**PSYCH-GA.2211/NEURL-GA.2201 – Fall 2023**
**Mathematical Tools for Neural and Cognitive Science**

# Homework 6

Due: 19 Dec 2023
(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. **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 (an *erf* function, `normcdf` in matlab) with mean $\mu$ and standard deviation $\sigma$, evaluated at $I$. The parameter $\lambda$ 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, estimating its parameters and comparing models.

(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` vs `I` (as points) and plot the psychometric function $p(I)$ (as a curve) on the same graph.

(c) Do the same with `T=ones(1,7)*10` and plot the results (including the psychometric function). What is the difference between this and the plot of the previous question?

(d) 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).

(e) 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 in (b). 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?

(f) 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.) The inverse of the Hessian provides an estimate of the covariance matrix of the parameter estimates. Use this to determine 95% confidence intervals on each parameter (Hint: a 95% confidence interval is the mean $\pm 1.96$ standard deviations of the parameter estimate. Compute the standard deviation of a marginal of the 3-D Gaussian that has covariance equal to the inverse Hessian.) Do the true parameter values (4, 1 and .05) fall within these confidence intervals?

(g) 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 part (f)?

(h) Simulate the experiment using `simpsych` twice, once using the original parameters and again changing the `mu` value to 6. We now consider a couple of approaches to test whether those two datasets differ significantly in `mu`. First, pool the datasets (treat it as one big dataset for a single psychometric function) and fit using your code from part (d). Now, write a new function, `nloglik2`, that computes the likelihood of a 4-parameter model: shared values of `sigma` and `lambda` for the two datasets, but separate parameters `mu1` and `mu2` for each dataset. Fit this model to the data from the two simulations. Report the fit parameters for these. Now, compare the two models using AIC and BIC (reusing the results of the fit from part (g)). What are the AIC and BIC statistics and do they "seem" large enough to support the more complex model (i.e., that the two `mu` values differ)?

(i) Finally, construct a permutation test of the null hypothesis (i.e., the hypothesis that there has been no change in `mu` between the two datasets). For each intensity, combine the 100 trials from each condition into a total of 200, then randomly partitioning this into two groups of 100. Fit both resampled datasets with your original one-sample model, noting the difference between the two `mu` estimates. Repeat this process 500 times to produce a null distribution of the differences. Now fit the two datasets separately and compute the difference between the two `mu` estimates. How likely (at what quantile; one-tailed $p$-value) is that difference in `mu` according to the null distribution? Do these results make sense given the true parameter values from which you simulated the datasets?

2. **Psychopathy.** You are interested in causes and treatment options for Psychopathy. You obtained a dataset, contained in the file `psychopathy.mat` obtained from a prison for violent offenders in upstate New York (not everyone in the prison is a psychopath, but they are more prevalent than in the general population). All study participants underwent a structural scan with a mobile, truck-mounted MRI. Each row of the matrix represents data from one prisoner. The first column contains the estimated cortical volume of paralimbic areas, relative to the population median, in cm$^3$. The second column contains the Hare Psychopathy Checklist (PCL-R) scores, which range from 0 to 40 (the higher the score, the more psychopathic traits someone exhibits). These scores are *not* distributed normally in either the general population (median = 4) or this prison subpopulation (median = 20). The third column indicates whether they already participated in an experimental treatment program known as "decompression therapy" (0 = did not yet participate, 1 = did already participate). To avoid self-selection effects, everyone in this dataset agreed to the therapy, but prisoners were randomly assigned to an earlier and a later treatment group, so that the untreated prisoners could serve as a control group.

   (a) Use polynomial regression to model PCL-R scores as a function of relative volume of paralimbic areas. (Note, you might make use of your code from HW2.) Use cross-validation to determine the best polynomial degree.

   (b) Use bootstrapping methods to estimate the 95% confidence interval of the average paralimbic volume of the decompression treatment group vs. the control group. If the random assignment worked, the confidence intervals should overlap. Do they? Also, do these data suggest that there is a statistically reliable difference from the general population, in terms of paralimbic volume?

   (c) Do a permutation test to assess whether decompression therapy has an effect. Designate an appropriate test statistic and calculate its p-value.

3. **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 cat vocalizations. 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, $\Sigma_{Data}$, 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?

(c) Fisher's discriminant suffers when there's not enough data to estimate the covariance matrices. Compute the *ridge-regularized* Fisher's discriminant, by estimating the covariance matrix as $\Sigma_{Estimated} = (1 - \lambda)\Sigma_{Data} + \lambda I$, where $\Sigma_{Data}$ is the sample covariance matrix of the data (as in part (b)), and $I$ is the identity matrix. The parameter $\lambda$ controls the regularization, allowing the solution to transition between the prototype classifier ($\lambda = 1$) and Fisher's Discriminant ($\lambda = 0$). Test the classifier for values $\lambda = [0 : 0.05 : 1]$ using 95%-5% cross-validation (i.e., 100 times, sample *without replacement* 95% of the data from each class, and test classification performance on the remaining 5%). Plot your cross-validated test-set performance (with error bars) as a function of $\lambda$ and state which $\lambda$ you think is best.

(d) Finally, consider the Quadratic Classifier that computes the ML decision rule for the general case of two Gaussian distributions (i.e., each with its own mean and covariance). Specifically, estimate the mean and covariance of data measured for each condition, and calculate the classifier that chooses the class of each data point based on which of the two Gaussians has higher probability at that location (write out the math). Repeat the plotting exercises of part (a) leaving out the discriminant vector (which doesn't exist for the quadratic classifier). Calculate the fraction of correctly classified data points. Which of the four classifiers (prototype, LDA, regularized LDA, or QDA) is best? Are there data scenarios in which you might prefer to use one of the inferior classifiers?