
Homework 8

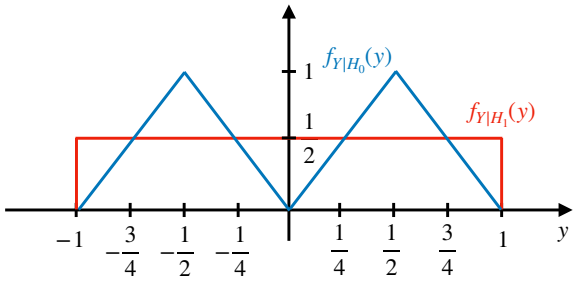
Due: 7:00pm, Friday, via Gradescope

Reading: Notes: Chapter 6, Chapter 7 **Videos:** 6.1 - 6.4; 7.1-7.2

Gradescope Uploading Instructions: As always, please be sure to assign problems to PDF pages when you are uploading to Gradescope. We will no longer grade submissions that fail to do so. Detailed instructions are available in the Homework folder on Blackboard.

Problem 8.1 (**Video 6.1, 6.2**, **Lecture Problem**)

Consider the following binary hypothesis testing scenario. (Note that all required integrals can be solved by calculating the areas of rectangles and triangles, so we are expecting exact answers.)



The hypothesis probabilities are $\mathbb{P}[H_0] = 2/3$ and $\mathbb{P}[H_1] = 1/3$.

(a) Determine the ML rule.

(b) Determine the MAP rule.

(c) Determine the probability of error under the ML rule.

(d) Determine the probability of error under the MAP rule.

Problem 8.2 (**Video 6.1, 6.2, 6.3**, **Quick Calculations**) For each of the scenarios below, determine the requested quantities.

(a) Under H_0, Y is Gaussian$(-1, 1)$. Under H_1, Y is Gaussian$(+1, 1)$. Let $\mathbb{P}[H_0] = 1/3$ and $\mathbb{P}[H_1] = 2/3$. Determine the ML and MAP decision rules.

(b) Under H_0, Y is Exponential(1). Under H_1, Y is Exponential(2). Let $\mathbb{P}[H_0] = 1/2$ and $\mathbb{P}[H_1] = 1/2$. Determine the likelihood ratio, the ML rule, and the probability of error under the ML rule.

(c) Under H_0, Y is Binomial$(4, 1/2)$. Under H_1, Y is Binomial$(3, 1/2)$. Let $\mathbb{P}[H_0] = 2/3$ and $\mathbb{P}[H_1] = 1/3$. Determine the probability of error under the ML and MAP decision rules.

Problem 8.3 (**Video 7.1, 7.2**, **Lecture Problem**)

Consider the following joint PDF

$$f_{X,Y}(x,y) = \begin{cases} \frac{4}{\pi} & x \geq 0,\ y \geq 0,\ x^2+y^2 \leq 1 \\ 0 & \text{otherwise.} \end{cases}$$

Note this is a uniform distribution over a quarter disk of radius 1.

(a) Determine the MMSE estimator $\hat{x}_{\text{MMSE}}(y)$ of X given $Y = y$.

(b) Determine the Mean Square Error of the MMSE estimator $\mathbb{E}\left[\left(X - \hat{x}_{\text{MMSE}}(Y)\right)^2\right]$. Please compute the integrals in order to get a numerical answer.

(c) Compute the LLSE estimate $\hat{x}_{\text{LLSE}}(y)$ of X given $Y = y$. Please compute the integrals to get numerical answers.

(d) Compute the Mean Square Error of the LLSE estimator. Please compute integrals to get numerical answers.

Problem 8.4

Let X, Y be joint Gaussian random variables, with zero mean, and $\mathsf{Var}[X] = \mathsf{Var}[Y] = 2$, $\mathsf{Cov}[X,Y] = 1$.

(a) Determine the MMSE estimator $\hat{x}_{\text{MMSE}}(y)$ of X given $Y = y$.

(b) Let U, V be independent continuous random variables with $U \sim$ Uniform(-1,1), $V \sim$Uniform(-1,1). Let $Z = U^3 + V$. compute linear least-squares estimate of Z based on observing $U = u$, denoted as $\hat{Z}_{\text{LLSE}}(u)$.

(c) Compute the Mean Square Error of both the MMSE estimate of Z given U and the LLSE estimate of Z given U.

Problem 8.5 (**Video 6.3, 6.4**, **Exploring Data**)

In this problem, we will combine ideas from Gaussian vectors and detection theory to come up with classifiers for high-dimensional datasets. The key assumption in detection theory is we know the distributions from which our observation vector $\underline{Y}$ is generated. Of course, this is not always the case in practice, but trying out the ML decision rule with a simple distribution is a good starting point.

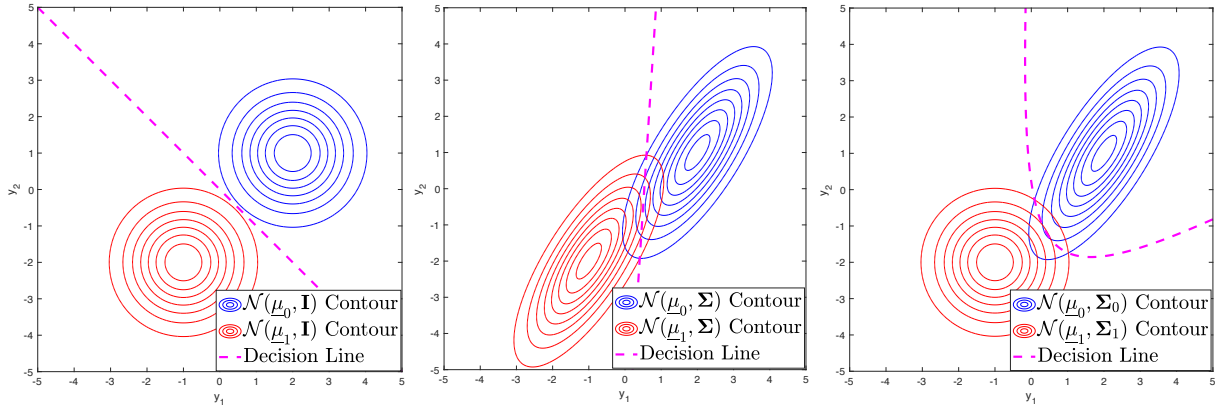
The shorthand notation $\underline{Y} \sim \mathcal{N}(\underline{\mu}, \mathbf{\Sigma})$ means that $\underline{Y}$ is a Gaussian vector with mean vector $\underline{\mu}$ and covariance matrix $\mathbf{\Sigma}$. Say that, under H_0, $Y \sim \mathcal{N}(\underline{\mu}_0, \mathbf{\Sigma}_0)$, and, under H_1, $Y \sim \mathcal{N}(\underline{\mu}_1, \mathbf{\Sigma}_1)$. Then, the maximum likelihood (ML) decision rule is $D(\underline{y}) = \begin{cases} 1 & f_{\underline{Y}|H_1}(\underline{y}) \geq f_{\underline{Y}|H_0}(\underline{y}) \\ 0 & f_{\underline{Y}|H_1}(\underline{y}) < f_{\underline{Y}|H_0}(\underline{y}) \end{cases}$ where

$$f_{\underline{Y}|H_1}(\underline{y}) = \frac{1}{\sqrt{\det(2\pi\mathbf{\Sigma}_1)}} e^{-\frac{1}{2}(\underline{y}-\underline{\mu}_1)^{\mathsf{T}}\mathbf{\Sigma}_1^{-1}(\underline{y}-\underline{\mu}_1)}$$

$$f_{\underline{Y}|H_0}(\underline{y}) = \frac{1}{\sqrt{(\det(2\pi\mathbf{\Sigma}_0)}} e^{-\frac{1}{2}(\underline{y}-\underline{\mu}_0)^T\mathbf{\Sigma}_0^{-1}(\underline{y}-\underline{\mu}_0)}$$

and d is the dimension of $\underline{Y}$. Let's look at a few special cases to understand the geometry of the Gaussian ML decision rule. See the plots below for illustrations.

- The covariance matrix is the identity matrix under both H_0 and H_1, $\mathbf{\Sigma}_0 = \mathbf{\Sigma}_1 = \mathbf{I}$. The decision is made by deciding which mean vector is closest, and the decision boundary is a line that is perpendicular to the line connecting $\underline{\mu}_0$ and $\underline{\mu}_1$.
- The covariance matrix is the same under both H_0 and H_1, $\mathbf{\Sigma}_0 = \mathbf{\Sigma}_1 = \mathbf{\Sigma}$. The decision boundary is still a line but now accounts for the tilt of the contour plot due to $\mathbf{\Sigma}$.
- The covariance matrix is the different under H_0 and H_1, $\mathbf{\Sigma}_0 \neq \mathbf{\Sigma}_1$. The decision boundary is no longer a line but is instead a curve.



Here is a simple procedure to implement these decision rules.

- Split the provided datasets into training data, which will be used to estimate the parameters of the distribution, and testing data, which will be used to evaluate the performance of the decision rule. (This is provided.)
- Estimate the mean vectors and covariance matrices under H_0 and H_1. (This is provided.)
- Evaluate the ML decision rule by calling the evaluating the PDFs under H_0 and H_1 for the test data provided, and selecting for each data point the decision that maximizes the likelihood. PDF evaluations can be done in MATLAB using the function `mvnpdf`, or in Python using `stats.multivariate_normal.pdf` appropriately. For example, `stats.multivariate_normal.pdf(x,mu,sigma)` returns the value of $f_{\underline{X}}(\underline{x})$ where $\underline{X} \sim \mathcal{N}(\underline{\mu}, \mathbf{\Sigma})$ and `mvnpdf(x,mu,sigma)` does the same in MATLAB.

We are providing you with a script in either Python or MATLAB, `hw8_python.ipynb` or `hw8_MATLAB.m`, that you will use in this problem, adding code where it is needed to perform the functions below. The script will read in the data set you choose, split the data into training data and test data for each class and then perform classification as indicated below. The script will also call a scoring function that is provided, which computes the probability of error as the fraction of test data that is classified incorrectly.

(a) Using the PDF functions discussed above, fill in the missing code in the hw8 script to run the Gaussian ML rule for the special case where the covariance matrix is the identity

matrix under both hypotheses. Specifically, under H_0, $Y \sim \mathcal{N}(\underline{\mu}_0, \mathbf{I})$ and, under H_1, $Y \sim \mathcal{N}(\underline{\mu}_1, \mathbf{I})$. Run the resulting code on the provided synthetic dataset `syntheticH0.csv` and `syntheticH1.csv` and write down the resulting probability of error.

(b) Using the PDF functions above, fill in the missing code in the script to run the Gaussian ML rule for the special case where the covariance matrix is equal under both hypotheses. Specifically, under H_0, $Y \sim \mathcal{N}(\underline{\mu}_0, \mathbf{\Sigma})$ and, under H_1, $Y \sim \mathcal{N}(\underline{\mu}_1, \mathbf{\Sigma})$. This requires you to estimate $\mathbf{\Sigma}$.

To estimate $\mathbf{\Sigma}$, we use a statistical technique known as pooling:

$$\hat{\mathbf{\Sigma}} = \frac{1}{n_{\text{train},0} + n_{\text{train},1} - 2}\left((n_{\text{train},0} - 1)\hat{\mathbf{\Sigma}}_0 + (n_{\text{train},1} - 1)\hat{\mathbf{\Sigma}}_1\right)$$

In the above equation, $\hat{\mathbf{\Sigma}}_0$ is the estimated covariance for the training data under hypothesis H_0, and $\hat{\mathbf{\Sigma}}_1$ is the estimated covariance for the training data under hypothesis H_1. Take care to only calculate the necessary matrices once, rather than every time inside a for loop (which will increase your run time to hours or days). Run the resulting code on the provided synthetic dataset `syntheticH0.csv` and `syntheticH1.csv` and write down the resulting probability of error.

(c) Using the PDF functions above, fill in the missing code in the script to run the Gaussian ML rule for the case where the covariance matrices are different. Specifically, under H_0, $Y \sim \mathcal{N}(\underline{\mu}_0, \mathbf{\Sigma}_0)$ and, under H_1, $Y \sim \mathcal{N}(\underline{\mu}_1, \mathbf{\Sigma}_1)$. Run the resulting code on the provided synthetic dataset `syntheticH0.csv` and `syntheticH1.csv` and write down the resulting probability of error.

(d) We will now repeat the process for a different data set. Specifically, we will use the cancer data sets `benignfull.csv` and `malignantfull.csv` from the previous homework, where `benignfull.csv` is the data under hypothesis H_0 (no cancer), and `malignantfull.csv` is the data under hypothesis H_1 (cancer). Run the completed above functions in the script for these data sets for the case of identity covariance. Write down the resulting probabilities of error.

Do you notice anything odd about the probability of error for the special case with identity covariances? You should have a value that is very close to 0.5, which is no better than guessing. This is happening because the values of $f_{\underline{Y}|H_0}(\underline{y})$ and $f_{\underline{Y}|H_1}(\underline{y})$ are so small ((10 dimensional data, exponential of negative quadratic) that they often go and get rounded to 0. Thus, the ML rule always picks H_1 by default due to the tie.

We can get around this issue by using the log-likelihood ratio

$$\log(L(\underline{y})) = \log\left(\frac{f_{\underline{Y}|H_1}(\underline{y}))}{f_{\underline{Y}|H_0}(\underline{y}))}\right) = \log(f_{\underline{Y}|H_1}(\underline{y})) - \log(f_{\underline{Y}|H_0}(\underline{y}))$$

Recall that the ML rule for the log-likelihood ratio is $D(\underline{y}) = \begin{cases} 1 & \log(L(\underline{y})) \geq 0, \\ 0 & \log(L(\underline{y})) < 0. \end{cases}$

It turns out that, for the special case where covariance matrix is the identity matrix under both hypotheses, this is equivalent to a "closest average":

$$D_{\text{avg}}(\underline{y}) = \begin{cases} 1, & \text{if } \|\underline{y} - \underline{\mu}_1\| < \|\underline{y} - \underline{\mu}_0\|, \\ 0, & \text{if } \|\underline{y} - \underline{\mu}_1\| \geq \|\underline{y} - \underline{\mu}_0\|. \end{cases}$$

Argue why this is the case mathematically. It may be helpful to recall that $\|\underline{a}\|^2 = \underline{a}^\mathsf{T}\underline{a}$.

(e) Continue filling out the classifier code in the script to complete the log-likelihood classifiers. For this part, call the provided function closestaverage to find which average is closest to a given data point. Run it on the breast cancer data set, and write down the resulting probability of error.

(f) For the second part, use the log-likelihood ratio to come up with an ML rule for the special case where the covariance matrix is equal under both hypotheses. Specifically, under H_0, $Y \sim \mathcal{N}(\underline{\mu}_0, \mathbf{\Sigma})$ and, under H_1, $Y \sim \mathcal{N}(\underline{\mu}_1, \mathbf{\Sigma})$, where Σ is the pooled covariance discussed in part b). Derive the log-likelihood rule for this case, and implement it in the script. When computing Σ^{-1}, be sure to use to compute it once (outside of any loop for classifying data) and use the matrix pseudoinverse (Python `np.linalg.pinv` or MATLAB lstinlinepinv) rather than the matrix inverse, as you are not guaranteed that the sample covariance matrices are invertible. Run the script on the breast cancer data set, and write down the resulting probability of error.

(g) Use the log-likelihood ratio to come up with an ML rule for the case where the covariance matrices are different. Complete the missing piece of the provided script. When computing the inverses of the covariances $\Sigma_0^{-1}, \Sigma_1^{-1}$, be sure to use the matrix pseudoinverse rather than the matrix inverse, as you are not guaranteed that the sample covariance matrices are invertible. Also, invert the matrices outside the loop so your code runs efficiently. Run this on the breast cancer data set, and write down the resulting probability of error.

(h) For this part, submit a printout of your code as part of your scanned submission.

To Submit: Answers to each of the parts. In particular, add a mathematical argument for part (d) that shows that the log likelihood ratio for the identity covariance case reduces to the closest average, in the answer to part (d).