Nikodem Kirsz 236559
Oskar Trela 236677

czwartek, 13:30
26.05.2022

Sztuczna inteligencja i systemy ekspertowe

Zadanie: Dopasowanie funkcji za pomocą sieci neuronowej

1. Wstęp

Celem zadania było zaimplementowanie oraz przetestowanie sieci neuronowej typu MLP (*Multi-layer Perceptron*). Sieć ma charakter uniwersalny gwarantujący poprawność działania i nauki perceptronu bez względu na liczbę warstw i neuronów. Neurony ukryte oraz wyjściowe wykorzystują sigmoidalną funkcję aktywacji (współczynnik nachylenia jest równy 1). Zgodnie z zaleceniami wagi sieci są inicjalizowane w sposób pseudolosowy z zakresu <-0.5, 0.5>.

Program działa w dwóch trybach: tryb nauki oraz tryb testowania.

Tryb nauki korzysta z metody on-line, która to stara się dostosować sieć do odpowiedniego zestawu danych po każdym jednym wzorcu zamiast na całym zestawie naraz tak jak w przypadku metody off-line. W ten sposób mamy możliwość przesyłania danych sekwencyjnie a model jest aktualizowany przy każdej zmianie danych wejściowych.

Nauka sieci polega na podaniu wartości na wejście sieci, a następnie sprawdzenie zgodności ze wzorcem wartości wyjściowych (*Forward Propagation*). W tym momencie obliczany jest błąd i rozpoczyna się praca algorytmu propagacji w tył (*Backpropagation Algorithm*). Błąd obliczony na wyjściu jest przekazywany do warstwy poprzedzającej dzięki czemu wagi oraz obciążenia są dostosowywane do poprawnego wyjścia. Algorytm jest propagowany w ten sposób aż do warstwy wejściowej.

Dodatkiem do naszej implementacji jest momentum, dzięki któremu gradient zejścia zmniejsza swój szum. Skutkiem tego jest zwiększona wydajność nauczania sieci.

Nasz wygenerowany model sieci będziemy uczyć na zbiorze danych Irysów. Drugi model będzie reprezentantem sieci typu autoenkoder – wzorce zostały przekazane w treści zadania.

Znalezienie najstabilniejszej oraz najwydajniejszej sieci polega na wykonaniu dużej ilości eksperymentów, tj. sprawdzenie wydajności sieci dla różnych ilości warstw, czy też sprawdzenie wydajności dla różnej ilości neuronów w warstwie.

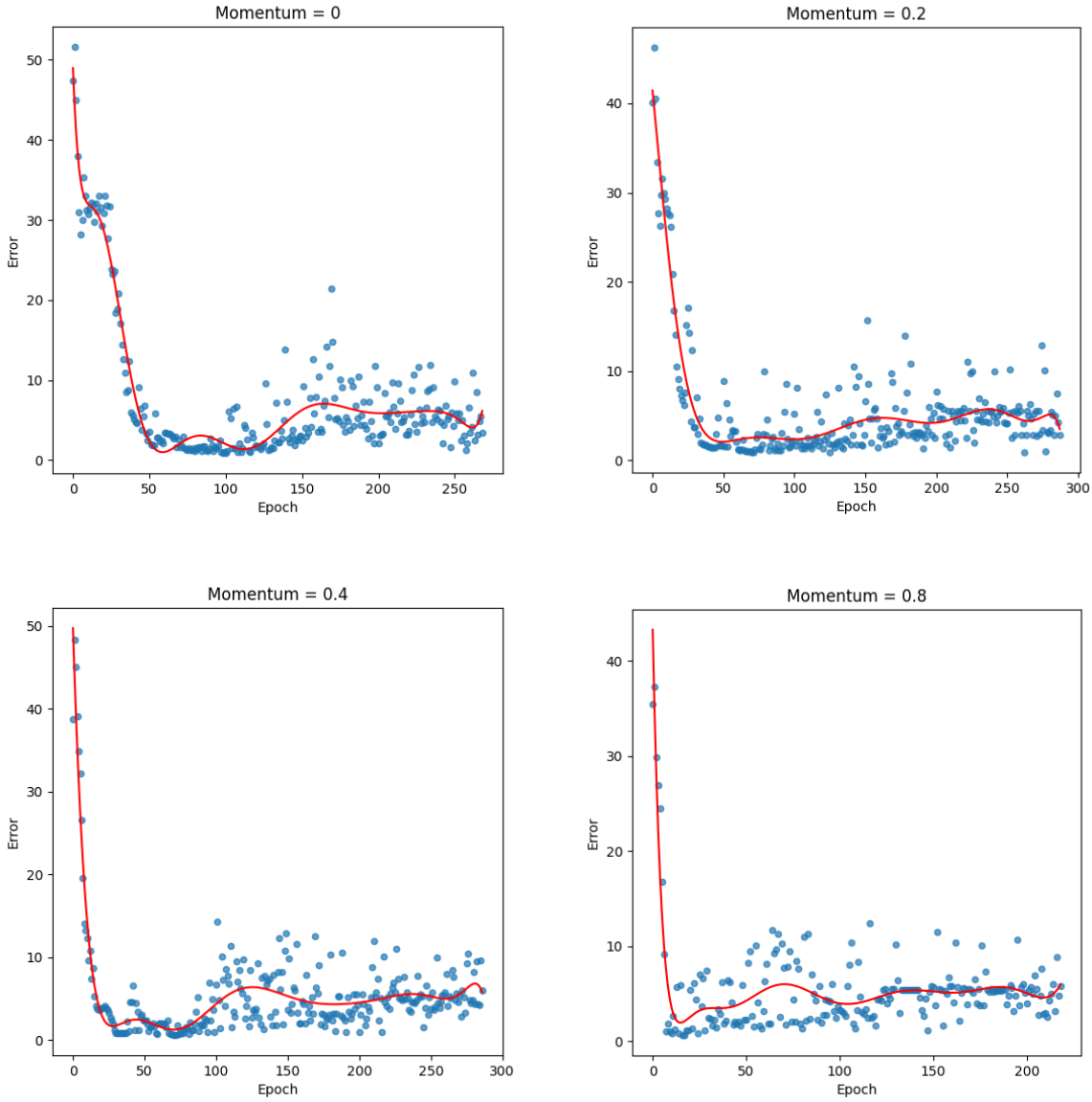
2. Badania

Implementacja miała nam posłużyć do realizacji dwóch zadań: klasyfikacji zbioru irysów oraz autoasocjacji.

2.1. Zbiór Irysów

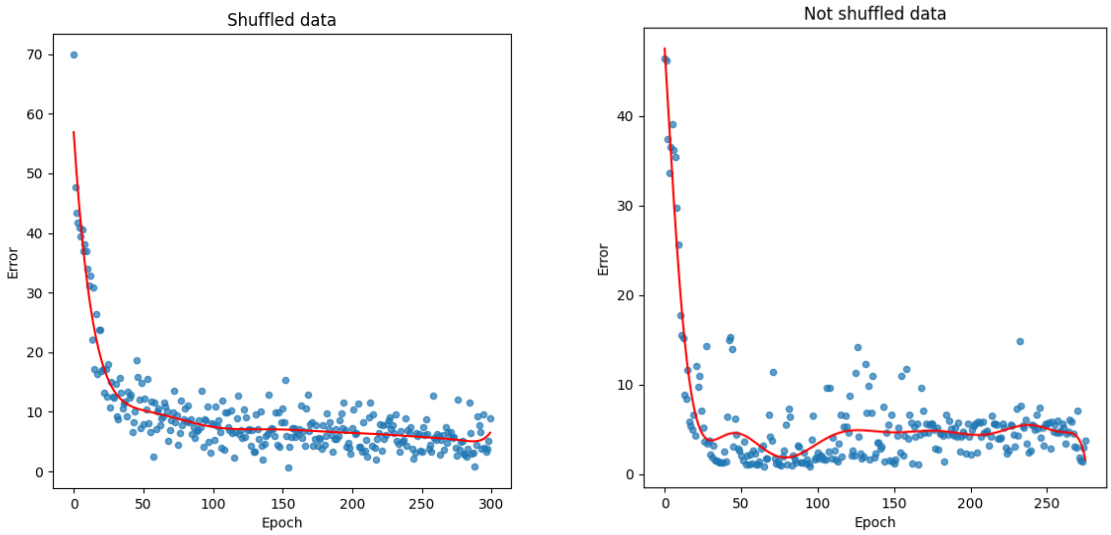
Jako najstabilniejszy model sieci dla danych treningowych oraz testowych Irysów przebadaliśmy architekturę sieci 4-4-3. Błąd całej sieci zachowywał się w granicach poniżej 1% i kończył proces nauki w 100-200 epokach.

Porównując szybkość uczenia z momentum oraz bez gołym okiem możemy zobaczyć różnicę. Przewaga po stronie nauki z tym parametrem wynika ze zmniejszonego szumu przy wyznaczaniu gradientu.



Rys. 1. Wykresy błędów podczas nauki sieci (model 4-4-3) względem epoki.

Dzięki uporządkowanym danym sieć jest w stanie szybciej zaadaptować się do zbioru wzorców. Jest to przyczyną tego, że dla dużej liczby kolejnych danych algorytm on-line dostosowuje wagi dla konkretnego wzorca danych wyjściowych. W przeciwieństwie do pomieszanego zbioru danych, gdzie sieć dostosowuje swoje wagi raz do jednego wzorca wyjściowego, a raz do drugiego.



Rys. 2. Wykresy błędów podczas nauki sieci (model 4-2-4) względem epoki.

Jak widzimy jest prawdą to, że uporządkowane dane mają wpływ na szybkość nauki. Wykresy pokazują również mniejsze zróżnicowanie błędów między epokami na korzyść danych pomieszanych. Jest to skutek tego, że sieć nie jest zaskakiwana nagłymi zmianami wzorców wyjściowych.

Dane Irysów zostały podzielone na zestaw treningowy oraz zestaw testowy dla rzetelniejszego przebadania powodzenia nauki naszej sieci. Najlepsza sieć jaką udało nam się uzyskać posiada 0.51 błędu całkowitego, co przekłada się na bardzo dobre wyniki przewidywania poprawnych danych wyjściowych. W przypadku danych testowych, które nie były częścią danych przekazanych w procesie trenowania sieci, uzyskaliśmy 100% poprawnych wyników dla każdej klasy kwiatu.

		Wynik na wyjściu				
		Setosa	Versicolor	Virginica		
Oczekiwany wynik	Setosa	10	0	0	Całkowita populacja	30
	Versicolor	0	10	0		
	Virginica	0	0	10		
	Precyzja	1	1	1		
	Recall	1	1	1		
	F-measure	1	1	1		

Tab. 1. Macierz pomyłek dla modelu 4-4-3 z błędem całkowitym 0.51

Przeuczenie to zjawisko, podczas którego sieć traci swoją wydajność na podanym zbiorze w wyniku zbyt dużej ilości generacji.

		Wynik na wyjściu				
		Setosa	Versicolor	Virginica		
Oczekiwany wynik	Setosa	10	0	0	Całkowita populacja	30
	Versicolor	0	9	1		
	Virginica	0	0	10		
	Precyzja	1	0.9	1		
	Recall	1	1	0.9		
	F-measure	1	0.95	0.95		

Tab. 2. Macierz pomyłek dla modelu 4-3-3 z błędem całkowitym 0.89

		Wynik na wyjściu				
		Setosa	Versicolor	Virginica		
Oczekiwany wynik	Setosa	10	0	0	Całkowita populacja	30
	Versicolor	0	7	3		
	Virginica	0	0	10		
	Precyzja	1	0.77	1		
	Recall	1	1	0.77		
	F-measure	1	0.87	0.87		

Tab. 3. Macierz pomyłek dla przeuczonego modelu 4-3-3 z błędem całkowitym 22.12

Jak widać przeuczona sieć gorzej radzi sobie z przewidywaniem wyników. Pierwszy model (tab. 2) wywodzi się z generacji nr 124, natomiast drugi z generacji nr 400.

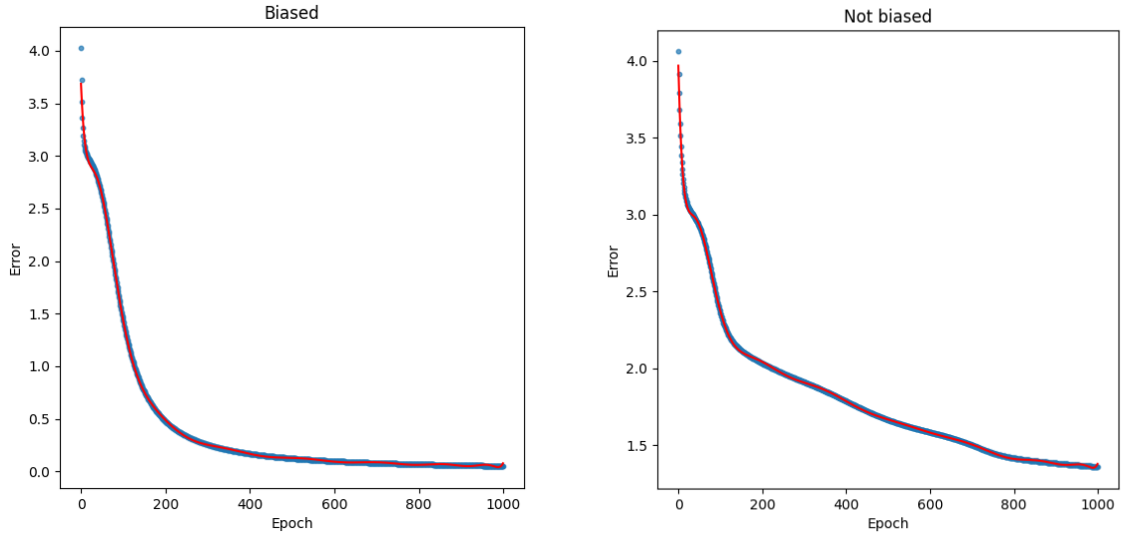
2.2 Sieć typu autoenkoder

Model sieci został narzucony z góry. Perceptron składa się z 3 warstw:

- Warstwa wejściowa (4 neurony)
- Warstwa ukryta (2 neurony)
- Warstwa wyjściowa (4 neurony)

Do zbadania mieliśmy wpływ uwzględnienia obciążenia w neuronach nieliniowych (wszystkich oprócz wejściowego) na skuteczność nauki podanych wzorców. Obciążenie, inaczej bias, jest inicjalizowane w sposób pseudolosowy wartościami z zakresu <-0.5, 0.5>.

Nauka jest prowadzona przy losowej kolejności wzorców podawanych na sieć w każdej epoce. Współczynnik nauki wynosi 0.6, a momentum nie jest w tym przypadku uwzględniane.

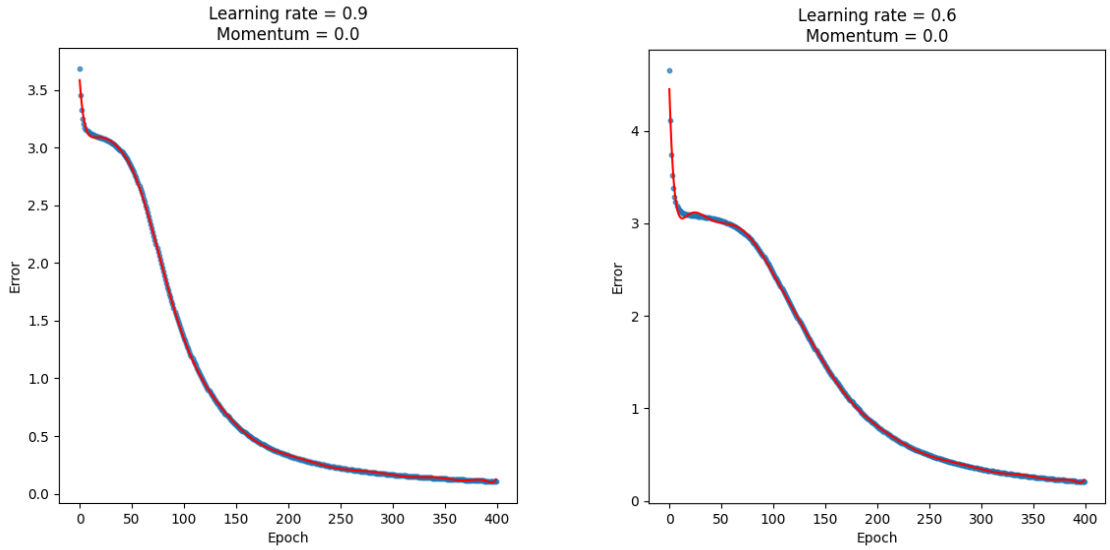


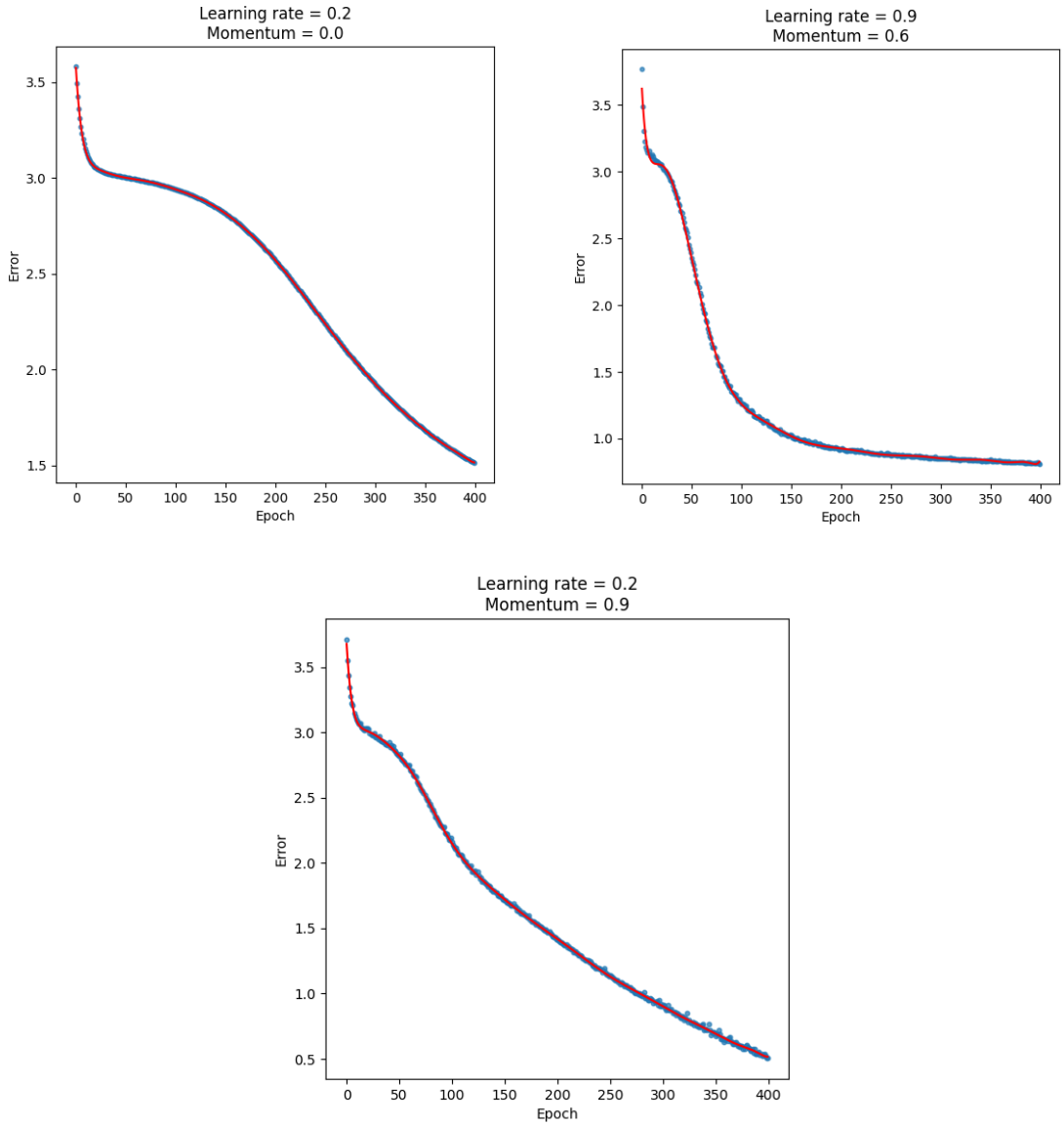
Rys. 3. Wykresy błędów podczas nauki sieci (model 4-2-4) względem epoki

Biorąc pod uwagę szybkość nauki oraz wydajność sieci zdecydowanym faworytem jest perceptron z uwzględnionym biasem. Przyczyną tego jest większa elastyczność przy obliczaniu wartości z funkcji aktywacji. Bias umożliwia przesuwanie funkcji po osi OX, dzięki czemu sieć ma większy zakres wartości wyjściowych dla różnych wartości wejściowych.

Do kolejnych eksperymentów zastosowaliśmy model z uwzględnionym biasem ze względu na stanowczo lepszą wydajność.

Mamy za zadanie zbadać szybkość uczenia perceptronu w zależności od uwzględnienia członu momentum i wartości współczynnika uczenia oraz momentum.





Rys. 4. Wykresy błędów podczas nauki sieci typu autoenkoder względem epoki

Widzimy, że uwzględnienie momentum istotnie wpływa na proces uczenia. Po wykresach możemy również stwierdzić, że większa wartość współczynnika uczenia proporcjonalnie wpływa na szybkość nauki perceptronu.

3. Wnioski końcowe

- Zbyt duża ilość warstw powoduje skomplikowanie kalkulacji dotyczących dostosowania wag do wzorca, tym samym do niepoprawnej nauki sieci.
- Zbyt duża ilość epok powoduje przeuczenie sieci skutkujące słabą wydajnością perceptronu.
- Momentum przyspiesza proces nauki, dzięki naprowadzaniu gradientu na rzeczywiste minimum.
- Bias ma duży wpływ na proces nauki sieci ze względu na większą elastyczność między wartościami wchodzącymi a wychodzącymi.
- Większa wartość współczynnika nauki powoduje szybszą, ale mniej dokładną naukę sieci i odwrotnie.