Sieci neuronowe

Tomasz Makowski
nr indeksu 291132

Kwiecień 2020

Spis treści

1 Wstęp

Celem pierwszej części laboratoriów jest zbadanie wpływu różnych czynników na trenowanie sieci neuronowych. Zarówno jakość wyników jak i sam czas treningu.

Sieci te mogą składać się z kilku warstw (przynajmniej 2 warstwy – wejściowa i wyjściowa) a najlepiej jeszcze kilka warstw ukrytych. Warstwy pomiędzy sobą są gęsto połączone – każdy neuron z poprzedniej warstwy z każdym neuronem z następnej warstwy. Sieć ma udostępniać zmianę wielu parametrów służących zarówno treningowi sieci jak i samej architektury. W sieci można dodawać tyle warstw ukrytych ile się chce o dowolnej liczbie neuronów, na każdej warstwie można wybrać jedną z pośród odpowiednio przygotowanych funkcji aktywacji: liniowa, sigmoidalna, tanh i ReLU. Na ostatniej warstwie jest również dostępny softmax - co pozwala rozwiązywać zadania klasyfikacji.

Sieć pozwala również inicjalizować wagi z rozkładu jednostajnego lub za pomocą metody Xavier. Trening sieci może odbywać się w batchach. Podczas treningu można korzystać z metody momentu lub RMSProp. Dodatkowo zaimplementowany został mechanizm regularyzacji wag oraz zatrzymanie uczenia przy braku poprawy przez dłuższy czas.

2 Zbiory danych

W każdym eksperymencie wszystkie dane zaraz po wczytaniu były normalizowane. Jest to najważniejszy element dzięki któremu w ogóle było możliwe trenowanie sieci. Bez normalizacji sieć było bardzo ciężko trenować.

Dane można podzielić na te dotyczące regresji i klasyfikacji. Dane dotyczące regresji zawsze przedstawiają wartości na osi x jako obserwacje do nauki i na ich podstawie trzeba przewidywać wartość na osi y. Dane klasyfikacyjne przedstawiają punkty pogrupowane na płaszczyźnie (zatem obserwacja ma x i y) natomiast trzeba przewidywać jedną z od 2 do 5 klas. Dane klasyfikacyjne po unormowaniu mieszczą się w kwadracie o wymiarach około 3x3 scentrowanym w punkcie (0,0).

2.1 Regresja

W regresji zostały użyte następujące dane:

1. Zbiór simple-square przedstawia funkcję kwadratową o dodatnim współczynniku przy x^2.
2. Zbiory z steps i multimodal w nazwie przedstawiają 4 stopnie – tzn. funkcja jest niemalejąca i zawiera 4 fragmenty stałe, pomiędzy którymi następują skoki do następnego fragmentu.

2.2 Klasyfikacja

Do klasyfikacji zostały użyte następujące zbiory danych:

1. Zbiór easy zawiera 2 klasy liniowo separowalne linią $x = y$.
2. Zbiór xor zawiera podział na 9 równych kwadratów. Korzystając z numeracji klawiatury numerycznej komputera liczby 2, 4, 6 i 8 to jedna klasa, a liczby 1, 3, 5, 7 i 9 to druga klasa.
3. Zbiór rings zawiera 2 okręgi - mniejszy i większy. Podzielone są one pionowo na pół. Każdy fragment takiego podziału może być innej klasy. Jedna klasa zawiera najczęściej kilka fragmentów. Rings3 zawiera 3 klasy, natomiast rings5 zawiera 5 klas i ma jeszcze więcej linii podziału.

3 Eksperymenty

3.1 Ręczne dopasowywanie wag

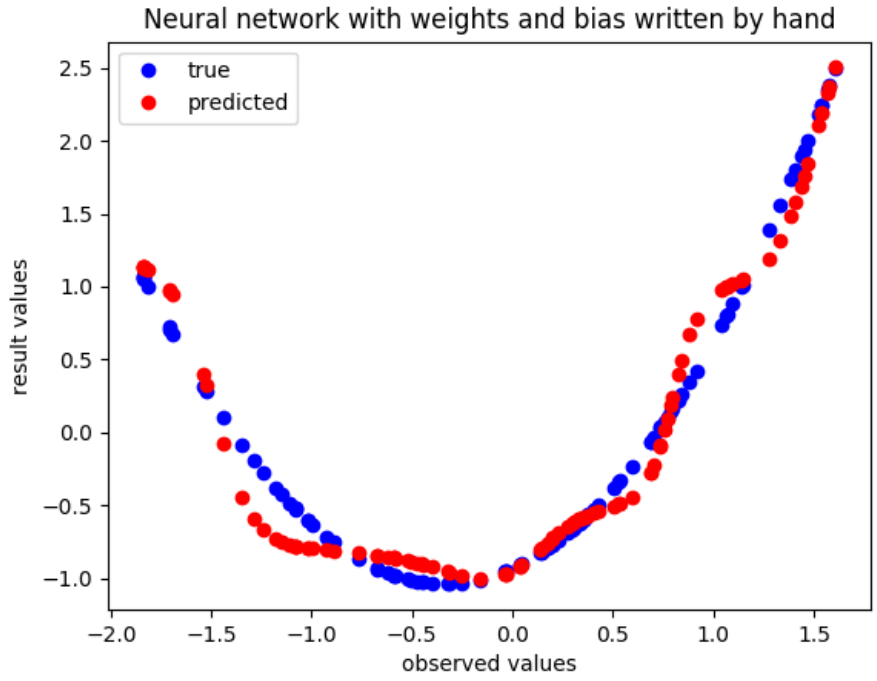
3.1.1 Opis

Pierwszym eksperymentem było ręczne dopasowanie wag do sieci bez użycia propagacji wstecznej. Testy zostały przeprowadzone na 2 zbiorach danych o nazwach simple-square i steps-large. W obu przypadkach zastosowano prostą architekturę: sieć miała jedną warstwę ukrytą z 5 neuronami, funkcją aktywacji na tej warstwie był sigmoid natomiast funkcją aktywacji na wyjściu była funkcja liniowa. Najistotniejszym elementem było ręczne wpisywanie wag i biasu dla sieci - łącznie 16 wartości.

Poniżej są wypisane wagi dla simple-square w_i^1 oznacza wagę na wejściu do i-tego neurona środowej warstwy, w_i^2 to waga wyjścia z i-tego neurona, b_i^1 to bias wejścia i-tego neurona, a b^2 to waga wejścia przy neuronie decyzyjnym sieci.

$w_1^1 = 15$	$b_1^1 = -12$	$w_1^2 = 1.5$	
$w_2^1 = 5$	$b_2^1 = 1$	$w_2^2 = -0.5$	
$w_3^1 = 10$	$b_3^1 = -15$	$w_3^2 = 2$	
$w_4^1 = -8$	$b_4^1 = 1$	$w_4^2 = 0.8$	
$w_5^1 = -10$	$b_5^1 = -15$	$w_5^2 = 2$	$b^2 = 0$

Na rysunku 1 zostało przedstawione porównanie prawdziwych danych z predykcją. Każdy z 5 neuronów ma



Rysunek 1: Simple square

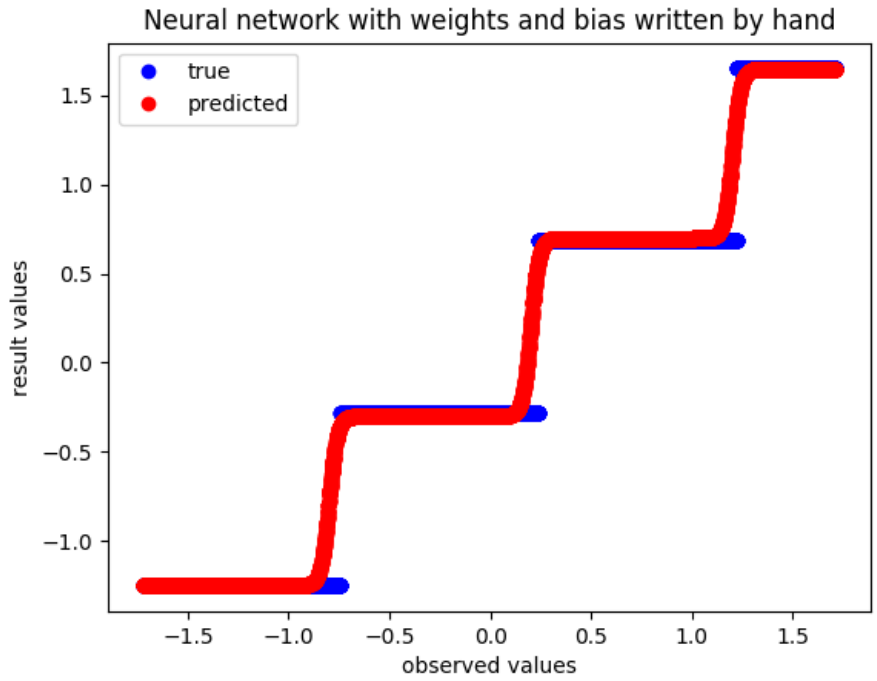
swoją interpretację i odpowiada za inny fragment krzywej z rysunku. Za fragment na osi x od -2 do około

-0.3 odpowiadają neurony 4 i 5. Charakteryzują się tym, że wagi w^1 są ujemne, zatem większe wartości y przypisują mniejszym wartością na osi x, co jest zgodne z wyglądem funkcji kwadratowej na tamtym odcinku. Dodatkowo bias b_5^1 ustawiony na -15 sprawia, że dla $x > 0$ wartość sigmoidu od neuronu 5 jest w przybliżeniu równa 0 - z tego powodu neuron 5 prawie nie ma wpływu na wartość dla $x > 0$. Analogicznie neurony 1, 2 i 3 są odpowiedzialne za wartości dla x¿0 i głównie za ten fragment wykresu.

W przypadku steps-large tak przedstawiają się wartości wag

$$
\begin{array}{lll}
w_4^1 = -50 & b_4^1 = 10 & w_4^2 = -0.3 \\
w_2^1 = 50 & b_2^1 = -10 & w_2^2 = 0.7 \\
w_3^1 = 0 & b_3^1 = 0 & w_3^2 = 0 \\
w_4^1 = -50 & b_4^1 = 10 & w_4^2 = -0.3 \\
w_5^1 = -50 & b_5^1 = -40 & w_5^2 = -0.95 \qquad b^2 = 0
\end{array}
$$

Natomiast na rysunku 2 zostało przedstawione porównanie prawdziwych danych z predykcją.



Rysunek 2: Steps large

Tutaj neuron 3 nie został nawet wykorzystany do przewidywania wyniku. Neurony 4 i 5 odpowiadają za ”2 niższe stopnie”. Neuron 4 powoduje wspólną obniżkę obu stopni do poziomu stopnia drugiego, a neuron 5 powoduje obniżkę tylko najniższego stopnia do odpowiedniego poziomu. Analogicznie neurony 1 i 2 podwyższają 2 ostatnie stopnie, gdzie 1 działa tylko na najwyższy stopień. To wszystko można było uzyskać dzięki biasom po -60 w przypadku 1 i 5 neuronu - dzięki temu działają tylko na mały fragment. Dodatkowo bardzo duże wagi w^1 powodują, że ”skoki” pomiędzy stopniami są gwałtowne.

3.1.2 Wnioski

Dopasowywanie wag bez korzystania z propagacji wstecznej jest zajęciem żmudnym. W szczególności ciężko jest estymować sigmoidami funkcję kwadratową. Zrobienie tego ręcznie z większą dokładnością byłoby trudne. W szczególności wykorzystanie wielu warstw sieci mogłoby sprawić dodatkowe problemy. Dodatkowo ręcznie dopasowana sieć działa dobrze na odcinku na którym została przygotowana, jednakże dla x nie należących do tego odcinka zachowanie może znacznie odbiegać od funkcji kwadratowej.

Ręcznie wytrenowana sieć ma wyraźny podział odpowiedzialności poszczególnych neuronów. W przypadku trenowania za pomocą spadku gradientu taki podział nie musi zostać zachowany i na przykład większość/wszystkie neurony mogą "chcieć" dopasowywać krzywą dla x¿0. Wtedy zbieganie do dobrego rozwiązania może trwać długo albo nawet może utknąć w lokalnym minimum i nie być wstanie dopasować krzywej po całości. Dodatkowo ręcznie wytrenowana sieć korzysta z bardzo małej liczby neuronów a mimo wszystko jest wstanie dobrze estymować krzywą.

3.2 Uczenie batchami

3.2.1 Opis

Drugim eksperymentem jest porównanie czasu uczenia za pomocą batchy z uczeniem na podstawie całego zbioru danych. Eksperyment został przeprowadzony na 3 zbiorach danych:

1. steps-small, który ma 50 obserwacji
2. square-simple, który ma 100 obserwacji
3. multimodal-large, który ma 10000 obserwacji

W ramach każdego zbioru danych zostały sprawdzone 3 podejścia:

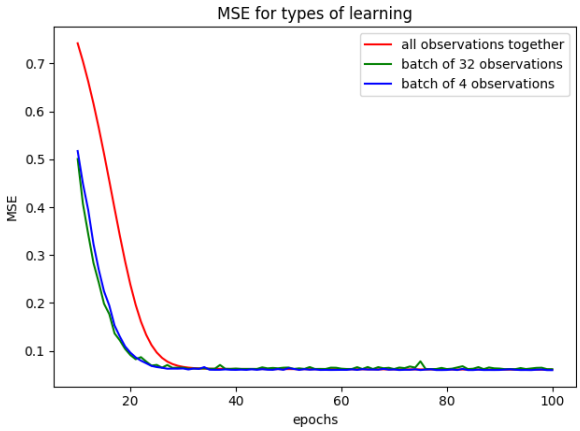
1. uczenie na podstawie całego zbioru danych naraz
2. batche po 32 obserwacje
3. mini batche po 4 obserwacje

Na wykresach 3 4 5 przedstawiono MSE w zależności od epoki poczynając od epoki 10 żeby czytelniejsza była właściwa zbieżność. Sieci były uczone z jedną warstwą ukrytą mającą 20 neuronów, funkcją aktywacji sigmoid, inicjalizacją wag typu Xavier oraz η (learning rate) dostosowaną dla każdej sieci oddzielnie. Zazwyczaj $\eta = 0.01$ jedyne wyjątki to w square-simple dla wszystkich obserwacji $\eta = 0.001$ oraz w multimodal-large $\eta = 0.00001$.

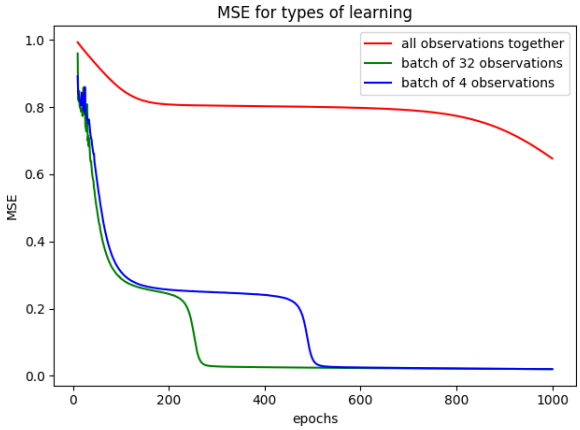
Na wykresach 6 7 8 można zaobserwować, że sieć faktycznie się nauczyła aczkolwiek nie najlepiej. Tam gdzie są stopnie nie widać dopasowania do stopni a jedynie "pofalowaną" linię. Również w przypadku funkcji kwadratowej sieć niekoniecznie jest wstanie się dopasować do obu zagięć funkcji.

3.2.2 Wnioski

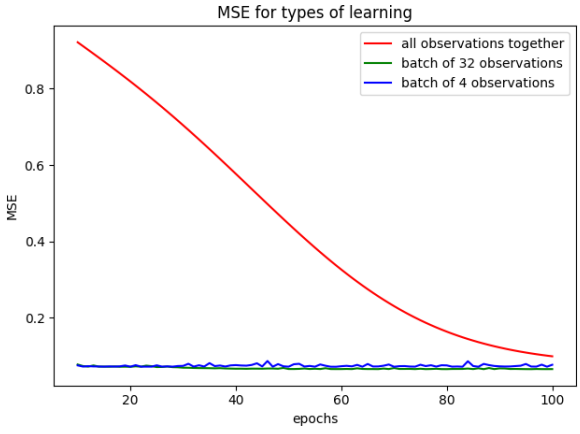
Najlepszym sposobem nauki są batche - w tym przypadku były sprawdzone batche po 32 obserwacje. Batche po 4 są za małe i powodują skoki funkcji celu, natomiast kompletny brak batchy sprawia, że uczenie jest wolne w szczególności na zbiorze mającym 10000 obserwacji jest to nieefektywny sposób. Batche większe



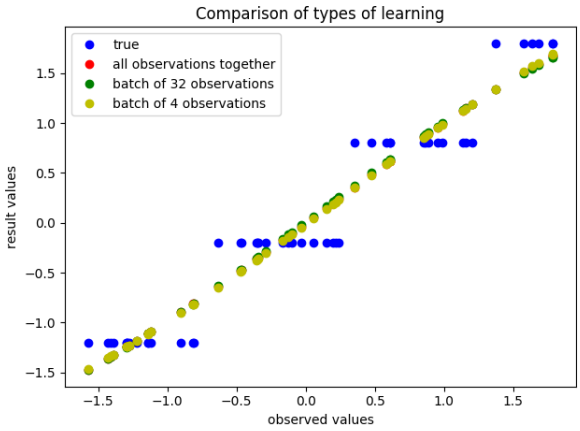
Rysunek 3: Steps small



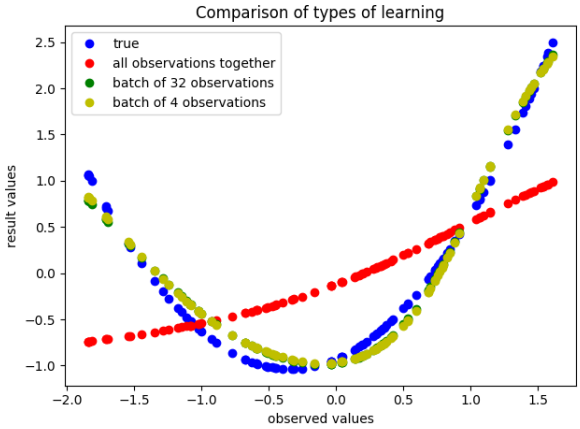
Rysunek 4: Square simple



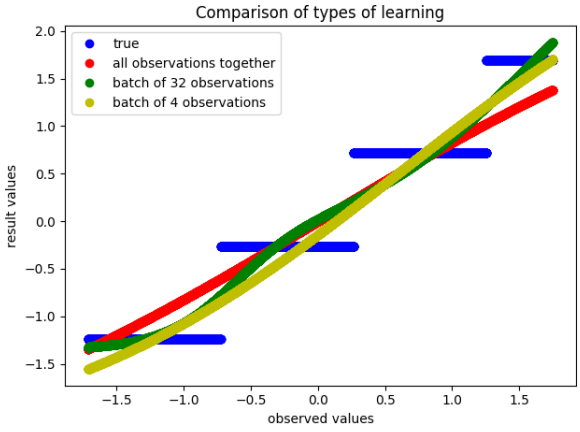
Rysunek 5: Multimodal large



Rysunek 6: Steps small



Rysunek 7: Square simple



Rysunek 8: Multimodal large

raczej nie mają uzasadnienia ponieważ steps-small miał 50 obserwacji i nauka na wszystkich obserwacjach na raz okazała się wolniejsza niż w 2 batchach po 32 i 18 obserwacji.

Od teraz wszystkie zastosowania będą miały naukę z batchami po 32 obserwacje.

3.3 Inicjalizacja wag

3.3.1 Opis

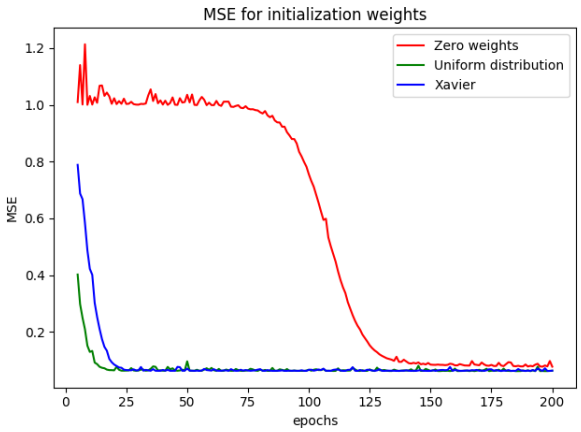
Kolejnym eksperymentem jest sposób inicjalizacji wag. Eksperyment został przeprowadzony na tych samych 3 zbiorach danych co poprzedni:

1. steps-small, który ma 50 obserwacji
2. square-simple, który ma 100 obserwacji
3. multimodal-large, który ma 10000 obserwacji

W ramach każdego zbioru danych zostały sprawdzone 3 sposoby inicjalizacji wag:

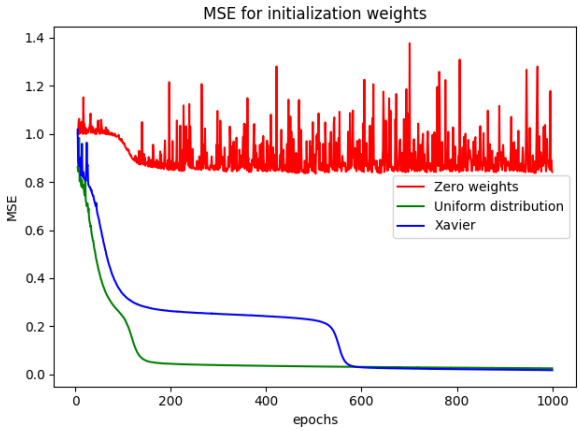
1. wszystkie wagi i bias równe zero
2. wagi z rozkładu jednostajnego na przedziale $[-\frac{1}{2}, \frac{1}{2}]$
3. wagi za pomocą metody Xavier, czyli z rozkładu jednostajnego na przedziale $[-\frac{\sqrt{6}}{IN+OUT}, \frac{\sqrt{6}}{IN+OUT}]$, gdzie IN to liczba neuronów z poprzedniej warstwy, a OUT to liczba neuronów w następnej warstwie

Na wykresach 9 10 11 przedstawiono MSE w zależności od epoki poczynając od epoki 5 żeby czytelniejsza była właściwa zbieżność. Sieci były uczone z jedną warstwą ukrytą mającą 20 neuronów, funkcją aktywacji sigmoid oraz $\eta = 0.01$ (learning rate).

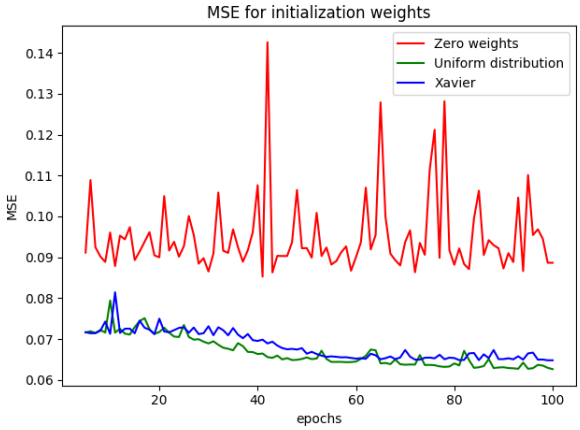


Rysunek 9: Steps small

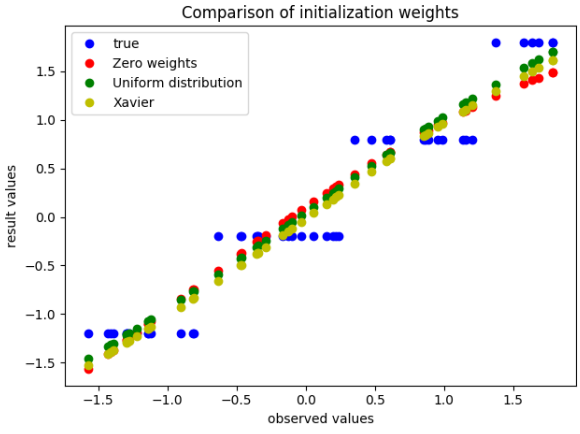
Na wykresach 12 13 14 można zaobserwować, że sieć nauczyła się podobnie jak w przypadku eksperymentu z wielkością batcha. Najgorzej tym razem wypadły wagi początkowe równe zero.



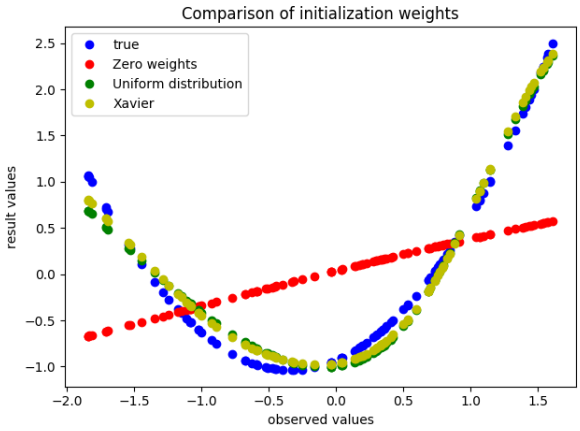
Rysunek 10: Square simple



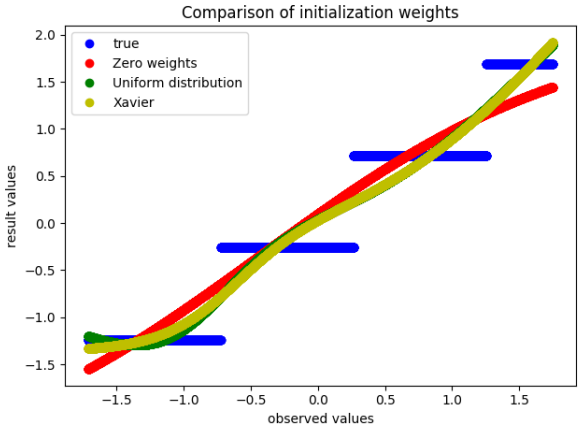
Rysunek 11: Multimodal large



Rysunek 12: Steps small



Rysunek 13: Square simple



Rysunek 14: Multimodal large

3.3.2 Wnioski

Najlepszym sposobem inicjalizacji wag jest metoda z rozkładem jednostajnym. Najczęściej zbiega szybciej niż metoda Xavier. Również dochodzi do stabilniejszych rozwiązań. MSE mniej "faluje". Wagi równe 0 są słabą metodą ponieważ sieć musi najpierw w jakiś sposób podzielić zadania dla poszczególnych neuronów. Inicjalizacja z przedziału ma tę przewagę, że neurony mają już jakieś zadania, trzeba je tylko lepiej nauczyć.

Od teraz wszystkie zastosowania będą miały naukę z metodą inicjalizacji wag z rozkładu jednostajnego.

3.4 Uczenie z momentem

3.4.1 Opis

Następnym eksperymentem jest skorzystanie z uczenia z momentem lub . Eksperyment został przeprowadzony na tych samych 3 zbiorach danych tylko z większą liczbą obserwacji:

1. steps-large, który ma 1000 obserwacji
2. square-large, który ma 10000 obserwacji

3. multimodal-large, który ma 10000 obserwacji

W ramach każdego zbioru danych zostały sprawdzone 3 sposoby zapamiętywania wag:

1. brak żadnego zapamiętywania poprzednich zmian wag
2. uczenie z momentem
3. RMSProp

Uczenie z momentem

Polega na zapamiętywaniu delty wag z poprzednich kroków i wygaszaniu tej wartości. Normalnie liczy się tylko deltę wag $\Delta\theta$ i w kroku zmiany wag dodaje się je z parametrem η: $\theta = \theta + \eta \cdot \Delta\theta$.

W przypadku uczenia z momentem $\Delta\theta$ dodaje się do zmiennej momentu M, a poprzedni moment wygasza się z parametrem λ: $M = \Delta\theta + \lambda \cdot M$. Następnie dopiero korzystając z momentu uaktualnia się wagi: $\theta = \theta + \eta \cdot M$.

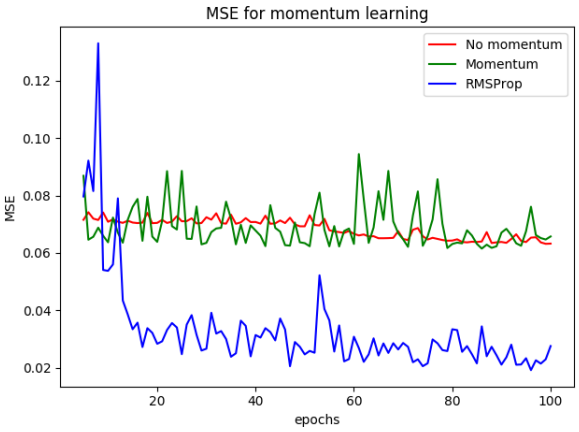
Podczas eksperymentu najlepszą wartością okazała się $\lambda = 0.9$

RMSProp

W przypadku uczenia typu RMSProp korzysta się z kwadratów momentów. Używany jest parametr β i zgodnie z wzorem $E = \beta \cdot E + (1 - \beta) \cdot g^2$ zapamiętuje się wartość E, gdzie $g = -\Delta\theta$, a g^2 jest kwadratem po współrzędnych. Następnie korzystając z E uaktualnia się wagi zgodnie ze wzorem $\theta = \theta - \eta \cdot \frac{g}{\sqrt{E}}$, gdzie $\frac{g}{\sqrt{E}}$ jest działaniem po współrzędnych.

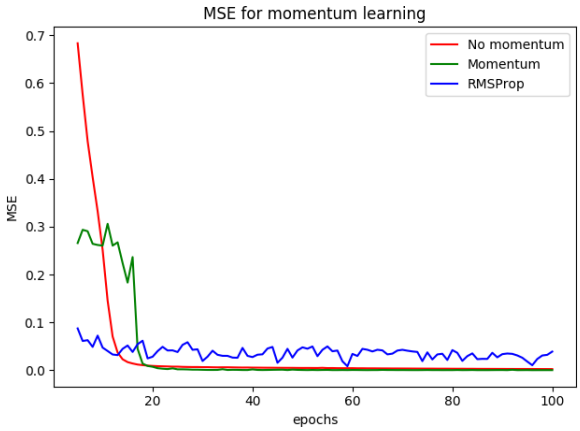
Podczas eksperymentu najlepszą wartością okazała się $\beta = 0.9$

Na wykresach 15 16 17 przedstawiono MSE w zależności od epoki poczynając od epoki 5 żeby czytelniejsza była właściwa zbieżność. Sieci były uczone z jedną warstwą ukrytą mającą 20 neuronów, funkcją aktywacji sigmoid oraz najczęściej $\eta = 0.01$ (learning rate). Dla RMSProp $\eta = 0.1$ - po testach okazało się że ta wartość daje lepszą zbieżność.

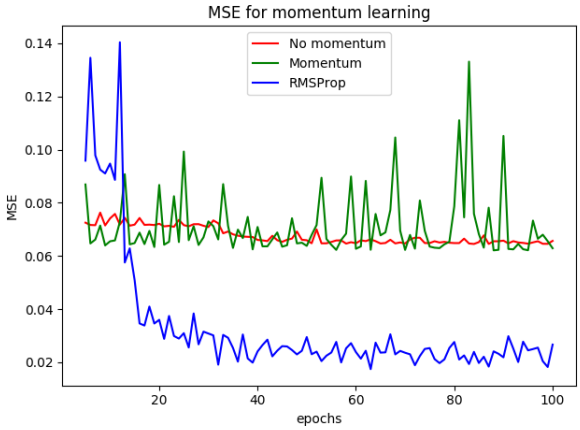


Rysunek 15: Steps large

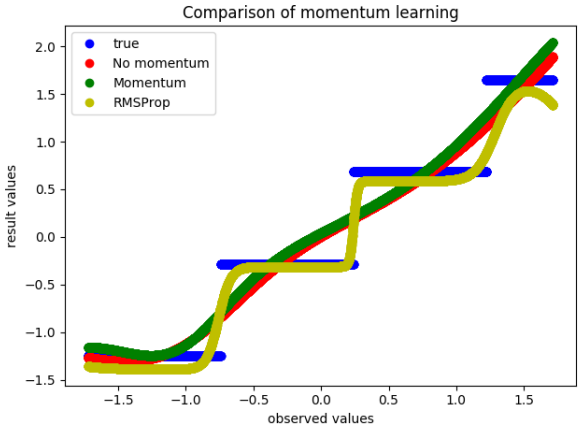
Na wykresach 18 19 20 można zaobserwować, że w końcu sieć nauczyła się stopni w przypadku uczenia z RMSProp. Pozostałe metody nadal nie były wstanie nauczyć się stopni. Wszystkie metody nauczyły się poprawnie funkcji kwadratowej.



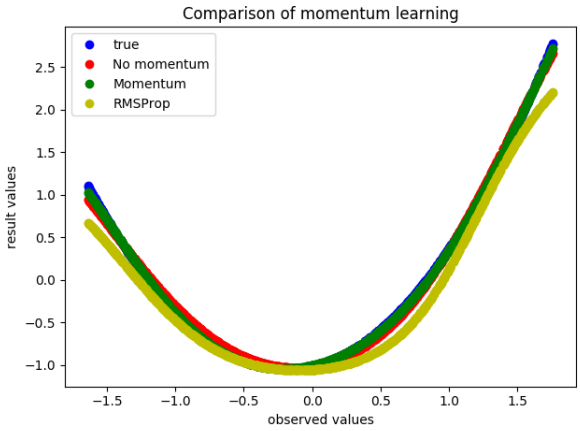
Rysunek 16: Square large



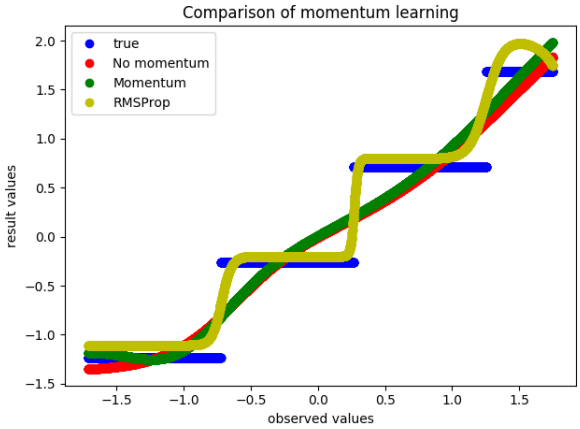
Rysunek 17: Multimodal large



Rysunek 18: Steps large



Rysunek 19: Square large



Rysunek 20: Multimodal large

3.4.2 Wnioski

RMSProp daje szybszą zbieżność niż uczenie bez momentu albo z momentem. Dzięki zastosowaniu RMSProp możliwe było nauczenie się przez sieć stopni tzn. zrobienie wypłaszczeń. Metoda z momentem okazała się ogólnie gorsza niż metoda bez momentu. Była mniej stabilna w przypadku steps i multimodal a w przypadku square zbiegła później. RMSProp niestety poradził sobie gorzej w przypadku zbioru square - co prawda zbiegł szybko, ale potem trzymał się bardzo niestabilnie z wyższym MSE niż pozostałe metody. Mimo tego RMSProp wydaje się najlepszą metodą do stosowania "w ciemno" dlatego od teraz dalsze eksperymenty będą korzystały z RMSProp.

3.5 Softmax dla problemu klasyfikacji

3.5.1 Opis

Następnym eksperymentem było zaimplementowanie funkcji aktywacji softmax i sprawdzenie czy działa lepiej niż liniowa funkcja aktywacji. Softmax jest typową funkcją aktywacji na ostatniej warstwie dla problemu klasyfikacji, gdy jedna klasa powinna być właściwa. Softmax skaluje wartości na wszystkich neuronach w

ostatniej warstwie tak, że ich suma jest równa 1. Zatem można traktować te wartości jako prawdopodobieństwa tego, że dana klasa jest właściwa.

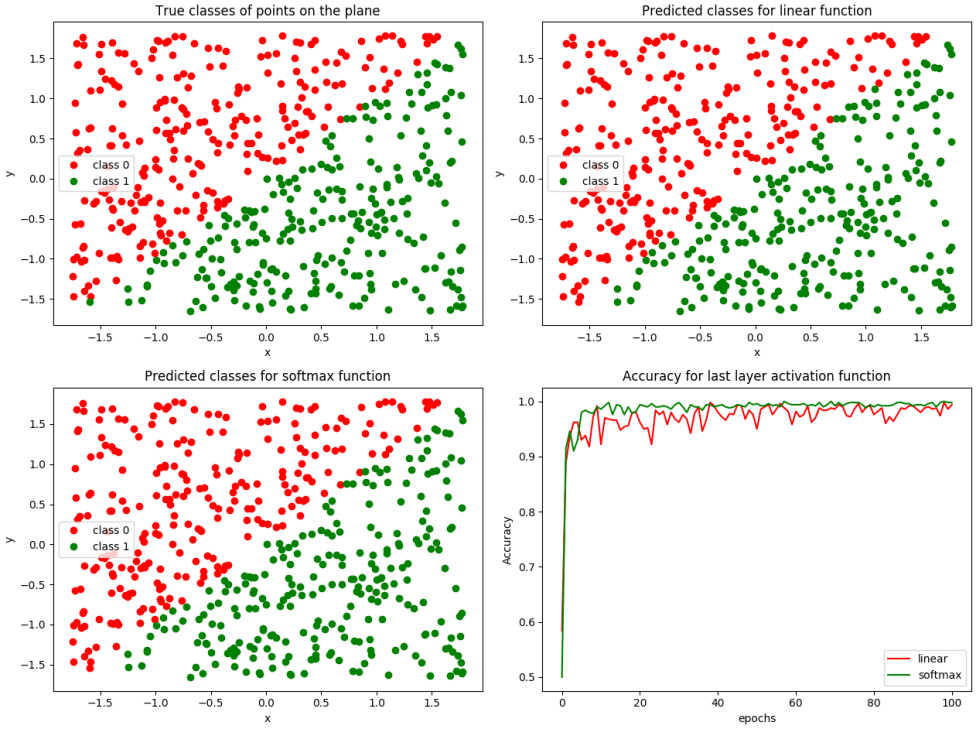
Eksperyment został przeprowadzony na następujących zbiorach danych:

1. easy, który ma 500 obserwacji
2. xor3, który ma 500 obserwacji
3. rings3-regular, który ma 1500 obserwacji

W ramach każdego zbioru danych zostały sprawdzone 2 funkcje aktywacji w ostatniej warstwie

1. liniowa
2. softmax

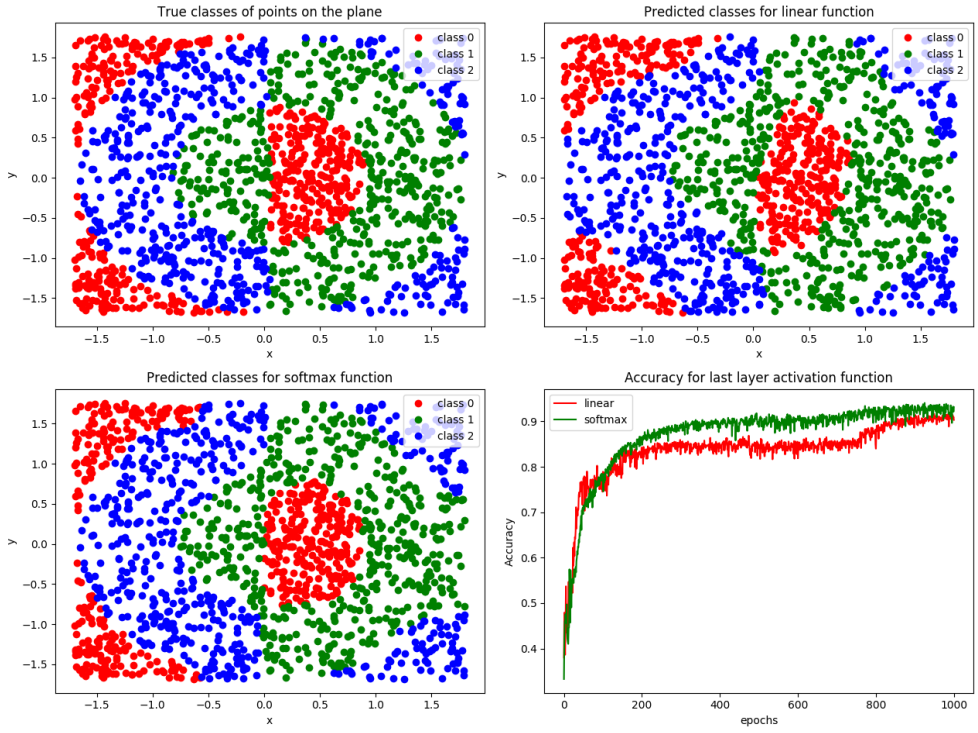
Na wykresach 21, 22 i 23 przedstawiono podział punktów na klasy, wyniki działania sieci z funkcją aktywacji liniową oraz softmax oraz accuracy w zależności od epoki poczynając od epoki 5 żeby czytelniejsza była właściwa zbieżność. Sieci były uczone z jedną warstwą ukrytą mającą 20 neuronów (z wyjątkiem xor3 - tam zostały użyte 2 warstwy ukryte po 20 neuronów inaczej sieć nie potrafiła nauczyć się środkowego kwadratu), funkcją aktywacji sigmoid na pozostałych warstwach oraz najczęściej $\eta = 0.01$ (learning rate).



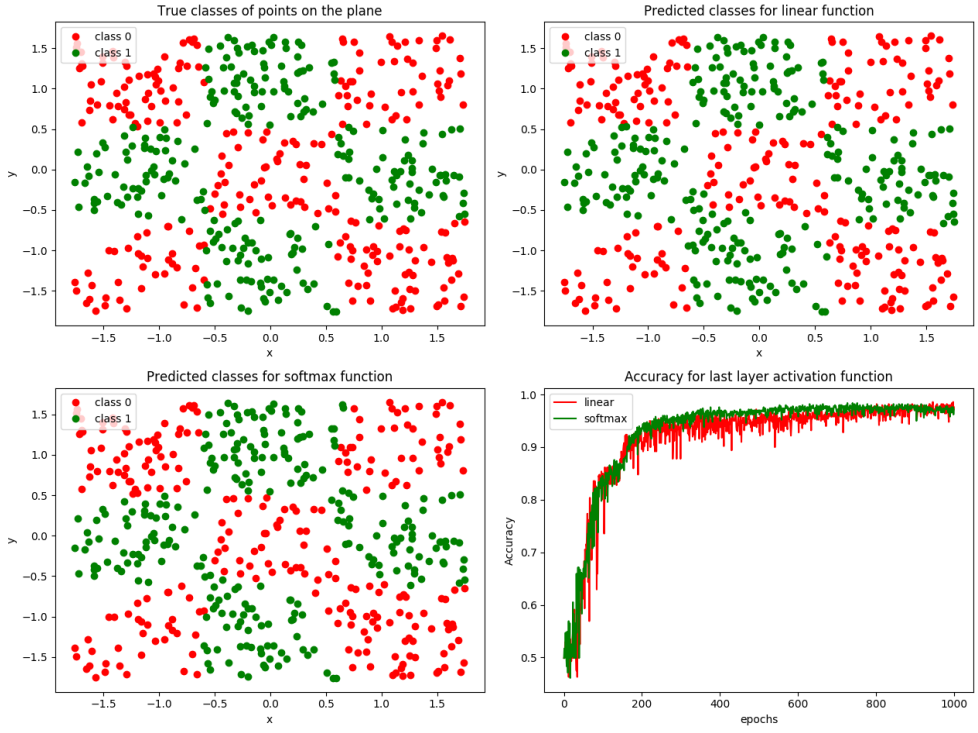
Rysunek 21: Easy

3.5.2 Wnioski

Zbiór easy został prawie perfekcyjnie dopasowany przez obie funkcje aktywacji. Softmax na ostatniej warstwie był nieznacznie lepszy. W przypadku zbioru ring3-regular softmax był znacznie lepszy, osiągnął prawie 10%



Rysunek 22: Ring3 regular



Rysunek 23: Xor3

lepszą skuteczność. Liniowa funkcja aktywacji miała problem ze znalezieniem okręgów. W przypadku zbioru Xor3 softmax również był lepszy. Różnice pomiędzy softmaxem a funkcją liniową były niewielkie.

Reasumując softmax jest lepszą funkcją aktywacji, a przynajmniej nie gorszą, niż liniowa funkcja aktywacji na ostatniej warstwie. Zatem od teraz w przypadku problemu klasyfikacji na ostatniej warstwie będzie używany softmax. Przy problemie regresji jak do tej pory funkcją aktywacji będzie funkcja liniowa.

3.6 Funkcje aktywacji

3.6.1 Opis

Kolejnym problemem było znalezienie najlepszych funkcji aktywacji oraz sprawdzenie tych funkcji aktywacji na różnych architekturach sieci.

Eksperyment został przeprowadzony na następujących zbiorach danych:

1. multimodal-large, który ma 10000 obserwacji, jest to zbiór regresyjny
2. steps-large, który ma 10000 obserwacji, jest to zbiór regresyjny
3. rings3-regular, który ma 1500 obserwacji, jest to zbiór klasyfikacyjny
4. rings5-regular, który ma 1500 obserwacji, jest to zbiór klasyfikacyjny

W ramach każdego zbioru danych zostały sprawdzone 4 funkcje aktywacji w ostatniej warstwie

1. liniowa
2. ReLU
3. sigmoid
4. tanh

Również zostały sprawdzone sieci mające 1, 2 lub 3 warstwy ukryte. Każda warstwa ukryta miała 50 neuronów. Użyty learning rate wynosił $\eta = 0.01$.

Na wykresach 25 i 24 przedstawiono wyniki działania sieci w postaci MSE dla problemu regresji oraz predykcje dopasowanego modelu. Wykresy MSE były zaczęte od 5 epoki, żeby uniknąć pokazywania dużych wartości. Dodatkowo w przypadku 3 epok został zaprezentowany wykres MSE bez funkcji aktywacji ReLU. Natomiast na wykresach 26 i 27 zostały przedstawione wyniki działania sieci w postaci accuracy dla problemu klasyfikacji wraz z zaprezentowanym zbiorem danych.

3.6.2 Wnioski

Regresja

Najgorszą funkcją aktywacji okazała się funkcja liniowa. W żadnym przypadku nie wykryła "schodów". Jeśli chodzi o pozostałe funkcje nie da się jednoznacznie wskazać najlepszej. Dla sieci z jedną warstwą ukrytą najlepiej sprawdzała się funkcja ReLU, ponieważ wykryła schody. Pozostałe funkcje wykryły tylko niektóre stopnie. Jednak dla sieci bardziej złożonych, czyli na przykład z 3 warstwami ukrytymi ReLU ma problem z gradientem i często on jest NaNem. Przez to ma wynik gorszy niż funkcja liniowa. W przypadku sieci z 3 warstwami ukrytymi najlepszy jest sigmoid, jednakże jest wolniej zbieżny niż tanh, który jest na 2 miejscu. Przy 2 warstwach tanh jest lepszy niż sigmoid. Najlepsze wyniki sieć osiąga przy 3 warstwach ukrytych,

jednakże są one tylko niewiele lepsze od wyników przy 2 warstwach ukrytych, a sprawiają, że sieć się dłużej uczy. Jedynie sieć z jedną warstwą ukrytą jest wyraźnie gorsza, jednakże stosując wtedy ReLU można znacznie przyspieszyć czas nauki.

Bazując na wynikach nie ma dużego znaczenia, czy się użyje 2 czy 3 warstwowej sieci. Jednakże istotne jest to, żeby używać przy nich sigmoidu, ewentualnie można wziąć pod uwagę użycie tanh i dokładniej sprawdzić wyniki. Jeżeli z jakiegoś powodu potrzebna jest prosta i szybka do nauczenia sieć można zastosować 1 warstwę ukrytą z ReLU jako funkcją aktywacji.

Klasyfikacja

Tak jak w przypadku regresji funkcja liniowa jest najgorsza. ReLU sprawdza się tylko przy jednej warstwie, jednakże jest gorszą funkcją aktywacji od pozostałych 2. Przy jednej warstwie ukrytej tanh jest lepszy od sigmoidu. Przy 2 i 3 warstwach ukrytych tanh też jest lepszy dla zbioru bardziej skomplikowanego – rings5. Natomiast dla zbioru mniej skomplikowanego – rings3 – sigmoid jest minimalnie lepszy, chociaż raczej te funkcje osiągają bardzo zbliżone rezultaty. Tanh szybciej zbiega do optimum lokalnego, w przypadku sigmoid zajmuje to więcej epok. Porównując sieci osiągają podobnie dobre wyniki niezależnie od liczby epok.

Dla problemu klasyfikacji najlepszą funkcją aktywacji jest tanh. Sieci z jedną warstwą ukrytą radzą sobie dobrze i najczęściej nie ma potrzeby dodawać więcej warstw.

3.7 Regularyzacja

3.7.1 Opis

Ostatnim eksperymentem było zaimplementowanie regularyzacji. Została zaimplementowana regularyzacja wag L1 i L2. L1 dodaje karę do funkcji celu, która jest proporcjonalna do wag, a w implementacji współczynnikiem proporcjonalności jest λ. L2 dodaje natomiast karę proporcjonalną do kwadratu wag ze współczynnikiem $\frac{\lambda}{2}$. Wartości tych współczynników powodują, że realnie przy każdym batchu wagi w przypadku L1 są pomniejszane o λ, a w przypadku L2 o $\lambda \cdot w$, gdzie w to waga poprzednia.

Eksperyment został przeprowadzony na następujących zbiorach danych (1 do regresji i 3 do klasyfikacji):

1. multimodal-sparse do regresji, który ma 40 obserwacji
2. rings3-balance, który ma 2060 obserwacji z 2 klasami po 1000 obserwacji i 60 z ostatniej klasy
3. ring5-sparse, który ma 200 obserwacji po 40 z 5 klas
4. xor3-balance, który ma 1050 obserwacji: 1000 jednej klasy i 50 drugiej

W ramach każdego zbioru danych zostały sprawdzone uczenie bez regularyzacji oraz uczenie z regularyzacjami L1 i L2, każde z nich z parametrem $\lambda \in 0.001, 0.0001, 0.00001$. Każda sieć miała 2 warstwy ukryte i funkcję aktywacji sigmoid. Każda warstwa ukryta miała 50 neuronów. Użyty learning rate wynosił $\eta = 0.01$.

Poza regularyzacją również zostało zastosowane zatrzymywane uczenie sieci jeśli w ciągu kolejnych 100 iteracji nie było poprawy najlepszego wyniku. W ten sposób każde uczenie sieci zatrzymało się samo przed granicą 10000 iteracji.

Na wykresie 28 zostały przedstawione wyniki i MSE dla regresji oraz na wykresach 29, 30 i 31 zostały pokazane wyniki działania sieci w postaci accuracy dla problemu klasyfikacji wraz z zaprezentowanym zbiorem danych.

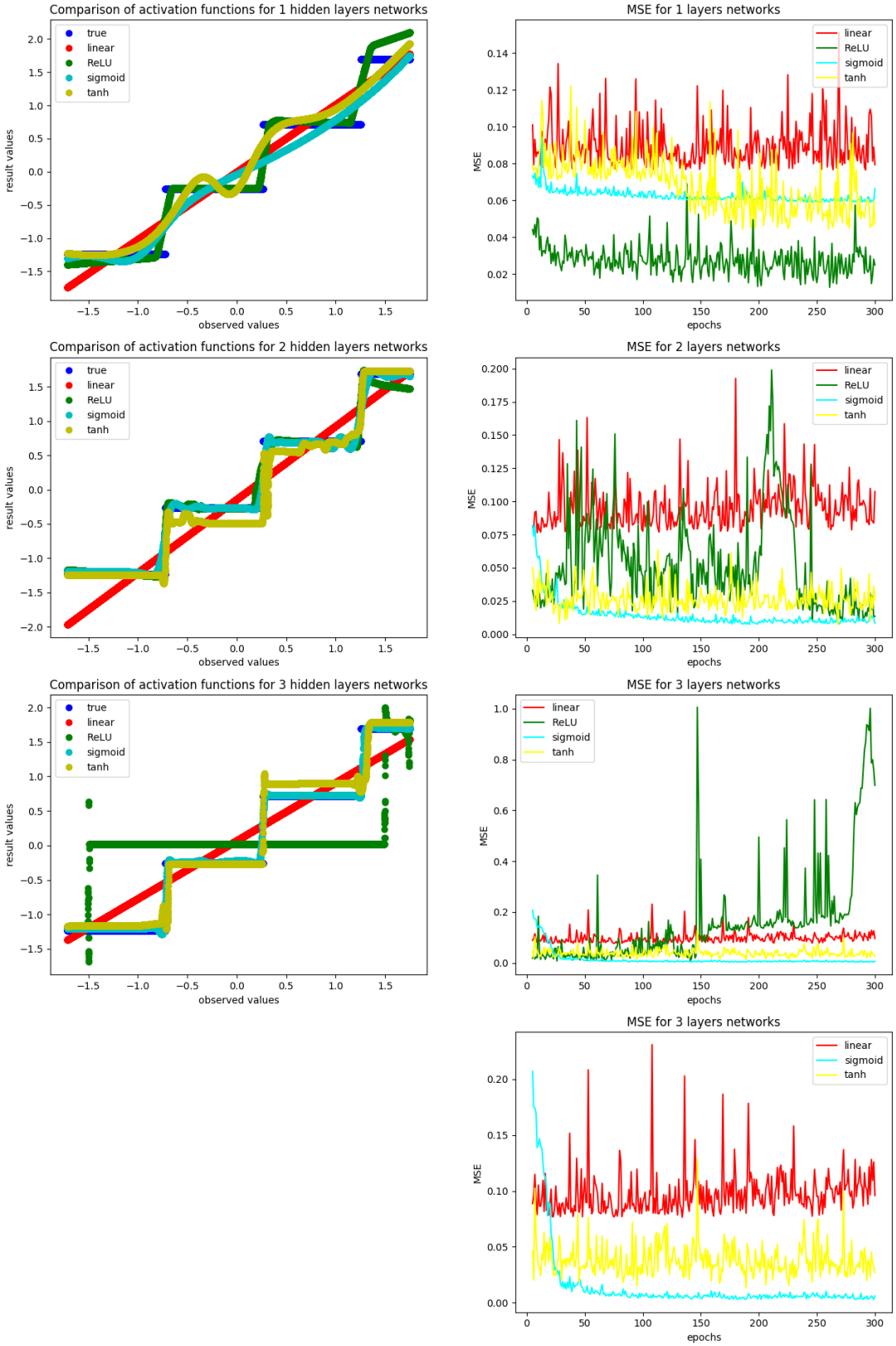
3.7.2 Wnioski

W wielu przypadkach regularyzacja dała pozytywny skutek. Jednakże jedynie dla niewielu λ jakość predykcji na zbiorze testowym była lepsza niż w przypadku modelu bez regularyzacji.

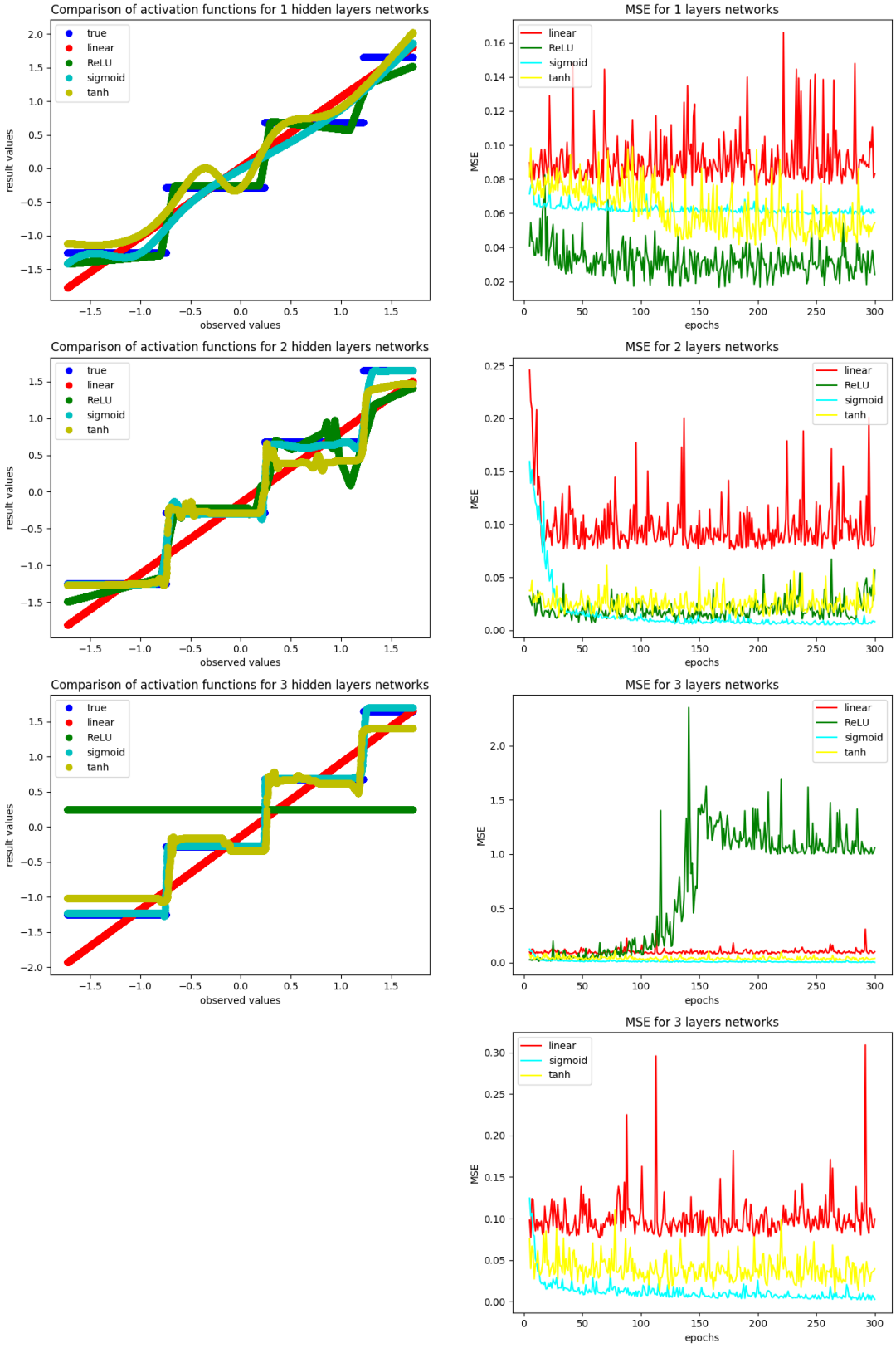
Okazało się, że dla bardzo rzadkich stopni jedna z regularyzacji poradziła sobie dużo lepiej niż model bez żadnej regularyzacji wag. W tym przypadku model "wychwycił" prawie wszystkie stopnie i przynajmniej częściowo się do nich dopasował. Takim rodzajem regularyzacji była regularyzacja L1 z $\lambda = 0.001$

W przypadku klasyfikacji wyniki nie różniły się tak bardzo jak w przypadku regresji, jednakże w każdym zbiorze istniała λ i rodzaj regularyzacji (L1/ L2) dająca lepszy wynik niż bazowy model.

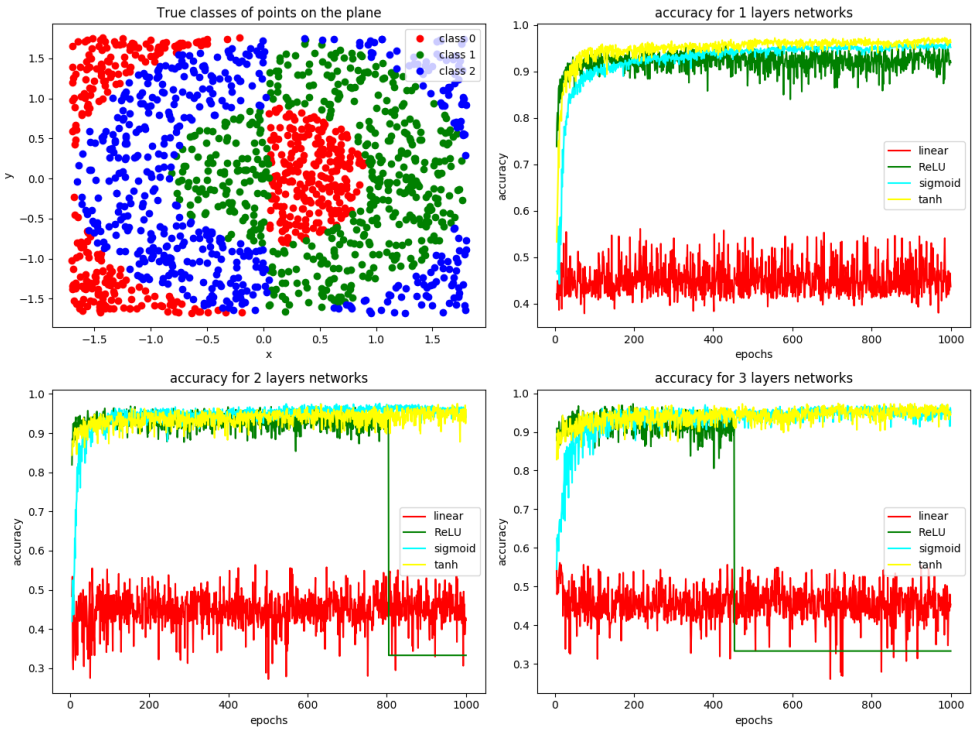
Generalnie $\lambda = 0.0001 \vee \lambda = 0.00001$ radzą sobie najlepiej. Także regularyzacja L1 jest zazwyczaj lepsza niż L2.



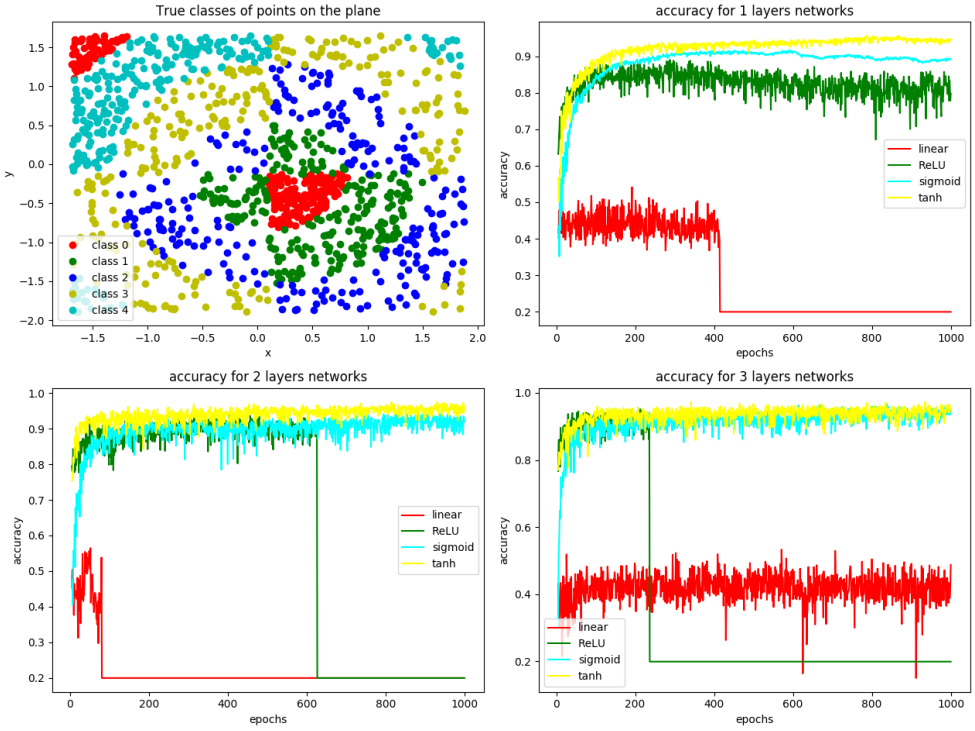
Rysunek 24: Multimodal-large



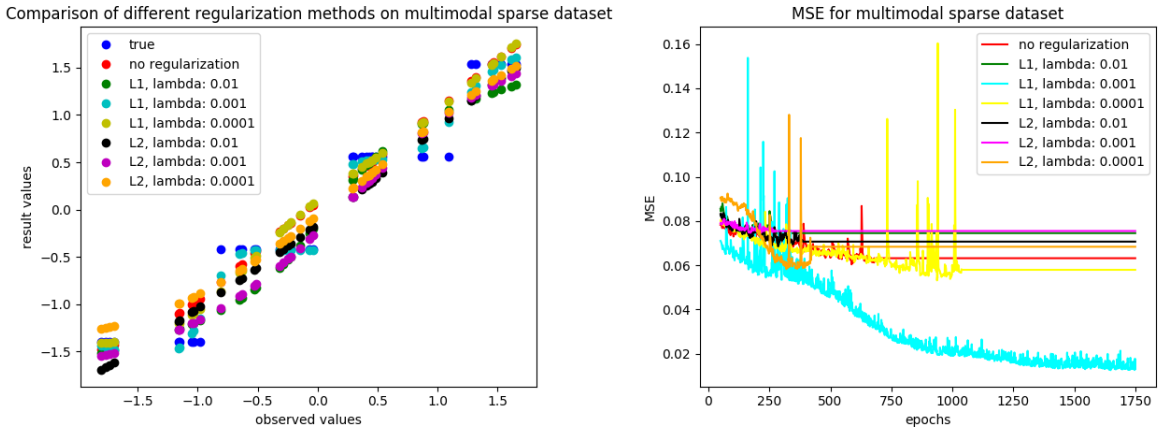
Rysunek 25: Steps-large



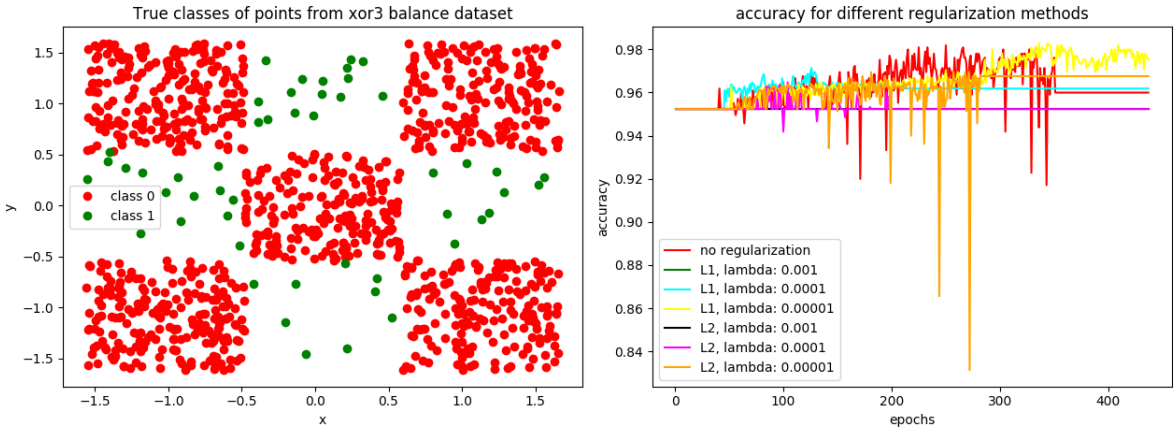
Rysunek 26: Rings3-regular



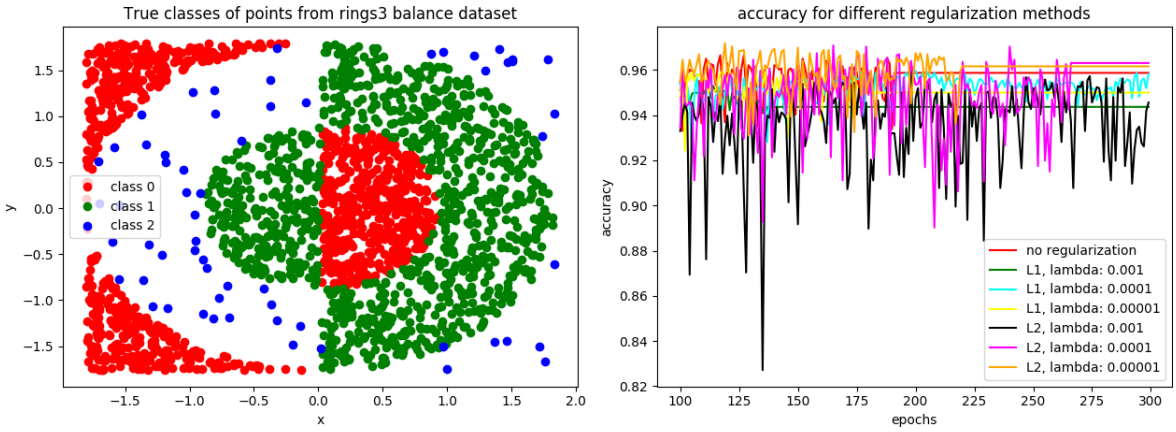
Rysunek 27: Rings5-regular



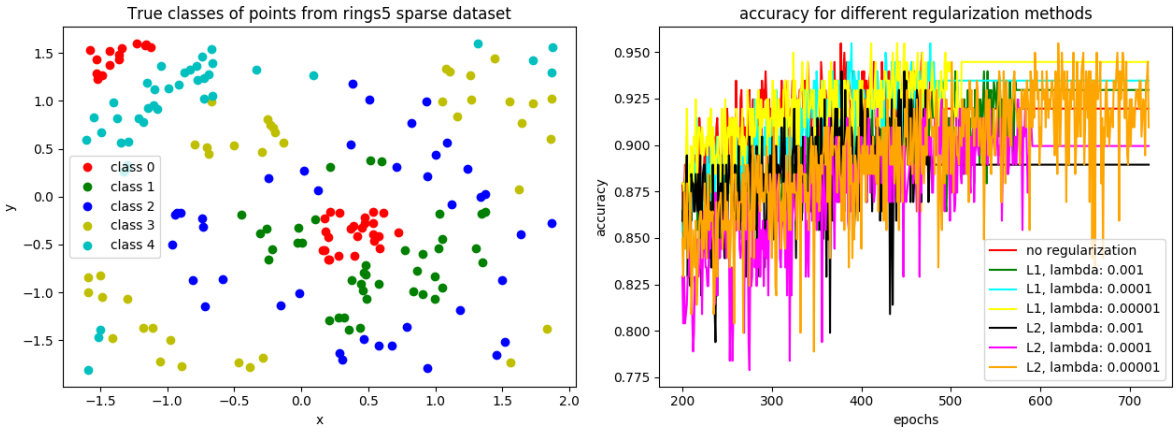
Rysunek 28: Multimodal-sparse



Rysunek 29: Xor3-balance



Rysunek 30: Rings3-balance



Rysunek 31: Rings5-sparse