JSC270 Assignment 3

Simulation (with `numpy`)

February 13, 2021

Submission Deadline: Monday, March 8th at 11:59AM EST

What and where to submit:

- A PDF report containing your responses, including any figures or derivations.
- A Google Colab notebook containing any relevant python code. Please provide a link to this notebook in your PDF report. Optionally, you may place the code on GitHub, and instead provide a link to that repo.

The PDF report is to be submitted through Quercus. This assignment is to be completed individually.

Grading Scheme: This assignment contains 3 questions, for a total of 50 marks, plus 4 optional bonus points.

1. Bayesian Simulation (18 pts)

Recall that the Bayesian approach to modelling assumes we have some prior belief about an unknown parameter of interest (say, θ). Our uncertainty about the true value is captured in a distribution called a **prior**. Instead of treating data as random, and assuming a fixed parameter to explain the sample we see, we assume the parameter θ is random, and we use a given sample of data (X) as evidence to update our belief about θ's true distribution. This process of updating beliefs is done via Bayes' Rule:

$$f(\theta|x) = \frac{f(x|\theta)f(\theta)}{f(x)} = \frac{f(x|\theta)f(\theta)}{\int_\theta f(x|\theta')f(\theta')d\theta'}$$

Each of the terms in the rule has its own name:

$$posterior \ = \ \frac{likelihood \cdot prior}{normalizing \ constant}$$

Where the **prior** encodes previous beliefs about θ, and the **likelihood** captures how well the data fits our current beliefs. The **posterior** distribution then represents our new (updated) belief about θ, given the data we have observed. The **normalizing constant** is often ignored, but is needed to ensure that the posterior integrates to 1. Although there are several ways of estimating each component, for this question, we'll focus on how the choice of prior affects the posterior.

Suppose we have a sample of n individuals, and we're interested in measuring the number of persons who contract COVID-19. For simplicity, let's assume that each person will either contract the disease or not (i.e. the outcome is binary). Thus, $X_i = 1$ if individual i contracted the disease, or 0 if not. If we assume that

individuals contract the disease independent of one another, the **likelihood** function of our n individuals is given by:

$$f(x_1, \cdots, x_n|\theta) = f(x|\theta) = (\theta)^{\Sigma_{i=1}^n x_i}(1-\theta)^{n-\Sigma_{i=1}^n x_i}$$

Note that this is just the joint density of our n data points, assuming each point follows a Bernoulli(θ) distribution. How do we reasonably estimate θ? Let's assume that $\theta \sim Beta(a, b)$. In other words, our prior distribution is given by:

$$f(\theta) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\theta^{a-1}(1-\theta)^{b-1}$$

for $\theta \in (0, 1)$. Note that $\Gamma(c) = (c-1)!$, and that, in general, the Beta distribution has expectation equal to $E(\theta) = \frac{a}{a+b}$.

A) (3 pts) Using Bayes rule, show that the posterior distribution is $Beta(a+\Sigma_{i=1}^n x_i, b+n-\Sigma_{i=1}^n x_i)$. *Hint: You don't actually need to integrate to obtain the normalizing constant. Just use the definition of the beta distribution, and the fact that the posterior must integrate to one (i.e.* $\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)}\int \theta^{\alpha-1}(1-\theta)^{\beta-1}d\theta = 1$*).*

B) (1 pt) What is the expectation of the posterior distribution (i.e. the posterior mean)? *Note: No need for a full derivation here, you can just state (and possibly simplify) your answer.*

C) (2 pts) Suppose $a = b = 1$. This gives a special case of the beta distribution. What is the name of this special case? In using this special prior distribution, what are we saying about our beliefs across possible values of θ?

D) (4 pts) Using the values of a and b above, simulate 1000 draws each from the prior and posterior distributions you derived earlier. Assume a dataset of 20 individuals, 4 of whom have contracted the disease (i.e. $n = 20$, $\Sigma_{i=1}^n x_i = 4$). Plot the prior and posterior distributions (histograms are fine). You may find the `numpy` function `np.random.beta()` useful here. How does the posterior mean compare to 1) the prior mean, and 2) the traditional likelihood estimate, $\frac{1}{n}\Sigma_{i=1}^n x_i$? Why do you think this is?

E) (4 pts) Suppose scientists are fairly certain that the contraction rate of COVID is close to that of the flu, which is fairly low. To encode this belief, we change the hyperparameters of our prior to $a = 1, b = 3$. Simulate 1000 draws each from the prior and posterior distributions using the new hyperparameters. Assume a dataset identical to the one in part D (ie. $n = 20$, $\Sigma_{i=1}^n x_i = 4$). Plot the new prior and posterior distributions. What are the new prior and posterior means, and how do they compare to those you found in part D? Explain the differences.

F) (4 pts) Now suppose we have the same prior as in E ($a = 1, b = 3$), but we get a different dataset: $n = 20$, $\Sigma_{i=1}^n x_i = 12$. Simulate 1000 draws each from the prior and posterior distributions using the new dataset. Plot the new prior and posterior distributions (again, histograms are fine). Compare the prior and posterior means to the likelihood estimate defined earlier. How do these results compare to those you obtained in parts D and E? Explain these differences.

2. Asymptotic Behavior (15 pts)

Suppose we have a random variable x that follows an **Exponential Distribution**. Thus, X has the following density:

$$f(x) = \lambda e^{-\lambda x}$$

for some parameter λ, and is defined over $x \in [0, \infty)$. Recall also the definition for the expectation (mean) of a random variable: $E(X) = \int_{-\infty}^{\infty} x \cdot f(x)dx$.

A) (4 pts) Determine the expectation of X, using the definition given (*Hint: Your answer should be a function of λ only*). What is the exact value of the expection of X when $\lambda = 4$?

B) (6 pts) Suppose now that we know the exact distribution of X is given by $\lambda = 2$. In python, generate 6 different random samples from this distribution with sizes $\{10, 20, 50, 100, 500, 2000\}$. You may find the function `np.random.exponential()` from the numpy package useful here (note that the parameter argument for this function is inverted, ie $\lambda = \frac{1}{\beta}$). Plot the distribution of each sample (i.e. 6 plots total, or optionally 1 plot with 6 curves).

C) (5 pts) Using your samples from part B, plot the mean of the sample as a function of sample size, with mean on the y-axis, and sample size on the x-axis(`numpy` has a handy `np.mean()` function for this). What is the theoretical mean when $\lambda = 2$? Include this as a horizontal line on your plot. What happens to the sample mean as sample size grows? What is the name of the statistical 'law' that explains the sample average behavior you've just seen?

3. Bootstrapping and Classification (17 pts)

In this question, you'll apply the statistical process of **bootstrapping** from scratch, using simulated model data. In class, we saw bootstrapping applied to decision trees (ie bagging), but its uses expand well beyond classification. In general, bootstrapping is quite valuable in cases where repeated sampling from a population is prohibitively expensive, or noisy.

Let's start by simulating data that follows a simple (one-covariate) logistic regression model. The true predictor (X) and response(Y) follow the following relationship:

$$\pi_i = \sigma(\beta_0 + \beta_1 x_{1i} + \beta_2 x_{2i} + \beta_3 d_i + \epsilon_i)$$

where: $i = 1, ..., N$ is our sample, $\pi_i = Pr(y_i = 1)$, $\epsilon_i \sim_{iid} N(0, \tau^2)$, and the *logistic* or *sigmoid function* is given by $\sigma(t) = \frac{exp(t)}{1+exp(t)}$. Also, x_1, x_2 are continuous covariates, while $d_i \in \{0, 1, 2, 3, 4\}$ is categorical. The class label (ie. 0 or 1) is generated by choosing all values of π_i above a certain threshold: $y_i = I\{\pi_i > c\}$.

A) (3 pts) Using values $\tau = 2, \beta_0 = 5$, $\beta_1 = 0.6$, $\beta_2 = 0.8$, $\beta_3 = 0.4$, and $c = 0.5$, simulate $N = 1000$ observations of (X_i, y_i), where $X_i = (x_{1i}, x_{2i}, d_i)$. For simplicity, assume $x_{1i} \sim Uniform(-100, 100)$, and $x_{2i} \sim Uniform(-150, 150)$. Please set a seed before you generate random numbers, to ensure your results are reproducible (`np.random.seed()`). Here is a suggested order of the steps you should take:

1. Set your seed.
2. Generate n observations of x_1, x_2, and d_1.
3. Generate n observations of ϵ.
4. Use x_i, β_0, β_1, β_2, β_3, and ϵ_i to generate the regression line.
5. Apply the sigmoid function to obtain probabilities.
6. Use the threshold c to generate labels (y_i).

B) (3 pts) Split your simulated data into separate training and test sets. Use an 80/20 split. Fit a logistic regression model to the training data, and plot the ROC curve associated with prediction of the test set. Again, please be sure to set your seed, for reproducibility. *Note: If you're planning on using* `sklearn.linear_model.LogisticRegression()`, *make sure to set the 'penalty' argument to 'None', to avoid regularization.*

C) (6 pts) Now you'll implement the bootstrapping algorithm described earlier to fit a large number of decision trees. The strategy here, using the entire simulated sample (not split into train and test) is:

i) Generate B resamples (x_j, y_j) from the main sample (sampling with replacement).
ii) Fit a Decision Tree Classifier to each resample.
iii) Use the fitted model to obtain a performance measure on the out-of-bag (OOB) observations. You may choose whichever performance measure you wish (classification error, AUC, precison, recall, F1 score, etc.).

For this question, you won't need to store the resamples themselves, only their evaluation metrics. To help you get started, we've provided some pseudocode below. Use a resample size of $M = 1000$, and use $B = 5000$ resamples. **Your result should be a single vector** (or list/array/etc. if you used a different data structure) **of length B**, containing, for example, F1 scores. You may find the function `np.random.choice` helpful in sampling with replacement. For fitting the model, you might use `sklearn.tree.DecisionTreeClassifier`.

Algorithm 1: Bootstrapping with Decision Tree Classification

```
Instantiate array of size B to store resample metrics
for j = 1 : B do
    generate resample j of size M from main sample(sampling uniformly with replacement)
    fit Decision Tree Classifier to resample
    Compute resample evaluation metric on remaining out of bag observations
    array[j] ← computed resample metric
end for
```

D) (3 pts) Provide a plot showing the distribution of the bootstrapped out-of-sample performance metrics. In your bootstrap algorithm (and thus in your plot), you may choose whichever metric you wish (classification error, AUC, precison, recall, F1 score, etc.). Your plot should be a histogram (or kernel density) composed of B different bootstrapped measures of fit.

E) (2 pts) Compute the mean of the distribution you found in part D. Compare it to the performance of your logistic classifier in part A (you may need to find the matching metric from your logistic classifier to make a proper comparison). Which method performs better? Why do you think that is?

4. Bonus: Choosing Reasonable Priors

(4 Bonus pts) Figure 1 shows 4 common types of statistical distributions. For each of the four quantities given below, state which of the four distributions is the best fit[1]. For each choice, explain your answer (possibly including why other distributions do not work well).

i) The average price of a cup of coffee in Toronto
ii) The proportion of individuals who catch a cold in February.
iii) The number of people (among a small group) who own a dog.
iv) The average weekly change in US unemployment (measured in hundreds of individuals).

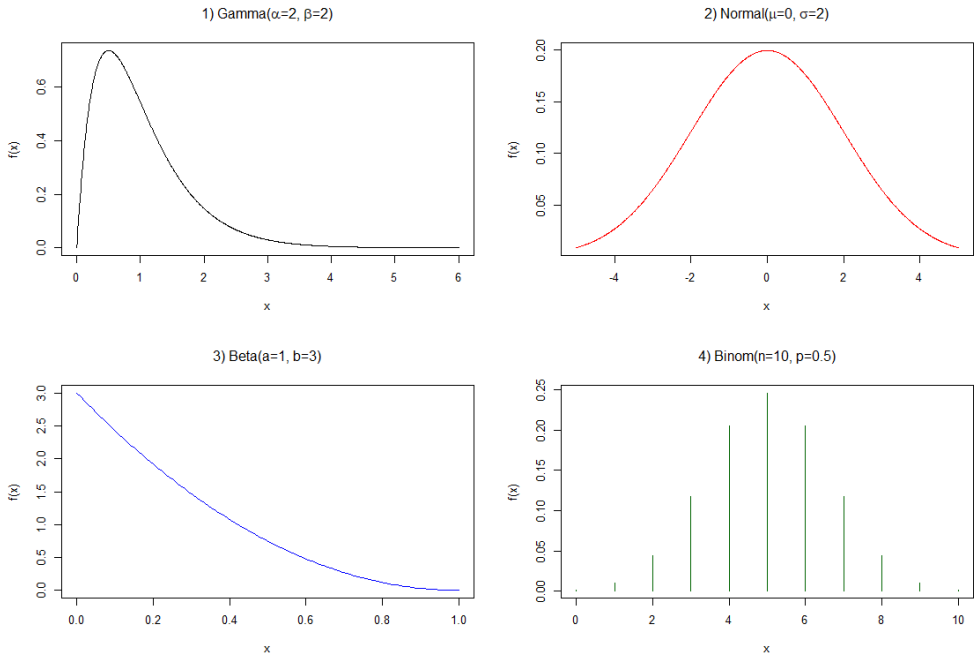

Figure 1: Some possible choices of prior.

[1]Note: I do not claim that these are the exact distributions of the given quantities. This exercise is just to show that what constitutes a reasonable choice of prior depends on the context.