Programowanie Sieciowe
Perceptron wielowarstwowy

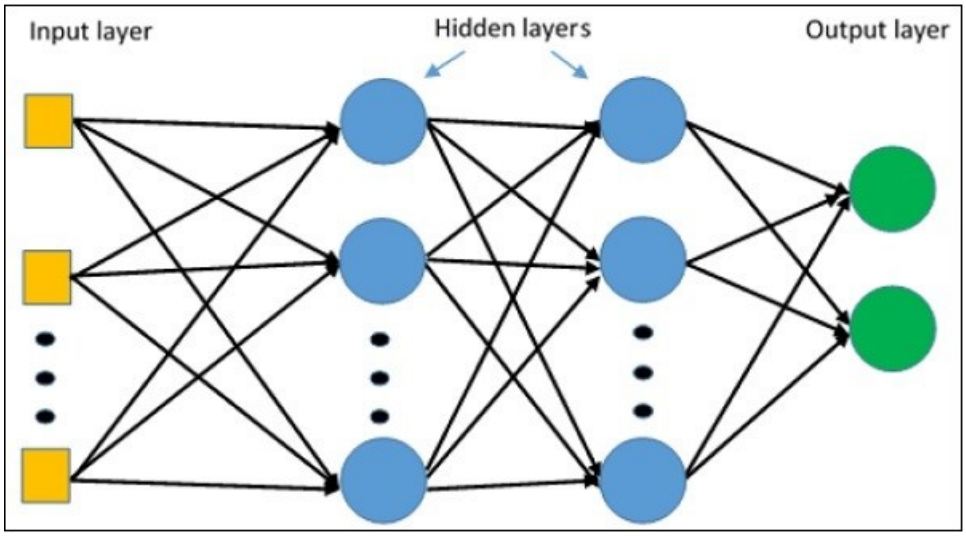
Imię i nazwisko: Dominik Ćwikowski
Indeks: 248914
Data zajęć: 24.03.2021

1. Opis ćwiczenia

Ćwiczenie ma na celu przetestowania działania sieci wielowarstwowej, wyszukiwanie optymalnych parametrów sieci i interpretacja jej wyników .

2. Opis rozwiązania problemu

Sieć tego typu składa się zwykle z jednej warstwy wejściowej, kilku warstw ukrytych oraz jednej warstwy wyjściowej. Warstwy ukryte składają się najczęściej z neuronów McCullocha-Pittsa.



Posłużymy się przy tym także, tak zwanym “One Hot Decoding”, którym będziemy kodować klasy. Jest to kod pozycyjny, o wielkości n bitów, gdzie tylko jeden z nich przyjmuje wartość 1.

Następnie stworzymy słownik który będzie zawierał wartości dla których bedziemy szukać najlepszego klasyfikatora.

```
{'learning_rate_init' : (0.1, 0.01, 0.001), 'hidden_layer_sizes' : [20, 40, 60, 80, 100],
'solver' : ['adam', 'lbfgs', 'sgd']}
```

A pod koniec sprawdzimy dokladność najlepszego klasyfikatora i dokonamy klasyfikacji.

3. Testy

Testy były wykonywane na zbiorach: “Iris”, “Digits” I “Mushrooms” (z Kaggle.com).
“Iris” to najbardziej znana baza danych dla uczenia maszynowego, zawierająca 4 wartości dla 3 różnych kwiatów.
Baza danych “Digits” jest częścią biblioteki “cklearn” do Pythona. Użyłem jej zamiast MNIST, ponieważ próba wykonania ćwiczenia na nim kończyła się otrzymaniem błędów związanych z alokacją pamięci przez funkcje z “cklearn”. “Digits” ma ok. 2000 próbek i rozmiary cyfr 8x8, co pozwoliło mi na pominięcie problemów.
Baza “Mushrooms” zawiera 8000 próbek, podzielone są na 2 klasy “jadalne” i “niejadalne”, i 22 wartości wejściowe.

Dla “Iris”:

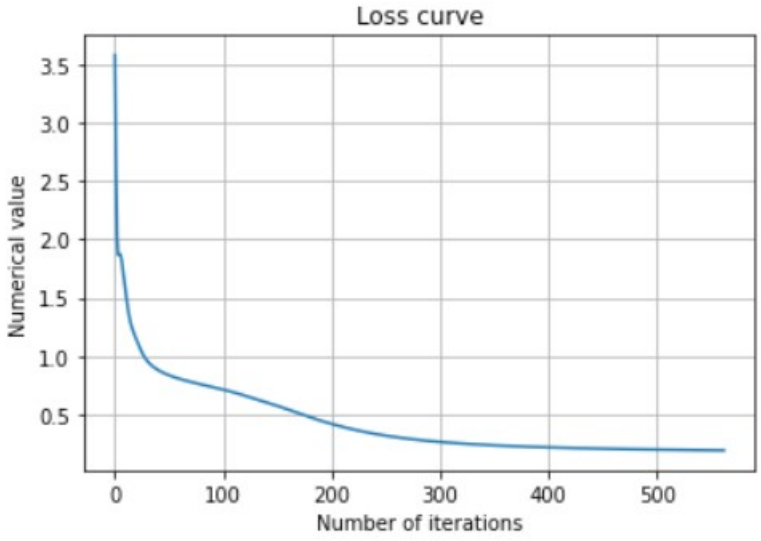
Najlepsze uzyskane parametry:

```
Dokladnosc dla danych treningowych: 98.0%
Dokladnosc dla danych testowych: 98.0%
Najlepsze parametry:  {'hidden_layer_sizes': 40, 'learning_rate_init': 0.01, 'solver': 'sgd'}
```

Dokładność ponownego testu dla najlepszych parametrów:

```
Dokladnosc dla danych treningowych: 97.5%
Dokladnosc dla danych testowych: 100.0%
```

Funkcja celu:



Klasyfikacja:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	11
1	1.00	1.00	1.00	13
2	1.00	1.00	1.00	6
micro avg	1.00	1.00	1.00	30
macro avg	1.00	1.00	1.00	30
weighted avg	1.00	1.00	1.00	30
samples avg	1.00	1.00	1.00	30

Dla “Digits”:

Najlepsze uzyskane parametry:

```
Dokladnosc dla danych treningowych: 100.0%
Dokladnosc dla danych testowych: 100.0%
Najlepsze parametry:  {'hidden_layer_sizes': 100, 'learning_rate_init': 0.1, 'solver': 'lbfgs'}
```

Dokładność ponownego testu dla najlepszych parametrów:

```
Dokladnosc dla danych treningowych: 100.0%
Dokladnosc dla danych testowych: 98.88888888888889%
```

Klasyfikacja:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	27
1	0.97	0.97	0.97	35
2	1.00	1.00	1.00	36
3	1.00	1.00	1.00	29
4	0.97	1.00	0.98	30
5	0.97	0.97	0.97	40
6	1.00	1.00	1.00	44
7	1.00	1.00	1.00	39
8	1.00	0.97	0.99	39
9	0.98	0.98	0.98	41
accuracy			0.99	360
macro avg	0.99	0.99	0.99	360
weighted avg	0.99	0.99	0.99	360

Dla “Mushrooms”:

Najlepsze uzyskane parametry:

```
Dokladnosc dla danych treningowych: 100.0%
Dokladnosc dla danych testowych: 100.0%
Najlepsze parametry:  {'hidden_layer_sizes': 60, 'learning_rate_init': 0.1, 'solver': 'lbfgs'}
```

Dokładność ponownego testu dla najlepszych parametrów:

```
Dokladnosc dla danych treningowych: 100.0%
Dokladnosc dla danych testowych: 100.0%
```

Klasyfikacja:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	852
1	1.00	1.00	1.00	773
accuracy			1.00	1625
macro avg	1.00	1.00	1.00	1625
weighted avg	1.00	1.00	1.00	1625

4. Wnioski

Użycie perceptronu wielowarstwowego, pozwala na uzyskanie lepszej dokładności, potrafiącej wynieść nawet 100%.

Im większa ilość danych wejściowych, tym większa szansa na wynik równy 100%.

Większa ilość warstw ukrytych nie musi wpływać pozytywnie na wynik.