Dokumentacja projektu TM.AE.2.

Autorzy:

Aleksander Krzemiński
Tomasz Sachanowski

1. Treść zadania

TM.AE.2. Niech D będzie danym zbiorem N d-wymiarowych wektorów $\mathbf{x}_j$ liczb rzeczywistych, w którym $x_d \in \{0, 1\}$, natomiast $h : \mathbf{R}^d \rightarrow \mathbf{R}$ funkcją $h(\mathbf{x}) = Threshold(w_0 + \Sigma_{i=1,\ldots,d-1}\ w_i{*}x_i)$. Zaprojektować i zaimplementować algorytm ewolucyjny, który będzie minimalizował wartość funkcji $Loss(h) = \Sigma_N (x_d - h(\mathbf{x}_j))^2$ względem współczynników w. Implementacja powinna umożliwiać wybór spośród dwóch definicji funkcji *Threshold*: 1) $Threshold(y) = 1$ jeśli y >= 0 i 0 wpp., 2) $Threshold(y) = 1/(1 - e^{-y})$. Działanie algorytmu należy zaprezentować w formie prostej aplikacji umożliwiającej zobrazowanie kolejnych kroków jego działania dla zbioru punktów umiejscowionych ręcznie na płaszczyźnie. Ponadto aplikacja powinna umożliwiać wczytanie dowolnego zbioru wektorów d-wymiarowych, wyznaczenie dla niego współczynników w zgodnie z powyższą procedurą, a następnie odczytanie wartości funkcji h dla zadanych wektorów: w tym celu należy przetestować działanie programu dla zbioru danych zawartych pod adresem: https://archive.ics.uci.edu/ml/datasets/Planning+Relax. (UWAGA: Przed wczytaniem danych do programu należy dokonać przekształcenia ostatniej współrzędnej ze zbioru {1, 2} do {0, 1}.) Aplikacja powinna także umożliwiać ustawianie innych parametrów stricte związanych z algorytmem ewolucyjnym (takich jak: wielkość populacji, prawdopodobieństwo mutacji, jej maksymalna siła, itp.). W dokumentacji należy przedstawić swoje przemyślenia nad możliwością wykorzystania powyższego podejścia do problemu klasyfikacji binarnej.

2. Kluczowe decyzje projektowe

Przed przystąpieniem do realizacji programu zdecydowaliśmy, że najlepiej pasującym algorytmem ewolucyjnym do tego zadania będzie *algorytm(μ+λ).* Do realizacji wybraliśmy technologię Python3.X, jako język powszechnie używany w zastosowaniach ML/AI.

3. Opis struktury programu

Program został podzielony na 4 moduły:

- *aplikacja.py*:

  Znajduje się kod interaktywnej aplikacji. Moduł służy do uruchomienia aplikacji dostępowej

- *generujDane.py:*

  Zawiera 2 metody: *iris_two_collection(iris, num_one, num_two)* oraz *generuj_dane()*. Pierwsza z nich jest używana do podziału klas zbiorów irisów. Funkcja ta łączy dwa z trzech zbiorów w jeden i zwraca całość jako dwa niezależne zbiory punktów. Druga funkcja służy do generowania punktów na płaszczyźnie tak, aby ich zbiory były dwoma rozłącznymi klasami.

- *osobnik.py*:

  Znajduje się klasa *Osobnik*, która reprezentuje pojedynczego osobnika populacji w naszym algorytmie ewolucyjnym. Oprócz tego znajdują się tam metody: *__Threshold_1* (funkcja aktywacji realizująca Threshold 1 z treści zadania), *__Threshold_2* (funkcja sigmoidalna), *wartosc_loss* (wyliczająca wartość funkcji straty) oraz metody pomocnicze.

- *populacja.py*:

  zawiera definicję klasy *Populacja* zbudowanej z metod:
  - ❏ *krzyżowanie_interpolacja,*
  - ❏ *krzyżowanie*
  - ❏ *mutacja*
  - ❏ *selekcja_loss_1*
  - ❏ *selekcja_loss_2*

  Jednym z podstawowych zadań, które klasa *Populacja* wykonuje to tworzenie osobników przystosowanych jak najlepiej do wymagań środowiska. Zawiera listy obiektów klasy Osobnik.
  Atrybuty klasy:
  - ❏ mi - wartość μ
  - ❏ lam - wartość *λ*
  - ❏ *pm - prawdopodobieństwo mutacji*
  - ❏ *data - zbiór danych służących klasyfikacji binarnej*

Dodatkowo w projekcie znajdują się dwa pliki jupyter-notebook. Działanie poszczególnych kroków algorytmu jest zobrazowane w main.ipynb oraz w iris_3d.ipynb dla zbioru irysów

4. Lista wykorzystanych narzędzi, bibliotek, itp.

Wykorzystane biblioteki:
- matplotlib - wizualizacja aplikacji
- numpy - operacji na danych
- sklearn.decomposition - wykorzystane do danych irysów
- random - zagwarantowanie losowości

5. Instrukcja użytkownika programu

W trakcie działania aplikacji interaktywnej wyświetlany jest stan osobników(czerwone proste) potomków (żółte proste) oraz zbiór danych(niebieskie i zielone punkty) Mamy możliwość zmiany parametrów za pomocą widgetów:

- wybór funkcji:
  - Threshold_1
  - Threshold_2
- parametry algorytmu ewolucyjnego:
  - μ
  - λ
  - *pm*
- Przycisk „GO” pozwala na przejście o 1 pokolenie do przodu
- „SKIP” do przeskoczenia o 10 pokoleń.

6. Wkład

- koncepcja rozwiązania problemu - T, A
- moduł osobnik.py- T
- moduł *generujDane.py- T,A*
- aplikacja.py- T,A
- populacja.py- T,A
- iris_3d.ipynb- T
- main.ipynb- T
- dokumentacja- A