UNIVERSITEIT TWENTE.

Data Science [201400174]

Course year 2018/2019, Quarter 1B

DATE
November 15, 2018

EXCERPT

Feature extraction from Time Series data [TS]

TEACHERS
Maurice van Keulen
Christin Seifert
Mannes Poel
Karin Groothuis-Oudshoorn
Faiza Bukhsh
Miha Lavric

COURSE COORDINATOR
Christin Seifert
Maurice van Keulen

PROJECT OWNERS
Faiza Bukhsh
Karin Groothuis-Oudshoorn
Maurice van Keulen
Mannes Poel
Michel van Putten
Luc Wismans

Topic 4

Feature extraction from Time Series data [TS]

4.1 Introduction

Space and time are the fundamental concepts with which we build our own reality. Hence, spatial and time series data are extensively available in every scientific and application domain. Typical examples include weather data in meteorology, stock market data in economics, energy data in industries, physiological signals such as ECG/EEG in medicine. In general, these spatio-temporal data are analyzed either to understand the underlying mechanism generating the data, to find patterns for inferring high level concepts (i.e., classification or clustering) or to predict the future behaviors of the systems based on their past data.

4.1.1 Global description of the practicum and project

This practicum will focus mainly on key techniques for analyzing time-series data such as peak detection, filters, Fourier analysis (FFT), Dynamic Time Warping (DTW) and prediction models.

4.1.2 Study material and tools

We suggest use of Python and Rodeo IDE[1] for this topic but you can also choose your language and IDE of convenience. Also, we recommend Anaconda[2] distribution to collectively install the required Python packages such as NumPy, Pandas, matplotlib, scikit-learn. Section 4.2.7 gives a short tutorial on the necessary basic concepts in Python and links for installing Anaconda and Rodeo. Note that the tutorial is optional and can be safely skipped by those who have sufficient knowledge in Python.

The following reading materials are suggested for different topics,

1. Time domain features
   (a) http://pandas.pydata.org/pandas-docs/stable/api.html#api-dataframe-stats

[1] https://github.com/yhat/rodeo
[2] https://docs.continuum.io

   (b) https://docs.scipy.org/doc/scipy-0.7.x/reference/stats.html

2. Frequency domain features

   (a) https://github.com/nipunbatra/ProgramaticallyUnderstandingSeries/blob/master/ft.ipynb

   (b) https://www.princeton.edu/~cuff/ele201/kulkarni_text/frequency.pdf

3. Dynamic time warping

   (a) http://preview.tinyurl.com/h2hmsp8

   (b) http://www.phon.ox.ac.uk/jcoleman/old_SLP/Lecture_5/DTW_explanation.html

   (c) https://github.com/nipunbatra/ProgramaticallyUnderstandingSeries/blob/master/dtw.ipynb

4. Time series prediction

   (a) https://github.com/aarshayj/Analytics_Vidhya/blob/master/Articles/Time_Series_Analysis/Time_Series_AirPassenger.ipynb

   (b) http://www.ulb.ac.be/di/map/gbonte/ftp/time_ser.pdf

   (c) https://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/

5. Using Python for data analysis

   There material goes beyond the scope of this topic but might be interesting for advanced readers,

   (a) https://github.com/icounter/python-data-mining-and-text-mining/blob/master/Python4DataAnalysis.pdf

   (b) https://www.youtube.com/user/sentdex/playlists?shelf_id=5&view=50&sort=dd

4.1.3 Deliverables and obligatory items

The practicum requires individual files including images, results and source code to be combined together as a zip files and upload to the blackboard. For each question, the source code and console outputs can be put together in a single text file. Also, add comments and explanations as needed to support the code and results.

Topic teacher: Maurice van Keulen

4.2 Description of the practical assignments

In this practicum, we will focus mainly on two topics, i.e., classifying time series data and time series prediction. We will begin with simple questions to describe and better understand the dataset recommended for this topic. Then, will explore key techniques for spatio-temporal mining such as peak detection, filters, Fourier analysis (FFT), Dynamic Time Warping (DWT) and correlation analysis.

4.2.1 Datasets

We will work with the following data sets,

1. For classifying time series data, we will work with an activity recognition data from inertial sensors (e.g., accelerometer and gyroscope). It has both raw data and common derived features. Also, a youtube video gives an overview of the dataset.

https://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones

The basic description about the dataset and the files are given in the README.txt

2. For time series prediction, we will work with data from the Berkeley Earth Surface Temperature Study.
https://www.kaggle.com/berkeleyearth/climate-change-earth-surface-temperature-data

This extensive data set is created by combining multiple existing archives and it allows groupings by country or city of interest.

4.2.2 Understanding the dataset

30 minutes

☞ **4.1**

Consider the activity recognition data set with *derived features* for this question.

(a) Describe the dataset by printing the summary statistics for the columns (i.e., all the input features)? Typical examples include count, mean, std, min, max, upper and lower percentiles. Refer to the tutorial in Section 4.2.7 for some useful built-in methods from Pandas.
Note: *UCI HAR Dataset/train/X_train.txt* and *UCI HAR Dataset/test/X_test.txt* are the training and testing data with all the input features.

(b) Plot the number of datapoints for each user activity (i.e., output class) as a histogram. Report if the data set balanced or not.
Note: *train/y_train.txt* and *test/y_test.txt* has the output classes (i.e., groundtruth or labels).

(c) Plot some of the derived features (atleast 10) and compare it for different user activities (i.e., output class). Report whether it is possible to discriminate between activities based on those chosen derived features. Visually inspect and report the nature of distributions (e.g., normal distribution or not, skewness, modality, etc.) for atleast 10 derived features.

□

4.2.3 Preprocessing and time-domain features

2 hours

Consider the activity recognition data set with *raw data* for this question.

☞ **4.2**

(a) Step detection is the most common output of the ubiquitous activity recognition apps and wearable devices. Although many sophisticated algorithms exist, the basic concept in all those methods is to find the peaks and count them.
Apply different peak finding methods described in the link below on the accelerometer data. You can choose to focus on just one axis which has the highest variations in acceleration. Report your views on their effectiveness and which one is favorable for this dataset.
https://github.com/MonsieurV/py-findpeaks#detect_peaks-from-marcos-duarte You will find the documention and sample code of each peak detection algorithm by clicking on the link under each graph (in the above github page).

(b) Accelerometer data is fairly noisy especially the high frequency components interfere with the signals of our interest (i.e., those corresponding to user steps).
Apply a low-pass and high-pass band filter to the raw data. You can choose to focus on just one axis which has the highest variations in acceleration. Report which of the filters is better for our data set and task. Note that different filters are readily supported by SciPy module.

https://docs.scipy.org/doc/scipy-0.18.1/reference/generated/scipy.signal.butter.html#scipy.signal.butter

(c) We notice many derived features in the activity recognition data set.
Write code to recreate some of those time-domain features features such as mean, standard deviation, kurtosis and signal magnitude area from the raw data. Report (by visual investigation) whether some of the features (e.g., SMA) have the potential to discriminate between multiple activities. Note: The derived data set uses fixed-width sliding windows of 2.56 sec and 50% overlap (128 readings/window) is used to generate the features.

□

4.2.4 Frequency domain features

2 hours

Consider the activity recognition data set with *raw data* for this question.

☞ **4.3**

(a) Fourier transform of a time signal helps us to understand the component frequencies present in that signal. A seemingly noisy signal in time domain can show useful patterns (dominating frequencies) in the frequency domain.
Apply FFT on the raw data and visualize the proportions of component frequencies under the following conditions,
  (a) Before and after using any filters for pre-processing.
  (b) For a short sequence of walking and being idle.
Also, report whether the FFT coefficients are useful to (visually) discriminate the individual activities.

□

4.2.5 Time series matching and classification

6 hours

For this question, we will work with both activity recognition data and global surface temperature data.

☞ **4.4**

As 2 time series data sequences usually vary in lengths and have local phase shifts (for example, it is not straight forward to use popular distance metrics such as Euclidean distance on time series data. Dynamic Time Warping (DTW)[8][9] is a similarity measuring technique often used for the above described scenarios. Simple classifiers such as K-Nearest Neighbors can achieve high accuracy while using DTW as a distance metric for classifying/clustering time series data.

The following tutorial applies DTW on some of the selected derived features from the activity recognition data set.

https://github.com/markdregan/K-Nearest-Neighbors-with-Dynamic-Time-Warping/blob/master/K_Nearest_Neighbor_Dynamic_Time_Warping.ipynb

(a) Apply the same technique on raw data (pre-processed with filters) and compare it with the results based on the derived features.
(b) Plot the yearly temperature for the following countries: Norway, Finland, Singapore and Cambodia. Use DTW to measure the similarities between the temperature data of these countries and reflect on the results.

□

4.2.6 Curve fitting and prediction

6 hours

Finally, we will cover time-series prediction. As this technique is not relevant for activity recognition, we will use other time-series data for reference as shown in [6,11].

☞ **4.5**

(a) Apply the techniques from [6,11] to determine whether global warming is true for your country and city of choice, and predict the raise in temperature over the next 5 years.
- Evaluate the stationarity of the original data and pre-processed data with various standard techniques (e.g., first difference, seasonal difference, etc.)
- Compare the quality of the prediction models and reason why a particular choice was made.

(b) Repeat the DTW assignment from the previous question on this data set on the de-trended data and reflect on the results.

□

4.2.7 Basic tutorial on Python and common modules

We will cover basic concepts such as data types, output to the console, conditions, loops and simple examples from NumPy and Pandas packages. This tutorial is inspired by some of the tutorials already available [1,2,12] which can be referred for advanced tutorials on Python and relevant modules.

1. Let's get started by printing the following statement:
The countdown begins , ... , 3 , 2 , 1 , 0.5 , Go!
Assign the comma separated values to individual variables.

2. Arrays or (are) Lists:
There is no native array data structure in Python but Lists which can support mixed data types [2]. Create an empty list and append the values from the individual variables created for the above question to the list.

3. Using operators on Strings and Lists:

   (a) How can we efficiently print repeating patterns such as, *blah blah blah blah blah blah blah blah blah blah*

   (b) Define 2 lists and join them by using '+' operator.

   (c) Repeat a list n times by using '*' operator.

4. IF statements, Boolean, in and is operators, and FOR loops:

   (a) Check if the value of the variable half used in question 1 is 0.5. If yes, print 'the value is correct', else, print 'the value is wrong'.

   (b) What is the output of the following code?

   ```
   Half = 0.5
   half = 0.5
   if Half is half:
       print("they are the same instances")
   elif Half == half:
       print("they have the same values")
   ```

   How can you make the if statement true?

   (c) Check if "Go!" is in the list created for question 2 and print the result. Hint: Use in operator.

   (d) Iterate over the individual elements in the list created for question 2 and printe each element.

Note: Python uses indentation to define the code blocks. The agreed standard is 4 spaces, although tab or other number of space would also work.

5. Functions and modules:

   (a) Write a recursive function to print the first ‘n’ numbers in the Fibonacci sequence.

   (b) Python modules are nothing but simple .py files that contain executable statements and functions.
   A library can be installed using pip as follows: *! pip install pandas*
   A library can be imported by: *import pandas as pd*

6. NumPy arrays:
   NumPy is an extension for Python with support for array data structures and high-level operations on arrays.

```
# Create 2 new lists height and weight
height = [1.87,  1.87, 1.82, 1.91, 1.90, 1.85]
weight = [81.65, 97.52, 95.25, 92.98, 86.18, 88.45]

# Import the numpy package as np
import numpy as np

# Create 2 numpy arrays from height and weight
np_height = np.array(height)
np_weight = np.array(weight)
```

   Now we can perform element-wise calculations on height and weight. For example, you could take all 6 of the height and weight observations above, and calculate the BMI for each observation with a single equation. These operations are very fast and computationally efficient. They are particularly helpful when you have 1000s of observations in your data.

```
# Calculate bmi
bmi = np_weight / np_height ** 2

# Print the result
print(bmi)
```

   Another great feature of Numpy arrays is the ability to subset. For instance, if you wanted to know which observations in our BMI array are above 23, we could quickly subset it to find out.

```
# For a boolean response
bmi > 23

# Print only those observations above 23
bmi[bmi > 23]
```

7. Pandas module:

   The most popular data structure (atleast for data science) in Python is from Pandas called DataFrame which allows to represent and manipulate data in tabular form. For example, here is a dictionary containing details about BRICS countries.

```
dictionary = {"country": ["Brazil", "Russia", "India", "China", "South Africa"],
       "capital": ["Brasilia", "Moscow", "New Dehli", "Beijing", "Pretoria"],
       "area": [8.516, 17.10, 3.286, 9.597, 1.221],
       "population": [200.4, 143.5, 1252, 1357, 52.98] }

print(dictionary)
```

   This dictionary can be simply converted into a dataframe as follows:

```
import pandas as pd
brics = pd.DataFrame(dictionary)
print(brics)

Now, various built-in functions can be used to
manipulate either the rows or columns.

print(brics.info())
print(brics.head())
print(brics.columns())
print(brics.values())
print(brics.[’population’])
print(brics.loc[:,[’population’,’area’]])
print(brics.sort_values(by=’population’))
```

8. matplotlib:

   The pyplot module from matplotlib provides MATLAB like interface to produce high-quality figures. A typical example is given below,

```
import numpy as np
import matplotlib.pyplot as plt

def f(t):
    return np.exp(-t) * np.cos(2*np.pi*t)

t1 = np.arange(0.0, 5.0, 0.1)
t2 = np.arange(0.0, 5.0, 0.02)

plt.figure(1)
plt.subplot(211)
plt.plot(t1, f(t1), ’bo’, t2, f(t2), ’k’)

plt.subplot(212)
plt.plot(t2, np.cos(2*np.pi*t2), ’r--’)
plt.savefig("path_to_file") # save figure
plt.show()
```

9. Reading and writing files:

   Here are some examples for reading files,

```
import pandas as pd

df = pd.read_csv(’path_to_file’,header=None, sep=’\s+’)

df.columns = [’CRIM’, ’ZN’, ’INDUS’, ’CHAS’,
              ’NOX’, ’RM’, ’AGE’, ’DIS’, ’RAD’,
              ’TAX’, ’PTRATIO’, ’B’, ’LSTAT’, ’MEDV’]
df.head()
```

   The file path can be a url as well,

   https://raw.githubusercontent.com/rasbt/python-machine-learning-book/master/code/datasets/housing/housing.data

4.2.8 Additional tutorials and references

1. http://pandas.pydata.org/pandas-docs/stable/10min.html

2. http://www.scipy-lectures.org
3. https://www.youtube.com/playlist?list=PLfZeRfzhgQzTMgwFVezQbnpc1ck0I6CQl
4. https://github.com/ThunderShiviah/samsung_android_time_series_analysis/blob/master/notebooks/time_series_analysis.ipynb
5. Analyzing features for activity recognition, Tâm Huynh et al.
6. https://github.com/aarshayj/Analytics_Vidhya/blob/master/Articles/Time_Series_Analysis/Time_Series_AirPassenger.ipynb
7. https://github.com/nipunbatra/ProgramaticallyUnderstandingSeries/blob/master/ft.ipynb
8. https://github.com/nipunbatra/ProgramaticallyUnderstandingSeries/blob/master/dtw.ipynb
9. http://www.phon.ox.ac.uk/jcoleman/old_SLP/Lecture_5/DTW_explanation.html
10. https://www.kaggle.com/berkeleyearth/climate-change-earth-surface-temperature-data
11. https://www.analyticsvidhya.com/blog/2015/12/complete-tutorial-time-series-modeling/
12. https://www.learnpython.org
13. https://github.com/markdregan/K-Nearest-Neighbors-with-Dynamic-Time-Warping/blob/master/K_Nearest_Neighbor_Dynamic_Time_Warping.ipynb
14. Activity Recognition from Accelerometer Data, Nishkam Ravi et al.