

Sprawozdanie 1: Sieci neuronowe

Autor:

Mateusz Stączek

Kwiecień, 2022

Nr indeksu:

305757

Temat Laboratoriów

Własna implementacja prostej sieci neuronowej w postaci MLP - perceptronu wielowarstwowego. Celem laboratoriów jest uzyskanie dobrych wyników na prostych problemach regresji i klasyfikacji oraz wizualizacja procesu uczenia oraz jego efektów.

Wykonana praca i wyniki eksperymentów

W trakcie każdego z laboratoriów dodawane były do sieci nowe funkcjonalności (np. moment, nowe funkcje aktywacji, crossentropy loss). Po każdym treningu wizualizowane były zarówno loss (MSE lub CrossEntropy - w zależności od używanego loss function), jak i f1 macro (wyłącznie przy klasyfikacji). Dodatkowo prezentowane były rezultaty przepuszczenia wszystkich obserwacji ze zbioru treningowego i testowego po zakończeniu treningu wraz z porównaniem z prawdziwymi wartościami.

Laboratorium 1

Wagi połączeń ręcznie dobrane (Geogebra) i przypisane odpowiednim parametrom sieci pozwoliły na dość dokładne odwzorowanie paraboli (zbior ‘square simple’) oraz funkcji schodkowej (zbior ‘steps large’) przy wykorzystaniu już 5 neuronów w 1 hidden layer. Rezultaty poniżej:

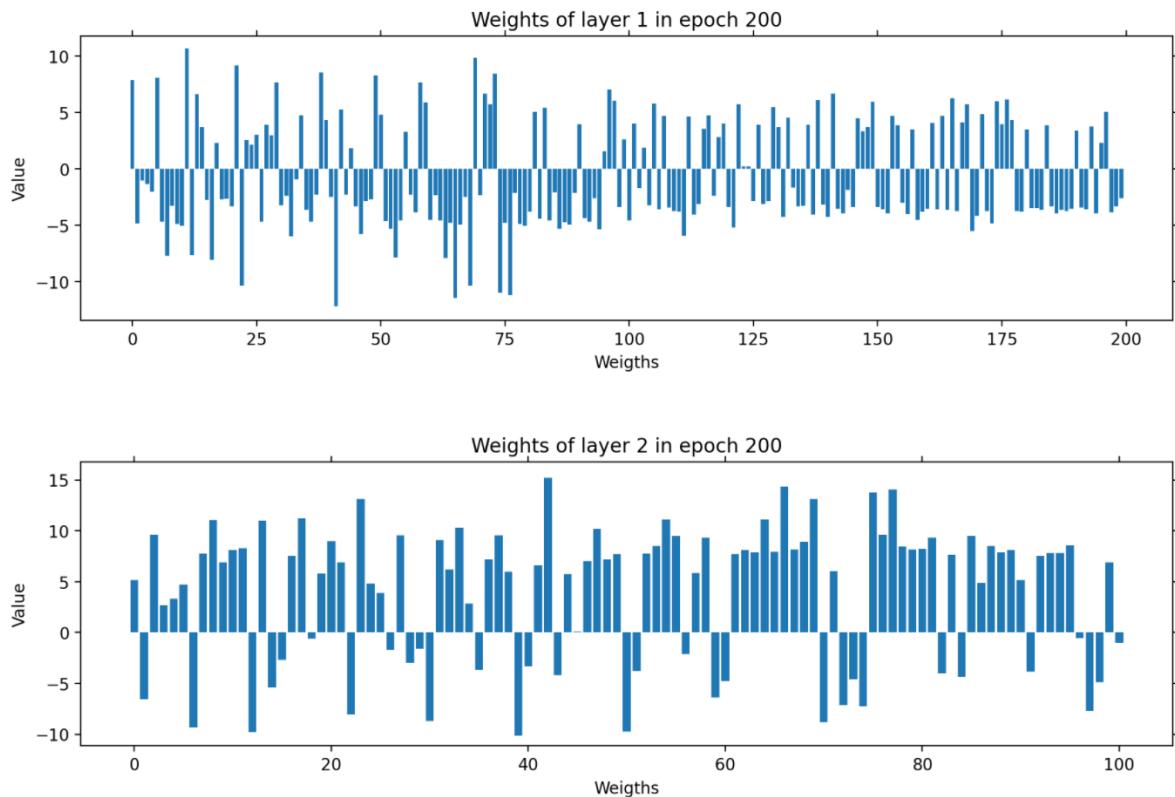
Dane	MSE train	MSE test
Square simple dataset	2.46	2.26
Steps large dataset	4.14	3.58

Laboratorium 2 i 3

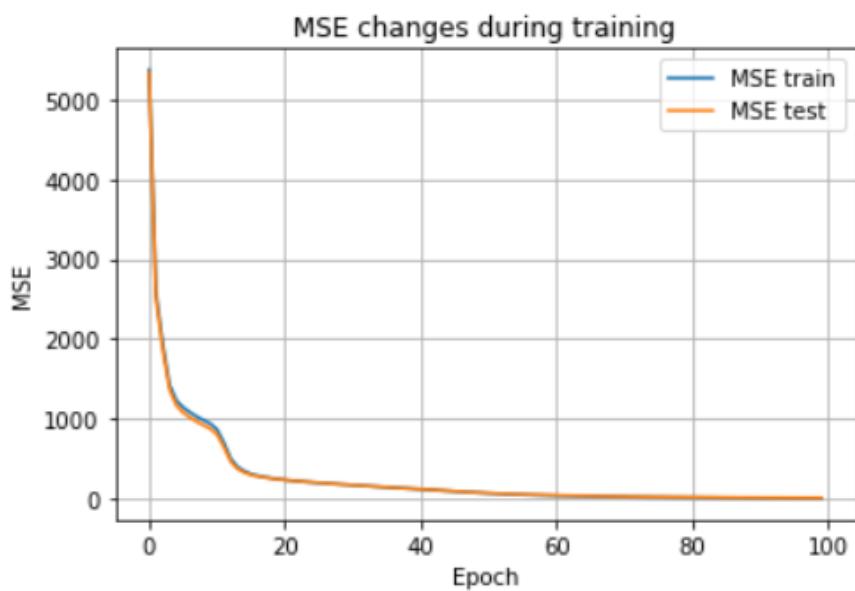
Dodanie backpropagacji pozwoliło na automatyczny wybór wag w sieci. Sprawdzone zostały następujące zbiory z wynikami przedstawionymi w tabeli poniżej:

Dane	MSE train	MSE test
Square simple	2.005	2.026
Steps small	0.913	97.558
Multimodal large	9.415	14.165

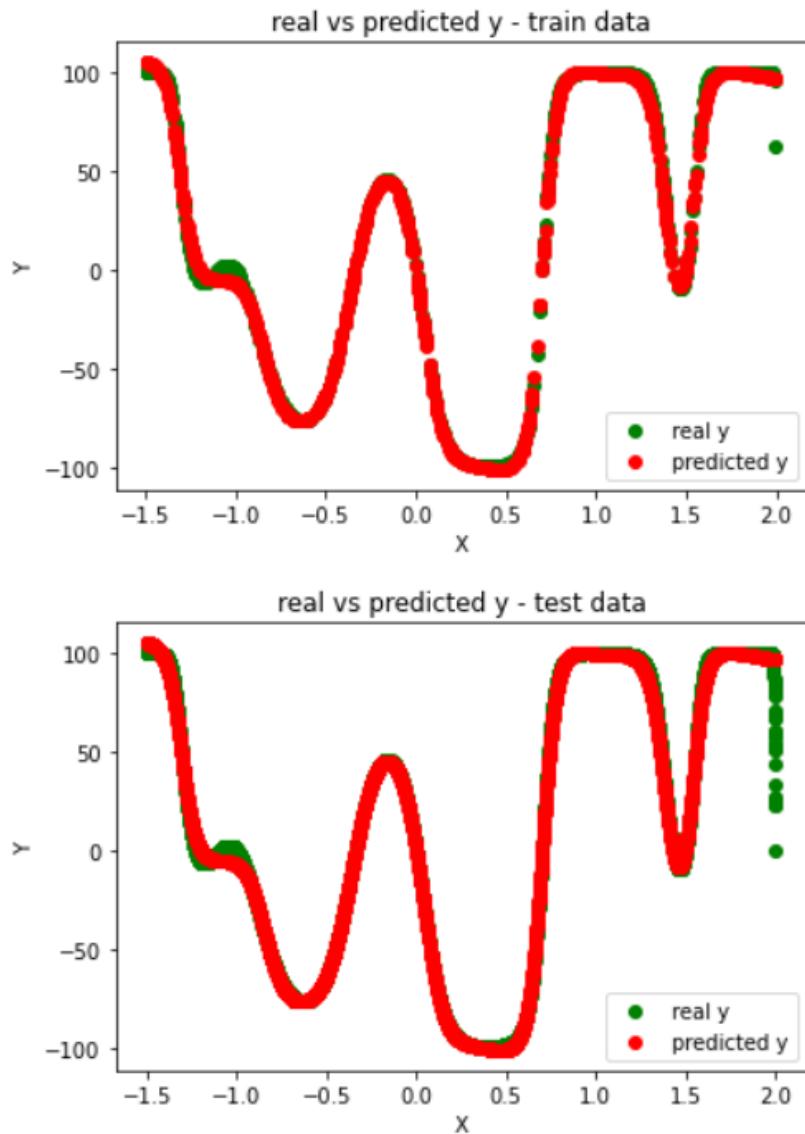
Po zakończeniu procesu uczenia stworzone zostały również wizualizacje wag w postaci gifów. Ostatnia klatka jednej z nich, zaprezentowana poniżej, reprezentuje wagi sieci mającej 1 hidden layer ze 100 neuronami:



Wizualizowane były także MSE loss na zbiorze treningowym i testowym w trakcie treningu. Przykładowy wykres poniżej dla sieci na zbiorze 'multimodal large':



Po każdym treningu wykonywany był również wykres przedstawiający przewidywania sieci na obu zbiorach. Sieć na 'multimodal large' sprawdziła się jak widać poniżej na wykresie:



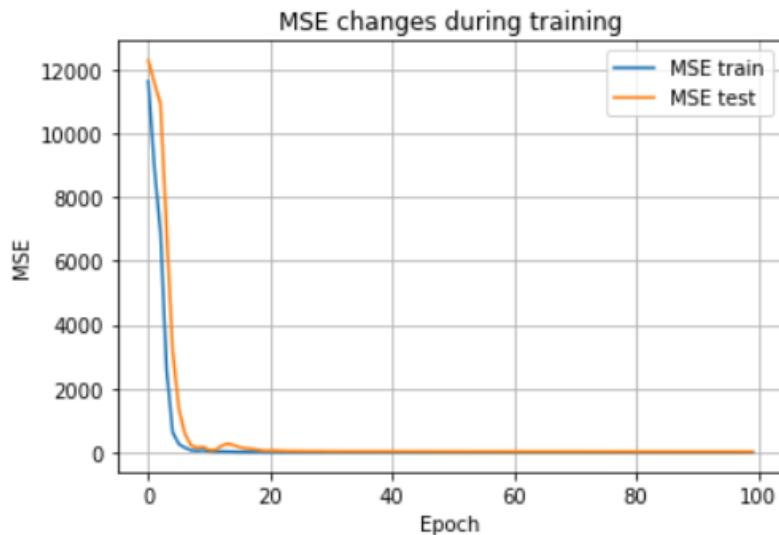
Jak widać, sieć się bardzo dobrze dopasowała i odwzorowuje skomplikowaną funkcję prawie całkiem poprawnie (MSE loss w tabeli z wynikami jest odpowiednio niski - około 10).

W tym laboratorium zaimplementowane zostały także inne sposoby losowania wag, które nie miały znaczącego wpływu na proces uczenia oraz wsparcie dla batch size - bez odpowiedniego zmniejszenia rozmiaru paczek danych do uczenia, proces uczenia trwałby znacznie dłużej, a dzięki zmniejszeniu batch size do 1, przebiegał dość sprawnie.

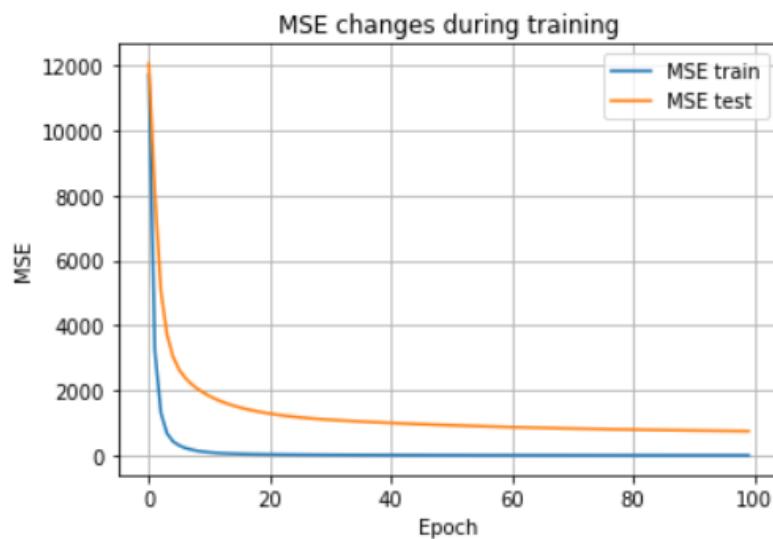
Laboratorium 4

W trakcie tego laboratorium porównane zostały dodatkowe elementy procesu uczenia - moment oraz RMS prop. Pierwsza z nich sprawdziła się doskonale i przyspieszała proces uczenia. Druga działała wolniej i nieco inaczej, ale również miała pozytywny wpływ na proces uczenia.

Przykładowy wykres MSE dla uczenia z momentem na 'square large':



Przykładowy wykres MSE dla uczenia z RMS prop na 'square large':



Rezultaty (najniższe MSE dla danego zbioru + metoda, z którą wynik został osiągnięty) umieszczone są w poniższej tabeli:

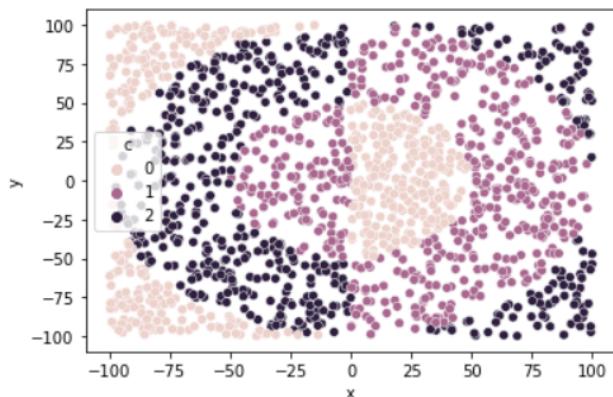
Dane	MSE train	MSE test
Square large (moment)	0.064	17.025
Steps large (moment)	24.625	16.543
Multimodal large (moment)	8.5	13.979

Laboratorium 5

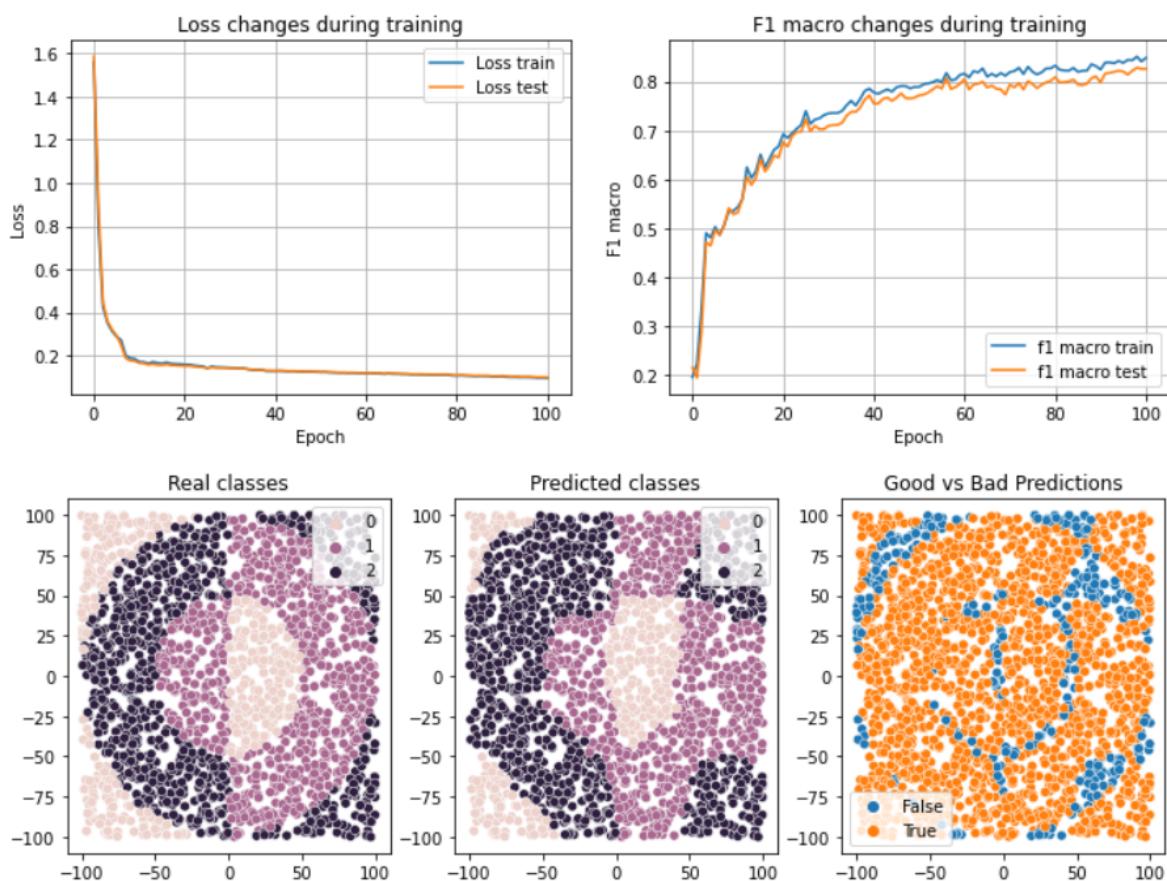
W tym laboratorium dodane zostały do implementacji możliwość liczenia F1 macro, a także możliwość użycia CrossEntropy loss z funkcją aktywacji Softmax na ostatniej warstwie (output). Testy zostały przeprowadzone na 3 różnych zbiorach, które wyglądały podobnie jak przedstawiony poniżej.

Zbiór rings3-regular

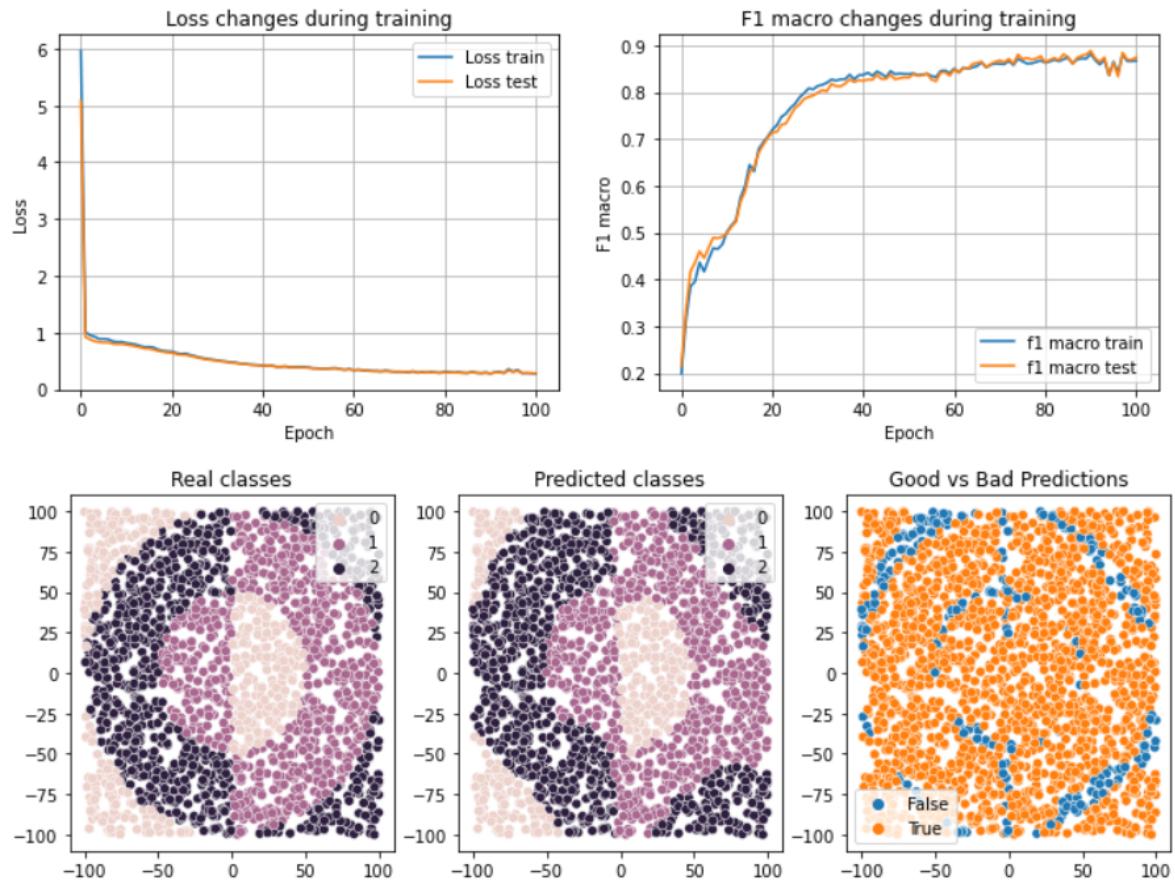
Tak wyglądał podział na klasy:



Efekty uczenia na nim były satysfakcjonujące, jak widać na kolejnych zestawach wykresów. Sieć bez softmax i crossentropy:



Sieć z softmax i crossentropy:

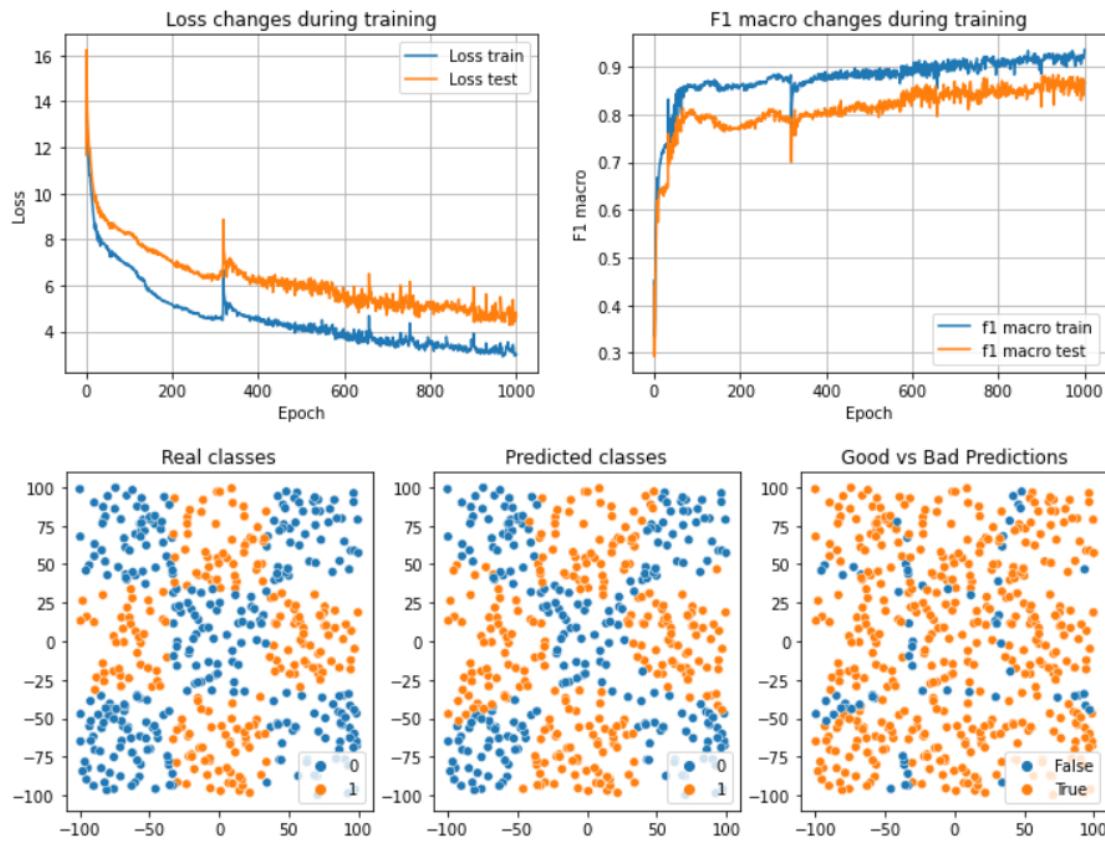


Obie sieci uczyły się dość podobnie z podobnym rezultatem.

Analogicznie eksperymenty przebiegły dla kolejnych zbiorów czyli 'easy' oraz 'xor3', choć dla ostatniego z nich problem był bardziej złożony i proces uczenia miał więcej niespodzianek. Mimo wszystko, uznać należy efekty za sukces, skoro sieć w głównej mierze dobrze klasyfikowała punkty jak zostało zaprezentowane w tabeli z rezultatami (f1 macro).

Zbiór xor3

Proces uczenia dla 'xor3' z softmax (bez softmax były podobne wyniki, z mniejszymi wahaniem, bardziej stabilnie).



Rezultaty

Dane	f1 macro train	f1 macro test
rings3-regular	0.867	0.875
easy	1	0.998
xor3	0.936	0.867

Różnic w skuteczności między zastosowaniem softmax a niezastosowaniem softmax nie było znaczących, jednak wyniki uzyskiwane w kolejnych eksperymentach były nieznacznie lepsze na korzyść tych, uzyskanych przez sieci wykorzystujące Crossentropy loss z Softmax funkcją aktywacji na ostatniej warstwie (w porównaniu z wykorzystaniem MSE loss i tanh na ostatniej warstwie).

Laboratorium 6

W trakcie tego dodana została możliwość zmiany funkcji aktywacji na tanh i relu (poza wcześniej dostępnymi sigmoid i linear). W ramach uściślenia - zaimplementowane ReLu przyjmowało wartość oraz wartość gradientu jako 0 dla x mniejszych od 0 (nie było to Leaky ReLu, czyli dla ujemnych x wartości są różne od 0, ale bardzo mało).

Porównanie 12 sieci - testy wstępne

Następnie trenowane było 12 sieci: dla każdej z 4 wymienionych wcześniej funkcji aktywacji trenowane były sieci z jedną, dwiema oraz trzema hidden layers. Aby sieci były do siebie możliwe podobne, każda z nich miała w sumie 120 neuronów w hidden layers, jednakże aby sieci się uczyły możliwie dobrze, dla każdej z nich dobrany zostały ręcznie parametr learning rate.

Wyniki trenowania tych 12 sieci na zbiorze ‘multimodal-large’ są w poniższej tabeli:

activation	hidden	learning rate	loss_train	loss_test
relu	1	0.0003	1602.898149	1566.403175
relu	2	0.0003	144.593680	143.693841
relu	3	0.0001	7.606620	12.495676
linear	1	0.0001	4459.693009	4424.354227
linear	2	0.0001	4583.188541	4548.167808
linear	3	0.0001	4649.009926	4613.984240
sigmoid	1	0.0030	4.527554	9.502291
sigmoid	2	0.0003	1583.002296	1445.241671
sigmoid	3	0.0003	416.797194	374.565892
tanh	1	0.0010	3.368971	8.402341
tanh	2	0.0003	737.841324	727.637919
tanh	3	0.0001	1152.339918	1042.079657

Wnioski z powyższej analizy są następujące:

- ReLu było tym lepsze, im więcej miało warstw
- Funkcje liniowe, niezależnie od liczby warstw, wciąż może zostać uproszczona do funkcji liniowej, więc zbioru 'multimodal-large' odwzorować (wielomian wysokiego stopnia) nie jest w stanie
- Sigmoid i Tanh są podobne, jednak Sigmoid jest nieco gorszy. Wynikać to może z tego, że nie przyjmuje wartości ujemnych.
- Sigmoid i Tanh są również podobne w kwestii szybkości uczenia względem liczby warstw - im więcej warstw, tym wolniej się uczyły.
- Warto używać numpy oraz nie używać wektoryzowanych funkcji - zaimplementowany w tym rozwiąaniu sigmoid był 3.2 razy wolniejszy na jednym treningu niż tanh.

Testy na pozostałych zbiorach danych

Następnie porównane zostały sieci używające Tanh z sieciami używającymi ReLu.

Porównanie dla regresji na 'steps large', obie sieci miały na output funkcję liniową.

Architecture	MSE train	MSE test
1x30 neurons, Tanh	28.101	19.797
3x30 neurons, ReLU	7.981	13.861

Porównanie dla klasyfikacji na 'rings5-regular', obie sieci miały na output funkcję tanh.

Architecture	f1 macro train	f1 macro test
3x20 neurons, Tanh	0.76	0.7
3x20 neurons, ReLU	0.529	0.407

Porównanie dla klasyfikacji na 'rings3-regular', obie sieci miały na output funkcję tanh.

Architecture	f1 macro train	f1 macro test
3x20 neurons, Tanh	0.79	0.774
3x20 neurons, ReLU	0.665	0.639

Należy wziąć pod uwagę, że wyniki zawarte w powyższej tabeli różnią się od najlepszych, jakie by można tą architekturą uzyskać, ponieważ przy większej liczbie epok wyniki byłyby nieco wyższe.

Wnioski z powyższych testów są takie, że tanh jest generalnie lepszy niż relu przy takiej samej architekturze sieci. Przy zwiększeniu sieci z aktywacjami ReLu, wyniki mogłyby być lepsze niż te osiągnięte z pomocą tanh, jednakże aby porównanie było bardziej przejrzyste, różnice zostały ograniczone do minimum.

Laboratorium 7

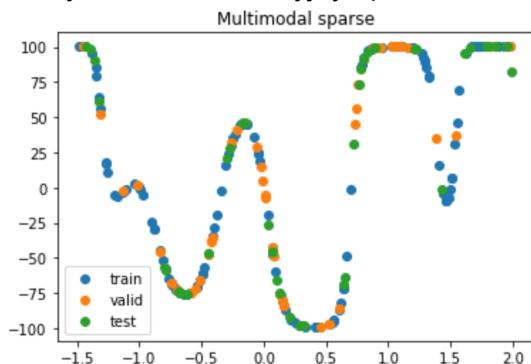
W ostatnim kroku zostały dodane regularizacje L1 i L2 oraz możliwość przerwania treningu, gdy błąd na zbiorze walidacyjnym wzrośnie o wskazaną liczbę razy względem błędu z poprzedniej epoki. Zaimplementowane zostało to w ten sposób, by dać możliwości przerywania dopiero przy znaczącym wzroście błędu, a nie jedynie o np 1% - w trakcie wykonywania testów, zazwyczaj parametr wynosił 1 lub 1.1, czyli pozwalający do 10% wzrostu.

Zbiory danych w tym laboratorium posiadały wysoce niezbalansowane dane lub dane rzadkie (jak wskazują nazwy zbiorów danych). Napotkane zostały trudności z wytrenowaniem skutecznych modeli.

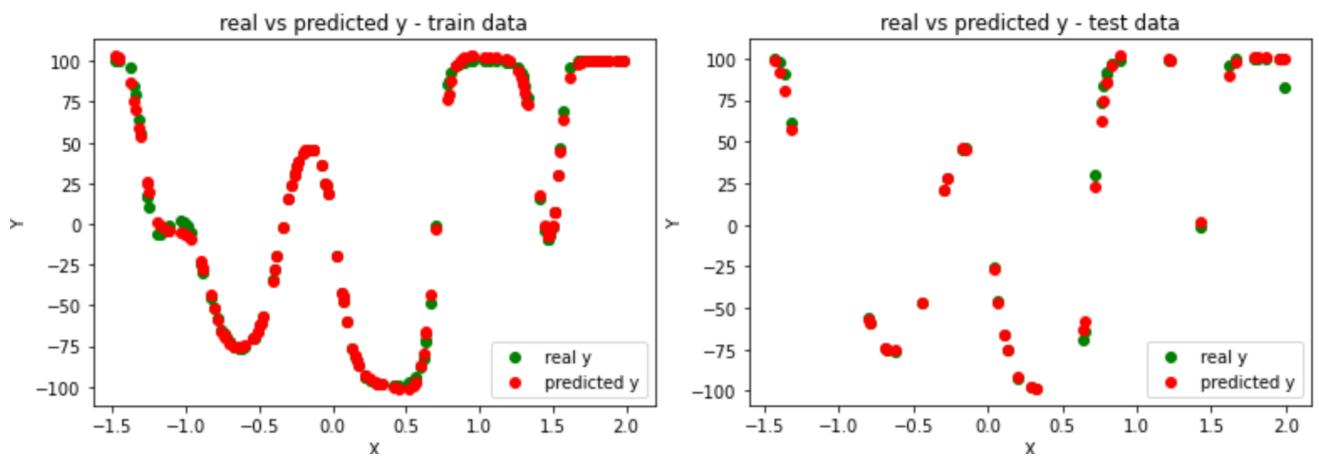
Ponadto, aby mieć prawdziwy zbiór testowy i walidacyjny, dane treningowe były dzielone na 2 zbiory wykorzystywane podczas treningu, a zbiór testowy jedynie przy ocenie wytrenowanego modelu podczas robienia wykresu f1 macro dla różnych parametrów regularizacji L1 i L2.

Zbiór 1 - multimodal-sparse

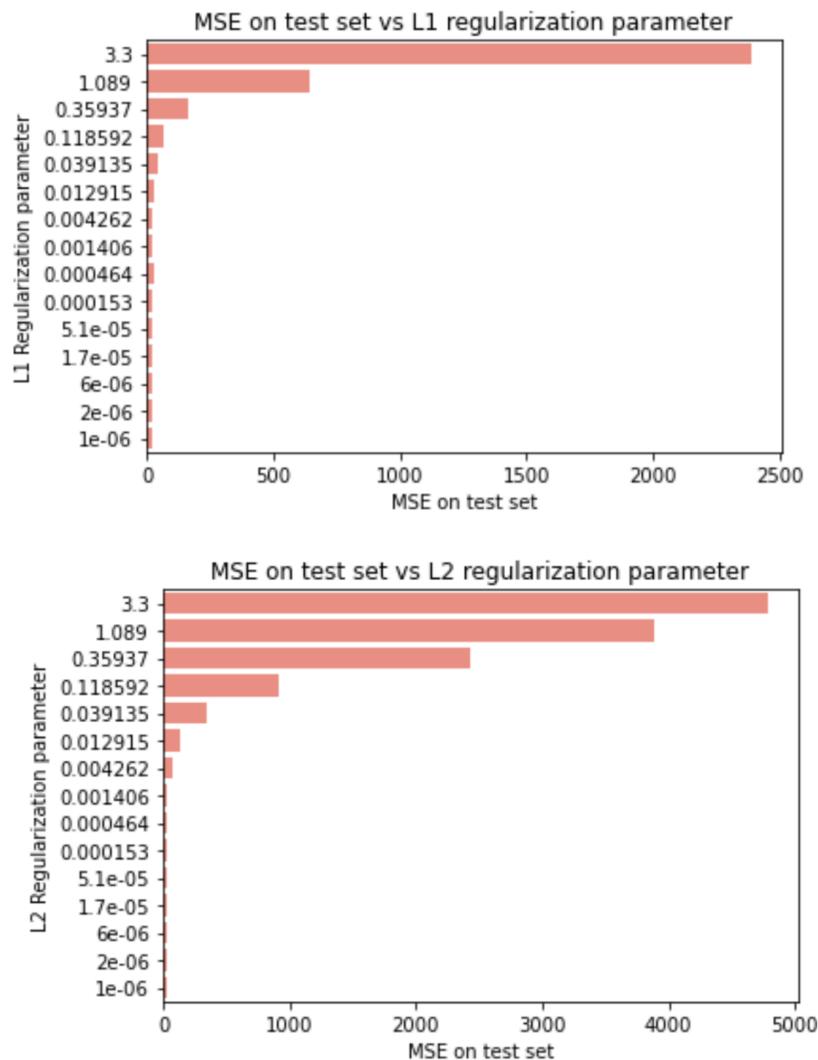
Zbiór danych miał postać znaną z wcześniejszych, lecz zawierał znacznie mniej punktów w zbiorze treningowym (a co za tym idzie i walidacyjnym).



Mimo tego model dość dobrze się dopasował:



Dla każdej z regularizacji model trenowany był dla wielu różnych parametrów regularizacji i osiągał poniższe wyniki:

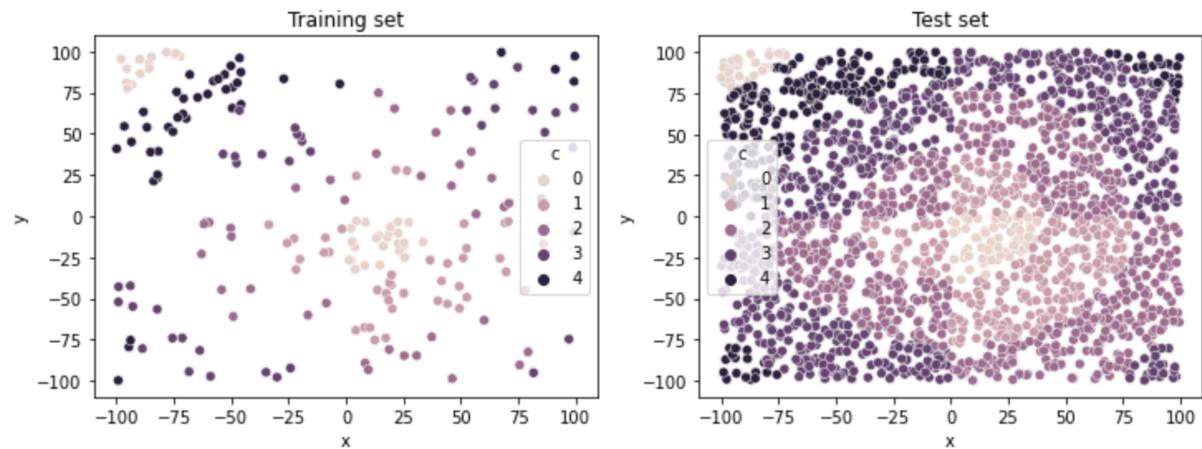


Jak można z powyższych wykresów wnioskować, parametr regularizacji ma duży wpływ na proces uczenia - przy zbyt dużym parametrze uczenie sieci jest utrudnione bądź niemożliwe, z kolei przy zbyt małym model zachowuje się tak, jakby tej regularizacji nie było (co oznacza, że wagi mogą być dowolnie duże, co nie jest pożądanym zjawiskiem).

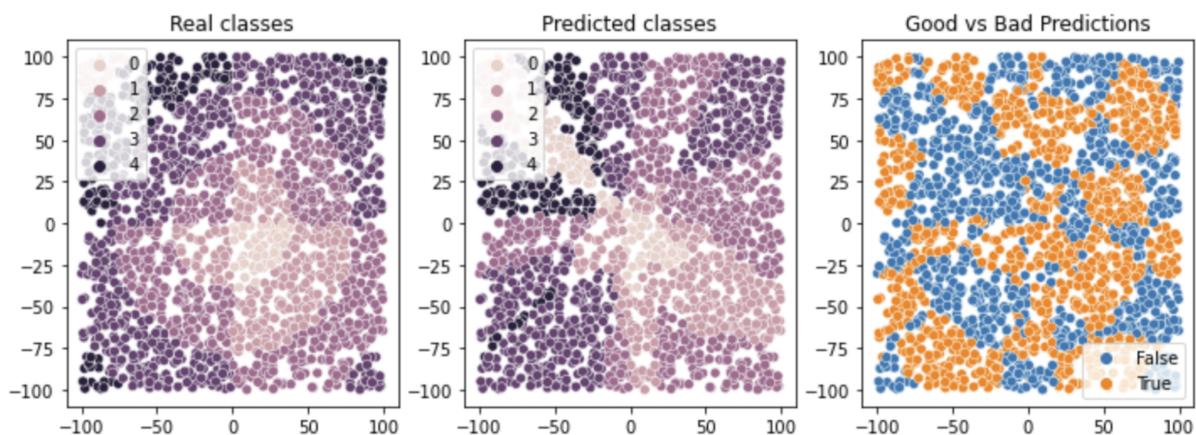
Na podstawie powyższych wykresów wnioskować można, że najlepszy parametr regularizacji jest ten, który jest największy spośród tych, dla których model osiągnął zadowalające rezultaty - np 0.0014 przy L2.

Zbiór 2 - rings5-sparse

Unique classes: 5

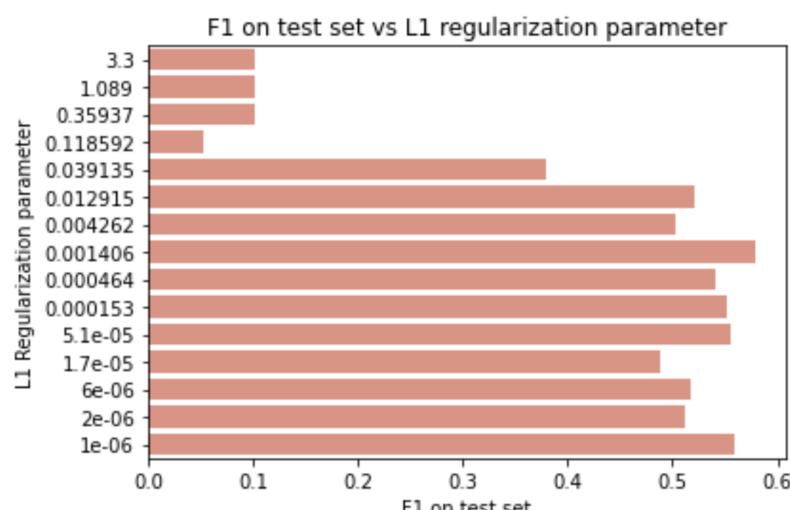


Danych w zbiorze treningowym praktycznie nie ma w porównaniu do testowego, stąd model niezależnie od regularizacji zachowywał się podobnie mało skutecznie:



Najlepsze f1 macro, to 0.7 na treningowym i walidacyjnym oraz 0.55 na testowym.

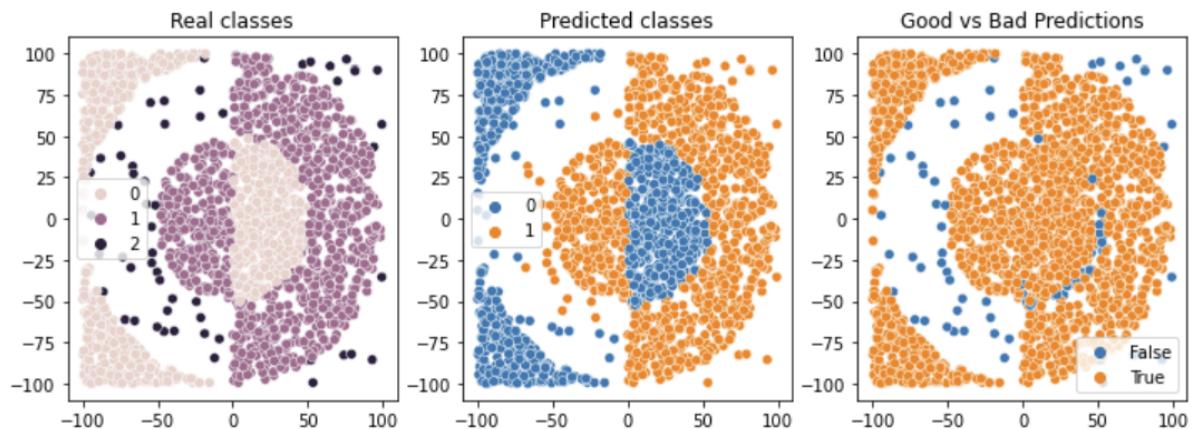
Wykres f1 macro na zbiorze testowym względem parametru regularizacji ponownie pokazuje, że zbyt duża regularizacja L1 lub L2 zatrzymuje trening:



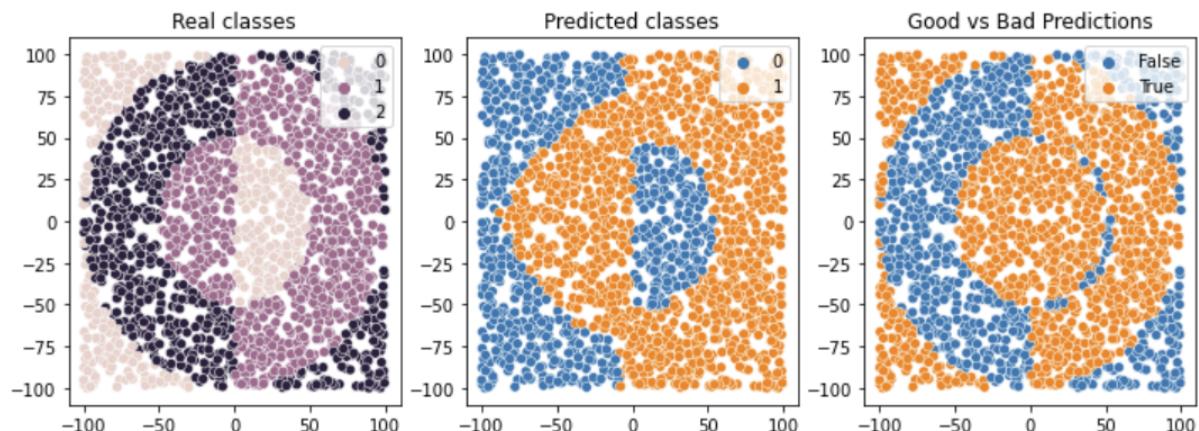
Zbiór 3 - rings-balance

Tym razem problemem było niebalansowanie danych treningowych - punktów z jednej z klas praktycznie nie było, co skutkowało, że losowo podzielony zbiór treningowy na zbiór treningowy i walidacyjny wciąż były niebalansowane i sieć nie uczyła się zupełnie przewidywać punktów z najmniej licznej klasy.

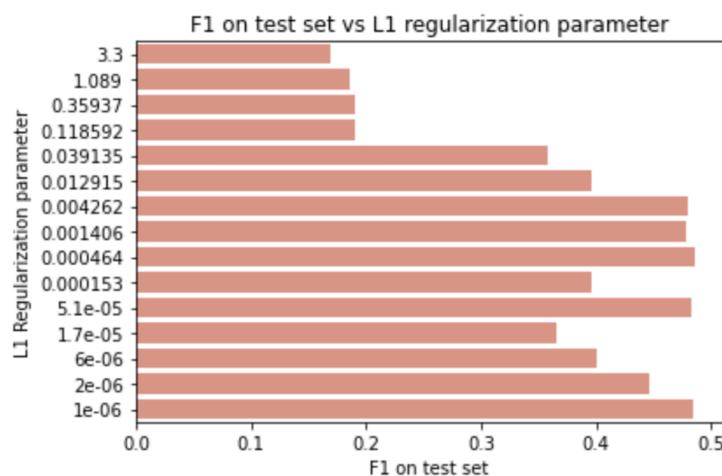
Zbiór treningowy:



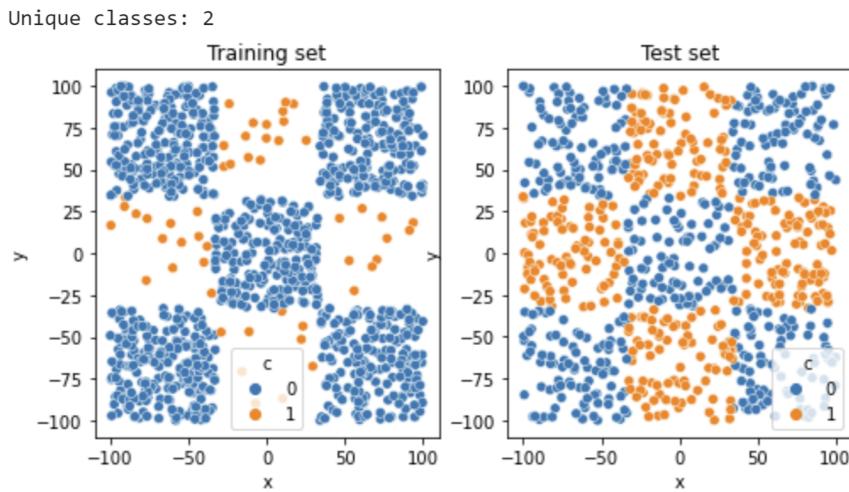
Zbiór testowy:



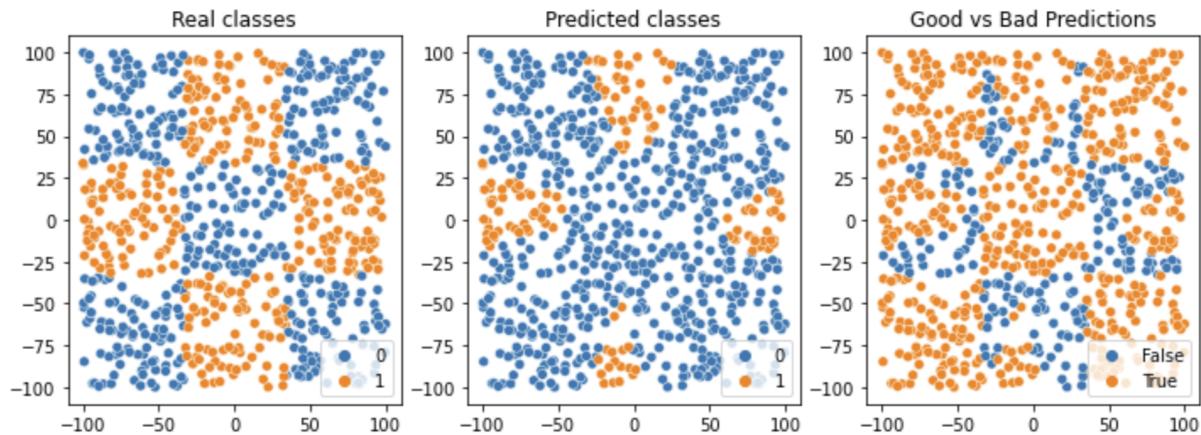
Wpływ parametru regularizacji L1 lub L2 był taki sam, jak poprzednio - zatrzymywał uczenie, gdy był za duży:



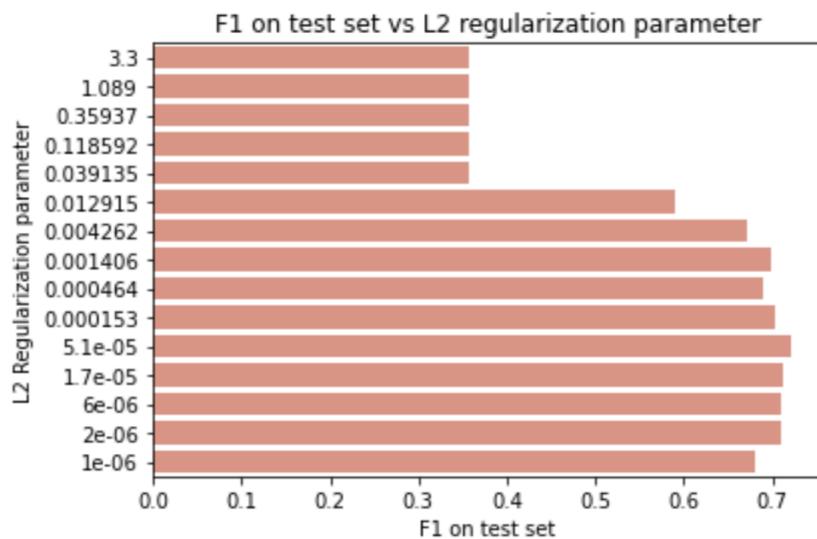
Zbiór 4 - xor3-balance



Ostatni zbiór był również niezbalansowany, jednak zawierał tylko 2 klasy i tym razem model się uczył, by przewidywać obie:



Regularyzacja zarówno L1, jak i L2, miała taki sam wpływ na proces uczenia:



Rezultaty

Na danych multimodal sparse sieci sprawdziły się dobrze przy obu sposobach regularyzacji:

Dane	MSE train	MSE valid	MSE test
Multimodal sparse	7.5	8.3	less than 20

Na danych do klasyfikacji modele miały problem się nauczyć, niezależnie od regularyzacji L1 lub L2. Jedyna drobna różnica jest widoczna przy danych 'xor3-balance', gdy regularyzacja L1 ma nieznacznie niższe wyniki od L2.

Dane	F1 macro train	F1 macro valid	F1 macro test
rings5-sparse	0.7	0.55	around 0.55
rings3-balance	0.63	0.63	around 0.48
xor3-balance	0.84 - L1 0.87 - L2	0.68 - L1 0.71 - L2	a bit less than 0.7 a bit more than 0.7

Wyniki średniej z metryki f1 dla każdej z klas na poziomie 0.6, gdy klas są 3, oznaczać może, że model zupełnie nie przewiduje jednej z klas - tak miało miejsce w 'rings3-balance', bo w zbiorze treningowym prawie nie było elementów z jednej z klas.

Wspólną cechą wszystkich treningów na każdym ze zbiorów jest to, że regularyzacja powstrzymywała lub utrudniała w znaczącym stopniu proces uczenia, jeśli parametr regularizacji był zbyt duży. Przy małych i bardzo małym parametrze regularizacji, modele były bliskie modelowi bez regularizacji i nie różniły się między sobą wynikami.

Early stopping czasem "przeszkadzało" w procesie uczenia, zatrzymując go zbyt szybko, mimo że w ogólności sieć się uczyła. Zmniejszenie parametrów learning rate i zwiększenie batch size pomogły uniknąć takich sytuacji. W celu zmniejszenia czasu uczenia, zastosowano przy treningach wzrostu błędów nie więcej niż 1.1 razy, czyli o nie więcej niż 10% względem błędu z poprzedniej epoki na zbiorze walidacyjnym,

Podsumowanie

Sieci przetestowane w trakcie opisanych wyżej laboratoriów i eksperymentów zachowywały się w dość przewidywalny sposób:

- Możliwe jest znalezienie kilku wag dla małej sieci ręcznie,
- Zwiększenie learning rate do pewnego stopnia przyspiesza proces uczenia, później sieć się nie uczy i błąd ciągle wzrasta,
- Zmniejszenie batch size przyspiesza proces uczenia kosztem stabilności,
- Moment i rms prop pomagają przy niskim batch size mieć bardziej stałe uczenie,
- Regularizacja ogranicza wartości wag, ale zbyt duża zatrzymuje uczenie,
- Funkcje aktywacji odgrywają ważną rolę - tahn i relu okazały najlepsze,
- Funkcja aktywacji softmax wraz z crossentropy jest dobrym wyborem do klasyfikacji,
- Niebalansowanie klas utrudnia lub uniemożliwia poprawne nauczenie się sieci.