POLITECHNIKA RZESZOWSKA
im. IGNACEGO ŁUKASIEWICZA

Katedra
Informatyki i Automatyki
Politechnika Rzeszowska

Sztuczna inteligencja - Projekt

Realizacja sieci neuronowej uczonej algorytmem wstecznej propagacji błędu z przyśpieszeniem metodą adaptacyjnego współczynnika uczenia uczącą się rozpoznawania jakości wina

Wykonał:
Krystian Szura
2 EF-DI P06

Rzeszów, 2019

Spis treści

1. Opis problemu

1.1. Cel projektu

Głównym założeniem projektu było przygotowanie algorytmu dla systemu eksperckiego, którego zadaniem było rozpoznawanie jakości wina. Została w tym celu stworzona sieć neuronowa uczona algorytmem wstecznej propagacji błędu. Jako metodę przyspieszającą uczenie zastosowano adaptacyjny współczynnik uczenia. Sieć zrealizowano przy pomocy funkcji *traingda* w programie *MATLAB* w wersji *2019a*.

Prace nad projektem obejmowały opracowanie danych do uczenia sieci, stworzenie oraz modyfikacje skryptu tworzącego wielowarstwową sieć neuronową, przeprowadzenie eksperymentów i opracowanie zebranych podczas nich danych a także przygotowanie spostrzeżeń i wniosków na podstawie wyników eksperymentów.

1.2. Czym jest *vihno verde*

Vinho verde (dosłownie z port. *zielone wino*) to portugalskie wino z regionu Minho na północnym zachodzie kraju oraz region winiarski, w którym jest produkowane. Nazwa odnosi się do młodości i świeżości wina, a nie do koloru, ponieważ butelkowane jest ono szybko po zbiorach i pite jako bardzo młode. Vinho verde najczęściej jest mieszanką, ale zdarzają się jednoszczepowe. Czołowe szczepy białe to Loureiro, Arinto, Alvarinho, Avesso, Trajadura. Czerwone – Vinhão i Alvarelhão. Białe wina są bardzo lekkie, mają 9–11% alk., są mocno kwaskowate, czasami delikatnie gazowane. Należy je pić bardzo zimne (nawet z zamrażarki). Czerwone vinho verde też są bardzo kwaśne i też należy je podawać z lodówki.[1]

[1] *https://winicjatywa.pl/10-rzeczy-o-vinho-verde/*

2. Specyfikacja danych

2.1. Przedstawienie zbioru danych

Opis problemu oraz przykładowe dane zostały przedstawione na stronie *http://archive.ics.uci.edu/ml/datasets/Wine+quality* przez prof. Paulo Corteza z Uniwersytetu Minho, Guimarães, Portugalia oraz A. Cerdeirę, F. Almeidę, T. Matosa i J. Reisa z Comissão de Viticultura da Região dos Vinhos Verdes (CVRVV)[2], Porto, Portugalia.

Dla problemu rozpoznawania jakości wina dostępne były dwa zbiory danych, powiązane z próbkami czerwonego i białego wina odmiany *vihno verde* z północnej Portugalii. Do dalszych rozważań używany będzie zbiór dla czerwonego wina.

Zbiór z danymi dla czerwonego wina zawierał w sobie 1599 przypadków. Do każdego przypadku przyporządkowane było 11 atrybutów wejściowych oraz dodatkowy atrybut wyjściowy. Atrybuty wejściowe zebrane na podstawie testów fizykochemicznych oznaczały:

1) stała kwasowość – większość kwasów związanych z winem, kwasy utrwalone i nielotne (g/dm^3),
2) lotna kwasowość – zawartość kwasu octowego w winie, która przy zbyt wysokich poziomach może prowadzić do octowego smaku (g/dm^3),
3) kwas cytrynowy – w niewielkich ilościach może dodać świeżości i smaku winom (g/dm^3),
4) cukier resztkowy – ilość cukru pozostałego po ustaniu fermentacji (g/dm^3),
5) chlorki – ilość soli w winie (g/dm^3),
6) wolny dwutlenek siarki – istnieje w równowadze między molekularnym SO_2 jako rozpuszczonym gazem i jonem wodorosiarczynowym; zapobiega rozwojowi drobnoustrojów i utlenianiu win (mg/dm^3),
7) całkowity dwutlenek siarki – ilość wolnych i związanych form SO_2, w niskich stężeniach niewyczuwalna, w większych staje się wyczuwalna w zapachu i smaku wina (mg/dm^3),
8) gęstość – zależna od procentowej zawartości alkoholu i cukru (g/dm^3),

[2] z port. *Komisja Uprawy Winorośli Regionu Vinho Verde*

9) pH – opisuje jak kwaśne lub zasadowe jest wino w skali od 0 (bardzo kwaśne) do 14 (bardzo zasadowe); większość win znajduje się pomiędzy 3-4 na skali pH,
10) siarczany – dodany do wina siarczan potasu, może przyczyniać się do poziomu SO_2, który działa jako środek przeciwdrobnoustrojowy i przeciwutleniacz (g/dm^3),
11) procentowa zawartość alkoholu (% objętości).

Atrybutem wyjściowym była jakość wina, która opiera się na danych sensorycznych (mediana co najmniej 3 ocen dokonanych przez ekspertów od wina). Każdy ekspert ocenił jakość wina pomiędzy 0 (bardzo złe) a 10 (bardzo dobre).

2.2. Normalizacja i sortowanie danych

W pierwotnej formie, dane były zapisane w arkuszu *.csv*, gdzie kolumny oznaczały atrybuty, a wiersze – rekordy. Dane zostały zaimportowane do programu MATLAB jako macierze, przy czym rozdzielono atrybuty wejściowe i wyjściowe, a następnie je transponowano.

W wyniku tego działania otrzymano macierz danych wejściowych, oznaczaną jako *P* (od ang. słowa *pattern*) o rozmiarze 11×1599 i macierz danych wyjściowych, oznaczoną jako *T* (od ang. słowa *target*) o rozmiarze 1×1599.

Następnie dane wejściowe i wyjściowe zostały posortowane. Na listingu 2.1. przedstawiono wykorzystany do tego skrypt.

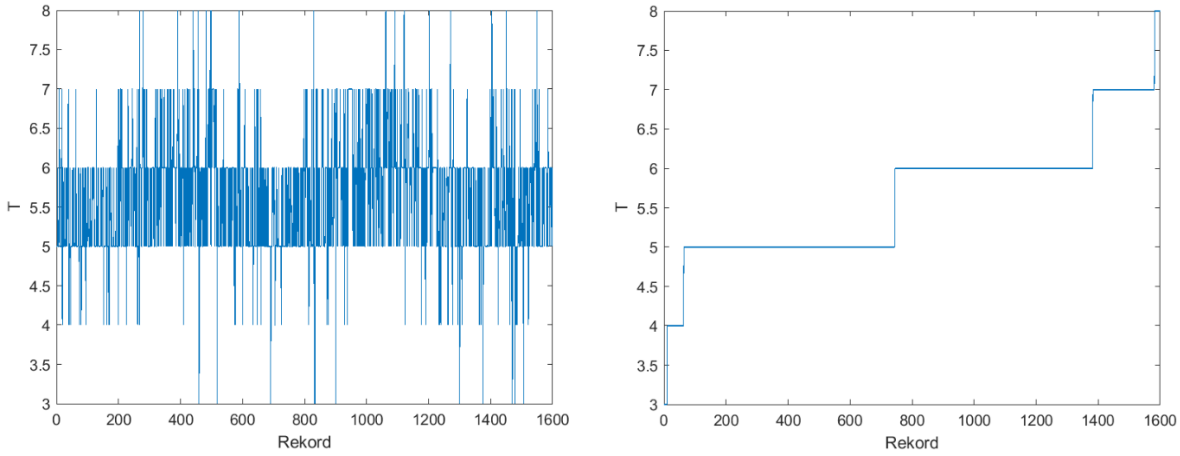
Listing 2.1. Sortowanie danych

```
[Ts, ind_Ts] = sort(Tn);      % posortowanie danych wzorcowych
                              % z zapamietaniem ich oryginalnych pozycji

Pns = zeros(size(Pn));        % utworzenie nowej macierzy, do ktorej
                              % skopiowane zostana posortowane dane uczące

for i = 1 : length(ind_Ts)    % wypelnienie macierzy posortowanymi danymi
    Pns(: , i) = Pn(: , ind_Ts(i));
end
```

Dzięki posortowaniu danych łatwiej jest zauważyć rozkład wartości wyjściowych dla poszczególnych rekordów, co zostało przedstawione na rysunku 2.1.



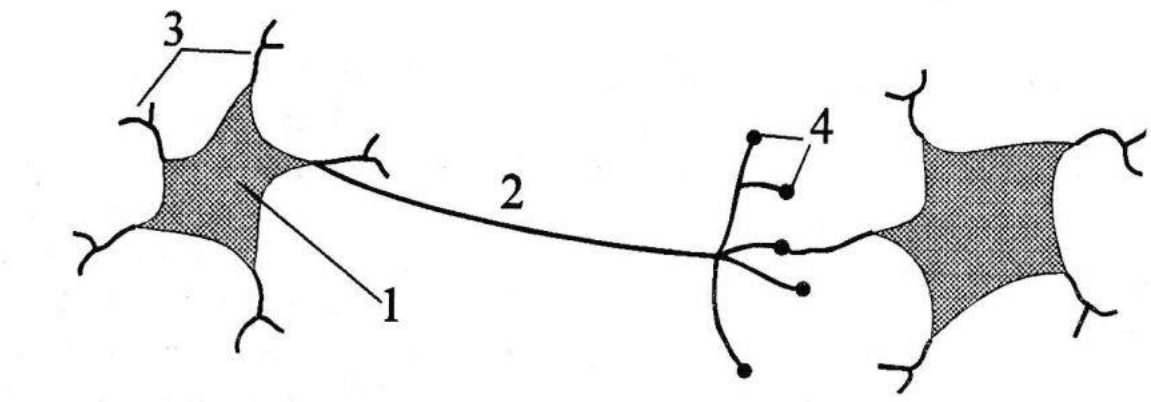
Rys. 2.1. Porównanie danych nieposortowanych (po lewej) i posortowanych (po prawej).

3. Zagadnienia teoretyczne

3.1. Neuron biologiczny i sztuczny

Podstawowym elementem systemu nerwowego jest komórka nerwowa zwana *neuronem*. W neuronie można wyróżnić ciało komórki nazywane *somą* oraz otaczające je dwa rodzaje wypustek: wypustki wprowadzające informacje do neuronu, tzw. *dendryty* i wypustkę wyprowadzającą, tzw. *akson.* Każdy neuron ma dokładnie jedną wypustkę wyprowadzającą, poprzez którą może wysyłać impulsy do innych neuronów.

Pojedynczy neuron przyjmuje pobudzenie od ogromnej liczby neuronów dochodzącej do tysiąca. Szacuje się, że w mózgu człowieka jest około 10^{11}, które oddziałują na siebie poprzez około 10^{15} połączeń. Jeden neuron przekazuje pobudzenie innym neuronom przez złącza nerwowe nazywane *synapsami*, przy czym transmisja sygnałów odbywa się na drodze skomplikowanych procesów chemiczno-elektrycznych. Synapsy pełnią rolę przekaźników informacji, w wyniku działania których pobudzenie może być wzmocnione lub osłabione. W rezultacie do neuronu dochodzą sygnały, z których część wywiera wpływ pobudzający, a część hamujący. Neuron sumuje impulsy pobudzające i hamujące. Jeżeli ich suma algebraiczna przekracza pewną wartość progową, to sygnał na wyjściu neuronu przesyłany jest – poprzez akson – do innych neuronów. Rysunek 3.1. przedstawia uproszczony schemat neuronu.[3]

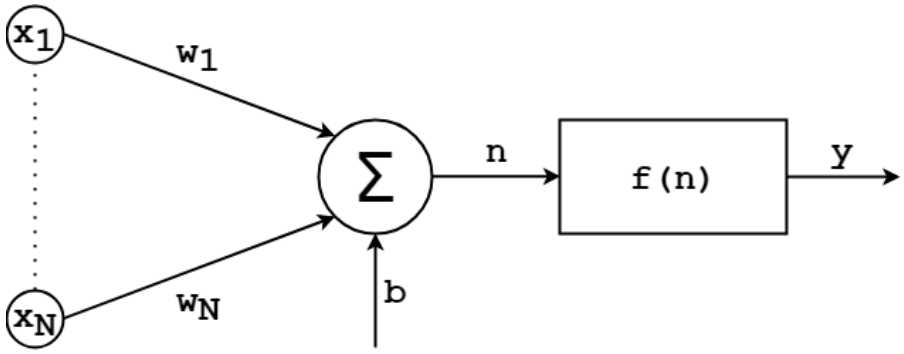


Rys. 3.1. Uproszczony schemat neuronu biologicznego i jego połączenia z sąsiednim neuronem: 1 – ciało komórki, 2 – akson, 3 – dendryty, 4 – synapsy. [3]

[3] Rutkowska D., Piliński M., Rutkowski L. *Sieci neuronowe, algorytmy genetyczne i systemy rozmyte.* Wydawnictwo Naukowe PWN, Warszawa 1997.

3.2. Neuron sztuczny

Podstawowym elementem sieci neuronowej jest neuron. Pojedynczy neuron odpowiada za przetwarzanie danych wejściowych na wartość wyjściową, która jest zależna od tego, jak dany neuron reaguje na te dane. Opis działania neuronu prezentuje model przedstawiony na rysunku 3.2.



Rys. 3.2. Model neuronu sztucznego:
x_1-x_N – sygnały wejściowe, w_1-w_N – współczynniki wagowe, b – bias, ∑ - sumator, n – łączne pobudzenie neuronu, $f(n)$ – funkcja aktywacji, y – wartość wyjściowa,

Sygnał wyjściowy pojedynczego neuronu y określony jest zależnością:

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

gdzie x_j jest j-tym sygnałem wejściowym, a w_j – j-tym współczynnikiem wagowym. Zapis może zostać skrócony poprzez stosowanie zapisu macierzowego do opisu działania neuronu. Wtedy $\boldsymbol{x} = [x_1, x_2, \dots, x_N]^T$ będzie wektorem sygnałów wejściowych, $\boldsymbol{w} = [w_1, w_2, \dots, w_N]$ – macierzą wierszową wag, a y i b – skalarami. Wówczas zależność (1) przyjmuje postać:

$$y = f(\boldsymbol{w}\boldsymbol{x} + b) \tag{2}$$

Ważona suma wejść wraz z przesunięciem jest nazywana często łącznym pobudzeniem neuronu. W dalszych rozważaniach oznaczana będzie ona symbolem n:

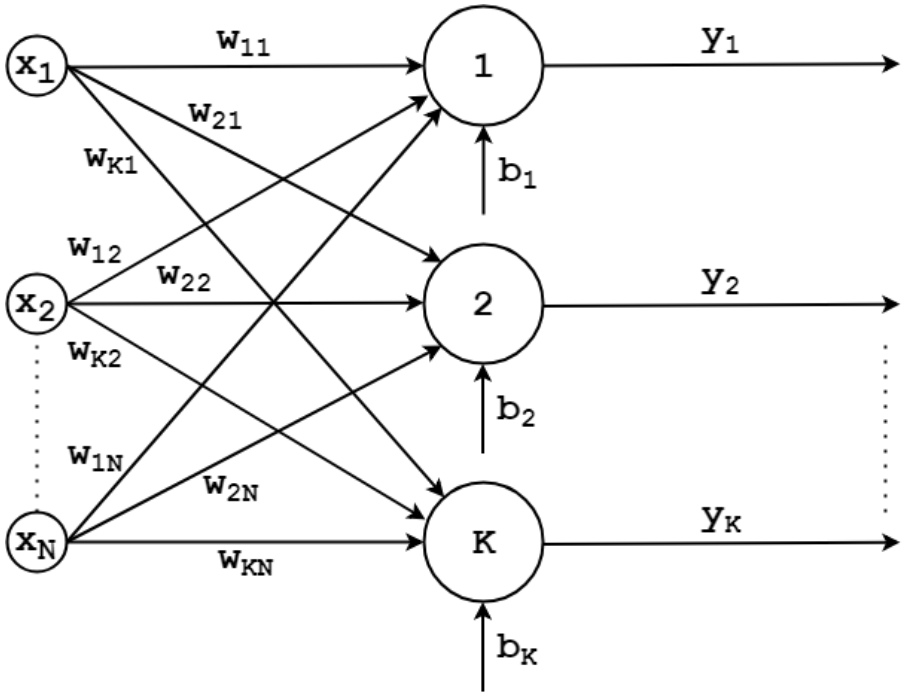
$$n = \sum_{j=1}^{N} w_j x_j + b. \tag{3}$$

W związku z zależnością (3), funkcję aktywacji można zapisać jako $f(n)$.[4]

[4] dr hab. inż. Roman Zajdel *Sztuczna inteligencja, Laboratorium, Ćw. 6. Model neuronu*, Politechnika Rzeszowska, Katedra Informatyki i Automatyki

3.3. Sieć jednokierunkowa jednowarstwowa

Sieć jednokierunkowa jednowarstwowa jest podstawowym elementem sieci wielowarstwowej. Neurony w tej sieci ułożone są w jednej warstwie. Przepływ sygnałów jest jednokierunkowy, tj. od wejścia do wyjścia. Schemat działania tej sieci przedstawiono na rysunku 3.3.



Rys. 3.3. Schemat sieci jednokierunkowej jednowarstwowej.
x_1-x_N – sygnały wejściowe, w_{11}-w_{KN} – współczynniki wagowe, *1*-*K* – kolejne neurony w warstwie, b_1-b_K – biasy dla *K*-tego neuronu, y_1-y_K –sygnały wyjściowe.

Wejściem dla każdego neuronu jest wektor sygnałów wejściowych $\boldsymbol{x} = [x_1, x_2, \dots, x_N]^T$. Każde wejście jest połączone z każdym neuronem. Dodatkowo, każdy neuron posiada swój własny wektor wag. Każdy neuron przedstawiony jest jako kółko z kolejną liczbą naturalną, aż do K. Stosując zapis macierzowy funkcjonowanie warstwy neuronów można opisać wzorem:

$$\boldsymbol{y} = f(\boldsymbol{w}\boldsymbol{x} + \boldsymbol{b}), \tag{4}$$

gdzie: $\boldsymbol{b} = [b_1, b_2, \dots, b_K]^T$ jest wektorem przesunięć, $\boldsymbol{y} = [y_1, y_2, \dots, y_K]^T$ jest wektorem sygnałów wyjściowych, a $\boldsymbol{w}$ jest prostokątna macierzą o wymiarach $K \times N$, gdzie K oznacza liczbę neuronów w warstwie, a N jest liczbą sygnałów wejściowych. Waga w_{pq} oznacza wagę sygnału wejściowego q wchodzącą do neuronu p. Budowę macierzy $\boldsymbol{w}$ przedstawiono poniżej.

$$\boldsymbol{w} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1N} \\ w_{21} & w_{22} & \cdots & w_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ w_{K1} & w_{K2} & \cdots & w_{KN} \end{bmatrix} \tag{5}$$

Celem uczenia pod nadzorem pojedynczej warstwy neuronów jest takie dobranie wag, które pozwoli na odwzorowanie danych wejściowych w wyjściowe z zadaną, ustaloną dokładnością. Zmiana wag odbywa się w kolejnych cyklach uczenia oznaczonych tutaj jako τ, zwanych epokami. Dla pojedynczej *j*-tej wagi dla *i*-tego neuronu może być ona opisana zależnością iteracyjną:

$$w_{ij}(\tau + 1) = w_{ij}(\tau) + \Delta w_{ij}(\tau) \tag{6}$$

Uczenie pod nadzorem zakłada, że każdemu wektorowi wejściowemu $\boldsymbol{x}$ towarzyszy pożądany wektor sygnałów wyjściowych $\boldsymbol{t} = [t_1, t_2, \dots, t_K]^T$. Para $(\boldsymbol{x}, \boldsymbol{t})$ używana jest w procesie uczenia. Jeżeli sieć nie jest nauczona, sygnał $\boldsymbol{t}$ różni się od sygnału wyjściowego $\boldsymbol{y}$. Popełniany błąd można zdefiniować jako:

$$\boldsymbol{e} = \boldsymbol{y} - \boldsymbol{t} \tag{7}$$

Celem uczenia jest uzyskanie zgodności odpowiedzi $\boldsymbol{y}$ z wartościami wymaganymi $\boldsymbol{t}$, co sprowadza się do minimalizacji funkcji błędu, którą najczęściej wyraża się jako:

$$E = \frac{1}{2}\sum_{j=1}^{K} e_j^2 \tag{8}$$

Poszukiwanie minimum tej funkcji możliwe jest za pomocą metody gradientowej, w szczególności metody największego spadku. Wektor przyrostu wag jest wyrażony jako:

$$\Delta \boldsymbol{w} = -\eta \nabla E(\boldsymbol{w}) \tag{9}$$

gdzie: ∇ jest gradientem, η – współczynnikiem uczenia. Znak minus wynika z faktu, iż gradient wskazuje kierunek najszybszego wzrostu funkcji, kiedy w minimalizacji chodzi o jak najszybszy spadek.

Rozważając wyznaczanie wzoru uczenia gradientowego jedynie dla *i*-tego neuronu i *j*-tej wagi otrzymujemy wzór:

$$\Delta w_{ij} = -\eta \frac{\partial E}{\partial w_{ij}} \tag{10}$$

Uwzględniając uwikłanie zależności E od w_{ij} poprzez y_i i n_i , czyli $E = E\left(y_i\left(n_i(w_{ij})\right)\right)$ pochodną można zapisać następująco:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial y_i} \cdot \frac{\partial y_i}{\partial n_i} \cdot \frac{\partial n_i}{\partial w_{ij}} \tag{11}$$

Uwzględniając zależności (7) i (8) otrzymujemy :

$$\begin{aligned} \frac{\partial E}{\partial y_i} &= \frac{\partial}{\partial y_i}\left(\frac{1}{2}(y_1 - t_1)^2 + \cdots + \frac{1}{2}(y_i - t_i)^2 + \cdots + \frac{1}{2}(y_K - t_K)^2\right) \\ &= y_i - t_i \ = e_i \end{aligned} \tag{12}$$

Na podstawie zależności (3) otrzymujemy:

$$\frac{\partial n_i}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}}(w_{i1}x_1 + \cdots + w_{ij}x_j + \cdots + w_{iN}x_N + b_i) = \ x_j \tag{13}$$

Podstawiając powyższe pochodne cząstkowe do równania (11) a następnie otrzymane równanie do zależności (10) otrzymujemy:

$$\Delta w_{ij} = \ -\eta \frac{\partial E}{\partial w_{ij}} = \eta \cdot (t_i - y_i) \cdot \frac{\partial y_i}{\partial n_i} \cdot x_j \tag{14}$$

gdzie $\frac{\partial y_i}{\partial n_i} = f_i'(n_i)$ jest równe pochodnej cząstkowej *i*-tej funkcji aktywacji po *i*-tym łącznym pobudzeniu neuronu.[5]

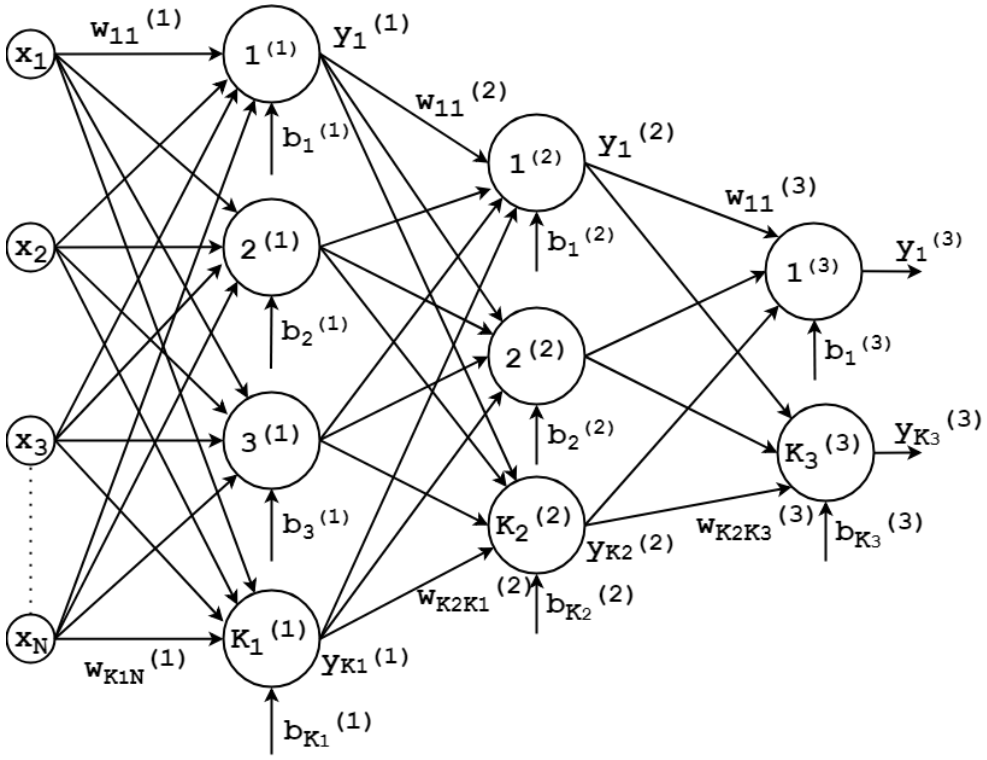
3.4. Sieć jednokierunkowa wielowarstwowa

Sieć jednokierunkowa wielowarstwowa najczęściej posiada jedną warstwę neuronów ukrytych pomiędzy warstwą wejściową a wyjściową. Taką sieć nazywa się siecią trójwarstwową. Połączenia pomiędzy warstwami neuronów są typu każdy z każdym. Sygnały wejściowe podawane są do warstwy wejściowej neuronów, których wyjścia stanowią sygnały źródłowe dla kolejnej warstwy.

Na mocy uniwersalnego twierdzenia o aproksymacji sieć trójwarstwowa ze skończoną ilością neuronów w warstwie ukrytej może aproksymować funkcje ciągłe na zwartych podzbiorach $\mathbb{R}^n$, przy drobnych założeniach dotyczących funkcji aktywacji. Pozwala to stwierdzić, że proste sieci neuronowe mogą reprezentować wiele różnych interesujących funkcji, gdy otrzymają odpowiednie parametry. Twierdzenie nie dotyka

[5] dr hab. inż. Roman Zajdel *Sztuczna inteligencja, Laboratorium, Ćw. 8. Sieć jednokierunkowa jednowarstwowa,* Politechnika Rzeszowska, Katedra Informatyki i Automatyki

jednak algorytmicznej zdolności uczenia się tych parametrów.[6] Schemat takiej sieci został przedstawiony na rysunku 3.4.



Rys. 3.4. Schemat sieci jednokierunkowej wielowarstwowej.

Każda warstwa neuronów posiada swoją macierz wag $\boldsymbol{w}$, wektor przesunięć $\boldsymbol{b}$, funkcję aktywacji f i wektor sygnałów wyjściowych $\boldsymbol{y}$. Aby rozróżniać je ze względu na warstwy, do każdej wielkości dodano indeks górny, oznaczający numer warstwy, której dotyczą. Dla przykładu, wektor sygnałów wyjściowych dla warstwy drugiej będzie oznaczany jako $\boldsymbol{y}^{(2)}$. Działanie każdej z warstw można rozpatrywać oddzielnie, na przykład warstwa druga posiada $N = K_1$ sygnałów wejściowych, $K = K_2$ neuronów i macierz wag $\boldsymbol{w} = \boldsymbol{w}^{(2)}$o wymiarze $K_2 \times K_1$. Wejściem warstwy drugiej jest wyjście warstwy pierwszej $\boldsymbol{x} = \boldsymbol{y}^{(1)}$ a wyjściem $\boldsymbol{y} = \boldsymbol{y}^{(2)}$. Działanie poszczególnych warstw jest dane jako:

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

$$\boldsymbol{y}^{(2)} = \boldsymbol{f}^{(2)}\left(\boldsymbol{w}^{(2)}\boldsymbol{y}^{(1)} + \boldsymbol{b}^{(2)}\right) \tag{16}$$

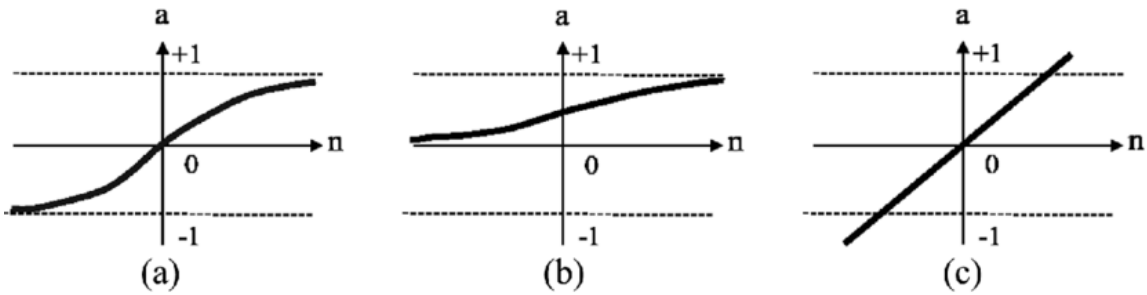
$$\boldsymbol{y}^{(3)} = \boldsymbol{f}^{(3)}\left(\boldsymbol{w}^{(3)}\boldsymbol{y}^{(2)} + \boldsymbol{b}^{(3)}\right) \tag{17}$$

[6] *https://en.wikipedia.org/wiki/Universal_approximation_theorem*

Działanie całej sieci można opisać jednym równaniem:

$$\boldsymbol{y}^{(3)} = \boldsymbol{f}^{(3)}\left(\boldsymbol{w}^{(3)}\boldsymbol{f}^{(2)}\left(\boldsymbol{w}^{(2)}\boldsymbol{f}^{(1)}\left(\boldsymbol{w}^{(1)}\boldsymbol{x} + \boldsymbol{b}^{(1)}\right) + \boldsymbol{b}^{(2)}\right) + \boldsymbol{b}^{(3)}\right) \qquad (18)$$

Sieci jednokierunkowe wielowarstwowe w warstwach wejściowej i ukrytej najczęściej wykorzystują funkcje aktywacji typu sigmoidalnego – *tansig* lub *logsig*, zaprezentowane na rysunku 3.5.a., 3.5.b. Zastosowanie tego typu funkcji w warstwach wyjściowych może być niekorzystne, głownie ze względu na wąski zakres (od 0 lub -1 do 1) sygnału wyjściowego. W związku z tym, w warstwie wyjściowej używa się głownie funkcji liniowej *purelin* (rys. 3.5. c.), która nie posiada takich ograniczeń.[7] Wzory funkcji aktywacji wraz z ich pochodnymi zawiera tabela 3.1.



Rys. 3.4. Funkcje aktywacji – a) *tansig*, b) *logsig*, c) *purelin*.[8]

Tabela 3.1. Najczęściej używane funkcje aktywacji wraz z pochodnymi.[9]

Nazwa funkcji	$f(n)$	$f'(n)$
tansig	$\frac{2}{1+exp(-2n)} - 1$	$1 - f(n)^2$
logsig	$\frac{1}{1 + exp(-n)}$	$f(n)\,(1 - f(n)\,)$
purelin	n	1

Funkcje te posiadają pochodne dające się przedstawić za pomocą tej samej funkcji, co niweluje konieczność używania numerycznego różniczkowania, przyspieszając działanie algorytmów uczących.

[7] dr hab. inż. Roman Zajdel *Sztuczna inteligencja, Laboratorium, Ćw. 9. Sieć jednokierunkowa wielowarstwowa*, Politechnika Rzeszowska, Katedra Informatyki i Automatyki

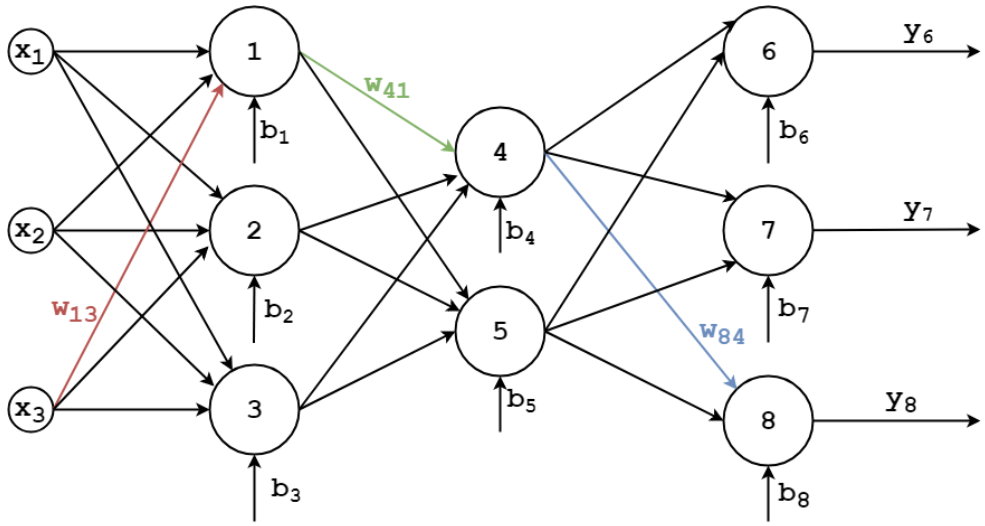
[8] https://www.researchgate.net/figure/ANN-transfer-functions-considered-a-Log-sigmoid-a-Logsign-1-1-e-n-b_fig3_327223187

[9] https://www.mathworks.com/help/deeplearning/

3.5. Algorytm wstecznej propagacji błędu

Podstawowym sposobem uczenia pod nadzorem sieci wielowarstwowej jest metoda wstecznej propagacji błędu. Polega ona na aktualizacji wag zaczynając od warstwy wyjściowej, następnie w kolejnych warstwach aż do warstwy wejściowej. Wagi każdej warstwy zmieniane są w zależności od parametrów tej warstwy i kolejnych w stronę wyjścia.

Sposób działania algorytmu wstecznej propagacji błędu najłatwiej przedstawić jest na przykładzie. Korzystając z schematu przykładowej sieci jednokierunkowej wielowarstwowej przedstawionego na rysunku 3.5. zaznaczono po jednej przykładowej wadze z każdej warstwy:



Rys. 3.5. Schemat przykładowej sieci jednokierunkowej wielowarstwowej.

Obliczając nową wagę warstwy ostatniej, w tym przypadku w_{84}, korzystamy z zależności (11) i (14):

$$\Delta w_{84} = -\eta \frac{\partial E}{\partial w_{84}} = -\eta \frac{\partial E}{\partial y_8} \cdot \frac{\partial y_8}{\partial n_8} \cdot \frac{\partial n_8}{\partial w_{84}} \tag{19}$$

Pochodna cząstkowa z błędu E po funkcji aktywacji y_8 jest równa $y_8 - t_8$, czyli błędowi cząstkowemu e_8 na wyjściu neuronu ósmego (patrz zależność (12)).

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 $\frac{\partial y_8}{\partial n_8} = f_8'(n_8)$.

Pochodna cząstkowa z łącznego pobudzenia neuronu n_8 po wadze w_{84} jest równy sygnałowi wejściowemu do neuronu 8 pochodzącego od neuronu 4 (patrz zależność (13)), a więc jest to sygnał y_4.

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

$$w_{84}^{*} = w_{84} + \eta \cdot e_8 \cdot f_8'(n_8) \cdot y_4 = w_{84} + \delta_8 \cdot y_4 \tag{20}$$

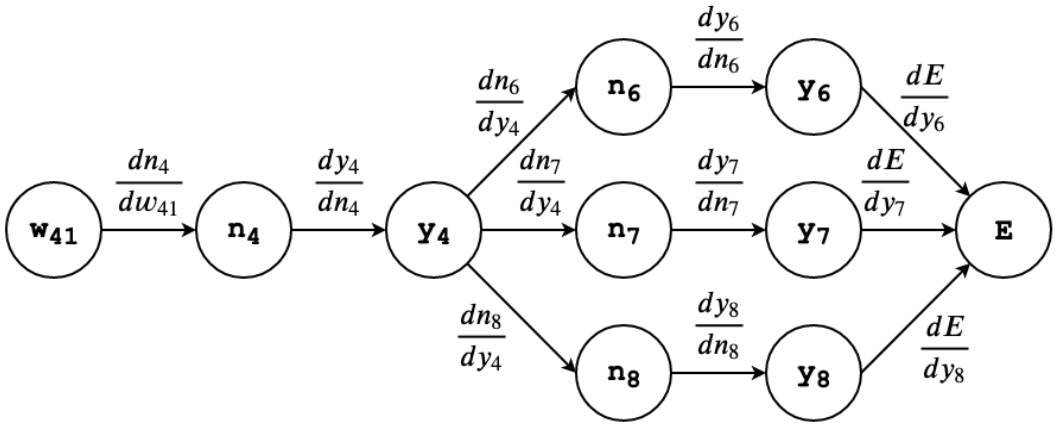
Bardzo często człon $\eta \cdot e_8 \cdot f_8'(n_8)$ jest zapisywany jako δ_8.[10] Bias można obliczyć w bardzo podobny sposób Dwie pierwsze pochodne cząstkowe liczone są w ten sam sposób. Jedynie ostatnia pochodna przyjmuje postać:

$$\frac{\partial n_i}{\partial b_i} = \frac{\partial}{\partial b_i}(w_{i1}x_1 + \cdots + w_{iN}x_N + b_i) = 1 \tag{21}$$

Stąd też wzór na nowy bias przyjmuje postać:

$$b_8^{*} = b_8 + \eta \cdot e_8 \cdot f_8'(n_8) \cdot 1 = b_8 + \delta_8 \tag{22}$$

Dla nowej wagi warstwy drugiej, dla przykładu wagi w_{41} najwygodniej jest skorzystać z pomocniczego schematu zależności (rys. 3.6.):



Rys. 3.6. Pomocniczy schemat zależności wagi w_{41}.

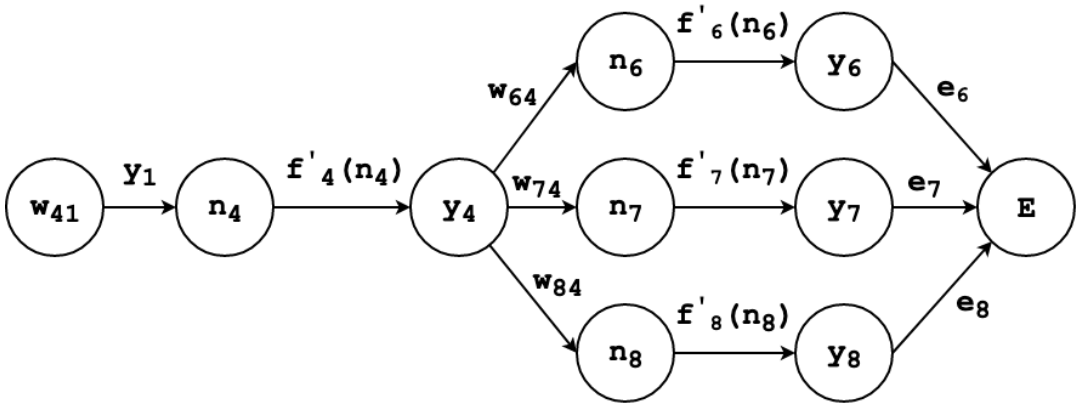
Na schemacie pokazane są kolejne zależności zmiennych. Błąd E zależy bezpośrednio od sygnałów wyjściowych w ostatniej warstwie (y_{6-8}), które zależą bezpośrednio od łącznego pobudzenia neuronów w odpowiednich neuronach (n_{6-8}), które zależą bezpośrednio od sygnału wyjściowego y_4, który zależy bezpośrednio 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 $\frac{\partial E}{\partial y_i} = e_i$, $\frac{\partial y_i}{\partial n_i} = f_i'(n_i)$ oraz $\frac{\partial n_i}{\partial w_{ij}} = y_j$. Pozostało obliczyć pochodną cząstkową typu $\frac{\partial n_i}{\partial y_j}$. Wynosi ona:

[10] https://en.wikipedia.org/wiki/Delta_rule

$$\frac{\partial n_i}{\partial y_j} = \frac{\partial}{\partial y_j}(w_{i1}y_1 + \cdots + w_{ij}y_j + \cdots + w_{iN}x_N + b_i) = w_{ij} \tag{23}$$

Stosując powyższe zależności, można nanieść je na schemat, który będzie wyglądał jak na rysunku 3.7.:



Rys. 3.7. Pomocniczy schemat zależności wagi w_{41} z obliczonymi pochodnymi.

Aby otrzymać poszukiwaną pochodną $\frac{\partial E}{\partial w_{41}}$ 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:

$$\frac{\partial E}{\partial w_{41}} = [e_6 \cdot f_6'(n_6) \cdot w_{64} + e_7 \cdot f_7'(n_7) \cdot w_{74} + e_8 \cdot f_8'(n_8) \cdot w_{84}] \cdot f_4'(n_4) \cdot y_1 \tag{24}$$

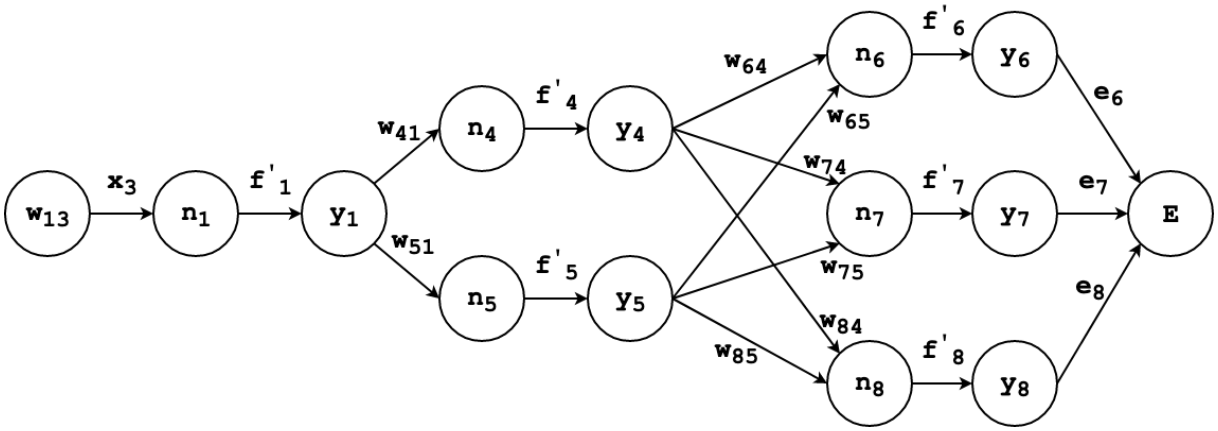
Dla skrócenia zapisu, przyjęto oznaczenie $f_x'(n_x) = f_x'$. Ostatecznie nowa waga w_{41}, oznaczona indeksem górnym z gwiazdką wynosi:

$$\begin{aligned} w_{41}^* = w_{41} + \eta \cdot [e_6 \cdot f_6' \cdot w_{64} + e_7 \cdot f_7' \cdot w_{74} + e_8 \cdot f_8' \cdot w_{84}] \cdot f_4' \cdot y_1 \; = \\ = w_{41} + \delta_4 \cdot y_1 \end{aligned} \tag{25}$$

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.[10] Podobnie jak w przypadku warstwy wyjściowej, poprzez zależność (21), nowy bias jest definiowany jako:

$$b_4^* = b_4 + \eta \cdot [e_6 \cdot f_6' \cdot w_{64} + e_7 \cdot f_7' \cdot w_{74} + e_8 \cdot f_8' \cdot w_{84}] \cdot f_4' \; = b_4 + \delta_4 \tag{26}$$

Dla nowej wagi w warstwie pierwszej, dla przykładu wagi w_{13}, należy postąpić w taki sam sposób, tworząc pomocniczy schemat (rys. 3.8.):



Rys. 3.8. Pomocniczy schemat zależności wagi w_{13} z obliczonymi pochodnymi.

Tak samo jak wcześniej, wiadomo, że $\frac{\partial E}{\partial y_i} = e_i$, $\frac{\partial y_i}{\partial n_i} = f_i'(n_i) = f_i'$ oraz $\frac{\partial n_i}{\partial y_j} = w_{ij}$. Jako iż jest to warstwa pierwsza, modyfikując równanie (23), pochodna łącznego pobudzenia neuronu po wadze będzie wynosić $\frac{\partial n_i}{\partial w_{ij}} = x_j$.

Ponownie, mnożąc wszystkie pochodne na drodze z E do w_{13}, uwzględniając rozgałęzienia, otrzymujemy:

$$\begin{aligned}\frac{\partial E}{\partial w_{13}} = [&e_6 \cdot f_6' \cdot (w_{64} \cdot f_4' \cdot w_{41} + w_{65} \cdot f_5' \cdot w_{51}) + \\ &+ e_7 \cdot f_7' \cdot (w_{74} \cdot f_4' \cdot w_{41} + w_{75} \cdot f_5' \cdot w_{51}) + \\ &+ e_8 \cdot f_8' \cdot (w_{84} \cdot f_4' \cdot w_{41} + w_{85} \cdot f_5' \cdot w_{51})] \cdot f_1' \cdot x_3\end{aligned} \tag{27}$$

Podstawiając powyższy wynik do zależności (14) a następnie do (6) otrzymujemy nową wagę w_{13}:

$$\begin{aligned}w_{13}^* = w_{13} + \eta \cdot [&e_6 \cdot f_6' \cdot (w_{64} \cdot f_4' \cdot w_{41} + w_{65} \cdot f_5' \cdot w_{51}) + \\ &+ e_7 \cdot f_7' \cdot (w_{74} \cdot f_4' \cdot w_{41} + w_{75} \cdot f_5' \cdot w_{51}) + \\ &+ e_8 \cdot f_8' \cdot (w_{84} \cdot f_4' \cdot w_{41} + w_{85} \cdot f_5' \cdot w_{51})] \cdot f_1' \cdot x_3 = \\ &= w_{13} + \delta_1 \cdot y_3\end{aligned} \tag{28}$$

Człon $\eta \cdot [e_6 \cdot f_6' \cdot (w_{64} \cdot f_4' \cdot w_{41} + \cdots + w_{85} \cdot f_5' \cdot w_{51})] \cdot f_1'$ jest często zapisywany jako δ_1.[10] Podobnie jak w przypadku poprzednich warstw, poprzez zależność (21), nowy bias jest definiowany jako:

$$b_1^* = b_1 + \eta \cdot [e_6 \cdot f_6' \cdot (w_{64} \cdot f_4' \cdot w_{41} + \cdots + w_{85} \cdot f_5' \cdot w_{51})] \cdot f_1' = b_1 + \delta_1 \tag{29}$$

3.6. Adaptacyjny współczynnik uczenia

Podstawowym elementem systemu nerwowego jest komórka nerwowa. Podstawowym elementem systemu nerwowego jest komórka nerwowa. Podstawowym elementem systemu nerwowego jest komórka nerwowa. Podstawowym elementem systemu nerwowego jest komórka nerwowa. Podstawowym elementem systemu nerwowego jest komórka nerwowa. Podstawowym elementem systemu nerwowego jest komórka nerwowa. Podstawowym elementem systemu nerwowego jest komórka nerwowa. Podstawowym elementem systemu nerwowego jest komórka nerwowa. Podstawowym elementem systemu nerwowego jest komórka nerwowa. Podstawowym elementem systemu nerwowego jest komórka nerwowa. Podstawowym elementem systemu nerwowego jest komórka nerwowa. Podstawowym elementem systemu nerwowego jest komórka nerwowa. Podstawowym elementem systemu nerwowego jest komórka nerwowa. Podstawowym elementem systemu nerwowego jest komórka nerwowa.

4. Algorytm

4.1. Kod programu

Podstawowym elementem systemu nerwowego jest komórka nerwowa. Podstawowym elementem systemu nerwowego jest komórka nerwowa. Podstawowym elementem systemu nerwowego jest komórka nerwowa. Podstawowym elementem systemu nerwowego jest komórka nerwowa. Podstawowym elementem systemu nerwowego jest komórka nerwowa. Podstawowym elementem systemu nerwowego jest komórka nerwowa. Podstawowym elementem systemu nerwowego jest komórka nerwowa. Podstawowym elementem systemu nerwowego jest komórka nerwowa. Podstawowym elementem systemu nerwowego jest komórka nerwowa. Podstawowym elementem systemu nerwowego jest komórka nerwowa. Podstawowym elementem systemu nerwowego jest komórka nerwowa. Podstawowym elementem systemu nerwowego jest komórka nerwowa. Podstawowym elementem systemu nerwowego jest komórka nerwowa. Podstawowym elementem systemu nerwowego jest komórka nerwowa. Podstawowym elementem systemu nerwowego jest komórka nerwowa. Podstawowym elementem systemu nerwowego jest komórka nerwowa.

4.2. Opis algorytmu

Podstawowym elementem systemu nerwowego jest komórka nerwowa. Podstawowym elementem systemu nerwowego jest komórka nerwowa. Podstawowym elementem systemu nerwowego jest komórka nerwowa. Podstawowym elementem systemu nerwowego jest komórka nerwowa. Podstawowym elementem systemu nerwowego jest komórka nerwowa. Podstawowym elementem systemu nerwowego jest komórka nerwowa. Podstawowym elementem systemu nerwowego jest komórka nerwowa. Podstawowym elementem systemu nerwowego jest komórka nerwowa. Podstawowym elementem systemu nerwowego jest komórka nerwowa. Podstawowym elementem systemu nerwowego jest komórka nerwowa.

5. Eksperymenty

5.1. Wpływ parametru *max_fail* na moment przerwania uczenia i poprawność klasyfikacji

Współczynnik *max_fail* odpowiada za liczbę sprawdzeń walidacyjnych, która reprezentuje liczbę sukcesywnych iteracji, w których błąd średniokwadratowy sieci nie zmniejsza się. Jeśli liczba ta osiągnie zadaną wartość, uczenie sieci zostanie przerwane.[11]

W eksperymencie ustawiono arbitralnie maksymalną liczbę epok na 100 000, liczbę neuronów w pierwszej warstwie na 50 oraz liczbę neuronów w drugiej warstwie na 20. Wartość parametru *max_fail* zmieniano na wartości [10 50 100 500 1000 2500 5000 10000]. Następnie badano, jaki wpływ na moment, w którym sieć przerwie naukę ma parametr *max_fail*.

Utworzono wektor *epoch_output* o rozmiarze 1×8, do którego zapisywano kolejne wartości epok, w których uczenie zostało przerwane (*best_epoch*) oraz wektor *pk_output* o tym samym rozmiarze, do którego zapisywano poprawność klasyfikacji dla sieci w aktualnej konfiguracji.

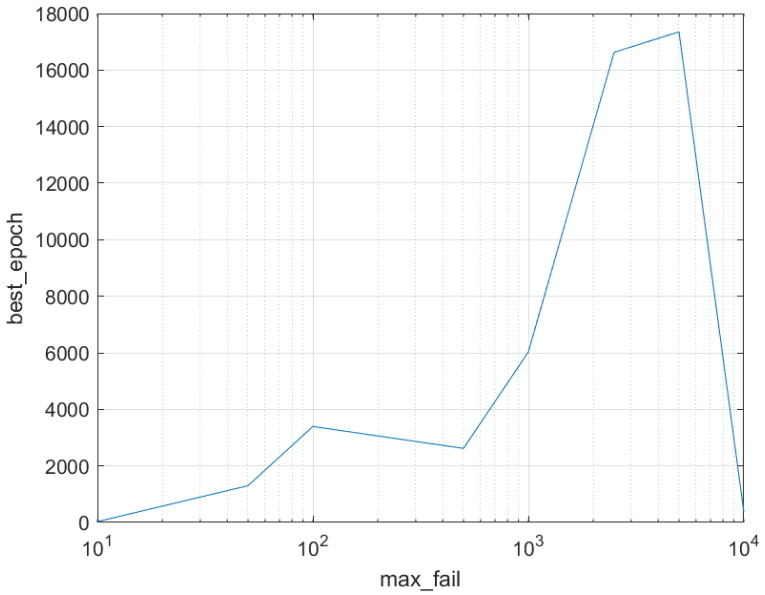
Parametr *max_fail* zmieniany był w pętli, po czym po wykonaniu jej, rysowano wykresy *epoch_output* (rys. 5.1.) oraz *pk_output* (rys. 5.2.) w zależności od *max_fail* w skali liniowo-logarytmicznej. Kod użyty do rysowania wykresów został zamieszczony poniżej, w listingu 5.1.

Listing 5.1. Rysowanie wykresów

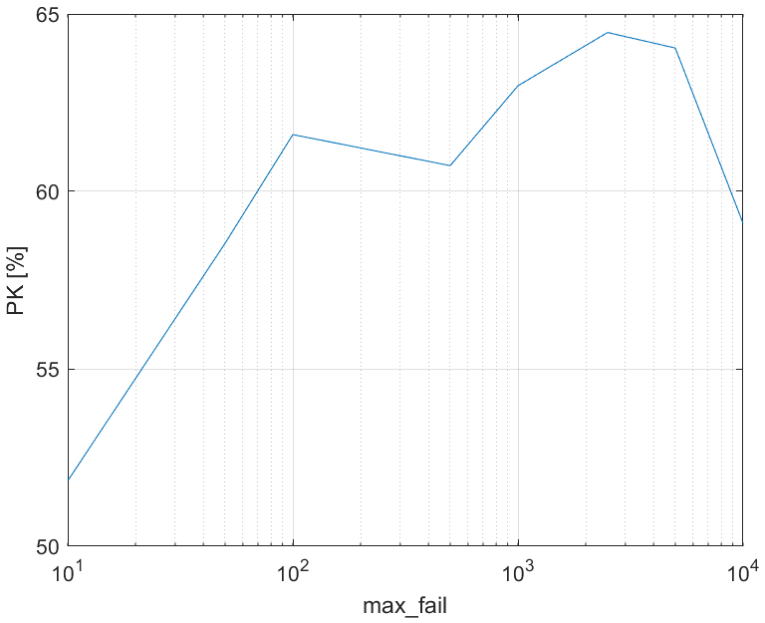
```
semilogx(maxerr_vec,epoch_output);
xlabel('max\_fail');
ylabel('best\_epoch');
grid on;

semilogx(maxerr_vec,pk_output);
xlabel('max\_fail');
ylabel('PK [%]');
grid on;
```

[11] *https://www.mathworks.com/help/deeplearning/ug/train-and-apply-multilayer-neural-networks.html#bss331l-17*



Rys. 5.1. Zależność liczby przebytych epok od parametru *max_fail*.



Rys. 5.2. Zależność poprawności klasyfikacji od parametru *max_fail*.

Sieć uzyskiwała najlepsze wyniki, gdy parametr *max_fail* ustawiony był na wartość 2500 lub 5000, co oznaczało odpowiednio 2,5 i 5% maksymalnej liczby epok. Zdecydowano więc na ustalenie parametru *max_fail* na 5% maksymalnej liczby epok w dalszych eksperymentach.

5.2. Wpływ maksymalnej liczby epok na poprawność klasyfikacji i błąd średniokwadratowy

W eksperymencie ponownie ustawiono liczbę neuronów w pierwszej warstwie na 50 oraz liczbę neuronów w drugiej warstwie na 20. Na podstawie poprzedniego eksperymentu, parametr *max_fail* został ustawiony na 5% maksymalnej liczby epok, która zmieniana była w zakresie [1000 5000 10000 30000 40000 50000 75000 100000]. Następnie badano, jaki wpływ na poprawność klasyfikacji i błąd średniokwadratowy ma maksymalna liczba epok.

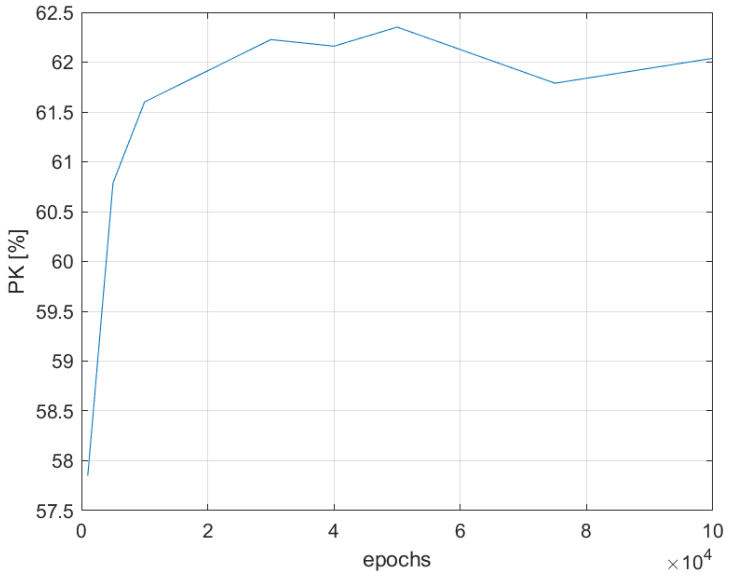
Utworzono wektor *pk_output* o rozmiarze 1×8, do którego zapisywano kolejne poprawności klasyfikacji oraz wektor *error_output* o tym samym rozmiarze, do którego zapisywano błąd średniokwadratowy dla sieci w aktualnej konfiguracji.

Maksymalna liczba epok zmieniana była w pętli, po czym po wykonaniu jej, rysowano wykresy *pk_output* (rys. 5.3.) oraz *error_output* (rys. 5.4.) w zależności od maksymalnej liczby epok. Kod użyty do rysowania wykresów został zamieszczony poniżej, w listingu 5.2.

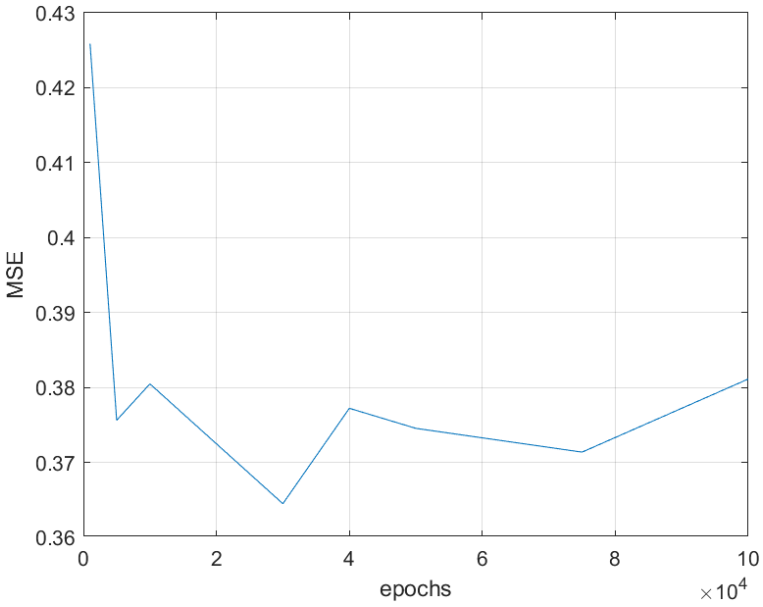
Listing 5.2. Rysowanie wykresów

```
plot(epochs,pk_output);
xlabel('epochs');
ylabel('PK [%]');
grid on

plot(epochs,error_output);
xlabel('epochs');
ylabel('MSE');
grid on
```



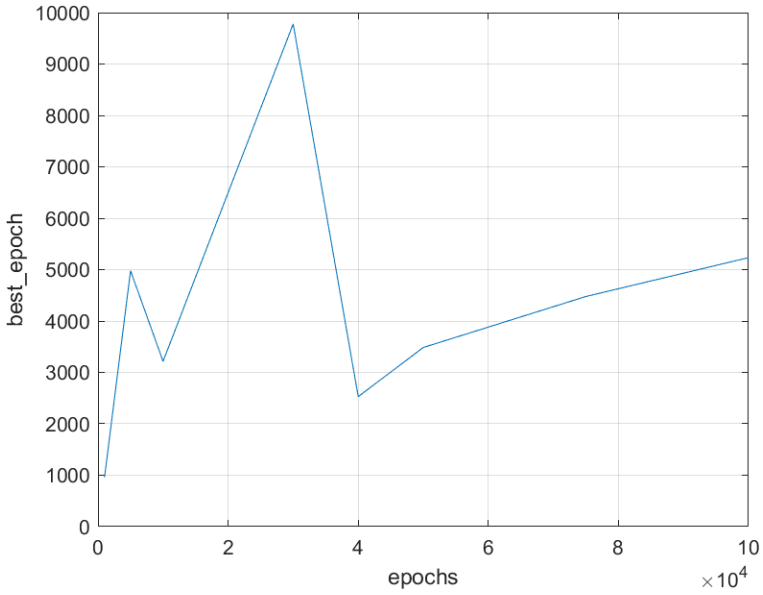
Rys. 5.3. Zależność poprawności klasyfikacji od maksymalnej liczby epok.



Rys. 5.4. Zależność błędu średniokwadratowego od maksymalnej liczby epok.

Obserwując powyższe wykresy oraz plik wynikowy eksperymentu można dojść do wniosku, że mała liczba epok wpływa negatywnie na poprawność klasyfikacji oraz błąd średniokwadratowy.

Sieć osiągnęła najmniejszy błąd średniokwadratowy dla wartości 30 000 epok. Również od tej wartości poprawność klasyfikacji zaczęła utrzymywać się na poziomie ponad 62%. Dla tej też wartości maksymalnej liczby epok osiągnięto największą liczbę przebytych epok (rys. 5.5.), co może uzasadniać uzyskany niższy błąd średniokwadratowy.



Rys. 5.5. Zależność liczby przebytych epok od maksymalnej liczby epok.

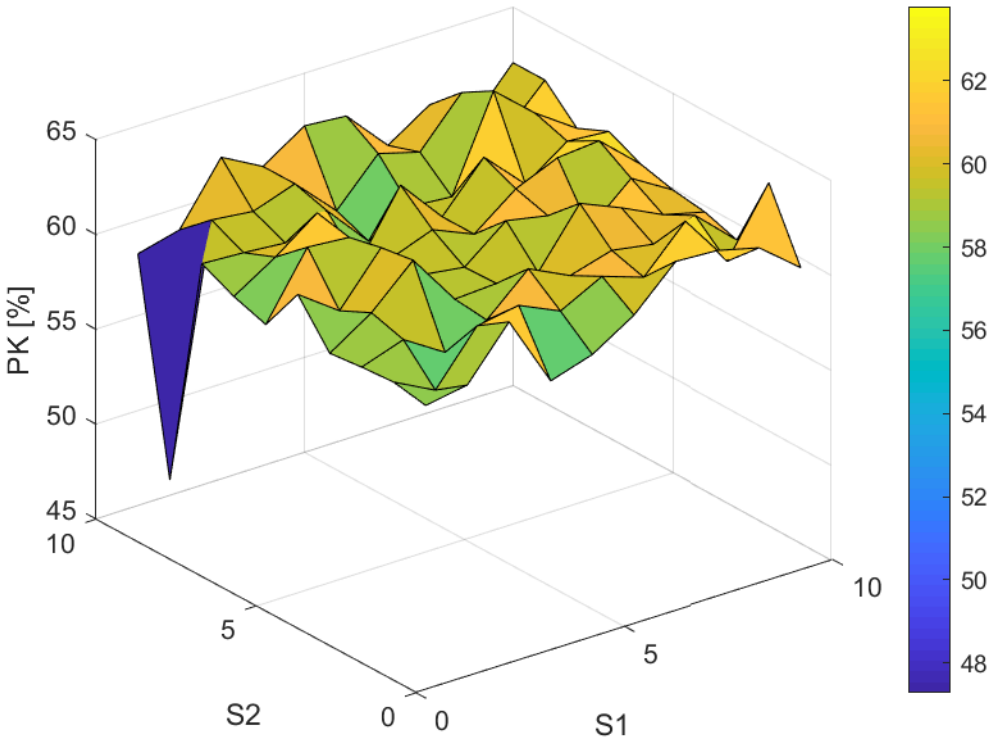
Na podstawie tego eksperymentu, zdecydowano na ustalenie maksymalnej liczby epok w dalszych eksperymentach na 30 000.

5.3. Badania przesiewowe dla małej ilości neuronów

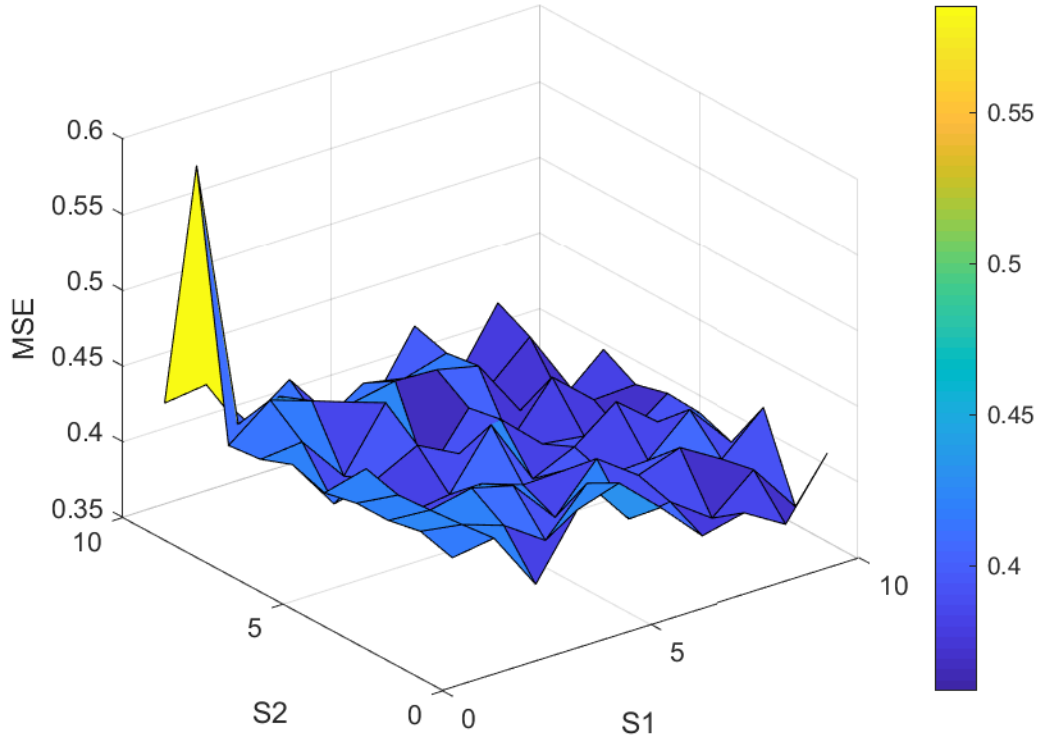
Na podstawie poprzednich eksperymentów, w tym eksperymencie ustawiono maksymalną liczbę epok na 30 000, parametr *max_fail* na 5% maksymalnej liczby epok, czyli 1500. Pozostałe parametry sieci zostały ustawione na domyślne, tj. współczynnik uczenia na 0.01, współczynnik zwiększania wartości współczynnika uczenia na 1.05, współczynnik zmniejszania wartości współczynnika uczenia na 0.7, dopuszczalną krotność przyrostu błędu na 1.04. Ilość neuronów w obu warstwach była zmieniana w zakresie od 1 do 10, z krokiem 1. Następnie badano, jaki wpływ na poprawność klasyfikacji i błąd średniokwadratowy ma mała liczba neuronów.

Utworzono pięciowymiarową macierz *PK_v5* o rozmiarze 10×10×1×1×1, do której zapisywano kolejne poprawności klasyfikacji oraz pięciowymiarową macierz *MSE_v5* o tym samym rozmiarze, do której zapisywano błąd średniokwadratowy dla sieci w aktualnej konfiguracji. Obie te macierze mogły zostać zredukowane do macierzy dwuwymiarowej, poprzez zastosowanie funkcji *squeeze*, która usuwa wymiary o długości 1.

Liczba neuronów w obu warstwach zmieniana była w zagnieżdżonych pętlach, po czym po wykonaniu jej, rysowano wykresy powierzchniowe *PK_v5* (rys. 5.6.) oraz *MSE_v5* (rys. 5.7.) w zależności od liczby neuronów w obu warstwach. Kod użyty do rysowania wykresów zawiera się w głównym listingu programu (listing 4.1.).



Rys. 5.6. Zależność poprawności klasyfikacji od liczby neuronów w warstwie pierwszej i drugiej.



Rys. 5.7. Zależność błędu średniokwadratowego od liczby neuronów w warstwie pierwszej i drugiej.

Obserwując powyższe wykresy oraz plik wynikowy eksperymentu można zauważyć, że dla małej liczby neuronów poprawność klasyfikacji zawiera się w większości w przedziale 58-63%, a błąd średniokwadratowy – w większości w przedziale 0.37-0.42

Największą poprawność klasyfikacji (63.79%) uzyskano dla 8 neuronów w warstwie pierwszej i drugiej. Błąd średniokwadratowy przy tej konfiguracji wynosił 0.373327.

Najmniejszy błąd średniokwadratowy (0.358708) został osiągnięty dla 10 neuronów w warstwie pierwszej i 9 neuronów w warstwie drugiej. Poprawność klasyfikacji przy tej konfiguracji wynosiła 62.04%.

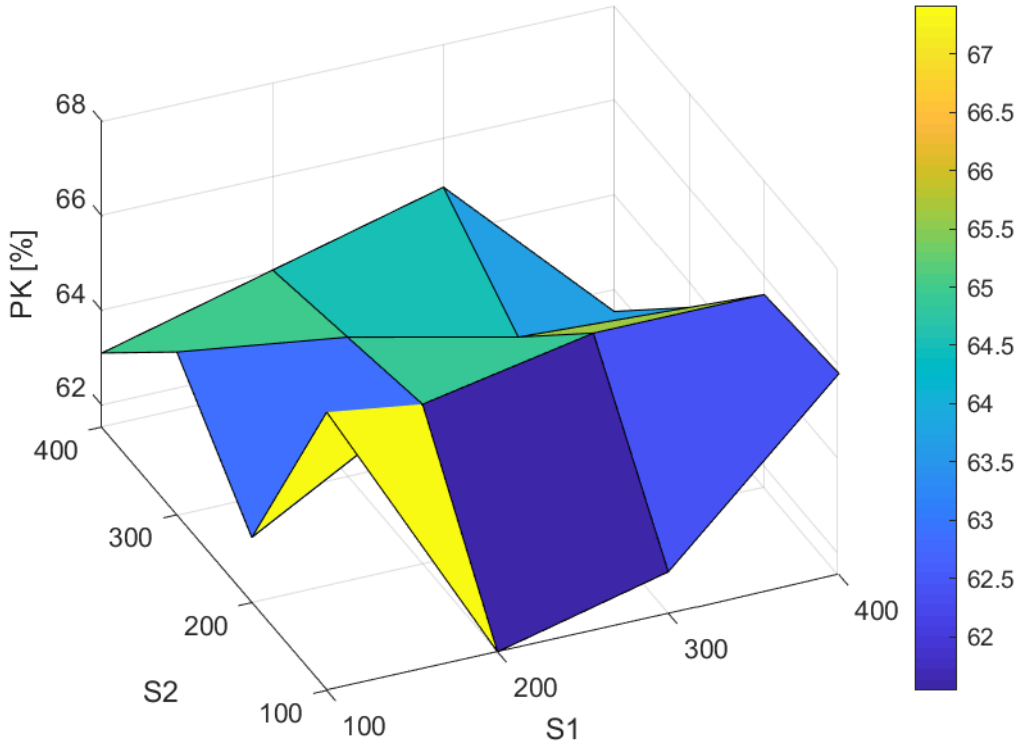
Najgorsze wyniki uzyskano przy 1 neuronie w warstwie pierwszej i 9 neuronach w warstwie drugiej. Dla tej konfiguracji poprawność klasyfikacji wyniosła 47.28%, a błąd średniokwadratowy wyniósł 0.585132.

5.4. Badania przesiewowe dla dużej ilości neuronów

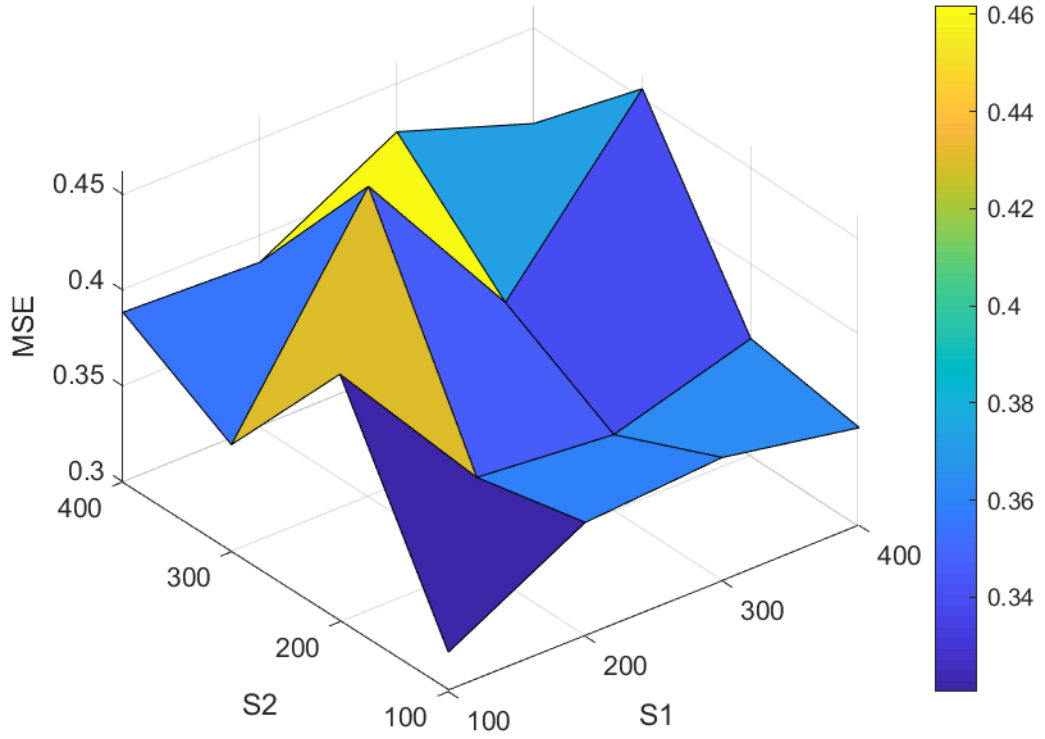
W tym eksperymencie wszystkie parametry oprócz współczynnika uczenia, który wynosił $5 \cdot 10^{-5}$ oraz liczby neuronów w obu warstwach były ustawione jak w poprzednim eksperymencie. Ilość neuronów w obu warstwach była zmieniana w zakresie od 100 do 400, z krokiem 100. Następnie badano, jaki wpływ na poprawność klasyfikacji i błąd średniokwadratowy ma duża liczba neuronów.

Utworzono pięciowymiarową macierz *PK_v5* o rozmiarze 4×4×1×1×1, do której zapisywano kolejne poprawności klasyfikacji oraz pięciowymiarową macierz *MSE_v5* o tym samym rozmiarze, do której zapisywano błąd średniokwadratowy dla sieci w aktualnej konfiguracji. Obie te macierze mogły zostać zredukowane do macierzy dwuwymiarowej, poprzez zastosowanie funkcji *squeeze*, która usuwa wymiary o długości 1.

Liczba neuronów w obu warstwach zmieniana była w zagnieżdżonych pętlach, po czym po wykonaniu jej, rysowano wykresy powierzchniowe *PK_v5* (rys. 5.8.) oraz *MSE_v5* (rys. 5.9.) w zależności od liczby neuronów w obu warstwach. Kod użyty do rysowania wykresów zawiera się w głównym listingu programu (listing 4.1.).



Rys. 5.8. Zależność poprawności klasyfikacji od liczby neuronów w warstwie pierwszej i drugiej.



Rys. 5.9. Zależność błędu średniokwadratowego od liczby neuronów w warstwie pierwszej i drugiej.

Obserwując powyższe wykresy oraz plik wynikowy eksperymentu można zauważyć, że dla dużej liczby neuronów poprawność klasyfikacji zawiera się w większości w przedziale 62-65%, a błąd średniokwadratowy – w większości w przedziale 0.34-0.42

Największą poprawność klasyfikacji (67.42%) uzyskano dla 100 neuronów w warstwie pierwszej i drugiej. Błąd średniokwadratowy przy tej konfiguracji wynosił 0.320591. Był to najmniejszy błąd uzyskany podczas tego eksperymentu.

Najgorszą poprawność klasyfikacji uzyskano dla dwóch konfiguracji: 200 i 100 neuronów oraz 400 i 400 neuronów odpowiednio w pierwszej i drugiej warstwie. Wynosiła ona 61.54%. Wartości błędu średniokwadratowego dla powyższych par neuronów wynosiły odpowiednio 0.359426 i 0.400275.

Największy błąd średniokwadratowy uzyskano dla 200 neuronów w warstwie pierwszej i 300 neuronów w warstwie drugiej. Wynosił on 0.461822. Poprawność klasyfikacji dla tej konfiguracji neuronów wynosiła 64.48%, a więc mieściła się w górnej części wyników otrzymanych podczas tego eksperymentu.

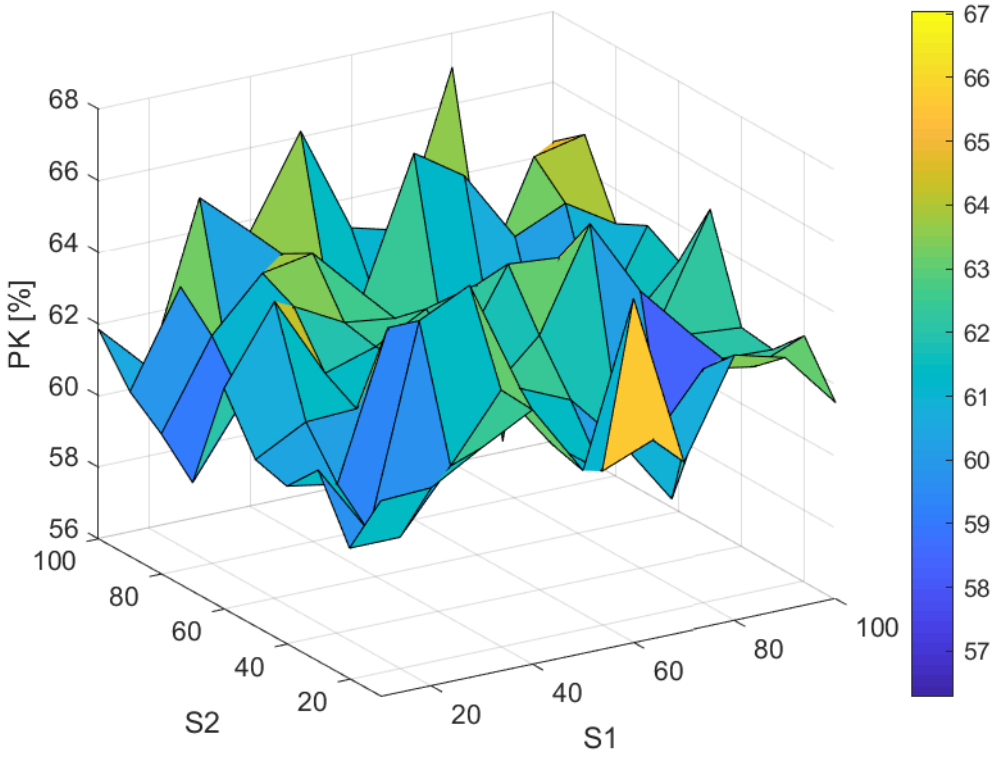
Można wyprowadzić z tego wniosek, iż wysoki błąd średniokwadratowy nie zawsze musi rzutować na niską poprawność klasyfikacji. Wniosek ten zostanie rozwinięty w punkcie 6., dotyczącym wniosków i spostrzeżeń.

5.5. Badania przesiewowe dla średniej ilości neuronów

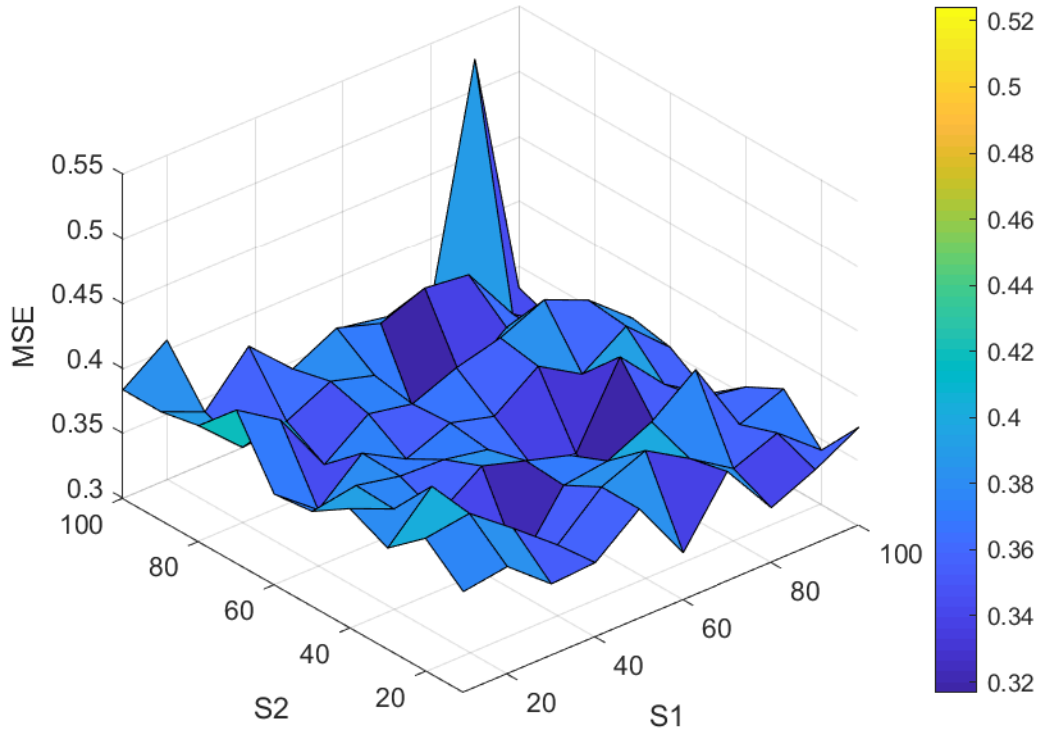
W tym eksperymencie wszystkie parametry oprócz współczynnika uczenia, który wynosił 0.01 oraz liczby neuronów w obu warstwach były ustawione jak w poprzednim eksperymencie. Ilość neuronów w obu warstwach była zmieniana w zakresie od 10 do 100, z krokiem 10. Następnie badano, jaki wpływ na poprawność klasyfikacji i błąd średniokwadratowy ma średnia liczba neuronów.

Utworzono pięciowymiarową macierz *PK_v5* o rozmiarze 10×10×1×1×1, do której zapisywano kolejne poprawności klasyfikacji oraz pięciowymiarową macierz *MSE_v5* o tym samym rozmiarze, do której zapisywano błąd średniokwadratowy dla sieci w aktualnej konfiguracji. Obie te macierze mogły zostać zredukowane do macierzy dwuwymiarowej, poprzez zastosowanie funkcji *squeeze*, która usuwa wymiary o długości 1.

Liczba neuronów w obu warstwach zmieniana była w zagnieżdżonych pętlach, po czym po wykonaniu jej, rysowano wykresy powierzchniowe *PK_v5* (rys. 5.10.) oraz *MSE_v5* (rys. 5.11.) w zależności od liczby neuronów w obu warstwach. Kod użyty do rysowania wykresów zawiera się w głównym listingu programu (listing 4.1.).



Rys. 5.10. Zależność poprawności klasyfikacji od liczby neuronów w warstwie pierwszej i drugiej.



Rys. 5.11. Zależność błędu średniokwadratowego od liczby neuronów w warstwie pierwszej i drugiej.

Obserwując powyższe wykresy oraz plik wynikowy eksperymentu można zauważyć, że dla dużej liczby neuronów poprawność klasyfikacji zawiera się w większości w przedziale 60-66%, a błąd średniokwadratowy – w większości w przedziale 0.32-0.38.

Największą poprawność klasyfikacji (67.04%) uzyskano dla 80 neuronów w warstwie pierwszej i 100 neuronów w warstwie drugiej. Błąd średniokwadratowy przy tej konfiguracji wynosił 0.34962.

Najgorszą poprawność klasyfikacji uzyskano dla 90 neuronów w warstwie pierwszej i 100 neuronów w warstwie drugiej. Wynosiła ona 56.29%. Wartości błędu średniokwadratowego przy tej konfiguracji wyniósł 0.524213. Była to największa wartość błędu średniokwadratowego w tym eksperymencie.

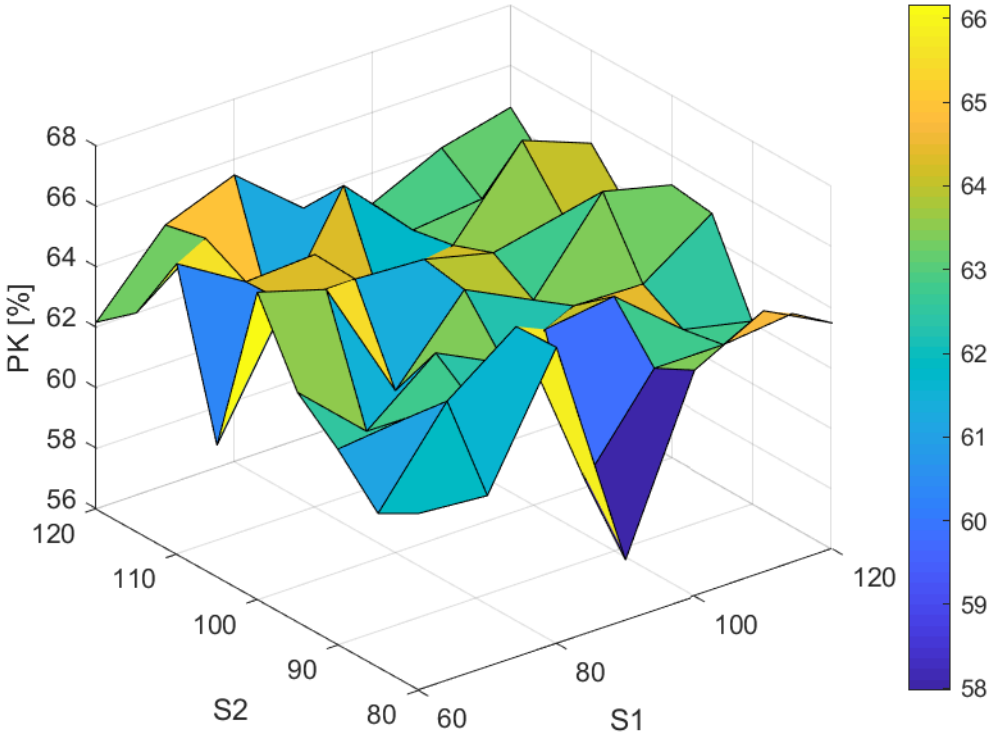
Najmniejszy błąd średniokwadratowy uzyskano dla 100 neuronów w obu warstwach. Wynosił on 0.333213. Poprawność klasyfikacji dla tej konfiguracji wynosiła 64.35%.

5.6. Badania w zakresie najbardziej efektywnych konfiguracji neuronów

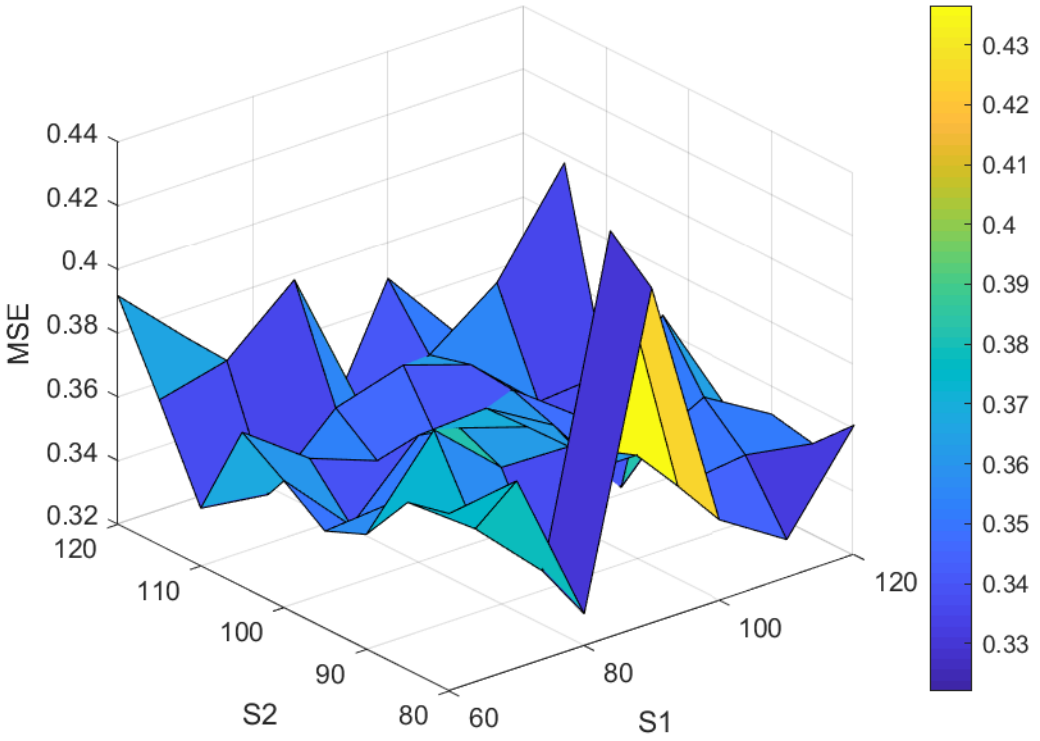
Na podstawie trzech poprzednich eksperymentów ustalono zakres neuronów, przy których sieć osiągała największą poprawność klasyfikacji i najmniejszy błąd średniokwadratowy. Zdecydowano się przeprowadzić dodatkowe badania dla ilości neuronów w pierwszej warstwie w zakresie od 60 do 120 z krokiem 10 i ilości neuronów w drugiej warstwie od 80 do 120 z krokiem 5. Współczynnik uczenia został ustawiony na 0.01. Wszystkie inne parametry były ustawione jak w poprzednich eksperymentach. Badano wpływ liczby neuronów w obu warstwach na poprawność klasyfikacji i błąd średniokwadratowy.

Utworzono pięciowymiarową macierz *PK_v5* o rozmiarze 7×9×1×1×1, do której zapisywano kolejne poprawności klasyfikacji oraz pięciowymiarową macierz *MSE_v5* o tym samym rozmiarze, do której zapisywano błąd średniokwadratowy dla sieci w aktualnej konfiguracji. Obie te macierze mogły zostać zredukowane do macierzy dwuwymiarowej, poprzez zastosowanie funkcji *squeeze*, która usuwa wymiary o długości 1.

Liczba neuronów w obu warstwach zmieniana była w zagnieżdżonych pętlach, po czym po wykonaniu jej, rysowano wykresy powierzchniowe *PK_v5* (rys. 5.12.) oraz *MSE_v5* (rys. 5.13.) w zależności od liczby neuronów w obu warstwach. Kod użyty do rysowania wykresów zawiera się w głównym listingu programu (listing 4.1.).



Rys. 5.12. Zależność poprawności klasyfikacji od liczby neuronów w warstwie pierwszej i drugiej.



Rys. 5.13. Zależność błędu średniokwadratowego od liczby neuronów w warstwie pierwszej i drugiej.

Obserwując powyższe wykresy oraz plik wynikowy eksperymentu można zauważyć, że dla dużej liczby neuronów poprawność klasyfikacji zawiera się w większości w przedziale 61-65%, a błąd średniokwadratowy – w większości w przedziale 0.33-0.39.

Największą poprawność klasyfikacji (66.17%) uzyskano dla 60 neuronów w warstwie pierwszej i 100 neuronów w warstwie drugiej. Błąd średniokwadratowy przy tej konfiguracji wynosił 0.359861.

Najgorszą poprawność klasyfikacji uzyskano dla 90 neuronów w warstwie pierwszej i 80 neuronów w warstwie drugiej. Wynosiła ona 57.97%. Wartości błędu średniokwadratowego przy tej konfiguracji wyniósł 0.425096.

Najmniejszy błąd średniokwadratowy uzyskano dla 110 neuronów w warstwie pierwszej i 100 neuronów w warstwie drugiej. Wynosił on 0.322062. Poprawność klasyfikacji dla tej konfiguracji wynosiła 65.6 %.

Największy błąd średniokwadratowy uzyskano dla 90 neuronów w warstwie pierwszej i 85 neuronów w warstwie drugiej. Wynosił on 0.436533. Poprawność klasyfikacji dla tej konfiguracji neuronów wynosiła 59.85%,

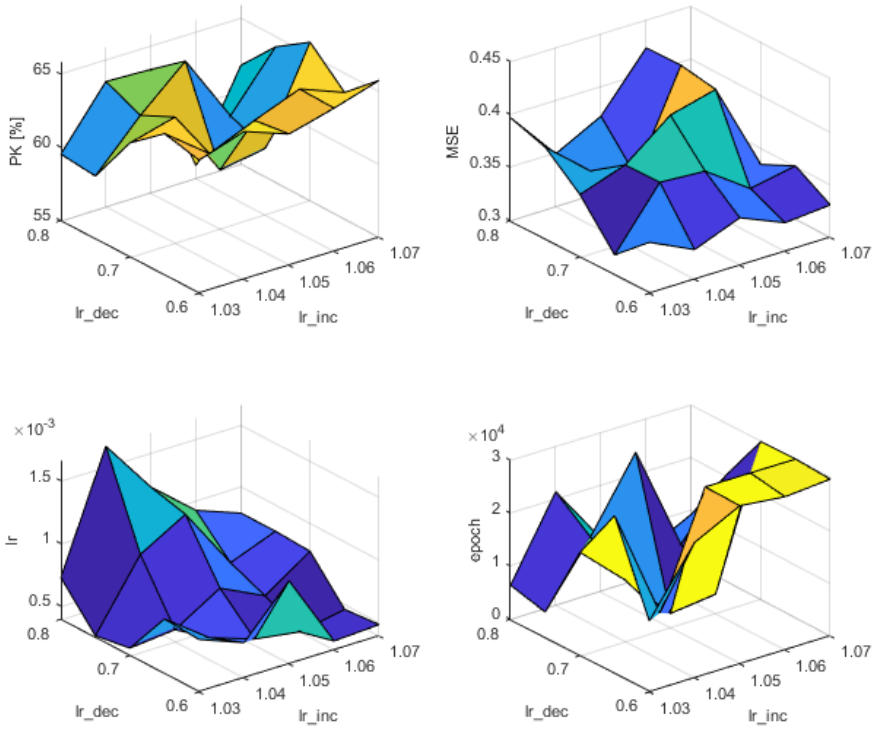
5.7. Wpływ parametrów adaptacyjnych na poprawność klasyfikacji, błąd średniokwadratowy i liczbę przebytych epok

Na podstawie poprzednich eksperymentów, ustawiono wartości neuronów w warstwach odpowiednio na 80 i 100 neuronów. Współczynnik uczenia został ustawiony na 0.001. W wykonanych eksperymentach (w szczególności w eksperymencie piątym) uzyskano dla tej konfiguracji poprawność klasyfikacji na poziomie ponad 67% i błąd średniokwadratowy na poziomie poniżej 0.35. Parametry adaptacyjne, czyli współczynniki zwiększania i zmniejszania wartości współczynnika uczenia oraz dopuszczalna krotność przyrostu błędu były odpowiednio zmieniane:

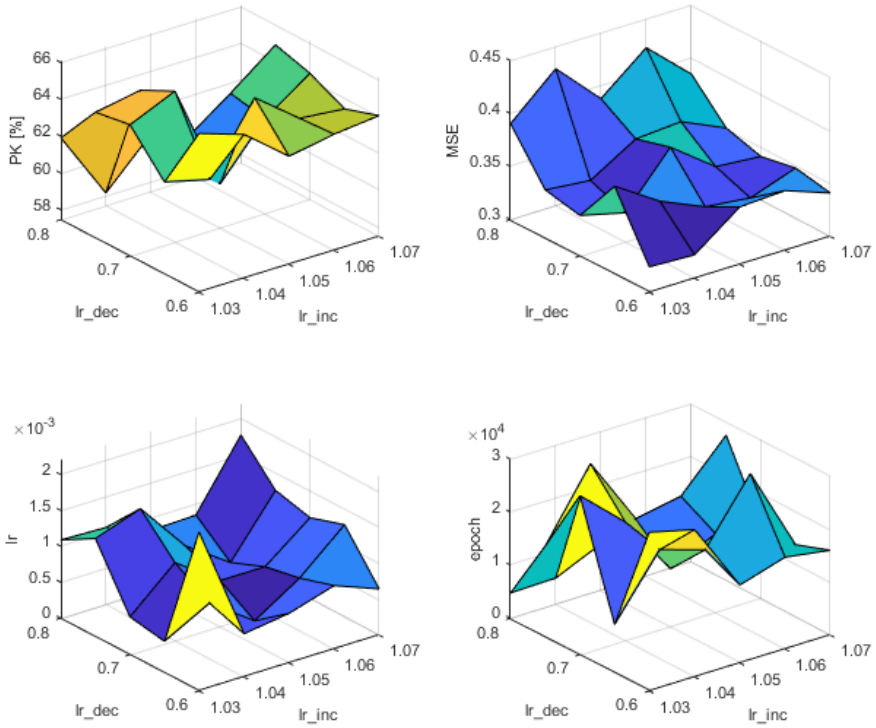
1) współczynnik zwiększania wartości współczynnika uczenia *lr_inc* zmieniany był w zakresie od 1.03 do 1.07 z krokiem 0.01,
2) współczynnik zmniejszania wartości współczynnika uczenia *lr_dec* zmieniany był w zakresie od 0.6 do 0.8 z krokiem 0.05,
3) dopuszczalna krotność przyrostu błędu *max_perf_inc* zmieniana był w zakresie od 1.02 do 1.06 z krokiem 0.01.

Utworzono pięciowymiarową macierz *PK_v5* o rozmiarze 1×1×5×5×5, do której zapisywano kolejne poprawności klasyfikacji, pięciowymiarową macierz *MSE_v5* o tym samym rozmiarze, do której zapisywano błąd średniokwadratowy dla sieci w aktualnej konfiguracji, macierz *EPOCH_v5* o tym samym rozmiarze, do której zapisywano najlepszą osiągniętą epokę, czyli taką, dla której był osiągany najmniejszy błąd średniokwadratowy oraz macierz *LR_v5* o tym samym rozmiarze, do której zapisywano współczynnik uczenia dla najlepszej epoki. Wszystkie te macierze mogły zostać zredukowane do macierzy trójwymiarowej, poprzez zastosowanie funkcji *squeeze*, która usuwa wymiary o długości 1.

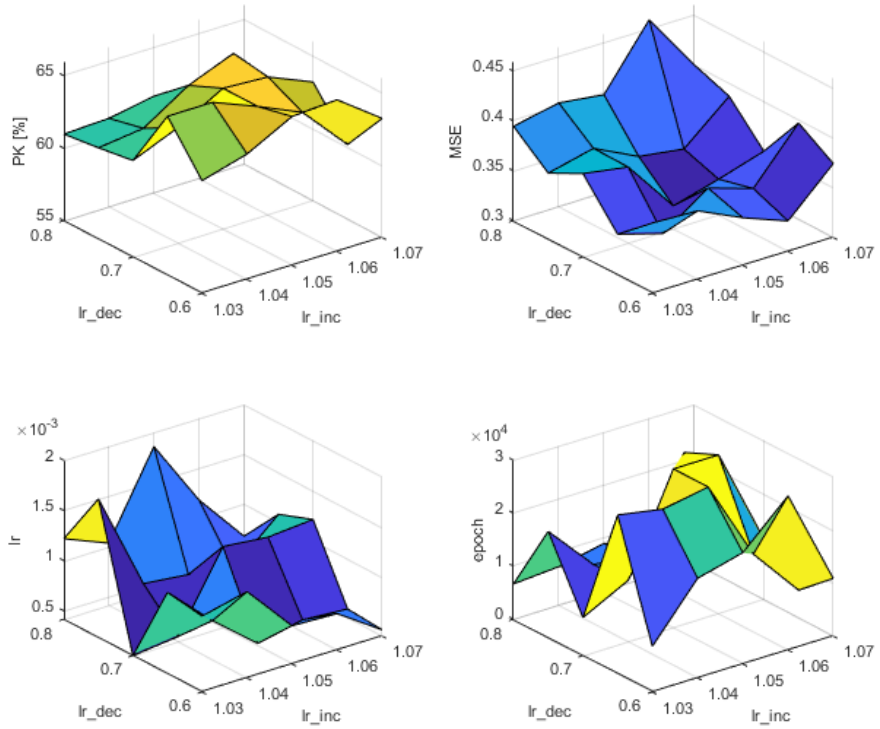
Parametry adaptacyjne zmieniane były w zagnieżdżonych pętlach, po czym po wykonaniu ich, narysowano pięć grup wykresów powierzchniowych *PK_v5, MSE_v5, LR_v5, EPOCH_v5* w zależności od współczynników zwiększania i zmniejszania wartości współczynnika uczenia (rys. 5.14. – 5.18.). Każda grupa przedstawiała powyższe zależności dla kolejnych wartości dopuszczalnej krotności przyrostu błędu. Na listingu 5.3. przedstawiono skrypt rysujący poniższe wykresy. Za każdym wywołaniem tego skryptu zmieniano parametr *curr_er_vec*, który odpowiadał za aktualny indeks w wektorze *er_vec*.



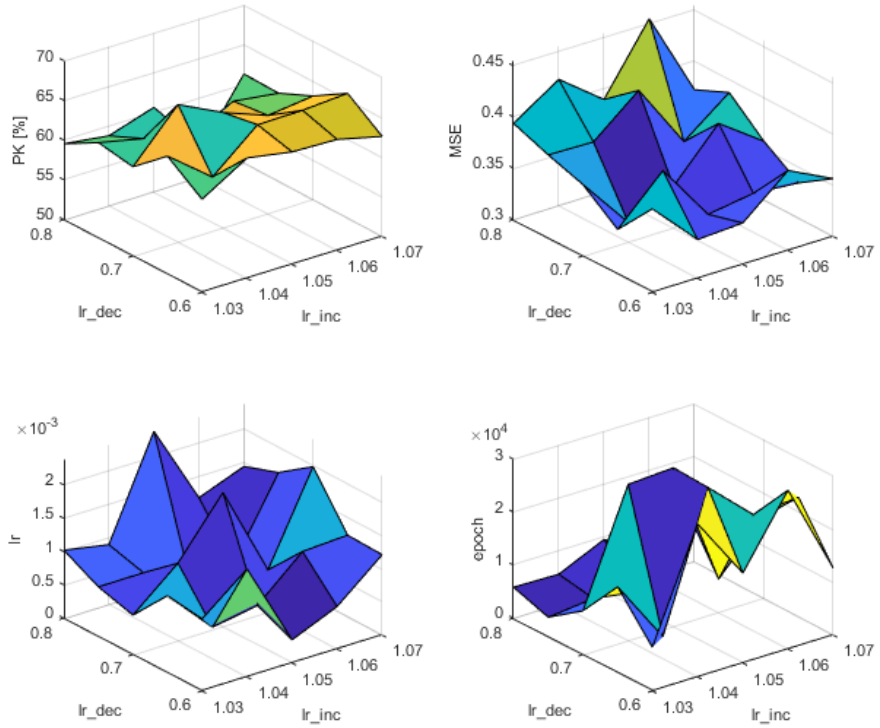
Rys. 5.14. Zależności *PK*, *MSE*, *LR*, *EPOCH* od *lr_inc* i *lr_dec* dla *max_perf_inc* = 1.02.



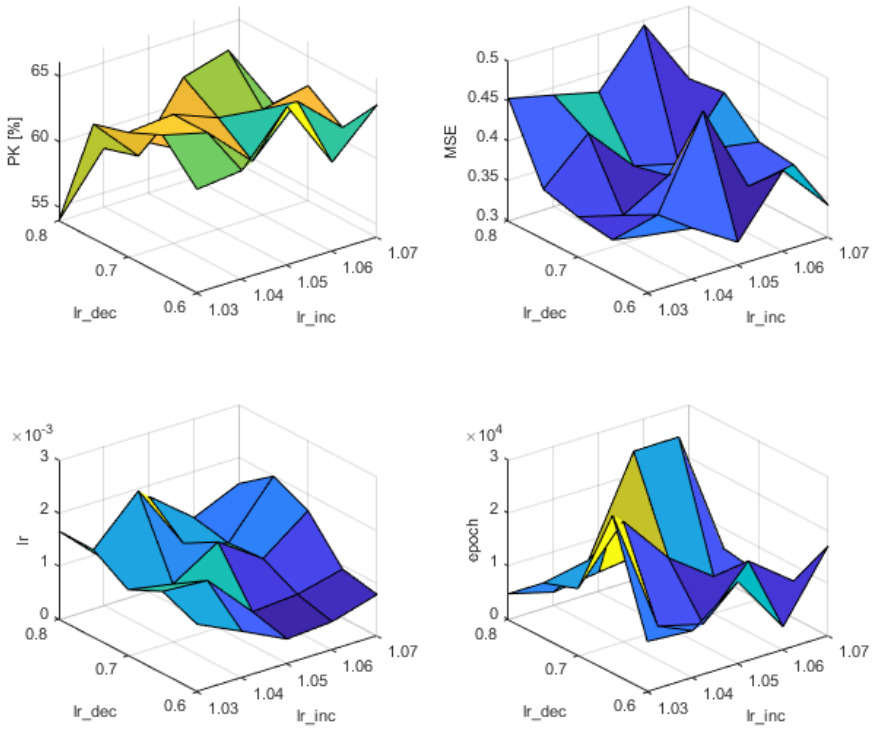
Rys. 5.15. Zależności *PK*, *MSE*, *LR*, *EPOCH* od *lr_inc* i *lr_dec* dla *max_perf_inc* = 1.03.



Rys. 5.16. Zależności *PK, MSE, LR, EPOCH* od *lr_inc* i *lr_dec* dla *max_perf_inc* = 1.04.



Rys. 5.17. Zależności *PK, MSE, LR, EPOCH* od *lr_inc* i *lr_dec* dla *max_perf_inc* = 1.05.



Rys. 5.18. Zależności *PK*, *MSE*, *LR*, *EPOCH* od *lr_inc* i *lr_dec* dla *max_perf_inc* = 1.06.

Listing 5.3. Rysowanie wykresów

```
close all;
figure('DefaultAxesFontSize',1);
set(gcf,'position',[100,100,700,576]);

curr_er_vec=1;

subplot(2,2,1)
surf(lr_inc_vec,lr_dec_vec,squeeze(PK_v5(1,1,:,:,curr_er_vec))')
xlabel('lr\_inc'); ylabel('lr\_dec'); zlabel('PK [%]');
axis([-inf inf -inf inf -inf inf]); axis 'auto z';

subplot(2,2,2)
surf(lr_inc_vec,lr_dec_vec,squeeze(MSE_v5(1,1,:,:,curr_er_vec))')
xlabel('lr\_inc'); ylabel('lr\_dec'); zlabel('MSE');
axis([-inf inf -inf inf -inf inf]); axis 'auto z';

subplot(2,2,3)
surf(lr_inc_vec,lr_dec_vec,squeeze(LR_v5(1,1,:,:,curr_er_vec))')
xlabel('lr\_inc'); ylabel('lr\_dec'); zlabel('lr');
axis([-inf inf -inf inf -inf inf]); axis 'auto z';

subplot(2,2,4)
surf(lr_inc_vec,lr_dec_vec,squeeze(EPOCH_v5(1,1,:,:,curr_er_vec))')
xlabel('lr\_inc'); ylabel('lr\_dec'); zlabel('epoch');
axis([-inf inf -inf inf -inf inf]); axis 'auto z';
```

Największą poprawność klasyfikacji (67.29%) uzyskano dla parametrów *lr_inc, lr_dec, max_perf_inc* równych odpowiednio 1.04, 0.7, 1.05. Błąd średniokwadratowy przy tej konfiguracji wynosił 0.319266, najlepszą epoką była epoka 29482, co oznacza, że uczenie nie zostało przerwane poprzez parametr *max_fail*. Współczynnik uczenia w tej epoce wynosił 0.0011638, co oznacza że stanowił on 116,38% współczynnika ustawionego na początku, tj. 0.001.

Najgorszą poprawność klasyfikacji (53.16%) uzyskano dla parametrów *lr_inc, lr_dec, max_perf_inc* równych odpowiednio 1.06, 0.8, 1.05. Błąd średniokwadratowy przy tej konfiguracji wynosił 0.455035, najlepszą epoką była epoka 4178, co oznacza, że uczenie zostało przerwane poprzez parametr *max_fail* po przebyciu ok. 14% zaplanowanej ilości epok. Współczynnik uczenia w tej epoce wynosił 0.00118079, co oznacza że stanowił on 118,79% współczynnika ustawionego na początku, tj. 0.001.

Najmniejszy błąd średniokwadratowy (0.319173) uzyskano dla parametrów *lr_inc, lr_dec, max_perf_inc* równych odpowiednio 1.03, 0.65, 1.02. Poprawność klasyfikacji przy tej konfiguracji wynosiła 64.67%, najlepszą epoką była epoka 29850, co oznacza, że uczenie nie zostało przerwane poprzez parametr *max_fail.* Współczynnik uczenia w tej epoce wynosił 0.000833166, co oznacza że stanowił on ok. 83,32% współczynnika ustawionego na początku, tj. 0.001.

Największy błąd średniokwadratowy (0.493483) uzyskano dla parametrów *lr_inc, lr_dec, max_perf_inc* równych odpowiednio 1.06, 0.8, 1.06. Poprawność klasyfikacji przy tej konfiguracji wynosiła 55.97%, najlepszą epoką była epoka 4652, co oznacza, że uczenie zostało przerwane poprzez parametr *max_fail* po przebyciu ok. 15.5% zaplanowanej ilości epok. Współczynnik uczenia w tej epoce wynosił 0.00112657, co oznacza że stanowił on ok. 112,66% współczynnika ustawionego na początku, tj. 0.001.

Najgorsze wyniki w tym eksperymencie występowały wówczas, gdy uczenie było przedwcześnie przerywane. Zestaw narzędzi do uczenia sieci neuronowych w środowisku MATLAB stosuje przedwczesne przerywanie procesu uczenia w celu eliminacji ilości obliczeń, które według niego nie posiadają perspektyw na polepszenie wyniku. Można ten efekt zredukować lub całkiem usunąć, wyłączając walidację w sieci neuronowej. Może to jednak prowadzić do nadmiarowych obliczeń, tak jak w tym eksperymencie, sieć nie dokonała polepszenia wyniku po ok. 15% zaplanowanych epok. Gdyby uczenie było kontynuowane, prawdopodobnie nie uzyskano by lepszego wyniku.

5.8. Sprawdzenie poprawności programu poprzez wykorzystanie innego zbioru danych

5.9. Badania dla bardzo dużej ilości neuronów bez przedwczesnego przerywania procesu uczenia

6. Wnioski i spostrzeżenia

Ponad 50% objętości pracy – część autorska:

a) założenia – dane,
b) opis zastosowanej metody rozwiązania lub analizy,
c) opis proponowanego rozwiązania, wyniki analizy teoretycznej, obliczenia, projekt konstrukcyjny, procesowy, technologiczny,
d) wyniki badań analitycznych, symulacyjnych lub eksperymentalnych itp.

7. Literatura

[1] http://weii.portal.prz.edu.pl/pl/materialy-do-pobrania. Dostęp 5.01.2015.

[2] Jakubczyk T., Klette A.: Pomiary w akustyce. WNT, Warszawa 1997.

[3] Barski S.: Modele transmitancji. Elektronika praktyczna, nr 7/2011, str. 15-18.

[4] Czujnik S200. Dokumentacja techniczno-ruchowa. Lumel, Zielona Góra. 2001.

[5] Pawluk K.: Jak pisać teksty techniczne poprawnie, Wiadomości Elektrotechniczne, Nr 12, 2001, str. 513-515.