Sprawozdanie z laboratorium
Perceptron Wielowarstwowy (MLP)

Bogumiła Okrojek
Numer indeksu: 327299

15 kwietnia 2025

1. Krótki opis tematu laboratoriów

Celem laboratoriów było zapoznanie się z zasadą działania perceptronu wielowarstwowego (MLP – Multi-Layer Perceptron), jego budową oraz procesem uczenia przy użyciu propagacji wstecznej. W ramach zajęć implementowane były kolejne elementy sieci: architektura, funkcje aktywacji, propagacja wsteczna, moment, RMSProp, softmax, różne funkcje aktywacji, mechanizmy regularyzacji i przeciwdziałania przeuczeniu.

2. Opis wykonanej pracy i wyniki eksperymentów

NN1: Bazowa implementacja MLP

Zaimplementowano sieć MLP z możliwością konfiguracji liczby warstw, liczby neuronów w każdej warstwie, funkcji aktywacji oraz inicjalizacji wag. W tej części wykorzystano funkcję sigmoidalną jako funkcję aktywacji wewnętrznych warstw oraz liniową funkcję aktywacji na wyjściu.

Przeprowadzono eksperymenty na dwóch zbiorach danych:

- **square-simple**
- **steps-large**

Zbiór: square-simple

Dla zbioru **square-simple** stworzono sieć o architekturze $[1, 5, 1]$, czyli jednowarstwową z pięcioma neuronami ukrytymi. Wagi i biasy zostały ustawione ręcznie w celu dopasowania kształtu wyjścia do danych. Użyto następujących parametrów:

$$
\begin{aligned}
W_0 &= \begin{bmatrix} 1.32 & -1.3 & 0.2 & 1.1 & -0.8 \end{bmatrix} \\
W_1 &= \begin{bmatrix} 450 \\ 550 \\ 550 \\ 550 \\ 550 \end{bmatrix} \\
b_0 &= \begin{bmatrix} -2.4 & -2.4 & -1.4 & -2.4 & -1.7 \end{bmatrix} \\
b_1 &= \begin{bmatrix} -452 \end{bmatrix}
\end{aligned}
$$

Wyniki:

- MSE na zbiorze treningowym: 6.49
- MSE na zbiorze testowym: 7.60

Zbiór: steps-large

Dla zbioru **steps-large** stworzono głębszą sieć o architekturze $[1, 5, 5, 1]$, umożliwiającą lepsze odwzorowanie wielomodalnych danych. Ręcznie dobrane wagi:

$$
\begin{aligned}
W_0 &= \begin{bmatrix} -44.5 & -33.2 & 52 & -58.6 & 32.3 \end{bmatrix} \\
W_1 &= \begin{bmatrix} -2.6 & -5.6 & -2.8 & -34.6 & -2.5 \\ -1.4 & 5.3 & -2.4 & -28.4 & -2.4 \\ 16.8 & -1.8 & 2.8 & 3.3 & 13.2 \\ -1.6 & -0.5 & -45.4 & 11.8 & -3 \\ 4 & -0.3 & 2.4 & 2.9 & 13.8 \end{bmatrix} \\
W_2 &= \begin{bmatrix} 38.8 \\ 35.1 \\ 78.6 \\ 79.3 \\ 44.5 \end{bmatrix} \\
b_0 &= \begin{bmatrix} -22.4 & -16.8 & -77.9 & 29 & -48.3 \end{bmatrix} \\
b_1 &= \begin{bmatrix} -9.8 & -2.1 & 20.4 & 16.9 & -13.7 \end{bmatrix} \\
b_2 &= \begin{bmatrix} -81.6 \end{bmatrix}
\end{aligned}
$$

Wyniki:

- MSE na zbiorze treningowym: 6.77
- MSE na zbiorze testowym: 4.01

Powyższe wyniki pokazują, że odpowiednie ręczne dobranie wag w niewielkiej sieci może prowadzić do satysfakcjonującej aproksymacji funkcji na obu zbiorach, bez konieczności stosowania uczenia maszynowego.

NN2: Implementacja propagacji wstecznej błędu

W ramach tego laboratorium zaimplementowano uczenie sieci neuronowej przy użyciu algorytmu propagacji wstecznej błędu (backpropagation). Wagi sieci inicjalizowano losowo z rozkładu jednostajnego na przedziale $[0, 1]$. Dodatkowo zaimplementowano mechanizm wizualizacji wartości wag w czasie uczenia, co pozwalało na analizę i diagnostykę problemów w przypadku trudności z konwergencją.

Zaimplementowano dwie wersje aktualizacji wag:

- aktualizacja po przejściu całego zbioru uczącego (full-batch),
- aktualizacja po każdej porcji danych (mini-batch).

Sieci przetestowano na czterech zestawach danych: `square-large`, `steps-large`, `steps-small` oraz `multimodal-large`. Dla każdego zbioru porównano efektywność obu wariantów uczenia, zatrzymując proces po osiągnięciu założonego progu błędu MSE na zbiorze treningowym.

Eksperyment 1: `square-large`, sieć [1,10,1] Uczenie zatrzymano po osiągnięciu $\text{MSE}_{\text{train}} < 2$.

- Full-batch: 1 532 330 epok, $\text{MSE}_{\text{test}} = 3.49$
- Mini-batch: 1 525 749 epok, $\text{MSE}_{\text{test}} = 3.49$

Eksperyment 2: `multimodal-large`, sieć [1,20,1] Uczenie zatrzymano po osiągnięciu $\text{MSE}_{\text{train}} < 40$.

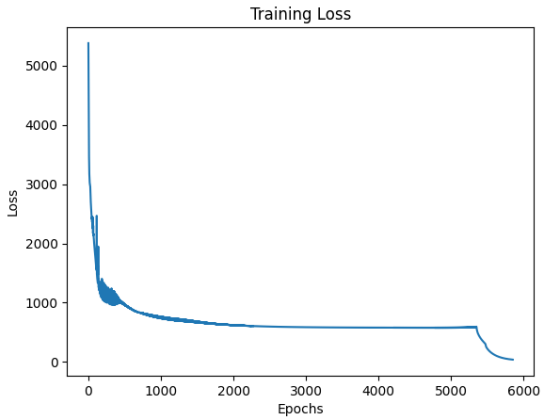
- Full-batch: 5 850 epok, $\text{MSE}_{\text{test}} = 34.98$
- Mini-batch: 1 916 epok, $\text{MSE}_{\text{test}} = 39.36$

Eksperyment 3: `steps-large`, sieć [1,5,5,1] Uczenie zatrzymano po osiągnięciu $\text{MSE}_{\text{train}} < 3.5$.

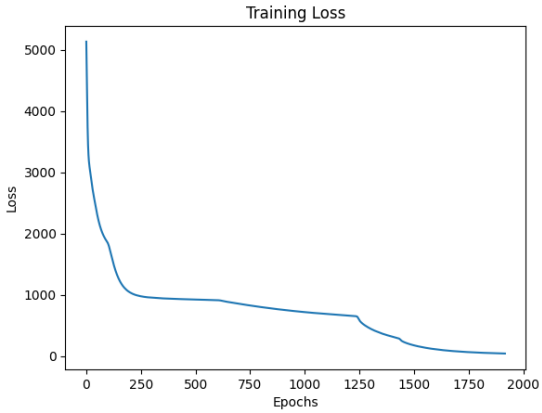
- Full-batch: 251 541 epok, $\text{MSE}_{\text{test}} = 2.47$
- Mini-batch: 86 522 epok, $\text{MSE}_{\text{test}} = 3.44$

Eksperyment 4: `steps-small`, sieć [1,4,4,1] Uczenie zatrzymano po osiągnięciu $\text{MSE}_{\text{train}} < 0.1$.

- Full-batch: 411 581 epok, $\text{MSE}_{\text{test}} = 45.66$
- Mini-batch: 376 814 epok, $\text{MSE}_{\text{test}} = 44.69$



Rysunek 1: Uczenie na całym zbiorze - zbiór multimodal large



Rysunek 2: Uczenie na batch - zbiór multimodal large

Rysunek 3: Porównanie przebiegu błędu treningowego (Train Loss) dla dwóch wariantów uczenia: na całym zbiorze i na batch.

Wnioski:

- Mini-batch learning w większości przypadków pozwalał na znaczące skrócenie liczby epok potrzebnych do osiągnięcia celu, szczególnie widoczne było to dla bardziej złożonych zbiorów jak `multimodal-large` czy `steps-large`.

- Full-batch learning dawał w niektórych przypadkach nieco niższe wartości MSE na zbiorze testowym (np. `steps-large`), jednak różnice te nie były istotne.
- Dla zbiorów prostszych lub o mniejszej liczbie przykładów (np. `steps-small`), liczba wymaganych epok była bardzo duża w obu trybach, a końcowy błąd testowy wysoki, co może sugerować problem z architekturą lub konieczność zastosowania lepszej metody inicjalizacji wag.
- Wizualizacja wag pozwalała zaobserwować, że w wielu przypadkach wagi ulegały stagnacji, co może wskazywać na konieczność zastosowania dodatkowych technik optymalizacyjnych (np. moment, RMSProp) lub innej funkcji aktywacji.
- Ogólnie rzecz biorąc, metoda mini-batch zapewniała lepszą efektywność uczenia i szybszą konwergencję, szczególnie w większych problemach.

NN3: Moment i RMSProp

Zaimplementowano dwie techniki wspomagające optymalizację: moment (momentum) oraz RMSProp. Celem było zbadanie wpływu tych metod na czas i jakość procesu uczenia sieci neuronowych. Przeprowadzono eksperymenty na trzech zbiorach danych (`square-large`, `steps-large`, `multimodal-large`), korzystając z tych samych architektur sieci dla każdego zbioru. Dla każdego przypadku porównano trzy konfiguracje treningu:

- bez dodatkowych technik optymalizacji,
- z momentem = 0.9,
- z RMSProp (również z momentem = 0.9).

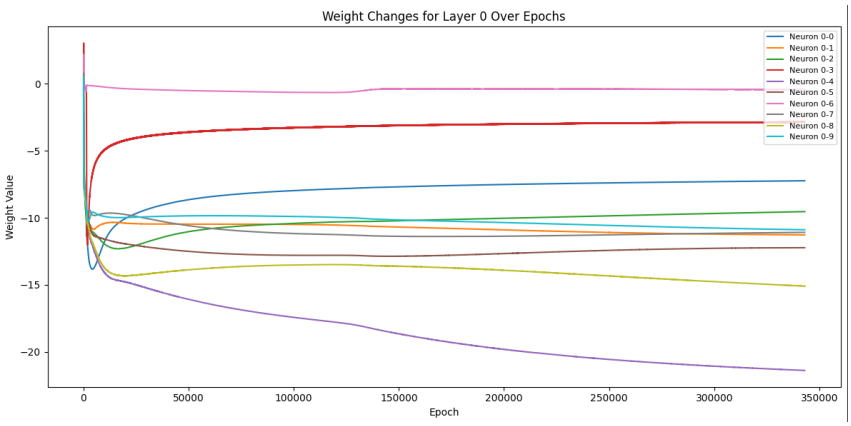
Uczenie zatrzymywano, gdy osiągnięto określony poziom błędu MSE na zbiorze treningowym.

Eksperyment 1: `square-large`, sieć [1,10,1] Uczenie zatrzymano po osiągnięciu $\text{MSE}_{\text{train}} < 1$.

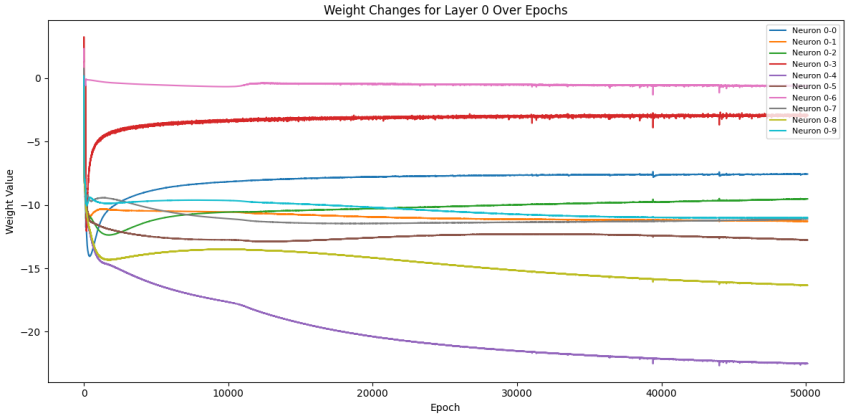
- Bez ulepszeń: 343 057 epok, $\text{MSE}_{\text{test}} = 261.18$
- Z momentem: 50 116 epok, $\text{MSE}_{\text{test}} = 268.22$
- Z RMSProp: 221 605 epok, $\text{MSE}_{\text{test}} = 258.88$

Eksperyment 2: `steps-large`, sieć [1,5,5,1] . Uczenie zatrzymano po osiągnięciu $\text{MSE}_{\text{train}} < 2.5$.

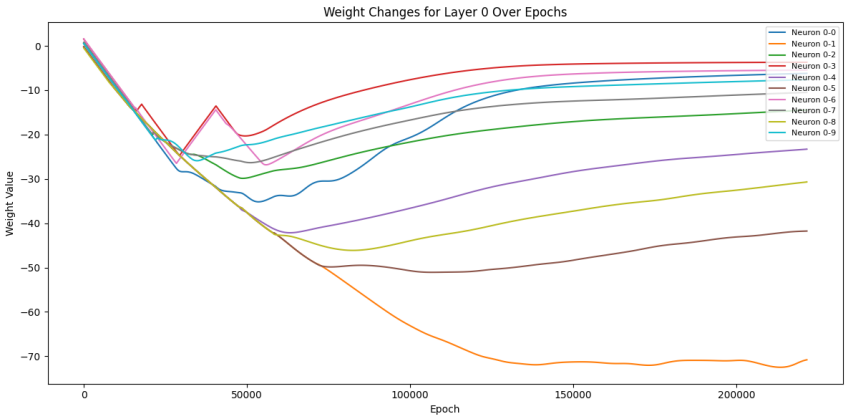
- Bez ulepszeń: 164 393 epok, $\text{MSE}_{\text{test}} = 2.67$
- Z momentem: 41 398 epok, $\text{MSE}_{\text{test}} = 1.88$
- Z RMSProp: 309 773 epok, $\text{MSE}_{\text{test}} = 2.27$



Bez ulepszeń



Z momentem



Z RMSProp

Rysunek 4: Zmiany wartości wag w warstwie ukrytej podczas uczenia na zbiorze `square-large` dla trzech metod optymalizacji. Każda linia przedstawia ewolucję jednej wagi w czasie (epoki).

Eksperyment 3: `multimodal-large`, sieć [1,30,1] Uczenie zatrzymano po osiągnięciu $\text{MSE}_{\text{train}} < 9$.

- Bez ulepszeń: 120 072 epok, $\text{MSE}_{\text{test}} = 3.77$
- Z momentem: 5 844 epok, $\text{MSE}_{\text{test}} = 4.25$

- Z RMSProp: 165 900 epok, $\text{MSE}_{\text{test}} = 4.01$

Wnioski:

- Zastosowanie momentu znacząco przyspieszało proces uczenia we wszystkich przypadkach — liczba epok potrzebna do osiągnięcia zakładanego poziomu błędu była od kilku do nawet kilkunastu razy mniejsza niż przy uczeniu bez ulepszeń.
- RMSProp dawał mieszane rezultaty: w niektórych przypadkach przyspieszał uczenie (np. *square-large*), w innych prowadził do znacznie większej liczby epok (np. *steps-large*), co pokazuje jego większą wrażliwość na specyfikę danych i architektury sieci.
- W przypadku bardziej złożonych zbiorów (np. *multimodal-large*) moment umożliwiał wyjątkowo szybkie osiągnięcie zadanego progu błędu, jednak czasami kosztem nieco wyższego błędu na zbiorze testowym, co może sugerować słabszą generalizację.
- Końcowe wartości MSE na zbiorze testowym były porównywalne niezależnie od zastosowanej metody optymalizacji, choć w niektórych przypadkach nieco lepsze wyniki uzyskiwało uczenie bez ulepszeń lub RMSProp.
- Ogólnie można stwierdzić, że moment to skuteczna i bezpieczna technika przyspieszająca uczenie modeli MLP, natomiast RMSProp może wymagać dokładniejszego dostrojenia hiperparametrów, aby osiągnąć optymalne efekty.

NN4: Softmax i klasyfikacja

Zaimplementowano funkcję aktywacji softmax oraz jej wersję propagacji wstecznej, umożliwiającą efektywne uczenie modeli MLP do zadań klasyfikacyjnych, zwłaszcza wieloklasowych. Przeprowadzono eksperymenty na trzech różnych zbiorach danych, porównując skuteczność modelu przy użyciu dwóch wariantów funkcji aktywacji w warstwie wyjściowej: *sigmoid* oraz *softmax*.

Dla każdego zbioru zastosowano tę samą architekturę sieci i identyczne warunki zatrzymania treningu.

Zbiory i konfiguracje eksperymentów:

- **rings3-regular**, sieć $[2, 8, 6, 4, 3]$, zatrzymano po osiągnięciu $\text{F1}_{train} > 0.85$
  - Sigmoid: epoka 69 887, $\text{F1}_{test} = 0.71$
  - Softmax: epoka 49 465, $\text{F1}_{test} = 0.72$
- **easy**, sieć $[2, 5, 2]$, zatrzymano po 1 000 000 epok
  - Sigmoid: $\text{F1}_{train} = 0.999$, $\text{F1}_{test} = 0.997$
  - Softmax: $\text{F1}_{train} = 0.999$, $\text{F1}_{test} = 0.998$
- **xor3**, sieć $[2, 32, 16, 8, 4, 2]$, zatrzymano po osiągnięciu $\text{F1}_{train} > 0.97$
  - Sigmoid: epoka 120 819, $\text{F1}_{test} = 0.85$
  - Softmax: epoka 474 415, $\text{F1}_{test} = 0.85$

Wnioski:

- Funkcja softmax okazała się kluczowa przy klasyfikacji wieloklasowej — szczególnie w zbiorze *rings3-regular*, gdzie jej zastosowanie umożliwiło szybsze osiągnięcie wymaganego poziomu F1 bez pogorszenia skuteczności na zbiorze testowym.
- W przypadku łatwych zbiorów danych (np. *easy*), oba podejścia — zarówno sigmoid, jak i softmax — osiągały niemal idealną skuteczność, z minimalnymi różnicami w wynikach testowych.
- W zbiorze *xor3* różnice w jakości klasyfikacji były niewielkie, ale uczenie z funkcją sigmoid było znacznie szybsze, co sugeruje, że dla niektórych specyficznych problemów binarnych może być ono wystarczające.
- Ogólnie rzecz biorąc, softmax sprawdza się najlepiej w zadaniach wieloklasowych, zapewniając interpretowalne wyjścia w postaci rozkładu prawdopodobieństwa oraz dobrą stabilność treningu.
- W zadaniach binarnych wybór pomiędzy sigmoid a softmax może mieć mniejsze znaczenie, ale softmax daje większą elastyczność, zwłaszcza w architekturach uniwersalnych, które mają działać zarówno dla klasyfikacji binarnej, jak i wieloklasowej.

NN5: Różne funkcje aktywacji

W tej części pracy testowano cztery popularne funkcje aktywacji: sigmoid, ReLU, tanh oraz funkcję liniową. Celem eksperymentów było zbadanie wpływu różnych funkcji aktywacji oraz liczby warstw ukrytych na skuteczność modeli MLP, zarówno w zadaniach regresyjnych, jak i klasyfikacyjnych.

Przeprowadzono eksperymenty na czterech różnych zbiorach danych: *multimodal-large*, *steps-large*, *rings3-regular* oraz *rings5-regular*. Dla każdego zbioru porównano skuteczność uczenia przy różnych funkcjach aktywacji i różnych architekturach sieci.

Eksperymenty:

- **Zbiór multimodal (regresja), sieć** $[1, 64, 1]$ – uczenie zatrzymano po 10 000 epokach:
  - Sigmoid: $\text{MSE}_{train} = 6.18$, $\text{MSE}_{test} = 2.23$
  - Linear: $\text{MSE}_{train} = 4398.22$, $\text{MSE}_{test} = 4433.71$
  - Tanh: $\text{MSE}_{train} = 5.03$, $\text{MSE}_{test} = 1.84$
  - ReLU: $\text{MSE}_{train} = 4792.69$, $\text{MSE}_{test} = 4807.56$
- **Zbiór multimodal (regresja), sieć** $[1, 128, 64, 1]$ – uczenie zatrzymano po 10 000 epokach:
  - Sigmoid: $\text{MSE}_{train} = 1167.85$, $\text{MSE}_{test} = 1272.23$
  - Linear: $\text{MSE}_{train} = 4398.22$, $\text{MSE}_{test} = 4433.71$
  - Tanh: $\text{MSE}_{train} = 11.60$, $\text{MSE}_{test} = 12.27$
  - ReLU: $\text{MSE}_{train} = 3940.52$, $\text{MSE}_{test} = 3844.23$
- **Zbiór multimodal (regresja), sieć** $[1, 256, 128, 64, 1]$ – uczenie zatrzymano po 10 000 epokach:

  - Sigmoid: $\text{MSE}_{train} = 1497.14$, $\text{MSE}_{test} = 1645.73$
  - Linear: $\text{MSE}_{train} = 4398.22$, $\text{MSE}_{test} = 4433.71$
  - Tanh: $\text{MSE}_{train} = 632.27$, $\text{MSE}_{test} = 667.64$
  - ReLU: $\text{MSE}_{train} = 5.62$, $\text{MSE}_{test} = 1.83$

- **Zbiór steps-large (regresja), sieć** $[1, 5, 5, 1]$ – uczenie zatrzymano po 10 000 epokach:
  - Sigmoid: $\text{MSE}_{train} = 5.27$, $\text{MSE}_{test} = 4.68$
  - Tanh: $\text{MSE}_{train} = 3.20$, $\text{MSE}_{test} = 0.49$

- **Zbiór rings3-regular (klasyfikacja), sieć** $[2, 8, 6, 4, 3]$ – uczenie zatrzymano po 10 000 epokach:
  - Tanh: $\text{F1}_{train} = 0.73$, $\text{F1}_{test} = 0.68$
  - ReLU: $\text{F1}_{train} = 0.79$, $\text{F1}_{test} = 0.77$

- **Zbiór rings5-regular (klasyfikacja), sieć** $[2, 64, 32, 16, 5]$ – uczenie zatrzymano po 10 000 epokach:
  - Tanh: $\text{F1}_{train} = 0.74$, $\text{F1}_{test} = 0.59$
  - ReLU: $\text{F1}_{train} = 0.77$, $\text{F1}_{test} = 0.62$

Wnioski:

- W zadaniach regresyjnych (np. *multimodal*) funkcja ReLU okazała się najbardziej efektywna, szczególnie przy większych sieciach (np. $[1, 256, 128, 64, 1]$), osiągając bardzo niski błąd testowy ($\text{MSE}_{test} = 1.83$). Natomiast funkcja *tanh* również osiągnęła dobre wyniki, ale była mniej stabilna przy większych architekturach.

- Funkcja *sigmoid* miała znacząco wyższy błąd na zbiorze testowym, szczególnie w przypadku bardziej złożonych architektur, co sugeruje, że jest mniej efektywna w zadaniach regresyjnych w porównaniu do ReLU i tanh.

- W zadaniach klasyfikacyjnych (np. *rings3-regular*, *rings5-regular*) ReLU również wykazała lepszą skuteczność niż tanh, osiągając wyższy F1-score na zbiorze testowym. ReLU zapewniała lepszą stabilność i skuteczność w zadaniach z wieloma klasami.

- Z kolei funkcja *tanh* okazała się lepsza w zadaniach regresyjnych przy mniejszych sieciach i w przypadku łatwiejszych zbiorów, takich jak *steps-large*, gdzie MSE był znacząco mniejszy niż dla sigmoid.

- Warto zauważyć, że funkcja liniowa miała bardzo wysokie błędy testowe, co sugeruje, że nie jest dobrym wyborem do równań nieliniowych, takich jak te w przypadku zadania regresji na zbiorze *multimodal*.

- Podsumowując, dla zadań regresyjnych i klasyfikacyjnych najskuteczniejsze okazały się funkcje ReLU oraz tanh. ReLU sprawdza się lepiej w większych i bardziej złożonych sieciach, a tanh lepiej w prostszych przypadkach. Funkcja sigmoid ma ograniczoną skuteczność, szczególnie w zadaniach o większej złożoności, a funkcja liniowa nie jest polecana w tego typu zadaniach.

NN6: Przeciwdziałanie przeuczeniu

W tej części pracy testowano dwie techniki regularyzacji: L1 i L2, a także mechanizm *early stopping*, który polega na zatrzymaniu procesu uczenia, gdy błąd na zbiorze testowym przestaje się poprawiać przez określoną liczbę epok. Celem eksperymentów było zbadanie wpływu tych technik na skuteczność modeli MLP, zarówno w zadaniach regresyjnych, jak i klasyfikacyjnych.

Przeprowadzono eksperymenty na czterech różnych zbiorach danych: *multimodal-sparse*, *rings5-sparse*, *rings3-balance* oraz *xor3-balance*. Dla każdego zbioru porównano skuteczność uczenia z różnymi technikami regularyzacyjnymi oraz zastosowaniem mechanizmu *early stopping*.

Eksperymenty:

- **Zbiór multimodal-sparse (regresja), sieć** $[1, 64, 1]$ – uczenie zatrzymano po 500,000 epokach:
  - Bez ulepszeń: $\text{MSE}_{train} = 229.7384$, $\text{MSE}_{test} = 530.8289$
  - Regulacja wag L1 z $\lambda = 0.1$: $\text{MSE}_{train} = 229.3595$, $\text{MSE}_{test} = 529.3372$
  - Regulacja wag L2 z $\lambda = 0.1$: $\text{MSE}_{train} = 233.4405$, $\text{MSE}_{test} = 526.5681$
  - Regulacja wag L1 z $\lambda = 1$: $\text{MSE}_{train} = 231.5205$, $\text{MSE}_{test} = 518.1967$
  - Regulacja wag L2 z $\lambda = 1$: $\text{MSE}_{train} = 286.5101$, $\text{MSE}_{test} = 570.5618$
  - Mechanizm early stopping: uczenie przerwano po 274,159 epokach, $\text{MSE}_{train} = 247.1653$, $\text{MSE}_{test} = 563.0493$
- **Zbiór rings5-sparse (klasyfikacja), sieć** $[2, 32, 16, 8, 5]$ – uczenie zatrzymano po 50,000 epokach:
  - Bez ulepszeń: $\text{F1}_{train} = 0.7210$, $\text{F1}_{test} = 0.5852$
  - Regulacja wag L1 z $\lambda = 0.1$: $\text{F1}_{train} = 0.7305$, $\text{F1}_{test} = 0.6046$
  - Regulacja wag L2 z $\lambda = 0.1$: $\text{F1}_{train} = 0.7283$, $\text{F1}_{test} = 0.6578$
  - Regulacja wag L1 z $\lambda = 1$: $\text{F1}_{train} = 0.7387$, $\text{F1}_{test} = 0.6039$
  - Regulacja wag L2 z $\lambda = 1$: $\text{F1}_{train} = 0.7395$, $\text{F1}_{test} = 0.6710$
  - Mechanizm early stopping: uczenie przerwano po 12,394 epokach, $\text{F1}_{train} = 0.5948$, $\text{F1}_{test} = 0.4988$
- **Zbiór rings3-balance (klasyfikacja), sieć** $[2, 8, 6, 4, 3]$ – uczenie zatrzymano po 100,000 epokach:
  - Bez ulepszeń: $\text{F1}_{train} = 0.7316$, $\text{F1}_{test} = 0.5810$
  - Regulacja wag L1 z $\lambda = 0.1$: $\text{F1}_{train} = 0.8196$, $\text{F1}_{test} = 0.7004$
  - Regulacja wag L2 z $\lambda = 0.1$: $\text{F1}_{train} = 0.8127$, $\text{F1}_{test} = 0.6995$
  - Mechanizm early stopping: uczenie przerwano po 3,662 epokach, $\text{F1}_{train} = 0.4317$, $\text{F1}_{test} = 0.3373$
- **Zbiór xor3-balance (klasyfikacja), sieć** $[2, 32, 16, 8, 4, 2]$ – uczenie zatrzymano po 50,000 epokach:

- Bez ulepszeń: $\mathrm{F1}_{train} = 0.8179$, $\mathrm{F1}_{test} = 0.5846$
- Regulacja wag L1 z $\lambda = 0.1$: $\mathrm{F1}_{train} = 0.4878$, $\mathrm{F1}_{test} = 0.3559$
- Regulacja wag L2 z $\lambda = 0.1$: $\mathrm{F1}_{train} = 0.4878$, $\mathrm{F1}_{test} = 0.3559$
- Regulacja wag L1 z $\lambda = 1$: $\mathrm{F1}_{train} = 0.4878$, $\mathrm{F1}_{test} = 0.3559$
- Regulacja wag L2 z $\lambda = 1$: $\mathrm{F1}_{train} = 0.4878$, $\mathrm{F1}_{test} = 0.3559$
- Mechanizm early stopping: uczenie przerwano po 1,001 epokach, $\mathrm{F1}_{train} = 0.4878$, $\mathrm{F1}_{test} = 0.3559$

Wnioski:

- **Regularyzacja L1** z $\lambda = 0.1$ okazała się efektywna w poprawie wyników testowych w porównaniu do modelu bez ulepszeń, szczególnie w zadaniach regresyjnych, jak w przypadku zbioru *multimodal-sparse*.
- **Regularyzacja L2** z $\lambda = 0.1$ również poprawiła wyniki na zbiorze testowym, jednak nie zawsze była bardziej skuteczna niż L1. Wysoka wartość $\lambda = 1$ prowadziła do gorszych wyników, zwłaszcza w przypadku zadania regresyjnego.
- **Mechanizm early stopping** okazał się bardzo przydatny, szczególnie w przypadkach, gdzie model zaczynał przeuczać się po długim czasie treningu, jak w przypadku zbioru *rings5-sparse*. Mechanizm pozwolił na zatrzymanie treningu po osiągnięciu stagnacji w wynikach.
- **Wnioski ogólne:** Zastosowanie regularyzacji L1 i L2 oraz mechanizmu early stopping skutecznie poprawia wyniki na zbiorach testowych, redukując ryzyko przeuczenia, szczególnie w zadaniach klasyfikacyjnych.

3. Wnioski

W trakcie realizacji laboratorium uzyskano kilka istotnych wniosków, które mogą pomóc w efektywnym wykorzystaniu perceptronów wielowarstwowych (MLP) w praktycznych zadaniach:

1. **Wybór funkcji aktywacji** – Funkcja aktywacji ma kluczowy wpływ na efektywność sieci neuronowej. W zadaniach regresyjnych funkcje takie jak ReLU i tanh sprawdzają się najlepiej, natomiast w przypadku klasyfikacji binarnej funkcja sigmoid może być odpowiednia. Softmax natomiast okazał się niezbędny w zadaniach wieloklasowych, gdzie jego zdolność do przekształcania wyników sieci na prawdopodobieństwa klasyfikacji jest szczególnie przydatna.

2. **Rola optymalizacji** – Testowanie różnych metod optymalizacji, w tym propagacji wstecznej z pełną paczką (full-batch) oraz z mini-paczkami (mini-batch), pokazało, że mini-batch znacznie przyspiesza proces uczenia, przy zachowaniu dobrych wyników, zwłaszcza w przypadku większych zbiorów danych. Full-batch może jednak prowadzić do lepszej konwergencji w niektórych przypadkach.

3. **Zastosowanie momentu i RMSProp** – Zastosowanie momentu poprawiło czas konwergencji, szczególnie w przypadku bardziej złożonych danych, podczas gdy RMSProp przynosił mieszane wyniki. Choć moment przyspieszał proces uczenia, w niektórych przypadkach negatywnie wpływał na generalizację modelu. To sugeruje, że dobór parametrów optymalizatora ma istotny wpływ na wydajność modelu.

4. **Zrozumienie konwergencji i regularyzacja** – Wizualizacja wag sieci neuronowych pozwala lepiej zrozumieć proces konwergencji i pomaga zidentyfikować problemy z przeuczeniem. Regularyzacja, np. przez techniki takie jak dropout czy L2 regularization, jest ważnym aspektem, który może poprawić zdolność sieci do generalizacji, szczególnie w przypadku dużych i złożonych zbiorów danych.

5. **Znaczenie inicjalizacji wag** – Odpowiednia inicjalizacja wag może znacznie ułatwić proces uczenia. W przypadku prostych sieci, wstępne ustawienie wag może wystarczyć, ale w bardziej złożonych architekturach odpowiednia inicjalizacja stanowi klucz do szybszego i skutecznego uczenia.

6. **Przeciwdziałanie przeuczeniu** – Zastosowanie regularyzacji L1 i L2 oraz mechanizmu *early stopping* skutecznie poprawia wyniki na zbiorach testowych, redukując ryzyko przeuczenia, szczególnie w zadaniach klasyfikacyjnych.

Wnioski te podkreślają znaczenie odpowiedniego dostosowania parametrów modelu, zarówno w kontekście struktury sieci, jak i wybranych metod optymalizacji oraz funkcji aktywacji, aby uzyskać najlepsze rezultaty w zadaniach regresyjnych i klasyfikacyjnych.