frac{\partial f^{(1)}(z_{i_1}^{(1)})}{\partial z_{i_1}^{(1)}} x_j \end{aligned} \quad (2.33)$$

### 3. Eksperymenty

#### 3.1. Eksperyment 1 - podział danych

Pierwszy eksperyment polega na dobraniu optymalnego podziału danych na dane uczące oraz dane testowe. Z racji braku wyznaczonej na początku optymalnej wartości współczynnika uczenia ( $lr$ ) oraz ilości neuronów na obu warstwach ( $S1$  i  $S2$ ) współczynnik uczenia został ustawiony na wartość domyślną, czyli 0.1, dlatego też dla tak dużego  $lr$  dobrana została niewielka ilość neuronów, tzn. 10 na pierwszej warstwie, oraz 9 na drugiej (ilość neuronów na warstwie pierwszej  $S1$  musi być większa od ilości neuronów na warstwie drugiej  $S2$ ). Stworzony w programie Matlab skrypt pozwala dziesięciokrotnie nauczyć sieć metodą wstecznej propagacji [3] dla każdego podziału danych. Ilość danych uczących zmienia się w zakresie od 10% do 90% całości zbioru, pozostała część to dane testujące - dzięki temu otrzymany średni wynik pozwala mieć większą pewność otrzymanych wyników. 150 danych znajdujących się w pliku jest podzielone po kolei klasami po 50, aby uwiarygodnić testy za każdym powtórzeniem dane są losowane dla każdej klasy indeksy, dla których określone dane z tablicy będą testowe, lub uczące - za pomocą funkcji `crossvalind` z argumentem `Holdout`[2]. Taka metoda pozwala uniknąć sytuacji w której powtarzane kolejne eksperymenty uczą się za każdym razem na tych samych danych i testują na tych samych danych, co poprzednio, oraz sytuacji w której losuje się różna ilość danych z różnych klas, przez co klasy, z których wylosowałoby się niewiele danych do uczenia mogłyby być niedouczona w jednym eksperymencie, a w kolejnym mieć bardzo dużo danych do uczenia i osiągnąć wysoką poprawność klasyfikacji. Taka sytuacja mogłaby spowodować, że otrzymane wyniki byłyby stochastyczne nie z powodu natury sieci, lecz z powodu doboru danych, co mogłoby doprowadzić do błędnych wniosków. W kodzie po wylosowaniu danych sieć się uczy, a poprawość klasyfikacji prezentowana jest na końcu za pomocą wykresu zależności poprawności klasyfikacji od podziału zbioru.



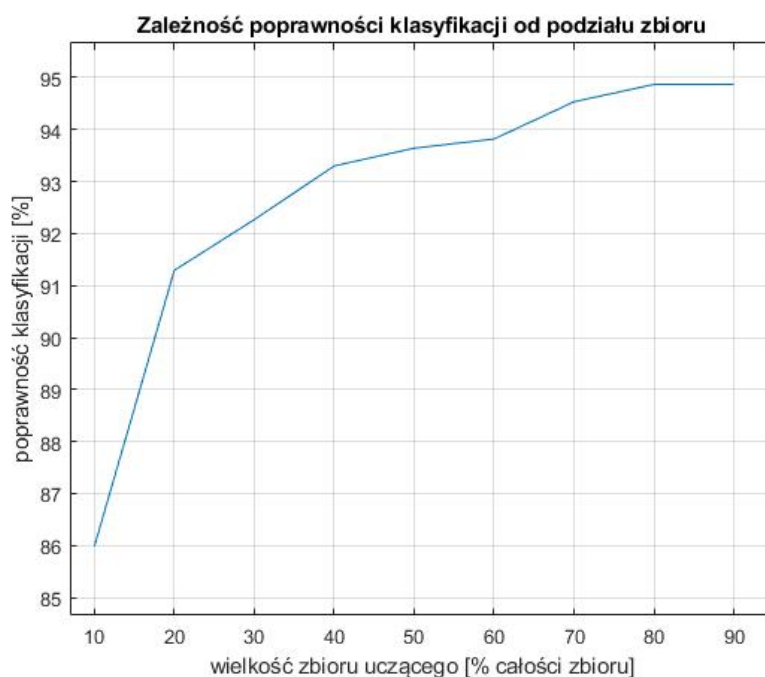
```

1 %wyczyszczenia środowiska i ustawienie wartości początkowych
2 close all; clear;
3 disp_freq = 100;
4 max_epoch = 40000;
5 max_fail = 10000;
6 load iris %wczytanie zbioru
7 %początkowe parametry ustawione są domyślnie
8 S1=10; %ilość neuronów na pierwszej warstwie
9 S2=9; %ilość neuronów na drugiej warstwie
10 lr = 0.1; %współczynnik uczenia
11 averages = zeros([1,9]); %pusta tablica dla zmiennych
12 for tries = 1:1:10 %dla 10 powtórzeń
13     for x = 1:9 %zbiór podzielony jest na 9 sposobów
14         r = 1;
15         t = 1;
16         for o = 1:3 %podział zbioru na klasy
17             [traind] = crossvalind('Holdout', 50, x/10); %krosvalidacja danych
18             for k = 1:50 %w każdej klasie dane są podzielone na uczące i testujące
19                 if (traind(k) == 1) %jeżeli indeks jest równy jeden
20                     %dana jest zapisywana do zbioru testującego
21                     Ptest(:,t) = Pn(:,k+50*(o-1));
22                     Ttest(1,t) = T(1,k+50*(o-1));
23                     t = t + 1; %iterowanie indeksu
24                 else
25                     %w przeciwnym wypadku jest zapisywana do zbioru uczącego
26                     Plearn(:,r) = Pn(:,k+50*(o-1));
27                     Tlearn(1,r) = T(1, k+50*(o-1));
28                     r = r + 1; %iterowanie indeksu
29                 end
30             end
31         end
32         net = feedforwardnet([S1, S2], 'traingd'); %definicja perceptronu
33         net.trainParam.epochs = max_epoch; %maksymalna liczba epok
34         net.trainParam.goal = 0.25/length(Ptest); %cel wydajności
35         net.trainParam.lr = lr; %learning rate
36         net.trainParam.max_fail = max_fail; %maksymalna ilość błędów walidacji
37         net.trainParam.showWindow = false; %czy pokazać okno uczenia
38         net.divideParam.trainRatio=1; %ilość danych do uczenia
39         net.divideParam.valRatio=0; %ilość danych do walidacji
40         net.divideParam.testRatio=0; %ilość danych do testowania
41         [net,tr] = train(net,Plearn,Tlearn); %uczenie sieci neuronowej
42         u = net(Ptest); %zapisanie do u tablicy otrzymanych wyjść
43         result = (1-sum(abs(Ttest-u)>=0.5)/length(Ttest))*100; %zapisanie PK
44         averages(x) = averages(x)+result; %dodawanie PK dla kolejnych powtórzeń
45         clear Plearn; clear Ptest; clear Tlearn; clear Ttest; %zresetowanie tablic
46     end
47 end
48 averages = averages/10; %wyciągnięcie średniej PK
49 percents = [10, 20, 30, 40, 50, 60, 70, 80, 90]; %dane do osi
50 plot(percents, averages); %rysowanie wykresu
51 title('Zależność poprawności klasyfikacji od podziału zbioru');
52 xlabel('wielkość zbioru uczącego [% całości zbioru]');
53 ylabel('poprawność klasyfikacji [%]');
54 grid;

```

Listing 1: Skrypt do analizy dotyczącej wpływu podziału danych na uczenie sieci

Jak widać na zamieszczonych poniżej wykresach oraz danych odpowiedni do optymalnego działania sieci jest podział, w którym 80% to dane uczące, taki podział stosowany będzie w kolejnych eksperymentach.



Rysunek 3.5: Wykres zależności poprawności klasyfikacji od podziału zbioru - Pk (zbiór uczący w %)

averages										
1x9 double										
	1	2	3	4	5	6	7	8	9	10
1	8.5985e+03	9.1292e+03	9.2267e+03	9.3300e+03	9.3640e+03	9.3817e+03	9.4533e+03	9.4867e+03	9.4867e+03	
2										

Rysunek 3.6: Dane do wykresu z rysunku 3.5 powyżej

## 3.2. Eksperyment 2 - ilość neuronów na poszczególnych warstwach

Kod w tym eksperymencie jest podobny do poprzedniego. Wykorzystany jest już tutaj wyznaczony wcześniej podział danych. Dla każdej wartości  $lr = 0.1, 0.01, 0.001, 0.0001$  oraz  $0.00001$  są w 10 powtórzeniach obliczane poprawności klasyfikacji i SSE (Sum Square Errors - suma kwadratów błędów) dla różnych ilości neuronów na pierwszej i drugiej warstwie, pozostałe wartości w przechowywanej wyniki tablicy dwuwymiarowej będące niezgodne z założeniem  $S1 > S2$  zostają zastąpione wartościami NaN, aby zapobiec ewentualnej nieczytelności wykresu.

```
1 %wyczyszczenia środowiska i ustawienie wartości początkowych
2 close all
3 clear
4 disp_freq = 100;
5 max_epoch = 40000;
6 max_fail = 10000;
7 load iris %wczytanie zbioru
8 %utworzenie potrzebnych tablic na dane i wartości
9 Ptest = zeros([4,30]);
10 Plearn = zeros([4,120]);
11 Ttest = zeros([1,30]);
12 Tlearn = zeros([1,120]);
13 results = zeros([11, 11]);
14 errors = zeros([11, 11]);
15
16 err_goal = 0.25/length(Plearn); %obliczenie pożądanej wartości błędu
17 j = 0; %j to zmienna odpowiadająca za pokazywanie progresu uczenia
18 lr = 1e-1; %współczynnik uczenia, zmienian dla kolejnych eksperymentów
19
20 for tries = 1:10 %dla 10 powtórzeń
21     r = 1;
22     t = 1;
23     for o = 1:3 %podział zbioru na klasy
24         [traind] = crossvalind('Holdout', 50, 0.8); %krosvalidacja danych
25         for k = 1:50 %w każdej klasie dane są podzielone na uczące i testujące
26             if (traind(k) == 1) %jeżeli indeks jest równy jeden
27                 %dana jest zapisywana do zbioru testującego
28                 Ptest(:,t) = Pn(:,k+50*(o-1));
29                 Ttest(1,t) = T(1,k+50*(o-1));
30                 t = t + 1; %iterowanie indeksu
31             else
32                 %w przeciwnym wypadku jest zapisywana do zbioru uczącego
33                 Plearn(:,r) = Pn(:,k+50*(o-1));
34                 Tlearn(1,r) = T(1, k+50*(o-1));
35                 r = r + 1; %iterowanie indeksu
36             end
37         end
38     end
39     for S1_vec = 1:10:101 %iteracja po pierwszej warstwie neuronów
40         for S2_vec = 1:10:S1_vec %iteracja po drugiej warstwie neuronów
41             j = j + 1; %zwiększanie progresu uczenia
42             net = feedforwardnet([S1_vec, S2_vec], 'traingd'); %definicja
perceptron
```

```

43     net.trainParam.epochs = max_epoch; %maksymalna liczba epok
44     net.trainParam.goal = err_goal; %cel wydajności
45     net.trainParam.lr = lr; %learning rate
46     net.trainParam.max_fail = max_fail; %maksymalna ilość błędów walidacji
47     net.trainParam.showWindow = false; %czy pokazać okno uczenia
48     net.divideParam.trainRatio=1; %ilość danych do uczenia
49     net.divideParam.valRatio=0; %ilość danych do walidacji
50     net.divideParam.testRatio=0; %ilość danych do testowania
51     [net,tr] = train(net,Plearn,Tlearn); %uczenie sieci neuronowej
52     u = net(Ptest); %zapisanie do u tablicy otrzymanych wyjść
53     error = sse(net, Ttest, u); %obliczenie SSE
54     pk = (1-sum(abs(Ttest-u)>=0.5)/length(Ttest))*100; %obliczenie PK
55     results((S1_vec+9)/10, (S2_vec+9)/10) = results((S1_vec+9)/10, (S2_vec
+9)/10) + pk; %zapisanie do result PK
56     errors((S1_vec+9)/10, (S2_vec+9)/10) = errors((S1_vec+9)/10, (S2_vec+9
/10) + error; %zapisanie do errors SSE
57     process = j/6.6;%obliczenie progresu
58     fprintf('progress: %.2f%%\n', process);%wyświetlenie progresu
59     end
60 end
61 end
62
63 %wypełnienie wartościami 'NaN' pustych miejsc w tablicy (bo S2<S1)
64 for S2_vec = 1:10:101
65     for S1_vec = 1:10:S2_vec
66         results((S1_vec+9)/10, (S2_vec+9)/10) = NaN;
67         errors((S1_vec+9)/10, (S2_vec+9)/10) = NaN;
68     end
69 end
70
71 %obliczenie średnich wyników
72 results = results / 10;
73 errors = errors / 10;
74 SL1 = [1, 11, 21, 31, 41, 51, 61, 71, 81, 91, 101];
75 SL2 = [1, 11, 21, 31, 41, 51, 61, 71, 81, 91, 101];
76
77 %wrysowanie wykresu pk
78 figure(1);
79 surf(SL1, SL2, results');
80 title('Zależność PK od ilości neuronów w poszczególnych warstwach');
81 xlabel('liczba neuronów w warstwie 1');
82 ylabel('liczba neuronów w warstwie 2');
83 zlabel('poprawność klasyfikacji [%]');
84 grid;
85
86 %wrysowanie wykresu sse
87 figure(2);
88 surf(SL1, SL2, errors');
89 title('Zależność błędu SSE od ilości neuronów w poszczególnych warstwach');
90 xlabel('liczba neuronów w warstwie 1');
91 ylabel('liczba neuronów w warstwie 2');
92 zlabel('SSE');
93 grid;

```

Listing 2: Skrypt do analizy dotyczącej wpływu ilości neuronów w obu warstwach na uczenie sieci

### 3.2.1. Dla współczynnika uczenia $lr = 0.1$

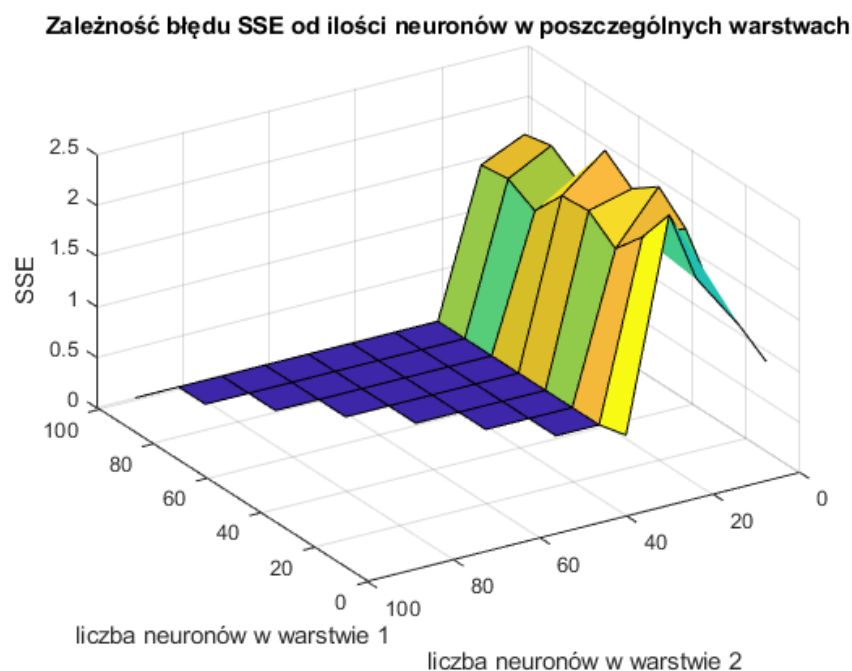
Dla najwyższego współczynnika uczenia udało się znaleźć dużą ilość par parametrów S1 oraz S2, które posiadały poprawność klasyfikacji na poziomie 100%, a sumę kwadratów błędów na poziomie 0. Najmniejsza ilość neuronów, która pozwalała osiągnąć taki wynik, to  $S1 = 41$  i  $S2 = 21$ . Większa ilość neuronów mogłaby przeuczać sieć, ponadto sieć mogłaby się uczyć dłużej, a już te wartości dają zadowalające wyniki, dlatego też będą podstawą do późniejszych rozważań, co zostanie jeszcze opisane po kolejnych eksperymentach podczas zagęszczonych poszukiwań 3.2.6. Minimalne PK jest tutaj równe 94%, a maksymalne SSE 2.1364.



Rysunek 3.7: Wykres zależności poprawności klasyfikacji od ilości neuronów w poszczególnych warstwach dla współczynnika uczenia równego  $1e-1$

	1	2	3	4	5	6	7	8	9	10	11
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	96.3333	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	95.3333	94.3333	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	95.3333	94.6667	99.6667	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	94.3333	95.6667	100	100	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6	94.6667	95.6667	100	100	100	NaN	NaN	NaN	NaN	NaN	NaN
7	95.3333	95	100	100	100	100	NaN	NaN	NaN	NaN	NaN
8	94	95.3333	100	100	100	100	100	NaN	NaN	NaN	NaN
9	94.3333	96.3333	100	100	100	100	100	100	NaN	NaN	NaN
10	94	94.6667	100	100	100	100	100	100	100	NaN	NaN
11	94.3333	95.6667	100	100	100	100	100	100	100	100	NaN

Rysunek 3.8: Dane do wykresu z rysunku 3.7 powyżej



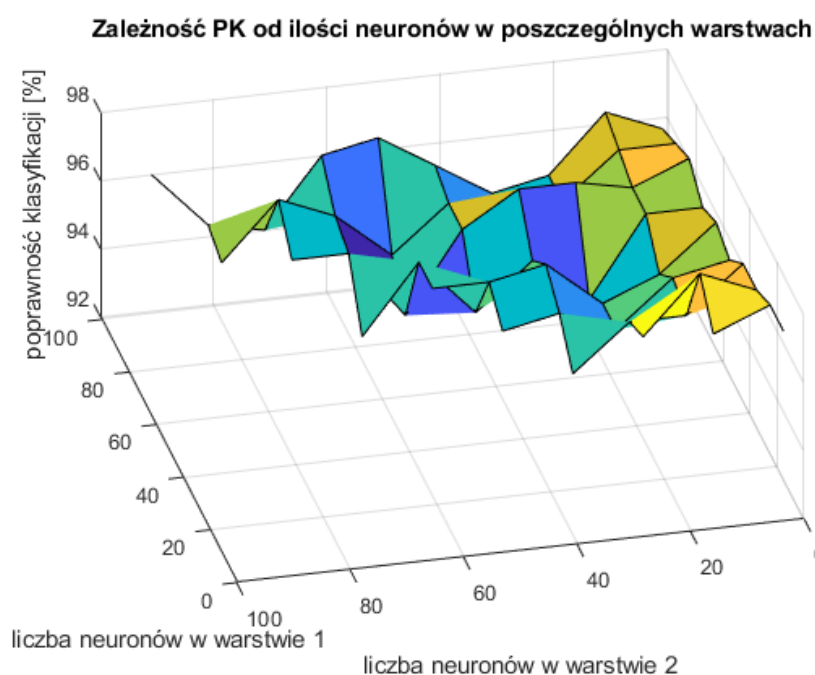
Rysunek 3.9: Wykres zależności sumy kwadratów błędów od ilości neuronów w poszczególnych warstwach dla współczynnika uczenia równego  $1e-1$

errors results											
11x11 double											
	1	2	3	4	5	6	7	8	9	10	11
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	0.9202	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	1.1062	1.6794	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	1.2665	2.1364	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	1.7228	1.7371	0	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6	1.9662	1.4611	0	0	0	NaN	NaN	NaN	NaN	NaN	NaN
7	1.7757	1.6655	0	0	0	0	NaN	NaN	NaN	NaN	NaN
8	1.9836	1.6433	0	0	0	0	0	NaN	NaN	NaN	NaN
9	1.5082	1.3244	0	0	0	0	0	0	NaN	NaN	NaN
10	1.6904	1.4694	0	0	0	0	0	0	0	NaN	NaN
11	1.6300	1.4365	0	0	0	0	0	0	0	0	NaN

Rysunek 3.10: Dane do wykresu z rysunku 3.9 powyżej

### 3.2.2. Dla współczynnika uczenia $lr = 0.01$

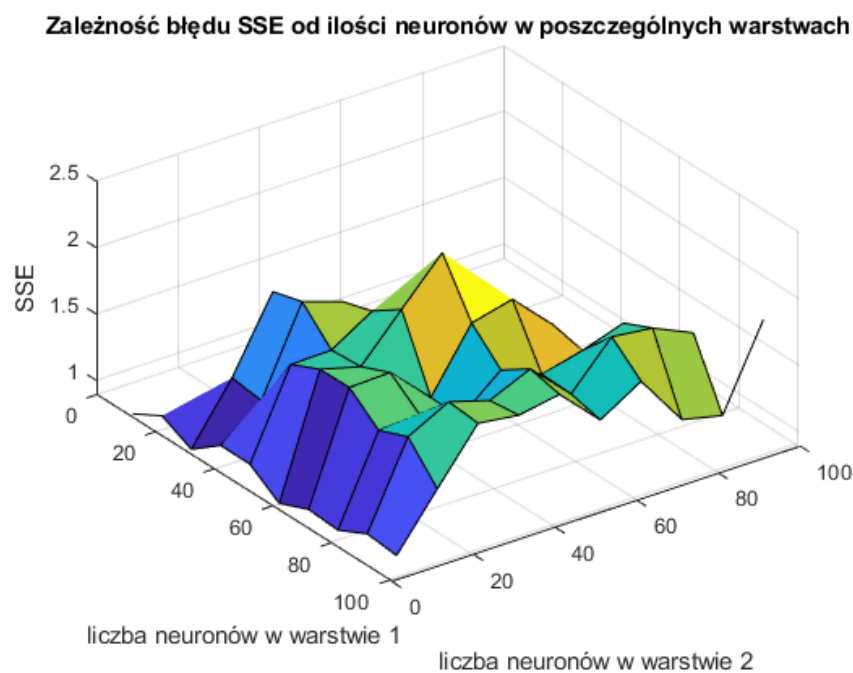
Dla mniejszego niż poprzednio współczynnika uczenia wykres traci swoje wcześniejsze właściwości, tzn. poprawność klasyfikacji nie rośnie już widocznie wraz ze wzrostem ilości neuronów, wyniki wydają się być bardziej stochastyczne w porównaniu do poprzednich, a największa poprawność klasyfikacji jest osiągana dla  $S1 = 31$ , oraz  $S2 = 11$ . Zdaje się, że większa losowość wyników pojawiają się od 31 neuronów na drugiej warstwie wzwyż. Suma kwadratów błędów mieści się w zakresie 0.0817 - 2.0794, natomiast poprawność klasyfikacji w przedziale 92.6667 %- 97%.



Rysunek 3.11: Wykres zależności poprawności klasyfikacji od ilości neuronów w poszczególnych warstwach dla współczynnika uczenia równego 1e-2

results											
11x11 double											
	1	2	3	4	5	6	7	8	9	10	11
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	96.6667	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	96.6667	96	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	96.3333	97.0000	95.3333	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	96.3333	95	95	93.6667	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6	95.6667	95.3333	94.0000	94.6667	94.3333	NaN	NaN	NaN	NaN	NaN	NaN
7	96	94.6667	94.0000	95.3333	95	95	NaN	NaN	NaN	NaN	NaN
8	95.6667	95.6667	93.3333	94.6667	93.3333	95	93.0000	NaN	NaN	NaN	NaN
9	96.3333	95.6667	96	96	95	92.6667	94.6667	94.6667	NaN	NaN	NaN
10	96.0000	96	94.6667	94.0000	95	93.6667	95.0000	95.6667	94	NaN	NaN
11	95.6667	96.3333	94.6667	94.3333	95.3333	96.3333	96	94	94.3333	96	NaN

Rysunek 3.12: Dane do wykresu z rysunku 3.11 powyżej



Rysunek 3.13: Wykres zależności sumy kwadratów błędów od ilości neuronów w poszczególnych warstwach dla współczynnika uczenia równego  $1e-2$

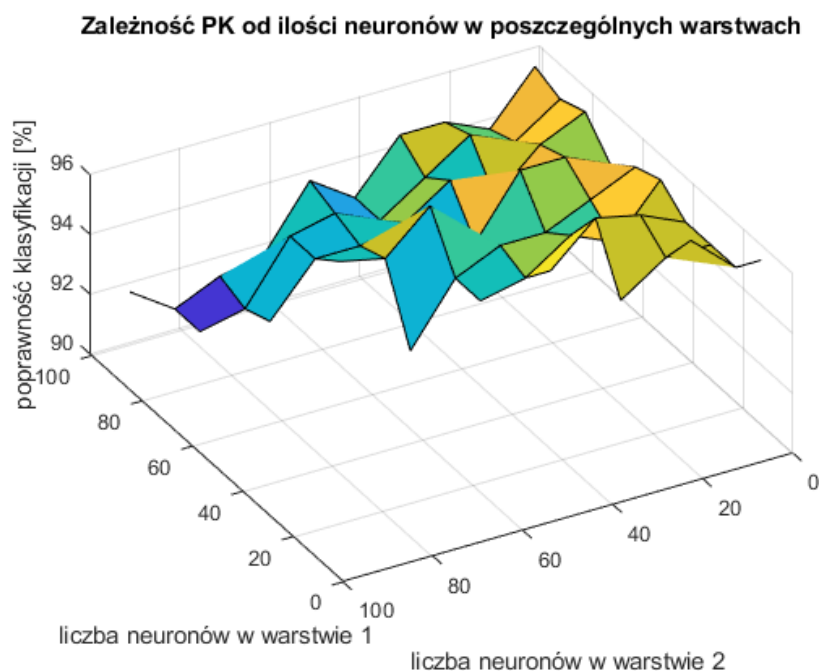
	1	2	3	4	5	6	7	8	9	10	11
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	0.8917	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	1.0160	0.9104	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	0.9031	1.3388	1.8867	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	1.0726	1.3075	1.9496	1.8472	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6	1.0704	1.7190	1.7252	1.9173	1.8876	NaN	NaN	NaN	NaN	NaN	NaN
7	0.9149	1.8222	1.7397	2.0689	2.3936	1.8340	NaN	NaN	NaN	NaN	NaN
8	1.0089	1.8251	1.8356	1.5442	2.0033	2.0794	1.7848	NaN	NaN	NaN	NaN
9	0.9924	1.6420	1.6599	1.5338	1.7891	1.6544	1.7313	1.8346	NaN	NaN	NaN
10	1.1037	1.7331	1.8940	1.8010	1.9370	1.6461	1.9741	1.9370	1.7990	NaN	NaN
11	1.0809	1.4798	1.8715	1.8301	1.8837	1.5894	1.7782	1.3874	1.3156	1.9354	NaN

Rysunek 3.14: Dane do wykresu z rysunku 3.13 powyżej



### 3.2.3. Dla współczynnika uczenia $lr = 0.001$

Po kolejnym zmniejszeniu współczynnika uczenia można zauważyć, że tym razem poprawność klasyfikacji maleje, a suma kwadratów błędów widocznie rośnie dla przy większej ilości neuronów - zwłaszcza na drugiej warstwie. Tym razem najlepsza poprawność klasyfikacji została otrzymana dla  $S1 = 21$ ,  $S2 = 11$ . Zakres poprawności klasyfikacji to 90.6667% - 96%, natomiast sumy kwadratów błędów 0.9743 - 2.6305.

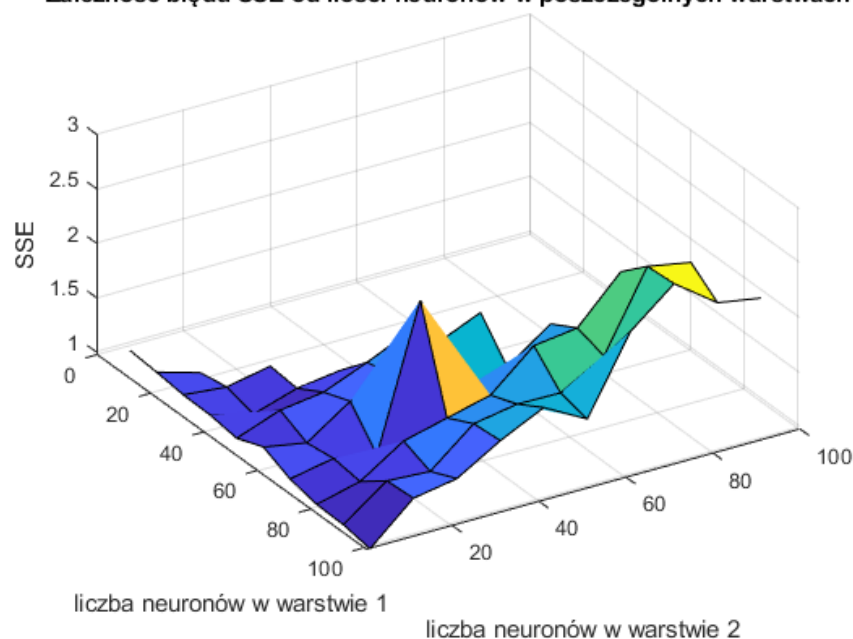


Rysunek 3.15: Wykres zależności poprawności klasyfikacji od ilości neuronów w poszczególnych warstwach dla współczynnika uczenia równego  $1e-3$

	1	2	3	4	5	6	7	8	9	10	11
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	95.6667	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	94.6667	96	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	94.6667	94.6667	93.6667	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	94.6667	95.3333	95.6667	94.3333	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6	95.3333	93.6667	94.3333	93.3333	93	NaN	NaN	NaN	NaN	NaN	NaN
7	95.0000	94.3333	93.6667	93.6667	93.0000	91	NaN	NaN	NaN	NaN	NaN
8	94.3333	95	95.0000	93.0000	94.6667	93.3333	93.6667	NaN	NaN	NaN	NaN
9	95.3333	94.6667	93.3333	94.3333	93.3333	93.0000	93	91.3333	NaN	NaN	NaN
10	95.0000	94	94.6667	93.6667	92.6667	93.3333	93	91.0000	90.6667	NaN	NaN
11	95.3333	93.6667	94.3333	94.3333	92.6667	93.6667	91.6667	91.3333	90.6667	91.6667	NaN

Rysunek 3.16: Dane do wykresu z rysunku 3.15 powyżej

Zależność błędu SSE od ilości neuronów w poszczególnych warstwach



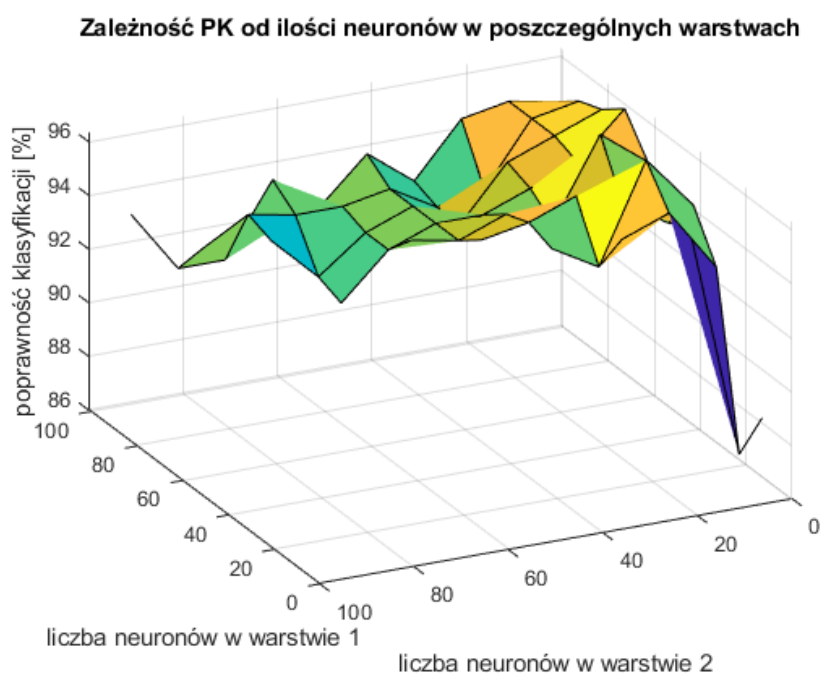
Rysunek 3.17: Wykres zależności sumy kwadratów błędów od ilości neuronów w poszczególnych warstwach dla współczynnika uczenia równego  $1e-3$

	1	2	3	4	5	6	7	8	9	10	11
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	1.1980	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	1.1758	1.0619	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	1.1458	1.1070	1.1875	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	1.1388	1.0353	1.1602	1.2147	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6	1.1022	1.2488	1.3033	1.4649	1.6467	NaN	NaN	NaN	NaN	NaN	NaN
7	1.1986	1.1262	1.3966	1.5281	1.7058	1.8792	NaN	NaN	NaN	NaN	NaN
8	1.0836	1.1602	1.0792	2.3904	1.4677	1.5548	1.8405	NaN	NaN	NaN	NaN
9	1.0267	1.1260	1.3864	1.4869	1.5849	1.9286	1.9802	2.3850	NaN	NaN	NaN
10	1.0176	1.3089	1.2849	1.6121	1.6805	1.6727	1.8999	2.6123	2.5377	NaN	NaN
11	0.9743	1.3274	1.3965	1.6339	1.8269	1.6094	2.3044	2.6305	2.3449	2.2763	NaN

Rysunek 3.18: Dane do wykresu z rysunku 3.17 powyżej

### 3.2.4. Dla współczynnika uczenia $lr = 0.0001$

Dla takiej wartości współczynnika uczenia widać, że wykresy przypominają poprzednie, nadal trend wzrostu PK i spadku SSE są podobne jak poprzednio, widać tutaj również, że dla bardzo małej ilości neuronów poprawność klasyfikacji spadała, a SSE wzrosła, jednak nie wydaje się to być istotne dla eksperymentu. Poprawność klasyfikacji mieści się w przedziale 86.3333% - 96.6667%, a suma kwadratów błędów 1.2877 - 4.1673.

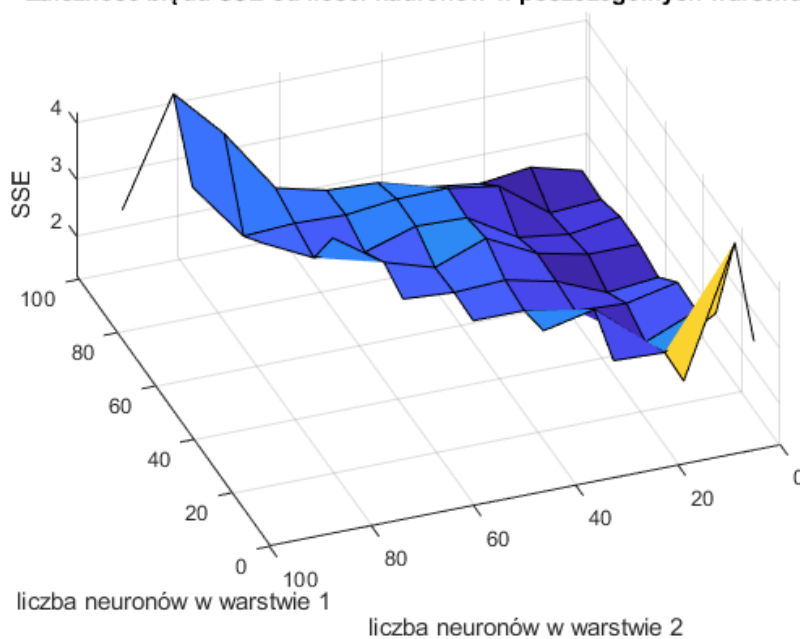


Rysunek 3.19: Wykres zależności poprawności klasyfikacji od ilości neuronów w poszczególnych warstwach dla współczynnika uczenia równego  $1e-4$

results errors											
11x11 double											
	1	2	3	4	5	6	7	8	9	10	11
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	88.3333	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	86.3333	94.6667	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	92.6667	95.0000	94.3333	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	94.3333	96.3333	92.6667	93.6667	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6	93.0000	94	94.6667	94	93.6667	NaN	NaN	NaN	NaN	NaN	NaN
7	94.6667	96.0000	93	93.6667	93.0000	93.3333	NaN	NaN	NaN	NaN	NaN
8	96.0000	94.3333	93.6667	93.0000	93	92.3333	90.6667	NaN	NaN	NaN	NaN
9	95.3333	94.6667	94	92	93	92.3333	91.0000	92.6667	NaN	NaN	NaN
10	95	94.6667	92.3333	92.0000	93	92.6667	92.6667	93	92	NaN	NaN
11	94.3333	94.6667	94.3333	92.3333	93.6667	92.0000	93.3333	90.6667	90.6667	93.0000	NaN

Rysunek 3.20: Dane do wykresu z rysunku 3.19 powyżej

Zależność błędu SSE od ilości neuronów w poszczególnych warstwach



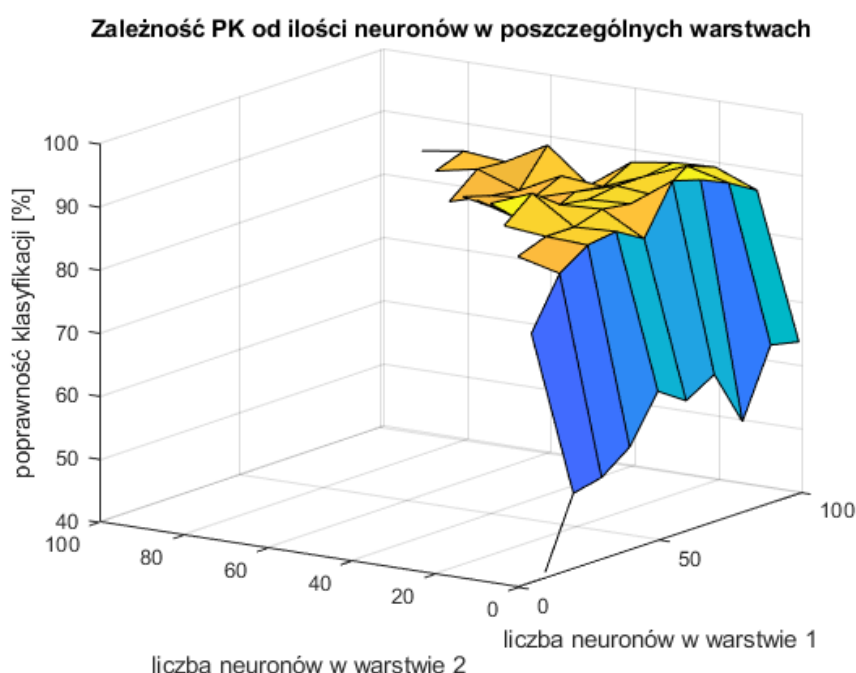
Rysunek 3.21: Wykres zależności sumy kwadratów błędów od ilości neuronów w poszczególnych warstwach dla współczynnika uczenia równego  $1e-4$

errors											
	1	2	3	4	5	6	7	8	9	10	11
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	2.6003	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	3.8724	1.6075	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	2.1494	1.6395	1.6759	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	1.7252	1.3555	2.1074	1.9105	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6	1.7577	1.5661	1.6115	1.7959	1.7996	NaN	NaN	NaN	NaN	NaN	NaN
7	1.3755	1.2877	1.6615	1.8664	1.8258	1.8892	NaN	NaN	NaN	NaN	NaN
8	1.4881	1.4900	1.6870	2.1312	1.7959	2.0669	2.6769	NaN	NaN	NaN	NaN
9	1.5029	1.3724	1.4994	2.0242	2.0533	1.7765	1.8557	2.2944	NaN	NaN	NaN
10	1.3350	1.2785	1.9302	2.0758	2.0615	1.9616	1.9941	1.9468	2.9911	NaN	NaN
11	1.3745	1.6128	1.5024	1.6253	1.8804	1.9262	2.1452	3.2791	4.1673	2.2962	NaN

Rysunek 3.22: Dane do wykresu z rysunku 3.21 powyżej

### 3.2.5. Dla współczynnika uczenia $\eta = 0.00001$

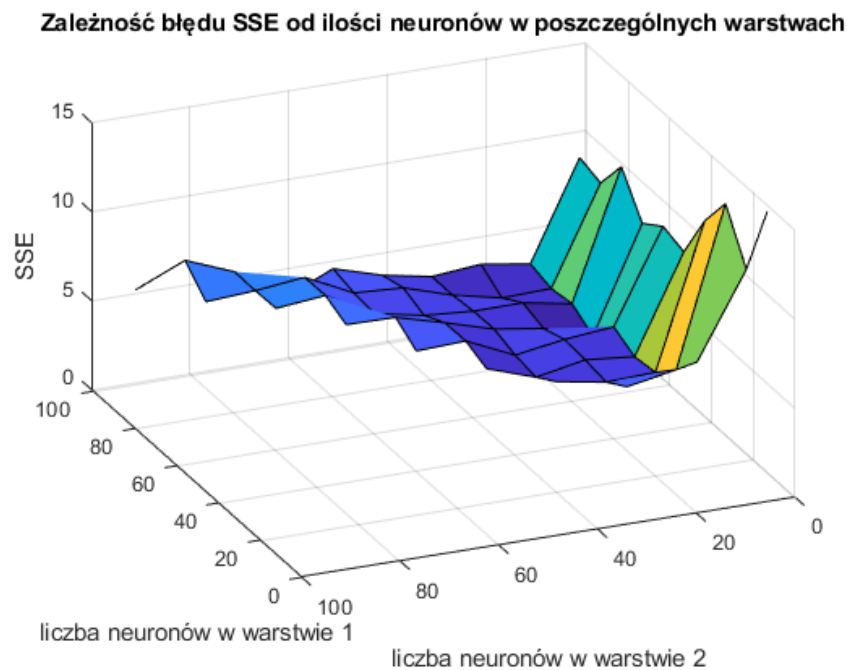
Dla najmniejszej wartości współczynnika uczenia można zauważyć, że zgodnie z oczekiwaniami wartości poprawności klasyfikacji są wyraźnie mniejsze, a SSE większe, niż wcześniej. Co więcej Pk jest najmniejsze, a SSE największe ze wszystkich eksperymentów. Nadal jednak poprawność klasyfikacji jest największa, a sumy kwadratów błędów najmniejsze dla ilości neuronów 11-31 w drugiej warstwie. Zakres Pk to 40.6667% - 92.6667%, natomiast SSE 2.1336 - 14.9306. W porównaniu do poprzednich te zakresy są spore, można więc uznać, że stochastyczność rośnie wraz z zmniejszaniem się współczynnika uczenia.



Rysunek 3.23: Wykres zależności poprawności klasyfikacji od ilości neuronów w poszczególnych warstwach dla współczynnika uczenia równego  $1e-5$

results											
errors											
11x11 double											
	1	2	3	4	5	6	7	8	9	10	11
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	40.6667	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	51.6667	76	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	52.6667	84	85.6667	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	56	87	87.3333	88	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6	63.3333	87.6667	87.3333	91.6667	89	NaN	NaN	NaN	NaN	NaN	NaN
7	60.3333	85	88.6667	88.3333	85	87	NaN	NaN	NaN	NaN	NaN
8	63	92.6667	88	87.6667	84	83.6667	85	NaN	NaN	NaN	NaN
9	54	91.3333	87.6667	88.3333	86.3333	84.3333	83.6667	81.6667	NaN	NaN	NaN
10	64.6667	89.3333	91	88.3333	85.6667	86.3333	83	85.3333	84	NaN	NaN
11	63.6667	86.6667	89.3333	89	88	83.3333	88.6667	85	85.6667	84.6667	NaN

Rysunek 3.24: Dane do wykresu z rysunku 3.23 powyżej



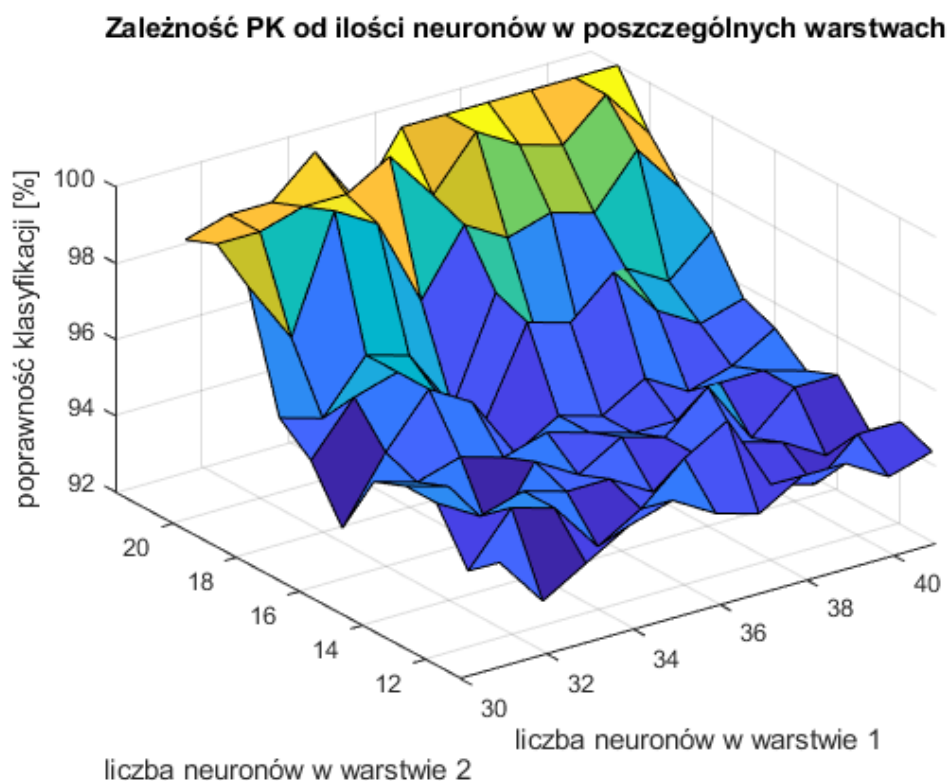
Rysunek 3.25: Wykres zależności sumy kwadratów błędów od ilości neuronów w poszczególnych warstwach dla współczynnika uczenia równego  $1e-5$

errors											
	1	2	3	4	5	6	7	8	9	10	11
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	14.9306	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	10.6456	5.8904	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	13.2944	4.4157	3.9093	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	11.2771	3.2258	3.1253	3.5966	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6	8.4975	3.0572	3.3346	2.8047	3.7399	NaN	NaN	NaN	NaN	NaN	NaN
7	8.8805	3.6954	3.4340	3.0018	4.2480	4.1308	NaN	NaN	NaN	NaN	NaN
8	8.0105	2.1336	3.0066	3.4203	4.2246	4.9791	5.0178	NaN	NaN	NaN	NaN
9	10.1411	2.8983	3.4001	3.3848	3.5948	4.1439	5.6130	5.3418	NaN	NaN	NaN
10	8.1844	2.7100	2.6314	3.2372	4.0990	4.2621	5.5879	5.2863	5.1178	NaN	NaN
11	8.5638	3.0448	3.4745	3.2349	3.7605	4.5779	3.3383	5.3081	6.3927	5.1949	NaN

Rysunek 3.26: Dane do wykresu z rysunku 3.25 powyżej

### 3.2.6. Zagęszczone poszukiwania

Przy ogólnym poszukiwaniu 3.2.1 najbardziej zadowalające wartości poprawności i błędu zostały osiągnięte przy współczynniku  $lr = 0.1$  i ilości neuronów na pierwszej oraz drugiej warstwie:  $S1 = 41$ ,  $S2 = 21$ . Kolejny eksperyment jest rozszerzeniem wspomnianego badania, tym razem zostaje jednak zawężony zakres poszukiwań, z zakresu  $S1 = 11, 21, \dots, 101$  do  $S1 = 31, 32, \dots, 41$  oraz z  $S2 = 1, 11, \dots, S1$  do  $S2 = 11, 12, \dots, 21$ . Taki zakres poszukiwań pozwala na znalezienie mniejszej wartości neuronów, dla których sieć osiąga minimalny błąd, dzięki czemu można zredukować ryzyko przeuczenia, oraz zmniejszyć czas uczenia sieci. Kod jest tutaj niemalże identyczny, jak wcześniej, zmienione są jedynie ilości neuronów oraz tablice do przechowywania wyników. Okazuje się, że udało się znaleźć zadowalające kombinacje liczb neuronów w obu warstwach. Z otrzymanych wyników można odczytać, że potrzebne jest 21 neuronów w drugiej warstwie, a na pierwszej wystarczy ich 36. Dodatkowo można zauważyć, że poprawność klasyfikacji rośnie, a suma kwadratów błędów maleje dla kolejnych wartości  $S2$  szybciej, niż dla kolejnych rosnących wartości  $S1$ . Same wartości PK nie spadają poniżej 93.3333%, a SSE nie osiąga wartości większej, niż 2.8768.

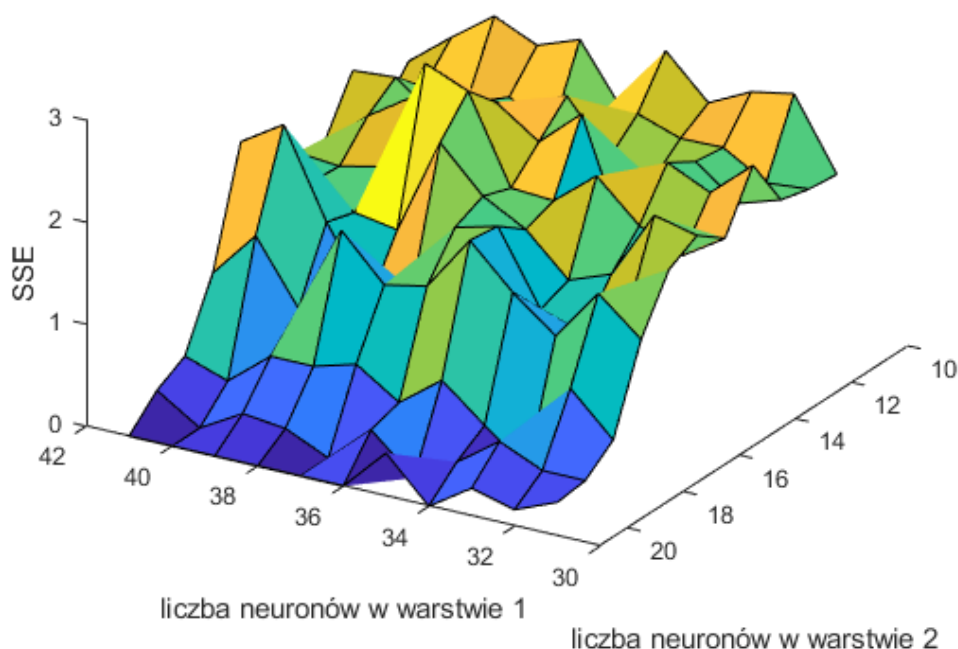


Rysunek 3.27: Wykres zależności poprawności klasyfikacji od ilości neuronów w poszczególnych warstwach dla współczynnika uczenia równego  $1e-1$  przy zagęszczonych poszukiwaniach

errors results											
11x11 double											
	1	2	3	4	5	6	7	8	9	10	11
1	94.6667	94.0000	95.3333	95.3333	95.0000	93.3333	94.6667	95.3333	98.3333	99.0000	98.6667
2	93.3333	95.3333	94.6667	95.0000	94.6667	95	96	95	96.6667	99	99
3	94.0000	95	95	93.3333	95	96.3333	96.3333	96.3333	99.6667	99.3333	99.0000
4	94.6667	93.6667	94.6667	95.0000	94.6667	95.6667	95.3333	96.0000	99	99.3333	100
5	95.0000	94.3333	94.6667	94.3333	95	94	94.3333	94.3333	96.6667	100	98.3333
6	94.3333	95.3333	94	95.0000	94.6667	94.6667	96.6667	97.0000	98.3333	98.6667	100
7	94	96	94.6667	94.6667	94.3333	94.3333	96.3333	95.3333	97.6667	100	100
8	94.6667	94	94.6667	95.6667	94.3333	95	97.3333	95.0000	98.0000	99.3333	100
9	94.6667	93.6667	94.3333	94	95.0000	94.3333	96.0000	96.6667	97.6667	99	100
10	94	94.6667	93.6667	95	95.0000	94.6667	95.3333	96	99	100	100
11	94.3333	94.6667	94	95.0000	94.6667	95.3333	95.6667	97.0000	97.6667	98.6667	100

Rysunek 3.28: Dane do wykresu z rysunku 3.27 powyżej

### Zależność błędu SSE od ilości neuronów w poszczególnych warstwach



Rysunek 3.29: Wykres zależności sumy kwadratów błędów od ilości neuronów w poszczególnych warstwach dla współczynnika uczenia równego  $1e-1$  przy zagęszczonych poszukiwaniach

errors results											
11x11 double											
	1	2	3	4	5	6	7	8	9	10	11
1	1.7603	1.7945	1.8671	2.3842	1.8474	2.0542	1.9005	1.3924	0.5859	0.3662	0.3320
2	2.4517	1.7570	1.7340	2.0193	1.5787	2.1532	1.3339	1.7456	0.9937	0.3674	0.1617
3	2.3731	2.0057	2.0466	2.2148	1.7753	1.4788	1.0792	1.2304	0.1115	0.5533	0.2741
4	2.1673	1.9971	1.7629	1.3527	2.1652	1.3230	1.2158	1.5607	0.4349	0.2556	0
5	2.5795	2.1401	1.8920	2.4974	1.8246	1.7220	1.9314	1.9706	0.7785	0	0.4230
6	1.8941	1.9944	2.3972	2.1245	1.8508	1.9495	1.8217	1.4252	0.4639	0.3741	0
7	2.4928	1.7656	2.2252	2.4152	2.7029	2.3756	1.1025	1.3397	0.7399	0	0
8	2.3391	2.0104	2.1414	1.7031	2.8768	1.4239	0.9064	1.7756	0.6223	0.1913	0
9	2.5268	1.8488	1.8643	2.2639	1.7681	1.4956	1.5811	0.9110	0.6223	0.2388	0
10	2.2217	1.9140	2.1864	1.9855	1.6723	1.7014	2.4377	1.5265	0.2827	0	0
11	1.8884	1.5343	2.1567	1.6378	1.7105	1.5617	2.1720	1.0641	0.4279	0.2703	0

Rysunek 3.30: Dane do wykresu z rysunku 3.29 powyżej



### 3.3. Eksperyment 3 - wielkość parametru współczynnika uczenia

W tym eksperymencie badany jest wpływ współczynnika uczenia na poprawność klasyfikacji oraz sumę kwadratów błędów. Współczynnik mieści się w takim samym zakresie, jak wcześniej:  $lr = [1e-5, 1e-4, 1e-3, 1e-2, 1e-1]$ . Aby otrzymać ogólny obraz wpływu współczynnika uczenia badany jest w pierwszym z eksperymentów tego dotyczącym jak zmienia się  $P_k$  i  $SSE$  w zależności od  $lr$ , jeżeli wynik dla danego  $lr$  jest brany dla każdej możliwej pary  $S1$  i  $S2$ , gdzie  $S1 = 11, 21, \dots, 101$ ; a  $S2 = 1, 11, \dots, S1$ . Drugi z eksperymentów pokazuje jak zmieniają się wyniki w zależności od współczynnika uczenia dla wyznaczonej w zagęszczonych poszukiwaniach pary liczb  $S1$  i  $S2$ .

#### 3.3.1. Współczynnik uczenia przy średniej ze wszystkich kombinacji ilości neuronów

Zostały tutaj wykorzystane tablice z wynikami eksperymentów, w których poszukiwane były wartości  $S1, S2$  dla różnych  $lr$ . Z eksperymentu, gdzie  $lr = 1e-1$  wzięte są tablice  $results1$  oraz  $errors1$ , dla  $lr = 1e-2$  tablice  $results2$  i  $errors2$ , itd. Następnie wartości z tablic sumowane tak, aby  $result1$  posiadało sumę poprawności klasyfikacji wszystkich par  $S1$  i  $S2$  dla  $lr=1e-1$ , podobnie  $error1$  sumę  $SSE$  i tak dalej dla każdej tablicy. Po zsumowaniu obliczona zostaje średnia i wyrysowane są wykresy.

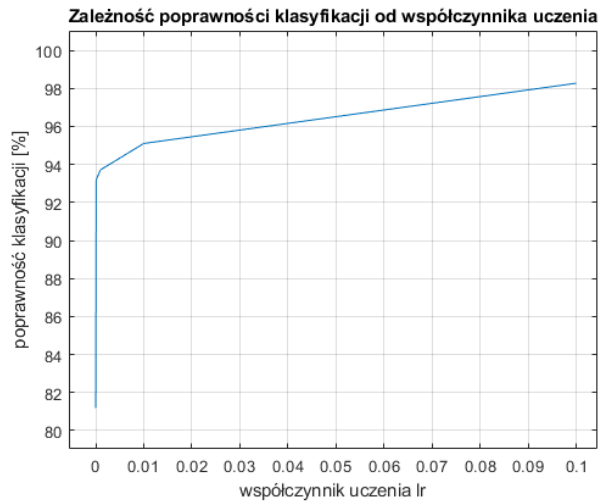
```

1  for x = 2:1:11 %dla indeksów odpowiadających S1
2      for y = 1:1:x-1 %dla indeksów odpowiadających S2
3          %sumowane są kolejne wartości Pk i SSE dla każdego lr
4              result1 = result1 + results1(x, y);
5              result2 = result2 + results2(x, y);
6              result3 = result3 + results3(x, y);
7              result4 = result4 + results4(x, y);
8              result5 = result5 + results5(x, y);
9              error1 = error1 + errors1(x, y);
10             error2 = error2 + errors2(x, y);
11             error3 = error3 + errors3(x, y);
12             error4 = error4 + errors4(x, y);
13             error5 = error5 + errors5(x, y);
14         end
15     end
16     %obliczenie średniej
17     result1 = result1 / 55;
18     result2 = result2 / 55;
19     result3 = result3 / 55;
20     result4 = result4 / 55;
21     result5 = result5 / 55;
22     error1 = error1 / 55;
23     error2 = error2 / 55;
24     error3 = error3 / 55;
25     error4 = error4 / 55;
26     error5 = error5 / 55;
27     %tablice potrzebne do osi
28     range = [1e-5, 1e-4, 1e-3, 1e-2, 1e-1];
29     results = [result5, result4, result3, result2, result1];
30     errors = [error5, error4, error3, error2, error1];
31     %wyrysowanie wykresów
32     figure(1);
33     plot(range,results);
34     title('Zależność poprawności klasyfikacji od współczynnika uczenia');
35     xlabel('współczynnik uczenia lr');
36     ylabel('poprawność klasyfikacji [%]');
37     grid;
38
39     figure(2);
40     plot(range,errors);
41     title('Zależność błędu SSE od współczynnika uczenia');
42     xlabel('współczynnik uczenia lr');
43     ylabel('SSE');
44     grid;

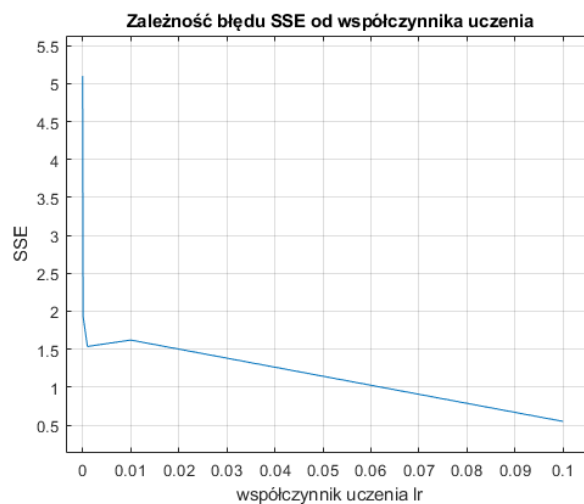
```

Listing 3: Skrypt do analizy dotyczącej wpływu wartości współczynnika uczenia na uczenie sieci

Z otrzymanych wyników można wywnioskować, że w tym przypadku wraz ze wzrostem współczynnika uczenia rośnie poprawność klasyfikacji, oraz maleje suma kwadratów błędów. Dla kolejnych coraz mniejszych wartości  $lr$  poprawność klasyfikacji coraz gwałtowniej maleje. Dla największej wartości  $lr = 0.1$  poprawność klasyfikacji nie osiąga 100%, a SSE 0.



Rysunek 3.31: Wykres zależności poprawności klasyfikacji od współczynnika uczenia dla średnich ze wszystkich kombinacji ilości neuronów



Rysunek 3.32: Wykres zależności sumy kwadratów błędów od współczynnika uczenia dla średnich ze wszystkich kombinacji ilości neuronów

### 3.3.2. Współczynnik uczenia dla optymalnej ilości neuronów

Zbadany również został wpływ współczynnika uczenia na poprawność klasyfikacji oraz sumę kwadratów błędów przy optymalnie dobranych ilościach neuronów na warstwie pierwszej i drugiej:  $S1 = 36$ ,  $S2 = 21$ . Działanie skryptu nie wymaga tłumaczenia, ponieważ bazuje na poprzednich umieszczonych w raporcie kodach - różni się jedynie wartościami w niektórych miejscach, ale zasada działania jest identyczna.

```
1 %wyczyszczenia środowiska i ustawienie wartości początkowych
2 close all
3 clear
4 disp_freq = 100;
5 max_epoch = 40000;
6 max_fail = 10000;
7 load iris
8 %ustawienie optymalnej ilości neuronów
9 S1=36;
10 S2=21;
11 %stworzenie potrzebnych tablic
12 Ptest = zeros([4,30]);
13 Plearn = zeros([4,120]);
14 Ttest = zeros([1,30]);
15 Tlearn = zeros([1,120]);
16 results = zeros([1,5]);
17 errors = zeros([1,5]);
18 err_goal = 0.25/length(Plearn);%błąd docelowy
19 j = 0;%j to zmienna odpowiadająca za pokazywanie progresu uczenia
20
21 for tries = 1:10 %dla 10 powtórzeń
22     i = 0; %zmienne iteracyjne do zapisu wyników w tablicach
23     r = 1;
24     t = 1;
25     for o = 1:3 %podział zbioru na klasy
26         [traind] = crossvalind('Holdout', 50, 0.8); %krosvalidacja danych
27         for k = 1:50 %w każdej klasie dane są podzielone na uczące i testujące
28             if (traind(k) == 1) %jeżeli indeks jest równy jeden
29                 %dana jest zapisywana do zbioru testującego
30                 Ptest(:,t) = Pn(:,k+50*(o-1));
31                 Ttest(1,t) = T(1,k+50*(o-1));
32                 t = t + 1; %iterowanie indeksu
33             else
34                 %w przeciwnym wypadku jest zapisywana do zbioru uczącego
35                 Plearn(:,r) = Pn(:,k+50*(o-1));
36                 Tlearn(1,r) = T(1, k+50*(o-1));
37                 r = r + 1; %iterowanie indeksu
38             end
39         end
40     end
41     for lr = [1e-5, 1e-4, 1e-3, 1e-2, 1e-1]
42         j = j + 1; %zwiększanie progresu uczenia
43         i = i + 1; %zwiększenie zmiennej iteracyjnej
44         net = feedforwardnet([S1, S2], 'traingd'); %definicja perceptron
45         net.trainParam.epochs = max_epoch; %maksymalna liczba epok
46         net.trainParam.goal = err_goal; %cel wydajności
47         net.trainParam.lr = lr; %learning rate
```

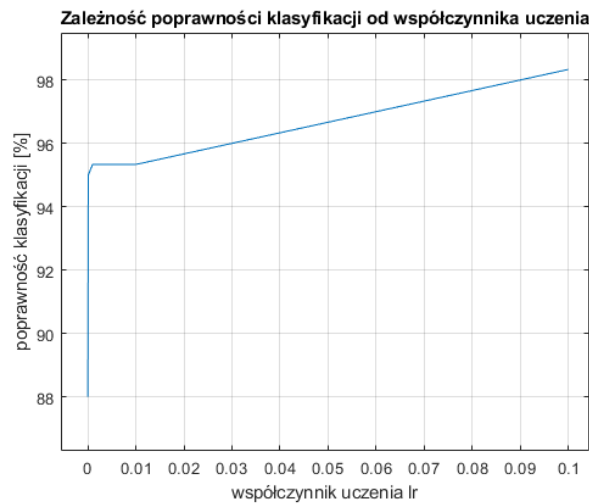
```

48     net.trainParam.max_fail = max_fail; %maksymalna ilość błędów walidacji
49     net.trainParam.showWindow = false; %czy pokazać okno uczenia
50     net.divideParam.trainRatio=1; %ilość danych do uczenia
51     net.divideParam.valRatio=0; %ilość danych do walidacji
52     net.divideParam.testRatio=0; %ilość danych do testowania
53     [net,tr] = train(net,Plearn,Tlearn); %uczenie sieci neuronowej
54     u = net(Ptest); %zapisanie do u tablicy otrzymanych wyjść
55     errors(i) = errors(i) + sse(net, Ttest, u); %zapisanie do errors SSE
56     results(i) = results(i) + (1-sum(abs(Ttest-u)>=0.5)/length(Ttest))*100; %
    zapisanie do result PK
57     process = j/0.5; %obliczenie progresu
58     fprintf('progress: %.2f%%\n', process); %wyświetlenie progresu
59 end
60 end
61
62 %obliczenie średnich wyników
63 results = results / 10;
64 errors = errors / 10;
65 %tablica do osi
66 LRrange = [1e-5, 1e-4, 1e-3, 1e-2, 1e-1];
67
68 %wyrysowanie wykresów
69 figure(1);
70 plot(LRrange,results);
71 title('Zależność poprawności klasyfikacji od współczynnika uczenia');
72 xlabel('współczynnik uczenia lr');
73 ylabel('poprawność klasyfikacji [%]');
74 grid;
75
76 figure(2);
77 plot(LRrange,errors);
78 title('Zależność błędu SSE od współczynnika uczenia');
79 xlabel('współczynnik uczenia lr');
80 ylabel('SSE');
81 grid;

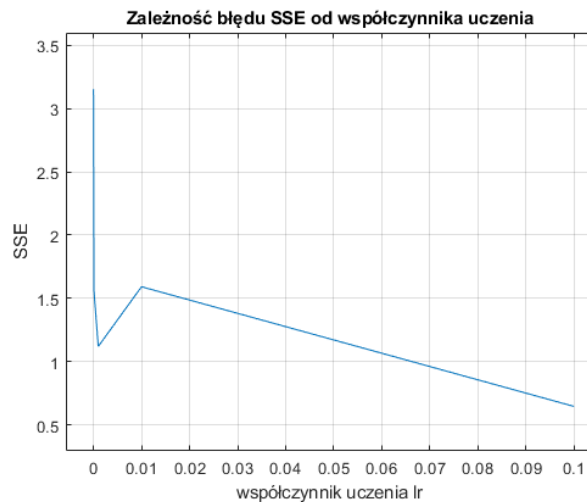
```

Listing 4: Skrypt do analizy dotyczącej wpływu wartości współczynnika uczenia przy ustalonej ilości neuronów na uczenie sieci

Otrzymane wykresy mają bardzo podobny charakter do poprzednich, potwierdzają więc, że dla tej sieci zwiększając poprawność klasyfikacji uzyskuje się lepsze wyniki. Jediną różnicą jest tutaj to, że dla  $lr = 1e-2$  SSE nie jest mniejsze, niż dla  $lr = 1e-3$ , ale jest to nieistotne dla badania, i nie wpływa na ostateczny kształt wniosków wypływających z tych dwóch eksperymentów.



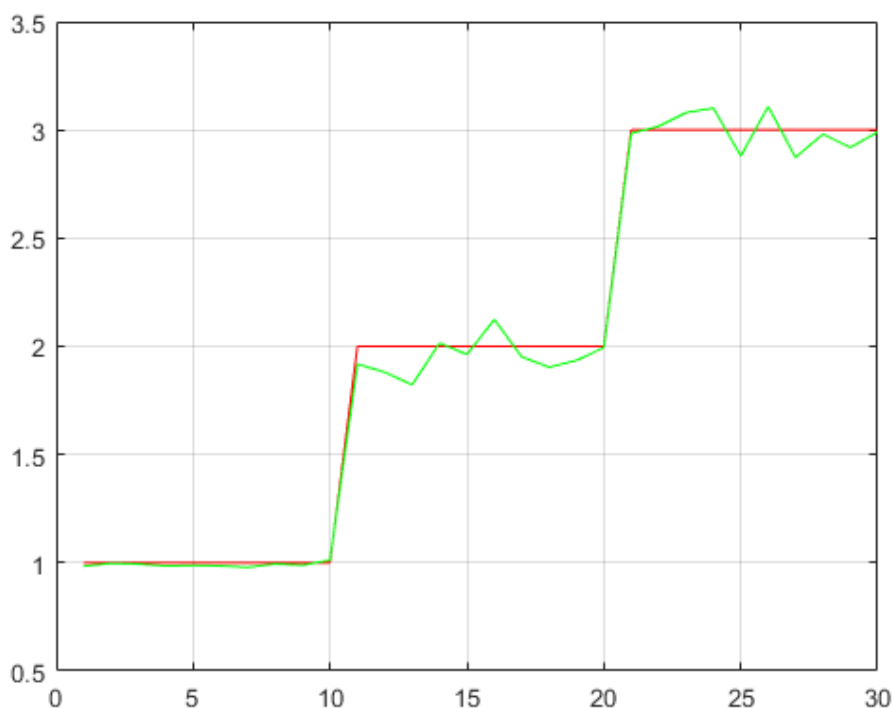
Rysunek 3.33: Wykres zależności poprawności klasyfikacji od współczynnika uczenia dla dobranych wcześniej 36 neuronów na pierwszej i 21 na drugiej warstwie



Rysunek 3.34: Wykres zależności sumy kwadratów błędów od współczynnika uczenia dla dobranych wcześniej 36 neuronów na pierwszej i 21 na drugiej warstwie

## 4. Wnioski

Dzięki znormalizowaniu danych i przeprowadzeniu serii eksperymentów udało się znaleźć pary liczb określających ilość neuronów na dwóch warstwach sieci, dla których wartości poprawności klasyfikacji osiągały 100%, a sumy kwadratów błędów 0, co jest wynikiem bardzo zadowalającym. Widać wówczas już było, że najlepsze wyniki sieć osiągała dla współczynnika uczenia równego  $lr = 0.1$ . Po dokładniejszych badaniach udało się określić pożądane wartości pozostałych parametrów: 36 neuronów w pierwszej warstwie, 21 neuronów w drugiej warstwie - dawały one maksymalne Pk i minimalne SSE. Przykład działania sieci z takimi parametrami został zaprezentowany poniżej. Czerwony wykres obrazuje pożądane wyjście sieci, zielony natomiast odpowiadające im otrzymane wyjścia z sieci neuronowej.



Rysunek 4.35: Wykres prównujący wyjście pożądane (docelowe, kolor czerwony) z aktualnym wyjściem sieci (kolor zielony)

Dla pozostałych parametrów  $lr$  można zauważyć, że otrzymane wyniki dla różnych ilości neuronów są bardziej stochastyczne, zakres otrzymywanych wartości zwiększa się, a same wartości maleją wraz ze zmniejszaniem współczynnika uczenia. Warto zauważyć, że ilość neuronów na drugiej warstwie zdaje się być bardziej istotna, niż ich ilość na warstwie pierwszej. Otrzymane dla tych parametrów wartości mogą być podstawą dla dalszych rozważań, w których można zbadać wpływ innych parametrów sieci (np. maksymalna ilość epok) na poprawność klasyfikacji, oraz to, czy zmieniając je udałoby się tę poprawność zwiększyć. Podczas badania współczynnika uczenia nie udało się osiągnąć poprawności klasyfikacji równej 100% i sumy kwadratów błędów 0 - zarówno wtedy, gdy wartości te były średnimi dla wszystkich par  $S1$  i  $S2$  z zakresu, jak i wtedy, kiedy badano  $lr$  dla pożądanых ilości neuronów. W obu przypadkach dla kolejnych wartości  $lr$  poprawność klasyfikacji rosła, a suma kwadratów błędów malała, jednak nie osiągały maksimum i minimum. Ten fakt może być podstawą do kolejnych eksperymentów, w których można by badać, czy nadal zwiększając współczynnik uczenia udałoby się osiągnąć lepsze rezultaty.



## Literatura

- [1] Kosiński, Robert: Sztuczne sieci neuronowe. Dynamika nieliniowa i chaos. 2014. – ISBN 978-83-7926-221-2
- [2] MathWorks: Generate indices for training and test sets. [https://in.mathworks.com/help/bioinfo/ref/crossvalind.html?s\\_tid=srchtitle](https://in.mathworks.com/help/bioinfo/ref/crossvalind.html?s_tid=srchtitle)
- [3] MathWorks: Gradient descent backpropagation. [https://in.mathworks.com/help/deeplearning/ref/traingd.html?s\\_tid=mwa\\_osa\\_a](https://in.mathworks.com/help/deeplearning/ref/traingd.html?s_tid=mwa_osa_a)
- [4] Roman Zajdel, dr hab. inż. prof. PRz .: Sztuczna inteligencja, Laboratorium, Ćwiczenie 6. Model neuronu
- [5] Roman Zajdel, dr hab. inż. prof. PRz .: Sztuczna inteligencja, Laboratorium, Ćwiczenie 8. Sieć jednokierunkowa jednowarstwowa
- [6] Roman Zajdel, dr hab. inż. prof. PRz .: Sztuczna inteligencja, Laboratorium, Ćwiczenie 9. Sieć jednokierunkowa wielowarstwowa
- [7] Zocca, Valentino ; Spacagna, Gianmario ; Slater, Daniel ; Roelantsi, Peter: Deep Learning. Uczenie głębokie z językiem Python. Sztuczna inteligencja i sieci neuronowe. 2018. – ISBN 978-83-283-4174-6