PSYCH-GA.2211/NEURL-GA.2201 – Fall 2025
Mathematical Tools for Neural and Cognitive Science

Homework 6

Due: 18 Dec 2025
(late homeworks penalized 10% per day)

See the course website 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. **Fitting a 2AFC psychometric function.**

   Consider a two-alternative forced-choice (2AFC) psychophysical experiment (fancy name: heterochromatic brightness matching). Subjects are shown a blue spot and a red spot and must decide which appears brighter. The intensity of the blue spot is fixed, and that of the red spot is randomly varied over trials. The purpose of the experiment is to estimate the intensity of red that matches the blue as well as the difference in intensity required for the two to be noticeably different. For a red spot of brightness I, the probability of the observer saying "The red spot is brighter" is:

   $$p(I) = \lambda * \frac{1}{2} + (1 - \lambda) * \Phi(I; \mu, \sigma),$$

   where $\Phi(I; \mu, \sigma)$ is the cumulative distribution function of the Gaussian (`normcdf` in matlab) with mean μ and standard deviation σ, evaluated at I. The parameter λ is called the "lapse rate" and is the proportion of trials the observer didn't pay attention and just guessed. The function $p(I)$ is known as the *psychometric function*.

   You will start by simulating performance in this task. Then, you'll simulate the *inverse* (scientific) side of the problem, and use this probabilistic model as a means of fitting/analyzing the simulated data set and estimating its parameters.

   (a) Write a function `B=simpsych(lambda,mu,sigma,I,T)` to simulate an experiment. The arguments `(I,T)` are vectors of equal length, the first containing a list of intensities and the second containing the number of trials to be run for each corresponding intensity. The function should generate draws from $p(I)$, and return a vector, B, (of the same length as I and T), containing the number of trials for which the simulated observer responded that the red spot was brighter, for each intensity I.

   (b) Illustrate the use of `simpsych` with `T=ones(1,7)*100` and `I=1:7` for $\lambda = 0.05$, $\mu = 4$ and $\sigma = 1$. Plot `B./T` as a function of `I` (as points) and plot the psychometric function $p(I)$ (as a curve) on the same graph.

   (c) Write a function `nll = nloglik(mu,sigma,lambda,I,T,B)` that returns the negative log likelihood of parameters `mu`, `sigma`, and `lambda` for data set `I,T,B` (we're negating it because we will be *minimizing* this function to solve for the optimal parameters).

   (d) Use the `matlab` function `fminsearch` to estimate the values of `mu`, `sigma`, and `lambda` that minimize the function `nloglik(mu,sigma,lambda,...)` for the dataset you generated above.

Two comments: first, the syntax for calling `nloglik` within `fminsearch` is a bit odd: `fminsearch(@(x) nloglik(x(1),x(2),x(3),I,T,B), <startpoint>)`.
Here, the `@` notation is used to create a temporary function, with argument `x` a vector containing the three variables being optimized (mean, standard deviation and lapse rate). Second, you'll need to specify a start point for the search – for this problem, `[2,2,.05]` is a reasonable choice. Were the estimates close to the true values used to generate the data?

(e) A variant of `fminsearch`, `fminunc`, also returns the *Hessian* (the matrix of second derivatives) of the negative log likelihood at the optimal values `mu`, `sigma` and `lambda`. (Note: `fminunc` is less robust than `fminsearch`, and if the optimizer strays too far from the true values, there may be numerical problems due to overflow of the likelihood; in this case, try a different starting point.) As discussed in class, the inverse of the Hessian provides an estimate of the covariance matrix of the parameter estimates. Use this to determine 95% confidence intervals (mean ± 1.96 standard deviations) for each parameter. Do the true parameter values fall within these confidence intervals? Do any pairs of parameters trade off (i.e., covary)?

(f) Produce a second set of confidence intervals for the parameters using a bootstrap method. For each of the 7 intensities, resample 100 trials (i.e., responses that the red spot is brighter or darker) from the 100 trials of that intensity in the original data, with replacement. Refit the model to the resampled data using `fminsearch`. Plot the histograms (function `hist`) of `mu`, `sigma` and `lambda` estimates obtained over 500 such resampled datasets, and define your confidence intervals as the region between the 2.5th and 97.5th percentiles of these distributions. How well do these values agree with those from the previous part?

2. **Regularization and cross-validation.**

(a) Start by generating a noisy dataset based on a polynomial $y = 9x^7 - 6x^5 + 4x^3 - 3x + 0.5 + \epsilon$, where $\epsilon \sim N(0, 0.3)$. Generate an x vector with 201 points ranging from -1 to 1 (e.g., using `linspace`) and the corresponding noisy values of y.

(b) Implement a leave-one-out cross-validation method, by fitting a polynomial to the data with one data point withheld. Compute the mean squared error (MSE) of the fitted polynomial on the training data, and on the test data, averaging over fits leaving out each of the 201 datapoints in turn. Plot these two values as a function of polynomial order, from 0 through 8, and plot the MSE as a function of order. What order polynomial provides the best solution, according to this analysis?

(c) Implement a LASSO regularized regression solution, as developed in class, using `fminunc` (or `scipy.optimize.minimize` for python users). Try a range of regularization strengths (λ values), plot cross-validated error as a function of λ. Select the best value of λ, and justify your choice. Which terms are used, and which are set to zero? How does this compare to the polynomial order chosen in the previous part and to the ground truth polynomial? Hint: start with very small values of λ (around 1e-6). To generate a logarithmically spaced range of λ values, you can use `np.geomspace` in Python or `logspace` in MATLAB.

(d) Eliminate the polynomial terms that were set to zero by the LASSO (i.e., eliminate those regressors from the matrix), and re-fit the data with this reduced set of terms. Compare the number of terms and cross-validated error of this solution to those of the solution in part b. Which do you prefer?

3. **Reverse Correlation.** From the course web page, download this function

   `[spikes, stimuli] = runGaussNoiseExpt(kernel, duration)`

   that simulates a white noise (reverse correlation) experiment. The `kernel` is a spatial weighting vector, and `duration` specifies the total number of random stimuli that will be shown. The function returns `spikes`, a binary vector indicating which stimuli produced spikes, and `stimuli`, a matrix whose rows contain the stimuli.

   (a) Generate a 100-sample response vector by running the function on the spatial kernel: `[1 2 1; 2 4 2; 1 2 1]/6`. This is a matrix, which you'll need to stretch out into a column vector (i.e., `kernel(:)`) before passing it into the function. Note that this kernel is unit-norm. Plot (on two subplots of the same figure) the linear filter response to the stimuli (you should be able to compute this with a single matrix multiplication!) and the spike train, both as functions of time. Do you see a relationship between these? Display a 2D scatter plot of the raw stimulus intensities at positions 1 and 6, which should look like samples of a 2D Gaussian. On top of this (use `hold on`), plot in red only the stimulus intensities that produced spikes. Use `axis equal` to use equally-scaled axes. Where in this 2D stimulus space do the spikes occur? Does this match your expectations?

   (b) Now compute the spike-triggered average (STA) for the simulated data from this "experiment". Rescale it to have unit norm, reshape it into a matrix, and display it as a grayscale image. Display the true kernel next to it (use `subplot`). How similar is the STA to the true kernel? Characterize the error of the STA, as a function of the duration of the experiment. For durations $100, 400, 1600, 6400, 25600, 102400$, run the experiment 100 times, compute the mean STA across these 100 runs, and subtract this from the true kernel, and compute the average of this difference kernel (the estimation bias). Also, subtract the computed mean from the 100 STAs, and compute the average squared error over these (the estimation variance). Plot the bias and square root of the variance as functions of the duration – you might want to look at a log-log plot (matlab has a function `loglog`). What do you conclude about how the bias and variance behave, as a function of the amount of data?

   (c) Estimate the nonlinearity of the response. Take the stimuli, spikes, and STA from a single run of the Gaussian noise experiment with duration 6400, project the stimuli onto the STA. Sort these projections from highest to lowest (using matlab's `sort` function) and re-order the spike vector to maintain correspondence (the sort function will give you the indices of the sorted values). Now collect the mean projection values, and the mean spike count, into bins containing each consecutive group of 200 indices (this should result in two vectors of length 32), and plot these against each other. You should see an estimate of the spiking nonlinearity. On the same graph, plot the spike responses (zeros and ones) as a function of the projection of their corresponding stimuli onto the STA.

4. **Classification (decision) in a 2-dimensional space**. Load the file `fisherData.mat` into your MATLAB environment. The file contains two data matrices, `data1` and `data2`, whose rows contain hypothetical normalized responses of 2 mouse auditory neurons to different stimuli – The first matrix contains responses to dogs barking, and the second are responses to cats meowing. You would like to know whether the responses of these two neurons could be used by the mouse to differentate the two types of sound. We'll implement three *classifiers.*

   (a) First consider the linear discriminant corresponding to the difference in means of the two data sets (the "prototype classifier"). Write the math to show that this solution is the Maximum Likelihood classifier under the assumption that the data are drawn from Gaussian distributions with different means and identity covariance (or any scalar multiple of the identity matrix). Visualize the solution by generating a binary image showing the classification output. Specifically, use `meshgrid` to generate X and Y coordinate images covering a region that extends a bit beyond the range of the data, create images of two Gaussians (with mean matching the corresponding data sets and identity covariance), and then calculate an image of the binary classifier by comparing the two Gaussian images. Display the image using `image` (make sure to provide x and y coordinates), and use `hold` to scatter plot the data on top of the image. Now compute the discriminant vector (compute the difference of the means of each data set, and normalize to unit length). Scatterplot the data (using different colors for the two data sets), and plot the discriminant vector and the decision boundary on top of this. What fraction of the points are correctly classified by this classifier?

Project the two data sets onto this discriminant, and plot histograms of each (use `hist`, and put them in the same plot by using `hold on` and `hold off`). How well separated are the two distributions?

(b) Now use Fisher's Linear Discriminant, which maximizes the average squared between-class mean distance, while minimizing the sum of within-class squared distances (see Notes on regression). Write the math to show that this classifier is the ML solution when the data are drawn from Gaussian distributions with different means, but the same covariance matrix (which need not be a multiple of the identity!). Estimate the common covariance, Σ_{Data}, by averaging together the sample covariances of the two data matrices. Repeat the plotting exercises of part (a) to visualize the solution. Again, what fraction of the points are correctly classified by this classifier?