NN - Raport

Aleksandra Samsel - 313396

1 Wstęp

W dobie szybkiego rozwoju technologii informatycznych i narastającej złożoności problemów, z jakimi mierzy się współczesna nauka, rośnie zapotrzebowanie na zaawansowane narzędzia analizy danych. Jednym z kluczowych obszarów, który przyczynia się do postępu w tej dziedzinie, jest sztuczna inteligencja, a w szczególności sieci neuronowe. W ramach projektu laboratoryjnego poświęconego perceptronowi wielowarstwowemu (MLP) - jednemu z fundamentalnych modeli w dziedzinie sieci neuronowych typu feedforward - postawiono sobie za cel nie tylko zrozumienie mechanizmów stojących za efektywnością tych systemów w rozwiązywaniu problemów predykcyjnych, ale przede wszystkim pogłębione badanie wpływu różnorodnych hiperparametrów na proces uczenia. Podjęte eksperymenty i analiza wyników mają na celu nie tylko empiryczną weryfikację teoretycznych założeń modelu MLP, ale także rozwój intuicji i umiejętności niezbędnych do efektywnego stosowania i modyfikowania sieci neuronowych w praktyce.

2 Opis zadania

W ramach laboratoriów, przechodzimy przez serię skoncentrowanych zadań, mających na celu praktyczne opanowanie konstrukcji i funkcjonowania sieci neuronowych. Projekt rozpoczyna się od podstawowej implementacji sieci MLP, gdzie eksperymentujemy z różnymi architekturami, liczbą neuronów, i funkcjami aktywacji, by rozwiązać zadania regresji na wybranych zbiorach danych. Kolejne etapy obejmują wprowadzenie do metod uczenia sieci, w tym implementację propagacji wstecznej błędu oraz technik takich jak moment i normalizacja gradientu, które mają na celu usprawnienie procesu uczenia. Mamy również za zadanie badać wpływ różnych funkcji aktywacji na skuteczność sieci i przeprowadzić eksperymenty mające na celu identyfikację i przeciwdziałanie zjawisku przeuczenia, poprzez implementację mechanizmów regularyzacji i zatrzymywania uczenia.

3 Propagacja wsteczna błędu

Opis wykonywanych eksperymentów

Aby porównać uczenie Sicie dla całej próbki i dla batchów wykonałam po 5 eksperymentów dla każdej z wersji oraz dla 3 typów danych: square simple, steps-small i multimodal-large. Aby oba sposoby uczenia miały równe szanse to parametry były za każdym razem takie same, dodatkowo porównałam uczenie dla dwóch typów sieci (różna liczba neuronów oraz różna liczba warstwy, eksperyment wykonałam dla sieci uczącej się dla całej próbki a następnie serię testów wykonywałam dla sieci która poradziła sobie lepiej). Funkcją aktywacji dla wszystkich warstw ukrytych jest sigmoid oraz warstwie wyjściowa jest funkcją identycznościową. Natomiast wagi i bajasy były inicjowane metodą He.

Opis parametrów

- eta - krok uczenia

Square-simple

Początkowo porównałam uczenie się sieci z jedną warstwą ukrytą z 20 neuronami oraz sieci z 2 warstwami ukrytymi po 5 neuronów. Lepiej uczyła się sieć z jedną warstwą ukrytą, dlatego też dalsze eksperymenty były przeprowadzane dla takiej sieci.

Parametry:

- warstwy oraz liczba neuronów $20 \longrightarrow 1$
- $eta = 0.003$
- epoki – 20000
- batch – 32

Wyniki:

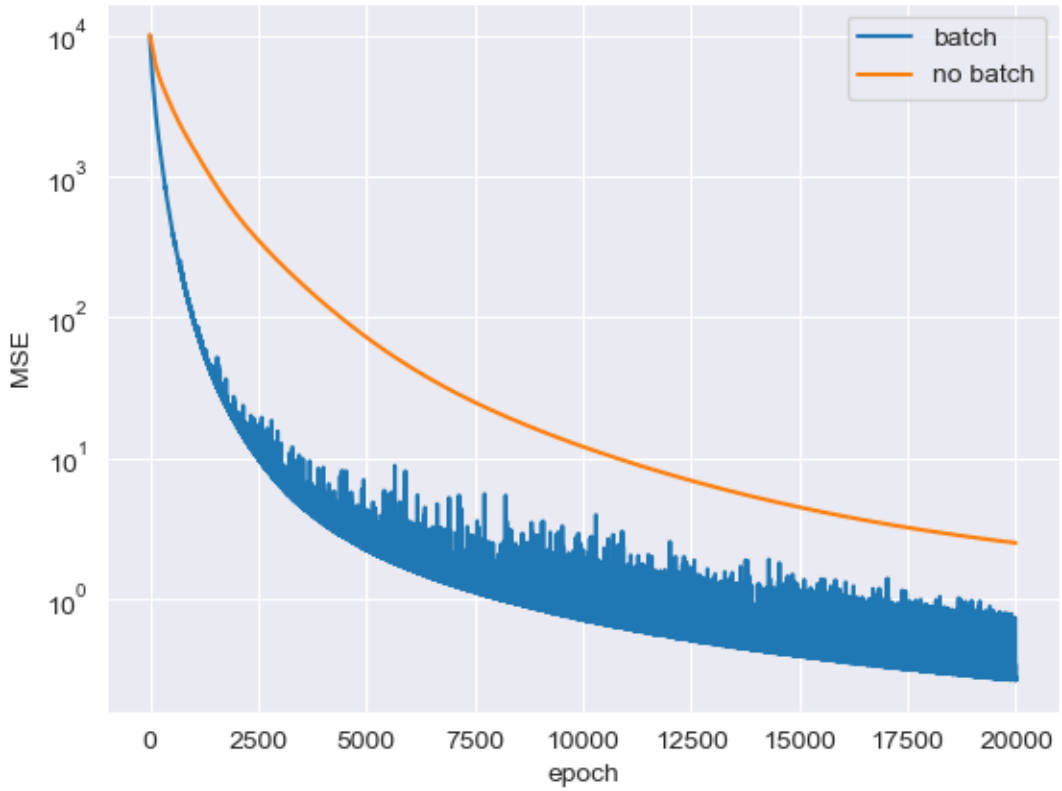
Bez batchów:

min	max	średnia
1.6903	10004.6152	238.5118
4.7366	10067.4208	306.8164
1.8482	10060.5598	267.9761
1.5488	10126.8335	259.99225
2.8633	10137.8792	301.8464

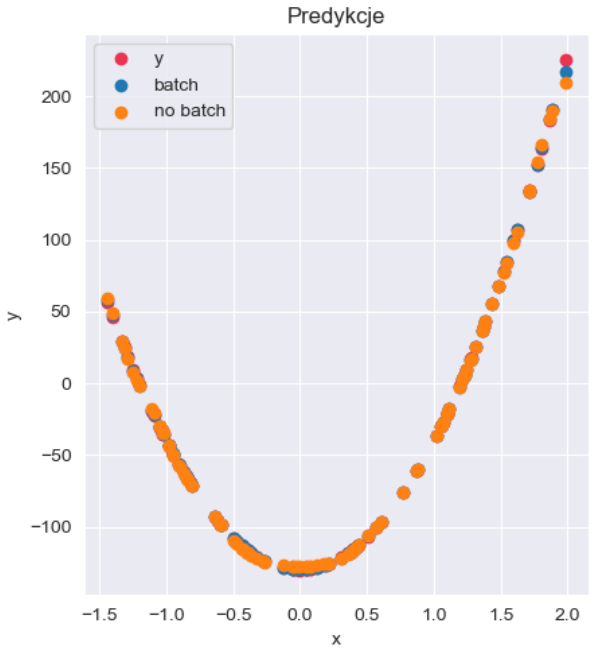
Z batchami:

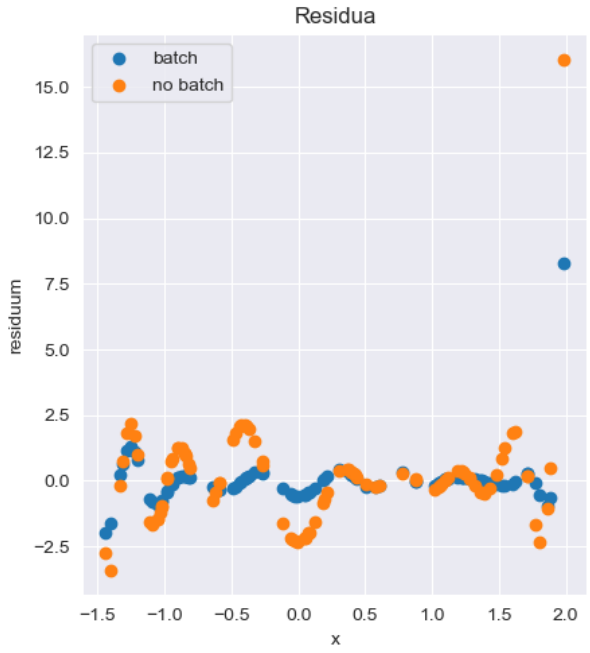
min	max	średnia
0.3257	9921.9955	61.5939
0.4591	9933.9485	69.6132
1.5593	9905.7019	104.9657
1.4670	9997.3476	86.0988
0.5950	9833.0080	76.1073

Widzimy dużą różnicę w osiągniętym minimalnym MSE dla uczenia na całej próbce i batchach. Obie metody osiągnęły zakładaną wartość MSE równą 4, jednak to eksperyment dla batchów wyszedł lepiej, widać też to na średnich wartościach, co wskazuje na szybszy spadek MSE. Możemy to zobaczyć na poniższym wykresie przedstawiającym wartość MSE dla kolejnych epok.



Możemy równie dobrze pokazać, że model uczący się na batchach jest lepszy poprzez porównanie wartości dla danych testowych na wykresie.





Steps-small

Parametry:

- warstwy oraz liczba neuronów 50 ⟶ 1
- eta = 0.01
- epoki – 100000
- batch – 32

Wyniki:

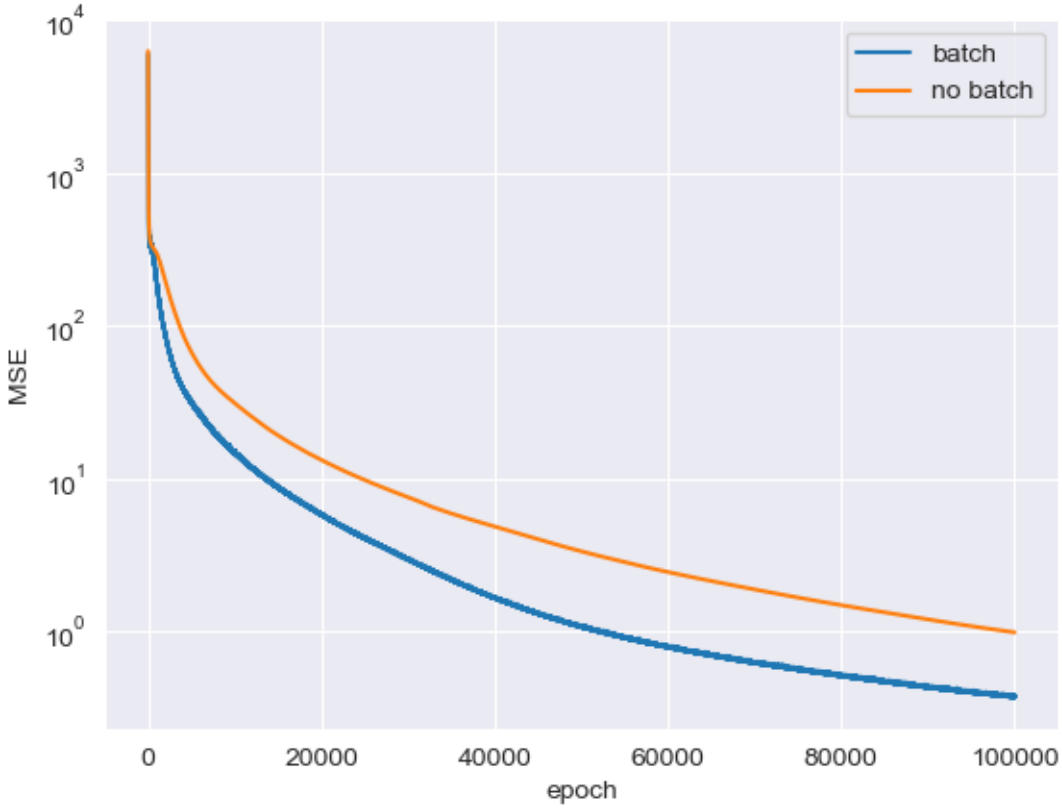
Bez batchów:

min	max	średnia
1.0131	6277.1801	16.4442
1.1414	6285.9833	17.2617
0.8493	6455.9342	18.0455
0.9963	6335.158	17.1660
0.9024	6424.6249	17.6423

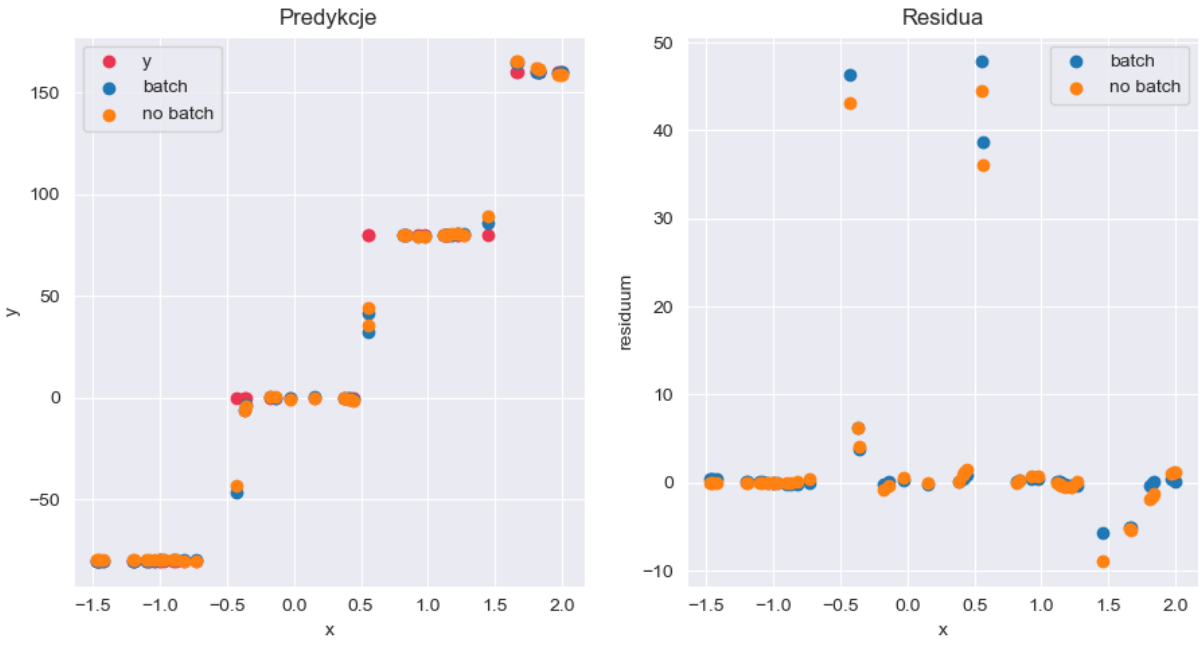
Z batchami:

min	max	średnia
0.3556	5880.0033	8.9520
0.2996	6074.8315	8.2123
0.2653	5970.1692	8.6196
0.3015	6154.9636	8.9634
0.349	6131.1497	9.1391

Po przyjrzeniu się wynikom można otrzymać podobne wnioski co dla danych square-simple. Liczby jasno wskazują na poprawienie modelu poprzez zastosowanie batchów. Ponownie możemy zobaczyć zbieżność mse dla obydwu modeli.



Dla danych testowych jednak lepiej poradził sobie model beż użycia batchów, jednak jest to tylko w punktach pomiędzy schodkami - te punkty są dość charakterystyczne i nie powinny ważyć na ocenie modeli.



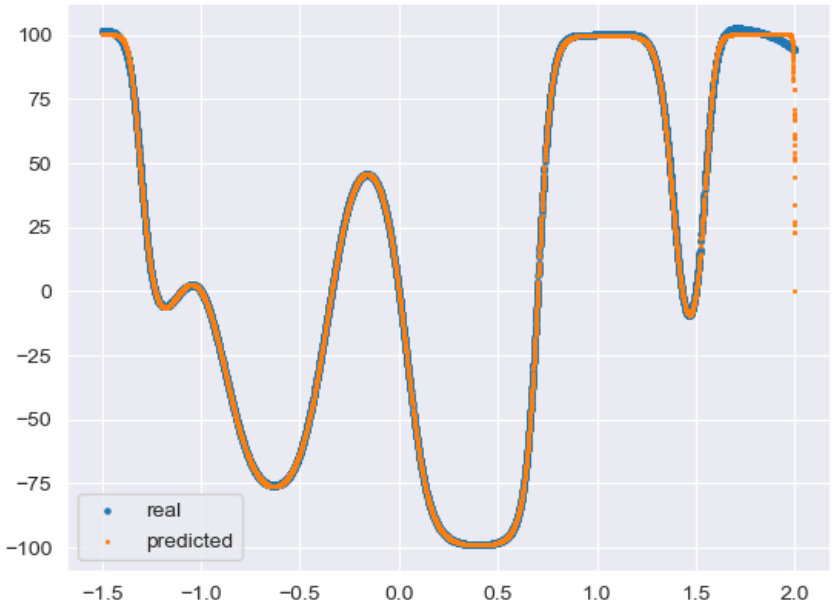
Multimodal-large

Można już wywnioskować, że batche rzeczywiście usprawniają działanie modelu. Dlatego tym razem przetestujemy model z wykorzystaniem batchów oraz zobaczymy jak model rzeczywiście przybliża dane punkty na wykresie.

Parametry:

- warstwy oraz liczba neuronów $50 \longrightarrow 1$
- eta = 0.01
- epoki – 100000
- batch – 32

Poniżej przedstawione jest porównanie rzeczywistych wartości oraz wartości, które przybliżył model. Wynika z tego, że modele rzeczywiście się uczą - jest to dobry wstęp do dalszych zadań.



4 Momentum vs RMSProp

Opis wykonywanych eksperymentów

Zadanie polegało na porównaniu dwóch nowych ulepszeń dla uczenia sieci: Momentum i RMSProp. Aby porównać te dwie metody wykonałam po 5 testów dla każdej sieci korzystającej z Momentum i RMSProp oraz dla wszystkich 3 typów danych: square-large, steps-large i multimodal-large. Dla każdego zestawu danych ustawione zostały odpowiednio dobrane parametry (metodą prób i błędów), jednak każdy z jednostkowych testów ma ten sam zestaw parametrów. Funkcją aktywacji dla wszystkich warstw ukrytych jest sigmoid oraz warstwie wyjściowa jest funkcją identycznościową. Natomiast wagi i bajasy były inicjowane metodą He. Podobnie jak w poprzednich testach.

Opis parametrów

- eta - krok uczenia
- momentum - współczynnik wygaszania momentu
- beta - współczynnik wygaszania

Square-large

Parametry:

- warstwy oraz liczba neuronów $50 \longrightarrow 1$
- eta = 0.01
- momentum = 0.9
- beta = 0.999
- epsilon = 10^{-8}
- epoki – 10000
- batch – 32

Wyniki:

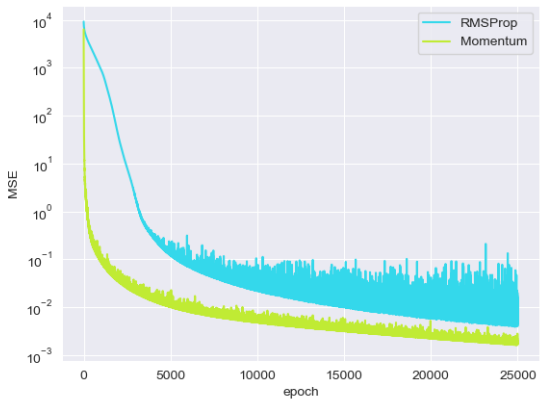
RMSProp:

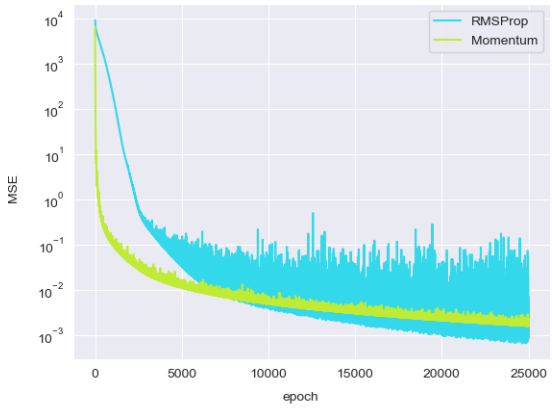
min	max	średnia
0.0027	9340.1912	194.5503
0.0092	9264.783	237.0785
0.0106	9265.1004	251.3468
0.0033	9277.908	205.9749
0.009	9260.8955	239.2131

Momentum:

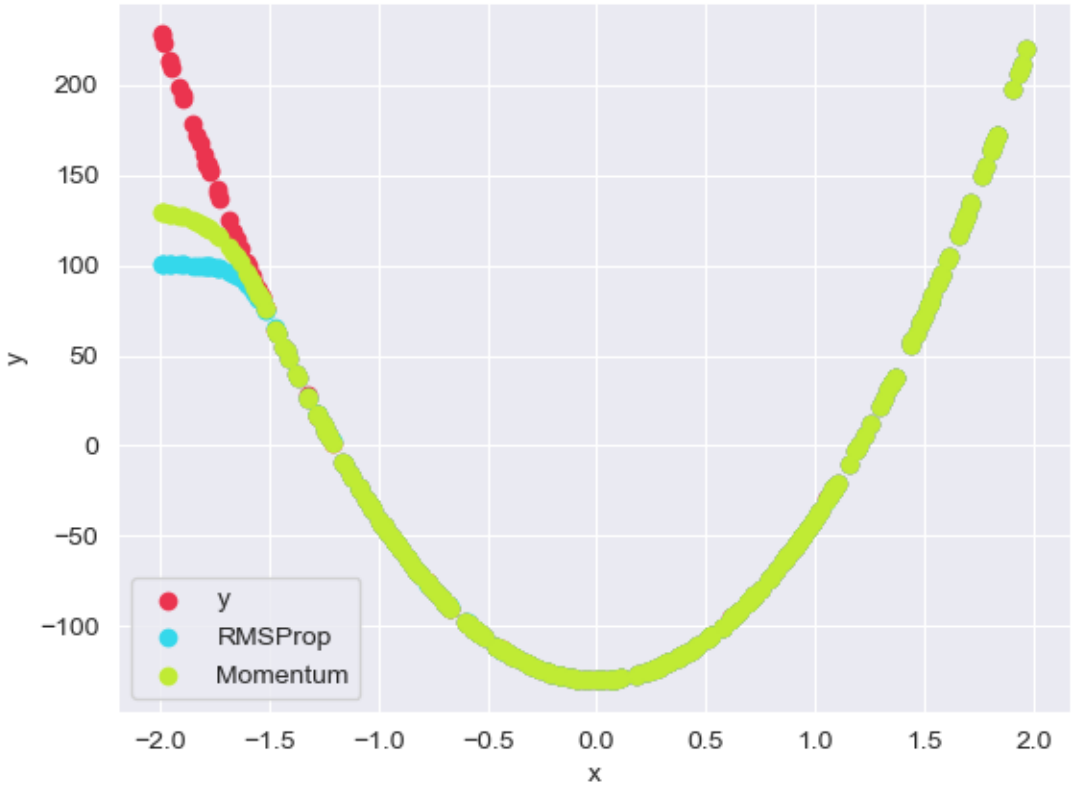
min	max	średnia
0.0041	6582.7716	2.0815
0.0049	6826.8005	2.5522
0.0038	6611.9139	2.2258
0.0052	6315.0216	2.1703
0.0042	6713.5468	2.2845

Widzimy, że ostateczne minimalne wartości były bardzo podobne dla obu metod. Średnia dla RMSProp jest zdecydowanie większa, co wynika z wyższych wartości maksymalnych, dlatego ważne w tym przypadku są dla nas tylko wartości minimalne, które nie wskazują nam na to, który model jest lepszy. Następnie wykonałam dodatkowe dwa testy na takich samych parametrach obydwu sieci korzystających z różnych metod, jednak tym razem liczba epok była zwiększona do 25 000. Poniżej przedstawione są wykresy wartości MSE dla porównywanych modeli.





Co widać dla obydwu eksperymentów to wartości MSE dla RMSProp są mniej „stabilne”. Nadal również widzimy, że nie jesteśmy jednoznacznie sprawdzić, która metoda jest lepsza. Dla tych samych modeli sprawdziłam jeszcze jak dokładne były predykcje w porównaniu z realną wartością dla danych testowych (korzystamy z najlepszego nauczonego modelu według MSE dla danych treningowych). Otrzymane wykresy były niemal identyczne dla obydwu eksperymentów, dlatego zamieszczam wykres jedynie dla jednego z nich.



Widzimy, że dla $x \in (-2, -1.5)$ zachodzi pewna anomalia, jednak wynika to z tego, że dane treningowe nie zawierały przykładów z takimi wartościami. W takim przypadku poradziła sobie nieznacznie lepiej metoda Momentum.

Steps-large

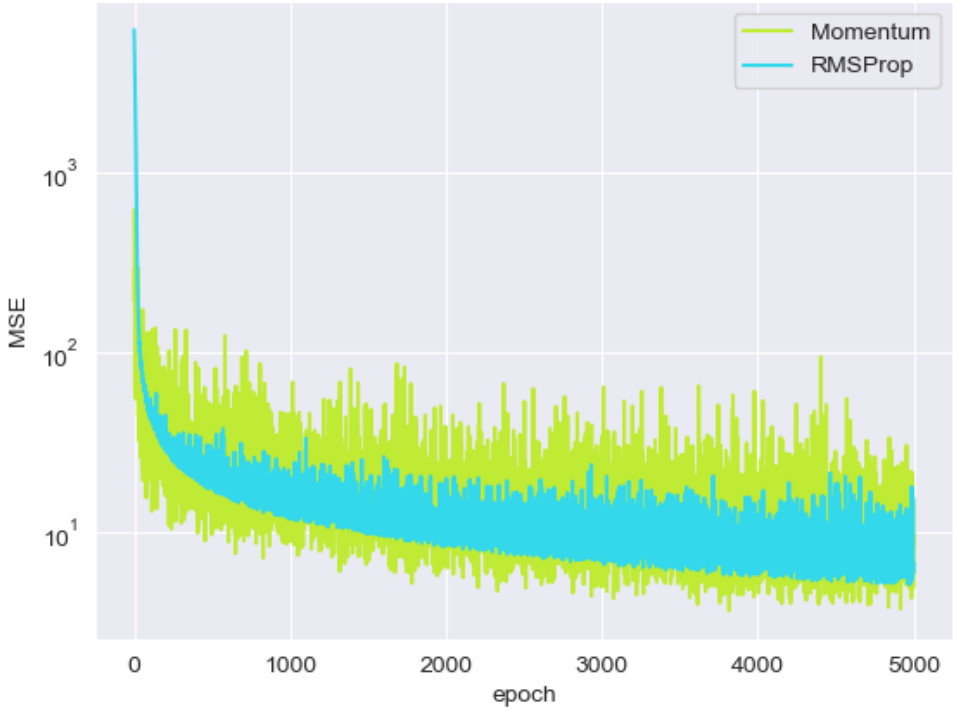
Parametry:

- warstwy oraz liczba neuronów $8 \longrightarrow 8 \longrightarrow 1$
- eta $= 0.01$
- momentum $= 0.5$
- beta $= 0.8$
- epsilon $= 10^{-8}$
- epoki – 5000
- batch – 64

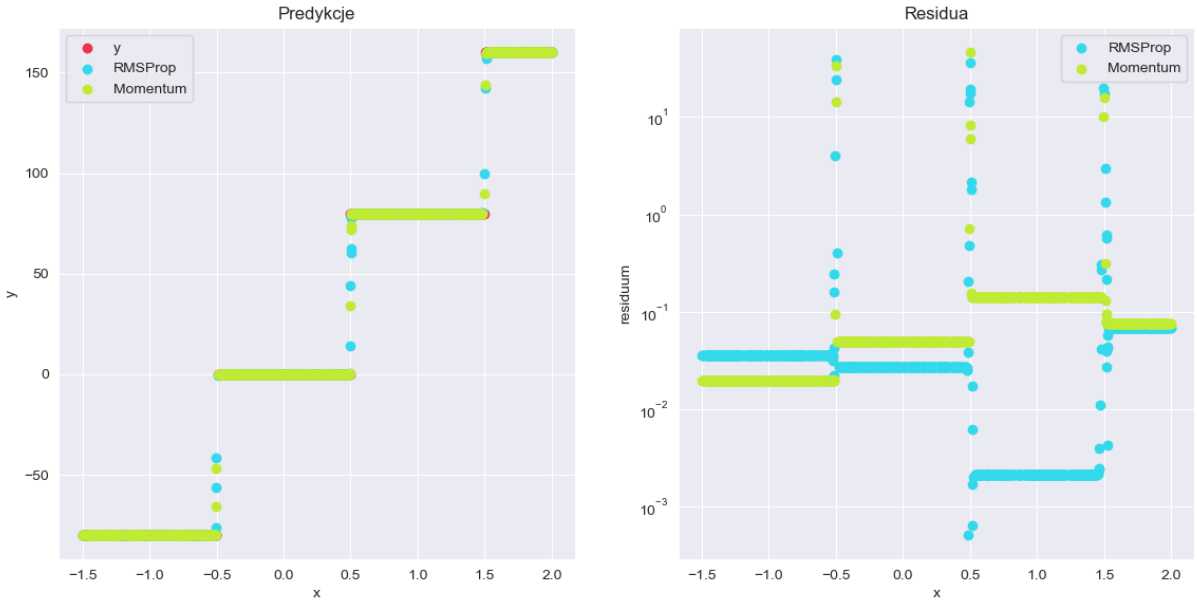
Wyniki:

RMSProp:			Momentum:		
min	max	średnia	min	max	średnia
4.9517	6303.6107	21.4114	3.3112	816.4302	18.1952
4.9619	6065.8332	21.5521	3.179	631.3268	17.6884
4.9569	6499.2434	21.827	3.1954	822.3158	18.079
4.9593	6119.9008	22.5754	3.677	620.5258	18.0257
4.9222	6292.2191	21.7706	3.5487	1251.0056	17.7263

Na podstawie wyników przedstawionych powyżej możemy po raz pierwszy stwierdzić, która z metod poradziła sobie lepiej. Widzimy, że mniejsze minimalne wartości MSE otrzymał model z użyciem metody Momentum. Wartości MSE jednak są bardzo niestabilne, co widać na poniższym wykresie.



W tym przypadku widzimy, że obydwie metody dają bardzo „niestabilne" wyniki MSE. Następnie sprawdziłam jak dobrze modele się uczą - pokazując to na wykresie dla danych testowych. Dodatkowo na drugim wykresie pokazane są wartości bezwzględne residuów w skali logarytmicznej.



Obie metody dają bardzo zadowalające wyniki dla danych testowych, jednak widać że dla wartości rzędu dziesiątek na drugim wykresie jest znacznie więcej punktów niebieskich co wpływa na mniejsza dokładność modelu z zastosowaną metoda RMSProp. Wyraźne błędy pojawiają się ponownie jak dla danych steps-small tylko pomiędzy schodkami, gdzie modele rzeczywiście mają problem z wyznaczeniem prawidłowej wartości.

Multimodal-large

Parametry:

- warstwy oraz liczba neuronów $50 \longrightarrow 1$
- eta = 0.1 (dla RMSProp), eta = 0.001 (dla Momentum)
- momentum = 0.8
- beta = 0.99
- epsilon = 10^{-6}
- epoki – 1000
- batch – 32

Wyniki:

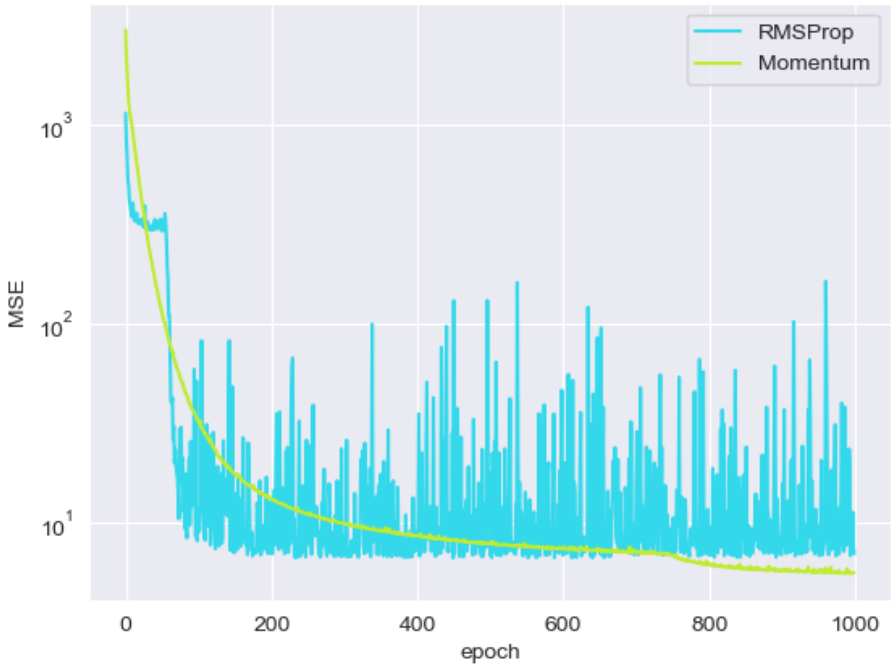
RMSProp:

min	max	średnia
7.5613	1176.9663	29.5096
5.4177	1199.3571	32.2966
3.8006	1176.8863	27.3341
8.1559	1261.3783	23.5566
3.872	1190.9768	36.5077

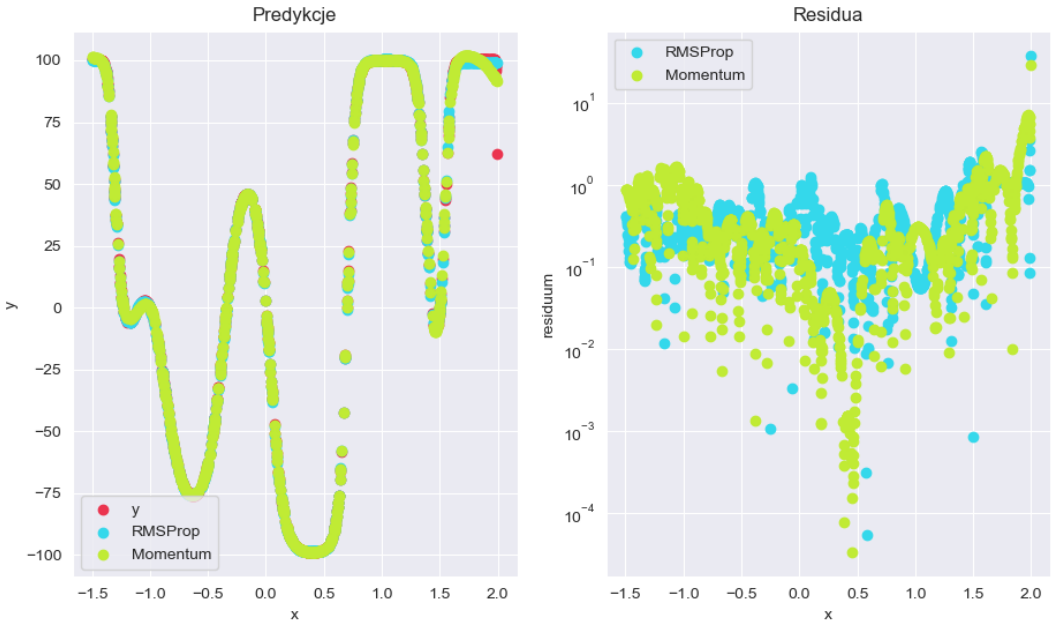
Momentum:

min	max	średnia
5.5928	2968.6577	42.8327
5.4183	2948.5266	35.7989
8.0303	2944.1537	40.3345
6.8298	3031.6506	39.488
6.795	3029.4242	38.5501

W tym przypadku, aby wyrównać szanse dla obydwu metod wartości kroku uczenia są różne. Wartości minimalne MSE są zbliżone, więc możemy jeszcze obejrzeć wykres pokazujący jak to MSE się zmieniało dla kolejnych epok.



Na tym wykresie bardzo ładnie widać, że na początku metoda RMSProp wpadła w minimum lokalne, natomiast metoda Momentum je ominęła - pokazuje to ich działanie. Dodatkowo tak jak poprzednio porównajmy wyniki dla danych testowych, co pokaże że nadal nie jesteśmy w stanie określić, która metoda lepiej wpłynęła na uczenie modelu - obie radzą sobie równie dobrze.



5 Zadania klasyfikacji

Kolejne zadanie dotyczy problemu klasyfikacji. Nowością w implementacji sieci jest funkcja softmax dla warstwy wyjściowej - do tej pory używana byłą funkcja identycznościowa. Naszym zadaniem było porównanie tych dwóch funkcji. W ramach eksperymentów wykonałam ponownie po 3 testy dla nowej i starej funkcji na warstwie wyjściowej na danych rings3-regular, easy i xor3. Ze względu na zadanie klasyfikacji pojawia się nowa miara pokazująca jak dobry jest model - f-score. Dla wszystkich sieci funkcją aktywacji dla wszystkich warstw ukrytych jest sigmoid oraz wagi i bajasy były inicjowane metodą He.

Rings3-regular

Parametry:

- warstwy oraz liczba neuronów 50 $\longrightarrow$ 50 $\longrightarrow$ 3
- eta = 0.005
- epoki – 10000
- batch – 128

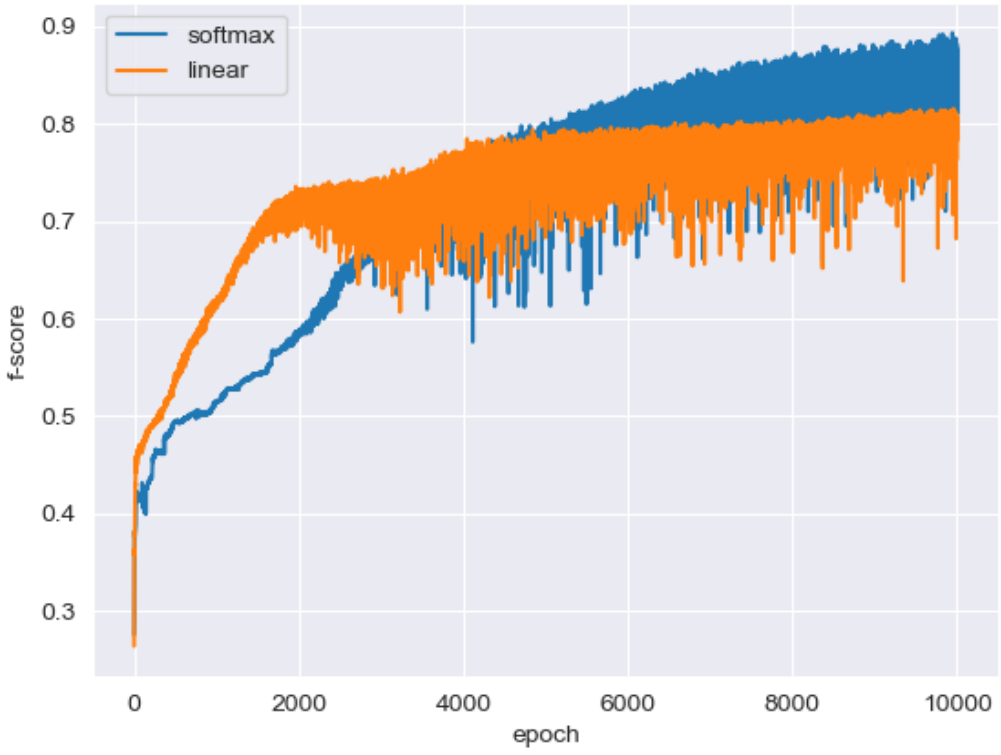
Wyniki:

Identyczność:

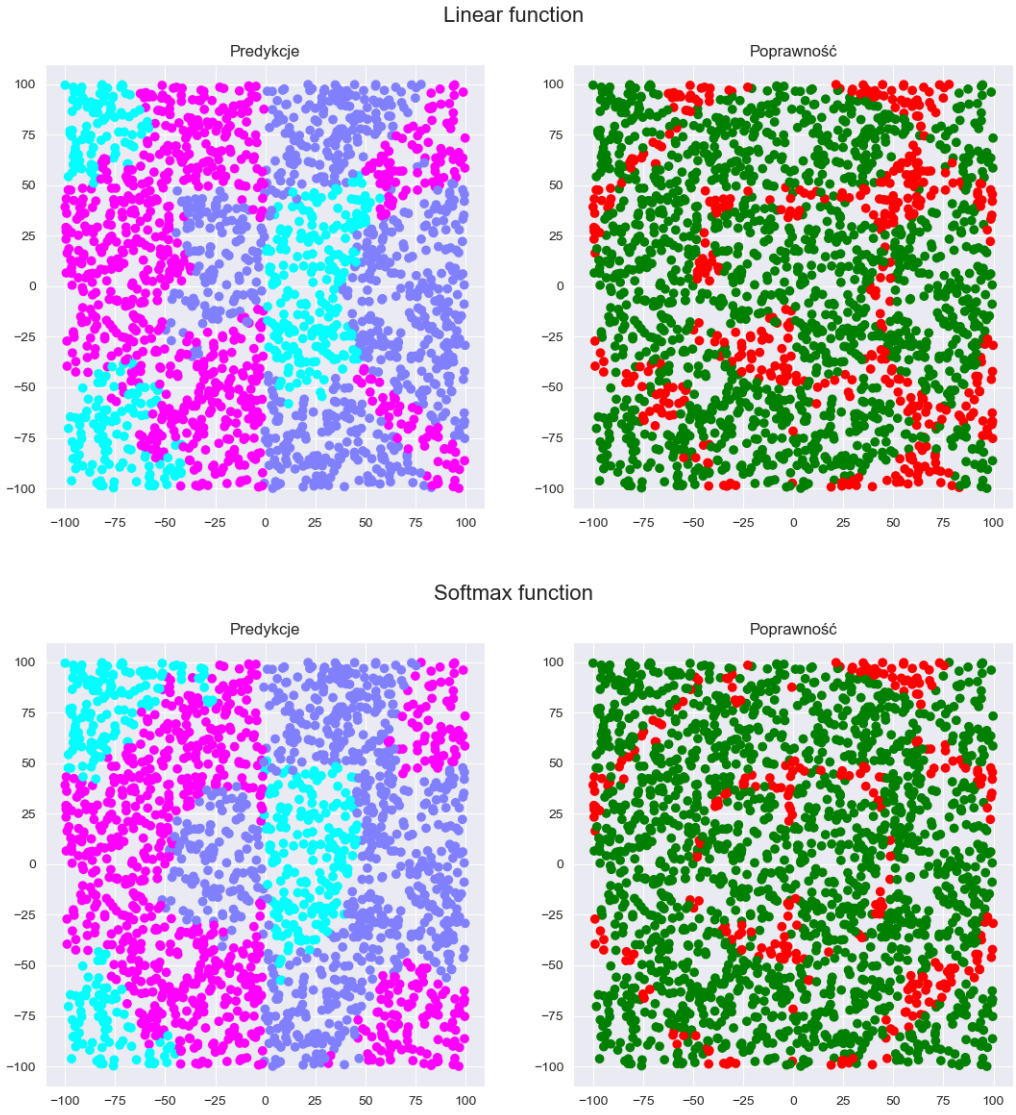
min	max	średnia
0.3351	0.8337	0.7302
0.197	0.827	0.6773
0.1801	0.8302	0.6872
0.2323	0.8288	0.7052
0.2641	0.8156	0.7301

Softmax:

min	max	średnia
0.2416	0.9178	0.772
0.1667	0.9186	0.7697
0.1667	0.9191	0.7605
0.1994	0.9001	0.6873
0.276	0.8933	0.7221



Porównując wartości f-score dążymy do jak najbliższej wartości do 1. Model korzystający z funkcji softmax osiąga maksymalne wartości większe niż model z funkcją identycznościową. Dla modeli, które osiągnęły wartości z ostatniego wiersza tabelek możemy zobaczyć jak wartości f-score zmieniały się na przestrzeni epok. Następnie zweryfikujmy jak dobrze modele poradziły sobie rzeczywiście z klasyfikacją. Możemy to zobaczyć na poniższych wykresach.



Po lewej stronie mamy predykcje obydwu modeli, natomiast po prawej na czerwono mamy zaznaczone punkty, które zostały źle zakwalifikowane a na zielono - dobrze zakwalifikowane. Dla modelu z funkcją identycznościową jest zauważalnie więcej czerwonych punktów, niż w drugim modelu.

Easy

Parametry:

- warstwy oraz liczba neuronów $30 \longrightarrow 30 \longrightarrow 2$
- eta $= 0.001$
- epoki – 3000
- batch – 128

Wyniki:

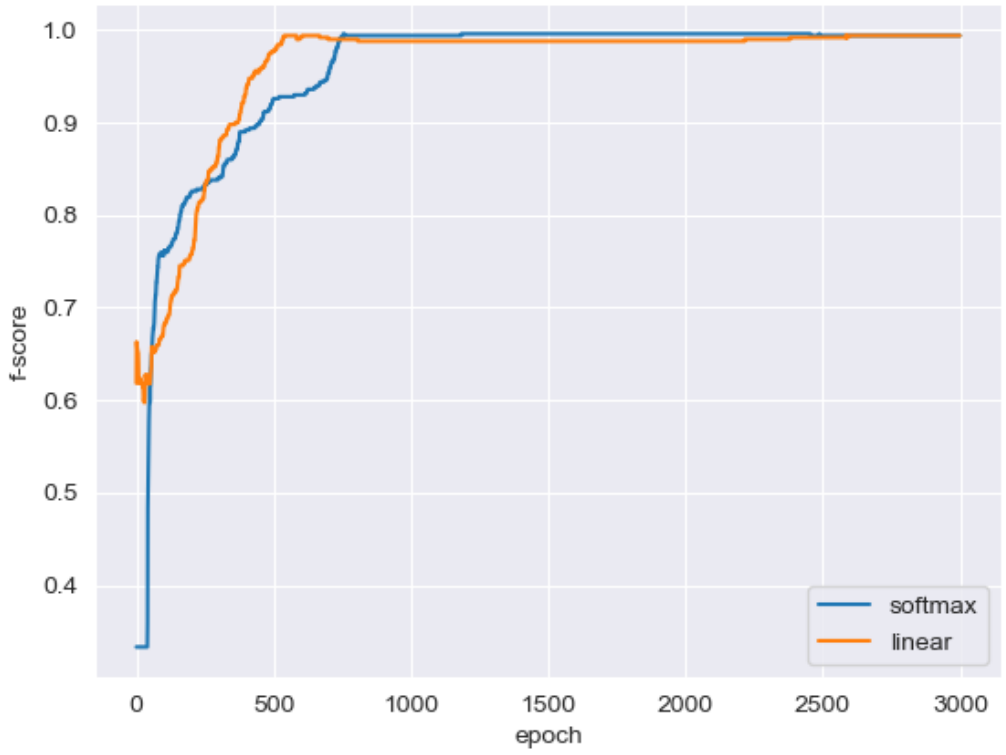
Identyczność:

min	max	średnia
0.3595	1.0	0.9812
0.3075	0.998	0.9352
0.1883	0.998	0.9443
0.2997	1.0	0.9624
0.5974	0.994	0.9597

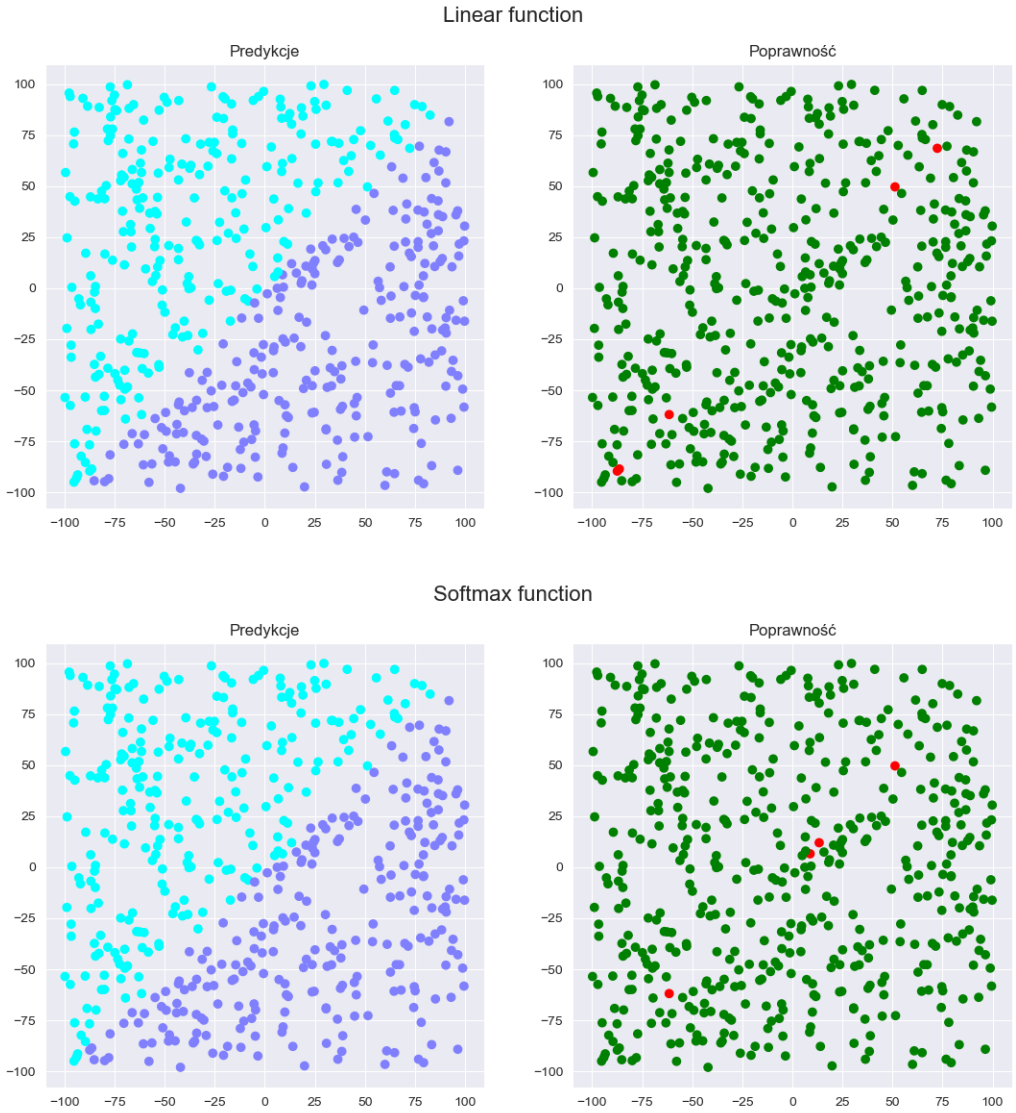
Softmax:

min	max	średnia
0.4261	0.998	0.9856
0.3333	1.0	0.9613
0.3333	0.992	0.9362
0.3333	1.0	0.9687
0.3333	0.996	0.9559

Tym razem mamy do czynienia z bardzo prostymi danymi - obszar jest podzielony prosta $y = x$. Widzimy, że obydwa modele uzyskały tak samo dobre wyniki. Możemy dodatkowo sprawdzić jak wyglądają wartości f-score dla kolejnych epok.



Wykres tylko potwierdza niemal identycznie dobre działanie obydwu modeli. Tak jak powyżej porównajmy rzeczywista klasyfikację na danych testowych



Jedyne co możemy zauważy, to że mamy minimalna liczbę czerwonych punktów dla obydwu modeli.

Xor

Parametry:

- warstwy oraz liczba neuronów $30 \longrightarrow 50 \longrightarrow 30 \longrightarrow 2$
- eta $= 0.008$
- epoki – 70000
- batch – 128

Wyniki:

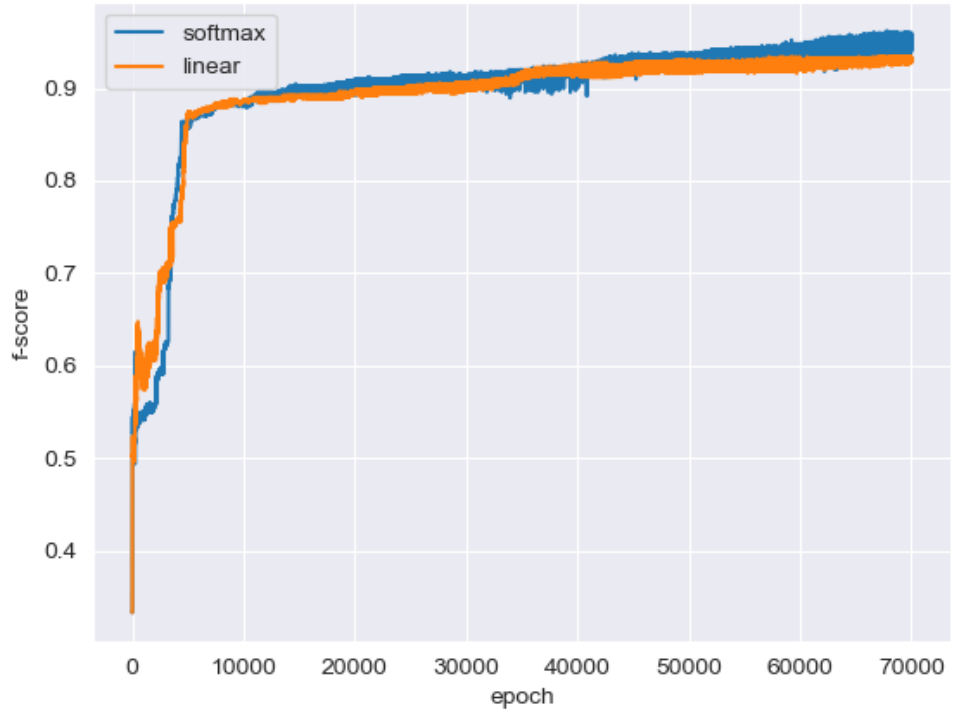
Identyczność:

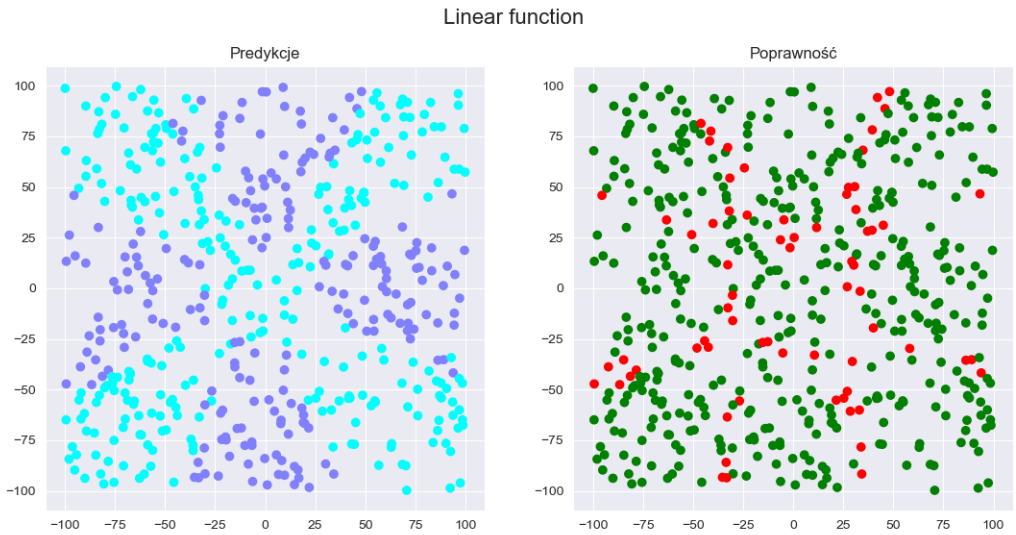
min	max	średnia
0.3333	0.95	0.9004
0.3052	0.91	0.8778
0.3333	0.934	0.8931

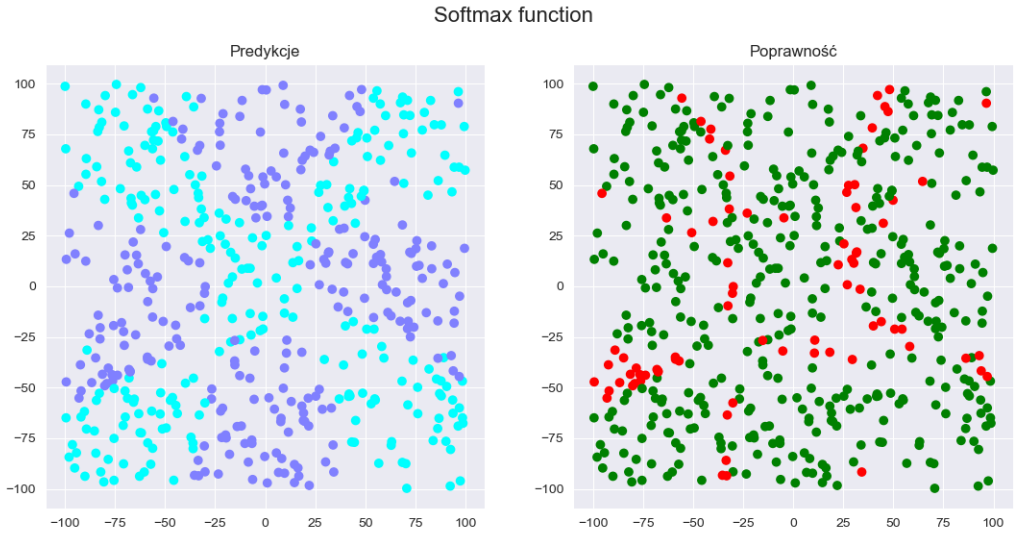
Softmax:

min	max	średnia
0.3316	0.992	0.9309
0.3333	0.972	0.916
0.3333	0.962	0.8983

Dla danych xor jesteśmy w stanie zauważyć nieznaczną przewagę funkcji softmax. Dalej porównajmy wartości f-score dla kolejnych epok oraz klasyfikację punktów dla danych testowych.







Obydwa modele poradziły sobie podobnie. Stworzyły pasy, tak jakby promyki wychodzące od środka, nie udało im się za dobrze odtworzyć klatek, które można było zobaczyć w oryginalnych danych

6 Porównanie funkcji aktywacji

Do tej pory we wszystkich sieciach używaną funkcją aktywacji był sigmoid, dlatego teraz przyszedł czas na porównanie innych opcji:

- identycznościowa, która do tej pory była stosowana na ostatniej warstwie
- tangens hiperboliczny - tanh
- rectified linear unit - ReLU

Pierwszym etapem zadania było porównanie powyższych funkcji aktywacji dla 3 różnych architektur sieci z 1, 2, i 3 warstwami ukrytymi dla już wcześniej używanych danych - multimodal-large. Następnie dla 2 najlepszych kombinacji ilości warstw i funkcji aktywacji mieliśmy przeprowadzić testy dla kolejnych danych dotyczących regresji: steps-large oraz klasyfikacji: rings5-regular i rings3-regular. W tym wypadku wszystkie sieci używały funkcji liniowej na ostatniej warstwie, reszta parametrów była odpowiednio dobrana do danego eksperymentu.

Sigmoid vs linear vs tanh vs ReLU

Przeprowadziłam serię 12 testów - dla każdej funkcji aktywacji z następującymi architekturami sieci:

- $30 \longrightarrow 1$
- $10 \longrightarrow 15 \longrightarrow 1$
- $10 \longrightarrow 15 \longrightarrow 10 \longrightarrow 1$

Dla każdego eksperymentu odpowiednio dobrałam krok uczenia i momentum. Zdecydowałam się na momentum ze względu na to, że testy z 4 rozdziału wskazywały na poprawę w uczeniu się sieci.

Wyniki:

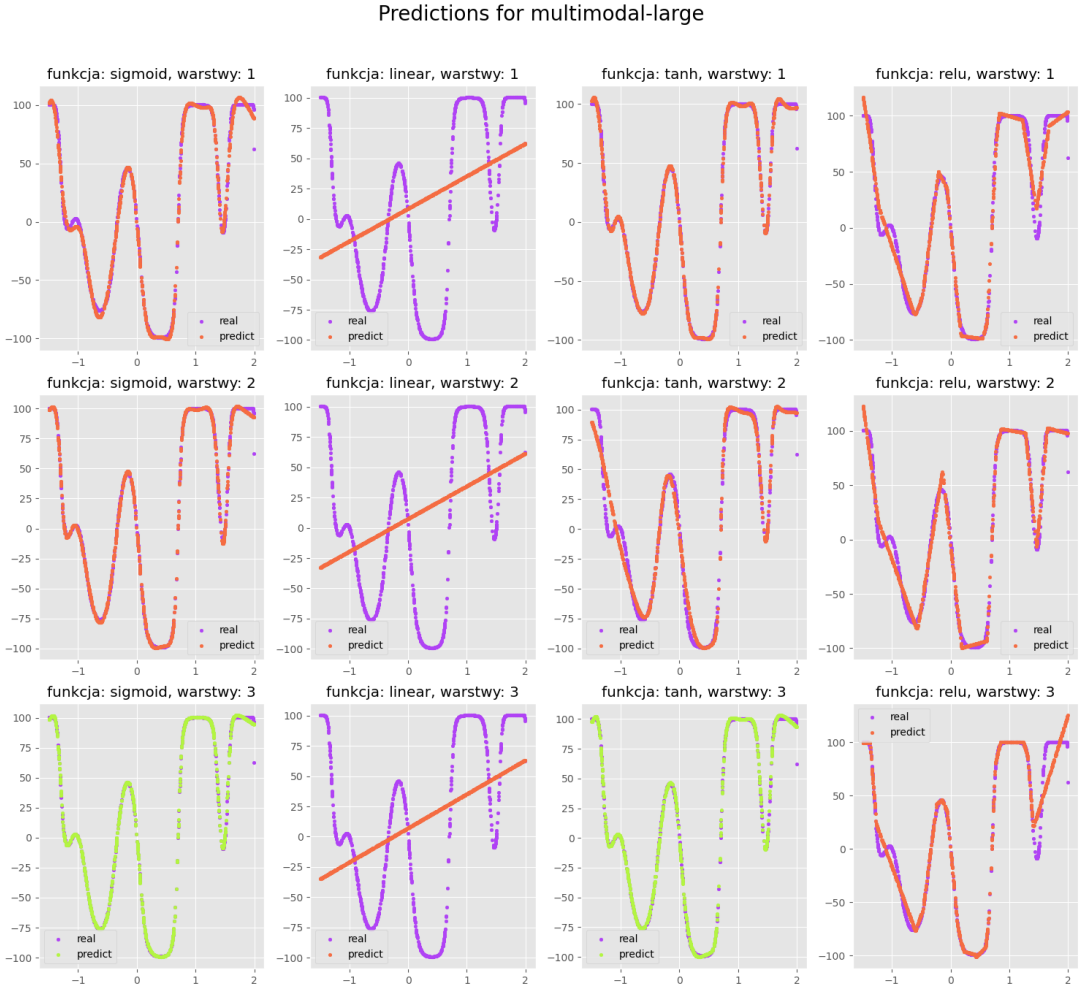
W poniższej tabeli przedstawione są minimalne wartości MSE dla danych konfiguracji. Każdy eksperyment trwał 4000 epok dla optymalnego porównania.

warstwy\funkcja	sigmoid	linear	tanh	ReLU
1	32.7870	4398.2223	9.2957	137.5037
2	7.2276	4398.2246	9.5022	212.8480
3	6.3478	4398.2222	4.2906	8.3885

Najlepsze wynika otrzymały sieci o 3 warstwach ukrytych dla funkcji sigmoid i tanh. Równie dobrze wypadły te same funkcje dla 2 warstw ukrytych oraz funkcja ReLU dla 3 warstw ukrytych. Te wyniki posłużył do dalszej części zadania. W ramach realizacji raportu przeprowadziłam dodatkowy zestaw testów, których wyniki przedstawione są poniżej.

warstwy\funkcja	sigmoid	linear	tanh	ReLU
1	18.8298	4398.2225	10.992	101.1104
2	7.6706	4398.2233	9.6158	65.8306
3	6.0237	4398.2228	6.2828	135.6678

Dalej możemy na wykresie zobaczyć jak te przybliżenia wyglądają dla danych testowych. Na zielono zostały zaznaczone konfiguracje sieci, które osiągnęły najniższe minimalne MSE dla danych treningowych.



Regresja

W ramach zbadania skuteczności wcześniej wytypowanych architektur użyjemy danych steps-large. Dla każdej architektury zostało przeprowadzonych 5 eksperymentów dla poniżej przedstawionych parametrów.

Parametry:

- warstwy oraz liczba neuronów $10 \longrightarrow 15 \longrightarrow 10 \longrightarrow 1$
- funkcja aktywacji - sigmoid (tanh)
- eta = 0.05 (0.01)
- momentum = 0.8 (0.9)
- epoki – 4000
- batch – 64

Wyniki:

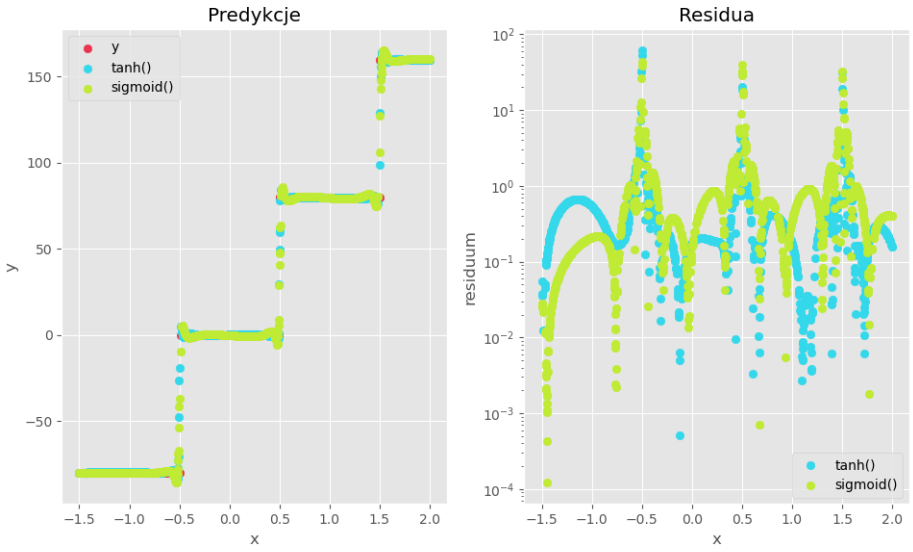
Sigmoid:

min	max	średnia
16.8868	543.8644	83.6047
14.0937	583.1361	57.5125
14.231	616.8484	74.1838
16.2832	620.0788	112.037
14.0181	649.0395	70.4551

Tanh:

min	max	średnia
8.4509	420.2773	20.2213
14.9252	1094.6479	39.9172
7.8658	429.647	20.8912
7.7305	451.9089	19.3481
8.2743	482.2446	21.5789

Dodatkowo porównajmy wyniki dla ostatniej trenowanej sieci dla każdej funkcji na danych testowych. Wykres po prawej stronie najlepiej pokazuje różnicę między obydwiema zastosowanymi funkcjami aktywacji, zauważmy że oś y jest w skali logarytmicznej. Wykresy oraz wyniki wskazują na niewielką przewagę funkcji tanh.



Klasyfikacja

Ring5-regular

Ponownie dla każdej architektury zostało przeprowadzonych 5 eksperymentów dla podanych niżej parametrów.

Parametry:

- warstwy oraz liczba neuronów $10 \longrightarrow 15 \longrightarrow 10 \longrightarrow 1$
- funkcja aktywacji - sigmoid (tanh)
- eta = 0.05 (0.006)
- momentum = 0.8 (0.9)
- epoki – 4000
- batch – 64

Wyniki:

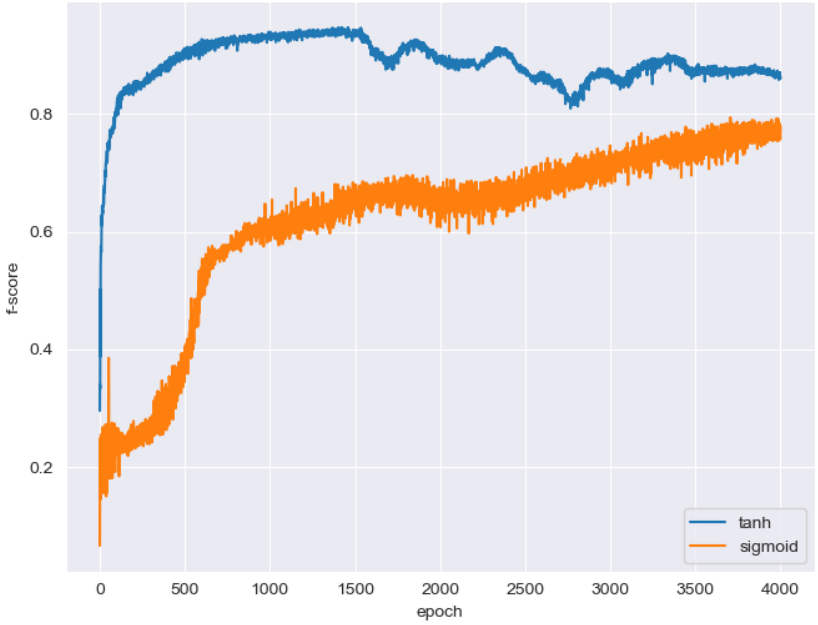
Sigmoid:

min	max	średnia
0.0667	0.8017	0.6235
0.0851	0.8747	0.6501
0.1024	0.8577	0.6623
0.0714	0.8634	0.6362
0.1378	0.8704	0.632

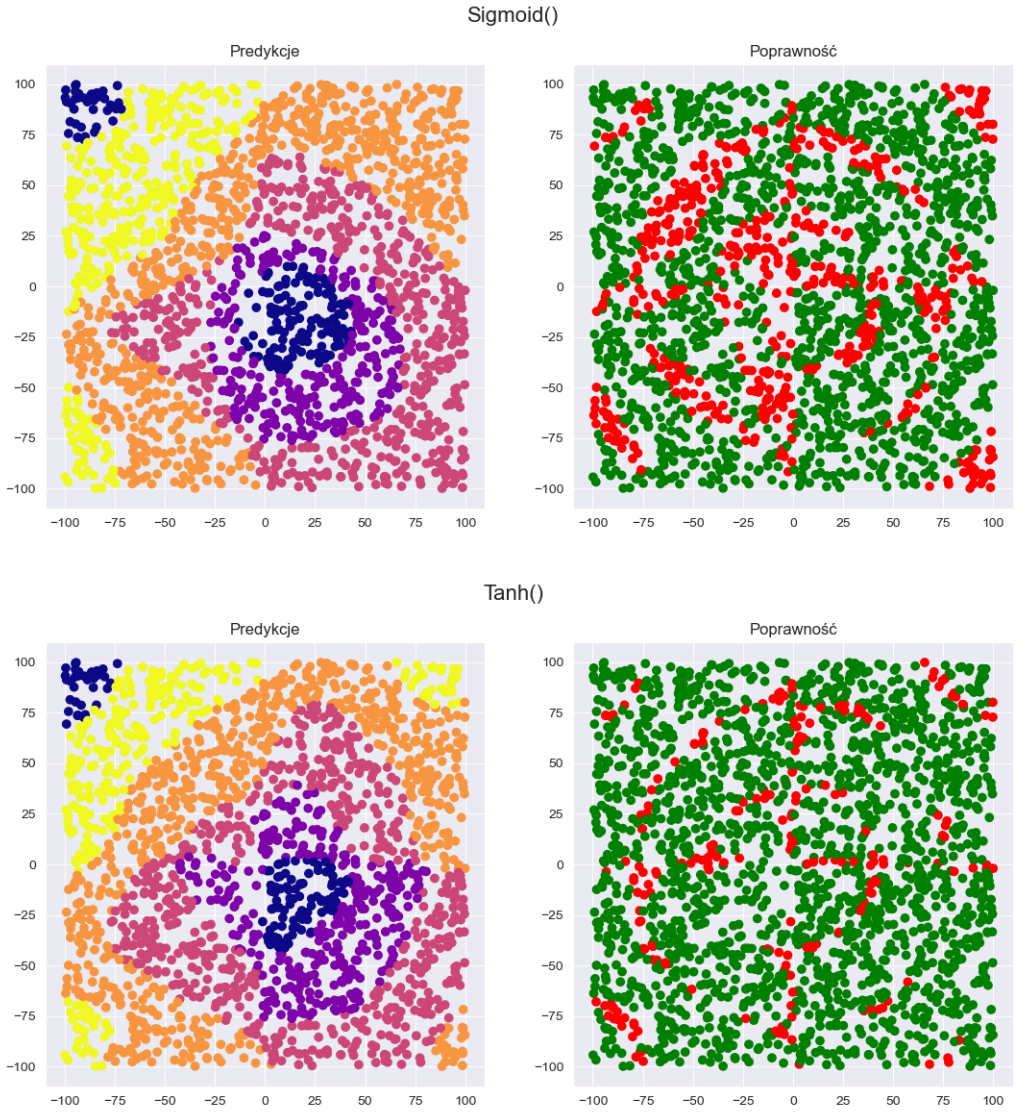
Tanh:

min	max	średnia
0.2147	0.9688	0.9289
0.1089	0.98	0.9439
0.3794	0.9542	0.8772
0.3152	0.9663	0.8864
0.1669	0.951	0.8115

Podobnie tak jak poprzednio sieć z funkcją aktywacji w postaci tanh uczy się lepiej - osiąga średnio lepsze maksymalne wartości f-score. Na poniższym wykresie możemy dodatkowo zobaczyć jak zmienia się wartość f-score dla kolejnych epok.



Dalej sprawdźmy jak nauczone sieci poradzą sobie z klasyfikacja na danych testowych. Tak jak można się było tego spodziewać sieć z funkcją aktywacji tanh daje lepsze wyniki. Widać to bardzo dobrze po ilości czerwonych puntów na prawych wykresach (punkty, które zostały źle zakwalifikowane).



Ring3-regular

Dla każdej sieci z różnymi funkcjami aktywacji zostało przeprowadzonych 5 eksperymentów dla podanych niżej parametrów.

Parametry:

- warstwy oraz liczba neuronów $10 \longrightarrow 15 \longrightarrow 10 \longrightarrow 1$
- funkcja aktywacji - sigmoid (tanh)
- eta = 0.01 (0.001)
- momentum = 0.8 (0.9)

- epoki – 4000
- batch – 64

Wyniki:

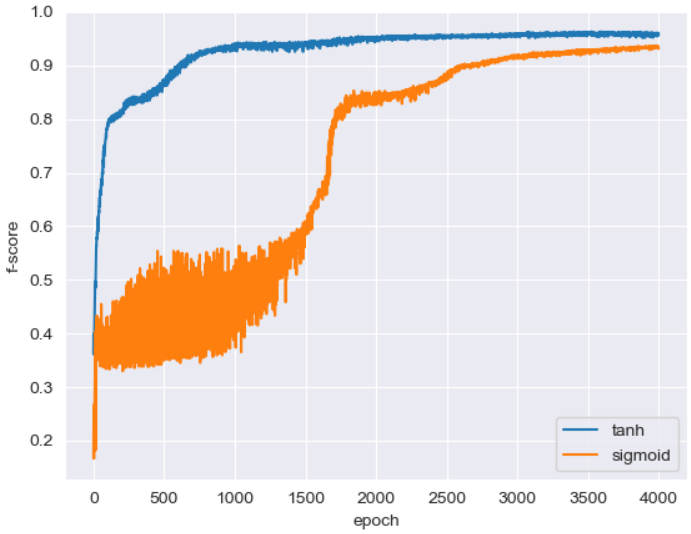
Sigmoid:

min	max	średnia
0.1667	0.9208	0.7145
0.2842	0.9213	0.829
0.1667	0.9434	0.7966
0.1821	0.9344	0.7903
0.1667	0.9367	0.7137

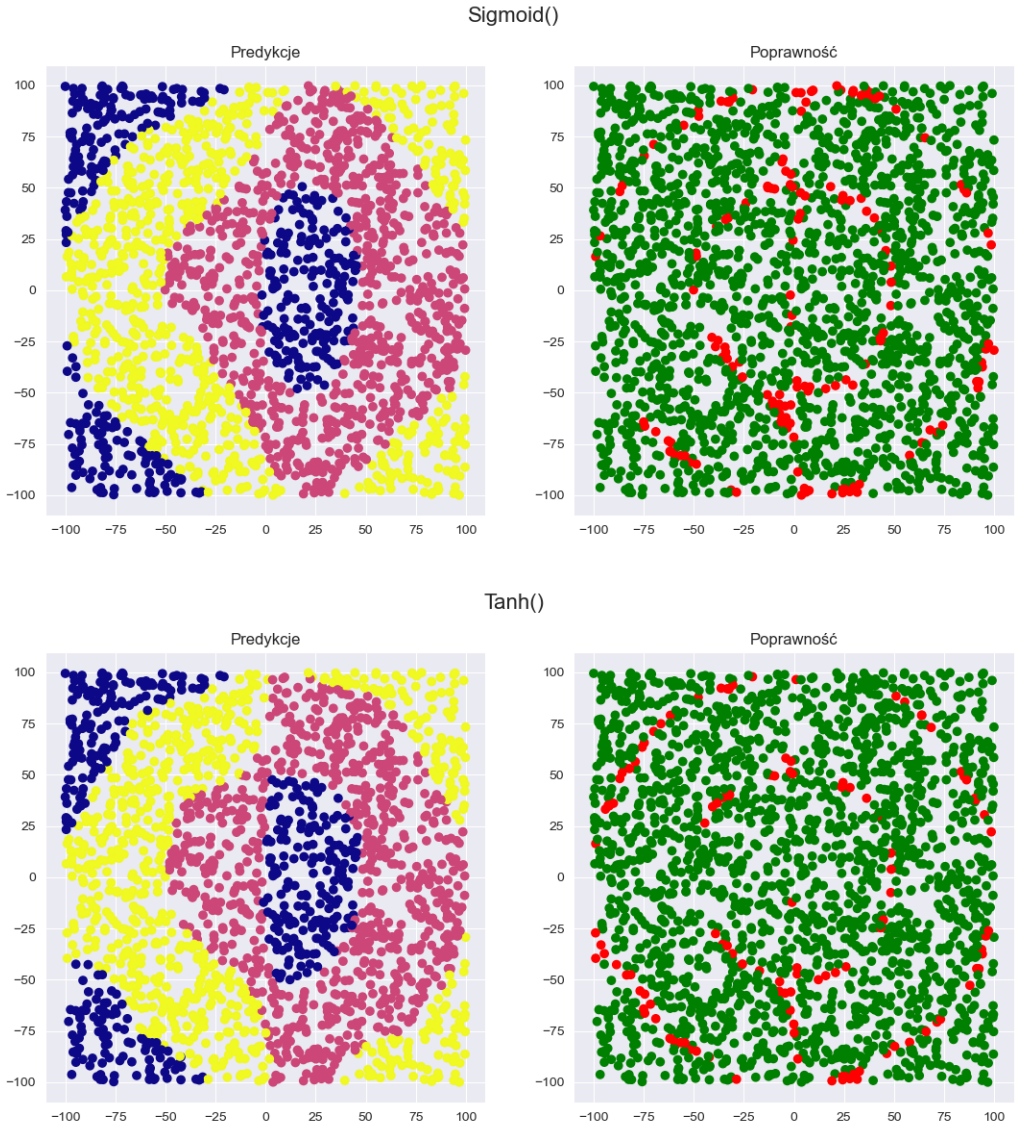
Tanh:

min	max	średnia
0.4101	0.9753	0.9437
0.2511	0.9607	0.9268
0.175	0.9653	0.916
0.2207	0.962	0.9106
0.3614	0.9633	0.9269

Tym razem nie widać dużej różnicy w wynikach pomiędzy sieciami. Obie osiągnęły bardzo zadowalające wyniki. Poniżej dodatkowo przedstawiony jest wykres wartości f-score dla kolejnych epok.



Natomiast na poniższych wykresach tak jak wcześniej naocznie możemy sprawdzić jak modele poradziły sobie z zadaniem klasyfikacji dla danych testowych. Mała liczba czerwonych punktów wskazuje na dużą dokładność modeli.



7 Zjawisko przeuczenia i regularyzacja

Ostatnią częścią projektu było zaimplementowanie mechanizmu zatrzymania w momencie wzrostu błędu na zbiorze walidacyjnym oraz mechanizmu regularyzacji. Następnie zdecydowałam się na porównanie 3 sieci korzystających z kolejno: mechanizmu stopującego, mechanizmu regularyzacji oraz sieci bez dodatkowych funkcji. Testy zostały wykonane dla danych: multimodal-sparse, rings5-sparse, rings3-balance i xor3-balance.

Mechanizm stopujący

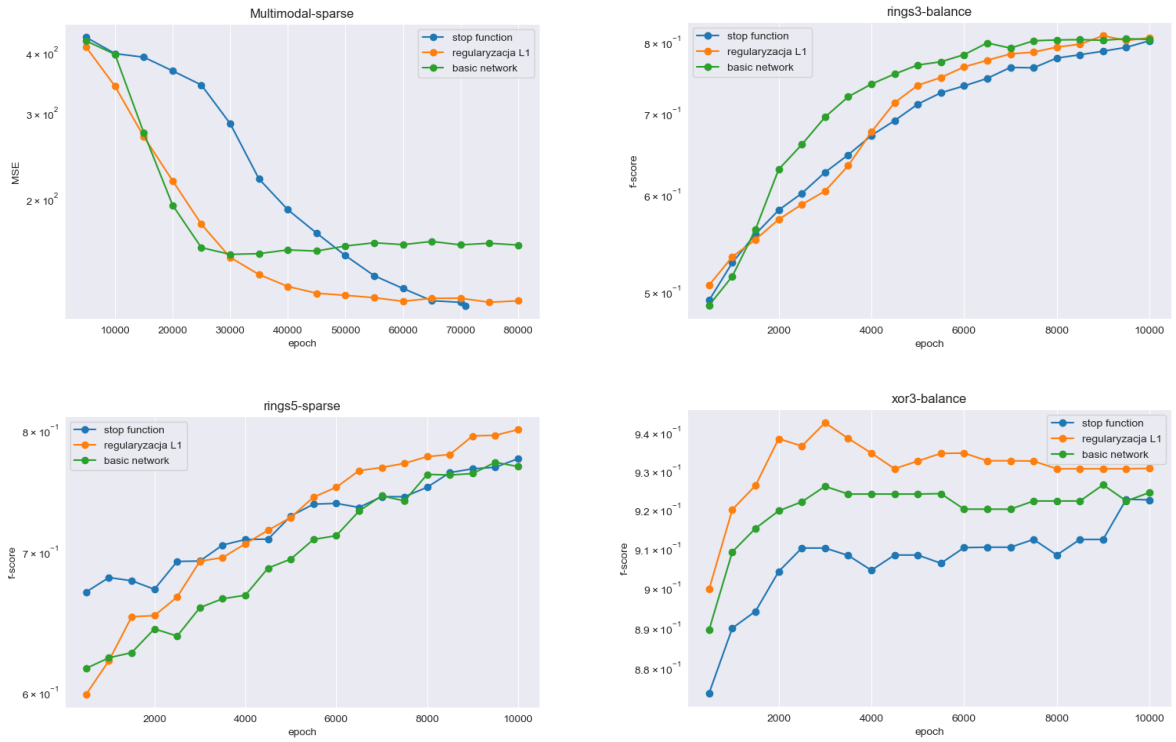
Jest to funkcja, która sprawdza czy obecna wartość błędu pomnożona przez stałą jest wyższa niż wartość dla epoki np. 1000 epok temu. Przetrzymywana jest liczba takich wydarzeń i jeśli ten licznik przekroczy pewną ustaloną wartość to uczenie sieci jest zatrzymywane.

Mechanizm regularyzacji

Na wykładzie poznaliśmy kilka mechanizmów regularyzacji: regularyzacja L1, regularyzacja L2 i dropout. Ja zdecydowałam się na zastosowanie regularyzacji L1, która polega na dodawaniu kar za zbyt duże wagi.

Wyniki

Na poniższych wykresach przedstawione zostały wyniki sieci dla wyżej opisanych architektur. Łatwo wywnioskować, że regularyzacja L1 zdecydowanie wypadła najlepiej, natomiast zadziałanie mechanizmu stopującego możemy zauważyć tylko dla danych multimodal-large (niebieski wykres ma mniej epok).



(*) Dla danych multimodal-large zdecydowałam się na zastosowanie innego kroku uczenia dla sieci bez ulepszeń, żeby dojść do momentu, w którym sieć zaczęła się overfitować (MSE na zbiorze treningowym malało a na zbiorze testowym - rosło).

8 Podsumowanie

Projekt skupiał się na pogłębionym badaniu i praktycznym opanowaniu sieci neuronowych. Celem było zrozumienie i analiza wpływu różnych hiperparametrów na proces uczenia oraz empiryczna weryfikacja teoretycznych założeń sieci neuronowych typu feedforward. W ramach projektu osiągnięto zakładany cel, którym było stworzenie efektywnego narzędzia analitycznego w postaci sieci neuronowych. Eksperymenty obejmowały praktyczne zastosowania różnorodnych architektur sieci, funkcji aktywacji, i technik uczenia. Projekt również skupiał się na identyfikacji i przeciwdziałaniu zjawisku przeuczenia, co miało na celu rozwinąć intuicję i umiejętności niezbędne do efektywnego stosowania sieci neuronowych w praktyce.