Visualising Data

Practical Assignment 3

Informatics 1 for Biomedical Engineers

Graz University of Technology

Abstract

Thus far we have read read the binary signal data and attempted to detect heartbeats. The heartbeat detection approach from assignment 2 was a little primitive and may have allowed false data into the results. Before working with this data, it is needed to remove these erroneous entries. As we detected the heartbeats we will now calculate the average beats per minute as a sliding window. Moreover, we will train functions to describe our signal data and the bpm. Finally, to help others understand any findings, we will now visualise the results using Matplotlib.

---

1. Tasks

This final assignment builds on-top of the previous tasks. We will use the outputs of assignments 1 and 2 as input for assignment 3. If you are unsure your code was correct feel free to use the reference solutions to generate your data[1][2].

Again, the data provided for this course are from the The PTB Diagnostic ECG Database[3].

Our program should visualise the data generated thus far. Additionally, we want to take care of a slight error we might have made during the heartbeat detection. This is vital as we want to calculate the BPM and also want to learn functions that describe our data. Finally, this time we want to check if any of the files we need are actually there!

The visualisation is done in matplotlib. Up to 5 Bonus points awarded for a nice plot. Please note, that you need to be positive without the bonus points (> 25 points combined from all assignments) to be eligible for them.

1.1. Open a data packages (2 pt)

Open the pickle you created in assignment 1 and the CSV from assignment 2. Use try-except blocks to check if the files are available and can be used. If no, then use sys.exit() to end the program. You are allowed to use a single general *except:* (no need to separate which error occurred)

1.2. Select sensor (1pt)

Similar to assignment 2 we wish to select which sensor data we want to visualise. The name of the signal is passed as the second command line parameter. You will need to look through your loaded data structure and check which index has the wanted name.

1.3. Minimum heartbeat threshold (2pt)

Due to the flaws in the simple detection mechanism you may have detected individual heartbeats twice. You will now have to implement a minimum distance between each beat. Reject all entries that fall within this limit. Example: minimum = 50 and beats = [1, 100, 120, 200, 500] should result

[1] `https://palme.iicm.tugraz.at/wiki/Info1BM/Assignment_1`
[2] `https://palme.iicm.tugraz.at/wiki/Info1BM/Assignment_2`
[3] `https://www.physionet.org/physiobank/database/ptbdb/`

in [1, 100, 200, 500]. 120 is rejected as it is within the threshold. You could do this in a single line in the form of a list comprehension. This is, however, not mandatory! (Note: Make sure you do not remove the very first entry!) How you set this threshold is up to you. Look at your data and set a sensible value. Explain your choice in a comment in your source code!

1.4. Beats per Minute (5pt)

Now we want to calculate the average BPM as a sliding window based on the last 10 heartbeats. The easiest way is to calculate the time between your current beat and the 10^{th} before that and use this to calculate the beats per minute. Naturally, you will not get a result for the first 9 beats. This is expected.

1.5. Fit polynomials for the signal and the bpm (5pt)

Use numpy and its *polyfit()* function to learn a polynomial that describes your data. Do this for the BPM and the signal data. The function will expect a parameter defining how many dimensions your resulting polynomials will have. Again, try multiple values and explain your final choice in a comment. (Hint: numpy provides the *poly1d()* function which can be very helpful during the visualisation)

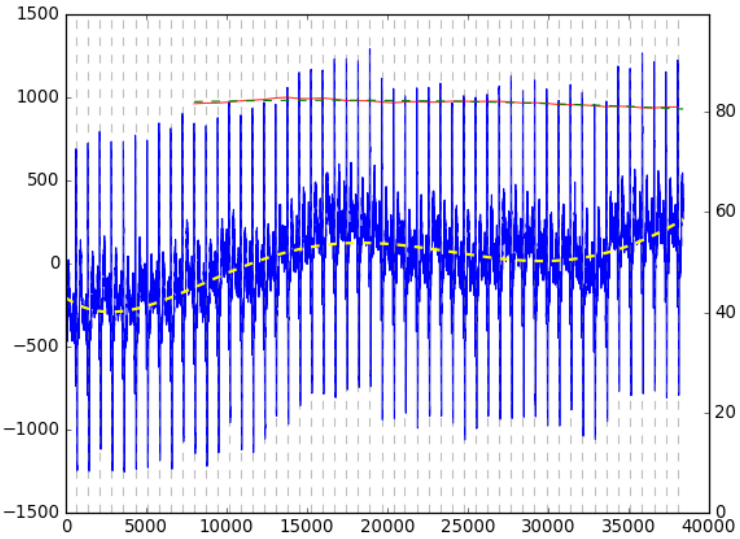
1.6. Plot all your data (10 pt)

Use *matplotlib* to plot all data. We want to see the following:

- The sensor data
- The learned function through the sensor data
- The detected heartbeats
- The calculated average bpm
- The learned function through the average bpm

Note that the averaged BPM will not start at 0 as you can't calculate an average for the first 9 beats. Below is an example how this visualisation might look like. (Run with *>python assignment_3.py s0010_re i*). There are no strict regulations how you select your lines and colours here. Just make sure everything is identifiable! Use the *twinx()* function to be able to

plot 2 different types of data onto the same image. (Your signal data related things on one axis and the heartbeat related on the other.)



For reference:

- Blue: signal data
- Yellow: function through signal data
- grey dashed: detected heartbeats
- red: average BPM
- green dashed: learned function through BPM

1.7. BONUS: Make your plot pretty (5 pt)

The plot thus far is rather useless. We need to annotate it and make sure the viewer understands what is what. You can get 1 bonus point for each of the following aspects:

- Add a legend
- Add axis labels
- Make sure you can identify everything in grey-scale (imagine you were to print it out).

- Add a padding within the image so the data values are not at the very edge. (Use xlim and ylim)
- Select a good image dimension so all data is nicely visible but no space is wasted. (Usually wide but flat)

For this bonus task you can use the *seaborn* package. You may need to install it but it is not mandatory!

1.8. Command Line Parameters

Your program should work with command line parameters. Execute it with the command *python assignment_3.py <dataset_name> <sensor_name>*. Please note that this time we do not want any file ending when we define the dataset! Look at the automated tests for an example call to your program. Your submission should expect any files in the same directory as the python source!

1.9. Output

Your program should output the visualised data as an image file. (Use .pdf as file type)

1.10. Debug Output

During development and testing debug outputs can be quite helpful. For finished programs on the other hand think about what information is vital to the user. In most cases it is only important to report when something went wrong. This means as long as everything goes as planned your submissions should **not** create any output to the console (no print statements). If you detect an error, you can broadcast it and exit the program. But please note that error handling is not part of this assignment.

2. File Headers

All python source files have to contain a comment header with the following information:

- Author: – Your name
- MatNr: – Your matriculate number
- Description: – Purpose of the file
- Comments: – Any comments as to why if you deviate from the task or restrictions

Please feel free to copy and paste the example and change its contents to your data. (Any comments will not be subject for plagiarism checks)! Note: Depending on your reader you may have to manually fix the whitespaces and indentation!

Example:

```
##############################################################################
# Author:      Patrick Kasper
# MatNr:       0730294
# Description: The main file. Assignment only has 1 file...
# Comments:    This is the example comment. I just made it a bit
#              longer so it spans across multiple lines.
##############################################################################
```

3. Allowed packages

The following packages are allowed to use in this assignment:

- sys
- pickle
- numpy
- matplotlib
- seaborn (can be helpful for the bonus task!)

4. Restrictions

- Do not add any bloat to your submission
  - No useless code
  - No debug output
  - No extra files
  - Try to avoid *.__MACOSX* or *.DS_STORE* folders in your submission archives.
- Use built-in functions where possible
  - except when explicitly stated—such as Task 1.3
- Do NOT load extra packages (no imports) except those listed in section 3
- Do NOT require any extra command line arguments!

5. Coding Standard

For this lecture and the practicals we use PEP 8 [4] as a coding standard guideline. Please note that whilst we do not strictly enforce all these rules we will not tolerate messy or unreadable code. Please focus especially on the following three aspects.

Language Programming is generally done in English. This is to ensure someone else at the other end of the world can read your code. Please make sure all sources you submit are thus written in English. Whilst grammatical or typographical errors won't be penalised in any way we ask you to double-check your code before submitting.

Spaces, not tabs. Indent your files with 4 spaces instead of tabs. PEP 8 forces this in python 3 (whilst allowing more freedom in python 2). Most editors have an option to indent with 4 spaces when you press the tab button!

[4] `https://www.python.org/dev/peps/pep-0008/`

Descriptive names. Use descriptive names for variables where possible! Whilst a simple i can be enough for a simple single loop code can become very messy really fast. If a variable has no purpose at all use a single underscore (_) for its name.

Max 72 characters. Do not have lines longer than 72 characters. Whilst PEP 8 allows 79 in some cases we ask you to use the lower cap of 72 characters per line. Please note that this includes indentation.

6. Self testing

There will be no self tests for this assignment. Look at your result to see if it makes sense (all source data was already tested via the previous tasks)

7. Automated Tests

We will test your submissions automatically. The test machines will have the latest anaconda version on the 5th of October 2016. Your file will be executed with the following command:

```
>python assignment_3.py s0010_re i
```

Please make sure your submission follows all the restrictions defined in this document. Your program will have a limited runtime of 2 minutes. If your submission exceeds this threshold then it will be considered non-executable. Please note that this is a very generous amount of time for any of the tasks in this course.
If your submission fails any of the automated tests we will look into your code to ensure the reason was not on our end and to evaluate which parts of your code are correct and awards points accordingly.

In particular, we will be testing the following aspects:

- General coding errors (Does the program even run?)
- Do the files contain the correct comment headers?
- Your output with the provided input (*assignment_2.py s0010_re i*)
- Your output with different input

8. Submission

8.1. Deadline

6^{th} of January 2017 at 23:59.

Any submission handed in too late will be ignored with the obvious exception of emergencies. In case the submission system is globally unreachable at the deadline it will automatically be pushed back for 24 hours.
If for whatever reason you are unable to submit your submission prior to the deadline please contact your tutor BEFORE the deadline.

8.2. Uploading your submission

Assignment submissions in this course will always be archives. You are allowed to use .zip or .tar.gz formats.
In addition to your python source files please also pack a *readme.txt*. The file is mandatory but its contents are optional. In this file please write down how much time you spent on the assignment and where you ran into issues. This is important to us as immediate feedback so we can adjust the tutor sessions. Please note that all submissions will be anonymised prior to being read. Hence please feel free to be honest (but do stay polite).
Upload your work to the Palme website. Before you do please double-check the following aspects:

- File names and folder structure (see below)
- Comment headers in every source file
- Coding standard upheld
- You do not hard-code the file name and the frame amount!

8.3. Your submission file

```
<your_matr_number>.zip
    assignment_3.py
    readme.txt
```

Do not submit any additional files. You do not need to submit the image your program creates!