Metody Inteligencji Obliczeniowej

Optymalizacja parametrów sieci neuronowej z użyciem metaheurystyk inspirowanych naturą

Kinga Miszczak

Przemysław Rewiś

Tomasz Szkaradek

Akademia Górniczo-Hutnicza w Krakowie

Wydział Fizyki i Informatyki Stosowanej

Opis projektu

Naszym zadaniem było opracowanie oraz przebadanie algorytmu dobierania wag sieci neuronowej MLP wybraną techniką inspirowaną naturą z listy Evolutionary Computation Bestiary link. W naszym przypadku wybór padł na algorytm sztucznej kolonii pszczół (ABC). Wyniki działania stworzonego algorytmu należało porównać z wynikami uzyskanymi z przykładu: link. Jako zbiór danych wykorzystaliśmy najpopularniejszy zestaw wszech czasów: *Iris flower data set* - zestaw pomiarów kwiatów irysa, udostępniony po raz pierwszy przez Ronalda Fishera w roku 1936. Zbiór irysów składa się z 4 wartości pomiarów jego płatków (szerokości i długość) oraz klasy do jakiej należy.

Algorytm sztucznej kolonii pszczół

Algorytm sztucznej kolonii pszczół (ABC) jest algorytmem optymalizacji, który symuluje zachowanie rodziny pszczół i został po raz pierwszy zaproponowany przez Karaboga w 2005 roku do optymalizacji parametrów rzeczywistych. W tym matematycznym modelu, kolonia pszczół składa się z trzech rodzajów pszczół:

- pszczół pracowniczych, które będą zbierać pokarm do ula w określonym źródle pokarmu
- pszczół obserwacyjnych, które będa wybierały dobre źródło pożywienia spośród źródeł znalezionych przez pszczoły pracownicze
- pszczół zwiadowczych, które będą szukały nowych lokalizacji źródeł pożywienia.

W algorytmie ABC miejsce źródła pokarmu stanowi możliwe rozwiązanie problemu optymalizacji, a początkowo liczba źródeł pożywienia jest równa liczbie pszczół pracowniczych w ulu. Jakość źródła pożywienia jest określona przez wartość funkcji celu na tym stanowisku (wartość przystosowania). Pszczoły zaczynają losowo eksplorować otoczenie w

poszukiwaniu dobrych źródeł pożywienia. Po znalezieniu źródła pożywienia pszczoła staje się pszczołą pracowniczą i zaczyna wydobywać pożywienie w odkrytym źródle. Następnie wraca z nektarem do ula i rozładowuje go. Po wyładowaniu nektaru może wrócić bezpośrednio do źródła lub podzielić się informacjami o źródle. Jeśli źródło pożywienia się wyczerpie, pszczoła pracownicza zostaje zwiadowcą i zaczyna losowo szukać nowego źródła. Pszczoły obserwacyjne czekające w ulu obserwują pszczoły pracownicze podczas zbierania pożywienia i wybierają źródło pożywienia w zależności od ilości nektaru. Wybór źródła pożywienia jest proporcjonalny do jakości źródła (wartości przystosowania).

Realizacja projektu

Projekt został zrealizowany w języku Python 3.

1. Pierwszym krokiem jest wczytanie zbioru danych za pomocą funkcji:

```
dataset=load_iris()
```

   z pakietu scikit-learn.

2. Dzielimy dataset na wektor wejściowy i wektor odpowiedzi żądanej:

```
data_inputs = np.array(dataset.data)

data_outputs = np.array(dataset.target)
```

3. Inicjalizujemy funkcję celu:

```
objective_function = Iris(dim=16)
```

4. Tworzymy obiekt klasy ABC (reprezentacja algorytmu) ustalamy parametry:

```
optimizer = ABC(obj_function=objective_function, colony_size=30, n_iter=300,
max_trials=100)
```

5. Wywołujemy optymalizacje

5.1 Schemat algorytmu optymalizacji ABC:

Procedura ABC

begin

Inicjalizuj populację n osobników x oraz oceń populację

repeat

Utwórz nowe rozwiązanie v_i dla bezrobotnych pszczół (*) Zastosuj zachłanną metodę selekcji dla pszczół pracowniczych

Oblicz prawdopodobieństwo p_i dla rozwiązania x_i przez (**) Utwórz nowe rozwiązania dla obserwatorów x w zależności od p_i

Zastosuj zachłanną metodę selekcji dla pszczół pracowniczych

Określ rozwiązania odrzucane, jeśli można i losowo wygeneruj nowe x_i (***) Zapamiętaj rozwiązanie najlepsze dotychczas znalezione

until warunek zakończenia niespełniony

end

Utworzenie nowego rozwiązania (położenie źródła pożywienia) v_i gdzie:

$$v_i = x_i + \varphi_i(x_i - x_k) \text{ (*)}$$

jest w sąsiedztwie x_i pszczoły robotnicy

x_k – losowo wybrany wektor ($k \neq i$)

φ_i – losowa liczba z zakresu [-1,1]

Sztuczna pszczoła obserwator wybiera źródło pożywienia na postawie prawdopodonieństwa

$$p_i = \frac{f_i}{\sum_{k=1}^{n} f_k} \quad (**),$$

gdzie:

f_i - wartość przystosowania i (proporcjonalna do ilości nektaru w położeniu i)

n - liczba źródeł pożywienia (równa liczbie pszczół robotnic)

Usuwane rozwiązane ze zwiadowca

$$x_i^j = x_{min}^j + \varepsilon \cdot (x_{max}^j - x_{min}^j) \quad (***),$$

gdzie:

$x_{min,\,max}^j$ - dolny/górny zakres j współrzędnej parametru x_i

ε - losowa liczba z zakresu [0,1]

Jako warunek stopu przyjęliśmy liczbę 300 iteracji, a funkcja aktywacji ma postać:

```
def relu(inpt):
    result = inpt
    result[inpt<0] = 0
    return result
```

6. Za pomocą funkcji predict_outputs sprawdzamy dokładność rozwiązania

```
def predict_outputs(weights_mat, data_inputs, data_outputs,
activation="relu"):

    predictions = np.zeros(shape=(data_inputs.shape[0]))

    for sample_idx in range(data_inputs.shape[0]):

        r1 = data_inputs[sample_idx, :]

        for curr_weights in weights_mat:

            r1 = np.matmul(r1, curr_weights)

            if activation == "relu":
```

```
            r1 = relu(r1)

        predicted_label = np.where(r1 == np.max(r1))[0][0]

        predictions[sample_idx] = predicted_label

    correct_predictions = np.where(predictions == data_outputs)[0].size

    accuracy = (correct_predictions/data_outputs.size)*100

    return accuracy, predictions
```

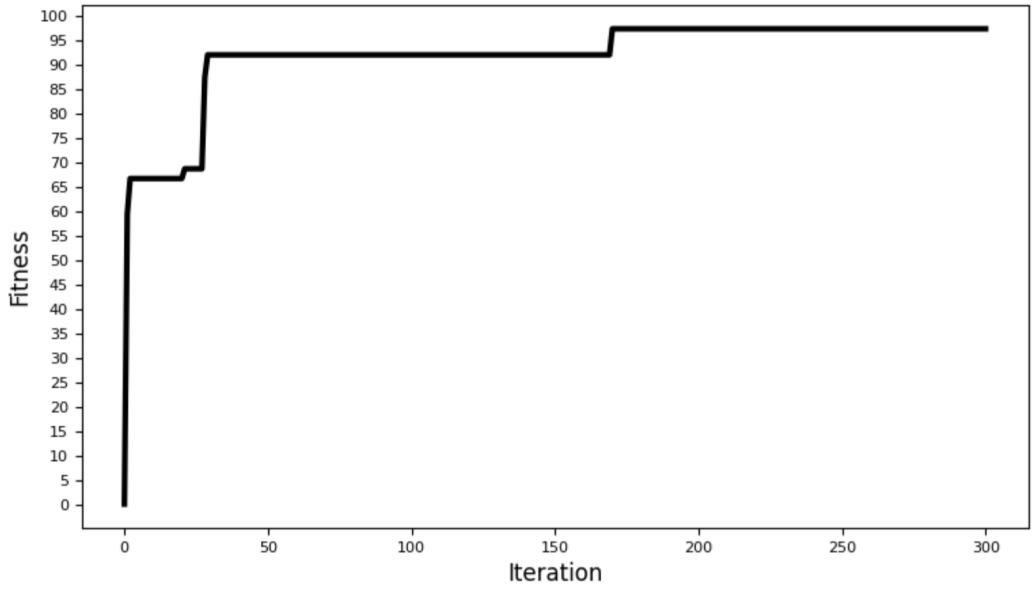
7. Rysujemy wykres dokładności (accuracy) od iteracji
8. Analizujemy wyniki

Analiza i porównanie wyników

Uzyskaliśmy następujące wyniki:

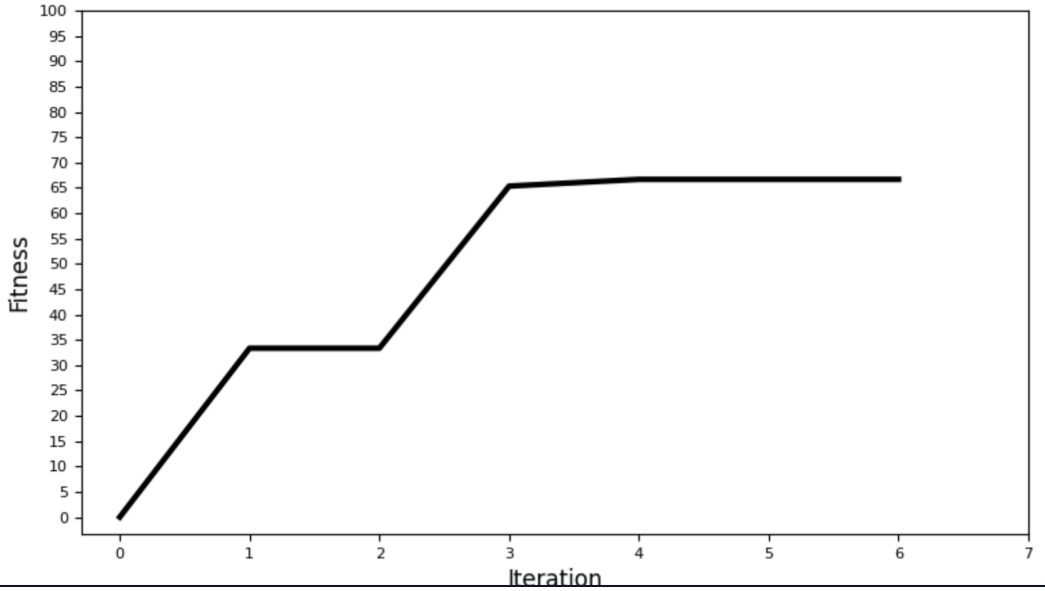
Uruchomienie	1	2	3	4	5	Średnia dokładność
Dokładność ABC	94.67	97.33	94.67	93.33	96.0	95.2
Dokładność PSO	61.33	64.67	60.0	66.67	66.67	63.87
Dokładność GA	72.0	33.33	33.33	66.67	48.67	50.8

Tabela 1. Zestawienie dokładności poszczególnych algorytmów

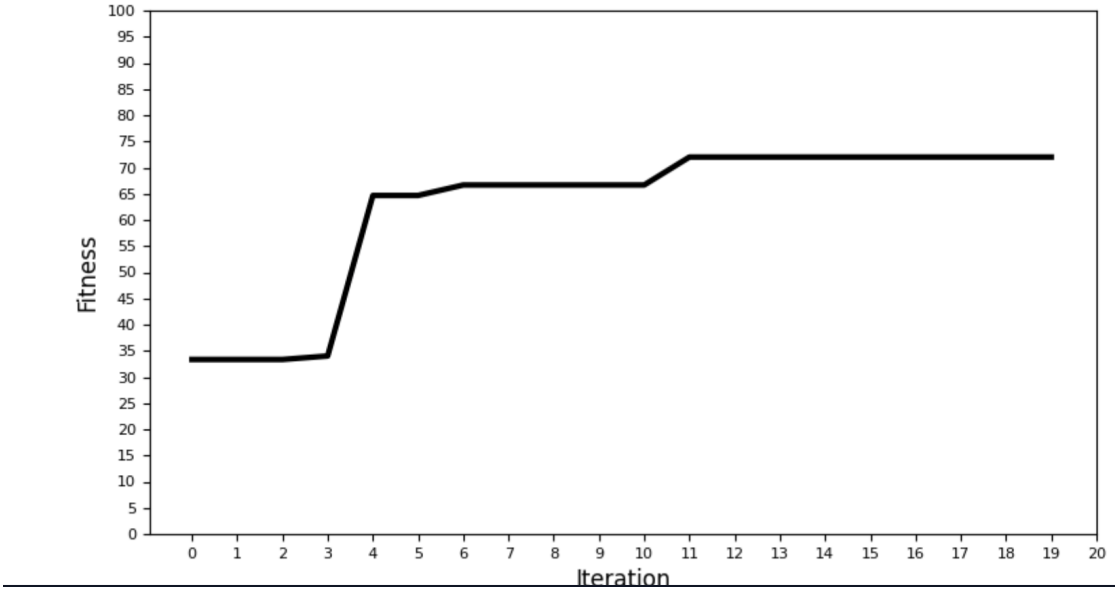


Rysunek 1. Najlepszy uzyskany wykres dokładności (accuracy) od iteracji dla ABC

Wagi optymalnego rozwiązania:
[0.04791622 0.12628144 0.04829219 0.]
[1. 0.1420232 0. 0.20859226]
[0.06844483 0.88226994 1. 0.92461814]
[0.37047176 0.04556195 0.2556746 0.04403644]



Rysunek 2. Najlepszy uzyskany wykres dokładności (accuracy) od iteracji dla PSO



Rysunek 3. Najlepszy uzyskany wykres dokładności (accuracy) od iteracji dla GA (z przykładu)

W naszym przypadku najlepsze rezultaty uzyskaliśmy korzystając z algorytmu ABC. Wyższa dokładność okupiona jest jednak znacząco dłuższym czasem wykonywania się programu. Algorytm GA miał większą maksymalną dokładność od PSO. Jednak korzystając z algorytmu PSO średnio otrzymaliśmy wyższą dokładność. Ogólnie podsumowując pracę nad projektem, można powiedzieć, że ABC jest przyjemnym algorytmem i nie wymaga wysokiego poziomu wiedzy do zrozumienia jego sposobu działania, ponieważ opiera się na metaheurystyce inspirowanej naturą. Wiedza nabyta w czasie robienia projektu jest bardzo przydatna i może być zastosowana przy napotkaniu zagadnień inteligencji obliczeniowej w przyszłości.

Źródła

- https://upwikipl.top/wiki/Artificial_bee_colony_algorithm
- http://itcraftsman.pl/ogolnodostepne-zbiory-danych-do-machine-learningu/
- https://medium.com/cesar-update/a-swarm-intelligence-approach-to-optimization-problems-using-the-artificial-bee-colony-abc-5d4c0302aaa4

- https://eti.pg.edu.pl/documents/176546/25263566/SZ_wyklad2.pdf