EE239AS.2, Spring 2017 **Homework #4**
Department of Electrical Engineering Prof. J.C. Kao
University of California, Los Angeles Special Reader M. Srinivasan

Due Wednesday, 17 May 2017, uploaded to Gradescope.
Covers material up to Discrete Classification II.
100 points total.

Please show your work and turn in all `MATLAB` code and plots. Please also **box your final answers for each question.**

A probabilistic generative model for classification comprises class-conditional densities $P(\mathbf{y} \mid \mathcal{C}_k)$ and class priors $P(\mathcal{C}_k)$, where $\mathbf{y} \in \mathbb{R}^D$ and $k = 1, \ldots, K$. We will consider three different generative models in this problem set:

i) Gaussian, shared covariance

$$\mathbf{y} \mid \mathcal{C}_k \sim \mathcal{N}(\boldsymbol{\mu}_k, \Sigma)$$

ii) Gaussian, class-specific covariance

$$\mathbf{y} \mid \mathcal{C}_k \sim \mathcal{N}(\boldsymbol{\mu}_k, \Sigma_k)$$

iii) Poisson

$$y_i \mid \mathcal{C}_k \sim \text{Poisson}(\lambda_{ki})$$

In iii), y_i is the ith element of the vector $\mathbf{y}$, where $i = 1, \ldots, D$. This is called a *naive Bayes* model, since the y_i are independent conditioned on $\mathcal{C}_k$.

1. (20 points) Maximum likelihood (ML) parameter estimation
In class, we derived the ML parameters for model i):

$$P(\mathcal{C}_k) = \frac{N_k}{N} \qquad \boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n \in \mathcal{C}_k} \mathbf{x}_n \qquad \Sigma = \sum_{k=1}^{K} \frac{N_k}{N} \cdot S_k,$$

where

$$S_k = \frac{1}{N_k} \sum_{n \in \mathcal{C}_k} (\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^{\mathrm{T}},$$

N_k is the number of data points in class $\mathcal{C}_k$, and N is the total number of data points in the data set.

   (a) (10 points) Find the ML parameters for model ii), i.e., find: $P(\mathcal{C}_k)$, $\boldsymbol{\mu}_k$, Σ_k.

   (b) (10 points) Find the ML parameters for model iii), i.e., find: $P(\mathcal{C}_k)$, λ_{ki}

2. (20 points) Decision boundaries
In class, we derived the decision boundary between class $\mathcal{C}_k$ and class $\mathcal{C}_j$ for model i):

$$(\mathbf{w}_k - \mathbf{w}_j)^{\mathrm{T}} \mathbf{x} + (w_{k0} - w_{j0}) = 0,$$

where

$$\mathbf{w}_k = \Sigma^{-1} \boldsymbol{\mu}_k \qquad w_{k0} = -\frac{1}{2} \boldsymbol{\mu}_k^{\mathrm{T}} \Sigma^{-1} \boldsymbol{\mu}_k + \log P(\mathcal{C}_k).$$

For each of the models ii) and iii), we'll want to derive the decision boundary between class $\mathcal{C}_k$ and class $\mathcal{C}_j$ and say whether it is linear in $\mathbf{y}$.

   (a) (10 points) What is the decision boundary for model (ii)? Is it linear?

   (b) (10 points) What is the decision boundary for model (iii)? Is it linear?

3. (30 points) Simulated data
We will now apply the results of Problems 1 and 2 to simulated data. The dataset can be found on CCLE as `ps4_simdata.mat`. The following describes the data format. The .mat file has a single variable named `trial`, which is a structure of dimensions (20 data points) $\times$ (3 classes). The `n`th data point for the `k`th class is a two-dimensional vector `trial(n,k).x`, where $\mathtt{n} = 1, \ldots, 20$ and $\mathtt{k} = 1, 2, 3$. To make the simulated data as realistic as possible, the data are non-negative integers, so one can think of them as spike counts. With this analogy, there are $D = 2$ neurons and $K = 3$ stimulus conditions.

   Please follow steps (a)–(e) below for *each* of the three models. The result of this problem should be three separate plots, one for each model. These plots will be similar in spirit to Figure 4.5 in *PRML*.

   (a) (4 points) Plot the data points in a two-dimensional space. For classes $\mathtt{k} = 1, 2, 3$, use a red $\times$, green $+$, and blue $\circ$ for each data point, respectively. Then, set the axis limits of the plot to be between 0 and 20. (`MATLAB` tip: call `axis([0 20 0 20])`)

(b) (7 points) Find the ML model parameters using results from Problem 1. Report the values of all the ML parameters for each model. (You can print out the parameters in MATLAB and copy and paste the MATLAB output.)

(c) (5 points) For each class, plot the ML mean on top of the data using a solid dot of the appropriate color. Set the marker size of this dot to be much larger than the marker sizes you used in part a, so the dot is easy to see.

(d) (7 points) For each class, plot the ML covariance using an ellipse of the appropriate color. Plot this on top of the data with the means. This part only needs to be donefor the Gaussian models i) and ii). Generate separate plots for models i) and ii). (Matlab tip: `contour` can be used to draw an iso-probability contour for each class. To aid interpretation, the contour should be drawn at the same probability level for each class. For this specific problem, please choose your contour level so you can see each ellipsoid reasonably. In our solutions, we drew the contour levels at probability 0.005, i.e., we called `contour(X, Y, Z, [0.007 0.007], 'r')`.)

(e) (7 points) Plot multi-class decision boundaries corresponding to the decision rule

$$\hat{k} = \underset{k}{\operatorname{argmax}}\, P(\mathcal{C}_k \mid \mathbf{x}) \tag{1}$$

and label each decision region with the appropriate class `k`. This should be plotted on top of your means and the covariance ellipsoids. Plot this by classifying a dense sampling of the two-dimensional data space. (Hint 1: You can do this by calling `[X,Y] = meshgrid(0:0.1:20, 0:0.1:20);` to partition the space, and then classifying each of these points. You should be able to compute this in a few lines of code, rather than iterating over each (x, y) pair, but we won't grade you on the efficiency of your script.) (Hint 2: You can check that you've done this properly by verifying that the decision boundaries pass through the intersection points of the contours drawn in part (d).)

4. (30 points) Real neural data
Neural prosthetic systems can be built based on classifying neural activity related to planning. As described in class, this is analogous to mapping patterns of neural activity to keys on a keyboard.

In this problem, we will apply the results of Problems 1 and 2 to real neural data. The neural data were recorded using a 100-electrode array in premotor cortex of a macaque monkey[1]. The dataset can be found on CCLE as `ps4_realdata.mat`.

The following describes the data format. The .mat file has two variables: `train_trial` contains the training data and `test_trial` contains the test data. Each variable is a structure of dimensions (91 trials) $\times$ (8 reaching angles). Each structure contains spike trains recorded simultaneously from 97 neurons while the monkey reached 91 times along each of 8 different reaching angles.

The spike train recorded from the `i`th neuron on the `n`th trial of the `k`th reaching angle is contained in `train_trial(n,k).spikes(i,:)`, where `n` $= 1, \ldots, 91$, `k` $= 1, \ldots, 8$, and `i` $= 1, \ldots, 97$. A spike train is represented as a sequence of zeros and ones, where time is discretized in 1 ms steps. A zero indicates that the neuron did not spike in the 1 ms

[1]The neural data have been generously provided by the laboratory of Prof. Krishna Shenoy at Stanford University. The data are to be used exclusively for educational purposes in this course.

bin, whereas a one indicates that the neuron spiked once in the 1 ms bin. The structure `test_trial` has the same format as `train_trial`.

These spike trains were recorded while the monkey performed a delayed reaching task, as described in class. Each spike train is 700 ms long (and is thus represented by a 1×700 vector), which comprises a 200 ms baseline period (before the reach target turned on) and a 500 ms planning period (after the reach target turned on). Because it takes time for information about the reach target to arrive in premotor cortex (due to the time required for action potentials to propagage and for visual processing), we will ignore the first 150 ms of the planning period. **For this problem, we will take spike counts for each neuron within a single 200 ms bin starting 150 ms after the reach target turns on.** In other words, we will only use `train_trial(n,k).spikes(i,351:550)` and `test_trial(n,k).spikes(i,351:550)` in this problem.

(a) (8 points) Fit the ML parameters of model i) to the training data (91×8 data points). Then, use these parameters to classify the test data (91×8 data points) according to the decision rule (1). What is the percent of test data points correctly classified?

(b) (6 points) Repeat part (a) for model ii). You should encounter a Matlab warning when classifying the test data. Why did the Matlab warning occur? What would we need to do to correct this error?

(c) (8 points) Correct the problem from part (b) by removing offending neurons. Now, what is the percent of test data points correctly classified? Is it higher or lower than your answer to part (a)? Why might this be?

(d) (8 points) Now we classify using a *naive Bayes* model. Repeat part (a) for model iii). Keep the convention in part (c), where offending neurons were removed from the analysis.