Sprawozdanie lab 4

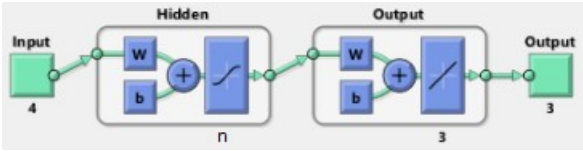
Metody inteligencji obliczeniowej – Informatyka Stosowana, WFIIS,
Jakub Salamon, II rok

Celem zajęć laboratoryjnych nr 4 było poznanie i zastosowanie
wielowarstwowych sieci neuronowych

Na zajęciach mieliśmy skonstruować sieć, która miała wybierać substancje konserwujące do napojów na podstawie czterech parametrów. Pierwszy z nich to ilość wody, drugi kwasowość, trzeci zawartość cukru(unormowana – 1.0 oznacza 10g/100ml) oraz napięcie powierzchniowe. Zbiór uczący składał się ze 130 przypadków, a testujący z 18, co daje nam 148 próbek. Wyjściem sieci będzie konserwant zakodowany jako macierz zer i jedynek o rozmiarze 3x1 dla każdej z próbek.

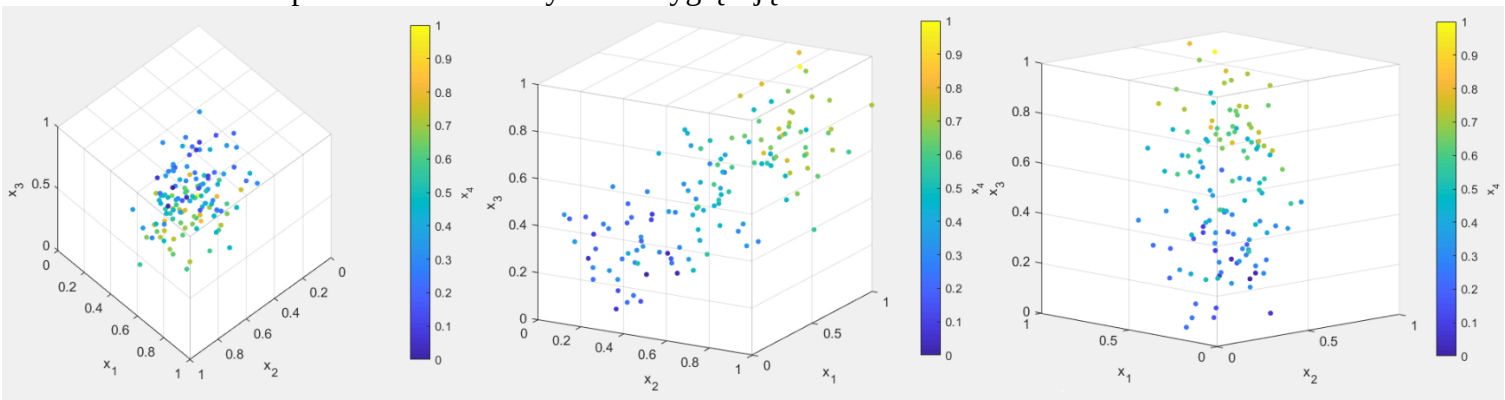
Po zaimplementowaniu takiej sieci w MATLABie musimy uwzględnić dodatkowo to, że trzeba zaokrąglać wyjścia sieci do zer i jedynek. Wartości wyjścia, które są mniejsze bądź równe 0.5 zaokrąglamy do 0, a powyżej 0.5 do 1. Maksymalna ilość epok jest ustawiona na 60, a minimalny gradient na 10^{-25}. Musimy też zablokować automatyczny podział MATLABa na danych na część uczącą, testującą i walidującą.

Schemat takiej sieci:



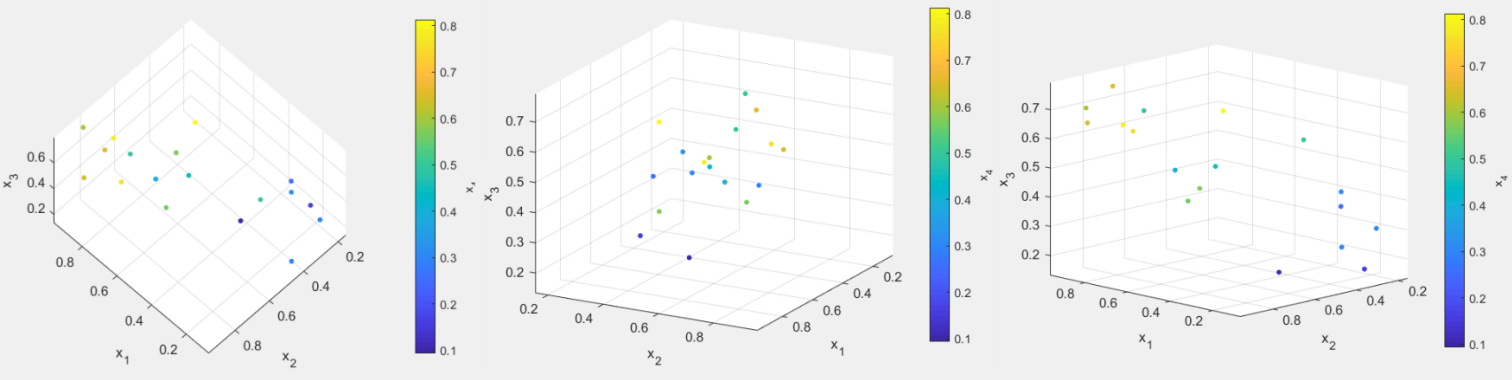
n – ilość neuronów ukrytych

Dane testowe przedstawione na wykresie wyglądają tak:



Możemy zauważyć, że dane są umieszczone w sposób liniowy.

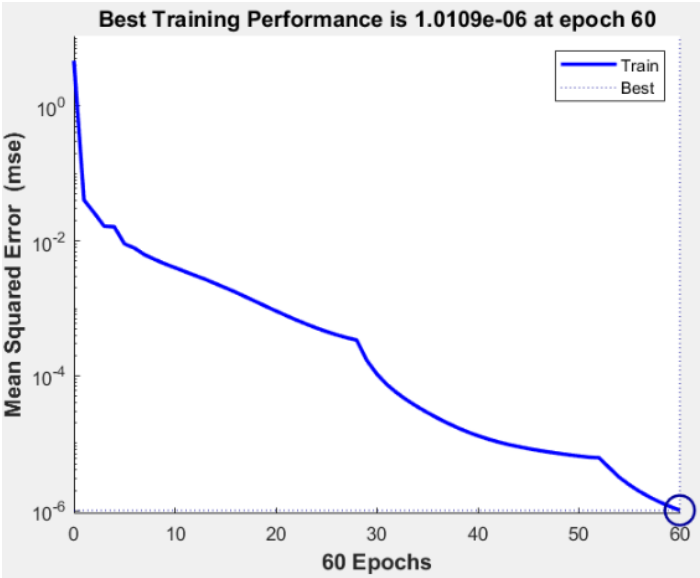
Natomiast dane testowe wyglądają tak:

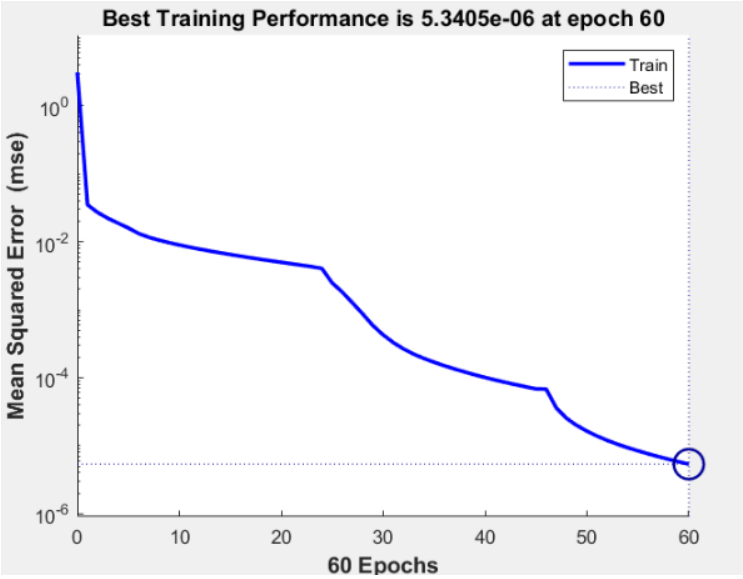


Przetestujmy naszą sieć dla różnej liczby neuronów i różnych funkcji aktywacji warstwy ukrytej. Będziemy używać sigmoidalnych funkcji aktywacji warstwy ukrytej, tj. tansig oraz logsig. W warstwie wyjściowej funkcją aktywacji będzie funkcja liniowa purelin.

50 neuronów ukrytych

Użycie funkcji aktywacji tansig (po lewej) oraz logsig (po prawej) przyniosło takie rezultaty:

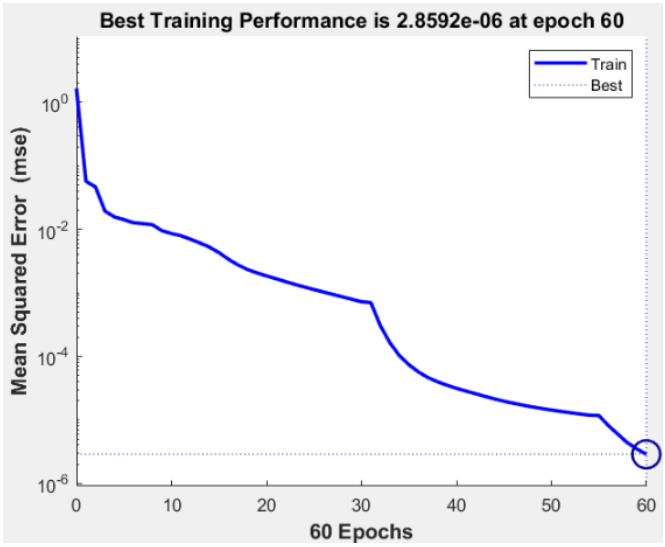


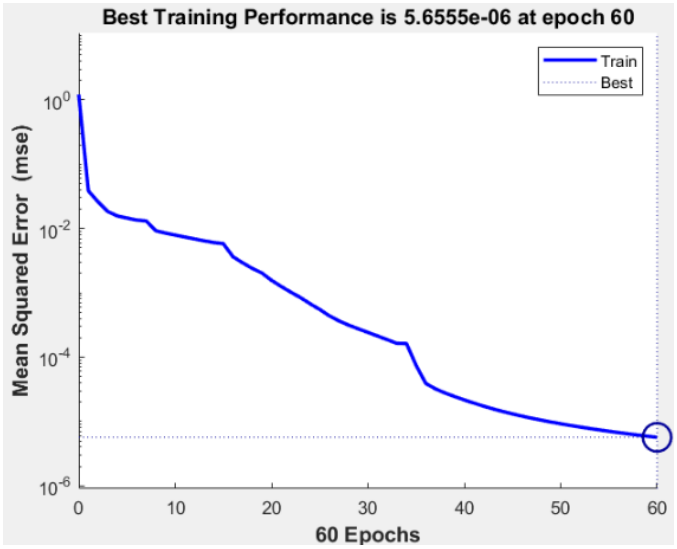


W pierwszym przypadku średnia z 10 uruchomień skryptu to 82.78% wydajności dla danych testowych, co daje nam około 3 błędnych wyników na 18 próbek, a w drugim przypadku odpowiednio 88.33% oraz 2 błędne wyniki.

Warto zauważyć, że pomimo mniejszego błędu średniokwadratowego dla danych uczących, w przypadku funkcji tansig, uzyskujemy mniejszą wydajność dla danych testowych.

35 neuronów ukrytych

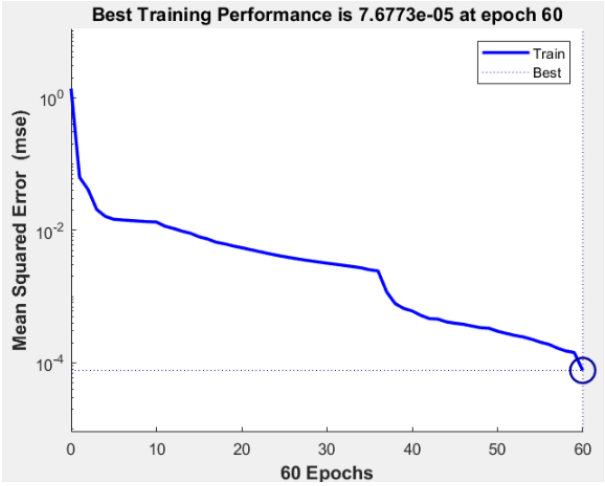


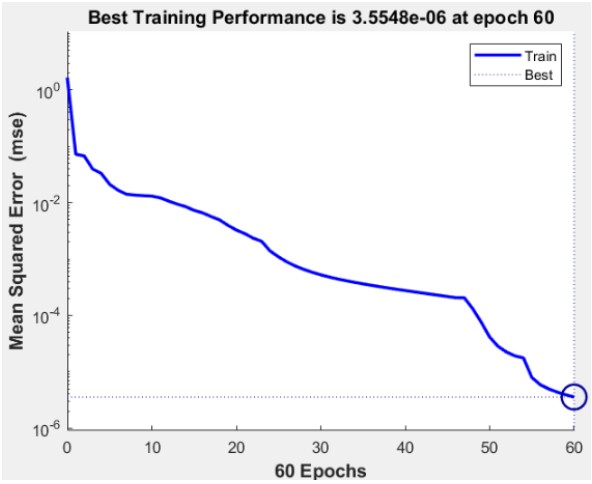


Wydajność w pierwszym przypadku wynosi 89.44% oraz dostajemy 2 pomyłki. W drugim przypadku 88.89% oraz taka sama ilość błędów.

W tym przypadku różne funkcje aktywacji dają podobny efekt, jednak funkcja logsig szybciej osiąga mniejszy błąd średniokwadratowy wydajności dla danych uczących.

20 neuronów ukrytych

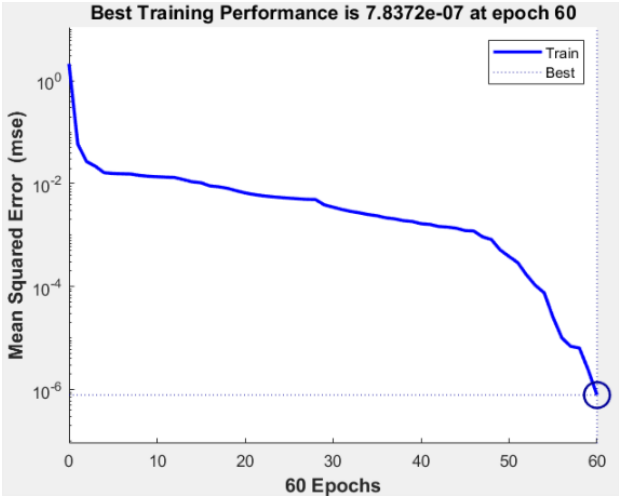


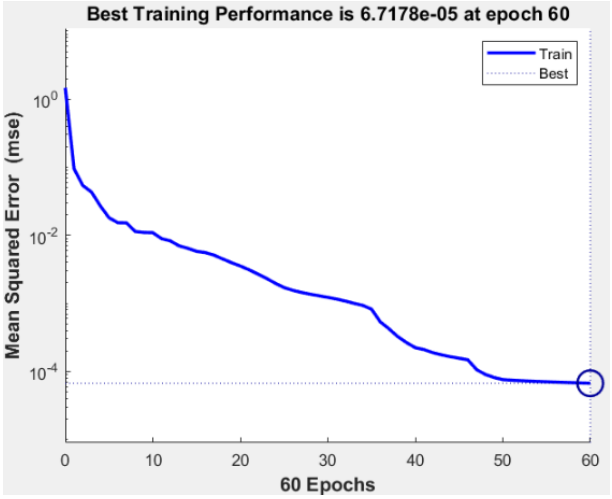


W pierwszym przypadku wydajność wynosi 95%, co daje średnio 1 błąd. Natomiast w drugim przypadku jest to odpowiednio 96.67% oraz także 1 błąd.

Błąd średniokwadratowy dla danych uczących jest podobny w obu przypadkach. Jednak w przypadku funkcji logsig osiągamy go szybciej.

15 neuronów ukrytych

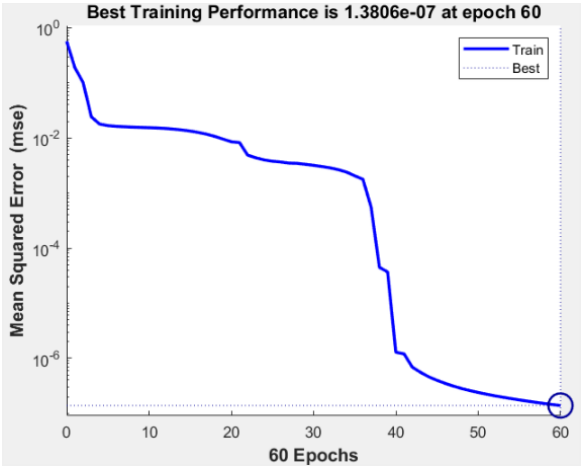


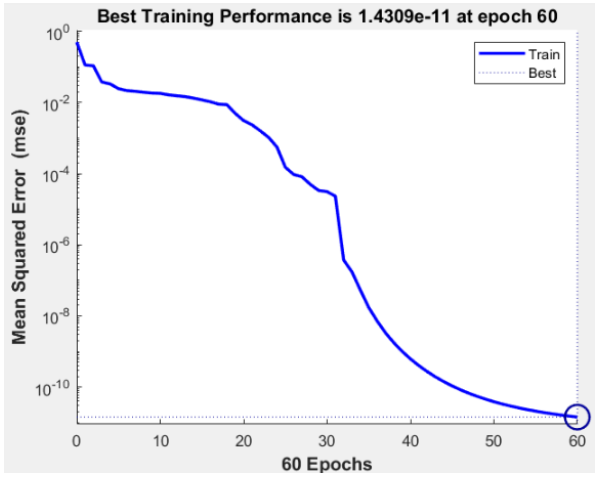


W przypadku funkcji tansig otrzymujemy 96.11% wydajności oraz średnio około 1 błąd. W drugim przypadku jest to odpowiednio 92.22% oraz także 1 pomyłka.

Funkcja tansig trochę gorzej radzi sobie przy 20 neuronach, ponieważ dopiero w końcowych epokach uzyskuje najmniejszy wynik wydajności dla danych uczących. Można także powiedzieć, że dla funkcji logsig najlepszą wydajność uzyskujemy już w około 50 epoce. Mimo to lepszy błąd średniokwadratowy uzyskujemy dla funkcji tansig.

10 neuronów ukrytych



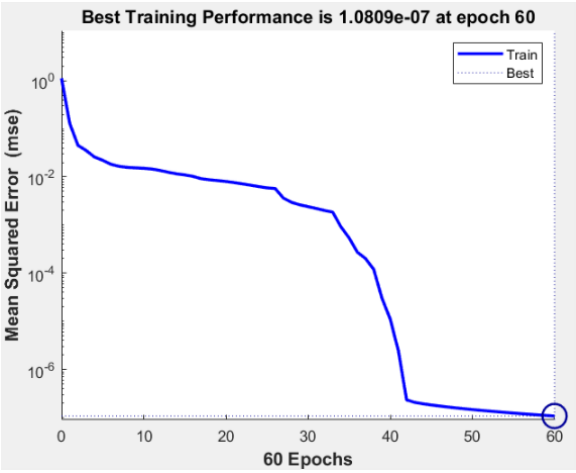


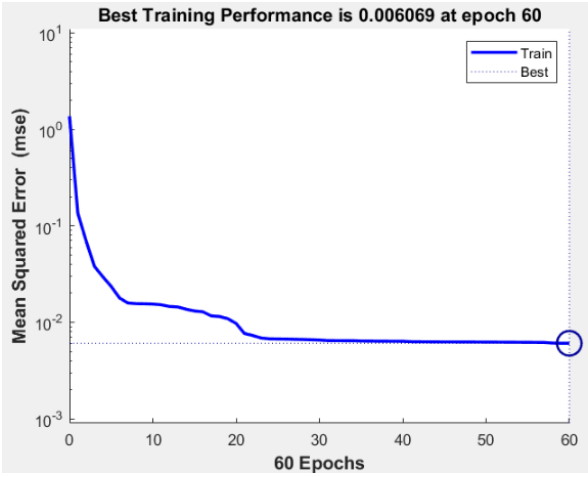
Otrzymujemy 97.22% wydajności dla danych testujących oraz średnio około 1 błąd przypasowania konserwantów. Natomiast w drugim przypadku wydajność to 95.55% oraz taka sama ilość pomyłek.

Dla funkcji tansig obserwujemy nagły spadek błędu średniokwadratowego w około 40 epoce uczenia. Dla logsig również występuje taki spadek jednak jest on łagodniejszy. Ogólnie jednak dla logsig błąd średniokwadratowy uczenia jest 10000 razy mniejszy czego nie widać dla danych

testowych. Jest to najmniejszy błąd dla danych uczących dla funkcji logsig jaki kiedykolwiek osiągnęliśmy.

8 neuronów ukrytych

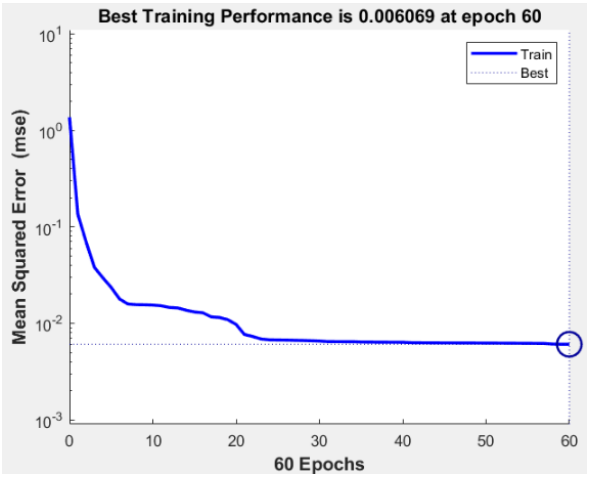


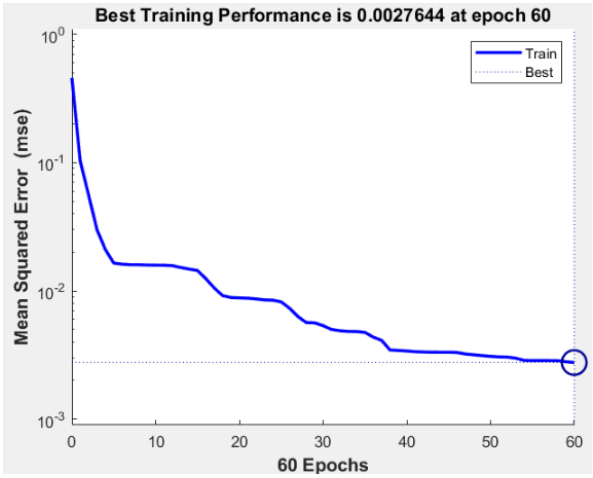


Otrzymujemy wydajność na poziomie 95.56%, co daje średnio 1 błąd dopasowania. Dla funkcji logsig jest to 97.22% oraz taka sama ilość błędów.

Pomimo tego, że dla funkcji logsig otrzymujemy gorszy błąd średniokwadratowy dla danych uczących, wydajność dla danych testowych jest większa. Warto również zauważyć, że funkcja logsig nie pozwala na zmniejszenie się błędu średniokwadratowego już od około 24 epoki. Otrzymujemy najmniejszy błąd dla danych uczących dla funkcji tansig jaki otrzymaliśmy we wszystkich testach.

5 neuronów ukrytych

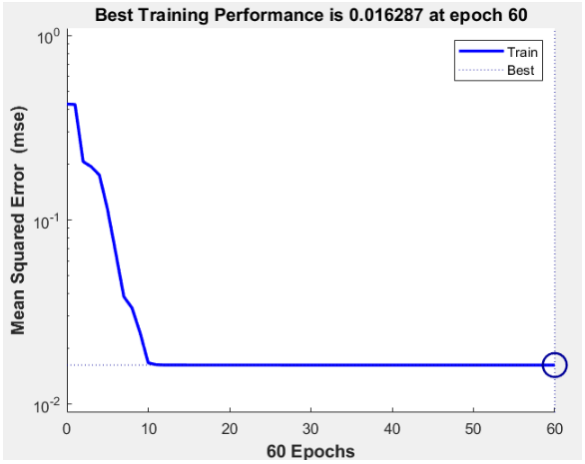


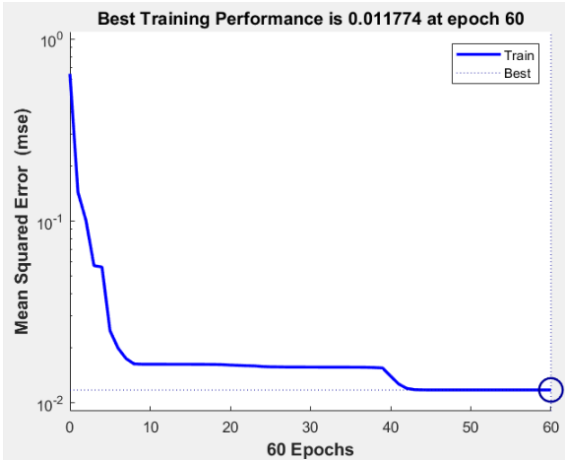


Otrzymujemy jak dotąd najwyższą wydajność, bo aż 98.33% i średnio 0.3 błędów, więc już możemy mówić o zerowej ilości błędów. Natomiast funkcja logsig przynosi nam gorsze rezultaty, bo 92.78% wydajności i średnio 1 pomyłkę w dopasowaniu konserwantów.

Błędy średniokwadratowe dla danych uczących są zbliżone. Dla funkcji tansig błąd ten już praktycznie nie maleje od około 24 epoki.

3 neurony ukryte

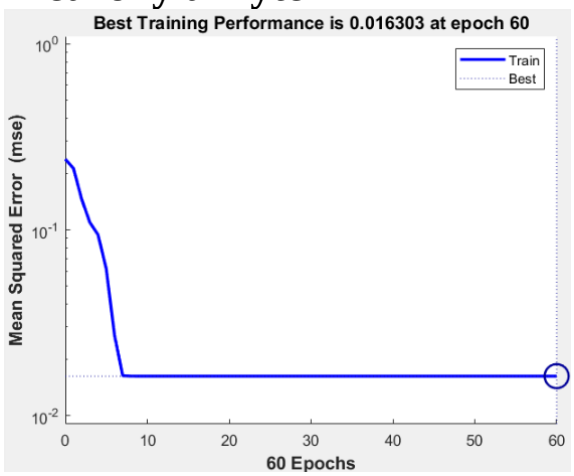


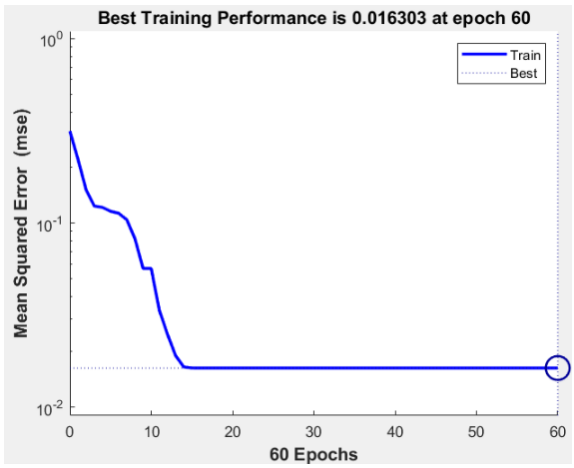


Wydajność dla danych testowych to 96.67%, co średnio w zaokrągleniu dane 1 błąd. W drugim przypadku otrzymujemy bardzo zbliżone wyniki, bo odpowiednio 96.11% oraz taką samą ilość błędów.

Błędy średniokwadratowe są najbardziej zbliżone jak do tej pory. Jednak w przypadku funkcji tansig jest on na takim poziomie już od około 10 epoki, a dla logsig dopiero w końcowych epokach.

2 neurony ukryte

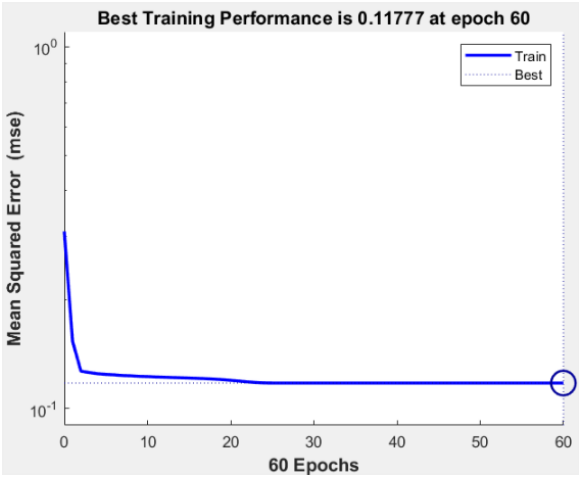


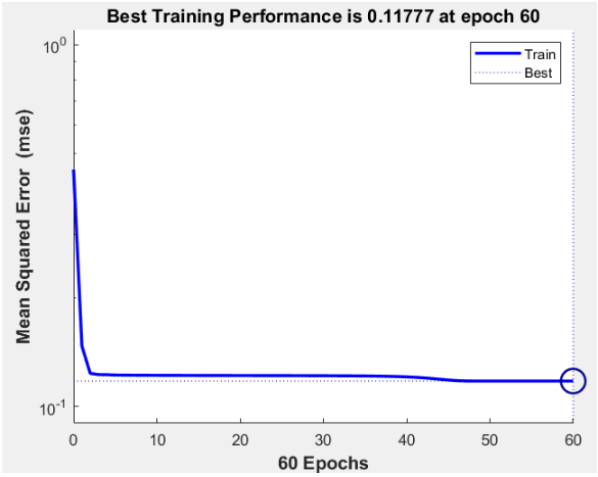


W przypadku funkcji aktywacji tansig wydajność w każdej iteracji była stała i wynosiła 94.44%. Dawało to 1 złe przypasowanie konserwantów. Co ciekawe dla funkcji logsig sytuacja wyglądała dokładnie tak samo.

Również warto zauważyć, że błędy średniokwadratowe w obu przypadkach były sobie równe. Widzimy jednak różnice na wykresach tych błędów. W przypadku logsig był on osiągany około 7 epok później.

1 neuron ukryty





Wydajność dla tansig jest za mała, bo wynosi tylko 38.89%, co daje aż 11 pomyłek. W przypadku logsig było podobnie, bo 38.33% oraz taka sama średnia ilość pomyłek. Oznacza to, że taka sieć nie spełnia swojego zadania. Można nawet powiedzieć, że odwracając wyniki, które otrzymujemy dostaniemy lepszą wydajność, ale nie o to chodzi nam w konstruowaniu sieci neuronowych.

Błędy średniokwadratowe w obu przypadkach były takie same oraz były najwyższe spośród wszystkich. Sieć oparta na jednym neuronie ukrytym nie nadaje się do realizacji tego zadania.

Podsumowanie

Liczba neuronów	Wydajność (dane testowe) [%]	
	tansig	logsig
50	82.78	88.33
35	89.44	88.89
20	95.00	96.67
15	96.11	92.11
10	97.22	95.55
8	95.56	97.22
5	98.33	92.78
3	96.67	96.11
2	94.44	94.44
1	38.89	38.33

Najlepszą wydajność dla tansig uzyskaliśmy dla 5 neuronów w warstwie ukrytej, natomiast dla logsig dla 8 neuronów ukrytych. Jednak tylko w przypadku funkcji aktywacji tansig możemy

mówić o braku błędów dopasowania konserwantów w najlepszej sieci. Warto też zauważyć, że największą wydajność otrzymujemy, gdy błąd średniokwadratowy dla danych uczących wynosi 0.006069 i wynosi on tyle samo dla 5 oraz 8 neuronów w warstwie ukrytej odpowiednio dla funkcji aktywacji tansig i logsig. Nie jest to najmniejszy błąd jaki dostaliśmy, bo aż 10^8 razy większy od najmniejszego otrzymanego.

Funkcja aktywacji warstwy wyjściowej hardlim

Zastosowanie takiej funkcji aktywacji spowodowało ogromny spadek wydajności sieci. Przypomnijmy, że najlepsza wydajność jaką otrzymaliśmy dla funkcji aktywacji warstwy ukrytej purelin (liniowej) wynosiła 98.33% oraz 97.22% odpowiednio dla funkcji aktywacji warstwy ukrytej tansig i logsig. Po zmianie na funkcje hardlim wydajność spadła do kolejno średnio 15% oraz 12.78%. Sieć działa bardzo źle w takim przypadku. Podobnie do przykładu z zastosowaniem jednego neuronu w warstwie ukrytej także tutaj możemy odwrócić wyniki otrzymane z naszej sieci, ale mimo wszystko działa ona źle. Jest to spowodowane tym, że hardlim jest skokową funkcją aktywacji, a jak pamiętamy z wcześniejszych przykładów otrzymywaliśmy wyniki na wyjściu, które niekoniecznie były zerami i jedynkami i musieliśmy je zaokrąglać.

Walidacja krzyżowa

Głównym celem walidacji krzyżowej jest podzielenie próby na kilka podzbiorów. Zastosowanie walidacji krzyżowej ma na celu także zapobieganie przeuczeniu się sieci.

Walidacja prosta
Polega na losowym przydzielaniu danych ze zbioru do uczących i testowych, gdzie ten drugi stanowi zazwyczaj około 33% całości.

Walidacja K-krotna
W metodzie tej dzielimy zbiór uczący na k podzbiorów, z czego k-1 wykorzystujemy jako dane uczące, a jeden pozostały jest zbiorem danych testowych. Wybór parametru k zależy głównie o ilości danych jakie mamy. Większy współczynnik k musimy zastosować do małej ilości danych, aby powstały zbiór uczący nie był zbyt mały. Warto zadbać o to, aby zestaw danych wejściowych był odpowiednio podzielony, tzn. że powinniśmy mieć elementy różnych typów w takich samych proporcjach w każdym ze zbiorów, tj. uczącym i testującym.

Procedura Leave-one-out
Jest to rodzaj walidacji K-krotnej. Przydziela wszystkie dane do danych testowych. Każdy element ma klasyfikator p wykorzystujący wszystkie elementy oprócz p. Element p jest klasyfikowany, a my zapisujemy wynik. Gdy przetestujemy wszystkie elementy sumujemy otrzymane wyniki. Stosuje się ją zazwyczaj do małych i prostych zbiorów, ponieważ jest to bardzo wolna metoda.

Jakub Salamon, 297914
Informatyka Stosowana, WFIIS, II rok