PSYCH-GA.2211/NEURL-GA.2201 – Fall 2021
Mathematical Tools for Neural and Cognitive Science

Homework 6

Due: 17 Dec 2021
(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. **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.

(d) Replot this nonlinearity with error bars, computed by bootstrapping. Draw a set of 6400 random integer indices in the range [1:6400], and use these to resample a set of projection/spike pairs, and recompute the nonlinearity for this bootstrap-resampled data. Unlike your previous estimates of the nonlinearity, you'll need to create fixed bins over which you'll average the projected values and spikes, which will allow you to compare across multiple bootstrap samples. Do this 100 times, to get 100 estimated nonlinearities. Plot the mean nonlinearity, and standard deviation, as points with error bars (use the matlab function `errorbar`).

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"). 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). 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?

(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?

(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 values $\lambda = [0 : 0.05 : 1]$ using 95%-5% cross-validation (i.e., 100 times, sample *without replacement* from 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 HW2: you sequentially fit models of increasing polynomial order to a dataset (from file `regress1.mat`) and assessed what the 'best' model was by eye and then by cross validation. Let's do that more objectively with regularization as well as 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 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 $K = 1, 2, \dots 7$, 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? Use one of the methods introduced in class to provide a justification. Then train this classifier on your pilot data and classify the company's data.