PSYCH-GA.2211/NEURL-GA.2201 – Fall 2019
Mathematical Tools for Neural and Cognitive Science

Homework 6

Due: 13 Dec 2019
(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 psychometric function.** In HW5-Q4, we simulated a subject performing a 2-alternative forced choice psychophysics experiment. We will now simulate the *inverse* (scientific) side of the problem, and use this probabilistic model as a means of fitting/analyzing a simulated data set.

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

   (b) Generate a contour plot (function `contour`, using 50 lines) of the negative log likelihood of the data set from part (c) of the previous problem, for all pairs of `mu` from `muall = [2:0.2:10]` and a `sigma` from `sigmaall = [0.5:0.2:6]`. What is the approximate location of the best fitting pair of parameters from this plot?

   (c) Use the `matlab` function `fminsearch` to get a more precise estimate of values `mu,sigma` that minimize the function `nloglik(mu,sigma,...)`. Two comments: first, the syntax for calling `nloglik` within `fminsearch` is a bit odd:
   `fminsearch(@(x) nloglik(x(1),x(2),I,T,C), <startpoint>)`.
   Here, the `@` notation is used to create a temporary function, with argument `x` a vector containing the two variables being optimized (mean and stdev). Second, you'll need to specify a start point for the search – for this problem, `[2,2]` is a reasonable choice.

   (d) A variant of `fminsearch, fminunc`, also returns the *Hessian* (the matrix of second derivatives) of the negative log likelihood at the optimal `mu` and `sigma`. (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 ± 1.96 standard deviations of the parameter estimate. Compute the standard deviation of a marginal of the 2-D Gaussian that has covariance equal to the inverse Hessian.) Do the true parameter values (4 and 1) fall within these confidence intervals?

   (e) Produce a second set of confidence intervals for the parameters using a bootstrap method. For each of the 7 intensities, resample 100 trials (correct or incorrect) 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` and `sigma` 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 (d)?

2. **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 (sometimes called the "prototype classifier"). Writek 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). Now compute the discriminant vector (compute the difference of the means of each data set, and normalize to unit length). Scatter plot 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?

   (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 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 *regularized* Fisher's discriminant, by estimating the covariance matrix as $\Sigma_{Estimated} = (1-\lambda)\Sigma_{Data} + \lambda I$, where Σ_{Data} is the mean covariance matrix estimated from the data (as in part (b)), I is the identity matrix. The parameter λ controls the regularization term, allowing the solution to transition between the prototype classifier ($\lambda = 1$) and Fisher's Discriminant ($\lambda = 0$). Test the classifier for $\lambda = [0 : 0.05 : 1]$ using 95%-5% cross-validation (i.e., sample *without replacement* and train on 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 λ and justify which λ 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?

3. **Revisiting old problems with new tools**.

   (a) From HW 2: you sequentially fit models of increasing polynomial order to a dataset and assessed what the 'best' model was by eye. Let's do that more objectively with regularization and cross-validation. Implement least-squares regression using the full polynomial model (up to 5th order) regularized by a term that penalizes large coefficients. For this penalty, try using both L2 (ridge) and L1 (LASSO) regularization terms, and determine the strength of the regularization using $95-5$ cross-validation. Are the models that you settled on via these regularizers different than the optimal models we found in HW2 and in the lab using cross-validation? Why? Explain in what ways these two regularization methods differ in terms of both their implementation and the resultant fits.

   (b) From HW 2: Professors Bell and Zell were given a mix of answers from Math Tools students informing them that their extracellular neural recordings arose from three *or* four neurons. Provide a more objective answer by implementing the (soft) K-means clustering algorithm. First, reduce the dimensionality of the data from 150 to 3 dimensions using PCA (using the SVD). Then, run the clustering algorithm with $K = 1, 2, \ldots 7$, repeating the estimation procedure from multiple random starting points for each K. For each K-iteration, compute the Euclidean distance of each point to its respective cluster centroid. Plot the *average* of point-to-centroid distances as a function of K. This plot should have a decreasing trend; however, increasing K yields diminishing returns in the reduction of point-to-centroid distances. Determine the number of clusters by choosing the K at which these distances stop decreasing substantially (This method is called the "elbow method"). For this hand-optimized value of K, plot a 3D scatter plot of your data with each point colored according to its assigned cluster. Using your new quantitative results, what would you now say to the professors about their data?

   (c) From HW 4: The research and development team at the international coffee conglomerate found your pilot experimental results too good to be true: They simply do not believe the scent of pumpkin spice evokes an increased response relative to control odorants in the amygdala, the structure associated with emotional responses in the brain. In order to validate your findings, the company performed the same experiment and collected 100 trials of their own, and now want you to classify their recorded responses (located in the file `newMeasurements.mat`). They ask you to train a prototype (nearest centroid) classifier on your pilot study data and use this trained model to classify the company's data as either control or pumpkin spice (they know the ground truth). Given what you know about the geometry of the data, what classifier would you propose the company use instead of the prototype classifier? This classifier likely has more free parameters than the prototype classifier, and so it wouldn't be surprising if it performs better; how can you justify the use of a more complicated classifier? Train this classifier on your pilot data and classify the company's data.