TAIO Input

Krzysztof Więcław

28 October 2019

Wersja: 1.0

1 Opis modułów

Wejście dla programu obejmuje trzy części:

1. Konfiguracja wczytywania szeregu czasowego
2. Szereg czasowy
3. Konfiguracja sieci neuronowej

Pliki z przykładowymi danymi oraz ich parsery znajdują się w repozytorium https://github.com/Wutus/DataParsersTAIO. Uwaga - ze względu na brak klas, których implementacja nie jest do tej pory jawna, nie jest to kod w wersji kompilowalnej.

2 Konfiguracja wczytywania szeregu czasowego

Plik określa w jaki sposób tworzone są wektory wejścia i oczekiwanego wyjścia dla sieci neuronowej.
Puste linie są ignorowane. Linie zawierające wyłącznie białe znaki są ignorowane. Linie, w których pierwszym niebiałym znakiem jest są ignorowane.
Linia niespełniająca żadnego z powyższych warunków jest parsowana następująco (z pominięciem białych znaków na początku i końcu linii):

```
CHRONOLOGIC|FULL|INDEPENDENT|CUSTOM DATA_PORTION TIME_OFFSET
```

Gdzie:

- MATRIX_TYPE - jeden z typów macierzy określających miejsca, dla których waga musi być zerem:
  - FULL dopuszcza wpływ przeszłości na przyszłość i odwrotnie (macierz z samymi jedynkami)

```
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1
```

  - CHRONOLOGIC dopuszcza tylko wpływ przeszłości na przyszłość (macierz dolno-trójkątna złożona z bloków będącymi macierzami dolnotrójkątnymi)

```
1 0 0 0 0 0 0 0 0 0
1 1 0 0 0 0 0 0 0 0
1 1 1 0 0 0 0 0 0 0
1 0 0 1 0 0 0 0 0 0
1 1 0 1 1 0 0 0 0 0
1 1 1 1 1 1 0 0 0 0
1 0 0 1 0 0 1 0 0 0
1 1 0 1 1 0 1 1 0 0
1 1 1 1 1 1 1 1 1 1
```

  - INDEPENDENT zakłada pełną niezależność przeszłości i przyszłości (macierz, w której jedyne niezerowe elementy znajdują się w blokach, będącymi macierzami dolno-trójkątnymi na przekątnej)

```
1 0 0 0 0 0 0 0 0 0
1 1 0 0 0 0 0 0 0 0
1 1 1 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0
0 0 0 1 1 0 0 0 0 0
0 0 0 1 1 1 0 0 0 0
0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 1 1 0 0
0 0 0 0 0 0 1 1 1 0
```

  - CUSTOM pozwala podać do programu dowolną macierz (program poprosi wówczas o plik wejściowy z taką macierzą w postaci zer i jedynek oddzielonych dowolnymi białymi znakami)

```
1 0 0 0 0 0 0 0 0 0
1 1 0 0 1 0 0 1 0
0 1 1 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0
0 1 0 1 1 0 0 0 0 0
0 0 0 0 1 1 0 0 0 0
0 0 0 0 0 0 1 0 0
0 1 0 0 1 0 1 1 0
0 0 0 0 0 0 0 1 1
```

Przykładowy plik zeros.txt dostępny w repozytorium

- DATA_PORTION - rozmiar porcji danych (wierszy), którą sieć przetwarza w trakcie jednej iteracji.
- TIME_OFFSET Odstęp czasowy (liczba wierszy) przyjmowany pomiędzy pierwszym szeregiem wejścia, a pierwszym szeregiem oczekiwanego wyjścia.

Przykładowy plik (ts_config.txt):

```
#Comments starting with # are allowed
#MATRIX_TYPE DATA_PORTION TIME_OFFSET

CHRONOLOGIC 3 1
```

Przykładowy sposób parsowania (z wykorzystaniem języka C++ i obiektów stringstream w pliku TimeSeriesConfigReader.cpp) oraz plik (ts_config.txt) są dostępne w repozytorium.

3 Szereg czasowy

Plik z szeregiem czasowym zawiera n linii w której każda zawiera k wartości zmiennoprzecinkowych (w których separatorem dziesiętnym jest kropka) oddzielonych od siebie spacjami. Wartości n i k są odczytywane ze struktury pliku i nie są nigdzie jawne podane. Przykładowy plik szeregu czasowego:

```
-0.7005 -0.49244 -0.58608 -0.44382 0.66156 -0.22074
-0.7005 -0.49244 -0.58608 -0.45872 0.70544 -0.13237
-0.70608 -0.49244 -0.56699 -0.51227 0.70105 -0.1147
-0.6952 -0.49244 -0.58608 -0.56732 0.71421 -0.088191
-0.70078 -0.49244 -0.56699 -0.6179 0.71421 -0.088191
```

Pełny plik (time_series_in.txt) i przykładowa implementacja z użyciem biblioteki boost::tokenizer (w pliku TimeSeriesParser.hpp) znajduje się w repozytorium .

4 Konfiguracja sieci neuronowej

Plik określa w jaki sposób przetwarzane są wektory wejścia i oczekiwanego wyjścia dla sieci neuronowej.
Puste linie są ignorowane. Linie zawierające wyłącznie białe znaki są ignorowane. Linie, w których pierwszym niebiałym znakiem jest są ignorowane.
Linia niespełniająca żadnego z powyższych warunków jest parsowana następująco (z pominięciem białych znaków na początku i końcu linii):

```
ERROR_TYPE MAX_ERROR MAX_ITERATIONS ACTIVATE_INPUT BIAS REVERT_ON_ERROR_INCREASE
```

Gdzie:

- ERROR_TYPE (string) - obecnie przyjmuje tylko jedną wartość, MEAN_SQUARED oznaczającą błąd średniokwadratowy,
- MAX_ERROR (float) - górna granica błędu przerywająca działanie algorytmu,
- MAX_ITERATIONS (int) - maksymalna liczba iteracji przerywająca działanie algorytmu,
- ACTIVATE_INPUT (bool) - flaga powodująca wywołanie funkcji aktywacji na danych wejściowych przed rozpoczęciem działania algorytmu,
- BIAS (bool) - flaga dodająca bias do struktury sieci neuronowej,

- REVERT_ON_ERROR_INCREASE (bool) - flaga powodująca powrót do poprzedniego rozwiązania w wypadku zwiększenie błędu.

Przykładowy plik (ts_config.txt):

```
#Comments starting with # are allowed
#ERROR_TYPE MAX_ERROR MAX_ITERATIONS ACTIVATE_INPUT BIAS REVERT_ON_ERROR_INCREASE

MEAN_SQUARED 0.01 6 1 1 0
```

Przykładowy sposób parsowania (z wykorzystaniem języka C++ i obiektów stringstream w pliku ConfigReader.hpp) oraz przykładowy plik (config.txt) są dostępne w repozytorium.