Politechnika Białostocka

Sprawozdanie z pracowni specjalistycznej

Sztuczna inteligencja

Ćwiczenie numer: 1 – 2

Temat: Sieci Neuronowe

Wykonujący ćwiczenie: Wojciech Mojsiejuk

Studia dzienne
Kierunek: Informatyka
Semestr: IV
Grupa zajęciowa: PS01
Prowadzący ćwiczenie: dr inż. Maciej Kopczyński

.....
OCENA

12.03.2019
……..
Data wykonania ćwiczenia

……
Data i podpis prowadzącego

I. Cel i treść zadań

Celem ćwiczeń było zapoznanie się z koncepcją sztucznych sieci neuronowych, w tym zrozumienie podstawowych terminów takimi jak: neuron, funkcja aktywacji, warstwy ukryte, atrybuty warunkowe, atrybut decyzyjny. Nauczenie się przygotowywania danych oraz obróbka już istniejących plików, interpretacji wyników, modyfikowania sieci.

i. Zadanie 1

Zaproponować dane (najlepiej rzeczywiste) dotyczące klasyfikacji binarnej (dwie klasy decyzyjne), gdzie zbiór obiektów U jest co najmniej dziesięcioelementowy, a zbiór atrybutów A co najmniej trzyelementowy. Na zaproponowanej tablicy decyzyjnej wykonać następujące zadania:

a. Wybrać losowo trzy obiekty (nowe obiekty).
b. Dokonać losowego podziału zbioru powstałego poprzez usunięcie obiektów z punktu a) na cześć treningową i testową w stosunku 2 : 1.
c. Na danych z punktu b) dokonać uczenia dowolnie skonstruowanej sieci.
d. Dokonać klasyfikacji nowych obiektów, tj. obiektów z punktu a) (Solve → CalculateOutput).
e. Powtórzyć dwukrotnie zadania z punktów c) i d), stosując za każdym razem inną dowolnie skonstruowaną sieć. W każdym z trzech eksperymentów ma być zachowany ten sam podział na cześć treningową i testową.
f. Dokonać oceny jakości klasyfikacji oraz wpływu sposobu konstrukcji sieci na wynik klasyfikacji.

ii. Zadanie 2

Dokonać klasyfikacji danych zaproponowanych przez prowadzącego zajęcia.

a. Wczytać zbiór danych (sieć o jednej warstwie ukrytej zawierającej jeden neuron).
b. Dokonać losowego podziału zbioru danych na cześć treningową i testową w stosunku 2 : 1.
c. Dokonać uczenia sieci. Sprawdzić wynik klasyfikacji.
d. Uczenie sieci powtórzyć przy następujących ustawieniach parametrów:
   - Trzy testy przy zastosowaniu pozostałych funkcji aktywacji.
   - Jeden test przy zastosowaniu normalizacji danych wejściowych (Neural Options→ Normalize Inputs).
   - Jeden test przy ustalonej indywidualnie stałej wartości parametru początkowego sieci neuronowej.
   - Jeden test przy zastosowaniu sieci neuronowej składającej się z jednej warstwy ukrytej zawierającej trzy neurony.
   - Jeden test przy zastosowaniu dowolnej sieci neuronowej (co najmniej dwie
   - warstwy ukryte).
e. Określić jak zmiany konstrukcji sieci wykonane w punktach i. – v. wpływają na wynik klasyfikacji.

II. Rozwiązanie zadań / Wyniki

i. Zadanie 1

Zaproponowanym zbiorem danych jest zbiór dotyczący uczestnictwa studentów w wykładach. Atrybutami warunkowymi będą godzina wykładu, dzień tygodnia w którym odbywa się wykład, ilość godzin snu w nocy poprzedzającej wykład oraz czas dojazdu na uczelnie (zbiór atrybutów **A = 4**). Argumentem decyzyjnym zaś będzie informacja zwrotna czy student pojawił się na danym wykładzie czy też nie. Dane zostały zebrane na próbie **n = 13** studentów z wykorzystaniem formularzy Google. Grupa wybrana różni się kierunkiem studiów, semestrem nauki oraz uczelnią, z czego większość jest przedstawicielami Politechniki Białostockiej. Po zebraniu informacji o wszystkich wykładach udało uzyskać się **U = 72** elementowy zbiór obiektów.

Tabele od Tab.1. do Tab.4. przedstawiają sposób w jaki argumenty warunkowe przedstawione jako dane tekstowe były konwertowane na dane liczbowe.

Tab.1. Konwersja dnia na dane liczbowe

Dzień tygodnia	Wartość parametru
poniedziałek	0.00
wtorek	0.25
środa	0.50
czwartek	0.75
piątek	1.00

Tab.2. Konwersja dojazdu na dane liczbowe

Czas dojazdu	Wartość parametru
<0,5) minut	0.00
<5,15) minut	0.25
<15,30) minut	0.50
<30,45) minut	0.75
<45,∞) minut	1.00

Tab.3. Konwersja godziny na dane liczbowe

Godzina rozpoczęcia wykładu	Wartość parametru
8:00	0.0
10:00	0.2
12:00	0.4
14:00	0.6
16:00	0.8
18:00	1.0

Tab.4. Konwersja ilości snu na dane liczbowe

Ilość snu	Wartość parametru
<0,2) h	0.0
<2,4) h	0.2
<4,6) h	0.4
<6,8) h	0.6
<8,10) h	0.8
<10,∞) h	1.0

W przypadku argumentu decyzyjnego zastosowano klasyfikację binarną, przedstawia ją Tab.5.

Tab.5. Konwersja argumentu decyzyjnego na dane liczbowe

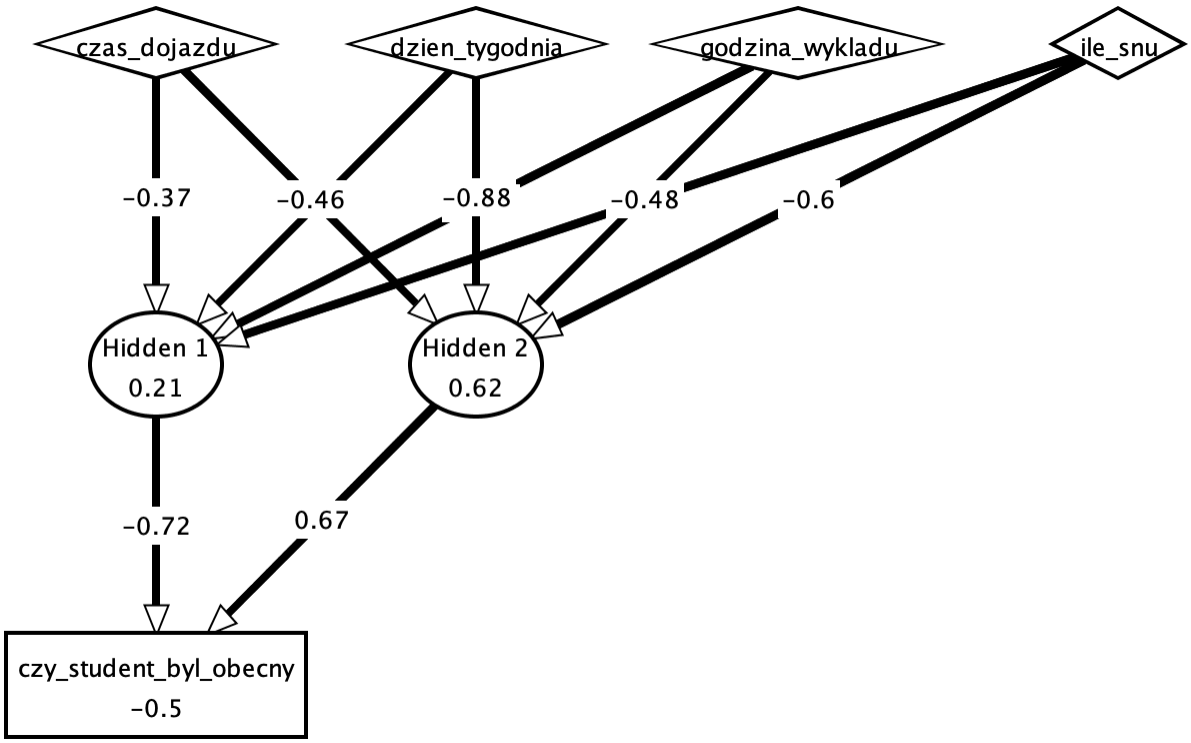
Czy student pojawił się na wykładzie	Wartość parametru
Tak	1.0
Nie	0.0

Na podstawie konwersji przedstawionych w tabelach Tab.1. - Tab.5. utworzony został skrypt w języku Python, pozwalający zmienić plik csv wygenerowany przez formularz Google, na plik odpowiedni do uczenia sztucznej sieci neuronowej. Kod programu znajduje się na Rys.1.

```
import pandas as pd
df = pd.read_csv("in.csv")
df.loc[df['Ile wynosi średni czas dotarcia przez ciebie na uczelnię?'] == '<0,5) minut','Ile wynosi średni czas dotarcia przez ciebie na uczelnię?']=0
df.loc[df['Ile wynosi średni czas dotarcia przez ciebie na uczelnię?'] == '<5,15) minut','Ile wynosi średni czas dotarcia przez ciebie na uczelnię?']=0.25
df.loc[df['Ile wynosi średni czas dotarcia przez ciebie na uczelnię?'] == '<15,30) minut','Ile wynosi średni czas dotarcia przez ciebie na uczelnię?']=0.5
df.loc[df['Ile wynosi średni czas dotarcia przez ciebie na uczelnię?'] == '<30,45) minut','Ile wynosi średni czas dotarcia przez ciebie na uczelnię?']=0.75
df.loc[df['Ile wynosi średni czas dotarcia przez ciebie na uczelnię?'] == '<45,∞) minut','Ile wynosi średni czas dotarcia przez ciebie na uczelnię?']=1
df.loc[df['Którego dnia tygodnia odbywa się wykład?'] == 'Poniedziałek','Którego dnia tygodnia odbywa się wykład?']=0
df.loc[df['Którego dnia tygodnia odbywa się wykład?'] == 'Wtorek','Którego dnia tygodnia odbywa się wykład?']=0.25
df.loc[df['Którego dnia tygodnia odbywa się wykład?'] == 'Środa','Którego dnia tygodnia odbywa się wykład?']=0.5
df.loc[df['Którego dnia tygodnia odbywa się wykład?'] == 'Czwartek','Którego dnia tygodnia odbywa się wykład?']=0.75
df.loc[df['Którego dnia tygodnia odbywa się wykład?'] == 'Piątek','Którego dnia tygodnia odbywa się wykład?']=1
df.loc[df['O której godzinie rozpoczyna się wykład?'] == 'ok. 8:00','O której godzinie rozpoczyna się wykład?']=0
df.loc[df['O której godzinie rozpoczyna się wykład?'] == 'ok. 10:00','O której godzinie rozpoczyna się wykład?']=0.2
df.loc[df['O której godzinie rozpoczyna się wykład?'] == 'ok. 12:00','O której godzinie rozpoczyna się wykład?']=0.4
df.loc[df['O której godzinie rozpoczyna się wykład?'] == 'ok. 14:00','O której godzinie rozpoczyna się wykład?']=0.6
df.loc[df['O której godzinie rozpoczyna się wykład?'] == 'ok. 16:00','O której godzinie rozpoczyna się wykład?']=0.8
df.loc[df['O której godzinie rozpoczyna się wykład?'] == 'ok. 18:00','O której godzinie rozpoczyna się wykład?']=1
df.loc[df['Ile godzin spałeś/aś w dniu poprzedzającym wykład'] == '<0,2) h','Ile godzin spałeś/aś w dniu poprzedzającym wykład']=0
df.loc[df['Ile godzin spałeś/aś w dniu poprzedzającym wykład'] == '<2,4) h','Ile godzin spałeś/aś w dniu poprzedzającym wykład']=0.2
df.loc[df['Ile godzin spałeś/aś w dniu poprzedzającym wykład'] == '<4,6) h','Ile godzin spałeś/aś w dniu poprzedzającym wykład']=0.4
df.loc[df['Ile godzin spałeś/aś w dniu poprzedzającym wykład'] == '<6,8) h','Ile godzin spałeś/aś w dniu poprzedzającym wykład']=0.6
df.loc[df['Ile godzin spałeś/aś w dniu poprzedzającym wykład'] == '<8,10) h','Ile godzin spałeś/aś w dniu poprzedzającym wykład']=0.8
df.loc[df['Ile godzin spałeś/aś w dniu poprzedzającym wykład'] == '<10,∞) h','Ile godzin spałeś/aś w dniu poprzedzającym wykład']=1
df.loc[df['Czy pojawiłeś się na danym wykładzie?'] == 'Nie','Czy pojawiłeś się na danym wykładzie?']=0
df.loc[df['Czy pojawiłeś się na danym wykładzie?'] == 'Tak','Czy pojawiłeś się na danym wykładzie?']=1
df = df.drop('Sygnatura czasowa', axis=1)
print(df)
df.to_csv('out.csv',index=False,header=False)
'''
HEADER:
Sygnatura czasowa,Ile wynosi średni czas dotarcia przez ciebie na uczelnię?,Którego dnia tygodnia odbywa się wykład?,O której godzinie rozpoczyna się wykład?,Ile godzin spałeś/aś w dniu poprzedzającym wykład,Czy pojawiłeś się na danym wykładzie?
'''
```

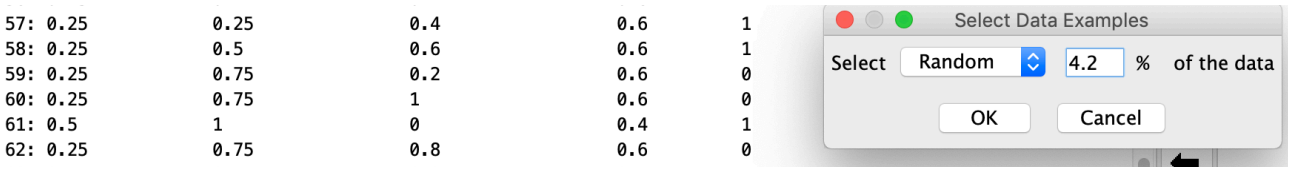
Rys.1. Skrypt zmieniający dane wygenerowane przez formularz Google na odpowiednie dane liczbowe

W taki sposób utworzone dane mogły zostać załadowane do programu AIspace Neural Networks. Rys.2. ukazuje domyślnie utworzoną przez program sieć neuronową zawierającą 1 ukrytą warstwę z 2 neuronami.



Rys.2. Schemat sztucznej sieci neuronowej wygenerowanej początkowo przez program

Ze zbioru obiektów zostały wyselekcjonowane w sposób losowy trzy obiekty, które następnie zostały usunięte ze zbioru. (Założenie: $\frac{3}{72} \approx 4.2\,\%$). Wybrane trzy obiekty znajdują się na Rys.3.



	czas_dojazdu	dzien_tygodnia	godzina_wykladu	ile_snu	czy_student_byl_obecny
1:	0.5	0	0.2	0.6	1
2:	1	0	1	0.4	0
3:	0.25	0	0.2	0.6	1

Rys.3. Wybrane 3 elementy ze zbioru 72 elementowego

Następnie pozostałe obiekty zostały losowo przydzielone na zbiór treningowy oraz zbiór testowy w stosunku 2:1 (Rys.4.), po czym przystąpiono do nauki sieci.

Training Examples (47)

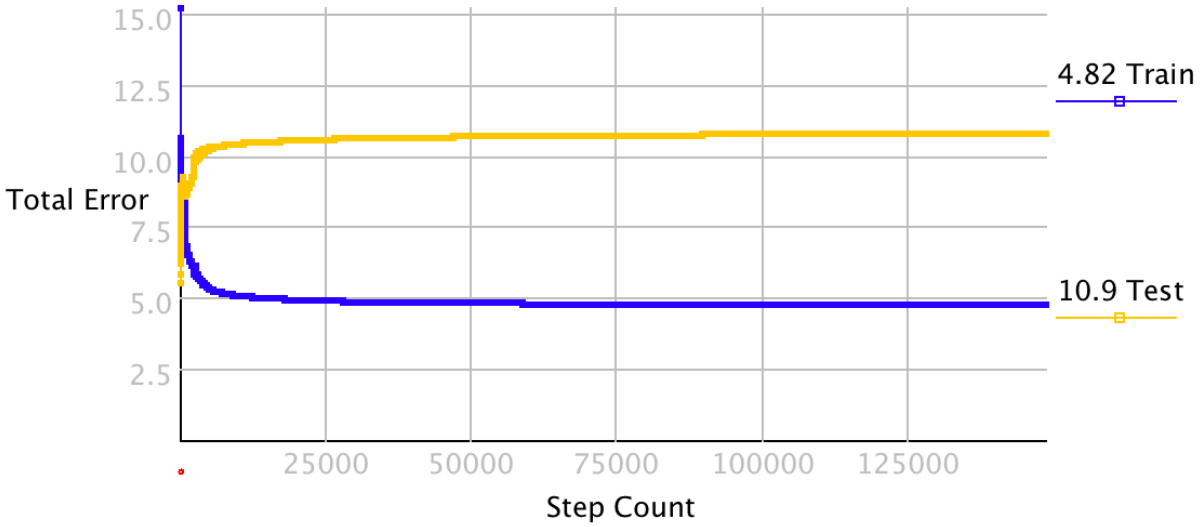
	czas_dojazdu	dzien_tygodnia	godzina_wykladu	ile_snu	czy_student_byl_obecny
1:	0.75	0	0.2	0.6	1
2:	0.75	0	1	0.6	1
3:	0	0	0.2	0.6	1
4:	0	1	0	0.4	1
5:	0	0.25	0	0.4	1
6:	0.5	0	1	0.6	0
7:	0.25	1	0	0.4	1
8:	1	1	0	0.6	0
9:	0.75	0.5	0.6	0.6	1
10:	0.75	0.5	0.8	0.6	0
11:	0.75	0.5	1	0.6	1
12:	0.25	0	0.8	0.6	1
13:	0.5	0.25	0.4	0.6	1
14:	0.5	0.5	0.6	0.6	1
15:	0.5	0.75	0.2	0.6	0
16:	0.5	0.75	1	0.6	1
17:	0	0	0.4	0.6	0
18:	0	0.5	0.6	0.6	0
19:	0	0.25	0.6	0.6	0
20:	0.75	0.25	0.4	0.6	1

Test Examples (22)

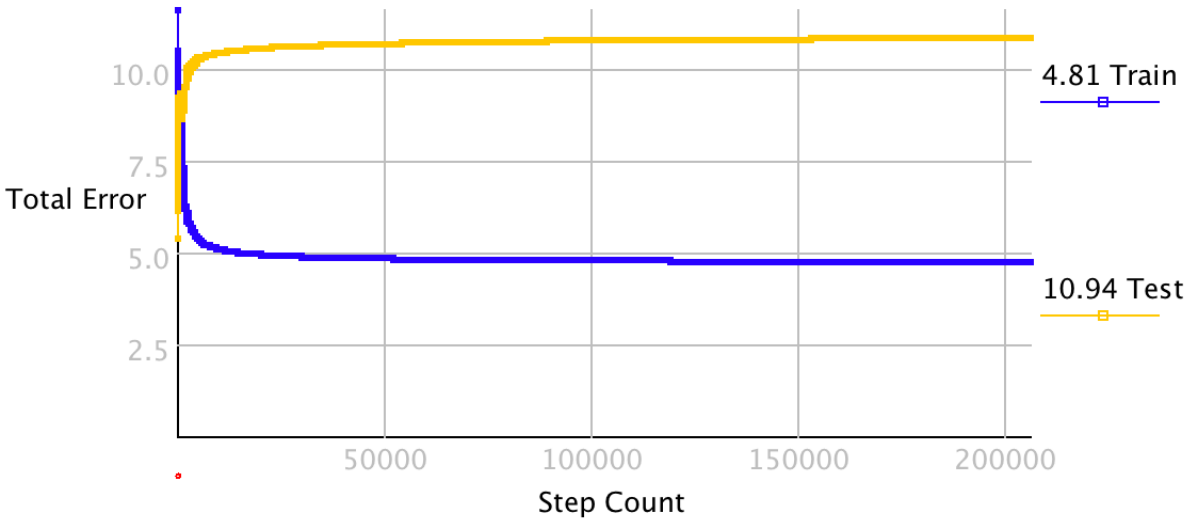
	czas_dojazdu	dzien_tygodnia	godzina_wykladu	ile_snu	czy_student_byl_obecny
1:	0.75	0.25	0.2	0	0
2:	0.75	0.5	0	1	0
3:	0.5	0	0.8	0.6	1
4:	0.25	0	1	0.6	1
5:	0.25	0	0	0.6	1
6:	0.25	0.25	1	0.6	1
7:	1	0	0.2	0.4	0
8:	0	0.25	0.2	0.6	0
9:	0	0.75	0.6	0.6	0
10:	0.25	0	0	0.6	1
11:	0	0	1	0.4	1
12:	0.25	0	0.2	0.6	1
13:	0.5	1	0	0.4	1
14:	0.25	0	0.8	0.6	0
15:	0.25	0	0.2	0.6	0
16:	0.25	0	1	0.6	0
17:	0	0.75	0.2	0.6	0
18:	0	1	0.2	0.6	0
19:	0.75	0	0	0.4	0
20:	0.25	0.75	0.2	0.6	0

Rys.4. Losowy podział obiektów na zbiór treningowy oraz na zbiór testowy

W celu minimalizacji przypadku niefortunnych parametrów wejściowych, proces uczenia powtórzono dwukrotnie (wykresy przedstawione na Rys.5. oraz Rys.6.).

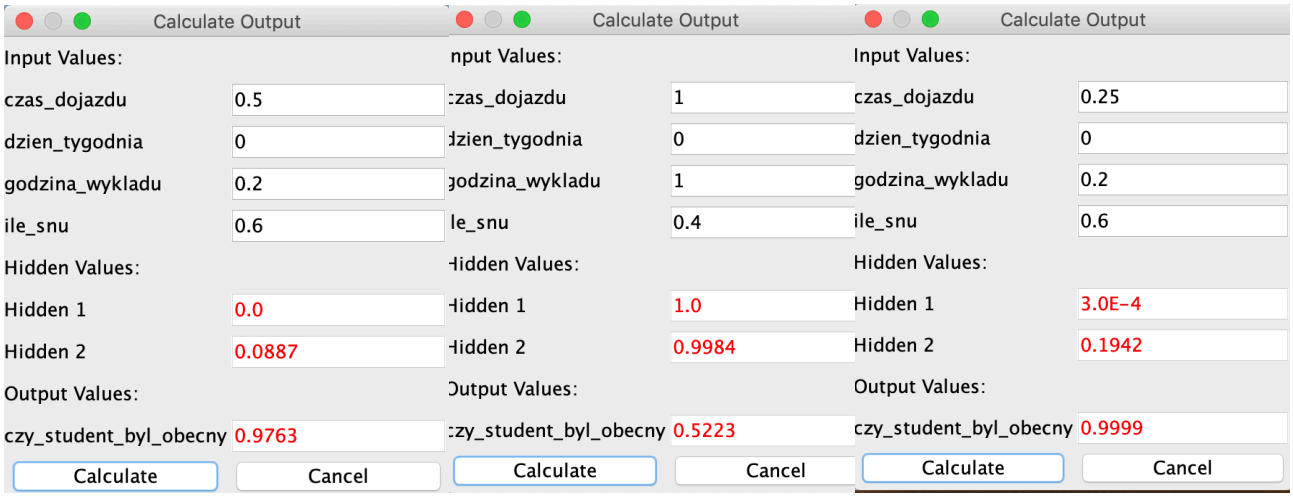


Rys.5. Wykres nauki sieci dla pierwszego podejścia

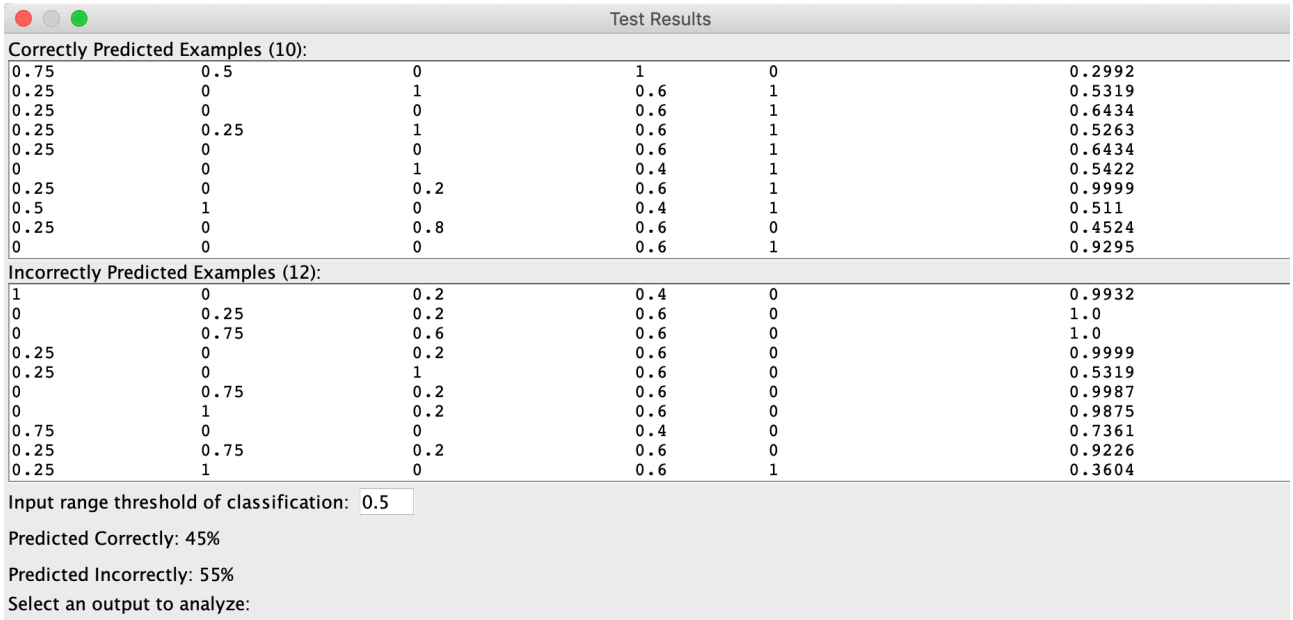


Rys.6. Wykres nauki sieci dla drugiego podejścia

Po czym spróbowano sklasyfikować obiekty z Rys.3., wynik klasyfikacji przedstawia Rys.7. zaś Rys.8. ukazuje statystyki podsumowujące wyniki testu.

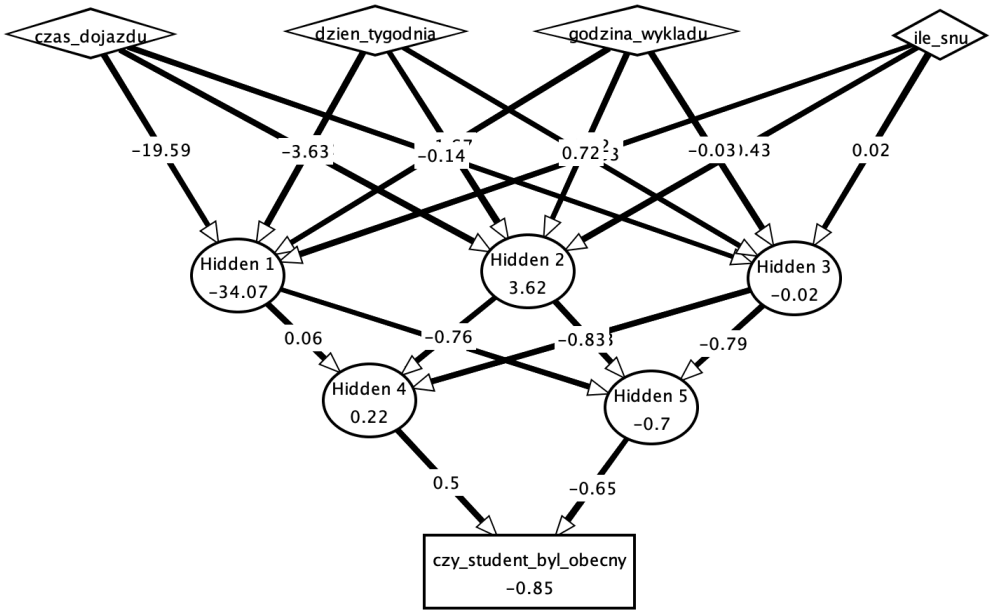


Rys.7. Klasyfikacja nowych obiektów



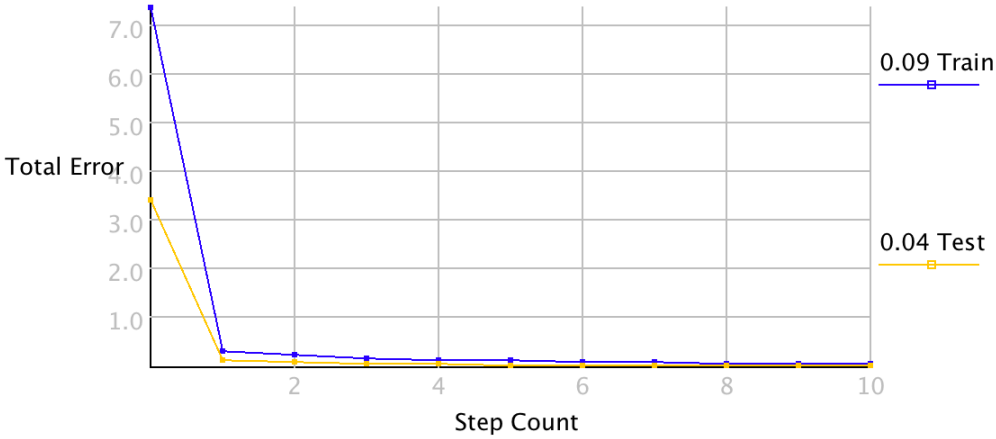
Rys.8. Wyniki testu

Po klasyfikacji obiektów przystąpiono do modyfikacji sieci neuronowej. Nowo utworzona sieć neuronowa ma 2 warstwy ukryte. W pierwszej warstwie ukrytej znajdują się 3 neurony, w drugiej zaś 2. Schemat utworzonej sieci został przedstawiony na Rys.9.

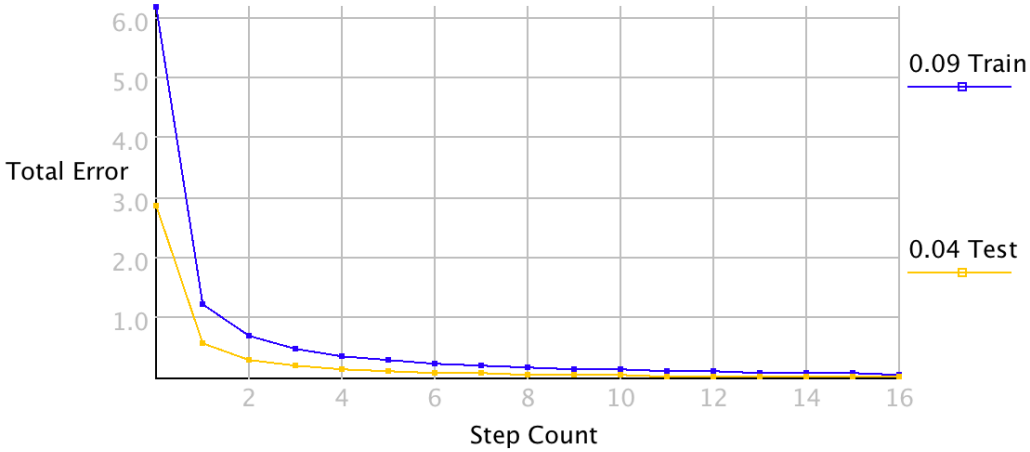


Rys.9. Schemat zaproponowanej sieci neuronowej

Przeprowadzono dwie próby nauczenia sieci (Rys.10. i Rys.11).

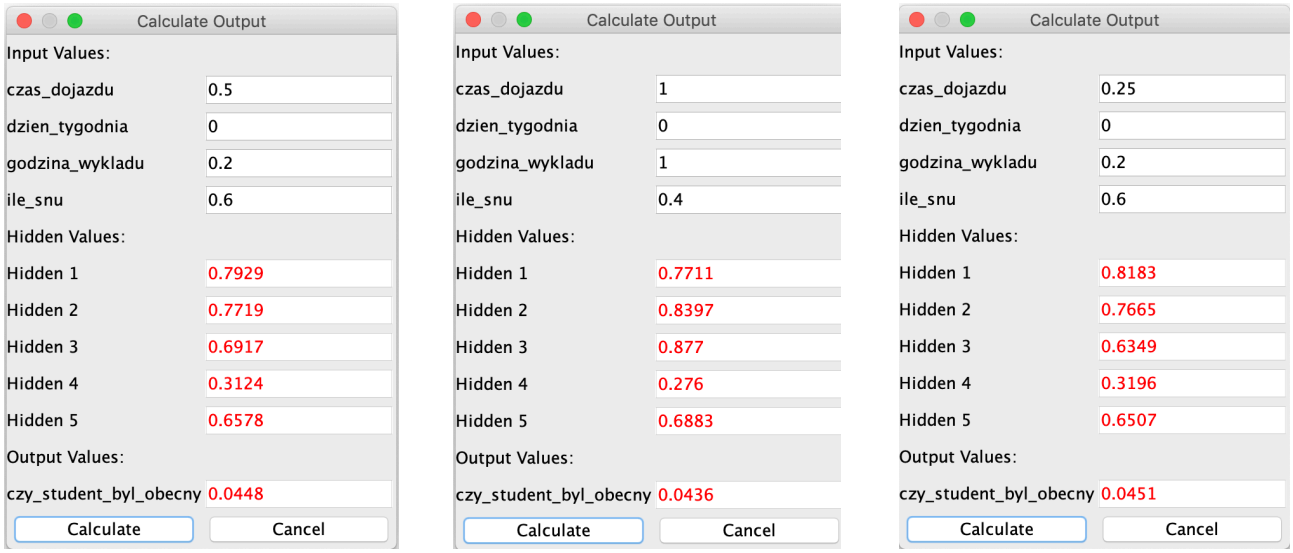


Rys.10. Wykres nauki sieci dla pierwszego podejścia

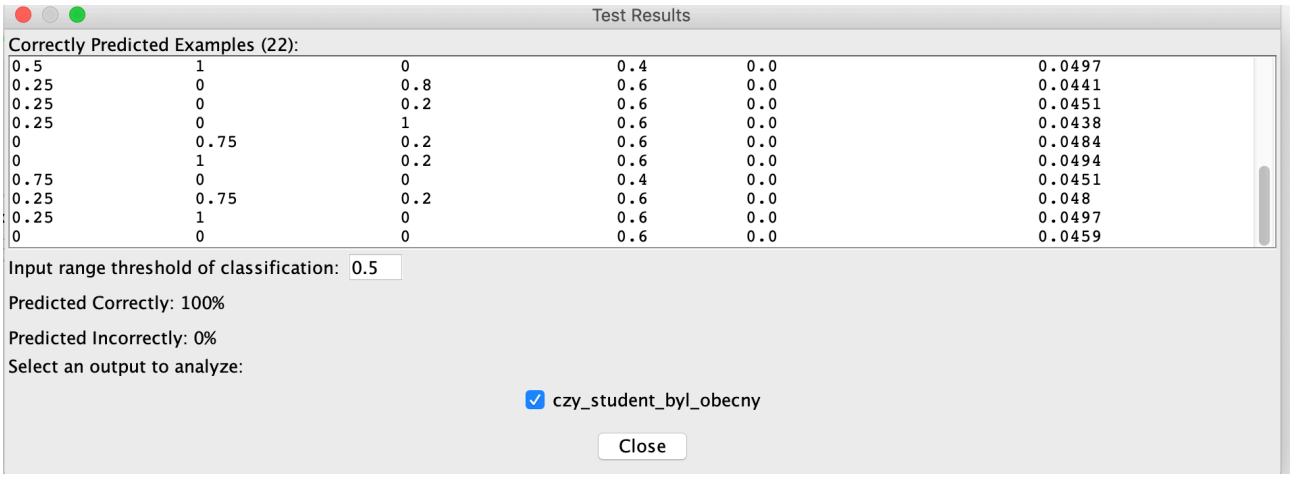


Rys.11. Wykres nauki sieci dla drugiego podejścia

Ponownie spróbowano sklasyfikować obiekty z Rys.3., wynik klasyfikacji przedstawia Rys. 12. zaś Rys.13. ukazuje statystyki podsumowujące wyniki testu.

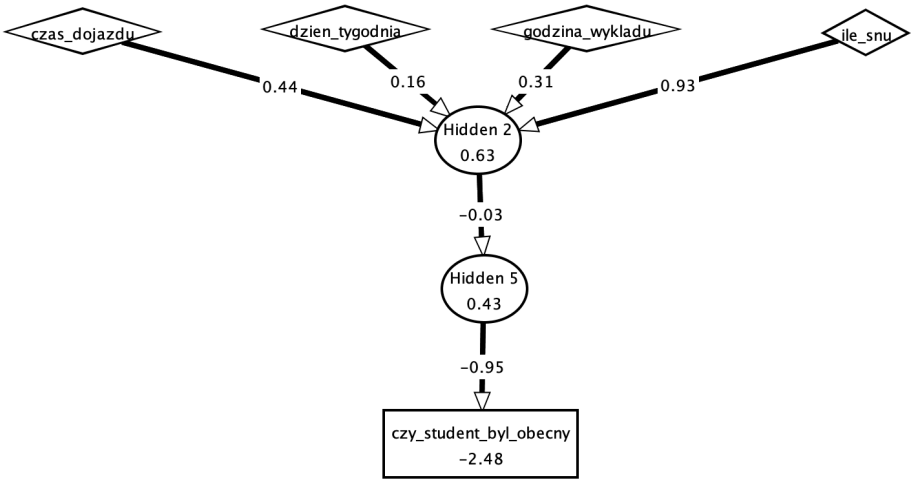


Rys.12. Klasyfikacja nowych obiektów



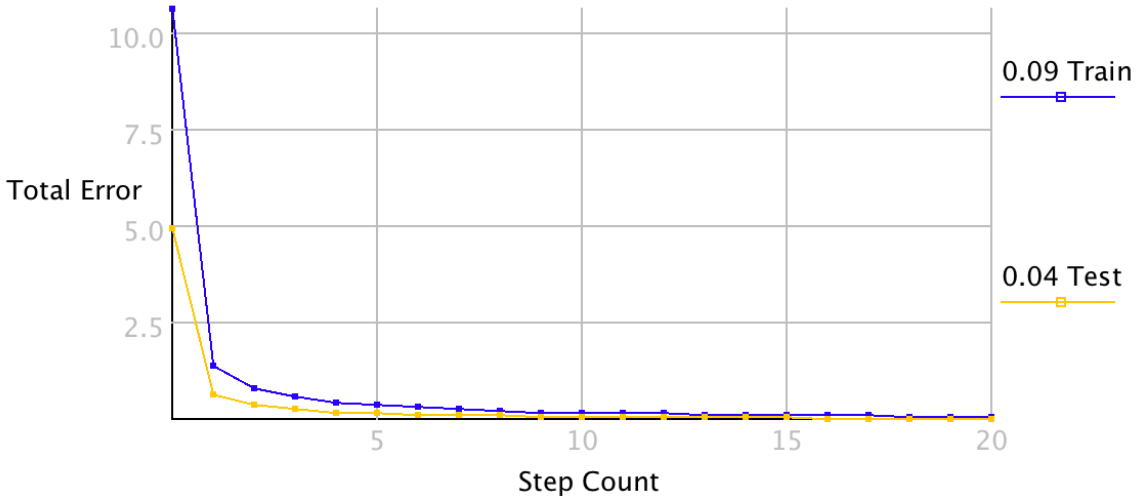
Rys.13. Wyniki testu

Po raz ostatni przystąpiono do modyfikacji sieci. Nowo utworzona sieć neuronowa ma 2 warstwy ukryte. Każda warstwa ma po 1 neuronie Schemat utworzonej sieci został przedstawiony na Rys.14.

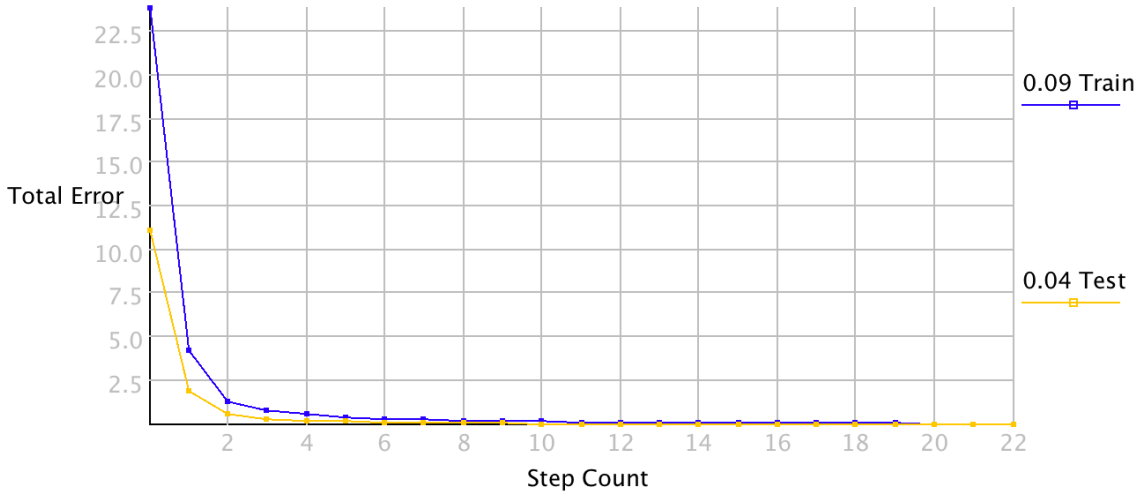


Rys.14. Schemat zaproponowanej sieci neuronowej

Przeprowadzono dwie próby nauczenia sieci (Rys.10. i Rys.11).

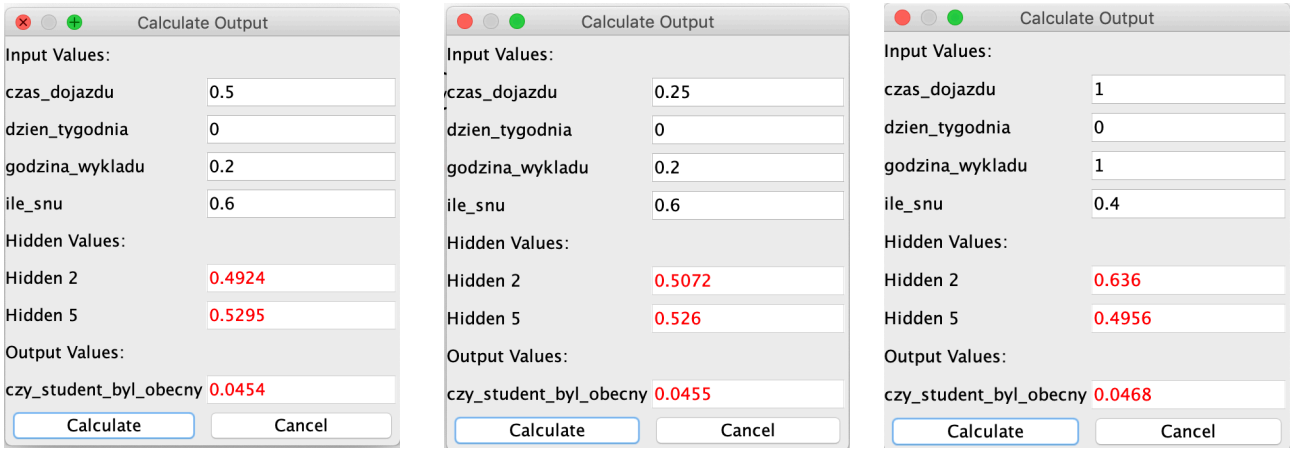


Rys.15. Wykres nauki sieci dla pierwszego podejścia

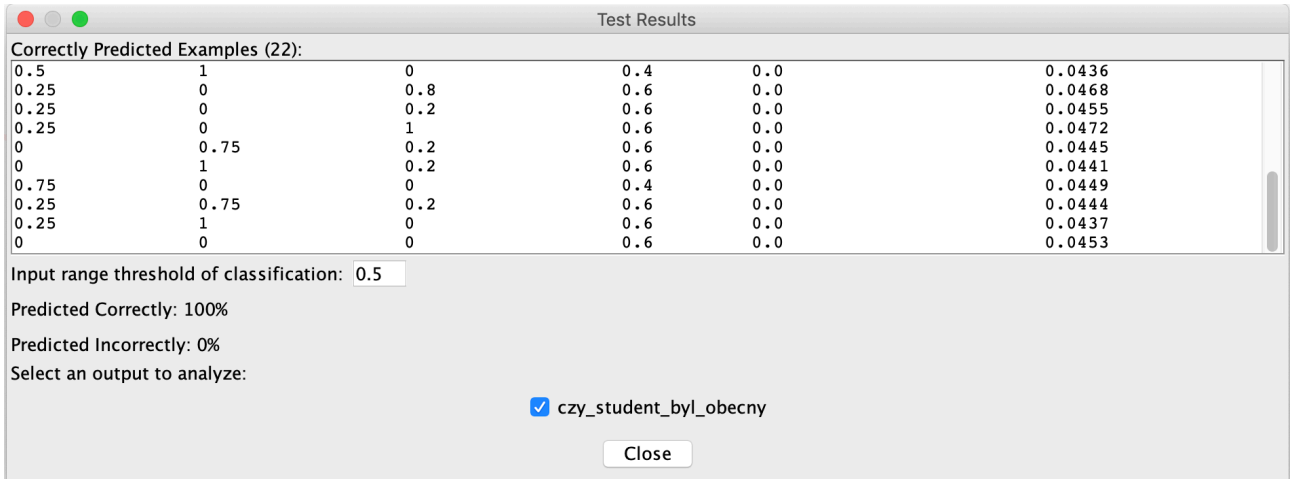


Rys.16. Wykres nauki sieci dla drugiego podejścia

Ponownie spróbowano sklasyfikować obiekty z Rys.3., wynik klasyfikacji przedstawia Rys. 17. zaś Rys.18. ukazuje statystyki podsumowujące wyniki testu.



Rys.17. Klasyfikacja nowych obiektów



Rys.18. Wyniki testu

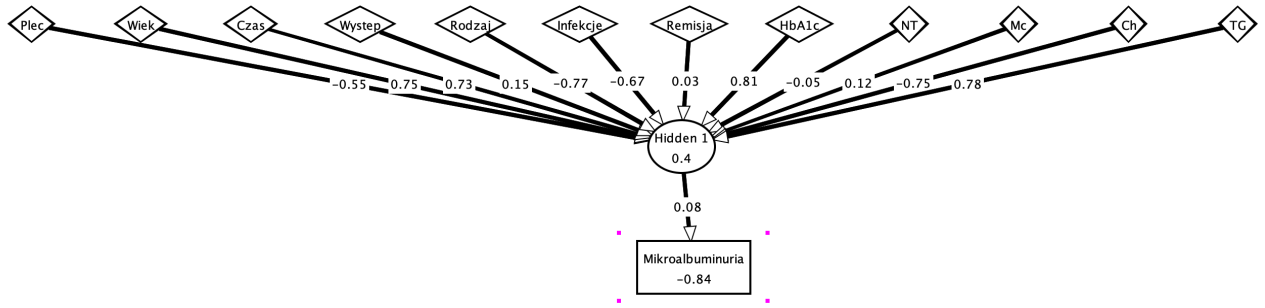
ii. Zadanie 2

Żeby móc wczytać dane wejściowe do programu AIspace Neural Networks muszą one zostać sprowadzone do reprezentacji liczbowej (sieć neuronowa uczy się tylko na podstawie typów liczbowych). W tym celu przy pomocy biblioteki pandas dane podane przez prowadzącego zostały przekształcone na typ liczbowy. Rys.19. przedstawia kod skryptu.

Przetworzone dane zostały załadowane do programu. W początkowym kreatorze wybrano opcje, żeby program wygenerował sieć o jednej warstwie ukrytej zawierającej jeden neuron. Schemat nowo utworzonej sieci przedstawia Rys.20.

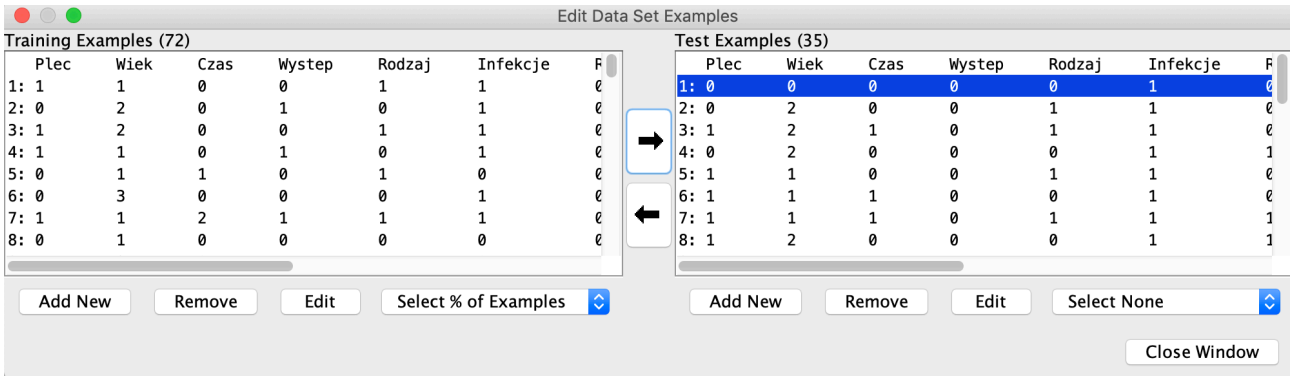
```
import pandas as pd
df = pd.read_csv("MK.csv")
df.loc[df['Plec'] == 'z','Plec']=1 #kobiety-1 mezczyzni-0
df.loc[df['Plec'] == 'm','Plec']=0
df.loc[df['Wystep'] == 'nie','Wystep']=0
df.loc[df['Wystep'] == 'tak','Wystep']=1
df.loc[df['Rodzaj'] == 'KIT_IIT','Rodzaj']=1
df.loc[df['Rodzaj'] == 'KIT','Rodzaj']=0
df.loc[df['Infekcje'] == 'tak','Infekcje']=1
df.loc[df['Infekcje'] == 'nie','Infekcje']=0
df.loc[df['Remisja'] == 'tak','Remisja']=1
df.loc[df['Remisja'] == 'nie','Remisja']=0
df.loc[df['NT'] == 'tak','NT']=1
df.loc[df['NT'] == 'nie','NT']=0
df.loc[df['Mc'] == '<3','Mc']=0
df.loc[df['Mc'] == '3-97','Mc']=1
df.loc[df['Mc'] == '>97','Mc']=2
df.loc[df['Ch'] == 'tak','Ch']=1
df.loc[df['Ch'] == 'nie','Ch']=0
df.loc[df['TG'] == 'tak','TG']=1
df.loc[df['TG'] == 'nie','TG']=0
df.loc[df['Mikroalbuminuria'] == 'tak','Mikroalbuminuria']=1
df.loc[df['Mikroalbuminuria'] == 'nie','Mikroalbuminuria']=0
print(df)
```

Rys.19. Skrypt zmieniający dane podane przez prowadzącego na odpowiednie dane liczbowe



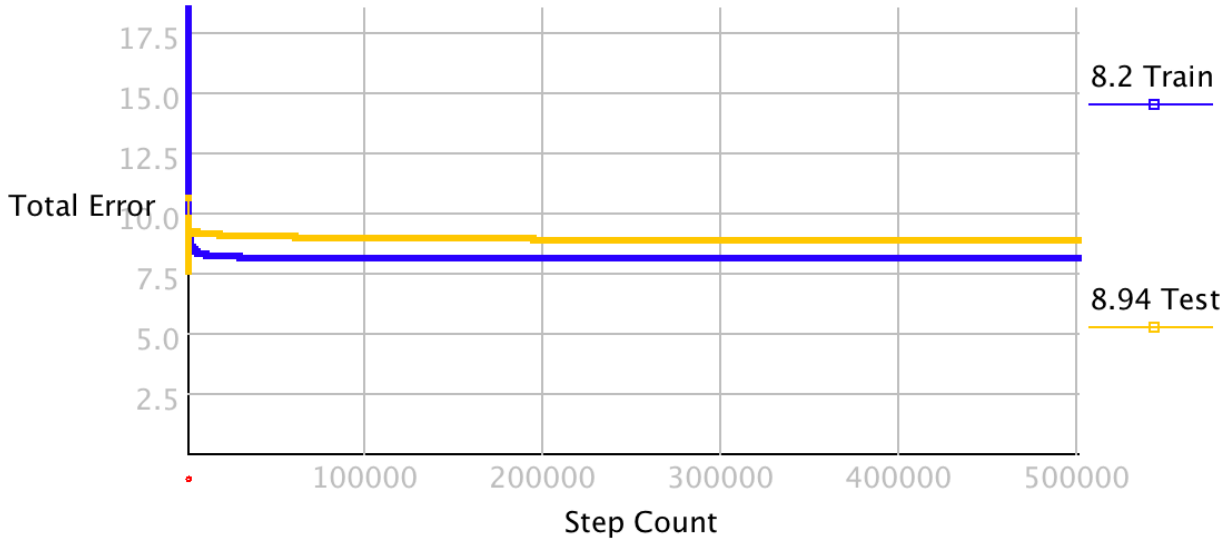
Rys.20. Schemat sztucznej sieci neuronowej wygenerowanej przez kreator programu

Następnie zbiór obiektów został podzielony na zbiór treningowy i testowy w stosunku 2:1.

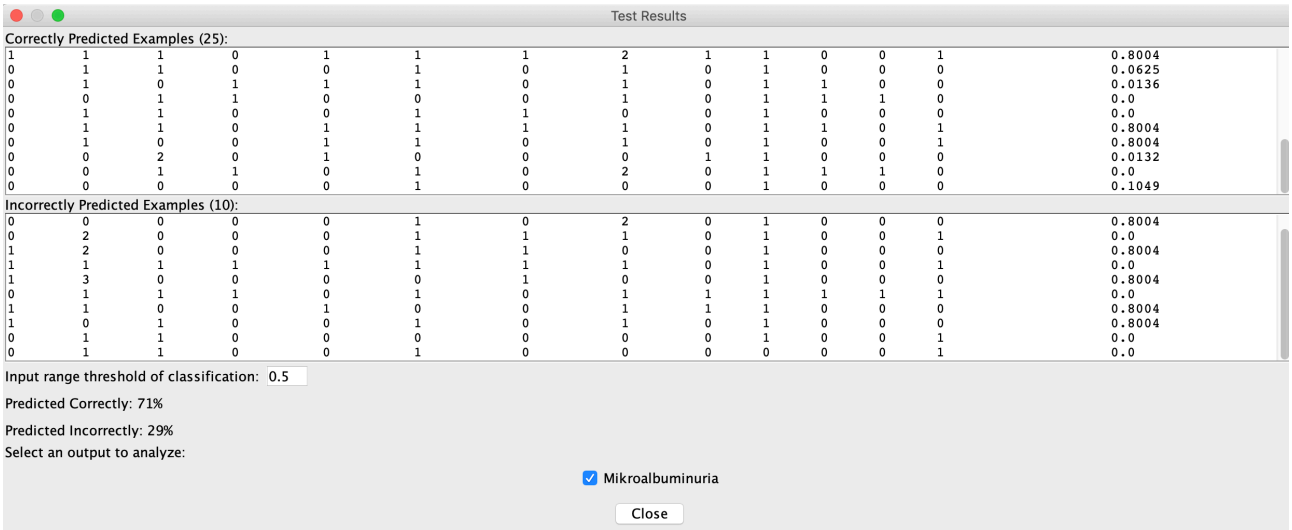


Rys.21. Losowy podział obiektów na zbiór treningowy oraz na zbiór testowy

Po czym przystąpiono do uczenia sieci. Rys.22. oraz Rys.23. przedstawiają wyniki uczenia.

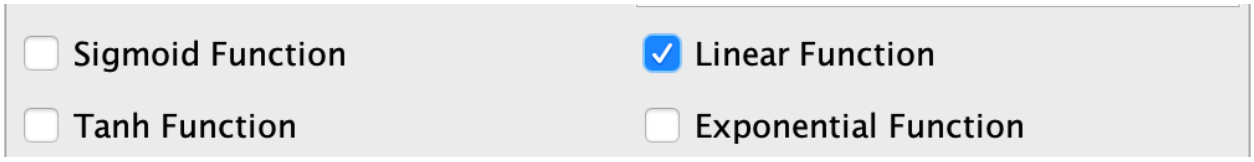


Rys.22. Wykres nauki sieci dla wygenerowanej domyślnie sieci



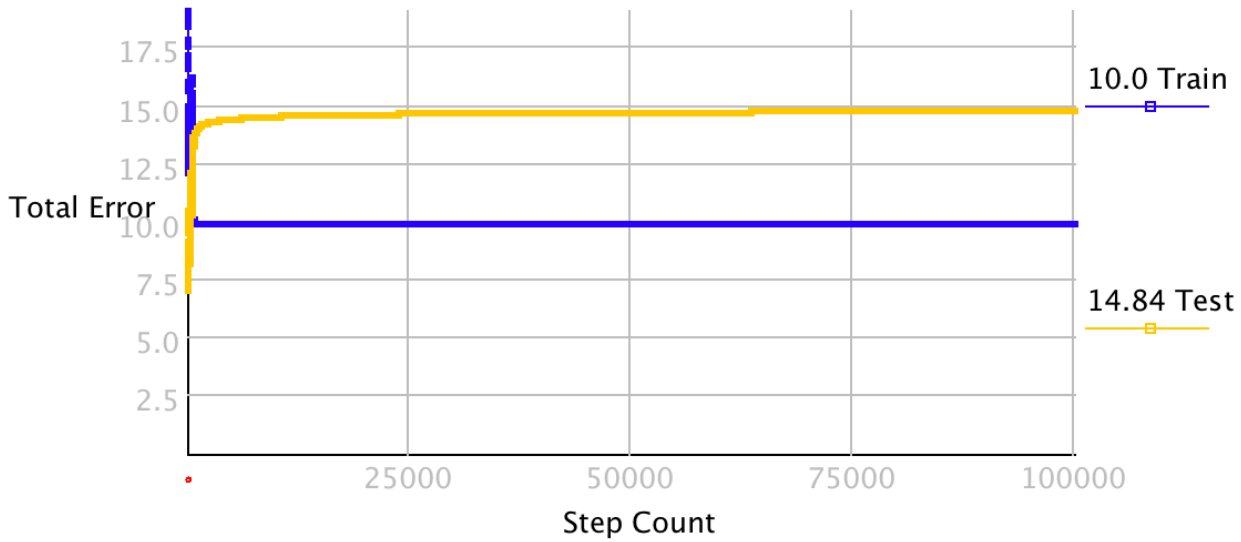
Rys.23. Wyniki testu klasyfikacji dla wygenerowanej domyślnie sieci

Zmieniono funkcje aktywacji z funkcji sigmoidalnej na funkcję liniową (Rys.24.).

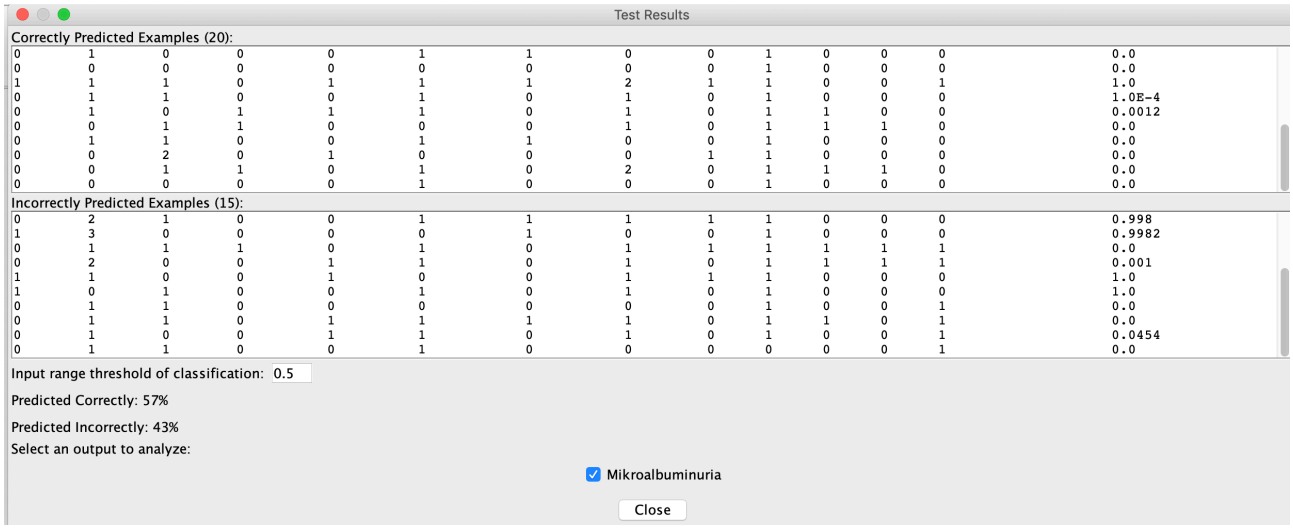


Rys.24. Zmiana funkcji aktywacji

A następnie po ponownym zainicjalizowaniu parametrów powtórzono proces uczenia sieci. Rys.25. oraz Rys.26. przedstawiają wyniki uczenia.

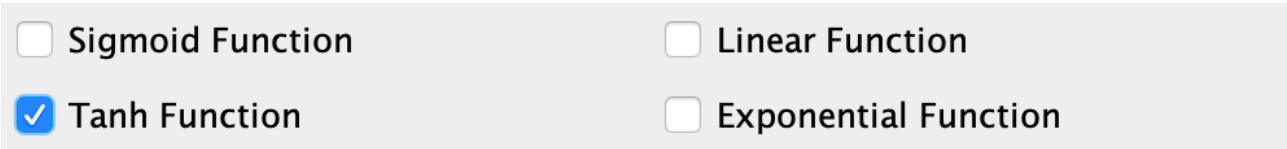


Rys.25. Wykres nauki sieci dla sieci z liniową funkcją aktywacji



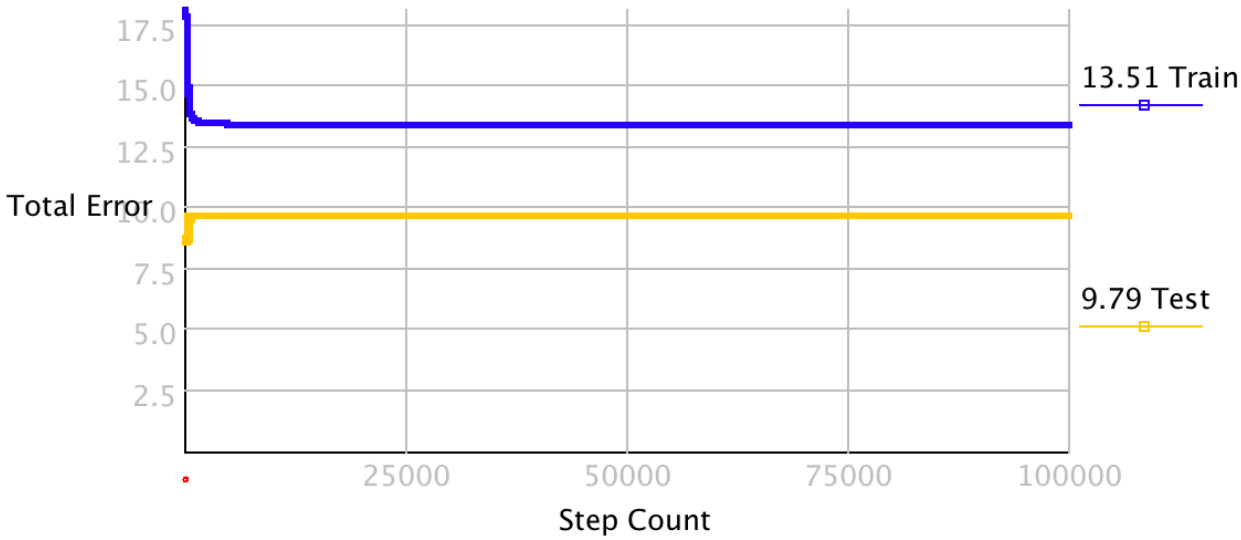
Rys.26. Wyniki testu klasyfikacji dla sieci z liniową funkcją aktywacji

Ponownie zmieniono funkcję aktywacji, tym razem na sigmoidalną funkcję bipolarna — tangens hiperboliczny (Rys.27.).

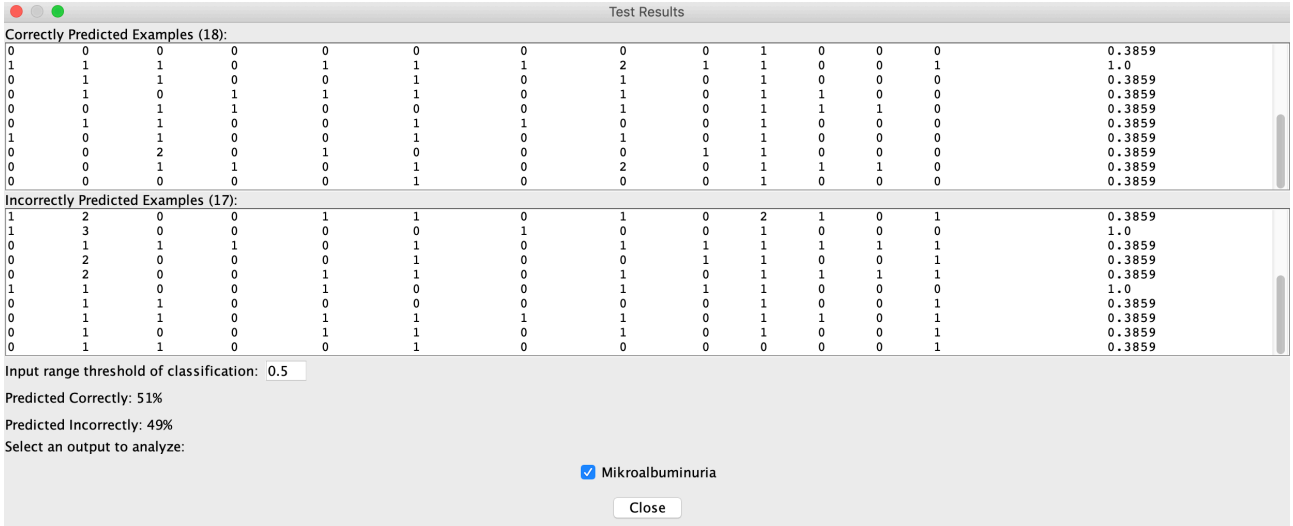


Rys.27. Zmiana funkcji aktywacji

A następnie po ponownym zainicjalizowaniu parametrów powtórzono proces uczenia sieci. Rys.27. oraz Rys.28. przedstawiają wyniki uczenia.

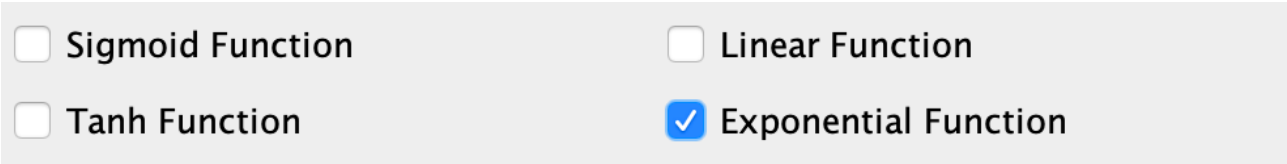


Rys.27. Wykres nauki sieci dla sieci z funkcją aktywacji w postaci tangensa hiperbolicznego



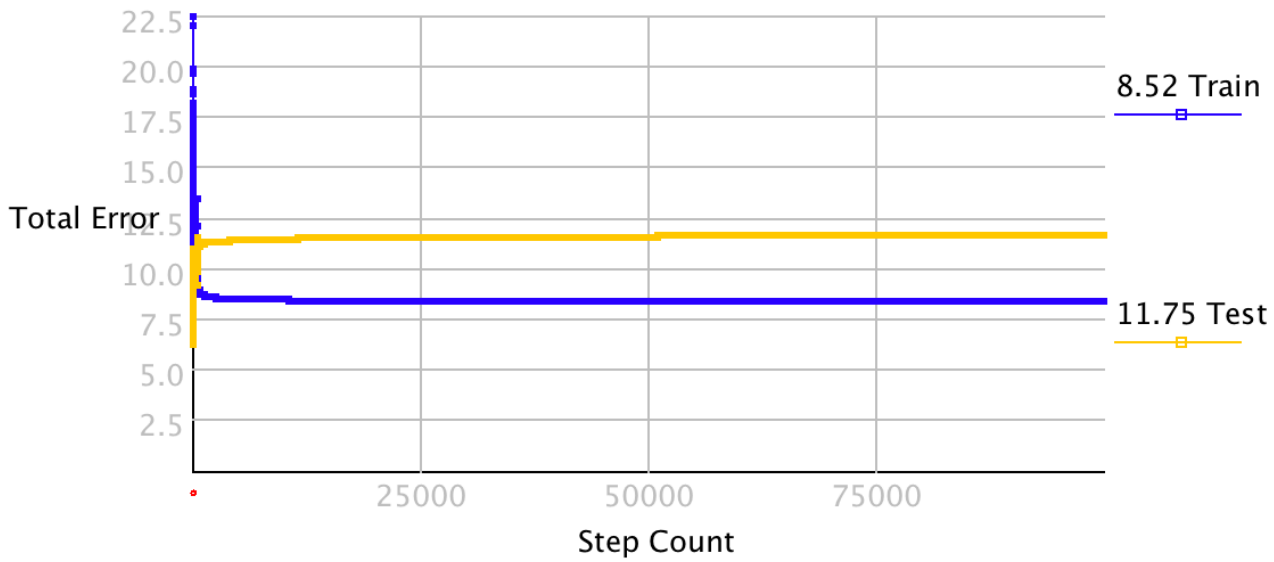
Rys.28. Wyniki testu klasyfikacji dla sieci z funkcją aktywacji w postaci tangensa hiperbolicznego

Funkcja aktywacji została ustawiona na funkcję wykładniczą (Rys.29.)

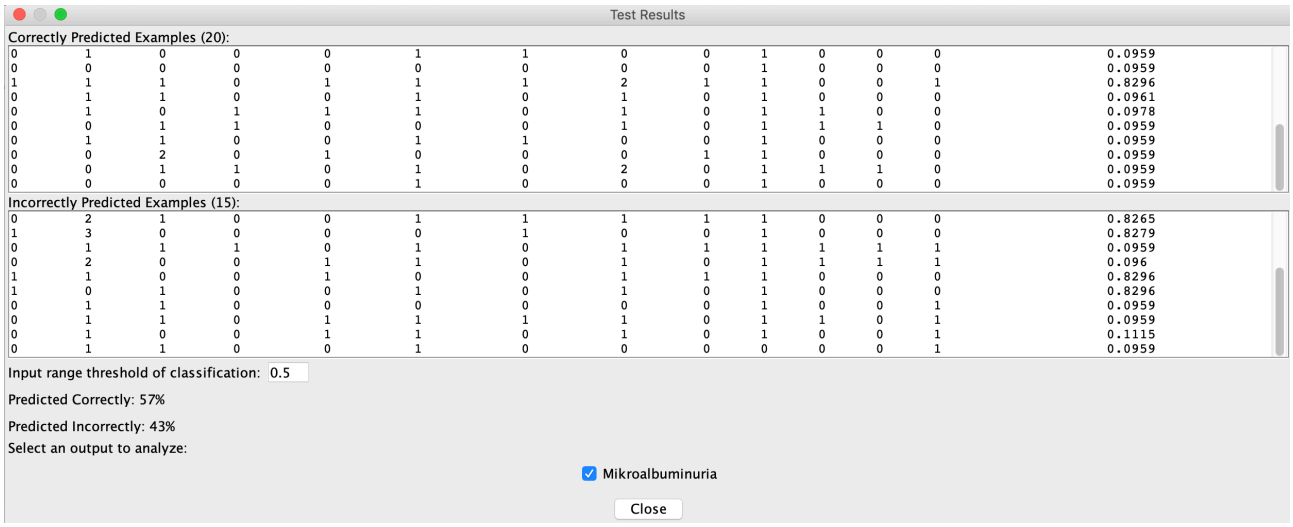


Rys.29. Zmiana funkcji aktywacji

Po ponownym zainicjalizowaniu parametrów powtórzono proces uczenia sieci. Rys.30. oraz Rys.31. przedstawiają wyniki uczenia.

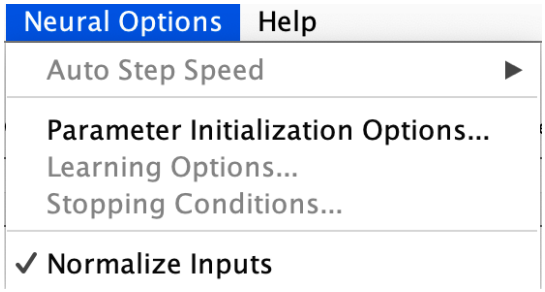


Rys.30. Wykres nauki sieci dla sieci z wykładniczą funkcją aktywacji



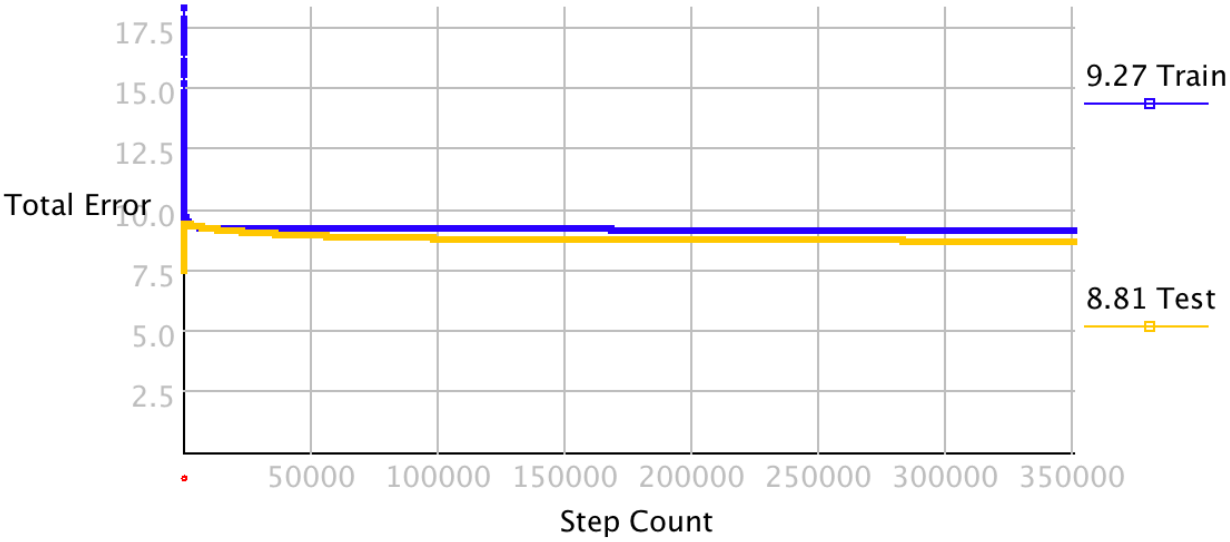
Rys.31. Wyniki testu klasyfikacji dla sieci z wykładniczą funkcją aktywacji

Przywrócona została domyślna funkcja aktywacji, a następnie znormalizowano dane wejściowe (Rys.32.).

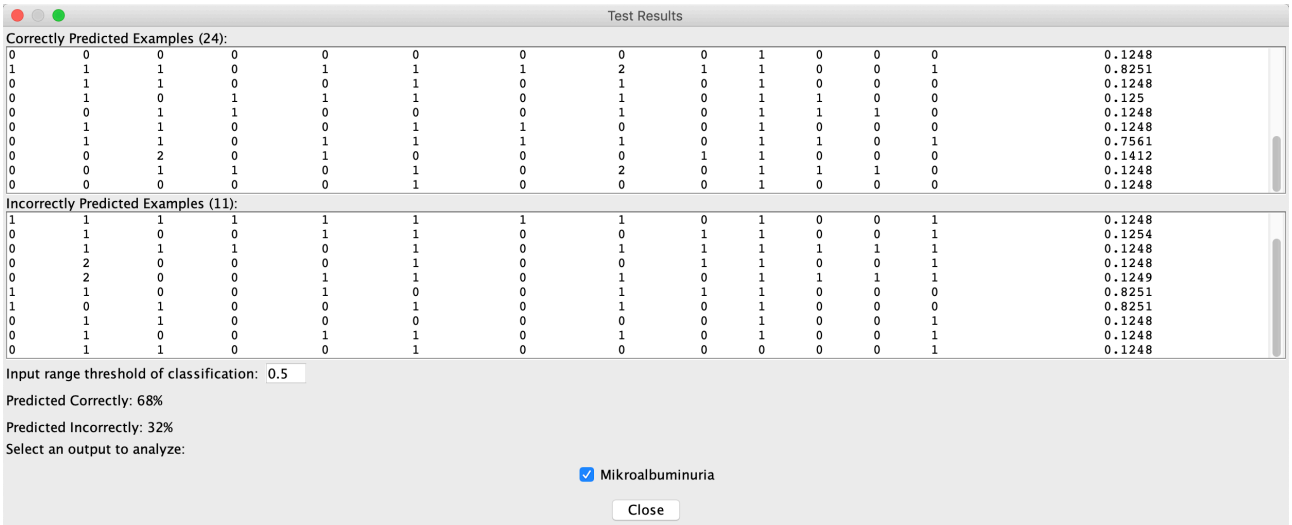


Rys.32. Normalizacja danych wejściowych

Po zainicjalizowaniu parametrów przystąpiono do uczenia sieci. Rys.33. oraz Rys.34. przedstawiają wyniki uczenia.

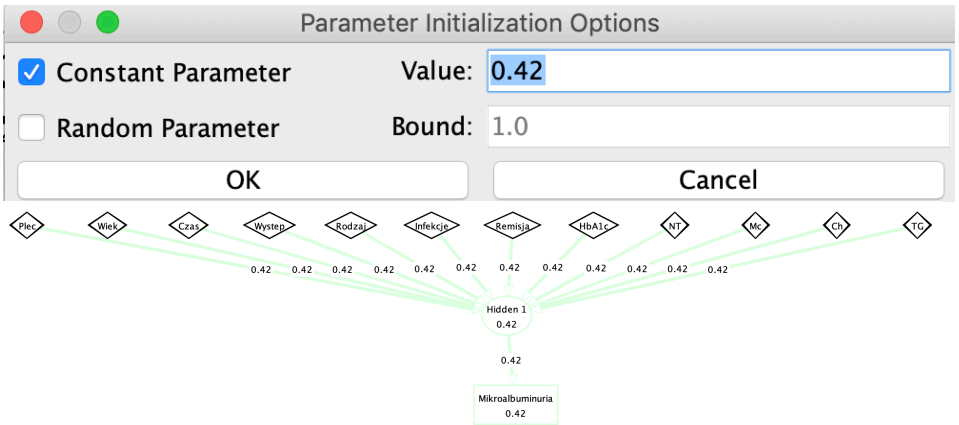


Rys.33. Wykres nauki sieci dla sieci po normalizacji danych wejściowych



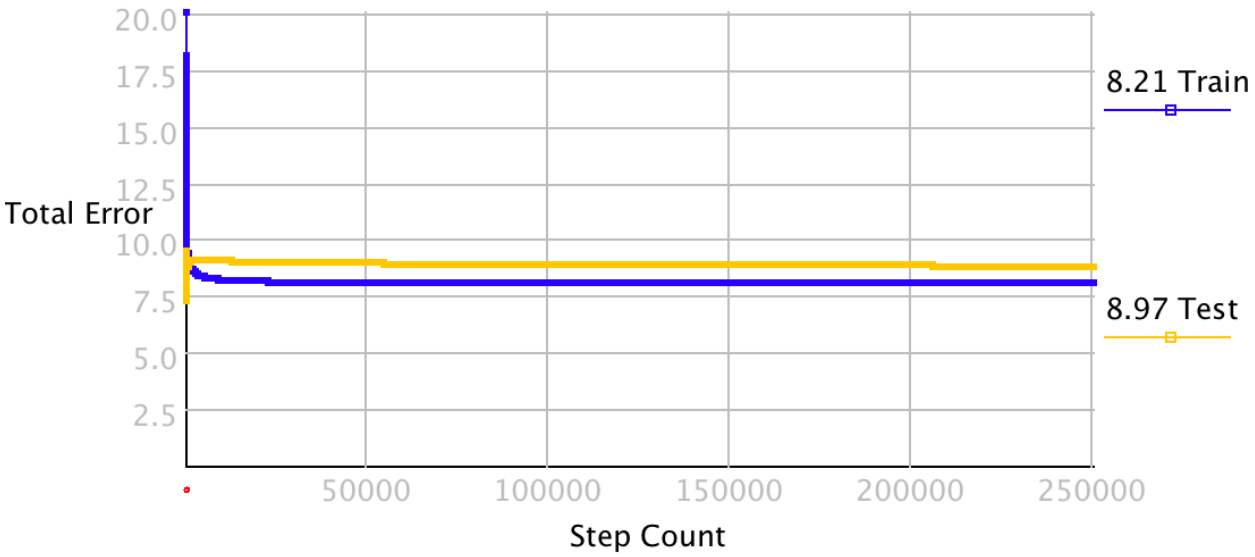
Rys.34. Wyniki testu klasyfikacji dla sieci po normalizacji danych wejściowych

Kolejny test przeprowadzono z ustalonym indywidualnie stałym parametrem początkowym sieci neuronowej. W Neural Options ustawiono stałą wartość na 0.42 (Rys.35.)

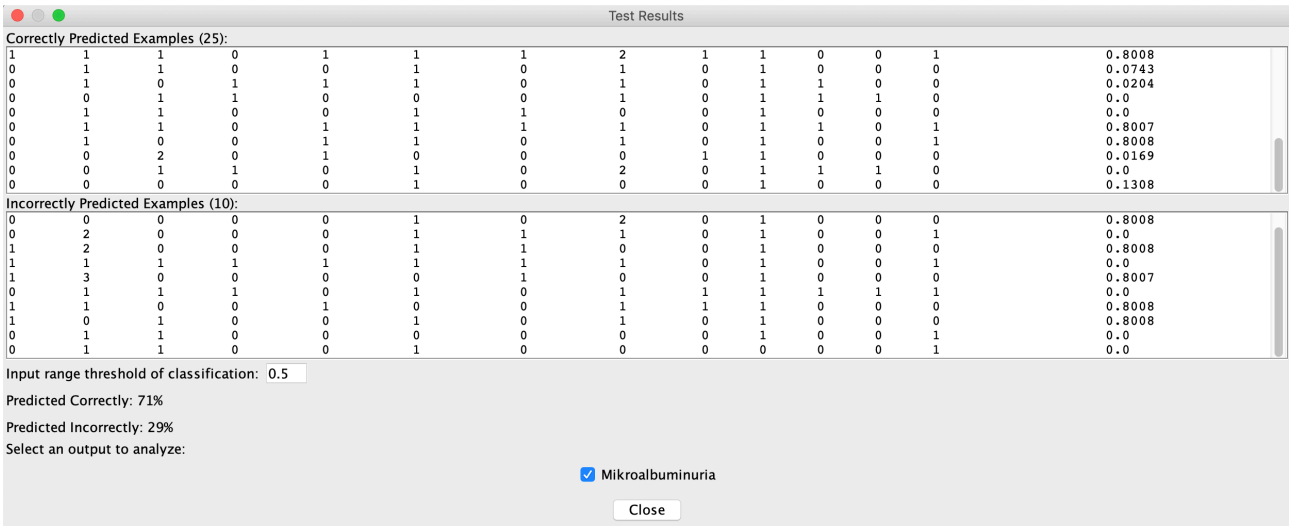


Rys.35. Ustawienie stałej wartości parametru początkowego sieci neuronowej

Po zainicjalizowaniu parametrów przystąpiono do uczenia sieci. Rys.33. oraz Rys.34. przedstawiają wyniki uczenia.



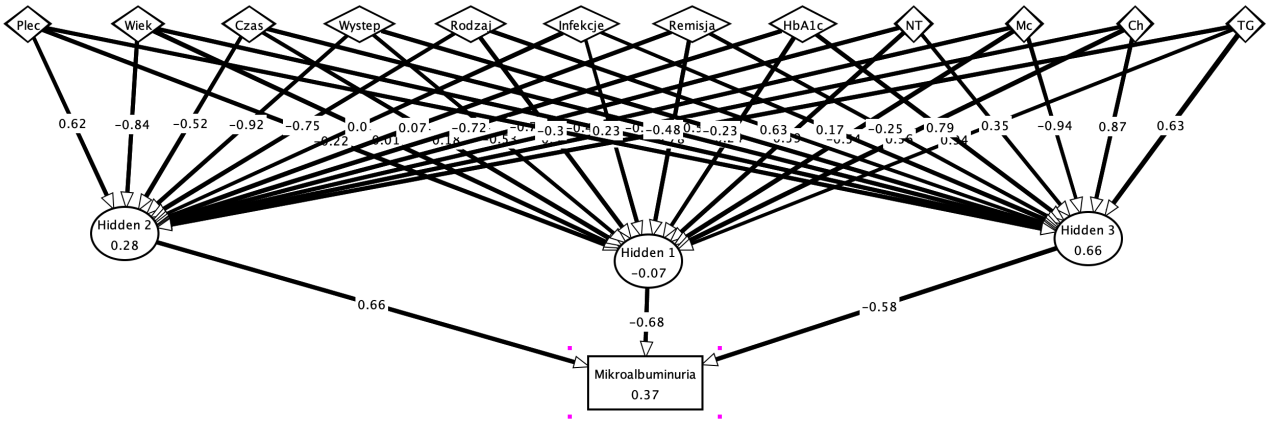
Rys.36. Wykres nauki sieci dla sieci z ustaloną stałą wartością parametrów wejściowych



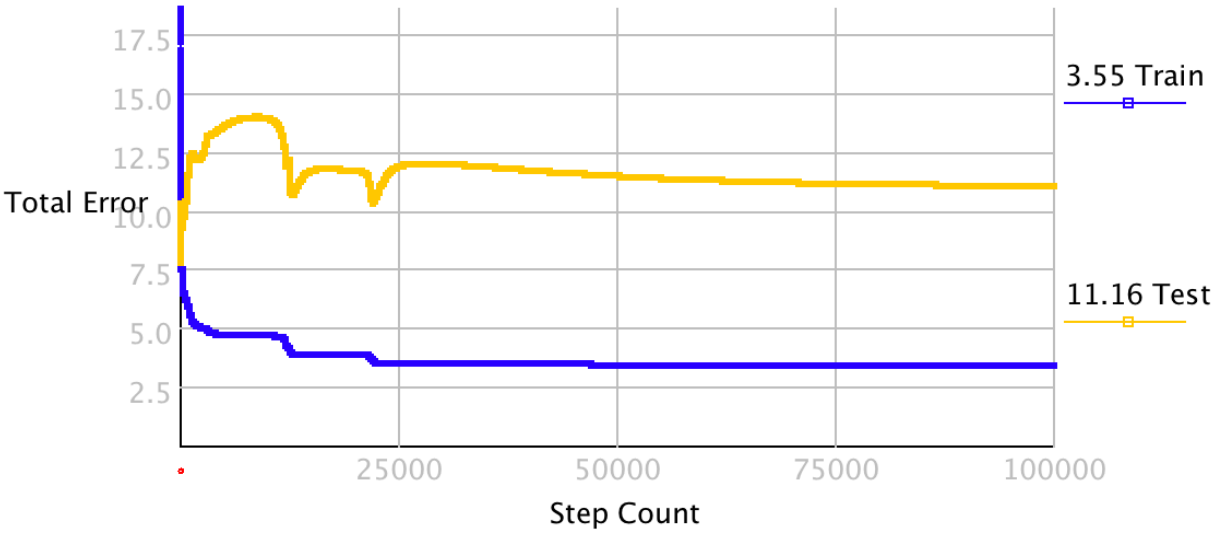
Rys.37. Wyniki testu klasyfikacji dla sieci z ustaloną stałą wartością parametrów wejściowych

W kolejnym kroku zmodyfikowano strukturę sieci neuronowej dodając zgodnie z instrukcją dwa dodatkowe neurony. Nowo utworzona sieć ma 1 warstwę ukrytą z 3 neuronami (Rys. 38.).

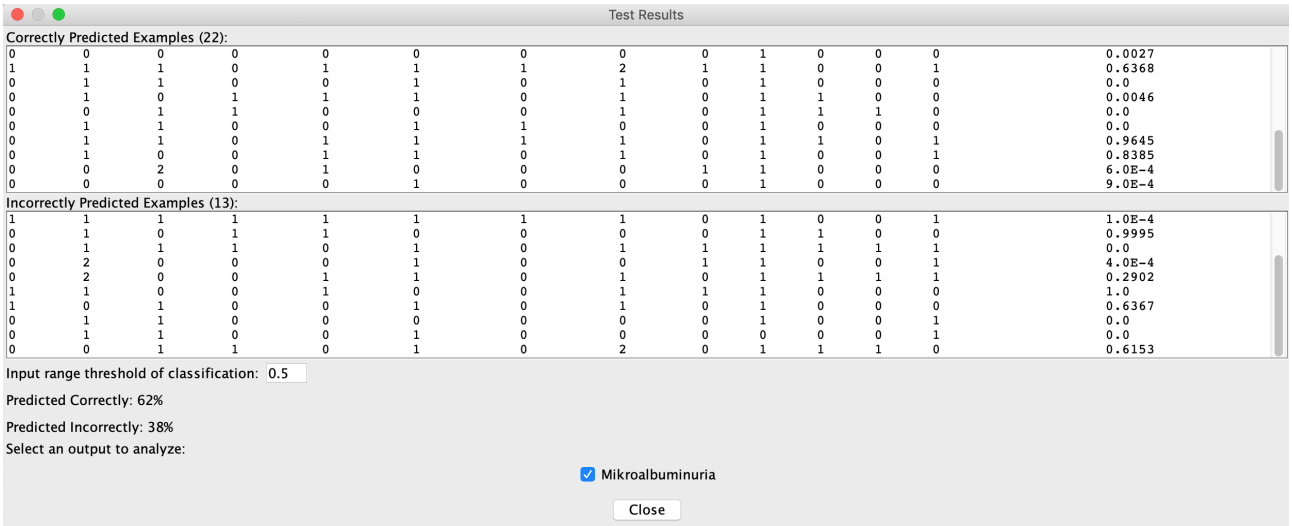
Po zainicjowaniu parametrów wejściowych przystąpiono do uczenia sieci. Rys.39. oraz Rys. 40. przedstawiają wyniki uczenia.



Rys.38. Schemat nowo utworzonej sieci neuronowej zaproponowanej przez prowadzącego



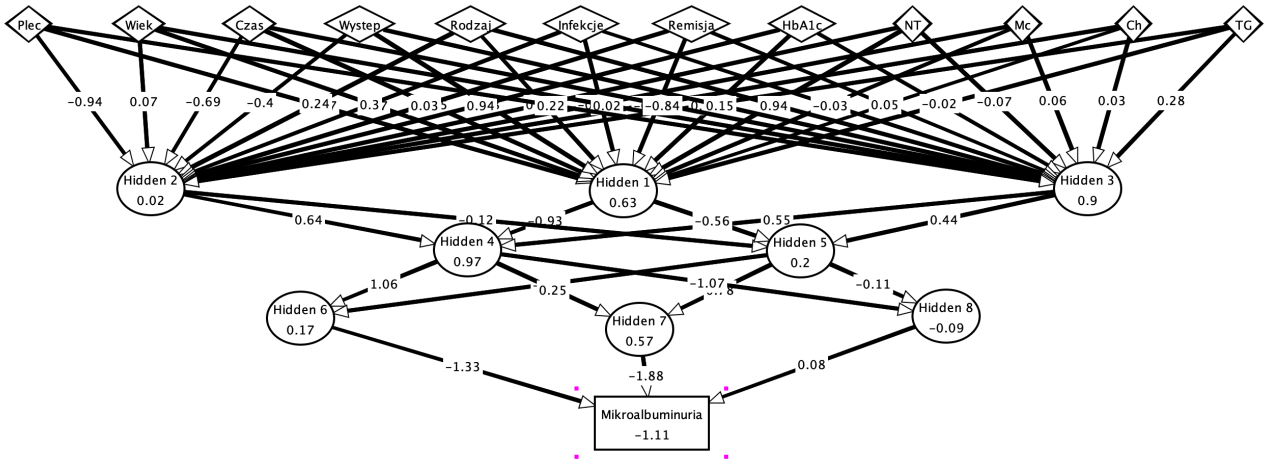
Rys.39. Wykres nauki sieci dla nowo utworzonej sieci zaproponowanej przez prowadzącego



Rys.40. Wyniki testu klasyfikacji dla nowo utworzonej sieci zaproponowanej przez prowadzącego

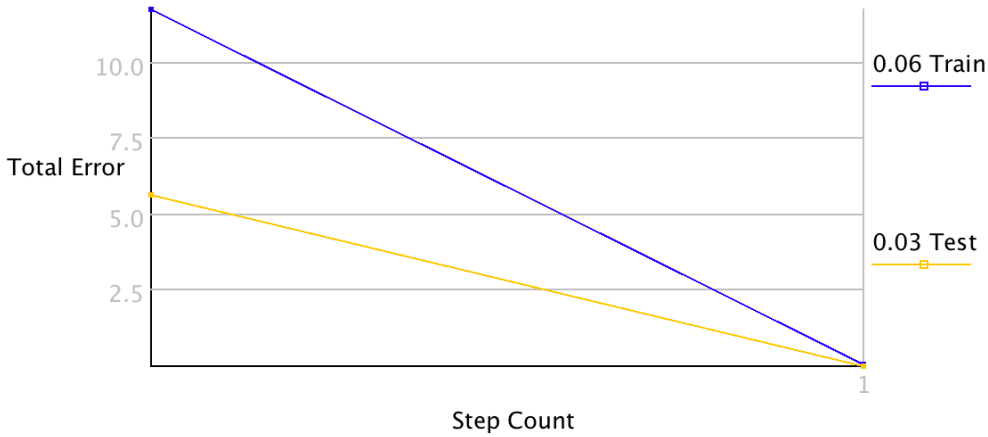
Ostatni test został przeprowadzony na sieci skonstruowanej wybranej przez wykonującego ćwiczenia.

Skonstruowana sieć składa się z trzech warstw ukrytych. Pierwsza warstwa posiada 3 neurony, druga 2, zaś trzecia 3. Rys.41. ukazuje schemat skonstruowanej sieci.

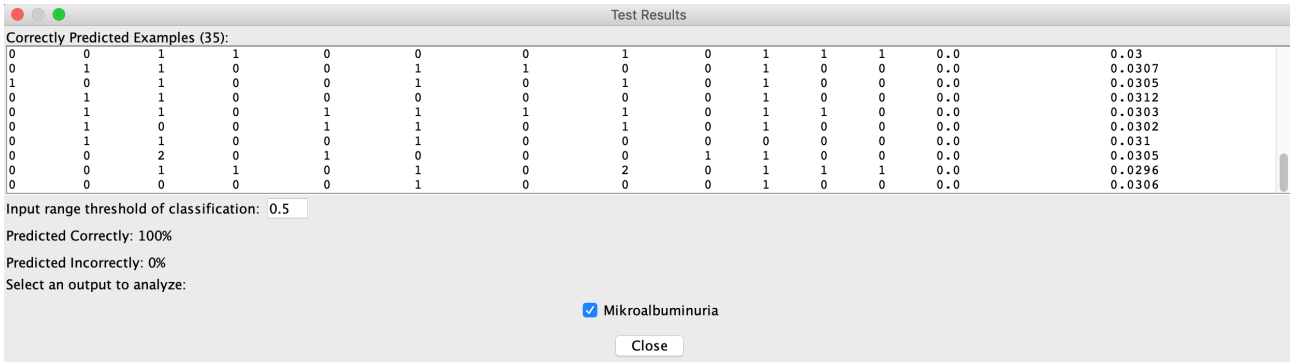


Rys.41. Schemat zaproponowanej sieci

Po zainicjalizowaniu parametrów wejściowych przystąpiono do uczenia sieci. Rys.42. oraz Rys.43. przedstawiają wyniki uczenia.



Rys.42. Wykres nauki sieci zaproponowanej sieci



Rys.43. Wyniki testu klasyfikacji dla zaproponowanej sieci

III. Wnioski

i. Zadanie 1

W przypadku sieci utworzonej domyślnie przez program (Rys.2.) proces uczenia się nie powiódł. Pomimo około 150 000 epok nie udało się w taki sposób wytrenować sieci neuronowej, by błąd obiektów testowych malał. Podejrzewając, że powodem takiego obrotu spraw są niefortunne parametry wejściowe postanowiono zainicjalizować je jeszcze raz i powtórzyć uczenie (Rys.6.), lecz otrzymano wykres o zbliżonym kształcie. Z powodu obserwacji wykresu zaprzestano oczekiwania do osiągnięcia target error i zakończono uczenie na około 200 000 epokach.

Mimo iż wyniki testów (Rys.8.) sugerowałyby, iż skuteczność sieci w klasyfikacji binarnej jest zbliżona do rzutu monetą, to przy próbie zakwalifikowania nowych obiektów poradził sobie stosunkowo dobrze (mając na uwagę niewielką próbę 3 obiektów, poprawnie sklasyfikował 2 z nich, zaś w przypadku trzeciego wartość otrzymana była na pograniczu obu odpowiedzi wynosząc 0.5223).

W przypadku dwóch następnych sieci (Rys.9. oraz Rys.14.) można wysnuć wniosek, że dodanie neuronów, oraz kolejnych warstw ukrytych zmniejsza czas potrzebny do wytrenowania sieci (jednocześnie zwiększając narzut obliczeniowy danej sieci, lecz przy tak małej próbie nie było to zauważalne). Czytając wykresy Rys.10-11. i Rys.15-16. można zauważyć że w przypadku obu modeli sieć osiągała zadowalający target error w ok. 20 epokach (co z porównaniem z 200 000 epokami jest gigantycznym skokiem, mając też w pamięci że w przypadku wykresu o 200 000 epokach nie udało się zejść do zadawalającego poziomu target error).

Oba modele w wynikach testów (Rys.13. i Rys.18.) osiągnęły 100% prawidłową klasyfikację, lecz w trakcie próby sklasyfikowania nowych obiektów, oba modele poradziły sobie gorzej niż pierwszy model. Może wynikać to z tego, że w tak małym zbiorze, zabranie 3 elementów mogło znacznie zaburzyć uczenie się sieci (nie miała w trakcie treningów podobnych przypadków). Mamy do czynienia z **overfitting** (przeuczeniem sieci), sieć dostosowała się bardzo dobrze do danych treningowych, zatracając zdolność do generalizacji, podając jej nowe dane, próbuje wykorzystać wiedzę zdobytą, lecz nie mając podobnych przykładów w zbiorze podejmuje złe decyzje jeżeli chodzi o klasyfikację.

ii. Zadanie 2

W przypadku zbioru podanego przez prowadzącego problem uczenia sieci był jeszcze bardziej widoczny. W przypadku pierwszego modelu sieci (Rys.20.) składającego się z 1 warstwy ukrytej i 1 neuronu proces uczenia szedł bardzo powoli (Rys.22.), pomimo ogólnej tendencji spadkowej, spadek ten był tak powolny iż po 500 000 epokach sprawiał wrażenie lini poziomej. Nie zmienia to faktu że udało się uzyskać 71% prawidłowej klasyfikacji (Rys. 23.). W przypadku zmiany funkcji aktywacji na funkcję liniową zauważono znaczne pogorszenie uczenia sieci (brak tendencji spadkowej), udało się uzyskać wynik prawidłowej

klasyfikacji na poziomie 57%. W przypadku ustawienia funkcji aktywacji na tangens hiperboliczny zaobserwowano z wykresu (Rys.27.) iż błąd treningu jak i błąd testu są raczej stałe (brak znaczącej tendencji spadkowej czy wzrostowej), co ciekawe błąd testu był niższy niż błąd treningu (co może być przypadkiem wylosowania parametrów wejściowych). Wynik klasyfikacji (Rys.28.) plasuje się na poziomie 51%. W przypadku funkcji wykładniczej na wykresie (Rys.30.) zaobserwowano gwałtowny spadek błędu treningowego przy jednoczesnym wzroście błędu testowego. Tak jak w przypadku funkcji tangensa hiperbolicznego otrzymano wynik klasyfikacji na poziomie 51% (Rys.31.).

W przypadku każdej funkcji aktywacji model sztucznej sieci neuronowej nie osiągnął zadowalającej wartości, lecz najbliższa zadowalającemu wynikowi okazała się domyślna sigmoidalna funkcja aktywacji (jako jedyna wyróżnia się procentem prawidłowej klasyfikacji, osiągając wynik lepszy niż rzut monetą), dlatego też była dalej wykorzystywana w kolejnych zadaniach.

Normalizacja jest przydatnym zabiegiem, daje możliwość iż sztuczna sieć neuronowa będzie uczyła się szybciej gdy wszystkie atrybuty warunkowe będą ze zbliżonego przedziału liczbowego (szczególnie gdy dane wejściowe łączą atrybuty o bardzo dużych jak i bardzo małych wartościach). W przypadku podanego zbioru konwersja liczbowa założona w programie (Rys.19.) była bardzo zbliżona do znormalizowanej, z tego powodu na wykresie (Rys.33.) nie zaobserwowano znaczącego usprawnienia, osiągając również nieznacznie gorszy wynik klasyfikacji niż w przypadku danych nieznormalizowanych (68% z Rys.34. w porównaniu z 71% z Rys.23., lecz jest to wynik na tyle nieznaczny iż prawdopodobnie może zależeć od zainicjalizowanych parametrów początkowych).

W przypadku testu przeprowadzonego z ustalonym indywidualnie stałym parametrem początkowym sieci neuronowej nie zaobserwowano znaczącej różnicy względem domyślnego modelu z Rys.20. (71% prawidłowej klasyfikacji na Rys.37. dla modelu ze stałym parametrem wejściowym wobec 71% prawidłowej klasyfikacji na Rys.23. dla domyślnego modelu z funkcją sigmoidalną).

Dodanie dwóch dodatkowych neuronów do warstwy ukrytej znacząco wpłynęło na proces nauki sieci (Rys.38.). Był to proces nieregularny z kilkoma skokami w górę, przy zachowaniu ogólnej tendencji spadkowej (Rys.39.). Z powodów skoków model ten okazał się być gorszy niż domyślny model, osiągając wynik prawidłowej klasyfikacji na poziomie 62% (Rys.40.)

Zaproponowana przez wykonującego ćwiczenia sieć (Rys.41.) znacząco usprawniła proces uczenia, jako jedynej udało się osiągnąć zadowalający wynik target error i to w zaledwie jednym kroku (Rys. 42.). Osiągając tym samym wynik 100% prawidłowej klasyfikacji (Rys. 43.).

Analizując otrzymane wyniki można wysnuć nastepujące wnioski (mając na uwadze, że część wniosków może być pochopna, wynika to z faktu małej próby czy też problemu z powtarzalnością testów):

- Dobre wyuczenie sieci na danych treningowych nie oznacza uzyskania dobrych wyników na danych testowych.Przykład taki ukazuje Rys.5. gdzie krzywa błędu treningowego maleje, podczas gdy krzywa błędu testowego rośnie
- Postać warstwy wejściowej sieci zależy od liczby atrybutów warunkowych. Ile kolumn ma plik wejściowy (z wyłączeniem atrybutu decyzyjnego) na podstawie którego uczona będzie sieć, tyle będzie neuronów w warstwie wejściowej.
- Postać warstwy wyjściowej zależy ile klas decyzyjnych będzie zawierał model sieci. W przypadku klasyfikacji binarnej będzie to jeden neuron.
- Liczba warstw ukrytych oraz liczba neuronów w danych warstwach ma znaczenie na uczenie sieci. Większa liczba neuronów i warstw ukrytych usprawnia proces uczenia, potrzeba mniej epok do wyuczenia sieci, lecz związane jest to z większym nakładem obliczeniowym
- Możliwym jest, że w wyniku przetrenowania sieć neuronowa będzie bardzo sprawna w klasyfikacji danych treningowych, lecz zupełnie bezradna w przypadku klasyfikacji nowych obiektów. Takie zjawisko nazywa się przeuczeniem sieci i jest zjawiskiem niepożądanym
- Modyfikacja sieci polega przede wszystkim na dodawaniu neuronów do kolejnych warstw ukrytych oraz na zmianie funkcji aktywacji (z założeniem, że w przypadku sieci powinna być ona jednolita)
- Inicjalizacja parametrów początkowych może wpłynąć na wynik uczenia, w takim przypadku można wylosować parametry początkowe i ponownie rozpocząć uczenie sieci