CS 464
Introduction to Machine Learning

Homework 1

Ufuk Girginkaya
21901886

07/11/2023

Contents

Probability Review 3
Question 1.1 3
Question 1.2 3
Question 1.3 4
MLE and MAP 4
Question 2.1 4
Question 2.2 4
Question 2.3 5
BBC News Classification 5
Question 3.1 5
Question 3.1.1 5
Question 3.1.2 6
Question 3.1.3 6
Question 3.1.4 6
Question 3.2 7
Question 3.3 7
Question 3.4 8
Appendices 9
Appendix A 9

Probability Review

Question 1.1

Let's define events A, B, C and D.

A: Choose yellow coin from Box 1, get two heads in a row.

B: Choose blue coin from Box 1, get two heads in a row.

C: Choose blue coin from Box 2, get two heads in a row.

D: Choose red coin from Box 2, get two heads in a row.

$$P(A) = \frac{1}{2} \times \frac{1}{3} \times \frac{1}{4} \times \frac{1}{4}$$

$$P(B) = \frac{1}{2} \times \frac{2}{3} \times \frac{1}{2} \times \frac{1}{2}$$

$$P(C) = \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2} \times \frac{1}{2}$$

$$P(D) = \frac{1}{2} \times \frac{1}{2} \times \frac{1}{10} \times \frac{1}{10}$$

The probability of getting two heads in a row:

$$P(A) + P(B) + P(C) + P(D) = 0.15875$$

Question 1.2

Since the blue coin is the only fair coin, the probability is:

$$P(Coin = Blue|2H) = \frac{P(Coin = Blue, 2H)}{P(2H)} = \frac{P(B) + P(C)}{P(A) + P(B) + P(C) + P(D)} = 0.91864$$

Question 1.3

If we have two heads in a row, the probability of selected coin was red is:

$$P(Coin = Red|2H) = \frac{P(Coin = Red, 2H)}{P(2H)} = \frac{P(D)}{P(A) + P(B) + P(C) + P(D)} = 0.01574$$

MLE and MAP

Question 2.1

MLE for the μ using the given data points is:

$$D = \{x_1, x_2, \dots, x_N\}$$

$$P(D|\mu, \sigma) = \left(\frac{1}{\sigma\sqrt{2\pi}}\right)^N \prod_{i=1}^{N} e^{\frac{-(x_i-\mu)^2}{2\sigma^2}}$$

$$\mu_{MLE} = argmax_{\mu} P(D|\mu, \sigma)$$

$$\frac{d\left(-Nln\left(\sigma\sqrt{2\pi}\right) - \sum_{i=1}^{N} \frac{(x_i - \mu)^2}{2\sigma^2}\right)}{d\mu} = 0$$

$$\mu_{MLE} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

Question 2.2

$$P(\mu|D) = P(D|\mu)P(\mu)$$

$$P(\mu|D) = \left(\prod_{i=1}^{N} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x_i-\mu)^2}{2\pi^2}}\right) \lambda e^{-\lambda\mu}$$

$$\frac{d}{d\mu} lnP(\mu|D) = \frac{1}{\sigma^2}\sum_{i=1}^{N}(x_i - \mu) - \lambda = 0$$

$$\mu_{MAP} = \frac{\sum_{i=1}^{N}\frac{x_i}{\sigma^2} - \lambda\sigma}{N}$$

Question 2.3

$f(1;1;1) = \frac{1}{2\pi} = 0.15915$

$f(2;1;1) = \frac{1}{2\pi}e^{-\frac{1}{2}} = 0.09653$

BBC News Classification

Question 3.1

Question 3.1.1

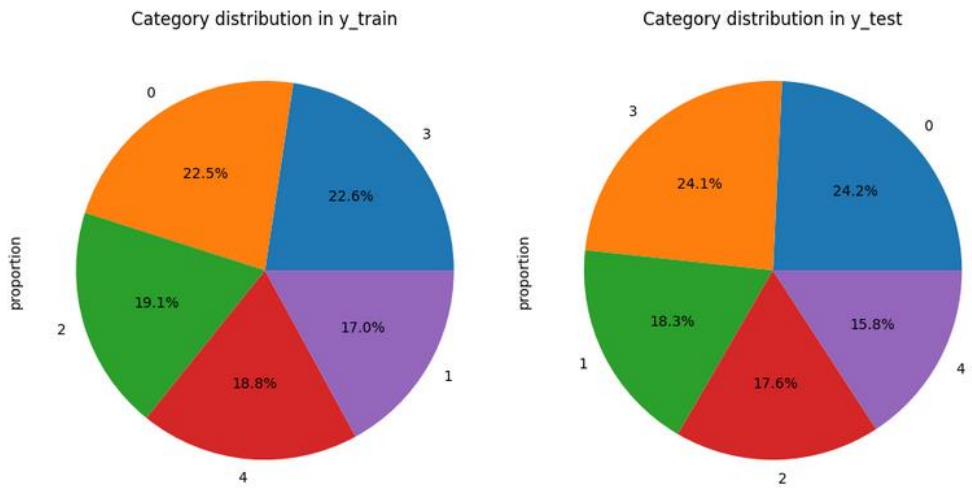

Figure 1

Question 3.1.2

Prior probabilities are calculated as shown below:

```
The prior probability of class 0 is 0.225
The prior probability of class 1 is 0.170
The prior probability of class 2 is 0.191
The prior probability of class 3 is 0.226
The prior probability of class 4 is 0.188
```

Figure 2

We can see that the calculations are the same as y_train data in Figure 1.

Question 3.1.3

By observing the pie chart of y_train we have relatively balanced data. An imbalanced training set would cause a model to be biased towards the majority class, leading to poor generalization for the minority class, which would result in a high overall accuracy in training set that masks the model's poor performance in the underrepresented class.

Question 3.1.4

```
The number of word 'alien' in Tech documents: 3
The number of word 'thunder' in Tech documents: 0
Log probability of 'alien' in Tech documents: -10.171387747476452
Log probability of 'thunder' in Tech documents: -inf
```

Figure 3

We can see that the word alien occurred three times and word thunder occurred 0 times. The log value '-inf' results from the log (0) value.

Codes for questions 3.1.1, 3.1.2 and 3.1.4 are given in Appendix A.

Question 3.2

```
Multinomial NB without smoothing with log(0) = -inf
Accuracy: 0.242
Confusion Matrix:
Predicted    0  All
Actual
0          135  135
1          102  102
2           98   98
3          134  134
4           88   88
All        557  557
Multinomial NB without smoothing with log(0) = 10^-12
Accuracy: 0.964
Confusion Matrix:
Predicted    0   1   2    3   4  All
Actual
0          129   0   3    1   2  135
1            0  94   2    0   6  102
2            1   1  94    0   2   98
3            0   0   0  134   0  134
4            0   2   0    0  86   88
All        130  97  99  135  96  557
```

Figure 4

Since log(0) translates to negative infinity in the first case, when it is included in the probability calculations, it causes the overall probability for a class to become negative infinity as well. As a result, the classifier ends up predicting the class with the least negative infinity occurrences, which could lead to only a few classes being predicted, which is single class in this case. That's why the confusion matrix looks that way in the first model. But, when we set log(0) value to 10^-12, the performance issues are fixed.

Question 3.3

```
Multinomial NB with smoothing Accuracy: 0.977
Confusion Matrix:
Predicted    0   1   2    3   4  All
Actual
0          131   0   2    0   2  135
1            0  97   0    0   5  102
2            1   0  96    0   1   98
3            0   0   1  133   0  134
4            1   0   0    0  87   88
All        133  97  99  133  95  557
```

Figure 5

Using a Dirichlet prior of $\alpha = 1$, which introduces smoothing, improves the classifier's ability. The theoretical basis for this improvement is that smoothing adjusts the word counts to prevent any feature's probability from being zero due to a lack of occurrence in the training data. In practice, this approach adjusts the model's estimates to be more robust against overfitting to the training data.

It is seen that, results are significantly better than the first case of Multinomial NB without smoothing, which has the accuracy of 0.242.

Question 3.4

```
Bernoulli NB with smoothing Accuracy: 0.966
Confusion Matrix:
Predicted    0   1   2    3   4  All
Actual
0          132   0   2    0   1  135
1            3  96   1    0   2  102
2            4   0  94    0   0   98
3            0   0   0  134   0  134
4            4   2   0    0  82   88
All        143  98  97  134  85  557
```

Figure 6

The accuracy is still very high but compared to the Multinomial case with smoothing, it is slightly lower. It performed better with classifying classes 0 and 3 but did worse in other classes.

The choices of models for further cases depend on the datasets and the purposes. For the given datasets it is seen that Bernoulli Naïve Bias and Multinomial Naïve Bias with smoothing does a great job.

Appendices

Appendix A

```
## Question 3.1.1

# Load the label data
y_train = pd.read_csv('y_train.csv', header = None, delim_whitespace=True)
y_test = pd.read_csv('y_test.csv', header=None, delim_whitespace=True)
x_train = pd.read_csv('X_train.csv', delim_whitespace=True, low_memory=False)
x_test = pd.read_csv('X_test.csv', delim_whitespace=True, low_memory=False)

# Calculate the distribution of categories in the training set
train_counts = y_train[0].value_counts(normalize=True) * 100
# Calculate the distribution of categories in the test set
test_counts = y_test[0].value_counts(normalize=True) * 100

# Plot the distribution for the training set
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
train_counts.plot(kind='pie', autopct='%1.1f%%')
plt.title('Category distribution in y_train')

# Plot the distribution for the test set
plt.subplot(1, 2, 2)
test_counts.plot(kind='pie', autopct='%1.1f%%')
plt.title('Category distribution in y_test')

plt.tight_layout()
plt.show()

## Question 3.1.2

train_counts = y_train[0].value_counts(normalize=True)
for i in range(len(train_counts)):
    print(f'The prior probability of class {i} is {train_counts[i]:.3f}')


## Question 3.1.4

train = pd.concat([x_train, y_train], axis=1)

# Get the rdocuments with the 4
tech_docs = train[train.iloc[:, -1] == 4]  # Replace 'Tech' with the actual value if it's numeric

# Count words "alien" and "thunder" in these documents
alien_count_in_tech = tech_docs['alien'].sum()
thunder_count_in_tech = tech_docs['thunder'].sum()

# Calculate the total number of words in class tech docs
total_words_in_tech = tech_docs.drop(tech_docs.columns[-1], axis=1).sum().sum()

# Calculate the probabilities
p_alien_given_tech = alien_count_in_tech / total_words_in_tech
p_thunder_given_tech = thunder_count_in_tech / total_words_in_tech

# Calculate the log probabilities
log_p_alien_given_tech = np.log(p_alien_given_tech)
log_p_thunder_given_tech = np.log(p_thunder_given_tech)

print(f"The number of word 'alien' in Tech documents: {alien_count_in_tech}")
print(f"The number of word 'thunder' in Tech documents: {thunder_count_in_tech}")

print(f"Log probability of 'alien' in Tech documents: {log_p_alien_given_tech}")
print(f"Log probability of 'thunder' in Tech documents: {log_p_thunder_given_tech}")
```