Informatyka, studia zaoczne, inż. III rok semestr VI
Inteligentna analiza danych **2015/2016**
prowadzący: dr inż. Bartłomiej Stasiak

Data oddania:_____ Ocena:_____

Marek Gadzalski 191422
Grzegorz Głąb 191425

Zadanie 2b. Perceptron wielowarstwowy

1. CEL

Celem zadania jest implementacja perceptronu wielowarstwowego (ang. Multi-Layer Perceptron, w skrócie: MLP) oraz wsteczną propagację błędów, jako metodę jego nauki.

2. WPROWADZENIE

2.1. Wstęp

Siec neuronowa składa się z warstw, każda warstwa zawiera pewna ilość neuronów. Każdy neuron posiada dowolna liczbę wejść oraz jedno wyjście. Sąsiednie warstwy są połączone iloczynem kartezjańskim także na wejście każdego neuronu trafiają wyjścia wszystkich neuronów z warstwy poprzedniej.
Omawiana tutaj sieć neuronowa posiada jedną warstwę neuronów kopiujących (Input Layer), wiele (jedną lub więcej) warstw ukrytych (Hidden Layers), oraz jedną warstwę wyjściową (Output Layer).

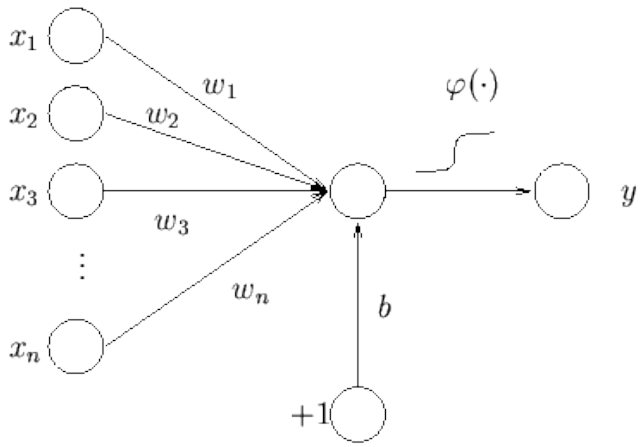


Figure 1: Neuron

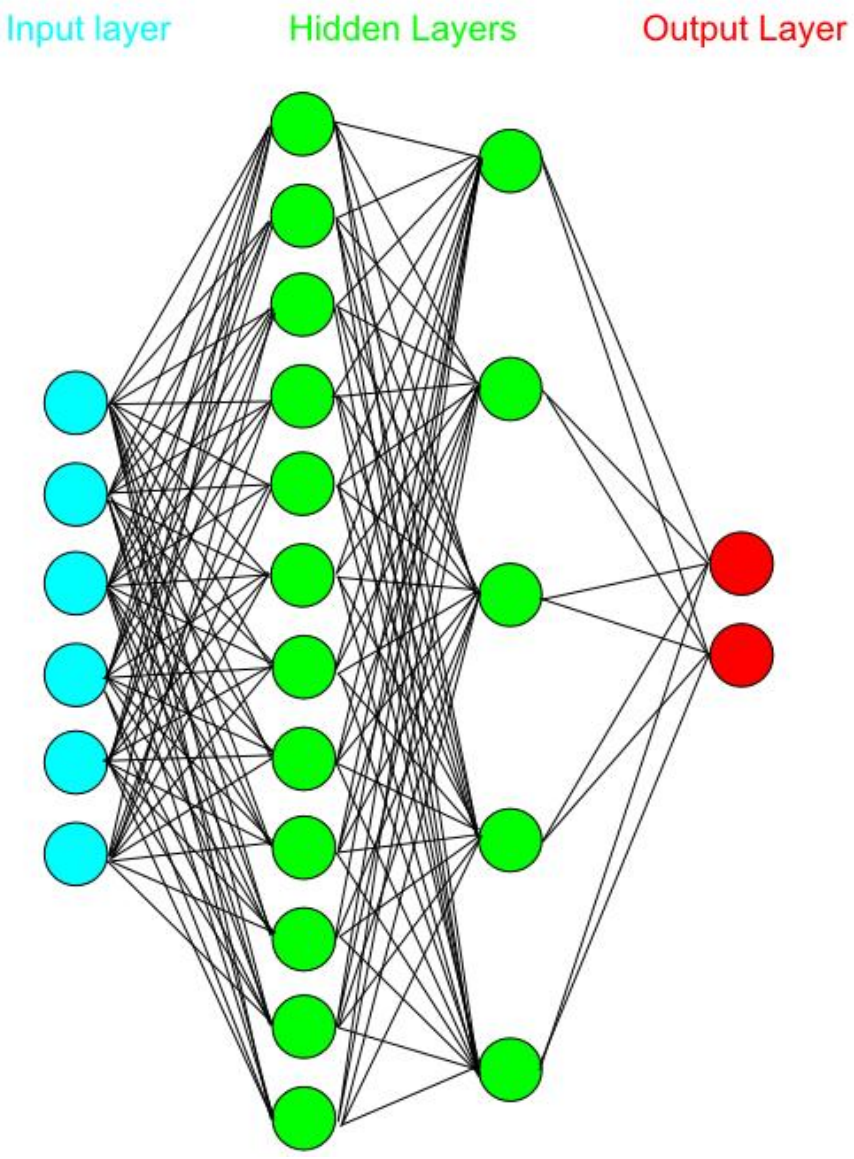


Figure 2: Perceptron Wielowarstwowy

2.2. Przetwarzanie sieci.

Aby uzyskać wyniki sieci wymagane jest przetworzenie każdego neuronu i obliczenie jego wyjścia. Wyjście neuronu obliczamy ze wzoru:

$$O = f\left(\sum_{i=1}^{n} \omega_i * x_i\right)$$

Gdzie:
O - wyjście neuronu
f - funkcja aktywacji neuronu
wi - waga na i-tym wejściu neuronu
xi - wartość na i-tym wejściu

W naszym przypadku pierwsza warstwa (nieprzetwarzająca) składa się z neuronów powielających, które przekazują dalej informacje, które otrzymują.
Neurony warstw ukrytych oraz warstwy wyjściowej są neuronami nieliniowymi - w tym przypadku zawsze oznacza to zastosowanie sigmoidalnej funkcji aktywacji o wzorze:

$$\mathrm{logsig(x)} = \frac{1}{1 + \mathrm{e}^{-\mathrm{x}}}$$

Zaletą logistycznej funkcji aktywacji jest łatwość obliczania jej pochodnej:

$$\mathrm{logsig'(x)} = \mathrm{logsig(x)} \cdot (1 - \mathrm{logsig(x)})$$

Wspomniane wzory opisują proces pozwalający obliczyć wyjścia wszystkich warstw oraz samo wyjście sieci na podstawie zadanego wejścia.

2.3. Wsteczna propagacja błędów.

Wsteczna propagacja błędu służy do nauki sieci neuronowej. Polega na liczeniu błędów każdego z neuronów zaczynając od warstwy wyjściowej. Wartość błędów neuronów na każdej warstwie jest zależna od wartości błędów na warstwie następnej. Mając wyliczone wartości błędu dla każdego neuronu możemy wyznaczyć modyfikacje wag wejść danej warstwy.

Wzór na błąd w warstwie wyjściowej:

$$\mathrm{E} = (\mathrm{I} - \mathrm{O}) * \mathrm{logsig'(x)}$$

Gdzie:
— E - wartość błędu
— I - oczekiwane wyjście
— O - otrzymane wyjście

Wzór na błąd na pozostałych warstwach:

$$\mathrm{E} = \left(\sum_{i=1}^{k} W_i * e_i\right) * \mathrm{logsig'(x)}$$

Gdzie:
— E - wartość błędu

— k - ilość neuronów w warstwie następnej
— W_i - waga wejścia powiązanego z wyjściem danego neuronu i-tego neuronu w warstwie następującej
— e_i - wartość błędu osiągniętego przez neuron, na którego wejście trafiają dane z aktualnego neuronu

Mając policzone błędy możemy obliczyć modyfikacje wag wg wzoru:

$$\delta_i = n * E * x_i + m * \delta^{prev}$$

Gdzie:
— n - współczynnik uczenia
— E - błąd
— m - współczynnik momentu
— δ^{prev}- wartosc z poprzedniego etapu procesu nauki

Nauka sieci odbywa się poprzez wielokrotnego obliczenia wyjść dla wejścia ze zbioru uczącego, a następnie porównania wyjścia z wyjściem oczekiwanym, propagacji błędu i aktualizacji wag. Sieć uczona jest w oparciu o wszystkie testy z zestawu, jednak kolejność może być losowana. Zastosowanym przez nas rozwiązaniem jest opcjonalne losowanie permutacji testów ze zbioru uczącego tak, aby każdy z nich był używany jednakowo często. Użycie do nauki całej permutacji elementów zbioru testowego nazywamy epoką. Dla każdego testu obliczany jest globalny błąd sieci, jako połowa sumy kwadratów między wejściami oczekiwanymi a otrzymanymi. Za wartość miarodajną postępów w nauce sieci uznajemy maksymalny globalny błąd sieci dla pojedynczego testu w danej epoce. Warunkiem zakończenia nauki jest, zatem dla nas osiągnięcie odpowiedniej epoki lub maksymalnego globalnego błędu w epoce poniżej zadanej wartości.

3. IMPLEMENTACJA

Aplikacja została napisana w języku Java z wykorzystaniem graficznego interfejsu Swing. Kod programu został podzielony na funkcjonalne pakiety

Pakiet net – zawiera model danych, składający się z klas:
Network – sieć, która zawiera kontener połączonych ze sobą warstw. Główna logika przetwarzania i uczenia się sieci jest zaimplementowane również w tej klasie. Jest to najistotniejsza klasa pod względem merytorycznym.
Layer – warstwa sieci, która zawiera kontener neuronów w warstwie
Neuron – jest to podstawowa encja, która implementuje nasz opisany wcześniej neuron.
TrainingData – klasa zawiera kontenery danych treningowych, walidacyjnych i testowych.
Pakiet func – zawiera implementacje funkcji aktywacyjnych i ich pochodnych
Pakiet services – Zawiera implementacje funkcjonalności pomocniczych, takich jak odczyt/zapis do pliku, obliczenia statystyk, tworzenie podzbiorów danych i inne dodatkowe funkcjonalności.
Pakiet GUI – zawiera implementacje prostego interfejsu użytkownika do obsługi programu.

4. MATERIAŁY

4.1. Dane

Materiałem był zbiór irisData zawierający dane dotyczące budowy morfologiczne kwiatów trzech gatunków irysów.
Na potrzeby części testów zbiór ten został podzielony na trzy rozdzielne podzbiory :

- zbiór treningowy: zbiór służący do obliczania błędów wykorzystanych w algorytmie wstecznej propagacji
- zbiór walidacyjny: przykłady tego zbioru prezentowane są sieci podczas nauki, a obliczony na jego podstawie błąd jest wykorzystywany do kontrolowania procesu nauki
- zbiór testowy: przykładu tego zbioru wykorzystywane są do klasyfikacji jakościowej i oceny zdolności sieci rozpoznawania wzorców

Program umożliwia podział danych w proporcjach:

- 40% zbiór testowy, 30% zbiór walidacyjny, 30% zbiór testowy
- 66% zbiór testowy, 18% zbiór walidacyjny, 18% zbiór testowy

Dodatkowo dane dzielono:

- sekwencyjnie - pierwsza część z każdej klasy stanowiła zbiór testowy, druga walidacyjny, a trzecia testowy
- losowo - każdy podzbiór zawierał losowe przykłady z każdej klasy

Program umożliwia także wykluczanie wybranych cech z procesu nauki.

4.2. Sieć

Liczba neuronów wejściowych sieci była równa liczbie parametrów w zbiorze danych (4). Natomiast liczba neuronów wyjściowych była równa liczbie klas w zbiorze danych: trzy. Klasy były reprezentowane przez następujące kombinacje odpowiedzi:

- Iris-setosa (1, 0, 0)
- Iris-versicolor (0, 1, 0)
- Iris-virginica (0, 0, 1)

Jakościowa klasyfikacja przykładów polegała na wyborze najbardziej pobudzonego neuronu wyjściowego.
Wewnętrzna struktura sieci była modyfikowana podczas testów.

5. WYNIKI I WNIOSKI

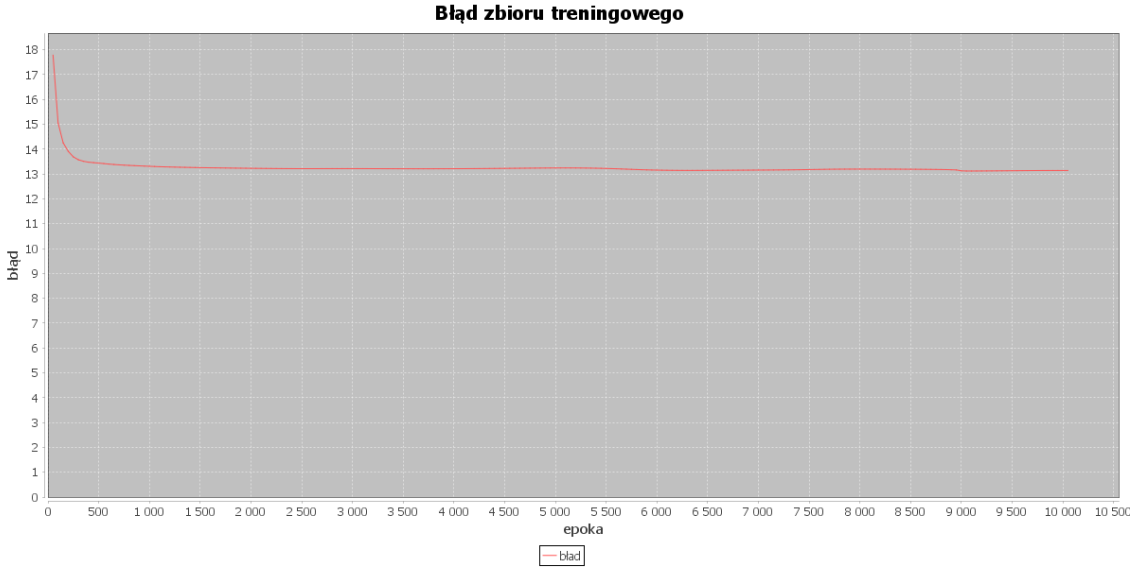
Parametry wspólne dla wszystkich przeprowadzonych doświadczeń:

parametr	wartość
liczba epok	10000
współczynnik nauki	0,3
człon momentum	0,8

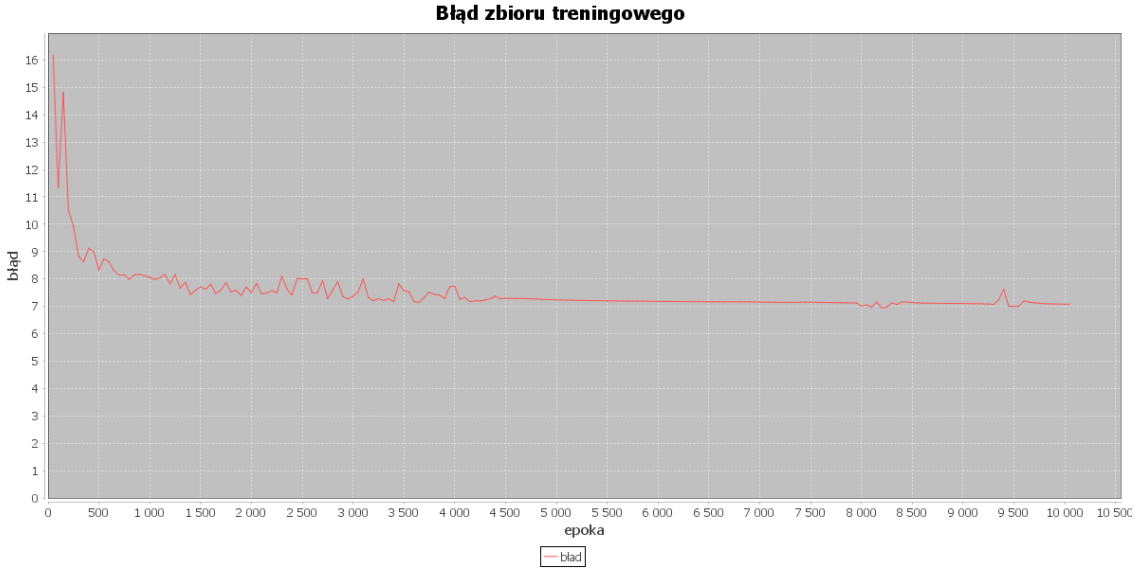
5.1. Badanie topologii sieci

Pierwszy zadaniem było ustalenie wpływu topologii warstw ukrytych na proces nauki

parametr	wartość
topologia warstw ukrytych	(1)
podział zbioru danych	100% trening

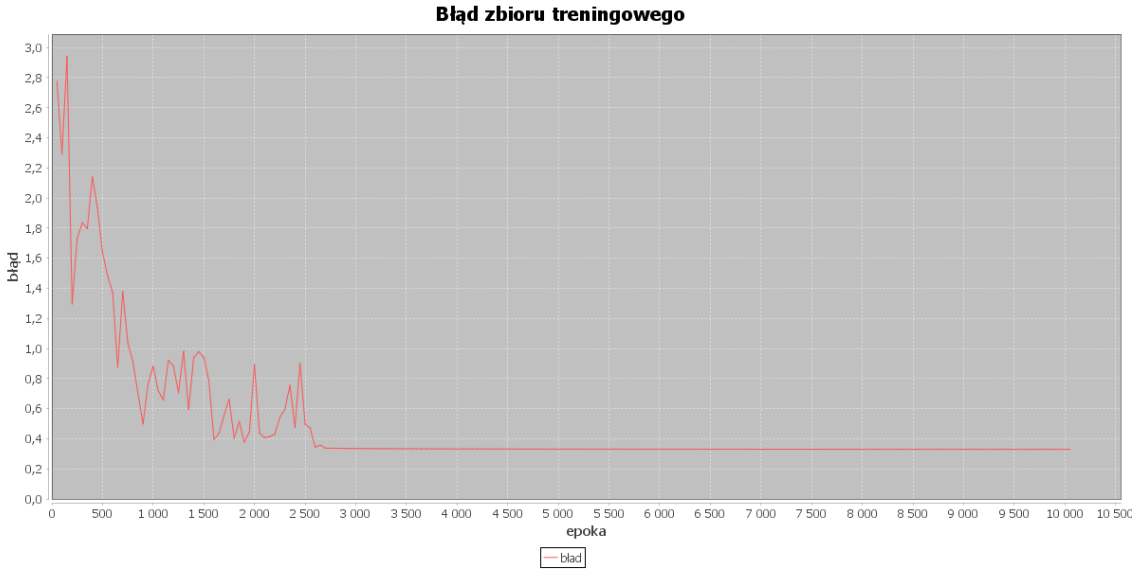


rametr	wartość
topologia warstw ukrytych	(2)
podział zbioru danych	100% trening



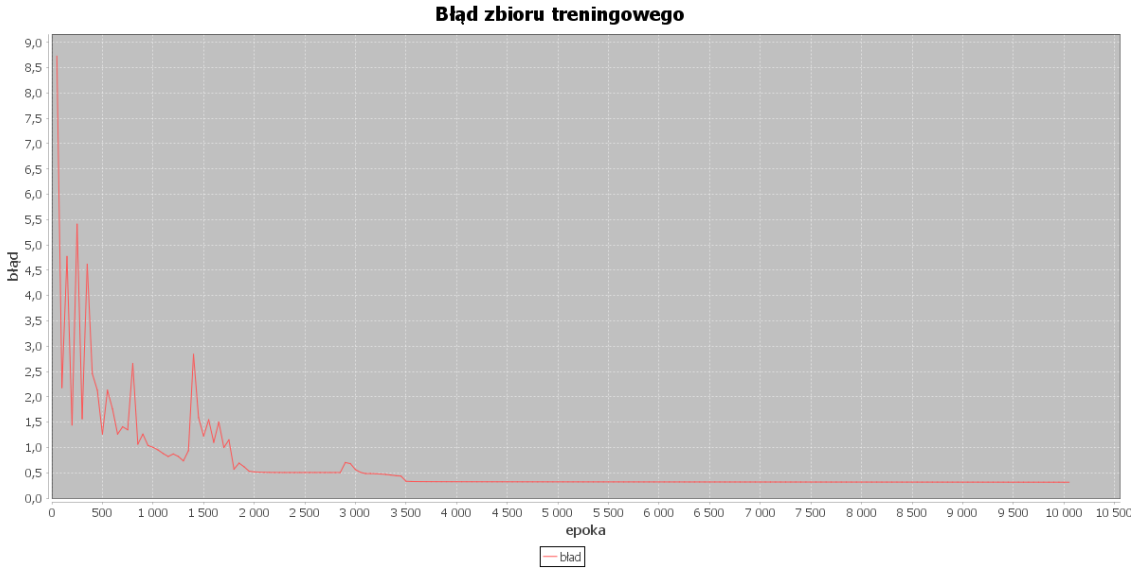
parametr	wartość
topologia warstw ukrytych	(3)
podział zbioru danych	100% trening

parametr	wartość
topologia warstw ukrytych	(8)
podział zbioru danych	100% trening



parametr	wartość
topologia warstw ukrytych	(16)
podział zbioru danych	100% trening

parametr	wartość
topologia warstw ukrytych	(32)(32)
podział zbioru danych	100% trening



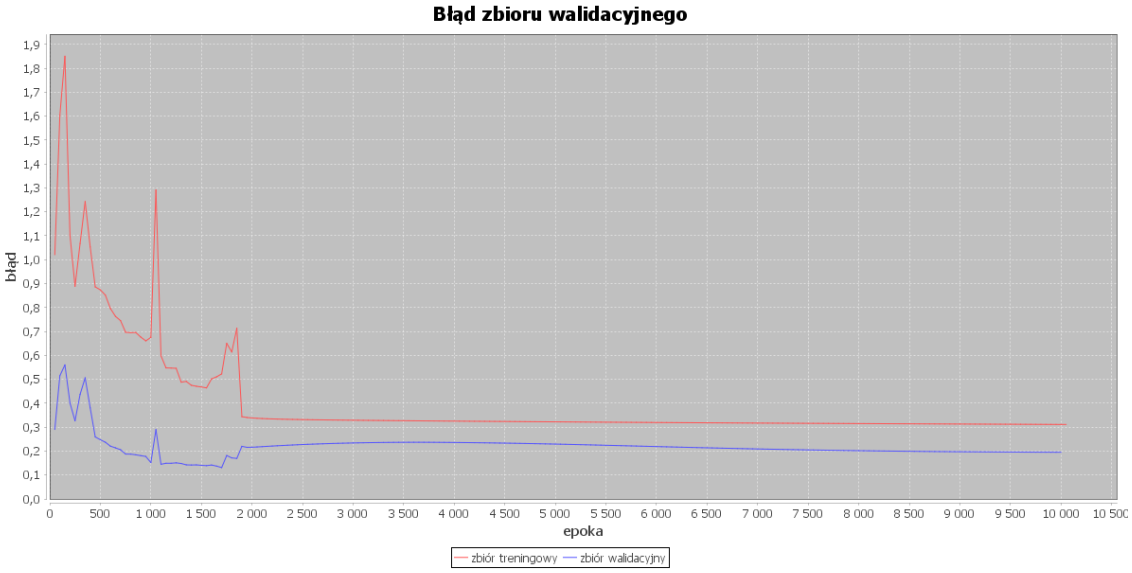
parametr	wartość
topologia warstw ukrytych	(8)(8)(8)
podział zbioru danych	100% trening

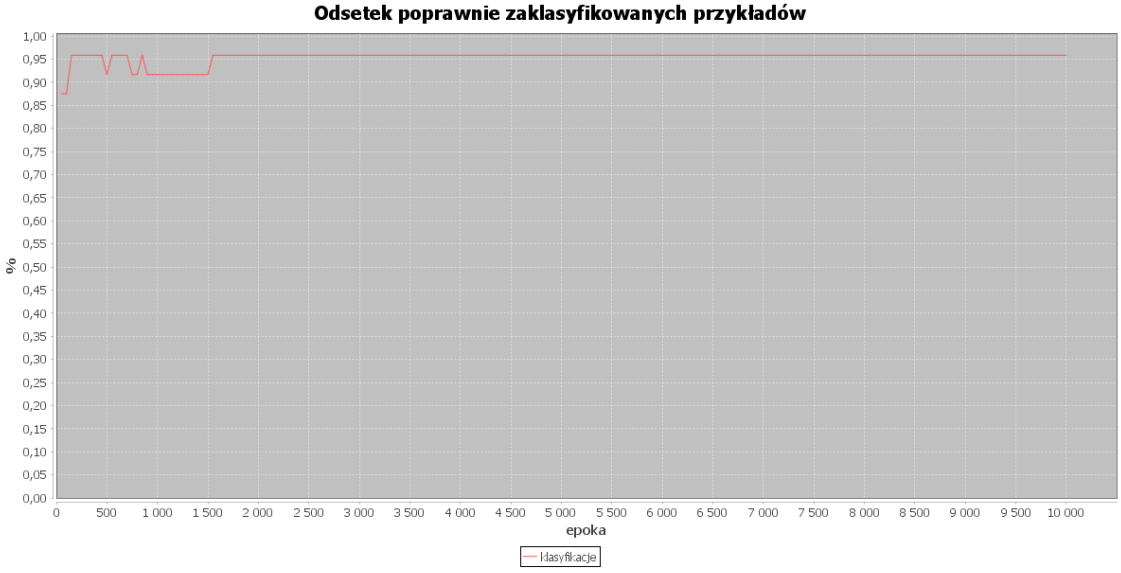
Wnioski

Aby sieć osiągała akceptowalny poziom błędu podczas treningu warstwa ukryta musi posiadać co najmniej trzy neurony. Natomiast zwiększanie liczby neuronów powyżej ośmiu, a także zwiększanie liczby warstw ukrytych nie zwiększa zdolności uczenia się sieci.
W związku z powyższym wszystkie kolejne doświadczenia będą prowadzone przy **pojedynczej warstwie ukrytej zbudowanej z ośmiu neuronów.**

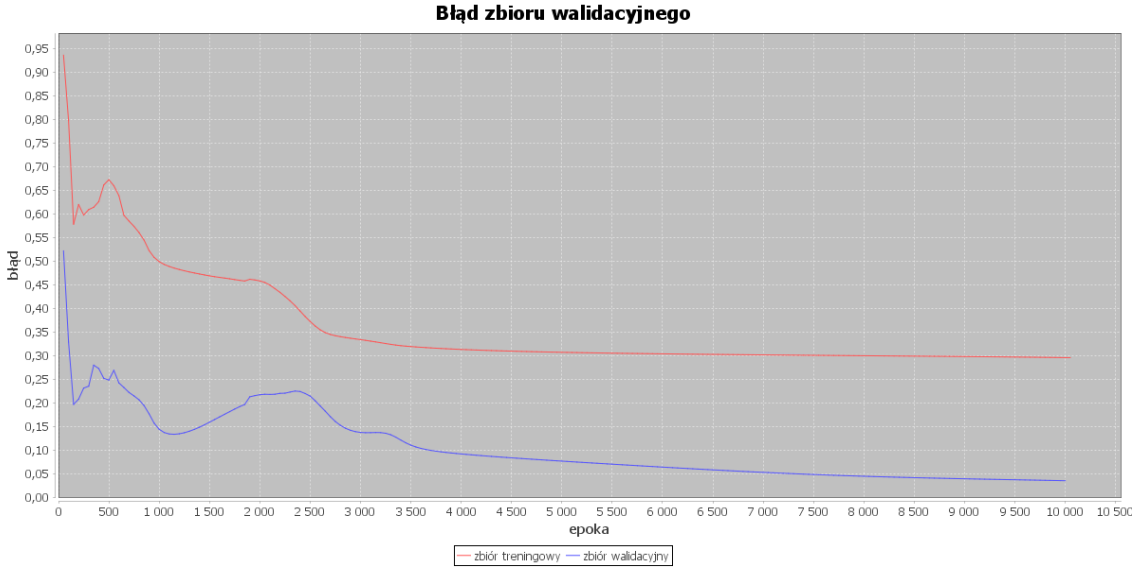
5.2. Badanie wpływu podziału danych na podzbiory na zdolność uczenia się sieci

parametr	wartość
topologia warstw ukrytych	(8)
podział zbioru danych	66% trening, 18% walid., 18% test. (sekwencyjnie)

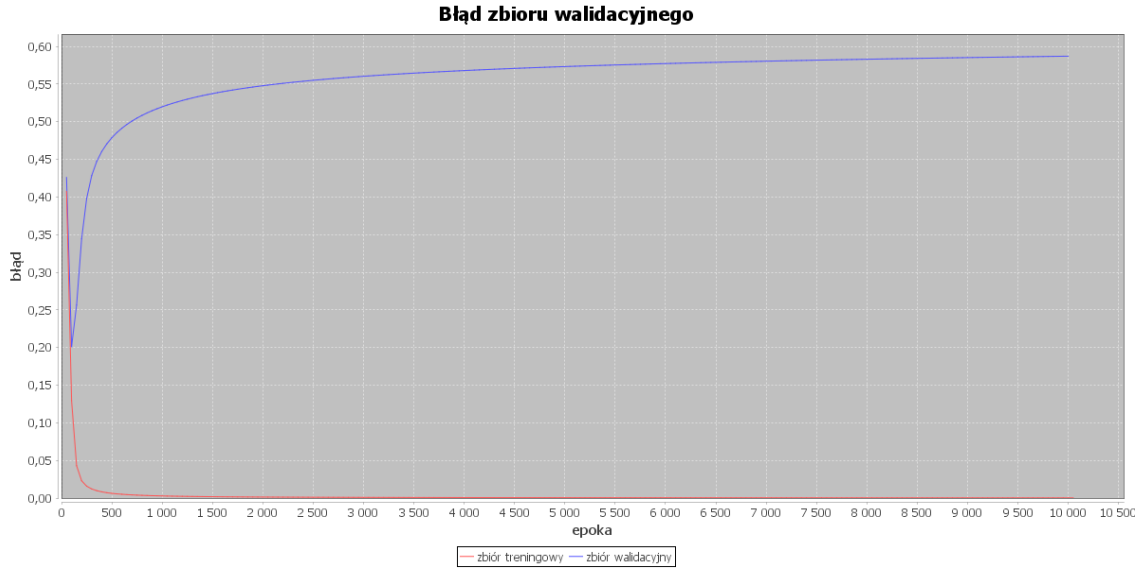




parametr	wartość
topologia warstw ukrytych	(8)
podział zbioru danych	66% trening, 18% walid., 18% test. (losowo)



parametr	wartość
topologia warstw ukrytych	(8)
podział zbioru danych	40% trening, 30% walid., 30% test. (losowo)



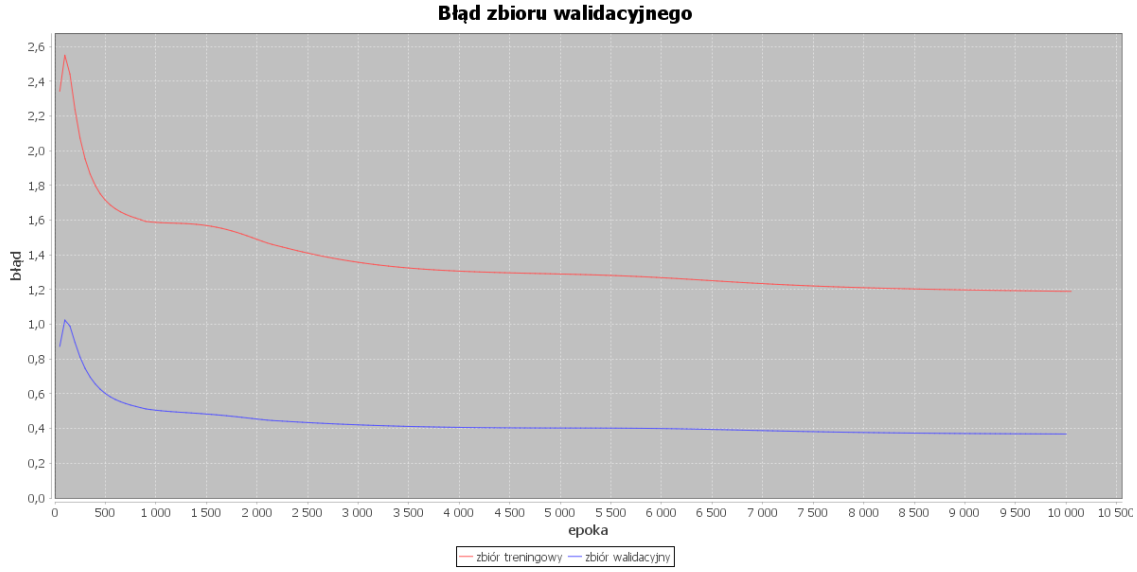
Wnioski

Nie zaobserwowano wpływu sposobu wyboru przykładów do podzbiorów (sekwencyjny i losowy). Natomiast wielkość zbioru treningowego ma znaczny wpływ na zdolność generalizowania sieci. Zmniejszenie wielkości zbioru treningowego powoduje znaczne przyśpieszenie nauki sieci, jednak analizując błąd na zbiorze walidacyjnym obserwuje się wyraźne zjawisko przeuczenia sieci. Zmniejsza się również odsetek poprawnie zaklasyfikowanych przykładów ze zbioru testowego.

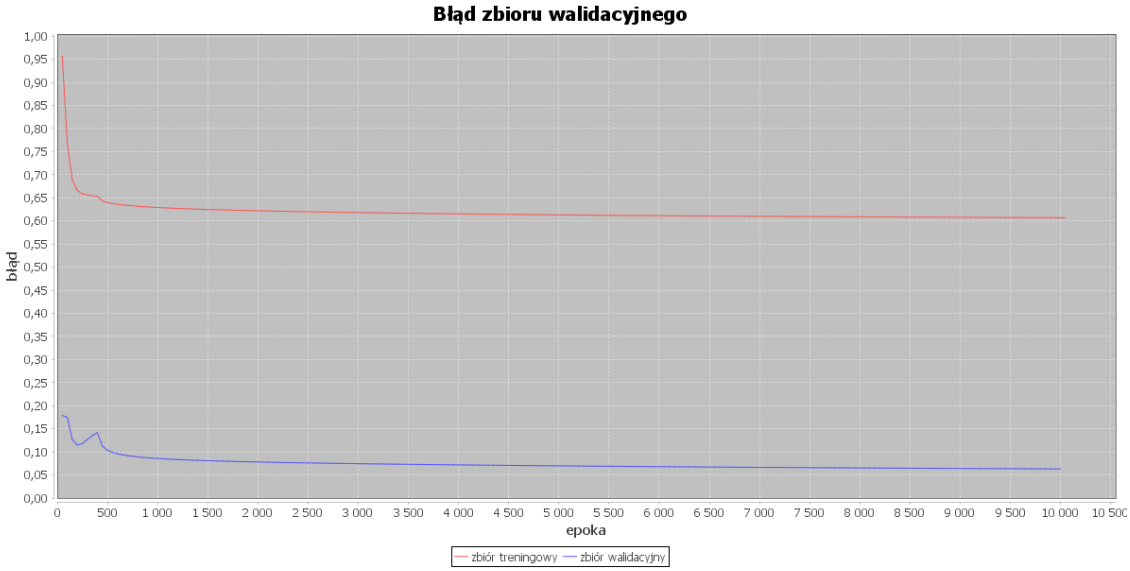
5.3. Badanie udziału cech w zbiorze badanych na zdolność uczenia się sieci

W kolejnej części doświadczeń wyłączano z procesu uczenia poszczególne cechy zbioru danych.

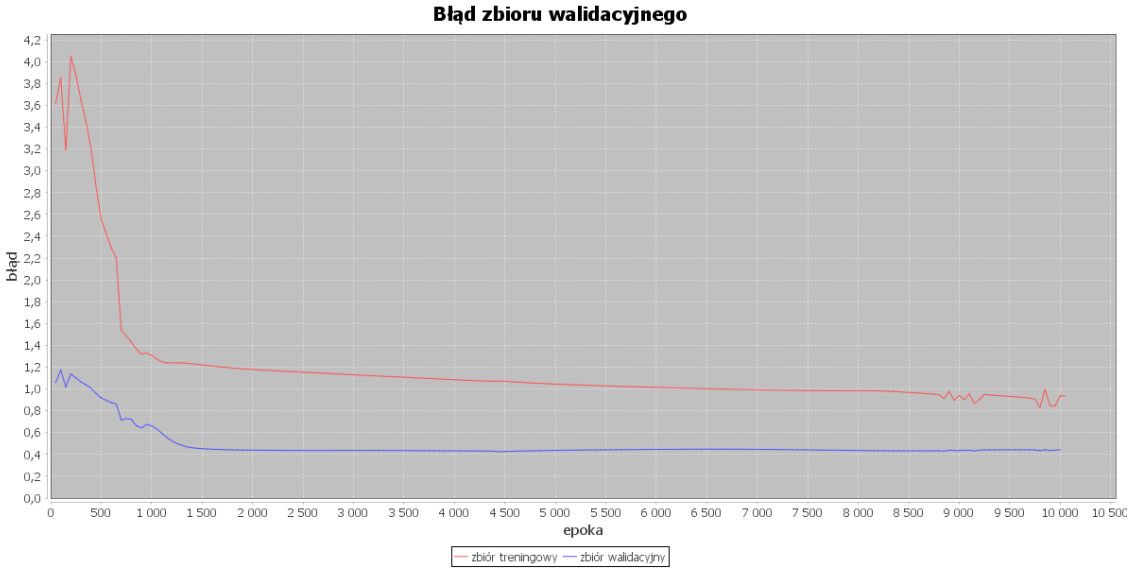
parametr	wartość
topologia warstw ukrytych	(8)
podział zbioru danych	66% trening, 18% walid., 18% test. (losowo)
wyłączone cechy	1



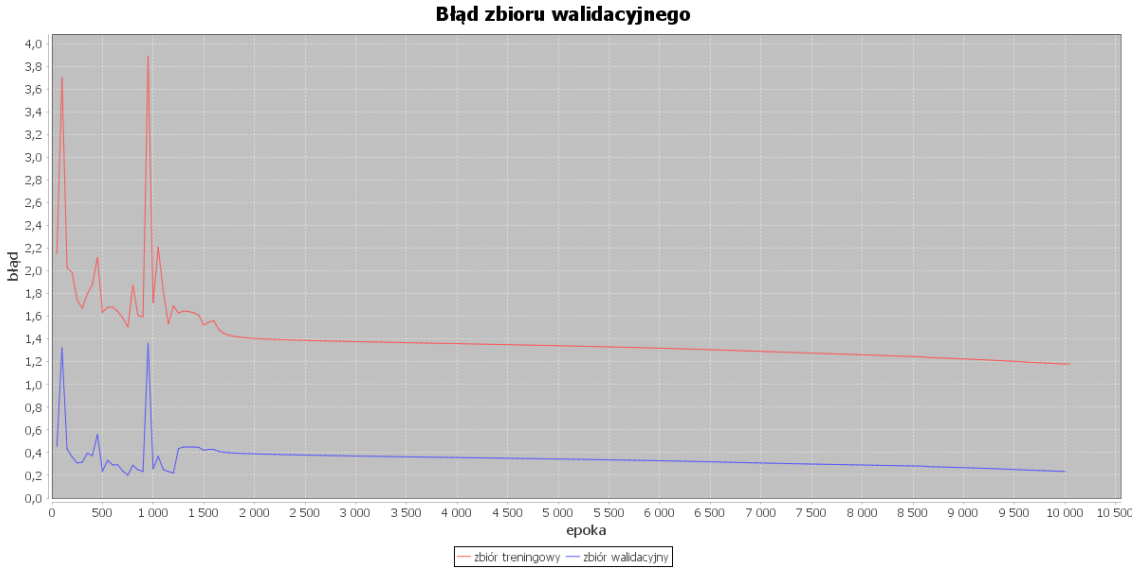
parametr	wartość
topologia warstw ukrytych	(8)
podział zbioru danych	66% trening, 18% walid., 18% test. (losowo)
wyłączone cechy	2



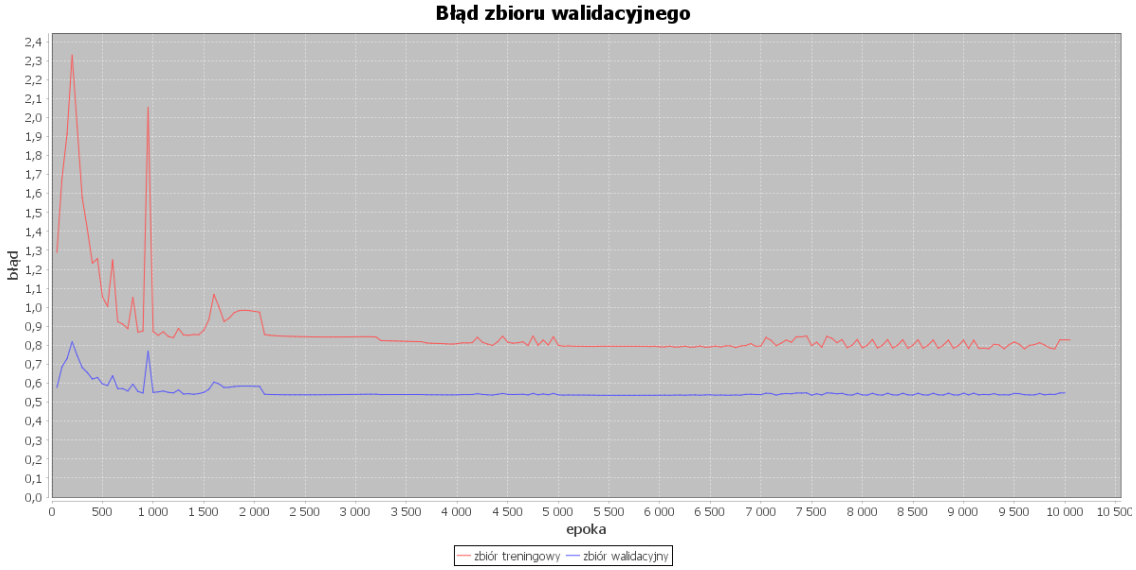
parametr	wartość
topologia warstw ukrytych	(8)
podział zbioru danych	66% trening, 18% walid., 18% test. (losowo)
wyłączone cechy	3



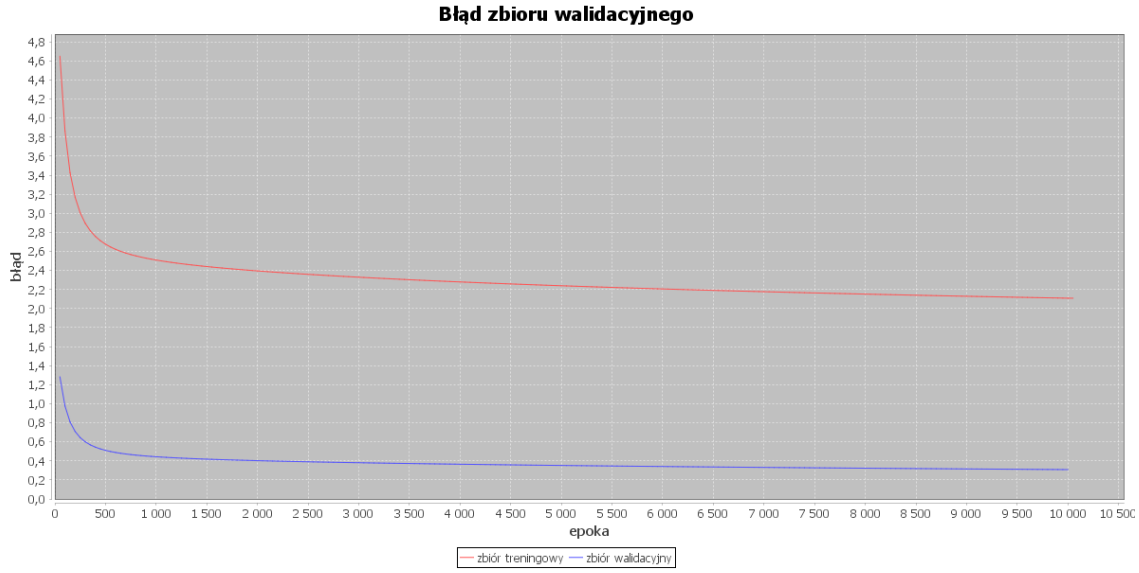
parametr	wartość
topologia warstw ukrytych	(8)
podział zbioru danych	66% trening, 18% walid., 18% test. (losowo)
wyłączone cechy	4

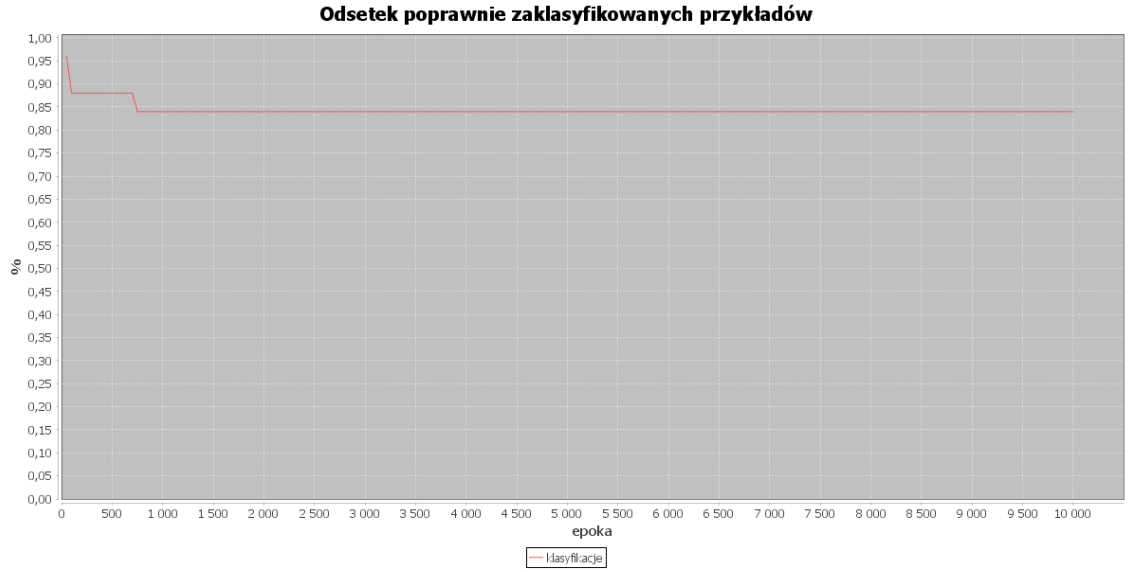


parametr	wartość
topologia warstw ukrytych	(8)
podział zbioru danych	66% trening, 18% walid., 18% test. (losowo)
wyłączone cechy	2,4

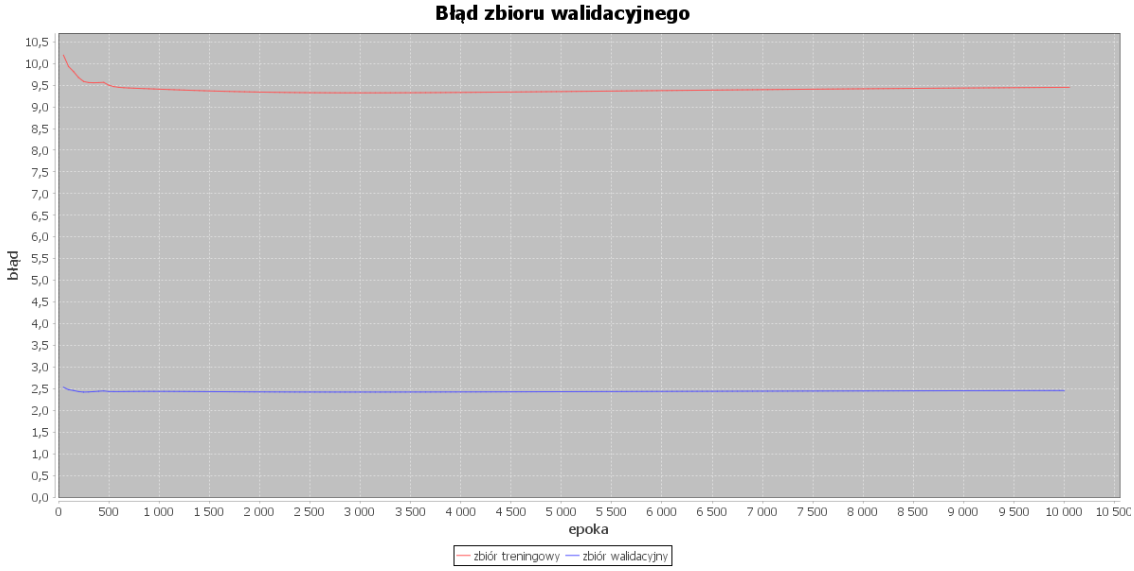


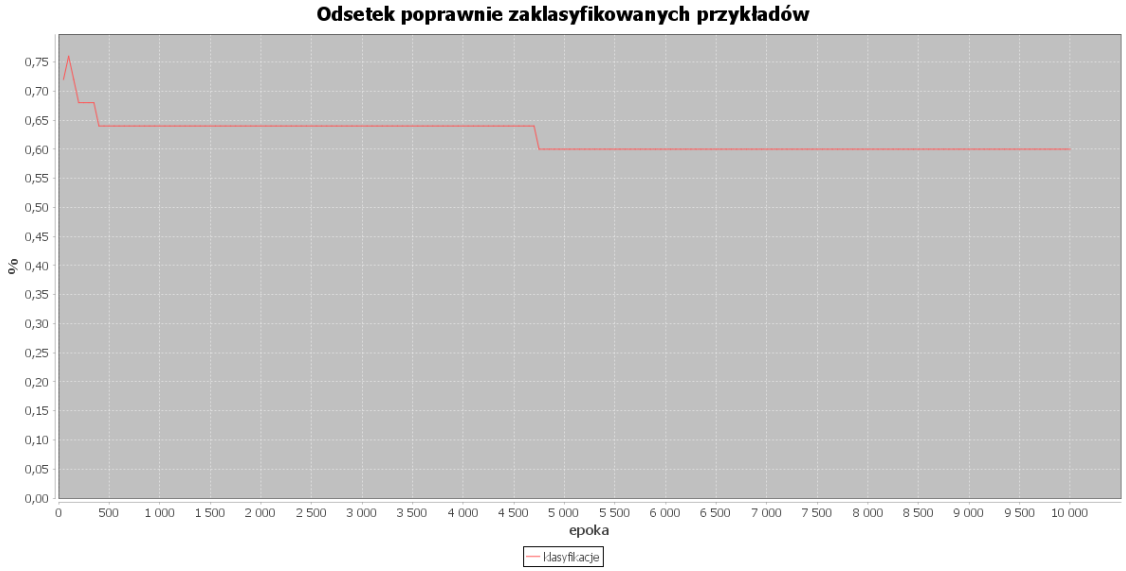
parametr	wartość
topologia warstw ukrytych	(8)
podział zbioru danych	66% trening, 18% walid., 18% test. (losowo)
wyłączone cechy	1,2,3





parametr	wartość
topologia warstw ukrytych	(8)
podział zbioru danych	66% trening, 18% walid., 18% test. (losowo)
wyłączone cechy	2,3,4





Wnioski

Najmniejszy wpływ na zdolność do klasyfikacji sieci ma cechy pierwsza i trzecia, natomiast największy cechy druga i czwarta. Ma to związek z rozkładem tych cech w zbiorze danych. Znormalizowane różnice średnich pomiędzy klasami mają największe wartości właśnie dla cech drugiej i czwartej. Cechy te zatem najlepiej charakteryzują klasy. Mimo wszystko wyłączanie z treningu poszczególnych cech nie obniża znacząco zdolności sieci do klasyfikowania przykładów. Dopiero wyłączenie trzech cech: drugiej, trzeciej i czwartej sprawia, że sieć klasyfikuje około 2/3 przykładów. Po dokładniejszej analizie ustalono, że to klasy druga i trzecie przestają być odróżniane.

6. Wnioski końcowe

- proces uczenia w znacznym stopniu zależy od początkowego stanu sieci
- zwiększanie liczby neuronów (i warstw) ukrytych tylko do pewnego momentu pozytywnie wpływa na zdolności uczenia się sieci
- dane irisdata wydają się być łatwe do klasyfikacji, gdyż wyłączenie z procesu uczenia nawet trzech wybranych cech w ograniczonym stopniu wpływa na zdolność rozpoznawania wzorców przez sieć