Rafał Lebioda 195639

Piotr Kaźmierczak 195611

Informatyka, studia dzienne, semestr VI, rok akademicki 2016/2017,

Laboratoria: czwartek, 8:15

Inteligentna Analiza Danych

Zadanie 1 – Sieci MLP

Spis treści

[Cel zadania 1](#_Toc479190540)

[Wstęp 1](#_Toc479190541)

[Obliczanie wyjścia neuronu 1](#_Toc479190542)

[Wsteczna propagacja błędu 2](#_Toc479190543)

[Opis implementacji 2](#_Toc479190544)

[Zadanie na 3 4](#_Toc479190545)

[Wyniki 4](#_Toc479190546)

# Cel zadania

Zadanie polegało na utworzeniu oraz wykorzystaniu wielowarstwowej sieci dla trzech różnych przypadków. Dla każdej utworzonej sieci należało przeprowadzić proces nauki w celu uzyskania sieci, która będzie w stanie rozwiązać dany problem

# Wstęp

Głównym elementem sieci neuronowych są neurony przetwarzające. Każdy neuron posiada dowolną ilość wejść oraz jedno wyjście. Dodatkowo neuron może być wyposażony w tak zwany bias, czyli dodatkowe wejście na którym występuje stała wartość. Waga dla tego wejścia jest modyfikowana jak wszystkie pozostałe wagi. Neurony są pogrupowane w warstwy, które można podzielić na warstwy: danych, ukryte oraz wyjściowe. Każde dwie warstwy są ze sobą połączone to oznacza, że na wejściach neuronów warstwy znajdują się wyjścia z neuronów warstwy poprzedniej.

# Obliczanie wyjścia neuronu

Do wejść doprowadzane są sygnały dochodzące z neuronów warstwy poprzedniej. Każdy sygnał mnożony jest przez odpowiadającą mu wartość liczbową zwaną wagą. Waga może być pobudzająca - dodatnia lub opóźniająca – ujemna.  Zsumowane iloczyny sygnałów i wag stanowią argument funkcjiaktywacji neuronu. Wartość funkcji aktywacji jest sygnałem wyjściowym neuronu i propagowana jest do neuronów warstwy następnej. Do naszego programu wykorzystaliśmy dwa rodzaje funkcji aktywacji :

* sigmoidalną funkcję aktywacji
* identycznościową funkcją aktywacji

W wypadku neuronów znajdujących się w warstwie danych ich wyjścia są wejściem całej sieci. W przypadku gdy neurony należą do warstwy wyjściowej, ich wyjścia są wyjściem całej sieci. Wyjście neuronu dane jest wzorem :

# Wsteczna propagacja błędu

Jest to metoda umożliwiająca modyfikację wag w sieci o architekturze warstwowej, we wszystkich jej warstwach. W wypadku wstecznej propagacji błędów proces nauki polega na policzeniu błędu każdego z neuronów rozpoczynając od neuronów wyjściowych. Dla neuronów w warstwie wyjściowej policzenie błędu jest dość proste i sprowadza się do wzoru :

gdzie:

s - suma wartości wejściowych pomnożonych przez wagi

y – wartość otrzymana

d – wartość oczekiwana

Natomiast obliczenie błędu dla neuronów w warstwie ukrytej polega na pobraniu z wszystkich neuronów z warstwy następnej błędów pomnożonych przez odpowiednią wagę, następnie należy zsumować wszystkie te wartości i pomnożyć przez pochodną funkcji aktywacji. Można to zapisać w postaci wzoru :

Z racji tego, że wagi są wykorzystywane w procesie obliczania błędu, należy obliczyć błędy dla wszystkich neuronów w sieci, a dopiero po tym procesie następuje zmiana wag jednocześnie we wszystkich warstwach. Modyfikowanie wag wyraża się wzorem:

gdzie:

µ - współczynnik nauki

- współczynnik momentum

- wartość z poprzedniej iteracji procesu nauki

# Opis implementacji

Implementacja programu została wykonana w języku C#. Program został napisany w taki sposób by można było utworzyć sieć o dowolnej ilości warstw ukrytych z dowolną ilością neuronów w warstwie. Zdecydowaliśmy się na implementacje typowo obiektową co zdecydowanie w efekcie końcowym zaważyło na wydajności programu w porównaniu gdybyśmy przeprowadzili implementacje przy pomocy macierzy. Główna klasą jest klasa Siec która posiada listę warstw oraz informację o poprzedniej oraz następnej warstwie. Każda warstwa posiada swoją listę neuronów. Interfejs użytkownika pozwala na ustalenie współczynników nauki oraz momentum, wykorzystanie biasu, ustawienie błędu jako warunek stopu nauki oraz ustawienie dowolnej ilości warstw i neuronów. Ponadto została dodana możliwość serializacji sieci do pliku .xml oraz wczytywanie sieci z takiego pliku co pozwala na wyuczenie sieci i zapisaniu jej. Umożliwia to w przyszłości wczytanie takiej sieci i kontynuowanie nauki lub wykorzystanie takiej wyuczonej już sieci do rozwiązania danego problemu. Program umożliwia przeprowadzanie nauki na danych pochodzących z różnych plików tekstowych oraz istnieje możliwość wczytania danych testowych w celu zweryfikowania czy sieć jest poprawnie nauczona.

# Część 1 - Transformacja

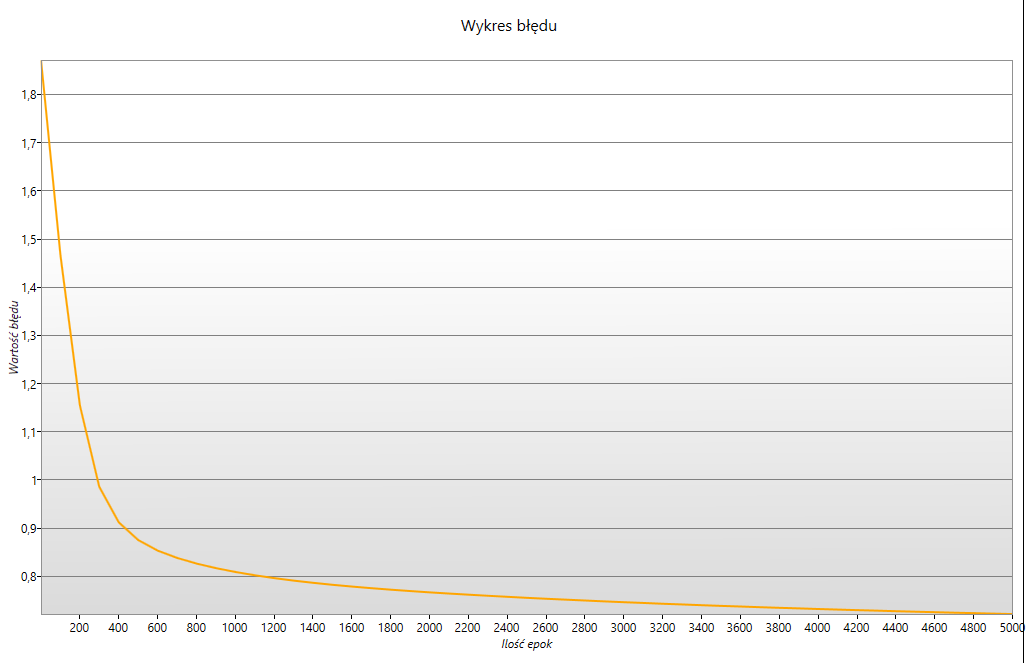
Zadanie polega na stworzeniu sieci neuronowej o 4 wejściach i 4 wyjściach oraz jednej warstwie ukrytej. Należało nauczyć sieć odtwarzać wprowadzone do niej dane z jak największą precyzją.

Sieć była uczona za pomocą następujących danych:

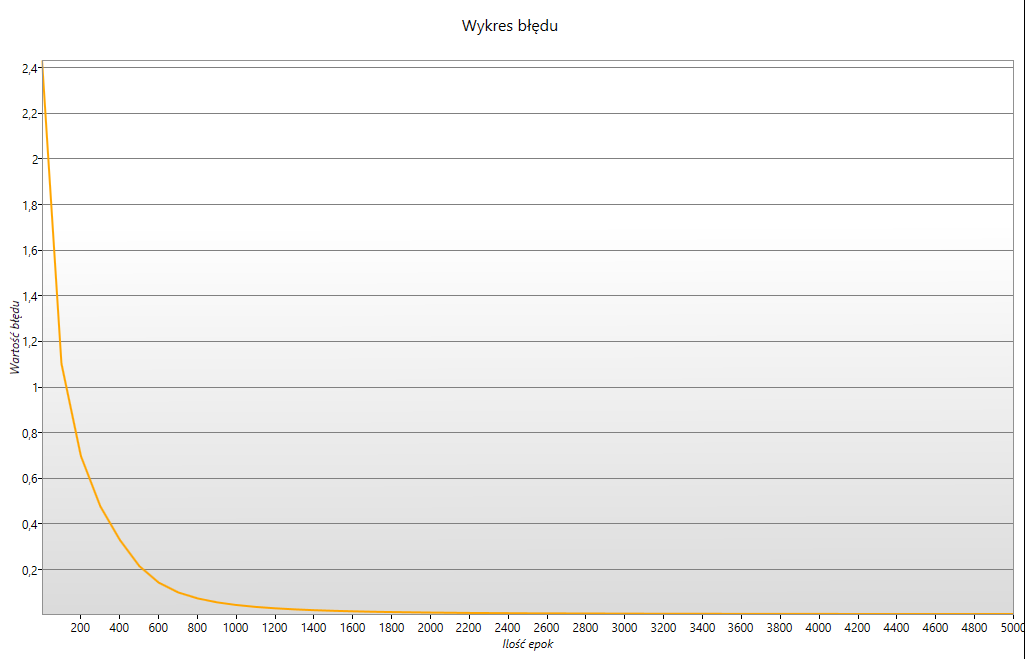
|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 |

## Wyniki

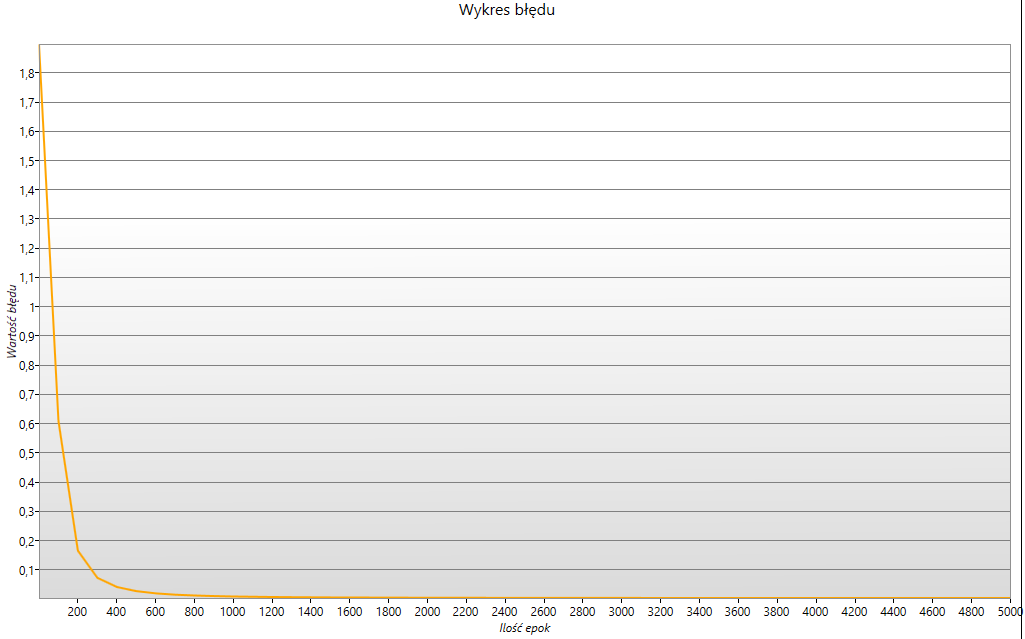
|  |  |
| --- | --- |
| Ilość epok | 5000 |
| Epsilon | 0,0001 |
| Momentum | 0,5 |
| Współczynnik nauki | 0,2 |
| Bias | tak |
| Ilość warstw ukrytych | 1 |
| Ilość neuronów | 1 |
| Ilość wejść | 4 |
| Ilość wyjść | 4 |



|  |  |
| --- | --- |
| Ilość epok | 5000 |
| Epsilon | 0,0001 |
| Momentum | 0,5 |
| Współczynnik nauki | 0,2 |
| Bias | nie |
| Ilość warstw ukrytych | 1 |
| Ilość neuronów | 2 |
| Ilość wejść | 4 |
| Ilość wyjść | 4 |



|  |  |
| --- | --- |
| Ilość epok | 5000 |
| Epsilon | 0,0001 |
| Momentum | 0,5 |
| Współczynnik nauki | 0,2 |
| Bias | nie |
| Ilość warstw ukrytych | 1 |
| Ilość neuronów | 3 |
| Ilość wejść | 4 |
| Ilość wyjść | 4 |

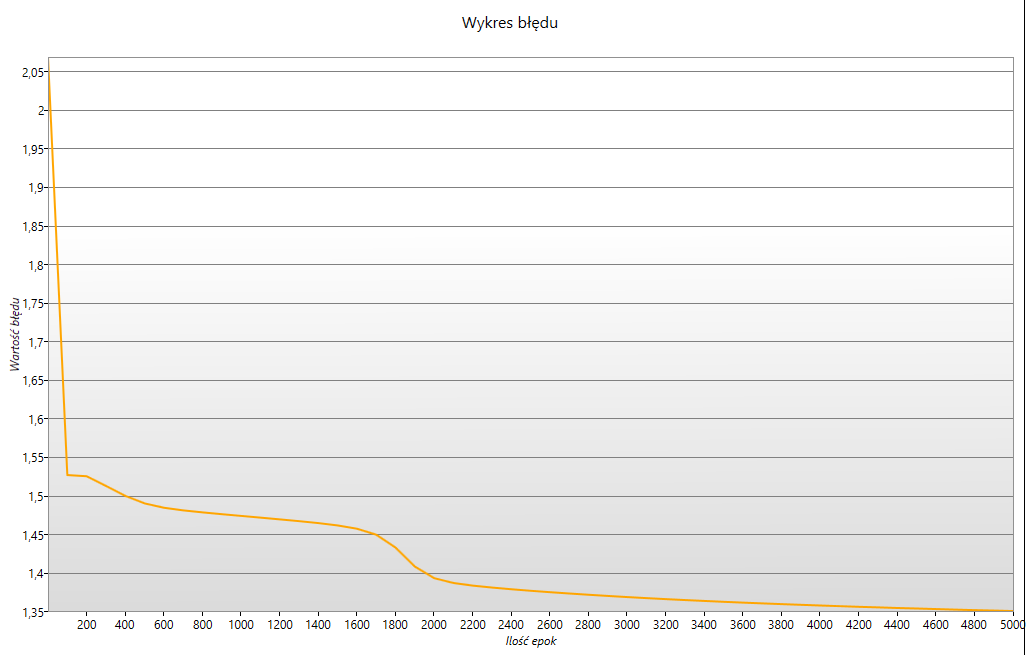


|  |  |
| --- | --- |
| Epoka 1 | Epoka 5000 |
| Błąd sieci równy : 2,90003810236001 | Błąd sieci równy : 0,000240571671351115 |

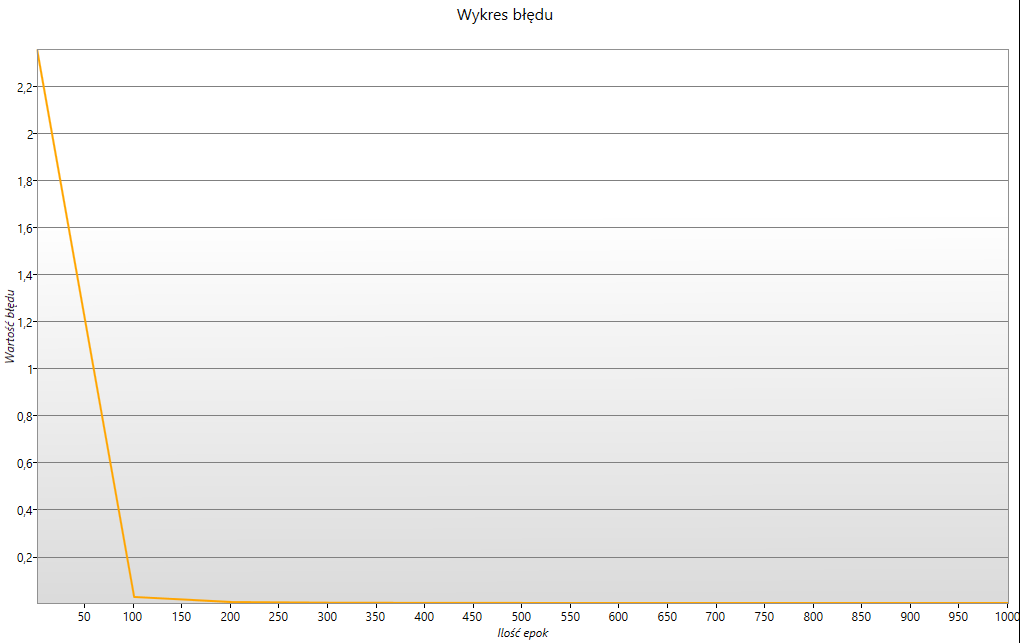
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Wartości spodziewane | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Wartości otrzymane | 0,995433475827978 | 9,29863618031561E-09 | 0,00389505703024347 | 0,00425599932475016 |
| 4,02883776931911E-06 | 0,99013225332963 | 0,0066782861048727 | 0,00582123597149931 |
| 0,00187487894868099 | 0,00396444198630648 | 0,989749016424863 | 2,44160544090309E-07 |
| 0,00240370992759943 | 0,00364912689515352 | 3,23103939406952E-07 | 0,989620313984464 |

|  |  |
| --- | --- |
| Ilość epok | 5000 |
| Epsilon | 0,0001 |
| Momentum | 0,5 |
| Współczynnik nauki | 0,2 |
| Bias | nie |
| Ilość warstw ukrytych | 1 |
| Ilość neuronów | 1 |
| Ilość wejść | 4 |
| Ilość wyjść | 4 |

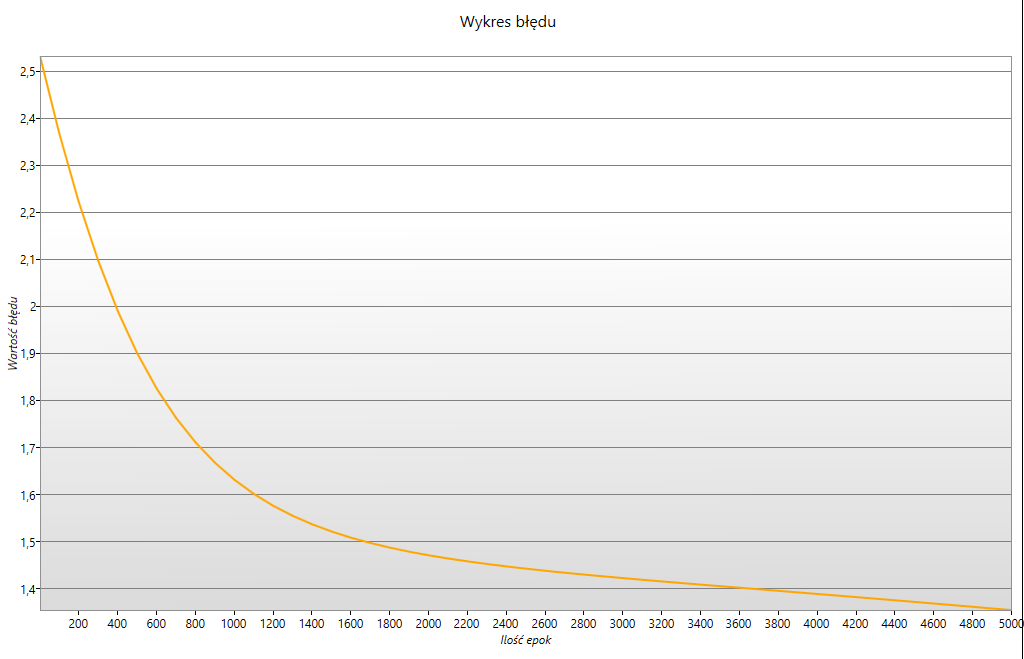


|  |  |
| --- | --- |
| Ilość epok | 5000 |
| Epsilon | 0,0001 |
| Momentum | 0,5 |
| Współczynnik nauki | 0,9 |
| Bias | tak |
| Ilość warstw ukrytych | 1 |
| Ilość neuronów | 3 |
| Ilość wejść | 4 |
| Ilość wyjść | 4 |



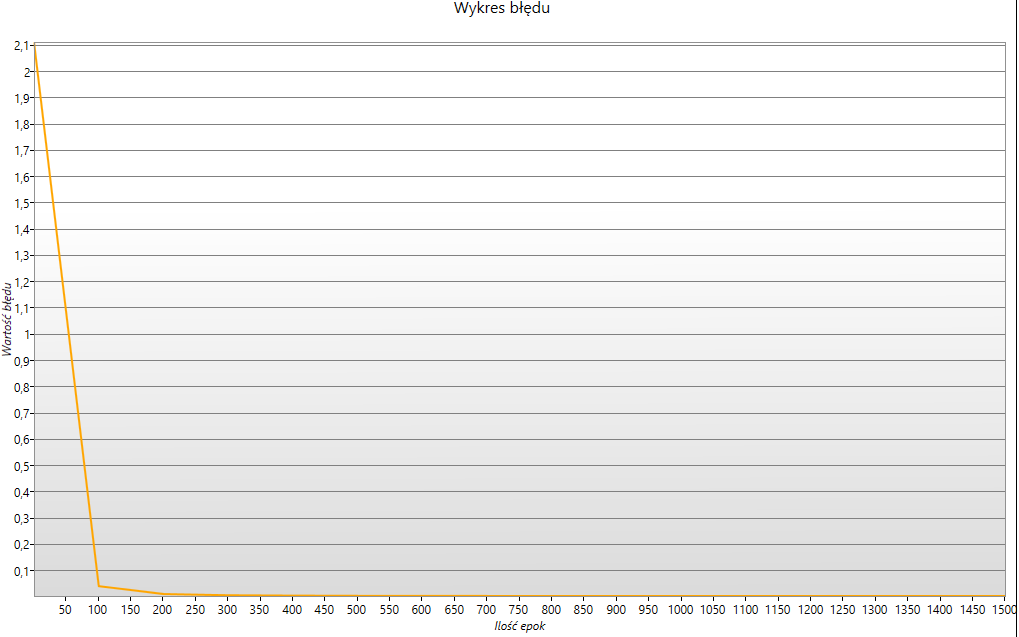
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Wartości otrzymane | 0,994256745065439 | 0,000934397566381756 | 0,000734944734527612 | 0,00260050082812886 |
| 0,00182015700082128 | 0,995099570200147 | 0,00106402957003907 | 0,000679002204036407 |
| 0,00565204299262853 | 0,00326929225079499 | 0,992080553163123 | 0,00279760133802165 |
| 5,99915543750312E-05 | 0,00170013729582777 | 0,00219872221777709 | 0,995235816416726 |

|  |  |
| --- | --- |
| Ilość epok | 5000 |
| Epsilon | 0,0001 |
| Momentum | 0,5 |
| Współczynnik nauki | 0,001 |
| Bias | tak |
| Ilość warstw ukrytych | 1 |
| Ilość neuronów | 3 |
| Ilość wejść | 4 |
| Ilość wyjść | 4 |



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Wartości otrzymane | 0,409696652236717 | 0,267398475608726 | 0,270669428267387 | 0,217194192771765 |
| 0,197320642505933 | 0,257431323243947 | 0,240772930133532 | 0,225349831307216 |
| 0,233919095705339 | 0,274658100019232 | 0,257932953243707 | 0,257976855309125 |
| 0,188793502920797 | 0,276432856392328 | 0,246162044408903 | 0,261462420713474 |

|  |  |
| --- | --- |
| Ilość epok | 5000 |
| Epsilon | 0,0001 |
| Momentum | 0,9 |
| Współczynnik nauki | 0,2 |
| Bias | tak |
| Ilość warstw ukrytych | 1 |
| Ilość neuronów | 3 |
| Ilość wejść | 4 |
| Ilość wyjść | 4 |



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Wartości otrzymane | 0,994775930185323 | 6,8123670717995E-07 | 0,00213113414516133 | 0,00275222837979915 |
| 2,37628458950642E-06 | 0,994819918308052 | 0,0017165751068957 | 0,00296382025659126 |
| 0,00145178539460571 | 0,00144298728443369 | 0,996938882102563 | 2,84306243237532E-09 |
| 0,00401126232008363 | 0,00402260772072745 | 8,45024765476142E-05 | 0,991058663301379 |

## Wnioski

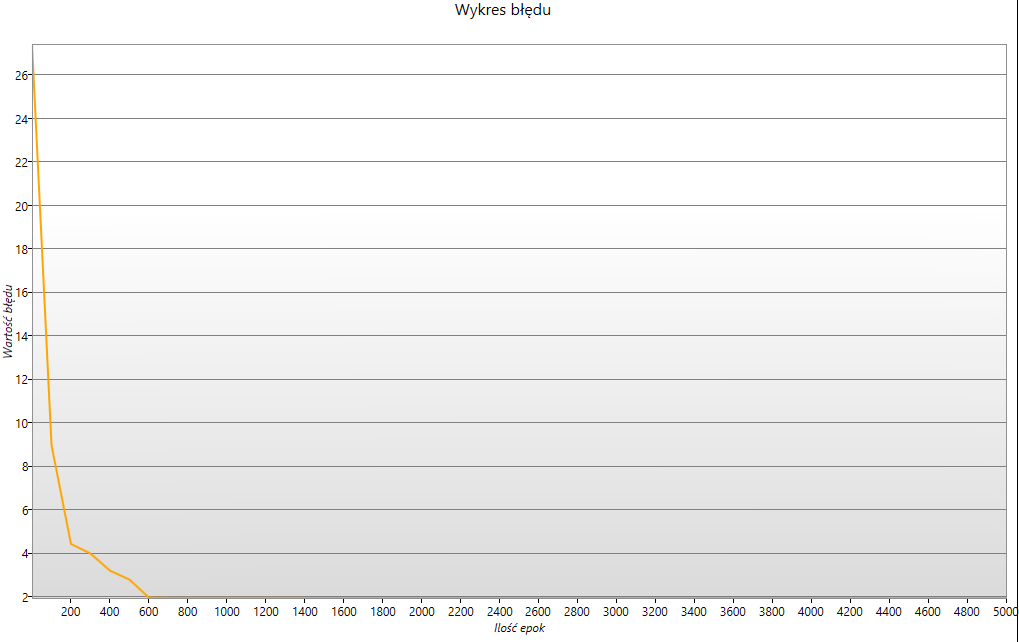
Po przeprowadzonych eksperymentach można stwierdzić, że większa ilość neuronów przyspiesza naukę sieci. Ich ilość powinna zależeć od stopnia skomplikowania problemu. Brak obecności biasu zmniejsza tempo zmniejszania błędu w przypadku, gdy sieć posiada jedynie jeden neuron w warstwie ukrytej. Jego obecność jest więc konieczna, by sieć poprawnie się uczyła. Wyższe wartości parametrów nauki zwiększały szybkość uczenia sieci, jednak zbyt wysoka wartość współczynnika nauki zmniejsza dokładność wartości otrzymywanych. Problem ten nie wystąpił w przypadku wysokiej wartości momentum, sieć nauczyła się odtwarzać wartości szybciej z wysoką dokładnością.

# Część 5 – Klasyfikacja

Zadanie polegało na nauczeniu sieci poprawnej klasyfikacji obiektów. Za wyjście sieci należy uznać odpowiednio zakodowany rodzaj obiektu: 1 – (0,0,0), 2 – (0,1,0), 3 – (0,0,1).

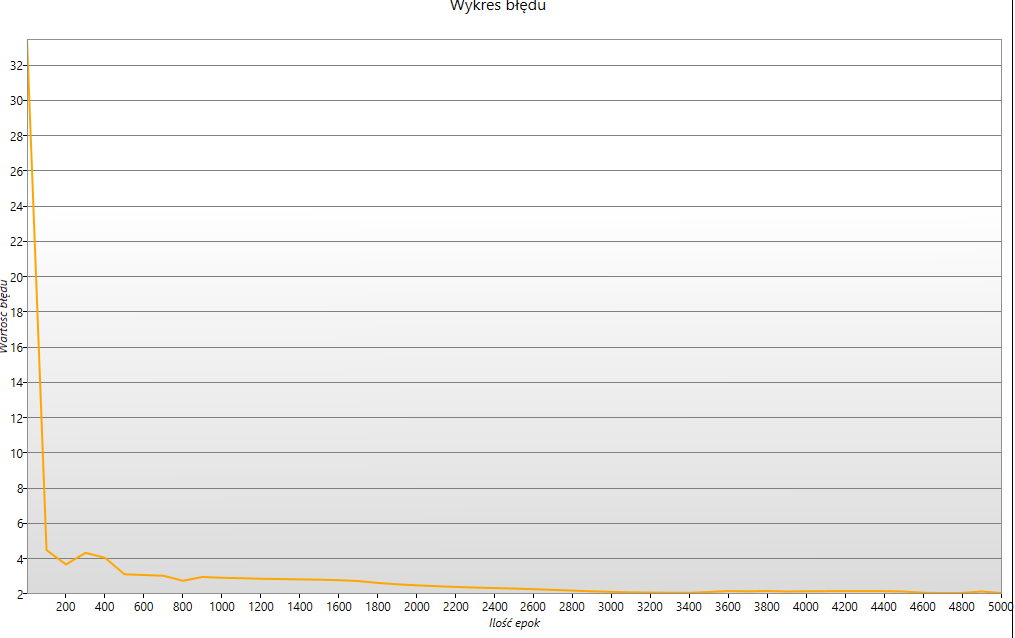
## Wyniki

|  |  |
| --- | --- |
| Ilość epok | 5000 |
| Epsilon | 0,0001 |
| Momentum | 0,5 |
| Współczynnik nauki | 0,2 |
| Bias | tak |
| Ilość warstw ukrytych | 1 |
| Ilość neuronów | 3 |
| Ilość wejść | 4 |
| Ilość wyjść | 3 |

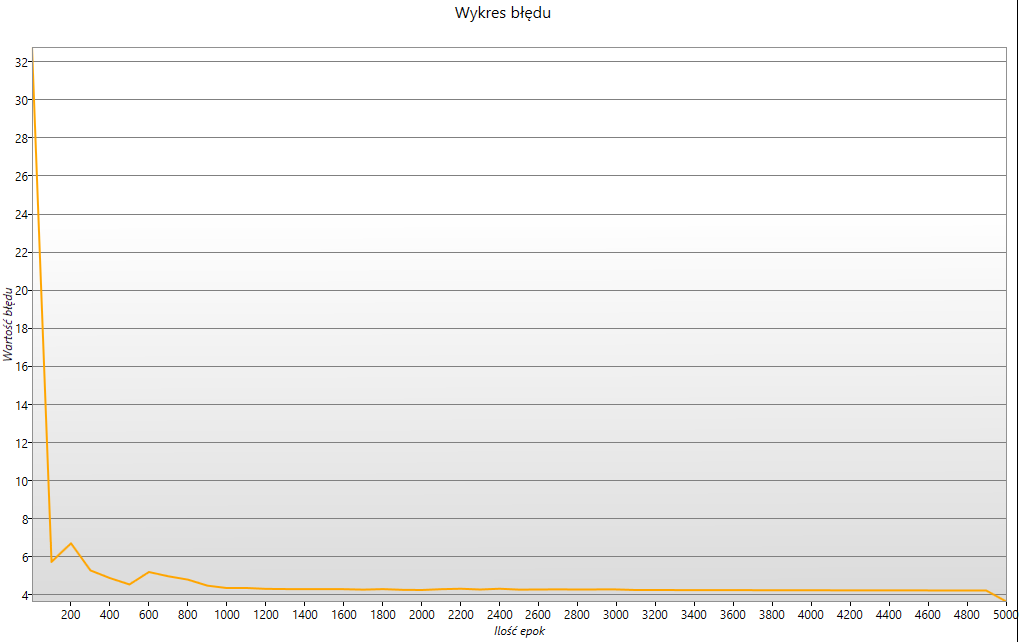


|  |  |  |  |
| --- | --- | --- | --- |
| Wartości spodziewane | 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |
| Wartości otrzymane | 0,999864378588616 | 3,97604793486828E-05 | ,03395074866108E-08 |
| 0,000118134747834204 | 0,999781341137589 | 4,61720827810573E-05 |
| 8,41565489727261E-06 | 0,112765563417179 | 0,946592662786655 |

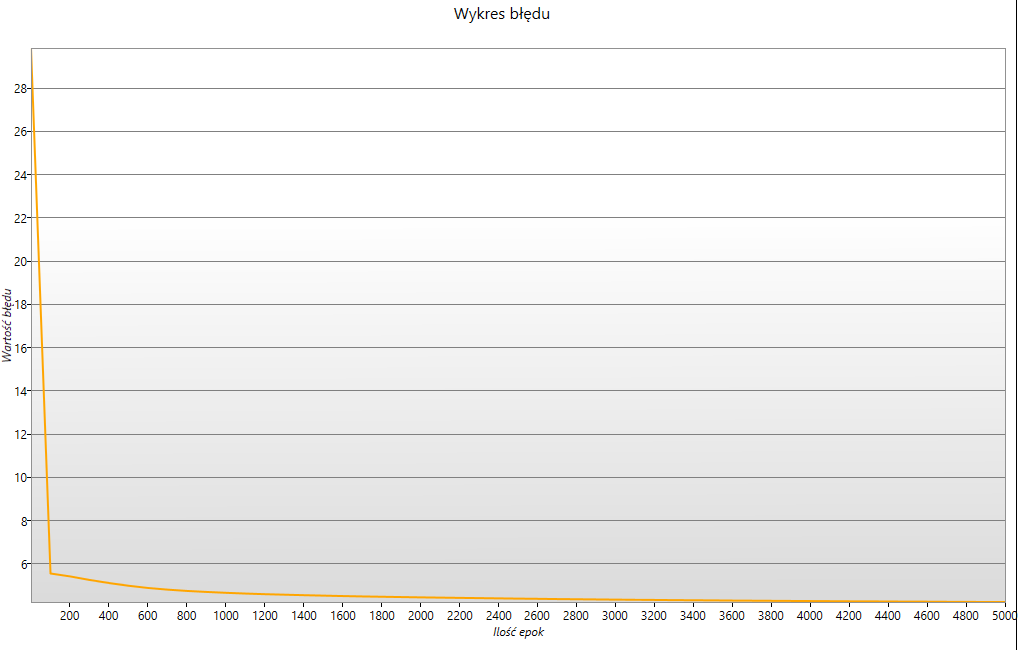
|  |  |
| --- | --- |
| Ilość epok | 5000 |
| Epsilon | 0,0001 |
| Momentum | 0,5 |
| Współczynnik nauki | 0,2 |
| Bias | tak |
| Ilość warstw ukrytych | 1 |
| Ilość neuronów | 3 |
| Ilość wejść | 3 |
| Ilość wyjść | 3 |



|  |  |
| --- | --- |
| Ilość epok | 5000 |
| Epsilon | 0,0001 |
| Momentum | 0,5 |
| Współczynnik nauki | 0,2 |
| Bias | tak |
| Ilość warstw ukrytych | 1 |
| Ilość neuronów | 3 |
| Ilość wejść | 2 |
| Ilość wyjść | 3 |



|  |  |
| --- | --- |
| Ilość epok | 5000 |
| Epsilon | 0,0001 |
| Momentum | 0,5 |
| Współczynnik nauki | 0,2 |
| Bias | tak |
| Ilość warstw ukrytych | 1 |
| Ilość neuronów | 3 |
| Ilość wejść | 1 |
| Ilość wyjść | 3 |



## Wnioski

Wraz z malejącą ilością wejść (wprowadzonych cech obiektów) błąd średniokwadratowy jest większy. Najważniejszym czynnikiem w przypadku naszej sieci był współczynnik nauki, który powinien mieć niską wartość. Zbyt wysoki współczynnik nauki uniemożliwia spadek błędu. Gdy przekroczy już ten próg błąd dla sieci nie ulega znacznej zmianie i można zakończyć naukę takiej sieci. Ilość neuronów wpływa na poziom nauczenia się sieci, jednakże powyżej 10 neuronów różnice już nie są tak bardzo zauważalne.