WSTĘP DO SZTUCZNEJ INTELIGENCJI

Ćwiczenie 5 – Perceptron Wielowarstwowy

JAKUB KWAŚNIAK 331396

MICHAŁ MIZIA 331407

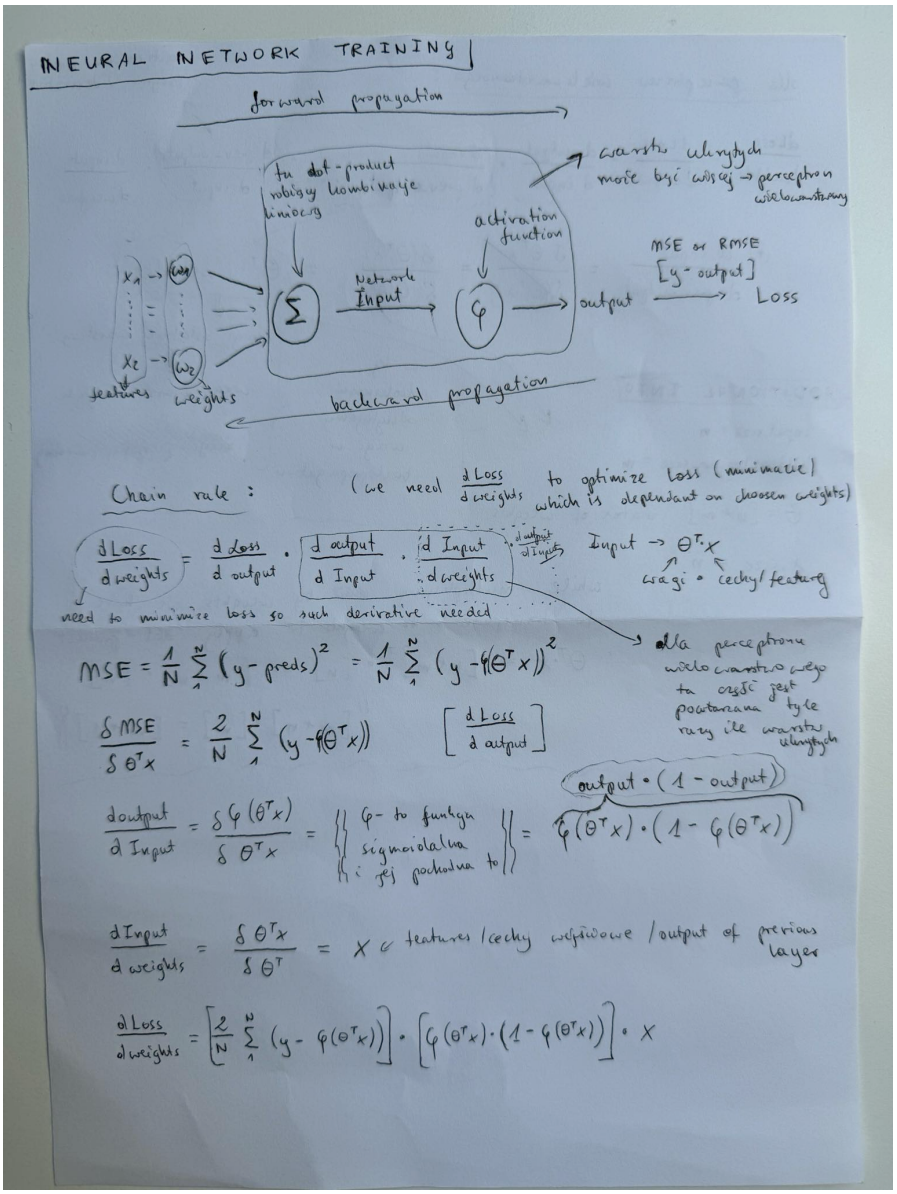
SPIS TREŚCI

Wstęp 3
Implementacja algorytmu 3
Wyniki 5
Wnioski 7

Wstęp

1. Zaimplementować sieć neuronową - perceptron wielowarstwowy, oraz mechanizm jej uczenia algorytmem propagacji wstecznej gradientu.
2. Sprawdzić jakość działania algorytmu dla zadania regresji na zbiorze danych Wine Quality https://archive.ics.uci.edu/dataset/186/wine+quality.
3. Spróbować znaleźć konfigurację sieci, która pozwala osiągać dobre rezultaty.

IMPLEMENTACJA ALGORYTMU



dla perceptronu wielowarstwowego:

activation function (sigmoid) derivative

$$\frac{dLoss}{dweights} = \boxed{\frac{dLoss}{doutput} \cdot \frac{doutput}{dinput}} \cdot \boxed{\frac{dinput}{dprev_output_n}} \cdot \dots \cdot \frac{dprev_output_1}{dinput} \cdot \frac{dinput}{dweights}$$

$$(*)\ \frac{dinput}{dprev_output} = \frac{\delta\, \cancel{\theta^T x}}{\cancel{\delta[\theta^T x)]}} = \frac{\delta(\theta^T x)}{\delta(G(\theta^T x))} = \boxed{\theta^T\ (weights)}$$

$\Downarrow$

wagi ostatniej warstwy

$\Downarrow$

wagi warst ukrytych

$\Downarrow$

~~wagi pierwszej warstwy~~

w tej kolejności odczytujemy wagi w backpropagation

ADDITIONAL INFO

input_size = n

hidden_layer_size = m

$\theta = [n * m]$ matrix of weights

X.size = n

while multiplying by weights we have to transpose weights to apply dot-product

$$\theta^T \cdot x \Rightarrow [n * m]^T \cdot [n] \Rightarrow [m * n] \cdot [n]$$

$$\{\{[m * n] \cdot [n] = [m * 1]\}\}$$

Wyniki

- **Najlepszy wynik:**

Mean Squared Error: 0.5468

- **Przykładowy zrzut ekranu danych, błędu średniokwadratowego dla wszystkich danych oraz przewidywań:**

```
      fixed_acidity  volatile_acidity  citric_acid  residual_sugar  chlorides  ...  total_sulfur_dioxide  density    pH  sulphates  alcohol
4152            6.3              0.13         0.42             1.1      0.043  ...                 146.0  0.99066  3.13       0.72     11.2
931             7.4              0.61         0.01             2.0      0.074  ...                  38.0  0.99748  3.48       0.65      9.8
4252            6.4              0.28         0.27            11.0      0.042  ...                 148.0  0.99786  3.14       0.46      8.7
1743            8.1              0.20         0.40             2.0      0.037  ...                  87.0  0.99210  3.12       0.54     11.2
1238            8.1              0.73         0.00             2.5      0.081  ...                  24.0  0.99798  3.38       0.46      9.6

[5 rows x 11 columns]
      quality
4152        7
931         5
4252        5
1743        6
1238        4
Mean Squared Error: 0.907698095533477
Predictions:
 [[5.97674635]
 [5.10788221]
 [5.63646462]
 [5.53257792]
 [4.12034158]] [[6.09539468]
 [5.11636046]
 [5.23518099]
 [5.44606696]
 [6.84578886]]
```

Poniższa tabela przedstawia wyniki dla poszczególnych ilości warstw ukrytych, te ilości przedstawione są w kolumnie configuration. Mean MSE to średni błąd średniokwadratowy na przestrzeni 10 prób przy czym jeśli próba się nie uda, czyli algorytm „utknie” / sieć obumrze (MSE wyjdzie > 3) – dzieje się tak gdy sieć nie będzie zmniejszać błędu lub będzie zmniejszać go w zbyt małym stopniu (lub nawet zwiększać) (underfitting) - wówczas próba jest powtarzana a ilość nieudanych prób jest powiększana o 1. Bad runs to stosunek nieudanych prób do wszystkich.

1. Próby dla wag inicjalizowanych losowo z wąsko ograniczonego przedziału (Wagi są losowane z przedziału [-0.1, 0.1])

lr=0.001, iters = 1000

Configuration	Mean MSE	Best MSE	Bad Runs
[10, 6, 4]	0.6708	0.6215	0.0000
[30, 15, 7]	0.7046	0.6757	0.0000
[20, 10]	0.6024	0.5669	0.0000
[10, 6]	0.5781	0.5511	0.0000
[20]	0.5657	0.5468	0.0000

lr=0.01, iters = 1000

Configuration	Mean MSE	Best MSE	Bad Runs
[10, 6, 4]	0.6812	0.6043	0.0000
[30, 15, 7]	0.6686	0.6333	0.0000
[20, 10]	0.5899	0.5556	0.0000
[10, 6]	0.6130	0.5562	0.0000
[20]	0.5604	0.5486	0.0000

2. Próby dla wag inicjalizowanych funkcją He (Wagi są losowane z rozkładu normalnego lub jednostajnego, gdzie wariancja zależy od liczby neuronów w warstwie wejściowej)

lr=0.01, iters = 1000

Configuration	Mean MSE	Best MSE	Bad Runs
[10, 6, 4]	0.6146	0.5415	0.0000
[30, 15, 7]	0.5578	0.5376	0.0000
[20, 10]	0.5497	0.5277	0.0000
[10, 6]	0.5570	0.5346	0.0000
[20]	0.5497	0.5317	0.0000

3. Porównanie StandardScaler i MinimaxScaler przy inicjalizacji wag metodą losową

Porównanie minmax scalera i standard scalera dla 5 iteracji, lr=0.01, iters=1000:

Configuration	Mean MSE	Best MSE
MinMax Scaler	0.6798	0.6614
Standar Scaler	0.5259	0.5240

Porównanie minmax scalera i standard scalera dla 5 iteracji, lr=0.001, iters=1000:

Configuration	Mean MSE	Best MSE
MinMax Scaler	0.6817	0.6579
Standar Scaler	33.0526	0.6560

Wnioski

Minmax scaler jest lepszym wyborem niż standar scaler ponieważ zmniejsza rozstrzał przestrzeni wejść, a więc zawęża możliwości błędu dla algorytmu – w przypadku wybrania Minimax scalera dla różnych hiperparametrów sieć działa przewidywalnie, natomiast dla Standard Scalera należy odpowiednio dobrać wszystkie hiperparametry sieci tak by nie doszło do underfittingu, co widoczne jest w tabelach porównań.

Również inicjalizacja wag losowo z przedziału [-0.1, 0.1] pozwala na uzyskanie sensowynych wyników bez błędów grubych, natomiast dla inicjalizacji He, która zależna jest od rozmiarów warstw ukrytych poprawne wyniki uzyskuje się tylko dla odpowiednio dobranych hiperparametrów, dla zbyt dużych, zbyt małych wartości może dojść do sytuacji w której sieć obumrze przestając minimalizować MSE lub do sytuacji gdy zostanie ono zmaksymalizowane generując błąd inf lub NaN