Lab 4 : Linear Algebra and Probability

Author	Jeongyeong Park
Module	INT104
Professor	Xiaobo Jin
Date	1st/April/2021

Introduction

The probability problem is to find the likelihood some event will happen. In order to compute probabilities of values in csv file, csv file is necessary to be parsed into a matrix. After that, each instance in matrix should be checked if it is under the particular condition required for probability. Also, the number of instances that are in same situation should be counted and recorded to calculate probabilities. In this project, there are many things to be counted and recorded, if all the cases are considered respectively, then code would be inefficient and complex. Therefore, it needs to be designed with various type of arrays(linear array and two-dimensional array) to reduce unnecessary repetition.

Design & Implementation

1. Design

   First of all, csv module and numpy module are imported: csv module to work with csv(comma-separated value) files and numpy module to work with arrays in python.

   1) Design for reading csv file and parsing its content into a matrix(two-dimensional array)

      First, csv file is read and instances in csv file are saved in list. Since all instances in the given matrix are either 0 or 1, which are integers, when list is converted to 2D array, the type of instances are changed into integer.

   2) Design for computing prior probabilities p(l =0) and p(l =1)

      For prior probabilities: p(l =0) and p(l =0), $\mathrm{p}(\mathrm{l}=0)=\frac{\#\{l=0\}}{Total\ number\ of\ the\ instances}$ and $\mathrm{p}(\mathrm{l}=1)=\frac{\#\{l=1\}}{Total\ number\ of\ the\ instances}$ are used. #{ l =0} represents the total number of the labels of the instances having value as 0(l =0) and #{l =1} represents the total number of the labels of the instances having value as 1(l =1). Linear array is used to count the number of cases under each condition and save final total number. Index 0 of it will store case for l=0 and index 1 of it will store case for l=1. In order to trace if l has a value of 0 or 1 in matrix, matrix[i,5] is checked since the label of the instance which is l located in column 5 in each row. i represents row and it goes from 0 to 99. Different linear array is also used to save prior probabilities. When prior probabilities are calculated with the formula mentioned earlier, the total number of the instances is 100 because given matrix has 100 rows. Index 0 of this linear array will store value of p(l=0) and index 1 of it will store value of p(l=1).

3) Design for computing conditional probabilities

For conditional probabilities: $p(a_i = 0|l = 0), i = 0,1,2,3,4$ and $p(a_i = 1|l = 0), i = 0,1,2,3,4$ and $p(a_i = 0|l = 1), i = 0,1,2,3,4$ and $p(a_i = 1|l = 1), i = 0,1,2,3,4,$ $p(a_i = x|l = y)$=$\frac{\#\{a_i=x \ and \ l=y\}}{\#\{l=y\}}$, i=0,1,2,3,4 is used: x and y will be either 0 or 1 in this lab. For example, $p(a_i = 0|l = 0)$=$\frac{\#\{a_i=0 \ and \ l=0\}}{\#\{l=0\}}$, i=0,1,2,3,4 is calculated in order to compute $p(a_i = 0|l = 0), i = 0,1,2,3,4$. $\#\{a_i = 0 \ and \ l = 0\}$ represents the total number of cases that correspond to attributes of the instances having value as 0($a_i = 0$) and labels of the instances having value as 0($l = 0$). As mentioned in design 2, #{ l =0} represents the total number of the labels of the instances having value as 0(l =0). In order to record the total number of 20 cases which have to be considered for conditional probabilities, 5x4 two-dimensional array is used. Each row is for recording each attribute: a0, a1, a2, a3, a4. In each row, column index 0 and 2 are for the case a_i has value of 0, and column index 1 and 3 are for the case a_i has value of 1. Moreover, column index 0 and 1 are for the case $l = 0$, and column index 2 and 3 are for the case $l = 1$.

count[0,0] $a_0 = 0, l = 0$	count[0,1] $a_0 = 1, l = 0$	count[0,2] $a_0 = 0, l = 1$	count[0,3] $a_0 = 1, l = 1$
count[1,0] $a_1 = 0, l = 0$	count[1,1] $a_1 = 1, l = 0$	count[1,2] $a_1 = 0, l = 1$	count[1,3] $a_1 = 1, l = 1$
count[2,0] $a_2 = 0, l = 0$	count[2,1] $a_2 = 1, l = 0$	count[2,2] $a_2 = 0, l = 1$	count[2,3] $a_2 = 1, l = 1$
count[3,0] $a_3 = 0, l = 0$	count[3,1] $a_3 = 1, l = 0$	count[3,2] $a_3 = 0, l = 1$	count[3,3] $a_3 = 1, l = 1$
count[4,0] $a_4 = 0, l = 0$	count[4,1] $a_4 = 1, l = 0$	count[4,2] $a_4 = 0, l = 1$	count[4,3] $a_4 = 1, l = 1$

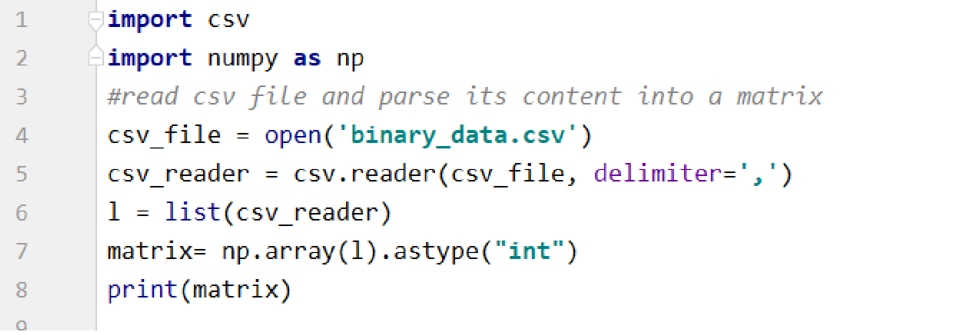
Figure1. Precise view of 5x4 matrix for counting

Depending on the value matrix[j,i] and matrix[j,5] have, each case of conditional probability would be considered. i will go from 0 to 4 since there are four attributes a_0, a_1, a_2, a_3, a_4 have to be considered. j will go from 0 to 99 since the given matrix has 100 rows. For the value of the label of the instances, matrix[j,5] will be used since the label of the instances located in column 5 in the given matrix.

When the conditional probabilities are computed with the formula mentioned above, denominator would be the value from the linear array that is used to save the total number of cases (l=0 and l=1) in design 2.

2. Implementation

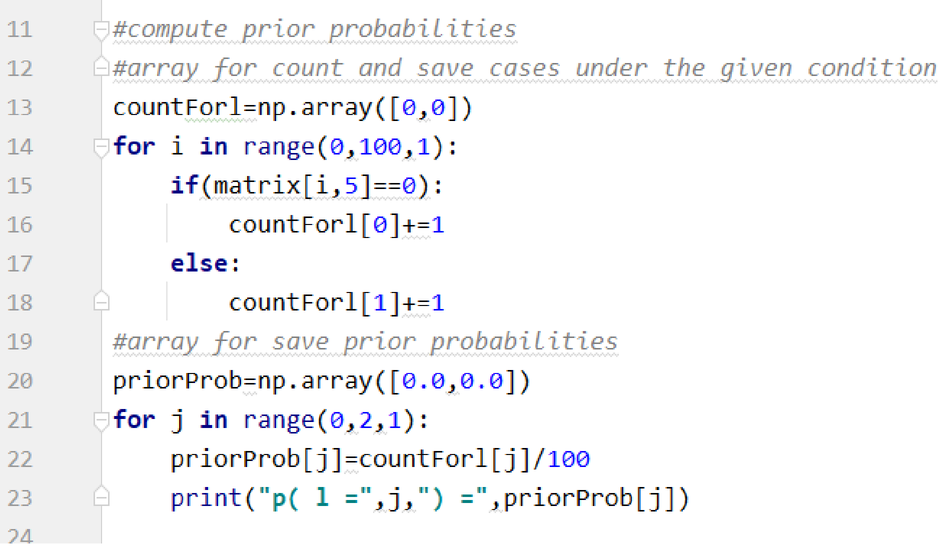
1) Code for read csv file and parse its content into a two-dimensional array(matrix)



```python
import csv
import numpy as np
#read csv file and parse its content into a matrix
csv_file = open('binary_data.csv')
csv_reader = csv.reader(csv_file, delimiter=',')
l = list(csv_reader)
matrix= np.array(l).astype("int")
print(matrix)
```

Figure2. Code for read file and parse its content

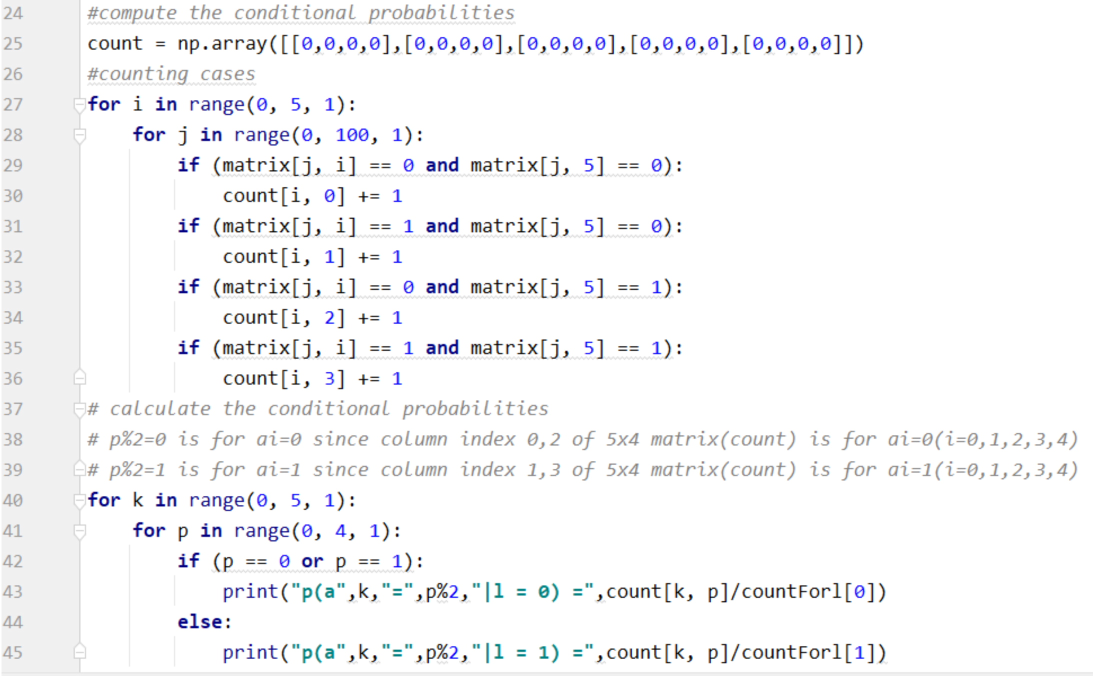
2) Code for compute prior probabilities p(l=0) and p(l=1)



```python
#compute prior probabilities
#array for count and save cases under the given condition
countForl=np.array([0,0])
for i in range(0,100,1):
    if(matrix[i,5]==0):
        countForl[0]+=1
    else:
        countForl[1]+=1
#array for save prior probabilities
priorProb=np.array([0.0,0.0])
for j in range(0,2,1):
    priorProb[j]=countForl[j]/100
    print("p( l =",j,") =",priorProb[j])
```

Figure3. Code for prior probabilities

3) Code for compute conditional probabilities

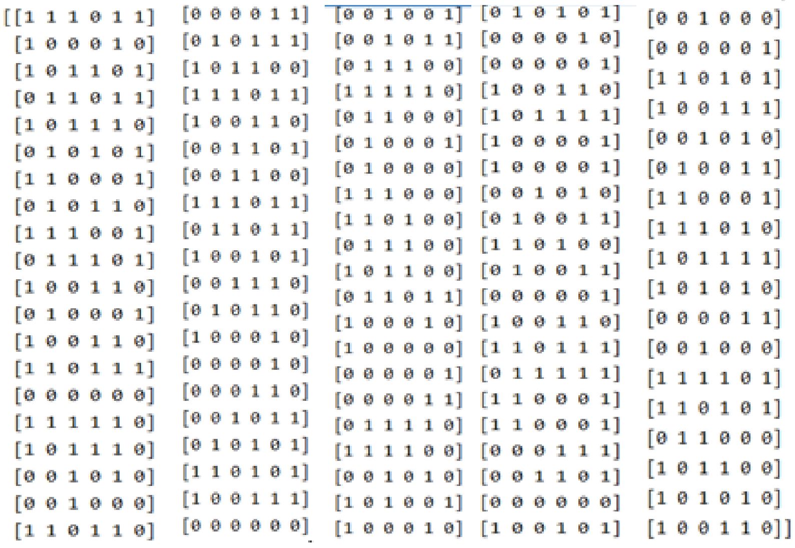


```
#compute the conditional probabilities
count = np.array([[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0]])
#counting cases
for i in range(0, 5, 1):
    for j in range(0, 100, 1):
        if (matrix[j, i] == 0 and matrix[j, 5] == 0):
            count[i, 0] += 1
        if (matrix[j, i] == 1 and matrix[j, 5] == 0):
            count[i, 1] += 1
        if (matrix[j, i] == 0 and matrix[j, 5] == 1):
            count[i, 2] += 1
        if (matrix[j, i] == 1 and matrix[j, 5] == 1):
            count[i, 3] += 1
# calculate the conditional probabilities
# p%2=0 is for ai=0 since column index 0,2 of 5x4 matrix(count) is for ai=0(i=0,1,2,3,4)
# p%2=1 is for ai=1 since column index 1,3 of 5x4 matrix(count) is for ai=1(i=0,1,2,3,4)
for k in range(0, 5, 1):
    for p in range(0, 4, 1):
        if (p == 0 or p == 1):
            print("p(a",k,"=",p%2,"|l = 0) =",count[k, p]/countForl[0])
        else:
            print("p(a",k,"=",p%2,"|l = 1) =",count[k, p]/countForl[1])
```

Figure4. Code for conditional probabilities

Test

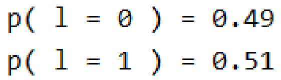
1) Read the text file and parse its content into a matrix



```
[[1 1 1 0 1 1]   [0 0 0 0 1 1]   [0 0 1 0 0 1]   [0 1 0 1 0 1]   [0 0 1 0 0 0]
 [1 0 0 0 1 0]   [0 1 0 1 1 1]   [0 0 1 0 1 1]   [0 0 0 0 1 0]   [0 0 0 0 0 1]
 [1 0 1 1 0 1]   [1 0 1 1 0 0]   [0 1 1 1 0 0]   [0 0 0 0 0 1]   [1 1 0 1 0 1]
 [0 1 1 0 1 1]   [1 1 1 0 1 1]   [1 1 1 1 1 0]   [1 0 0 1 1 0]   [1 0 0 1 1 1]
 [1 0 1 1 1 0]   [1 0 0 1 1 0]   [0 1 1 0 0 0]   [1 0 1 1 1 1]   [0 0 1 0 1 0]
 [0 1 0 1 0 1]   [0 0 1 1 0 1]   [0 1 0 0 0 1]   [1 0 0 0 0 1]   [0 1 0 0 1 1]
 [1 1 0 0 0 1]   [0 0 1 1 0 0]   [0 1 0 0 0 0]   [1 0 0 0 0 1]   [1 1 0 0 0 1]
 [0 1 0 1 1 0]   [1 1 1 0 1 1]   [1 1 1 0 0 0]   [0 0 1 0 1 0]   [1 1 1 0 1 0]
 [1 1 1 0 0 1]   [0 1 1 0 1 1]   [1 1 0 1 0 0]   [0 1 0 0 1 1]   [1 0 1 1 1 1]
 [0 1 1 1 0 1]   [1 0 0 1 0 1]   [0 1 1 1 0 0]   [1 1 0 1 0 0]   [1 0 1 0 1 0]
 [1 0 0 1 1 0]   [0 0 1 1 1 0]   [1 0 1 1 0 0]   [0 1 0 0 1 1]   [0 0 0 0 1 1]
 [0 1 0 0 0 1]   [0 1 0 1 1 0]   [0 1 1 0 1 1]   [0 0 0 0 0 1]   [0 0 1 0 0 0]
 [1 0 0 1 1 0]   [1 0 0 0 1 0]   [1 0 0 0 1 0]   [1 0 0 1 1 0]   [1 1 1 1 0 1]
 [1 1 0 1 1 1]   [0 0 0 0 1 0]   [1 0 0 0 0 0]   [1 1 0 1 1 1]   [1 1 0 1 0 1]
 [0 0 0 0 0 0]   [0 0 0 1 1 0]   [0 0 0 0 0 1]   [0 1 1 1 1 1]   [0 1 1 0 0 0]
 [1 1 1 1 1 0]   [0 0 1 0 1 1]   [0 0 0 0 1 1]   [1 1 0 0 0 1]   [1 0 1 1 0 0]
 [1 0 1 1 1 0]   [0 1 0 1 0 1]   [0 1 1 1 1 0]   [1 1 0 0 0 1]   [1 0 1 0 1 0]
 [0 0 1 0 1 0]   [1 1 0 1 0 1]   [1 1 1 1 0 0]   [0 0 0 1 1 1]   [1 0 0 1 1 0]]
 [0 0 1 0 0 0]   [1 0 0 1 1 1]   [0 0 1 0 1 0]   [0 0 1 1 0 1]
 [1 1 0 1 1 0]   [0 0 0 0 0 0]   [1 0 1 0 0 1]   [0 0 0 0 0 0]
                                 [1 0 0 0 1 0]   [1 0 0 1 0 1]
```

Figure5. The test result of matrix(Read from left to right)

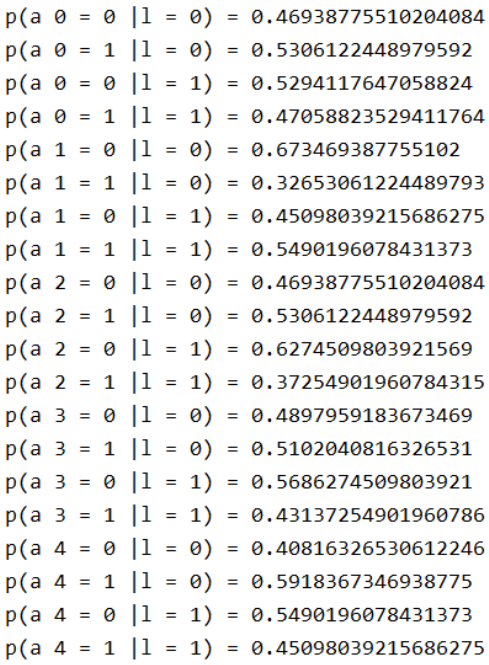
2) Compute the prior probabilities p(l =0) and p(l =1)



```
p( l = 0 ) = 0.49
p( l = 1 ) = 0.51
```

Figure6. The test result of prior probabilities

3) Compute the conditional probabilities



```
p(a 0 = 0 |l = 0) = 0.46938775510204084
p(a 0 = 1 |l = 0) = 0.5306122448979592
p(a 0 = 0 |l = 1) = 0.5294117647058824
p(a 0 = 1 |l = 1) = 0.47058823529411764
p(a 1 = 0 |l = 0) = 0.673469387755102
p(a 1 = 1 |l = 0) = 0.32653061224489793
p(a 1 = 0 |l = 1) = 0.45098039215686275
p(a 1 = 1 |l = 1) = 0.5490196078431373
p(a 2 = 0 |l = 0) = 0.46938775510204084
p(a 2 = 1 |l = 0) = 0.5306122448979592
p(a 2 = 0 |l = 1) = 0.6274509803921569
p(a 2 = 1 |l = 1) = 0.37254901960784315
p(a 3 = 0 |l = 0) = 0.4897959183673469
p(a 3 = 1 |l = 0) = 0.5102040816326531
p(a 3 = 0 |l = 1) = 0.5686274509803921
p(a 3 = 1 |l = 1) = 0.43137254901960786
p(a 4 = 0 |l = 0) = 0.40816326530612246
p(a 4 = 1 |l = 0) = 0.5918367346938775
p(a 4 = 0 |l = 1) = 0.5490196078431373
p(a 4 = 1 |l = 1) = 0.45098039215686275
```

Figure7. The text result of conditional probabilities

Conclusion

In conclusion, using arrays(linear array and 2D array) to save values required for probabilities works better than using separate variables and if statements. If individual variable and if statement which are needed to compute 22 probabilities are used, many lines with only different variables are repeated. In addition, when the probabilities are computed at the final step, since there are many variables, mistakes are more likely to occur. However, if arrays are used to store various variables that are used to compute probabilities, there is no need to repeat unnecessary statements. When the saved value in array is needed, it can be obtained through calling corresponded index. Moreover, if new matrix having different size or values of instances in csv file requires to be considered for probabilities, arrays for counting and saving numbers can be changed according to a given conditions. It is much more effective than creating new variables and classifying each case.