# Data 102, Fall 2023
# Homework 3

Due: **5:00 PM** Friday, October 20, 2023

## Submission Instructions

Homework assignments throughout the course will have a written portion and a code portion. Please follow the directions below to properly submit both portions.

**Written Portion:**

- Every answer should contain a calculation or reasoning.
- You may write the written portions on paper or in $\LaTeX$.
- If you type your written responses, please make sure to put it in a markdown cell instead of writing it as a comment in a code cell.
- Please start each question on a new page.
- It is your responsibility to check that work on all the scanned pages is legible.

**Code Portion:**

- You should append any code you wrote in the PDF you submit. You can either do so by copy and paste the code into a text file or convert your Jupyter Notebook to PDF.
- Run your notebook and make sure you print out your outputs from running the code.
- It is your responsibility to check that your code and answers show up in the PDF file.

**Submitting:**

You will submit a PDF file to Gradescope containing all the work you want graded (including your math and code).

- When downloading your Jupyter Notebook, make sure you go to File → Save and Export Notebook As → PDF; do not just print page from your web browser because your code and written responses will be cut off.
- Combine the PDFs from the written and code portions into one PDF. Here is a useful tool for doing so. As a Berkeley student, you get free access to Adobe Acrobat, which you can use to merge as many PDFs as you want.
- Please see this guide for how to submit your PDF on Gradescope. In particular, for each question on the assignment, please make sure you understand how to select the corresponding page(s) that contain your solution (see item 2 on the last page).

Late assignments will count towards your slip days; it is your responsibility to ensure you have enough time to submit your work.

Data science is a collaborative activity. While you may talk with others about the homework, please write up your solutions individually. If you discuss the homework with your peers, please include their names on your submission. Please make sure any handwritten answers are legible, as we may deduct points otherwise.

## GLM for Dilution Assay

1. In this question, you'll go beyond the four GLM types you saw in class, and explore a new kind of GLM for solving a specific scientific problem.

   Being able to reformulate problems as generalized linear models (GLMs) enables you to solve a wide variety of problems with existing packages. We recommend reviewing the examples of GLMs from Lectures 10 and 11. In particular, make sure you understand that formulating a GLM involves choosing an 1) output distribution and 2) link function that are appropriate for the application at hand.

   In this problem, you'll retrace the footsteps of the statistician R. A. Fisher and develop one of the very first applications of GLMs. In a 1922 paper, Fisher formulated a GLM he used to estimate the unknown concentration $\rho_0$ of an infectious microbe in a solution. Without specialized technology to directly measure $\rho_0$ from the solution, Fisher devised the following procedure: we will progressively dilute the original solution, and after each dilution, we'll pour out some small volume $v$ onto a sterile plate. If zero microbes land on the plate, it will remain sterile, but if any microbes land on a plate, they will grow visibly on it (we call this an "infected plate"). By observing whether or not the plate is infected at each dilution, and by formulating the relationship between this data and $\rho_0$ as a GLM, we can estimate $\rho_0$ from this data.

   Specifically, let $\rho_t$ denote the concentration at dilution $t$. Each time, we dilute the solution to be half its concentration, such that

   $$\rho_t = \frac{\rho_0}{2^t} \tag{1}$$

   for $t = 0, 1, \ldots$. When we pour out volume $v$ of the solution onto the plate, and wait awhile to allow for microbe growth, we can observe whether a plate was infected (*i.e.*, has a non-zero number of microbes) or is sterile (*i.e.*, has zero microbes). Therefore, our data $Y_t \in \{0, 1\}$ is whether or not the plate is infected at each dilution.

   **In other words, we observe a sequence of binary values $Y_0, \ldots, Y_t$, and from that, we want to estimate the initial concentration $\rho_0$.** In this question, we'll formulate a GLM that relates $\rho_0$ and $t$ to the data $Y_t$. Estimating the parameters of this GLM will then allow us to estimate $\rho_0$, as will become clear in the last part.

   (a) (2 points) At dilution $t$, the data $Y_t \in \{0, 1\}$ indicates whether or not the plate is infected. The chance that a plate gets infected is denoted by $\mu(t) := \mathbb{E}[Y_t]$. Write down an output distribution for $Y_t$ that is appropriate for the values it takes on, using $\mu(t)$ as a parameter. (We'll derive what $\mu(t)$ should be in the next part).

(b) (3 points) At dilution $t$, we pour out volume $v$ onto a plate, so the expected number of microbes on the plate is $\rho_t v$. The actual number of microbes is distributed as a Poisson random variable with this mean $\rho_t v$:

$$\#\text{ microbes on plate at dilution } t \sim \text{Poisson}(\rho_t v). \tag{3}$$

Using this fact, write out an expression for $\mu(t) := \mathbb{E}[Y_t]$. Start with

$$\begin{aligned}\mu(t) &= \mathbb{P}(\text{plate is infected at dilution } t) &(4)\\ &= 1 - \mathbb{P}(\text{there are 0 microbes on plate at dilution } t). &(5)\end{aligned}$$

(c) (3 points) Use your findings from part (b), along with Equation (1), to find a link function $g$ such that

$$g(\mu(t)) = \beta_0 + \beta_1 t \tag{8}$$

for some constants $\beta_0$ and $\beta_1$. (Remember that in class, we talked about the inverse link function $g^{-1}$, such that $\mu(t) = g^{-1}(\beta_0 + \beta_1 t)$). Your answer should be of the form "$\beta_0 = \ldots$ and $\beta_1 = \ldots$".

(d) (2 points) Choosing an appropriate output distribution and link function as we've done in Parts (a) and (c) completes the GLM specification. Now, suppose you've estimated $\beta_0$ and $\beta_1$ (*e.g.*, using maximum-likelihood estimation). Write down an estimate of $\rho_0$.

*Hint:* For this question, you do not need to estimate $\beta_0$ and $\beta_1$: assume you know them, and find a way to estimate $\rho_0$ from them.

## Image Denoising with Gibbs Sampling

2. In this problem, we derive a Gibbs sampling algorithm to restore a corrupted image [1]. A grayscale image can be represented by a 2-dimensional array $X$ of shape $n \times m$, where the intensity of the $(i, j)$-th pixel is $X_{ij}$. In this problem, we are given an image $X$ whose pixels have been corrupted by noise, and the goal is to recover the original image $Z$.

(a) (2 points) Load the grayscale image `X.pkl` as a numpy array $X$. Visualize the image. From plotting the image $X$, it is clear that it has been corrupted with noise. Let $Z$ denote the original image, which we also represent as an $n \times m$ array. Let $\mathcal{I} = \{(i, j) : 1 \le i \le n \text{ and } 1 \le j \le m\}$ denote the collection of all pixels in the image, represented by the corresponding index of the array. Given a pixel $(i, j)$, define the set of *neighboring pixels* to be

$$N_{(i,j)} = \left\{(i', j') \in \mathcal{I} : (i = i' \text{ and } |j - j'| = 1) \textbf{ or } (|i - i'| = 1 \text{ and } j = j')\right\}.$$

To capture the fact that, in natural images, neighboring pixels are likely be similar, we consider the following prior over the original image:

$$p(Z) \propto \exp\left(-\frac{1}{2} \sum_{(i,j)\in\mathcal{I}} \left[aZ_{ij}^2 - b \sum_{(i',j')\in N_{(i,j)}} Z_{ij} Z_{i'j'}\right]\right).$$

Assuming the image has been corrupted with Gaussian noise $X_{(i,j)} \mid Z_{(i,j)} \sim \mathcal{N}(Z_{(i,j)}, \tau^{-1})$ (independently across pixels $(i,j) \in \mathcal{I}$), the complete posterior can be written as

$$p(Z \mid X) \propto \exp\left(-\frac{1}{2}\sum_{(i,j)}\left[(a+\tau)Z_{ij}^2 - 2\tau Z_{ij}X_{ij} - b\sum_{(i',j')\in N_{(i,j)}} Z_{ij}Z_{i'j'}\right]\right) \tag{11}$$

Let $S_{ij} = \sum_{(i',j')\in N_{(i,j)}} Z_{i'j'}$. By completing the square in the posterior (11), we have

$$Z_{ij} \mid (Z_{i'j'})_{(i',j')\neq(i,j)}, X \sim \mathcal{N}\left(\frac{\tau X_{ij} + bS_{ij}}{a+\tau}, \frac{1}{a+\tau}\right) \tag{12}$$

(b) (2 points) Fill in the missing line of pseudocode for a Gibbs sampler of the posterior, $p(Z|X)$. **Be specific with each conditioned variable and sub/superscript!**

```
• Initialize $Z^{(0)} = X$.
• For $t = 1, \dots, T$:
    – Sample $Z_{1,1}^{(t)} \sim p(Z_{1,1} \mid Z_{1,2} = Z_{1,2}^{(t-1)}, Z_{1,3} = Z_{1,3}^{(t-1)}, \dots, Z_{n,m} = Z_{n,m}^{(t-1)}, X)$.
    – Sample $Z_{1,2}^{(t)} \sim p(Z_{1,2} \mid Z_{1,1} = Z_{1,1}^{(t)}, Z_{1,3} = Z_{1,3}^{(t-1)}, \dots, Z_{n,m} = Z_{n,m}^{(t-1)}, X)$.
    – Sample $Z_{1,3}^{(t)} \sim$ # TODO: fill this in.
    – ...
    – Sample $Z_{n,m}^{(t)} \sim p(Z_{n,m} \mid Z_{1,1} = Z_{1,1}^{(t)}, Z_{1,2} = Z_{1,2}^{(t)}, \dots, Z_{n,m-1} = Z_{n,m-1}^{(t)}, X)$
```

(c) (3 points) Write the pseudo-code from Part (b) more explicitly both by using a double for-loop over $(i,j) \in \mathcal{I}$ and by being explicit about the conditional distributions of the form $p(Z_{1,1} \mid Z_{1,2} = Z_{1,2}^{(t-1)}, Z_{1,3} = Z_{1,3}^{(t-1)}, \dots, Z_{n,m} = Z_{n,m}^{(t-1)}, X)$.

(d) (5 points) Implement the Gibbs sampler from Part (c) with $a = 250, b = 62.5$, and $\tau = 0.01$. Run your code for $T = 1$ iteration, i.e. update each coordinate exactly once. Visualize the resulting image $Z^{(1)}$. Time your code and estimate how long it would take to compute $Z^{(100)}$.

*Hint*: To convert your pseudo-code into real code, it might be helpful to use `np.random.randn()` to generate a $\mathcal{N}(0,1)$ random variable at each step.

(e) (2 points) The bottleneck in running the Gibbs sampler from Part (d) is sampling a single pixel $Z_{ij}$ with the values of all others held fixed. Fortunately, it is possible to speed up the sampling process with an improvement known as *blocked Gibbs sampling*. Specifically, define two subsets of the pixels $\mathcal{I}_{\text{even}} = \{(i,j) : i+j \text{ is even}\}$ and $\mathcal{I}_{\text{odd}} = \{(i,j) : i+j \text{ is odd}\}$. The blocked Gibbs sampler proceeds as follows:

```
• Initialize $Z^{(0)} = X$.
• For $t = 1, \dots, T$:
    – Let $Z = Z^{(t-1)}$.
    – Let $\Delta$ be an $n \times m$ matrix with $\mathcal{N}(0, \frac{1}{a+\tau})$ entries.
    – For $(i,j) \in \mathcal{I}_{\text{even}}$:
        * Let $S_{ij} = \sum_{(i',j')\in N_{(i,j)}} Z_{i'j'}$
    – Update $Z_{\mathcal{I}_{\text{even}}} = \frac{\tau}{a+\tau} X_{\mathcal{I}_{\text{even}}} + \frac{b}{a+\tau} S_{\mathcal{I}_{\text{even}}} + \Delta_{\mathcal{I}_{\text{even}}}$.
    – For $(i,j) \in \mathcal{I}_{\text{odd}}$:
        * Let $S_{ij} = \sum_{(i',j')\in N_{(i,j)}} Z_{i'j'}$
```

- Update $Z_{\mathcal{I}_{\text{odd}}} = \frac{\tau}{a+\tau} X_{\mathcal{I}_{\text{odd}}} + \frac{b}{a+\tau} S_{\mathcal{I}_{\text{odd}}} + \Delta_{\mathcal{I}_{\text{odd}}}$.
- Let $Z^{(t)} = Z$.

The advantage of this approach is that the inner for-loops can be *vectorized*. Explain why updating half the variables $Z_{\mathcal{I}_{\text{even}}}$ (and then $Z_{\mathcal{I}_{\text{odd}}}$) at once is justified.
*Hint*: if you're not sure why, try drawing out a small grid of pixels and label each one with $i + j$.

(f) (1 point) Implement the Gibbs sampler from Part (e) using $a = 250, b = 62.5$ and $\tau = 0.01$. Run your code for $T = 100$ iterations, and visualize the resulting image $Z^{(100)}$. Time your code and report how long it took.
*Hint:* Compute the entire $n \times m$ matrix $S$ at once using matrix operations on $Z$. You may find it helpful to pad the matrix $Z$ with a border of zeros using `Z_bar = np.pad(Z, 1)`. Then use slicing on the $(n + 2) \times (m + 2)$ matrix `Z_bar` to compute $S$.

## Rejection Sampling

3. (6 points) Consider the function

$$q(x) = \cos^2(12x) \times |x^3 + 6x - 2| \times \mathbb{1}\{x \in (-1, -.25) \cup (0, 1)\}.$$

In this problem, we use rejection sampling to generate random variables with pdf $p(x) = Zq(x)$.

(a) (2 points) Plot $q$ over its domain. What is a uniform proposal distribution $f$ that covers the support of $p$? What is the largest possible constant $M$ such that the scaled target distribution $h(x) = Mq(x)$ satisfies $h(x) \leq f(x)$ for all $x$?

(b) (2 points) Suppose you run rejection sampling with target $h$ and proposal $f$ from part (a) until you generate $n$ samples and your sampler runs a total of $N \geq n$ times, including $n$ acceptances and $N - n$ rejections. Explain how you can use $n, N$ and $M$ to estimate $Z$.
*Hint*: the ratio of acceptances $n$ to total runs $N$ is an approximation of the ratio between the area under the curve $h(x)$ and the area under $f(x)$.
*Hint*: remember what happens if you integrate a pdf over its entire support.

(c) (2 points) Use rejection sampling to generate a sample of size $10^3$ from $h(x)$. Since $p(x)$ is a pdf and it's proportional to $h(x)$, we can display its estimate easily: plot a normalized histogram of your sample, and overlay a smooth kernel density estimate, that will provide more information on the shape of the estimated distribution.
Repeat the previous steps increasing the number of samples to $10^6$.

## References

[1] Stuart Geman and Donald Geman (1984). *Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images.* IEEE Transactions on Pattern Analysis and Machine Intelligence, (6), 721-741.