Homework 2

Shao-Ting Chiu (UIN:433002162)

9/29/22

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 as st # scientific computing
import pandas as pd # data managing
import numpy as np # numerical comuptation
import scipy.optimize as opt
import sympy as sp
import matplotlib.pyplot as plt
from scipy.special import erf
from numpy.linalg import inv, det
from scipy.linalg import block_diag
from scipy.stats import norm # for problem 2.17 (b)
from scipy.stats import multivariate_normal
# Matplotlib setting
plt.rcParams['text.usetex'] = True
matplotlib.rcParams['figure.dpi']= 300
RES_GRID = 90
```

Version

```
print(sys.version)
print(matplotlib.__version__)
print(st.__version__)
print(np.__version__)
print(pd.__version__)
```

```
3.8.12 (default, Oct 22 2021, 18:39:35)
[Clang 13.0.0 (clang-1300.0.29.3)]
3.3.1
1.5.2
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 $\mu_0 = (0, \ldots, 0)$, $\mu_1 = (1, \ldots, 1)$, $P(Y = 0) = P(Y = 1)$, 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.

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.

(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.

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

(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

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

(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

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

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

(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.

(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

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 ≤ 35) and high (SFE ≥ 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.