Zastosowania informatyki w medycynie

Automatyczne rozpoznawanie faz snu na podstawie diagramów EEG

Szymon Bagiński

Prowadzący: Dr hab. inż. Robert Burduk

Czerwiec 2018

Spis treści

Wstęp

Celem projektu było stworzenie programistycznej metody automatycznego rozpoznawania faz snu człowieka na podstawie zapisu sygnału elektroencefalograficznego (*EEG*). Częścią zadania nie było opracowanie metody gromadzenia zapisów ani wykonywanie pomiarów sygnału. Zdecydowano się skorzystać z bazy danych dostępnej pod adresem `https://www.physionet.org/physiobank/database/sleep-edfx/`, udostępniającej pliki w formacie *EDF* (European Data Format).

Z uwagi na brak wiedzy dziedzinowej, przy klasyfikacji zdecydowano się na zastosowanie sztucznej sieci neuronowej zamiast analitycznego podejścia do problemu. Przy implementacji sieci skorzystano z otwartoźródłowej biblioteki programistycznej *TensorFlow*. Wykorzystano wersję biblioteki wykorzystującą technologię *CUDA*, dzięki czemu skrócono czas uczenia sieci poprzez przeniesienie wielu równoległych obliczeń na kartę graficzną.

Do wykonania skryptów wczytujących i przygotowujących dane, oraz do uczenia maszynowego został użyty język programowania *Python*. Szczególnie przydatne okazały się pakiety:

- NumPy - obsługa danych tablicowych,
- PyEDFlib - obsługa plików *EDF*,
- PyWavelets - przetwarzania sygnału, Dyskretna Transformata Falkowa,
- TensorFlow - interfejs biblioteki do języka *Python*.

Wszystkie skrypty, które powstały w ramach projektu można znaleźć w publicznym repozytorium pod adresem `www`.

Rozdział 1

Pozyskanie danych

Użyte w projekcie dane zostały pobrane ze strony internetowej []. Wykorzystane zapisy uzyskano w latach 1987-1991 podczas badań nad wpływem wieku na sen. Rekordy pochodzą od osób rasy kaukaskiej w wieku od 25 do 101 lat , które nie przyjmowały żadnych leków związanych ze snem. Sygnały były próbkowane z częstotliwością 100 Hz, a ich klasyfikacja została przeprowadzona przy pomocy reguł Rechtschaffen'a i Kales'a na bazie trzydziestosekundowych segmentów z kanałów *Fpz-Cz* i *Pz-Oz*. Każdy z segmentów został sklasyfikowany jako: przebudzony, nie-REM1, nie-REM2, nie-REM3, REM, niewielka aktywność, poruszenie lub nieokreślony. W tym projekcie został użyty jednak tylko kanał *Pz-Oz*, a rozpoznawanymi fazami były: przebudzony, nie-REM1 + REM, nie-REM2, nie-REM3 + niewielka aktywność. Dokładny opis przebiegu badań znajduje się stronie internetowej bazy danych.

Do trenowania sztucznej sieci neuronowej zostało wykorzystanych dwadzieścia siedem ponad dwudziestogodzinnych zapisów *EEG*, a skuteczność klasyfikacji sprawdzono przy pomocy pięciu rekordów nie użytych przy trenowaniu. Obie listy plików znajdują się w dodatku A i B.

1.1 Wczytywanie danych z pliku EDF

W module *EdfFile.py* została stworzona klasa *EdfFile*, która jako argument konstruktora przyjmuje ścieżkę do pliku z elektroencefalogramem oraz ścieżkę do pliku, w którym znajdują się adnotacje.

W funkcji *__init__* zostało wykorzystanych kilka funkcjonalności z biblioteki PyEDFlib. Funkcja *EdfReader* służy do odczytania danych z pliku i zwraca obiekt go reprezentujący, przez który mamy dostęp do właściwości zapisu takich jak między innymi:

- liczba sygnałów (kanałów) w pliku,
- lista etykiet sygnałów,
- wartości napięcia dla zczytanych próbek,
- lista częstotliwości sygnałów,
- lista adnotacji do faz snu,
- lista czasów przejść między fazami snu.

W tym miejscu jest tworzona także kolekcja obiektów typu *Signal*, które są "opakowaniem" na sygnały obecne w pliku.

W klasie *EdfFile* znajdują się także dwie funkcje odpowiedzialne za przygotowanie danych dla sieci neuronowej. Są to *createOutput* oraz *createInput*. Zwracają one dane odczytane z pliku w postaci macierzy, które następnie w skrypcie *prepare.py* są agregowane. Tutaj także, ma miejsce kontrola poprawności danych i usunięcie błędnych próbek, aby nie umieszczać ich na wejściu sieci.

1.2 Przetwarzanie wstępne

Klasa *Signal* posiada tylko jedną metodę. Jest nią *getEpochs*, która dzieli sygnał na trzydziestosekundowe segmenty, tworzy odpowiadające im obiekty typu *Epoch* i zwraca je w liście. Następnie, w skrypcie *prepare.py* otrzymane w ten sposób obiekty są przekazywane do metody *createOutput* i *createInput* z klasy *EdfFile*. Na każdym segmencie wywoływana jest metoda *extractFeatures*, której zadaniem jest wyekstrahowanie dwunastu cech, które posłużą jako pobudzenia wejść sieci neuronowej.

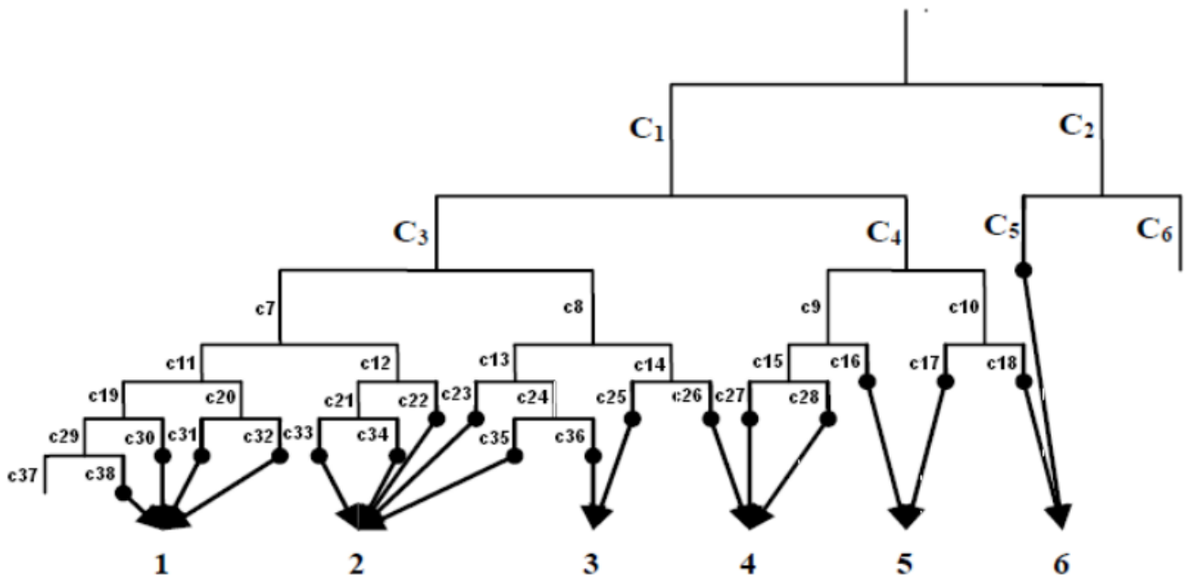
Rozdział 2

Wyekstrahowanie cech z sygnału EEG

Wszystkie cechy zostały pozyskane przy użyciu Dyskretnej Transformaty Falkowej (DWPT - Discrete Wavelet Transform). Jest to przekształcenie podobne do transformaty Fouriera. Główna różnica polega na użyciu innego jądra przekształcenia. W transformacie Fouriera jądrem jest funkcja sinusoidalna, która w dziedzinie czasu jest nieograniczona, a w dziedzinie częstotliwości jest punktem. To zapewnia jej idealną rozdzielczość w dziedzinie częstotliwości natomiast czyni ją nieprzydatną w kontekście analizy czasowej. Ponieważ nośniki falek jak i ich widm nie są punktowe, analiza przy pomocy transformaty falkowej nie posiada idealnej rozdzielczości, ani w dziedzinie czasu ani w dziedzinie częstotliwości (następstwo zasady nieoznaczoności). transformata falkowa służy więc do analizy sygnałów niestacjonarnych, ponieważ dostarcza informacji o czasowo-częstotliwościowych zmianach sygnałów.

W projekcie została użyta Dyskretna transformata pakietów falkowych (DWPT - Discrete wavelet packet transform), która jest rozszerzeniem Transformaty Falkowej. Dzieli ona oryginalny sygnał na dwie części: wysokie i niskie częstotliwości. Transformata falkowa rozkłada tylko niskie częstotliwości dokładnie, a DWPT to jej uogólniona wersja, która również rozkłada pasma wysokiej częstotliwości i prowadzi do powstania kompletnego drzewa pakietów falkowych (rysunek 2.1).

W tym przypadku została użyta DWPT z falką Daubechies rzędu pierwszego do głębokości poziomu siódmego. Skorzystano z gotowej funkcjonalności biblioteki PyWavelets - WaveletPacket.



Rysunek 2.1: Drzewo Dyskretnej transformaty pakietów falkowych

2.1 Podział sygnału na poszczególne pasma częstotliwości

- {0.39 - 3.13 Hz}, Delta, współczynnik = [C38, C30, C31, C32]
- {3.13 - 8.46 Hz}, Theta, współczynnik = [C33, C34, C22, C23, C35]
- {8.46 - 10.93 Hz}, Alpha, współczynnik = [C36, C25]
- {10.93 - 15.63 Hz}, Spindle, współczynnik = [C26, C27, C28]
- {15.63 - 21.88 Hz}, Beta1, współczynnik = [C16, C17]
- {21.88 - 37.50 Hz}, Beta2, współczynnik = [C18, C5]

2.2 Cechy użyte do klasyfikacji sygnału

- Średnie energie (E1, E2, E3, E4, E5, E6) współczynników reprezentujących każde z sześciu wymienionych wyżej pasm
- Całkowita energia (E7), która jest sumą energii wymienionych powyżej
- Stosunek energii pasma Alpha do sumy energii pasm Delta i Theta (E8)
- Stosunek energii pasma Delta do sumy energii pasm Alpha i Theta (E9)
- Stosunek energii pasma Theta do sumy energii pasm Delta i Alpha (E10)
- Średnia wartość współczynników we wszystkich pasmach
- Odchylenie standardowe współczynników we wszystkich pasmach

Rozdział 3

Klasyfikacja faz snu w oparciu o sieć neuronową

normalizacja struktura sieci uzyte funkcjonalnosci TensorFlow utworzone funkcje trenowanie wyniki, może w podsumowaniu

Podsumowanie

Dodatek A

Pliki wykorzystane do treningu

SC4001E0-PSG.edf
SC4002E0-PSG.edf
SC4011E0-PSG.edf
SC4012E0-PSG.edf
SC4021E0-PSG.edf
SC4022E0-PSG.edf
SC4031E0-PSG.edf
SC4032E0-PSG.edf
SC4041E0-PSG.edf
SC4042E0-PSG.edf
SC4051E0-PSG.edf
SC4052E0-PSG.edf
SC4061E0-PSG.edf
SC4062E0-PSG.edf
SC4071E0-PSG.edf
SC4072E0-PSG.edf
SC4081E0-PSG.edf
SC4082E0-PSG.edf
SC4091E0-PSG.edf
SC4092E0-PSG.edf
SC4122E0-PSG.edf
SC4131E0-PSG.edf
SC4141E0-PSG.edf
SC4142E0-PSG.edf
SC4151E0-PSG.edf
SC4152E0-PSG.edf
SC4161E0-PSG.edf

Dodatek B

Pliki wykorzystane do testowania

SC4101E0-PSG.edf
SC4102E0-PSG.edf
SC4111E0-PSG.edf
SC4112E0-PSG.edf
SC4121E0-PSG.edf