Cognitive Modeling: Homework Assignment 1
Technical Stack

February 3, 2024

All answers and solutions to non-programming questions should be submitted to LMS as a **legible** write-up (either fully digital or a scan). All code should be committed to and merged into the `main` branch of your team's GitHub repository.

Problem 1: True-False Questions (4 points)

Mark all statements which are **FALSE**.

1. A random variable is discrete if its support is countable and there exists an associated probability density function (pdf).
2. Probability mass functions have a lower bound of 0 and an upper bound of 1.
3. The set of all possible realizations of a random variable is called its probability density.
4. The expected value of a discrete random variable is always part of its support, that is, $\mathbb{E}[X] \in R_X$.
5. Continuous random variables are functions which map points from the sample space to the real numbers.
6. We can formulate most parametric Bayesian models as a generative process, by which we first sample from the likelihood and then use the synthetic data point to sample from the prior.
7. The Bayesian posterior $p(\theta \mid y)$ for continuous parameter vectors $\theta \in \mathbb{R}^D$ is just another density function. That means, its integral $\int p(\theta \mid Y = y)\, d\theta \neq 1$ for some y.
8. Each realization of a continuous random variable has a probability of zero.

Problem 2: Git and GitHub (8 points)

1. Create a public GitHub repository, create and add a team logo to the `README` file, along with some basic introductory notes on why cognitive modeling is important for psychology and cognitive science. Create an `environment.yml` file and add all dependencies we have discussed so far. Then, in addition to the `main` branch, create separate branches for each of the two team members, from which you will be merging working code into the `main` branch.

2. Create a *merge conflict* (either for some of the coding exercises or a mock conflict) and resolve it.

3. Explain the differences between the following git commands

   (a) `git restore`

   (b) `git checkout`

   (c) `git reset`

   (d) `git revert`

   in terms of undoing changes to a repository by providing a minimal (actual or a synthetic) example.

Problem 3: Expectations I (4 points)

Suppose that you want to invest some money in the oil market. You believe that the probability of the market going up is 0.8 and the market going down is 0.2. Further, if the market goes up, oil prices will increase by 1%, if it goes down, oil prices will drop by 10%. What is your expectation? Would you invest in this market? Assuming the increase/drop in prices is fixed, what is the minimal probability of prices going up in order for you to invest rationally (according to expectation) in this market? Discuss a limitation of expectations when making single-shot, real-life decisions.

Problem 4: Expectations II (4 points)

1. Show the following identity for the variance of a random variable X:

$$\text{Var}\left[X\right] = \mathbb{E}\left[X^2\right] - \mathbb{E}\left[X\right]^2 \tag{1}$$

2. Show the following property for the variance of a random variable X and a scalar α:

$$\text{Var}\left[\alpha X + \beta\right] = \alpha^2 \text{Var}\left[X\right] \tag{2}$$

3. Assume you are given a random variable X with a standard normal distribution (mean zero, variance one). We can write this as

$$X \sim \text{Normal}(\mu = 0, \sigma = 1) \tag{3}$$

One way to sample from this distribution is using NumPy's function `numpy.random.randn`. What transformation do you need to apply to the sampled values (i.e., the outputs of the function) such that they are now distributed according to

$$\tilde{X} \sim \text{Normal}(\mu = 3, \sigma = 5)? \tag{4}$$

Problem 5: Simple Bayesian Inference (4 points)

The inhabitants of an island tell the truth one third of the time. They lie with probability 2/3. On an occasion, after one of them made a statement, you ask another "Was that statement true?" and he says "yes". What is the probability that the statement was indeed true?

Problem 6: Murder Mystery Revised (6 points)

Construct a small story of your own choosing in the spirit of the "Murder Mystery" we considered in class (can be from a completely different domain). Define a "prior" distribution, a "likelihood", and justify your selection of probabilities. Create a first folder in your team repository and write a script containing fully vectorized code which simulates your story and outputs a table representing the approximate joint probabilities (which we approximated through the relative frequencies across N simulation runs). Compare this table to the analytic probabilities. How large should N be for the approximate probabilities to become almost indistinguishable from the analytic ones?

Problem 7: Priors, Sensitivity, Specificity (6 points)

Let's revisit the task we disease problem we tackled during class. Imagine you are a medical researcher analyzing the effectiveness of a new diagnostic test for a rare disease **X**. This disease affects 1% of the population. The probability of a true positive (the test correctly identifies an individual as having the disease) is 95%. This is also known as the *sensitivity* of the test. The probability of a true negative (the test correctly identifies an individual as not having the disease) is 90%. This is also known as the *specificity* of the test.

We will now consider a question of sensitivity analysis (not to be confused with the sensitivity of a test): How would the posterior probability change if the prior, the sensitivity, or the specificity of the test were to test. Write a Python program which produces three pretty and annotated 2D graphs depicting

1. The posterior probability (Y-axis) of actually having the disease given a positive test as a function of the prior probability (X-axis), assuming fixed sensitivity and specificity.

2. The posterior probability (Y-axis) of actually having the disease given a positive test as a function of the test's sensitivity (X-axis), assuming fixed prior and specificity.

3. The posterior probability (Y-axis) of actually having the disease given a positive test as a function of the test's specificity (X-axis), assuming fixed prior and sensitivity.

Briefly discuss how the posterior changes as a function of each of the quantities.

Bonus (4 points) Generate three 3D plots (either surface plots or scatter plots) depicting the same posterior probability as a function of the combination of two quantities (prior - sensitivity, prior - specificity, sensitivity - specificity).

Problem 8: Monte Carlo Approximation (4 points)

Write a Python program that approximates the value of π using Monte Carlo approximation. Your program should generate a sequence of random points and use these points to estimate the value of π. The accuracy of the approximation should improve as the number of points increases. Here are some hints:

- Your program should generate random points with x and y coordinates ranging between -1 and 1. This will simulate points within a 2×2 square that circumscribes a unit circle centered at the origin $(0, 0)$.
- For each generated point, determine whether it falls inside the unit circle. A point $p = (x, y)$ is inside the circle if $x^2 + y^2 \leq 1$.
- Use the ratio of the number of points that fall inside the circle to the total number of generated points to approximate π. The formula is given by $\pi \approx 4 \times$ (number of points inside/total number of points).

Plot the approximation error of your Monte Carlo estimator as a function of the total number of points N. You can use `np.pi` as the ground truth.

Problem 9: AI-Assisted Programming (4 points)

Use ChatGPT or any other large language model (LLM) to generate a function called `multivariate_normal_density(x, mu, Sigma)` which returns the density of a D-dimensional vector `x` given a D-dimensional mean (location) vector `mu` and a $D \times D$-dimensional covariance matrix `Cov`. Compare the outputs of your function with those obtained using SciPy's `scipy.stats.multivariate_normal` for a few parameterizations including a spherical Gaussian (zero covariance, shared variance across dimensions), a diagonal Gaussian (zero covariance, different variance for each dimension), and a full-covariance Gaussian (non-zero covariance, different variance for each dimension). Describe briefly how the LLM performed.