Assignment 5: Probabilistic Learning

CS 589 - ML

Shubham Shetty (shubhamshett@umass.edu)
Brinda Murulidhara (bmurulidhara@umass.edu)
Adarsh Kolya (akolya@umass.edu)

April 2021

Preface

Before running our codes, the following packages are imported and test & training data are assigned to variables. Also, some reusable functions are defined -

```
# Import Statements
import numpy as np
from matplotlib import pyplot as plt
from scipy.stats import norm


# Load Data

## Training Data
X = np.loadtxt('x.csv')
Y = np.loadtxt('y.csv')

## Test Data
x = np.loadtxt('x_test.csv')
y = np.loadtxt('y_test.csv')


# Predictor Function
def f_m(x,m):
    if m>0:
        return x**m
    else:
        return 0
```

Answer 1

```
def prior(m):
    p = {0: 0.2,
         1: 0.2,
         2: 0.2,
         3: 0.1,
         4: 0.1,
         5: 0.05,
         6: 0.05,
         7: 0.05,
         8: 0.025,
         9: 0.025}

    return p[m]
```

Answer 2

```
def show_priors():
    for m in range(10):
        print("m", m, "prior(m)", prior(m))

    x = [0,1,2,3,4,5,6,7,8,9]
    y = [prior(m) for m in x]
    plt.bar(x, y, label="Priors Bar Chart")
    plt.xlabel("m")
    plt.ylabel("Prior")
    plt.xticks(np.arange(0, 10, 1))
    plt.show()

show_priors()
```

Generated output -

```
m 0 prior(m) 0.2
m 1 prior(m) 0.2
m 2 prior(m) 0.2
m 3 prior(m) 0.1
m 4 prior(m) 0.1
m 5 prior(m) 0.05
m 6 prior(m) 0.05
m 7 prior(m) 0.05
m 8 prior(m) 0.025
m 9 prior(m) 0.025
```

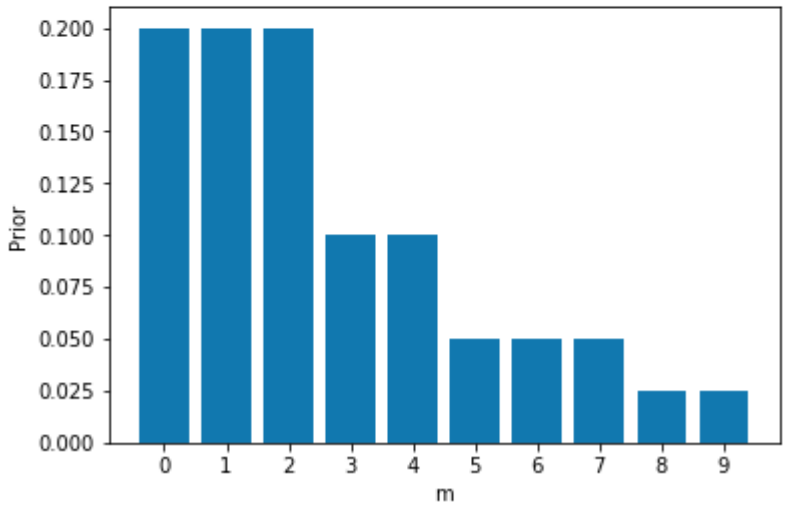

Figure 1: Priors Bar Chart

Answer 3

```
def likelihood_single(x, y, m):
    mu = 0 if m == 0 else x ** m
    sigma = 0.1
    return norm.pdf(y, mu, sigma)
```

Answer 4

```
def likelihood(X,Y,m):
    p = np.product(np.array(list(map(lambda x,y: likelihood_single(x,y,m), X, Y))))
    return p
```

Answer 5

```
X = np.loadtxt('x.csv')
Y = np.loadtxt('y.csv')


def plot_likelihood(X,Y):
    x = [0,1,2,3,4,5,6,7,8,9]
    y = [likelihood(X,Y,m) for m in x]
    plt.bar(x, y)
    plt.xlabel("m")
    plt.ylabel("Likelihood")
    plt.xticks(np.arange(0, 10, 1))
    plt.show()


plot_likelihood(X,Y)
```

Generated output -

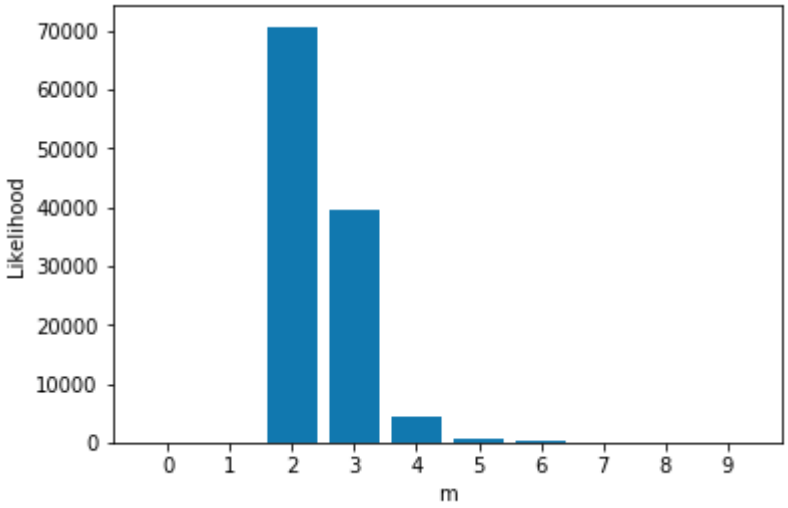

Figure 2: Likelihood Bar Chart

Answer 6

$$p(m|\text{Data}) = \frac{p(m)\ p(\text{Data}|m)}{p(\text{Data})} \tag{1}$$

We know that m can range from 0 - 9. So

$$\begin{aligned} &\sum_{m=0}^{9} (p(m|\text{Data})) = 1 \\ \implies &\sum_{m=0}^{9} \left(\frac{p(m)\ p(\text{Data}|m)}{p(\text{Data})} \right) = 1 \\ \implies &p(\text{Data}) = \sum_{m=0}^{9} (p(m)\ p(\text{Data}|m)) \end{aligned} \tag{2}$$

Substituting (2) in (1) we get

$$\boxed{\mathbf{p(m|Data)} = \frac{\mathbf{p(m)\ p(Data|m)}}{\sum_{\mathbf{m=0}}^{\mathbf{9}} (\mathbf{p(m)\ p(Data|m)})}}$$

Answer 7

```
def posterior(X,Y,m):
    m_i = [0,1,2,3,4,5,6,7,8,9]
    normalizer = np.sum(np.array([prior(i)*likelihood(X,Y,i) for i in
    m_i]))
    p = (likelihood(X,Y,m)*prior(m))/normalizer
    return p
```

Answer 8

```
def plot_posterior(X,Y):
    x = [0,1,2,3,4,5,6,7,8,9]
    y = [posterior(X,Y,m) for m in x]
    #sum_all = np.sum(np.array([posterior(X,Y,m) for m in x]))
    #print(sum_all)
    plt.bar(x, y)
    plt.xlabel("m")
    plt.ylabel("Posterior")
    plt.xticks(np.arange(0, 10, 1))
    #plt.yticks(np.arange(0, 1.1, 0.1))
    plt.show()


plot_posterior(X,Y)
```

Generated output -

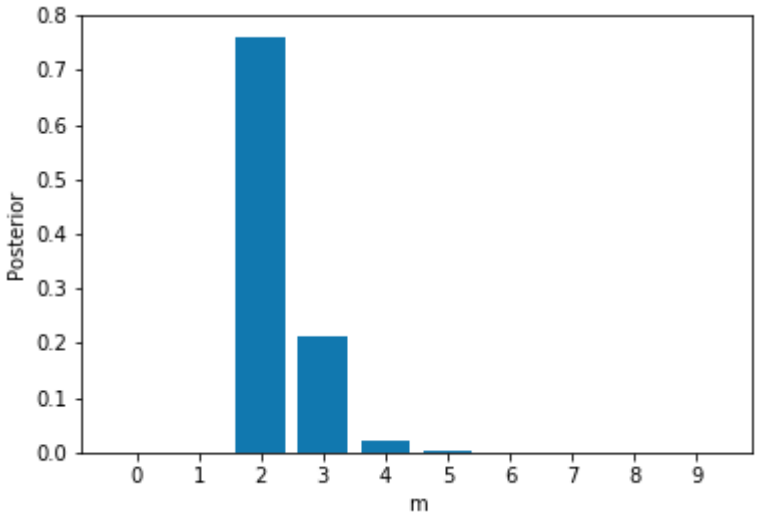

Figure 3: Posteriors Bar Chart

Answer 9

```
def MAP(X,Y):
    m_i = [0,1,2,3,4,5,6,7,8,9]
    p = [posterior(X,Y,i) for i in m_i]
    m = m_i[p.index(max(p))]
    return m
```

Answer 10

$m_{MAP} = \mathbf{2}$
Posterior probability = $p(m_{MAP}|\text{Data}) = \mathbf{0.7610021202297498}$

Answer 11

```
def predict_MAP(x,X,Y):
    m = MAP(X,Y)
    f = x ** m if m > 0 else 0
    return f
```

Answer 12

```
x = np.loadtxt('x_test.csv')
y = np.loadtxt('y_test.csv')

def MAP_plot(X,Y,x_test,y_test):
    y_pred = list(map(lambda x:predict_MAP(x,X,Y), x_test))
    plt.scatter(x_test, y_test, marker="x", label="Test True Output")
    plt.scatter(x_test, y_pred, marker="o", label="Test Prediction
   Output")
    plt.xlabel("Test Input x")
    plt.ylabel("Test Output y")
    plt.legend()
    plt.show()


MAP_plot(X,Y,x,y)
```

Generated output -

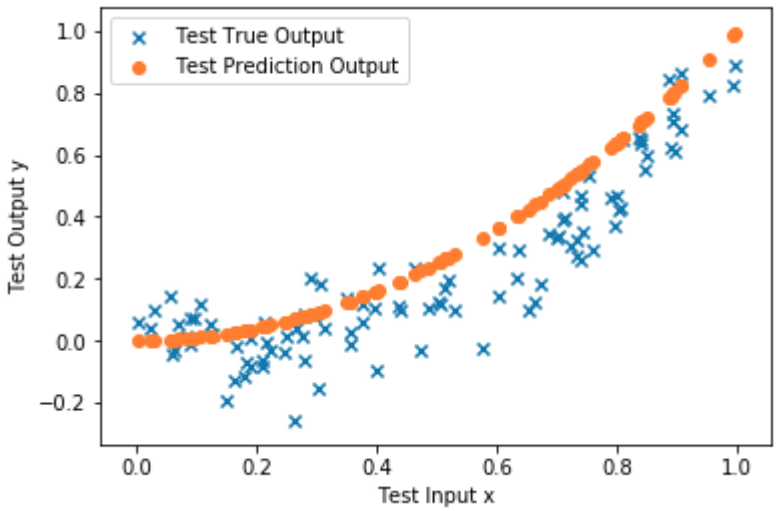

Figure 4: MAP Test Predictions vs True Output

Answer 13

```
def MAP_MSE(X,Y,x_test,y_test):
    mse = np.sum(np.square((np.array(y_test) - np.array(list(map(
    lambda x:predict_MAP(x,X,Y), x_test))))))/100
    return mse

print(f"MAP_MSE: {MAP_MSE(X,Y,x,y)}")
```

Generated MSE value - **0.02146290417007275**

Answer 14

```
def f_m(x,m):
    # Predictor Function
    if m>0:
        return x**m
    else:
        return 0

def predict_Bayes(x,X,Y):
    m_i = [0,1,2,3,4,5,6,7,8,9]
    f = np.sum(np.array([posterior(X,Y,m)*f_m(x,m) for m in m_i]))
    return f
```

Answer 15

```
def Bayes_plot(X,Y,x_test,y_test):
    y_pred = list(map(lambda x:predict_Bayes(x,X,Y), x_test))
    plt.scatter(x_test, y_test, marker="x", label="Test True Output")
    plt.scatter(x_test, y_pred, marker="o", label="Test Prediction
   Output")
    plt.xlabel("Test Input x")
    plt.ylabel("Test Output y")
    plt.legend()
    plt.show()


Bayes_plot(X,Y,x,y)
```

Generated output -

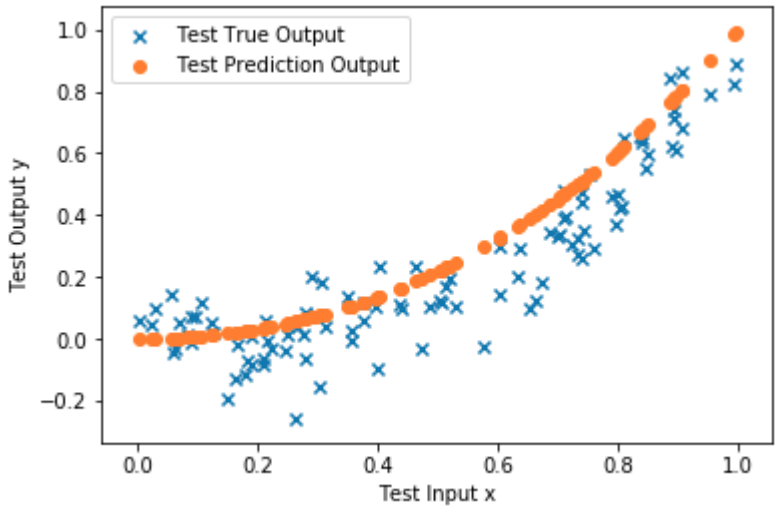

Figure 5: Bayes Test Predictions vs True Output

Answer 16

```
def Bayes_MSE(X,Y,x_test,y_test):
    mse = np.sum(np.square((np.array(y_test) - np.array(list(map(
    lambda x:predict_Bayes(x,X,Y), x_test))))))/100
    return mse

print(f"Bayes_MSE: {Bayes_MSE(X,Y,x,y)}")
```

Generated MSE value - **0.01635700904708008**

Answer 17

The MSE for Bayes is lower than that of MAP.

The MAP estimate returns the model which provides the highest posterior probability and disregards all other models. On the other hand, Bayes estimator returns a fully calculated probability distribution based on all posteriors. Hence the Bayes estimator's prediction is more accurate than MAP estimator as it considers all models rather than just one. MAP estimator prediction tends to get closer to the Bayes' estimator predictions as the value of max posterior probability grows.