Exercise session 1 / 14

February 12, 2019

The objective of this first exercise session are the following:

- Get familiar with the specialized libraries of Python that are necessary to complete these exercise sheets.
- Apply Bayesian statistics to a first simple example: the Bernoulli model.

1 Useful libraries for Python

1.1 Introduction to Python

Quoting Wikipedia:

> Python is an interpreted high-level programming language for general-purpose programming. [...] Python has a design philosophy that emphasizes code readability, and a syntax that allows programmers to express concepts in fewer lines of code [...]. It provides constructs that enable clear programming on both small and large scales.

It's two main drawbacks are:

1. It is so flexible that it does tend to breed bad habits compared to more rigorous languages.
2. It is not as efficient as C in terms of speed.
   However, it is possible to have Python code interact with C code in order to get a best of both worlds.

Python has gained widespread recognition, especially in the Machine Learning community, and a lot of code is freely available online for a lot of learning algorithms.

If you wish to code the exercises of the course in Python (which is recommended), you will need to use the following two libraries:

1. Numpy: this is a critical library which endows Python with arrays. Since scientific programing without arrays is senseless, numpy is indispensable.

2. Matplotlib: this is my prefered library for plotting.

Both can be installed by installing the Anaconda Python distribution (hyperlink).

The following libraries could also prove useful:

1. Pandas: handle datasets using an R-style *dataframe* class. Many useful bits and pieces for data analysis.

2. Pytorch and Tensorflow: optimized linear algebra, derivatives using the backpropagation algorithm.

3. Pickle: save and load data to the disk.

4. Seaborn: fancier plotting than matplotlib.

5. Scipy: some fancier math than numpy.

1.2 Programming assignments

Before you proceed further, you should be familiar with Numpy and Matplotlib. If you aren't:

1. Find a tutorial for both libraries.

2. Find out how to define arrays of various sizes with:

   ```
   numpy.array()
   numpy.zeros()
   numpy.ones()
   numpy.eye()
   ```

3. Find out how to do basic plots with Matplotlib with:

   ```
   matplotlib.pyplot.plot()
   matplotlib.pyplot.show()
   matplotlib.pyplot.hist()
   matplotlib.pyplot.scatter()
   ```

4. Find how to perform the following operations on arrays:

   (a) Array addition, scalar product, elementwise multiplication.

   (b) Matrix multiplication (use the @ operator; hyperlink).

5. Find out how to generate draws from various classical distribution using the numpy.random sublibrary.

2 Bernoulli model

Let $x_1 \dots x_n$ be data such that each datapoint can only take one out of two values (e.g. Success/Failure). e.g:

```
x_array = np.array( [0,1,1,0,1,0,1,1,0,1,1], dtype = np.int )
```

We will model such data using a Bernouli model:

$$X_i \overset{IID}{\sim} \mathcal{B}(\theta) \;\; \theta \in [0,1]$$

where θ is the unknown success probability.

In order for us to have a Bayesian model of this data, we further need a **prior distribution** on θ. We will model θ to be a-priori distributed from a **beta distribution:**

$$\theta \sim Beta(\alpha, \beta)$$
$$f(\theta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \theta^{\alpha-1} (1-\theta)^{\beta-1}$$

where $\Gamma(t)$ is the Gamma function, a classical function of analysis.

2.1 Mathematical assignments

First, we will derive some of the properties of this model. In particular, we will describe how

1. Open the Wikipedia page on the Beta distribution.
   Find the prior mean and variance of θ.

2. The uniform distribution is a particular case of the Beta distribution.
   Which value of α and β does it correspond to?

3. Assume we are in a situation were our prior knowledge is that θ is roughly equal to 0.3 ± 0.1.
   Which value of α and β does that correspond to?

4. Prove that the unnormalized posterior distribution for θ:

   $$\tilde{f}(\theta | x_1 \dots x_n) = f(\theta) f(x_1 \dots x_n | \theta)$$

   is an unnormalized Beta distribution.
   What are the parameters of this distribution?

5. Compute the normalization constant:

   $$f(x_1 \dots x_n) = \int f(\theta) f(x_1 \dots x_n | \theta)$$

6. The Beta distribution is approximately Gaussian in any limit where α, β both go to ∞ while their ratio remains constant[1].
   For a fixed pair (α, β), figure out the mean and variance of a Gaussian approximation.
   Use this fact to give the formula for an approximate **0.95 credible interval**, i.e. an interval such that:

$$\mathbb{P}\left(\theta \in I | x_1 \dots x_n\right) \approx 0.95$$

   where the approximation is the Gaussian approximation.

7. Try to comment on the size of the credible interval as n tends to infinity.

8. Check that either $\alpha = 0$ and $\beta = 0$ both give *improper priors*, i.e priors which do not have an integral equal to 1.

2.2 Programming assignments

Now that we have derived the properties of the model, we will implement:

- A procedure to generate a new dataset.
- A procedure to give a Bayesian analysis of a given dataset.

In practice:

1. Implement a function to generate a dataset.
   Your function should take as input:
   (a) A true probability θ_0.
   (b) The number of points n.

2. Implement a function to compute the posterior distribution given an inputed dataset.
   Your function should take as input:

[1]Informal proof: if γ_1, γ_2 are two independent Gamma variables:

$$\gamma_1 \sim \Gamma\left(\alpha, 1\right)$$
$$\gamma_2 \sim \Gamma\left(\beta, 1\right)$$

then the ratio:

$$\frac{\gamma_1}{\gamma_1 + \gamma_2} \sim \mathcal{B}\left(\alpha, \beta\right)$$

The claimed result then can be obtained:

- A Gamma variable is a sum of $n = \alpha$ IID exponential random variables and thus converges to a Gaussian (CLT).
- Apply the law of large numbers to $\gamma_1 + \gamma_2$ and the CLT to γ_1 and combining the two halves with Slutsky's theorem.

More direct proofs are also possible.

(a) A dataset $x_1 \dots x_n$.

(b) The prior parameters α, β (by default, use the uniform prior $\alpha = \beta = 1$).

Your fonction should return:

(a) A function which takes as input a value of θ and return the unnormalized posterior $\tilde{f}(\theta|x_1 \dots x_n)$.

(b) The value of the normalization constant.

(c) The bounds of the approximate 0.95 credible interval.

3. Implement a function which takes as input the unnormalized posterior $\tilde{f}(\theta|x_1 \dots x_n)$ and plots it.

4. Implement a function which plots the normalized posterior instead.

5. Implement a function which plots the 0.95 credible interval.

2.3 Experimental assignments

We now get to play around with our functions.

1. Generate three datasets with the same n and same θ_0.
   Plot the three posteriors to observe how they vary with the data.

2. Generate multiple datasets with the same θ_0 but growing n (e.g. $n = \{2, 10, 100\}$).
   How does the shape of the posterior change with n? What do you think it converges to?

3. For fixed θ_0, plot the size of the credible interval as n increases.
   How does the size of the credible interval scale with n?

4. For fixed θ_0, plot the log of the normalization constant as n increases.
   How are these two quantities related?
   (NB: you might need to use the scipy.special.loggamma function to ensure numerical stability).

5. Test empirically how the posterior-mean compares to the natural frequentist estimator:
   $$\hat{\theta} = \frac{\sum_{i=1}^{n} x_i}{n}$$
   for various values of α, β, θ_0 and n.

6. The following claim is true:

   > The 0.95 credible interval is approximately a 0.95 frequentist confidence interval.

   Check experimentaly that the claim is true.