Laboratorium 5
Wprowadzenie do sztucznej inteligencji

Agnieszka Głuszkiewicz

1 Wprowadzenie i cel zadania

Niniejsze sprawozdanie przedstawia implementację i analizę dwuwarstwowej sieci neuronowej rozwiązującej problem klasyfikacji binarnej. Zadanie polegało na określeniu, czy dwie liczby rzeczywiste $x_1, x_2 \in [-1, 1] \setminus \{0\}$ posiadają ten sam znak. W przypadku posiadania tego samego znaku, oczekiwana wartość wyjściowa wynosiła 1; w przeciwnym razie 0. Struktura zaimplementowanej sieci neuronowej jest następująca:

- Warstwa wejściowa: 2 neurony (odpowiadające za wejścia x_1 i x_2).
- Warstwa ukryta: 4 neurony.
- Warstwa wyjściowa: 1 neuron.

Przeprowadzone eksperymenty uwzględniały porównanie dwóch funkcji aktywacji:

- **Funkcja Sigmoidalna** ($\sigma(z) = \frac{1}{1+e^{-z}}$).
- **Funkcja ReLU** ($\mathrm{ReLU}(z) = \max(0, z)$).

Badałam również wpływ trzech metod normalizacji danych wejściowych na proces uczenia:

- Dane Nieznormalizowane.
- **Normalizacja L1** ($||\mathbf{x}||_1 = |x_1| + |x_2|$), z postacią znormalizowaną $\mathbf{x}_{\mathrm{norm\_L1}} = \frac{\mathbf{x}}{||\mathbf{x}||_1}$.
- **Normalizacja L2** ($||\mathbf{x}||_2 = \sqrt{x_1^2 + x_2^2}$), z postacią znormalizowaną $\mathbf{x}_{\mathrm{norm\_L2}} = \frac{\mathbf{x}}{||\mathbf{x}||_2}$.

Uczenie sieci odbywało się z wykorzystaniem algorytmu propagacji wstecznej, z funkcją kosztu średniokwadratowego błędu (MSE) i różnymi wartościami współczynnika uczenia (η).

2 Implementacja algorytmu propagacji wstecznej

Implementacja sieci neuronowej obejmuje metody: 'forward' (przetwarzanie w przód), 'backward' (propagacja wsteczna) oraz 'train' (zarządzanie procesem uczenia). Funkcje aktywacji (σ, ReLU) oraz ich pochodne zostały zaimplementowane jako metody pomocnicze. Wagi sieci są inicjalizowane małymi losowymi wartościami, natomiast biasy wartościami zerowymi.

2.1 Faza przetwarzania w przód (Forward Pass):

Dla pojedynczego wejścia $\mathbf{x} = [x_1, x_2]$:

1. **Obliczenia w warstwie ukrytej:** Dla każdego neuronu j w warstwie ukrytej (gdzie $j \in \{1, 2, 3, 4\}$), sumowane są ważone wejścia z dodatkiem biasu:

$$z_j^{(1)} = \sum_{i=1}^{2} w_{ji}^{(1)} x_i + b_j^{(1)}$$

Następnie, do wartości $z_j^{(1)}$ aplikowana jest wybrana funkcja aktywacji f (σ lub ReLU), co daje aktywację neuronu $a_j^{(1)}$:

$$a_j^{(1)} = f(z_j^{(1)})$$

2. **Obliczenia w warstwie wyjściowej:** Dla jedynego neuronu wyjściowego, podobnie obliczany jest ważony sygnał z warstwy ukrytej z dodatkiem biasu:

$$z^{(2)} = \sum_{j=1}^{4} w_j^{(2)} a_j^{(1)} + b^{(2)}$$

Aplikacja funkcji aktywacji f do $z^{(2)}$ daje przewidywane wyjście sieci $\hat{y}$:

$$\hat{y} = f(z^{(2)})$$

2.2 Faza propagacji wstecznej (Backward Pass):

Funkcja kosztu to błąd średniokwadratowy (MSE) dla pojedynczej próbki: $E = \frac{1}{2}(y-\hat{y})^2$, gdzie y jest rzeczywistą wartością docelową, a $\hat{y}$ przewidywaną przez sieć. Celem jest wyznaczenie gradientów $\frac{\partial E}{\partial w}$ i $\frac{\partial E}{\partial b}$ dla wszystkich wag i biasów.

1. **Obliczanie błędu (delty) dla warstwy wyjściowej:** Zaczynamy od obliczenia, jak bardzo błąd E zmienia się w zależności od ważonej sumy wejść do neuronu wyjściowego ($z^{(2)}$). Jest to tzw. "delta"dla warstwy wyjściowej, $\delta^{(2)}$. Stosujemy regułę łańcuchową:

$$\delta^{(2)} = \frac{\partial E}{\partial z^{(2)}} = \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z^{(2)}} = (\hat{y} - y) \cdot f'(z^{(2)})$$

$\delta^{(2)}$ mówi nam, jak duży jest "niespodziewany"błąd sieci ($\hat{y} - y$) i jak bardzo neuron był wrażliwy na zmianę swojego wejścia przed aktywacją ($f'(z^{(2)})$).

2. **Wyznaczenie gradientów dla wag i biasu warstwy wyjściowej:** Mając $\delta^{(2)}$, możemy obliczyć, jak każda waga i bias w warstwie wyjściowej wpływa na całkowity błąd E.

   - Gradient dla wagi $w_j^{(2)}$ (łączącej neuron j warstwy ukrytej z neuronem wyjściowym):

   $$\frac{\partial E}{\partial w_j^{(2)}} = \frac{\partial E}{\partial z^{(2)}} \cdot \frac{\partial z^{(2)}}{\partial w_j^{(2)}} = \delta^{(2)} \cdot a_j^{(1)}$$

   Gradient wagi zależy od błędu wyjściowego ($\delta^{(2)}$) i aktywacji neuronu ($a_j^{(1)}$), z którego wychodzi połączenie. Aktywny neuron, który przyczynił się do dużego błędu, będzie miał mocniej korygowaną wagę.

   - Gradient dla biasu $b^{(2)}$ neuronu wyjściowego:

   $$\frac{\partial E}{\partial b^{(2)}} = \frac{\partial E}{\partial z^{(2)}} \cdot \frac{\partial z^{(2)}}{\partial b^{(2)}} = \delta^{(2)} \cdot 1 = \delta^{(2)}$$

   Bias jest korygowany bezpośrednio o wartość błędu $\delta^{(2)}$, ponieważ jego wpływ na aktywację neuronu jest stały.

3. **Obliczanie błędu (delty) dla warstwy ukrytej:** Błąd z warstwy wyjściowej jest propagowany wstecz do warstwy ukrytej. Dla każdego neuronu j w warstwie ukrytej, jego błąd $\delta_j^{(1)}$ jest obliczany. Błąd ten zależy od wkładu tego neuronu w błąd warstwy wyjściowej oraz od pochodnej jego funkcji aktywacji.

$$\delta_j^{(1)} = \left(w_j^{(2)} \cdot \delta^{(2)} \right) \cdot f'(z_j^{(1)})$$

Błąd neuronu ukrytego jest "dziedziczony"z błędów neuronów warstw wyższych, do których jest połączony, i jest modyfikowany przez własną wrażliwość na zmiany.

4. **Wyznaczenie gradientów dla wag i biasów warstwy ukrytej:** Dla każdego połączenia z neuronu wejściowego i do neuronu ukrytego j, gradienty dla wag $w_{ji}^{(1)}$ i biasów $b_j^{(1)}$ są obliczane.

   - Gradient dla wagi $w_{ji}^{(1)}$ (łączącej wejście x_i z neuronem j warstwy ukrytej):

   $$\frac{\partial E}{\partial w_{ji}^{(1)}} = \frac{\partial E}{\partial z_j^{(1)}} \cdot \frac{\partial z_j^{(1)}}{\partial w_{ji}^{(1)}} = \delta_j^{(1)} \cdot x_i$$

   Gradient wagi zależy od błędu neuronu ukrytego ($\delta_j^{(1)}$) i wartości wejściowej (x_i), którą ta waga "przetwarza".

- Gradient dla biasu $b_j^{(1)}$ neuronu ukrytego j:

$$\frac{\partial E}{\partial b_j^{(1)}} = \frac{\partial E}{\partial z_j^{(1)}} \cdot \frac{\partial z_j^{(1)}}{\partial b_j^{(1)}} = \delta_j^{(1)} \cdot 1 = \delta_j^{(1)}$$

  Bias jest korygowany bezpośrednio o wartość błędu $\delta_j^{(1)}$, analogicznie do biasu w warstwie wyjściowej.

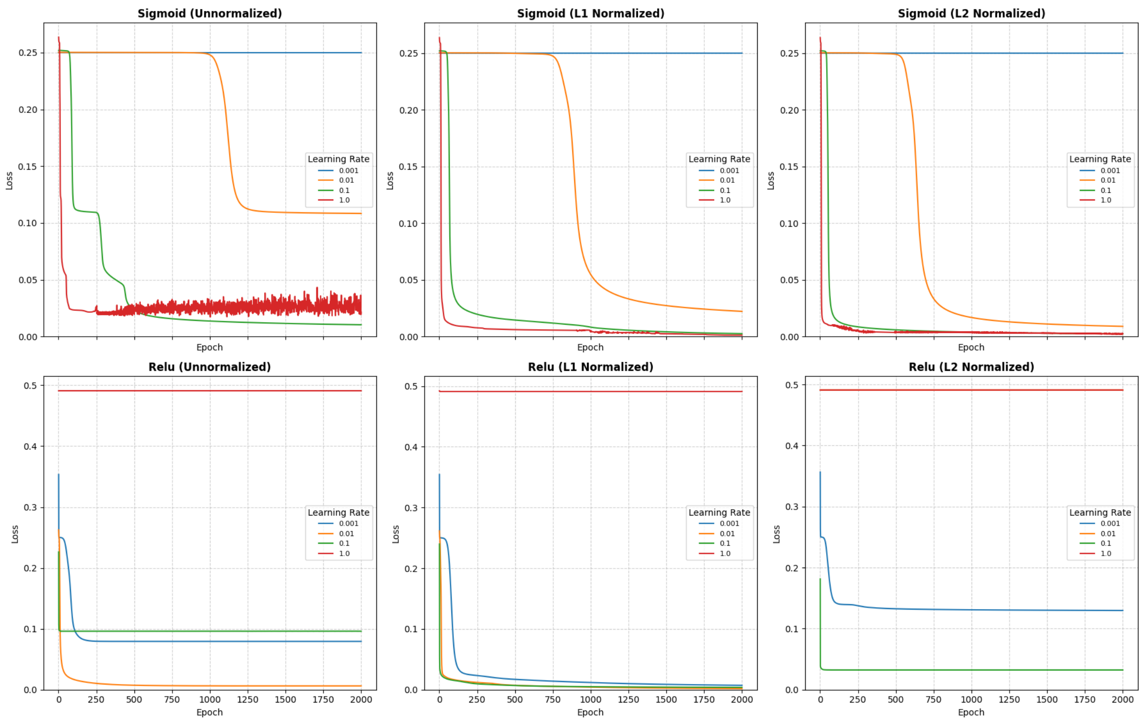
5. **Aktualizacja wag i biasów:** Wagi i biasy są korygowane w kierunku malejącym gradientu, z uwzględnieniem współczynnika uczenia (η). Ponieważ obliczone gradienty ($\frac{\partial E}{\partial w}$, $\frac{\partial E}{\partial b}$) wskazują kierunek wzrostu funkcji kosztu, musimy odejmować je, aby zminimalizować błąd.

$$w_{\text{new}} = w_{\text{old}} - \eta \frac{\partial E}{\partial w}$$
$$b_{\text{new}} = b_{\text{old}} - \eta \frac{\partial E}{\partial b}$$

   Ten proces aktualizacji jest powtarzany dla każdej próbki treningowej w ramach danej epoki, a następnie przez zadaną liczbę epok treningowych.

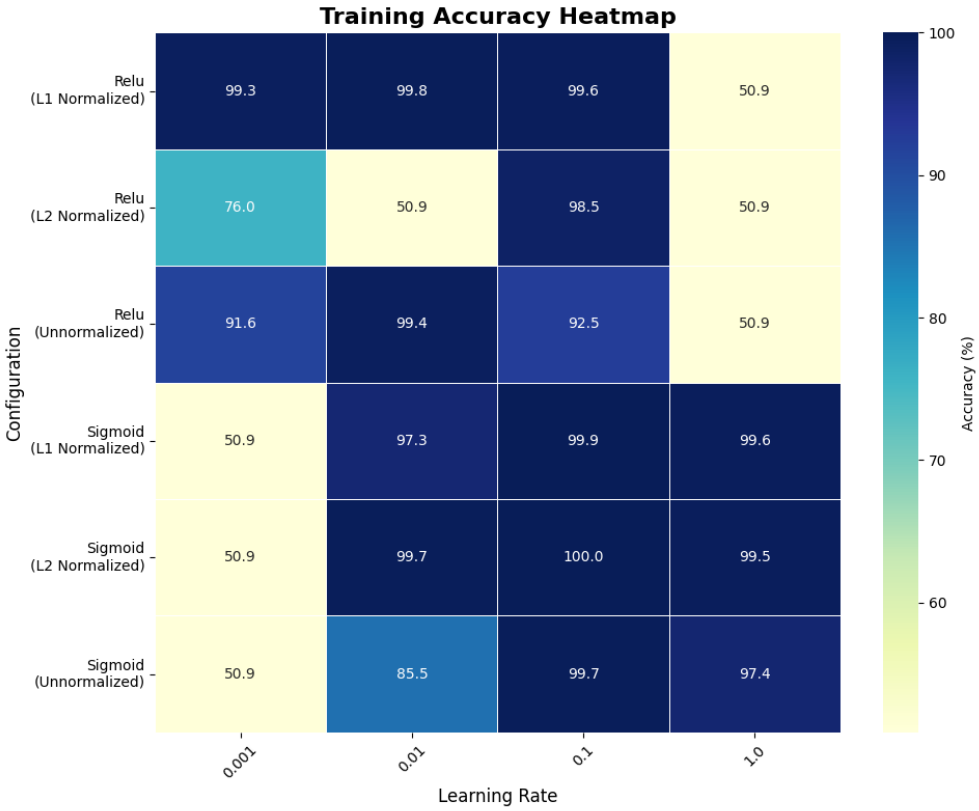
3 Analiza tempa uczenia

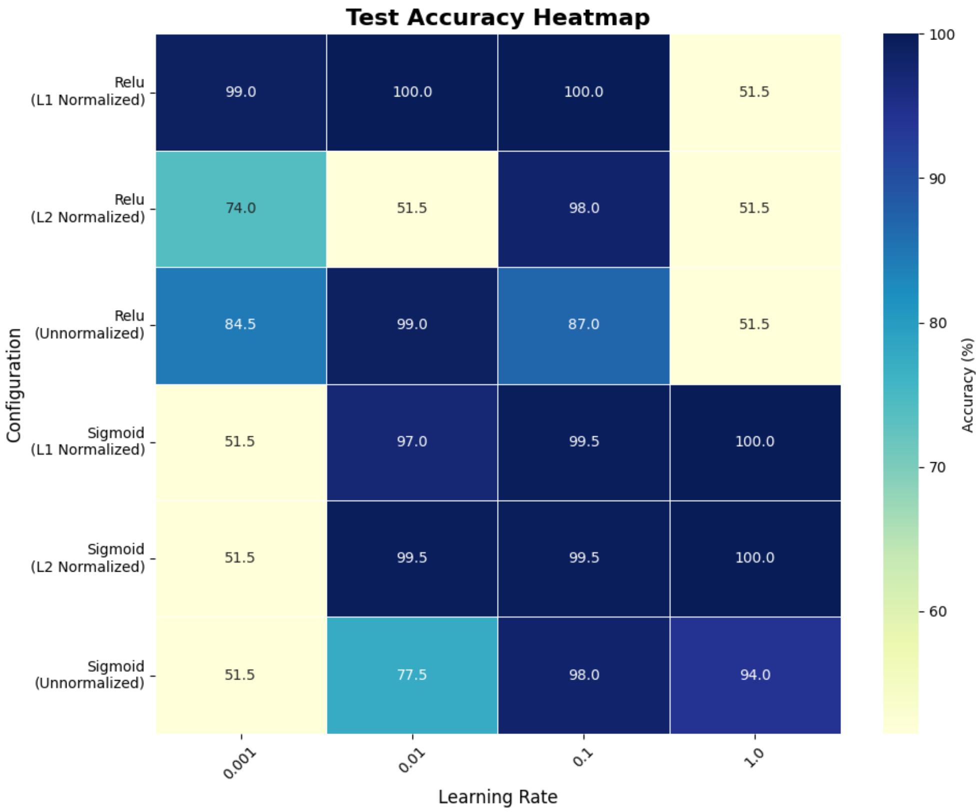
Tempo uczenia jest bezpośrednio regulowane przez współczynnik uczenia (η).



Rysunek 1: Krzywe straty (błędu) dla różnych konfiguracji.

4 Heatmapy dokładności





5 Wnioski końcowe

- **Współczynnik uczenia** (η) jest kluczowy. Zbyt niskie η spowalnia, zbyt wysokie rozbiega. Optymalne znalezione η dla rozważanych danych to $0.01 - 0.1$.
- **Normalizacja** poprawia stabilność uczenia.
- **Funkcja Sigmoid** zapewnia niższą stratę i płynniejszą konwergencję przy dobrym η. **ReLU** charakteryzuje się szybszym początkowym spadkiem, ale może stabilizować się na wyższym poziomie.