Dawid Urbaniak 204023 204023@edu.p.lodz.pl
Igor Oryński 203959 203959@edu.p.lodz.pl

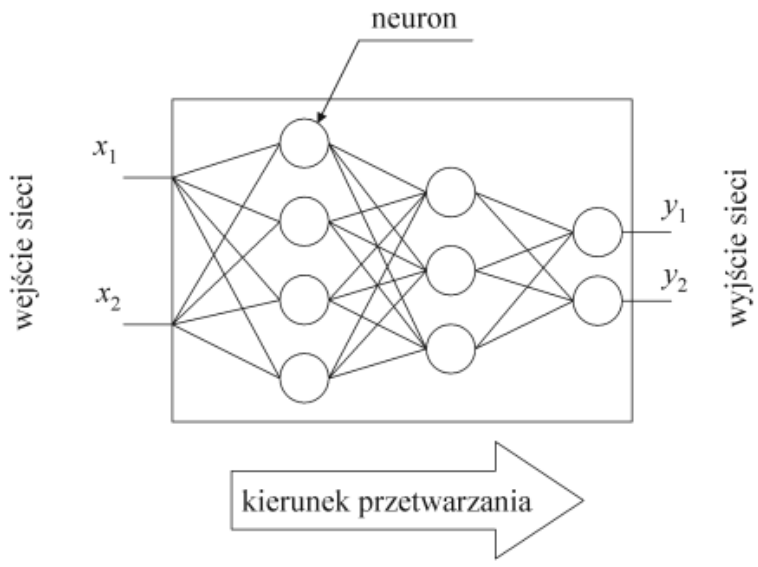
Zadanie 2.: Perceptron wielowarstwowy

1. Cel

Celem zadania jest stworzenie aplikacji implementującej perceptron wielowarstwowy oraz wsteczną propagację błędów jako metodę jego nauki.

2. Wprowadzenie

Perceptron wielowarstwowy jest jednym z najprostszych typów sieci neuronowych składający się z co najmniej dwóch neuronów ułożonych warstwowo, implementujący algorytm uczenia nadzorowanego. Neurony są zgrupowane w warstwy w taki sposób, że nie istnieją żadne połączenia między neuronami danej warstwy. Wyróżnia się 3 rodzaje warstw: wejściowa, wyjściowa oraz warstwy ukryte.



Rysunek 1. Graf przedstawiający budowę perceptronu wielowarstwowego[1]

3. Opis implementacji

Aplikacja została zaimplementowana w języku **Python3** korzystając z bibliotek **numpy** oraz **gnuplot** do tworzenia wykresów.

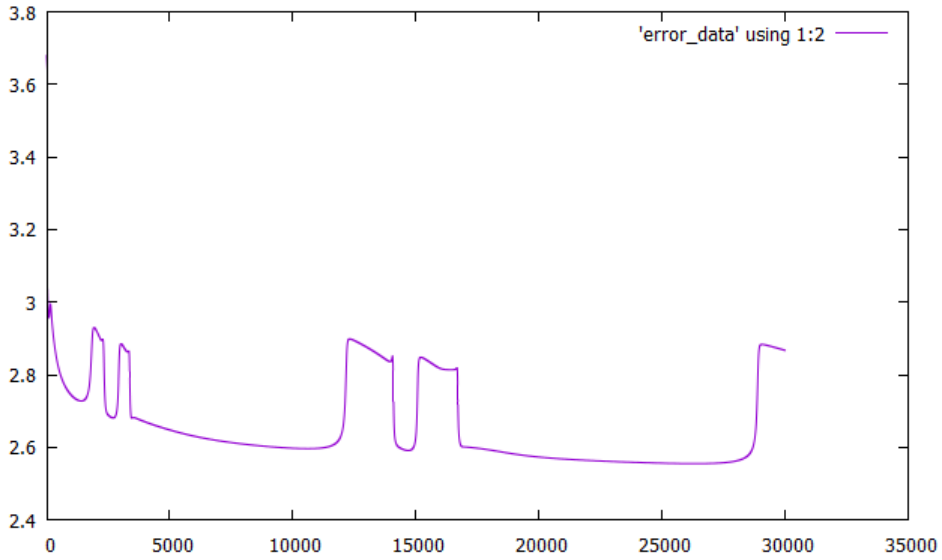
Wszystkie stworzone metody służące do obliczeń znajdują się w przestrzeni klasy **BackPropagationNetwork**, w tym:
— **sigmoid** - metoda obliczająca sigmoidalną funkcję aktywacji,
— **trainEpoch** - metoda trenująca sieć.

4. Materiały i metody

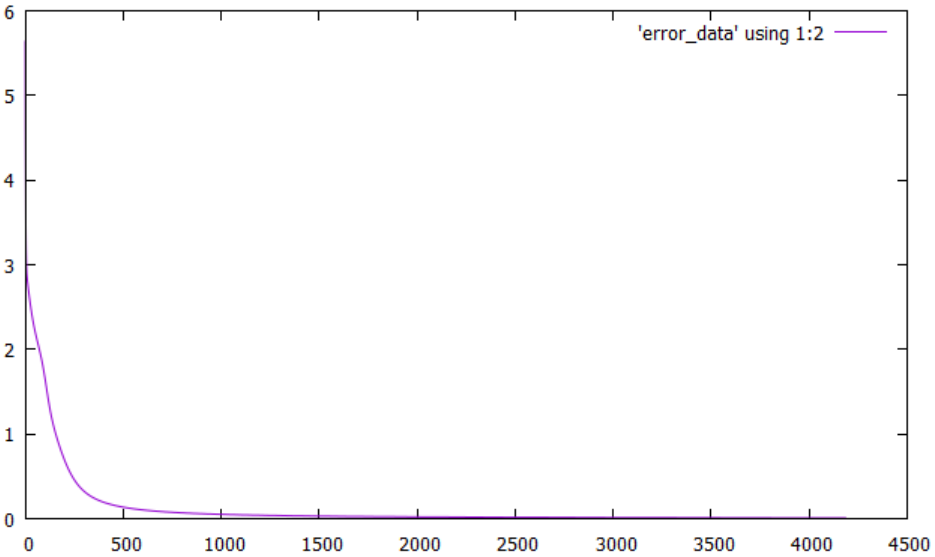
Na początku badane są dwa przypadki nauki przy losowej kolejności podawania wzorców w każdej epoce oraz przy współczynniku nauki równym 0.6 (współczynnik momentum nie jest uwzględniany): z biasem oraz bez.

Następnie badana jest szybkość uczenia perceptronu w zależności od wartości współczynników nauki oraz momentum w następujących przypadkach:
— współczynnik nauki - 0,9, współczynnik momentum - 0,0,
— współczynnik nauki - 0,6, współczynnik momentum - 0,0,
— współczynnik nauki - 0,2, współczynnik momentum - 0,0,
— współczynnik nauki - 0,9, współczynnik momentum - 0,6,
— współczynnik nauki - 0,2, współczynnik momentum - 0,9.

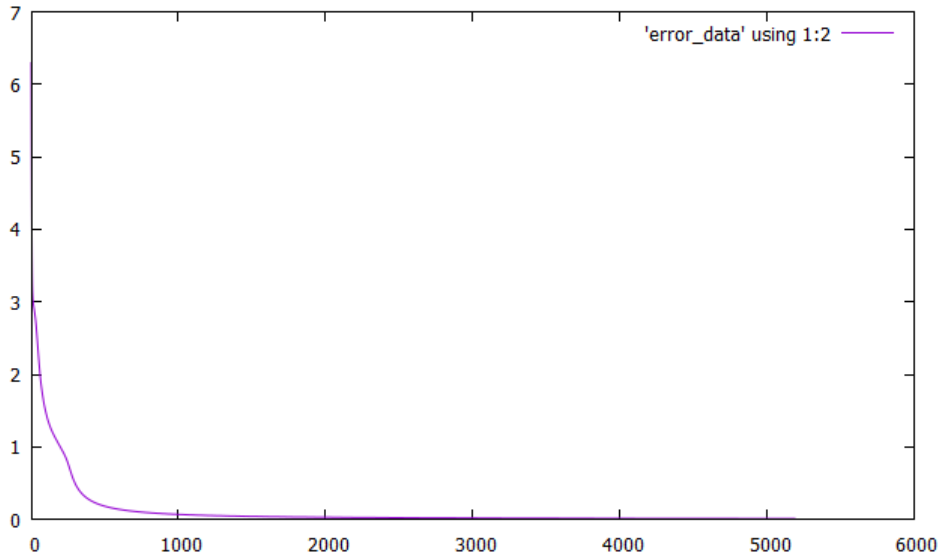
5. Wyniki



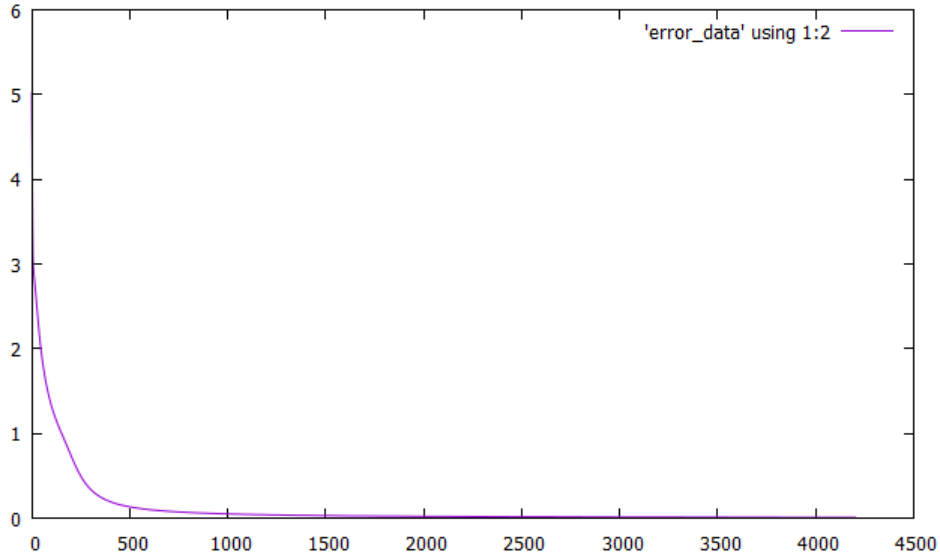
Rysunek 2. Wykres zmian funkcji kosztu: losowa kolejność wzorców, bez biasu, wsp. nauki - 0.6



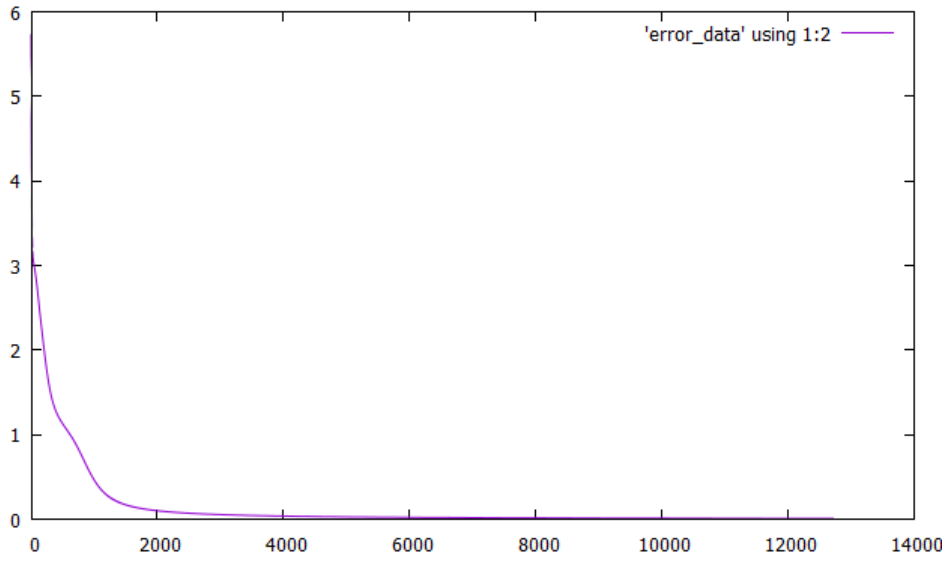
Rysunek 3. Wykres zmian funkcji kosztu: losowa kolejność wzorców, z biasem, wsp. nauki - 0.6



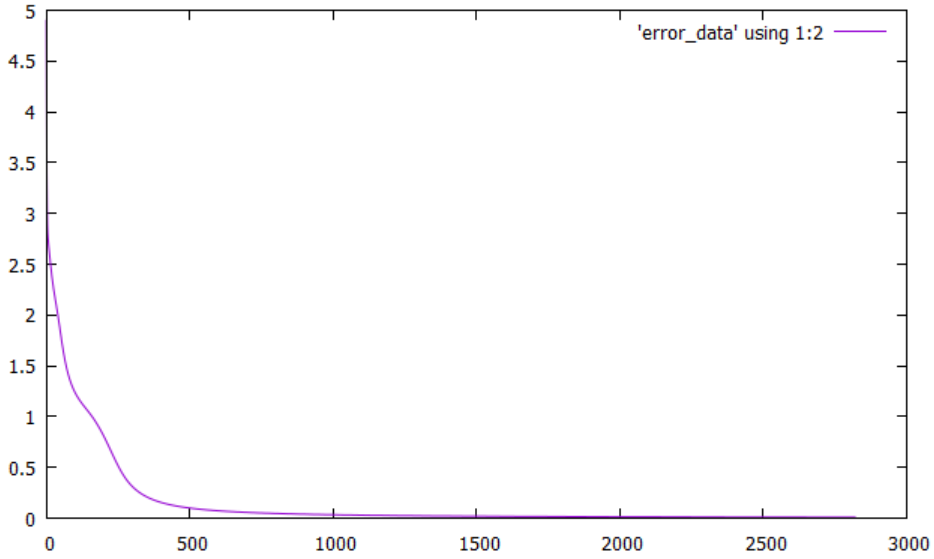
Rysunek 4. Wykres zmian funkcji kosztu: losowa kolejność wzorców, z biasem, wsp. nauki - 0.9, wsp. momentum - 0.0, liczba iteracji - 5188



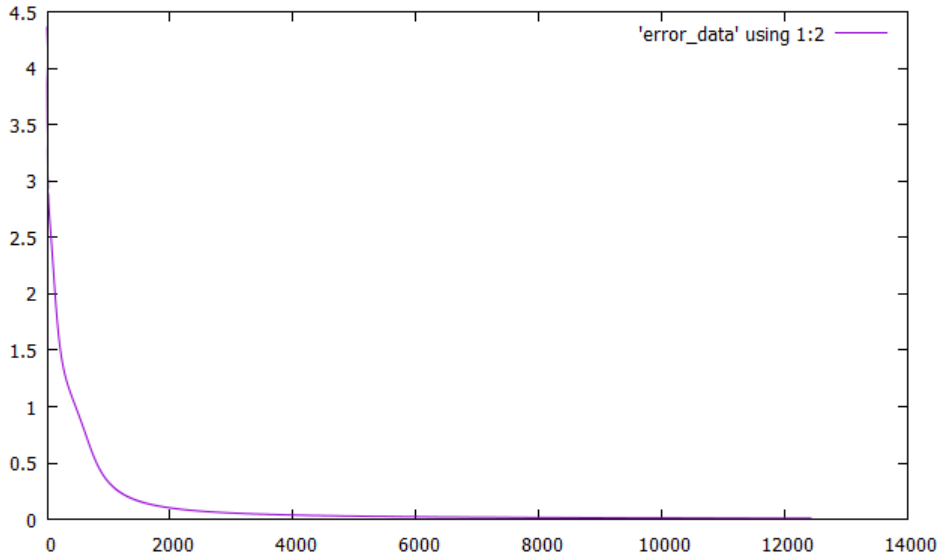
Rysunek 5. Wykres zmian funkcji kosztu: losowa kolejność wzorców, z biasem, wsp. nauki - 0.6, wsp. momentum - 0.0, liczba iteracji - 4202



Rysunek 6. Wykres zmian funkcji kosztu: losowa kolejność wzorców, z biasem, wsp. nauki - 0.2, wsp. momentum - 0.0, liczba iteracji - 12714



Rysunek 7. Wykres zmian funkcji kosztu: losowa kolejność wzorców, z biasem, wsp. nauki - 0.9, wsp. momentum - 0.6, liczba iteracji - 2819



Rysunek 8. Wykres zmian funkcji kosztu: losowa kolejność wzorców, z biasem, wsp. nauki - 0.2, wsp. momentum - 0.9, liczba iteracji - 12423

6. Dyskusja

Po wykonaniu pierwszych dwóch przypadków możemy zauważyć, że nauka z wykorzystaniem biasu jest o wiele bardzie efektywna. Wykonanie uczenia sieci bez użycia biasu doprowadziła do maksymalnej ustawionej ilości iteracji - 30000, podczas gdy algorytm z biasem zatrzymał się przy ok. 4200 iteracjach osiągając warunek stopu dopuszczalnego błędu.

Na podstawie kolejnych testów możemy wywnioskować, że przy pominięciu współczynnika momentum najlepsze efekty uzyskujemy przy umiarkowanym współczynniku nauki (w naszym przypadku wynosił on 0.6). Zauważmy, że wysoka wartość współczynnika nauki nie oznacza wcale, że nasza sieć będzie uzyskiwać lepsze wyniki. Nasze wnioski praktycznie nie zmienią się nawet gdy weźmiemy pod uwagę współczynnik momentum. Na podstawie rysunków 7 i 8 możemy zauważyć, że wysoki współczynnik nauki w połączeniu z umiarkowaną wartością momentum dał najlepszy rezultat z dotychczasowych - poniżej 3000 iteracji. Ustawienie odpowiednio niskiego i wysokiego współczynnika nauki i momentum dało odwrotny efekt - liczba iteracji ma drugą najwyższą wartość spośród badanych, co automatycznie oznacza bardzo niską efektywność takiego rozwiązania.

Należy jednakże pamiętać, że wniosek ten opisuje bardzo mały i prosty zestaw treningowy. Intuicja oraz literatura sugerują, że oparcie nauki na niewielkim współczynniku uczenia oraz dużym momentum pozwala uczynić naukę bardziej elastyczną, szczególnie w przypadku kiedy ważne jest dopasowanie się do dokładnie wszystkich testów z nieco liczniejszego zbioru treningowego.

7. Wnioski

— Wersja algorytmu z biasem jest bardziej efektywna niż bez niego.
— Bez współczynnika momentum najlepiej zachować umiarkowaną wartość współczynnika nauki.
— Z współczynnikiem momentum najlepsze wyniki uzyskamy utrzymując oba współczynniki na umiarkowanym poziomie.
— Zbytnie obniżenie lub podwyższenie wartości współczynników znacząco zmniejsza efektywność uczenia sieci.
— Wszystkie powyższe wnioski oparte są na bardzo małym i prostym zestawie treningowym, w przypadku struktur bardziej złożonych największą efektywności osiągnie oparcie nauki o niski współczynnik uczenia i wysoki momentum.

Literatura

[1] K. Diks [http://wazniak.mimuw.edu.pl/] *Sztuczna inteligencja/SI Moduł 12*
[2] P. Tarasiuk [https://ftims.edu.p.lodz.pl/] *Sztuczne sieci neuronowe z propagacją w przód: przydatne wzory*
[3] Wikipedia [https://pl.wikipedia.org/] *Perceptron wielowarstwowy*