Lab 9, NumPy and Simple Linear Regression

Due date: Wednesday, May 11, 10:00am.

Overview

For this lab you will implement Simple Linear Regression and will test it on the Jester dataset.

Assignment Preparation

This is an individual assignment.

For this assignment you will build one Jupyter notebook, `SLR.ipynb`.

To download the instructor's version of the notebook, open a Jupyter terminal, change to your lab assignments directory and type the following command:

```
$ nbgrader fetch --course=dekhtyar Lab9
```

The notebook will be made available at the end of the day Friday, May 6. You will receive an email when it is made available.

Data

*This section is a copy of the **Data** section of Lab 7 document. It is kept here to give you easier access to reference material.*

You will be using *joke ratings data* from the **Jester** project, run by Professor Ken Goldberg at UC Berkeley. **Jester**[1] is an on-line joke recommender system available at

http://shadow.ieor.berkeley.edu/humor/

Disclaimer. **Please read this note before proceeding! Jester** has a database consisting of 100 jokes. The jokes are shown to the user, and the user's reaction to them is measured on a continuous scale. **Please be aware** that the jokes available through `Jester` may contain some examples that you personally will find objectionable. Please know, that *it is not my intent to offend anyone's sensibilities* (and neither is it the intent of the authors of the dataset and **Jester**).

The Dataset

The portion of the **Jester** dataset that you will be studying consists of two files uploaded to the `/home/dekhtyar/data` directory on the Jupyter server.

List of jokes. The first file is `Jokes.xml`. This file contains the list of jokes that **Jester** uses. The file has the following simple format:

```
<jokes>
  <joke>
    Joke 1 goes here...
  </joke>
   ...
  <joke>
     Joke 100 goes here...
  </joke>
</jokes>
```

There are 100 jokes in the file — these are all the jokes **Jester** uses. This file can be easily parsed with `BeautifulSoup` using the techniques you have aquired from the previous labs. The jokes are not numbered internally in the XML file, but their order corresponds to the joke ids in the ratings data file.

Ratings Data. The `jester-data-1.csv` file contains our joke ratings data. The full dataset contains three data files of roughly equal size. Of the three files, you will be using only the first one, `jester-data-1`.

The **Jester** dataset web page describes the format of the data as follows[2]:

> "Format:
>
> 1. 3 Data files contain anonymous ratings data from 73,421 users.
> 2. Data files are in .zip format, when unzipped, they are in Excel (.xls) format
> 3. Ratings are real values ranging from -10.00 to +10.00 (the value "99" corresponds to "null" = "not rated").
> 4. One row per user
> 5. The first column gives the number of jokes rated by that user. The next 100 columns give the ratings for jokes 01 - 100.

6. The sub-matrix including only columns $\{5, 7, 8, 13, 15, 16, 17, 18, 19, 20\}$ is dense. Almost all users have rated those jokes (see discussion of "universal queries" in the above paper)."

There are 24,983 rows (ratings records) in the `jester-data-1.csv` file. Each row has 101 values as described above.

Lab Assignment

Before starting your work, make sure you download the instructor's version of the `Ttest.ipynb` notebook.

The notebook asks you to perform the following set of actions:

1. **Import Joke Ratings.** Reuse your Lab 7 code to import the array of joke ratings in the format/formats convenient for future use (see the remainder of instructions to determine what specific format the data should be imported in).
2. **Develop simple linear regression routine.** You will implement a function that takes as input two one-dimensional arrays and performs a linear regression between them using the minimization of the sum squared error technique.
3. **Data Analysis.** Using the simple linear regression function you have implemented, you will study the Jester dataset and look for linear relationships in the data there. The full list of question is below.
4. **Explanations.** The final part of the notebook is your explanations and thought on what you have observed.

Step 1. Import of Ratings.

For the first part of the notebook, import and if necessary extend your Lab 7 code for loading the Joke ratings data into main memory. For this lab, we will not be using the texts of the jokes, so there is no need to import your inverted index construction.

Determine in what format you want to store the ratings, and what auxillary data structures you want to have based on the tasks you are asked to do in Step 3 of the process.

All ratings data must be stored in `NumPy arrays`.

Step 2. Simple Linear Regerssion.

Implement simple linear regression procedure by creating a function

```
slr(sample1, sample2)
```

where the input parameters, `sample1` and `sample2` are two one-dimensional NumPy arrays that have the same size (test for it in the code). The function shall return two values: `alpha` and `beta`: the coefficients of the regression line. See below.

Simple Linear regression in nutshell

Given two observations for the same collection of data points $x = (x_1, \dots, x_n)$ and $y = (y_1 \dots, y_n)$ we ask whether x and y have an observable (and predictable) relationship.

The simplest form of such a relationship is a *linear relationship* where we represent observations y via observations x as follows:

$$y = \beta \cdot x + \alpha$$

Usually, the relationship in the data are not exact (even if observed), so we assume that in addition to the linear relationship, an error may also be present:

$$y = \beta \cdot x + \alpha + \epsilon$$

When broken into individual observations, we obtain a series of relationships:

$$y_1 = \beta \cdot x_1 + \alpha + \epsilon_1$$
$$\dots$$
$$y_n = \beta \cdot x_n + \alpha + \epsilon_n$$

We are interested in finding the values α and β that minimize the error.

Least Squares method. The most straightforward method to construct simple linear regression is to use the least squares method for estimating α and β. In this method, we want to optimize the objective function

$$Q(\alpha, \beta) = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - (\beta x_i + \alpha))^2$$

This function is a quadratic fully differentiable function of two variables, which means that it has exactly one global optimum and it is reached in the point $(al\hat{p}ha, be\hat{t}a)$ where the partial derivatives of $Q(\alpha, \beta)$ are equal to 0.

Therefore, we need to solve the following system of equations:

$$\frac{\partial Q}{\partial \alpha} = \sum_{i=1}^{n} (2\alpha - 2y_i - 2\beta x_i) = 0$$

$$\frac{\partial Q}{\partial \beta} = \sum_{i=1}^{n} \left(2\beta x_i^2 - 2y_i x_i + 2\alpha x_i \right) = 0$$

From the first equation:

$$\sum_{i}^{n} \alpha = \sum_{i}^{n} (y_i - \beta x_i)$$

Since $\sum_i^\alpha = n \cdot \alpha$, we get:

$$\alpha = \frac{\sum_i^n (y_i - \beta x_i)}{n}$$

Subsituting into the second equation for α we get:

$$\sum_{i=1}^{n} \left(2\beta x_i^2 - 2y_i x_i + 2\alpha x_i \right) = \beta \sum_{i=1}^{n} x_i^2 - \sum_{i=1}^{n} y_i x_i + \alpha \sum_{i=1}^{n} x_i =$$

$$= \beta \sum_{i=1}^{n} x_i^2 - \sum_{i=1}^{n} y_i x_i + \frac{1}{n} \sum_{i=1}^{n} (y_i - \beta x_i) \sum_{i=1}^{n} x_i =$$

$$\beta \left(\sum_{i=1}^{n} x_i^2 - \frac{1}{n} \left(\sum_{i=1}^{n} x_i \right)^2 \right) - \left(\sum_{i=1}^{n} y_i x_i - \frac{1}{n} \sum_{i=1}^{n} y_i \sum_{i=1}^{n} x_i \right) = 0$$

This, in turn, means that:

$$\hat{\beta} = \frac{\sum_{i=1}^{n} y_i x_i - \frac{1}{n} \sum_{i=1}^{n} y_i \sum_{i=1}^{n} x_i}{\sum_{i=1}^{n} x_i^2 - \frac{1}{n} \left(\sum_{i=1}^{n} x_i \right)^2}$$

Substituting back into α:

$$\hat{\alpha} = \frac{1}{n} \left(\sum_{i=1}^{n} y_i - \hat{b} \sum_{i=1} x_i \right)$$

Step 3. Analysis of Relationships in Jester Data

You will perform three analyses using your linear regression function.

Preliminaries. Before analyzing the Jester data, implement one more function:

```
def commonUsers(ratings1, ratings2)
```

Recall that not every user rated every joke. When trying to find a linear relationship either between two jokes or between two users, it is useful to be able to extract from the ratings data only the pairs of observations where a single user rated both jokes (or a pair of users rated the same joke). The `commonUsers()` function shall do exactly that. It shall take as input two arrays of joke ratings (generally speaking, either rows or columns of the joke ratings array), and return back **two** arrays that contain all the common ratings in the two arrays. For example, if the two arrays are

```
a= np.array([1, 2, 99, -1, 3, 99])
b= np.array([99,-3, 3,-2, 2, 4])
```

the function call

```
r1, r2 = commonUsers(a,b)
```

shall return the following:

```
r1
np.array([2,-1,3])
r2
np.array([-3,-2,2])
```

Use this function to obtain the observation array that you will be passing to the `slr()` function.

MatPlotLib code. The notebook text contains some basic `Matplotlib` code for displaying scatterplots of observations and regression lines. One of the notebook cells contains a full example of building a scatterplot and a regression line for two jokes (jokes 14 and 50 in the current version of the notebook). Feel free to reuse this code in the other parts of the notebook where you need to visualize your results. In-line comments explain what each matplotlib function does.

Note: This notebook also uses `seaborn`, Python's package that creates a nice(r) visualization of the plots. To check the effects of `seaborn`, find the following code in the notebook cell that contains the import statements:

```
import seaborn
seaborn.set()
```

Comment out this code and rerun the notebook. You will see a distinct difference in how the plots are rendered (hint: they are prettier with `seaborn`).

Question 1. Find relationships between Joke 50 and all other jokes. This question is fairly straightforward. Recall that in Lab 7 you determined that Joke 50 in the Jester dataset has the highest average rating from the users. For this round of analysis, we want to see what the linear relationships between this joke and all other jokes in the dataset are.

Perform the following steps. For each other joke, compute the linear regression between that joke and Joke 50. Report the α, β regression coefficients and the *Sum Squared Error Per User* (the total sum squared error divided by the number of users who rated both jokes). (Yes - your output will contain 99 printed lines). The notebook has instructions for how to format the output.

Once you perform these computations (make sure you archive all the results), study the following, somewhat open-ended, question:

> ***Which joke has the best linear relationship with Joke 50?***

The notebook provides a Python code cell for any followup computations you need to perform to determine the answer to this question, and a Markdown cell for you to put your answer explaining which joke it is, and why/how you selected it.

Question 2. Study the necessary sample size to produce reliable approximations. For this question, we choose to compare Joke 49 and Joke 50.

The total number of people who rated both jokes is fairly high. We want to try to build a reasonably exact regression line based on only a sample from the entire population of people who rated both jokes. We want to know what sample size we need to draw to produce **reliable results**. We define **reliable results** as follows:

> for an ε value of 0.1, and given a sample $u = (u_1, \ldots, u_n)$ of users randomly selected from a set of all users who rated jokes 49 and 50, both α_u and β_u parameters of the linear regression line for the sample are within the ε value of the α and β parameters for the population **about 95% of the time.**

Your code shall test a variety of sample sizes, and determine what percentage of regression lines computed based on random samples of the given size is *reliable*. Once you reach the 95% reliability threshold, you can stop. The notebook provides instructions on how you shall report the observed results for every sample size. **Note:** you code must conduct an actual search for the appropriate sample size.

Question 3. Visual Examination of Linear Relationships. In addition to comparing ratings for two different jokes, we can compare ratings two different users gave to all the jokes.

For this question you are asked to select a few pairs of users, and visualize both the scatterplots of the joke ratings (user1 vs. user2 ratings) and the regression lines obtained.

You can start the examination with couple of arbitrarily selected users pair to see what is going on, but I would like to ask you to find some pairs of users with *interesting* relationships (either strong positive correlations or strong negative correlations).

I included a number of empty notebook cells for this questions - use them and create new cells as needed (only one notebook cell in that block is an autograded cell - you must fill it with code for one of the computations).

Step 4. Reflections

After completing your studies, include a Markdown cell that contains your observations, thoughts and reflections of the observed results.

Deliverables and submission instructions

Submit your lab using `nbgrader`:

```
$ nbgrader submit --course=dekhtyar Lab9
```

References

[1] Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Eigentaste: A Constant Time Collaborative Filtering Algorithm. *Information Retrieval*, 4(2), 133-151. July 2001.

[2] Ken Goldberg, Anonymous Ratings Data from the Jester Online Joke Recommender System, http://www.ieor.berkeley.edu/~goldberg/jester-data/.