Homework 2

Shao-Ting Chiu (UIN:433002162)

10/7/22

Table of contents

Homework Description

- Course: ECEN649, Fall2022

  Problems from the book:

  3.6 (10 pt)

  4.2 (10 pt)

  4.3 (10 pt)

4.4 (10 pt)

4.8 (20 pt)

- Deadline: `Oct. 12th, 11:59 am`

Computational Enviromnent Setup

Third-party libraries

```python
%matplotlib inline
import sys # system information
import matplotlib # plotting
import scipy.stats as st # scientific computing
import pandas as pd # data managing
import numpy as np # numerical comuptation
from numpy import linalg as LA
import scipy as sp
import scipy.optimize as opt
import sympy as sp
import matplotlib.pyplot as plt
from numpy.linalg import inv, det
from numpy.random import multivariate_normal as mvn
from numpy.random import binomial as binom
# Matplotlib setting
plt.rcParams['text.usetex'] = True
matplotlib.rcParams['figure.dpi']= 300
```

Version

```python
print(sys.version)
print(matplotlib.__version__)
print(sp.__version__)
print(np.__version__)
print(pd.__version__)
```

```
3.8.14 (default, Sep  6 2022, 23:26:50)
[Clang 13.1.6 (clang-1316.0.21.2.5)]
3.3.1
1.6.2
1.19.1
1.1.1
```

Problem 3.6 (Python Assignment)

Using the synthetic data model in Section A8.1 for the homoskedastic case with $\mu_0 = (0, \dots, 0)$, $\mu_1 = (1, \dots, 1)$, $P(Y = 0) = P(Y = 1)$, and $k = d$ (independent features), generate a large number (e.g., $M = 1000$) of training data sets for each sample size $n = 20$ to $n = 100$, in steps of 10, with $d = 2, 5, 8$, and $\sigma = 1$. Obtain an approximation of the expected classification error $E[\epsilon_n]$ of the nearest centroid classifier in each case by averaging ϵ_n, computed using the exact formula (3.13), over the M synthetic training data sets. Plot $E[\epsilon_n]$ as a function of the sample size, for $d = 2, 5, 8$ (join the individual points with lines to obtain a smooth curve). Explain what you see.

- The formula in Braga-Neto (2020, 56, Eq. 3.13)
  - $\epsilon_n = \frac{1}{2}\left(\Phi\left(\frac{a_n^T\hat{\mu}_0+b_n}{\|a_n\|}\right) + \Phi\left(-\frac{a_n^T\hat{\mu}_1+b_n}{\|a_n\|}\right)\right)$
    - $\mu_0 = (0, \dots, 0)$ $\hat{\mu}_0 = \frac{1}{N_0}\sum_{i=1}^{n} X_i I_{Y_i=0}$
    - $\mu_1 = (1, \dots, 1)$ $\hat{\mu}_1 = \frac{1}{N_1}\sum_{i=1}^{n} X_i I_{Y_i=1}$
    - $a_n = \hat{\mu}_1 - \hat{\mu}_0$
    - $b_n = \frac{(\hat{\mu}_1-\hat{\mu}_0)(\hat{\mu}_1+\hat{\mu}_0)}{2}$

```python
def hat_mu(m):
    return np.mean(m, axis=0)

def get_an(hm0,hm1):
    return hm1 - hm0

def get_bn(hm0,hm1):
    return (hm1 - hm0)*(hm1+hm0).T/2

def epsilon(hmu0, hmu1, p0=0.5):
    p1 = 1-p0
    an = get_an(hmu0, hmu1)
    bn = get_bn(hmu0, hmu1)
    epsilon0 = st.norm.cdf((an*hmu0.T + bn)/LA.norm(an))
    epsilon1 = st.norm.cdf(-(an*hmu1.T+ bn)/LA.norm(an))
    return (p0*epsilon0 + p1*epsilon1)[0][0]

class GaussianDataGen:
    def __init__(self, n, d, s=1, mu=0):
        self.n = n
        self.d = d
        self.mu = np.ones(d) * mu
        self.s = s
        self.cov = self.get_cov()

    def get_cov(self):
        return np.identity(self.d) * self.s

    def sample(self):
        hmuV = np.zeros(self.d)
        for i in range(0,self.d):
            hmuV[i] = np.mean(np.random.normal(self.mu[0], self.s, self.n))
        return np.matrix(hmuV)

def cal_eps(dg0, dg1, p0=0.5):
    hmuV0 = dg0.sample()
    hmuV1 = dg1.sample()
    return epsilon(hmuV0, hmuV1, p0=0.5)
cal_eps_func = np.vectorize(cal_eps)

def exp_try_nd(n, d, s=1,M=1000):
    gX0 = GaussianDataGen(n=n, d=d, s= s,mu=0)
    gX1 = GaussianDataGen(n=n, d=d, s= s, mu=1)
    eps = cal_eps_func([gX0 for i in range(0,M)], gX1)
    return np.mean(eps)
exp_try_nd_func = np.vectorize(exp_try_nd)

"""
M = 1000
ns = np.arange(20,80, 10)
s = 1
dres = {2:[],5:[],8:[]}


for k in dres.keys():
    dres[k] = exp_try_nd_func(ns,k,M)


fig, ax = plt.subplots()
for k in dres.keys():
    ax.plot(ns, dres[k], 'o',label="d={}".format(k))
ax.set_xlabel("n")
ax.set_ylabel("$E[\\epsilon_n]$")
ax.legend();
"""
```

```
'\nM = 1000\nns = np.arange(20,80, 10)\ns = 1\ndres = {2:[],5:[],8:[]}\n\n\nfor k in dres.ke
```

Problem 4.2

A common method to extend binary classification rules to K classes, $K > 2$, is the *one-vs-one approach*, in which $K(K-1)$ classifiers are trained between all pairs of classes, and a majority vote of assigned labels is taken.

(a)

Formulate a multiclass version of parametric plug-in classification using the one-vs-one approach.

Let $\psi^*_{i,j}$ be a one-one classifiers that $i \neq j$, and $\{(i,j)|i \in [1,k], j \in [1,k], i \neq j\}$. For K classes, there are $K(K-1)$ classifiers; for each classifier $\psi^*_{i,j}$ and $x \in R^d$,

$$\psi^*_{ij,n} = \begin{cases} 1, & D_{ij,n}(x) > k_{ij,n} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

where

- $D_{ij,n}(x) = \ln \frac{p(x|\theta_{i,n})}{p(x|\theta_{j,n})}$
- $k_{ij,n} = \ln \frac{P(Y=j)}{P(Y=i)}$
- Noted that feature-label distribution is expressed via a familty of PDF $\{p(x|\theta_i)|\theta \in \Theta \subseteq R^m\}$, for $i = 1, \dots, K$.

Let $\psi^*_{i,n} = \sum_{j \neq i} I_{\psi^*_{ij,n}=1}$, and the one-vs-one classifier is

$$\psi^*_n(x) = \arg \max_{k=1,\dots,K} \psi^*_{k,n}$$

(b)

Show that if the threshold $k_{ij,n}$ between classes i and j is given by $\frac{\ln \hat{c}_j}{\ln \hat{c}_i}$, then the one-vs-one parametric classification rule is equivalent to the simple decision.

$$\psi_n(x) = \arg \max_{k=1,\dots,K} \hat{c}_k p(x|\theta_{k,n}), x \in R^d$$

(For simplicity, you may ignore the possibility of ties.)

(c)

Applying the approach in items (a) and (b), formulate a multiclass version of Gaussian discriminant analysis. In the case of multiclass NMC, with all thresholds equal to zero, how does the decision boundary look like?

Problem 4.3

Under the general Gaussian model $p(x|Y=0) \sim \mathcal{N}_d(\mu_0, \sum_0)$ and $p(x|Y=1) \sim \mathcal{N}_d(\mu_1, \sum_1)$, the classification error $\epsilon_n = P(\psi_n(X) \neq Y|S_n)$ of *any* linear classifier in the form

$$\psi_n(x) = \begin{cases} 1, & a_n^T x + b_n > 0, \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

(examples discussed so far include LDA and its variants, and the logistic classifier) can be readily computed in terms of Φ (the CDF of a standard normal random variable), the classifier parameters a_n and b_n, and the distributional parameters $c = P(Y=1)$, μ_0, μ_1, Σ_0, and Σ_1.

(a)

Show that

$$\epsilon_n = (1-c)\Phi\left(\frac{a_n^T\mu_0 + b_n}{\sqrt{a_n^T\Sigma_0 a_n}}\right) + c\Phi\left(-\frac{a_n^T\mu_1 + b_n}{\sqrt{a_n^T\Sigma_1 a_n}}\right)$$

Hint: the discriminant $a_n^T x + b_n$ has a simple Gaussian distribution in each class.

(b)

Compute the errors of the NMC, LDA, and DLDA classifiers in Example 4.2 if $c = 1/2$,

$$\mu_0 = \begin{bmatrix} 2 \\ 3 \end{bmatrix}, \mu 1 = \begin{bmatrix} 6 \\ 5 \end{bmatrix}, \Sigma_0 = \begin{bmatrix} 1 & 1 \\ 1 & 2 \end{bmatrix}, \text{ and } \Sigma_1 = \begin{bmatrix} 4 & 0 \\ 0 & 1 \end{bmatrix}$$

Which classifier does the best?

Problem 4.4

Even in the Gaussian case, the classification error of quadratic classifiers in general require numerical integration for its computation. In some special simple cases, however, it is possible to obtain exact solutions. Assume a two-dimensional Gaussian problem with $P(Y=1) = \frac{1}{2}$, $\mu_0 = \mu_1 = 0$, $\Sigma_0 = \sigma_0^2 I_2$, and $\Sigma_1 = \sigma_1^2 I_2$. For definiteness, assume that $\sigma_0 < \sigma_1$.

(a)

Show that the Bayes classifier is given by

$$\psi^*(x) = \begin{cases} 1, & \|x\| > r^*, \\ 0, & \text{otherwise} \end{cases}, \qquad \text{where } r^* = \sqrt{2\left(\frac{1}{\sigma_0^2} - \frac{1}{\sigma_1^2}\right)^{-1} \ln \frac{\sigma_1^2}{\sigma_0^2}} \tag{3}$$

In particular, the optimal decision boundary is a circle of radius r^*.

The inverted Σ_1 and Σ_2 are[1]

$$\Sigma_0 = \sigma_0^2 I_2 = \begin{bmatrix} \sigma_0^2 & 0 \\ 0 & \sigma_0^2 \end{bmatrix} \tag{5}$$

$$\Sigma_0^{-1} = \frac{1}{\sigma_0^4} \begin{bmatrix} \sigma_0^2 & 0 \\ 0 & \sigma_0^2 \end{bmatrix} = \sigma_0^{-2} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \sigma_0^{-2} I_2 \tag{6}$$

$$\Sigma_1^{-1} = \sigma_1^{-2} I_2 \tag{7}$$

Use the derivation in Braga-Neto (2020, 74),

$$A_n = \begin{bmatrix} a_{11} & a_{12} \\ a_{12} & a_{22} \end{bmatrix} = \frac{-1}{2} \Sigma_1^{-1} - \Sigma_0^{-1} = \frac{-1}{2} (\sigma_1^{-2} - \sigma_0^{-2}) \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{8}$$

$$b_n = \begin{bmatrix} b_{n,1} \\ b_{n,2} \end{bmatrix} = \Sigma_1^{-1} \underbrace{\mu_1}_{=0} - \Sigma_0^{-1} \underbrace{\mu_0}_{=0} \tag{9}$$

$$= \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{10}$$

$$c = -\frac{1}{2} \ln \frac{|\Sigma_1|}{|\Sigma_0|} = \frac{-1}{2} \ln \frac{\sigma_1^4}{\sigma_0^4} = -\ln \frac{\sigma_1^2}{\sigma_0^2}$$

According to Braga-Neto (2020, Eq. 4.26), the 2-dimensional QDA decision boundary is

[1]
$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \tag{4}$$

$$D(x) = a_{11}x_1^2 + 2a_{12}x_1x_2 + a_{22}x_2^2 + b_1x_1 + b_2x_2 + c = 0 \tag{11}$$

$$a_{11}(x_1^2 + x_2^2) = \ln\frac{\sigma_1^2}{\sigma_0^2} \tag{12}$$

$$x_1^2 + x_2^2 = 2(\frac{1}{\sigma_0^2} - \frac{1}{\sigma_1^2})^{-1}\ln\frac{\sigma_1^2}{\sigma_0^2} \tag{13}$$

$$r^* = \sqrt{x_1^2 + x_2^2} = \sqrt{2(\frac{1}{\sigma_0^2} - \frac{1}{\sigma_1^2})^{-1}\ln\frac{\sigma_1^2}{\sigma_0^2}} \tag{14}$$

Noted that $\left(\frac{1}{\sigma_0^2} - \frac{1}{\sigma_1^2}\right) > 0$ because $\sigma_0 < \sigma_1$

For any point $\|x_j\| > r^*$, the discriminant (D) is larger than 0, and $\psi^*(x_j) = 1$.

(b)

Show that the corresponding Bayes error is given by

$$\epsilon^* = \frac{1}{2} - \frac{1}{2}(\frac{\sigma_1^2}{\sigma_0^2} - 1)e^{-(1-\frac{\sigma_0^2}{\sigma_1^2})^{-1}\ln\frac{\sigma_1^2}{\sigma_0^2}}$$

In particular, the Bayes error is a function only of the ratio of variances $\frac{\sigma_1^2}{\sigma_0^2}$, and $\epsilon^* \to 0$ as $\frac{\sigma_1^2}{\sigma_0^2} \to \infty$.

Hint: use polar coordinates to solve the required integrals analytically.

Part I: Definition of errors

$$\begin{aligned} \epsilon^0[\psi^*] &= P(D^*(X) > k^*|Y = 0) \qquad (15) \\ &= P(\|x\| > r^*|Y = 0) \qquad (16) \end{aligned}$$

$$\begin{aligned} \epsilon^1[\psi^*] &= P(D^*(X) \le k^*|Y = 1) \qquad (17) \\ &= P(\|x\| \le r^*|Y = 1) \qquad (18) \end{aligned}$$

Part II: PDF of 2D Gaussian

$$
\begin{align}
p(x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}) &= \frac{1}{\sqrt{(2\pi)^2\sigma^4}} \exp(-\frac{1}{2} x^T \Sigma^{-1} x) \tag{19} \\
&= \frac{1}{\sqrt{(2\pi)^2\sigma^4}} \exp(-\frac{1}{2}\frac{x_1^2 + x_2^2}{\sigma^2}) \tag{20} \\
&= \frac{1}{2\pi\sigma^2} \exp(-\frac{x_1^2 + x_2^2}{2\sigma^2}) \tag{21} \\
& \tag{22}
\end{align}
$$

Use the polar coordination, $x_1 = r\cos\theta$ and $x_2 = r\sin\theta$. $x_1^2 + x_2^2 = r^2$. We can transform 2D gaussian into polar coordination:

$$p(r, \theta) = \frac{1}{2\pi\sigma^2} \exp(-\frac{r^2}{2\sigma^2})$$

Part III: Integration

$$
\begin{align}
\epsilon^0[\psi^*] &= \int_{\theta=0}^{\theta=2\pi} \int_{r=r^*}^{\infty} \frac{1}{2\pi\sigma_0^2} \exp(-\frac{r^2}{2\sigma_0^2}) r dr d\theta \tag{23} \\
&= \frac{1}{2\pi\sigma_0^2} \int_{\theta=0}^{\theta=2\pi} \int_{r=r^*}^{\infty} \exp(-\frac{r^2}{2\sigma_0^2}) r dr d\theta \tag{24} \\
&= \frac{1}{2\pi\sigma_0^2} \int_{\theta=0}^{\theta=2\pi} \sigma_0^2 \exp(-\frac{r_*^2}{2\sigma_0^2}) d\theta \tag{25} \\
&= \exp(-\frac{r_*^2}{2\sigma_0^2}) \tag{26} \\
&= \exp\left(-\frac{1}{\sigma_0^2(\sigma_0^{-2} - \sigma_1^{-2})} \ln\frac{\sigma_1^2}{\sigma_0^2}\right) \tag{27} \\
&= \exp\left(\frac{-1}{(1 - \frac{\sigma_0^2}{\sigma_1^2})} \ln\frac{\sigma_1^2}{\sigma_0^2}\right) \tag{28} \\
&= \exp\left(-(1 - \frac{\sigma_0^2}{\sigma_1^2})^{-1} \ln\frac{\sigma_1^2}{\sigma_0^2}\right) \tag{29}
\end{align}
$$

$$
\begin{align}
\epsilon^1[\psi^*] &= \int_{\theta=0}^{\theta=2\pi}\int_{r=0}^{r=r^*} \frac{1}{2\pi\sigma_1^2}\exp(-\frac{r^2}{2\sigma_1^2}) r dr d\theta \tag{30}\\
&= 1 - \exp(-\frac{r_*^2}{2\sigma_1^2}) \tag{31}\\
&= 1 - \exp\left(-\frac{1}{\sigma_1^2(\sigma_0^{-2} - \sigma_1^{-2})}\ln\frac{\sigma_1^2}{\sigma_0^2}\right) \tag{32}\\
&= 1 - \exp\left(-\frac{1}{(\frac{\sigma_1^2}{\sigma_0^2} - 1)}\ln\frac{\sigma_1^2}{\sigma_0^2}\right) \tag{33}\\
&= 1 - \exp\left(-\frac{\frac{\sigma_0^2}{\sigma_1^2}}{(1 - \frac{\sigma_0^2}{\sigma_1^2})}\ln\frac{\sigma_1^2}{\sigma_0^2}\right) \tag{34}\\
&= 1 - \exp\left(-\frac{\sigma_0^2}{\sigma_1^2}(1 - \frac{\sigma_0^2}{\sigma_1^2})^{-1}\ln\frac{\sigma_1^2}{\sigma_0^2}\right) \tag{35}
\end{align}
$$

Issue

- Can not find the proper simplification.
- https://ardianumam.wordpress.com/2017/10/19/deriving-gaussian-distribution/
- Integrate the area outside the circle with (Y=0) and Integrate the area inside the circle with (Y=1)

(c)

Compare the optimal classifier to the QDA classifier in Example 4.3. Compute the error of the QDA classifier and compare to the Bayes error.

Problem 4.8 (Python Assignment)

Apply linear discriminant analysis to the stacking fault energy (SFE) dataset (see Braga-Neto (2020, sec. A8.4)), already mentioned in Braga-Neto (2020, ch. 1). Categorize the SFE values into two classes, low (SFE ≤ 35) and high (SFE ≥ 45), excluding the middle values.

(a)

Apply the preprocessing steps in `c01_matex.py` to obtain a data matrix of dimensions 123(number of sample points) $\times$ 7(number of features), as described in Braga-Neto (2020, sec. 1.8.2).

Define low (SFE $\leq$ 35) and high (SFE $\geq$ 45) labels for the data. Pick the first 20% of the sampe point s to be the training data and the remaining 80% to be test data.

(b)

Using the function `ttest_ind` from the `scipy.stats` module, apply Welch's two-sample t-test on the training data, and produce a table with the predictors, T statistic, and p-value, ordered with largest absolute T statistics at the top.

(c)

Pick the top two predictors and design an LDA classifier. (This is an example of *filter feature selection*, to be discussed in Chapter 9.). Plot the training data with the superimposed LDA decision boundary. Plot the testing data with the superimposed previously-obtained LDA decision boundary. Estimate the classification error rate on the training and test data. What do you observe?

(d)

Repeat for the top three, and five predictors. Estimate the errors on the training and testing data (there is no need to plot the classifiers). What can you observe?

References

Braga-Neto, Ulisses. 2020. *Fundamentals of Pattern Recognition and Machine Learning*. Springer.