EE 511: Homework Assignment 1 Solutions

Solution 1

Solution 1 Given in the problem,

* m characters
* d possible values
* Text is described by Vector V_i where i^{th} characters.

$$m = \sum_{i=1}^{d} V_i$$

By using poisson's distribution to find $P(V|\ell)$

$$P(V|\ell) = \frac{\bar{e}^{\lambda} \lambda^{V_1, V_2 \ldots V_d}}{V_1, V_2, \ldots V_d}$$

Using the MAP decission rule for detecting the language.

$$\hat{\ell} = \operatorname{argmax} \left[\log p(V|\ell) + \log p(\ell) \right]$$

$$= \operatorname{argmax} \left[\log \left(\frac{\bar{e}^{\lambda} \lambda^{V_1, V_2 \ldots V_d}}{V_1, V_2 \ldots V_d} \right) + \log p(\ell) \right]$$

In the problem prior is q_k, putting that into the above equation.

$$= \operatorname{argmax} \left[\log \frac{\bar{e}^{\lambda} \lambda^{V_i}}{V_i !} + \log q_k \right]$$

$$= \operatorname{argmax} \left[\sum_{i=1}^{d} -\lambda + \sum_{i=1}^{d} \log \left(\frac{\lambda^{V_i}}{V_i !} \right) + \log(q_k) \right]$$

$$= \arg\max \left[-d\lambda + \log \lambda \sum_{i=1}^{d} V_i - \sum_{i=1}^{d} \log (V_i !) + \log q_K \right]$$

differentiating and equating to zero,

$$\frac{dl}{d\lambda} = -d + \frac{1}{\lambda} \left(\sum_{i=1}^{d} V_i \right) = 0$$

$$\frac{1}{\lambda} \left(\sum_{i=1}^{d} V_i \right) = d$$

$$\boxed{\lambda_{MLE} = \frac{\sum_{i=1}^{d} V_i}{d}.}$$

Solution 2

Let the number of sample messages be N. For spam, the number of sample messages be Ns and for non-spam be Nns , where Ns = 3 and Nns = 4. Hence, the total number of messages ($Ns + Nns$) will be 7.

Now, creating the table to check number of frequency of words for spam messages

Words in spam messages	Frequency
Million	1
dollar	1
offer	2
secret	2

today	1
is	1
Remaining words	0

Similarly, creating the table to check number of frequency of words for non- spam messages

Words in non-spam messages	**Frequency**
Low	2
price	2
for	1
valued	1
customer	1
play	1
secret	1
sports	1
today	1
is	1
healthy	1
pizza	1
Remaining words	0

Common words that appeared in the spam and non-spam were, “today” , “secret” and “is”.

Now, determining the MLE for the following :

$\Theta s = \frac{3}{7}$, $\Theta secret/S = \frac{2}{3}$, $\Theta secret/NS = \frac{1}{4}$, $\Theta sports/NS = \frac{2}{4} = \frac{1}{2}$, $\Theta dollar/S = \frac{1}{3}$

Solution 3

Solution 3. Given in the problem,

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

$$S = \frac{1}{n} \sum_{i=1}^{n} |x_i| \longrightarrow \text{i.i.d}$$

Laplacian random vatriable is,

$$P(x|\theta) = \frac{\theta}{2} e^{-\theta|x|}$$

(a) To find the MLE estimate, we have n data samples.

$$\hat{\theta} = \arg\max_{\theta} L(\theta) = \arg\max_{\theta} \sum_{i=1}^{n} \log P(x_i|\theta)$$

$$= \arg\max_{\theta} \sum_{i=1}^{n} \left[\log\left(\frac{\theta}{2} e^{-\theta|x_i|}\right)\right]$$

$$= \arg\max_{\theta} \sum_{i=1}^{n} \left[\log\frac{\theta}{2} - \theta|x_i|\right]$$

$$= \arg\max_{\theta} \left[n \log\frac{\theta}{2} - \theta \sum_{i=1}^{n} |x_i|\right]$$

Now, taking the derivative of the above expression with respect to θ and setting it to zero,

$$\frac{dL(\theta)}{d\theta} = \frac{n}{\theta} - \sum_{i=1}^{n} |x_i| = 0$$

$$\frac{n}{\theta} - \sum_{i=1}^{n} |x_i| = 0$$

$$\frac{n}{\theta} = \sum_{i=1}^{n} |x_i|$$

$$\boxed{\hat{\theta}_{MLE} = \frac{n}{\sum_{i=1}^{n} |x_i|}}$$

(b) To find the MAP estimate $p(\theta) = \alpha e^{-\theta\alpha}$ if $\theta > 0$.

Putting the value of prior $(p(\theta))$ in the MAP estimate formula

$$\theta_{MAP} = \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{n} \left[\log p(x_i|\theta) + \log p(\theta) \right]$$

$$= \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{n} \left[\log\left(\frac{\theta}{2} e^{-\theta|x_i|}\right) + \log\left(\alpha e^{-\theta\alpha}\right) \right]$$

$$= \underset{\theta}{\operatorname{argmax}} \left[n \log\frac{\theta}{2} - \theta \sum_{i=1}^{n} |x_i| + \log\alpha - \alpha\theta \right]$$

now taking the derivative of the above equation with respect to θ.

$$\left(\frac{dL(\theta)}{d\theta}\right)_{\theta_{MAP}} = \frac{n}{\theta} - \sum_{i=1}^{n} |x_i| - \alpha = 0$$

$$\frac{n}{\theta} = \alpha + \sum_{i=1}^{n} |x_i|$$

$$\theta_{MAP} = \frac{n}{\alpha + \sum_{i=1}^{n} |x_i|}$$

$$\theta_{MAP} = \frac{1}{\frac{\alpha}{n} + \frac{\sum_{i=1}^{n} |x_i|}{n}}$$

when n tends to ∞ then the α term goes to 0 (zero), therefore

$\theta_{MAP} \longrightarrow \theta_{ML}$, when $n \to \infty$.

Solution 4 is on the next page

Solution 4 Given,

* $P_S(x|b) \sim N(b, \sigma^2 = 9)$
* Gaussian prior, $p(b) \sim N(0, \sigma_0^2 = 4)$
* Data sample $D_n = \{x_1 \dots x_n\}$
* Posterior of the sensor bias is

$$P(b|D_n) \sim N(b_n, \sigma_n^2)$$

where, $\boxed{b_n = \alpha \hat{b}_{ML} + (1-\alpha) b_0}$ —(1)

and $\boxed{\sigma_n^2 = \frac{\sigma^2}{n+R}}$ —(2)

$\Rightarrow$ α and R can be calulated by the following formulas $\alpha = \frac{n}{n+R}$ and $R = \frac{\sigma^2}{\sigma_0^2}$

4(a) In order to calculate no. of samples (n), we will use formula (2).
In formula (2), σ_n^2 (variance) is unknown, therefore we need to determine σ_n^2 first by using information given.

* 95% of probability mass of Gaussian is within ± 1.96
* b_n is within ± 0.5 of the true bias

therefore, b_n interval will be of 1.

$\Rightarrow$ 95% confident that the true mean lies in the interval

$$b_n + 1.96\sigma_n - (b_n - 1.96\sigma_n) = 1$$

$$3.92\sigma_n = 1$$

$$\sigma_n = \frac{1}{3.92}$$

$$\sigma_n^2 = \left(\frac{1}{3.92}\right)^2 = 0.0650$$

Now putting the value of σ_n^2 in formula (2)

$$\sigma_n^2 = \frac{\sigma^2}{n+R} \quad \text{where, } \sigma^2 = 9$$

$$R = \frac{\sigma^2}{\sigma_0^2} = \frac{9}{4} = 2.25$$

$$n+R = \frac{\sigma^2}{\sigma_n^2} = \frac{9}{0.0650}$$

$$n+R = 138.46$$

$$n = 138.46 - R$$

$$n = 138.46 - 2.25$$

$n = 136.21$, Hence minimum number of samples for 95% CI $N = 136$

4(b) If the variance of the bias prior was large let's assume $\sigma_0^2 = 18$ and $\sigma^2 = 9$, putting the values in the formula (2) to check the sample size.

$$n + R = \frac{\sigma^2}{\sigma_n^2}$$ where $\sigma_n^2 = 0.0650$ (using same value)

$$n + R = \frac{9}{0.0650} = 138.46$$

$$n = 13.8.46 - R$$

where, $R = \frac{\sigma^2}{\sigma_0^2} = \frac{9}{18} = 0.5$

$$n = 138.46 - 0.5$$

$$\boxed{n = 137.96}$$

Here sample size is slightly larger than the sample size obtained in part (a)

Hence, we can conclude if $\sigma_0^2 \uparrow$ then sample size increase slightly.

4(c) If the mean (prior) $\mu_0 = 0.5$ then using formula (1)

$$b_n = \alpha \hat{b}_{ML} + (1-\alpha) b_0$$

where $\mu_0 = b_0 = 0.5$

$$b_n = \left(\frac{n}{n+R}\right)\left(\frac{S_{n_2}}{\cancel{n}}\right) + \left(1 - \left(\frac{n}{n+R}\right)\right) 0.5$$

$$b_n = \frac{S_{n_2}}{n+R} + \left(\frac{R}{n+R}\right) \times 0.5$$

$$b_n = \frac{S_{n_2}}{136 + 2.25} + \left(\frac{2.25}{136+2.25}\right) \times 0.5$$

$$b_n = \frac{S_{n_2}}{138.25} + 0.0081$$

where $S_{n_2} = \sum_{i=1}^{n} c(x_i)$

Solution 5 Computer Assignment is on the next page

Part (a) by KNN
Part (b) by Gaussian

In [1]:
```
# In the section below we are importing libraries which will be used in the
# <numpy> to compute mean error for the predicted values
# <matplotlib.pyplot> to plot the error graph
# <pandas> to load and parse the csv file into meaningful dataframes
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [2]:
```
# In the section below we are loading training data csv file into dataframe
df_train = pd.read_csv('zip.train.p.csv')
df_train.head()
```

Out[2]:

	6.0000	-1.0000	-1.0000.1	-1.0000.2	-1.0000.3	-1.0000.4	-1.0000.5	-1.0000.6	-0.6310	0.8620	..
0	5.0	-1.0	-1.0	-1.0	-0.813	-0.671	-0.809	-0.887	-0.671	-0.853	..
1	4.0	-1.0	-1.0	-1.0	-1.000	-1.000	-1.000	-1.000	-1.000	-1.000	..
2	7.0	-1.0	-1.0	-1.0	-1.000	-1.000	-0.273	0.684	0.960	0.450	..
3	3.0	-1.0	-1.0	-1.0	-1.000	-1.000	-0.928	-0.204	0.751	0.466	..
4	6.0	-1.0	-1.0	-1.0	-1.000	-1.000	-0.397	0.983	-0.535	-1.000	..

5 rows × 258 columns

In [3]:
```
# In the section below we are separating attributes and labels for training
X_train = df_train.iloc[:, 1:256].values
Y_train = df_train.iloc[:, 0].values
```

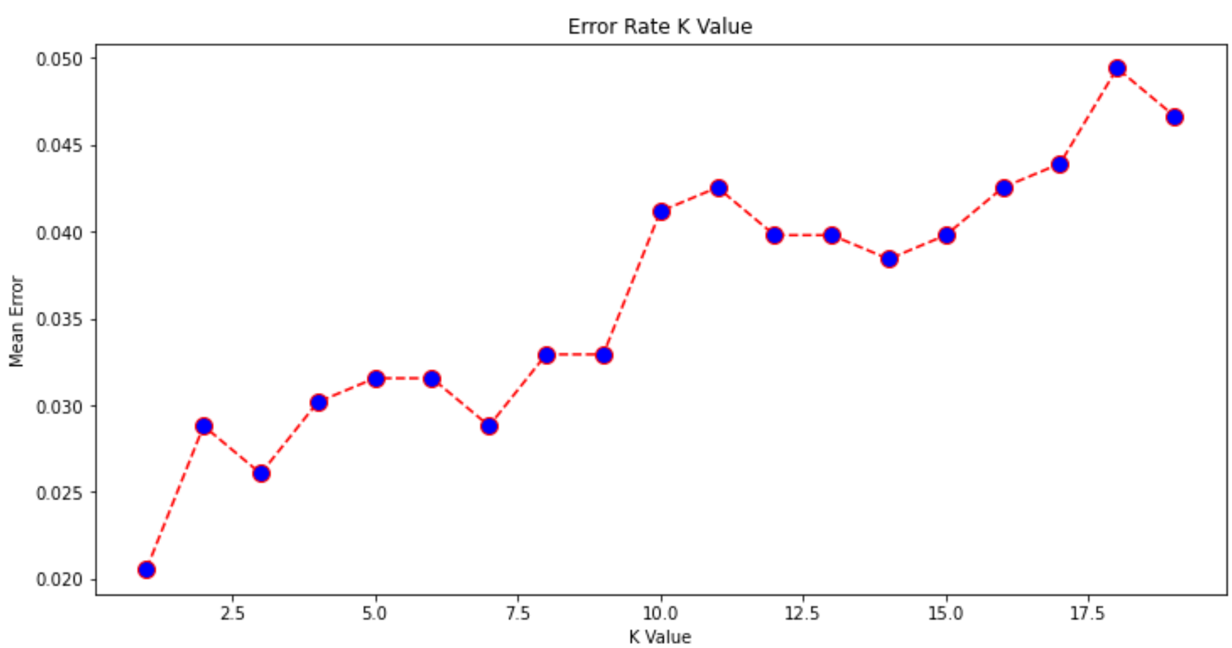
In [4]:
```
# In the section below we will use 10% of the training data as validation d
from sklearn.model_selection import train_test_split
# split dataset into training and validation data
X_train, X_validation, Y_train, Y_validation = train_test_split(
    X_train, Y_train, test_size=0.1, random_state=1, stratify=Y_train)
```

In [17]:
```python
# In the section below we are training and validating KNN classifier ...
# ... for different values of K (1 to 20).
from sklearn.neighbors import KNeighborsClassifier
error = []
min_error = 1
best_knn = KNeighborsClassifier(n_neighbors=0)
# find the best classifier for k = {1:20} based on minimum mean error.
for i in range(1, 20):
    knn = KNeighborsClassifier(n_neighbors=i)
    # training the model
    knn.fit(X_train, Y_train)
    # predicting the label using test data
    y_pred = knn.predict(X_validation)
    error_i = np.mean(y_pred != Y_validation)
    error.append(error_i)
    if(error_i < min_error):
        print(i, error_i)
        min_error = error_i
        best_knn = knn
```

```
1 0.0205761316872428
```

In [18]:
```python
# In the section below we are plotting to visualize mean error agaist diffe
plt.figure(figsize=(12, 6))
plt.plot(range(1, 20), error, color='red', linestyle='dashed', marker='o',
         markerfacecolor='blue', markersize=10)
plt.title('Error Rate K Value')
plt.xlabel('K Value')
plt.ylabel('Mean Error')
```

Out[18]:
```
Text(0, 0.5, 'Mean Error')
```



In [19]:
```
# In the section below we are loading test data csv file into dataframe nam
df_test = pd.read_csv('zip.test.p.csv')
print(df_test.head())
```

```
    9   -1  -1.1  -1.2   -1.3  -1.4  -0.948  -0.561  0.148  0.384  ...  -
1.136  \
0  6 -1.0  -1.0  -1.0 -1.000  -1.0    -1.0  -1.000 -1.000 -1.000  ...  -
1.000
1  3 -1.0  -1.0  -1.0 -0.593   0.7     1.0   1.000  1.000  1.000  ...
1.000
2  6 -1.0  -1.0  -1.0 -1.000  -1.0    -1.0  -1.000 -1.000 -1.000  ...  -
1.000
3  6 -1.0  -1.0  -1.0 -1.000  -1.0    -1.0  -1.000 -0.858 -0.106  ...
0.901
4  0 -1.0  -1.0  -1.0 -1.000  -1.0    -1.0   0.195  1.000  0.054  ...
0.224

   -0.908   0.43  0.622  -0.973  -1.137  -1.138  -1.139  -1.140  -1.141
0  -1.000 -1.000 -1.000  -1.000  -1.000  -1.000    -1.0    -1.0    -1.0
1   0.717  0.333  0.162  -0.393  -1.000  -1.000    -1.0    -1.0    -1.0
2  -1.000 -1.000 -1.000  -1.000  -1.000  -1.000    -1.0    -1.0    -1.0
3   0.901  0.901  0.290  -0.369  -0.867  -1.000    -1.0    -1.0    -1.0
4   1.000  0.988  0.187   0.139  -0.641  -0.812    -1.0    -1.0    -1.0

[5 rows x 257 columns]
```

In [20]:
```
# In the section below we are separating attributes and labels for test dat
X_test = df_test.iloc[:, 1:256].values
Y_test = df_test.iloc[:, 0].values
```

In [21]:
```
# let's predict the labels using the best model
Y_pred = best_knn.predict(X_test)
```

In [22]:
```
# Print out classification report
from sklearn.metrics import classification_report, confusion_matrix
print(classification_report(Y_test, Y_pred))
# Print out confusion matrix
print(confusion_matrix(Y_test, Y_pred))
```

```
           6       0.96      0.96      0.96       170
           7       0.92      0.94      0.93       147
           8       0.95      0.89      0.92       166
           9       0.89      0.95      0.92       176

    accuracy                           0.94      2006
   macro avg       0.94      0.94      0.94      2006
weighted avg       0.94      0.94      0.94      2006

[[355   0   2   0   0   0   0   1   0   1]
 [  0 255   0   0   6   0   2   1   0   0]
 [  6   1 183   2   1   0   0   2   3   0]
 [  3   0   2 153   0   6   0   0   0   2]
 [  0   3   1   0 179   1   2   3   1  10]
 [  2   0   2   4   0 145   2   0   3   2]
 [  0   0   1   0   2   3 164   0   0   0]
 [  0   1   1   1   4   0   0 138   0   2]
 [  5   0   2   5   1   1   0   1 148   3]
 [  0   0   1   0   2   0   0   4   1 168]]
```

In [1]:
```
# In the section below, we are importing libraries which will be used in the pro
# <numpy> to compute mean error for the predicted values
# <pandas> to load and parse the csv file into meaningful dataframes
import numpy as np
import pandas as pd
```

In [2]:
```
# In the section below, we are loading training data csv file into dataframe nam
df_train = pd.read_csv('zip.train.p.csv')
print(df_train.head())
```

```
   6.0000  -1.0000  -1.0000.1  -1.0000.2  -1.0000.3  -1.0000.4  -1.0000.5  \
0     5.0     -1.0       -1.0       -1.0     -0.813     -0.671     -0.809
1     4.0     -1.0       -1.0       -1.0     -1.000     -1.000     -1.000
2     7.0     -1.0       -1.0       -1.0     -1.000     -1.000     -0.273
3     3.0     -1.0       -1.0       -1.0     -1.000     -1.000     -0.928
4     6.0     -1.0       -1.0       -1.0     -1.000     -1.000     -0.397

   -1.0000.6  -0.6310  0.8620  ...  0.8230  1.0000.39  0.4820  -0.4740  \
0     -0.887   -0.671  -0.853  ...  -0.671     -0.033   0.761    0.762
1     -1.000   -1.000  -1.000  ...  -1.000     -1.000  -0.109    1.000
2      0.684    0.960   0.450  ...   1.000      0.536  -0.987   -1.000
3     -0.204    0.751   0.466  ...   0.639      1.000   1.000    0.791
4      0.983   -0.535  -1.000  ...   0.015     -0.862  -0.871   -0.437

   -0.9910  -1.0000.121  -1.0000.122  -1.0000.123  -1.0000.124  Unnamed: 257
0    0.126       -0.095       -0.671       -0.828         -1.0           NaN
1   -0.179       -1.000       -1.000       -1.000         -1.0           NaN
2   -1.000       -1.000       -1.000       -1.000         -1.0           NaN
3    0.439       -0.199       -0.883       -1.000         -1.0           NaN
4   -1.000       -1.000       -1.000       -1.000         -1.0           NaN

[5 rows x 258 columns]
```

In [3]:
```
# In the section below, we are separating attributes and labels for training dat
X_train = df_train.iloc[:, 1:256].values
Y_train = df_train.iloc[:, 0].values
```

In [4]:
```
# In the section below, we are loading test data csv file into dataframe named d
df_test = pd.read_csv('zip.test.p.csv')
df_test.head()
```

Out[4]:

	9	-1	-1.1	-1.2	-1.3	-1.4	-0.948	-0.561	0.148	0.384	...	-1.136	-0.908	0.43	0.
0	6	-1.0	-1.0	-1.0	-1.000	-1.0	-1.0	-1.000	-1.000	-1.000	...	-1.000	-1.000	-1.000	-1.
1	3	-1.0	-1.0	-1.0	-0.593	0.7	1.0	1.000	1.000	1.000	...	1.000	0.717	0.333	0
2	6	-1.0	-1.0	-1.0	-1.000	-1.0	-1.0	-1.000	-1.000	-1.000	...	-1.000	-1.000	-1.000	-1.
3	6	-1.0	-1.0	-1.0	-1.000	-1.0	-1.0	-1.000	-0.858	-0.106	...	0.901	0.901	0.901	0.
4	0	-1.0	-1.0	-1.0	-1.000	-1.0	-1.0	0.195	1.000	0.054	...	0.224	1.000	0.988	0

5 rows × 257 columns

In [5]:
```
# In the section below, we are separating attributes and labels for test data.
X_test = df_test.iloc[:, 1:256].values
Y_test = df_test.iloc[:, 0].values
```

In [6]:
```
#Import Gaussian Naive Bayes model
from sklearn.naive_bayes import GaussianNB

#Create a Gaussian Classifier
model = GaussianNB()

#Train the model using the training sets
model.fit(X_train,Y_train)

#Predict Output
Y_pred= model.predict(X_test)
```

In [7]:
```
# Print out classification report
from sklearn.metrics import classification_report, confusion_matrix
print(classification_report(Y_test, Y_pred))
# Print out confusion matrix
print(confusion_matrix(Y_test, Y_pred))
```

```
              precision    recall  f1-score   support

           0       0.94      0.82      0.87       359
           1       0.91      0.96      0.94       264
           2       0.77      0.72      0.74       198
           3       0.90      0.48      0.62       166
           4       0.75      0.28      0.41       200
           5       0.81      0.48      0.60       160
           6       0.66      0.89      0.76       170
           7       0.85      0.90      0.87       147
           8       0.44      0.66      0.53       166
           9       0.45      0.86      0.59       176

    accuracy                           0.72      2006
   macro avg       0.75      0.70      0.69      2006
weighted avg       0.77      0.72      0.72      2006

[[294   1  13   0   4   2  12   1  31   1]
 [  0 254   1   0   1   0   4   0   1   3]
 [  2   0 142   3   3   4  12   1  30   1]
 [  6   0  11  79   1   3   4   6  47   9]
 [  0   4   4   0  57   1   7   5   5 117]
 [  9   1   2   5   2  77  36   2  19   7]
 [  3   6   5   0   0   2 151   0   3   0]
 [  0   1   1   0   3   0   0 132   1   9]
 [  0   1   5   1   3   6   2   0 109  39]
 [  0  10   0   0   2   0   0   9   3 152]]
```