Sieci MLP raport

Laura Hoang, 313366

Kwiecień 2023

1 NN1: Bazowa implementacja

W ramach tej części przygotowany został szkielet klasy, który pozwalał na inicjalizację sieci z różną liczbą warstw i neuronów w każdej z warstwie.

Na tym etapie parametry sieci zostały dobrane ręcznie (m.in. korzystając z kalkulatora Desmos). Najlepsze wyniki wyszły dla poniższych architektur:

Zbiór danych	**Architektura**	**MSE (train)**	**MSE (test)**
square_simple	1-5-1 oraz 1-10-1	4.466	5.252
steps_large	1-5-1 oraz 1-10-1	4.19	4.655

Architektury **1-5-1** i **1-10-1** miały identycznie dobrane wagi, tylko na innej liczbie neuronów, co w efekcie dało tą samą wartość MSE dla zbioru treningowego i testowego, dla **square_simple** oraz **steps_large**.

2 NN2: Implementacja propagacji wstecznej błędu

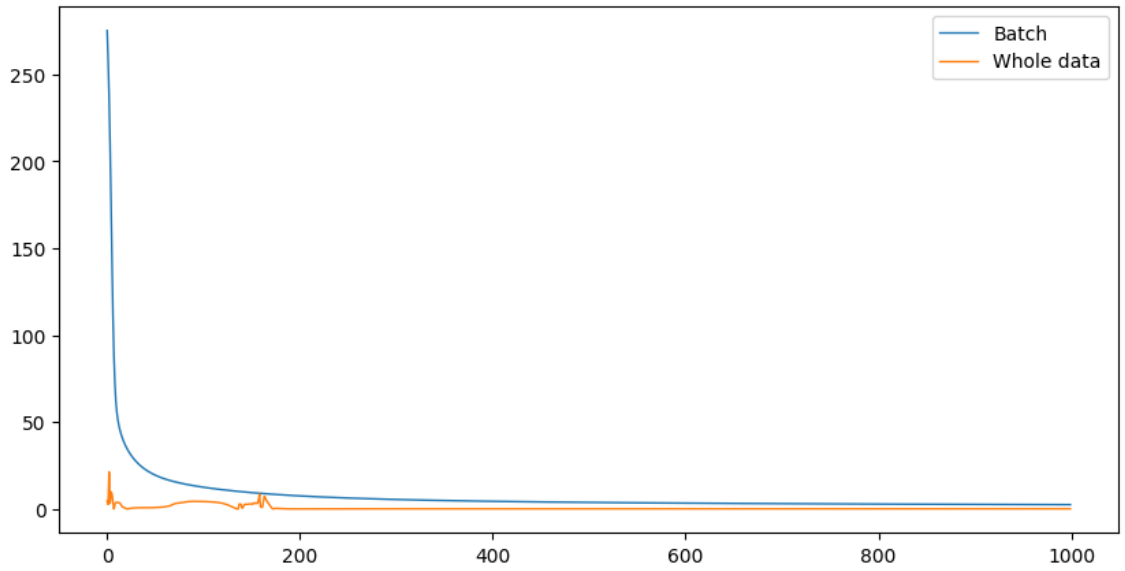
W ramach tej części, zaimplementowana została propagacja wsteczna sieci neuronowej. Udało się uzyskać wyniki przedstawione w poniższej tabeli:

Zbiór danych	**Architektura**	**MSE (train)**	**MSE (test)**
square_simple	1-400-1	0.25	0.32
steps_small	1-10-10-10-10-1	0.33	42.7
multimodal_large	1-50-50-50-1	5.65	3.34

Wszystkie wyniki spełniają wymogi z zadania, poza zbiorem **steps-small**.

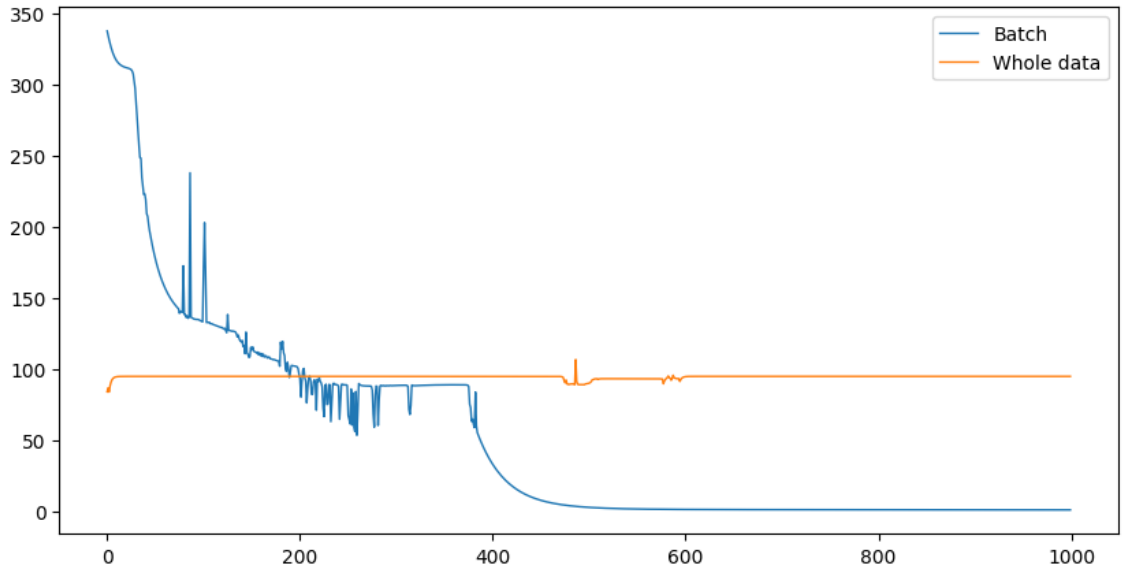
Oprócz tego, porównane zostało, jaki wpływ na szybkość zbieżności dla danych ma wielkość batch size'a [1]. Porównania zostały wykonane parami na tych samych zbiorach i przy tych samych architekturach. Poniżej pokazane są wykresy jak zmieniała się wartość error ($||\hat{y}||_F = \sqrt{\sum_i (y_i - \hat{y}_i)^2}$) wraz z kolejnymi epochami:

[1] Przy implementacji gdy batch size jest 1, uczenie przebiega na całym zbiorze.



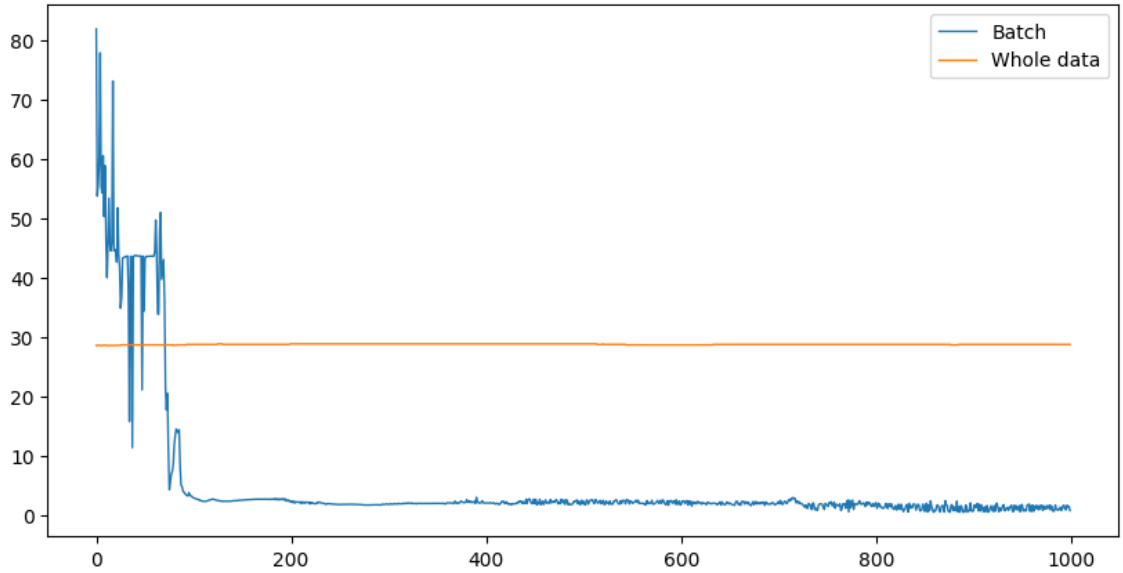
Rysunek 1: Porównanie zbieżności przy aktualizacji wag po batcha'ch vs. przy aktualizacji po całym zbiorze dla zbioru **square-simple**

W tym wypadku, zbieżność jest lepsza przy wyborze metody uczenia na całym zbiorze niż przy dobranym batch size (tutaj batch size jest równy 10). Im większy batch size, tym gorsze były wyniki.



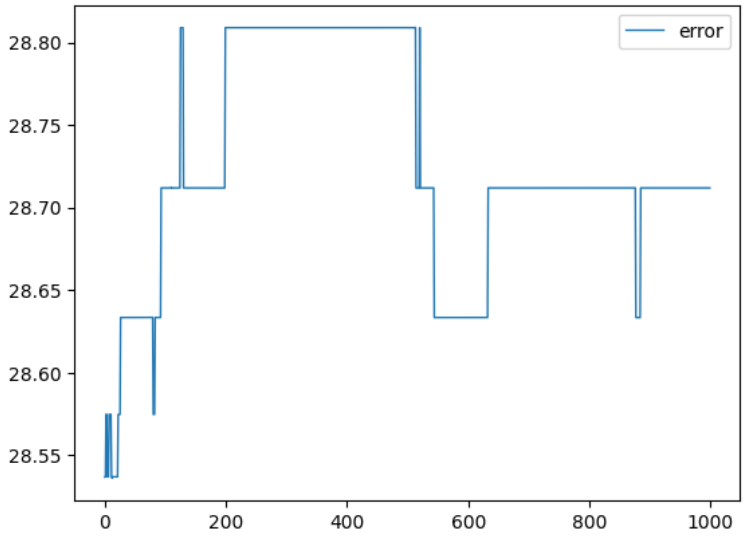
Rysunek 2: Porównanie zbieżności przy aktualizacji wag po batcha'ch vs. przy aktualizacji po całym zbiorze dla zbioru **steps-small**

Natomiast, dla tego przypadku, przy wyborze metody uczenia na całym zbiorze, zbieżność jest słaba, a dla dobranego batch size jest znacznie lepiej (tutaj batch size równy 40).



Rysunek 3: Porównanie zbieżności przy aktualizacji wag po batcha'ch vs. przy aktualizacji po całym zbiorze dla zbioru **multimodal-large**

W przypadku tego zbioru, dla dobranego batch size, metoda dobrze zbiega (tutaj batch size jest równy 30), a przy uczeniu modelu na całym zbiorze, error "skacze"i nie maleje. Poniższy wykres pokazuje opisane zjawisko [2]:



Rysunek 4: Zbieżność przy aktualizacji wag po całym zbiorze dla zbioru **multimodal-large**

Z wykresów można wywnioskować, że aktualizacja po batch'u cały czas zmniejszała wartości, a aktualizacja na całym zbiorze "skakała", lecz globalnie nie zmieniała error'u.

[2]Rysunek 4 reprezentuje pomarańczową linię z Rysunku 3.

3 NN3: Implementacja momentu i normalizacji gradientu

W ramach tej pracy należało zaimplementować dwa usprawnienia uczenia gradientowego sieci neuronowej. Były to:

- moment, czyli stochastic gradient descent z momentem,
- normalizację gradientu - RMSProp.

Uzyskane MSE dla kolejnych metod zostały przedstawione w poniższej tabeli[3]:

Zbiór danych	**Klasyczna**	**Momentum**	**RMS**
square_large	0.015 (47.058)	0 (45.997)	3.163 (87.246)
steps_large	16.55	12.438	40.787
multimodal_large	0.656	1.194	2.592

W badanych przypadkach najlepsze wyniki daje metoda klasyczna lub z zastosowaniem momentum. Być może RMS można dopracować przy odpowiednim doborze parametrów. Oprócz tego, nie udało się odpowiednio dobrać modelu aby ulepszyć MSE dla zbioru 'steps-large', co było problemem również przy NN2.

4 NN4: Rozwiązywanie zadania klasyfikacji

W ramach tego tematu zaimplementowane zostały: funkcja softmax dla warstwy wyjściowej sieci neuronowej oraz nowa funkcja straty, f-measure.

Należało porównać działanie modelu przy problemie klasyfikacji bez i z wykorzystaniem softmax.

Ta implementacja zapewniła otrzymanie poniższych wyników, które spełniły wymagania dla większości przypadków. Jedynie na zbiorze testowym **xor3** nie udało się osiągnąć f-measure na wystarczającym poziomie.

Zbiór danych	**Bez softmax**	**Z softmax**
rings3_regular	0.7015	0.819
easy	0.998	1.0
xor3	0.87	0.876

Jak widać po otrzymanych wynikach, softmax ulepsza modele.

[3]Dla zbioru **square_large**, MSE na oryginalnych danych testowych były wysokie, ponieważ zawierały x należące do przedziału, którego nie było w zbiorze treningowym. Z tego powodu zbadane zostały dane testowe "ucięte"(dla $x \geq 1.5$). Wyniki w nawiasach odnoszą się do oryginalnych danych testowych (nieuciętych)

5 NN5: Testowanie różnych funkcji aktywacji

W ramach tego laboratorium należało rozszerzyć implementację sieci o możliwość wyboru poniższych funkcji aktywacji:

- sigmoid,
- liniowa,
- tanh,
- ReLU.

Należało również uwzględnić zmianę pochodnej funkcji aktywacji.

W przypadku ReLU, nie implementuje się dokładnie gradientu dla tej funkcji, ponieważ nie różniczkuje się w 0. Co więcej, niekiedy dla $x < 0$ daje się pochodną równą 0.01, aby dalej pomimo aktywacji zerowej gradient nie został wyzerowany.

Poniżej przedstawione są wyniki (wartości MSE) testów wstępnych dla zbioru **multimodal-large**:

Funkcja aktywacji	**1-16-1**	**1-32-16-1**	**1-32-32-16-1**
sigmoid	995.1	27.8	**1.13**
linear	4436	4438.7	4495.1
tanh	461.4	448.5	391.9
ReLU	1717.8	**8.15**	320

Jak widać w powyższej tabelce, najlepsze wyniki zostały osiągnięte dla funkcji sigmoid z architekturą 1-32-32-16-1 oraz dla funkcji ReLU z architekturą 1-32-16-1.

W kolejnym etapie należało zbadać skuteczność funkcji aktywacji dla innych zbiorów danych: dla zadania regresji (**steps-large**) oraz klasyfikacji (**rings3-regular** i **rings5-regular**).

MSE na zbiorze **steps-large** prezentuje się w następujący sposób dla najlepszych wariantów:

Funkcja aktywacji	**Architektura**	**MSE**
sigmoid	1-32-32-16-1	15.9
ReLU	1-32-16-1	34.4

Następnie, zaprezentowane zostały wyniki dla zadań klasyfikacji [4]: dla **rings3-regular** osiągnięte wartości F-measure to:

Funkcja aktywacji	**Architektura**	**F-measure**
sigmoid	2-64-64-3	0.73
tanh	2-64-64-3	0.8

[4]Pierwsza i ostatnia warstwa w architekturze to wejście i wyjście. W związku z tym, dla zadania klasyfikacji jest wielowymiarowy input.

a dla **rings5-regular** osiągnięte wartości F-measure to:

Funkcja aktywacji	**Architektura**	**F-measure**
sigmoid	2-50-50-5	0.71
ReLU	2-10-5	0.66

6 NN6: Zjawisko przeuczenia + regularyzacja

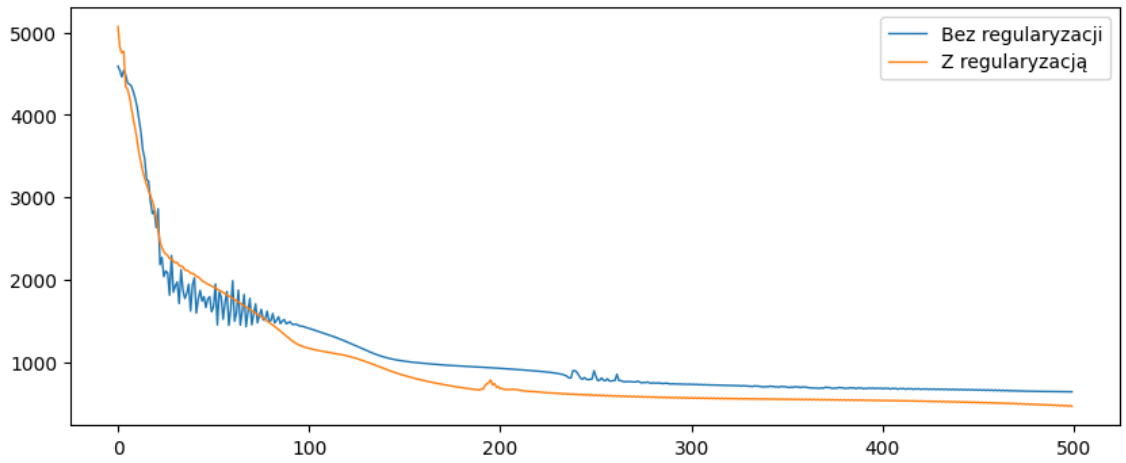
W ramach tego tematu należało zaimplementować mechanizm regularyzacji wag w sieci oraz mechanizm zatrzymywania uczenia przy wzroście błędu na zbiorze walidacyjnym. W tej pracy jako mechanizm regularyzacji wag zaimplementowana została regularyzacja L2.

Natomiast mechanizm zatrzymywania uczenia przy wzroście błędu na zbiorze walidacyjnym działa w ten sposób, że jeśli w epoce wynik się nie poprawi, to uczenie jest zatrzymywane i zwracana jest sieć z wagami, która osiągnęła najlepsze wyniki na zbiorze walidacyjnym.

Wyniki eksperymentu sprawdzającego wpływ regularyzacji na wynik na zbiorze testowym są przedstawione w poniższej tabeli:

Zbiór danych	**Bez regularyzacji**	**Regularyzacja**
multimodal_sparse	MSE: 505.5	MSE: 433.9
rings5_sparse	Log_loss: 0.7	Log_loss: 1.60
rings3_balance	Log_loss: 0.61	Log_loss: 1.09
xor3_balance	Log_loss: 0.01	Log_loss: 0.69

Przykładowo, porównanie zbieżności obu metod w kolejnych epokach dla zbioru **multimodal_sparse** jest przedstawione na poniższym wykresie:



Rysunek 5: Porównanie zbieżności bez regularyzacji vs. z regularyzacją L2 dla zbioru **multimodal_sparse**

7 Podsumowanie

- Aktualizacja po batch'u daje stabilne spadki wartości, a aktualizacja na całym zbiorze - nie oraz globalnie nie zmienia error'u.
- Lepszą metodą usprawnienia uczenia gradientu jest zastosowanie momentum, niż RMS.
- W przypadku klasyfikacji binarnej, softmax może osiągąć lepsze wyniki F-score niż sigmoid.
- W przypadku regresji najlepiej używać ReLU lub sigmoid jako funkcję aktywacji. Natomiast w przypadku klasyfikacji najlepiej używać sigmoid lub tanh jako funkcję aktywacji.
- Aktywacja liniowa w warstwie ukrytych nie powinno się stosować, ponieważ powstaje tylko przekształcenie liniowe neuronów wejściowych.
- Regularyzacja może pomóc w przypadku, gdy dane wejściowe są niezbilansowane lub jest ich mało.