# Homework 1: Probability

Due: 11:59PM, April 10, 2024

*Preliminaries*

In a 2D space $\mathcal{X} = [0,1] \times [0,1]$, there is a signal source continuously emitting a stochastic signal. You are equipped with a sensor to detect the signal with a binary sensor reading. This means at any location of the 2D space, you can take a measurement $z$, the measurement will be either *Positive* or *Negative*. The likelihood of reading a specific measurement $z$ at a specific location $x$ with a given source location $s$ is governed by the following Bernoulli distribution:

$$p(z|x;s) = \begin{cases} \exp\left(-100 \cdot (\|x-s\| - 0.2)^2\right), & \textbf{if } z = \mathit{Positive} \\ 1 - \exp\left(-100 \cdot (\|x-s\| - 0.2)^2\right), & \textbf{if } z = \mathit{Negative} \end{cases} \tag{1}$$

*Hint*

Taking a measurement is conceptually similar to flipping a potentially unfair coin. But here the "fairness" of the coin depends on the location of the sensor and the location of the source. The function $f(x) = \exp\left(-100 \cdot (\|x-s\| - 0.2)^2\right)$, with a given source $s$, describes how "biased" the measurement is. The larger the value $f(x)$ is at a given sensor location $x$, the more likely the measurement will be *Positive*. This function $f(x)$ has a "ring-shape" around the source, as seen in the figure below.

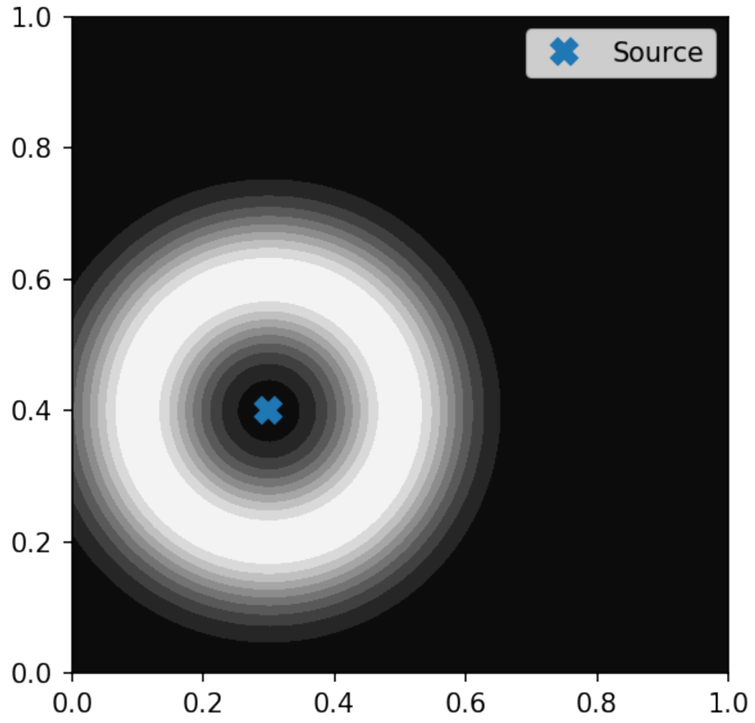

Figure 1: Illustration of the measurement model. The "brighter" the region is, the more likely the sensor will have a *Positive* reading there.

*Software*

Python packages *NumPy* and *Matplotlib* are sufficient for this assignment. To simulate samples from a Bernoulli distribution, you can use a third-party package, or draw a sample uniformly between $[0, 1]$ and see if the sample value is larger than $p(Positive|x; s)$—if this is true, you get a *Positive* reading; otherwise, you get a *Negative* reading. The computation of this assignment could take a while to finish on your computer, but it should not take more than 10 minutes per problem. Even though not required, if you are interested in accelerating the computation, we encourage using Python package *JAX* or *PyTorch* with GPU acceleration (Google Colab provides free GPU access). Feel free to use Jupyter Notebook as the programming environment.

1. (20 pts) Assume the source location is $s = [0.3, 0.4]$, uniformly sample 100 locations in the space, and simulate one measurement for each location. Plot the measurements in the space, if the measurement is positive, plot it as a green dot, otherwise plot it as a red dot. Visualize the "ring-shaped" background seen in Figure 1 as well.
   **Turn in:** Visualization plot only, no code. You should have a plot similar the one below, but the samples should be different.

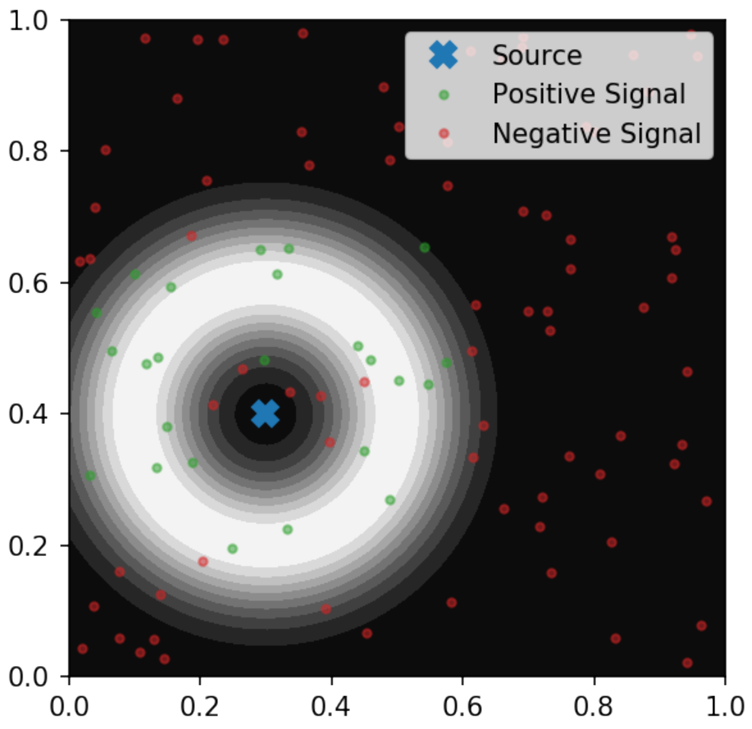

Figure 2: Example visualization of Problem 1.

2. (20 pts) Given the 100 measurements generated from Problem 1, denoted as $\{(x_i, z_i)\}_{i=1 \dots N}$, now assume you do not know the source location and instead would like to estimate it from the 100 measurements. You can estimate the likelihood of the source location $s$ through the following equation:

$$\mathcal{L}(s) = \prod_{i=1}^{N} p(z_i | x_i; s) \tag{2}$$

Visualize the likelihood function $\mathcal{L}(s)$, which is in the same domain as the sensor, across the whole search space $\mathcal{X}$ with the measurements from Problem 1. Visualize the measurements on top of the likelihood function plot (with the same color scheme from Problem 1). Note that you will need a mesh grid with at least 50 grid cells per dimension to sufficiently visualize the likelihood function. It is recommended to have 100 grid cells per dimension.
**Turn in:** Visualization plot only, no code.

3. (20 pts) Repeat the process for Problem 2, but uniformly sample *one* location in the space and assume that the sensor is stuck at that location for all 100 measurements. Note that the results may vary based on your samples. So randomize your samples for multiple times and feel free to include up to 3 plots if the results are different.
   **Turn in:** Visualization plot(s) only, no code.

4. (20 pts) Assume the same source location and assume you do not know the source location but instead would like to localize the source from sequential online observations. Uniformly sample *one* location in the space and assume that the sensor is stuck at that location. Simulate 10 measurements *sequentially* at that location. For each new measurement from the sensor, use Bayes' rule to update the belief of the source location, with the initial belief being a uniform distribution.

   For each measurement, visualize the source location (even though your algorithm does not know it), the location where you take the measurement, the measurement (green for positive, red for negative), and the current belief. Organize the plots from the 10 measurements into a 2-by-5 grid.
   **Turn in:** Visualization plot only, no code.

5. (20 pts) Repeat the process for Problem 4, but instead of fixing the location for measurement, you can sample at a new location to take each new measurement. You can choose the sampling strategy for the sensor location, though uniform sampling is sufficient.
   **Turn in:** Visualization plot only, no code.