Sprawozdanie z sieci MLP w ramach przedmiotu Metody Inteligencji Obliczeniowej w Analizie Danych

Adrianna Grudzień
nr indeksu: 305721

24 V Anno Domini MMXXII

Spis treści

1 Wprowadzenie

Założeniem tej części laboratorium było zapoznanie się z budową modelu perceptronu wielowarstwowego – sieci neuronowej typu feedforward. Ćwiczenia wykonywane na laboratorium nie miały na celu jedynie uzyskania dobrych wynikach w dość prostych problemach predykcyjnych. Głównym celem była wizualizacja budowy sieci i zmian wartości parametrów w trakcie procesu uczenia. Podsumowaniem serii laboratoriów na temat MLP jest to sprawozdania, w którym zawarłam obserwacje na temat zachowania się procesu uczenia sieci w zależności od różnych ustawień hiperparametrów.

W związku z tym, że w ramach zajęć modyfikowałam różne elementy i obserwowałam ich wpływ, nie korzystałam z gotowych bibliotek implementujących sieć neuronową. Do implementacji użyłam jedynie bibliotek realizujących działania macierzowe, np. numpy.

2 Zrealizowane tematy

2.1 NN1: Bazowa implementacja

2.1.1 Opis zadania

Zaimplementowałam sieć neuronową typu MLP, w której można ustawić: liczbę warstw, liczbę neuronów w każdej z warstw i wagi poszczególnych połączeń, w tym biasów. Sieć używa sigmoidalnej funkcji aktywacji. Na wyjściu dopuszczana jest funkcja liniowa.

Implementacja sieci została przygotowana w taki sposób, żeby łatwo zmieniać:

- Architekturę, to znaczy liczbę wejść, wyjść, neuronów w warstwach ukrytych.
- Funkcję aktywacji.

Tak przygotowaną implementację wykorzystałam do rozwiązania zadania regresji na dostarczonych danych. Parametry sieci dobrałam ręcznie, tak aby uzyskać możliwie dobre wyniki na zbiorach danych:

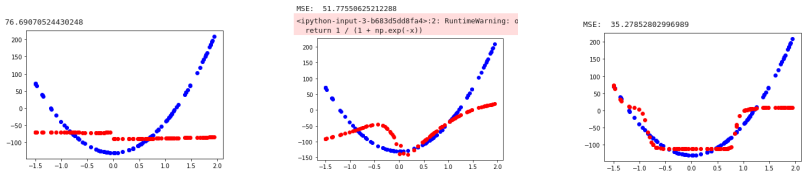
- square-simple
- steps-large

Rozważyłam architektury sieci:

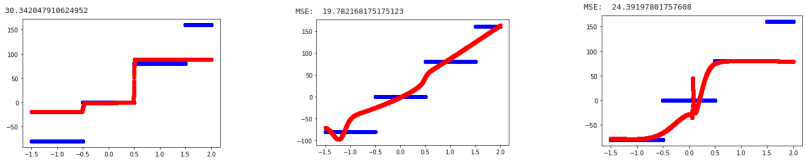
- jedna warstwa ukryta, 5 neuronów,
- jedna warstwa ukryta, 10 neuronów,
- dwie warstwy ukryte, po 5 neuronów każda.

2.1.2 Przebieg eksperymentu

Wagi dobierałam ręcznie. Niestety, mimo dobrych chęci, tak uzyskane wartości wag nie pozwoliły na otrzymanie dobrych wyników modelu. Aby to zwizualizować, stworzyłam wykresy przedstawiające prawdziwe wartości obserwacji oraz te „przewidziane" przez model wartości.



Rysunek 1: Predykcje modelu na zbiorze **square-simple** (kolor niebieski - rzeczywiste wartości, czerwony - predykcje) (a) jedna warstwa ukryta, 5 neuronów (b) jedna warstwa ukryta, 10 neuronów (c) dwie warstwy ukryte, po 5 neuronów każda.



Rysunek 2: Predykcje modelu na zbiorze **steps-large** (kolor niebieski - rzeczywiste wartości, czerwony - predykcje) (a) jedna warstwa ukryta, 5 neuronów (b) jedna warstwa ukryta, 10 neuronów (c) dwie warstwy ukryte, po 5 neuronów każda.

2.1.3 Wnioski

- Ręczne dobieranie wag to sposób czasochłonny i mało skuteczny. Lepiej tego nie robić.

2.2 NN2: Implementacja propagacji wstecznej błędu

2.2.1 Opis zadania

W ramach tego laboratorium zaimplementowałam uczenie sieci neuronowej propagacją wsteczną błędu. Aby sprawdzić implementację, wykonałam uczenie na prostych danych do uczenia dostarczonych na zajęciach. Następnie zaimplementowałam metodę wizualizacji wartości wag sieci. Zaimplementowałam wersję z aktualizacją wag po prezentacji wszystkich wzorców i wersję z aktualizacją po

prezentacji kolejnych porcji (batch). Porównać szybkość uczenia dla każdego z wariantów.

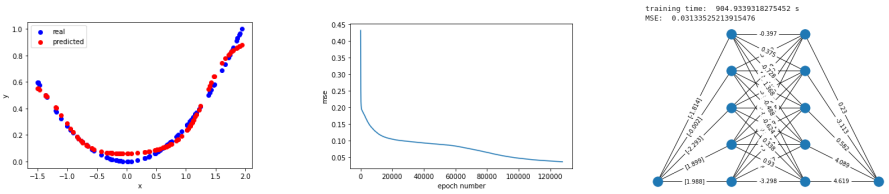
Początkowo inicjowałam wagi z rozkładu jednostajnego na przedziale [0,1], ale lepsze wyniki wychodziły dla rozkładu normalnego. Dla metody Xavier nie zaobserwowałam poprawy działania modelu.

Przetestowałam uczenie sieci na następujących zbiorach:

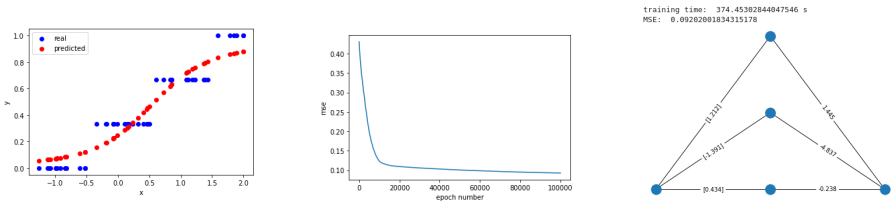
- square-simple,
- steps-small,
- multimodal-large.

2.2.2 Przebieg eksperymentu

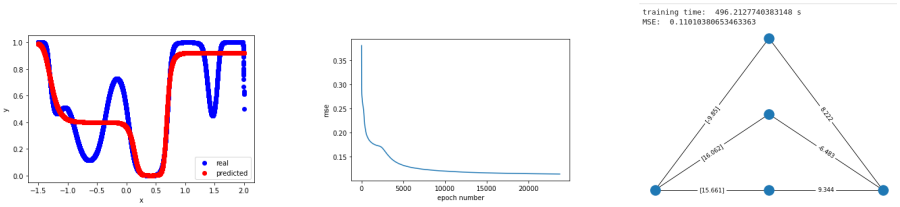
Niebacznie przed przystąpieniem do treningu sieci dokonałam normalizacji danych (tylko kolumny y). Uznałam, że wartościowe będzie pokazanie wyników dla tego eksperymentu w tymże sprawozdaniu. Okazało się, że przy znormalizowanych danych sieć uczy się szybciej i zgodnie z intuicją i definicją błędu MSE, błąd MSE od samego początku jest mały.
Sieć szybciej uczy się przy aktualizacji wag po prezentacji wszystkich wzorców.



Rysunek 3: Wyniki eksperymentu na zbiorze **square-simple** dla architektury [1,5,5,1] (a) predykcja modelu po wytrenowaniu (b) błąd MSE (c) wizualizacja sieci i jej wag.



Rysunek 4: Wyniki eksperymentu na zbiorze **steps-small** dla architektury [1,3,1] (a) predykcja modelu po wytrenowaniu (b) błąd MSE (c) wizualizacja sieci i jej wag.



Rysunek 5: Wyniki eksperymentu na zbiorze **multimodal-large** dla architektury [1,3,1] (a) predykcja modelu po wytrenowaniu (b) błąd MSE (c) wizualizacja sieci i jej wag.

2.2.3 Wnioski

- Dla architektur o większej liczbie neuronów sieć szybciej osiąga małe błędy MSE.
- Wynik uczenia sieci bardzo zależy od dobranych parametrów, tj. architektura sieci czy rozmiar parametru batch_size.
- Sieć szybciej uczy się przy aktualizacji wag po prezentacji wszystkich wzorców niż wersja z aktualizacją po prezentacji kolejnych porcji (batch).

2.3 NN3: Implementacja momentu i normalizacji gradientu

2.3.1 Opis zadania

Zaimplementowałam dwa usprawnienia uczenia gradientowego sieci neuronowej:

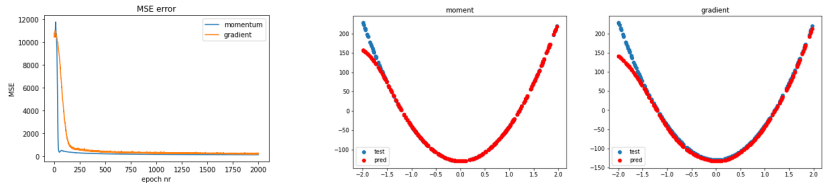
- moment,
- normalizację gradientu RMSProp.

Porównałam szybkość zbieżności procesu uczenia dla obu wariantów.
Przeprowadziłam eksperymenty na zbiorach:

- square-large,
- steps-large,
- multimodal-large.

2.3.2 Przebieg eksperymentu

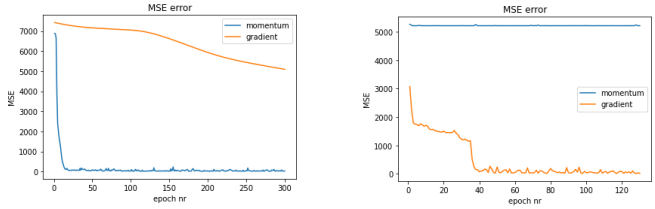
Wykonałam trening dla metody momentu oraz gradientu, a następnie porównałam wyniki (błąd MSE, czas treningu, predykcje). Następnie sprawdziłam wpływ zmian parametru learning_rate na błąd MSE.
Okazało się, że obie metody działają w podobnym czasie - w zależności od wybranych parametrów treningu różnica ta się zmieniała, dlatego nie można jednoznacznie określić, która metoda jest szybsza.



Rysunek 6: Wyniki eksperymentu na zbiorze **square-large** dla architektury [10,5,1] (a) predykcja modelu po wytrenowaniu (b) błąd MSE (c) wizualizacja sieci i jej wag.

```
Momentum fitting time:  240.99221301078796
Gradient fitting time:  211.8213996887207
```

Rysunek 7: Wyniki eksperymentu na zbiorze **square-large** dla architektury [10,5,1] - czas treningu w sekundach.



Rysunek 8: Błąd MSE dla eksperymentów na zbiorze (a) **steps-small** dla architektury [32, 32, 32, 32, 1] (b) **multimodal-large** dla architektury [32, 32, 32, 32, 1].

2.3.3 Wnioski

- Niskie wyniki błędu MSE osiąga wcześniej metoda momentu, jednakowoż przebieg treningu jest bardzo podobny. W obu metodach pojawia się pik w górę błędu MSE w początkowych fazach trenowania.
- Szybciej zbiega metoda momentu.
- Obie metody działają w porównywalnym czasie.

2.4 NN4: Rozwiązywanie zadania klasyfikacji

2.4.1 Opis zadania

Zaimplementować funkcję softmax dla warstwy wyjściowej sieci neuronowej. Sprawdzić szybkość i skuteczność w wariancie, gdy sieć używa funkcji softmax

na ostatniej warstwie i gdy jest użyta zwykła funkcja aktywacji.

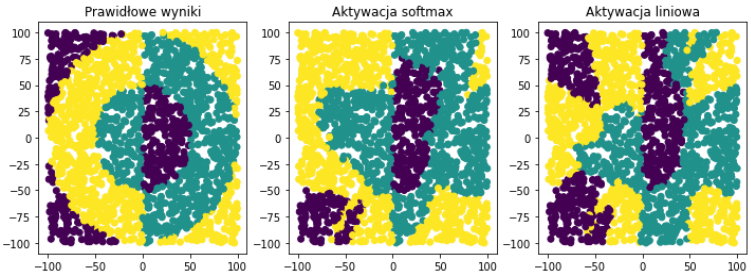
Przeprowadzić eksperymenty na zbiorach (w nawiasach wymagana wartość F-measure na 2 punkty):

- rings3-regular,
- easy,
- xor3

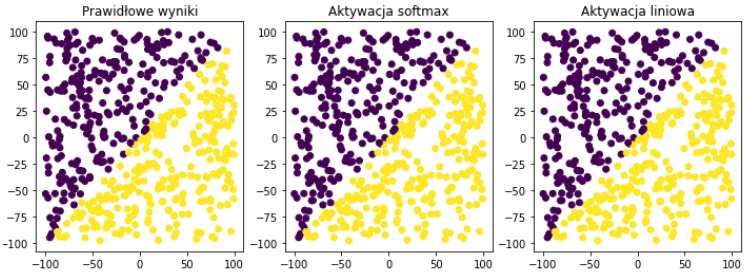
2.4.2 Przebieg eksperymentu

Wytrenowałam 6 sieci - po dwie dla każdego zbioru danych - które różniły się funkcją aktywacji w warstwie wyjściowej. Porównywałam działanie funkcji Softmax oraz funkcji liniowe na ostatniej warstwie. Dla funkcji Softmax funkcją straty była funkcja cross entropy, a dla funkcji liniowej - mse.

Zauważyłam duże podobieństwo wyników klasyfikacji w obu wariantach.



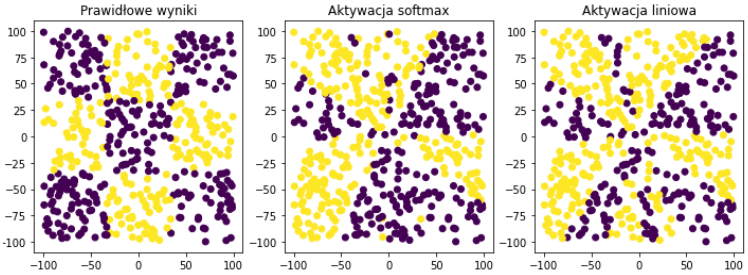
Rysunek 9: Wyniki eksperymentu na zbiorze **rings3-regular** dla architektury [30, 11, 3].



Rysunek 10: Wyniki eksperymentu na zbiorze **easy** dla architektury [20, 2].

2.4.3 Wnioski

- Wyniki klasyfikacji dla funkcji softmax przy funkcji straty - cross entropy są bardzo podobne do tych uzyskanych przy użyciu funkcji liniowej i mse.



Rysunek 11: Wyniki eksperymentu na zbiorze **xor** dla architektury [20, 2].

2.5 NN5: Testowanie różnych funkcji aktywacji

2.5.1 Opis zadania

Należy rozszerzyć istniejącą implementację sieci i metody uczącej o możliwość wyboru funkcji aktywacji:

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

Porównać szybkość uczenia i skuteczność sieci w zależności od liczby neuronów w poszczególnych warstwach i rodzaju funkcji aktywacji. Należy wziąć pod uwagę fakt, że różne funkcje aktywacji mogą dawać różną skuteczność w zależności od liczby neuronów i liczby warstw. Sprawdzić sieci z jedną, dwiema i trzema warstwami ukrytymi.

Przeprowadzić testy wstępne dla zbioru multimodal-large (regresja), dla wszystkich trzech architektur i wszystkich czterech funkcji aktywacji.

Dla pozostałych zbiorów wybrać dwa najlepsze zestawy i zbadać ich skuteczność:

- regresja
  - steps-large,
- klasyfikacja
  - rings5-regular
  - rings3-regular

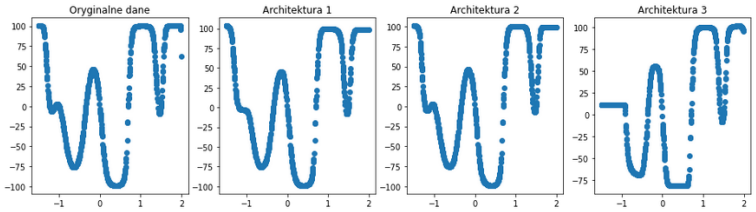
2.5.2 Przebieg eksperymentu

Wykonałam eksperymenty na zbiorze do regresji **multimodal-large** dla następujących architektur:

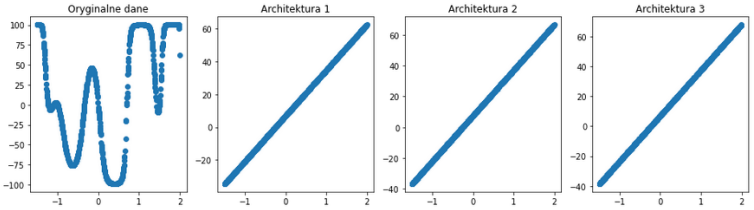
- architektura 1.: [32, 1] (jedna warstwa ukryta),
- architektura 2.: [40, 40, 1] (dwie warstwy ukryte),
- architektura 3.: [32, 32, 16, 1] (trzy warstwy ukryte).

Obserwacje dla poszczególnych funkcji aktywacji:

- Sigmoid - dużo zależy od doboru parametrów, ale ogólnie radzi sobie bardzo dobrze. Przy trzech warstwach ukrytych trochę gorzej.
- Linear - radzi sobie równie słabo dla dowolnej architektury.
- Tanh - radzi sobie całkiem dobrze, lepiej - gdy jest więcej warstw ukrytych. Jest trochę szybsza niż Sigmoid. Dla dużych rozmiarów parametru batch sieć osiągała gorsze wyniki (np. lepsze przy batch_size=4 niż 64).
- ReLU - osiąga słabe wyniki dla architektur z małą liczbą warstw ukrytych. Im więcej takich warstw, tym lepiej.

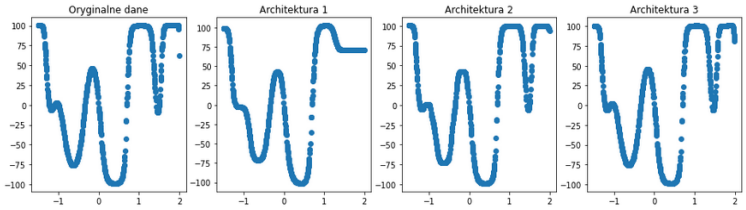


Rysunek 12: Wyniki eksperymentu dla funkcji **Sigmoid** na zbiorze **multimodal-large** dla poszczególnych architektur.

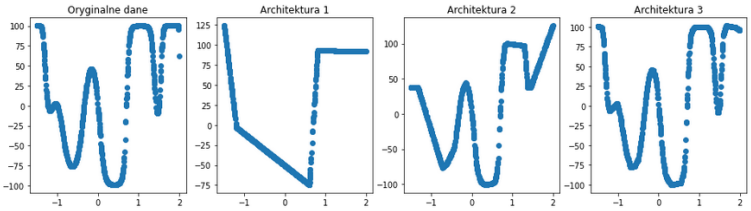


Rysunek 13: Wyniki eksperymentu dla funkcji **Linear** na zbiorze **multimodal-large** dla poszczególnych architektur.

Najlepsze pod względem błędu MSE okazały się zestawy:



Rysunek 14: Wyniki eksperymentu dla funkcji **Tanh** na zbiorze **multimodal-large** dla poszczególnych architektur.



Rysunek 15: Wyniki eksperymentu dla funkcji **ReLU** na zbiorze **multimodal-large** dla poszczególnych architektur.

	MSE - arch1	MSE - arch2	MSE - arch3
Sigmoid	4.110139	1.824580	495.202067
Linear	4433.789883	4437.943508	4442.128375
Tanh	329.240084	6.515091	2.590387
ReLU	1431.778197	355.191391	7.694468

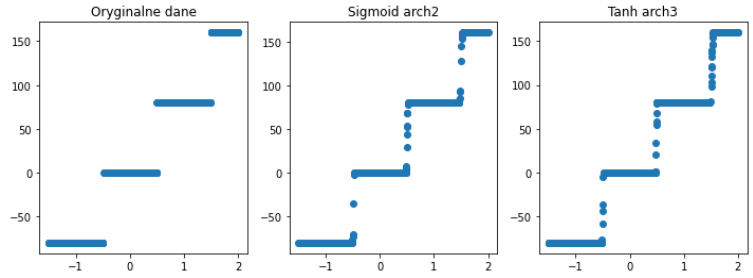
Rysunek 16: Wyniki eksperymentów dla poszczególnych funkcji aktywacji i poszczególnych architektur.

- funkcja ‘Sigmoid‘, architektura ‘nr 2‘, czyli z dwoma warstwami ukrytymi,
- funkcja ‘Tanh‘, architektura ‘nr 3‘, czyli z trzema warstwami ukrytymi.

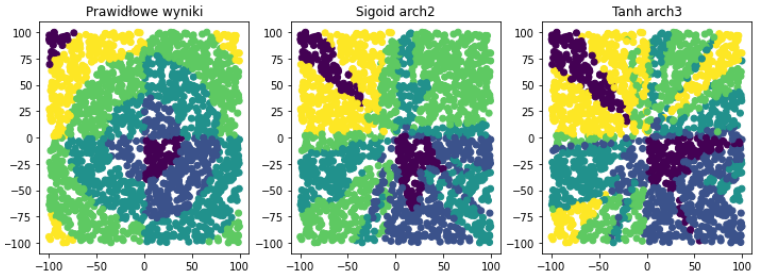
Dla tych dwóch wybranych zestawień przeprowadziłam eksperymenty na zbiorach **steps-large**, **rings5-regular**, **rings3-regular**.

2.5.3 Wnioski

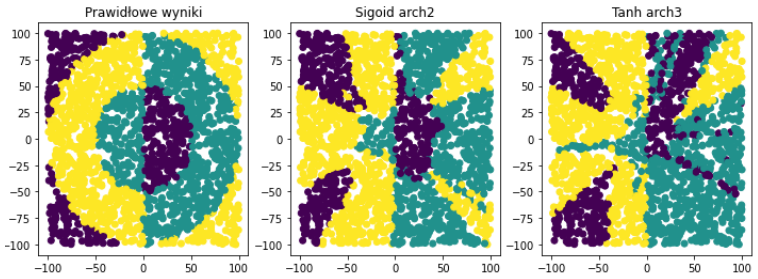
- Sieć jest bardzo wrażliwa na zmiany parametrów. Optymalny dobór ich to sztuka.



Rysunek 17: Wyniki eksperymentu na zbiorze **steps-large** dla optymalnych parametrów wybranych na podstawie wcześniejszych eksperymentów.



Rysunek 18: Wyniki eksperymentu na zbiorze **rings5-regular** dla optymalnych parametrów wybranych na podstawie wcześniejszych eksperymentów.



Rysunek 19: Wyniki eksperymentu na zbiorze **rings3-regular** dla optymalnych parametrów wybranych na podstawie wcześniejszych eksperymentów.

2.6 NN6: Zjawisko przeuczenia + regularyzacja

2.6.1 Opis zadania

Zaimplementować mechanizm regularyzacji wag w sieci oraz mechanizm zatrzymywania uczenia przy wzroście błędu na zbiorze walidacyjnym.

Przeprowadzić eksperymenty na zbiorach i porównać skuteczność na zbiorze

testowym dla różnych wariantów przeciwdziałania przeuczeniu sieci:

- multimodal-sparse,
- rings5-sparse,
- rings3-balance,
- xor3-balance.

Wykorzystałam regularyzację L2

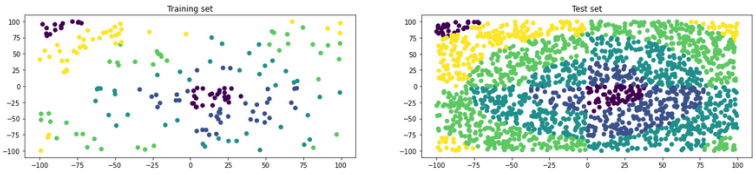
$$Cost\,function = Loss + \frac{\lambda}{2m} * \sum \|w\|^2$$

Rysunek 20: Wzór na funkcję kosztu dla regularyzacji L2 (lambda - parametr regularyzacji).

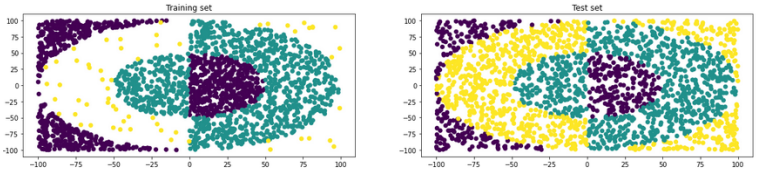
Zmniejszając wartości w macierzy wagowej (regularyzacja), zmniejsza się również wartość **z** na neuronie, co z kolei zmniejsza efekt funkcji aktywacji. W związku z tym do danych zostaje dopasowana mniej złożona funkcja, co skutecznie ogranicza nadmierne dopasowanie i powstrzymuje sieć przed przeuczeniem.

Wytrenowałam po kilka modeli dla każdego zbioru - zarówno dla zadania z regularyzacją, jak i bez, a następnie wyliczyłam na ich podstawie średnią metrykę - mse dla zadania regresji, f score dla zadania klasyfikacji - niezależnie na zbiorach treningowym oraz testowym. Tak uzyskane wyniki zamieściłam w tabelce podsumowującej.

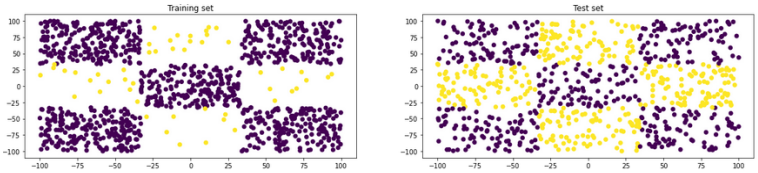
2.6.2 Przebieg eksperymentu



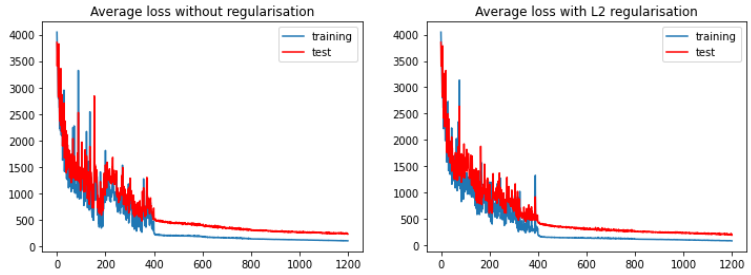
Rysunek 21: Zbiór **rings5-sparse**.



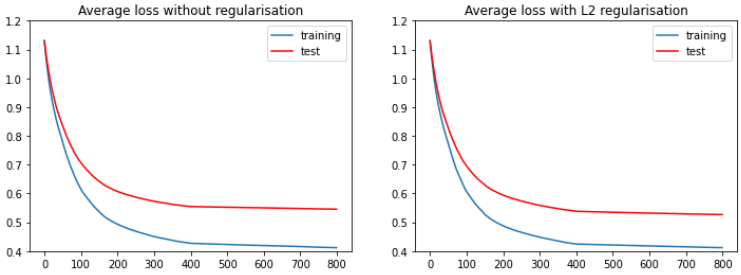
Rysunek 22: Zbiór **rings3-balance**.



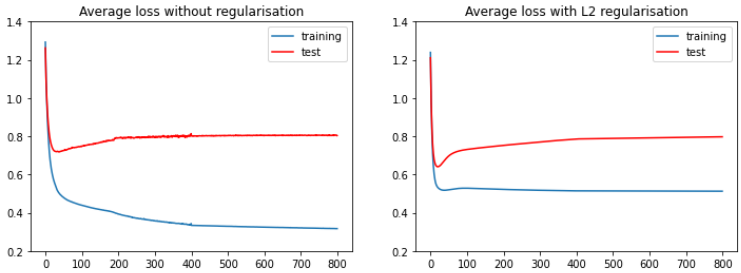
Rysunek 23: Zbiór **xor3-balance**.



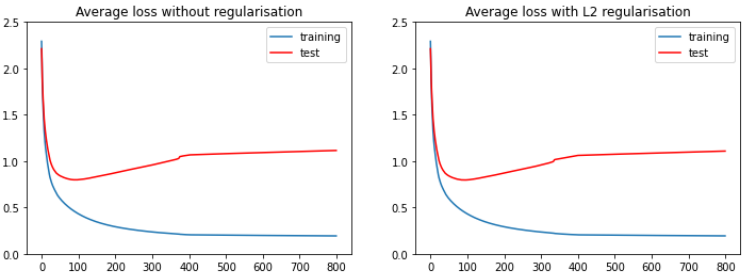
Rysunek 24: Uśredniona funkcja straty bez oraz z regularyzacją dla zbioru **multimodal-sparse**.



Rysunek 25: Uśredniona funkcja straty bez oraz z regularyzacją dla zbioru **rings5-sparse**.



Rysunek 26: Uśredniona funkcja straty bez oraz z regularyzacją dla zbioru **rings3-balance**. Przeuczenie.



Rysunek 27: Uśredniona funkcja straty bez oraz z regularyzacją dla zbioru **xor3-balance**. Przeuczenie.

2.6.3 Wnioski

- Regularyzacja (L2) wpływa pozytywnie na wyniki modelu. Sprawia, że sieć nie ulega przeuczeniu. Dla zadania regresji błąd MSE istotnie maleje po zastosowaniu regresji. Dla zadania klasyfikacji metryka f score rośnie.
- To, czy sieć poprawia wynik przy regularyzacji w dużej mierze zależy od

	metric	mean metric train	mean metric test
multimodal-sparse-no-reg	mse	99.827135	225.507490
multimodal-sparse-l2-0.01	mse	86.420571	195.392992
rings5-sparse-no-reg	f_score	0.447151	0.311712
rings5-sparse-l2-0.1	f_score	0.447817	0.312089
rings3-balanse-no-reg	f_score	0.817360	0.408309
rings3-balanse-l2-2	f_score	0.317279	0.136900
xor3-balance-no-reg	f_score	0.934801	0.405436
xor3-balance-l2-0.01	f_score	0.933491	0.400492
xor3-balanse-l2-3	f_score	0.929152	0.393245
xor3-balanse-l2-5	f_score	0.929152	0.393245
xor3-balanse-l2-0.01-2	f_score	0.949096	0.470508
rings5-sparse-l2-3	f_score	0.066667	0.016501

Rysunek 28: Tabelka podsumowująca zebrane wyniki. (Nazwy wierszy utworzone są według schematu: 'nazwa zbioru - no-reg/l2 - wartość parametru regularyzacji'.

doboru parametrów, w szczególności parametru regularyzacji.

- Dla zbioru rings5-sparse wynik się nieznacznie poprawił dla parametru 0.1, natomiast dla 3 drastycznie zmalał (f score).
- Dla zbioru rings3-balance wynik pogorszył się dla parametru 2.
- Dla zbioru xor3-balance wynik (f score) pogorszył się dla parametrów: 0.01, 3, 5, natomiast poprawił się dla parametru 2.
- Dla przeprowadzonych eksperymentów zależność zarówno wzrostowa, jak i malejąca zachodzi jednocześnie i na zbiorze treningowym, i na zbiorze testowym.

Koniec.