# Homework 3

Spring 2021
(Due: Friday, Mar 5, 2021, 11:59 pm Eastern Time)

Please submit your homework through **gradescope**. You can write, scan, type, etc. But for the convenience of grading, please merge everything into a **single PDF**.

## Objective

There are three things you will learn in this homework:

(a) Understanding why the maximum-likelihood estimate of the covariance matrix is the sample covariance.

(b) Implement a Bayesian decision rule for an image segmentation problem.

(c) Analyze the ROC curve.

You will be asked some of these questions in Quiz 3. The Quiz will be open on Mar 6, 8am Eastern Time, and close on Mar 7, 8am Eastern Time. The Quiz is 30 minutes long.

## Exercise 1

Suppose that we are given a dataset $\mathcal{D} \overset{\text{def}}{=} \{\boldsymbol{x}_n\}_{n=1}^N$, where each sample $\boldsymbol{x}_n \in \mathbb{R}^d$ is an iid copy of the random variable $\boldsymbol{X}$. For simplicity, we assume that the distribution of $\boldsymbol{X}$ is a multi-dimensional Gaussian of mean $\boldsymbol{\mu} \in \mathbb{R}^d$ and covariance $\boldsymbol{\Sigma} \in \mathbb{R}^{d\times d}$. We further assume that the mean vector $\boldsymbol{\mu}$ is known and is given. Therefore, the likelihood of observing a sample $\boldsymbol{x}_n$ is fully controlled by the covariance matrix, i.e.,

$$p(\boldsymbol{x}_n \,|\, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\Big\{ -\frac{1}{2}(\boldsymbol{x}_n - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_n - \boldsymbol{\mu})\Big\} \tag{1}$$

Taking into consideration of all the samples in the dataset $\mathcal{D}$, the likelihood of $\mathcal{D}$ is

$$p(\mathcal{D} \,|\, \boldsymbol{\Sigma}) = \prod_{n=1}^N \left\{ \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}} \exp\Big\{ -\frac{1}{2}(\boldsymbol{x}_n - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}_n - \boldsymbol{\mu})\Big\} \right\}. \tag{2}$$

The goal of this analytical exercise is to derive the maximum-likelihood estimate of $\boldsymbol{\Sigma}$:

$$\widehat{\boldsymbol{\Sigma}}_{\text{ML}} = \underset{\boldsymbol{\Sigma}}{\text{argmax}} \quad p(\mathcal{D} \,|\, \boldsymbol{\Sigma}). \tag{3}$$

To make things simpler we assume that $\boldsymbol{\Sigma}$ and $\widetilde{\boldsymbol{\Sigma}} = \frac{1}{N}\sum_{n=1}^N (\boldsymbol{x}_n - \boldsymbol{\mu})(\boldsymbol{x}_n - \boldsymbol{\mu})^T$ are invertible in this exercise.

(a) Recall that the trace operator is defined as $\text{tr}[\boldsymbol{A}] = \sum_{i=1}^d [\boldsymbol{A}]_{i,i}$. Prove the matrix identity

$$\boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x} = \text{tr}[\boldsymbol{A}\boldsymbol{x}\boldsymbol{x}^T], \tag{4}$$

where $\boldsymbol{A} \in \mathbb{R}^{d\times d}$.

(b) Show that the likelihood function in (3) can be written as:

$$p(\mathcal{D}|\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{Nd/2}} |\boldsymbol{\Sigma}^{-1}|^{N/2} \exp\left\{ -\frac{1}{2}\text{tr}\left[ \boldsymbol{\Sigma}^{-1} \sum_{n=1}^N (\boldsymbol{x}_n - \boldsymbol{\mu})(\boldsymbol{x}_n - \boldsymbol{\mu})^T \right]\right\}. \tag{5}$$

© 2021 Stanley Chan. All Rights Reserved. 

(c) Let $\widetilde{\boldsymbol{\Sigma}} = \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{x}_n - \boldsymbol{\mu})(\boldsymbol{x}_n - \boldsymbol{\mu})^T$, and let $\boldsymbol{A} = \boldsymbol{\Sigma}^{-1}\widetilde{\boldsymbol{\Sigma}}$, and $\lambda_1, ..., \lambda_d$ be the eigenvalues of $\boldsymbol{A}$. Show that the result from the previous part leads to:

$$p(\mathcal{D}|\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{Nd/2}|\widetilde{\boldsymbol{\Sigma}}|^{N/2}}\left(\prod_{i=1}^{d}\lambda_i\right)^{N/2}\exp\left\{-\frac{N}{2}\sum_{i=1}^{d}\lambda_i\right\} \tag{6}$$

**Hint**: For matrix $\boldsymbol{A}$ with eigenvalues $\lambda_1, ..., \lambda_d$, $\text{tr}[\boldsymbol{A}] = \sum_{i=1}^{d}\lambda_i$.

(d) Find $\lambda_1, \ldots, \lambda_d$ such that (6) is maximized.

(e) With the choice of $\lambda_i$ given in (d), prove that the ML estimate $\widehat{\boldsymbol{\Sigma}}_{\text{ML}}$ is

$$\widehat{\boldsymbol{\Sigma}}_{\text{ML}} = \frac{1}{N}\sum_{n=1}^{N}(\boldsymbol{x}_n - \boldsymbol{\mu})(\boldsymbol{x}_n - \boldsymbol{\mu})^T. \tag{7}$$

(f) What would be the alternative way of finding $\widehat{\boldsymbol{\Sigma}}_{\text{ML}}$? You do not need to prove. Just briefly mention the idea.

(g) If $\boldsymbol{\mu}$ is also estimated from the data so that it is $\widehat{\boldsymbol{\mu}} = \frac{1}{N}\sum_{n=1}^{N}\boldsymbol{x}_n$, the ML estimate $\widehat{\boldsymbol{\Sigma}}_{\text{ML}} = (1/N)\sum_{n=1}^{N}(\boldsymbol{x}_n - \widehat{\boldsymbol{\mu}})(\boldsymbol{x}_n - \widehat{\boldsymbol{\mu}})^T$ will a *biased* estimate of the covariance matrix because $\mathbb{E}[\widehat{\boldsymbol{\Sigma}}_{\text{ML}}] \neq \boldsymbol{\Sigma}$. Can you suggest an unbiased estimate $\widehat{\boldsymbol{\Sigma}}_{\text{unbias}}$ such that $\mathbb{E}[\widehat{\boldsymbol{\Sigma}}_{\text{unbias}}] = \boldsymbol{\Sigma}$? No need to prove. Just state the result.

# Exercise 2

In this exercise I want you to implement a Bayesian decision rule for a (super classical) problem of image segmentation. The image we work with consists of a cat and some grass [1]. The size of this image is $500 \times 375$ pixels. The left hand side of Figure 1 shows the image, and the right hand side of Figure 1 shows a manually labeled "ground truth". Your task is to do as much as you can to extract the cat from the grass, and compare your result with the "ground truth".

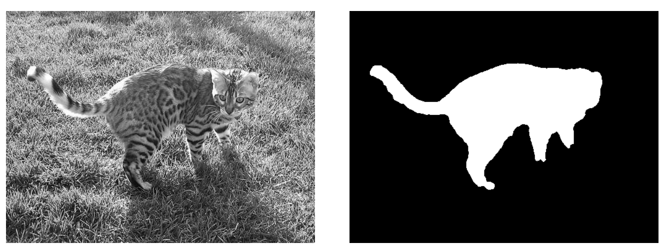

Figure 1: The "Cat and Grass" image.

## Preparation Steps (No need to hand in)

First of all, go to the course website and download the data. Write the Python script to read the data and convert it into a data matrix.

```
train_cat   = np.matrix(np.loadtxt('train_cat.txt',   delimiter = ','))
train_grass = np.matrix(np.loadtxt('train_grass.txt', delimiter = ','))
```

The data matrices are $64 \times K_1$ and $64 \times K_0$, respectively, where $K_1$ is the number of training samples for Class 1 (cat), and $K_0$ is the number of training samples for Class 0 (grass).

Throughout this exercise, you need to read images and extract patches. To read an image, you can call `cv2` library or you can call `plt.imread`. For example, you can do

[1] Image Source: `http://www.robots.ox.ac.uk/vgg/data/pets/`

© 2021 Stanley Chan. All Rights Reserved.

```
Y = plt.imread('cat_grass.jpg') / 255
```

The decision making of this problem is performed for every pixel. Therefore, you need to write a `for` loop to loop through all the pixels of the image. Moreover, you need to extract $8 \times 8$ neighbors surrounding each pixel. These can be done using the following commands:

```
M,N = Y.shape
for i in range(M-8):
    for j in range(N-8):
        block = Y[i:i+8, j:j+8] # This is a 8x8 block
        #
        # Something
        #
```

To make your life easier, it is okay to set the running index `i in range(M-8)` by neglecting the boundary pixels. In this case, the ground truth mask will have 8 rows and 8 columns less.

## Your Tasks (Please hand in)

The Bayesian decision rule we are going to implement is based on the posterior distribution. We define the likelihood functions:

$$p_{\boldsymbol{X}|Y}(\boldsymbol{x}|C_1) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}_1|^{1/2}} \exp\Big\{ -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_1)^T\boldsymbol{\Sigma}_1^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1)\Big\},$$

$$p_{\boldsymbol{X}|Y}(\boldsymbol{x}|C_0) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}_0|^{1/2}} \exp\Big\{ -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_0)^T\boldsymbol{\Sigma}_0^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_0)\Big\}, \tag{8}$$

and also the prior distributions $p_Y(C_1) = \pi_1$ and $p_Y(C_0) = \pi_0$. For simplicity, we assume that $\pi_1 = \frac{K_1}{K_1+K_0}$ and $\pi_0 = \frac{K_0}{K_1+K_0}$. The Bayesian decision rule says that

$$p_{Y|\boldsymbol{X}}(C_1|\boldsymbol{x}) \gtrless_{C_0}^{C_1} p_{Y|\boldsymbol{X}}(C_0|\boldsymbol{x}), \tag{9}$$

which is based on the **posterior** distribution.

(a) Substitute the multi-dimensional Gaussian likelihood (8) and the priors $\pi_1$ and $\pi_0$ into (9). Show that the decision rule is equivalent to

$$-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_1)^T\boldsymbol{\Sigma}_1^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_1) + \log\pi_1 - \frac{1}{2}\log|\boldsymbol{\Sigma}_1| \gtrless_{C_0}^{C_1} -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_0)^T\boldsymbol{\Sigma}_0^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_0) + \log\pi_0 - \frac{1}{2}\log|\boldsymbol{\Sigma}_0|.$$

(b) Estimate $\boldsymbol{\mu}_1$, $\boldsymbol{\mu}_0$, $\boldsymbol{\Sigma}_1$, $\boldsymbol{\Sigma}_0$, $\pi_1$ and $\pi_0$ in Python. Report:

(i) The first 2 entries of the vector $\boldsymbol{\mu}_1$ and the first 2 entries of the vector $\boldsymbol{\mu}_0$.

(ii) The first $2 \times 2$ entries of the matrix $\boldsymbol{\Sigma}_1$ and the first $2 \times 2$ entries of the matrix $\boldsymbol{\Sigma}_0$.

(iii) The values of $\pi_1$ and $\pi_0$.

(c) Write a double for loop to loop through the pixels of the testing image. At each pixel location, consider a $8 \times 8$ neighborhood. This will be the testing vector $\boldsymbol{x} \in \mathbb{R}^{64}$. Dump this testing vector $\boldsymbol{x}$ into the decision rule you proved in (a), and determine whether the testing vector belongs to Class 1 or Class 0. Repeat this for other pixel locations.

```
for i in range(M-8):
    for j in range(N-8):
        block = Y[i:i+8, j:j+8]
        #
```

© 2021 Stanley Chan. All Rights Reserved. 

```
        # Something
        #
        prediction[i,j] = # Something
```

If you do everything right, you will get a binary image. Submit this predicted binary image. Remark: My program runs for about 10-15 seconds. If your code takes forever to run, something is wrong.

(d) Consider the ground truth image `truth.png`. Report the mean absolute error (MAE) between your prediction and the ground truth:

$$\text{MAE} = \frac{1}{\#\text{ of pixels}} \sum_{i,j} \left| \texttt{prediction}[i,j] - \texttt{truth}[i,j] \right|.$$

Report your MAE. Remark: Because we are not dealing with the boundary pixels (which explains why I set `i in range(M-8)`), when computing the MAE you need to set the true mask to `truth[0:M-8, 0:N-8]`.

(e) Go to the internet and download an image with similar content: an animal on grass or something like that. Apply your classifier to the image, and submit your resulting mask. You probably do not have the ground truth mask, so please just show the predicted mask. Does it perform well? If not, what could go wrong? Write one to two bullet points to explain your findings. Please be brief.

## Exercise 3

The objective of this exercise is to plot the ROC curve. You may want to read Chapter 9.4 and Chapter 9.5 of my book.

(a) The Bayesian decision rule you derived in Exercise 2 is actually equivalent to the likelihood ratio test:

$$\frac{p_{\boldsymbol{X}|Y}(\boldsymbol{x}|C_1)}{p_{\boldsymbol{X}|Y}(\boldsymbol{x}|C_0)} \gtrless_{C_0}^{C_1} \tau, \tag{10}$$

for some threshold constant $\tau$. Determine $\tau$ that corresponds to the decision rule in Exercise 2.

(b) Implement this likelihood ratio test rule for different values of $\tau$. For every $\tau$, compute the number of true positives and the number of false positives. Then, we can define the probability of detection $p_D(\tau)$ and the probability of miss $p_F(\tau)$ as:

$$p_D(\tau) = \frac{\#\text{ true positives}}{\text{total }\#\text{ of positives in ground truth}}$$
$$p_F(\tau) = \frac{\#\text{ false positives}}{\text{total }\#\text{ of negatives in ground truth}}.$$

Plot the ROC curve. That is, plot $p_D(\tau)$ as a function of $p_F(\tau)$. Your ROC curve should cover the range $[0,1] \times [0,1]$. Remark: Generating this ROC curve will take a minute or two.

(c) On your ROC curve, mark a red dot to indicate the operating point of the Bayesian decision rule.

(d) Implement a linear regression classifier for this problem, and plot the ROC curve. The idea is to construct a matrix system:

$$\underbrace{\begin{bmatrix} \boldsymbol{X}_1 \\ \boldsymbol{X}_0 \end{bmatrix}}_{=\boldsymbol{A}} \boldsymbol{\theta} = \underbrace{\begin{bmatrix} \boldsymbol{1} \\ -\boldsymbol{1} \end{bmatrix}}_{=\boldsymbol{b}},$$

where $\boldsymbol{X}_1 \in \mathbb{R}^{K_1 \times d}$ and $\boldsymbol{X}_0 \in \mathbb{R}^{K_0 \times d}$ are the training data matrix of Class 1 and Class 0. Solve the regression problem

$$\widehat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta} \in \mathbb{R}^d}{\text{argmin}} \, \|\boldsymbol{A}\boldsymbol{\theta} - \boldsymbol{b}\|^2.$$

During testing, write a double for loop to through all the $8 \times 8$ neighbors of the image pixels. The decision rule per neighbor is

$$\widehat{\boldsymbol{\theta}}^T \boldsymbol{x} \quad \gtrless_{C_0}^{C_1} \quad \tau, \tag{11}$$

where $\widehat{\boldsymbol{\theta}}$ is the trained model parameter, and $\boldsymbol{x}$ is the testing $8 \times 8$ neighbor. By varying the threshold $\tau$, you can obtain another ROC curve. Plot it.

## Exercise 4: Project Check Point

This is check point # 3. As we go to this check point, you should have become familiar with your project. You should have known the basic literature, and you should have played some existing codes. The goal of this check point is to jump into the core problem by making some small modifications to the existing code. Because everyone has a slightly different project goal, I can only offer some very generic advices:

- If your proposed theme requires training on a different dataset, it is the time to program it and try training your model using the new dataset (and new configurations).
- If you want to compare several baseline methods, it is the time to setup an infrastructure and define the evaluation benchmark. Make sure that the comparisons you are making are fair to all competing methods.
- If you want to attack classifier, you should have an attacker ready. If you want to defend a classifier, you need to start thinking about how to do adversarial training.
- If you want to re-implement a label noise algorithm, start with the simplest toy problem (e.g., MNIST or even smaller dataset) and consider the simplest model (few layers of networks). Implement something.

You will certainly hit a lot of obstacles because you are still learning the code. This is normal, and you need to overcome it. No matter what you do, please keep a track record of your **quantitative experiments**. What experiments have you done? How do you know your training is on the right track? Do you have any training loss curve? Do you have any baseline benchmark that is working? Any troubleshoot plan?

Please focus on one specific (small) task and do not be too ambitious. I am not asking you to win the Turing award. I am asking you to learn something.

Append a one-page summary of the things you have done in check point #3 to the previous check point # 1 and #2. (So altogether you should have 3 pages after check point #3.) They do not need to contain any complete story, but you need to put down the plots, tables, figures, etc. Write a few major bullet points.

© 2021 Stanley Chan. All Rights Reserved. 