Sprawozdanie lab 3

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

Celem zajęć laboratoryjnych nr 3 była aproksymacja funkcji z użyciem sztucznej sieci neuronowej

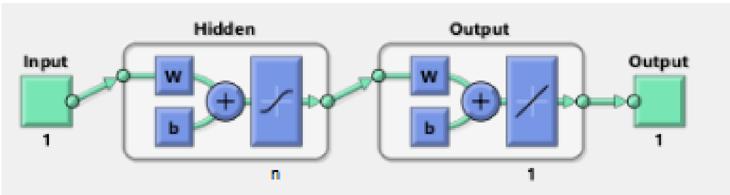
Aproksymowana funkcja

$$f(x) = \sin(3x+7) + \cos(2x+9)$$

$$x \in [-2, 5]$$

Liczność próby uczącej wynosiła kolejno 10, 20, 50, 100, 200. Próba testowa to zawsze 1000 elementów. Sieć składa się z warstwy wejściowej ukrytej, o n neuronach, oraz warstwy wyjściowej, o 1 neuronie, z funkcją aktywacji purelin, czyli liniową. Minimalny gradient wydajności jest ustawiony na 10^{-15}, a maksymalny czas uczenia na nieskończoność. Wartości te są zmienione, z powodu chęci uzyskania lepszych rezultatów uczenia.

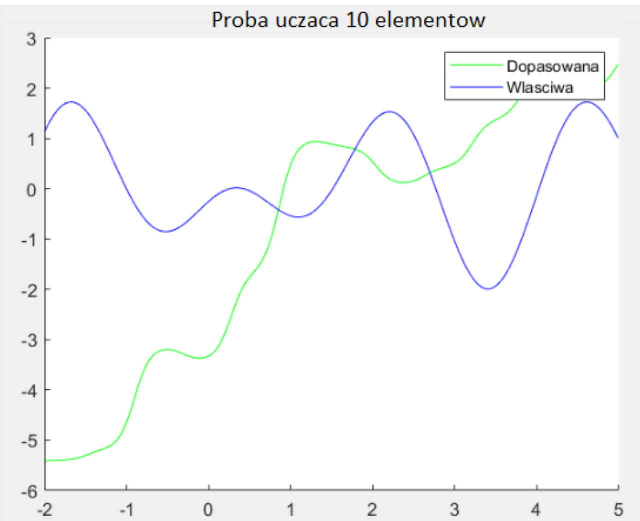
Schemat takiej sieci neuronowej, gdzie n to liczba neuronów ukrytych

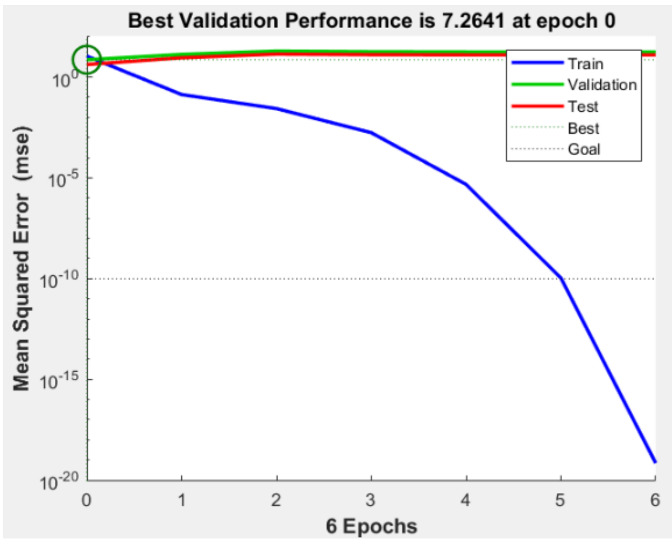


10 elementów uczących

Na początek ustawmy próbę uczącą na 10 elementów, a ilość neuronów warstwy ukrytej na 11. Funkcja aktywacji warstwy ukrytej to tansig.

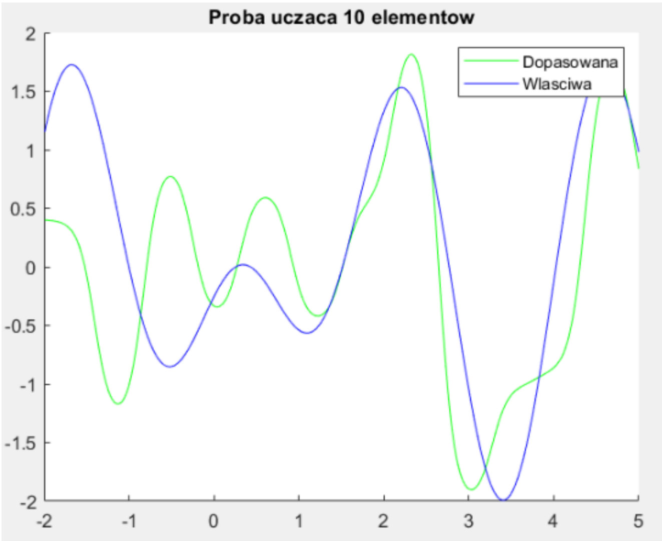
I przypadek – najgorsza sieć

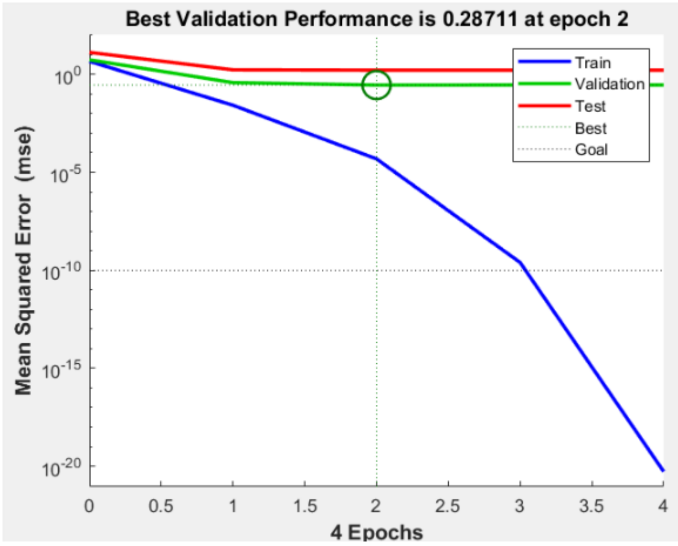




Średni błąd kwadratowy w przypadku danych testowych nie maleje (a nawet rośnie), a dla danych uczących maleje gwałtownie.

II przypadek – najlepsza sieć





Główną różnicą jest to, że błąd średniokwadratowy dla danych testujących maleje wraz ze wzrostem liczby epok.

Jak widzimy powstała funkcja nie jest dobrze dopasowana. Wykres zależności błędu uczenia (po prawej stronie) pokazuje nam, że pomimo dość małego błędu średniokwadratowego w wydajności danych uczących, funkcja źle określa wartości dla danych testowych. Spowodowane jest to małą ilością danych uczących i dużym zróżnicowaniem aproksymowanej funkcji, co uniemożliwia znalezienie dobrego rozwiązania.

Wzór powstałej funkcji

Znając wagi oraz biasy konkretnych warstw jesteśmy w stanie wyznaczyć wzór powstałej funkcji.

$$Q(x) = purelin[w_2 \sum_{i=1}^{n} [f_{aktywacji} * (w_{1,i} * x + bias_{1,i})] + bias_2]$$

gdzie,
purelin – funkcja aktywacji warstwy wyjściowej
$f_{aktywacji}$ - funkcja aktywacji warstwy ukrytej
n – liczba neuronów w warstwie ukrytej
$w_{1,i}$ – waga i-tego neuronu warstwy ukrytej
w_2 - waga warstwy wyjściowej
$bias_{1,i}$ – bias i-tego neuronu warstwy ukrytej
$bias_2$ – bias warstwy wyjściowej

Warto też dodać, że aby narysować wykres powstałej funkcji z tego wzoru musimy postąpić z parametrem x w taki sposób w jaki postępuje z nim sieć neuronowa. Argumenty wejściowe do sieci są najpierw normalizowane, wkładane do sieci a następnie denormalizowane. Musimy postąpić w ten sam sposób. Możemy użyć w tym celu funkcji mapminmax() oraz mapminmax.reverse().

W pierwszym przypadku
Q(x) = -0.64702(-0.28984 * tansig(-16.8x + 16.8)-0.10243 * tansig(16.8x -13.745) +0.27385 *tansig(16.8x -10.691) +0.2835 * tansig(16.8x -7.6364)-0.10233 * tansig(-16.8x + 4.5818)-0.22381* tansig(16.8x -1.5273)-0.04756 *

tansig(16.8x + 1.5273) +0.78744 * tansig(16.8x +4.5818) +0.52591 * tansig(16.8x + 7.6364) +0.090949 * tansig(-16.8x -10.691)-0.61183 * tansig(-6.8x -13.745)) -0.64702

Patrząc na funkcję Q(x) widzimy, że wagi warstwy ukrytej są mało zróżnicowane, co skutkuje złym działaniem sieci.

W drugim przypadku
Q(x) = 0.17979(0.4993 * tansig(-16.801x + 16.8) +0.81866 * tansig(16.799x -13.747) +0.032382 * tansig(16.8x-10.693) +0.30203 * tansig(16.692x -7.8799)-1.21 * tansig(16.799x -4.5961)-0.45478 * tansig(-16.767x + 1.762) +0.30449 * tansig(16.775x + 1.3165)-0.34679 * tansig(16.8x + 4.5824)-0.33803 * tansig(-16.754x -7.7352) +0.41016 * tansig(-16.801x -10.69)-0.66713 * tansig(-16.8x -13.745)) +0.17979

W tym przypadku wagi warstwy ukrytej są bardziej zróżnicowane czego efekt możemy zaobserwować na wykresie. Funkcja jest lepiej dopasowana.

Błąd aproksymacji średniokwadratowej punktowej

Możemy określić jakość naszego rozwiązania na podstawie obliczonego błędu aproksymacji średniokwadratowej punktowej, który wyraża się wzorem

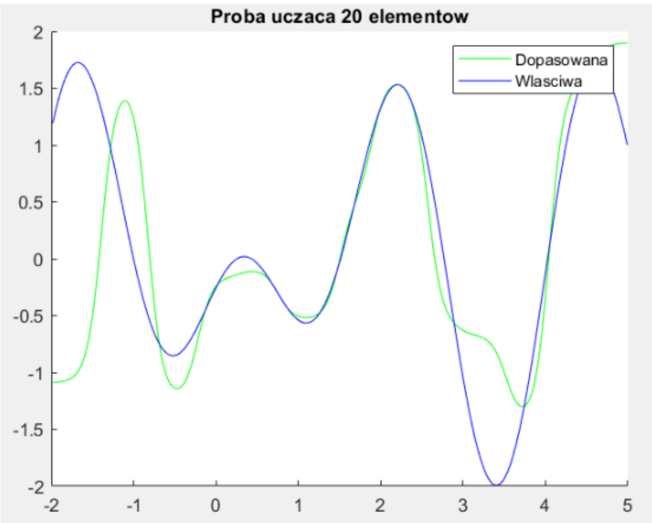
$$S=\frac{1}{n}\sum_{i=1}^{n}\left(f(x_i)-Q(x_i)\right)^2$$

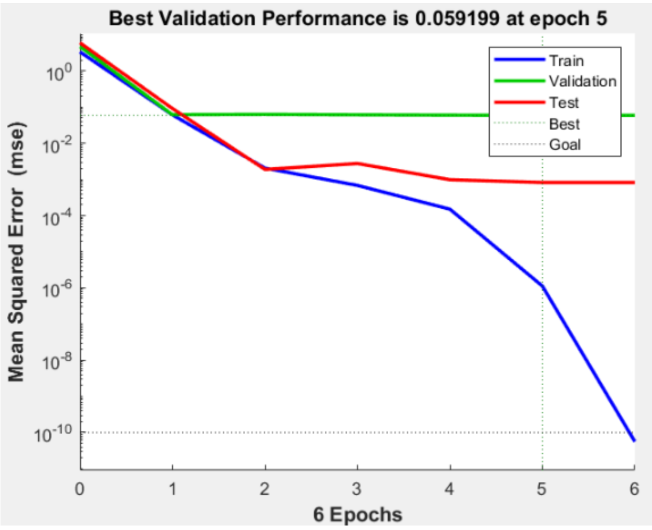
gdzie
f - funkcja oryginalna,
Q - funkcja dopasowana przez sieć neuronową

W pierwszym przypadku błąd wyniósł 8.5603
W drugim przypadku 0.71254

20 elementów uczących

I przypadek – najgorsza sieć



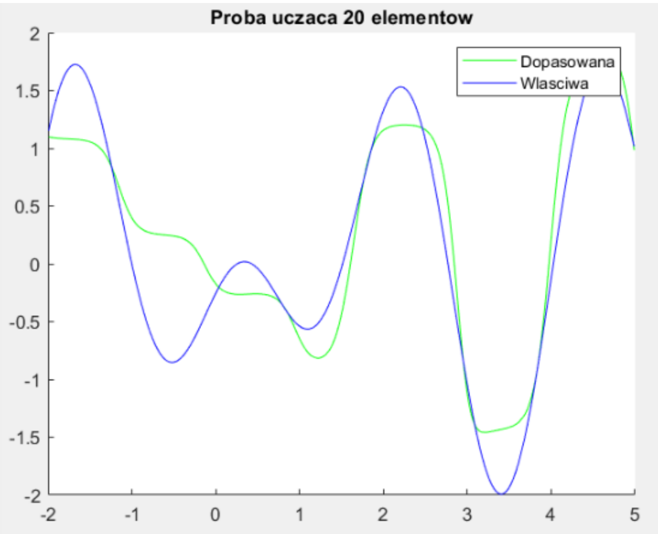


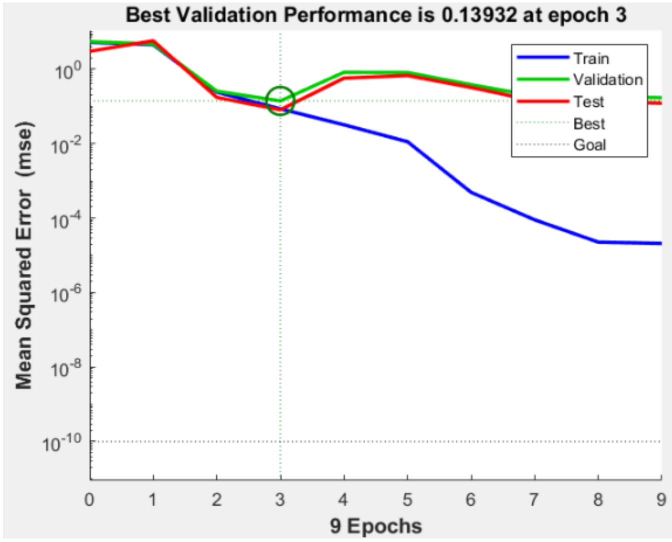
Q(x) = 0.12957(-0.112 * tansig(-16.774x + 16.826)-0.99858 * tansig(-16.782x + 13.771) +0.25667 * tansig(-16.75x + 10.76)-0.027485 * tansig(16.871x -7.4677) +0.72561 * tansig(-16.657x + 5.2587) +

0.35936 * tansig(16.565x -1.4337)-0.34875 * tansig(-16.699x -0.79775)-0.14871 * tansig(16.694x + 5.0556) +0.0242 * tansig(16.754x + 7.7155)-0.36111 * tansig(-16.798x -10.695)-0.94965 * tansig(16.593x + 14.011)) +0.12957
Błąd aproksymacji średniokwadratowej punktowej 0.7387

Błąd średniokwadratowy dla danych uczących w pewnym momencie nagle maleje przez co szybko osiąga on zadane minimum. Nie jest to korzystne, ponieważ zakończa to uczenie i uzyskujemy małą wydajność.

II przypadek – najlepsza sieć



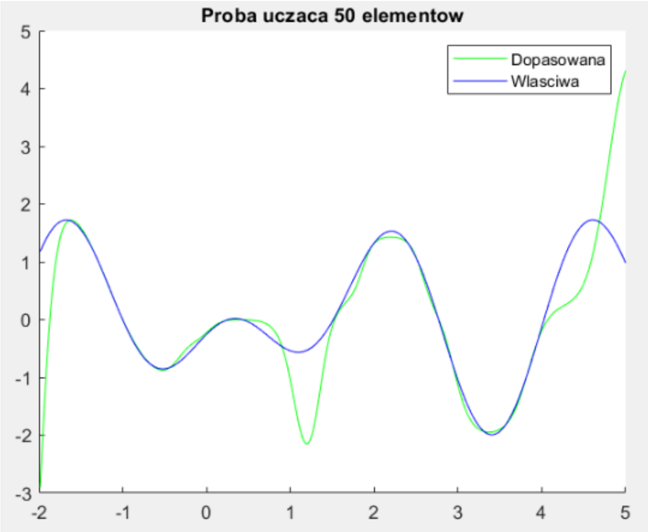


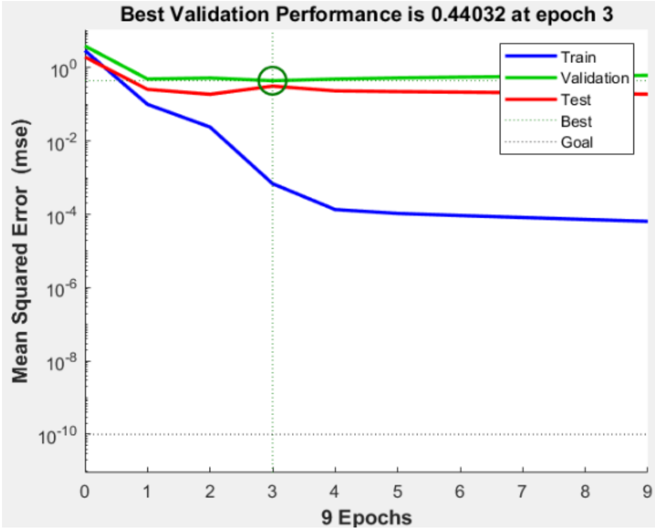
Q(x) = 1.2376(-0.83033 * tansig(16.605x -16.995)-0.066226 * tansig(-16.796x + 13.75)-0.95384 * tansig(-16.434x + 11.303)-0.3972 * tansig(-15.495x + 5.9155)-1.2044 * tansig(16.646x -5.7223)-0.4268 * tansig(-17.003x -0.67088)-0.2127 * tansig(-16.701x -1.4477)-0.19339 * tansig(16.798x + 4.5879) -0.0093168 * tansig(-16.765x -7.6963)-0.16188 * tansig(16.963x + 10.401)-0.25442 * tansig(15.769x + 14.768)) +1.2376
Błąd aproksymacji średniokwadratowej punktowej 0.20503

Patrząc na wykres widzimy, aproksymacja funkcji przebiegła lepiej niż dla 10 elementów uczących. Liczba epok jest podobna do wcześniejszych. Zmalał błąd średniokwadratowy dla danych testowych w I przypadku. Funkcja błędu dla danych uczących jest dalej gwałtowna. W II przypadku błędy średniokwadratowe zmieniają się łagodniej, jednak błąd ten dla danych testowych jest większy niż w najgorszym przypadku dla 20 elementów uczących. W obu przypadkach wagi neuronów są bardziej zróżnicowane niż poprzednio. Można zauważyć tutaj przeuczenie sieci, ponieważ w pewnym momencie błąd średniokwadratowy dla danych testowych rośnie.

50 elementów uczących

I przypadek – najgorsza sieć



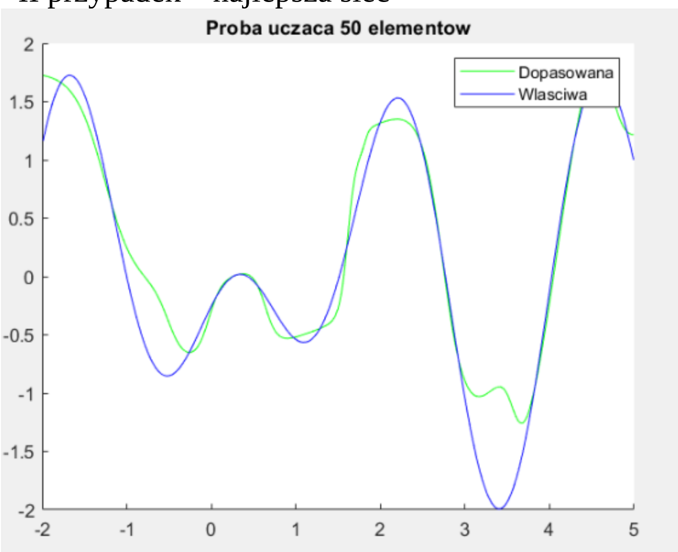


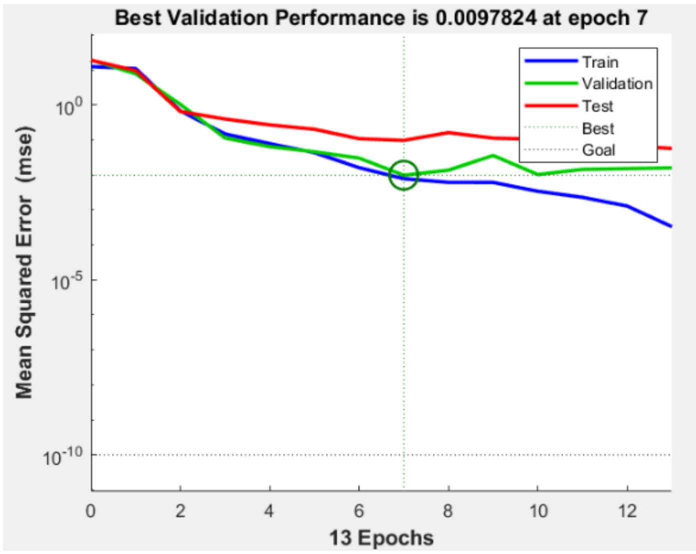
Q(x) = -1.0967(-0.84166 * tansig(-16.802x + 16.797)-0.41366 * tansig(-15.932x + 14.988)-0.62651 * tansig(-14.033x + 9.6619)-0.61841 * tansig(16.28x -6.7017) +0.33564 * tansig(-22.044x + 6.4068)-0.31159 * tansig(-24.129x + 1.6506) +0.92034 * tansig(-17.272x -3.1187) +1.0077 * tansig(19.249x + 1.8089) -0.11852 * tansig(-19.571x -9.8915)-0.23858 * tansig(-17.434x -10.854) +1.067 * tansig(-5.8818x -5.2052)) -1.0967

Błąd aproksymacji średniokwadratowej punktowej 0.50072

Już na samym początku wykresu widzimy, że błąd średniokwadratowy dla danych uczących maleje a dla testowych już nie. W takich przypadkach uzyskujemy małą wydajność, co zaobserwowaliśmy też w tym przypadku.

II przypadek – najlepsza sieć





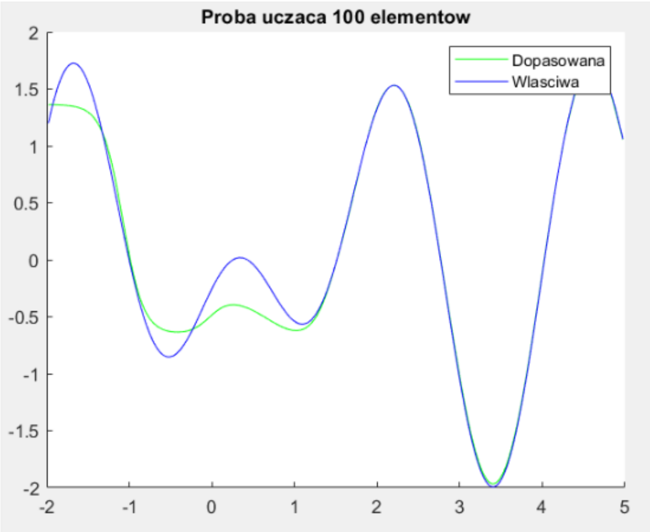
Q(x) = 0.91686(-0.48482 * tansig(16.819x -16.968)-1.5146 * tansig(-5.7024x + 4.6695)-0.3601 * tansig(21.943x -14.378)-0.44444 * tansig(-1.3559x + 0.16984)-0.84271 * tansig(12.336x -4.9038) -0.050436 * tansig(-63.192x + 7.0708)-0.0055704 * tansig(13.427x + 1.3129) +0.20425 * tansig(-22.121x -5.6157)-0.18509 * tansig(-21.694x -9.9837)-0.19917 * tansig(16.871x + 10.24)-0.54974 * tansig(8.8786x + 7.4685)) +0.91686

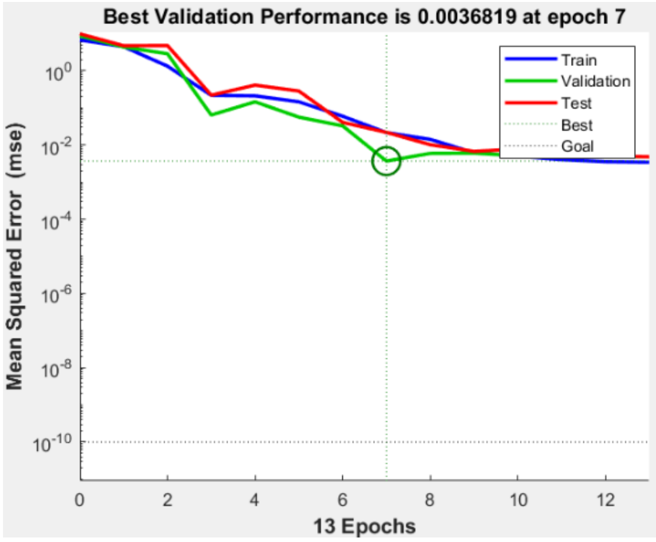
Błąd aproksymacji średniokwadratowej punktowej 0.088929

Błędy średniokwadratowe maleją w miarę jednostajnie.

100 elementów uczących

I przypadek – najgorsza sieć

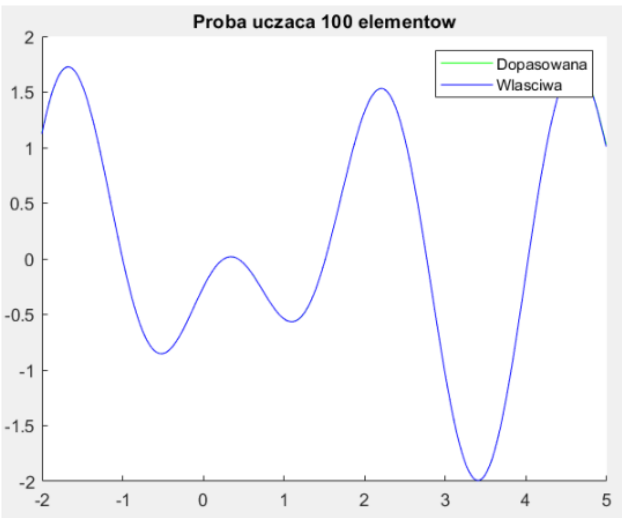


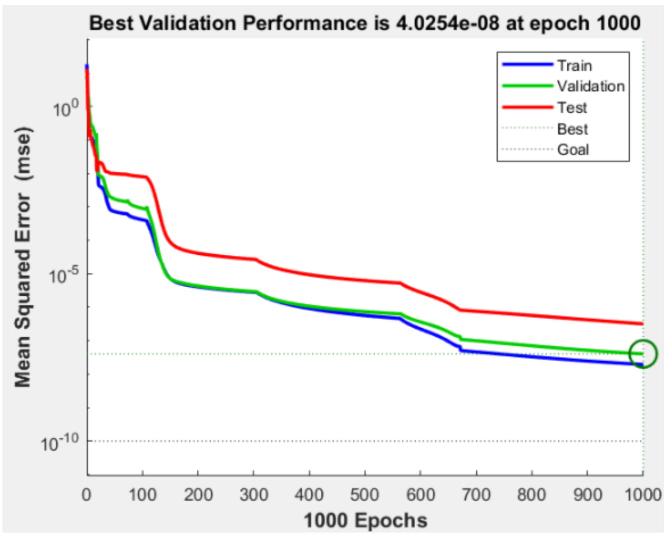


Q(x) = 0.6379(-0.67922 * tansig(9.9585x -9.7293) +1.6374 * tansig(5.9686x -4.2049)-0.0015764 * tansig(26.567x -14.738) +1.0285 * tansig(-7.457x + 3.2279) +0.43059 * tansig(-9.8019x + 3.1082) + 0.67755 * tansig(8.6521x -0.67817) +0.088958 * tansig(16.958x + 0.50478) +0.13559 * tansig(-7.0343x -1.5903) +0.095795 * tansig(14.744x + 6.4207)-0.54232 * tansig(13.996x + 10.336) +1.6632 * tansig(17.768x + 42.042)) +0.6379
Błąd aproksymacji średniokwadratowej punktowej 0.026922

Patrząc na wykres widzimy, że funkcja w prawie wszystkich punktach jest dobrze dopasowana. Możemy domyślać się, że dane uczące znajdowały się głównie po prawej stronie dziedziny, ponieważ głównie tam funkcja jest dobrze dopasowana.

II przypadek – najlepsza sieć





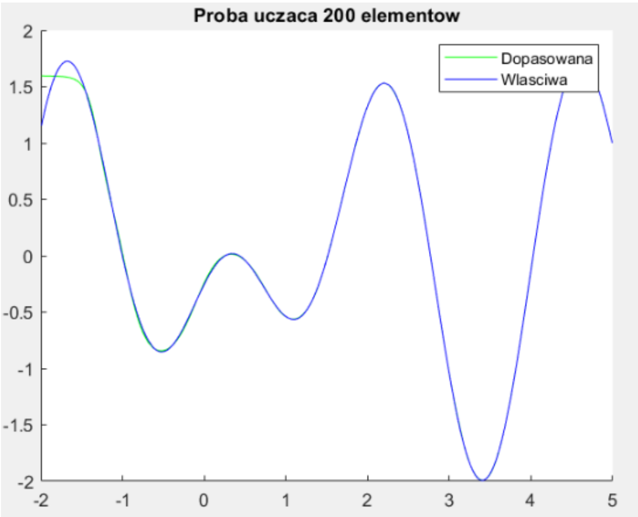
Q(x) = 1.6362(11.975 * tansig(-3.2024x + 3.3262) +44.967 * tansig(1.5892x -0.92242) +0.00095711 * tansig(-29.593x + 17.413) +0.00015228 * tansig(106970x -48111) +0.54411 * tansig(4.8997x -0.64205) +

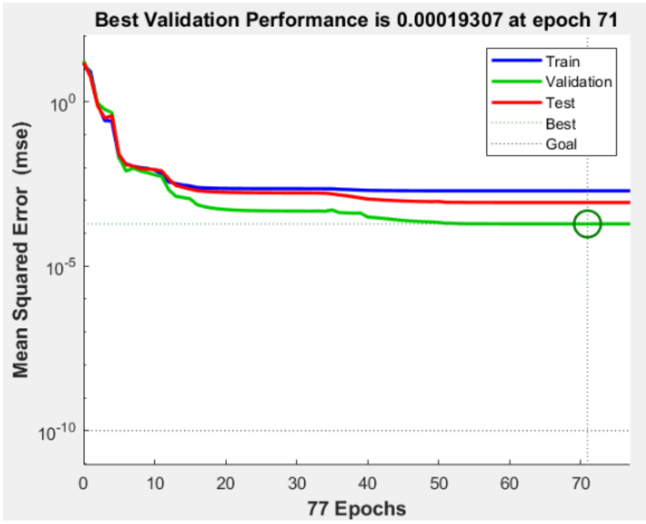
26.602 * tansig(-2.7458x + 1.2363)-0.00056608 * tansig(1483.6x + 111.94) +4.2006 * tansig(-3.7469x -0.65989) +0.57098 * tansig(5.2705x + 2.4407) +1.8961 * tansig(-5.0607x -3.7581) +0.056585 * tansig(-11.004x -9.0403)) +1.6362
Błąd aproksymacji średniokwadratowej punktowej 3.3032e-06

Tym razem uczenie trwa długo, ale dzięki temu uzyskujemy bardzo mały błąd aproksymacji. Nie jesteśmy w stanie zaobserwować dwóch wykresów. Są one niemalże identyczne. Wagi neuronów są zróżnicowane, a w niektórych przypadkach są one bardzo duże.

200 elementów uczących

I przypadek – najgorsza sieć

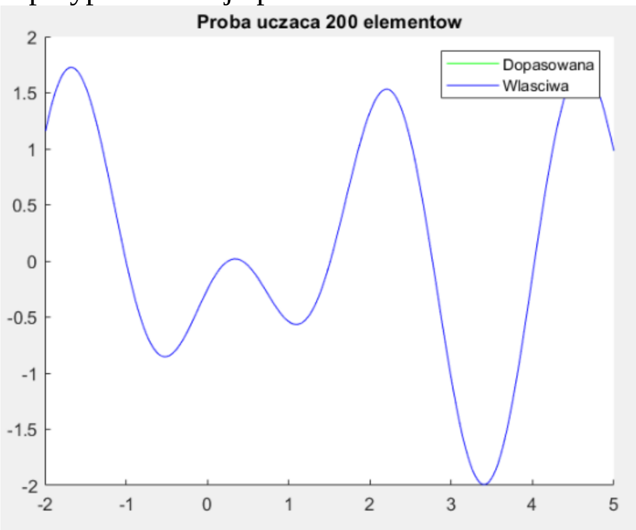


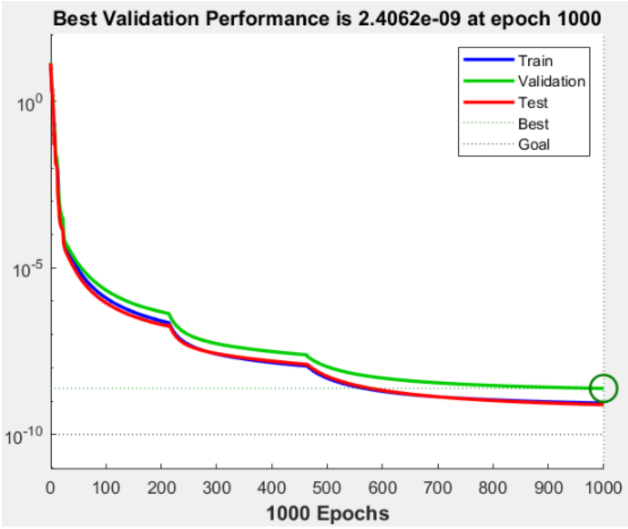


Q(x) = 0.33913(-1.1015 * tansig(6.8477x -6.8788) +1.8969 * tansig(5.6604x -4.0347)-0.02397 * tansig(-19.611x + 11.995) +0.0042477 * tansig(-124.34x + 59.218)-1.0017 * tansig(-6.665x + 0.61852) + 0.057319 * tansig(15.627x + 0.11105)-1.8547 * tansig(5.5624x -2.1393) +0.29116 * tansig(-9.5837x -1.99) -0.30455 * tansig(-11.052x -5.0521)-0.46002 * tansig(13.618x + 9.5918) +0.23128 * tansig(-23.95x -19.208)) +0.33913
Błąd aproksymacji średniokwadratowej punktowej 0.0023125

Wykresy prawie idealnie się pokrywają. Wagi są bardzo zróżnicowane. Błędy średniokwadratowe są stabilne i równo zmierzają do swojego minimum. Co ciekawe błąd średniokwadratowy dla danych testujących jest mniejszy niż dla danych uczących.

II przypadek – najlepsza sieć





Błąd aproksymacji średniokwadratowej punktowej 1.508e-09
Błąd ten jest najniższy spośród powyższych przypadków, dlatego możemy uznać, że poniższy wzór najlepiej opisuje aproksymowaną funkcję.

Q(x) = -3.6575(-2.254 * tansig(4.5374x -10.074)-0.00013883 * tansig(-66.534x + 50.518)-2.701 * tansig(4.4398x -4.3947) +5.5935 * tansig(3.5918x -2.4817) +6.6861 * tansig(-3.3647x + 1.3075) + 2.4101 * tansig(3.9924x + 1.9003) +5.0999 * tansig(3.4404x -0.3143) +2.9514 * tansig(-3.7341x -0.7279) -0.00023792 * tansig(28.069x + 12.779)-2.8438 * tansig(4.1573x + 3.1686)-0.00047174 * tansig(28.505x + 23.342)) -3.6575

Nie jesteśmy już w stanie zaobserwować już dwóch wykresów. Funkcja została bardzo dobrze aproksymowana. Uczenie trwało bardzo długo, bo aż przez wszystkie możliwe epoki, czyli 1000. Spadek błędów średniokwadratowych był stabilny. Wagi neuronów są bardzo zróżnicowane.

Za każdym razem, gdy zwiększaliśmy liczbę elementów uczących rosła dokładność i wydajność sieci. Obserwując wykresy zależności błędu uczenia/testowania od epok dostrzegamy, że im lepsza jest sieć tym łagodniej osiąga najmniejsze najmniejszy możliwy błąd średniokwadratowy. Dla małej próby uczącej algorytm szybko kończył swoje działanie. Im większa była ta próba tym dłużej trwało uczenie.

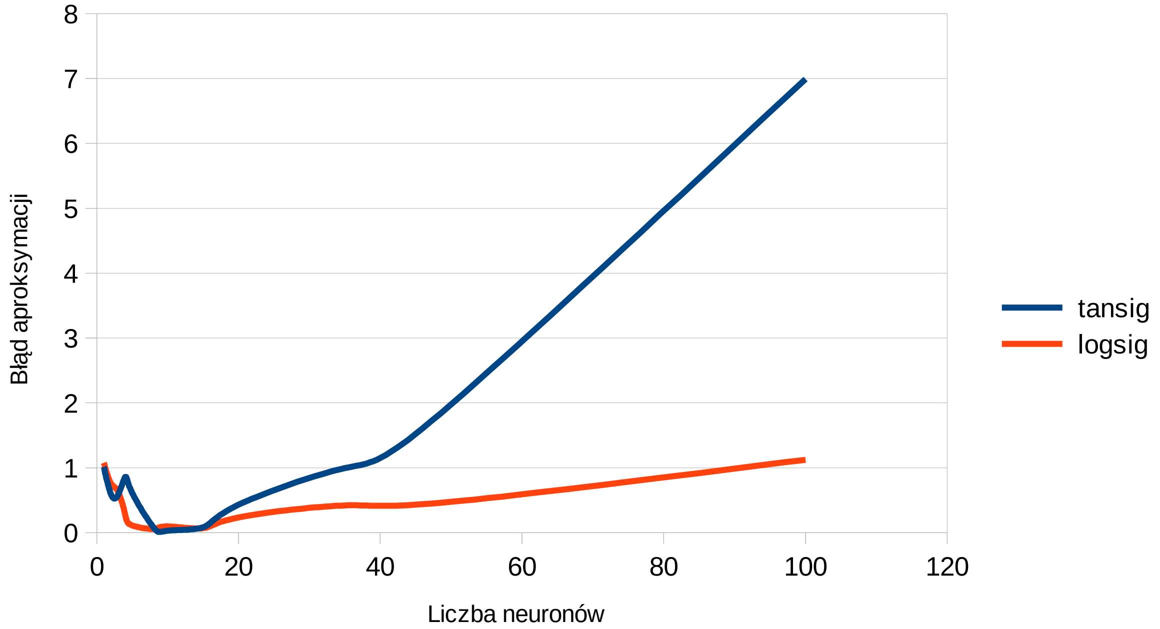
Wpływ ilości neuronów ukrytych oraz rodzaju funkcji aktywującej warstwy ukrytej na sieć neuronową

Teraz przetestujemy wpływ ilości neuronów ukrytych oraz rodzaju funkcji aktywującej warstwy ukrytej na sieć neuronową. Testy będziemy przeprowadzać na sieci z 20 elementami uczącymi w 5 iteracjach sieci. Do tabeli jest wybierana sieć z najbardziej korzystnym błędem aproksymacji średniokwadratowej punktowej spośród tych 5 iteracji.

Liczba neuronów	**Funkcja aktywacji**	
	tansig	**logsig**
1	1.0136	1.0763
2	0.5930	0.7595
3	0.5811	0.6334
4	0.8566	0.2817
5	0.6124	0.1105
8	0.0819	0.0583
10	0.0298	0.0958
15	0.0838	0.0686
20	0.4292	0.2344
35	0.9942	0.4193
50	1.9772	0.4750
100	6.9933	1.1242

Widzimy, że funkcja logsig działa przeważnie lepiej niż funkcja tansig. Obie funkcje mają podobną najlepsza wydajność dla podobnej ilości neuronów, tj. 8-10. Obserwujemy też zależność, że do pewnego momentu wzrost liczby neuronów wpływa pozytywnie na wydajność, jednak po przekroczeniu pewnej ich liczby wydajność spada. Sieć ma trudności we właściwym wyznaczeniu wag dla coraz większej liczby neuronów. Trzeba też zauważyć, że liczba epok uczenia spada wraz ze wzrostem liczby neuronów. Sieć szybciej stwierdza, że osiągnęła swoje cele. Warto się zastanowić czego oczekujemy od naszej sieci i jakie dane możemy jej zaoferować jako uczące. Po określeniu swoich celów warto zbadać, kiedy i jakim kosztem osiągniemy najlepszą wydajność i czy jest ona tego warta. Nie warto jednak stosować zbyt dużej liczby neuronów, gdy ma się mało danych wejściowych.

Wyniki powyższych testów można przedstawić na wykresie, co pozwala na zaobserwowanie dla jakiej liczby neuronów uzyskujemy najlepszą wydajność (9 neuronów ukrytych).



Jakub Salamon, 297914
Informatyka Stosowana, WFIIS, II rok