**Sieci neuronowe i neurosterowniki**

**Projekt nr.1**

|  |  |
| --- | --- |
| **Prowadzący:**  Dr. Inż. Krzysztof Halawa | **Termin:**  Środa 13:15 TP |

**Autorzy:**

Armand Piecyk 235293

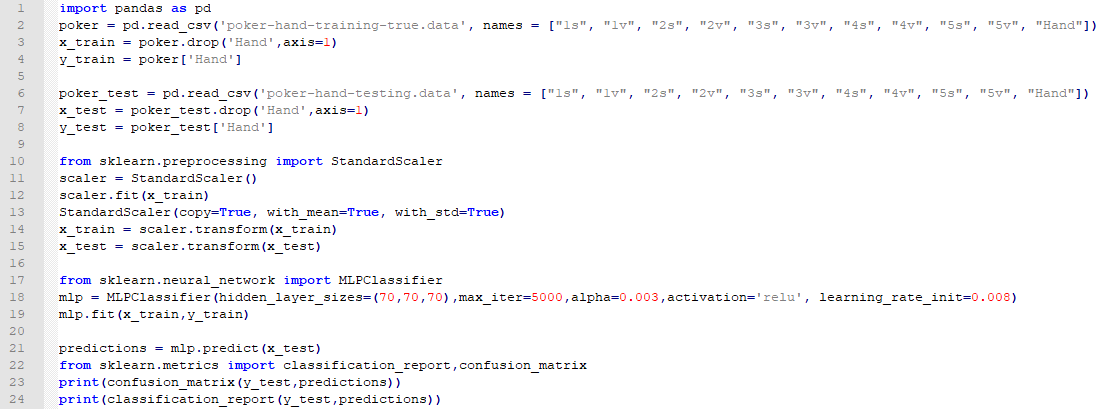
Radosław Bugiel 235118

1.Wstęp

Zadanie polegało na stworzeniu sieci neuronowej- wielowarstwowego perceptronu. Dane pobrano ze strony <https://archive.ics.uci.edu/ml/index.php>, oznaczone jako Poker Hand. Dane składają się z 11 atrybutów zawierających informację o pięciu kartach pobranych ze standardowego zestawu zawierającego 52 karty. Pierwsze 10 atrybutów opisuje kolejno kolor i figurę pięciu kart, a ostatni atrybut opisuje układ pokerowy tworzony przez zestaw tych pięciu kart. Zestaw danych zawiera 25 010 instancji treningowych i 1 000 000 instancji testowych.

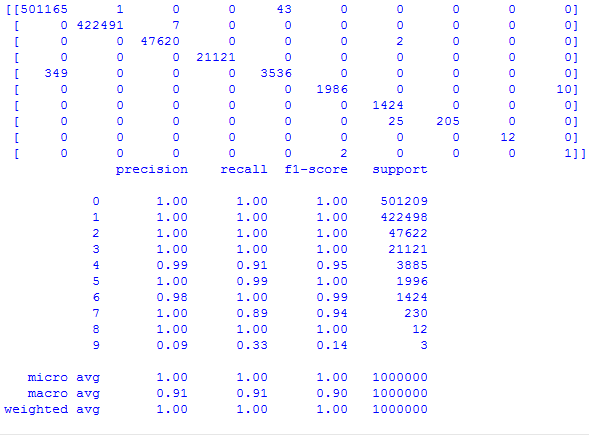
2.Program i testy

Do stworzenia sieci wykorzystano język Python i bibliotekę Scikit-learn**.**



*1.Kod programu w języku Python*

Głównym celem testów było uzyskanie zadowalającej precyzji ocenianych danych. Wyniki wahają się na granicy 80- 90% poprawnej klasyfikacji. Macierz znajdująca się na poniżej pokazuje jak sieć klasyfikował dane. Każdy wiersz odpowiada wartości układu pokerowego tworzonego przez zestaw 5 kart(0-9), a każda kolumna oznacza ilość sklasyfikowanych tam układów. Pierwszy wiersz pokazuje, że z 501209 zestawów danych o wartości 0 sieć poprawnie sklasyfikowała 501165 zestawów, 1 zestaw sklasyfikowała z wartością 1, a 43 zestawom przypisał wartość 4. Następnie w drugim wierszu sieć przypisała złe wartości tylko 7 zestawom, gdzie 422491 zostało ocenione poprawnie. Najmniejsza precyzja jest dla układu o wartości 9, gdzie w całym zestawie danych są 3 takie instancje i tylko jedna została poprawnie sklasyfikowana.

****

*2.Wyniki działania programu.*

Testy przeprowadzono poprzez zmianę lub dodawanie dodatkowych argumentów w funkcji MLPClassifier.

Zmiana argumentu :

Dla solvera „adam”:

1. **hidden\_layer\_sizes**

dla 2 warstw po:

* 110 neuronów poprawność predykcji wynosi poniżej 60%
* 90 neuronów poprawność predykcji wynosi około 70%
* 70 neuronów poprawność predykcji wynosi ponad 70%
* 50 neuronów poprawność predykcji wynosi około 70%
* 30 neuronów poprawność predykcji wynosi poniżej 60%

dla 3 warstw po:

* 110 neuronów poprawność predykcji wynosi około 80%
* 90 neuronów poprawność predykcji wynosi ponad 80%
* 70 neuronów poprawność predykcji wynosi poniżej 90%
* 50 neuronów poprawność predykcji wynosi ponad 80%
* 30 neuronów poprawność predykcji wynosi około 70%

dla 4 warstw po:

* 110 neuronów poprawność predykcji wynosi ponad 70%
* 90 neuronów poprawność predykcji wynosi około 80%
* 70 neuronów poprawność predykcji wynosi ponad 80%
* 50 neuronów poprawność predykcji wynosi około 70%
* 30 neuronów poprawność predykcji wynosi około 70%

1. **max\_iter**

Ilość iteracji nie wpływa znacząco na poprawność predykcji.

1. **Alpha**

Wartości współczynnika alpha w przedziale 0.0001- 0.03 skutkują poprawność predykcji w granicach 85%.

1. **Learning\_rate\_init**

Wartości w przedziale 0.001-0.01 skutkują poprawność predykcji w przedziale 80- 90%. Przy wartościach rzędu 0.0001 oraz 0.05 poprawność predykcji spada do 20%.

1. **Activation**

* relu: poprawność predykcji wynosi poniżej 90%
* tanh: poprawność predykcji wynosi poniżej 60%
* logistic: poprawność predykcji wynosi około 20%

1. **Epsilon**

Domyślnie 1e-8. Wartości w przedziale 1e-3 -- 1e-11 nie powodują widocznych zmian precyzji. Dla wartości 1e-1 otrzymano 20% precyzji, dla 2e-1 40% ,a dla 2e-8 86%.

3. Wnioski

Największy wpływ na poprawność predykcji ma wybór funkcji aktywacji. Najlepsze wyniki otrzymano dla funkcji ReLU. Kolejnym ważnym elementem była ilość ukrytych warstw oraz liczba neuronów w poszczególnej warstwie. Najlepsza okazały się 3 warstwy po 70 neuronów. Zwiększanie lub zmniejszanie obu wartości skutkowało pogarszaniem efektywności. Pozostałe współczynniki dla szerokich zakresów wartości miały nieznaczny wpływ poprawność predykcji. Mogło to wynikać z ilości ukrytych neuronów.