Data Forecasting and Error Calculation

Practical Assignment 2

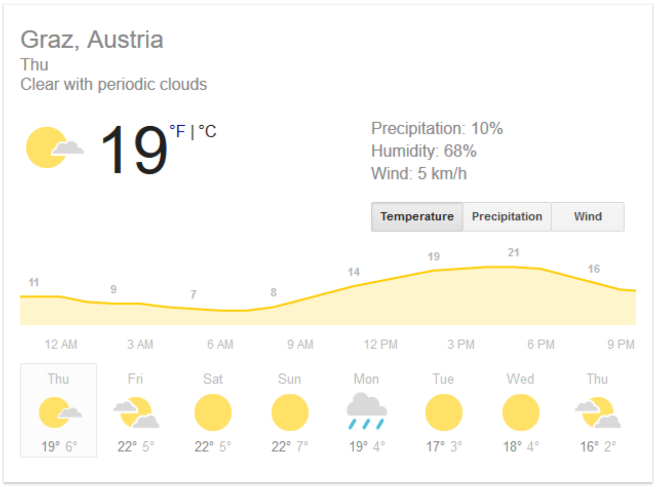
Informatics 2 for Biomedical Engineers

Graz University of Technology

Abstract

Forecasting—predicting the net few steps— is one primary purpose of time series analysis. Common applications range from weather or stock price prediction to error detection in machinery. Python provides a set of very powerful tool-kits to quickly analyse and predict time dependent data. Among others, the packages scipy and scikit-learn are widely used for these kind of tasks. The most challenging task is to know what to use when and how to prepare the data appropriately.

This second assignment has two components. First, forecasting the sensor data on the ptbdb dataset and second, implementing multiple error calculation methods such as root mean squared error (RMSE) or mean absolute error (MAE) to evaluate the predictions. This is a first introduction to computer science in python.



Weather Forecast Graz
`https://google.com`, `https://weather.com`

1. File Headers

All python source files (apart from the __init__.py) 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. (Depending on your PDF-viewer you may have to fix the whitespaces)!

2. Tasks

This assignment build upon the framework developed in assignment 1. However, it does not necessarily rely on it.

You are given a lot of freedom regarding how to implement your program. We do not strictly dictate how you get your parameters. You can require the via command lines, import a dedicated settings file or have them directly in your code. However, you have to document your program clearly so anyone is able to adapt it to their needs (such as, loading a different data file).

2.1. Sensor Data Prediction (8pt)

We want to predict a future sensor value based the last few. You can use the Linear Regression module provided by sklearn[1] (Note: There are many different possible algorithms. Feel free to use any you like!). The data needs to be prepared for this. First you need to split all your data into a training and a test set. A good rule of thumb is to assign the first 80% of your data as training data. Then you want to prepare sets for the algorithm to train. These sets contain of a series of the last sensor on the one hand. And on

[1]`http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html`

the other you want to have the value Y iterations in the future. Do This for every set in your training data.

Example: You have 20 values. You assign the first 15 as training set and want to consider the last 5 values and want to look 5 into the future. Your training set would look as follows: (Note, use numpy arrays, not lists)

```
# d is the training part of your data
X = [
     [d0,d1,d2,d3,d4],
     [d1,d2,d3,d4,d5],
     [d2,d3,d4,d5,d6],
     [d3,d4,d5,d6,d7],
     [d4,d5,d6,d7,d8],
     [d5,d6,d7,d8,d9]
    ]

y = [d9, d10, d11, d12, d13, 15]
```

To train the algorithm you call the fit() function. Note: X and y are acceptable common variable names for these tasks. However, add a short comment explaining them when you first create them. This fitting process can take a long time the more past values you use. (Hint: start out with 100 values and predict 10 values in the future. Then increase those values and observe the results)

Once your data is trained call .predict() to obtain your predictions. The parameter for this function is similar to the X in fit but uses the test set (the remainder of your data). It will return a list of predicted values. Proceed to compare them with your test set using your own error calculations (see below).

2.2. Error Calculation (5pt)

Implement a set of classes each implementing a different method to calculate the error of a data series. For example, the root mean squared error (RMSE)[2]. Use the interface class below as parent for your classes and implement the required functionality. These sources are also available online. [3] Note: Implement different calculation methods to see their effect. Your choice of algorithm will not be graded.

[2] https://www.kaggle.com/wiki/RootMeanSquaredError

[3] https://raw.githubusercontent.com/pkasper/info2-bm/master/assignments/error_calculator.py

Data integrity checks (1pt)

Extend the interface class and implement checks for the data getter and setter. The 3 types allowed as data are List, Tuple, and numpy.array().

Implement Subclasses ($4 \times 1pt$)

Implement 4 classes each implementing a different error metric. Implement the calc_error() function by hand, and do *NOT* call functions provided by packages such as sklearn. (You can should however use them to compare your results). You can use the functionality provided by the interface class to to perform data integrity checks. Note: You may want to add more checks to the abstract class (for example: Check if the passed parameter is actually a variable of the correct type). Carefully document your classes so it is clear how you calculate the errors.

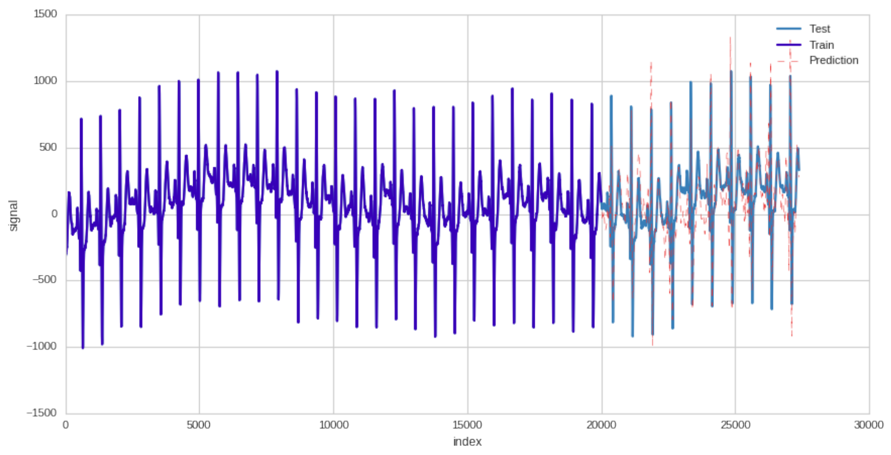
Abstact Interface Class ErrorCalculator()

```
import abc

class ErrorCalculator(metaclass=abc.ABCMeta):
    """A class built to calculate errors on data series.
    This is an abstract class and thus, concrete implementations of the
    error calculation methods have to be defined in the child classes.

    Args:
      data (list): The series of true/test data. Can be of np.array().

    Attributes:
      _data (str): The stored list of of true/test data. Only to be set
                   via the properties.
    """
    def __init__(self, data=list()):
        if self.__class__ is ErrorCalculator:
            raise NotImplementedError("Instantiated␣base␣class.␣Use␣a␣derivative!")
        self._data = data

    @property
    def data(self):
        """
        Property to access the true/test data. Can either be set in the class constructor
        or via this property.

        TODO: The setter method should perform type safety checks! Best create a separate
        (non-abstract) function for this.
        """
        return self._data

    @data.setter
    def data(self, data):
        self._data = data

    @abc.abstractmethod
    def calc_error(self, y):
        """
        This method calculates the error of series y on the stored test data.

        TODO: Implement typechecks for y

        Args:
          y: The series to test with

        Returns:
          The return value. The calculated error of series y on self_data
        """
        if self._data is None:
            raise ValueError("Data␣is␣not␣set")
        if len(y) != len(self._data[1]):
            raise ValueError("Series␣are␣not␣of␣identical␣length")
```

2.3. Visualisation (2pt)

Visualise your data and results in a meaningful way. Make sure all information is labelled. Your program should create two ".png" or ".pdf" files as output and name according to the information depicted.

First, visualize the data and your prediction results. Make sure the reader can distinct training from test data. Here is an example minimal visualisation:



Second, visualise the errors for each of your implemented error calculation methods. There are two values of interesting for each algorithm. The error on the training part of the data and the error on the test set. Choose an appropriate visualization style to display this kind of information.

3. Your Program Environment

You can assume both data and the info1bm package of assignment 1 will be present in your project directors. The ptdbd data files will be available in a `/data/` subfolder in the identical structure to the original data[4]. Feel free to use them to read and process the binary data.

4. File Headers

All python source files (apart from the __init__.py) 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. (Depending on your PDF-viewer you may have to fix the whitespaces)! **Example:**

```
######################################################################
# Author:      Patrick Kasper
# MatNr:       0730294
# Description: The main file. Here we import the package and do stuff
# Comments:    This is the example comment. I just made it a bit
#              longer so it spans across multiple lines.
######################################################################
```

5. External Code

If you use **any** code from any online source you must always make a note about this in the code. A short comment explaining where you took the code from and what it does is enough. Please note, this includes and code you take from the reference solutions of informatics 1. Else these lines of code will be considered plagiarism! Remember however, this is still a 1-student assignment so do not copy the code of your colleagues!

[4] `https://www.physionet.org/physiobank/database/ptbdb/`

6. Restrictions

- Do not add any bloat to your submission
- Use built-in functions where possible
- Try to avoid *.__MACOSX* or *.DS_STORE* folders in your submission archives.

6.1. Allowed packages

Any that came bundled with anaconda! See this list: `https://docs.continuum.io/anaconda/pkg-docs#python-3-6`. Any packages marked as *In Installer* can be used. Note, this link leads to the 3.6 specifications. This is intended as 3.5 is technically outdated. However, it is safe to assume everything that is listed in the 3.6 package is also available in 3.5.

7. Coding Standard

For this lecture and the practicals we use PEP 8 [5] 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.

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!

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.

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

8. Automated Tests

We will monitor the standard output. For this task please make sure you do not have any other output and you use proper capitalisation (as defined by the task).
Please make sure your submission follows all the restrictions defined in this document. Your program will have a limited runtime of 5 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.

9. Submission

9.1. Deadline

7th of May 2017 at 23:59:59.

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 on time please contact your tutor as soon as possible.

9.2. 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 file
- Coding standard upheld

9.3. Your submission file

```
assignment_2.zip (or assignment_2.tar.gz)
    assignment_2.py
    readme.txt
    <Error Calculators>
```