ECE C143A/C243A, Spring 2022 **Homework #4 Solutions:**
Department of Electrical and Computer Engineering Prof. J.C. Kao
University of California, Los Angeles TAs T. Monsoor, W. Yu

Covers material up to Discrete Classification II.
100 points total.

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.

Solution: We make some notational simplifications.

- We let $\pi_k = P(\mathcal{C}_k)$. We use these two interchangeably.
- Because we prior defined y_i as the i^{th} element of $\mathbf{y}$, we'll label the i^{th} training example of $\mathbf{y}$ as $\mathbf{y}^i$.
- Similarly, the class of the i^{th} training example will be denoted $\mathcal{C}^i$.
- Thus, an observation of training data is given by the pair $(\mathbf{y}^i, \mathcal{C}^i)$. The set of all data is given by $\{\mathbf{y}^i, \mathcal{C}^i\}$.
- The i^{th} training example in class $\mathcal{C}_k$ can be written as $i : \mathcal{C}^i = \mathcal{C}_k$. However, for shorthand, we will usually write this as $i \in \mathcal{C}_k$.
- We let N_k be the number of training examples $i \in \mathcal{C}_k$.
- We let N be the total number of training examples.

With this, let's go ahead and write the data likelihood.

$$\begin{aligned} \mathcal{L} &= P(\{\mathbf{y}^i, \mathcal{C}^i\}) \\ &= \prod_k \prod_{i \in \mathcal{C}_k} P(\mathbf{y}^i, \mathcal{C}^i) \\ &= \prod_k \prod_{i \in \mathcal{C}_k} P(\mathcal{C}^i) P(\mathbf{y}^i | \mathcal{C}^i) \end{aligned}$$

To simplify calculations, we compute the log-likelihood.

$$\log \mathcal{L} = \sum_k \sum_{i \in \mathcal{C}_k} \left[\log P(\mathcal{C}^i) + \log P(\mathbf{y}^i | \mathcal{C}^i) \right]$$

Now, to calculate the class probabilities, we need the additional constraint that $\sum_k \pi_k = 1$. We need to take this into account via a Lagrange multiplier. With this, let's go ahead and calculate the ML parameters.

ML π_k:
We differentiate:

$$\begin{aligned}
\frac{\partial \left(\log \mathcal{L} + \lambda \sum_k \pi_k - 1\right)}{\partial \pi_k} &= \frac{\partial}{\partial \pi_k} \sum_k \left[\left(\sum_{i \in \mathcal{C}_k} \log P(\mathcal{C}^i) \right) - \lambda(\pi_k - 1) \right] \\
&= \frac{\partial}{\partial \pi_k} \sum_k \left[N_k \log P(\mathcal{C}_k) - \lambda(\pi_k - 1) \right] \\
&= \frac{\partial}{\partial \pi_k} \sum_k \left[N_k \log \pi_k - \lambda(\pi_k - 1) \right] \\
&= \frac{N_k}{\pi_k} - \lambda \\
[=] \quad & 0
\end{aligned}$$

Solving this equation gives $\pi_k = \frac{N_k}{\lambda}$. Thus, we also need to solve for λ. We do this by differentiating w.r.t. λ.

$$\begin{aligned}
\frac{\partial \left(\log \mathcal{L} + \lambda \sum_k (\pi_k - 1)\right)}{\partial \lambda} &= \frac{\partial}{\partial \lambda} \sum_k \left[-\lambda(\pi_k - 1) \right] \\
&= \sum_k -\pi_k + 1 \\
[=] \quad & 0
\end{aligned}$$

This returns to us our constraint equation, $\sum_k \pi_k = 1$. Hence, we have that

$$\begin{aligned}
\sum_k \pi_k &= \sum_k \frac{N_k}{\lambda} \\
&= 1
\end{aligned}$$

and thus, $\lambda = \sum_k N_k = N$.
This gives us the answer that

$$\boxed{\pi_k = \frac{N_k}{N}}$$

ML $\boldsymbol{\mu}_k$:

We differentiate:

$$\begin{aligned}
\frac{\partial \log \mathcal{L}}{\partial \boldsymbol{\mu}_k} &= \frac{\partial}{\partial \boldsymbol{\mu}_k} \sum_{i \in \mathcal{C}_k} -\frac{1}{2} (\mathbf{y}^i - \boldsymbol{\mu}_k)^T \Sigma_k^{-1} (\mathbf{y}^i - \boldsymbol{\mu}_k) \\
&= \frac{\partial}{\partial \boldsymbol{\mu}_k} \sum_{i \in \mathcal{C}_k} -\frac{1}{2} \left[(\mathbf{y}^i)^T \Sigma_k^{-1} \mathbf{y}^i - 2 \boldsymbol{\mu}_k^T \Sigma_k^{-1} \mathbf{y}^i + \boldsymbol{\mu}_k^T \Sigma_k^{-1} \boldsymbol{\mu}_k \right] \\
&= \sum_{i \in \mathcal{C}_k} \left[-\Sigma_k^{-1} \mathbf{y}^i + \Sigma_k^{-1} \boldsymbol{\mu}_k \right] \\
&= \sum_{i \in \mathcal{C}_k} \left[-\mathbf{y}^i \right] + N_k \boldsymbol{\mu}_k \\
[=] \quad & 0
\end{aligned}$$

This gives:

$$\boxed{\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{i \in \mathcal{C}_k} \mathbf{y}^i}$$

ML Σ_k:

We differentiate:

$$\begin{aligned} \frac{\partial \log \mathcal{L}}{\partial \Sigma_k} &= \frac{\partial}{\partial \Sigma_k} \sum_{i \in \mathcal{C}_k} \left[-\frac{1}{2} \mathrm{Tr}\left(\Sigma_k^{-1} (\mathbf{y}^i - \boldsymbol{\mu}_k)(\mathbf{y}^i - \boldsymbol{\mu}_k)^T \right) - \frac{1}{2} \log |\Sigma_k| \right] \\ &= \sum_{i \in \mathcal{C}_k} \left[\frac{1}{2} \Sigma_k^{-1} (\mathbf{y}^i - \boldsymbol{\mu}_k)(\mathbf{y}^i - \boldsymbol{\mu}_k)^T \Sigma_k^{-1} - \frac{1}{2} \Sigma_k^{-1} \right] \\ [=] \quad & 0 \end{aligned}$$

Solving this expression, we arrive at:

$$N_k \Sigma_k^{-1} = \sum_{i \in \mathcal{C}_k} \Sigma_k^{-1} (\mathbf{y}^i - \boldsymbol{\mu}_k)(\mathbf{y}^i - \boldsymbol{\mu}_k)^T \Sigma_k^{-1}$$

And therefore,

$$\boxed{\Sigma_k = \frac{1}{N_k} \sum_{i \in \mathcal{C}_k} (\mathbf{y}^i - \boldsymbol{\mu}_k)(\mathbf{y}^i - \boldsymbol{\mu}_k)^T}$$

These answers are all intuitively what we'd expect. The class means and covariances are the means and covariances of the data belonging to those classes. The class priors are the proportion of data we observe in each class.

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

Solution: The calculation for the ML π_k are exactly analogous to part (a), and thus,

$$\boxed{\pi_k = \frac{N_k}{N}}$$

To calculate the ML λ_{ki}, we recognize that

$$P(y_i | \mathcal{C}_k) = \frac{\lambda_{ki}^{y_i}}{y_i!} \exp(-\lambda_{ki})$$

Because the y_i's are independent conditioned on the class, we have that

$$P(\mathbf{y}^j | C^j) = \prod_i P(y_i^j | C^j)$$

Thus, the likelihood is

$$\mathcal{L} = \prod_k \prod_{j \in \mathcal{C}_k} \prod_i P(y_i^j | \mathcal{C}^j) P(\mathcal{C}^j)$$

and the log-likelihood is

$$\begin{aligned}
\log \mathcal{L} &= \sum_k \sum_{j \in \mathcal{C}_k} \sum_i \Big[\log P(y_i^j | \mathcal{C}^j) + \log P(\mathcal{C}^j) \Big] \\
&= \sum_k \sum_{j \in \mathcal{C}_k} \sum_i \left[\log \left(\frac{\lambda_{ki}^{y_i^j}}{y_i^j!} \exp(-\lambda_{ki}) \right) + \log P(\mathcal{C}^j) \right] \\
&= \sum_k \sum_{j \in \mathcal{C}_k} \sum_i \Big[y_i^j \log \lambda_{ki} - \lambda_{ki} - \log(y_i^j!) + \log P(\mathcal{C}^j) \Big]
\end{aligned}$$

We differentiate w.r.t. λ_{ki}, and set to 0, i.e.,

$$\begin{aligned}
\frac{\partial \log \mathcal{L}}{\partial \lambda_{ki}} &= \sum_{j \in C_k} y_i^j \left[\frac{1}{\lambda_{ki}} - 1 \right] \\
&= 0
\end{aligned}$$

This gives that

$$\boxed{\lambda_{ki} = \frac{1}{N_k} \sum_{j \in \mathcal{C}_k} y_i^j}$$

This answer makes intuitive sense: the rate parameter for neuron i when it is in class $\mathcal{C}_k$ is the empirical average of the firing rates of neuron i in class $\mathcal{C}_k$.

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?

Solution: The decision boundary is when the data likelihood for one class is equal to the data likelihood of another class. Let's call these two classes $\mathcal{C}_j$ and $\mathcal{C}_k$. Then we want to evaluate when $P(\mathbf{y}|\mathcal{C}_j) = P(\mathbf{y}|\mathcal{C}_k)$. This is equivalent to:

$$\pi_j P(\mathbf{y}|\mathcal{C}_j) = \pi_k P(\mathbf{y}|\mathcal{C}_k)$$

To simplify the calculation, we take the logs of both sides to arrive at:

$$\log \pi_j + \log P(\mathbf{y}|\mathcal{C}_j) = \log \pi_k + \log P(\mathbf{y}|\mathcal{C}_k)$$

We consider comparing the terms that are different between the j and k classes. Specifically, we compare:

$$-\frac{1}{2}\log|\Sigma_j| - \frac{1}{2}(\mathbf{y}-\boldsymbol{\mu}_j)^T\Sigma_j^{-1}(\mathbf{y}-\boldsymbol{\mu}_j) + \log \pi_j$$
$$\text{vs}$$
$$-\frac{1}{2}\log|\Sigma_k| - \frac{1}{2}(\mathbf{y}-\boldsymbol{\mu}_k)^T\Sigma_k^{-1}(\mathbf{y}-\boldsymbol{\mu}_k) + \log \pi_k$$

This simplifies to:

$$-\frac{1}{2}\log|\Sigma_j| - \frac{1}{2}\mathbf{y}^T\Sigma_j^{-1}\mathbf{y} + \boldsymbol{\mu}_j^T\Sigma_j^{-1}\mathbf{y} - \frac{1}{2}\boldsymbol{\mu}_j^T\Sigma_j^{-1}\boldsymbol{\mu}_j + \log \pi_j$$
$$\text{vs}$$
$$-\frac{1}{2}\log|\Sigma_k| - \frac{1}{2}\mathbf{y}^T\Sigma_k^{-1}\mathbf{y} + \boldsymbol{\mu}_k^T\Sigma_k^{-1}\mathbf{y} - \frac{1}{2}\boldsymbol{\mu}_k^T\Sigma_k^{-1}\boldsymbol{\mu}_k + \log \pi_k$$

Setting these terms equal, and consolidating terms, we see the decision boundary is:

$$\boxed{\mathbf{y}^T(\mathbf{W}_j - \mathbf{W}_k)\mathbf{y} + (\mathbf{w}_j - \mathbf{w}_k)^T\mathbf{y} + (w_{j0} - w_{k0}) = 0}$$

with

$$\begin{aligned}
\mathbf{W}_j &= -\frac{1}{2}\Sigma_j^{-1} \\
\mathbf{W}_k &= -\frac{1}{2}\Sigma_k^{-1} \\
\mathbf{w}_j &= \Sigma_j^{-1}\boldsymbol{\mu}_j \\
\mathbf{w}_k &= \Sigma_k^{-1}\boldsymbol{\mu}_k \\
w_{j0} &= -\frac{1}{2}\log|\Sigma_j| + \log P(\mathcal{C}_j) \\
w_{k0} &= -\frac{1}{2}\log|\Sigma_k| + \log P(\mathcal{C}_k)
\end{aligned}$$

It is clear from the form that

The decision boundary is not linear; in fact it is quadratic.

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

Solution: Like before, the decision boundary is

$$\log \pi_j + \log P(\mathbf{y}|\mathcal{C}_j) = \log \pi_k + \log P(\mathbf{y}|\mathcal{C}_k)$$

Here, assuming there are n neurons,

$$\begin{aligned}\log P(\mathbf{y}|C_j) &= \log \prod_{i=1}^{n} P(y_i|\mathcal{C}_j) \\ &= \sum_{i=1}^{n} \log P(y_i|\mathcal{C}_j) \\ &= \sum_{i=1}^{n} -\lambda_{ji} + y_i \log \lambda_{ji} - \log(y_i!)\end{aligned}$$

The decision boundary is thus,

$$\log \pi_j + \sum_{i=1}^{n} -\lambda_{ji} + y_i \log \lambda_{ji} - \log(y_i!) = \log \pi_k + \sum_{i=1}^{n} -\lambda_{ki} + y_i \log \lambda_{ki} - \log(y_i!)$$

Re-arranging terms, and defining λ_k as the vertical concatenation of the λ_{ki}'s, we have that the decision boundary is of the form:

$$\boxed{(\mathbf{w}_j - \mathbf{w}_k)^T \mathbf{y} + (w_{j0} - w_{k0}) = 0}$$

with

$$\begin{aligned}\mathbf{w}_j &= \log \lambda_j \\ \mathbf{w}_k &= \log \lambda_k \\ w_{j0} &= \log P(\mathcal{C}_j) - \sum_{i=1}^{n} \lambda_{ji} \\ w_{k0} &= \log P(\mathcal{C}_k) - \sum_{i=1}^{n} \lambda_{ki}\end{aligned}$$

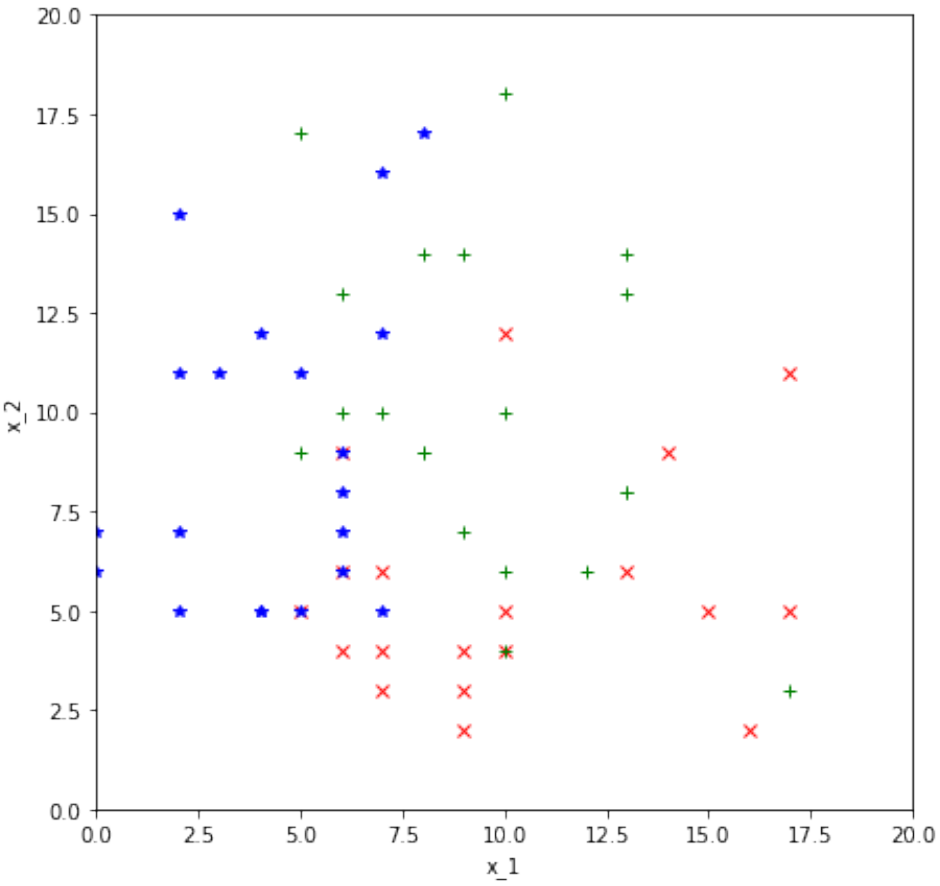
It is clear from the form that

$$\boxed{\text{The decision boundary is linear.}}$$

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 `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) 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.

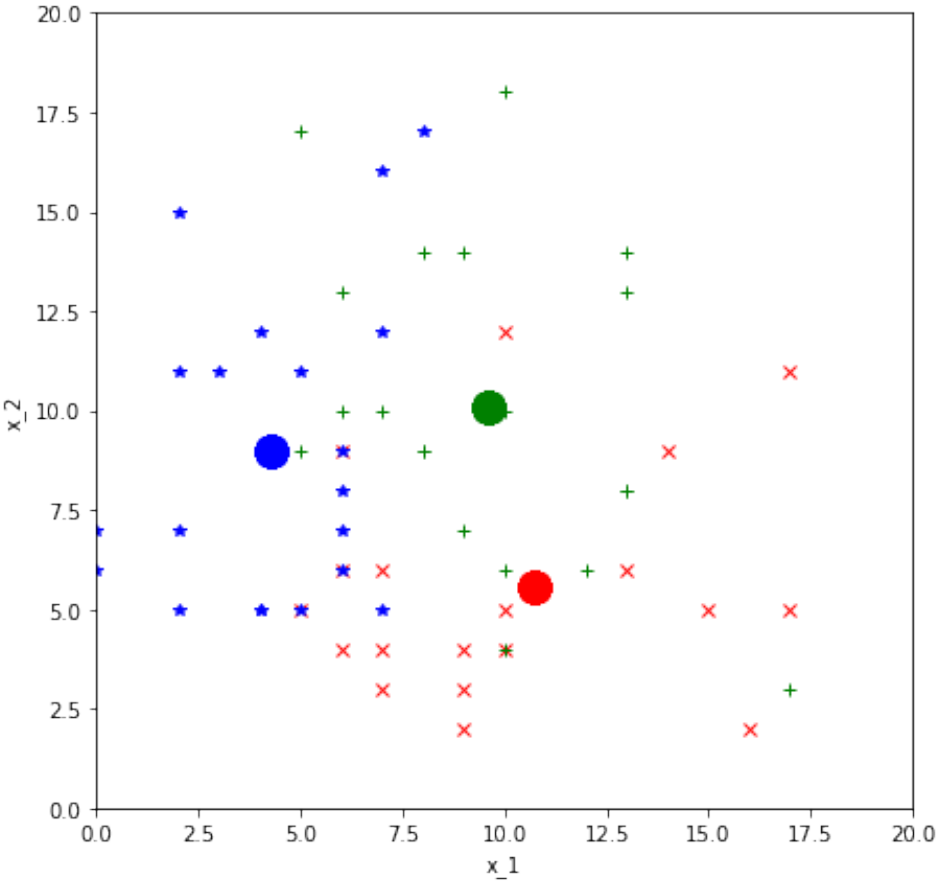


(b) (15 points) Find the ML model parameters using results from Problem 1. Report the values of all the ML parameters for each model. The printed parameters are:

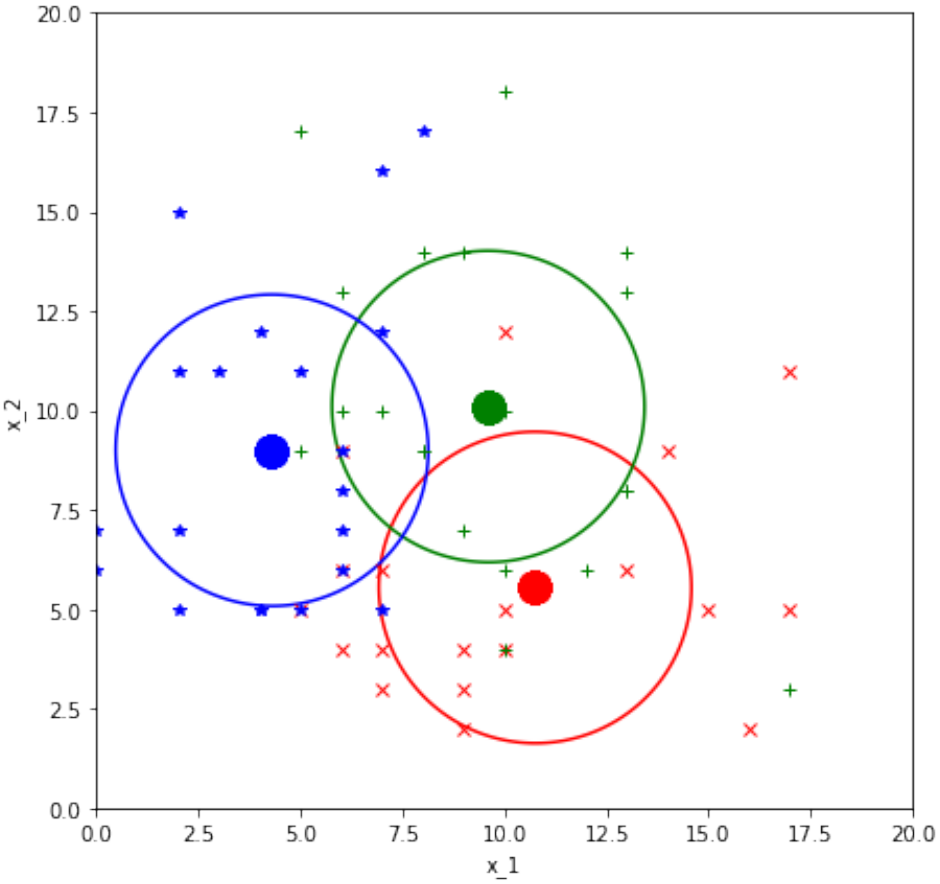
```
For Model 1...
Class priors:
    0.3333
    0.3333
    0.3333

Mean:
   10.7500    5.5500
    9.6000   10.1000
    4.3000    9.0000

Cov:
   12.1822   -0.0246
   -0.0246   12.7246

For Model 2...
Class priors:
    0.3333
    0.3333
```

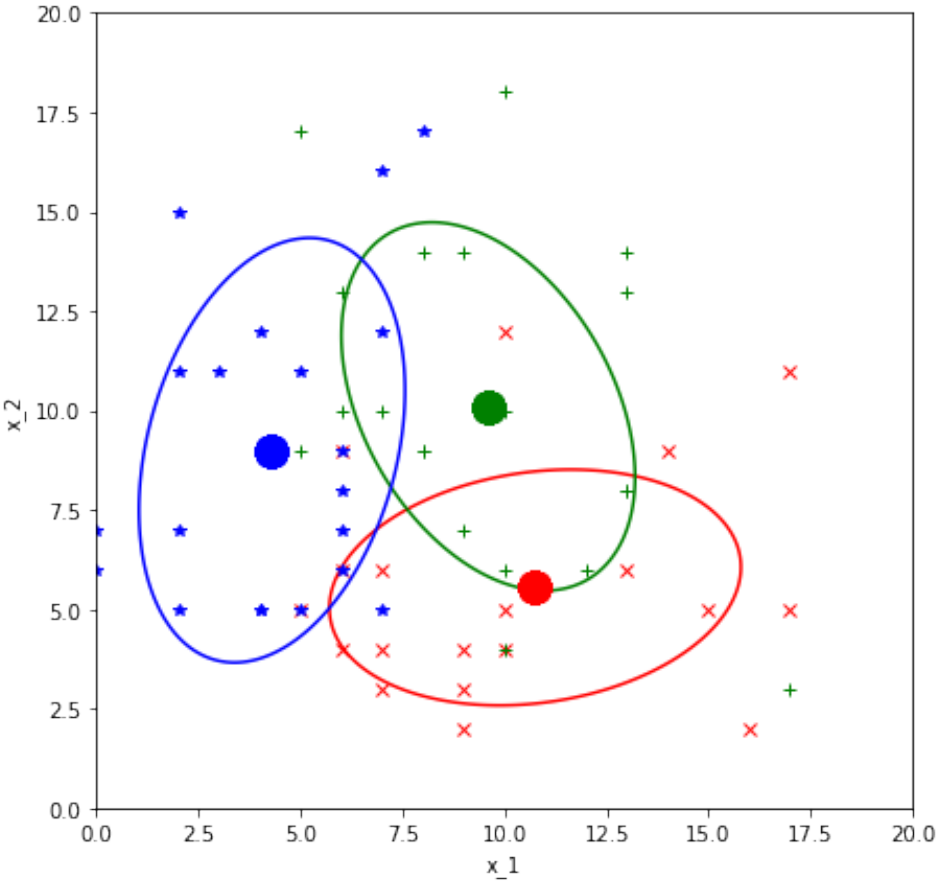
```
    0.3333

Mean:
   10.7500    5.5500
    9.6000   10.1000
    4.3000    9.0000

Cov class 1:
   22.0921    2.2500
    2.2500    7.6289

Cov class 2:
   10.0421   -4.9579
   -4.9579   16.6211

Cov class 3:
    5.6947    2.6316
    2.6316   15.2632

For Model 3...
Class priors:
    0.3333
    0.3333
    0.3333

Lambdas:
   10.7500    5.5500
    9.6000   10.1000
    4.3000    9.0000
```

(c) 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) 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).

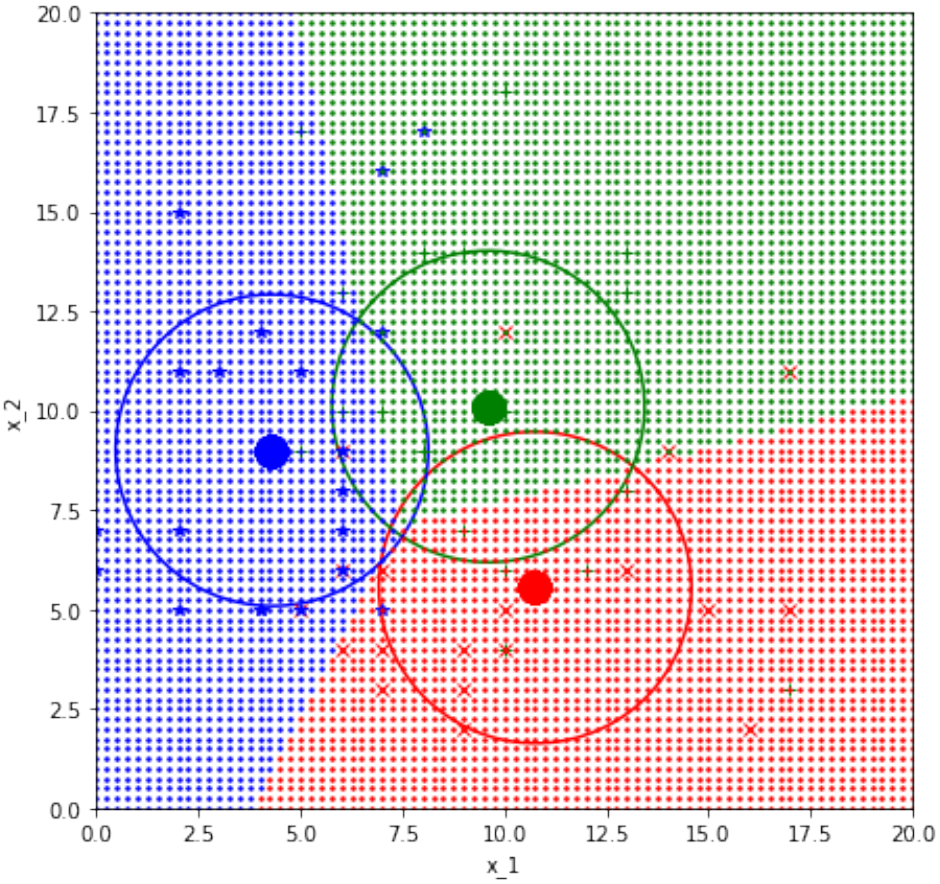

20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
x_2
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
x_1

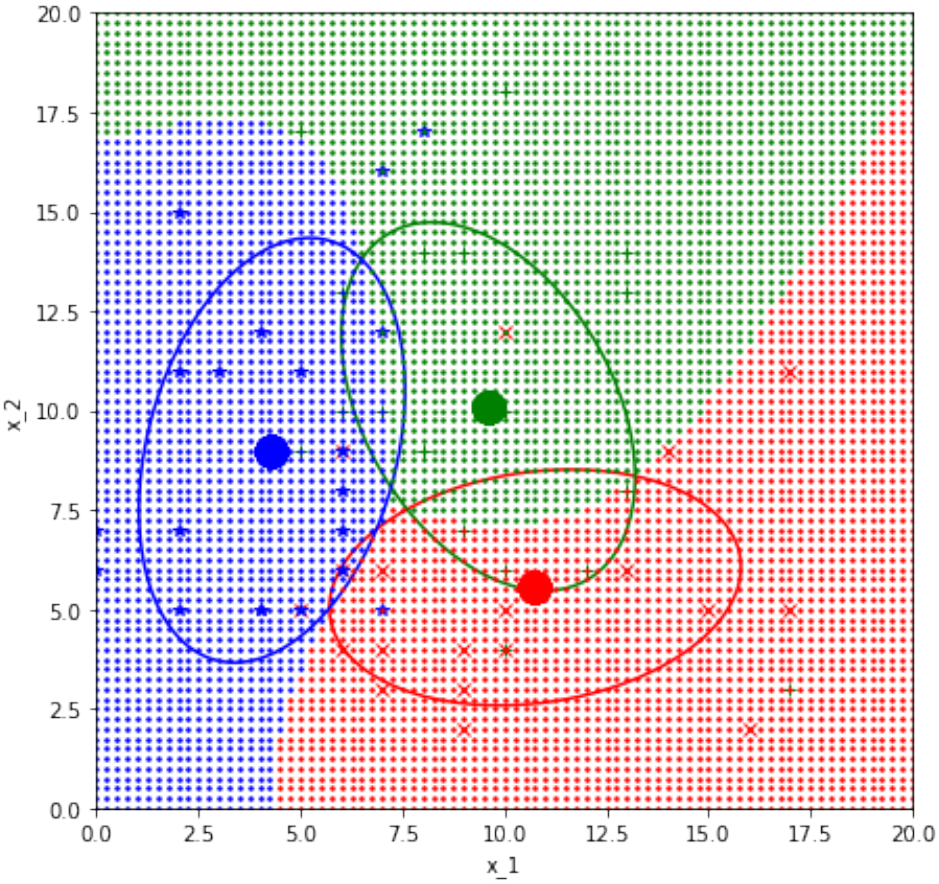


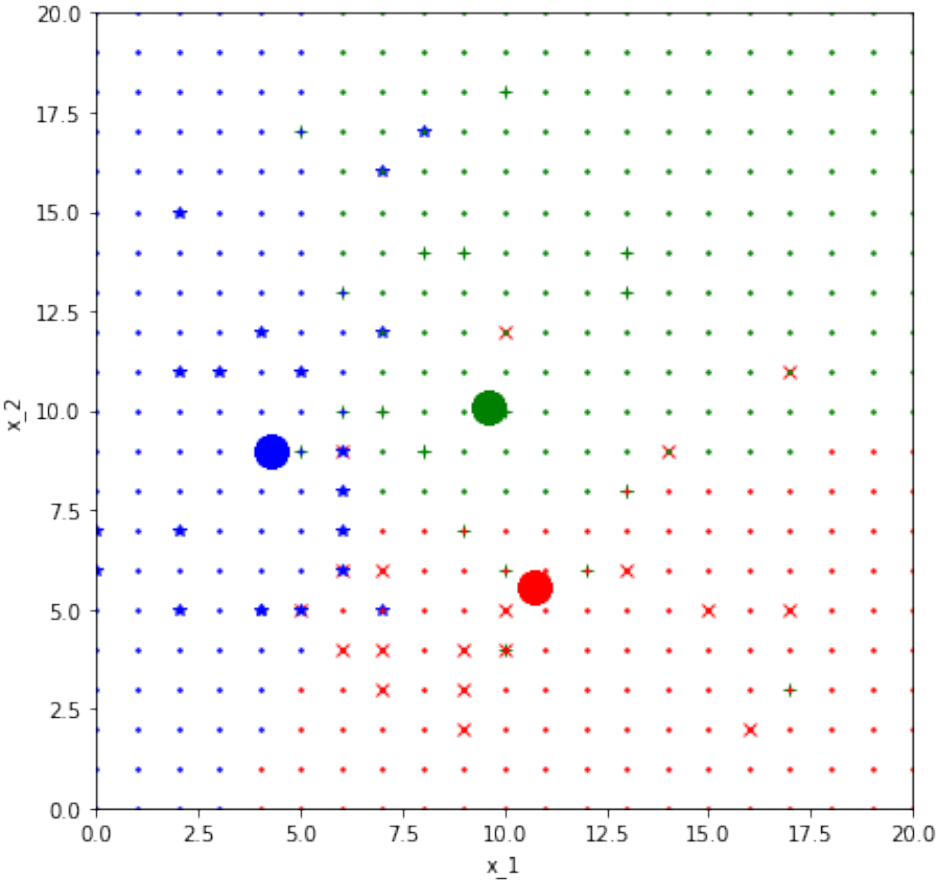
(e) (15 points) Plot multi-class decision boundaries corresponding to the decision rule

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

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.


20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
x_2
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
x_1


20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
x_2
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
x_1



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 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`.

[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.

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?

Solution: Approximately 96% of test data points are correctly classified.

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

Solution: There are neurons that have no spikes on some conditions, resulting in singular (non-invertible) covariance matrices.

(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?

Solution:

We remove the neurons without spikes on a certain condition (resulting in a covariance matrix with zero rows and columns). With this, our classification accuracy is approximately 44%. This isn't great, but above chance. Multiple problems could have led to this. For example, it could be that the covariance matrices for each condition aren't well estimated since we have less training data for each covariance matrix. It could be that the algorithm is weighting quiescent neurons (neurons that don't fire strongly) too heavily since these neurons have smaller variances, and thus their inverses are large. This could be addressed by different regularizations.

(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.

Solution: We get that the accuracy is 92%.