Metody klasyfikacji i redukcji wymiaru 15.03.2024

Projekt nr 3

Paweł Lorek

Termin oddania: xx.xx.2024, 23:59 (SKOS)

1 Wstęp

Do oddania: Raport w postaci pliku `.pdf` oraz pliki w `Python` (ver 3).
Raport ma być napisany tak, by był zrozumiały również dla osoby, która nie uczęszcza na wykład: Oprócz wyników powinno się znaleźć sformułowanie problemu, itp.

Uwaga: Poniżej znajduje się opis co powinno znaleźć się w projekcie. Twój projekt powinien te wszystkie elementy zawierać, ale nie musi się tylko i wyłącznie do nich sprowadzać.

Projekt można oddawać w zespołach 2-osobowych.

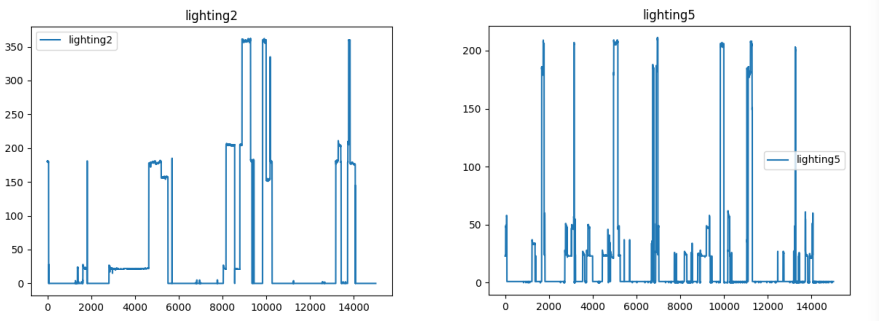
2 Klasyfikacją urządzeń domowych na podstawie poboru prądu.

W pliku `house3_5devices_train.csv` podany jest pobór prądu (tzw. "active power") zmierzony dla 5 urządzeń w odstępach ok 20 sekundowych. Dane są fragmentem zbioru REDD http://redd.csail.mit.edu/, były mierzone w jednym domu w okresie od 16.04.2011, godz. 5:11:43 do 21.04.2011 godz. 7:21:44 (ok. 15000 pomiarów). Plik ma następujący format:

```
time,lighting2,lighting5,lighting4,refrigerator,microwave
1302930703,180,23,195,117,2
1302930721,181,23,195,119,2
1302930738,180,23,195,117,2
1302930765,181,23,195,117,2
```

Pierwsza kolumna to timestamp (można z niego odczytać dokładną datę i godzinę) – ignorujemy go w procesie uczenia i testowania, kolejne wiersze traktujemy jako kolejne punkty czasowe. Pobór mocy każdego z urządzeń podany jest w kolejnych 5 kolumnach.

Jest tam 5 urządzeń: `ligthing2, lighting5, lighting4, refrigerator, microwave`. Oto jak wyglądają przykładowe dwa:



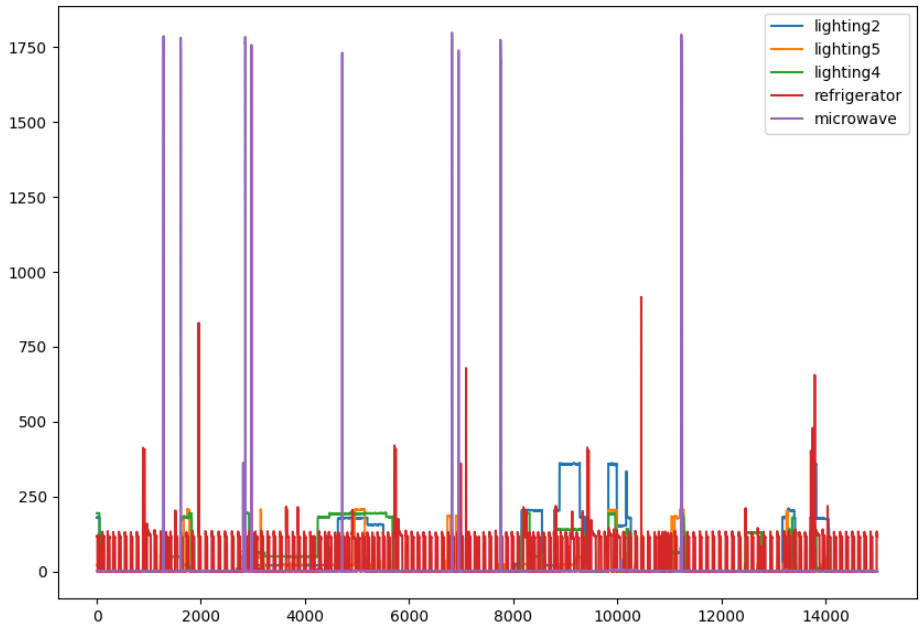
Skrypt `data_visualise.py` tworzy powyższe wykresy. Trzeba podać plik .csv w opcji `--file`, z kolei opcja `--separate` oznacza czy każde urządzenie ma być na osobnym obrazku (`yes`) czy też wszystkie na jednym (`no`, domyślnie). Powyższe obrazki były "wyprodukowane" poleceniem

```
python data_visualise.py --file house3_5devices_train.csv --separate yes
```

Z kolei polecenie

```
python data_visualise.py --file house3_5devices_train.csv
```

pokazuje



3 Zadanie

Napisz skrypt, który uruchamia się następująco:

```
python classify_devs_IndexNr.py --train house3_5devices_train.csv
                            --test test_folder --output results.txt
```

gdzie `house3_5devices_train.csv` to wspomniany plik, natomiast `test_folder` zawiera pliki

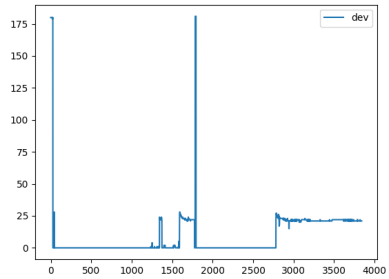
```
dev1.csv
dev2.csv
dev3.csv
...
```

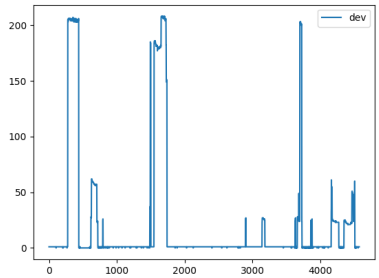
W Moodle załaczony jest przykładowy folder (`test_folder.tar.gz`) – tam jest 6 plików, ale może być ich dowolna liczba. Format każdego z plików jest następujący:

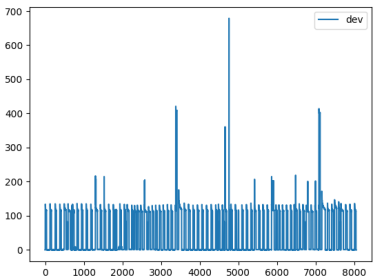
```
time,dev
1303001413,0
1303001430,0
1303001487,134
1303001509,132
1303001526,131
```

Czyli pierwsza kolumna to timestamp (ignorujemy), druga to pobór prądu. W przykładowym `test_folder.tar.gz` umieściłem 6 plików, które de facto wziąłem z pliku treningowego. Przy sprawdzaniu właściwym wezmę dane z okresu 21.04.2011 7:21:44 – 22.04.2011 11:37:18. Proszę zauważyć, że każdy z plików `dev*.csv` może mieć różną długość (pomiar może być z kilku minut lub kilku godzin).

Na tych plikach również można używać `data_visualise.py`







Zadanie. Trzeba sklasyfikować każdy z plików `dev*.csv` jako jedno z urządzeń: `ligthing2, lighting5, lighting4, refrigerator, microwave`.
Następnie wynik proszę zapisać do pliku podanym w parametrze `--output`, plik powinien mieć następujący format

```
file,dev_classified
dev1.csv,ligthing2
dev2.csv,ligthing2
dev3.csv,refrigerator
dev4.csv,microwave
dev5.csv,lighting5
dev6.csv,lighting4
```

Przy okazji – w powyższym przykładzie jest podana poprawna klasyfikacja wszystkich 6 plików z przykładowego `test_folder.tar.gz`.

4 Ukryte modele Markowa

Zauważmy, że zadaniem jest de facto klasyfikacja szeregów czasowych. Można tutaj zastosować ukryte modele Markowa (HMM) korzystając z gotowych bibliotek, proponuję zapoznać się z jedną z bibliotek:

- https://hmmlearn.readthedocs.io/en/latest/
- https://github.com/lopatovsky/HMMs/

  (tutaj warto oglądnąć

  https://github.com/lopatovsky/HMMs/blob/master/hmms.ipynb)

Podejście może być następujące:

- Dla każdego z urządzeń `1=ligthing2, 2=lighting5, 3=lighting4, 4=refrigerator, 5=microwave` wyuczamy się osobno pięciu różnych modeli HMM $\lambda_1, \ldots, \lambda_5$.
- W każdym modelu przyjmujemy, że jest ileś ukrytych stanów, natomiast każdy stan "emituje" (=obserwacje) zmienną losową o rozkładzie normalnym.
- We wspomnianych bibliotekach są wbudowane wersje HMM, gdzie obserwacje mają rozkład normalny.
- Uczenie się parametrów (= alg. EM = alg. Bauma-Welcha) jest również w tych bibliotekach. Jednym z parametrów jest liczba ukrytych stanów, które trzeba podać przed uczeniem. Sugeruję zawsze spróbowanie kilku (np. $2, 3, \ldots, 8$) i wybranie "najlepszego" modelu. "Najlepszy" można wybrać patrząc jak model pasuje do danych treningowych, można np. patrzeć na likelihood – również dostępne w bibliotekach, lub (bo łatwo model przetrenować) zapoznać się (i użyć) z kryteriami AKAIKE[1] lub BIC[2].
- **Klasyfikacja**. Dla każdego z pięciu wyuczonych modeli $\lambda_1, \ldots, \lambda_5$ patrzymy jak dobrze dane urządzenie `dev*.csv` do niego "pasuje" (licząc likelihood). Ostatecznie jeśli największe likelihood było np. dla modelu λ_1 to je klasyfikujemy jako odpowiadające modelowi urządzenie (model λ_1 odpowiada `lighting2`).

[1] https://en.wikipedia.org/wiki/Akaike_information_criterion

[2] Bayesian information criterion,
https://en.wikipedia.org/wiki/Bayesian_information_criterion

Wymagane jest, by użyć powyższego sposobu wykorzystującego HMM. **Dodatkowo** można użyć innych sposobów klasyfikacji. Jeśli będzie więcej niż 1 sposób, to plik `classify_devs_IndexNr.py` powinien przyjmować dodatkowy argument `--alg ALG`, gdzie `ALG` to algorytm, domyślnie powinno to być `hmm`, pozostałe opcje proszę jasno opisać w raporcie.