18-698 / 42-632
Neural Signal Processing
Prof. Byron Yu Spring 2013

Problem Set 3

This problem set is due in class on **Tuesday, March 5, 1:30pm**. Please show your work and turn in all Matlab code and plots.

If you have questions, please post them on the Discussion Board on the Blackboard course website, rather than emailing the course staff. This will allow other students with the same question to see the response and any ensuing discussion.

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

i) Gaussian, shared covariance

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

ii) Gaussian, class covariance

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

iii) Poisson

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

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

1. 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.

   For each of the models ii) and iii), find the ML parameters:

   ii) **(5 points)**$P(\mathcal{C}_k)$, $\boldsymbol{\mu}_k$, Σ_k

   iii) **(5 points)** $P(\mathcal{C}_k)$, λ_{ki}

2. **(10 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), derive the decision boundary between class $\mathcal{C}_k$ and class $\mathcal{C}_j$ and say whether it is linear.

3. Simulated data
   We will now apply the results of Problems 1 and 2 to simulated data. The dataset can be found on the Blackboard course website under "Course Content → Data sets → ps3_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) × (3 classes). The `n`th data point for the `k`th class is a two-dimensional vector `trial(n,k).x`, where `n` $= 1, \ldots, 20$ and `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) **(3 points)** Plot the data points in a two-dimensional space. For classes `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) **(3 points)** Find the ML model parameters using results from Problem 1.

(c) **(3 points)** For each class, plot the ML mean using a solid dot of the appropriate color.

(d) **(3 points)** For each class, plot the ML covariance using an ellipse of the appropriate color. This part only needs to be done for the Gaussian 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 example, call `contour(X, Y, Z, 0.007, 'r')`.)

(e) **(3 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 can be done either by plotting appropriate segments of the pairwise decision boundaries derived in Problem 2, or by classifying a dense sampling of the two-dimensional data space. (Hint: 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. Real neural data

Neural prosthetic systems can be built based on classifying neural activity related to movement 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 the Blackboard course website under "Course Content → Data sets → ps3_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` $= 1, \ldots, 8$,

[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.

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 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) **(5 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) **(3 points)** Repeat part (a) for model ii). You should encounter a Matlab warning when classifying the test data. Why did the Matlab warning occur?

(c) **(2 points)** The answer to part (b) is a motivation for using a *naive Bayes* model. Repeat part (a) for model iii). You should encounter a different Matlab warning when classifying the test data. Why did this Matlab warning occur?

(d) **(5 points)** To get around the warning in part (c), one might consider removing the offending neurons from the analysis. However, if a neuron is removed, it must be removed for all reaching angles (i.e., it cannot be removed for only the reaching angles at which it's problematic). Thus, we may need to remove a large number of neurons in total, which is undesirable.

An alternative is to set a minimum variance for each neuron. (Yes, this is a hack, but it works!) Although we might consider sweeping the minimum variance to optimize classification performance, here we will simply set the minimum variance to 0.01. Now, repeat part (a) for model iii) with this minimum variance.