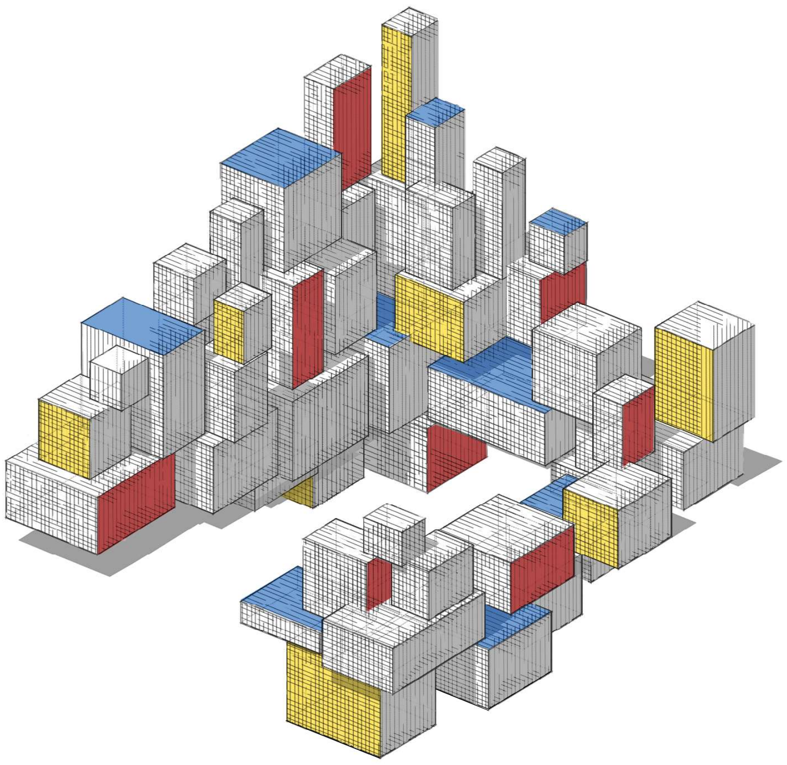
Solutions Manual

to Accompany

Data Science and Machine Learning: Mathematical and Statistical Methods



Dirk P. Kroese Zdravko I. Botev Thomas Taimre
Slava Vaisman Robert Salomone

25th August 2021

CONTENTS

Preface

We believe that the only effective way to master the theory and practice of Data Science and Machine learning is through exercises and experiments. For this reason, we included many exercises and algorithms in *Data Science and Machine Learning: Mathematical and Statistical Methods* (DSML), Chapman and Hall/CRC, 2019.

This companion volume to DSML is written in the same style and contains a wealth of additional material: worked solutions for the over 150 exercises in DSML, many Python programs and additional illustrations.

Like DSML, this solution manual is aimed at anyone interested in gaining a better understanding of the mathematics and statistics that underpin the rich variety of ideas and machine learning algorithms in data science. One of the main goals of the manual is to provide a comprehensive solutions guide to instructors, which will aid student assessment and stimulate further student development. In addition, this manual offers a unique complement to DSML for self-study. All too often a stumbling block for learning is the unavailability of worked solutions and actual algorithms.

The solutions manual covers a wide range of exercises in data analysis, statistical learning, Monte Carlo methods, unsupervised learning, regression, regularization and kernel methods, classification, decision trees and ensemble methods, and deep learning. Our choice of using Python was motivated by its ease of use and clarity of syntax.

Reference numbers to DSML are indicated in boldface blue font. For example, Definition **1.1.1** refers to the corresponding definition in DSML, and **(1.7)** refers to equation (1.7) in DSML, whereas Figure 1.1 refers to the first numbered figure in the present document.

This solutions manual was financially supported by the Australian Research Council *Centre of Excellence for Mathematical & Statistical Frontiers*, under grant number CE140100049.

Dirk Kroese, Zdravko Botev,
Thomas Taimre, Radislav Vaisman, and Robert Salomone
Brisbane and Sydney

CHAPTER 1

Importing, Summarizing, and Visualizing Data

1. Visit the UCI Repository https://archive.ics.uci.edu/. Read the description of the data and download the Mushroom data set `agaricus-lepiota.data`. Using **pandas**, read the data into a `DataFrame` called `mushroom`, via **read_csv**.

We can import the file directly via its URL:

```
import pandas as pd

URL = 'http://archive.ics.uci.edu/ml/machine-learning-databases/
    mushroom/agaricus-lepiota.data'
mushroom = pd.read_csv(URL,header=None)
```

(a) How many features are in this data set?

Solution: There are 23 features.

```
mushroom.info()
```

```
    <class 'pandas.core.frame.DataFrame'>
RangeIndex: 8124 entries, 0 to 8123
Data columns (total 23 columns):
0      8124 non-null object
1      8124 non-null object
2      8124 non-null object
3      8124 non-null object
4      8124 non-null object
5      8124 non-null object
6      8124 non-null object
7      8124 non-null object
8      8124 non-null object
9      8124 non-null object
10     8124 non-null object
11     8124 non-null object
12     8124 non-null object
13     8124 non-null object
14     8124 non-null object
15     8124 non-null object
```

```
16     8124 non-null object
17     8124 non-null object
18     8124 non-null object
19     8124 non-null object
20     8124 non-null object
21     8124 non-null object
22     8124 non-null object
dtypes: object(23)
memory usage: 1.4+ MB
```

(b) What are the initial names and types of the features?

Solution: From the output of `mushroom.info()`, we see that the initial names of the features are $0, 1, 2, \ldots, 22$, and that they all have the type `object`.

(c) Rename the first feature (index 0) to `'edibility'` and the sixth feature (index 5) to `'odor'` [Hint: the column names in **pandas** are immutable; so individual columns cannot be modified directly. However it is possible to assign the entire column names list via `mushroom.columns = newcols`.]

Solution:

```
# create a list object that contains the column names
newcols = mushroom.columns.tolist()

# assign new values in the list
newcols[0] = 'edibility'
newcols[5] = 'odor'

# replace the column names with our list
mushroom.columns = newcols
```

(d) The 6th column lists the various odors of the mushrooms: encoded as `'a'`, `'c'`, ... Replace these with the names `'almond'`, `'creosote'`, etc. (categories corresponding to each letter can be found on the website). Also replace the `'edibility'` categories `'e'` and `'p'` with `'edible'` and `'poisonous'`.

Solution:

```
DICT = {'a': "almond", 'c': "creosote", 'f': "foul",
'l':"anise",'m': "musty",'n':"none", 'p': "pungent",
's':"spicy", 'y':"fishy"}
mushroom.odor = mushroom.odor.replace(DICT)

DICT = {'e': "edible", 'p':"poisonous"}
mushroom.edibility = mushroom.edibility.replace(DICT)
```

(e) Make a contingency table cross-tabulating `'edibility'` and `'odor'`.

Solution:

```
pd.crosstab(mushroom.odor,mushroom.edibility)
```

```
    edibility  edible  poisonous
odor
almond         400          0
```

```
anise          400          0
creosote         0        192
fishy            0        576
foul             0       2160
musty            0         36
none          3408        120
pungent          0        256
spicy            0        576
```

(f) Which mushroom odors should be avoided, when gathering mushrooms for consumption?

Solution: From the table in the previous question, we see that the data indicates that all mushroom odors except almond and anise have observations that are poisonous. Thus, all odors other than these two should be avoided.

(g) What proportion of odorless mushroom samples were safe to eat?

Solution: We can calculate the proportion by obtaining values directly off the contingency table and directly calculating:

```
3408/(3408+120)
```

```
0.9659863945578231
```

Alternatively, we can find the answer without the table:

```
mushroom[(mushroom.edibility=='edible') & (mushroom.odor == '
    none')].shape[0]/mushroom[mushroom.odor=='none'].shape[0]
```

```
0.9659863945578231
```

2. Change the type and value of variables in the **nutri** data set according to Table 1.2 and save the data as a CSV file. The modified data should have eight categorical features, three floats, and two integer features.

Solution:

3. It frequently happens that a table with data needs to be restructured before the data can be analyzed using standard statistical software. As an example, consider the test scores in Table 1.3 of 5 students before and after specialized tuition.

Table 1.3: Student scores.

Student	Before	After
1	75	85
2	30	50
3	100	100
4	50	52
5	60	65

This is not in the standard format described in Section 1.1. In particular, the student scores are divided over two columns, whereas the standard format requires that they are collected in one column, e.g., labelled 'Score'. Reformat the table in standard format, using three features:

- 'Score', taking continuous values,
- 'Time', taking values 'Before' and 'After',
- 'Student', taking values from 1 to 5.

Solution: Up to a possible reordering of the rows, your table should look like the one given below, which was made with the **melt** method of **pandas**.

```
# manually create dataframe with data from table
values = [[1,75,85],[2,30,50],[3,100,100],[4,50,52],[5,60,65]]

import pandas as pd
df = pd.DataFrame(values, columns=['Student','Before', 'After'])

# format dataframe as required
df = pd.melt(df, id_vars=['Student'], var_name="Time", value_vars=['
    Before','After'])
print(df)
```

```
   Student    Time  value
0        1  Before     75
1        2  Before     30
2        3  Before    100
3        4  Before     50
4        5  Before     60
5        1   After     85
6        2   After     50
7        3   After    100
8        4   After     52
9        5   After     65
```

4. Create a similar barplot as in Figure 1.5, but now plot the corresponding *proportions* of males and females in each of the three situation categories. That is, the heights of the bars should sum up to 1 for both barplots with the same 'gender' value. [Hint: **seaborn** does not have this functionality built in, instead you need to first create a contingency table and use **matplotlib.pyplot** to produce the figure.]

Solution:

☞ 2 5. The **iris** data set, mentioned in Section 1.1, contains various features, including 'Petal.Length' and 'Sepal.Length', of three species of iris: setosa, versicolor, and virginica.

(a) Load the data set into a **pandas** DataFrame object.

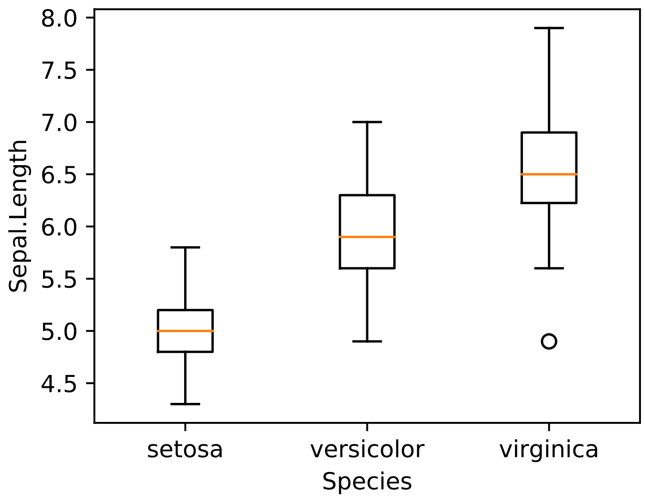
Solution:

```
import pandas as pd
urlprefix = 'http://vincentarelbundock.github.io/Rdatasets/csv/'
dataname = 'datasets/iris.csv'
iris = pd.read_csv(urlprefix + dataname)
```

(b) Using **matplotlib.pyplot**, produce boxplots of 'Petal.Length' for each the three species, in one figure.

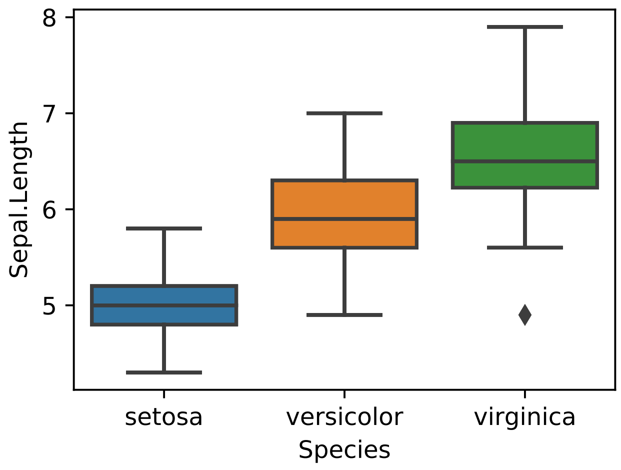
Solution:

```
import matplotlib.pyplot as plt
labels = ["setosa","versicolor","virginica"]
plt.boxplot([setosa["Sepal.Length"],versicolor["Sepal.Length"],
    virginica["Sepal.Length"]], labels=labels)
```



Note that it is simpler to create a similar plot in **seaborn** via:

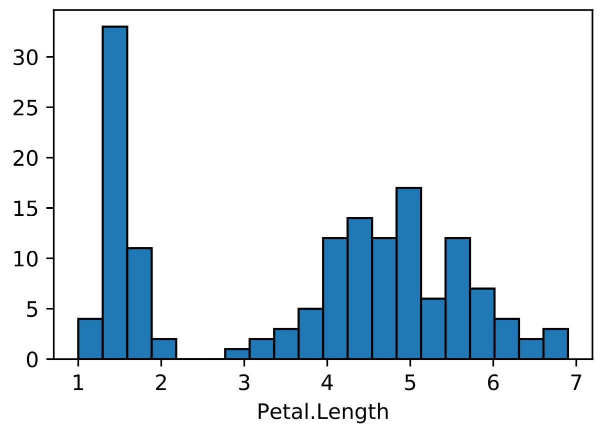
```
import seaborn as sns
sns.boxplot(x='Species', y='Sepal.Length', data=iris)
```



(c) Make a histogram with 20 bins for `'Petal.Length'`.

Solution:

```
plt.hist(iris["Petal.Length"], bins=20, edgecolor='black')
plt.xlabel("Petal.Length")
```



(d) Produce a similar scatter plot for `'Sepal.Length'` against `'Petal.Length'` to that of the left plot in Figure **1.9**. Note that the points should be colored according to the `'Species'` feature as per the legend in the right plot of the figure.

Solution:

```
DICT = {0:(setosa, 'b','setosa'), 1: (versicolor,'g','versicolor
    '), 2: (virginica,'r','virginica') }

for k in DICT:
  plt.scatter(DICT[k][0]["Petal.Length"], DICT[k][0]["Sepal.
      Length"], color=DICT[k][1], alpha=0.5)

plt.xlabel('Petal.Length')
plt.ylabel('Sepal.Length')
plt.show()
```

(e) Using the **`kdeplot`** method of the **`seaborn`** package, reproduce the right plot of Figure **1.9**, where kernel density plots for `'Petal.Length'` are given. ☞ 131

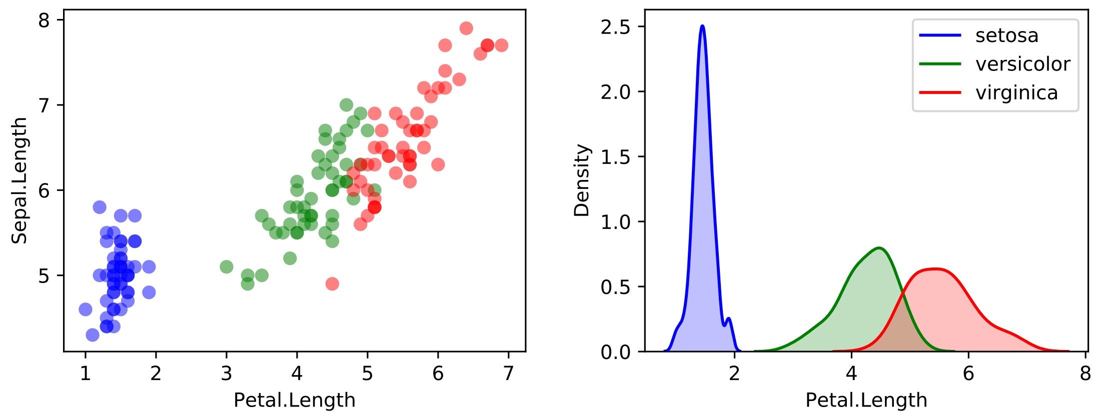


Figure 1.9: Left: scatter plot of `'Sepal.Length'` against `'Petal.Length'`. Right: kernel density estimates of `'Petal.Length'` for the three species of iris.

Solution:

```
sns.kdeplot(setosa["Petal.Length"],color = 'b',shade=True)
sns.kdeplot(versicolor["Petal.Length"], color='g',shade=True)
sns.kdeplot(virginica["Petal.Length"], color='r',shade=True)
plt.legend(('setosa','versicolor','virginica'))
plt.xlabel('Petal.Length')
plt.ylabel('Density')
```

6. Import the data set `EuStockMarkets` from the same website as the `iris` data set above. The data set contains the daily closing prices of four European stock indices during the 1990s, for 260 working days per year.

Solution:

(a) Create a vector of times (working days) for the stock prices, between 1991.496 and 1998.646 with increments of 1/260.

Solution:

(b) Reproduce Figure 1.10. [Hint: Use a dictionary to map column names (stock indices) to colors.]

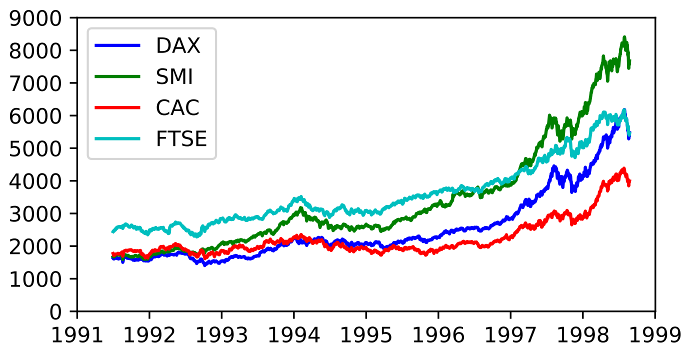


Figure 1.10: Stocks

Solution:

7. Consider the KASANDR data set from the UCI Machine Learning Repository, which can be can be downloaded from

http://archive.ics.uci.edu/ml/machine-learning-databases/00385/de.tar.bz2.

This archive file has a size of 900Mb, so it may take a while to download. Uncompressing the file (e.g., via 7-Zip) yields a directory de containing two large CSV files: test_de.csv and train_de.csv, with sizes 372Mb and 3Gb, respectively. Such large data files can still be processed efficiently in **pandas**, provided there is enough processor memory. The files contain records of user information from www.kelkoo.com web logs in Germany as well as meta-information on users, offers, and merchants. The data sets have 7 attributes and 1919561 and 15844717 rows, respectively. The data sets are anonymized via hex strings.

(a) Load train_de.csv into a **pandas** DataFrame object **de**, using

read_csv('train_de.csv', delimiter = '\t').

If not enough memory is available, load test_de.csv instead. Note that entries are separated here by tabs, not commas. Time how long it takes for the file to load, using the **time** package. (It took 38 seconds for train_de.csv to load on one of our computers).

Solution: Note that we must first ensure that train_de.csv is in our working directory.

```python
import time
import pandas as pd

start  = time.time()
de = pd.read_csv('train_de.csv',delimiter='\t')
print("Import took {:.2f} seconds.".format(time.time()-start))
```

```
Import took 60.14 seconds.
```

(b) How many unique users and merchants are in this data set?

Solution:

```python
print("unique users :", len(de.userid.unique()))
print("unique merchants :", len(de.merchant.unique()))
```

```
unique users : 291485
unique merchants :  703
```

8. Visualizing data involving more than two features requires careful design, which is often more of an art than a science.

(a) Go to Vincent Arel-Bundocks's website (URL given in Section 1.1) and read the `Orange` data set into a `pandas` `DataFrame` object called `orange`. Remove its first (unnamed) column.

Solution:

(b) The data set contains the circumferences of 5 orange trees at various stages in their development. Find the names of the features.

Solution:

(c) In Python, import `seaborn` and visualize the growth curves (circumference against age) of the trees, using the `regplot` and `FacetGrid` methods.

Solution:

CHAPTER 2

STATISTICAL LEARNING

1. Suppose that the loss function is the piecewise linear function:

$$\text{Loss}(y, y') = \alpha(y' - y)^+ + \beta(y - y')^+, \quad \alpha, \beta > 0\,,$$

where c^+ is equal to c if $c > 0$, and zero otherwise. Show that the minimizer of the risk $\ell(g) = \mathbb{E}\,\text{Loss}(Y, g(\boldsymbol{X}))$ satisfies

$$\mathbb{P}[Y < g^*(\boldsymbol{x}) \,|\, \boldsymbol{X} = \boldsymbol{x}] = \frac{\beta}{\alpha + \beta}\,.$$

In other words, $g^*(\boldsymbol{x})$ is the $\beta/(\alpha+\beta)$ quantile of Y, conditional on $\boldsymbol{X} = \boldsymbol{x}$.

Solution: Assume all expectation and probability operators are conditional on $\boldsymbol{X} = \boldsymbol{x}$ and write $g(\boldsymbol{x}) = \widehat{y}$ for short. We have

$$\begin{aligned}
\mathbb{E}\,\text{Loss}(Y, \widehat{y}) &= \alpha \mathbb{E}(\widehat{y} - Y)^+ + \beta \mathbb{E}(Y - \widehat{y})^+ \\
&= \alpha\, \mathbb{E}[(\widehat{y} - Y)\, \mathbb{1}\{Y < \widehat{y}\}] + \beta\, \mathbb{E}[(Y - \widehat{y})\, \mathbb{1}\{Y > \widehat{y}\}] \\
&= \widehat{y}\, (\alpha\, \mathbb{P}[Y < \widehat{y}] - \beta\, \mathbb{P}[Y > \widehat{y}]) + \beta\, \mathbb{E}[Y\, \mathbb{1}\{Y > \widehat{y}\}] - \alpha\, \mathbb{E}[Y\, \mathbb{1}\{Y < \widehat{y}\}] \\
&= \widehat{y}\Big((\alpha+\beta)\, \mathbb{P}[Y < \widehat{y}] - \beta\Big) - (\alpha+\beta)\, \mathbb{E}[Y\, \mathbb{1}\{Y < \widehat{y}\}] + \beta\, \mathbb{E} Y.
\end{aligned}$$

Let y^* be the $\beta/(\alpha+\beta)$ quantile of the distribution of Y, that is,

$$\mathbb{P}[Y < y^*] = \beta/(\alpha+\beta)\,.$$

Then,

$$\begin{aligned}
&\mathbb{E}\,\text{Loss}(Y, \widehat{y}) - \mathbb{E}\,\text{Loss}(Y, y^*) = \\
&= \widehat{y}\Big((\alpha+\beta)\, \mathbb{P}[Y < \widehat{y}] - \beta\Big) - (\alpha+\beta)\Big(\mathbb{E}[Y\, \mathbb{1}\{Y < \widehat{y}\}] - \mathbb{E}[Y\, \mathbb{1}\{Y < y^*\}]\Big) \\
&= (\alpha+\beta)\left\{\widehat{y}\Big(\mathbb{P}[Y < \widehat{y}] - \mathbb{P}[Y < y^*]\Big) + \mathbb{E}[Y\, \mathbb{1}\{Y < y^*\}] - \mathbb{E}[Y\, \mathbb{1}\{Y < \widehat{y}\}]\right\}.
\end{aligned}$$

Hence, if $y^* > \widehat{y}$, then

$$\begin{aligned}
\frac{\mathbb{E}\,\text{Loss}(Y, \widehat{y}) - \mathbb{E}\,\text{Loss}(Y, y^*)}{(\alpha+\beta)} &= \mathbb{E}[Y\, \mathbb{1}\{\widehat{y} < Y < y^*\}] - \widehat{y}\, \mathbb{P}[\widehat{y} < Y < y^*] \\
&= \mathbb{P}[\widehat{y} < Y < y^*]\, \underbrace{(\mathbb{E}[Y \,|\, \widehat{y} < Y < y^*] - \widehat{y})}_{\geqslant 0} \geqslant 0.
\end{aligned}$$

The case for $y^* < \widehat{y}$ is dealt with similarly.

An alternative solution is as follows. Suppose $y^* \geqslant \widehat{y}$. We have three cases to consider.

(a) $y^* \geqslant Y \geqslant \widehat{y}$, so that

$$\mathrm{Loss}(Y, \widehat{y}) - \mathrm{Loss}(Y, y^*) = \beta(Y - \widehat{y}) - \alpha(y^* - Y) \geqslant \alpha(\widehat{y} - y^*).$$

(b) $Y \geqslant y^* \geqslant \widehat{y}$, so that

$$\mathrm{Loss}(Y, \widehat{y}) - \mathrm{Loss}(Y, y^*) = \beta(Y - \widehat{y}) - \beta(Y - y^*) = \beta(y^* - \widehat{y}) \geqslant 0.$$

(c) $y^* \geqslant \widehat{y} \geqslant Y$, so that

$$\mathrm{Loss}(Y, \widehat{y}) - \mathrm{Loss}(Y, y^*) = \alpha(\widehat{y} - Y) - \alpha(y^* - Y) = \alpha(\widehat{y} - y^*).$$

Therefore, when $y^* \geqslant \widehat{y}$,

$$\mathrm{Loss}(Y, \widehat{y}) - \mathrm{Loss}(Y, y^*) \geqslant \begin{cases} \alpha(\widehat{y} - y^*) & Y \leqslant y^* \\ \beta(y^* - \widehat{y}) & Y \geqslant y^*. \end{cases}$$

Hence, when $y^* \geqslant \widehat{y}$,

$$\begin{aligned} \mathbb{E}\mathrm{Loss}(Y, \widehat{y}) - \mathbb{E}\mathrm{Loss}(Y, y^*) &\geqslant \mathbb{E}\alpha(\widehat{y} - y^*)\mathbb{I}\{Y \leqslant y^*\} + \mathbb{E}\beta(y^* - \widehat{y})\mathbb{I}\{Y \geqslant y^*\} \\ &= \alpha(\widehat{y} - y^*)\frac{\beta}{\alpha + \beta} + \beta(y^* - \widehat{y})\frac{\alpha}{\alpha + \beta} = 0. \end{aligned}$$

We obtain the same result when $y^* \leqslant \widehat{y}$.

2. Show that, for the squared-error loss, the approximation error $\ell(g^{\mathcal{G}}) - \ell(g^*)$ in (2.16), is equal to $\mathbb{E}(g^{\mathcal{G}}(\boldsymbol{X}) - g^*(\boldsymbol{X}))^2$. [Hint: expand $\ell(g^{\mathcal{G}}) = \mathbb{E}(Y - g^*(\boldsymbol{X}) + g^*(\boldsymbol{X}) - g^{\mathcal{G}}(\boldsymbol{X}))^2$.]

Solution:

3. Suppose $\mathcal{G}$ is the class of *linear* functions. A linear function evaluated at a feature $\boldsymbol{x}$ can be described as $g(\boldsymbol{x}) = \boldsymbol{\beta}^\top \boldsymbol{x}$ for some parameter vector $\boldsymbol{\beta}$ of appropriate dimension. Denote $g^{\mathcal{G}}(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{\beta}^{\mathcal{G}}$ and $g_\tau^{\mathcal{G}}(\boldsymbol{x}) = \boldsymbol{x}^\top \widehat{\boldsymbol{\beta}}$. Show that

$$\mathbb{E}(g_\tau^{\mathcal{G}}(\boldsymbol{X}) - g^*(\boldsymbol{X}))^2 = \mathbb{E}(\boldsymbol{X}^\top \widehat{\boldsymbol{\beta}} - \boldsymbol{X}^\top \boldsymbol{\beta}^{\mathcal{G}})^2 + \mathbb{E}(\boldsymbol{X}^\top \boldsymbol{\beta}^{\mathcal{G}} - g^*(\boldsymbol{X}))^2 \ .$$

Hence, deduce that the statistical error in (2.16) is $\ell(g_\tau^{\mathcal{G}}) - \ell(g^{\mathcal{G}}) = \mathbb{E}(g_\tau^{\mathcal{G}}(\boldsymbol{X}) - g^{\mathcal{G}}(\boldsymbol{X}))^2$.

Solution: Since $\boldsymbol{\beta}^{\mathcal{G}} = \mathrm{argmin}_{\boldsymbol{\beta}}\, \mathbb{E}(\boldsymbol{\beta}^\top \boldsymbol{X} - g^*(\boldsymbol{X}))^2$, we must have

$$\frac{\partial}{\partial \boldsymbol{\beta}} \mathbb{E}(\boldsymbol{\beta}^\top \boldsymbol{X} - g^*(\boldsymbol{X}))^2 \Big|_{\boldsymbol{\beta} = \boldsymbol{\beta}^{\mathcal{G}}} = \boldsymbol{0} \ .$$

In other words, after simplification, $\boldsymbol{\beta}^{\mathcal{G}}$ satisfies

$$2\mathbb{E}\boldsymbol{X}\Big(\boldsymbol{X}^\top \boldsymbol{\beta}^{\mathcal{G}} - g^*(\boldsymbol{X})\Big) = \boldsymbol{0} \ .$$

Therefore, using the identity $(a + b)^2 = a^2 + b^2 + 2ab$, we obtain

$$\begin{aligned} \mathbb{E}(g_\tau^{\mathcal{G}}(\boldsymbol{X}) - g^*(\boldsymbol{X}))^2 &= \mathbb{E}(g_\tau^{\mathcal{G}}(\boldsymbol{X}) - g^{\mathcal{G}}(\boldsymbol{X}) + g^{\mathcal{G}}(\boldsymbol{X}) - g^*(\boldsymbol{X}))^2 \\ &= \mathbb{E}((\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}^{\mathcal{G}})^\top \boldsymbol{X})^2 + \mathbb{E}(\boldsymbol{X}^\top \boldsymbol{\beta}^{\mathcal{G}} - g^*(\boldsymbol{X}))^2 + 2(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}^{\mathcal{G}})^\top \underbrace{\mathbb{E}\boldsymbol{X}(\boldsymbol{X}^\top \boldsymbol{\beta}^{\mathcal{G}} - g^*(\boldsymbol{X}))}_{=\boldsymbol{0}} \ . \end{aligned}$$

4. Show that formula (2.24) holds for the 0–1 loss with 0–1 response.

Solution:

5. Let $\boldsymbol{X}$ be an n-dimensional normal random vector with mean vector $\boldsymbol{\mu}$ and covariance matrix Σ, where the determinant of Σ non-zero. Show that $\boldsymbol{X}$ has joint probability density

$$f_{\boldsymbol{X}}(\boldsymbol{x}) = \frac{1}{\sqrt{(2\pi)^n\,|\Sigma|}}\,\mathrm{e}^{-\frac{1}{2}\,(\boldsymbol{x}-\boldsymbol{\mu})^\top\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}, \quad \boldsymbol{x}\in\mathbb{R}^n\,.$$

Solution: By (C.27), write $\boldsymbol{X} = \boldsymbol{\mu} + \mathbf{B}\boldsymbol{Z}$, where $\boldsymbol{Z}$ is n-dimensional *standard* normal and $\mathbf{B}\mathbf{B}^\top = \Sigma$. The joint probability density function of $\boldsymbol{Z}$ is

$$f_{\boldsymbol{Z}}(\boldsymbol{z}) = (2\pi)^{-\frac{n}{2}}\,\mathrm{e}^{-\frac{1}{2}\,\boldsymbol{z}^\top\boldsymbol{z}}, \quad \boldsymbol{z}\in\mathbb{R}^n.$$

By the transformation rule (C.23), the joint probability density function of $\boldsymbol{Y} := \boldsymbol{X}-\boldsymbol{\mu} = \mathbf{B}\boldsymbol{Z}$ is thus

$$f_{\boldsymbol{Y}}(\boldsymbol{y}) = (2\pi)^{-\frac{n}{2}}\frac{1}{|\det(\mathbf{B})|}\,\mathrm{e}^{-\frac{1}{2}\,\boldsymbol{y}^\top\mathbf{B}^{-\top}\mathbf{B}^{-1}\boldsymbol{y}} = (2\pi)^{-\frac{n}{2}}\frac{1}{\sqrt{\det(\boldsymbol{\Sigma})|}}\,\mathrm{e}^{-\frac{1}{2}\,\boldsymbol{y}^\top\boldsymbol{\Sigma}^{-1}\boldsymbol{y}}, \quad \boldsymbol{y}\in\mathbb{R}^n.$$

The joint probability density function of $\boldsymbol{X} = \boldsymbol{\mu} + \boldsymbol{Y}$ is obtained by a shift of $f_{\boldsymbol{Y}}$; that is, $f_{\boldsymbol{X}}(\boldsymbol{x}) = f_{\boldsymbol{Y}}(\boldsymbol{y}-\boldsymbol{\mu})$, giving the desired result.

6. Let $\widehat{\boldsymbol{\beta}} = \mathbf{A}^+\boldsymbol{y}$. Using the defining properties of the pseudo-inverse, show that for any $\boldsymbol{\beta}\in\mathbb{R}^p$,

$$\|\mathbf{A}\widehat{\boldsymbol{\beta}} - \boldsymbol{y}\| \leqslant \|\mathbf{A}\boldsymbol{\beta} - \boldsymbol{y}\|.$$

Solution:

7. Suppose that in the polynomial regression Example 2.1 we select the linear class of functions $\mathcal{G}_p$ with $p \geqslant 4$. Then, $g^* \in \mathcal{G}_p$ and the approximation error is zero, because $g^{\mathcal{G}_p}(\boldsymbol{x}) = g^*(\boldsymbol{x}) = \boldsymbol{x}^\top\boldsymbol{\beta}$, where $\boldsymbol{\beta} = [10, -140, 400, -250, 0, \dots, 0]^\top \in \mathbb{R}^p$. Use the tower property to show that the learner $g_\tau(\boldsymbol{x}) = \boldsymbol{x}^\top\widehat{\boldsymbol{\beta}}$ with $\widehat{\boldsymbol{\beta}} = \mathbf{X}^+\boldsymbol{y}$, assuming $\mathrm{rank}(\mathbf{X}) \geqslant 4$, is *unbiased*:

$$\mathbb{E}\,g_{\mathcal{T}}(\boldsymbol{x}) = g^*(\boldsymbol{x})\,.$$

Solution: To see this note that conditional on $\mathbf{X}$, we have

$$\begin{aligned}\mathbb{E}_{\mathbf{X}}[g_{\mathcal{T}}(\boldsymbol{x})] &= \boldsymbol{x}^\top\mathbb{E}_{\mathbf{X}}[\widehat{\boldsymbol{\beta}}] = \boldsymbol{x}^\top\mathbf{X}^+\mathbb{E}_{\mathbf{X}}[\boldsymbol{Y}]\\ &= \boldsymbol{x}^\top\mathbf{X}^+\mathbf{X}\boldsymbol{\beta}^* = \boldsymbol{x}^\top\boldsymbol{\beta}^* = g^*(\boldsymbol{x})\,.\end{aligned}$$

The tower property then yields $\mathbb{E}[g_{\mathcal{T}}(\boldsymbol{x})] = \mathbb{E}[\mathbb{E}[g_{\mathcal{T}}(\boldsymbol{x})\,|\,\mathbf{X}]] = g^*(\boldsymbol{x})$.

8. (Exercise 7 continued.) Observe that the learner $g_{\mathcal{T}}$ can be written as a linear combination of the response variable: $g_{\mathcal{T}}(\boldsymbol{x}) = \boldsymbol{x}^\top\mathbf{X}^+\boldsymbol{Y}$. Prove that for any learner of the form $\boldsymbol{x}^\top\mathbf{A}\boldsymbol{y}$, where $\mathbf{A}\in\mathbb{R}^{p\times n}$ is some matrix and that satisfies $\mathbb{E}_{\mathbf{X}}[\boldsymbol{x}^\top\mathbf{A}\boldsymbol{Y}] = g^*(\boldsymbol{x})$, we have

$$\mathbb{V}\mathrm{ar}_{\mathbf{X}}[\boldsymbol{x}^\top\mathbf{X}^+\boldsymbol{Y}] \leqslant \mathbb{V}\mathrm{ar}_{\mathbf{X}}[\boldsymbol{x}^\top\mathbf{A}\boldsymbol{Y}]\,,$$

where the equality is achieved for $\mathbf{A} = \mathbf{X}^+$. This is called the *Gauss–Markov inequality*. Hence, using the Gauss–Markov inequality deduce that for the unconditional variance:

$$\mathbb{V}\mathrm{ar}[g_{\mathcal{T}}(\boldsymbol{x})] \leqslant \mathbb{V}\mathrm{ar}[\boldsymbol{x}^\top \mathbf{A} \boldsymbol{Y}] \,.$$

Deduce that $\mathbf{A} = \mathbf{X}^+$ also minimizes the expected generalization risk.

Solution:

9. Consider again the polynomial regression Example 2.1. Use the fact that $\mathbb{E}_{\mathbf{X}} \widehat{\boldsymbol{\beta}} = \mathbf{X}^+ \boldsymbol{h}^*(\boldsymbol{u})$, where $\boldsymbol{h}^*(\boldsymbol{u}) = \mathbb{E}[\boldsymbol{Y} \,|\, \boldsymbol{U} = \boldsymbol{u}] = [h^*(u_1), \ldots, h^*(u_n)]^\top$, to show that the expected in-sample risk is:

$$\mathbb{E}_{\mathbf{X}}\, \ell_{\mathrm{in}}(g_{\mathcal{T}}) = \ell^* + \frac{\|\boldsymbol{h}^*(\boldsymbol{u})\|^2 - \|\mathbf{X}\mathbf{X}^+ \boldsymbol{h}^*(\boldsymbol{u})\|^2}{n} + \frac{\ell^* p}{n} \,.$$

Also, use Theorem C.2 to show that the expected statistical error is:

$$\mathbb{E}_{\mathbf{X}}\, (\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta})^\top \mathbf{H}_p (\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}) = \ell^* \mathrm{tr}(\mathbf{X}^+ (\mathbf{X}^+)^\top \mathbf{H}_p) + (\mathbf{X}^+ \boldsymbol{h}^*(\boldsymbol{u}) - \boldsymbol{\beta})^\top \mathbf{H}_p (\mathbf{X}^+ \boldsymbol{h}^*(\boldsymbol{u}) - \boldsymbol{\beta}) \,.$$

Solution: Using the fact that $\mathbb{E}[Y_i - h^*(U_i) \,|\, U_i = u_i] = 0$ and the identity $(a+b)^2 = a^2 + 2ab + b^2$, we have

$$\begin{aligned}
\mathbb{E}_{\mathbf{X}}\, \ell_{\mathrm{in}}(g_{\mathcal{T}}) &= \frac{1}{n}\sum_{i=1}^n \mathbb{E}[(Y_i - h_{\mathcal{T}}(U_i))^2 \,|\, U_i = u_i] \\
&= \frac{1}{n}\sum_{i=1}^n \mathbb{E}[(Y_i - h^*(U_i))^2 \,|\, U_i = u_i] + \frac{1}{n}\sum_{i=1}^n \mathbb{E}[(h^*(U_i) - h_{\mathcal{T}}(U_i))^2 \,|\, U_i = u_i] \\
&= \ell^* + \frac{1}{n}\sum_{i=1}^n \mathbb{E}[(h^*(U_i) - h_{\mathcal{T}}(U_i))^2 \,|\, U_i = u_i]
\end{aligned}$$

Next, recall that $\mathbb{E}[\boldsymbol{h}_{\mathcal{T}}(\boldsymbol{U}) \,|\, \boldsymbol{U} = \boldsymbol{u}] = \mathbb{E}_{\mathbf{X}} \mathbf{X} \widehat{\boldsymbol{\beta}} = \mathbf{X}\mathbf{X}^+ \boldsymbol{h}^*(\boldsymbol{u}) = \boldsymbol{h}^{\mathcal{H}_p}(\boldsymbol{u})$, that is, the learner $h_{\mathcal{T}}(u)$ is an unbiased estimator of $h^{\mathcal{H}_p}(u)$, conditional on $\mathbf{X}$ (or $\boldsymbol{U} = \boldsymbol{u}$). As a result of this, $\mathbb{E}[(h^*(U_i) - h_{\mathcal{T}}(U_i))^2 \,|\, \boldsymbol{U} = \boldsymbol{u}]$ can be decomposed as:

$$\mathbb{E}[(h^*(U_i) - h_{\mathcal{T}}(U_i))^2 \,|\, \boldsymbol{U} = \boldsymbol{u}] = (h^*(u_i) - h^{\mathcal{H}_p}(u_i))^2 + \mathbb{E}[(h_{\mathcal{T}}(U_i) - h^{\mathcal{H}_p}(U_i))^2 \,|\, \boldsymbol{U} = \boldsymbol{u}] \,.$$

Therefore,

$$\begin{aligned}
\frac{1}{n}\sum_{i=1}^n \mathbb{E}[(h^*(U_i) - h_{\mathcal{T}}(U_i))^2 \,|\, U_i = u_i] &= \frac{\|\boldsymbol{h}^*(\boldsymbol{u}) - \boldsymbol{h}^{\mathcal{H}_p}(\boldsymbol{u})\|^2}{n} + \frac{\sum_i \mathbb{V}\mathrm{ar}_{\mathbf{X}}[\boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}]}{n} \\
&= \frac{\|(\mathbf{I} - \mathbf{X}\mathbf{X}^+)\boldsymbol{h}^*(\boldsymbol{u})\|^2}{n} + \frac{\mathrm{tr}(\mathbb{V}\mathrm{ar}_{\mathbf{X}}[\mathbf{X}\widehat{\boldsymbol{\beta}}])}{n}
\end{aligned}$$

Using the idempotent property of $\mathbf{X}\mathbf{X}^+$ and $\mathbf{I} - \mathbf{X}\mathbf{X}^+$:

$$\mathbf{X}\mathbf{X}^+(\mathbf{X}\mathbf{X}^+) = \mathbf{X}\mathbf{X}^+, \qquad (\mathbf{I} - \mathbf{X}\mathbf{X}^+)(\mathbf{I} - \mathbf{X}\mathbf{X}^+) = \mathbf{I} - \mathbf{X}\mathbf{X}^+,$$

we can simplify $\|(\mathbf{I} - \mathbf{X}\mathbf{X}^+)\boldsymbol{h}^*(\boldsymbol{u})\|^2 = \|\boldsymbol{h}^*(\boldsymbol{u})\|^2 - \|\mathbf{X}\mathbf{X}^+\boldsymbol{h}^*(\boldsymbol{u})\|^2$. As in Example 2.3, we can show that $\mathrm{tr}(\mathbb{V}\mathrm{ar}_{\mathbf{X}}[\mathbf{X}\widehat{\boldsymbol{\beta}}]) = \mathrm{tr}(\mathbb{V}\mathrm{ar}_{\mathbf{X}}[\mathbf{X}\mathbf{X}^+\boldsymbol{Y}]) = \ell^* p$. Therefore, the estimation error is $\ell^* p/n$, the approximation error is $\frac{\|\boldsymbol{h}^*(\boldsymbol{u})\|^2 - \|\mathbf{X}\mathbf{X}^+\boldsymbol{h}^*(\boldsymbol{u})\|^2}{n}$, and the irreducible error is ℓ^*.

Finally, note that $\mathbb{V}\mathrm{ar}_{\mathbf{X}}[\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}] = \mathbb{V}\mathrm{ar}_{\mathbf{X}}[\mathbf{X}^+\boldsymbol{Y}] = \mathbf{X}^+\mathbb{V}\mathrm{ar}_{\mathbf{X}}[\boldsymbol{Y}](\mathbf{X}^+)^\top = \ell^*\mathbf{X}^+(\mathbf{X}^+)^\top$. Also, $\mathbb{E}[\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}] = \mathbf{X}^+\boldsymbol{h}^*(\boldsymbol{u}) - \boldsymbol{\beta}$. Therefore, an application of Theorem C.2 yields:

$$\mathbb{E}_{\mathbf{X}}\,(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta})^\top \mathbf{H}_p(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}) = \ell^*\mathrm{tr}(\mathbf{X}^+(\mathbf{X}^+)^\top\mathbf{H}_p) + (\mathbf{X}^+\boldsymbol{h}^*(\boldsymbol{u}) - \boldsymbol{\beta})^\top\mathbf{H}_p(\mathbf{X}^+\boldsymbol{h}^*(\boldsymbol{u}) - \boldsymbol{\beta}) .$$

10. Consider the setting of the polynomial regression in Example 2.2. Use Theorem C.19 to prove that

$$\sqrt{n}(\widehat{\boldsymbol{\beta}}_n - \boldsymbol{\beta}_p) \xrightarrow{\mathrm{d}} \mathsf{N}(\mathbf{0}, \ell^*\mathbf{H}_p^{-1} + \mathbf{H}_p^{-1}\mathbf{M}_p\mathbf{H}_p^{-1}) ,$$

where $\mathbf{M}_p := \mathbb{E}[\boldsymbol{X}\boldsymbol{X}^\top(g^*(\boldsymbol{X}) - g^{\mathcal{G}_p}(\boldsymbol{X}))^2]$ is the matrix with (i, j)-th entry:

$$\int_0^1 u^{i+j-2}(h^{\mathcal{H}_p}(u) - h^*(u))^2 \mathrm{d}u ,$$

and $\mathbf{H}_p^{-1}$ is the $p \times p$ *inverse Hilbert matrix* with (i, j)-th entry:

$$(-1)^{i+j}(i + j - 1)\binom{p+i-1}{p-j}\binom{p+j-1}{p-i}\binom{i+j-2}{i-1}^2 .$$

Observe that $\mathbf{M}_p = \mathbf{0}$ for $p \geqslant 4$, so that the matrix $\mathbf{M}_p$ term is due to choosing a restrictive class $\mathcal{G}_p$ that does not contain the true prediction function.

Solution:

11. In Example 2.2 we saw that the statistical error can be expressed, see (2.20), as

$$\int_0^1 \left([1, \ldots, u^{p-1}](\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}_p)\right)^2 \mathrm{d}u = (\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}_p)^\top\mathbf{H}_p(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}_p) .$$

By Problem 10 the random vector $\boldsymbol{Z}_n := \sqrt{n}(\widehat{\boldsymbol{\beta}}_n - \boldsymbol{\beta}_p)$ has, for large n, approximately a multivariate normal distribution with mean vector $\mathbf{0}$ and covariance matrix $\mathbf{V} := \ell^*\mathbf{H}_p^{-1} + \mathbf{H}_p^{-1}\mathbf{M}_p\mathbf{H}_p^{-1}$. Use Theorem C.2 to show that for large n the *expected* statistical error is

$$\mathbb{E}\,(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}_p)^\top\mathbf{H}_p(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}_p) \simeq \frac{\ell^* p}{n} + \frac{\mathrm{tr}(\mathbf{M}_p\mathbf{H}_p^{-1})}{n}, \quad n \to \infty .$$

Plot this large-sample approximation of the expected statistical error and compare it with the outcome of the statistical error.

Solution: By Theorem C.2 we have that

$$\begin{aligned} n\mathbb{E}\,(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}_p)^\top\mathbf{H}_p(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}_p) = \mathbb{E}\boldsymbol{Z}_n^\top\mathbf{H}_p\boldsymbol{Z}_n &= \mathrm{tr}(\mathbf{H}_p\mathbf{V}) \\ &= \ell^*\mathrm{tr}(\mathbf{H}_p\mathbf{H}_p^{-1}) + \mathrm{tr}(\mathbf{H}_p\mathbf{H}_p^{-1}\mathbf{M}_p\mathbf{H}_p^{-1}) \\ &= \ell^* p + \mathrm{tr}(\mathbf{H}_p^{-1}\mathbf{M}_p), \end{aligned}$$

whence the result follows. The plot looks like the following.

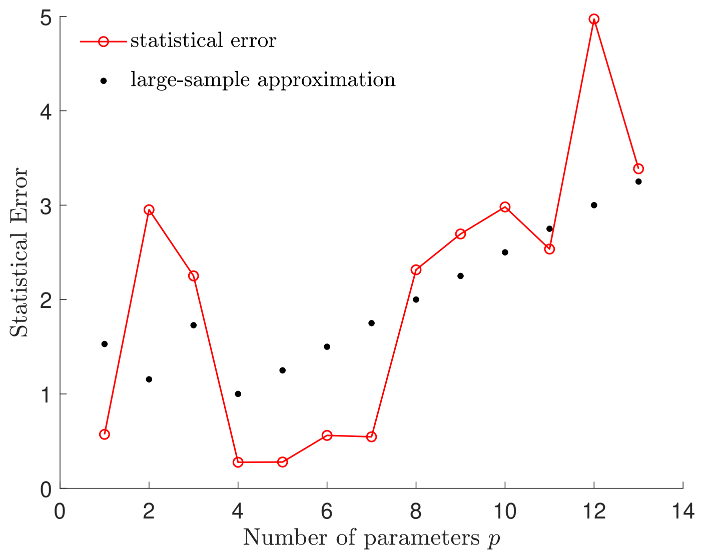


Figure 2.1: The statistical error as a function of the number of parameters p of the model. Superimposed on the figure is the large-sample approximation (2.54).

12. Consider again Example 2.2. The result in (2.53) suggests that $\mathbb{E}\widehat{\boldsymbol{\beta}} \to \boldsymbol{\beta}_p$ as $n \to \infty$, where $\boldsymbol{\beta}_p$ is the solution in the class $\mathcal{G}_p$ given in (2.18). Thus, the large-sample approximation of the pointwise bias of the learner $g_{\mathcal{T}}^{\mathcal{G}_p}(\boldsymbol{x}) = \boldsymbol{x}^\top\widehat{\boldsymbol{\beta}}$ at $\boldsymbol{x} = [1,\dots,u^{p-1}]^\top$ is

$$\mathbb{E}g_{\mathcal{T}}^{\mathcal{G}_p}(\boldsymbol{x}) - g^*(\boldsymbol{x}) \simeq [1,\dots,u^{p-1}]\boldsymbol{\beta}_p - [1,u,u^2,u^3]\boldsymbol{\beta}^*, \quad n \to \infty.$$

Use Python to reproduce Figure 2.17, which shows the (large-sample) pointwise squared bias of the learner for $p \in \{1,2,3\}$. Note how the bias is larger near the end points $u = 0$ and $u = 1$. Explain why the areas under the curves correspond to the approximation errors.

Solution:

13. For our running Example 2.2 we can use (2.53) to derive a large-sample approximation of the pointwise variance of the learner $g_{\mathcal{T}}(\boldsymbol{x}) = \boldsymbol{x}^\top\widehat{\boldsymbol{\beta}}_n$. In particular, show that for large n

$$\mathbb{V}\mathrm{ar}[g_{\mathcal{T}}(\boldsymbol{x})] \simeq \frac{\ell^*\boldsymbol{x}^\top\mathbf{H}_p^{-1}\boldsymbol{x}}{n} + \frac{\boldsymbol{x}^\top\mathbf{H}_p^{-1}\mathbf{M}_p\mathbf{H}_p^{-1}\boldsymbol{x}}{n}, \quad n \to \infty.$$

Figure 2.18 shows this (large-sample) variance of the learner for different values of the predictor u and model index p. Observe that the variance ultimately increases in p and that it is smaller at $u = 1/2$ than closer to the endpoints $u = 0$ or $u = 1$. Since the bias is also larger near the end points, we deduce that the pointwise mean squared error (2.21) is larger near the end points of the interval $[0,1]$ than near its middle. In other words, the error is much smaller in the center of the data cloud than near its periphery.

Solution: By Problem 10 the random vector $\sqrt{n}(\widehat{\boldsymbol{\beta}}_n - \boldsymbol{\beta}_p)$ has, for large n, approximately a multivariate normal distribution with mean vector $\mathbf{0}$ and covariance matrix $\mathbf{V} := \ell^*\mathbf{H}_p^{-1} + \mathbf{H}_p^{-1}\mathbf{M}_p\mathbf{H}_p^{-1}$. So, $\widehat{\boldsymbol{\beta}}$ has approximately a multivariate normal distribution with mean vector

$\boldsymbol{\beta}_p$ and covariance matrix $\mathbf{V}/n$. It follows that (asymptotically),

$$\mathbb{V}\mathrm{ar}[g_{\mathcal{T}}(\boldsymbol{x})] = \mathbb{V}\mathrm{ar}[\boldsymbol{x}^\top \widehat{\boldsymbol{\beta}}] = \frac{1}{n}\boldsymbol{x}^\top \mathbf{V}\boldsymbol{x},$$

which yields the desired result.

14. Let $h : \boldsymbol{x} \mapsto \mathbb{R}$ be a convex function and let $\boldsymbol{X}$ be a random variable. Use the subgradient definition of convexity to prove *Jensen's inequality*:

$$\mathbb{E}h(\boldsymbol{X}) \geqslant h(\mathbb{E}\boldsymbol{X}) \ .$$

Solution:

15. Using Jensen's inequality, show that the Kullback–Leibler divergence between probability densities f and g is always positive; that is,

$$\mathbb{E}\ln\frac{f(\boldsymbol{X})}{g(\boldsymbol{X})} \geqslant 0,$$

where $\boldsymbol{X} \sim f$.

Solution: The function $-\ln$ is convex and so by Jensen's inequality we have

$$\mathbb{E}\ln\frac{f(\boldsymbol{X})}{g(\boldsymbol{X})} = \mathbb{E} - \ln\frac{g(\boldsymbol{X})}{f(\boldsymbol{X})} \geqslant -\ln\left(\mathbb{E}\frac{g(\boldsymbol{X})}{f(\boldsymbol{X})}\right) = -\ln\int\frac{g(\boldsymbol{x})}{f(\boldsymbol{x})}f(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = -\ln 1 = 0.$$

16. The purpose of this exercise is to prove rigorously the following *Vapnik–Chernovenkis bound*: for any *finite* class $\mathcal{G}$ (containing only a finite number $|\mathcal{G}|$ of possible functions) and a general *bounded* loss function, $l \leqslant \text{Loss} \leqslant u$, the expected statistical error is bounded from above according to:

$$\mathbb{E}\ell(g^{\mathcal{G}}_{\mathcal{T}_n}) - \ell(g^{\mathcal{G}}) \leqslant \frac{(u-l)\sqrt{2\ln(2|\mathcal{G}|)}}{\sqrt{n}} \ . \tag{2.57}$$

Note how this bound conveniently does not depend on the distribution of the training set $\mathcal{T}_n$ (which is typically unknown), but only on the complexity (i.e., cardinality) of the class $\mathcal{G}$. We can break up the proof of (2.57) into the following four parts:

(a) For a general function class $\mathcal{G}$, training set $\mathcal{T}$, risk function ℓ, and training loss $\ell_{\mathcal{T}}$, we have, by definition, $\ell(g^{\mathcal{G}}) \leqslant \ell(g)$ and $\ell_{\mathcal{T}}(g^{\mathcal{G}}_{\mathcal{T}}) \leqslant \ell_{\mathcal{T}}(g)$ for all $g \in \mathcal{G}$. Show that

$$\ell(g^{\mathcal{G}}_{\mathcal{T}}) - \ell(g^{\mathcal{G}}) \leqslant \sup_{g\in\mathcal{G}}|\ell_{\mathcal{T}}(g) - \ell(g)| + \ell_{\mathcal{T}}(g^{\mathcal{G}}) - \ell(g^{\mathcal{G}}).$$

Since $\mathbb{E}\ell_{\mathcal{T}}(g) = \mathbb{E}\ell(g)$, we obtain, after taking expectations on both sides of the inequality above:

$$\mathbb{E}[\ell(g^{\mathcal{G}}_{\mathcal{T}})] - \ell(g^{\mathcal{G}}) \leqslant \mathbb{E}\sup_{g\in\mathcal{G}}|\ell_{\mathcal{T}}(g) - \ell(g)|.$$

Solution:

(b) If X is a zero-mean random variable taking values in the interval $[l, u]$, then *Hoeffding's inequality* states that the moment generating function satisfies

$$\mathbb{E}\, \mathrm{e}^{tX} \leqslant \exp\left(\frac{t^2(u-l)^2}{8}\right), \quad t \in \mathbb{R}. \tag{2.58}$$

Prove this result by using the fact that the line segment joining points $(l, \exp(tl))$ and $(u, \exp(tu))$ bounds the convex function $x \mapsto \exp(tx)$ for $x \in [l, u]$; that is:

$$\mathrm{e}^{tx} \leqslant \mathrm{e}^{tl}\frac{u-x}{u-l} + \mathrm{e}^{tu}\frac{x-l}{u-l}, \quad x \in [l, u] .$$

Solution:

(c) Let $Z_1, \ldots, Z_n$ be (possibly dependent and non-identically distributed) zero-mean random variables with moment generating functions that satisfy $\mathbb{E}\exp(tZ_k) \leqslant \exp(t^2\eta^2/2)$ for all k and some parameter η. Use Jensen's inequality (2.56) to prove that for any $t > 0$,

$$\mathbb{E}\max_k Z_k = \frac{1}{t}\mathbb{E}\ln\max_k \mathrm{e}^{tZ_k} \leqslant \frac{1}{t}\ln(n) + \frac{t\eta^2}{2}.$$

From this derive that

$$\mathbb{E}\max_k Z_k \leqslant \eta\sqrt{2\ln(n)} .$$

Finally, show that this last inequality implies that

$$\mathbb{E}\max_k |Z_k| \leqslant \eta\sqrt{2\ln(2n)} . \tag{2.1}$$

Solution:

(d) Returning to the objective of this exercise, denote the elements of $\mathcal{G}$ by $g_1, \ldots, g_{|\mathcal{G}|}$, and let $Z_k = \ell_{\tau_n}(g_k) - \ell(g_k)$. By part (a) it is sufficient to bound $\mathbb{E}\max_k |Z_k|$. Show that the $\{Z_k\}$ satisfy the conditions of (c) with $\eta = (u-l)/\sqrt{n}$. For this you will need to apply part (b) to a random variable $\mathrm{Loss}(g(\boldsymbol{X}), Y) - \ell(g)$, where $(\boldsymbol{X}, Y)$ is a generic data point. Now complete the proof of (2.57).

Solution:

17. Consider the problem in Exercise 16a above. Show that

$$|\ell_{\mathcal{T}}(g_{\mathcal{T}}^{\mathcal{G}}) - \ell(g^{\mathcal{G}})| \leqslant 2\sup_{g\in\mathcal{G}}|\ell_{\mathcal{T}}(g) - \ell(g)| + \ell_{\mathcal{T}}(g^{\mathcal{G}}) - \ell(g^{\mathcal{G}}).$$

From this, conclude:

$$\mathbb{E}\,|\ell_{\mathcal{T}}(g_{\mathcal{T}}^{\mathcal{G}}) - \ell(g^{\mathcal{G}})| \leqslant 2\mathbb{E}\sup_{g\in\mathcal{G}}|\ell_{\mathcal{T}}(g) - \ell(g)| .$$

The last bound allows us to assess how close the training loss $\ell_{\mathcal{T}}(g_{\mathcal{T}}^{\mathcal{G}})$ is to the optimal risk $\ell(g^{\mathcal{G}})$ within class $\mathcal{G}$.

Solution: By definition, we have $\ell(g^{\mathcal{G}}) \leqslant \ell(g)$ and $\ell_{\mathcal{T}}(g_{\mathcal{T}}^{\mathcal{G}}) \leqslant \ell_{\mathcal{T}}(g)$ for any $g \in \mathcal{G}$. We have

$$\begin{aligned}
|\ell_{\mathcal{T}}(g_{\mathcal{T}}^{\mathcal{G}}) - \ell(g^{\mathcal{G}})| &\leqslant |\ell_{\mathcal{T}}(g_{\mathcal{T}}^{\mathcal{G}}) - \ell(g_{\mathcal{T}}^{\mathcal{G}})| + \ell(g_{\mathcal{T}}^{\mathcal{G}}) - \ell(g^{\mathcal{G}}) \\
&\leqslant \sup_{g\in\mathcal{G}} |\ell_{\mathcal{T}}(g) - \ell(g)| + \ell(g_{\mathcal{T}}^{\mathcal{G}}) - \ell_{\mathcal{T}}(g_{\mathcal{T}}^{\mathcal{G}}) + \ell_{\mathcal{T}}(g^{\mathcal{G}}) - \ell(g^{\mathcal{G}}) \\
&\leqslant \sup_{g\in\mathcal{G}} |\ell_{\mathcal{T}}(g) - \ell(g)| + |\ell(g_{\mathcal{T}}^{\mathcal{G}}) - \ell_{\mathcal{T}}(g_{\mathcal{T}}^{\mathcal{G}})| + \ell_{\mathcal{T}}(g^{\mathcal{G}}) - \ell(g^{\mathcal{G}}) \\
&\leqslant 2 \sup_{g\in\mathcal{G}} |\ell_{\mathcal{T}}(g) - \ell(g)| + \ell_{\mathcal{T}}(g^{\mathcal{G}}) - \ell(g^{\mathcal{G}}) \,.
\end{aligned}$$

Hence, taking expectation yields

$$\mathbb{E}|\ell_{\mathcal{T}}(g_{\mathcal{T}}^{\mathcal{G}}) - \ell(g^{\mathcal{G}})| \leqslant 2\mathbb{E} \sup_{g\in\mathcal{G}} |\ell_{\mathcal{T}}(g) - \ell(g)| \,.$$

18. Show that for the normal linear model $\boldsymbol{Y} \sim \mathcal{N}(\mathbf{X}\boldsymbol{\beta}, \sigma^2\mathbf{I}_n)$, the maximum likelihood estimator of σ^2 is identical to the method of moments estimator (2.37).

Solution:

19. Let $X \sim \mathsf{Gamma}(\alpha, \lambda)$. Show that the pdf of $Z = 1/X$ is equal to

$$\frac{\lambda^\alpha (z)^{-\alpha-1} \mathrm{e}^{-\lambda (z)^{-1}}}{\Gamma(\alpha)}, \quad z > 0 \,.$$

Solution: This is a consequence of the transformation rule (C.23): for any $z > 0$ we have

$$f_Z(z) = \frac{f_X(x)}{(1/x)^2} = \frac{\lambda^\alpha x^{\alpha-1}\mathrm{e}^{-\lambda x}}{\Gamma(\alpha)/x^2} = \frac{\lambda^\alpha z^{-\alpha+1}\mathrm{e}^{-\lambda/z}}{\Gamma(\alpha) z^2},$$

which gives the desired result.

20. Consider the sequence $w_0, w_1, \ldots$, where $w_0 = g(\boldsymbol{\theta})$ is a non-degenerate initial guess and $w_t(\boldsymbol{\theta}) \propto w_{t-1}(\boldsymbol{\theta}) g(\tau \,|\, \boldsymbol{\theta}),\ t > 1$. We assume that $g(\tau \,|\, \boldsymbol{\theta})$ is not the constant function (with respect to $\boldsymbol{\theta}$) and that the maximum likelihood value

$$g(\tau \,|\, \widehat{\boldsymbol{\theta}}) = \max_{\boldsymbol{\theta}} g(\tau \,|\, \boldsymbol{\theta}) < \infty$$

exists (is bounded). Let

$$l_t := \int g(\tau \,|\, \boldsymbol{\theta}) w_t(\boldsymbol{\theta}) \,\mathrm{d}\boldsymbol{\theta} \,.$$

Show that $\{l_t\}$ is a strictly increasing and bounded sequence. Hence, conclude that its limit is $g(\tau \,|\, \widehat{\boldsymbol{\theta}})$.

Solution:

21. Consider the Bayesian model for $\tau = \{x_1, \dots, x_n\}$ with likelihood $g(\tau \,|\, \mu)$ such that $(X_1, \dots, X_n \,|\, \mu) \sim_{\text{iid}} \mathsf{N}(\mu, 1)$ and prior pdf $g(\mu)$ such that $\mu \sim \mathsf{N}(\nu, 1)$ for some hyperparameter ν. Define a sequence of densities $w_t(\mu), t \geqslant 2$ via $w_t(\mu) \propto w_{t-1}(\mu)\, g(\tau \,|\, \mu)$, starting with $w_1(\mu) = g(\mu)$. Let a_t and b_t denote the mean and precision of μ under the posterior $g_t(\mu \,|\, \tau) \propto g(\tau \,|\, \mu) w_t(\mu)$. Show that $g_t(\mu \,|\, \tau)$ is a normal density with precision $b_t = b_{t-1} + n,\ b_0 = 1$ and mean $a_t = (1 - \omega_t) a_{t-1} + \omega_t \bar{x}_n,\ a_0 = \nu$, where $\omega_t := n/(b_{t-1} + n)$. Hence, deduce that $g_t(\mu \,|\, \tau)$ converges to a degenerate density with a point-mass at $\bar{x}_n$.

Solution: We use induction. The claim $(a_m, b_m) = (\nu, 1)$ is true for $m = 0$, by definition. Assume that the formula is true for $m = t-1$, that is, $g_{t-1}(\mu \,|\, \tau)$ is the pdf of $\mathsf{N}(a_{t-1}, b_{t-1})$. To show that the formula is true for $m = t$, we use the result **(2.45)** in Example **2.7** to calculate the mean and variance of $g_t(\mu \,|\, \tau)$:

$$\begin{aligned} a_t &= \frac{n}{b_{t-1} + n} \bar{x}_n + \frac{b_{t-1}}{b_{t-1} + n} a_{t-1} \\ \frac{1}{b_t} &= \frac{1}{b_{t-1} + n}\ . \end{aligned}$$

Hence, $g_t(\mu \,|\, \tau)$ is a normal density with limiting mean $a_t \to \bar{x}_n$ and limiting variance $b_t^{-1} \to 0$.

22. Consider again Example **2.8**, where we have a normal model with improper prior $g(\boldsymbol{\theta}) = g(\mu, \sigma^2) \propto 1/\sigma^2$. Show that the prior predictive pdf is an improper density $g(x) \propto 1$, but that the posterior predictive density is

$$g(x \,|\, \tau) \propto \left(1 + \frac{(x - \bar{x}_n)^2}{(n+1) S_n^2}\right)^{-n/2} .$$

Deduce that $\frac{X - \bar{x}_n}{S_n \sqrt{(n+1)/(n-1)}} \sim \mathsf{t}_{n-1}$.

Solution:

23. Assuming that $\boldsymbol{X}_1, \dots, \boldsymbol{X}_n \overset{\text{iid}}{\sim} f$, show that **(2.48)** holds and that $\ell_n^* = -n \mathbb{E} \ln f(\boldsymbol{X})$.

Solution: By independence, we have

$$\begin{aligned} -\ell_n^* &= \int f(\tau_n) \ln f(\tau_n) \mathrm{d}\tau_n = \mathbb{E} \ln f(\mathcal{T}_n) \\ &= \mathbb{E} \ln \prod_{i=1}^n f(\boldsymbol{X}_i) = \mathbb{E} \sum_{i=1}^n \ln f(\boldsymbol{X}_i) \\ &= \sum_{i=1}^n \mathbb{E} \ln f(\boldsymbol{X}_i) = n \mathbb{E} \ln f(\boldsymbol{X})\ . \end{aligned}$$

To show **(2.48)**, note that

$$\ln g_{\tau_n}(\tau_n') = \ln g(\tau_n' \,|\, \tau_n) = \ln g(\tau_n', \tau_n) - \ln g(\tau_n)\ .$$

Therefore, by independence

$$\begin{aligned} \mathbb{E} \ell(g_{\mathcal{T}_n}) = -\mathbb{E} \int f(\tau_n') \ln g(\tau_n' \,|\, \mathcal{T}_n) \mathrm{d}\tau_n' &= -\mathbb{E} \ln g(\mathcal{T}_n' \,|\, \mathcal{T}_n) \\ &= -\mathbb{E} \ln g(\mathcal{T}_n', \mathcal{T}_n) + \mathbb{E} \ln g(\mathcal{T}_n) \\ &= -\mathbb{E} \ln g(\mathcal{T}_{2n}) + \mathbb{E} \ln g(\mathcal{T}_n)\ . \end{aligned}$$

24. Suppose that $\tau = \{x_1, \ldots, x_n\}$ are observations of iid continuous and strictly positive random variables, and that there are two possible models for their pdf. The first model $p = 1$ is

$$g(x \,|\, \theta, p = 1) = \theta \exp(-\theta x)$$

and the second $p = 2$ is

$$g(x \,|\, \theta, p = 2) = \left(\frac{2\theta}{\pi}\right)^{1/2} \exp\left(-\frac{\theta x^2}{2}\right).$$

For both models, assume that the prior for θ is a gamma density

$$g(\theta) = \frac{b^t}{\Gamma(t)} \theta^{t-1} \exp(-b\theta),$$

with the same hyperparameters b and t. Find a formula for the Bayes factor, $g(\tau \,|\, p = 1)/g(\tau \,|\, p = 2)$, for comparing these models.

Solution:

25. Suppose that we have a total of m possible models with prior probabilities $g(p), p = 1, \ldots, m$. Show that the posterior probability of model $g(p \,|\, \tau)$ can be expressed in terms of all the $p(p-1)$ Bayes factors:

$$g(p = i \,|\, \tau) = \left(1 + \sum_{j \neq i} \frac{g(p = j)}{g(p = i)} B_{j \,|\, i}\right)^{-1}.$$

Solution: We have

$$\begin{aligned}
\left(1 + \sum_{j \neq i} \frac{g(p = j)}{g(p = i)} B_{j \,|\, i}\right)^{-1} &= \frac{g(p = i)}{g(p = i) + \sum_{j \neq i} g(p = j) B_{j \,|\, i}} \\
&= \frac{g(p = i) g(\tau \,|\, p = i)}{g(p = i) + \sum_{j \neq i} g(p = j) g(\tau \,|\, p = j)} \\
&= g(p = i \,|\, \tau).
\end{aligned}$$

26. Given the data $\tau = \{x_1, \ldots, x_n\}$, suppose that we use the Bayesian likelihood $(X \,|\, \boldsymbol{\theta}) \sim \mathsf{N}(\mu, \sigma^2)$ with parameter $\boldsymbol{\theta} = (\mu, \sigma^2)^\top$ and wish to compare the following two nested models.

(a) Model $p = 1$, where $\sigma^2 = \sigma_0^2$ is known and this is incorporated via the prior

$$g(\boldsymbol{\theta} \,|\, p = 1) = g(\mu \,|\, \sigma^2, p = 1) g(\sigma^2 \,|\, p = 1) = \frac{1}{\sqrt{2\pi}\sigma} \mathrm{e}^{-\frac{(\mu - x_0)^2}{2\sigma^2}} \times \delta(\sigma^2 - \sigma_0^2).$$

(b) Model $p = 2$, where both mean and variance are unknown with prior

$$g(\boldsymbol{\theta} \,|\, p = 2) = g(\mu \,|\, \sigma^2) g(\sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \mathrm{e}^{-\frac{(\mu - x_0)^2}{2\sigma^2}} \times \frac{b^t (\sigma^2)^{-t-1} \mathrm{e}^{-b/\sigma^2}}{\Gamma(t)}.$$

Show that the prior $g(\boldsymbol{\theta}\,|\,p = 1)$ can be viewed as the limit of the prior $g(\boldsymbol{\theta}\,|\,p = 2)$ when $t \to \infty$ and $b = t\sigma_0^2$. Hence, conclude that

$$g(\tau\,|\,p = 1) = \lim_{\substack{t\to\infty \\ b=t\sigma_0^2}} g(\tau\,|\,p = 2)$$

and use this result to calculate $B_{1\,|\,2}$. Check that the formula for $B_{1\,|\,2}$ agrees with the Savage–Dickey density ratio:

$$\frac{g(\tau\,|\,p = 1)}{g(\tau\,|\,p = 2)} = \frac{g(\sigma^2 = \sigma_0^2\,|\,\tau)}{g(\sigma^2 = \sigma_0^2)} ,$$

where $g(\sigma^2\,|\,\tau)$ and $g(\sigma^2)$ are the posterior and prior, respectively, under model $p = 2$.

Solution:

CHAPTER 3

MONTE CARLO METHODS

1. We can modify the Box–Muller method in Example 3.1 to draw X and Y uniformly on the unit disc: $\{(x, y) \in \mathbb{R}^2 : x^2 + y^2 \leqslant 1\}$ in the following way: Independently draw a radius R and an angle $\Theta \sim \mathcal{U}(0, 2\pi)$, and return $X = R\cos(\Theta)$, $Y = R\sin(\Theta)$. The question is how to draw R.

   (a) Show that the cdf of R is given by $F_R(r) = r^2$ for $0 \leqslant r \leqslant 1$ (with $F_R(r) = 0$ and $F_R(r) = 1$ for $r < 0$ and $r > 1$, respectively).

   Solution: Take a disc D_r of radius $0 \leqslant r \leqslant 1$. We have $F_R(r) = \mathbb{P}[(X, y) \in D_r] = \pi r^2/\pi = r^2$, as required. For $r \neq [0, 1]$ the value of $F_R(r)$ follows from the monotonicity of F_R.

   (b) Explain how to simulate R using the inverse-transform method.

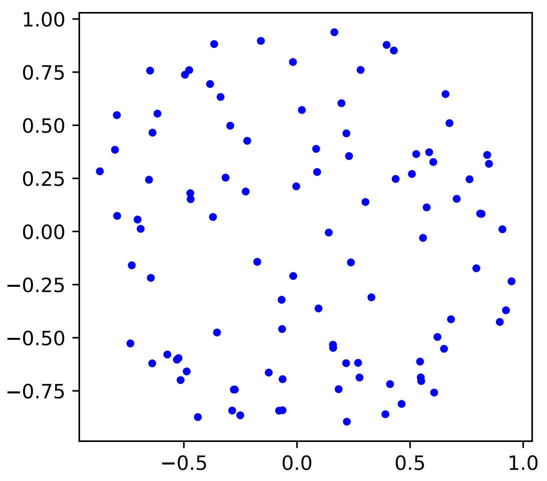
   Solution: For $0 \leqslant r \leqslant 1$, solving $u = F_R(r)$ for r yields the inverse cdf $r = F_R^{-1}(u) = \sqrt{u}$. Thus, we can simulate R using the inverse-transform method by first drawing $U \sim \mathcal{U}(0, 1)$, and then returning $R = \sqrt{U}$.

   (c) Simulate 100 independent draws of $[X, Y]^\top$ according to the method described above.

   Solution: Python code and example output is provided below.

```
import matplotlib.pyplot as plt
import numpy as np
rand = np.random.rand

R, Theta = np.sqrt(rand(100)), 2*np.pi*rand(100)
X, Y = R * np.cos(Theta), R * np.sin(Theta)
plt.plot(X, Y, 'b.')
```



2. A simple acceptance–rejection method to simulate a vector $\boldsymbol{X}$ in the unit d-ball $\{\boldsymbol{x} \in \mathbb{R}^d : \|\boldsymbol{x}\| \leqslant 1\}$ is to first generate $\boldsymbol{X}$ uniformly in the cube $[-1,1]^d$ and then to accept the point only if $\|\boldsymbol{X}\| \leqslant 1$. Determine an analytic expression for the probability of acceptance as a function of d and plot this for $d = 1, \dots, 50$.

   Solution:

3. Let the random variable X have pdf

$$f(x) = \begin{cases} \frac{1}{2}\,x\,, & 0 \leqslant x < 1\,, \\ \frac{1}{2}\,, & 1 \leqslant x \leqslant \frac{5}{2}\,. \end{cases}$$

   Simulate a random variable from $f(x)$, using

   (a) the inverse-transform method,

   Solution: The cdf of X is given by

$$F(x) = \begin{cases} 0 & x < 0\,, \\ \frac{x^2}{4} & 0 \leqslant x < 1\,, \\ \frac{1}{4} + \frac{x-1}{2} & 1 \leqslant x \leqslant 5/2\,, \\ 1 & x > 5/2\,. \end{cases}$$

   Hence, to draw from this distribution using the inverse-transform method, first draw $U \sim \mathcal{U}(0,1)$. If $\frac{1}{4} \leqslant U < 1$, output $X = \frac{1}{2} + 2U$; otherwise, output $X = 2\sqrt{U}$.

   (b) the acceptance–rejection method, using the proposal density

$$g(x) = \frac{8}{25}\,x\,, \quad 0 \leqslant x \leqslant \frac{5}{2}\,.$$

   Solution: The acceptance–rejection method with proposal density $g(x) = 8x/25$ has an acceptance probability of $C = \max_{0 \leqslant x \leqslant 5/2} 25f(x)/(8x) =$

$25f(1)/8 = 25/16$. To draw a random variable $X \sim g$, we can generate $U \sim \mathcal{U}(0,1)$ and return $X = \frac{5}{2}\sqrt{U}$ (this is the inverse-transform way of drawing from g). Based on Algorithm 3.2.4, the acceptance–rejection algorithm is as follows.

Algorithm 3.0.1: Acceptance–Rejection Method for this exercise

```
1 found ← false
2 while not found do
3     Generate $U_1 \sim \mathcal{U}(0,1)$ and let $X = \frac{5}{2}\sqrt{U_1}$.
4     Generate $U_2 \sim \mathcal{U}(0,1)$ independently of $U_1$.
5     $Z \leftarrow \frac{f(X)}{Cg(X)} = \begin{cases} 1, & \text{for } 0 \leqslant X < 1, \\ \frac{1}{X}, & \text{for } 1 \leqslant X \leqslant \frac{5}{2}. \end{cases}$
6     if $U_2 \leqslant Z$ then found ← true
7 return $X$
```

A histogram created from 10,000 random variables with pdf f is given in Figure **??**.

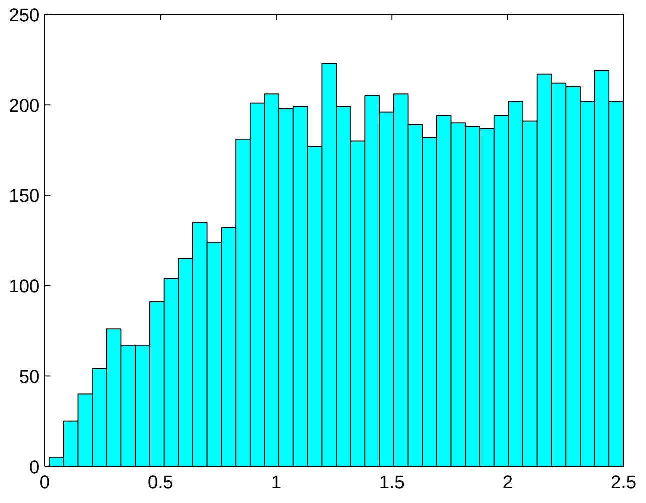


Figure 3.1: Histogram of 10,000 samples from f.

4. Construct generation algorithms for the following distributions:

   (a) The $\mathsf{Weib}(\alpha, \lambda)$ distribution, with cdf $F(x) = 1 - \mathrm{e}^{-(\lambda x)^\alpha}$, $x \geqslant 0$, where $\lambda > 0$ and $\alpha > 0$.

   Solution:

   (b) The $\mathsf{Pareto}(\alpha, \lambda)$ distribution, with pdf $f(x) = \alpha\lambda(1 + \lambda x)^{-(\alpha+1)}$, $x \geqslant 0$, where $\lambda > 0$ and $\alpha > 0$.

   Solution:

5. We wish to sample from the pdf

$$f(x) = x\,\mathrm{e}^{-x}, \quad x \geqslant 0,$$

using acceptance–rejection with the proposal pdf $g(x) = \mathrm{e}^{-x/2}/2$, $x \geqslant 0$.

(a) Find the smallest C for which $Cg(x) \geqslant f(x)$ for all x.

Solution: We require that

$$C \geqslant \frac{f(x)}{g(x)} = \frac{x\,\mathrm{e}^{-x}}{0.5\,\mathrm{e}^{-0.5x}} = 2x\,\mathrm{e}^{-0.5x}, \qquad \text{for all } x \in \mathbb{R}. \tag{3.1}$$

Thus, the smallest C is the maximum of $2x\,\mathrm{e}^{-0.5x}$ over $\mathbb{R}$. We have

$$\frac{\mathrm{d}}{\mathrm{d}x} 2x\,\mathrm{e}^{-0.5x} = \mathrm{e}^{-0.5x}\,(2 - x)\,,$$

and thus a critical point at $x = 2$. The fact that the critical point is a maximum can be verified by observing that the second derivative is negative at $x = 2$. Therefore, $C = f(2)/g(2) = 4\mathrm{e}^{-1}$ is the is smallest C that satisfies (**??**).

(b) What is the efficiency of this acceptance–rejection method?

Solution: The efficiency (acceptance probability) is given by $1/C = \mathrm{e}^1/4 \approx 0.6796$.

6. Let $[X, Y]^\top$ be uniformly distributed on the triangle $(0, 0), (1, 2), (-1, 1)$. Give the distribution of $[U, V]^\top$ defined by the linear transformation

$$\begin{bmatrix} U \\ V \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} X \\ Y \end{bmatrix}.$$

Solution:

7. Explain how to generate a random variable from the *extreme value distribution*, which has cdf

$$F(x) = 1 - \mathrm{e}^{-\exp(\frac{x-\mu}{\sigma})}\,, \quad -\infty < x < \infty, \quad (\sigma > 0),$$

via the inverse-transform method.

Solution: By solving $y = F(x)$ for x, the inverse of the cdf of the extreme value distribution is given by

$$F^{-1}(y) = \mu + \sigma\, \ln(-\ln(1 - y))\,.$$

Hence, to generate X from the extreme value distribution, draw $U \sim \mathcal{U}(0, 1)$ and return $X = \mu + \sigma\,\ln(-\ln U)$. Here, we have used the fact that if $U \sim \mathcal{U}(0, 1)$ then also $1 - U \sim \mathcal{U}(0, 1)$.

8. Write a program that generates and displays 100 random vectors that are uniformly distributed within the ellipse

$$5\,x^2 + 21\,x\,y + 25\,y^2 = 9\,.$$

[Hint: Consider generating uniformly distributed samples within the circle of radius 3 and use the fact that linear transformations preserve uniformity to transform the circle to the given ellipse.]

Solution:

9. Suppose that $X_i \sim \mathsf{Exp}(\lambda_i)$, independently, for all $i = 1, \dots, n$. Let $\mathbf{\Pi} = [\Pi_1, \dots, \Pi_n]^\top$ be the random permutation induced by the ordering $X_{\Pi_1} < X_{\Pi_2} < \cdots < X_{\Pi_n}$, and define $Z_1 := X_{\Pi_1}$ and $Z_j := X_{\Pi_j} - X_{\Pi_{j-1}}$ for $j = 2, \dots, n$.

   (a) Determine an $n \times n$ matrix $\mathbf{A}$ such that $\mathbf{Z} = \mathbf{A}\boldsymbol{X}$ and show that $\det(\mathbf{A}) = 1$.

   Solution: We have that $\mathbf{Z} = \mathbf{A}\boldsymbol{X}$, where

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & \cdots & \cdots & 0 \\ -1 & 1 & 0 & \cdots & 0 \\ 0 & -1 & 1 & 0 & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & -1 & 1 \end{bmatrix}.$$

   Hence,

$$\mathbf{A}^{-1} = \begin{bmatrix} 1 & 0 & \cdots & \cdots & 0 \\ 1 & 1 & 0 & \cdots & 0 \\ 1 & 1 & 1 & 0 & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 1 & \cdots & 1 & 1 & 1 \end{bmatrix}.$$

   with $\det(\mathbf{A}^{-1}) = 1 = \det(\mathbf{A})$, because $\mathbf{A}$ is lower triangular.

   (b) Denote the joint pdf of $\boldsymbol{X}$ and $\mathbf{\Pi}$ as

$$f_{\boldsymbol{X},\mathbf{\Pi}}(\boldsymbol{x}, \boldsymbol{\pi}) = \prod_{i=1}^{n} \lambda_{\pi_i} \exp\left(-\lambda_{\pi_i} x_{\pi_i}\right) \times \mathbb{1}\{x_{\pi_1} < \cdots < x_{\pi_n}\}, \quad \boldsymbol{x} \geqslant \mathbf{0},\ \boldsymbol{\pi} \in \mathcal{P}_n,$$

   where $\mathcal{P}_n$ is the set of all $n!$ permutations of $\{1, \dots, n\}$. Use the multivariate transformation formula (C.22) to show that ☞ 432

$$f_{\mathbf{Z},\mathbf{\Pi}}(\boldsymbol{z}, \boldsymbol{\pi}) = \exp\left(-\sum_{i=1}^{n} z_i \sum_{k \geqslant i} \lambda_{\pi_k}\right) \prod_{i=1}^{n} \lambda_i, \qquad \boldsymbol{z} \geqslant \mathbf{0},\ \boldsymbol{\pi} \in \mathcal{P}_n.$$

   Hence, conclude that the probability mass function of the random permutation $\mathbf{\Pi}$ is:

$$\mathbb{P}[\mathbf{\Pi} = \boldsymbol{\pi}] = \prod_{i=1}^{n} \frac{\lambda_{\pi_i}}{\sum_{k \geqslant i} \lambda_{\pi_k}}, \qquad \boldsymbol{\pi} \in \mathcal{P}_n.$$

   Solution: By the multivariate transformation formula (C.22), we have ☞ 432

$$f_{\boldsymbol{Z},\mathbf{\Pi}}(\boldsymbol{z}, \boldsymbol{\pi}) = \frac{f_{\boldsymbol{X},\mathbf{\Pi}}(\mathbf{A}^{-1}\boldsymbol{z}, \boldsymbol{\pi})}{\det(\mathbf{A})} = \prod_{i=1}^{n} \lambda_{\pi_i} \exp\left(-z_i \sum_{k \geqslant i} \lambda_{\pi_k}\right), \quad \boldsymbol{z} \geqslant \mathbf{0}.$$

   Hence, by integrating/marginalizing the $\boldsymbol{z}$, we obtain $f_{\mathbf{\Pi}}(\boldsymbol{\pi}) = \prod_{i=1}^{n} \frac{\lambda_{\pi_i}}{\sum_{k \geqslant i} \lambda_{\pi_k}}$.

   (c) Write pseudo-code to simulate a *uniform* random permutation $\mathbf{\Pi} \in \mathcal{P}_n$; that is, such that $\mathbb{P}[\mathbf{\Pi} = \boldsymbol{\pi}] = \frac{1}{n!}$, and explain how this uniform random permutation can be used to reshuffle a training set τ_n.

Solution:

```
Algorithm 3.0.2: Simulating a Random Permutation
    input: Length of permutation $n$.
    output: Random permutation $\mathbf{\Pi} \in \mathcal{P}_n$.
1   Draw $U_1, \ldots, U_n \overset{\text{iid}}{\sim} \mathcal{U}(0,1)$ and sort them in ascending order.
2   Let $\mathbf{\Pi}$ be the permutation of $\{1, \ldots, n\}$ that satisfies $U_{\Pi_1} < \cdots < U_{\Pi_n}$.
3   return $\mathbf{\Pi}$
```

10. Consider the Markov chain with transition graph given in Figure **3.17**, starting in state 1.

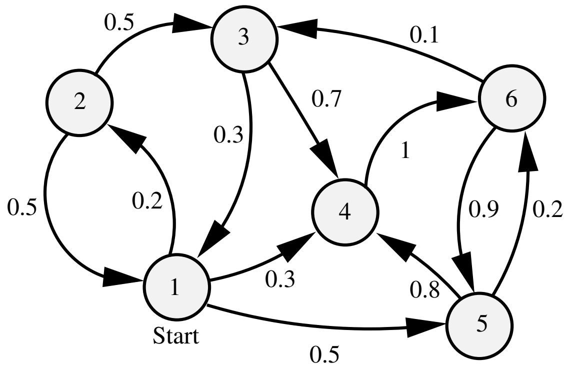


Figure 3.2: The transition graph for the Markov chain $\{X_t, t = 0, 1, 2, \ldots\}$.

   (a) Construct a computer program to simulate the Markov chain, and show a realization for $N = 100$ steps.

   Solution:

   (b) Compute the limiting probabilities that the Markov chain is in state 1,2,. . . ,6, by solving the global balance equations **(C.42)**.

   Solution:

   (c) Verify that the exact limiting probabilities correspond to the average fraction of times that the Markov process visits states 1,2,. . . ,6, for a large number of steps N.

   Solution:

11. As a generalization of Example **C.9**, consider a random walk on an arbitrary undirected connected graph with a finite vertex set $\mathcal{V}$. For any vertex $v \in \mathcal{V}$, let $d(v)$ be the number of neighbors of v — called the *degree* of v. The random walk can jump to each one of the neighbors with probability $1/d(v)$ and can be described by a Markov chain. Show that, if the chain is *aperiodic*, the limiting probability that the chain is in state v is equal to $d(v)/\sum_{v' \in \mathcal{V}} d(v')$.

Solution: This random walk can be viewed as a Markov chain with transition probabilities $P(u, v) = A(u, v)/d(u)$ for $u, v \in \mathcal{V}$, where $A(u, v) = 1$ if there is an edge connecting vertices u and v, and $A(u,v) = 0$ otherwise. Note that $A(u,v) = A(v,u)$ holds, since the graph is undirected.

Defining $c = \sum_{v' \in \mathcal{V}} d(v')$ and $\pi(v) = d(v)/c$, we have for any $u, v \in \mathcal{V}$ that

$$\pi(u)P(u,v) = \frac{d(u)}{c}\,\frac{A(u,v)}{d(u)} = \frac{A(u,v)}{c} = \frac{A(v,u)}{c} = \frac{d(v)}{c}\,\frac{A(v,u)}{d(v)} = \pi(v)P(v,u)\,,$$

so that $\boldsymbol{\pi} = [\pi(v), v \in \mathcal{V}]$ satisfies the detailed balance equations and is therefore stationary for the chain.

Since the chain is irreducible (as the graph is connected) and finite, if the chain is aperiodic, then the stationary distribution $\boldsymbol{\pi}$ is the (unique) limiting distribution.

12. Let $U, V \sim_{\text{iid}} \mathcal{U}(0,1)$. The reason why in Example **3.7** the sample mean and sample median behave very differently, is that $\mathbb{E}[U/V] = \infty$, while the median of U/V is finite. Show this, and compute the median. [Hint: start by determining the cdf of $Z = U/V$ by writing it as an expectation of an indicator function.]

    Solution:

13. Consider the problem of generating samples from $Y \sim \mathsf{Gamma}(2, 10)$.

    (a) Direct simulation: Let $U_1, U_2 \sim_{\text{iid}} \mathcal{U}(0,1)$. Show that $-\ln(U_1)/10 - \ln(U_2)/10 \sim \mathsf{Gamma}(2, 10)$. [Hint: derive the distribution of $-\ln(U_1)/10$ and use Example **C.1**.]

    Solution: Defining $X_1 = -\ln(U_1)/10$ and $X_2 = -\ln(U_2)/10$, we note that X_1 and X_2 are iid. Moreover, we are asked to show that $Z = X_1 + X_2 \sim \mathsf{Gamma}(2, 10)$. Following the hint, we shall determine the cdf F of X_1 as follows. Noting that $X_1 \geqslant 0$, given $x \geqslant 0$, we have

    $$F(x) = \mathbb{P}(X_1 \leqslant x) = \mathbb{P}(-\ln(U_1)/10 \leqslant x) = \mathbb{P}(U_1 \geqslant \exp(-10x)) = 1 - \mathrm{e}^{-10\,x}\,.$$

    We recognise F as a the cdf of an exponential random variable with parameter 10. Thus $X_1, X_2 \sim_{\text{iid}} \mathsf{Exp}(10)$, and further we know that the common mgf of X_1, X_2 is given by $M_X(s) = 10/(10 - s)$ for $s < 10$. Since X_1 and X_2 are iid, the mgf of Z is given by $M_Z(s) = (M_X(s))^2 = (10/(10 - s))^2$ for $s < 10$. Referring to Example C.1, we recognise $M_Z(s)$ as the mgf of a $\mathsf{Gamma}(2, 10)$ random variable. Thus, we have shown that $-\ln(U_1)/10 - \ln(U_2)/10 \sim \mathsf{Gamma}(2, 10)$.

    (b) Simulation via MCMC: Implement an independence sampler to simulate from the $\mathsf{Gamma}(2, 10)$ target pdf

    $$f(x) = 100\,x\,\mathrm{e}^{-10x}\,, \qquad x \geqslant 0\,,$$

    using proposal transition density $q(y \,|\, x) = g(y)$, where $g(y)$ is the pdf of an $\mathsf{Exp}(5)$ random variable. Generate $N = 500$ samples, and compare the true cdf with the empirical cdf of the data.

    Solution: Python code and example output are provided below.

```
import numpy as np
from numpy.random import rand , choice
from statsmodels.distributions.empirical_distribution
    import ECDF
import matplotlib.pyplot as plt

np.random.seed(321)

N = int(5e2)
B = 0 # burn in period

samples = np.zeros(N + B)
samples[0] = 0 # x = 0

for i in range(1,N + B):
    Y = -np.log(rand())/5 # Propose Y ~ g(y)

    # Compute the acceptance probability
    alpha = 100*Y*np.exp(-10*Y)*5*np.exp(-5*samples[i-1])
        /(100*samples[i-1]*np.exp(-10*samples[i-1])*5*np.exp
        (-5*Y))
    alpha = min(alpha,1)

    if rand() <= alpha: # Accept proposal?
        samples[i]=Y
    else:
        samples[i]=samples[i-1]

samples = samples[B:] # take post burn-in samples

# Compute the true cdf
x = np.arange(0, 1, 0.01)
y = 1-np.exp(-10*x)*(1+10*x)

# Compute the ecdf
samp_cdf = ECDF(samples)

# Plot the true cdf and the ecdf
plt.plot(x, y,'b-',label='$F(x)$')
plt.step(samp_cdf.x, samp_cdf.y,'r-',label='$F_n(x)$')
plt.xlabel('$x$')
plt.tight_layout()
plt.legend()
plt.savefig('GammaIndSamp.pdf',format='pdf') # saving as
    pdf
plt.show() # both plots will now be drawn
```

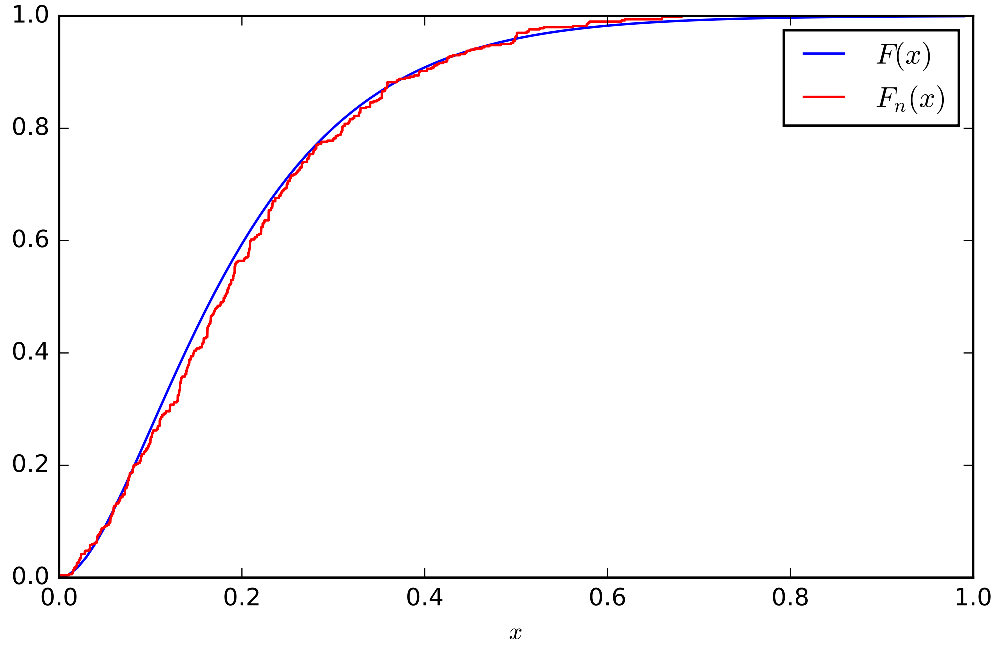


Figure 3.3: Empirical cdf $F_n(x)$ from the MCMC samples vs true cdf $F(x)$.

14. Let $\boldsymbol{X} = [X, Y]^\top$ be a random column vector with a bivariate normal distribution with expectation vector $\boldsymbol{\mu} = [1, 2]^\top$ and covariance matrix

$$\Sigma = \begin{bmatrix} 1 & a \\ a & 4 \end{bmatrix} .$$

   (a) What are the conditional distributions of $(Y \mid X = x)$ and $(X \mid Y = y)$? [Hint: use Theorem C.8.]

   Solution:

   (b) Implement a Gibbs sampler to draw 10^3 samples from the bivariate distribution $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ for $a = 0$, 1, and 1.75, and plot the resulting samples.

   Solution:

15. Here the objective is to sample from the 2-dimensional pdf

$$f(x, y) = c\, \mathrm{e}^{-(xy+x+y)}, \quad x \geqslant 0, \quad y \geqslant 0,$$

for some normalization constant c, using a Gibbs sampler. Let $(X, Y) \sim f$.

   (a) Find the conditional pdf of X given $Y = y$, and the conditional pdf of Y given $X = x$.

   Solution: We have

$$f(x \mid y) \propto \mathrm{e}^{-(xy+x)} = \mathrm{e}^{-(y+1)x},$$

   which, up to a constant of proportionality, is the pdf of an $\mathsf{Exp}(y + 1)$ random variable. Thus,

$$f(x \mid y) = (y + 1)\mathrm{e}^{-(y+1)}, \quad x \geqslant 0$$

and by symmetry

$$f(y \mid x) = (x+1)\mathrm{e}^{-(x+1)}, \quad y \geqslant 0,$$

which is, up to a constant of proportionality, the pdf of an $\mathsf{Exp}(x+1)$ random variable.

(b) Write working Python code that implements the Gibbs sampler and outputs 1000 points that are approximately distributed according to f.

Solution: Python code and example output are provided below.

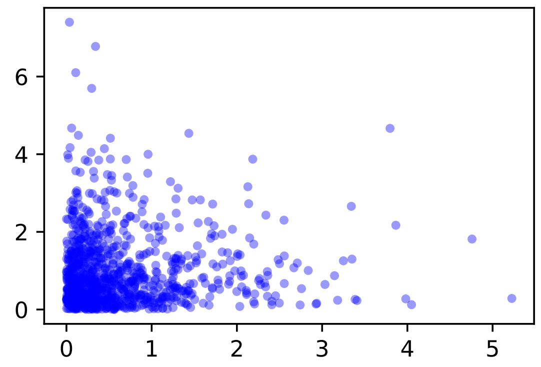
```python
import numpy as np
from numpy.random import rand
import matplotlib.pyplot as plt

n = 1000
b = 500 # burn in period
samples = np.zeros((n + b,2))
samples[0,:] = [1,1]# x = 1, y = 1

for i in range(1,n + b):
    samples[i,0] = -np.log(1-rand())/(samples[i-1,1]+1)
    samples[i,1] = -np.log(1-rand())/(samples[i,1]+1)

samples = samples[b:,:] # take samples after burn-in period

plt.clf()
plt.plot(samples[:,0], samples[:,1], 'b.')
plt.show()
```



(c) Describe how the normalization constant c could be estimated via Monte Carlo simulation, using random variables $X_1, \ldots, X_N, Y_1, \ldots, Y_N \overset{\text{iid}}{\sim} \mathsf{Exp}(1)$.

Solution: Observe that

$$\begin{aligned} c^{-1} &= \int_{\mathbb{R}} \mathrm{e}^{-xy-x-y} \mathbb{1}\{x \geqslant 0\} \mathbb{1}\{y \geqslant 0\}\, \mathrm{d}x\, \mathrm{d}y \\ &= \int_{\mathbb{R}} \mathrm{e}^{-xy} \underbrace{\mathrm{e}^{-x} \mathbb{1}\{x \geqslant 0\}}_{g_X(x)} \underbrace{\mathrm{e}^{-y} \mathbb{1}\{y \geqslant 0\}}_{g_Y(y)}\, \mathrm{d}x\, \mathrm{d}y\,. \end{aligned}$$

Noting that g_X and g_Y are both pdfs of an $\mathsf{Exp}(1)$ random variable, we may write

$$c^{-1} = \mathbb{E}\, \mathrm{e}^{-XY}, \quad X, Y \overset{\text{iid}}{\sim} \mathsf{Exp}(1)\,,$$

and therefore

$$c = \left(\mathbb{E}\, \mathrm{e}^{-XY}\right)^{-1}, \quad X, Y \overset{\text{iid}}{\sim} \mathsf{Exp}(1)\,.$$

Thus, we can estimate c via Monte Carlo simulation via

$$\widehat{c} = \left(\frac{1}{N}\sum_{k=1}^{N} \mathrm{e}^{-X_k Y_k}\right)^{-1}, \quad X_1, \ldots, X_N, Y_1, \ldots, Y_N \overset{\text{iid}}{\sim} \mathsf{Exp}(1)\,.$$

16. We wish to estimate $\mu = \int_{-2}^{2} \mathrm{e}^{-x^2/2}\,\mathrm{d}x = \int H(x) f(x)\,\mathrm{d}x$ via Monte Carlo simulation using two different approaches: (1) defining $H(x) = 4\,\mathrm{e}^{-x^2/2}$ and f the pdf of the $\mathcal{U}[-2, 2]$ distribution and (2) defining $H(x) = \sqrt{2\pi}\,\mathbb{1}\{-2 \leqslant x \leqslant 2\}$ and f the pdf of the $\mathcal{N}(0, 1)$ distribution.

    (a) For both cases estimate μ via the estimator $\widehat{\mu}$

    $$\widehat{\mu} = N^{-1} \sum_{i=1}^{N} H(\boldsymbol{X}_i)\,. \tag{3.34}$$

    Use a sample size of $N = 1000$.

    Solution:

    (b) For both cases estimate the relative error κ of $\widehat{\mu}$ using $N = 100$.

    Solution:

    (c) Give a 95% confidence interval for μ for both cases using $N = 100$.

    Solution:

    (d) From part (b), assess how large N should be such that the relative width of the confidence interval is less than 0.01, and carry out the simulation with this N. Compare the result with the true value of μ.

    Solution:

17. Consider estimation of the tail probability $\mu = \mathbb{P}[X \geqslant \gamma]$ of some random variable X, where γ is large. The crude Monte Carlo estimator of μ is

    $$\widehat{\mu} = \frac{1}{N}\sum_{i=1}^{N} Z_i, \tag{3.35}$$

    where $X_1, \ldots, X_N$ is are iid copies of X and $Z_i = \mathbb{1}\{X_i \geqslant \gamma\}$, $i = 1, \ldots, N$.

    (a) Show that $\widehat{\mu}$ is unbiased; that is, $\mathbb{E}\,\widehat{\mu} = \mu$.

    Solution: Using the properties of expectation, and noting that $Z_1, \ldots, Z_N \overset{\text{iid}}{\sim} \mathsf{Ber}(\mathbb{P}[X \geqslant \gamma])$, we have

    $$\mathbb{E}\widehat{\mu} = \mathbb{E}\frac{1}{N}\sum_{i=1}^{N} Z_i = \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}\, Z_i = \frac{1}{N} N\, \mathbb{E}\, Z_1 = \mathbb{E}\, Z_1 = \mathbb{P}[X \geqslant \gamma] = \mu\,.$$

(b) Express the relative error of $\widehat{\mu}$, i.e.,

$$\mathrm{RE} = \frac{\sqrt{\mathbb{V}\mathrm{ar}\,\widehat{\mu}}}{\mathbb{E}\widehat{\mu}},$$

in terms of N and μ.

Solution: From the unbiasedness of $\widehat{\mu}$ and the fact that $Z_1, \ldots, Z_N \overset{\text{iid}}{\sim} \mathsf{Ber}(\mu)$, we have

$$\mathrm{RE} = \frac{\sqrt{\mathbb{V}\mathrm{ar}\left(\frac{1}{N}\sum_{i=1}^{N} Z_i\right)}}{\mathbb{E}\widehat{\mu}} = \frac{\sqrt{\mu(1-\mu)}}{\mu\sqrt{N}} . \tag{3.2}$$

(c) Explain how to estimate the relative error of $\widehat{\mu}$ from outcomes $x_1, \ldots, x_N$ of $X_1, \ldots, X_N$, and how to construct a 95% confidence interval for μ.

Solution: An estimator of the relative error of $\widehat{\mu}$ is obtained by replacing all occurrences of μ in (**??**) with $\widehat{\mu}$, i.e.,

$$\widehat{\mathrm{RE}} = \frac{\sqrt{\widehat{\mu}(1-\widehat{\mu})}}{\widehat{\mu}\sqrt{N}} .$$

Thus, an *estimate* from outcomes $x_1, \ldots, x_N$ is given by replacing all occurrences $\widehat{\mu}$ in the above equation with $\frac{1}{N}\sum_{i=1}^{N} \mathbb{1}\{x_i \geqslant \gamma\}$. We can construct a 95% confidence interval for μ via the normal approximation to the Binomial distribution, yielding the interval

$$\left(\widehat{\mu} - 1.96\,\frac{\sqrt{\widehat{\mu}(1-\widehat{\mu})}}{\sqrt{N}},\ \widehat{\mu} + 1.96\,\frac{\sqrt{\widehat{\mu}(1-\widehat{\mu})}}{\sqrt{N}}\right),$$

which can also be written as

$$\left(\widehat{\mu} - 1.96\widehat{\mu}\,\widehat{\mathrm{RE}},\ \widehat{\mu} + 1.96\widehat{\mu}\,\widehat{\mathrm{RE}}\right).$$

(d) An unbiased estimator Z of μ is said to be *logarithmically efficient* if

$$\lim_{\gamma\to\infty} \frac{\ln \mathbb{E}Z^2}{\ln \mu^2} = 1 . \tag{3.3}$$

Show that the CMC estimator (3.35) with $N = 1$ is not logarithmically efficient.

Solution: As Z is a Bernoulli random variable here, $\mathbb{E}\,Z^2 = \mathbb{E}\,Z = \mathbb{P}[X \geqslant \gamma] = \mu$, and thus

$$\frac{\ln \mathbb{E}Z^2}{\ln \mu^2} = \frac{\ln \mathbb{E}Z^2}{\ln \mathbb{P}[X \geqslant \gamma]^2} = \frac{\mathbb{P}[X \geqslant \gamma]}{2\ln \mathbb{P}[X \geqslant \gamma]} = \frac{1}{2}, \tag{3.4}$$

for **any** value of γ. Thus the CMC estimator is not logarithmically efficient.

18. One of the test cases in [70] involves the minimization of the *Hougen* function. Implement a cross-entropy and a simulated annealing algorithm to carry out this optimization task.

Solution:

19. In the the *binary knapsack problem*, the goal is to solve the optimization problem:

$$\max_{\boldsymbol{x}\in\{0,1\}^n} \boldsymbol{p}^\top \boldsymbol{x}\,,$$

subject to the constraints

$$\mathbf{A}\boldsymbol{x} \leqslant \mathbf{c}\,,$$

where $\boldsymbol{p}$ and $\boldsymbol{w}$ are $n \times 1$ vectors of non-negative numbers, $\mathbf{A} = (a_{ij})$ is an $m \times n$ matrix, and $\mathbf{c}$ is an $m \times 1$ vector. The interpretation is that $x_j = 1$ or 0 depending on whether item j with value p_j is packed into the knapsack or not , $j = 1, \dots, n$; The variable a_{ij} represents the i-th attribute (e.g., volume, weight) of the j-th item. Associated with each attribute is a maximal capacity, e.g., c_1 could be the maximum volume of the knapsack, c_2 the maximum weight, etc.

Write a CE program to solve the `Sento1.dat` knapsack problem at `http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/mknap2.txt`, as described in [16].

Solution: We provide Python code and example output for the cross-entropy algorithm, incorporating constant smoothing for the vector of Bernoulli probabilities employing a penalization approach to incorporate the constraints into the objective function, as suggested in [16]. Note that the algorithm frequently (but not always) obtains the optimal solution.

```python
import numpy as np
from numpy import array, square, sqrt, zeros, ones
import matplotlib.pyplot as plt

def SKnap(x,N):

    m=30 # Number of knapsacks
    # Object weights
    p=array([2, 77, 6, 67, 930, 3, 6, 270, 33, 13, 110, 21, 56,
         974, 47, 734, 238, 75, 200, 51, 47, 63, 7, 6, 468, 72,
        95, 82, 91, 83, 27, 13, 6, 76, 55, 72, 300, 6, 65, 39,
        63, 61, 52, 85, 29, 640, 558, 53, 47, 25, 3, 6, 568, 6,
        2, 780, 69, 31, 774, 22])

    # Knapsack capacities
    c=array([6000, 6000, 6000, 6000, 6000, 6000, 6000, 6000,
        6000, 4000, 6000, 6000, 6000, 6000, 6000, 6000, 6000,
        6000, 6000, 4000, 6000, 6000, 6000, 6000, 6000, 6000,
        6000, 6000, 6000, 4000])

    # Constraint matrix
    A=array([[47, 774, 76, 56, 59, 22, 42, 1, 21, 760, 818, 62,
         42, 36, 785, 29, 662, 49, 608, 116, 834, 57, 42, 39,
        994, 690, 27, 524, 23, 96, 667, 490, 805, 46, 19, 26,
        97, 71, 699, 465, 53, 26, 123, 20, 25, 450, 22, 979, 75,
         96, 27, 41, 21, 81, 15, 76, 97, 646, 898, 37],
              [73, 67, 27, 99, 35, 794, 53, 378, 234, 32, 792,
                 97, 64, 19, 435, 712, 837, 22, 504, 332, 13,
                 65, 86, 29, 894, 266, 75, 16, 86, 91, 67, 445,
                 118, 73, 97, 370, 88, 85, 165, 268, 758, 21,
                 255, 81, 5, 774, 39, 377, 18, 370, 96, 61, 57,
                 23, 13, 164, 908, 834, 960, 87],
              [36, 42, 56, 96, 438, 49, 57, 16, 978, 9, 644,
                 584, 82, 550, 283, 340, 596, 788, 33, 350, 55,
                 59, 348, 66, 468, 983, 6, 33, 42, 96, 464, 175,
                  33, 97, 15, 22, 9, 554, 358, 587, 71, 23, 931,
                  931, 94, 798, 73, 873, 22, 39, 71, 864, 59,
                 82, 16, 444, 37, 475, 65, 5],
              [47, 114, 26, 668, 82, 43, 55, 55, 56, 27, 716, 7,
                  77, 26, 950, 320, 350, 95, 714, 789, 430, 97,
                 590, 32, 69, 264, 19, 51, 97, 33, 571, 388,
                 602, 140, 15, 85, 42, 66, 778, 936, 61, 23,
                 449, 973, 828, 33, 53, 297, 75, 3, 54, 27, 918,
                  11, 620, 13, 28, 80, 79, 3],
              [61, 720, 7, 31, 22, 82, 688, 19, 82, 654, 809,
                 99, 81, 97, 830, 826, 775, 72, 9, 719, 740,
                 860, 72, 30, 82, 112, 66, 638, 150, 13, 586,
                 590, 519, 2, 320, 13, 964, 754, 70, 241, 72,
                 12, 996, 868, 36, 91, 79, 221, 49, 690, 23, 18,
                  748, 408, 688, 97, 85, 777, 294, 17],
              [698, 53, 290, 3, 62, 37, 704, 810, 42, 17, 983,
                 11, 45, 56, 234, 389, 712, 664, 59, 15, 22, 91,
                  57, 784, 75, 719, 294, 978, 75, 86, 105, 227,
                 760, 2, 190, 3, 71, 32, 210, 678, 41, 93, 47,
                 581, 37, 977, 62, 503, 32, 85, 31, 36, 30, 328,
                  74, 31, 56, 891, 62, 97],
              [71, 37, 978, 93, 9, 23, 47, 71, 744, 9, 619, 32,
                 214, 31, 796, 103, 593, 16, 468, 700, 884, 67,
                 36, 3, 93, 71, 734, 504, 81, 53, 509, 114, 293,
                  31, 75, 59, 99, 11, 67, 306, 96, 218, 845,
                 303, 3, 319, 86, 724, 22, 838, 82, 5, 330, 58,
                 55, 66, 53, 916, 89, 56],
              [33, 27, 13, 57, 6, 87, 21, 12, 15, 290, 206, 420,
                  32, 880, 854, 417, 770, 4, 12, 952, 604, 13,
                 96, 910, 34, 460, 76, 16, 140, 100, 876, 622,
                 559, 39, 640, 59, 6, 244, 232, 513, 644, 7,
                 813, 624, 990, 274, 808, 372, 2, 694, 804, 39,
                 5, 644, 914, 484, 1, 8, 43, 92],
              [16, 36, 538, 210, 844, 520, 33, 73, 100, 284,
                 650, 85, 894, 2, 206, 637, 324, 318, 7, 566,
                 46, 818, 92, 65, 520, 721, 90, 53, 174, 43,
                 320, 812, 382, 16, 878, 678, 29, 92, 755, 827,
                 27, 218, 143, 12, 57, 480, 154, 944, 7, 730,
                 12, 65, 67, 39, 390, 32, 39, 318, 47, 86],
              [45, 51, 59, 21, 53, 43, 25, 7, 42, 27, 310, 45,
                 72, 53, 798, 304, 354, 79, 45, 44, 52, 76, 45,
                 26, 27, 968, 86, 16, 62, 85, 790, 208, 390, 36,
                  62, 83, 93, 16, 574, 150, 99, 7, 920, 860, 12,
                  404, 31, 560, 37, 32, 9, 62, 7, 43, 17, 77,
                 73, 368, 66, 82],
              [11, 51, 97, 26, 83, 426, 92, 39, 66, 2, 23, 93,
                 85, 660, 85, 774, 77, 77, 927, 868, 7, 554,
                 760, 104, 48, 202, 45, 75, 51, 55, 716, 752,
                 37, 95, 267, 91, 5, 956, 444, 529, 96, 99, 17,
                 99, 62, 7, 394, 580, 604, 89, 678, 476, 97,
                 234, 1, 608, 19, 69, 676, 51],
              [410, 89, 414, 81, 130, 491, 6, 238, 79, 43, 5,
                 288, 910, 204, 948, 19, 644, 21, 295, 11, 6,
                 595, 904, 67, 51, 703, 430, 95, 408, 89, 11,
                 495, 844, 13, 417, 570, 9, 429, 16, 939, 430,
                 270, 49, 72, 65, 66, 338, 994, 167, 76, 47,
                 211, 87, 39, 1, 570, 85, 134, 967, 12],
              [553, 63, 35, 63, 98, 402, 664, 85, 458, 834, 3,
                 62, 508, 7, 1, 72, 88, 45, 496, 43, 750, 222,
                 96, 31, 278, 184, 36, 7, 210, 55, 653, 51, 35,
                 37, 393, 2, 49, 884, 418, 379, 75, 338, 51, 21,
                  29, 95, 790, 846, 720, 71, 728, 930, 95, 1,
                 910, 5, 804, 5, 284, 128],
              [423, 6, 58, 36, 37, 321, 22, 26, 16, 27, 218,
                 530, 93, 55, 89, 71, 828, 75, 628, 67, 66, 622,
                  440, 91, 73, 790, 710, 59, 83, 968, 129, 632,
                 170, 67, 613, 608, 43, 71, 730, 910, 36, 92,
                 950, 138, 23, 95, 460, 62, 189, 73, 65, 943,
                 62, 554, 46, 318, 13, 540, 90, 53],
              [967, 654, 46, 69, 26, 769, 82, 89, 15, 87, 46,
                 59, 22, 840, 66, 35, 684, 57, 254, 230, 21,
                 586, 51, 19, 984, 156, 23, 748, 760, 65, 339,
                 892, 13, 13, 327, 65, 35, 246, 71, 178, 83, 3,
                 34, 624, 788, 200, 980, 882, 343, 550, 708,
                 542, 53, 72, 86, 51, 700, 524, 577, 948],
              [132, 900, 72, 51, 91, 150, 22, 110, 154, 148, 99,
                  75, 21, 544, 110, 11, 52, 840, 201, 2, 6, 663,
                  22, 20, 89, 10, 93, 964, 924, 73, 501, 398, 3,
                  2, 279, 5, 288, 80, 91, 132, 620, 628, 57, 79,
                  2, 874, 36, 497, 846, 22, 350, 866, 57, 86,
                 83, 178, 968, 52, 399, 628],
              [869, 26, 710, 37, 81, 89, 6, 82, 82, 56, 96, 66,
                 46, 13, 934, 49, 394, 72, 194, 408, 5, 541, 88,
                  93, 36, 398, 508, 89, 66, 16, 71, 466, 7, 95,
                 464, 41, 69, 130, 488, 695, 82, 39, 95, 53, 37,
                  200, 87, 56, 268, 71, 304, 855, 22, 564, 47,
                 26, 26, 370, 569, 2],
              [494, 2, 25, 61, 674, 638, 61, 59, 62, 690, 630,
                 86, 198, 24, 15, 650, 75, 25, 571, 338, 268,
                 958, 95, 898, 56, 585, 99, 83, 21, 600, 462,
                 940, 96, 464, 228, 93, 72, 734, 89, 287, 174,
                 62, 51, 73, 42, 838, 82, 515, 232, 91, 25, 47,
                 12, 56, 65, 734, 70, 48, 209, 71],
              [267, 290, 31, 844, 12, 570, 13, 69, 65, 848, 72,
                 780, 27, 96, 97, 17, 69, 274, 616, 36, 554,
                 236, 47, 7, 47, 134, 76, 62, 824, 55, 374, 471,
                  478, 504, 496, 754, 604, 923, 330, 22, 97, 6,
                 2, 16, 14, 958, 53, 480, 482, 93, 57, 641, 72,
                 75, 51, 96, 83, 47, 403, 32],
              [624, 7, 96, 45, 97, 148, 91, 3, 69, 26, 22, 45,
                 42, 2, 75, 76, 96, 67, 688, 2, 2, 224, 83, 69,
                 41, 660, 81, 89, 93, 27, 214, 458, 66, 72, 384,
                  59, 76, 538, 15, 840, 65, 63, 77, 33, 92, 32,
                 35, 832, 970, 49, 13, 8, 77, 75, 51, 95, 56,
                 63, 578, 47],
              [33, 62, 928, 292, 2, 340, 278, 911, 818, 770,
                 464, 53, 888, 55, 76, 31, 389, 40, 864, 36, 35,
                  37, 69, 95, 22, 648, 334, 14, 198, 42, 73,
                 594, 95, 32, 814, 45, 45, 515, 634, 254, 42,
                 29, 15, 83, 55, 176, 35, 46, 60, 296, 262, 598,
                  67, 644, 80, 999, 3, 727, 79, 374],
              [19, 780, 400, 588, 37, 86, 23, 583, 518, 42, 56,
                 1, 108, 83, 43, 720, 570, 81, 674, 25, 96, 218,
                  6, 69, 107, 534, 158, 56, 5, 938, 9, 938, 274,
                  76, 298, 9, 518, 571, 47, 175, 63, 93, 49, 94,
                  42, 26, 79, 50, 718, 926, 419, 810, 23, 363,
                 519, 339, 86, 751, 7, 86],
              [47, 75, 55, 554, 3, 800, 6, 13, 85, 65, 99, 45,
                 69, 73, 864, 95, 199, 924, 19, 948, 214, 3,
                 718, 56, 278, 1, 363, 86, 1, 22, 56, 114, 13,
                 53, 56, 19, 82, 88, 99, 543, 674, 704, 418,
                 670, 554, 282, 5, 67, 63, 466, 491, 49, 67,
                 154, 956, 911, 77, 635, 2, 49],
              [53, 12, 79, 481, 218, 26, 624, 954, 13, 580, 130,
                  608, 3, 37, 91, 78, 743, 1, 950, 45, 41, 718,
                 36, 30, 534, 418, 452, 359, 759, 88, 29, 499,
                 55, 974, 93, 56, 108, 257, 93, 171, 13, 92, 63,
                  714, 9, 84, 890, 16, 930, 967, 748, 5, 7, 6,
                 327, 894, 33, 629, 448, 21],
              [9, 19, 7, 535, 75, 3, 27, 928, 21, 7, 864, 27,
                 73, 61, 25, 75, 876, 16, 92, 22, 248, 11, 86,
                 944, 872, 996, 252, 2, 800, 334, 93, 107, 254,
                 441, 930, 744, 97, 177, 498, 931, 694, 800, 9,
                 36, 6, 539, 35, 79, 130, 860, 710, 7, 630, 475,
                  903, 552, 2, 45, 97, 974],
              [17, 36, 77, 843, 328, 22, 76, 368, 39, 71, 35,
                 850, 96, 93, 87, 56, 972, 96, 594, 864, 344,
                 76, 17, 17, 576, 629, 780, 640, 56, 65, 43,
                 196, 520, 86, 92, 31, 6, 593, 174, 569, 89,
                 718, 83, 8, 790, 285, 780, 62, 378, 313, 519,
                 2, 85, 845, 931, 731, 42, 365, 32, 33],
              [65, 59, 2, 671, 26, 364, 854, 526, 570, 630, 33,
                 654, 95, 41, 42, 27, 584, 17, 724, 59, 42, 26,
                 918, 6, 242, 356, 75, 644, 818, 168, 964, 12,
                 97, 178, 634, 21, 3, 586, 47, 382, 804, 89,
                 194, 21, 610, 168, 79, 96, 87, 266, 482, 46,
                 96, 969, 629, 128, 924, 812, 19, 2],
              [468, 13, 9, 120, 73, 7, 92, 99, 93, 418, 224, 22,
                  7, 29, 57, 33, 949, 65, 92, 898, 200, 57, 12,
                 31, 296, 185, 272, 91, 77, 37, 734, 911, 27,
                 310, 59, 33, 87, 872, 73, 79, 920, 85, 59, 72,
                 888, 49, 12, 79, 538, 947, 462, 444, 828, 935,
                 518, 894, 13, 591, 22, 920],
              [23, 93, 87, 490, 32, 63, 870, 393, 52, 23, 63,
                 634, 39, 83, 12, 72, 131, 69, 984, 87, 86, 99,
                 52, 110, 183, 704, 232, 674, 384, 47, 804, 99,
                 83, 81, 174, 99, 77, 708, 7, 623, 114, 1, 750,
                 49, 284, 492, 11, 61, 6, 449, 429, 52, 62, 482,
                  826, 147, 338, 911, 30, 984],
              [35, 55, 21, 264, 5, 35, 92, 128, 65, 27, 9, 52,
                 66, 51, 7, 47, 670, 83, 76, 7, 79, 37, 2, 46,
                 480, 608, 990, 53, 47, 19, 35, 518, 71, 59, 32,
                  87, 96, 240, 52, 310, 86, 73, 52, 31, 83, 544,
                  16, 15, 21, 774, 224, 7, 83, 680, 554, 310,
                 96, 844, 29, 61]])

    # Optimum value: 7772

    betap=-np.sum(p) # Penalty Constant to apply to the
       constraint function

    out=zeros([N,1])

    for i in range(0,N):
        for j in range(0,m):
            if (np.dot(A[j,:],x[i,:])>c[j]):
                out[i]=out[i]+betap
        out[i]=out[i]+np.dot(p,x[i,:])
    return out

S = SKnap
n=60 # Number of objects

np.set_printoptions(precision=3)
v = 0.5*ones([1,n])
N=1000
Nel = 20
eps = 10**(-2)
alpha=0.5 # Constant Smoothing

vv=v
xbest=ones([1,n])
Sbest=np.Inf

while (np.max(np.min(np.vstack((v,1-v)),axis=0))>eps).any():
    X = np.random.binomial(1,p=v,size=[N,n])
    Sx = np.hstack((X, -S(X,N)))
    sortSx = Sx[Sx[:,-1].argsort(),]
    Elite = sortSx[0:Nel,:-1]
    vnew = np.mean(Elite, axis=0)

    if sortSx[0,-1]<Sbest:
        Sbest=sortSx[0,-1]
        xbest=Elite[0,:]

    v=alpha*vnew+(1-alpha)*v

    vv=np.vstack((vv,v))

print('S(xbest)= {}, xbest: {}, v: {}\n'.format(-Sbest, xbest,v
    ))
plt.plot(vv)
plt.show()
```

```
S(xbest)= 7772.0,
xbest: [ 0.  1.  0.  0.  1.  0.  0.  1.  1.  0.
         1.  0.  1.  1.  0.  1.  0.  1.  1.  0.
         1.  0.  0.  0.  1.  0.  1.  0.  0.  1.
         0.  0.  0.  0.  0.  0.  1.  0.  0.  0.
         0.  0.  0.  0.  0.  1.  1.  0.  0.  0.
         0.  0.  1.  0.  0.  1.  0.  0.  1.  0.],
v: [[ 0.  1.  0.  0.  1.  0.  0.  1.  1.  0.
      1.  0.  1.  1.  0.  1.  0.  1.  1.  0.
      1.  0.  0.  0.  1.  0.  1.  0.  0.  1.
      0.  0.  0.  0.  0.  0.  1.  0.  0.  0.
      0.  0.  0.  0.  0.  1.  1.  0.  0.  0.
      0.  0.  1.  0.  0.  1.  0.  0.  1.  0.]]
```

20. Let $(C_1, R_1), (C_2, R_2), \ldots$ be a renewal reward process, with $\mathbb{E}R_1 < \infty$ and $\mathbb{E}C_1 < \infty$. Let $A_t = \sum_{i=1}^{N_t} R_i/t$ be the average reward at time $t = 1, 2, \ldots$, where $N_t = \max\{n : T_n \leqslant t\}$ and we have defined $T_n = \sum_{i=1}^{n} C_i$ as the time of the n-th renewal.

    (a) Show that $T_n/n \xrightarrow{\text{a.s.}} \mathbb{E}C_1$ as $n \to \infty$.

    Solution:

    (b) Show that $N_t \xrightarrow{\text{a.s.}} \infty$ as $t \to \infty$.

    Solution:

    (c) Show that $N_t/t \xrightarrow{\text{a.s.}} 1/\mathbb{E}C_1$ as $t \to \infty$. [Hint: Use the fact that $T_{N_t} \leqslant t \leqslant T_{N_t+1}$ for all $t = 1, 2, \ldots$.]

    Solution:

    (d) Show that

$$A_t \xrightarrow{\text{a.s.}} \frac{\mathbb{E}R_1}{\mathbb{E}C_1} \quad \text{as} \quad t \to \infty.$$

    Solution:

21. Prove Theorem 3.3:

Theorem: Control Variable Estimation

Let $Y_1, \dots, Y_N$ be the output of N independent simulation runs, and let $\widetilde{Y}_1, \dots, \widetilde{Y}_N$ be the corresponding control variables, with $\mathbb{E}\widetilde{Y}_k = \widetilde{\mu}$ known. Let $\varrho_{Y,\widetilde{Y}}$ be the correlation coefficient between each Y_k and $\widetilde{Y}_k$. For each $\alpha \in \mathbb{R}$ the estimator

$$\widehat{\mu}^{(c)} = \frac{1}{N}\sum_{k=1}^{N}\left[Y_k - \alpha\left(\widetilde{Y}_k - \widetilde{\mu}\right)\right]$$

is an unbiased estimator for $\mu = \mathbb{E}Y$. The minimal variance of $\widehat{\mu}^{(c)}$ is

$$\mathbb{V}\text{ar}(\widehat{\mu}^{(c)}) = \frac{1}{N}(1 - \varrho^2_{Y,\widetilde{Y}})\mathbb{V}\text{ar}(Y)$$

which is obtained for $\alpha = \varrho_{Y,\widetilde{Y}}\sqrt{\mathbb{V}\text{ar}(Y)/\mathbb{V}\text{ar}(\widetilde{Y})}$.

Solution: First, we establish unbiasedness for all $\alpha \in \mathbb{R}$,

$$\begin{aligned}\mathbb{E}\widehat{\mu}^{(c)} &= \mathbb{E}\left[\frac{1}{N}\sum_{k=1}^{N}\left[Y_k - \alpha\left(\widetilde{Y}_k - \widetilde{\mu}\right)\right]\right] = \frac{1}{N}\left[\mathbb{E}\left[\sum_{k=1}^{N}Y_k\right] - \alpha\,\mathbb{E}\left[\sum_{k=1}^{N}\widetilde{Y}_k\right] + \alpha N\widetilde{\mu}\right] \\ &= \frac{1}{N}\left[N\mathbb{E}Y - \alpha\,N\,\mathbb{E}\widetilde{Y} + \alpha\,N\,\widetilde{\mu}\right] = \mathbb{E}Y - \alpha\,\mathbb{E}\widetilde{Y} + \alpha\,\widetilde{\mu} \\ &= \mu - \alpha\widetilde{\mu} + \alpha\,\widetilde{\mu} = \mu.\end{aligned}$$

Next, we derive the α that yields the minimum variance of $\widehat{\mu}^{(c)}$. We have:

$$\mathbb{V}\text{ar}\,\widehat{\mu}^{(c)} = \frac{1}{N^2}\mathbb{V}\text{ar}\left[\sum_{k=1}^{N}\left[Y_k - \alpha\left(\widetilde{Y}_k - \widetilde{\mu}\right)\right]\right] = \frac{1}{N}\left[\mathbb{V}\text{ar}Y + \alpha^2\mathbb{V}\text{ar}\widetilde{Y}_1 - 2\,\alpha\mathbb{C}\text{ov}(Y,\widetilde{Y})\right]. \tag{3.5}$$

Observing that not all terms above depend on α, we have

$$\underset{\alpha\in\mathbb{R}}{\text{argmin}}\left\{\mathbb{V}\text{ar}\widehat{\mu}^{(c)}\right\} = \underset{\alpha\in\mathbb{R}}{\text{argmin}}\left\{\alpha^2\mathbb{V}\text{ar}\widetilde{Y} - 2\alpha\mathbb{C}\text{ov}(Y,\widetilde{Y})\right\}.$$

Taking the derivative with respect to α and setting to zero yields the single critical point (a minimizer as the second derivative is positive)

$$\alpha_* = \frac{\mathbb{C}\text{ov}(Y,\widetilde{Y})}{\mathbb{V}\text{ar}\widetilde{Y}} = \frac{\mathbb{C}\text{ov}(Y,\widetilde{Y})}{\sqrt{\mathbb{V}\text{ar}\widetilde{Y}}\sqrt{\mathbb{V}\text{ar}Y}}\frac{\sqrt{\mathbb{V}\text{ar}Y}}{\sqrt{\mathbb{V}\text{ar}\widetilde{Y}}} = \varrho_{Y,\widetilde{Y}}\sqrt{\frac{\mathbb{V}\text{ar}Y}{\mathbb{V}\text{ar}\widetilde{Y}}}.$$

Finally, substituting α^* into (**??**) yields

$$\begin{aligned}
&\frac{1}{N}\left[\mathbb{V}\mathrm{ar}Y + \alpha_*^2\mathbb{V}\mathrm{ar}\widetilde{Y} - 2\,\alpha_*\mathbb{C}\mathrm{ov}(Y,\widetilde{Y})\right] \\
&= \frac{1}{N}\left[\mathbb{V}\mathrm{ar}Y + \varrho_{Y,\widetilde{Y}}^2\frac{\mathbb{V}\mathrm{ar}Y}{\mathbb{V}\mathrm{ar}\widetilde{Y}}\mathbb{V}\mathrm{ar}\widetilde{Y} - 2\varrho_{Y,\widetilde{Y}}\sqrt{\frac{\mathbb{V}\mathrm{ar}Y}{\mathbb{V}\mathrm{ar}\widetilde{Y}}}\mathbb{C}\mathrm{ov}(Y,\widetilde{Y})\right] \\
&= \frac{1}{N}\left[\mathbb{V}\mathrm{ar}Y + \varrho_{Y,\widetilde{Y}}^2\mathbb{V}\mathrm{ar}Y - 2\varrho_{Y,\widetilde{Y}}\sqrt{\frac{\mathbb{V}\mathrm{ar}Y}{\mathbb{V}\mathrm{ar}\widetilde{Y}}}\frac{\sqrt{\mathbb{V}\mathrm{ar}Y}}{\sqrt{\mathbb{V}\mathrm{ar}Y}}\mathbb{C}\mathrm{ov}(Y,\widetilde{Y})\right] \\
&= \frac{1}{N}\left[\mathbb{V}\mathrm{ar}Y + \varrho_{Y,\widetilde{Y}}^2\mathbb{V}\mathrm{ar}Y - 2\varrho_{Y,\widetilde{Y}}^2\mathbb{V}\mathrm{ar}Y\right] = \frac{1}{N}(1-\varrho_{Y,\widetilde{Y}}^2\mathbb{V}\mathrm{ar}Y).
\end{aligned}$$

22. Prove that if $H(\boldsymbol{x}) \geqslant 0$ the importance sampling pdf g^* in **(3.22)** gives the zero-variance importance sampling estimator $\widehat{\mu} = \mu$.

    Solution:

23. Let X and Y be random variables (not necessarily independent) and suppose we wish to estimate the expected difference $\mu = \mathbb{E}[X - Y] = \mathbb{E}X - \mathbb{E}Y$.

    (a) Show that if X and Y are *positively correlated*, the variance of $X - Y$ is smaller than if X and Y are *independent*.

    Solution: We have

    $$\mathbb{V}\mathrm{ar}[X - Y] = \mathbb{V}\mathrm{ar}X + \mathbb{V}\mathrm{ar}Y - 2\mathbb{C}\mathrm{ov}(X,Y)\,.$$

    If X and Y are positively correlated, $\mathbb{C}\mathrm{ov}(X,Y) > 0$, while $\mathbb{C}\mathrm{ov}(X,Y) = 0$ if X and Y are independent. Clearly, the variance is smaller by $2\mathbb{C}\mathrm{ov}(X,Y)$ whenever X and Y are positively correlated.

    (b) Suppose now that X and Y have cdfs F and G, respectively, and are simulated via the inverse-transform method: $X = F^{-1}(U), Y = G^{-1}(V)$, with $U, V \sim \mathcal{U}(0,1)$, not necessarily independent. Intuitively, one might expect that if U and V are positively correlated, the variance of $X - Y$ would smaller than if U and V are independent. Show that this is not always the case by providing a counter-example.

    Solution: Consider the following simple example. Let $U \sim \mathcal{U}(0,1)$ and, for a parameter $\alpha \in [0,1]$, set $V_\alpha = (\alpha - U)\mathbb{1}\{U \leqslant \alpha\} + U\mathbb{1}\{U > \alpha\}$. It is not hard to show that $V \sim \mathcal{U}(0,1)$, and also that $\mathbb{C}\mathrm{ov}(U, V_\alpha) = \frac{1}{2}(1 - 2\alpha^3)$, so that U and V_α are positively correlated whenever $\alpha < 2^{-1/3} \approx 0.7937$.
    Now suppose

    $$F(x) = \begin{cases} 0\,, & \text{for } x < 0\,, \\ x\,, & \text{for } 0 \leqslant x \leqslant 1\,, \\ 1\,, & \text{for } x > 1\,, \end{cases} \quad \text{and} \quad G(y) = \begin{cases} 0\,, & \text{for } y < 0\,, \\ y^2\,, & \text{for } 0 \leqslant y \leqslant 1\,, \\ 1\,, & \text{for } y > 1\,. \end{cases}$$

    Note that we have simply taken $X \sim \mathcal{U}(0,1)$. Then inverse–transform method tells us to simulate X and Y via $X = U$ and $Y_\alpha = \sqrt{V_\alpha}$, respectively. One can explicitly compute $\mathbb{C}\mathrm{ov}(X, Y_\alpha) = \frac{1}{15}(1 - 2\alpha^{5/2})$ which is positive only when $\alpha < 2^{-2/5} \approx 0.7578$.

Thus, setting $\alpha \in (2^{-2/5}, 2^{-1/3})$ gives a situation where U and V_α are positively correlated, but X and Y_α are negatively correlated, and hence $\mathbb{V}\mathrm{ar}(X - Y_\alpha)$ is *greater* than using independent U and V.

(c) Continuing (b), assume now that F and G are continuous. Show that the variance of $X - Y$ by taking *common random numbers* $U = V$ is no larger than when U and V are independent. [Hint: Assume the following lemma of Hoeffding (1940): If (X, Y) have joint cdf H with marginal cdfs of X and Y being F and G, respectively, then

$$\mathbb{C}\mathrm{ov}(X, Y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (H(x, y) - F(x)G(y))\,\mathrm{d}x\mathrm{d}y\,,$$

provided $\mathbb{C}\mathrm{ov}(X, Y)$ exists.]

Solution: Observe that $H(x, y) = \mathbb{P}[X \leqslant x, Y \leqslant y] = \mathbb{P}[U \leqslant F(x), U \leqslant G(y)] = \mathbb{P}[U \leqslant \min\{F(x), G(y)\}] = \min\{F(x), G(y)\}$, where the second equality follows by continuity of F and G. Hence, for any $(x, y) \in \mathbb{R}^2$,

$$H(x, y) - F(x)G(y) = \left.\begin{cases} F(x)(1 - G(y))\,, & \text{for } F(x) \leqslant G(y)\,, \\ G(y)(1 - F(x))\,, & \text{for } F(x) > G(y)\,, \end{cases}\right\} \geqslant 0\,,$$

so that $\mathbb{C}\mathrm{ov}(X, Y) \geqslant 0$ from the lemma of Hoeffding. By (a), the variance of $X - Y$ using common random numbers is no larger than when X and Y are independent (i.e., if U and V are independent).

CHAPTER 4

UNSUPERVISED LEARNING

1. This exercise is to show that the Fisher information matrix $\mathbf{F}(\boldsymbol{\theta})$ in (4.8) is equal to the matrix $\mathbf{H}(\boldsymbol{\theta})$ in (4.9), in the special case where $f = g(\cdot;\boldsymbol{\theta})$, and under the assumption that integration and differentiation orders can be interchanged.

   (a) Let $\boldsymbol{h}$ be a vector-valued function and k a real-valued function. Prove the following quotient rule for differentiation:

$$\frac{\partial[\boldsymbol{h}(\boldsymbol{\theta})/k(\boldsymbol{\theta})]}{\partial\boldsymbol{\theta}} = \frac{1}{k(\boldsymbol{\theta})}\frac{\partial\boldsymbol{h}(\boldsymbol{\theta})}{\partial\boldsymbol{\theta}} - \frac{1}{k^2(\boldsymbol{\theta})}\frac{\partial k(\boldsymbol{\theta})}{\partial\boldsymbol{\theta}}\boldsymbol{h}(\boldsymbol{\theta})^\top. \qquad (4.45)$$

   Solution: Write out the matrix using the ordinary quotient rule for real-valued functions: $(h/k)' = (h'k - k'h)/k^2$.

   (b) Now take $\boldsymbol{h}(\boldsymbol{\theta}) = \frac{\partial g(\boldsymbol{X};\boldsymbol{\theta})}{\partial\boldsymbol{\theta}}$ and $k(\boldsymbol{\theta}) = g(\boldsymbol{X};\boldsymbol{\theta})$ in (4.45) and take expectations with respect to $\mathbb{E}_{\boldsymbol{\theta}}$ on both sides to show that

$$-\mathbf{H}(\boldsymbol{\theta}) = \underbrace{\mathbb{E}_{\boldsymbol{\theta}}\left[\frac{1}{g(\boldsymbol{X};\boldsymbol{\theta})}\frac{\partial\frac{\partial g(\boldsymbol{X};\boldsymbol{\theta})}{\partial\boldsymbol{\theta}}}{\partial\boldsymbol{\theta}}\right]}_{\mathbf{A}} - \mathbf{F}(\boldsymbol{\theta}).$$

   Solution: We have

$$\begin{aligned}
\frac{\partial[\boldsymbol{h}(\boldsymbol{\theta})/k(\boldsymbol{\theta})]}{\partial\boldsymbol{\theta}} &= \left[\frac{\partial^2 \ln g(\boldsymbol{X};\boldsymbol{\theta})}{\partial\theta_i\partial\theta_j}\right] \quad \text{(matrix)}\\
&= \frac{1}{g(\boldsymbol{X};\boldsymbol{\theta})}\frac{\partial\frac{\partial g(\boldsymbol{X};\boldsymbol{\theta})}{\partial\boldsymbol{\theta}}}{\partial\boldsymbol{\theta}} - \frac{1}{g^2(\boldsymbol{X};\boldsymbol{\theta})}\frac{\partial g(\boldsymbol{X};\boldsymbol{\theta})}{\partial\boldsymbol{\theta}}\left(\frac{\partial g(\boldsymbol{X};\boldsymbol{\theta})}{\partial\boldsymbol{\theta}}\right)^\top\\
&= \frac{1}{g(\boldsymbol{X};\boldsymbol{\theta})}\frac{\partial\frac{\partial g(\boldsymbol{X};\boldsymbol{\theta})}{\partial\boldsymbol{\theta}}}{\partial\boldsymbol{\theta}} - \boldsymbol{S}(\boldsymbol{\theta};\boldsymbol{X})\boldsymbol{S}(\boldsymbol{\theta};\boldsymbol{X})^\top.
\end{aligned}$$

   Taking expectations on both sides (remember that we consider the case $f = g(\cdot;\boldsymbol{\theta})$), we obtain

$$-\mathbf{H}(\boldsymbol{\theta}) = \underbrace{\mathbb{E}_{\boldsymbol{\theta}}\left[\frac{1}{g(\boldsymbol{X};\boldsymbol{\theta})}\frac{\partial\frac{\partial g(\boldsymbol{X};\boldsymbol{\theta})}{\partial\boldsymbol{\theta}}}{\partial\boldsymbol{\theta}}\right]}_{\mathbf{A}} - \mathbf{F}(\boldsymbol{\theta}).$$

(c) Finally show that $\mathbf{A}$ is the zero matrix.

Solution: This follows from an interchange of integration and differentiation:

$$\mathbf{A} = \int \frac{1}{g(\boldsymbol{x};\boldsymbol{\theta})} \frac{\partial \frac{\partial g(\boldsymbol{x};\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}}{\partial \boldsymbol{\theta}} g(\boldsymbol{x};\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{x} = \frac{\partial \int \frac{\partial g(\boldsymbol{x};\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \mathrm{d}\boldsymbol{x}}{\partial \boldsymbol{\theta}} = \mathbf{O},$$

because, again by interchange of differentiation and integration,

$$\int \frac{\partial g(\boldsymbol{x};\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}\,\mathrm{d}\boldsymbol{x} = \frac{\partial \int g(\boldsymbol{x};\boldsymbol{\theta})\,\mathrm{d}\boldsymbol{x}}{\partial \boldsymbol{\theta}} = \frac{\partial 1}{\partial \boldsymbol{\theta}} = \mathbf{0}.$$

2. Plot the mixture of $\mathcal{N}(0,1)$, $\mathcal{U}(0,1)$, and $\mathsf{Exp}(1)$ distributions, with weights $w_1 = w_2 = w_3 = 1/3$.

   Solution:

3. Denote the pdfs in Exercise 2 by f_1, f_2, f_3. Suppose that X is simulated via the following two-step procedure: First, draw Z uniformly from $\{1,2,3\}$, then draw X from f_Z. How likely is it that the outcome $x = 0.5$ of X has come from the uniform pdf f_2?

   Solution: By Bayes' formula, we have

$$f(z\,|\,\boldsymbol{x}) \propto f(z) f(\boldsymbol{x}\,|\,z) \propto f(\boldsymbol{x}\,|\,z).$$

   For the normal pdf, $f(0.5\,|\,z=1) = \mathrm{e}^{-\frac{1}{2}(0.5)^2}/\sqrt{2\pi} = 0.3521$, for the uniform pdf, $f(0.5\,|\,z=2) = 1$, and for the exponential pdf $f(0.5\,|\,z=3) = \mathrm{e}^{-0.5} = 0.6065$. The posterior probabilities are thus 0.1789, 0.5106, and 0.3097. It is thus almost 3 times more likely that the outcome 0.5 has come from the uniform distribution than from the normal one.

4. Simulate an iid training set of size 100 from the $\mathsf{Gamma}(2.3, 0.5)$ distribution, and implement the Fisher scoring method in Example 4.1 to find the maximum likelihood estimate. Plot the true and approximate pdfs.

   Solution:

5. Let $\mathscr{T} = \{\boldsymbol{X}_1,\ldots,\boldsymbol{X}_n\}$ be iid data from a pdf $g(\boldsymbol{x};\boldsymbol{\theta})$ with Fisher matrix $\mathbf{F}(\boldsymbol{\theta})$. Explain why, under the conditions where (4.7) holds,

$$\boldsymbol{S}_{\mathscr{T}}(\boldsymbol{\theta}) := \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{S}(\boldsymbol{X}_i;\boldsymbol{\theta})$$

   for large n has approximately a multivariate normal distribution with expectation vector $\mathbf{0}$ and covariance matrix $\mathbf{F}(\boldsymbol{\theta})$.

   Solution: This follows directly from the (multivariate) central limit theorem, as $\boldsymbol{S}_{\mathscr{T}}(\boldsymbol{\theta})$ is the average of iid random vectors with expectation $\mathbb{E}\boldsymbol{S}(\boldsymbol{X};\boldsymbol{\theta}) = \mathbf{0}$ and covariance matrix $\mathbf{F}(\boldsymbol{\theta})$.

6. Figure 4.15 shows a Gaussian KDE with bandwidth $\sigma = 0.2$ on the points $-0.5, 0, 0.2, 0.9$, and 1.5. Reproduce the plot in Python. Using the same bandwidth, plot also the KDE for the same data, but now with $\phi(z) = 1/2, z \in [-1, 1]$.

   Solution:

7. For fixed x', the Gaussian kernel function

$$f(x;t) := \frac{1}{\sqrt{2\pi t}} \mathrm{e}^{-\frac{1}{2}\frac{(x-x')^2}{t}}$$

   is the solution to Fourier's *heat equation*

$$\frac{\partial}{\partial t} f(x;t) = \frac{1}{2}\frac{\partial^2}{\partial x^2} f(x;t), \quad x \in \mathbb{R}, t > 0,$$

   with initial condition $f(x;0) = \delta(x - x')$ (the Dirac function at x'). Show this. As a consequence, the Gaussian KDE is the solution to the same heat equation, but now with initial condition $f(x;0) = n^{-1}\sum_{i=1}^{n} \delta(x - x_i)$. This was the motivation for the theta KDE, which is a solution to the same heat equation but now on a *bounded* interval.

   Solution: As $t \to 0$, the Gaussian kernel function $f(x;t)$ converges to the Dirac delta function at x'. Moreover,

$$\frac{\partial}{\partial t} f(x;t) = \frac{(x-x')^2 \mathrm{e}^{-\frac{(x-x')^2}{2t}}}{2\sqrt{2\pi}\, t^{5/2}} - \frac{\mathrm{e}^{-\frac{(x-x')^2}{2t}}}{2\sqrt{2\pi}\, t^{3/2}} = \frac{\mathrm{e}^{-\frac{(x-x')^2}{2t}}\left((x-x')^2 - t\right)}{2\sqrt{2\pi}\, t^{5/2}}$$

   and

$$\frac{\partial}{\partial x} f(x;t) = -\frac{(x-x')\mathrm{e}^{-\frac{(x-x')^2}{2t}}}{\sqrt{2\pi}\, t^{3/2}}$$

   Taking the derivative of the last expression with respect to x gives

$$\frac{\partial^2}{\partial x^2} f(x;t) = \frac{(x-x')^2 \mathrm{e}^{-\frac{(x-x')^2}{2t}}}{\sqrt{2\pi}\, t^{5/2}} - \frac{\mathrm{e}^{-\frac{(x-x')^2}{2t}}}{\sqrt{2\pi}\, t^{3/2}} = \frac{\mathrm{e}^{-\frac{(x-x')^2}{2t}}\left((x-x')^2 - t\right)}{\sqrt{2\pi}\, t^{5/2}},$$

   which is twice that of the partial derivative with respect to t.

8. Show that the Ward linkage in (4.41) is equal to

$$d_{\text{Ward}}(\mathcal{I}, \mathcal{J}) = \frac{|\mathcal{I}|\,|\mathcal{J}|}{|\mathcal{I}| + |\mathcal{J}|} \|\overline{\boldsymbol{x}}_{\mathcal{I}} - \overline{\boldsymbol{x}}_{\mathcal{J}}\|^2.$$

   Solution:

9. Carry out the agglomerative hierarchical clustering of Example 4.8 via the `linkage` method from `scipy.cluster.hierarchy`. Show that the linkage matrices are the same. Give a scatterplot of the data, color coded into $K = 3$ clusters.

   Solution:

```
import scipy.cluster.hierarchy as h
X = np.genfromtxt('clusterdata.csv',delimiter=',')
Z = h.linkage(X, method = 'ward')
h.dendrogram(Z) # SciPy can produce a dendogram from L
cl = h.fcluster(Z, criterion = 'maxclust', t=3)

import matplotlib.pyplot as plt
cols = ['red','green','blue']
colors = [cols[i-1] for i in cl]
plt.scatter(X[:,0], X[:,1],c=colors)
plt.show()
```

10. Suppose that we have the data $\tau_n = \{x_1, \ldots, x_n\}$ in $\mathbb{R}$ and decide to train the two-component Gaussian mixture model

$$g(x; \boldsymbol{\theta}) = w_1 \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left(-\frac{(x-\mu_1)^2}{2\sigma_1^2}\right) + w_2 \frac{1}{\sqrt{2\pi\sigma_2^2}} \exp\left(-\frac{(x-\mu_2)^2}{2\sigma_2^2}\right),$$

where the parameter vector $\boldsymbol{\theta} = [\mu_1, \mu_2, \sigma_1, \sigma_2, w_1, w_2]^\top$ belongs to the set

$$\Theta = \{\boldsymbol{\theta} : w_1 + w_2 = 1, w_1 \in [0, 1], \mu_i \in \mathbb{R}, \sigma_i > 0, \ \forall i\}\ .$$

Suppose that the training is via the maximum likelihood in (2.28). Show that

$$\sup_{\boldsymbol{\theta} \in \Theta} \frac{1}{n} \sum_{i=1}^{n} \ln g(x_i; \boldsymbol{\theta}) = \infty\ .$$

In other words, find a sequence of values for $\boldsymbol{\theta} \in \Theta$ such that the likelihood grows without bound. How can we restrict the set Θ to ensure that the likelihood remains bounded?

Solution:

11. A d-dimensional normal random vector $\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ can be defined via an affine transformation, $\boldsymbol{X} = \boldsymbol{\mu} + \boldsymbol{\Sigma}^{1/2}\boldsymbol{Z}$, of a standard normal random vector $\boldsymbol{Z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$, where $\boldsymbol{\Sigma}^{1/2}(\boldsymbol{\Sigma}^{1/2})^\top = \Sigma$. In a similar way, we can define a d-dimensional Student random vector $\boldsymbol{X} \sim \mathrm{t}_\alpha(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ via a transformation

$$\boldsymbol{X} = \boldsymbol{\mu} + \frac{1}{\sqrt{S}} \boldsymbol{\Sigma}^{1/2} \boldsymbol{Z}, \tag{4.46}$$

where, $\boldsymbol{Z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$ and $S \sim \mathsf{Gamma}(\frac{\alpha}{2}, \frac{\alpha}{2})$ are independent, $\alpha > 0$, and $\boldsymbol{\Sigma}^{1/2}(\boldsymbol{\Sigma}^{1/2})^\top = \Sigma$. Note that we obtain the multivariate normal distribution as a limiting case for $\alpha \to \infty$.

(a) Show that the density of the $\mathrm{t}_\alpha(\mathbf{0}, \mathbf{I}_d)$ distribution is given by

$$t_\alpha(\boldsymbol{x}) := \frac{\Gamma((\alpha + d)/2)}{(\pi\alpha)^{d/2}\Gamma(\alpha/2)} \left(1 + \frac{1}{\alpha}\|\boldsymbol{x}\|^2\right)^{-\frac{\alpha+d}{2}}.$$

By the transformation rule (C.23), it follows that the density of $\boldsymbol{X} \sim \mathsf{t}_\alpha(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is given by $t_{\alpha,\boldsymbol{\Sigma}}(\boldsymbol{x}-\boldsymbol{\mu})$, where

$$t_{\alpha,\boldsymbol{\Sigma}}(\boldsymbol{x}) := \frac{1}{|\boldsymbol{\Sigma}^{1/2}|} t_\alpha(\boldsymbol{\Sigma}^{-1/2}\boldsymbol{x}).$$

[Hint: conditional on $S = s$, $\boldsymbol{X}$ has a $\mathcal{N}(\mathbf{0}, \mathbf{I}_d/s)$ distribution.]

Solution: In this case, $\boldsymbol{X} = \boldsymbol{Z}/\sqrt{S}$. We have $(\boldsymbol{X}\,|\,S = s) \sim \mathcal{N}(\mathbf{0}, s^{-1}\mathbf{I}_d)$. Hence,

$$\begin{aligned}
f_{\boldsymbol{X}}(\boldsymbol{x}) &= \int_0^\infty \frac{1}{(2\pi/s)^{d/2}} \mathrm{e}^{-\frac{1}{2}\|\boldsymbol{x}\|^2 s} \times \frac{\frac{\alpha}{2}(s\frac{\alpha}{2})^{\frac{\alpha}{2}-1}\mathrm{e}^{-\frac{\alpha}{2}s}}{\Gamma(\alpha/2)}\,\mathrm{d}s \\
&= \frac{1}{\Gamma(\alpha/2)} \int_0^\infty \frac{(s/2)^{d/2}}{\pi^{d/2}} \mathrm{e}^{-\frac{1}{2}(\|\boldsymbol{x}\|^2/\alpha+1)\alpha s} \times \frac{\alpha}{2}(s\frac{\alpha}{2})^{\frac{\alpha}{2}-1}\,\mathrm{d}s \\
&= \frac{1}{\Gamma(\alpha/2)\pi^{d/2}} \int_0^\infty \left(\frac{u}{\alpha}\right)^{d/2} \mathrm{e}^{-(\|\boldsymbol{x}\|^2/\alpha+1)u} u^{\frac{\alpha}{2}-1}\,\mathrm{d}u \\
&= \frac{1}{\Gamma(\alpha/2)(\alpha\pi)^{d/2}} \int_0^\infty \mathrm{e}^{-(\|\boldsymbol{x}\|^2/\alpha+1)u} u^{\frac{\alpha+d}{2}-1}\,\mathrm{d}u \\
&= \frac{\Gamma((\alpha+d)/2)}{\Gamma(\alpha/2)(\alpha\pi)^{d/2}} \left(\|\boldsymbol{x}\|^2/\alpha+1\right)^{-\frac{\alpha+d}{2}} \underbrace{\int_0^\infty \frac{\left(\|\boldsymbol{x}\|^2/\alpha+1\right)^{\frac{\alpha+d}{2}} \mathrm{e}^{-(\|\boldsymbol{x}\|^2/\alpha+1)u} u^{\frac{\alpha+d}{2}-1}}{\Gamma((\alpha+d)/2)}\,\mathrm{d}u}_{=1}.
\end{aligned}$$

(b) We wish to fit a $\mathbf{t}_\nu(\boldsymbol{\mu}, \Sigma)$ distribution to given data $\tau = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ in $\mathbb{R}^d$ via the EM method. We use the representation (4.46) and augment the data with the vector $\boldsymbol{S} = [S_1, \ldots, S_n]^\top$ of hidden variables. Show that complete-data likelihood function is given by

$$g(\tau, \boldsymbol{s}\,|\,\boldsymbol{\theta}) = \prod_i \frac{(\alpha/2)^{\alpha/2} s_i^{(\alpha+d)/2-1} \exp(-\frac{s_i}{2}\alpha - \frac{s_i}{2}\|\boldsymbol{\Sigma}^{-1/2}(\boldsymbol{x}_i - \boldsymbol{\mu})\|^2)}{\Gamma(\alpha/2)(2\pi)^{d/2}|\boldsymbol{\Sigma}^{1/2}|}. \qquad (4.47)$$

Solution: The complete-data likelihood is given by $g(\tau, \boldsymbol{s}\,|\,\boldsymbol{\theta}) = g(\boldsymbol{s}\,|\,\boldsymbol{\theta})g(\tau\,|\,\boldsymbol{s}, \boldsymbol{\theta})$. The result now follows from the fact that the $\{(\boldsymbol{X}_i, S_i)\}$ are independent, $(\boldsymbol{X}_i\,|\,S_i = s_i) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}/s_i)$ and $S_i \sim \mathsf{Gamma}(\alpha/2, \alpha/2)$.

(c) Show that, as a consequence, conditional on the data τ and parameter $\boldsymbol{\theta}$, the hidden data are mutually independent, and

$$(S_i\,|\,\tau, \boldsymbol{\theta}) \sim \mathsf{Gamma}\left(\frac{\alpha+d}{2}, \frac{\alpha + \|\boldsymbol{\Sigma}^{-1/2}(\boldsymbol{x}_i - \boldsymbol{\mu})\|^2}{2}\right), \quad i = 1, \ldots, n.$$

Solution: This follows directly from (4.47), by viewing the right-hand side as a function of the $\{s_i\}$ only.

(d) At iteration t of the EM algorithm, let $g^{(t)}(\boldsymbol{s}) = g(\boldsymbol{s}\,|\,\tau, \boldsymbol{\theta}^{(t-1)})$ be the density of the missing data, given the observed data τ and the current parameter guess $\boldsymbol{\theta}^{(t-1)}$. Verify that the expected complete-data log-likelihood is given by:

$$\begin{aligned}
\mathbb{E}_{g^{(t)}} \ln g(\tau, \boldsymbol{S}\,|\,\boldsymbol{\theta}) = \frac{n\alpha}{2}\ln\frac{\alpha}{2} - \frac{nd}{2}\ln(2\pi) - n\ln\Gamma(\frac{\alpha}{2}) - \frac{n}{2}\ln|\boldsymbol{\Sigma}| \\
+ \frac{\alpha+d-2}{2}\sum_{i=1}^n \mathbb{E}_{g^{(t)}} \ln S_i - \sum_{i=1}^n \frac{\alpha + \|\boldsymbol{\Sigma}^{-1/2}(\boldsymbol{x}_i - \boldsymbol{\mu})\|^2}{2}\mathbb{E}_{g^{(t)}} S_i.
\end{aligned}$$

Show that

$$\mathbb{E}_{g^{(t)}} S_i = \frac{\alpha^{(t-1)} + d}{\alpha^{(t-1)} + \| (\boldsymbol{\Sigma}^{(t-1)})^{-1/2} (\boldsymbol{x}_i - \boldsymbol{\mu}^{(t-1)})\|^2} =: w_i^{(t-1)}$$

$$\mathbb{E}_{g^{(t)}} \ln S_i = \psi\left(\frac{\alpha^{(t-1)} + d}{2}\right) - \ln\left(\frac{\alpha^{(t-1)} + d}{2}\right) + \ln w_i^{(t-1)},$$

where $\psi := (\ln \Gamma)'$ is *digamma* function.

Solution: Take the logarithm and then the expectation of (4.47) to obtain the expected complete-date log-likelihood. The expectations follow from (c) and the fact that if $X \sim \mathsf{Gamma}(\alpha,\beta)$, then $\mathbb{E}X = \alpha/\beta$ and $\mathbb{E}\ln X = \psi(\alpha) - \ln\beta$.

(e) Finally, show that in the M-step of the EM algorithm $\boldsymbol{\theta}^{(t)}$ is updated from $\boldsymbol{\theta}^{(t-1)}$ as follows:

$$\boldsymbol{\mu}^{(t)} = \frac{\sum_{i=1}^n w_i^{(t-1)} \boldsymbol{x}_i}{\sum_{i=1}^n w_i^{(t-1)}}$$

$$\boldsymbol{\Sigma}^{(t)} = \frac{1}{n} \sum_{i=1}^n w_i^{(t-1)} (\boldsymbol{x}_i - \boldsymbol{\mu}^{(t)})(\boldsymbol{x}_i - \boldsymbol{\mu}^{(t)})^\top$$

And $\alpha^{(t)}$ is defined implicitly through the solution of the nonlinear equation:

$$\ln\left(\frac{\alpha}{2}\right) - \psi\left(\frac{\alpha}{2}\right) + \psi(\frac{\alpha^{(t)} + d}{2}) - \ln(\frac{\alpha^{(t)} + d}{2}) + 1 + \frac{\sum_{i=1}^n \left(\ln(w_i^{(t-1)}) - w_i^{(t-1)}\right)}{n} = 0.$$

Solution: For fixed α the expected complete-data log-likelihood is maximal when

$$\sum_{i=1}^n \|\boldsymbol{\Sigma}^{-1/2}(\boldsymbol{x}_i - \boldsymbol{\mu})\|^2 w_i^{(t-1)}$$

is minimal. This leads to the stated updating rules or $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. Finally, for any fixed $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, the expected complete-data log-likelihood is maximized when

$$\frac{n\alpha}{2} \ln \frac{\alpha}{2} - n \ln \Gamma(\frac{\alpha}{2}) + \frac{\alpha}{2} \sum_{i=1}^n \left(\ln(w_i^{(t-1)}) - w_i^{(t-1)}\right) + \frac{n\alpha}{2} \left(\psi(\frac{\alpha^{(t)} + d}{2}) - \ln(\frac{\alpha^{(t)} + d}{2})\right)$$

is maximal. Setting the derivative with respect to α equal to 0 gives the stated equation.

12. A generalization of both the gamma and inverse-gamma distribution is the *generalized inverse-gamma distribution*, which has density

$$f(s) = \frac{(a/b)^{p/2}}{2K_p(\sqrt{ab})} s^{p-1} \mathrm{e}^{-\frac{1}{2}(as + b/s)}, \quad a, b, s > 0, \; p \in \mathbb{R}, \tag{4.48}$$

where K_p is the *modified Bessel function of the second kind*, which can be defined as the integral

$$K_p(x) = \int_0^\infty \mathrm{e}^{-x\cosh(t)} \cosh(pt) \, \mathrm{d}t, \quad x > 0, \; p \in \mathbb{R}. \tag{4.49}$$

We write $S \sim \mathsf{GIG}(a, b, p)$ to denote that S has a pdf of the form **(4.48)**. The function K_p has many interesting properties. Special cases include

$$\begin{aligned}
K_{1/2}(x) &= \sqrt{\frac{x\pi}{2}}\mathrm{e}^{-x}\,\frac{1}{x} \\
K_{3/2}(x) &= \sqrt{\frac{x\pi}{2}}\mathrm{e}^{-x}\left(\frac{1}{x} + \frac{1}{x^2}\right) \\
K_{5/2}(x) &= \sqrt{\frac{x\pi}{2}}\mathrm{e}^{-x}\left(\frac{1}{x} + \frac{3}{x^2} + \frac{3}{x^3}\right).
\end{aligned}$$

More generally, K_p follows the recursion

$$K_{p+1}(x) = K_{p-1}(x) + \frac{2p}{x}K_p(x). \tag{4.50}$$

(a) Using the change of variables $\mathrm{e}^z = s\sqrt{a/b}$, show that

$$\int_0^\infty s^{p-1}\mathrm{e}^{-\frac{1}{2}(as+b/s)}\,\mathrm{d}s = 2K_p(\sqrt{ab})(b/a)^{p/2}.$$

Solution:

(b) Let $S \sim \mathsf{GIG}(a, b, p)$. Show that

$$\mathbb{E}S = \frac{\sqrt{b}K_{p+1}(\sqrt{ab})}{\sqrt{a}K_p(\sqrt{ab})} \tag{4.51}$$

and

$$\mathbb{E}S^{-1} = \frac{\sqrt{a}K_{p+1}(\sqrt{ab})}{\sqrt{b}K_p(\sqrt{ab})} - \frac{2p}{b}. \tag{4.52}$$

Solution:

13. In Exercise **11** we viewed the multivariate Student t_α distribution as a *scale-mixture* of the $\mathcal{N}(\mathbf{0}, \mathbf{I}_d)$ distribution. In this exercise, we consider a similar transformation, but now $\mathbf{\Sigma}^{1/2}\mathbf{Z} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$ is not divided but is *multiplied* by $\sqrt{S}$, with $S \sim \mathsf{Gamma}(\alpha/2, \alpha/2)$:

$$\boldsymbol{X} = \boldsymbol{\mu} + \sqrt{S}\,\mathbf{\Sigma}^{1/2}\,\mathbf{Z}, \tag{4.53}$$

where S and $\mathbf{Z}$ are independent and $\alpha > 0$.

(a) Show, using Exercise **12**, that for $\mathbf{\Sigma}^{1/2} = \mathbf{I}_d$ and $\boldsymbol{\mu} = \mathbf{0}$, the random vector $\boldsymbol{X}$ has a d-dimensional *Bessel distribution*, with density:

$$\kappa_\alpha(\boldsymbol{x}) := \frac{2^{1-(\alpha+d)/2}\alpha^{(\alpha+d)/4}\|\boldsymbol{x}\|^{(\alpha-d)/2}}{\pi^{d/2}\Gamma(\alpha/2)}K_{(\alpha-d)/2}\left(\|\boldsymbol{x}\|\sqrt{\alpha}\right), \quad \boldsymbol{x} \in \mathbb{R}^d,$$

where K_p is the modified Bessel function of the second kind in **(4.49)**. We write $\boldsymbol{X} \sim \mathsf{Bessel}_\alpha(\mathbf{0}, \mathbf{I}_d)$. A random vector $\boldsymbol{X}$ is said to have a $\mathsf{Bessel}_\alpha(\boldsymbol{\mu}, \mathbf{\Sigma})$ distribution if it can be written in the form **(4.53)**. By the transformation rule

(C.23), its density is given by $\frac{1}{\sqrt{|\Sigma|}}\kappa_\alpha(\boldsymbol{\Sigma}^{-1/2}(\boldsymbol{x}-\boldsymbol{\mu}))$. Special instances of the Bessel pdf include:

$$\begin{aligned}
\kappa_2(x) &= \frac{\exp(-\sqrt{2}|x|)}{\sqrt{2}}\\
\kappa_4(x) &= \frac{1+2|x|}{2}\exp(-2|x|)\\
\kappa_4(x_1,x_2,x_3) &= \frac{1}{\pi}\exp\left(-2\sqrt{x_1^2+x_2^2+x_3^2}\right)\\
\kappa_{d+1}(\boldsymbol{x}) &= \frac{((d+1)/2)^{d/2}\sqrt{\pi}}{(2\pi)^{d/2}\Gamma((d+1)/2)}\exp\left(-\sqrt{d+1}\,\|\boldsymbol{x}\|\right), \quad \boldsymbol{x}\in\mathbb{R}^d.
\end{aligned}$$

Note that k_2 is the (scaled) pdf of the double-exponential or *Laplace* distribution.

Solution: We have $(\boldsymbol{X}\,|\,S=s)\sim\mathcal{N}(\boldsymbol{0},\mathbf{I}_d s)$, so

$$f_{\boldsymbol{X}}(\boldsymbol{x}) = \int_0^\infty \frac{1}{(2\pi s)^{d/2}}\mathrm{e}^{-\|\boldsymbol{x}\|^2/(2s)}\times\frac{\mathrm{e}^{-s\alpha/2}s^{\alpha/2-1}(\alpha/2)^{\alpha/2}}{\Gamma(\alpha/2)}\mathrm{d}s.$$

Defining $a=\alpha$, $b=\|\boldsymbol{x}\|^2$, and $p=(\alpha-d)/2$, we write the integral in terms of the density of the $\mathsf{GIG}(a,b,p)$ distribution as follows:

$$\begin{aligned}
f_{\boldsymbol{X}}(\boldsymbol{x}) &= \frac{(\alpha/2)^{\alpha/2}}{(2\pi)^{d/2}\Gamma(\alpha/2)}\times\frac{2K_p(\sqrt{ab})}{(a/b)^{p/2}}\underbrace{\int_0^\infty\frac{(a/b)^{p/2}}{2K_p(\sqrt{ab})}\mathrm{e}^{-\frac{1}{2}(as+b/s)}s^{p-1}\,\mathrm{d}s}_{=1}\\
&= \frac{2^{1-(\alpha+d)/2}\alpha^{(\alpha+d)/4}\|\boldsymbol{x}\|^{(\alpha-d)/2}}{\pi^{d/2}\Gamma(\alpha/2)}K_{(\alpha-d)/2}\left(\|\boldsymbol{x}\|\sqrt{\alpha}\right).
\end{aligned}$$

(b) Given the data $\tau=\{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n\}$ in $\mathbb{R}^d$, we wish to fit a Bessel pdf to the data by employing the EM algorithm, augmenting the data with the vector $\boldsymbol{S}=[S_1,\ldots,S_n]^\top$ of missing data. We assume that α is known and $\alpha>d$. Show that conditional on τ (and given $\boldsymbol{\theta}$), the missing data vector $\boldsymbol{S}$ has independent components, with $S_i\sim\mathsf{GIG}(\alpha,b_i,(\alpha-d)/2)$, with $b_i:=\|\Sigma^{-1/2}(\boldsymbol{x}_i-\boldsymbol{\mu})\|^2$, $i=1,\ldots,n$.

Solution: Writing out $g(\tau,\boldsymbol{s}\,|\,\boldsymbol{\theta})=g(\boldsymbol{s}\,|\,\boldsymbol{\theta})g(\tau\,|\,\boldsymbol{s},\boldsymbol{\theta})$, we see that the complete-data likelihood is given by

$$g(\tau,\boldsymbol{s}\,|\,\boldsymbol{\theta})=\prod_{i=1}^n\frac{(\alpha/2)^{\alpha/2}s_i^{(\alpha-d)/2-1}\exp(-\frac{s_i}{2}\alpha-\frac{1}{2s_i}\|\Sigma^{-1/2}(\boldsymbol{x}_i-\boldsymbol{\mu})\|^2)}{\Gamma(\alpha/2)(2\pi)^{d/2}|\Sigma|^{1/2}}. \qquad (4.1)$$

As a consequence, conditional on τ, the components of $\boldsymbol{S}$ are independent and each has a GIG pdf, of the form (4.48).

(c) At iteration t of the EM algorithm, let $g^{(t)}(\boldsymbol{s})=g(\boldsymbol{s}\,|\,\tau,\boldsymbol{\theta}^{(t-1)})$ be the density of the missing data, given the observed data τ and the current parameter guess $\boldsymbol{\theta}^{(t-1)}$. Show that expected complete-data log-likelihood is given by:

$$Q^{(t)}(\boldsymbol{\theta}):=\mathbb{E}_{g^{(t)}}\ln g(\tau,\boldsymbol{S}\,|\,\boldsymbol{\theta})=-\frac{1}{2}\sum_{i=1}^n b_i(\boldsymbol{\theta})w_i^{(t-1)}+\text{constant}, \qquad (4.54)$$

where $b_i(\boldsymbol{\theta}) = \|\Sigma^{-1/2}(\boldsymbol{x}_i - \boldsymbol{\mu})\|^2$ and

$$w_i^{(t-1)} := \frac{\sqrt{\alpha}K_{(\alpha-d+2)/2}(\sqrt{\alpha b_i(\boldsymbol{\theta}^{(t-1)})})}{\sqrt{b_i(\boldsymbol{\theta}^{(t-1)})}K_{(\alpha-d)/2}(\sqrt{\alpha b_i(\boldsymbol{\theta}^{(t-1)})})} - \frac{\alpha - d}{b_i(\boldsymbol{\theta}^{(t-1)})}, \quad i = 1, \dots, n.$$

Solution: From (**??**)

$$Q^{(t)}(\boldsymbol{\theta}) := \mathbb{E}_{g^{(t)}} \ln g(\tau, \boldsymbol{S} \,|\, \boldsymbol{\theta}) = \mathbb{E}_{g^{(t)}} \sum_{i=1}^{n} \left(\text{const.} + \frac{\alpha - d - 2}{2} \ln S_i - \frac{1}{2} b_i(\boldsymbol{\theta}) S_i^{-1} - \frac{\alpha}{2} S_i \right)$$
$$= -\frac{1}{2} \sum_{i=1}^{n} b_i(\boldsymbol{\theta})\, \mathbb{E}_{g^{(t)}} S_i^{-1} + \text{const.}$$

Now define $w_i^{(t-1)} := \mathbb{E}_{g^{(t)}} S_i^{-1}$ and apply (4.52).

(d) From (4.54) derive the M-step of the EM algorithm. That is, show how $\boldsymbol{\theta}^{(t)}$ is updated from $\boldsymbol{\theta}^{(t-1)}$.

Solution: We need to maximize the quadratic form

$$\sum_{i=1}^{n} w_i^{(t-1)} (\boldsymbol{x}_i - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\boldsymbol{x}_i - \boldsymbol{\mu})$$

with respect to $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$. Taking gradients with respect to $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ and equating these to $\mathbf{0}$ and $\mathbf{O}$, respectively, gives

$$\boldsymbol{\mu}^{(t)} = \frac{\sum_{i=1}^{n} w_i^{(t-1)} \boldsymbol{x}_i}{\sum_{i=1}^{n} w_i^{(t-1)}}$$
$$\Sigma^{(t)} = \frac{1}{n} \sum_{i=1}^{n} w_i^{(t-1)} (\boldsymbol{x}_i - \boldsymbol{\mu}^{(t)})(\boldsymbol{x}_i - \boldsymbol{\mu}^{(t)})^\top.$$

14. Consider the ellipsoid $E = \{\boldsymbol{x} \in \mathbb{R}^d : \boldsymbol{x}\Sigma^{-1}\boldsymbol{x} = 1\}$ in (4.42). Let $\mathbf{U}\mathbf{D}^2\mathbf{U}^\top$ be an SVD of $\boldsymbol{\Sigma}$. Show that the linear transformation $\boldsymbol{x} \mapsto \mathbf{U}^\top\mathbf{D}^{-1}\boldsymbol{x}$ maps the points on E onto the unit sphere $\{z \in \mathbb{R}^d : \|z\| = 1\}$.

    Solution:

15. Figure 4.13 shows how the centered "surfboard" data are projected onto the first column of the principal component matrix $\mathbf{U}$. Suppose we project the data instead onto the plane spanned by the first *two* columns of $\mathbf{U}$. What are a and b in the representation $ax_1 + bx_2 = x_3$ of this plane?

    Solution: For any vector $\boldsymbol{x} = [x_1, x_2, x_3]^\top$ and its projection onto the plane, $\boldsymbol{y} = \mathbf{U}_2\mathbf{U}_2^\top\boldsymbol{x}$, the difference $\boldsymbol{x} - \boldsymbol{y}$ is a normal vector to the plane. Scale this vector to a normal vector $\boldsymbol{h} = [a, b, 1]$. Points on the plane are now represented by $\langle \boldsymbol{h}, \boldsymbol{x} \rangle = 0$; i.e., $ax_1 + bx_2 = x_3$. The following Python program gives the numerical values for a and b.

```python
import numpy as np

X = np.genfromtxt('pcadat.csv', delimiter=',')
n = X.shape[0]
X = X - X.mean(axis=0)

G = X.T @ X
U, _ , _ = np.linalg.svd(G/n)

Y = X @ U[:,0:2] @ U[:,0:2].T

normal = X[0,:] - Y[0,:]
h = - normal/normal[2]
print("a = {}, b = {}".format(h[0], h[1]))
```

```
a = -0.5139978194692603, b = 1.1504692429116332
```

16. Figure 4.14 suggests that we can assign each feature vector $\boldsymbol{x}$ in the **iris** data set to one of two clusters, based on the value of $\boldsymbol{u}_1^\top \boldsymbol{x}$, where $\boldsymbol{u}_1$ is the first principal component. Plot the sepal lengths against petal lengths and color the points for which $\boldsymbol{u}_1^\top \boldsymbol{x} < 1.5$ differently to points for which $\boldsymbol{u}_1^\top \boldsymbol{x} \geqslant 1.5$. To which species of iris do these cluster correspond?

Solution:

CHAPTER 5

REGRESSION

1. Following his mentor Francis Galton, the mathematician/statistician Karl Pearson conducted comprehensive studies comparing hereditary traits between members of the same family. Figure 5.10 depicts the measurements of the heights of 1078 fathers and their adult sons (one son per father). The data is available from the book's website as `pearson.csv`.

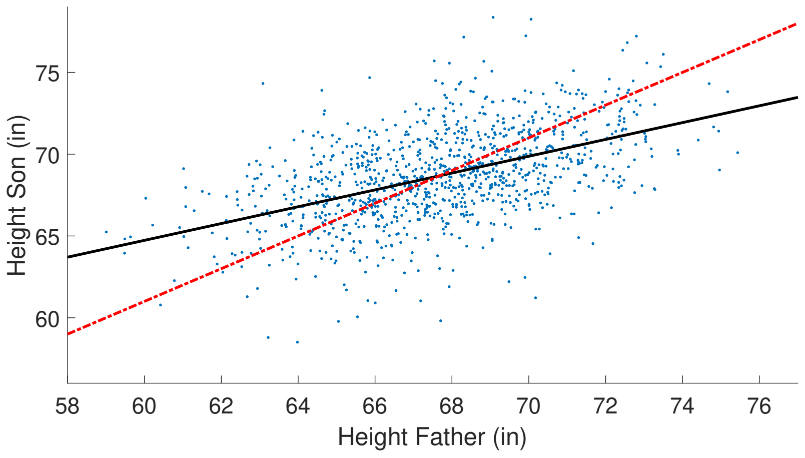


Figure 5.10: A scatter plot of heights from Pearson's data.

   (a) Show that sons are on average 1 inch taller than the fathers.

   (b) We could try to "explain" the height of the son by taking the height of his father and adding 1 inch. The prediction line $y = x + 1$ (red dashed) is given Figure 5.10. The black solid line is the fitted regression line. This line has a slope less than 1, and demonstrates Galton's "regression" to the average. Find the intercept and slope of the fitted regression line.

Solution:

   (a) The average height of the fathers is 67.69 inches, and of the sons 68.68 inches.

   (b) Intercept $\widehat{\beta_0} = 33.9$, slope $\widehat{\beta_1} = 0.514$.

2. For the simple linear regression model, show that the values for $\widehat{\beta}_1$ and $\widehat{\beta}_0$ that solve the equations (5.9) are:

$$\widehat{\beta}_1 = \frac{\sum_{i=1}^n (x_i - \overline{x})(y_i - \overline{y})}{\sum_{i=1}^n (x_i - \overline{x})^2} \tag{5.40}$$

$$\widehat{\beta}_0 = \overline{y} - \widehat{\beta}_1 \overline{x}\,, \tag{5.41}$$

provided that not all x_i are the same.

Solution:

3. Edwin Hubble discovered that the universe is expanding. If v is a galaxy's recession velocity (relative to any other galaxy) and d is its distance (from that same galaxy), Hubble's law states that

$$v = H\,d,$$

where H is known as Hubble's constant. The following are distance (in millions of light-years) and velocity (thousands of miles per second) measurements made on 5 galactic clusters.

distance	68	137	315	405	700
velocity	2.4	4.7	12.0	14.4	26.0

State the regression model and estimate H.

Solution: This is a simple linear regression model with intercept 0; that is, the random velocity Y for a fixed distance x is given by $Y = Hx + \varepsilon$. The model matrix is $\mathbf{X} = [x_1, \ldots, x_5]^\top$, where $\{x_i\}$ are the observed distances. Solving (5.9) gives

$$\widehat{H} = \frac{\sum_{i=1}^5 x_i y_i}{\sum_{i=1}^5 x_i^2} = 0.0368\,.$$

4. The multiple linear regression model (5.6) can be viewed as a first-order approximation of the general model

$$Y = g(\boldsymbol{x}) + \varepsilon\,, \tag{5.1}$$

where $\mathbb{E}\,\varepsilon = 0$, $\mathbb{V}\mathrm{ar}\,\varepsilon = \sigma^2$, and $b(\boldsymbol{x})$ is some known or unknown function of a d-dimensional vector $\boldsymbol{x}$ of explanatory variables. To see this, replace $g(\boldsymbol{x})$ with its first-order Taylor approximation around some point $\boldsymbol{x}_0$ and write this as $\beta_0 + \boldsymbol{x}^\top \boldsymbol{\beta}$. Express β_0 and $\boldsymbol{\beta}$ in terms of g and $\boldsymbol{x}_0$.

Solution:

5. Table 5.6 shows data from an agricultural experiment crop yield is measured for two levels of pesticide and tree levels of fertilizer. There are three responses for each combination.

Table 5.6: Crop yield.

	Fertilizer		
Pesticide	Low	Medium	High
No	3.23, 3.20, 3.16	2.99, 2.85, 2.77	5.72, 5.77, 5.62
Yes	6.78, 6.73, 6.79	9.07, 9.09, 8.86	8.12, 8.04, 8.31

(a) Organize the data in standard form, where each row corresponds to a single measurement and the columns correspond to the response variable and the two factor variables.

Solution:

The following table gives the data organised in standard form.

Table : Crop yield data as a spreadsheet.

Crop Yield	Pesticide	Fertilizer
3.23	No	Low
3.20	No	Low
3.16	No	Low
2.99	No	Medium
2.85	No	Medium
2.77	No	Medium
5.72	No	High
5.77	No	High
5.62	No	High
6.78	Yes	Low
6.73	Yes	Low
6.79	Yes	Low
9.07	Yes	Medium
9.09	Yes	Medium
8.86	Yes	Medium
8.12	Yes	High
8.04	Yes	High
8.31	Yes	High

(b) Let Y_{ijk} be the response for the k-th replication at level i for factor 1 and level j for factor 2. To assess which factors best explain the response variable, we use the ANOVA model

$$Y_{ijk} = \mu + \alpha_i + \beta_j + \gamma_{ij} + \varepsilon_{ijk}, \tag{5.43}$$

where $\sum_i \alpha_i = \sum_j \beta_j = \sum_i \gamma_{ij} = \sum_j \gamma_{ij} = 0$. Define $\boldsymbol{\beta} = [\mu, \alpha_1, \alpha_2, \beta_1, \beta_2, \beta_3, \gamma_{11}, \gamma_{12}, \gamma_{13}, \gamma_{21}, \gamma_{22}, \gamma_{23}]^\top$. Give the corresponding 18×12 model matrix.

Solution:

The 18×12 model matrix is given by

$$\mathbf{X} = \begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix},$$

where $\mathbf{1}$ and $\mathbf{0}$ are 3×1 vectors of ones and zeros, respectively.

(c) Note that in this case the parameters are linearly dependent. For example, $\alpha_2 = -\alpha_1$ and $\gamma_{13} = -(\gamma_{11} + \gamma_{12})$. To retain only 6 linearly independent variables consider the 6-dimensional parameter vector $\widetilde{\boldsymbol{\beta}} = [\mu, \alpha_1, \beta_1, \beta_2, \gamma_{11}, \gamma_{12}]^\top$. Find the matrix $\mathbf{M}$ such that $\mathbf{M}\widetilde{\boldsymbol{\beta}} = \boldsymbol{\beta}$.

Solution:

The matrix $\mathbf{M}$ is given below:

$$\underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}}_{\mathbf{M}} \underbrace{\begin{bmatrix} \mu \\ \alpha_1 \\ \beta_1 \\ \beta_2 \\ \gamma_{11} \\ \gamma_{12} \end{bmatrix}}_{\widetilde{\boldsymbol{\beta}}} = \underbrace{\begin{bmatrix} \mu \\ \alpha_1 \\ \alpha_2 \\ \beta_1 \\ \beta_2 \\ \beta_3 \\ \gamma_{11} \\ \gamma_{12} \\ \gamma_{13} \\ \gamma_{21} \\ \gamma_{22} \\ \gamma_{23} \end{bmatrix}}_{\boldsymbol{\beta}} = \begin{bmatrix} \mu \\ \alpha_1 \\ -\alpha_1 \\ \beta_1 \\ \beta_2 \\ -\beta_1 - \beta_2 \\ \gamma_{11} \\ \gamma_{12} \\ -\gamma_{11} - \gamma_{12} \\ -\gamma_{11} \\ -\gamma_{12} \\ \gamma_{11} + \gamma_{12} \end{bmatrix}.$$

(d) Give the model matrix corresponding to $\widetilde{\boldsymbol{\beta}}$.

Solution: We have $\mathbf{X}\boldsymbol{\beta} = \mathbf{X}\mathbf{M}\widetilde{\boldsymbol{\beta}}$, so the model matrix $\mathbf{A} = \mathbf{X}\mathbf{M}$ can be explicitly computed to be the 18×6 matrix

$$\mathbf{A} = \begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & -\mathbf{1} & -\mathbf{1} & -\mathbf{1} & -\mathbf{1} \\ \mathbf{1} & -\mathbf{1} & \mathbf{1} & \mathbf{0} & -\mathbf{1} & \mathbf{0} \\ \mathbf{1} & -\mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} & -\mathbf{1} \\ \mathbf{1} & -\mathbf{1} & -\mathbf{1} & -\mathbf{1} & \mathbf{1} & \mathbf{1} \end{bmatrix},$$

where $\mathbf{1}$ and $\mathbf{0}$ are 3×1 vectors of ones and zeros, respectively.

6. Show that for the birthweight data in Section 5.6.6.2 there is no significant decrease in birthweight for smoking mothers. [Hint: create a new variable `nonsmoke` = 1-`smoke`, which reverses the encoding for the smokers and nonsmokers. Then, the

parameter $\beta_1 + \beta_3$ in the original model is the same as the parameter β_1 in the model

$$\texttt{Bwt} = \beta_0 + \beta_1\texttt{age} + \beta_2\texttt{nonsmoke} + \beta_3\texttt{age} \times \texttt{nonsmoke} + \varepsilon\,.$$

Now find a 95% for β_3 and see if it contains 0.]

Solution:

7. Prove (5.37) and (5.38).

Solution: Recall the definition of $r_\tau(\boldsymbol{\beta})$ from Example 5.10:

$$r_\tau(\boldsymbol{\beta}) := \frac{1}{n}\sum_{i=1}^{n}\left[(1-y_i)\boldsymbol{x}_i^\top\boldsymbol{\beta} + \ln\left(1+\mathrm{e}^{-\boldsymbol{x}_i^T\boldsymbol{\beta}}\right)\right].$$

We seek to show that the gradient ∇r_τ and Hessian $\mathbf{H}$ of r_τ are given by

$$\nabla\, r_\tau = \frac{1}{n}\sum_{i=1}^{n}(\mu_i - y_i)\,\boldsymbol{x}_i$$

and

$$\mathbf{H} = \frac{1}{n}\sum_{i=1}^{n}\mu_i(1-\mu_i)\,\boldsymbol{x}_i\,\boldsymbol{x}_i^\top\,,$$

where

$$\mu_i = \frac{1}{1+\mathrm{e}^{-\boldsymbol{x}_i^\top\boldsymbol{\beta}}}\,.$$

To compute the gradient, we proceed as follows.

$$\begin{aligned}
\nabla\, r_\tau &= \nabla\,\frac{1}{n}\sum_{i=1}^{n}\left[(1-y_i)\boldsymbol{x}_i^\top\boldsymbol{\beta} + \ln\left(1+\mathrm{e}^{-\boldsymbol{x}_i^T\boldsymbol{\beta}}\right)\right]\\
&= \frac{1}{n}\sum_{i=1}^{n}\left[(1-y_i)(\nabla\,\boldsymbol{x}_i^\top\boldsymbol{\beta}) + (\nabla\,\ln\left(1+\mathrm{e}^{-\boldsymbol{x}_i^T\boldsymbol{\beta}}\right))\right]\\
&= \frac{1}{n}\sum_{i=1}^{n}\left[(1-y_i)\,\boldsymbol{x}_i + \frac{1}{1+\mathrm{e}^{-\boldsymbol{x}_i^T\boldsymbol{\beta}}}\nabla\,(1+\mathrm{e}^{-\boldsymbol{x}_i^T\boldsymbol{\beta}})\right]\\
&= \frac{1}{n}\sum_{i=1}^{n}\left[(1-y_i)\,\boldsymbol{x}_i - \frac{\mathrm{e}^{-\boldsymbol{x}_i^T\boldsymbol{\beta}}}{1+\mathrm{e}^{-\boldsymbol{x}_i^T\boldsymbol{\beta}}}\boldsymbol{x}_i\right]\\
&= \frac{1}{n}\sum_{i=1}^{n}\left[(1-y_i)\,\boldsymbol{x}_i - (1-\mu_i)\,\boldsymbol{x}_i\right]\\
&= \frac{1}{n}\sum_{i=1}^{n}(\mu_i - y_i)\,\boldsymbol{x}_i\,,
\end{aligned}$$

as required.

To compute the Hessian, we proceed as follows.

$$
\begin{aligned}
\mathbf{H} &= \frac{\partial}{\partial \boldsymbol{\beta}} \nabla r_\tau \\
&= \frac{\partial}{\partial \boldsymbol{\beta}} \left[\frac{1}{n} \sum_{i=1}^{n} (\mu_i - y_i)\, \boldsymbol{x}_i \right] \\
&= \frac{1}{n} \sum_{i=1}^{n} \left(\frac{\partial}{\partial \boldsymbol{\beta}} \mu_i \, \boldsymbol{x}_i \right) \\
&= \frac{1}{n} \sum_{i=1}^{n} \left(\boldsymbol{x}_i \, \frac{\partial}{\partial \boldsymbol{\beta}} \mu_i \right) \\
&= \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{x}_i \left(- \frac{1}{(1 + \mathrm{e}^{-\boldsymbol{x}_i^\top \boldsymbol{\beta}})^2} \times \mathrm{e}^{-\boldsymbol{x}_i^\top \boldsymbol{\beta}} \times (-\boldsymbol{x}_i^\top) \right) \\
&= \frac{1}{n} \sum_{i=1}^{n} -\mu_i^2 \times \frac{(1-\mu_i)}{\mu_i} \times (-\boldsymbol{x}_i \boldsymbol{x}_i^\top) \\
&= \frac{1}{n} \sum_{i=1}^{n} \mu_i (1-\mu_i) \boldsymbol{x}_i \boldsymbol{x}_i^\top \,,
\end{aligned}
$$

as required.

8. In the *Tobit regression* model with normally distributed errors, the response is modelled as:

$$
Y_i = \begin{cases} Z_i, & \text{if } u_i < Z_i \\ u_i, & \text{if } Z_i \leqslant u_i \end{cases}, \qquad \boldsymbol{Z} \sim \mathcal{N}(\mathbf{X}\boldsymbol{\beta}, \sigma^2 \mathbf{I}_n),
$$

where the model matrix $\mathbf{X}$ and the thresholds $u_1, \ldots, u_n$ are given. Typically, $u_i = 0, i = 1, \ldots, n$. Suppose we wish to estimate $\boldsymbol{\theta} := (\boldsymbol{\beta}, \sigma^2)$ via the Expectation–Maximization method, similar to the censored data Example 4.2. Let $\boldsymbol{y} = [y_1, \ldots, y_n]^\top$ be the vector of observed data.

(a) Show that the likelihood function of $\boldsymbol{y}$ is:

$$
g(\boldsymbol{y} \,|\, \boldsymbol{\theta}) = \prod_{i: y_i > u_i} \varphi_{\sigma^2}(y_i - \boldsymbol{x}_i^\top \boldsymbol{\beta}) \times \prod_{i: y_i = u_i} \Phi((u_i - \boldsymbol{x}_i^\top \boldsymbol{\beta})/\sigma),
$$

where Φ is the cdf of the $\mathcal{N}(0,1)$ distribution and φ_{σ^2} the pdf of the $\mathcal{N}(0, \sigma^2)$ distribution.

Solution:

(b) Let $\overline{\boldsymbol{y}}$ and $\underline{\boldsymbol{y}}$ be vectors that collect all $y_i > u_i$ and $y_i = u_i$, respectively. Denote the corresponding matrix of predictors by $\overline{\mathbf{X}}$ and $\underline{\mathbf{X}}$, respectively. For each observation $y_i = u_i$ introduce a latent variable z_i and collect these into a vector $\boldsymbol{z}$. For the same indices i collect the corresponding u_i into a vector $\boldsymbol{c}$. Show that the complete-data likelihood is given by

$$
g(\boldsymbol{y}, \boldsymbol{z} \,|\, \boldsymbol{\theta}) = \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(- \frac{\|\overline{\boldsymbol{y}} - \overline{\mathbf{X}}\boldsymbol{\beta}\|^2}{2\sigma^2} - \frac{\|\boldsymbol{z} - \underline{\mathbf{X}}\boldsymbol{\beta}\|^2}{2\sigma^2} \right) \mathbb{1}\{\boldsymbol{z} \leqslant \boldsymbol{c}\}.
$$

Solution:

(c) For the E-step, show that, for a fixed $\boldsymbol{\theta}$,

$$g(\boldsymbol{z} \,|\, \boldsymbol{y}, \boldsymbol{\theta}) = \prod_i g(z_i \,|\, \boldsymbol{y}, \boldsymbol{\theta}),$$

where each $g(z_i \,|\, \boldsymbol{y}, \boldsymbol{\theta})$ is the pdf of the $\mathcal{N}(\{\underline{\mathbf{X}}\boldsymbol{\beta}\}_i, \sigma^2)$ distribution, truncated to the interval $(-\infty, c_i]$.

Solution:

(d) For the M-step, compute the expectation of the complete log-likelihood

$$-\frac{n}{2}\ln\sigma^2 - \frac{n}{2}\ln(2\pi) - \frac{\|\overline{\boldsymbol{y}} - \overline{\mathbf{X}}\boldsymbol{\beta}\|^2}{2\sigma^2} - \frac{\mathbb{E}\|\boldsymbol{Z} - \underline{\mathbf{X}}\boldsymbol{\beta}\|^2}{2\sigma^2}.$$

Then, derive the formulas for $\boldsymbol{\beta}$ and σ^2 that maximize the expectation of the complete log-likelihood.

Solution:

9. **Analysis of Women's Wages via EM Algorithm.** Dowload dataset `WomenWage.csv` from the book's website. This dataset is a tidied-up version of the women's wage dataset from [91]. The first column of the data (`hours`) is the response variable Y. It shows the hours spent in the labor force by married women in the 1970's. We want to understand what factors determine the participation rate of women in the labor force. The predictor variables are:

Table 5.7: Features for the women's wage data set.

Feature	Description
`kidslt6`	Number of children younger than 6 years
`kidsge6`	Number of children older than 6 years
`age`	Age of the married woman
`educ`	Number of years of formal education
`exper`	Number of years of "work experience"
`nwifeinc`	Non-wife income, that is, the income of the husband
`expersq`	The square of `exper`, to capture any nonlinear relationships

We observe that some of the responses are $Y = 0$, that is, some women did not participate in the labor force. For this reason, we model the data using the Tobit regression model, in which the response Y is given as:

$$Y_i = \begin{cases} Z_i, & \text{if } 0 < Z_i \\ 0, & \text{if } Z_i \leqslant 0 \end{cases}, \qquad \boldsymbol{Z} \sim \mathcal{N}(\mathbf{X}\boldsymbol{\beta}, \sigma^2\mathbf{I}).$$

With $\boldsymbol{\theta} = (\boldsymbol{\beta}, \sigma^2)$, the likelihood function for data $\boldsymbol{y} = (y_1, \ldots, y_n)$ is:

$$g(\boldsymbol{y} \,|\, \boldsymbol{\theta}) = \prod_{i:y_i>0} \varphi_{\sigma^2}(y_i - \boldsymbol{x}_i^\top\boldsymbol{\beta}) \times \prod_{i:y_i=0} \Phi((u_i - \boldsymbol{x}_i^\top\boldsymbol{\beta})/\sigma),$$

where Φ is the standard normal cdf. In the previous question, we derived the EM algorithm for maximizing the log-likelihood.

(a) Write down the EM algorithm in pseudocode as it applies to this Tobit regression.

(b) Implement the EM algorithm pseudocode in Python. Comment on which factor you think is important in determining the labor participation rate of women living in the USA in the 1970's.

Solution: The pseudocode is:

Algorithm 5.0.1: EM algorithm for Tobit model

```
input: Initial guesstimate $\boldsymbol{\theta}_1 = (\boldsymbol{\beta}_1, \sigma_1^2)$. Responses $\overline{\boldsymbol{y}}$ and $\underline{\boldsymbol{y}}$ that collect all $y_i > 0$
       and $y_i = 0$, respectively, with corresponding predictors $\overline{\mathbf{X}}$ and $\underline{\mathbf{X}}$.
output: Approximate maximum likelihood estimate $\boldsymbol{\theta}_t$.
$t \leftarrow 1$
$n' \leftarrow \dim(\underline{\boldsymbol{y}})$
$\mathbf{C} \leftarrow (\overline{\mathbf{X}}^\top \overline{\mathbf{X}} + \underline{\mathbf{X}}^\top \underline{\mathbf{X}})^{-1}$
while $\boldsymbol{\theta}_t$ has not converged do
    Expectation Step:
    for $i = 1$ to $n'$ do
        $$\mu \leftarrow \boldsymbol{\beta}_t^\top \underline{\boldsymbol{x}}_i$$
        $$a_i \leftarrow \mu - \sigma_t \frac{\varphi(\mu/\sigma_t)}{\Phi(-\mu/\sigma_t)}$$
        $$b_i^2 \leftarrow \mu^2 + \sigma_t^2 - \mu\sigma_t \frac{\varphi(\mu/\sigma_t)}{\Phi(-\mu/\sigma_t)}$$
    $\boldsymbol{\mu}^\top \leftarrow (a_1, \ldots, a_m)$
    $\varsigma^2 \leftarrow \sum_{i=1}^{n'} (b_i^2 - a_i^2)$
    Maximization Step:
        $$\boldsymbol{\beta}_{t+1} \leftarrow \mathbf{C}(\overline{\mathbf{X}}^\top \overline{\boldsymbol{y}} + \underline{\mathbf{X}}^\top \boldsymbol{\mu})$$
        $$\sigma_{t+1}^2 \leftarrow \frac{\|\overline{\boldsymbol{y}} - \overline{\mathbf{X}}\boldsymbol{\beta}_{t+1}\|^2 + \|\boldsymbol{\mu} - \underline{\mathbf{X}}\boldsymbol{\beta}_{t+1}\|^2 + \varsigma^2}{n}$$
    $\boldsymbol{\theta}_{t+1} \leftarrow (\boldsymbol{\beta}_{t+1}, \sigma_{t+1}^2)$
    $t \leftarrow t + 1$
return $\boldsymbol{\theta}_t$
```

The following Python code implements the EM algorithm and displays the estimated coefficients $\boldsymbol{\beta}$.

```python
import numpy as np
from numpy import random
from scipy.stats import norm
# load data
data = np.genfromtxt('WomenWage.csv',delimiter=',', skip_header
    =1)
```

```
X = np.array(data[:,1:]);
n,p=X.shape
y = data[:,0]
yu=y[y!=0].reshape(-1,1)
Xu=X[y!=0,:]
Xd=X[y==0,:]

C=Xu.T@Xu+Xd.T@Xd;C=np.linalg.inv(C);

b=random.rand(p).reshape(p,1);s=120; tol=1 # inital values
while tol>10**-6:
    # E-step
    m=Xd@b
    a=m-s*norm.pdf(m/s)/norm.cdf(-m/s)
    b2=m**2+s**2-m*s*norm.pdf(m/s)/norm.cdf(-m/s)
    ss=np.sum(b2-a**2)
    # M-step
    bnew=C@(Xu.T@yu+Xd.T@m)
    tol=np.linalg.norm(bnew-b); b=bnew;
    s=np.sum((yu-Xu@b)**2)+np.sum((m-Xd@b)**2)+ss
    s=np.sqrt(s/n)
print(b)
```

```
[[-187.58025183]
 [   7.41516791]
 [   6.23710946]
 [  35.65870167]
 [  72.53461786]
 [  -2.02751486]
 [  -1.37320273]]
```

10. Let $\mathbf{P}$ be a projection matrix. Show that the diagonal elements of $\mathbf{P}$ all lie in the interval $[0, 1]$. In particular, for $\mathbf{P} = \mathbf{X}\mathbf{X}^+$ in Theorem 5.1, the leverage value $p_i := \mathbf{P}_{ii}$ satisfies $0 \leqslant p_i \leqslant 1$ for all i.

    Solution:

11. Consider the linear model $\boldsymbol{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$ in (5.8), with $\mathbf{X}$ being the $n \times p$ model matrix and $\boldsymbol{\varepsilon}$ having expectation vector $\mathbf{0}$ and covariance matrix $\sigma^2\mathbf{I}_n$. Suppose that $\widehat{\boldsymbol{\beta}}_{-i}$ is the least squares estimate obtained by omitting the i-th observation, Y_i; that is,

$$\widehat{\boldsymbol{\beta}}_{-i} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \sum_{j \neq i} (Y_j - \boldsymbol{x}_j^\top \boldsymbol{\beta})^2,$$

where $\boldsymbol{x}_j^\top$ is the j-th row of $\mathbf{X}$. Let $\widehat{Y}_{-i} = \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}_{-i}$ be the corresponding fitted value at $\boldsymbol{x}_i$. Also, define $\boldsymbol{B}_i$ as the least squares estimator of $\boldsymbol{\beta}$ based on the response data

$$\boldsymbol{Y}^{(i)} := [Y_1, \ldots, Y_{i-1}, \widehat{Y}_{-i}, Y_{i+1}, \ldots, Y_n]^\top.$$

(a) Prove that $\widehat{\boldsymbol{\beta}}_{-i} = \boldsymbol{B}_i$; that is, the linear model obtained from fitting all responses except the i-th is the same as the one obtained from fitting the data $\boldsymbol{Y}^{(i)}$.

(b) Use the previous result to verify that

$$Y_i - \widehat{Y}_{-i} = (Y_i - \widehat{Y}_i)/(1 - \mathbf{P}_{ii}) ,$$

where $\mathbf{P} = \mathbf{X}\mathbf{X}^{+}$ is the projection matrix onto the columns of $\mathbf{X}$. Hence, deduce the PRESS formula in Theorem **5.1**.

Solution: We have that

$$\sum_{j \neq i} (Y_j - \boldsymbol{x}_j^\top \widehat{\boldsymbol{\beta}}_{-i})^2 \leqslant \sum_{j \neq i} (Y_j - \boldsymbol{x}_j^\top \boldsymbol{\beta})^2$$

for any $\boldsymbol{\beta}$. It follows that

$$(\widehat{Y}_{-i} - \boldsymbol{x}_i^\top \boldsymbol{\beta})^2 + \sum_{j \neq i} (Y_j - \boldsymbol{x}_j^\top \widehat{\boldsymbol{\beta}}_{-i})^2 \leqslant \sum_{j} (Y_j^{(i)} - \boldsymbol{x}_j^\top \boldsymbol{\beta})^2. \tag{5.2}$$

We also have

$$\sum_{j} (Y_j^{(i)} - \boldsymbol{x}_j^\top \boldsymbol{B}_i)^2 \leqslant \sum_{j} (Y_j^{(i)} - \boldsymbol{x}_j^\top \boldsymbol{\beta})^2 = \sum_{j \neq i} (Y_j - \boldsymbol{x}_j^\top \boldsymbol{\beta})^2 + (\widehat{Y}_{-i} - \boldsymbol{x}_i^\top \boldsymbol{\beta})^2 \tag{5.3}$$

for any $\boldsymbol{\beta}$. Evaluating (**??**) at $\boldsymbol{\beta} = \boldsymbol{B}_i$ and (**??**) at $\boldsymbol{\beta} = \widehat{\boldsymbol{\beta}}_{-i}$ we obtain:

$$(\widehat{Y}_{-i} - \boldsymbol{x}_i^\top \boldsymbol{B}_i)^2 + \sum_{j \neq i} (Y_j - \boldsymbol{x}_j^\top \widehat{\boldsymbol{\beta}}_{-i})^2 \leqslant \sum_{j \neq i} (Y_j - \boldsymbol{x}_j^\top \widehat{\boldsymbol{\beta}}_{-i})^2 + (\widehat{Y}_{-i} - \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}_{-i})^2.$$

But, since $\widehat{Y}_{-i} = \boldsymbol{x}_i^\top \widehat{\boldsymbol{\beta}}_{-i}$, this implies that $(\widehat{Y}_{-i} - \boldsymbol{x}_i^\top \boldsymbol{B}_i)^2 \leqslant 0$ or, equivalently, that $0 = \boldsymbol{x}_i^\top (\boldsymbol{B}_i - \widehat{\boldsymbol{\beta}}_{-i})$, which is true for any arbitrary predictor $\boldsymbol{x}_i$ only if $\boldsymbol{B}_i = \widehat{\boldsymbol{\beta}}_{-i}$.
For the second part, observe that

$$\boldsymbol{Y} - \boldsymbol{Y}^{(i)} = (Y_i - \widehat{Y}_{-i}) \boldsymbol{u}_i ,$$

where $\boldsymbol{u}_i$ is the unit length vector with an entry 1 in the i-th position. Since,

$$\begin{aligned}
\widehat{\boldsymbol{Y}} &= \mathbf{P}\boldsymbol{Y} \\
&= \mathbf{P}(\boldsymbol{Y}^{(i)} + (Y_i - \widehat{Y}_{-i})\boldsymbol{u}_i) \\
&= \mathbf{X}\boldsymbol{B}_i + (Y_i - \widehat{Y}_{-i})\mathbf{P}\boldsymbol{u}_i \\
&= \mathbf{X}\boldsymbol{\beta}_{-i} + (Y_i - \widehat{Y}_{-i})\mathbf{P}\boldsymbol{u}_i ,
\end{aligned}$$

we have

$$\boldsymbol{u}_i^\top \widehat{\boldsymbol{Y}} = \boldsymbol{u}_i^\top (\mathbf{X}\boldsymbol{\beta}_{-i}) + (Y_i - \widehat{Y}_{-i}) \boldsymbol{u}_i^\top \mathbf{P} \boldsymbol{u}_i .$$

In other words,

$$\widehat{Y}_i = \widehat{Y}_{-i} + (Y_i - \widehat{Y}_{-i}) \mathbf{P}_{ii} ,$$

which after rearrangement becomes $Y_i - \widehat{Y}_i = (1 - \mathbf{P}_{ii})(Y_i - \widehat{Y}_{-i})$. Therefore,

$$\sum_i (Y_i - \widehat{Y}_{-i})^2 = \sum_i \left(\frac{Y_i - \widehat{Y}_i}{1 - \mathbf{P}_{ii}} \right)^2 .$$

12. Take the linear model $\boldsymbol{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$, where $\mathbf{X}$ is an $n \times p$ model matrix, $\boldsymbol{\varepsilon} = \mathbf{0}$, and $\mathbb{C}\text{ov}(\boldsymbol{\varepsilon}) = \sigma^2 \mathbf{I}_n$. Let $\mathbf{P} = \mathbf{X}\mathbf{X}^+$ be the projection matrix onto the columns of $\mathbf{X}$.

    (a) Using the properties of the pseudo-inverse (see Definition A.2), show that $\mathbf{P}\mathbf{P}^\top = \mathbf{P}$.

    Solution:

    (b) Let $\boldsymbol{E} = \boldsymbol{Y} - \widehat{\boldsymbol{Y}}$ be the (random) vector of residuals, where $\widehat{\boldsymbol{Y}} = \mathbf{P}\boldsymbol{Y}$. Show that the i-th residual has a normal distribution with expectation 0 and variance $\sigma^2 \mathbf{P}_{ii}$ (that is, σ^2 times the i-th leverage).

    Solution:

    (c) Show that σ^2 can be unbiasedly estimated via

    $$S^2 := \frac{1}{n-p} \|\boldsymbol{Y} - \widehat{\boldsymbol{Y}}\|^2 = \frac{1}{n-p} \|\boldsymbol{Y} - \mathbf{X}\widehat{\boldsymbol{\beta}}\|^2. \tag{5.44}$$

    [Hint: use the cyclic property of the trace as in Example 2.3.]

    Solution:

13. Consider a normal linear model $\boldsymbol{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$, where $\mathbf{X}$ is an $n \times p$ model matrix and $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_n)$. Exercise 12 shows that for any such model the i-th standardized residual $E_i / (\sigma \sqrt{1 - \mathbf{P}_{ii}})$ has a standard normal distribution. This motivates the use of the leverage $\mathbf{P}_{ii}$ to assess whether the i-th observation is an outlier depending on the size of the i-th residual relative to $\sqrt{1 - \mathbf{P}_{ii}}$. A more robust approach is to include an estimate for σ using all data except the i-th observation. This gives rise to the *studentized residual* T_i, defined as

    $$T_i = \frac{E_i}{S_{-i}\sqrt{1 - \mathbf{P}_{ii}}},$$

    where S_{-i} is an estimate of σ obtained by fitting all the observations except the i-th and $E_i = Y_i - \widehat{Y}_i$ is the i-th (random) residual. Exercise 12 shows that we can take, for example,

    $$S^2_{-i} = \frac{1}{n-1-p} \|\boldsymbol{Y}_{-i} - \mathbf{X}_{-i}\widehat{\boldsymbol{\beta}}_{-i}\|^2, \tag{5.45}$$

    where $\mathbf{X}_{-i}$ is the model matrix $\mathbf{X}$ with the i-th row removed, is an unbiased estimator of σ^2. We wish to compute S^2_{-i} efficiently, using S^2 in (5.44), as the latter will typically be available once we have fitted the linear model. To this end, define $\boldsymbol{u}_i$ as the i-th unit vector $[0, \ldots, 0, 1, 0, \ldots, 0]^\top$, and let

    $$\boldsymbol{Y}^{(i)} := \boldsymbol{Y} - (Y_i - \widehat{Y}_{-i})\boldsymbol{u}_i = \boldsymbol{Y} - \frac{E_i}{1 - \mathbf{P}_{ii}}\boldsymbol{u}_i,$$

    where we have used the fact that $Y_i - \widehat{Y}_{-i} = E_i/(1 - \mathbf{P}_{ii})$, as derived in the proof of Theorem 5.1. Now apply Exercise 11 to prove that

    $$S^2_{-i} = \frac{(n-p)\,S^2 - E_i^2/(1 - \mathbf{P}_{ii})}{n - p - 1}.$$

Solution: Define $C := E_i/(1 - \mathbf{P}_{ii})$. Recall that $\|\boldsymbol{Y} - \mathbf{X}\widehat{\boldsymbol{\beta}}\|^2 = \boldsymbol{Y}^\top(\mathbf{I} - \mathbf{P})\boldsymbol{Y}$, where $\mathbf{P}$ is the projection matrix. Then, from Exercise 11 we have

$$\begin{aligned}(n-p)S^2 - (n-p-1)S_{-i}^2 &= \boldsymbol{Y}^\top(\mathbf{I} - \mathbf{P})\boldsymbol{Y} - \boldsymbol{Y}^{(i)\top}(\mathbf{I} - \mathbf{P})\boldsymbol{Y}^{(i)} \\ &= 2C\boldsymbol{u}_i^\top(\mathbf{I} - \mathbf{P})\boldsymbol{Y} - C^2\boldsymbol{u}_i^\top(\mathbf{I} - \mathbf{P})\boldsymbol{u}_i \\ &= 2C(Y_i - \widehat{Y}_i) - C^2(1 - \mathbf{P}_{ii}) \\ &= 2\frac{E_i^2}{1 - \mathbf{P}_{ii}} - \frac{E_i^2}{1 - \mathbf{P}_{ii}} \\ &= \frac{E_i^2}{1 - \mathbf{P}_{ii}}.\end{aligned}$$

14. Using the notation from Exercises 11–13, *Cook's distance* for observation i is defined as

$$D_i := \frac{\|\widehat{\boldsymbol{Y}} - \widehat{\boldsymbol{Y}}^{(i)}\|^2}{p\,S^2}.$$

It measures the change in the fitted values when the i-th observation is removed, relative to the residual variance of the model (estimated via S^2).

By using similar arguments as those in Exercise 13, show that

$$D_i = \frac{\mathbf{P}_{ii}\,E_i^2}{(1 - \mathbf{P}_{ii})^2 p\,S^2}.$$

It follows that there is no need to "omit and refit" the linear model in order to compute Cook's distance for the i-th response.

Solution:

15. Prove that if we add an additional feature to the general linear model, then R^2, the coefficient of determination, is necessarily non-decreasing in value and hence cannot be used to compare models with different number of predictors.

Solution: Suppose we initially have a model with p_1 features, giving a residual of $\|\boldsymbol{Y} - \boldsymbol{Y}^{(1)}\|^2$ and coefficient of determination R_1^2, and we add one or more features to obtain the new residual $\|\boldsymbol{Y} - \boldsymbol{Y}^{(2)}\|^2$ and coefficient of determination R_2^2. From the Pythagorean identity

$$\|\boldsymbol{Y} - \boldsymbol{Y}^{(2)}\|^2 + \|\boldsymbol{Y}^{(1)} - \boldsymbol{Y}^{(2)}\|^2 = \|\boldsymbol{Y} - \boldsymbol{Y}^{(1)}\|^2,$$

we have that

$$-\|\boldsymbol{Y} - \boldsymbol{Y}^{(1)}\|^2 \leqslant -\|\boldsymbol{Y} - \boldsymbol{Y}^{(2)}\|^2.$$

In other words,

$$\begin{aligned}
R_1^2 &= \frac{\|\boldsymbol{Y}^{(1)} - \bar{Y}\mathbf{1}\|^2}{\|\boldsymbol{Y} - \bar{Y}\mathbf{1}\|^2} \\
&= \frac{\|\boldsymbol{Y} - \bar{Y}\mathbf{1}\|^2 - \|\boldsymbol{Y} - \boldsymbol{Y}^{(1)}\|^2}{\|\boldsymbol{Y} - \bar{Y}\mathbf{1}\|^2} \\
&\leqslant \frac{\|\boldsymbol{Y} - \bar{Y}\mathbf{1}\|^2 - \|\boldsymbol{Y} - \boldsymbol{Y}^{(2)}\|^2}{\|\boldsymbol{Y} - \bar{Y}\mathbf{1}\|^2} \\
&= \frac{\|\boldsymbol{Y}^{(2)} - \bar{Y}\mathbf{1}\|^2}{\|\boldsymbol{Y} - \bar{Y}\mathbf{1}\|^2} = R_2^2.
\end{aligned}$$

Hence, adding more predictors to the model never reduces the coefficient of variation, and is not suitable for model selection (it will always choose the most complex model).

16. Let $\boldsymbol{X} := [X_1, \ldots, X_n]^\top$ and $\boldsymbol{\mu} := [\mu_1, \ldots, \mu_n]^\top$. In the fundamental Theorem C.9, we use the fact that if $X_i \sim \mathcal{N}(\mu_i, 1)$, $i = 1, \ldots, n$ are independent, then $\|\boldsymbol{X}\|^2$ has (per definition) a noncentral χ_n^2 distribution. Show that $\|\boldsymbol{X}\|^2$ has moment generating function

$$\frac{\mathrm{e}^{t\|\boldsymbol{\mu}\|^2/(1-2t)}}{(1-2t)^{n/2}}, \qquad t < 1/2,$$

and so the distribution of $\|\boldsymbol{X}\|^2$ depends on $\boldsymbol{\mu}$ only through the norm $\|\boldsymbol{\mu}\|$.

Solution:

17. Carry out a logistic regression analysis on a (partial) *wine* dataset classification problem. The data can be loaded using the following code.

```
from sklearn import datasets
import numpy as np
data = datasets.load_wine()
X = data.data[:, [9,10]]
y = np.array(data.target==1,dtype=np.uint)
X = np.append(np.ones(len(X)).reshape(-1,1),X,axis=1)
```

The model matrix has three features, including the constant feature. Instead of using Newton's method (5.39) to estimate $\boldsymbol{\beta}$, implement a simple gradient descent procedure

$$\boldsymbol{\beta}_t = \boldsymbol{\beta}_{t-1} - \alpha \nabla r_\tau(\boldsymbol{\beta}_{t-1}),$$

with learning rate $\alpha = 0.0001$, and run it for 10^6 steps. Your procedure should deliver three coefficients; one for the intercept and the rest for the explanatory variables. Solve the same problem using the `Logit` method of `statsmodels.api` and compare the results.

Solution:

```
from sklearn import datasets
import numpy as np
data = datasets.load_wine()
X = data.data[:, [9,10]]
y = np.array(data.target==1,dtype=np.uint)
X = np.append(np.ones(len(X)).reshape(-1,1),X,axis=1)
#######################################################
def sigmoid(z):
    return 1 / (1 + np.exp(-z))

theta = np.zeros(3)
lr = 0.0001

for i in range(1000000):
    g = sigmoid(np.dot(X, theta))
    gradient = np.dot(X.T, (g - y))
    theta -= lr * gradient

print("gradient descent: ", theta)


#######################################################
import statsmodels.api as sm
model = sm.Logit(y, X)
result = model.fit()
print("statsmodels.api: ", result.params)
```

```
gradient descent:  [ 6.82750898 -2.14602587  1.940977  ]

statsmodels.api:  [ 6.8275091  -2.14602588  1.94097692]
```

18. Consider again Example 5.10, where we train the learner via the Newton iteration (5.39). If $\mathbf{X}^\top := [\boldsymbol{x}_1, \dots, \boldsymbol{x}_n]$ defines the matrix of predictors and $\boldsymbol{\mu}_t := \boldsymbol{h}(\mathbf{X}\boldsymbol{\beta}_t)$, then the gradient (5.37) and Hessian (5.38) for Newton's method can be written as: ☞ 205

$$\nabla r_\tau(\boldsymbol{\beta}_t) = \frac{1}{n}\mathbf{X}^\top(\boldsymbol{\mu}_t - \boldsymbol{y}) \quad \text{and} \quad \mathbf{H}(\boldsymbol{\beta}_t) = \frac{1}{n}\mathbf{X}^\top \mathbf{D}_t \mathbf{X},$$

where $\mathbf{D}_t := \text{diag}(\boldsymbol{\mu}_t \odot (\mathbf{1} - \boldsymbol{\mu}_t))$ is a diagonal matrix. Show that the Newton iteration (5.39) can be written as the *iterative reweighted least squares* method:

ITERATIVE REWEIGHTED LEAST SQUARES ☞ 349

$$\boldsymbol{\beta}_t = \underset{\boldsymbol{\beta}}{\text{argmin}}\, (\widetilde{\boldsymbol{y}}_{t-1} - \mathbf{X}\boldsymbol{\beta})^\top \mathbf{D}_{t-1} (\widetilde{\boldsymbol{y}}_{t-1} - \mathbf{X}\boldsymbol{\beta}),$$

where $\widetilde{\boldsymbol{y}}_{t-1} := \mathbf{X}\boldsymbol{\beta}_{t-1} + \mathbf{D}_{t-1}^{-1}(\boldsymbol{y} - \boldsymbol{\mu}_{t-1})$ is the so-called *adjusted response*. [Hint: use the fact that $(\mathbf{M}^\top\mathbf{M})^{-1}\mathbf{M}^\top \boldsymbol{z}$ is the minimizer of $\|\mathbf{M}\boldsymbol{\beta} - \boldsymbol{z}\|^2$.]

Solution:

19. In *multi-output linear regression*, the response variable is a real-valued vector of dimension, say, m. Similar to (5.8), the model can be written in matrix notation:

$$\mathbf{Y} = \mathbf{X}\mathbf{B} + \begin{bmatrix} \boldsymbol{\varepsilon}_1^\top \\ \vdots \\ \boldsymbol{\varepsilon}_n^\top \end{bmatrix},$$

where:

- $\mathbf{Y}$ is an $n \times m$ matrix of n independent responses (stored as row vectors of length m);
- $\mathbf{X}$ is the usual $n \times p$ model matrix;
- $\mathbf{B}$ is an $p \times m$ matrix of model parameters;
- $\boldsymbol{\varepsilon}_1, \dots, \boldsymbol{\varepsilon}_n \in \mathbb{R}^m$ are independent error terms with $\mathbb{E}\boldsymbol{\varepsilon} = \mathbf{0}$ and $\mathbb{E}\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^\top = \Sigma$.

We wish to learn the matrix parameters $\mathbf{B}$ and Σ from the training set $\{\mathbf{Y}, \mathbf{X}\}$. To this end, consider minimizing the training loss:

$$\frac{1}{n}\operatorname{tr}\left((\mathbf{Y}-\mathbf{X}\mathbf{B})\Sigma^{-1}(\mathbf{Y}-\mathbf{X}\mathbf{B})^\top\right),$$

where $\operatorname{tr}(\cdot)$ is the trace of a matrix.

(a) Show that the minimizer of the training loss, denoted $\widehat{\mathbf{B}}$, satisfies the normal equations:

$$\mathbf{X}^\top\mathbf{X}\,\widehat{\mathbf{B}} = \mathbf{X}^\top\mathbf{Y}.$$

Solution: By Theorem A.1, we have

$$\begin{aligned}
\operatorname{tr}\left((\mathbf{Y}-\mathbf{X}\mathbf{B})\Sigma^{-1}(\mathbf{Y}-\mathbf{X}\mathbf{B})^\top\right) &= \operatorname{tr}\left(\Sigma^{-1}(\mathbf{Y}-\mathbf{X}\mathbf{B})^\top(\mathbf{Y}-\mathbf{X}\mathbf{B})\right)\\
&= \operatorname{tr}\left(\Sigma^{-1}(\mathbf{Y}^\top\mathbf{Y} - 2\mathbf{Y}^\top\mathbf{X}\mathbf{B} + \mathbf{B}^\top\mathbf{X}^\top\mathbf{X}\mathbf{B})\right)\\
&= \operatorname{tr}(\Sigma^{-1}\mathbf{Y}^\top\mathbf{Y}) - 2\operatorname{tr}(\mathbf{Y}^\top\mathbf{X}\mathbf{B}) + \operatorname{tr}(\mathbf{X}^\top\mathbf{X}\mathbf{B}\mathbf{B}^\top).
\end{aligned}$$

Suppose that $\mathbf{U}$ is an all zero matrix with a one in the (i, j)-th position. Then,

$$\begin{aligned}
\lim_{\delta\downarrow 0}\frac{\operatorname{tr}(\mathbf{X}^\top\mathbf{X}(\mathbf{B}+\delta\mathbf{U})(\mathbf{B}+\delta\mathbf{U})^\top) - \operatorname{tr}(\mathbf{X}^\top\mathbf{X}\mathbf{B}\mathbf{B}^\top)}{\delta} &= \operatorname{tr}(\mathbf{X}^\top\mathbf{X}(\mathbf{U}\mathbf{B}^\top + \mathbf{B}\mathbf{U}^\top))\\
&= [\mathbf{B}^\top\mathbf{X}^\top\mathbf{X}]_{ji} + [\mathbf{X}^\top\mathbf{X}\mathbf{B}]_{ij}
\end{aligned}$$

gives the derivative of $\operatorname{tr}(\mathbf{X}^\top\mathbf{X}\mathbf{B}\mathbf{B}^\top)$ with respect to the (i, j)-th element of $\mathbf{B}$. Clearly,

$$\frac{\partial\operatorname{tr}(\mathbf{X}^\top\mathbf{X}\mathbf{B}\mathbf{B}^\top)}{\partial\mathbf{B}} = 2\mathbf{X}^\top\mathbf{X}\mathbf{B}.$$

Hence, taking a scalar/matrix derivative of the training loss with respect to $\mathbf{B}$, we obtain:

$$\frac{-2\mathbf{X}^\top\mathbf{Y} + 2\mathbf{X}^\top\mathbf{X}\mathbf{B}}{n}.$$

Hence, the least-squares estimator $\widehat{\mathbf{B}}$ satisfies the normal equations.

(b) Noting that

$$(\mathbf{Y}-\mathbf{X}\mathbf{B})^\top(\mathbf{Y}-\mathbf{X}\mathbf{B}) = \sum_{i=1}^{n}\boldsymbol{\varepsilon}_i\boldsymbol{\varepsilon}_i^\top,$$

explain why

$$\widehat{\Sigma} := \frac{(\mathbf{Y}-\mathbf{X}\widehat{\mathbf{B}})^\top(\mathbf{Y}-\mathbf{X}\widehat{\mathbf{B}})}{n}$$

is a method-of-moments estimator of Σ, just like the one given in **(5.10)**.

Solution:

Since $\mathbb{E}\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^\top = \Sigma$, the average

$$\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{\varepsilon}_i\boldsymbol{\varepsilon}_i^\top = \frac{(\mathbf{Y}-\mathbf{X}\mathbf{B})^\top(\mathbf{Y}-\mathbf{X}\mathbf{B})}{n}$$

is an unbiased estimator of Σ. In practice, we do not know the parameter matrix $\mathbf{B}$, but we can replace it by its least-squares estimator to obtain the (biased) estimator of Σ:

$$\widehat{\Sigma} = \frac{(\mathbf{Y}-\mathbf{X}\widehat{\mathbf{B}})^\top(\mathbf{Y}-\mathbf{X}\widehat{\mathbf{B}})}{n}.$$

This is the same argument used to derive **(5.10)**.

CHAPTER 6

KERNEL METHODS

1. Let $\mathcal{G}$ be a RKHS with reproducing kernel κ. Show that κ is a positive semi-definite kernel.

   Solution: Using the properties of the inner product $\langle \cdot, \cdot \rangle$ on $\mathcal{G}$, we have for every $n \geqslant 1$ and $\alpha_1, \ldots, \alpha_n \in \mathbb{R}$, $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathscr{X}$,

$$\begin{aligned}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)\alpha_j &= \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \langle \kappa_{\boldsymbol{x}_i}, \kappa_{\boldsymbol{x}_j}\rangle \alpha_j \\ &= \textstyle\sum_{i=1}^{n} \alpha_i \langle \kappa_{\boldsymbol{x}_i}, \sum_{j=1}^{n} \alpha_j \kappa_{\boldsymbol{x}_j}\rangle \\ &= \textstyle\langle \sum_{i=1}^{n} \alpha_i \kappa_{\boldsymbol{x}_i}, \sum_{j=1}^{n} \alpha_j \kappa_{\boldsymbol{x}_j}\rangle \\ &= \textstyle\| \sum_{j=1}^{n} \alpha_j \kappa_{\boldsymbol{x}_j} \|^2 \geqslant 0.\end{aligned}$$

2. Show that a reproducing kernel, if it exists, is unique.

   Solution:

3. Let $\mathcal{G}$ be a Hilbert space of functions $g : \mathcal{X} \to \mathbb{R}$. Given $\boldsymbol{x} \in \mathcal{X}$, the *evaluation functional* is the map $\delta_{\boldsymbol{x}} : g \mapsto g(\boldsymbol{x})$. Show that evaluation functionals are linear operators.

   Solution: For any $a, b \in \mathbb{R}$ and $f, g \in \mathcal{G}$ we have

$$\delta_{\boldsymbol{x}}(af + bg) = (af + bg)(\boldsymbol{x}) = af(\boldsymbol{x}) + bg(\boldsymbol{x}) = a\delta_{\boldsymbol{x}} f + b\delta_{\boldsymbol{x}} g\,.$$

4. Let $\mathcal{G}_0$ be the pre-RKHS $\mathcal{G}_0$ constructed in the proof of Theorem 6.2. Thus, $g \in \mathcal{G}_0$ is of the form $g = \sum_{i=1}^{n} \alpha_i \kappa_{\boldsymbol{x}_i}$ and

$$\langle g, \kappa_{\boldsymbol{x}} \rangle_{\mathcal{G}_0} = \sum_{i=1}^{n} \alpha_i \langle \kappa_{\boldsymbol{x}_i}, \kappa_{\boldsymbol{x}} \rangle_{\mathcal{G}_0} = \sum_{i=1}^{n} \alpha_i \kappa(\boldsymbol{x}_i, \boldsymbol{x}) = g(\boldsymbol{x})\,.$$

   Therefore, we may write the evaluation functional of $g \in \mathcal{G}_0$ at $\boldsymbol{x}$ as $\delta_{\boldsymbol{x}} g := \langle g, \kappa_{\boldsymbol{x}} \rangle_{\mathcal{G}_0}$. Show that $\delta_{\boldsymbol{x}}$ is bounded on $\mathcal{G}_0$ for every $\boldsymbol{x}$; that is, $|\delta_{\boldsymbol{x}} f| < \gamma \|f\|_{\mathcal{G}_0}$, for some $\gamma < \infty$.

   Solution:

5. Continuing Exercise 4, let (f_n) be a Cauchy sequence in $\mathcal{G}_0$ such that $|f_n(\boldsymbol{x})| \to 0$ for all $\boldsymbol{x}$. Show that $\|f_n\|_{\mathcal{G}_0} \to 0$.

   Solution: As (f_n) is a Cauchy sequence on $\mathcal{G}_0$, it is of necessity bounded in the norm on $\mathcal{G}_0$; that is, $\|h_n\|_{\mathcal{G}_0} \leqslant B$, for some $B < \infty$. Moreover, by the definition of Cauchy sequence, for every $\varepsilon > 0$ there is an N such that for all $m, n \geqslant N$, $\|f_n - f_m\|_{\mathcal{G}_0} < \frac{\varepsilon}{2B}$. As $f_N \in \mathcal{G}_0$, it is of the form $f_N = \sum_{t=1}^{T} \alpha_t \kappa_{\boldsymbol{x}_t}$. Since f_n converges pointwise to 0, we can find an M such that for all $n > M$, $f_n(\boldsymbol{x}_t) < \varepsilon/(2T|\alpha_t|)$ for all $t \in \{1, \dots, T\}$.

   Consequently, for all $n > \max(N, M)$, we have:

$$
\begin{aligned}
\|f_n\|^2_{\mathcal{G}_0} &= |\langle f_n - f_N + f_N, f_n\rangle_{\mathcal{G}_0}| \\
&\leqslant |\langle f_n - f_N, f_n\rangle_{\mathcal{G}_0}| + |\langle f_N, f_n\rangle_{\mathcal{G}_0}| \\
&\leqslant \|f_n\|_{\mathcal{G}_0}\|f_n - f_N\|_{\mathcal{G}_0} + \left|\sum_{t=1}^{T} \alpha_t \langle \kappa_{\boldsymbol{x}_t}, f_n\rangle_{\mathcal{G}_0}\right| \\
&\leqslant \|f_n\|_{\mathcal{G}_0}\|f_n - f_N\|_{\mathcal{G}_0} + \sum_{t=1}^{T} |\alpha_t|\, |f_n(\boldsymbol{x}_t)| \\
&< B\frac{\varepsilon}{2B} + (\frac{\varepsilon|\alpha_1|}{2T|\alpha_1|} + \cdots + \frac{\varepsilon|\alpha_n|}{2T|\alpha_n|}) = \varepsilon.
\end{aligned}
$$

   Note that above we have made use of the Cauchy–Schwarz inequality and the reproducing property.

6. Continuing Exercises 5 and 4, to show that the inner product (6.14) is well defined, a number of facts have to be checked.

   (a) Verify that the limit converges.

   (b) Verify that the limit is independent of the Cauchy sequences used.

   (c) Verify that the properties of an inner product are satisfied. The only non-trivial property to verify is that $\langle f, f\rangle_{\mathcal{G}} = 0$ if and only if $f = 0$.

   Solution:

7. Exercises 4–6 show that $\mathcal{G}$ defined in the proof of Theorem 6.2 is an inner product space. It remains to prove that $\mathcal{G}$ is a RKHS. This requires us to prove that the inner product space $\mathcal{G}$ is complete (and thus Hilbert), and that its evaluation functionals are bounded and hence continuous (see Theorem A.16). This is done in a number of steps.

   (a) Show that $\mathcal{G}_0$ is dense in $\mathcal{G}$ in the sense that every $f \in \mathcal{G}$ is a limit point (with respect to the norm on $\mathcal{G}$) of a Cauchy sequence (f_n) in $\mathcal{G}_0$.

   Solution: By definition of $\mathcal{G}$, any f is the pointwise limit of a Cauchy sequence $\{f_n\} \in \mathcal{G}_0$. We need to show that such a Cauchy sequence converges to f in $\|\cdot\|_{\mathcal{G}}$. Note that we already have proved convergence in $\|\cdot\|_{\mathcal{G}_0}$ (Problem 5). Hence, for every $\varepsilon > 0$ there exist an N such that for all $n > N$, $\|f - f_n\|_{\mathcal{G}_0} < \varepsilon$. As, for fixed n, $\|f - f_n\|_{\mathcal{G}} = \lim_{m\to\infty} \|f - f_n\|_{\mathcal{G}_0}$ (by definition), we have

$$\|f - f_n\|_{\mathcal{G}} = \lim_{m\to\infty} \|f_m - f_n\|_{\mathcal{G}_0} < \varepsilon\,,$$

showing convergence with respect to $\|\cdot\|_{\mathcal{G}}$.

(b) Show that every evaluation functional $\delta_{\boldsymbol{x}}$ on $\mathcal{G}$ is continuous at the 0 function. That is,

$$\forall \varepsilon > 0 : \exists \delta > 0 : \forall f \in \mathcal{G} : \|f\|_{\mathcal{G}} < \delta \Rightarrow |f(\boldsymbol{x})| < \varepsilon. \tag{6.40}$$

Continuity at the whole space then follows automatically from linearity.

Solution: (Consider some $f \in \mathcal{G}$ and some Cauchy sequence (f_n) in $\mathcal{G}_0$ converging pointwise to f. Let $\varepsilon > 0$. We combine three continuity/convergence results. First, by pointwise convergence, there exist an N_1 such that for all $n \geqslant N_1$, $|f_n(\boldsymbol{x}) - f(\boldsymbol{x})| < \varepsilon/2$. Second, by continuity of evaluation functionals on $\mathcal{G}_0$ at 0, there exists a $\delta > 0$ such that $\|g\|_{\mathcal{G}_0} < \delta \Rightarrow |g(\boldsymbol{x})| < \varepsilon/2$. Third, as a result of (a), we can choose an $N \geqslant N_1$ such that $\|f_n - f\|_{\mathcal{G}} < \delta/2$ for all $n \geqslant N$.
In particular, we have

$$\|f_N\|_{\mathcal{G}_0} < \delta \implies |f_N(\boldsymbol{x})| < \varepsilon/2\,.$$

Note that f_N is part of the Cauchy sequence converging to f, so $\|f_N\|_{\mathcal{G}_0} = \|f_N\|_{\mathcal{G}}$. By taking $\|f\|_{\mathcal{G}} < \delta/2$ we have $\|f_N\|_{\mathcal{G}} \leqslant \|f\|_{\mathcal{G}} + \|f - f_N\|_{\mathcal{G}} < \delta/2 + \delta/2 = \delta$, which implies $|f_N(\boldsymbol{x})| < \varepsilon/2$. Consequently,

$$|f(\boldsymbol{x})| \leqslant |f(\boldsymbol{x}) - f_N(\boldsymbol{x})| + |f_N(\boldsymbol{x})| < \varepsilon/2 + \varepsilon/2 = \varepsilon\,,$$

meaning that $\|f\|_{\mathcal{G}} < \delta/2 \implies |\delta_{\boldsymbol{x}}(f)| = |f(\boldsymbol{x})| < \varepsilon$. Hence the evaluation functionals are continuous on $\mathcal{G}$ at 0.

(c) Show that $\mathcal{G}$ is complete; that is, every Cauchy sequence $(f_n) \in \mathcal{G}$ converges in the norm $\|\cdot\|_{\mathcal{G}}$.

Solution: (Take some Cauchy sequence (f_n) in $\mathcal{G}$. By (b), we have that for any $\boldsymbol{x} \in \mathscr{X}$, $\|f_n - f_m\|_{\mathcal{G}} < \delta \Rightarrow |f_n(\boldsymbol{x}) - f_m(\boldsymbol{x})| < \varepsilon$. So $(f_n(\boldsymbol{x}))$ is a Cauchy sequence in $\mathbb{R}$. By the completeness of $\mathbb{R}$, there exists some $f(\boldsymbol{x}) = \lim_{n\to\infty} f_n(\boldsymbol{x}) \in \mathbb{R}$ for all $\boldsymbol{x} \in \mathscr{X}$. By the denseness result (a) there exists, for each n, a sequence $(g_{n,j}, j = 1, 2, \ldots) \in \mathcal{G}_0$ such that for $j \geqslant N_n$ it holds that $\|f_n - g_{n,j}\|_{\mathcal{G}} < 1/n$. If we define $g_n = g_{n,N_n}$, then g_n converges pointwise to f, since

$$|g_n(\boldsymbol{x}) - f(\boldsymbol{x})| \leqslant |g_n(\boldsymbol{x}) - f_n(\boldsymbol{x})| + |f_n(\boldsymbol{x}) - f(\boldsymbol{x})|,$$

and both terms on the right hand side converge to 0. Next, note that

$$\|g_m - g_n\|_{\mathcal{G}_0} = \|g_m - g_n\|_{\mathcal{G}} \leqslant \|g_m - f_m\|_{\mathcal{G}} + \|f_m - f_n\|_{\mathcal{G}} + \|f_n - g_n\|_{\mathcal{G}},$$

where all three right-hand terms converge to 0 as $m, n \to \infty$. Hence, (g_n) is Cauchy in $\mathcal{G}_0$. We noted in the proof of (a) that Cauchy sequences in $\mathcal{G}_0$ with pointwise limits in $\mathcal{G}$ also converge to those limits in $\|\cdot\|_{\mathcal{G}}$, thus $\|g_n - f\|_{\mathcal{G}} \to 0$. Since $\|f_n - g_n\|_{\mathcal{G}} \to 0$, it follows that $\|f_n - f\|_{\mathcal{G}} \to 0$, and so $\mathcal{G}$ is complete.

8. If κ_1 and κ_2 are reproducing kernels on $\mathcal{X}$ and $\mathcal{Y}$, then $\kappa_+((\boldsymbol{x},\boldsymbol{y}),(\boldsymbol{x}',\boldsymbol{y}')) := \kappa_1(\boldsymbol{x},\boldsymbol{x}')+\kappa_2(\boldsymbol{y},\boldsymbol{y}')$ and $\kappa_\times((\boldsymbol{x},\boldsymbol{y}),(\boldsymbol{x}',\boldsymbol{y}')) := \kappa_1(\boldsymbol{x},\boldsymbol{x}')\kappa_2(\boldsymbol{y},\boldsymbol{y}')$ are reproducing kernels on the Cartesian product $\mathcal{X}\times\mathcal{Y}$. Prove this.

   Solution:

9. A RKHS enjoys the following desirable smoothness property: if (g_n) is a sequence belonging to RKHS $\mathcal{G}$ on $\mathcal{X}$, and $\|g_n - g\|_{\mathcal{G}} \to 0$, then $g(\boldsymbol{x}) = \lim_n g_n(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathcal{X}$. Prove this, using Cauchy–Schwarz.

   Solution: By the Cauchy–Schwarz inequality,

$$|g_n(\boldsymbol{x}) - g(\boldsymbol{x})| = |\langle g_n - g, \kappa_{\boldsymbol{x}}\rangle_{\mathcal{G}}| \leqslant \|g_n - g\|_{\mathcal{G}}\|\kappa_{\boldsymbol{x}}\|_{\mathcal{G}} \downarrow 0\,.$$

10. Let $\boldsymbol{X}$ be an $\mathbb{R}^d$-valued random variable that is symmetric about the origin (that is, $\boldsymbol{X}$ and $(-\boldsymbol{X})$ are identically distributed). Denote by μ is its distribution and $\psi(\boldsymbol{t}) = \mathbb{E}\mathrm{e}^{\mathrm{i}\boldsymbol{t}^\top \boldsymbol{X}} = \int \mathrm{e}^{\mathrm{i}\boldsymbol{t}^\top \boldsymbol{x}}\,\mu(\mathrm{d}\boldsymbol{x})$ for $\boldsymbol{t} \in \mathbb{R}^d$ is its characteristic function. Verify that $\kappa(\boldsymbol{x},\boldsymbol{x}') = \psi(\boldsymbol{x}-\boldsymbol{x}')$ is a real-valued positive semi-definite function.

    Solution:

11. Suppose RKHS $\mathcal{G}$ of functions from $\mathcal{X} \to \mathbb{R}$ (with kernel κ) is invariant under a group $\mathcal{T}$ of transformations $T : \mathcal{X} \to \mathcal{X}$; that is, for all $f, g \in \mathcal{G}$ and $T \in \mathcal{T}$, we have (i) $f \circ T \in \mathcal{G}$ and (ii) $\langle f \circ T, g \circ T\rangle_{\mathcal{G}} = \langle f, g\rangle_{\mathcal{G}}$. Show that $\kappa(T\boldsymbol{x}, T\boldsymbol{x}') = \kappa(\boldsymbol{x},\boldsymbol{x}')$ for all $\boldsymbol{x},\boldsymbol{x}' \in \mathcal{X}$ and $T \in \mathcal{T}$.

    Solution: For any $g \in \mathcal{G}$ and $T \in \mathcal{T}$ we have $g(\boldsymbol{x}) = \langle g, \kappa_{\boldsymbol{x}}\rangle_{\mathcal{G}}$ by definition. We also have $g(\boldsymbol{x}) = g \circ T^{-1}(T\boldsymbol{x}) = \langle g \circ T^{-1}, \kappa_{T\boldsymbol{x}}\rangle_{\mathcal{G}} = \langle g, \kappa_{T\boldsymbol{x}} \circ T\rangle_{\mathcal{G}}$ where the last equality follows from (ii). Hence, by uniqueness of the kernel, we must have $\kappa_{\boldsymbol{x}} = \kappa_{T\boldsymbol{x}} \circ T$ for all $\boldsymbol{x} \in \mathcal{X}$, and for all $\boldsymbol{x}' \in \mathcal{X}$ it follows that $\kappa(\boldsymbol{x},\boldsymbol{x}') = \kappa_{\boldsymbol{x}}(\boldsymbol{x}') = \kappa_{T\boldsymbol{x}}(T\boldsymbol{x}') = \kappa(T\boldsymbol{x}, T\boldsymbol{x}')$.

12. Given two Hilbert spaces $\mathcal{H}$ and $\mathcal{G}$, we call a mapping $A : \mathcal{H} \to \mathcal{G}$ a *Hilbert space isomorphism* if it is

    (i) a linear map; that is, $A(af + bg) = aA(f) + bA(g)$ for any $f, g \in \mathcal{H}$ and $a, b \in \mathbb{R}$.

    (ii) a surjective map; and

    (iii) an isometry; that is for all $f, g \in \mathcal{H}$, it holds that $\langle f, g\rangle_{\mathcal{H}} = \langle Af, Ag\rangle_{\mathcal{G}}$.

    Let $\mathcal{H} = \mathbb{R}^p$ (equipped with the usual Euclidean inner product) and construct its (continuous) *dual space* $\mathcal{G}$, consisting of all continuous linear functions from $\mathbb{R}^p$ to $\mathbb{R}$, as follows: (a) For each $\boldsymbol{\beta} \in \mathbb{R}^p$, define $g_{\boldsymbol{\beta}} : \mathbb{R}^p \to \mathbb{R}$ via $g_{\boldsymbol{\beta}}(\boldsymbol{x}) = \langle \boldsymbol{\beta}, \boldsymbol{x}\rangle = \boldsymbol{\beta}^\top \boldsymbol{x}$, for all $\boldsymbol{x} \in \mathbb{R}^p$. (b) Equip $\mathcal{G}$ with the inner product $\langle g_{\boldsymbol{\beta}}, g_{\boldsymbol{\gamma}}\rangle_{\mathcal{G}} := \boldsymbol{\beta}^\top\boldsymbol{\gamma}$.

    Show that $A : \mathcal{H} \to \mathcal{G}$ defined by $A(\boldsymbol{\beta}) = g_{\boldsymbol{\beta}}$ for $\boldsymbol{\beta} \in \mathbb{R}^p$ is a Hilbert space isomorphism.

    Solution:

13. Let $\mathbf{X}$ be an $n \times p$ model matrix. Show that $\mathbf{X}^\top\mathbf{X} + n\gamma\mathbf{I}_p$ for $\gamma > 0$ is invertible.

    Solution: The matrix $\mathbf{X}^\top\mathbf{X}$ is not only symmetric but also positive semi-definite, and as such all of its eigenvalues $\lambda_1, \dots, \lambda_p$ are real and non-negative. Thus $\mathbf{X}^\top\mathbf{X} - \lambda\mathbf{I}_p$ is only singular for non-negative λ. Setting $\lambda = -n\gamma < 0$ which cannot be an eigenvalue of $\mathbf{X}^\top\mathbf{X}$, and so the matrix $\mathbf{X}^\top\mathbf{X} + n\gamma\mathbf{I}_p$ is invertible.

    A similar argument applies to show invertibility of $\mathbf{X}\mathbf{X}^\top + n\gamma\mathbf{I}_n$. Moreover, Sylvester's determinant identity tells us that in fact $\det(\mathbf{X}^\top\mathbf{X} + n\gamma\mathbf{I}_p) = \det(\mathbf{X}\mathbf{X}^\top + n\gamma\mathbf{I}_n)$ which we have just shown is non-zero whenever $\gamma > 0$.

14. As Example 6.8 clearly illustrates, the pdf of a random variable that is symmetric about the origin is not in general a valid reproducing kernel. Take two iid random variables X and X' with common pdf f, and define $Z = X + X'$. Denote by ψ_Z and f_Z the characteristic function and pdf of Z, respectively.

    Show that if ψ_Z is in $L^1(\mathbb{R})$, f_Z is a positive semidefinite function. Use this to show that $\kappa(x, x') = f_Z(x - x') = \mathbb{1}\{|x - x'| \leqslant 2\}(1 - |x - x'|/2)$ is a valid reproducing kernel.

    Solution:

15. For the smoothing cubic spline of Section 6.6, show that $\kappa(x,u) = \frac{\max\{x,u\}\min\{x,u\}^2}{2} - \frac{\min\{x,u\}^3}{6}$.

    Solution: Noting that $\kappa(x,u) = \langle \kappa_x, \kappa_u \rangle_{\mathcal{H}}$, we have

$$
\begin{aligned}
\kappa(x,u) &= \int_0^1 \frac{\partial^2 \kappa(x,s)}{\partial s^2}\frac{\partial^2 \kappa(u,s)}{\partial s^2}\,\mathrm{d}s \\
&= \int_0^1 (x-s)_+(u-s)_+\,\mathrm{d}s \\
&= \int_0^{\min\{x,u\}} (x-s)_+(u-s)_+\,\mathrm{d}s + \int_{\min\{x,u\}}^{\max\{x,u\}} (x-s)_+(u-s)_+\,\mathrm{d}s + \int_{\max\{x,u\}}^1 (x-s)_+(u-s)_+\,\mathrm{d}s \\
&= \int_0^{\min\{x,u\}} (\max\{x,u\} - s)(\min\{x,u\} - s)\,\mathrm{d}s + 0 + 0 \\
&= \frac{\max\{x,u\}\min\{x,u\}^2}{2} - \frac{\min\{x,u\}^3}{6}.
\end{aligned}
$$

16. Let $\mathbf{X}$ be an $n \times p$ model matrix and let $\boldsymbol{u} \in \mathbb{R}^p$ be the unit-length vector with k-th entry equal to one ($u_k = \|\boldsymbol{u}\| = 1$). Suppose that the k-th column of $\mathbf{X}$ is $\boldsymbol{v}$ and that it is replaced with a new predictor $\boldsymbol{w}$, so that we obtain the new model matrix:

$$\widetilde{\mathbf{X}} = \mathbf{X} + (\boldsymbol{w} - \boldsymbol{v})\boldsymbol{u}^\top.$$

    (a) Denoting

$$\boldsymbol{\delta} := \mathbf{X}^\top(\boldsymbol{w} - \boldsymbol{v}) + \frac{\|\boldsymbol{w} - \boldsymbol{v}\|^2}{2}\boldsymbol{u},$$

    show that

$$\widetilde{\mathbf{X}}^\top\widetilde{\mathbf{X}} = \mathbf{X}^\top\mathbf{X} + \boldsymbol{u}\boldsymbol{\delta}^\top + \boldsymbol{\delta}\boldsymbol{u}^\top = \mathbf{X}^\top\mathbf{X} + \frac{(\boldsymbol{u}+\boldsymbol{\delta})(\boldsymbol{u}+\boldsymbol{\delta})^\top}{2} - \frac{(\boldsymbol{u}-\boldsymbol{\delta})(\boldsymbol{u}-\boldsymbol{\delta})^\top}{2}.$$

In other words, $\widetilde{\mathbf{X}}^\top\widetilde{\mathbf{X}}$ differs from $\mathbf{X}^\top\mathbf{X}$ by a symmetric matrix of rank two.

Solution:

(b) Suppose that $\mathbf{B} := (\mathbf{X}^\top\mathbf{X} + n\gamma\mathbf{I}_p)^{-1}$ is already computed. Explain how the Sherman–Morrison formulas in Theorem A.10 can be applied twice to compute the inverse and log-determinant of the matrix $\widetilde{\mathbf{X}}^\top\widetilde{\mathbf{X}} + n\gamma\mathbf{I}_p$ in $O((n+p)p)$ computing time, rather than the usual $O((n+p^2)p)$ computing time.[1]

Solution:

(c) Write a Python program for updating a matrix $\mathbf{B} = (\mathbf{X}^\top\mathbf{X} + n\gamma\mathbf{I}_p)^{-1}$ when we change the k-th column of $\mathbf{X}$, as shown in the following pseudo-code.

```
Algorithm 6.8.1: Updating via Sherman–Morrison Formula
  input: Matrices $\mathbf{X}$ and $\mathbf{B}$, index $k$, and replacement $\boldsymbol{w}$ for the $k$-th column of $\mathbf{X}$.
  output: Updated matrices $\mathbf{X}$ and $\mathbf{B}$.
1 Set $\boldsymbol{v} \in \mathbb{R}^n$ to be the $k$-th column of $\mathbf{X}$.
2 Set $\boldsymbol{u} \in \mathbb{R}^p$ to be the unit-length vector such that $u_k = \|\boldsymbol{u}\| = 1$.
3 $\mathbf{B} \leftarrow \mathbf{B} - \dfrac{\mathbf{B}\boldsymbol{u}\boldsymbol{\delta}^\top\mathbf{B}}{1+\boldsymbol{\delta}^\top\mathbf{B}\boldsymbol{u}}$
4 $\mathbf{B} \leftarrow \mathbf{B} - \dfrac{\mathbf{B}\boldsymbol{\delta}\boldsymbol{u}^\top\mathbf{B}}{1+\boldsymbol{u}^\top\mathbf{B}\boldsymbol{\delta}}$
5 Update the $k$-th column of $\mathbf{X}$ with $\boldsymbol{w}$.
6 return $\mathbf{X}, \mathbf{B}$
```

Solution:

17. Use Algorithm 6.8.1 from Exercise 16 to write Python code that computes the ridge regression coefficient $\boldsymbol{\beta}$ in (6.5) and use it to replicate the results on Figure 6.1. The following pseudo-code (with running cost of $O((n+p)p^2)$) may help with the writing of the Python code. ☞ 217

```
Algorithm 6.8.2: Ridge Regression Coefficients via Sherman–Morrison Formula
  input: Training set $\{\mathbf{X}, \boldsymbol{y}\}$ and regularization parameter $\gamma > 0$.
  output: Solution $\widehat{\boldsymbol{\beta}} = (n\gamma\mathbf{I}_p + \mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\boldsymbol{y}$.
1 Set $\mathbf{A}$ to be an $n \times p$ matrix of zeros and $\mathbf{B} \leftarrow \mathbf{I}_p/(n\gamma)$.
2 for $j = 1, \ldots, p$ do
3     Set $\boldsymbol{w}$ to be the $j$-th column of $\mathbf{X}$.
4     Update $\{\mathbf{A}, \mathbf{B}\}$ via Algorithm 6.8.1 with inputs $\{\mathbf{A}, \mathbf{B}, j, \boldsymbol{w}\}$.
5 $\widehat{\boldsymbol{\beta}} \leftarrow \mathbf{B}(\mathbf{X}^\top\boldsymbol{y})$
6 return $\widehat{\boldsymbol{\beta}}$
```

Solution:

[1]This Sherman–Morrison updating is not always numerically stable. A more numerically stable method will perform two consecutive rank-one updates of the Cholesky decomposition of $\mathbf{X}^\top\mathbf{X} + n\gamma\mathbf{I}_p$.

The following code implements the algorithm above using $\gamma = 1$ as a default value.

```
def ridge(X,y,gam=1):
    n,p=X.shape
    A=np.zeros([n,p])
    B=np.identity(p)/(n*gam)
    for j in range(1,p+1):
          w=X[:,j-1]; w=w.reshape(n,1)
          A,B=SherMorr(A,B,j,w)
          bhat=B@(X.T@y)
    return bhat
```

18. Consider Example 2.10 with $\mathbf{D} = \mathrm{diag}(\lambda_1, \dots, \lambda_p)$ for some nonnegative vector $\boldsymbol{\lambda} \in \mathbb{R}^p$, so that twice the negative logarithm of the *model evidence* can be written as

$$-2\ln g(\boldsymbol{y}) = l(\boldsymbol{\lambda}) := n\ln[\boldsymbol{y}^\top(\mathbf{I} - \mathbf{X}\boldsymbol{\Sigma}\mathbf{X}^\top)\boldsymbol{y}] + \ln|\mathbf{D}| - \ln|\boldsymbol{\Sigma}| + c,$$

where c is a constant that depends only on n.

(a) Use the *Woodbury identities* (A.15) and (A.16) to show that

$$\mathbf{I} - \mathbf{X}\boldsymbol{\Sigma}\mathbf{X}^\top = (\mathbf{I} + \mathbf{X}\mathbf{D}\mathbf{X}^\top)^{-1}$$
$$\ln|\mathbf{D}| - \ln|\boldsymbol{\Sigma}| = \ln|\mathbf{I} + \mathbf{X}\mathbf{D}\mathbf{X}^\top|.$$

Deduce that $l(\boldsymbol{\lambda}) = n\ln[\boldsymbol{y}^\top\mathbf{C}\boldsymbol{y}] - \ln|\mathbf{C}| + c$, where $\mathbf{C} := (\mathbf{I} + \mathbf{X}\mathbf{D}\mathbf{X}^\top)^{-1}$.

Solution:

(b) Let $[\boldsymbol{v}_1, \dots, \boldsymbol{v}_p] := \mathbf{X}$ denote the p columns/predictors of $\mathbf{X}$. Show that

$$\mathbf{C}^{-1} = \mathbf{I} + \sum_{k=1}^{p} \lambda_k \boldsymbol{v}_k \boldsymbol{v}_k^\top.$$

Explain why setting $\lambda_k = 0$ has the effect of excluding the k-th predictor from the regression model. How can this observation be used for model selection?

Solution:

(c) Prove the following formulas for the gradient and Hessian elements of $l(\boldsymbol{\lambda})$:

$$\begin{aligned}\frac{\partial l}{\partial \lambda_i} &= \boldsymbol{v}_i^\top\mathbf{C}\boldsymbol{v}_i - n\frac{(\boldsymbol{v}_i^\top\mathbf{C}\boldsymbol{y})^2}{\boldsymbol{y}^\top\mathbf{C}\boldsymbol{y}} \\ \frac{\partial^2 l}{\partial\lambda_i\,\partial\lambda_j} &= (n-1)(\boldsymbol{v}_i^\top\mathbf{C}\boldsymbol{v}_j)^2 - n\left[\boldsymbol{v}_i^\top\mathbf{C}\boldsymbol{v}_j - \frac{(\boldsymbol{v}_i^\top\mathbf{C}\boldsymbol{y})(\boldsymbol{v}_j^\top\mathbf{C}\boldsymbol{y})}{\boldsymbol{y}^\top\mathbf{C}\boldsymbol{y}}\right]^2.\end{aligned} \tag{6.1}$$

Solution:

(d) One method to determine which predictors in $\mathbf{X}$ are important is to compute

$$\boldsymbol{\lambda}^* := \underset{\boldsymbol{\lambda}\geqslant\mathbf{0}}{\mathrm{argmin}}\, l(\boldsymbol{\lambda})$$

using, for example, the interior-point minimization Algorithm B.4.1 with gradient and Hessian computed from (6.41). Write Python code to compute $\boldsymbol{\lambda}^*$ and use it to select the best polynomial model in Example 2.10.

Solution:

19. (Exercise 18 continued.) Consider again Example 2.10 with $\mathbf{D} = \mathrm{diag}(\lambda_1, \dots, \lambda_p)$ for some nonnegative model-selection parameter $\boldsymbol{\lambda} \in \mathbb{R}^p$. A Bayesian choice for $\boldsymbol{\lambda}$ is the maximizer of the marginal likelihood $g(\boldsymbol{y} \,|\, \boldsymbol{\lambda})$; that is,

$$\boldsymbol{\lambda}^* = \operatorname*{argmax}_{\boldsymbol{\lambda} \geqslant \mathbf{0}} \iint g(\boldsymbol{\beta}, \sigma^2, \boldsymbol{y} \,|\, \boldsymbol{\lambda}) \, \mathrm{d}\boldsymbol{\beta} \, \mathrm{d}\sigma^2,$$

where

$$\ln g(\boldsymbol{\beta}, \sigma^2, \boldsymbol{y} \,|\, \boldsymbol{\lambda}) = -\frac{\|\boldsymbol{y} - \mathbf{X}\boldsymbol{\beta}\|^2 + \boldsymbol{\beta}^\top \mathbf{D}^{-1} \boldsymbol{\beta}}{2\sigma^2} - \frac{1}{2} \ln |\mathbf{D}| - \frac{n+p}{2} \ln(2\pi\sigma^2) - \ln \sigma^2.$$

To maximize $g(\boldsymbol{y} \,|\, \boldsymbol{\lambda})$, one can use the *EM algorithm* with $\boldsymbol{\beta}$ and σ^2 acting as *latent variables* in the *complete-data log-likelihood* $\ln g(\boldsymbol{\beta}, \sigma^2, \boldsymbol{y} \,|\, \boldsymbol{\lambda})$. Define

$$\begin{aligned} \boldsymbol{\Sigma} &:= (\mathbf{D}^{-1} + \mathbf{X}^\top \mathbf{X})^{-1} \\ \overline{\boldsymbol{\beta}} &:= \boldsymbol{\Sigma} \mathbf{X}^\top \boldsymbol{y} \\ \widehat{\sigma}^2 &:= \left(\|\boldsymbol{y}\|^2 - \boldsymbol{y}^\top \mathbf{X} \overline{\boldsymbol{\beta}} \right) / n. \end{aligned} \tag{6.2}$$

(a) Show that the conditional density of the latent variables $\boldsymbol{\beta}$ and σ^2 is such that

$$\begin{aligned} \left(\sigma^{-2} \,\middle|\, \boldsymbol{\lambda}, \boldsymbol{y}\right) &\sim \mathsf{Gamma}\left(\frac{n}{2}, \frac{n}{2} \widehat{\sigma}^2\right) \\ \left(\boldsymbol{\beta} \,\middle|\, \boldsymbol{\lambda}, \sigma^2, \boldsymbol{y}\right) &\sim \mathcal{N}\left(\overline{\boldsymbol{\beta}}, \sigma^2 \boldsymbol{\Sigma}\right). \end{aligned}$$

Solution: As in Example 2.10, we can write

$$g(\boldsymbol{\beta}, \sigma^2, \boldsymbol{y} \,|\, \boldsymbol{\lambda}) = \frac{(\sigma^2)^{-1}}{(2\pi\sigma^2)^{(n+p)/2} |\mathbf{D}|^{1/2}} \exp\left(-\frac{\|\boldsymbol{\Sigma}^{-1/2}(\boldsymbol{\beta} - \overline{\boldsymbol{\beta}})\|^2}{2\sigma^2} - \frac{n\widehat{\sigma}^2}{2\sigma^2} \right).$$

Hence,

$$\begin{aligned} g(\sigma^2 \,|\, \boldsymbol{\lambda}, \boldsymbol{y}) &\propto \int g(\boldsymbol{\beta}, \sigma^2, \boldsymbol{y} \,|\, \boldsymbol{\lambda}) \, \mathrm{d}\boldsymbol{\beta} \\ &\propto \frac{1}{(\sigma^2)^{1+n/2}} \exp\left(-\frac{n\widehat{\sigma}^2/2}{\sigma^2} \right), \end{aligned}$$

which we recognize as the density of σ^2, where $\sigma^{-2} \sim \mathsf{Gamma}\left(\frac{n}{2}, \frac{n}{2}\widehat{\sigma}^2\right)$; see Exercise 19. Similarly,

$$g(\boldsymbol{\beta} \,|\, \sigma^2, \boldsymbol{\lambda}, \boldsymbol{y}) \propto \exp\left(-\frac{\|\boldsymbol{\Sigma}^{-1/2}(\boldsymbol{\beta} - \overline{\boldsymbol{\beta}})\|^2}{2\sigma^2} \right),$$

which we recognize as the density of the $\mathcal{N}\left(\overline{\boldsymbol{\beta}}, \sigma^2 \boldsymbol{\Sigma}\right)$ distribution.

(b) Use Theorem C.2 to show that the expected complete-data log-likelihood is

$$-\frac{\overline{\boldsymbol{\beta}}^\top \mathbf{D}^{-1} \overline{\boldsymbol{\beta}}}{2\widehat{\sigma}^2} - \frac{\mathrm{tr}(\mathbf{D}^{-1}\boldsymbol{\Sigma}) + \ln |\mathbf{D}|}{2} + c_1,$$

where c_1 is a constant that does not depend on $\boldsymbol{\lambda}$.

Solution: Taking the expectation with respect to $\boldsymbol{\beta}$, given $\boldsymbol{y}, \boldsymbol{\lambda}, \sigma^2$, we obtain

$$-\frac{\|\boldsymbol{y}-\mathbf{X}\overline{\boldsymbol{\beta}}\|^2+\overline{\boldsymbol{\beta}}^\top\mathbf{D}^{-1}\overline{\boldsymbol{\beta}}+\sigma^2\mathrm{tr}(\boldsymbol{\Sigma}(\mathbf{D}^{-1}+\mathbf{X}^\top\mathbf{X}))}{2\sigma^2}-\frac{1}{2}\ln|\mathbf{D}|+\text{const.}$$

Since $\mathbb{E}[1/\sigma^2\,|\,\boldsymbol{\lambda},\boldsymbol{y}]=1/\widehat{\sigma}^2$, taking an expectation with respect to $1/\sigma^2$, given $\boldsymbol{\lambda},\boldsymbol{y}$, we obtain

$$-\frac{\overline{\boldsymbol{\beta}}^\top\mathbf{D}^{-1}\overline{\boldsymbol{\beta}}}{2\widehat{\sigma}^2}-\frac{\mathrm{tr}(\boldsymbol{\Sigma}\mathbf{D}^{-1})+\ln|\mathbf{D}|}{2}+\text{const.}$$

(c) Use Theorem A.2 to simplify the expected complete-data log-likelihood and to show that it is maximized at $\lambda_i=\boldsymbol{\Sigma}_{ii}+(\overline{\beta}_i/\widehat{\sigma})^2$ for $i=1,\dots,p$. Hence, deduce the following E and M steps in the EM algorithm:

E-step. Given $\boldsymbol{\lambda}$, update $(\boldsymbol{\Sigma},\overline{\boldsymbol{\beta}},\widehat{\sigma}^2)$ via the formulas (6.42).

M-step. Given $(\boldsymbol{\Sigma},\overline{\boldsymbol{\beta}},\widehat{\sigma}^2)$, update $\boldsymbol{\lambda}$ via $\lambda_i=\boldsymbol{\Sigma}_{ii}+(\overline{\beta}_i/\widehat{\sigma})^2,\ i=1,\dots,p$.

Solution: Since $\ln|\mathbf{D}|=\sum_i\ln\lambda_i$ and $\overline{\boldsymbol{\beta}}^\top\mathbf{D}^{-1}\overline{\boldsymbol{\beta}}=\mathrm{tr}(\mathbf{D}^{-1}\overline{\boldsymbol{\beta}}\,\overline{\boldsymbol{\beta}}^\top)=\sum_i\overline{\beta}_i^2/\lambda_i$, and $\mathrm{tr}(\boldsymbol{\Sigma}\mathbf{D}^{-1})=\sum_i\boldsymbol{\Sigma}_{ii}/\lambda_i$, we need to minimize:

$$\sum_i\frac{(\overline{\beta}_i/\widehat{\sigma})^2+\boldsymbol{\Sigma}_{ii}}{\lambda_i}+\sum_i\ln\lambda_i.$$

Differentiating with respect to λ_i and setting the derivative to zero yields

$$-\frac{(\overline{\beta}_i/\widehat{\sigma})^2+\boldsymbol{\Sigma}_{ii}}{\lambda_i^2}+\frac{1}{\lambda_i}=0.$$

Solving the equation, yields $\lambda_i=(\overline{\beta}_i/\widehat{\sigma})^2+\boldsymbol{\Sigma}_{ii}$.
As with the solution of 18 d), we can use the MM algorithm

(d) Write Python code to compute $\boldsymbol{\lambda}^*$ via the EM algorithm, and use it to select the best polynomial model in Example 2.10. A possible stopping criterion is to terminate the EM iterations when

$$\ln g(\boldsymbol{y}\,|\,\boldsymbol{\lambda}_{t+1})-\ln g(\boldsymbol{y}\,|\,\boldsymbol{\lambda}_t)<\varepsilon$$

for some small $\varepsilon>0$, where the marginal log-likelihood is

$$\ln g(\boldsymbol{y}\,|\,\boldsymbol{\lambda})=-\frac{n}{2}\ln(n\pi\widehat{\sigma}^2)-\frac{1}{2}\ln|\mathbf{D}|+\frac{1}{2}\ln|\boldsymbol{\Sigma}|+\ln\Gamma(n/2).$$

Solution: The following function implements the EM algorithm and runs it on the data used in Exercise 18.

```
import numpy as np; from numpy.linalg import solve
from numpy.linalg import inv; from numpy import sqrt
from numpy import random
```

```
n=1000;p=4
X = random.randn(n*p).reshape(n,p)
beta=np.array([1,0,-4,0]).reshape(p,1)
y=X@beta+random.randn(n).reshape(n,1)

def EMalg(X,y):
    n,p=X.shape; XX=X.T@X # precompute
    lam=random.rand(p) # initial lambdas
    err=1
    while err >10**-6:
     Dinv=np.diag(1/lam)
     S=inv(Dinv+XX)
     b=S@X.T@y
     s2=np.sum((y-X@b)**2)/n
     b=b.reshape(-1)
     lam_new=np.diag(S)+b**2/s2
     err=sqrt(np.sum((lam-lam_new)**2))
     lam=lam_new
    return lam,b

lam,b=EMalg(X,y)
```

Note that the stopping criterion here is $\|\boldsymbol{\lambda}_{t+1} - \boldsymbol{\lambda}_t\| < 10^{-3}$, which is an alternative to the one suggested above. Overall, the EM algorithm does not perform as well as the algorithm in Exercise **18**, because it frequently fails to force any of the λ_i's to be (close to) zero.

20. In this exercise we explore how the *early stopping* of the *gradient descent* iterations (see Example **B.10**),

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \alpha \nabla f(\boldsymbol{x}_t), \quad t = 0, 1, \ldots,$$

is (approximately) equivalent to the global minimization of $f(\boldsymbol{x}) + \frac{1}{2}\gamma\|\boldsymbol{x}\|^2$ for certain values of the *ridge regularization* parameter $\gamma > 0$ (see Example **6.1**). We illustrate the *early stopping* idea on the quadratic function $f(\boldsymbol{x}) = \frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^\top \mathbf{H}(\boldsymbol{x} - \boldsymbol{\mu})$, where $\mathbf{H} \in \mathbb{R}^{n\times n}$ is a symmetric positive-definite (Hessian) matrix with eigenvalues $\{\lambda_k\}_{k=1}^n$.

(a) Verify that for a symmetric matrix $\mathbf{A} \in \mathbb{R}^n$ such that $\mathbf{I} - \mathbf{A}$ is invertible, we have

$$\mathbf{I} + \mathbf{A} + \cdots + \mathbf{A}^{t-1} = (\mathbf{I} - \mathbf{A}^t)(\mathbf{I} - \mathbf{A})^{-1}.$$

Solution:

(b) Let $\mathbf{H} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^\top$ be the diagonalization of $\mathbf{H}$ as per Theorem **A.8**. If $\boldsymbol{x}_0 = \mathbf{0}$, show that the formula for $\boldsymbol{x}_t$ is

$$\boldsymbol{x}_t = \boldsymbol{\mu} - \mathbf{Q}(\mathbf{I} - \alpha\boldsymbol{\Lambda})^t\mathbf{Q}^\top\boldsymbol{\mu}.$$

Hence, deduce that a necessary condition for $\boldsymbol{x}_t$ to converge is $\alpha < 2/\max_k \lambda_k$.

Solution:

(c) Show that the minimizer of $f(\boldsymbol{x}) + \frac{1}{2}\gamma\|\boldsymbol{x}\|^2$ can be written as

$$\boldsymbol{x}^* = \boldsymbol{\mu} - \mathbf{Q}(\mathbf{I} + \gamma^{-1}\boldsymbol{\Lambda})^{-1}\mathbf{Q}^\top\boldsymbol{\mu}.$$

Solution:

(d) For a fixed value of t, let the learning rate $\alpha \downarrow 0$. Using part (b) and (c), show that if $\gamma \simeq 1/(t\,\alpha)$ as $\alpha \downarrow 0$, then $\boldsymbol{x}_t \simeq \boldsymbol{x}^*$. In other words, $\boldsymbol{x}_t$ is approximately equal to $\boldsymbol{x}^*$ for small α, provided that γ is inversely proportional to $t\,\alpha$.

Solution:

CHAPTER 7

CLASSIFICATION

Exercises

1. Let $0 \leqslant w \leqslant 1$. Show that the solution to the convex optimization problem

$$\begin{aligned} &\min_{p_1,\dots,p_n} \sum_{i=1}^n p_i^2 \\ &\text{subject to: } \sum_{i-1}^{n-1} p_i = w \text{ and } \sum_{i=1}^n p_i = 1, \end{aligned} \tag{7.28}$$

is given by $p_i = w/(n-1), i = 1, \dots, n-1$ and $p_n = 1 - w$.

Solution: The Lagrangian function is

$$\mathcal{L}(p_1, \dots, p_n, \lambda, \mu) = \sum_{i=1}^n p_i^2 + \lambda \left(\sum_{i=1}^{n-1} p_i - w \right) + \mu \left(\sum_{i=1}^n p_i - 1 \right).$$

Differentiation with respect to p_i gives $2p_i + \lambda + \mu = 0$, for $i = 1, \dots, n-1$. Thus $p_1 = \dots = p_{n-1} = w/(c-1)$, and $p_n = 1 - w$.

2. Derive the formulas (7.14) by minimizing the cross-entropy training loss:

$$-\frac{1}{n} \sum_{i=1}^n \ln g(\boldsymbol{x}_i, y_i \,|\, \boldsymbol{\theta}),$$

where $g(\boldsymbol{x}, y \,|\, \boldsymbol{\theta})$ is such that:

$$\ln g(\boldsymbol{x}, y \,|\, \boldsymbol{\theta}) = \ln \alpha_y - \frac{1}{2} \ln |\boldsymbol{\Sigma}_y| - \frac{1}{2} (\boldsymbol{x} - \boldsymbol{\mu}_y)^\top \boldsymbol{\Sigma}_y^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_y) - \frac{p}{2} \ln(2\pi).$$

Solution:

3. Adapt the code in Example 7.2 to plot the estimated decision boundary instead of the true one in Figure 7.3. Compare the results.

Solution: We found estimates

$$\widehat{\boldsymbol{\mu}_1} = \begin{bmatrix} 0.0628 \\ 0.0348 \end{bmatrix}, \quad \widehat{\boldsymbol{\mu}_2} = \begin{bmatrix} 2.1137 \\ 4.0610 \end{bmatrix}, \quad \text{and} \quad \widehat{\Sigma} = \begin{bmatrix} 2.0421 & 0.7410 \\ 0.7410 & 2.0111 \end{bmatrix},$$

and the estimated decision boundary is very close to the true one.

4. Recall from equation (7.16) that the decision boundaries of the multi-logit classifier are linear, and that the pre-classifier can be written as a conditional pdf of the form:

$$g(y \,|\, \mathbf{W}, \boldsymbol{b}, \boldsymbol{x}) = \frac{\exp(z_{y+1})}{\sum_{i=1}^{c} \exp(z_i)}, \quad y \in \{0, \dots, c-1\},$$

where $\boldsymbol{x}^\top = [1, \widetilde{\boldsymbol{x}}^\top]$ and $\boldsymbol{z} = \mathbf{W}\widetilde{\boldsymbol{x}} + \boldsymbol{b}$.

(a) Show that the linear discriminant pre-classifier in Section 7.4 can also be written as a conditional pdf of the form ($\boldsymbol{\theta} = \{\alpha_y, \boldsymbol{\Sigma}_y, \boldsymbol{\mu}_y\}_{y=0}^{c-1}$):

$$g(y \,|\, \boldsymbol{\theta}, \boldsymbol{x}) = \frac{\exp(z_{y+1})}{\sum_{i=1}^{c} \exp(z_i)}, \quad y \in \{0, \dots, c-1\},$$

where $\boldsymbol{x}^\top = [1, \widetilde{\boldsymbol{x}}^\top]$ and $\boldsymbol{z} = \mathbf{W}\widetilde{\boldsymbol{x}} + \boldsymbol{b}$. Find formulas for the corresponding $\boldsymbol{b}$ and $\mathbf{W}$ in terms of the linear discriminant parameters $\{\alpha_y, \boldsymbol{\mu}_y, \boldsymbol{\Sigma}_y\}_{y=0}^{c-1}$, where $\boldsymbol{\Sigma}_y = \boldsymbol{\Sigma}$ for all y.

Solution:

(b) Explain which pre-classifier has smaller approximation error: the linear discriminant or multi-logit one? Justify your answer by proving an inequality between the two approximation errors.

Solution:

5. Consider a binary classification problem where the response Y takes values in $\{-1, 1\}$. Show that optimal prediction function for the hinge loss $\text{Loss}(y, y') = (1 - yy')_+ := \max\{0,\ 1 - yy'\}$ is the same as the optimal prediction function g^* for the indicator loss:

$$g^*(\boldsymbol{x}) = \begin{cases} 1 & \text{if} \quad \mathbb{P}(Y = 1 \,|\, \boldsymbol{X} = \boldsymbol{x}) > 1/2, \\ -1 & \text{if} \quad \mathbb{P}(Y = 1 \,|\, \boldsymbol{X} = \boldsymbol{x}) < 1/2. \end{cases}$$

That is, show that

$$\mathbb{E}\,(1 - Y\,h(\boldsymbol{X}))_+ \geqslant \mathbb{E}\,(1 - Y\,g^*(\boldsymbol{X}))_+ \tag{7.29}$$

for all functions h.

Solution: We have

$$\begin{aligned}\mathbb{E}[(1 - Yh(\boldsymbol{X}))_+ \,|\, \boldsymbol{X} = \boldsymbol{x}] &= \sum_{y \in \{-1,1\}} (1 - yh(\boldsymbol{x}))_+ \mathbb{P}(Y = y \,|\, \boldsymbol{X} = \boldsymbol{x}) \\ &= (1 - h(\boldsymbol{x}))_+ \alpha(\boldsymbol{x}) + (1 + h(\boldsymbol{x}))_+ (1 - \alpha(\boldsymbol{x})),\end{aligned}$$

where $\alpha(\boldsymbol{x}) = \mathbb{P}(Y = 1 \,|\, \boldsymbol{X} = \boldsymbol{x})$. Evaluating the last expression at the conjectured optimal $g^*(\boldsymbol{x}) = \text{sign}(2\mathbb{P}(Y = 1 \,|\, \boldsymbol{X} = \boldsymbol{x}) - 1)$ yields $2\min\{\alpha(bx), 1 - \alpha(\boldsymbol{x})\}$. Hence, we need to show that for any real $h(\boldsymbol{x})$ and $\alpha(\boldsymbol{x}) \in [0, 1]$, we have

$$(1 - h(\boldsymbol{x}))_+ \alpha(\boldsymbol{x}) + (1 + h(\boldsymbol{x}))_+ (1 - \alpha(\boldsymbol{x})) \geqslant 2\min\{\alpha(\boldsymbol{x}), 1 - \alpha(\boldsymbol{x})\}.$$

Suppressing the $\boldsymbol{x}$ in the notation below, we consider three possible cases. First, $h \geqslant 1$ yields

$$\begin{aligned}(1-h)_+\alpha + (1+h)_+(1-\alpha) &= (1+h)(1-\alpha)\\ &\geqslant 2(1-\alpha) \geqslant 2\min\{\alpha, 1-\alpha\}.\end{aligned}$$

Second, $h \leqslant -1$ yields

$$\begin{aligned}(1-h)_+\alpha + (1+h)_+(1-\alpha) &= (1-h)\alpha\\ &\geqslant 2\alpha \geqslant 2\min\{\alpha, 1-\alpha\}.\end{aligned}$$

Third, for $h \in (-1, 1)$ we have

$$\begin{aligned}(1-h)_+\alpha + (1+h)_+(1-\alpha) &= (1-h)\alpha + (1+h)(1-\alpha)\\ &\geqslant (1-h)\min\{\alpha, 1-\alpha\} + (1+h)\min\{\alpha, 1-\alpha\}\\ &= 2\min\{\alpha, 1-\alpha\}.\end{aligned}$$

Therefore, in all three cases, whatever h is, the lower bound holds.

6. In Example 4.12, we applied a principal component analysis (PCA) to the `iris` data, but ignored to classify the flowers based on their feature vectors $\boldsymbol{x}$. Implement a 1-nearest neighbor algorithm, using a training set of 50 randomly chosen data pairs $(\boldsymbol{x}, y)$ from the `iris` data set. How many of the remaining 100 flowers are correctly classified? Now classify these entries with an off-the-shelf multi-logit classifier, such as can be found in the `statsmodels` and `sklearn` packages.

Solution:

7. Figure 7.13 displays two groups of data points, given in Table 7.8. The convex hulls have also been plotted. It is possible to separate the two classes of points via a straight line. In fact, many of such lines are possible. SVM gives the best separation, in the sense that the gap (margin) between the points in maximal.

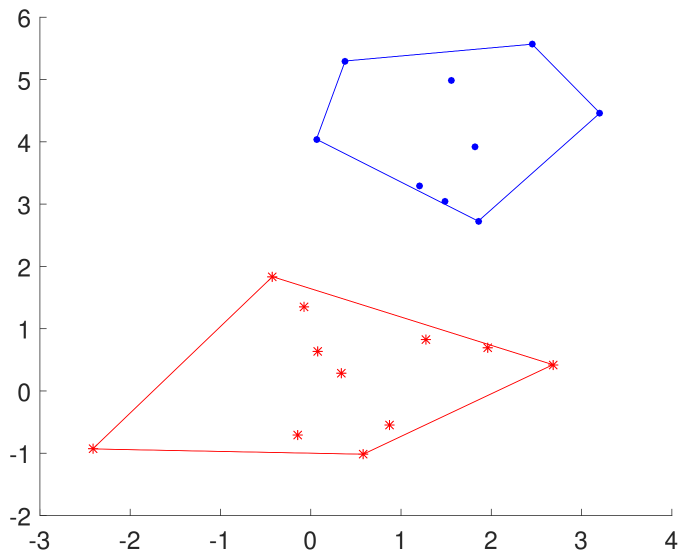


Figure 7.13: Separate the points by a straight line so that the separation between the two groups is maximal.

Table 7.8: Data for Figure 7.13.

x_1	x_2	y	x_1	x_2	y
2.4524	5.5673	-1	0.5819	-1.0156	1
1.2743	0.8265	1	1.2065	3.2984	-1
0.8773	-0.5478	1	2.6830	0.4216	1
1.4837	3.0464	-1	-0.0734	1.3457	1
0.0628	4.0415	-1	0.0787	0.6363	1
-2.4151	-0.9309	1	0.3816	5.2976	-1
1.8152	3.9202	-1	0.3386	0.2882	1
1.8557	2.7262	-1	-0.1493	-0.7095	1
-0.4239	1.8349	1	1.5554	4.9880	-1
1.9630	0.6942	1	3.2031	4.4614	-1

(a) Identify from the figure the three support vectors.

Solution: Visually, we can guess the three support vectors as follows:

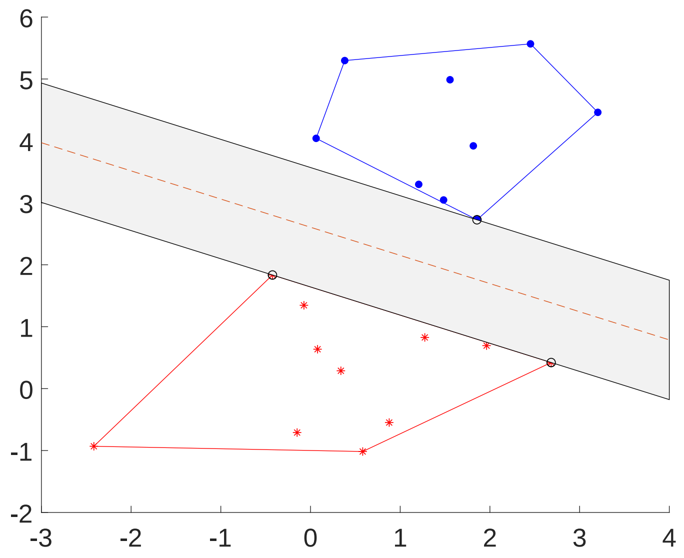


Figure 7.1: Optimal "slab" separating of the two groups and identifying the three support vectors.

(b) For a separating boundary (line) given by $\beta_0 + \boldsymbol{\beta}^\top \boldsymbol{x} = 0$, show that the margin width is $2/\|\boldsymbol{\beta}\|$.

Solution: The two margins are $P_1 : \{\boldsymbol{x} : \beta_0 + \boldsymbol{\beta}^\top \boldsymbol{x} = 1\}$ and $P_{-1} : \{\boldsymbol{x} : \beta_0 + \boldsymbol{\beta}^\top \boldsymbol{x} = -1\}$. Clearly, these are affine planes whose normal vector is $\boldsymbol{\beta}$. Take a point $\boldsymbol{x}$ on P_1. How far lies $\boldsymbol{x}$ from P_{-1}? The point closest to $\boldsymbol{x}$ must be of the form $\boldsymbol{x} + a\boldsymbol{\beta}$, as $\boldsymbol{\beta}$ is the normal vector of both planes. As this point lies in P_{-1}, it must satisfy

$$\beta_0 + \boldsymbol{\beta}^\top(\boldsymbol{x} + a\boldsymbol{\beta}) = -1.$$

Since $\beta_0 + \boldsymbol{\beta}^\top \boldsymbol{x} = 1$, it follows that $a = -2/\boldsymbol{\beta}^\top\boldsymbol{\beta} = -2/\|\boldsymbol{\beta}\|^2$. The distance between $\boldsymbol{x}$ and $\boldsymbol{x} + a\boldsymbol{\beta}$ is thus $2/\|\boldsymbol{\beta}\|$.

(c) Show that the parameters β_0 and $\boldsymbol{\beta}$ that solve the convex optimization problem (7.24) provide the maximal width between the margins.

Solution: Consider the SVM problem (7.20) and substitute: $\{\mathbf{K}\boldsymbol{\alpha}\}_i$ with $\boldsymbol{\beta}^\top\boldsymbol{x}_i$, β_0 with α_0, and $\boldsymbol{\alpha}^\top\mathbf{K}\boldsymbol{\alpha}$ with $\boldsymbol{\beta}^\top\boldsymbol{\beta} = \|\boldsymbol{\beta}\|^2$. As the data are perfectly separable here, the optimal solution will have $\beta_0 + \boldsymbol{\beta}^\top\boldsymbol{x}_i \geqslant 1$ for all i. Hence, we may remove the sum in (7.20) (which is zero for the optimal solution) and add the constraints $y_i(\beta_0 + \boldsymbol{x}_i^\top\boldsymbol{\beta}) \geqslant 1,\ i = 1, \ldots, n$. The factor γ in (7.20) is now irrelevant and we may set it to 2, giving (7.24). This is in turn equivalent to

$$\max_{\boldsymbol{\beta},\beta_0}\ 2/\|\boldsymbol{\beta}\|$$
$$\text{subject to:}\quad y_i(\beta_0 + \boldsymbol{x}_i^\top\boldsymbol{\beta}) \geqslant 1,\ i = 1, \ldots, n.$$

That is, in view of (b): maximize the margin width subject to the points being separated perfectly.

(d) Solve (7.24) using a penalty approach; see Section B.4. In particular, minimize the penalty function

$$S(\boldsymbol{\beta}, \beta_0) = \|\boldsymbol{\beta}\|^2 - C\sum_{i=1}^{n} \min\left\{(\beta_0 + \boldsymbol{\beta}^\top\boldsymbol{x}_i)\, y_i - 1,\ 0\right\}$$

for some positive penalty constant C.

Solution: Below, we minimize the costfunction S using the `fmin` method of `sciply.optimize`, setting $C = 10$.

```
import scipy.optimize
import numpy.linalg as lin
import numpy as np

from numpy import genfromtxt
Xy = genfromtxt('Xy.csv', delimiter=',')
X = Xy[:,0:2].T
z = Xy[:,2].T

C = 10

def costfun(u):
  b0 = u[0]
  b = u[1:3]
  return lin.norm(b)**2 - C*np.sum(
            np.minimum((b0 + b.T @ X)*z.T - 1, 0))

xopt = scipy.optimize.fmin(costfun, [1,1,1],
                    ftol = 1e-10, maxiter = 1000)
print(xopt)
```

```
Optimization terminated successfully.
         Current function value: 1.298388
         Iterations: 296
         Function evaluations: 543
[ 2.70314925 -0.47180565 -1.03720193]
```

We find $\beta_0 = 2.703$ and $\boldsymbol{\beta} = [-0.4718, -1.0372]^\top$, giving a maximimal margin width of 1.7552.

(e) Find the solution via **sklearn**'s **SVC** method of the dual optimization problem (7.21). Note that, as the two point sets are separable, the constraint $\boldsymbol{\lambda} \leqslant 1$ may be removed, and the value of γ can be set to 1.

Solution:

```python
import numpy as np
from numpy import genfromtxt
from sklearn.svm import SVC

Xy = genfromtxt('Xy.csv', delimiter=',')
X = Xy[:,0:2]
y = Xy[:,2]

clf = SVC(C = 1, kernel='linear')
clf.fit(X,y)
print("Support Vectors \n", clf.support_vectors_)
print("Support Vector Labels ",y[clf.support_])
print("Nu",clf.dual_coef_)
print("Bias",clf.intercept_)
print("beta",clf.coef_)
```

```
Support Vectors
 [[ 1.8557   2.7262 ]
 [-0.42393  1.8349 ]
 [ 2.683    0.42161]]
Support Vector Labels  [-1.  1.  1.]
Nu [[-0.64933185  0.32486229  0.32446956]]
Bias [2.70379174]
beta [[-0.47213216 -1.03731906]]
```

The β_0 and $\boldsymbol{\beta}$ found as a result of this "dual" optimization method agree with (although are not exactly the same as) the ones found via the approximate optimization method used for the "primal" problem in (d). Of course one could have used there more sophisticated methods.

8. In Example 7.6 we used the feature map $\boldsymbol{\phi}(\boldsymbol{x}) = [x_1, x_2, x_1^2 + x_2^2]^\top$ to classify the points. An easier way is to map the points into $\mathbb{R}^1$ via the feature map $\boldsymbol{\phi}(\boldsymbol{x}) = \|\boldsymbol{x}\|$ or any monotone functon thereof. Translated back into $\mathbb{R}^2$ this yields a circular separating boundary. Find the radius and centre of this circle, using the fact that here the sorted norms for the two groups are $\ldots, 0.4889, 0.5528, \ldots,$

Solution:

9. Let $Y \in \{0, 1\}$ be a response variable and let $h(x)$ be the regression function:

$$h(x) := \mathbb{E}[Y \,|\, X = x] = \mathbb{P}[Y = 1 \,|\, X = x]$$

Recall that the Bayes classifier is $g^*(x) = \mathbb{1}[h(x) > 1/2]$. Let $g : \mathbb{R} \mapsto \{0, 1\}$ be any other classifier function. Denote all probabilities and expectations conditional on $X = x$ as $\mathbb{P}_x[\cdot]$ and $\mathbb{E}_x[\cdot]$.

(a) Show that

$$\mathbb{P}_x[g(x) \neq Y] = \overbrace{\mathbb{P}_x[g^*(x) \neq Y]}^{\text{irreducible error}} + |2h(x) - 1| \mathbb{1}_{[g(x) \neq g^*(x)]}$$

Hence, deduce that for an learner $g_{\mathcal{T}}$ constructed from a training set $\mathcal{T}$, we have

$$\mathbb{E}[\mathbb{P}_x[g_{\mathcal{T}}(x) \neq Y \,|\, \mathcal{T}]] = \mathbb{P}_x[g^*(x) \neq Y] + |2h(x) - 1| \mathbb{P}[g_{\mathcal{T}}(x) \neq g^*(x)],$$

where the first expectation and last probability operations are with respect to $\mathcal{T}$.

Solution: We first show that for any classifier g, we have the identity

$$\mathbb{P}_{\boldsymbol{x}}[g(\boldsymbol{x}) \neq Y] - \mathbb{P}_{\boldsymbol{x}}[g^*(\boldsymbol{x}) \neq Y] = |2h(\boldsymbol{x}) - 1| \mathbb{1}\{g^*(\boldsymbol{x}) \neq g(\boldsymbol{x})\}.$$

To this end, consider the four possible cases for the value of $(g^*(\boldsymbol{x}), g(\boldsymbol{x}))$.

$(1, 1)$ **case:** here $\mathbb{P}_{\boldsymbol{x}}[1 \neq Y] - \mathbb{P}_{\boldsymbol{x}}[1 \neq Y] = 0 = |2h(\boldsymbol{x}) - 1| \mathbb{1}\{1 \neq 1\}$, so that the identity is valid;

$(0, 0)$ **case:** here $\mathbb{P}_{\boldsymbol{x}}[0 \neq Y] - \mathbb{P}_{\boldsymbol{x}}[0 \neq Y] = 0 = |2h(\boldsymbol{x}) - 1| \mathbb{1}\{0 \neq 0\}$, so that the identity is valid;

$(1, 0)$ **case:** here $\mathbb{P}_{\boldsymbol{x}}[0 \neq Y] - \mathbb{P}_{\boldsymbol{x}}[1 \neq Y] = \mathbb{P}_{\boldsymbol{x}}[Y = 1] - \mathbb{P}_{\boldsymbol{x}}[Y = 0] = 2\mathbb{P}_{\boldsymbol{x}}[Y = 1] - 1 = 2h(\boldsymbol{x}) - 1 = |2h(\boldsymbol{x}) - 1|$, since $\mathbb{1}\{2h(\boldsymbol{x}) - 1 > 0\} = g^*(\boldsymbol{x}) = 1$;

$(0, 1)$ **case:** here $\mathbb{P}_{\boldsymbol{x}}[1 \neq Y] - \mathbb{P}_{\boldsymbol{x}}[0 \neq Y] = \mathbb{P}_{\boldsymbol{x}}[Y = 0] - \mathbb{P}_{\boldsymbol{x}}[Y = 1] = 1 - 2\mathbb{P}_{\boldsymbol{x}}[Y = 1] = 1 - 2h(\boldsymbol{x}) = |2h(\boldsymbol{x}) - 1|$, since $\mathbb{1}\{2h(\boldsymbol{x}) - 1 < 0\} = 1 - g^*(\boldsymbol{x}) = 1$.

Thus, in all possible cases, the identity is valid. Therefore, replacing $g(\boldsymbol{x})$ with $g_{\mathcal{T}}(\boldsymbol{x})$ and taking expectation with respect to $\mathcal{T}$ yields:

$$\mathbb{E}[\mathbb{P}_{\boldsymbol{x}}[g_{\mathcal{T}}(\boldsymbol{x}) \neq Y \,|\, \mathcal{T}]] - \mathbb{P}_{\boldsymbol{x}}[g^*(\boldsymbol{x}) \neq Y] = |2h(\boldsymbol{x}) - 1| \mathbb{P}[g^*(\boldsymbol{x}) \neq g_{\mathcal{T}}(\boldsymbol{x})],$$

as required.

(b) Using the previous result, deduce that for the unconditional error (that is, we no longer condition on $X = x$), we have

$$\mathbb{P}[g^*(X) \neq Y] \leqslant \mathbb{P}[g_{\mathcal{T}}(X) \neq Y].$$

Solution: Since

$$\mathbb{E}[\mathbb{P}_{\boldsymbol{x}}[g_{\mathcal{T}}(\boldsymbol{x}) \neq Y \,|\, \mathcal{T}]] - \mathbb{P}_{\boldsymbol{x}}[g^*(\boldsymbol{x}) \neq Y] \geqslant 0,$$

taking an expectation with respect to $\boldsymbol{X}$ on both sides yields

$$\mathbb{P}[g_{\mathcal{T}}(\boldsymbol{X}) \neq Y] - \mathbb{P}[g^*(\boldsymbol{X}) \neq Y] \geqslant 0.$$

(c) Show that, if $g_{\mathcal{T}} := \mathbb{1}[h_{\mathcal{T}}(x) > 1/2]$ is a classifier function such that as $n \uparrow \infty$

$$h_{\mathcal{T}}(x) \xrightarrow{\text{d}} Z \sim \mathcal{N}(\mu(x), \sigma^2(x))$$

for some mean and variance functions $\mu(x)$ and $\sigma^2(x)$, respectively, then

$$\mathbb{P}_x[g_{\mathcal{T}}(x) \neq g^*(x)] \longrightarrow \Phi\left(\frac{\operatorname{sign}(1-2h(x))(2\mu(x)-1)}{2\sigma(x)}\right),$$

where Φ is the cdf of a standard normal random variable.

Solution: Direct enumeration of the possible values of $(g_{\mathcal{T}_n}(\boldsymbol{x}), g^*(\boldsymbol{x}))$ immediately confirms the identity:

$$\mathbb{1}\{g_{\mathcal{T}_n}(\boldsymbol{x}) \neq g^*(\boldsymbol{x})\} = \operatorname{sign}(2h(\boldsymbol{x})-1)\left(\mathbb{I}\{h_{\mathcal{T}_n}(\boldsymbol{x}) < 1/2\} - \mathbb{1}\{g^*(\boldsymbol{x}) = 0\}\right).$$

Therefore, taking by taking an expectation with respect to $\mathcal{T}_n$, we obtain:

$$\mathbb{P}_{\boldsymbol{x}}[g_{\mathcal{T}_n}(\boldsymbol{x}) \neq g^*(\boldsymbol{x})] = \operatorname{sign}(2h(\boldsymbol{x})-1)(\mathbb{P}_{\boldsymbol{x}}[h_{\mathcal{T}_n}(\boldsymbol{x}) < 1/2] - \mathbb{1}\{g^*(\boldsymbol{x}) = 0\}).$$

Since as $n \to \infty$ $\mathbb{P}_{\boldsymbol{x}}[h_{\mathcal{T}_n}(\boldsymbol{x}) < 1/2] \to \mathbb{P}[Z < 1/2]$ with $Z \sim \mathcal{N}(\mu(\boldsymbol{x}), \sigma^2(\boldsymbol{x}))$, we can write:

$$\begin{aligned}\mathbb{P}_{\boldsymbol{x}}[g_{\mathcal{T}_n}(\boldsymbol{x}) \neq g^*(\boldsymbol{x})] &\to \operatorname{sign}(2h(\boldsymbol{x})-1)\left(\Phi\left(\frac{1-2\mu(\boldsymbol{x})}{2\sigma(\boldsymbol{x})}\right) - \mathbb{1}\{g^*(\boldsymbol{x}) = 0\}\right) \\ &= \Phi\left(\frac{\operatorname{sign}(2h(\boldsymbol{x})-1)(1-2\mu(\boldsymbol{x}))}{2\sigma(\boldsymbol{x})}\right).\end{aligned}$$

10. The purpose of this exercise is to derive the dual program (7.21) from the primal program (7.20). The starting point is to introduce a vector of auxiliary variables $\boldsymbol{\xi} := [\xi_1, \ldots, \xi_n]^\top$ and write the primal program as

$$\begin{aligned}\min_{\alpha,\alpha_0,\xi} \ & \sum_{i=1}^n \xi_i + \frac{\gamma}{2}\boldsymbol{\alpha}^\top \mathbf{K}\boldsymbol{\alpha} \\ \text{subject to:} \quad & \boldsymbol{\xi} \geqslant \mathbf{0}, \\ & y_i(\alpha_0 + \{\mathbf{K}\boldsymbol{\alpha}\}_i) \geqslant 1 - \xi_i, \ i = 1, \ldots, n.\end{aligned} \tag{7.30}$$

(a) Apply the Lagrangian optimization theory from Section B.2.2 to obtain the Lagrangian function $\mathcal{L}(\{\alpha_0, \boldsymbol{\alpha}, \boldsymbol{\xi}\}, \{\boldsymbol{\lambda}, \boldsymbol{\mu}\})$, where $\boldsymbol{\mu}$ and $\boldsymbol{\lambda}$ are the Lagrange multipliers corresponding to the first and second inequality constraints, respectively.

Solution:

(b) Show that the Karush–Kuhn–Tucker (see Theorem B.2) conditions for optimizing $\mathcal{L}$ are:

$$\begin{aligned}\boldsymbol{\lambda}^\top \boldsymbol{y} &= 0 \\ \boldsymbol{\alpha} &= \boldsymbol{y} \odot \boldsymbol{\lambda}/\gamma \\ \mathbf{0} \leqslant \boldsymbol{\lambda} &\leqslant \mathbf{1} \\ (\mathbf{1}-\boldsymbol{\lambda}) \odot \boldsymbol{\xi} &= \mathbf{0}, \quad \lambda_i\left(y_i g(\boldsymbol{x}_i) - 1 + \xi_i\right) = 0, \ i = 1, \ldots, n \\ \boldsymbol{\xi} &\geqslant \mathbf{0}, \quad y_i g(\boldsymbol{x}_i) - 1 + \xi_i \geqslant 0, \ i = 1, \ldots, n.\end{aligned} \tag{7.31}$$

Here $\odot$ stands for componentwise multiplication; e.g., $\boldsymbol{y} \odot \boldsymbol{\lambda} = [y_1\lambda_1, \ldots, y_n\lambda_n]^\top$, and we have abbreviated $\alpha_0 + \{\mathbf{K}\boldsymbol{\alpha}\}_i$ to $g(\boldsymbol{x}_i)$, in view of (7.19). [Hint: one of the KKT conditions is $\boldsymbol{\lambda} = \mathbf{1} - \boldsymbol{\mu}$; thus we can eliminate $\boldsymbol{\mu}$.]

Solution:

(c) Using the KKT conditions (7.31), reduce the Lagrange dual function $\mathcal{L}^*(\boldsymbol{\lambda}) := \min_{\alpha_0,\boldsymbol{\alpha},\boldsymbol{\xi}} \mathcal{L}(\{\alpha_0, \boldsymbol{\alpha}, \boldsymbol{\xi}\}, \{\boldsymbol{\lambda}, \mathbf{1} - \boldsymbol{\lambda}\})$ to

$$\mathcal{L}^*(\boldsymbol{\lambda}) = \sum_{i=1}^{n} \lambda_i - \frac{1}{2\gamma} \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_i \lambda_j y_i y_j \, \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j). \tag{7.32}$$

Solution:

(d) As a consequence of (7.19) and (a)–(c), show that the optimal prediction function g_τ is given by

$$g_\tau(\boldsymbol{x}) = \alpha_0 + \frac{1}{\gamma} \sum_{i=1}^{n} y_i \lambda_i \, \kappa(\boldsymbol{x}_i, \boldsymbol{x}), \tag{7.33}$$

where $\boldsymbol{\lambda}$ is the solution to

$$\begin{aligned} &\max_{\boldsymbol{\lambda}} && \mathcal{L}^*(\boldsymbol{\lambda}) \\ &\text{subject to:} && \boldsymbol{\lambda}^\top \boldsymbol{y} = 0, \ \mathbf{0} \leqslant \boldsymbol{\lambda} \leqslant \mathbf{1}, \end{aligned} \tag{7.34}$$

and $\alpha_0 = y_j - \frac{1}{\gamma} \sum_{i=1}^{n} y_i \lambda_i \, \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$ for any j such that $\lambda_j \in (0, 1)$.

Solution:

11. Consider SVM classification as illustrated in Figure 7.7. The goal of this exercise is to classify the training points $\{(\boldsymbol{x}_i, y_i)\}$ based on the value of the multipliers $\{\lambda_i\}$ in Exercise 10. Let ξ_i be the auxilliary variable in Exercise 10, $i = 1, \ldots, n$.

(a) For $\lambda_i \in (0, 1)$ show that $(\boldsymbol{x}_i, y_i)$ lies exactly on the decision border.

Solution: From KKT condition 4 in (7.31) we have $\xi_i = 0$ and hence $g_\tau(\boldsymbol{x}_i) = y_i$; so $(\boldsymbol{x}_i, y_i)$ lies exactly on the decision border.

(b) For $\lambda_i = 1$, show that $(\boldsymbol{x}_i, y_i)$ lies strictly inside the margins.

Solution: By KKT conditions 4 and 5, we must have $y_i g(\boldsymbol{x}_i) \leqslant 1$ so these points, which are also support vectors, lie inside the margins.

(c) Show that for $\lambda_i = 0$ the point $(\boldsymbol{x}_i, y_i)$ lies outside the margins and is correctly classified.

Solution: In this case $\xi_i = 0$, which implies $y_i g_\tau(\boldsymbol{x}_i) \geqslant 1$ by KKT condition 5, so the point is correctly classified and lies outside the margins.

12. A famous data set is the MNIST handwritten digit database, containing many thousands of digitalized numbers (from 0 to 9), each described by a 28×28 matrix of gray scales. A similar but much smaller data set is described in [63]. Here, each handwritten digit is summarized by a 8×8 matrix with integer entries from 0 (white) to 15 (black). Figure 7.14 shows the first 50 digitized images. The data set can be accessed with Python using the `sklearn` package as follows.

```
from sklearn import datasets
digits = datasets.load_digits()
```

```
x_digits = digits.data    # explanatory variables
y_digits = digits.target  # responses
```

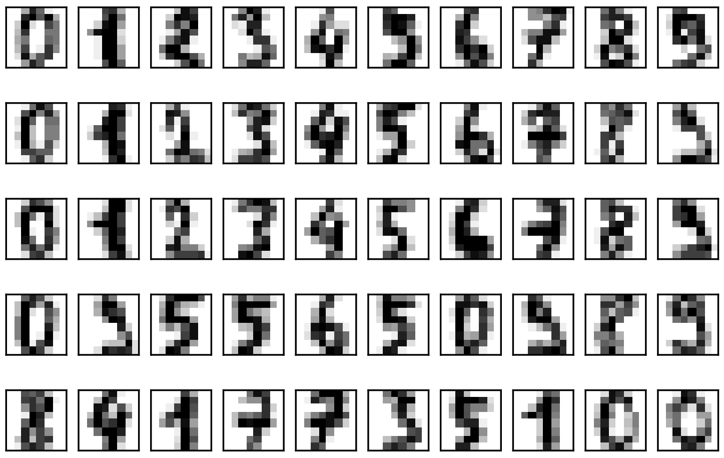


Figure 7.14: Classify the digitized images.

(a) Divide the data into a 75% training set and 25% test set.

Solution:

(b) Compare the effectiveness of the K-nearest neighbors and naive Bayes method to classify the data.

Solution:

(c) Assess which K to use in the K-nearest neigbor classification.

Solution:

13. Download the `winequality-red.csv` dataset from `https://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality`. The response here is the wine quality (from 0 to 10) as specified by a wine "expert" and the explanory variables are various characteristics such as acidity and sugar content. Use the **`SVC`** classifier of **`sklearn.svm`** with a linear kernel and penalty parameter `C` = 1 (see Remark 7.2) to fit the data. Use the method **`cross_val_score`** from **`sklearn.model_selection`** to obtain a 5-fold cross-validation score as an estimate of the probability that the predicted class matches the expert's class.

Solution:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import cross_val_score
from sklearn import svm
```

```
df = pd.read_csv('winequality-red.csv', sep=';')
y = df.quality
X = df.drop(["quality"],axis=1)

np.random.seed(12345)

clf = svm.SVC(kernel='linear', C=1)
scores = cross_val_score(clf, X, y, cv=5)
print("CV score = ",np.mean(scores))
```

```
CV score =  0.5717266874501472
```

14. Consider the credit approval dataset `crx.data` from http://archive.ics.uci.edu/ml/datasets/credit+approval. The dataset is concerned with credit card applications. The last column in the data set indicates whether the application is approved (+) or not (−). With the view of preserving data privacy, all 15 explanatory variables were anonymized. Note that some explanatory variables are continuous and some are categorical.

(a) Load and prepare the data for analysis with **`sklearn`**. First, eliminate data rows with missing values. Next,encode categorical explanatory variables using a **`OneHotEncoder`** object from **`sklearn.preprocessing`** to create a model matrix **X** with indicator variables for the categorical variables, as described in Section **5.3.5**.

(b) The model matrix should contain 653 rows and 46 columns. The response variable should be a 0/1 variable (reject/approve). We will consider several classification algorithms and test their performance (using a zero-one loss) via 10-fold cross validation.

   i. Write a function which takes 3 parameters: **X**, $\boldsymbol{y}$, and a model, and returns the 10-fold cross-validation estimate of the expected generalization risk.

   ii. Consider the following **`sklearn`** classifiers: **`KNeighborsClassifier`** ($k = 5$), **`LogisticRegression`**, and **`MPLClassifier`** (multilayer perceptron). Use the function from (i) to identify the best performing classifier.

Solution:

15. Consider a synthetic dataset that was generated in the following fashion. The explanatory variable follows a standard normal distribution. The response label is 0 if the explanatory variable is between the 0.95 and 0.05 quantiles of the standard normal distribution, and 1, otherwise. The dataset was generated using the following code.

```
import numpy as np
import scipy.stats
# generate data

np.random.seed(12345)
N = 100
X = np.random.randn(N)
q = scipy.stats.norm.ppf(0.95)
y = np.zeros(N)
```

```
y[X>=q] = 1
y[X<=-q] = 1
X = X.reshape(-1,1)
```

Compare the K-nearest neighbors classifier with $K = 5$ and Logistic regression classifier. Without computation, which classifier is likely to be better for these data? Verify your answer by coding both classifiers and printing the corresponding training 0–1 loss.

Solution: The K-nearest neighbors classifier should outperform the logistic regression classifier, since the latter separates the data into classes $(-\infty, c)$ and (c, ∞), which will both have a mix of 0 and 1 labels.

```
from sklearn.metrics import zero_one_loss
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression

model = KNeighborsClassifier(n_neighbors=5)
model.fit(X,y)
y_pred = model.predict(X)
loss = zero_one_loss(y,y_pred)
print("KNN classifier loss = ", loss)

model = LogisticRegression(solver="lbfgs")
model.fit(X,y)
y_pred = model.predict(X)
loss = zero_one_loss(y,y_pred)
print("Logistic regression loss = ", loss)
```

```
KNN classifier loss =  0.0
Logistic regression loss =  0.10999999999999999
```

16. Consider the digits dataset from Exercise 12. In this exercise, we would like to train a binary classifier for the identification of digit 8.

(a) Divide the data such that the first 1000 rows are used as the training set and the rest are used as the test set.

(b) Train the `LogisticRegression` classifier from the `sklearn.linear_model` package.

(c) “Train” a naive classifier that always returns 0. That is, the naive clasifier identifies each instance as being not 8.

(d) Compare the zero-one test losses of the logistic regression and the naive classifiers.

(e) Find the confusion matrix, the precision, and the recall of the logistic regression classifier.

(f) Find the fraction of eights that are correctly detected by the logistic regression classifier.

Solution:

17. Repeat Exercise 16 with the original MNIST dataset. Use the first 60000 rows as the train set and the remaining 10000 rows as the test set. The original dataset can be obtained using the following code.

```
from sklearn.datasets import fetch_openml

X, y = fetch_openml('mnist_784', version=1, return_X_y=True)
```

Solution:

```
import numpy as np
from sklearn.datasets import fetch_openml
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import zero_one_loss
from sklearn.metrics import confusion_matrix
from sklearn.metrics import precision_score, recall_score

X, y = fetch_openml('mnist_784', version=1, return_X_y=True)
X_train, X_test, y_train, y_test = X[:60000], X[60000:], y[:60000],
    y[60000:]

y_train = (y_train == '8')
y_test = (y_test == '8')



model = LogisticRegression(solver = "lbfgs")
model.fit(X_train,y_train)
y_pred = model.predict(X_test)
loss = zero_one_loss(y_test, y_pred)
print ("Logistic regression test loss = ", loss)

# constant prediction
y_pred_naive = np.zeros(len(y_pred))
loss = zero_one_loss(y_test, y_pred_naive)
print ("Naive classifier test loss = ", loss)

print("confusion matrix")
print(confusion_matrix(y_test, y_pred))
print("--------------------------------")
print("precision: ", precision_score(y_test, y_pred))
print("recall: ",recall_score(y_test, y_pred))
print("--------------------------------")
```

```
Logistic regression test loss =  0.05359999999999998
Naive classifier test loss =  0.09740000000000004
confusion matrix
[[8809  217]
 [ 319  655]]
--------------------------------
precision:  0.7511467889908257
recall:  0.6724845995893224
--------------------------------
```

The fraction of eights that are correctly detected by the logistic regression classifier is about 67.2%.

18. For the breast cancer data in Section 7.8, investigate and discuss whether *accuracy* is the relevant metric to use or if other metrics discussed in Section 7.2 are more appropriate.

Solution:

CHAPTER 8

Tree Methods

1. Show that any training set $\tau = \{(\boldsymbol{x}, y_i), i = 1, \ldots, n\}$ can be fitted via a tree with zero training loss.

Solution: Take, for example, a tree whose root note has the condition $\boldsymbol{x} = \boldsymbol{x}_1$ and has two child nodes. The left child is a leaf node with value y_1. The right child has the condition $\boldsymbol{x} = \boldsymbol{x}_2$. This child has also two children. The left is a leaf node with value y_2, the right has the condition $\boldsymbol{x} = \boldsymbol{x}_3$, and so on.

2. Suppose during the construction of a decision tree we wish to specify a constant regional prediction function g^w on the region $\mathcal{R}_w$, based on the training data in $\mathcal{R}_w$, say $\{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_k, y_k)\}$. Show that $g^w(\boldsymbol{x}) := k^{-1} \sum_{i=1}^{k} y_i$ minimizes the squared-error loss.

Solution:

3. Using the program from Section 8.2.4, write a basic implementation of a decision tree for a binary classification problem. Implement the misclassification, the Gini index, and the entropy score criterion. Compare the Gini index and the entropy with the misclassification score. What metric is more sensitive to node's impurity?

Solution:

```python
# BasicTreeExtended.py
import numpy as np
from sklearn.datasets import make_friedman1, make_blobs
from sklearn.model_selection import train_test_split
from enum import Enum
from sklearn.metrics import zero_one_loss

# deferent loss types
class LossType(Enum):
    SQER    = 1
    MISSCLF = 2
    GINI    = 3
    CE      = 4

def makedata():
  n_points = 500 # points
  X, y =  make_friedman1(n_samples=n_points, n_features=5,
                         noise=1.0, random_state=100)

  return train_test_split(X, y, test_size=0.5, random_state=3)

def makedata_clf():
  n_points = 500 # points

  X, y =  make_blobs(n_samples=n_points, centers=2, n_features=5,
                   random_state=10, cluster_std=5)

  return train_test_split(X, y, test_size=0.5, random_state=3)

# tree node
class TNode:
   def __init__(self, depth, X, y, lossType):
      global n
      self.depth = depth
      self.X = X   # matrix of explanatory variables
      self.y = y   # vector of response variables
      self.lossType = lossType
      # initialize optimal split parameters
      self.j = None
      self.xi = None
      # initialize children to be None
      self.left = None
      self.right = None
      # initialize the regional predictor
      self.g = None

   def CalculateLoss(self):
       if(len(self.y)==0):
           return 0
       if(self.lossType == LossType.SQER):
           return np.sum(np.power(self.y- self.y.mean(),2))

       # count class instances
       c0 = len(self.y[self.y==0])
       c1 = len(self.y[self.y==1])

       if(self.lossType == LossType.MISSCLF):
           return (1 - max(c0,c1)/(c0+c1))*(len(self.y)/n)

       p0 = c0/(c0+c1)
       p1 = c1/(c0+c1)
       if(self.lossType == LossType.GINI):
           return (p0*(1-p0) + p1*(1-p1))*(len(self.y)/n)
       if(self.lossType == LossType.CE):
           if(p0==0 or p1==0):
               return 0
           else:
               return (-0.5*(p0*np.log2(p0) + p1*np.log2(p1)))*(len(
                   self.y)/n)
```

```python
def Construct_Subtree(node, max_depth,lossType):
    if(node.depth == max_depth or len(node.y) == 1):
        node.g  = node.y.mean()
    else:
        j, xi = CalculateOptimalSplit(node, lossType)
        node.j = j
        node.xi = xi
        Xt, yt, Xf, yf = DataSplit(node.X, node.y, j, xi)

        if(len(yt)>0):
            node.left = TNode(node.depth+1,Xt,yt, lossType)
            Construct_Subtree(node.left, max_depth, lossType)

        if(len(yf)>0):
            node.right = TNode(node.depth+1, Xf,yf, lossType)
            Construct_Subtree(node.right, max_depth, lossType)

    return node

# split the data-set
def DataSplit(X,y,j,xi):
    ids = X[:,j]<=xi
    Xt  = X[ids == True,:]
    Xf  = X[ids == False,:]
    yt  = y[ids == True]
    yf  = y[ids == False]
    return Xt, yt, Xf, yf

def CalculateOptimalSplit(node, lossType):
    X = node.X
    y = node.y
    best_var = 0
    best_xi = X[0,best_var]
    best_split_val = node.CalculateLoss()

    m, n  = X.shape

    for j in range(0,n):
        for i in range(0,m):
            xi = X[i,j]
            Xt, yt, Xf, yf = DataSplit(X,y,j,xi)
            tmpt = TNode(0, Xt, yt, lossType)
            tmpf = TNode(0, Xf, yf, lossType)
            loss_t = tmpt.CalculateLoss()
            loss_f = tmpf.CalculateLoss()
            curr_val =  loss_t + loss_f
            if (curr_val < best_split_val):
                best_split_val = curr_val
                best_var = j
                best_xi = xi
    return best_var,  best_xi


def Predict(X,node):
    if(node.right == None and node.left != None):
        return Predict(X,node.left)

    if(node.right != None and node.left == None):
        return Predict(X,node.right)

    if(node.right == None and node.left == None):
        return node.g
    else:
        if(X[node.j] <= node.xi):
            return Predict(X,node.left)
        else:
            return Predict(X,node.right)

if __name__ == "__main__":
    X_train, X_test, y_train, y_test = makedata_clf()
    maxdepth = 10 # maximum tree depth

    n = len(X_train)

    # define loss type
    lossType = LossType.GINI

    # Create tree root at depth 0
    treeRoot = TNode(0, X_train,y_train,lossType)

    # Build the regression tree with maximal depth equal to
        max_depth
    Construct_Subtree(treeRoot, maxdepth,lossType)

    # Predict
    y_hat = np.zeros(len(X_test))
    for i in range(len(X_test)):
        y_hat[i] = Predict(X_test[i],treeRoot)

    print("Basic tree: tree loss = ",  zero_one_loss(y_test, np.
        int64(y_hat)))
```

4. Suppose in the decision tree of Example 8.1, there are 3 blue and 2 red data points in a certain tree region. Calculate the misclassification impurity, the Gini impurity, and the entropy impurity. Repeat these calculations for 2 blue and 3 red data points.

Solution:

5. Consider the procedure of finding the best splitting rule for categorical variable with c labels from Section 8.3.4. Show that one needs to consider 2^{c-1} subsets of $\{1, \dots, c\}$ to find the optimal partition of labels.

Solution: Denote the partitions by 0 and 1. Fix the first label to be in partition 0. Then, the number of possible subsets of the remaining labels is equal to the number of binary vectors of length $c - 1$ and the answer follows.

6. Reproduce Figure 8.6 using the following classification data.

```
from sklearn.datasets import make_blobs
X, y =  make_blobs(n_samples=5000, n_features=10, centers=3,
                              random_state=10, cluster_std=10)
```

Solution:

7. Prove (8.13); that is, show that

$$\sum_{r=1}^{|T|}\left(\sum_{i=1}^{n}\mathbb{1}_{\{\boldsymbol{x}_i\in\mathcal{R}_r\}}\,\text{Loss}(y_i, g_r(\boldsymbol{x}_i))\right) = n\,\ell_\tau\,(g)\,.$$

Solution: It holds that:

$$\sum_{r=1}^{|T|}\left(\sum_{i=1}^{n}\mathbb{1}_{\{\boldsymbol{x}_i\in\mathcal{R}_r\}}\,\text{Loss}(y_i, g_r(\boldsymbol{x}_i))\right) = n\sum_{r=1}^{|T|}\frac{n_r}{n}\left(\frac{1}{n_r}\sum_{i=1}^{n}\mathbb{1}_{\{\boldsymbol{x}_i\in\mathcal{R}_r\}}\,\text{Loss}(y_r, g_r(\boldsymbol{x}_i))\right)\underbrace{=}_{(8.2)} n\,\ell_\tau\,(g)\,.$$

8. Suppose τ is a training set with n elements and τ^*, also of size n, is obtained from τ by bootstrapping; that is, resampling with replacement. Show that for large n, τ^* does not contain a fraction of about $\mathrm{e}^{-1}\approx 0.37$ of the points from τ.

Solution:

9. Prove Equation (8.17).

Solution: First note that

$$\varrho = \frac{\text{Cov}(X,Y)}{\sqrt{\mathbb{V}\text{ar}(X)\mathbb{V}\text{ar}(Y)}} \Rightarrow \text{Cov}(X,Y) = \varrho\sigma^2.$$

Therefore,

$$\begin{aligned}\mathbb{V}\text{ar}(\overline{X}_B) &= \mathbb{V}\text{ar}\left(\frac{1}{B}\sum_{b=1}^{B}X_b\right) = \frac{1}{B^2}\left[\sum_{b=1}^{B}\mathbb{V}\text{ar}\,(X_b) + 2\sum_{1\leqslant i<j\leqslant B}\text{Cov}(X_i,X_j)\right]\\ &= \frac{1}{B^2}\left[B\sigma^2 + 2\frac{B(B-1)}{2}\varrho\sigma^2\right] = \frac{1}{B^2}\left[B\sigma^2 + B^2\varrho\sigma^2 - B\varrho\sigma^2\right] = \varrho\sigma^2 + \sigma^2\frac{1-\varrho}{B}.\end{aligned}$$

10. Consider the following train/test split of the data. Construct random forest regressor and identify the optimal parameter m in the sense of R^2 score, (see Remark 8.3).

```
import numpy as np
from sklearn.datasets import make_friedman1
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score

# create regression problem
n_points = 1000 # points
x, y =  make_friedman1(n_samples=n_points, n_features=15,
                            noise=1.0, random_state=100)

# split to train/test set
x_train, x_test, y_train, y_test = \
        train_test_split(x, y, test_size=0.33, random_state=100)
```

Solution:

11. Explain why bagging decision trees is a special case of random forest.

Solution: It is easy to see that bagging is a special case of random forest. Specifically, a random forest g_{rf} with $m = p$, satisfies $g_{\text{rf}} := g_{\text{bag}}$.

12. Show that **(8.28)** holds.

Solution:

13. Consider the following classification data and module imports:

```
from sklearn.datasets import make_blobs
from sklearn.metrics import zero_one_loss
from sklearn.model_selection import train_test_split
import numpy as np
import matplotlib.pyplot as plt
from sklearn.ensemble import GradientBoostingClassifier

X_train, y_train =  make_blobs(n_samples=5000, n_features=10,
   centers=3, random_state=10, cluster_std=5)
```

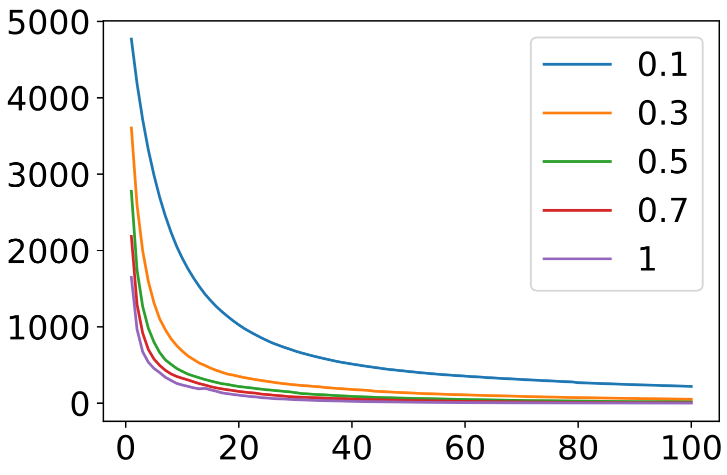
Using the gradient boosting algorithm with $B = 100$ rounds, plot the training loss as a function of γ, for $\gamma = 0.1, 0.3, 0.5, 0.7, 1$. What is your conclusion regarding the relation between B and γ?

Solution:

```
from sklearn.datasets import make_blobs
from sklearn.metrics import zero_one_loss
from sklearn.model_selection import train_test_split
import numpy as np
import matplotlib.pyplot as plt
from sklearn.ensemble import GradientBoostingClassifier

if __name__ == "__main__":
    X_train, y_train =  make_blobs(n_samples=5000, n_features=10,
                  centers=3,random_state=10, cluster_std=5)

    gamma = [0.1,0.3,0.5,0.7,1]
    fig = plt.figure()
    for g in gamma:
        bclf = GradientBoostingClassifier(learning_rate=g)
        bclf.fit(X_train,y_train)
        plt.plot(np.linspace(1,100,100),bclf.train_score_,label=g)
        plt.legend()
    fig.show()
```


5000
4000
3000
2000
1000
0
0
20
40
60
80
100
0.1
0.3
0.5
0.7
1

CHAPTER 9

Deep Learning

1. Show that the softmax function

$$\text{Softmax} : \boldsymbol{z} \mapsto \frac{\exp(\boldsymbol{z})}{\sum_k \exp(z_k)}.$$

satisfies the invariance property:

$$\text{Softmax}(\boldsymbol{z}) = \text{Softmax}(\boldsymbol{z} + c \times \mathbf{1}), \quad \text{for any constant } c.$$

Solution: Let $\boldsymbol{w} := \text{Softmax}(\boldsymbol{z} + c \times \mathbf{1})$ and $\boldsymbol{u} := \text{Softmax}(\boldsymbol{z})$. For every i, we have from the definition of the softmax function:

$$\begin{aligned} w_i &= \frac{\exp(z_i + c)}{\sum_k \exp(z_k + c)} \\ &= \frac{\exp(c)\exp(z_i)}{\sum_k \exp(c)\exp(z_k)} \\ &= \frac{\exp(z_i)}{\sum_k \exp(z_k)} \\ &= u_i. \end{aligned}$$

This implies the identity.

2. *Projection Pursuit* is a network with one hidden layer that can be written as:

$$g(\boldsymbol{x}) = S(\boldsymbol{\omega}^\top \boldsymbol{x}),$$

where S is a univariate *smoothing cubic spline*. If we use squared-error loss with $\tau_n = \{y_i, \boldsymbol{x}_i\}_{i=1}^n$, we need to minimize the training loss:

$$\frac{1}{n}\sum_{i=1}^n (y_i - S(\boldsymbol{\omega}^\top \boldsymbol{x}_i))^2$$

with respect to $\boldsymbol{\omega}$ and all cubic smoothing splines. This training of the network is typically tackled iteratively in a manner similar to the *EM algorithm*. In particular, we iterate ($t = 1, 2, \ldots$) the following steps until convergence.

(a) Given the *missing data* $\boldsymbol{\omega}_t$, compute the spline S_t by training a cubic smoothing spline on $\{y_i, \boldsymbol{\omega}_t^\top \boldsymbol{x}_i\}$. The smoothing coefficient of the spline may be determined as part of this step.

(b) Given the spline function S_t, compute the next projection vector $\boldsymbol{\omega}_{t+1}$ via *iterative reweighted least squares*:

$$\boldsymbol{\omega}_{t+1} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\ (\boldsymbol{e}_t - \mathbf{X}\boldsymbol{\beta})^\top \boldsymbol{\Sigma}_t (\boldsymbol{e}_t - \mathbf{X}\boldsymbol{\beta}), \tag{9.11}$$

where

$$e_{t,i} := \boldsymbol{\omega}_t^\top \boldsymbol{x}_i + \frac{y_i - S_t(\boldsymbol{\omega}_t^\top \boldsymbol{x}_i)}{S_t'(\boldsymbol{\omega}_t^\top \boldsymbol{x}_i)}, \quad i = 1, \ldots, n$$

is the adjusted response, and $\boldsymbol{\Sigma}_t^{1/2} = \operatorname{diag}(S_t'(\boldsymbol{\omega}_t^\top \boldsymbol{x}_1), \ldots, S_t'(\boldsymbol{\omega}_t^\top \boldsymbol{x}_n))$ is a diagonal matrix.

Apply Taylor's Theorem B.1 to the function S_t and derive the iterative reweighted least squares optimization program (9.11).

Solution:

3. Suppose that in the *stochastic gradient descent* method we wish to repeatedly draw minibatches of size N from τ_n, where we assume that $N \times m = n$ for some large integer m. Instead of repeatedly resampling from τ_n, an alternative is to reshuffle τ_n via a random permutation $\boldsymbol{\Pi}$ and then advance sequentially through the reshuffled training set to construct m non-overlapping minibatches. A single traversal of such a reshuffled training set is called an *epoch*. The following pseudo-code describes the procedure.

Algorithm 9.5.1: Stochastic Gradient Descent with Reshuffling

```
input: Training set $\tau_n = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^n$, initial weight matrices and bias vectors
       $\{\mathbf{W}_l, \boldsymbol{b}_l\}_{l=1}^L \to \boldsymbol{\theta}_1$, activation functions $\{\boldsymbol{S}_l\}_{l=1}^L$, learning rates $\{\alpha_1, \alpha_2, \ldots\}$.
output: The parameters of the trained learner.
1  $t \leftarrow 1$ and epoch $\leftarrow 0$
2  while stopping condition is not met do
3      Draw $U_1, \ldots, U_n \overset{\text{iid}}{\sim} \mathcal{U}(0,1)$.
4      Let $\boldsymbol{\Pi}$ be the permutation of $\{1, \ldots, n\}$ that satisfies $U_{\Pi_1} < \cdots < U_{\Pi_n}$.
5      $(\boldsymbol{x}_i, \boldsymbol{y}_i) \leftarrow (\boldsymbol{x}_{\Pi_i}, \boldsymbol{y}_{\Pi_i})$ for $i = 1, \ldots, n$            // reshuffle $\tau_n$
6      for $j = 1, \ldots, m$ do
7          $\widehat{\ell_\tau} \leftarrow \frac{1}{N}\sum_{i=(j-1)N+1}^{jN} \text{Loss}(\boldsymbol{y}_i, \boldsymbol{g}(\boldsymbol{x}_i \mid \boldsymbol{\theta}))$
8          $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \alpha_t \frac{\partial \widehat{\ell_\tau}}{\partial \boldsymbol{\theta}}(\boldsymbol{\theta}_t)$
9          $t \leftarrow t + 1$
10     epoch $\leftarrow$ epoch + 1          // number of reshuffles or epochs
11 return $\boldsymbol{\theta}_t$ as the minimizer of the training loss
```

Write Python code that implements the stochastic gradient descent with data reshuffling, and use it to train the neural net in Section 9.5.1.

Solution: The following code implements the stochastic gradient descent with data reshuffling. Note that the code uses `np.random.permutation(n)` to simulate a random permutation.

```
import numpy as np
import matplotlib.pyplot as plt
#%%
# import data
data = np.genfromtxt('polyreg.csv',delimiter=',')
X = data[:,0].reshape(-1,1)
y = data[:,1].reshape(-1,1)
# Network setup
p = [X.shape[1],20,20,1] # size of layers
L = len(p)-1 # number of layers
#%%
def initialize(p, w_sig = 1):
    W, b = [[]]*len(p), [[]]*len(p)
    for l in range(1,len(p)):
        W[l]= w_sig * np.random.randn(p[l], p[l-1])
        b[l]= w_sig * np.random.randn(p[l], 1)
    return W,b
W,b = initialize(p) # initialize weight matrices and bias
    vectors
#%%
def RELU(z,l): # RELU activation function: value and derivative
    if l == L: return z, np.ones_like(z)
    else:
        val = np.maximum(0,z) # RELU function element -wise
        J = np.array(z>0, dtype = float) # derivative of RELU
            element -wise
        return val, J
def loss_fn(y,g):
    return (g - y)**2, 2 * (g - y)
S=RELU
#%%
def feedforward(x,W,b):
    a, z, gr_S = [0]*(L+1), [0]*(L+1), [0]*(L+1)
    a[0] = x.reshape(-1,1)
    for l in range(1,L+1):
        z[l] = W[l] @ a[l-1] + b[l] # affine transformation
        a[l], gr_S[l] = S(z[l],l) # activation function
    return a, z, gr_S
#%%
def backward(W,b,X,y):
    n =len(y)
    delta = [0]*(L+1)
    dC_db , dC_dW = [0]*(L+1), [0]*(L+1)
    loss=0
    for i in range(n): # loop over training examples
        a, z, gr_S = feedforward(X[i,:].T, W, b)
        cost , gr_C = loss_fn(y[i], a[L]) # cost i and gradient
```

```
                wrt g
            loss += cost/n
            delta[L] = gr_S[L] @ gr_C
            for l in range(L,0,-1): # l = L,...,1
                dCi_dbl = delta[l]
                dCi_dWl = delta[l] @ a[l-1].T
                # ---- sum up over samples ----
                dC_db[l] = dC_db[l] + dCi_dbl/n
                dC_dW[l] = dC_dW[l] + dCi_dWl/n
                # -----------------------------
                delta[l-1] = gr_S[l-1] * W[l].T @ delta[l]
    return dC_dW , dC_db , loss
#%%
def list2vec(W,b):
# converts list of weight matrices and bias vectors into
# one column vector
    b_stack = np.vstack([b[i] for i in range(1,len(b))] )
    W_stack = np.vstack(W[i].flatten().reshape(-1,1) for i in
        range
    (1,len(W)))
    vec = np.vstack([b_stack , W_stack])
    return vec
#%%
def vec2list(vec, p):
# converts vector to weight matrices and bias vectors
    W, b = [[]]*len(p) ,[[]]*len(p)
    p_count = 0
    for l in range(1,len(p)): # construct bias vectors
        b[l] = vec[p_count:(p_count+p[l])].reshape(-1,1)
        p_count = p_count + p[l]
    for l in range(1,len(p)): # construct weight matrices
        W[l] = vec[p_count:(p_count + p[l]*p[l-1])].reshape(p[l
            ], p[
        l-1])
        p_count = p_count + (p[l]*p[l-1])
    return W, b
#%%
batch_size = 20
lr = 0.005
beta = list2vec(W,b)
loss_arr = []
n = len(X)
num_epochs = 1000
print("epoch | batch loss")
print("----------------------------")
for epoch in range(1,num_epochs+1):
    perm=np.random.permutation(n)
    for j in range(1,int(n/batch_size)):
        batch_idx = range((j-1)*batch_size,j*batch_size)
        batch_idx=perm[batch_idx]
        batch_X = X[batch_idx].reshape(-1,1)
        batch_y=y[batch_idx].reshape(-1,1)
        dC_dW , dC_db , loss = backward(W,b,batch_X ,batch_y)
        d_beta = list2vec(dC_dW ,dC_db)
        loss_arr.append(loss.flatten()[0])
        if(epoch==1 or np.mod(epoch ,1000)==0):
            print(epoch ,": ",loss.flatten()[0])
        beta = beta - lr*d_beta
        W,b = vec2list(beta ,p)
    # calculate the loss of the entire training set
dC_dW , dC_db , loss = backward(W,b,X,y)
print("entire training set loss = ",loss.flatten()[0])
xx = np.arange(0,1,0.01)
y_preds = np.zeros_like(xx)
for i in range(len(xx)):
    a, _, _ = feedforward(xx[i],W,b)
    y_preds[i], = a[L]
plt.plot(X,y, 'r.', markersize = 4,label = 'y')
plt.plot(np.array(xx), y_preds , 'b',label = 'fit')
plt.legend()
plt.xlabel('x')
plt.ylabel('y')
plt.show()
plt.plot(np.array(loss_arr), 'b')
plt.xlabel('iteration')
plt.ylabel('Training Loss')
plt.show()
```

4. Denote the pdf of the $\mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$ distribution by $\varphi_{\mathbf{\Sigma}}(\cdot)$, and let

$$\mathcal{D}(\boldsymbol{\mu}_0, \mathbf{\Sigma}_0 \,|\, \boldsymbol{\mu}_1, \mathbf{\Sigma}_1) = \int_{\mathbb{R}^d} \varphi_{\mathbf{\Sigma}_0}(\boldsymbol{x} - \boldsymbol{\mu}_0) \ln \frac{\varphi_{\mathbf{\Sigma}_0}(\boldsymbol{x} - \boldsymbol{\mu}_0)}{\varphi_{\mathbf{\Sigma}_1}(\boldsymbol{x} - \boldsymbol{\mu}_1)} \,\mathrm{d}\boldsymbol{x}$$

be the Kullback–Leibler divergence between the densities of the $\mathcal{N}(\boldsymbol{\mu}_0, \mathbf{\Sigma}_0)$ and $\mathcal{N}(\boldsymbol{\mu}_1, \mathbf{\Sigma}_1)$ distributions on $\mathbb{R}^d$. Show that

$$2\mathcal{D}(\boldsymbol{\mu}_0, \mathbf{\Sigma}_0 \,|\, \boldsymbol{\mu}_1, \mathbf{\Sigma}_1) = \mathrm{tr}(\mathbf{\Sigma}_1^{-1}\mathbf{\Sigma}_0) - \ln |\mathbf{\Sigma}_1^{-1}\mathbf{\Sigma}_0| + (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)^\top \mathbf{\Sigma}_1^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0) - d.$$

Hence, deduce the formula in **(B.22)**.

Solution:

5. Suppose that we wish to compute the inverse and log-determinant of the matrix

$$\mathbf{I}_n + \mathbf{U}\mathbf{U}^\top,$$

where $\mathbf{U}$ is an $n \times h$ matrix with $h \ll n$. Show that

$$(\mathbf{I}_n + \mathbf{U}\mathbf{U}^\top)^{-1} = \mathbf{I}_n - \mathbf{Q}_n\mathbf{Q}_n^\top,$$

where $\mathbf{Q}_n$ contains the first n rows of the $(n + h) \times h$ matrix $\mathbf{Q}$ in the QR factorization of the $(n + h) \times h$ matrix:

$$\begin{bmatrix} \mathbf{U} \\ \mathbf{I}_h \end{bmatrix} = \mathbf{QR}.$$

In addition, show that $\ln |\mathbf{I}_n + \mathbf{U}\mathbf{U}^\top| = \sum_{i=1}^h \ln r_{ii}^2$, where $\{r_{ii}\}$ are the diagonal elements of the $h \times h$ matrix $\mathbf{R}$.

Solution: First note that

$$\begin{bmatrix} \mathbf{U}^\top & \mathbf{I}_h \end{bmatrix} \begin{bmatrix} \mathbf{U} \\ \mathbf{I}_h \end{bmatrix} = (\mathbf{QR})^\top \mathbf{QR}.$$

Therefore,

$$\mathbf{I}_h + \mathbf{U}^\top \mathbf{U} = \mathbf{R}^\top \mathbf{R}$$

and $|\mathbf{R}|^2 = |\mathbf{I}_h + \mathbf{U}^\top\mathbf{U}|$. Hence, from the Woodbery determinant identity (A.16), we conclude that

$$2 \ln |\mathbf{R}| = \ln |\mathbf{I}_h + \mathbf{U}^\top\mathbf{U}| = \ln |\mathbf{I}_n + \mathbf{U}\mathbf{U}^\top|.$$

From the Woodbery matrix identity (A.15) we also have

$$(\mathbf{I}_n + \mathbf{U}\mathbf{U}^\top)^{-1} = \mathbf{I}_n - \mathbf{U}(\mathbf{I}_h + \mathbf{U}^\top\mathbf{U})^{-1}\mathbf{U}^\top.$$

Since $\mathbf{U} = \mathbf{Q}_n\mathbf{R}$, we have

$$\begin{aligned} \mathbf{U}(\mathbf{I}_h + \mathbf{U}^\top\mathbf{U})^{-1}\mathbf{U}^\top &= \mathbf{Q}_n\mathbf{R}(\mathbf{R}^\top\mathbf{R})^{-1}\mathbf{R}^\top\mathbf{Q}_n^\top \\ &= \mathbf{Q}_n\mathbf{R}\mathbf{R}^{-1}\mathbf{R}^{-\top}\mathbf{R}^\top\mathbf{Q}_n^\top \\ &= \mathbf{Q}_n\mathbf{Q}_n^\top, \end{aligned}$$

whence we deduce the desired result.

6. Suppose that

$$\mathbf{U} = [\boldsymbol{u}_0, \boldsymbol{u}_1, \ldots, \boldsymbol{u}_{h-1}],$$

where all $\boldsymbol{u} \in \mathbb{R}^n$ are column vectors and we have computed $(\mathbf{I}_n + \mathbf{U}\mathbf{U}^\top)^{-1}$ via the QR factorization method in Exercise 5. If the columns of matrix $\mathbf{U}$ are updated to

$$[\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{h-1}, \boldsymbol{u}_h],$$

show that the inverse $(\mathbf{I}_n + \mathbf{U}\mathbf{U}^\top)^{-1}$ can be updated in $O(h\,n)$ time (rather than computed from scratch in $O(h^2\,n)$ time). Deduce that the computing cost of updating the Hessian approximation (9.10) is the same as that for the *limited-memory BFGS* Algorithm 9.4.3.

In your solution you may use the following facts. Suppose we are given the $\mathbf{Q}$ and $\mathbf{R}$ factors in the QR factorization of a matrix $\mathbf{A} \in \mathbb{R}^{n \times h}$. If a row/column is added to matrix $\mathbf{A}$, then the $\mathbf{Q}$ and $\mathbf{R}$ factors need not be recomputed from scratch (in $O(h^2\,n)$ time), but can be updated efficiently in $O(h\,n)$ time. Similarly, if a row/column is removed from matrix $\mathbf{A}$, then the $\mathbf{Q}$ and $\mathbf{R}$ factors can be updated in $O(h^2)$ time.

Solution:

7. Suppose that $\mathbf{U} \in \mathbb{R}^{n \times h}$ has its k-th column $\boldsymbol{v}$ replaced with $\boldsymbol{w}$, giving the updated $\widetilde{\mathbf{U}}$.

(a) If $\boldsymbol{e} \in \mathbb{R}^h$ denotes the unit-length vector such that $e_k = \|\boldsymbol{e}\| = 1$ and

$$\boldsymbol{r}_\pm := \frac{\sqrt{2}}{2}\mathbf{U}^\top(\boldsymbol{w}-\boldsymbol{v}) + \frac{\sqrt{2}\,\|\boldsymbol{w}-\boldsymbol{v}\|^2}{4}\boldsymbol{e} \pm \frac{\sqrt{2}}{2}\boldsymbol{e},$$

show that

$$\widetilde{\mathbf{U}}^\top\widetilde{\mathbf{U}} = \mathbf{U}^\top\mathbf{U} + \boldsymbol{r}_+\boldsymbol{r}_+^\top - \boldsymbol{r}_-\boldsymbol{r}_-^\top.$$

[Hint: You may find Exercise **16** in Chapter **6** useful.]

Solution: We have

$$\widetilde{\mathbf{U}} = \mathbf{U} + (\boldsymbol{w}-\boldsymbol{v})\boldsymbol{e}.$$

Set

$$\boldsymbol{\delta} = \mathbf{U}^\top(\boldsymbol{w}-\boldsymbol{v}) + \frac{\|\boldsymbol{w}-\boldsymbol{v}\|^2}{2}\boldsymbol{u}.$$

Then, in Exercise **16** part a), we showed that

$$\widetilde{\mathbf{U}}^\top\widetilde{\mathbf{U}} = \mathbf{U}^\top\mathbf{U} + \frac{(\boldsymbol{\delta}+\boldsymbol{e})(\boldsymbol{\delta}+\boldsymbol{e})^\top}{2} - \frac{(\boldsymbol{\delta}-\boldsymbol{e})(\boldsymbol{\delta}-\boldsymbol{e})^\top}{2}.$$

Hence, the result follows by recognizing that

$$\boldsymbol{r}_+ = \frac{\sqrt{2}}{2}(\boldsymbol{\delta}+\boldsymbol{e}), \qquad \boldsymbol{r}_- = \frac{\sqrt{2}}{2}(\boldsymbol{\delta}-\boldsymbol{e}).$$

(b) Let $\mathbf{B} := (\mathbf{I}_h + \mathbf{U}^\top\mathbf{U})^{-1}$. Use the *Woodbury identity* **(A.15)** to show that

$$(\mathbf{I}_n + \widetilde{\mathbf{U}}\widetilde{\mathbf{U}}^\top)^{-1} = \mathbf{I}_n - \widetilde{\mathbf{U}}\left(\mathbf{B}^{-1} + \boldsymbol{r}_+\boldsymbol{r}_+^\top - \boldsymbol{r}_-\boldsymbol{r}_-^\top\right)^{-1}\widetilde{\mathbf{U}}^\top.$$

Solution:

Applying the identity **(A.15)**, we have

$$\begin{aligned}(\mathbf{I}_n + \widetilde{\mathbf{U}}\widetilde{\mathbf{U}}^\top)^{-1} &= \mathbf{I}_n - \widetilde{\mathbf{U}}(\mathbf{I}_h + \widetilde{\mathbf{U}}^\top\widetilde{\mathbf{U}})^{-1}\widetilde{\mathbf{U}}^\top \\ &= \mathbf{I}_n - \widetilde{\mathbf{U}}(\mathbf{I}_h + \mathbf{U}^\top\mathbf{U} + \boldsymbol{r}_+\boldsymbol{r}_+^\top - \boldsymbol{r}_-\boldsymbol{r}_-^\top)^{-1}\widetilde{\mathbf{U}}^\top \\ &= \mathbf{I}_n - \widetilde{\mathbf{U}}(\mathbf{B}^{-1} + \boldsymbol{r}_+\boldsymbol{r}_+^\top - \boldsymbol{r}_-\boldsymbol{r}_-^\top)^{-1}\widetilde{\mathbf{U}}^\top.\end{aligned}$$

(c) Suppose that we have stored $\mathbf{B}$ in computer memory. Use Algorithm **6.8.1** and parts (a) and (b) to write pseudo-code that updates $(\mathbf{I}_n + \mathbf{U}\mathbf{U}^\top)^{-1}$ to $(\mathbf{I}_n + \widetilde{\mathbf{U}}\widetilde{\mathbf{U}}^\top)^{-1}$ in $O((n+h)h)$ computing time.

Solution: Let

$$\mathbf{C} := (\mathbf{B}^{-1} + \boldsymbol{r}_+\boldsymbol{r}_+^\top)^{-1} = \mathbf{B} - \frac{\mathbf{B}\boldsymbol{r}_+\boldsymbol{r}_+^\top\mathbf{B}}{1 + \boldsymbol{r}_+\mathbf{B}\boldsymbol{r}_+^\top},$$

then

$$(\mathbf{B}^{-1} + \boldsymbol{r}_+\boldsymbol{r}_+^\top - \boldsymbol{r}_-\boldsymbol{r}_-^\top)^{-1} = \mathbf{C} + \frac{\mathbf{C}\boldsymbol{r}_-\boldsymbol{r}_-^\top\mathbf{C}}{1 - \boldsymbol{r}_-\mathbf{C}\boldsymbol{r}_-^\top}.$$

This suggests the following pseudo-code for updating $(\mathbf{I}_n + \mathbf{U}\mathbf{U}^\top)^{-1}$ to $(\mathbf{I}_n + \widetilde{\mathbf{U}}\widetilde{\mathbf{U}}^\top)^{-1}$ in $O((n+h)h)$ computing time..

Algorithm 6.8.1: Updating via Sherman–Morrison Formula

```
input: Matrices $\mathbf{U}$ and $\mathbf{B} = (\mathbf{I}_n + \mathbf{U}\mathbf{U}^\top)^{-1}$, index $k$, and replacement $\boldsymbol{w}$ for the $k$-th
       column $\boldsymbol{v}$ of $\mathbf{U}$.
output: Updated matrices $\widetilde{\mathbf{U}}$ and $(\mathbf{I}_n + \widetilde{\mathbf{U}}\widetilde{\mathbf{U}}^\top)^{-1}$.
1 Set $\boldsymbol{e} \in \mathbb{R}^h$ to be the unit-length vector such that $e_k = \|\boldsymbol{e}\| = 1$.
2 Set $\boldsymbol{r}_\pm \leftarrow \frac{\sqrt{2}}{2}\mathbf{U}^\top(\boldsymbol{w} - \boldsymbol{v}) + \frac{\sqrt{2}\,\|\boldsymbol{w}-\boldsymbol{v}\|^2}{4}\boldsymbol{e} \pm \frac{\sqrt{2}}{2}\boldsymbol{e}$.
3 $\mathbf{B} \leftarrow \mathbf{B} - \dfrac{\mathbf{B}\boldsymbol{r}_+\boldsymbol{r}_+^\top\mathbf{B}}{1 + \boldsymbol{r}_+^\top\mathbf{B}\boldsymbol{r}_+}$
4 $\mathbf{B} \leftarrow \mathbf{B} + \dfrac{\mathbf{B}\boldsymbol{r}_-\boldsymbol{r}_-^\top\mathbf{B}}{1 - \boldsymbol{r}_-^\top\mathbf{B}\boldsymbol{r}_-}$
5 $\widetilde{\mathbf{U}} \leftarrow \mathbf{U} + (\boldsymbol{w} - \boldsymbol{v})\boldsymbol{e}$.
6 return $\widetilde{\mathbf{U}}, \mathbf{B}$
```

8. Equation (9.7) gives the rank-two BFGS update of the inverse Hessian $\mathbf{C}_t$ to $\mathbf{C}_{t+1}$. Instead, of using a two-rank update, we can consider a one-rank update, in which $\mathbf{C}_t$ is updated to $\mathbf{C}_{t+1}$ by the general rank-one formula:

$$\mathbf{C}_{t+1} = \mathbf{C}_t + \upsilon_t\, \boldsymbol{r}_t \boldsymbol{r}_t^\top$$

Find values for the scalar υ_t and vector $\boldsymbol{r}_t$, such that $\mathbf{C}_{t+1}$ satisfies the secant condition $\mathbf{C}_{t+1}\boldsymbol{g}_t = \boldsymbol{\delta}_t$.

Solution:

9. Show that the *BFGS formula* (B.23) can be written as:

$$\mathbf{C} \leftarrow \left(\mathbf{I} - \upsilon \boldsymbol{g}\boldsymbol{\delta}^\top\right)^\top \mathbf{C} \left(\mathbf{I} - \upsilon \boldsymbol{g}\boldsymbol{\delta}^\top\right) + \upsilon \boldsymbol{\delta}\boldsymbol{\delta}^\top,$$

where $\upsilon := (\boldsymbol{g}^\top\boldsymbol{\delta})^{-1}$.

Solution: Proceeding with a direct expansion, we obtain:

$$\begin{aligned}
\left(\mathbf{I} - \upsilon \boldsymbol{g}\boldsymbol{\delta}^\top\right)^\top \mathbf{C} \left(\mathbf{I} - \upsilon \boldsymbol{g}\boldsymbol{\delta}^\top\right) + \upsilon \boldsymbol{\delta}\boldsymbol{\delta}^\top &= \left(\mathbf{I} - \upsilon \boldsymbol{g}\boldsymbol{\delta}^\top\right)^\top \left(\mathbf{C} - \upsilon \mathbf{C}\boldsymbol{g}\boldsymbol{\delta}^\top\right) + \upsilon \boldsymbol{\delta}\boldsymbol{\delta}^\top \\
&= \mathbf{C} - \upsilon \mathbf{C}\boldsymbol{g}\boldsymbol{\delta}^\top - \left(\upsilon \boldsymbol{\delta}\boldsymbol{g}^\top\mathbf{C} - \upsilon^2 \boldsymbol{\delta}\boldsymbol{g}^\top\mathbf{C}\boldsymbol{g}\boldsymbol{\delta}^\top\right) + \upsilon \boldsymbol{\delta}\boldsymbol{\delta}^\top \\
&= \mathbf{C} + \left(\upsilon + \upsilon^2 \boldsymbol{g}^\top\mathbf{C}\boldsymbol{g}\right)\boldsymbol{\delta}\boldsymbol{\delta}^\top - \upsilon\left(\mathbf{C}\boldsymbol{g}\boldsymbol{\delta}^\top + \boldsymbol{\delta}\boldsymbol{g}^\top\mathbf{C}\right),
\end{aligned}$$

which is in agreement with (B.23).

10. Show that the *BFGS formula* (B.23) is the solution to the constrained optimization problem:

$$\mathbf{C}_{\text{BFGS}} = \underset{\mathbf{A} \text{ subject to } \mathbf{A}\boldsymbol{g} = \boldsymbol{\delta},\, \mathbf{A} = \mathbf{A}^\top}{\operatorname{argmin}} \mathcal{D}(\mathbf{0}, \mathbf{C} \,|\, \mathbf{0}, \mathbf{A}),$$

where $\mathcal{D}$ is the Kullback-Leibler distance defined in (B.22). On the other hand, show that the *DPF formula* (B.24) is the solution to the constrained optimization problem:

$$\mathbf{C}_{\text{DFP}} = \underset{\mathbf{A} \text{ subject to } \mathbf{A}\boldsymbol{g}=\boldsymbol{\delta},\ \mathbf{A}=\mathbf{A}^\top}{\operatorname{argmin}} \mathcal{D}(\mathbf{0}, \mathbf{A} \,|\, \mathbf{0}, \mathbf{C}).$$

Solution:

11. Consider again the multi-logit regression model in Exercise 5.18, which used the *iterative reweighted least squares* for training the learner. Repeat all the computations, but this time using the *limited-memory BFGS* Algorithm 9.4.4. Which training algorithm converges faster to the optimal solution?

Solution: We used the following Python code to implement both methods.

```
import numpy as np; from numpy.linalg import solve
from numpy import sqrt; from numpy import random

np.random.seed(123)
n=2000;p=4
X = random.randn(n*p).reshape(n,p)
beta1=np.array([1,0,-4,0]).reshape(p,1)
y=1/(1+np.exp(-X@beta1));
y=y<random.rand(n).reshape(n,1); y = y*1 # simulated data
#%% IRLS
def IRLS(X,y,b):
    n,p=X.shape
    err=1; count=1;
    while err >10**-6:
     mu=1/(1+np.exp(-X@b))
     D=np.diag((mu*(1-mu)).reshape(-1))
     yt=X@b+np.linalg.inv(D)@(y-mu)
     b_new=solve(X.T@D@X,(D@X).T@yt)
     err=np.linalg.norm(b-b_new)
     b=b_new
     count=count+1
    return b, count

#%% training loss
def f(theta,X,y):
    n,p=X.shape
    mu=1/(1+np.exp(-X@theta))
    loss=(1-y)*(X@theta)+np.log(1+np.exp(-X@theta))
    loss=np.array([np.sum(loss)/n]) # loss
    u=X.T@(mu-y)/n # gradient
    return u, loss

#%% LM-BFGS
def LmBfgs(u,g,dell):
    t = len(dell); q=u
    upsilon=np.zeros((t,1)); tau=np.zeros((t,1))

    for i in range(t-1, -1, -1):
        upsilon[i]=dell[i].T @ g[i]
        upsilon[i]=1/upsilon[i]
        tau[i]=dell[i].T @ q
        q=q-upsilon[i]*tau[i]*g[i]

    for i in range(t):
        q=q+upsilon[i]*(tau[i]-(g[i].T @ q))  * dell[i]

    return q
#%% BFGS
def Bfgs(f,beta,X,y):
    x = beta; n = len(x)
    h = 15 # length of LM-BFGS history
    delHist=[]; gHist=[] # BGFS history of updates
    x_best=x; f_best=np.inf
    grad=np.zeros((n,1)); dell=np.ones((n,1))*.1
    print("iteration : loss")
    for t in range(140):
        grad_new, fval = f(x,X,y)

        if (fval<f_best):
            f_best=fval
            x_best=x

        g=grad_new-grad; gHist.append(g)
        delHist.append(dell)

        if (len(gHist)>h):
           gHist.pop(0); delHist.pop(0)

        d = -LmBfgs(grad_new,gHist,delHist)

        alpha=10; u_t, loss = f(x+alpha*d,X,y)
        tmp = (fval+10**-4*(alpha*d.T @ grad_new))[0][0]
        while (loss>tmp):
            u_t, loss = f(x+alpha*d,X,y)
            tmp = (fval+10**-4*alpha*d.T @ grad_new)[0][0]
            alpha=alpha/1.5;

        dell=alpha*d; xnew=x+dell; x=xnew; grad=grad_new
        err=np.linalg.norm(grad)
        print(t,": ",fval)
        if (err<10**-6 or np.isnan(fval)):
            break

    return x_best
bo=random.randn(p).reshape(p,1)/10 # inital random guess
b_IRLS,count=IRLS(X,y,bo)
opt_beta = Bfgs(f,bo,X,y)
```

After running the code above, the iterative reweighted least squares converged to the solution after 9 iterations, whereas the limited-memory BFGS converged to the solution in 17 iterations.

12. Download the `seeds_dataset.txt` data set from the book's GitHub site, which

contains 210 independent examples. The categorical output (response) here is the type of wheat grain: Kama, Rosa, and Canadian (encoded as 1, 2, and 3), so that $c = 3$. The seven continuous features (explanatory variables) are measurements of the geometrical properties of the grain (area, perimeter, compactness, length, width, asymmetry coefficient, and length of kernel groove). Thus, $\boldsymbol{x} \in \mathbb{R}^7$ (which does not include the constant feature 1) and the multi-logit pre-classifier in Example 9.2 can be written as $\boldsymbol{g}(\boldsymbol{x}) = \text{softmax}(\mathbf{W}\boldsymbol{x} + \boldsymbol{b})$, where $\mathbf{W} \in \mathbb{R}^{3\times 7}$ and $\boldsymbol{b} \in \mathbb{R}^3$. Implement and train this pre-classifier on the first $n = 105$ examples of the seeds data set using, for example, Algorithm 9.4.1. Use the remaining $n' = 105$ examples in the data set to estimate the generalization risk of the learner using the cross-entropy loss. [Hint: Use the cross-entropy loss formulas from Example 9.4.]

Solution:

13. In Exercise 12 above, we train the multi-logit regression model whilst keeping z_1 to be an arbitrary constant (say $z_1 = 0$). This has the effect of removing a node from the output layer of the network, giving a weight matrix $\mathbf{W} \in \mathbb{R}^{(c-1)\times(p-1)}$ and bias vector $\boldsymbol{b} \in \mathbb{R}^{c-1}$ of smaller dimensions ($c - 1$, instead of c). Repeat the training of the multi-logit regression model, but this time using a weight matrix $\mathbf{W} \in \mathbb{R}^{c\times(p-1)}$ and bias vector $\boldsymbol{b} \in \mathbb{R}^c$.

Solution:

We modify the softmax and the loss function code in the previous Exercise to obtain the following.

```
def SOFTMAX2(X):
    # pad X with zero
    X_tmp = np.array([0,X[0][0],X[1][0]])
    exps = np.exp(X_tmp - np.max(X_tmp))
    exps = exps / np.sum(exps)
    return np.array([exps[1],exps[2]]).reshape(-1,1), None

#%%
def loss_fn2(y,g):
    y_truth = np.zeros(len(g))
    if(y!=0):
        y_truth[y-1] = 1
    return -y_truth @ np.log1p(g)
```

Next, we only modify the definition of the network, with the rest of the code still unchanged.

```
# load data
np.random.seed(1234)
df = pd.DataFrame(np.loadtxt("seeds_dataset.txt"))
df.columns = ["x1","x2","x3","x4","x5","x6","x7","y"]
y = np.array(df.y.values.reshape(-1,1),dtype=int)
y = y-1
df.drop(columns=["y"], inplace = True)

# split train test
```

```
X_train, X_test, y_train, y_test = train_test_split(
    df, y, test_size=0.5, random_state=42)


X = X_train.values; y = y_train

# define network
p = [X.shape[1],100,2] # size of layers
L = len(p)-1             # number of layers
W,b = initialize(p) # initialize weight matrices and bias
   vectors


theta_t = list2vec(W,b)
SteepestDescent(f,p,theta_t,X,y)

# calculate the e generalization risk
X = X_test.values; y = y_test
ut,loss = f(theta_t,p,X,y)
print("generalization risk = ",loss)
```

```
iteration : loss
0 :  [-0.21124485]  alpha= 0.1
100 :  [-0.43264695]  alpha= 0.0012060439128182574
200 :  [-0.43528644]  alpha= 6.824379332841874e-05
300 :  [-0.43764303]  alpha= 0.0011627218236705203
400 :  [-0.43808258]  alpha= 0.00027624515706391515
500 :  [-0.43822065]  alpha= 0.00026509289170559925
600 :  [-0.43834471]  alpha= 0.006971026764143862
700 :  [-0.43871281]  alpha= 0.0021657789142644265
800 :  [-0.43884083]  alpha= 0.0013479918278602348
900 :  [-0.43899964]  alpha= 0.00018690830239791452
generalization risk =  [-0.25085326]
```

Thus, in this instance, removing an ostensibly redundant node from the network did not improve the generalization risk.

14. Consider again Example 9.4, where we used a *softmax* output function $\boldsymbol{S}_L$ in conjunction with the *cross-entropy* loss: $C(\boldsymbol{\theta}) = -\ln g_{y+1}(\boldsymbol{x} \,|\, \boldsymbol{\theta})$. Find formulas for $\frac{\partial C}{\partial \boldsymbol{g}}$ and $\frac{\partial \boldsymbol{S}_L}{\partial \boldsymbol{z}_L}$. Hence, verify that:

$$\frac{\partial \boldsymbol{S}_L}{\partial \boldsymbol{z}_L} \frac{\partial C}{\partial \boldsymbol{g}} = \boldsymbol{g}(\boldsymbol{x} \,|\, \boldsymbol{\theta}) - \boldsymbol{e}_{y+1},$$

where $\boldsymbol{e}_i$ is the unit length vector with an entry of 1 in the i-th position.

Solution:

15. Derive the formula (B.25) for a diagonal Hessian update in a quasi-Newton method for minimization. In other words, given a current minimizer $\boldsymbol{x}_t$ of $f(\boldsymbol{x})$, a diagonal matrix $\mathbf{C}$ of approximating the Hessian of f, and a gradient vector $\boldsymbol{u} = \nabla f(\boldsymbol{x}_t)$, find

the solution to the constrained optimization program:

$$\min_{\mathbf{A}} \mathcal{D}(\boldsymbol{x}_t, \mathbf{C} \,|\, \boldsymbol{x}_t - \mathbf{A}\boldsymbol{u}, \mathbf{A})$$
$$\text{subject to: } \mathbf{A}\boldsymbol{g} \geqslant \boldsymbol{\delta}, \; \mathbf{A} \text{ is diagonal},$$

where $\mathcal{D}$ is the Kullback-Leibler distance defined in **(B.22)** (see Exercise **4**).

Solution: We first substitute and simplify:

$$\begin{aligned} 2\mathcal{D}(\boldsymbol{x}_t, \mathbf{C} \,|\, \boldsymbol{x}_t - \mathbf{A}\boldsymbol{u}, \mathbf{A}) &= \operatorname{tr}(\mathbf{C}\mathbf{A}^{-1}) - \ln|\mathbf{A}^{-1}\mathbf{C}| + \boldsymbol{u}^\top \mathbf{A}\mathbf{A}^{-1}\mathbf{A}\boldsymbol{u} - d \\ &= \operatorname{tr}(\mathbf{C}\mathbf{A}^{-1}) + \ln|\mathbf{A}| + \boldsymbol{u}^\top \mathbf{A}\boldsymbol{u} + \text{const.} \end{aligned}$$

Assuming that **A** and **C** are diagonal, we then obtain:

$$2\mathcal{D}(\boldsymbol{x}_t, \mathbf{C} \,|\, \boldsymbol{x}_t - \mathbf{A}\boldsymbol{u}, \mathbf{A}) = \sum_j \frac{c_j}{a_j} + \sum_j \ln(a_j) + \sum_j u_j^2 a_j + \text{const.}$$

If the inequality constraint is active for some k', then $a_{k'} = \delta_{k'}/g_{k'}$. Alternatively, if the inequality constraints are inactive for some $k \in \mathcal{A}$, then the solution is obtained by differentiating $2\mathcal{D}(\boldsymbol{x}_t, \mathbf{C} \,|\, \boldsymbol{x}_t - \mathbf{A}\boldsymbol{u}, \mathbf{A})$ with respect to $a_k, k \in \mathcal{A}$ and setting the result to zero:

$$-\frac{c_k}{a_k^2} + \frac{1}{a_k} + u_k^2 = 0, \quad k \in \mathcal{A}.$$

Therefore, we have the quadratic $u_k^2 a_k^2 + a_k - c_k = 0$ with solution

$$a_k = \frac{-1 + \sqrt{1 + 4c_k u_k^2}}{2u_k^2} = \frac{2c_k}{1 + \sqrt{1 + 4c_k u_k^2}}, \quad k \in \mathcal{A}.$$

Hence, the formula **(B.25)**.

16. Consider again the Python implementation of the polynomial regression in Section **9.5.1**, where the *stochastic gradient descent* was used for training.

    Using the polynomial regression dataset, implement and run the following four alternative training methods listed below. Note that the following code implements the neural network and will be required for running all the training methods.

```python
import numpy as np; import matplotlib.pyplot as plt

def initialize(p, w_sig = 1):
    W, b = [[]]*len(p), [[]]*len(p)

    for l in range(1,len(p)):
       W[l]= w_sig * np.random.randn(p[l], p[l-1])
       b[l]= w_sig * np.random.randn(p[l], 1)
    return W,b

def list2vec(W,b):
    # converts list of weight matrices and bias vectors into
```

```
        one column vector
    b_stack = np.vstack([b[i] for i in range(1,len(b))] )
    W_stack = np.vstack(W[i].flatten().reshape(-1,1) for i in
        range(1,len(W)))
    vec = np.vstack([b_stack, W_stack])
    return vec
#%%
def vec2list(vec, p):
    # converts vector to weight matrices and bias vectors
    W, b = [[]]*len(p),[[]]*len(p)
    p_count = 0

    for l in range(1,len(p)): # construct bias vectors
        b[l] = vec[p_count:(p_count+p[l])].reshape(-1,1)
        p_count = p_count + p[l]

    for l in range(1,len(p)): # construct weight matrices
        W[l] = vec[p_count:(p_count + p[l]*p[l-1])].reshape((p[
            l], p[l-1]))
        p_count = p_count + (p[l]*p[l-1])

    return W, b
#%%
def RELU(z,l):
    # RELU activation function: value and derivative
    if l == L: return z, np.ones_like(z) # if last layer return
         identity
    else:
        val = np.maximum(0,z) # RELU function element-wise
        J = np.array(z>0, dtype = float) # derivative of RELU
            element-wise
        return val, J

#%%
def loss_fn(y,g):
    return (g - y)**2, 2 * (g - y)

#%%
def feedforward(x,W,b):
    a, z, gr_S = [0]*(L+1), [0]*(L+1), [0]*(L+1)

    a[0] = x.reshape(-1,1)
    for l in range(1,L+1):
        z[l] = W[l] @ a[l-1] + b[l] # affine transformation
        a[l], gr_S[l] = S(z[l],l) # activation function
    return a, z, gr_S

#%%
def backward(W,b,X,y):
    n = len(y)
    delta = [0]*(L+1)
    dC_db, dC_dW = [0]*(L+1), [0]*(L+1)
    loss=0
    for i in range(n): # loop over training examples
        if(len(X)==1):
            a, z, gr_S = feedforward(X[i].T, W, b)
        else:
            a, z, gr_S = feedforward(X[i,:].T, W, b)
        cost, gr_C = loss_fn(y[i], a[L]) # cost i and gradient
            wrt g
        loss += cost/n

        delta[L] = gr_S[L] @ gr_C

        for l in range(L,0,-1): # l = L,...,1
            dCi_dbl = delta[l]
            dCi_dWl = delta[l] @  a[l-1].T

            # ---- sum up over samples ----
            dC_db[l] = dC_db[l] + dCi_dbl/n
            dC_dW[l] = dC_dW[l] + dCi_dWl/n
            # -----------------------------

            delta[l-1] =  gr_S[l-1] * W[l].T @ delta[l]

    return dC_dW, dC_db, loss, a
#-------------------------------------------
def f(theta,p,X,y):

    W,b = vec2list(theta,p)
    dC_dW, dC_db, loss, a = backward(W,b,X,y)
    u_t = list2vec(dC_dW,dC_db)
    return u_t, loss
```

(a) Implement the steepest-descent Algorithm 9.4.1;

Solution:

(b) Implement the Levenberg–Marquardt Algorithm B.3.3, in conjunction with Algorithm 9.4.2 for computing the Jacobian matrix;

Solution:

(c) Implement the *limited-memory BFGS* Algorithm 9.4.4;

Solution:

(d) Implement the *ADAM* Algorithm 9.4.5, which uses past gradient values to determine the next search direction.

Solution:

For each training algorithm, using trial and error, tune any algorithmic parameters so that the network training is as fast as possible. Comment on the relative advantages and disadvantages of each training/optimization method. For example, comment on which optimization method makes rapid initial progress, but gets trapped in a suboptimal solution, and which method is slower, but more consistent in finding good optima.

Solution:

17. Consider again the `Pytorch` code in Section 9.5.2. Repeat all the computations, but this time using the *momentum* method for training of the network. Comment on which method is preferable: the *momentum* or the *Adam* method?

Solution:

The following figure with the evolution of the loss values was obtained using the Python code below.

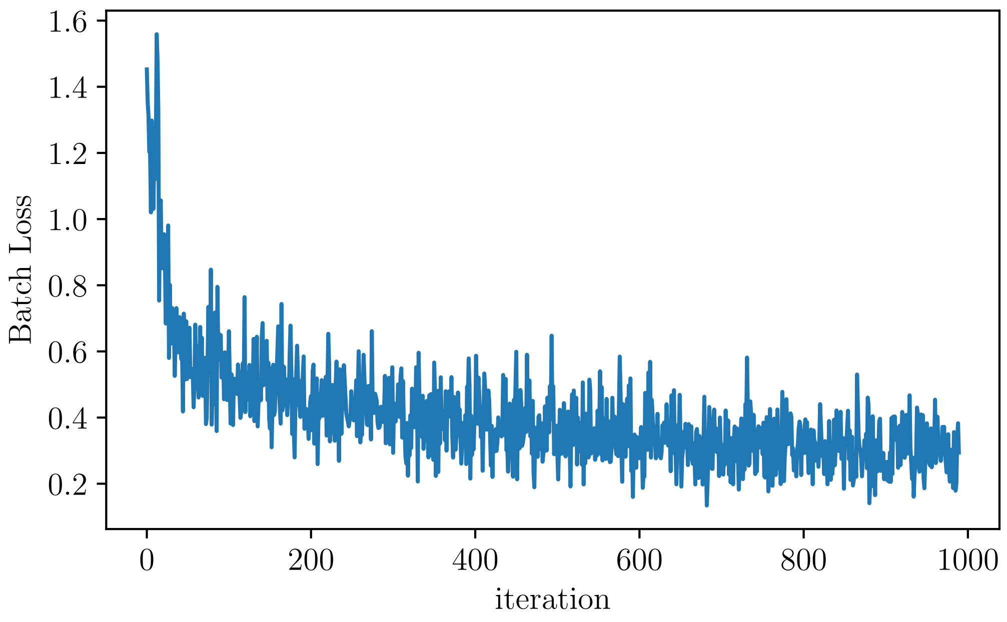


Figure 9.1: The batch loss history using momentum method.

The main change in the Python code is the change of the options in the training method:

```
optimizer = torch.optim.SGD(cnn.parameters(), lr=learning_rate,
    momentum=0.9)
```

Running the Python code with the above modification, we obtained comparable results with the following output.

```
Epoch :  1 , Training Loss:  0.5184073448181152
Epoch :  10 , Training Loss:  0.10343176126480103
Epoch :  20 , Training Loss:  0.008296280167996883
Epoch :  30 , Training Loss:  9.205818059854211e-05
Epoch :  40 , Training Loss:  4.760742012877017e-05
Epoch :  50 , Training Loss:  9.422302355233114e-06
Test Accuracy of the model on 10000 training test images: 91.64 %
```