INDR421 – HW1
15-10-2018
Oğuzhan Akan (30223)

Approach

1- **Reading and Separating Data:** There were two .csv files within the homework zip file, data points and labels. I first changed the labels from A, B, C, D, E to 1, 2, 3, 4, 5 for a more convenient use, then binded the data points and labels into one matrix. Then, to extract the training data from the raw data, I sliced every 25 row starting from row number 1 and skipping 14 rows after 25th row, i.e. 1:25, 40:64, 79:103, 118:142, 157:181 with the following line:

```
training_data <- data_set[rep(0:(classes-1), each=take) *
(take+skip) + (1:take),]
```

where classes = 5, take = 25, and skip = 14. Here, *classes* represent the letters which are A...E in this case. *take* represents the number of rows we should take as training data and *skip* the number of rows we should skip to match row number with subsequent label.

I followed a similar approach to extract the test data as well.

```
test_data <- data_set[take + rep(0:(classes-1),each=skip) *
(take+skip) + (1:skip),]
```

Here, the difference in the code is that it starts with *take*, because first row of test data should start from row number 26 (*take+1*). There is also an interchange between *take* and *skip* in the rest of the line.

Finally, I separated the data into training and test with the following:

```
y_train <- training_data[,321]
x_train <- training_data[,1:320]
y_test <- test_data[,321]
x_test <- test_data[,1:320]
```

2- **Estimating Parameters:** Since the aim was to classify given multivariate data points, I assumed that the data points follow a Bernoulli distribution. Therefore according to the scoring function of Bernoulli density, I needed to estimate the parameters p(y=c) and pcd(x). I found the p(y=c) by counting the number of label and then dividing by the total number of data, for each label:

```
probabilities <- sapply(X = 1:classes, FUN = function(c)
{length((y_train == labels[c])[(y_train ==
labels[c])==TRUE])/length(y_train)})
```

pcd(x) is defined as “the probability of a pixel being 1 at position d for class c”, therefore I needed a nested function that calculates probability of each pixel for each position and class:

```
pcd <- sapply(X=1:5, FUN = function(c){
       sapply(X=1:320, FUN = function(x){
             sum(x_train[c:(c+24)+24*(c-1),x]) /length((y_train
             == labels[c])[(y_train == labels[c])==TRUE])})})
```

3- **Scoring Function:** Because I would need to apply the scoring function both to training and test data, I wrote a function that takes the data and pcd(x) matrix as input and returns gc(x) matrix as output.

4- **Finding y hats and building confusion matrix:** For each data point of training and test data, I extracted the column number where the data point had the highest score, which gives the estimated label for that specific data. For example:

```
train_y_hat <- sapply(X=1:nrow(x_train), FUN = function(c)
{which.max(train_data_scores[c,])})
```

5- **Printing results:** In this section, I displayed the results that are asked. I also shared the results in the next pages, in case of my r code does not work.

INDR421 – HW1
15-10-2018
Oğuzhan Akan (30223)

Results

```
> print(prior)
     labels probabilities
[1,]      1           0.2
[2,]      2           0.2
[3,]      3           0.2
[4,]      4           0.2
[5,]      5           0.2
> print(pcd[,1])
  [1] 0.00 0.00 0.00 0.04 0.04 0.04 0.16 0.20 0.16 0.12 0.12 0.24 0.20 0.28 0.36 0.44 0.48 0.56 0.52 0.40 0
.00 0.04 0.08 0.12 0.16 0.16 0.28 0.28 0.32 0.48 0.56 0.64 0.72 0.76 0.88 0.96
 [37] 1.00 0.92 0.76 0.60 0.00 0.04 0.12 0.24 0.28 0.28 0.36 0.44 0.52 0.68 0.80 0.80 0.80 0.88 1.00 1.00 0
.92 0.80 0.76 0.40 0.04 0.16 0.16 0.28 0.44 0.56 0.68 0.76 0.76 0.80 0.76 0.88
 [73] 0.92 0.92 0.80 0.72 0.68 0.56 0.32 0.20 0.08 0.24 0.36 0.36 0.56 0.64 0.60 0.68 0.68 0.64 0.76 0.80 0
.80 0.64 0.48 0.40 0.28 0.12 0.00 0.00 0.16 0.36 0.44 0.60 0.64 0.72 0.56 0.52
[109] 0.48 0.44 0.60 0.68 0.68 0.52 0.40 0.20 0.08 0.00 0.00 0.00 0.32 0.52 0.64 0.64 0.56 0.56 0.40 0.44 0
.32 0.44 0.44 0.56 0.64 0.52 0.44 0.12 0.08 0.00 0.00 0.00 0.36 0.64 0.76 0.56
[145] 0.56 0.52 0.40 0.40 0.32 0.28 0.36 0.52 0.60 0.44 0.32 0.16 0.16 0.00 0.00 0.00 0.48 0.80 0.76 0.64 0
.44 0.36 0.28 0.16 0.20 0.24 0.32 0.48 0.60 0.52 0.40 0.20 0.16 0.00 0.00 0.00
[181] 0.60 0.84 0.84 0.72 0.52 0.40 0.32 0.28 0.32 0.28 0.24 0.52 0.56 0.52 0.40 0.20 0.12 0.00 0.04 0.00 0
.56 0.76 0.84 0.76 0.60 0.48 0.40 0.40 0.44 0.36 0.40 0.68 0.60 0.56 0.44 0.32
[217] 0.12 0.08 0.08 0.08 0.48 0.68 0.64 0.64 0.56 0.52 0.56 0.52 0.52 0.48 0.64 0.76 0.68 0.52 0.48 0.48 0
.40 0.36 0.40 0.36 0.20 0.56 0.52 0.52 0.44 0.56 0.56 0.64 0.60 0.56 0.68 0.80
[253] 0.84 0.76 0.68 0.76 0.64 0.64 0.48 0.48 0.20 0.48 0.48 0.48 0.48 0.44 0.48 0.52 0.60 0.68 0.76 0.80 0
.80 0.88 0.92 1.00 0.88 0.88 0.76 0.72 0.16 0.32 0.36 0.48 0.56 0.60 0.60 0.64
[289] 0.64 0.68 0.72 0.80 0.76 0.84 0.88 0.84 0.84 0.76 0.72 0.64 0.08 0.20 0.28 0.36 0.40 0.48 0.48 0.48 0
.48 0.52 0.60 0.68 0.72 0.72 0.68 0.56 0.72 0.68 0.68 0.64
> print(pcd[,2])
  [1] 0.04 0.24 0.24 0.20 0.12 0.08 0.12 0.16 0.24 0.32 0.28 0.32 0.28 0.28 0.28 0.32 0.40 0.44 0.24 0.20 0
.12 0.36 0.48 0.36 0.40 0.52 0.56 0.52 0.60 0.68 0.72 0.68 0.68 0.76 0.76 0.72
 [37] 0.72 0.68 0.60 0.44 0.28 0.56 0.56 0.52 0.52 0.56 0.56 0.60 0.72 0.76 0.72 0.64 0.72 0.80 0.76 0.76 0
.80 0.84 0.68 0.56 0.36 0.64 0.64 0.48 0.44 0.32 0.44 0.52 0.60 0.60 0.64 0.60
 [73] 0.56 0.52 0.44 0.48 0.64 0.64 0.84 0.76 0.48 0.76 0.56 0.36 0.16 0.20 0.36 0.32 0.56 0.68 0.60 0.44 0
.36 0.24 0.16 0.24 0.28 0.40 0.88 0.88 0.48 0.84 0.68 0.28 0.16 0.16 0.36 0.40
[109] 0.68 0.76 0.52 0.40 0.28 0.12 0.08 0.08 0.20 0.44 0.88 0.96 0.48 0.84 0.68 0.24 0.08 0.16 0.28 0.40 0
.60 0.64 0.44 0.40 0.24 0.08 0.04 0.08 0.12 0.24 0.68 1.00 0.52 0.88 0.68 0.24
[145] 0.08 0.20 0.36 0.68 0.68 0.64 0.48 0.36 0.16 0.08 0.00 0.00 0.00 0.16 0.56 1.00 0.56 0.92 0.72 0.24 0
.20 0.28 0.44 0.64 0.72 0.72 0.56 0.36 0.12 0.08 0.00 0.00 0.04 0.12 0.52 0.96
[181] 0.56 0.96 0.80 0.56 0.36 0.40 0.60 0.64 0.72 0.88 0.60 0.36 0.20 0.12 0.08 0.00 0.04 0.12 0.56 0.96 0
.36 0.80 0.84 0.64 0.60 0.64 0.64 0.56 0.64 0.84 0.64 0.48 0.24 0.12 0.08 0.00
[217] 0.04 0.24 0.72 0.92 0.28 0.64 0.76 0.64 0.56 0.56 0.60 0.44 0.56 0.80 0.68 0.60 0.32 0.16 0.16 0.08 0
.08 0.36 0.64 0.88 0.20 0.44 0.48 0.44 0.48 0.48 0.48 0.36 0.44 0.36 0.68 0.64
[253] 0.52 0.40 0.24 0.16 0.20 0.52 0.80 0.76 0.12 0.40 0.44 0.28 0.36 0.44 0.24 0.20 0.20 0.28 0.52 0.76 0
.84 0.60 0.36 0.40 0.48 0.84 0.88 0.68 0.12 0.28 0.28 0.28 0.28 0.24 0.16 0.04
[289] 0.04 0.12 0.28 0.72 0.88 0.88 0.80 0.84 0.88 0.88 0.68 0.40 0.04 0.16 0.20 0.20 0.16 0.16 0.04 0.00 0
.00 0.04 0.12 0.40 0.72 0.80 0.80 0.88 0.88 0.80 0.56 0.24
> print(pcd[,3])
  [1] 0.00 0.00 0.00 0.00 0.00 0.12 0.20 0.24 0.40 0.56 0.64 0.76 0.72 0.64 0.44 0.36 0.20 0.08 0.00 0.00 0
.00 0.00 0.12 0.28 0.32 0.56 0.72 0.88 0.92 0.96 0.96 0.96 1.00 1.00 1.00 0.92
 [37] 0.80 0.48 0.20 0.08 0.00 0.20 0.28 0.44 0.60 0.80 0.88 0.92 0.88 0.80 0.76 0.76 0.76 0.88 1.00 0.96 0
.96 0.84 0.44 0.20 0.04 0.36 0.40 0.64 0.80 0.72 0.68 0.56 0.60 0.44 0.36 0.28
 [73] 0.36 0.40 0.68 0.80 0.96 0.96 0.80 0.56 0.08 0.40 0.60 0.76 0.76 0.60 0.52 0.36 0.20 0.16 0.08 0.04 0
.08 0.12 0.24 0.48 0.88 0.96 0.96 0.68 0.24 0.56 0.80 0.80 0.64 0.44 0.44 0.16
[109] 0.12 0.08 0.04 0.00 0.00 0.00 0.08 0.28 0.56 0.96 1.00 0.88 0.40 0.72 0.84 0.76 0.56 0.40 0.16 0.12 0
.04 0.04 0.00 0.00 0.00 0.00 0.00 0.04 0.40 0.92 1.00 0.96 0.44 0.84 0.92 0.68
[145] 0.56 0.28 0.12 0.04 0.04 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.16 0.68 0.96 1.00 0.52 0.88 0.88 0.60 0
.48 0.16 0.04 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.44 0.92 1.00
[181] 0.68 0.84 0.84 0.72 0.36 0.20 0.12 0.04 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.04 0.40 0.96 1.00 0
.64 0.92 0.88 0.68 0.32 0.20 0.16 0.04 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00
[217] 0.00 0.32 0.88 1.00 0.36 0.64 0.80 0.56 0.32 0.24 0.12 0.08 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0
.16 0.32 0.80 1.00 0.36 0.52 0.64 0.48 0.24 0.24 0.16 0.04 0.00 0.00 0.00 0.00
```

INDR421 – HW1
15-10-2018
Oğuzhan Akan (30223)

```
[253] 0.00 0.00 0.00 0.00 0.20 0.40 0.84 1.00 0.24 0.48 0.52 0.40 0.20 0.20 0.12 0.04 0.00 0.00 0.00 0.00 0
.00 0.00 0.00 0.04 0.20 0.48 0.88 0.96 0.16 0.40 0.44 0.36 0.20 0.12 0.08 0.00
[289] 0.00 0.00 0.00 0.00 0.00 0.00 0.04 0.12 0.24 0.56 0.84 0.84 0.16 0.28 0.32 0.28 0.20 0.16 0.08 0.00 0
.00 0.00 0.00 0.00 0.00 0.00 0.08 0.16 0.20 0.60 0.88 0.80
> print(pcd[,4])
  [1] 0.12 0.44 0.40 0.16 0.12 0.08 0.08 0.08 0.12 0.08 0.04 0.04 0.04 0.04 0.04 0.04 0.12 0.32 0.40 0.56 0
.44 0.64 0.64 0.44 0.32 0.20 0.24 0.24 0.24 0.28 0.28 0.28 0.32 0.32 0.32 0.44
 [37] 0.48 0.68 0.68 0.76 0.52 0.76 0.72 0.48 0.40 0.40 0.40 0.44 0.48 0.48 0.48 0.48 0.52 0.48 0.56 0.56 0
.60 0.64 0.72 0.80 0.48 0.80 0.76 0.60 0.48 0.32 0.28 0.40 0.40 0.40 0.40 0.48
 [73] 0.52 0.48 0.40 0.32 0.36 0.52 0.72 0.88 0.44 0.76 0.76 0.68 0.48 0.36 0.28 0.28 0.36 0.40 0.40 0.40 0
.40 0.36 0.28 0.32 0.36 0.60 0.76 0.92 0.40 0.80 0.80 0.76 0.44 0.40 0.28 0.24
[109] 0.32 0.36 0.28 0.32 0.32 0.32 0.32 0.32 0.32 0.60 0.84 0.96 0.24 0.76 0.76 0.72 0.44 0.32 0.20 0.20 0
.24 0.28 0.28 0.40 0.36 0.36 0.32 0.32 0.32 0.48 0.84 0.96 0.16 0.64 0.76 0.72
[145] 0.40 0.36 0.36 0.32 0.24 0.24 0.28 0.24 0.16 0.16 0.12 0.16 0.28 0.52 0.84 0.92 0.24 0.60 0.72 0.64 0
.52 0.44 0.36 0.28 0.28 0.28 0.24 0.24 0.12 0.12 0.04 0.08 0.20 0.48 0.80 0.92
[181] 0.24 0.60 0.72 0.72 0.56 0.36 0.40 0.36 0.28 0.24 0.08 0.08 0.08 0.12 0.16 0.20 0.24 0.64 0.80 0.84 0
.16 0.52 0.68 0.72 0.64 0.48 0.40 0.36 0.20 0.08 0.04 0.04 0.12 0.12 0.20 0.20
[217] 0.48 0.64 0.84 0.72 0.08 0.32 0.48 0.64 0.56 0.60 0.48 0.24 0.20 0.08 0.08 0.08 0.08 0.08 0.24 0.44 0
.60 0.76 0.76 0.64 0.08 0.24 0.44 0.60 0.64 0.60 0.48 0.40 0.24 0.24 0.20 0.16
[253] 0.20 0.20 0.36 0.52 0.64 0.76 0.64 0.40 0.08 0.20 0.28 0.36 0.60 0.72 0.68 0.60 0.56 0.48 0.44 0.44 0
.44 0.52 0.56 0.64 0.68 0.64 0.48 0.24 0.04 0.16 0.20 0.28 0.48 0.68 0.76 0.80
[289] 0.72 0.72 0.68 0.72 0.80 0.80 0.84 0.80 0.72 0.60 0.24 0.16 0.00 0.04 0.16 0.24 0.28 0.40 0.56 0.84 0
.88 0.92 0.96 0.88 0.80 0.84 0.80 0.64 0.56 0.28 0.16 0.08
> print(pcd[,5])
  [1] 0.00 0.12 0.12 0.08 0.12 0.16 0.12 0.04 0.12 0.12 0.20 0.20 0.24 0.24 0.32 0.52 0.52 0.56 0.56 0.32 0
.00 0.16 0.24 0.28 0.32 0.40 0.36 0.44 0.48 0.64 0.60 0.72 0.68 0.80 0.84 0.92
 [37] 0.92 0.96 0.84 0.68 0.04 0.24 0.36 0.40 0.48 0.48 0.44 0.56 0.56 0.64 0.72 0.80 0.80 0.80 0.76 0.80 0
.80 0.96 0.88 0.80 0.04 0.40 0.48 0.56 0.60 0.48 0.60 0.60 0.56 0.60 0.80 0.84
 [73] 0.88 0.64 0.52 0.44 0.72 0.84 0.96 0.80 0.04 0.48 0.52 0.60 0.56 0.48 0.48 0.52 0.52 0.72 0.76 0.88 0
.68 0.40 0.28 0.16 0.40 0.80 0.92 0.88 0.04 0.52 0.68 0.72 0.52 0.36 0.44 0.44
[109] 0.40 0.68 0.88 0.84 0.48 0.28 0.04 0.08 0.24 0.80 0.92 0.88 0.16 0.60 0.80 0.84 0.64 0.48 0.28 0.28 0
.28 0.68 0.76 0.80 0.40 0.12 0.00 0.04 0.32 0.68 0.96 0.88 0.28 0.76 0.88 0.80
[145] 0.56 0.32 0.12 0.16 0.28 0.76 0.76 0.84 0.28 0.08 0.04 0.12 0.28 0.72 0.96 0.88 0.28 0.88 0.96 0.64 0
.32 0.16 0.04 0.04 0.36 0.76 0.76 0.72 0.32 0.08 0.04 0.12 0.32 0.68 1.00 0.88
[181] 0.28 1.00 1.00 0.60 0.28 0.12 0.00 0.08 0.36 0.80 0.88 0.72 0.32 0.12 0.04 0.04 0.28 0.64 0.96 0.92 0
.40 1.00 1.00 0.60 0.16 0.12 0.00 0.16 0.36 0.80 0.80 0.76 0.28 0.12 0.04 0.04
[217] 0.32 0.64 0.92 0.92 0.60 1.00 1.00 0.48 0.24 0.00 0.00 0.20 0.40 0.80 0.72 0.60 0.28 0.04 0.00 0.04 0
.36 0.64 0.92 0.92 0.72 1.00 0.96 0.44 0.24 0.00 0.00 0.16 0.44 0.80 0.76 0.52
[253] 0.24 0.04 0.00 0.08 0.32 0.56 0.84 0.80 0.68 0.96 0.92 0.48 0.20 0.04 0.04 0.12 0.36 0.68 0.48 0.36 0
.20 0.04 0.00 0.12 0.32 0.56 0.76 0.64 0.64 0.88 0.76 0.28 0.08 0.00 0.04 0.08
[289] 0.24 0.40 0.32 0.28 0.12 0.04 0.00 0.08 0.32 0.44 0.52 0.52 0.60 0.84 0.64 0.28 0.08 0.00 0.04 0.08 0
.20 0.24 0.24 0.20 0.08 0.04 0.00 0.04 0.28 0.32 0.48 0.44
> print(train_confusion_matrix)
           y_train
y_hat_train  1  2  3  4  5
          1 25  0  0  0  0
          2  0 24  1  0  1
          3  0  0 24  0  0
          4  0  1  0 25  0
          5  0  0  0  0 24
> print(test_confusion_matrix)
          y_test
y_hat_test  1  2  3  4  5
         1  7  0  0  0  0
         2  0 11  3  2  4
         3  0  0  7  0  0
         4  7  3  3 12  0
         5  0  0  1  0 10
```