Homework 3: Applied Probability and Statistics

UA CSC 380: Principles of Data Science, Fall 2022

Homework due at 11:59pm on February 17

Deliverables Your must make two submissions: (1) your homework as a SINGLE PDF file by the stated deadline to the gradescope. Include your code and output of the code as texts in the PDF. and (2) your codes as a ZIP file to a separate submission. Each subproblem is worth 10 points. More instructions:

- You can hand-write your answers and scan them to make it a PDF. If you use your phone camera, I recommend using TurboScan (smartphone app) or similar ones to avoid uploading a slanted image or showing the background. Make sure you rotate it correctly.
- Watch the video and follow the instruction for the submission: https://youtu.be/KMPoby5g_nE
- **Show all work along with answers to get the full credit**.
- **Paste all your codes and outputs in the report to get full credit.**
- Place your final answer into an 'answer box' that can be easily identified.
- There will be no late days. Late homeworks result in zero credit.

Failure to follow the submission instruction will result in a minor penalty in credit.

You can choose to work individually or in pairs.

- If you choose to work in pairs, you are free to discuss whatever you want with your partner; please make only one submission per group.
- Please do not discuss with people outside your group about the homework (refer to the academic integrity policy in Lecture 1).
- If you have clarification questions, please feel free to post on Piazza so that it can promote discussion.

Problem 1: Programming Setup

In this problem, let's set up the python environment using Conda. You do not need to submit any code for this problem.

Install Python 3 with Conda. Create your own virtual environment (see https://numpy.org/install/ for an example), and in your virtual environment, install numpy, scipy, matplotlib. Install Jupyter lab. Start the jupyter lab and start a jupyter notebook. Import matplotlib.pyplot and plot anything in there (e.g., pyplot.plot([1,2,3],[4,5,6])).

Please screen capture the jupyter notebook screen (the code and the shown plot) and report it in your submission. Once you are done, you may want to deactivate your own virtual environment. If you have already been using Jupyter Lab, no need to reinstall; paste the screen shots in your report.

Problem 2: Law of Large Numbers / Central Limit Theorem

This problem is about verifying theoretical results learned in the class by visualizing them. You will have to be familiar with googling and reading python library documents to solve these problems. You may also take a look at basic numpy operators like how addition works for vectors and matrices, and what 'mean(A,0)' does to a 2d numpy array A. I believe you must already be good at googling and reading API documents – one of the basic skills a computer science graduate must have. All the coding answers must be done with Jupyter lab.

a) Let us numerically verify the law of large numbers. We will simulate $m = 100$ sample mean trajectories of $X_1, \ldots, X_N \sim$ Bernoulli($\mu = 0.2$) and plot them altogether in one plot. Here, a sample mean trajectory means a sequence of $\bar{X}_1, \bar{X}_2, \ldots, \bar{X}_N$ where $\bar{X}_i$ is the sample mean using samples $X_1, \ldots, X_i$. We will plot $\bar{X}_n$ as a function of n, but do this multiple times. Take n from 1 to $N = 1000$. An ideal plot would look like the following:

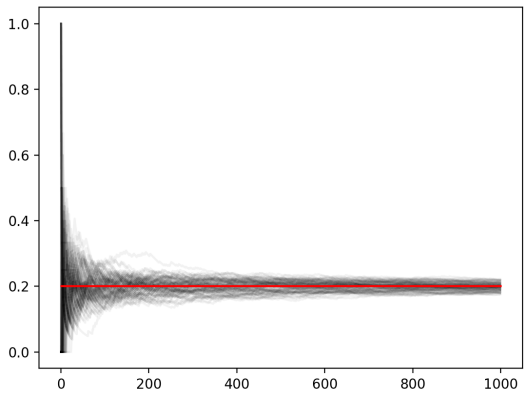

Figure 1:

You must use the 'alpha' option to pyplot.plot() to give some transparency (you should obtain a similar look visualization as above). You may want to use the 'color' option to specify the color.

b) Let us verify the central limit theorem (CLT) by simulation. For $N \in \{10, 100, 1000, 10000\}$, perform:

- Take N samples from Bernoulli($\mu = 0.05$) and compute the sample mean. Repeat this 1000 times.
- Plot those 1000 numbers as a histogram (pyplot.hist) with a proper number of bins. Use density=True.
- With a red line, overlay the pdf of a Gaussian distribution with the parameters suggested by the CLT (figure this out!).

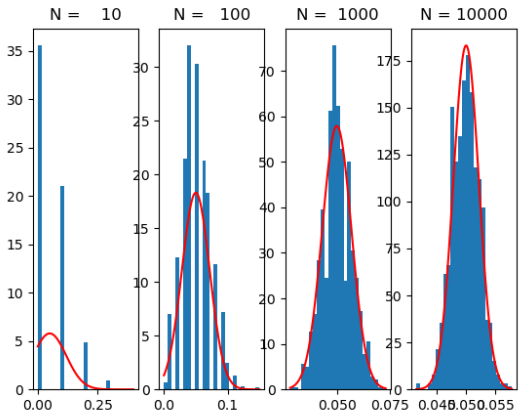

Figure 2:

An ideal answer would look like Figure 2. To receive full credit, you must use pyplot.subplot to have four plots in one figure.

c) Consider a function like $f(n) = a/\sqrt{n}$. I claim that $\log(f(n))$ will be a linear function of $\log(n)$ (see wikipedia for the meaning of linear function). First, plot this using jupyter to confirm this. Next, write a mathematical proof for the claim. Finally, report the slope of this linear function (a specific numerical value).

d) The CLT is inherently about the distribution of the sample mean. However, as seen from the lecture, Gaussian random variables almost have no tails; i.e., 3 times of standard deviation contains 99% of the probable values. Now, CLT says that $\hat{\mu}_n - \mu$ has standard deviation of $\sigma/\sqrt{n}$. This means $|\hat{\mu}_n - \mu|$ will be no larger than $3 \cdot \sigma/\sqrt{n}$. It is very tempting to think that $|\hat{\mu}_n - \mu|$ essentially behaves like $\sigma/\sqrt{n}$ which would imply that the rate of convergence is $O(1/\sqrt{n})$. Let's verify this numerically.

To verify our conjecture, let us numerically estimate how $|\hat{\mu}_n - \mu|$ behaves as a function of n, in expectation. Follow the setup of a) but instead of plotting trajectories, save the trajectories into a matrix variable A (2d numpy array) where A[i,n] would be i-th trajectory's $\hat{\mu}_n$. Also, let us use $m = 2000$. For each $n \in \{1, \ldots, N\}$ (recall $N = 1000$),

- compute $|\hat{\mu}_n - \mu|$ for each trajectory ($m \in \{1, \ldots, 2000\}$)
- take the average over these numbers

This gives us an estimate on $\mathbf{E}[|\hat{\mu}_n - \mu|]$. Finally, use the trick of the previous subproblem to plot a curve that verifies our conjecture. Please also draw a dashed line of the form $y = -(1/2)x + b$ for a suitably chosen b so we can provide a visual reference. An ideal answer would look like Figure 3. To receive full credit, be sure to label each axis.

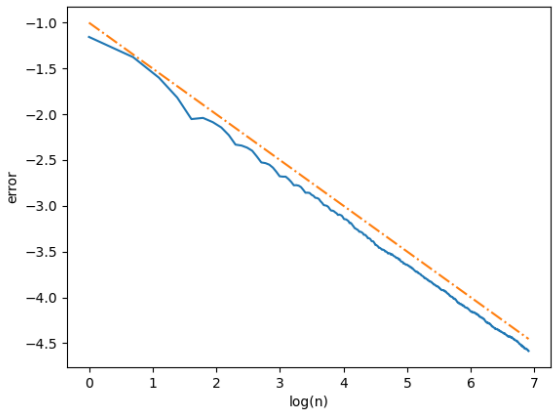

Figure 3:

Problem 3: Maximum Likelihood Estimation

I would like to build a simple model to predict how many students are likely to come to my office hours this semester. Because this is an arrival process, I will model the number of arrivals during office hours as Poisson distributed. Recall that the Poisson is a discrete distribution over the number of arrivals (or events) in a fixed time-frame. The Poisson distribution has a probability mass function (PMF) of the form,

$$\text{Poisson}(x; \lambda) = \frac{1}{x!}\lambda^x e^{-\lambda}.$$

The parameter λ is the *rate* parameter, and represents the expected number of arrivals $\mathbb{E}[x] = \lambda$. To fit the model I will need to estimate the rate parameter using some data.

a) During my last three office hours I received $X_1 = 5, X_2 = 6, X_3 = 8$ students. Write the logarithm of the joint probability distribution $\log p(X_1, X_2, X_3; \lambda)$.

b) Compute the maximum likelihood estimate (MLE) of the rate parameter λ^{MLE} which maximizes the joint probability in part (a). The log-likelihood function is concave and so the MLE can be computed by finding the zero-derivative solution. Make sure to show all of your calculations. How many arrivals should I expect at my next office hours under this model?

c) I have assigned a particularly challenging homework which has led to a lot of students $X_4 = 15$ arriving at my office hours. Compute the MLE again, but include this new training point. How has the model changed with this new data point?

Problem 4: Estimating Pearson Correlation

This question will walk you through the process of estimating the Pearson correlation of two random variables, along with confidence intervals using the bootstrap method. Lecture slides and Chapter 8 of the textbook (Wasserman) will be helpful if you need a refresher. For our chosen model, we will use a bivariate Gaussian distribution: $P(X, Y; \mu, \Sigma) = \mathcal{N}(\mu, \Sigma)$. For simplicity, let us assume that the distribution is zero-mean $\mu = (0, 0)^T$. Let us assume that the true (unknown) covariance matrix is,

$$\Sigma = \begin{pmatrix} \mathrm{Var}(X) & \mathrm{Cov}(X,Y) \\ \mathrm{Cov}(Y,X) & \mathrm{Var}(Y) \end{pmatrix} = \begin{pmatrix} \sigma_X^2 & \rho\sigma_X\sigma_Y \\ \rho\sigma_X\sigma_Y & \sigma_Y^2 \end{pmatrix} = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix}.$$

We have written the covariance matrix in terms of the Pearson correlation between X and Y,

$$\rho = \frac{\mathrm{Cov}(X,Y)}{\sigma_X\sigma_Y} = \frac{\mathbb{E}\left[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])\right]}{\sqrt{\mathbb{E}\left[(X - \mathbb{E}[X])^2\right]\mathbb{E}\left[(Y - \mathbb{E}[Y])^2\right]}} = \frac{0.5}{1 \cdot 1} = 0.5.$$

Using **numpy.random.seed** set your random number generator seed to 0 and answer the following:

*a) Create a dataset by drawing $N = 100$ samples from our model using **numpy.random** function **multivariate_normal**. Create a scatterplot of your data using **matplotlib.pyplot** functions **scatter** or **plot**. Label your axes X and Y. Note: If you use plot() then you need to adjust options to make it a scatter plot.*

b) Compute and report the plug-in estimator of correlation, given by:

$$\hat{\rho}_N = \frac{\sum_i (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_i (X_i - \bar{X})^2 \sum_j (Y_j - \bar{Y})^2}}$$

Where $\bar{X} = \frac{1}{N}\sum_i X_i$ is the sample mean (and similarly for $\bar{Y}$).

*c) Repeat the above process $B = 5,000$ times to generate $\hat{\rho}_{N,1}, \ldots, \hat{\rho}_{N,B}$, each one based on a fresh set of $N = 100$ samples. Display a histogram of your B estimates using **matplotlib.pyplot.hist** with 30 bins. Label your axes.*

d) Use the B estimates obtained in the above question to estimate $\mathbb{E}\left[(\hat{\rho}_N - \rho)^2\right]$, the mean square error (MSE) of plug-in estimator $\hat{\rho}_N$. What is the value of your MSE estimate?