Foundations of Data Science (CS F320)
Assignment 1

Submission by 23:59 on 06-Oct-2018

The Problem

Two BPHC students, Rabbit and Tortoise, are arguing about different approaches to use in their Data Science Assignment. Rabbit wants to use all the data for learning at once. Tortoise wants to use data sequentially. Your task is to perform a Bayesian analysis of this problem.

Dataset

The question is based on the Coin Tossing problem. You will have to randomly generate a dataset for this problem. The dataset should be of size around 150.

The Model

As we know,

$$Posterior \propto Likelihood \text{ x } Prior$$

$$p(\mu|D, a, b) \propto P(D|\mu) \text{ x } P(\mu|a, b)$$

where,
Prior is the information known by us before the coin tosses,
Likelihood is the distribuiton of the data points in the coin tosses and
Posterior is the distribution of the mean after observint the coin tosses.
Our goal is to find the distribution followed by the mean of the coin tosses after observing data points.

As we know, coin tossing follows a Bernoulli distribution. It's probability density function is given by

$$\text{Bern}(x|\mu) = \mu^x(1-\mu)^{1-x}$$

where μ is the mean of the bernoulli distributiuon.

Thus, for a dataset D of N points, we get the likelihood function as

$$p(D|\mu) = \prod_{n=1}^{N} p(x_n|\mu) = \prod_{n=1}^{N} \mu^{x_n}(1-\mu)^{1-x_n}$$

We will take the prior to be a beta distribution. The PDF for a beta distribution is given by

$$\text{Beta}(\mu|a,b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\mu^{a-1}(1-\mu)^{b-1}$$

where a and b are the parameters and Γ is the gamma function. Choose appropirate a and b such that the mean of the prior is 0.4.

Multiplying the 2 distributions, you get the posterior (Hint: This will also be a Beta distribution). Your goal is to computer this distribution.

Part A: Sequential Learning

This is Tortoise's view of the problem. Since you will take examples one at a time, the likelihood function will have only one term in it's product.

As with any sequential learning problem, this posterior after vieweing the first example becomes the prior for the next example. Thus, you start off with a prior and will observe how examples coming in change the appearance of the prior. You will have to make plots of the prior at each stage (similar to Fig 2.2 in Bishop). Optionally, combine all the plots together to make a GIF.

Part B

This is Rabit's view of the problem. In this case, the likelihood will be over the entire dataset. Compute the posterior after viewing the entire dataset at once and make plots of the same.

Part C

Remark on the differences(or similarities) between the two models that you have obtained. Also, come up with a reason to why the dataset was restricted to 150 datapoints. What happens if more points are added?

Implementation Details

You will have to implement this in *python* using *numpy*, *scipy* and *matplotlib*. No other libraries are allowed.

Hints

1. *numpy.random* has a large number of functions to generate random numbers. There are functions to generate integers within a range (*numpy.random.randint*) or real numbers within $[0, 1]$ (*numpy.random.random*). You will need to use these.

2. A small code snippet is given below on how to generate an array of 5 numbers within the range $[0, 1]$ and to exponentiate them to the fourth power. You will need to do something like this when you want to compute the Beta function PDF.

```
>>> import numpy as np
>>> np.random.random([5,1])
>>> n=np.random.random([5,1])
>>> n
array([[0.95486375],
       [0.56752836],
       [0.47852305],
       [0.03370669],
       [0.01789102]])
>>> n**4
array([[8.31315015e-01],
       [1.03740964e-01],
       [5.24338141e-02],
       [1.29081664e-06],
       [1.02456767e-07]])
```

3. You will need to compute the value of the gamma function along the way. For that, you can use *scipy.special.gamma* from the *scipy* module

```
>>> import scipy.special
>>> scipy.special.gamma(5)
24.0
>>> scipy.special.gamma(0.1)
9.513507698668732
>>> scipy.special.gamma(0.01)
99.43258511915059
```

4. Given below is code in *matplotlib* to plot the probability density function. x are points in the range $[0, 1]$ and y is the computed pdf value.

```
import matplotlib.pyplot as plt
import matplotlib.ticker as mt


#i represents the figure number for saving the figure as file
def plot(x,y,i):
  plt.figure(1)
  ax = plt.gca()

  mx = 10
  ax.set_ylim(0, mx)
  ax.xaxis.set_major_locator(
                mt.FixedLocator([i*0.1 for i in range(11)]))
  ax.plot(x, y,linewidth=0,marker='.',markersize=4)

  #To show the figure, use this
  #plt.show()

  #To save the figure to a file, use this
  #plt.savefig("Fig/{}.png".format(i))

  plt.close(1)
```

Submission

Submission will be via CMS. Gather all your code into a single .py file. Along with that, make a document with the plots. You can put plots after every 20 examples for Part A, and the final figure for Part B. Also, write down your answer for Part C. Zip the code and this document and name it with your ID numbers.

For Queries

Anirudh Srinivasan (2015A7PS0382H)
Phani Shankar Ede (2015B3A70420H)

References

Section 2.1, Christopher M Bishop. 2006. Pattern Reccognition and Machine Learning.