

Projekt 2: Analiza możliwości algorytmów optymalizacji

Sebastian Michoń 136770, Marcin Zatorski 136834

grupa L5

1 Zarys idei

1. Obliczenia przeprowadzano dla 2 architektur:
 - (a) Standardowa sieć neuronowa, złożona z warstw gęstych o kolejno 10-50-100-100-100-5 neuronach (10 neuronów wejściowych, 5 wyjściowych).
2. Porównanie optymalizatorów było elementem hiperparameter tuningu i badania wpływu poszczególnych metod regularyzacji na poszczególne optymalizatory.
3. Operacje przeprowadzone dla pierwszej architektury:
 - (a) Stworzono pewną sieć neuronową, dane treningowe i dane testowe. Dane treningowe składały się z 50.000 instancji, dane testowe z 10.000 instancji (instancje składały się z 10 wartości).
 - (b) Uruchamiano losowo zainicjalizowaną (z biasem z rozkładu normalnego i wagami pochodzącymi z jednorodnego inicjalizatora Xaviera) sieć neuronową dla danych treningowych. Będzie ona nazywana dalej przewzorcową siecią neuronową.
 - (c) W każdym pojedynczym eksperymencie porównywano określone optymalizatory w następujący sposób:
 - i. Ustalano ground truth jako rezultat propagacji zestawu treningowego i testowego przez sieć neuronową z wagami pochodzącymi z przewzorcowej sieci neuronowej i regularyzacją (gdyby używać przewzorcowej sieci neuronowej z takimi samymi wagami i bez regularyzacji, wyniki mogłyby się różnić dla regularyzacji 'batch normalization'; Dzięki rozwiązaniu zadania w taki sposób spełniono założenie o identycznej architekturze sieci neuronowych jednocześnie - dzięki kopiowaniu wag - umożliwiając porównywanie rezultatów dla różnych rodzajów regularyzacji). Sieć ta nazywana będzie wzorcową siecią neuronową.
 - ii. Tworzono sieć neuronową dla podanej metody regularyzacji i podanych hiperparametrów (np. dropout_rate=0.2). Nazywana ona będzie dalej testową siecią neuronową
 - iii. Trenowano testową sieć neuronową w 3 epokach.
 - iv. Po wytrenowaniu testowej sieci neuronowej ewaluowano ją na zbiorze treningowym, testowym i porównywano wagi w dwóch sieciach neuronowych: testowej i wzorcowej.
4. Funkcją kosztu dla porównania danych wyjściowych było MSE.
5. Dla pierwszej architektury eksperymenty przeprowadzono dla następujących typów regularyzacji:
 - (a) **batch_normalization**: przeprowadzano testy dla 3 optymalizatorów: SGD, Adam i AdamW(weight_decay = 0.0001). Przeprowadzono testy dla każdej pary złożonej z wartości z ciągu [4, 16, 64, 256] dla parametru batch_size i wartości z ciągu [0.1, 0.5, 0.9, 0.95, 0.99, 0.999] dla parametru momentum - w sumie wykonano 24 testy dla tej regularyzacji.

- (b) **weight_decay**: przeprowadzano testy dla 2 optymalizatorów: SGD i AdamW. Przeprowadzono testy dla każdej wartości z ciągu [0.5, 0.1, 0.01, 0.001, 0.0001, 0.00001] dla parametru weight_decay - w sumie wykonano 6 testów dla tej regularyzacji.
- (c) **dropout**: przeprowadzano testy dla 3 optymalizatorów: SGD, Adam i AdamW(weight_decay = 0.0001). Przeprowadzono testy dla każdej wartości z ciągu [0, 0.1, 0.2, 0.3, 0.4, 0.5] dla parametru dropout_rate - w sumie wykonano 6 testów dla tej regularyzacji.

Wykonano zatem w sumie 36 testów. Wartości Dropout / Batch normalization wstawiono pomiędzy każdą parę warstw gęstych z wyłączeniem pierwszych 2 warstw (czyli na przykład 10-50-Dropout-100-Dropout-100-Dropout-100-Dropout-5).

6. Funkcja kosztu wag:

- (a) Zaimplementowano standardową funkcję liczącą MSE będącą uśrednioną sumą błędów kwadratowych dla wszystkich wag i biasów (bias liczony był w średniej jako jedna z wag).
- (b) Aby porównywać wagi neuronów, które są w jakiś sposób związane, przed wyliczeniem funkcji kosztu modyfikowano macierze wag w wytrenowanej testowej sieci neuronowej tak, aby neurony, które mają podobne wartości wag i biasa na wejściu były na tych samych pozycjach w obydwu porównywanych sieciach neuronowych.
- (c) W *tensorflow* element macierzy wag $W_{i,j}$ oznacza wagę i -tego wyjścia z poprzedniej warstwy dla j -tego wejścia kolejnej warstwy. Co za tym idzie, wektor $V_j = [W_{1,j}, W_{2,j} \dots W_{n,j}, b_j]$ reprezentuje kolejne wagi na wejściu j -tego neurona w kolejnej warstwie.
- (d) Dla wszystkich macierzy wag i biasów - począwszy od pierwszej - porównywano ujemne podobieństwo kosinusowe pomiędzy każdą parą wektorów V_j w obu sieciach (wzorcowej i testowej) dla tej samej warstwy.
- (e) Celem metody była maksymalizacja podobieństwa neuronów na tych samych pozycjach; w tym celu wykorzystano metodę węgierską, aby wybrać minimalną możliwą sumę ujemnych podobieństw kosinusowych przy pewnej zamianie miejscami pozycji neuronów. Rezultatem metody węgierskiej była sekwencja par $1 : a_1, 2 : a_2, \dots m : a_m$ oznaczająca, że aby zminimalizować sumę ujemnych podobieństw kosinusowych wektorów wag (z biasem) dla pojedynczego neurona należy dokonać takiej transformacji na macierzy wag, aby kolumna o indeksie a_i była na pozycji i -tej po transformacji testowej sieci neuronowej. Transformację tę osiągnięto przez:

$$\begin{bmatrix} k_1 & k_2 & \dots & k_m \end{bmatrix} \begin{bmatrix} e_{a_1} & e_{a_2} & \dots & e_{a_m} \end{bmatrix} = \begin{bmatrix} k'_1 & k'_2 & \dots & k'_m \end{bmatrix}$$

gdzie e_i oznacza pionowy wektor jednostkowy o wypełniony zerami i jedną jedynką w pozycji i -tej, wektory jednostkowe mają rozmiar m (macierz z prawej jest kwadratowa), wektory k_j to kolumny macierzy wag przed transformacją. W ten sam sposób transformowano wektor biasów.

- (f) Analogicznie, jeśli za daną warstwą była inna warstwa gęsta, zamieniono kolejnością wiersze następnej macierzy wag tak, aby kolejność neuronów była taka sama w obu macierzach (a zatem należało zmienić kolejność wartości wychodzących z poprzedniej warstwy):

$$\begin{bmatrix} e_{a_1}^T \\ e_{a_2}^T \\ \dots \\ e_{a_m}^T \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \dots \\ w_m \end{bmatrix} = \begin{bmatrix} w'_1 \\ w'_2 \\ \dots \\ w'_m \end{bmatrix}$$

gdzie w_i oznacza i -ty wiersz pierwotnej macierzy. Nie transformowano wektora biasów, ponieważ nie miał on związku z transformacją pozycji wyjść z poprzedniej warstwy.

2 Rezultaty i ich omówienie

2.1 Testy pierwszej sieci

2.1.1 Testy batch normalization

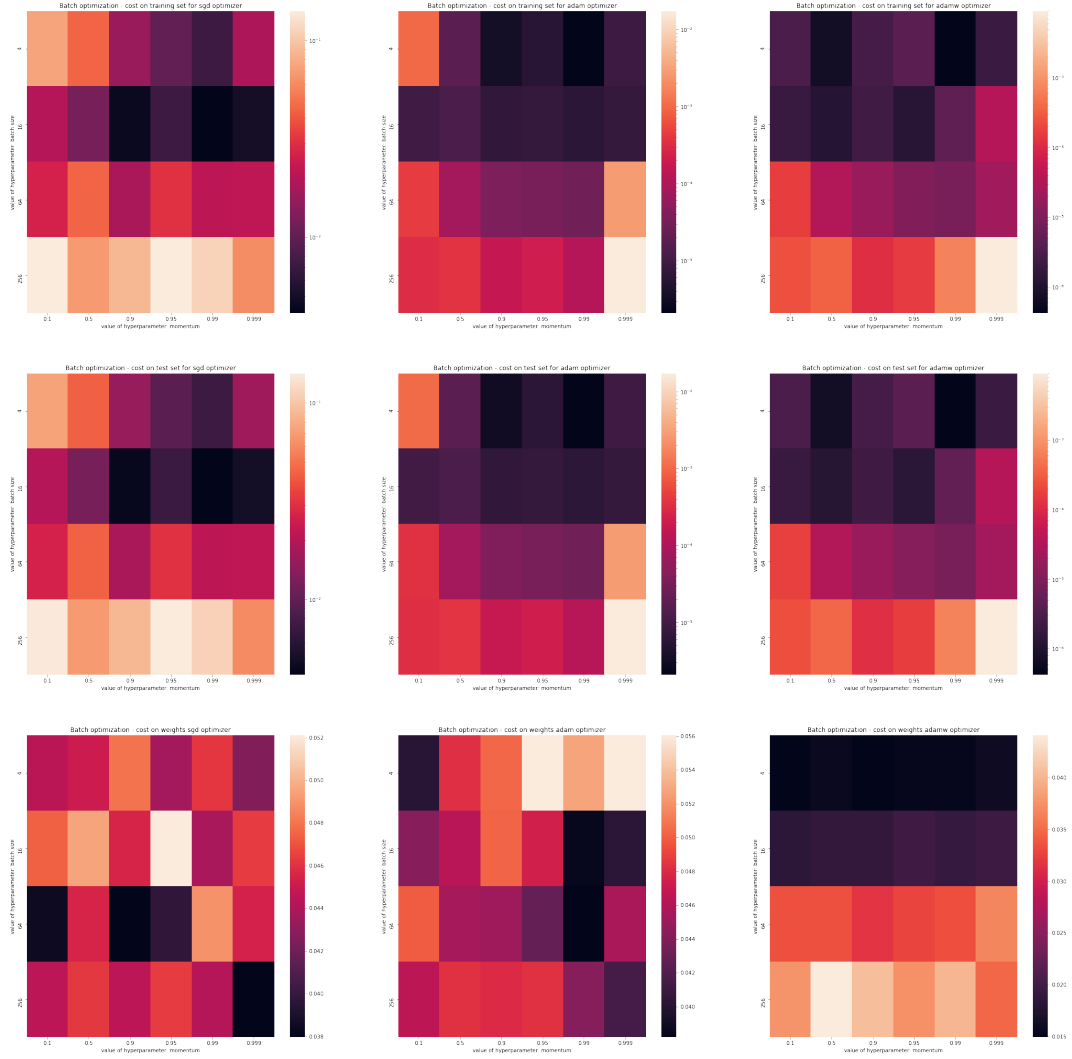


Figure 1: Rezultaty nauki sieci z regularyzacją typu batch normalization

Opis testów i ich rezultatów:

1. Na tym i każdym kolejnym wykresie wartości funkcji kosztu dla zestawu testowego i treningowego będą przedstawiane w skali logarytmicznej, nawet dla SGD.
2. Wartości funkcji kosztu dla zbioru treningowego i testowego są nieomal identyczne - wynika to z:

- (a) 50.000 Instancji danych treningowych pochodzących z tego samego rozkładu - pociąga to za sobą możliwość efektywnego wytrenowania sieci.
- (b) Braku szumu w danych wyjściowych - Wzorcowe wartości na wyjściu są funkcją zależną jedynie od inputu.
- (c) Identycznej architektury obu sieci neuronowych.

Obserwacja ta będzie zauważalna we wszystkich kolejnych testach.

- 3. Najlepsze rezultaty optymalizacji SGD (wartość MSE rzędu około 0.01) są porównywalne z najgorszymi rezultatami optymalizacji Adam i AdamW. Obserwacja ta będzie się powtarzała w kolejnych testach.
- 4. Wszystkie optymalizatory uzyskały najniższe wartości funkcji kosztu dla momentum=0.99.
- 5. Dla SGD najlepsze wyniki osiągnęto dla batch_size=16, dla Adam i AdamW dla batch_size=4.
- 6. Wartości fukcji kosztu dla batch_size wyższego równego 64 są prawie zawsze wyższe niż dla mniejszego batch_size.
- 7. AdamW dla batch_size=4 bardzo dobrze dopasowywał się do wag sieci wzorcowej; żaden inny optymalizator nie osiągnął podobnych rezultatów dla opisanej w paragrafie 6 funkcji kosztu (0.15 AdamW względem 0.38 SGD i Adama). W ogólności Adam_W lepiej dopasowywał wagi do sieci wzorcowej niż pozostałe optymalizatory.
- 8. Dla wartości funkcji kosztu osiąganey przez optymalizaotry Adam i AdamW niemożliwość perfekcyjnego dopasowania do wzorcowej sieci neuronowej może wynikać między innymi z błędów zaokrągleń.

2.1.2 Testy weight decay

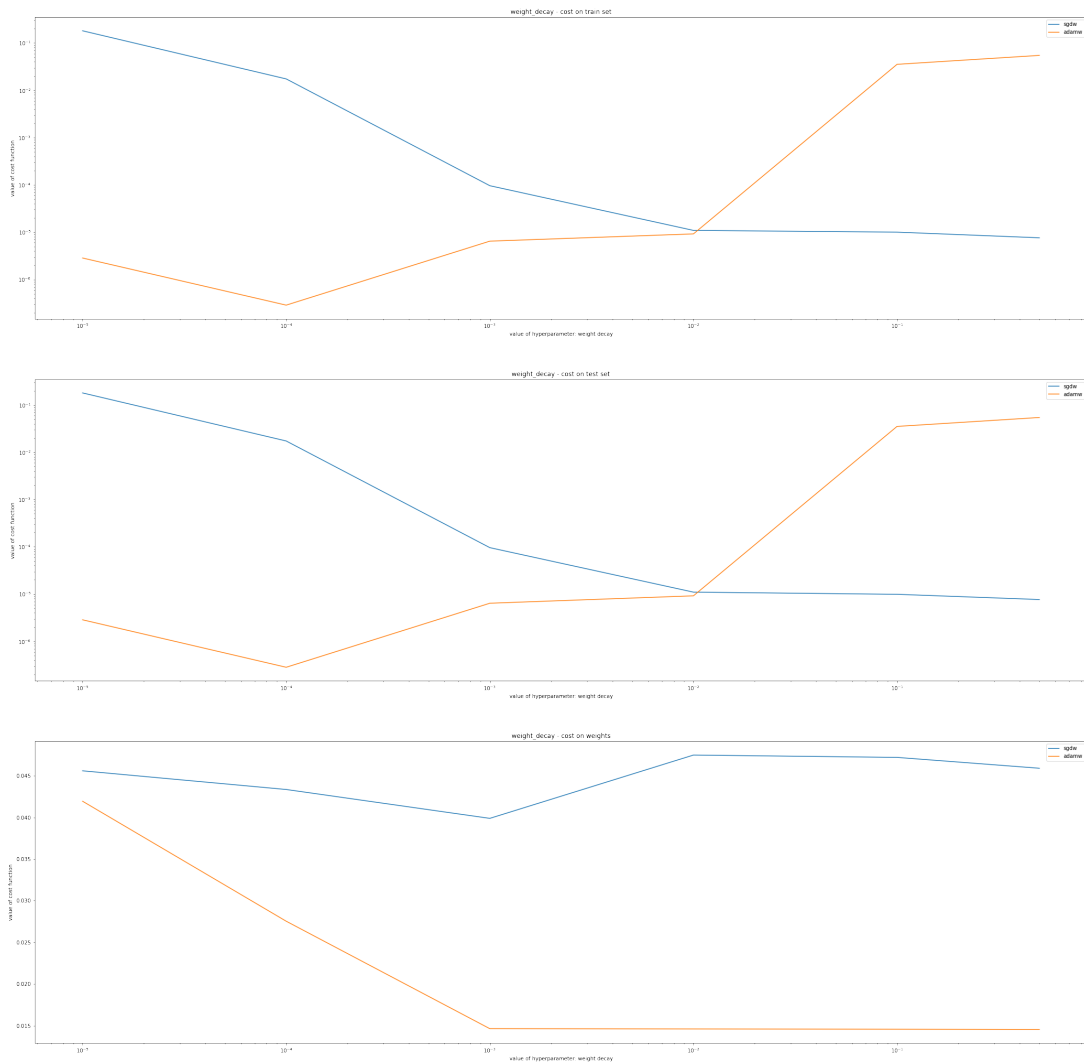


Figure 2: Rezultaty nauki sieci z różnymi wartościami weight decay w optymalizatorze.

Opis testów i ich rezultatów:

1. Skale na obu osiach są skalami logarytmicznymi z wyjątkiem ostatniego rysunku, gdzie na osi y jest skala liniowa.
2. Optymalizator SGDW osiągał najlepsze rezultaty dla weight decay większego niż 0.1, z kolei AdamW osiągał najlepsze rezultaty dla weight decay rzędu 10^{-4} .
3. Pomimo ponad stukrotnie wyższej wartości funkcji kosztu dla zestawu testowego i treningowego dla AdamW niż SGDW dla weight decay rzędu 0.1, funkcja kosztu dla wag była około 3 razy

niższa dla algorytmu AdamW niż dla SGD. W ogólności, funkcja kosztu wag była prawie stała dla SGD (na poziomie 0.45) i malejąca dla AdamW.

4. Najlepsze rezultaty osiągane przez SGD i AdamW są podobne (około 10 razy niższe dla AdamW), co pokazuje wyższą skuteczność optymalizatora SGD z odpowiednio dobraną wagą niż standardowego optymalizatora SGD.

2.1.3 Testy dropoutu

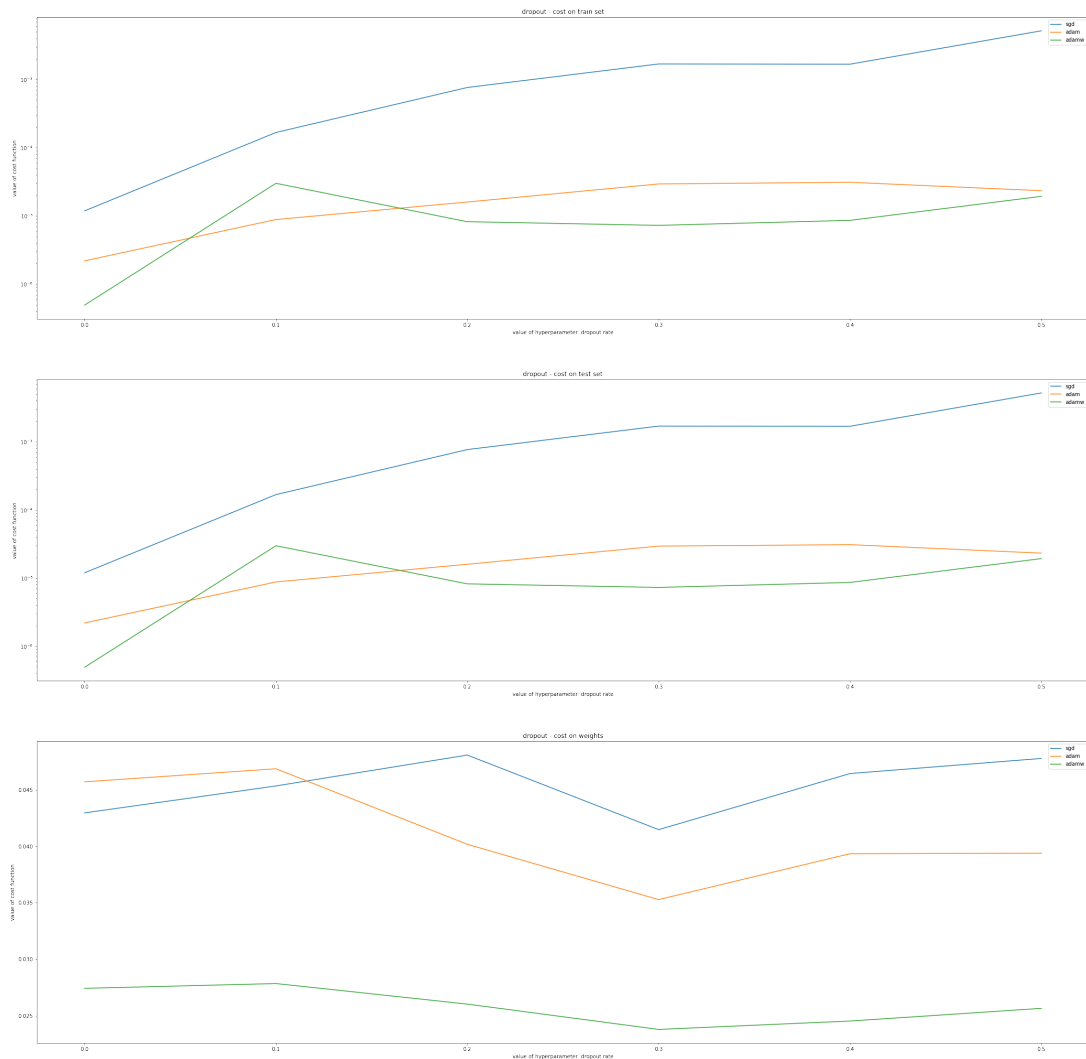


Figure 3: Rezultaty nauki sieci z różnymi wartościami weight decay w optymalizatorze.

Opis testów i ich rezultatów:

1. Skale na osi y są skalami logarytmicznymi z wyjątkiem ostatniego rysunku, gdzie na osi y jest skala liniowa.

2. Wzrost dropout rate często prowadził do wyższych wartości funkcji kosztu na zestawach treningowym i testowym - może to wynikać z kilku czynników:
 - (a) Celem dodania dropoutu jest uniknięcie przetrenowania i dostosowania się sieci neuronowej do wyników obarczonym pewnym szumem; w tych danych nie ma żadnego szumu, teoretycznie można osiągnąć funkcję kosztu równą 0 dla każdych danych.
 - (b) Istnienie dropoutu może penalizować próbę upodobnienia sieci testowej do wzorcowej sieci neuronowej, np. przez odrzucanie neuronów istniejących we wzorcowej sieci mających kluczowy wpływ na rezultat.
3. Wszystkie 3 optymalizatory osiągały najbardziej podobną sieć neuronową do sieci wzorcowej dla `dropout_rate=0.3`.

2.1.4 Uwagi ogólne

1. Optymalizator SGD osiągał gorsze rezultaty na zbiorach treningowym i testowym z normalizacją niż bez niej. Jego najlepszy rezultat to $MSE \approx 10^{-5}$, dla Adama ta wartość wynosiła $MSE \approx 2 * 10^{-6}$, a dla AdamW $MSE \approx 5 * 10^{-7}$.
2. Regularyzacja prowadziła do zmniejszenia funkcji kosztu wag, co można zauważyć na wykresie 3 (pokazującym funkcję kosztu wag w zależności od dropoutu).

3 Wnioski

1. Optymalizator SGD zawsze osiągał gorsze rezultaty niż AdamW z $\text{weight_decay}=10^{-4}$, ponadto nie wykonywał się szybciej niż AdamW.
2. Wykorzystanie optymalizatora AdamW z dobrze dopasowanym weight_decay praktycznie zawsze prowadzi do lepszych rezultatów niż wykorzystanie optymalizatora Adam - zarówno w kontekście podobieństwa wag sieci neuronowych, jak i funkcji kosztu.
3. Wartość weight_decay ma bardzo duży wpływ na działanie optymalizatora, może powodować nawet osiągnięcie nawet 10^5 razy mniejszej funkcji kosztu niż dowolna wartość tego hiperparametru. Optymalna wartość tego hiperparametru jest zależna od algorytmu.
4. Parametr momentum dla batch normalization na poziomie 0.99 zdaje się być odpowiedni dla testowanych optymalizatorów, taka jest też jego domyślna wartość w tensorflowie.
5. Przy używaniu batch normalization niższa wartość batch_size niekoniecznie prowadziła do lepszej wartości funkcji kosztu pomimo dłuższego przetwarzania i większej liczby aktualizacji wag dla tej samej liczby epok (jest to widoczne dla optymalizatora SGD).
6. W specyficznym przypadku bez żadnego "szumu" w danych wyjściowych dropout nie prowadził do polepszenia predykcji na zbiorach treningowym i testowym, wręcz przeciwnie. Wartość dropoutu na poziomie 0.3 pozwalała natomiast najlepiej dopasować się do macierzy wag. W ogólności, w testach pierwszej architektury regularyzacja niewiele przynosiła w kontekście funkcji kosztu zarówno dla zbioru treningowego, jak i testowego.