PSYCH-GA.2211/NEURL-GA.2201 – Fall 2026
Mathematical Tools for Neural and Cognitive Science

Homework 4

Due: 10 Nov 2026
(late homeworks penalized 10% per day)

See the course web site for submission details. For each problem, show your work - if you only provide the answer, and it is wrong, then there is no way to assign partial credit! And, please don't procrastinate until the day before the due date... *start now*!

1. **Bayesian inference of eye color.** A male and female chimpanzee have blue and brown eyes, respectively. The brown-eyed allele is denoted with a capital B, and the blue-eyed allele with a lowercase b. Assume a simple genetic model in which the gene for brown eyes is always dominant (so that the trait of blue eyes can only arise from two blue-eyed genes, but the trait of brown eyes can arise from two brown-eyed genes, or one blue and one brown). Children get one allele from each parent. You can also assume: i) the *a priori* probability of the mother being BB (or Bb) is 50%; and ii) the *a priori* probability that a child is born with any one of the four gene configurations from the two parents is 25%. For each question, provide the math, and explain your reasoning.

   (a) Suppose you observe that they have a single child with brown eyes. Conditioned on this observation, what is the probability that the female chimp has a blue-eyed gene?

   (b) Suppose you also observe that they have a second child with brown eyes. Condiitoned on both observations, what is the probability that the female chimp has a blue-eyed gene?

   (c) Generalizing, suppose you observe that they have N children, all of whom have brown eyes... express the probability that the femle chip has a blue-eyed gene, as a function of N.

2. **Poisson neurons.** The Poisson distribution is often used to model neural spike counts:

$$p(k) = \frac{\mu^k e^{-\mu}}{k!},$$

   where k (a non-negative integer) represents the spike count over some pre-specified time interval, and μ (a non-negative real number) is the rate, which specifies the *expected* spike count in that interval.

   (a) Visualize the (truncated) Poisson distribution. Set the expected number of spikes to $\mu = 6$ spikes/interval then create a vector `p` of length 21, whose elements contain the probabilities of Poisson spike counts for $k = [0...20]$. Since we're truncating the range at a maximum value of 20, you'll need to normalize the vector so it sums to one (the distribution given above is normalized over the range from 0 to infinity) so that the vector `p` represents a valid probability distribution. Plot `p` in a bar plot and compute and mark the mean firing rate. Is it equal to μ? Why or why not?

(b) Write a function that generates `num` samples from a distribution `p` specified by a vector (such as the truncated Poisson distribution you created in the previous part), `samples = randp(p, num)`. [Hint: see class slides: use the `rand` function, which generates real values over the interval [0...1], and partition this interval into portions proportional in size to the probabilities in `p`]. Test your function by drawing 1,000 samples from the truncated Poisson distribution in (a), plotting a histogram of how many times each value is sampled, and comparing this to the true frequencies specified by `p`. Verify qualitatively that the answer gets closer (converges) as you increase the number of samples (try 10 raised to powers $[2, 3, 4, 5]$).

(c) Imagine you're recording with an electrode from two neurons simultaneously, whose spikes have very similar waveforms (and thus are not easily distinguished by your spike sorting software). Create a probability vector, `q`, for the second neuron, assuming a mean rate of 3 spikes/interval. What is the probability distribution of the observed spike counts, which will be the sum of spike counts from the two neurons derived from `p` and `q`? [Hint: the distribution vector should have length $m+n-1$ when m and n are the lengths of the two input PDFs.] Verify your answer by comparing it to the histogram of 1,000 samples generated by adding samples from the two distributions (generated using `randp`).

(d) Now imagine you are recording from a neuron with mean rate 9 spikes/interval (the sum of the rates from the two neurons above). Plot the distribution of spike counts for this neuron, in comparison with the distribution of the sum of the previous two neurons. Based on the results of these two experiments, if we record a new spike train, can you tell whether the measured spikes came from one or two neurons just by looking at the distribution of spike counts? Can you explain this based on properties of the Poisson distribution?

3. **The Central Limit theorem.** The Central Limit theorem states that the distribution of the average of n independent samples drawn from any fixed distribution with finite mean and variance, gets closer and closer to a Normal (Gaussian) distribution as n increases. Specifically, if the mean and variance of the original distribution are μ and σ, the distribution of $\sqrt{n}(\bar{x} - \mu)/\sigma$ converges to $\mathcal{N}(0, 1)$ as n increases (where $\bar{x}$ is the average of n samples).

(a) Generate 1,000 samples of two values each from a uniform distribution (use `rand`). Compute the average of each sample (pair of values), and plot a histogram of these. What shape is it, approximately? What shape should it have in the limit, as you gather more and more samples (try with 100,000 samples)? Why?

(b) Now try this again with samples containing 3 values. How has the histogram changed? Try sample sizes of 4 and 5 as well. When do you judge that the histogram starts looking Normal?

(c) Test the Normality of the distribution a bit more carefully, using a "Q-Q" (quantile-quantile) plot (plot the quantiles of one distribution against another). If the two distributions match, the values should lie on a unit-slope line. For this problem, you can use the matlab function `normplot`, which plots the quantiles of a sample of data against those of a Normal distribution of the same mean and variance. First, try this on a sample of 1,000 values from a normal distribution (use `randn`). The points should fall (close to) a straight line, indicating that the sample is close to normal, as expected. Try this a few times to see how the plot varies (you might want to put them on the same graph, using matlab's `hold on` command). Now call `normplot` on a sample of 1,000

values from a uniform distribution. Explain qualitatively why it has the shape it does (hint: think about the quantiles of the uniform and Normal distributions). Do this for averages of uniform samples of different size (2, 3, 4, ...). Keep increasing sample size until you cannot tell the resulting QQ plot from the QQ plots for samples from the Normal distribution. Roughly how big does the sample have to be?

4. **Analyzing and simulating experimental data.** An international coffee conglomerate recruits you to characterize the neuropsychology underlying their customers' adoration of pumpkin spice flavoring. You devise a blood-oxygen level dependent (BOLD) fMRI pilot experiment in which you present one of two classes of odorants to an individual while monitoring the activity of three key voxels located in the amygdala, a structure known to be associated with emotional responses. The file `experimentData.mat` contains: a $(N \times 3)$ matrix `data`, where each row is the BOLD response of the three voxels on a given trial relative to some baseline; and a $(N \times 1)$ vector `trialConds` indicating the experimental condition of each trial. Condition 1 includes trials in which you present an odorant selected randomly from a library of possible control odorants, and condition 2 includes trials in which the trade-secret pumpkin-spice odorant is presented.

   (a) Before doing anything quantitative with your data, it is always good practice to visualize it. First, determine how many trials of each condition were completed. Display this information as a 2-bin histogram with each bin representing each of the two possible conditions, and their heights representing their respective trial counts. Next, plot a 3D scatter plot of the recorded responses, with each point color-coded according to its associated condition (use the function `scatter3` in Matlab (or see footnote[1] for Python) and be sure to label your axes). Describe your data qualitatively using this figure. Is there a noticeable difference between the two conditions? What geometric shape are these 'response clouds', and what distribution would you use to model them?

   (b) Quantify the response statistics of each individual condition. Calculate the means of each response cloud, as well as their respective covariance matrices. Compute the covariance matrices of each response cloud using matrix multiplication (remember to center the data first). Verify that your calculation is correct by comparing with the output given by the `cov` function. How do the covariance matrices compare between condition 1 and condition 2 (are they similar at all or wildly different)?

   (c) Next, compute the SVD of each covariance matrix. Plot the three singular vectors originating from the center of each response cloud and scale their amplitude by the square root of the singular values. Relative to how similar the covariance matrices were before computing their SVD, how do each condition's respective set of singular values compare? Describe what this tells us about the relationship between the two conditions and, more fundamentally, the relationship between the three voxels across conditions.

   (d) A powerful method to validate a model is by *generating* (i.e., simulating) new data matching your quantitative description of the real data, and then comparing them with real data. Write a function `samples = ndRandn(mean, cov, num)` that generates a set of samples drawn from an N-dimensional Gaussian distribution with the specified `mean` (an N-vector) and `cov`ariance (an NxN matrix). The parameter `num` should be

[1] Make sure you run `from mpl_toolkits.mplot3d import Axes3D` and `%matplotlib notebook` at some point. Then run `fig = plt.figure(); ax = fig.add_subplot(111, projection('3d')); ax.plot(''whatever you want'')`. Note that this does *not* work in Colab, and you need to have Jupyter notebook on your own computer for interactive 3D plots.

optional (defaulting to 1) and should specify the number of samples to return. The returned value should be a matrix with `num` rows each containing a sample of N elements. [Hint: generate samples from an N-dimensional Gaussian with zero mean and identity covariance matrix, and then transform these to achieve the desired `mean` and covariance.] Test your function for $\mu = [3, 5]$ and $C = [10, -4; -4, 5]$, drawing 1,000 points. Compute the sample mean and sample covariance of your simulated data, and compare to the requested mean and covariance.

(e) Now create a function `simResponses = odorExperiment(numTrials1,numTrials2)` where `numTrials1` and `numTrials2` are the number of trials in a simulated experiment for condition 1 and 2, respectively. `simResponses` is an $(N \times 3)$ matrix containing simulated responses of each of your 3 voxels during $N =$ `numTrials1` + `numTrials2` trials. Plot the simulated and real responses in the same figure (use subplots if you wish). Is your simulated response data a good characterization of the real amygdala voxel responses?

5. **Optional (if didn't get it sufficiently in Homework 2): Principal components.** Load the file `PCA.mat` into your MATLAB environment. You'll find a matrix M containing responses of a population of 14 neurons, under 150 different experimental conditions (each column contains the estimated firing rate of one neuron under the 150 conditions). The conditions correspond to an animal reaching for a target in different directions, and different distances away from a central resting position. We cannot directly visualize data of this many dimensions, but we can use linear algebra to project them into a lower-dimensional space.

   (a) Compute the principal components of the 14-dimensional population responses (these will each be vectors with 14 components, i.e., one weight per neuron, thus treating conditions as 150 samples of 14-dimensional neural population responses). First, center the data by subtracting the mean response $mean(M)$ from every row of the matrix (hint: you can use an outer product as described in class, or you might find the function `repmat` helpful). Call this re-centered data matrix $\tilde{M}$. Then compute the eigenvectors and eigenvalues of $\tilde{M}^T\tilde{M}$ (alternatively, you can compute the singular values of $\tilde{M}$). Plot the eigenvalues (or singular values). What do you think is the "true" dimensionality of the responses? How does this change if you do not mean-center the data first?

   (b) Project the data in $\tilde{M}$ onto the first principal component (i.e., the eigenvector corresponding to the maximal eigenvalue). Plot a histogram (using `hist`) of these values. Verify that the sum of squares of these values is equal to the first eigenvalue λ_1. What proportion of the total variance of the data (i.e., sum of squares of all entries of $\tilde{M}$) does this component account for?

   (c) Show a scatter plot of the data projected onto the first two principal components (that is, plot the inner product of the data with the first component versus the inner product with the second component). You can use `plot` (with circular plot symbols and no connecting lines), or use `scatter`. Use `axis('equal')` to set the two axes to use equal scales. Show that the sum of the squared lengths of these projected vectors is equal to $\lambda_1 + \lambda_2$. What proportion of the total variability of the data do these two components account for?