دانشگاه تهران دانشکده مهندسی برق و کامپیوتر
درس یادگیری ماشین تمرین اول

نام و نام خانوادگی	میلاد محمدی
شماره دانشجویی	**810100462**
تاریخ ارسال گزارش	**1401/08/08**

سوال اول

خطا به صورت زیر تعریف می‌شود:

$$l_1 = \lambda_{11}\, p(x|\omega_1)P(\omega_1) + \lambda_{21}\, p(x|\omega_2)\, P(\omega_2)$$

$$l_2 = \lambda_{12}\, p(x|\omega_1)\, P(\omega_1) + \lambda_{22}\, p(x|\omega_2)\, P(\omega_2)$$

چون $\lambda_{11} = \lambda_{22} = 0$:

$$\begin{cases} l_1 = \lambda_{21}\, p\,(x|\omega_2)\, p(\omega_2) \\ l_1 = \lambda_{12}\, p\,(x|\omega_1)\, p(\omega_1) \end{cases} \quad => \quad \frac{p(x|\omega_1)}{p(x|\omega_2)} = \frac{\lambda_{21}\, p(\omega_2)}{\lambda_{12}\, p(\omega_1)}$$

با توجه به توزیع احتمال دو کلاس :

$$\frac{p(x|\omega_1)}{p(x|\omega_2)} = \frac{\frac{1}{\sqrt{2\pi}\,\sigma} * e^{-\frac{1}{2} * \frac{x^2}{\sigma^2}}}{\frac{1}{\sqrt{2\pi}\,\sigma} * e^{-\frac{1}{2} * \frac{(x-1)^2}{\sigma^2}}} = exp\,(\frac{-1}{2}\,\frac{1}{\sigma^2}\,(2x-1))$$

$$\frac{\lambda_{21}\, p(\omega_2)}{\lambda_{12}\, p(\omega_1)} = exp\,(\frac{-1}{2}\,\frac{1}{\sigma^2}\,(2x-1))$$

$$X = \frac{1}{2} - \sigma^2\, \frac{\lambda_{21}\, p(\omega_2)}{\lambda_{12}\, p(\omega_1)}$$

سوال دوم

قسمت الف)

احتمال کلاس‌بندی صحیح به صورت زیر است:

$$P_c = \sum_{i=1}^{M} p(x \epsilon R_i,\ \omega_i)$$

با توجه به رابطه‌ی زیر می‌توانیم عبارت بالا را گسترش دهیم:

$$p(x \epsilon R_i, \omega_i) = P\ (x\ \in\ R_i|\omega_i)P\ (\omega_i) = (\int_{R_i} p(x|\omega_i\ dx))\ p(\omega_i)$$

$$P_c = \sum_{i=1}^{M}\ (\int_{R_i} p(x|\omega_i)p(\omega_i)\ dx\ )$$

از آنجا که رابطه بالا مجموع M کلاس مختلف برای به حداکثر رساندن احتمال است، R_i را به عنوان ناحیه‌ای از x تعریف می‌کنیم که در آن :

$$p(x|\omega_i)P(\omega_i)\ >\ p(x|\omega_j)P(\omega_j) \quad \forall j\ \neq\ i\ .$$

با توجه به R_i و رابطه‌ی $(\int_{R_i} p(x|\omega_i)p(\omega_i))\ dx$ مقدار P_c بیشترین مقدار خواهد بود و با تقسیم طرفین رابطه و قاعده بیز رابطه زیر را داریم:

$$P(\omega_i|x)\ >\ P(\omega_j|x) \quad \forall j\ \neq\ i\ .$$

قسمت ب)

از آنجایی که $\sum_{i=1}^{M} p(\omega_i|x) = 1$، حداقل باید یکی از $p(\omega_i|x)$ بزرگتر از $\frac{1}{M}$ باشد. پس :

$$P(\omega_i * |x) = \max_i P(\omega_i|x) \quad => \quad P(\omega_i * |x)\ \geq\ \frac{1}{M}$$

$$P_e = 1\ -\max_i P(\omega_i|x)\ = 1 - \frac{1}{M}\ = \frac{M-1}{M}$$

قسمت ج)

در حالت M کلاسه، یکی از کلاس‌ها را Positive و بقیه را Negative در نظر می‌گیریم و یک نمودار ROC رسم می‌کنیم. این کار را تا جایی تکرار می‌کنیم که برای تمام کلاس‌ها نمودار ROC را رسم کرده باشیم. در واقع برای تمامی کلاس‌ها یک نمودار ROC جدا رسم می‌کنیم که به روش One vs Rest معروف است.

قسمت د)

در حالتی که کاملا از هم مستقل باشند، روشی بهتر از Naïve Bayes وجود ندارد. در این حالت روش Naïve Bayes تنها بر اساس احتمال پیشین عمل می‌کند و با احتمال 50 درصد درستی یا نادرستی را تعیین می‌کند.

سوال سوم

احتمال μ به صورت زیر تعریف می‌شود:

$$\mathrm{l}(\mu) = \left(\prod_{i=1}^{N} p(x_i | \mu, \sigma^2)\right) p(\mu)$$

اگر از رابطه بالا لگاریتم بگیریم:

$$\ln \mathrm{l}(\mu) = \ln p(\mu) + \sum_{i=1}^{N} \ln p(x_i | \mu, \sigma^2) =$$

$$\ln \frac{\mu \exp(-\mu^2/2\sigma_\mu^2)}{\sigma_\mu^2} + \sum_{i=1}^{N} \left[\ln\left(\frac{1}{\sqrt{2\pi\,\sigma^2}}\right) - \frac{(x_i - \mu)^2}{2\sigma^2}\right] =$$

$$\ln \mu - \mu^2/2\sigma_\mu^2 - \ln \sigma_\mu^2 + N \ \ln \frac{1}{\sqrt{2\pi\,\sigma^2}} - \frac{1}{2\sigma^2} \sum_{i=1}^{N} (x_i - \mu)^2$$

حال از عبارت بالا نسبت به μ مشتق می‌گیریم:

$$\frac{1}{\mu} - \frac{\mu}{\sigma_\mu^2} + \frac{1}{\sigma^2} \sum_{i=1}^{N} x_i - \mu =$$

$$\left(\frac{1}{\sigma_\mu^2} + \frac{N}{\sigma^2}\right) \mu^2 - \left(\frac{1}{\sigma^2}\right)\left(\frac{1}{N} \sum_{i=1}^{N} x_i\right) \mu - 1 = 0$$

عبارات رنگی بالا به ترتیب Z و R باشند طبق معادلات درجه 2 به رابطه‌ی زیر می‌رسیم:

$$\mu = \frac{Z \pm \sqrt{Z^2 + 4R}}{2R} = \frac{Z}{2R}\left(1 \pm \sqrt{1 + \frac{4R}{Z^2}}\right)$$

سوال چهارم

قسمت الف)

احتمال یافتن ویژگی x_i برای یک بودن در دسته ω_1 با p نشان داده می‌شود:

$$p(x_i = 1|\omega_1) = 1 - p(x_i = 0|\omega_1) = p_{i1} = p > \frac{1}{2}$$

نمونه x از کلاس ω_1 گرفته شده است. بنابراین:

$$p(x|\omega_1) = \prod_{i=1}^{d} p(x_i|\omega_1) = \prod_{i=1}^{d} p^{x_i} (1-p)^{1-(x_i)}$$

تابع log-likelihood برای p به صورت زیر است:

$$\mathrm{l}(p) = \ln p(x|\omega_1) = \sum_{i=1}^{d}[x_i \ \ln p + (1-x_i)\ln(1-p)]$$

مشتق رابطه بالا را حساب کرده و برابر صفر قرار می‌دهیم:

$$\frac{1}{\hat{p}}\sum_{i=1}^{d} x_i - \frac{1}{1-\hat{p}}\sum_{i=1}^{d}(1-x_i) = 0$$

$$\frac{1}{\hat{p}}\sum_{i=1}^{d} x_i = \frac{1}{1-\hat{p}}\sum_{i=1}^{d}(1-x_i) \quad => \quad \frac{1-\hat{p}}{\hat{p}}\sum_{i=1}^{d} x_i = d - \sum_{i=1}^{d} x_i$$

$$d = \frac{1}{\hat{p}}\sum_{i=1}^{d} x_i \quad => \quad \hat{p} = \frac{1}{d}\sum_{i=1}^{d} x_i$$

قسمت ب) با توجه به رابطه‌ی به دست آمده در قسمت الف، اگر واریانس را برای رابطه‌ی حساب کنیم به چنین رابطه‌ای خواهیم رسید: $\frac{p(1-p)}{d}$ که در مخرج کسر d قرار دارد. بنابراین با افزایش d به سمت بی‌نهایت مقدار واریانس برابر صفر خواهد بود و احتمال خطا صفر خواهد شد حتی اگر از هر کلاس فقط یک نمونه داشته باشیم.

قسمت ج)

$$T = \frac{1}{d}\sum_{j=1}^{d} x_j$$

با افزایش d به سمت بی‌نهایت داریم:

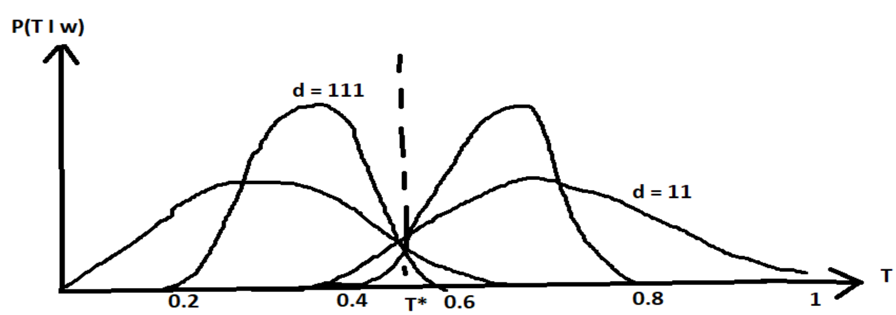
$$T = \frac{1}{d}\sum_{j=1}^{d} \varepsilon(x_i|\omega_1) = [1 * p + 0 * (1-p)] = p$$

با توجه به اینکه فقط یک کلاس را در نظر می‌گیریم واریانس T به صورت زیر است:

$$var(T|\omega_1) = \frac{1}{d}\sum_{j=1}^{d} var(x_j|\omega_1) = \frac{1}{d^2}\sum_{j=1}^{d}[1^2 * p + 0^2 * (1-p) - p * p]$$

$$var(T|\omega_1) = \frac{p(1-p)}{d}$$

طبق رابطه بالا با افزایش d به سمت بی‌نهایت، مقدار واریانس صفر می‌شود بنابراین احتمال خطا صفر می‌شود.



1 Question 5

a: Part A

i. Naive-bayes

naive bayes is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

ii. Optimal-bayes

The Bayes Optimal Classifier is a probabilistic model that makes the most probable prediction for a new example. It is described using the Bayes Theorem that provides a principled way for calculating a conditional probability. It is also closely related to the Maximum a Posteriori: a probabilistic framework referred to as MAP that finds the most probable hypothesis for a training dataset.

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler, scale
from sklearn.metrics import plot_confusion_matrix, accuracy_score,␣
 ↪confusion_matrix, precision_score, recall_score
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
```

[2]:
```
df = pd.read_csv("D:\ML\ML_HW1\Data\Breast_cancer_data.csv")
```

[3]:
```
X, y = df.iloc[:, :-1], df.iloc[:, -1]
```

[4]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
```

[5]:
```
classes,counts = np.unique(y, return_counts=True)
print('Class:',list(classes),'Count:',counts)
```

```
Class: [0, 1] Count: [212 357]
```

[6]:
```
X_train_scaled = scale(X_train)
X_test_scaled = scale(X_test)
```

[7]:
```
#Finding Prior Probability
classes_train,counts_train = np.unique(y_train, return_counts=True)
print('Class:',list(classes_train),'Count:',counts_train)

total_samples=len(y_train)
print(total_samples)
```

```
prior = np.array([ x*1.0/total_samples for x in counts_train ])
```

```
Class: [0, 1] Count: [169 286]
455
```

[8]:
```
#Finding likelihoods assuming gaussian distribution for all.
mean={}
std={}
for i in classes:
    mean[i]=list(np.mean(X_train_scaled[y_train==i],axis=0))
    std[i]=list(np.std(X_train_scaled[y_train==i],axis=0))


def gaussian_distribution(x, mean, std):
    g = np.sqrt(1.0/2 * np.pi* std**2) * np.exp(-((x - mean)**2/(2 * std**2 )))
    return g
```

[9]:
```
def likelihood(sample,mean,std):

    feature_prob=np.zeros((len(sample),1))
    for i in range(len(sample)):
        feature_prob[i]=gaussian_distribution(sample[i],mean[i],std[i])
    return np.prod(feature_prob)
```

[10]:
```
#Putting likelihood together with prior probabilities to calculate posterior␣
↪probabilities.

y_pred=[]
for i in X_test_scaled:

    class_likelihood=np.zeros(len(classes))
    for cls in classes:
        class_likelihood[int(cls)]=likelihood(i,mean[cls],std[cls])
    posterior=np.multiply(class_likelihood,prior)
    max_index=posterior.argmax()
    y_pred.append(max_index)
```

b: Part B

[11]:
```
accuracy=accuracy_score(y_test,y_pred)*100
print(f'accuracy of our model: ',accuracy)
precision = precision_score(y_test,y_pred)
print(f'Precision of our model',precision)
recall = recall_score(y_test,y_pred)
print(f'Recall of our model',recall)
```

```
accuracy of our model:  89.47368421052632
Precision of our model 1.0
Recall of our model 0.8309859154929577
```

[12]:
```
confusion_matrix(y_pred, y_test)
```

[12]:
```
array([[43, 12],
       [ 0, 59]], dtype=int64)
```

i. Analysis.

The number of samples of class one is 357 and class zero is 212. All samples of class one are predicted correctly and this can be because of the more number of samples in class one.

c: Part C

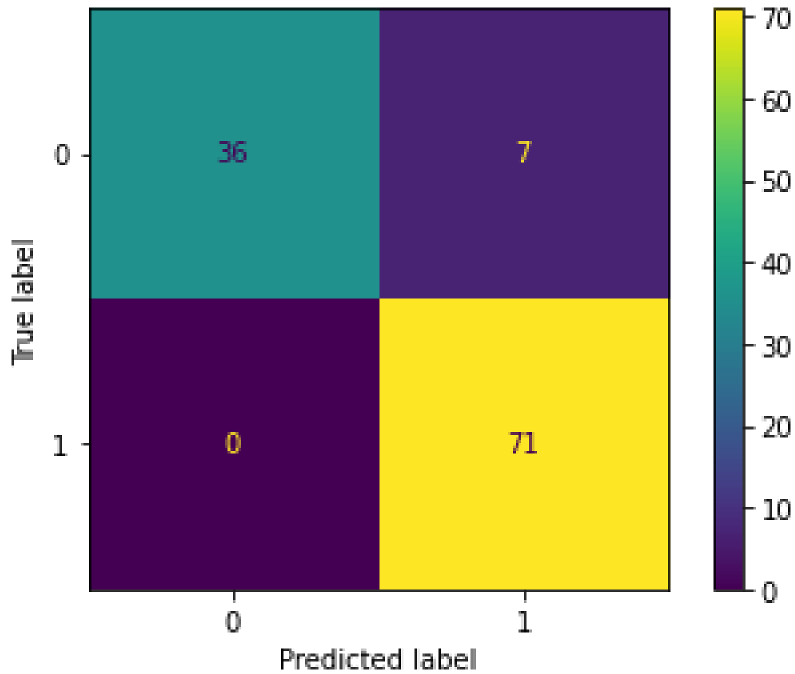
i. Naive Bayes Library

[13]:
```
gnb = GaussianNB()
library_pred = gnb.fit(X_train_scaled, y_train).predict(X_test_scaled)
```

[14]:
```
accuracy=accuracy_score(y_test,library_pred)*100
print(f'accuracy of naive bayes library: ',accuracy)
precision = precision_score(y_test,library_pred)
print(f'Precision of naive bayes library',precision)
recall = recall_score(y_test,library_pred)
print(f'Recall of naive bayes library',recall)
```

```
accuracy of naive bayes library:  93.85964912280701
Precision of naive bayes library 0.9102564102564102
Recall of naive bayes library 1.0
```

[15]:
```
plot_confusion_matrix(gnb, X_test_scaled, y_test)
plt.show()
```



ii. Analysis.

The number of samples of class one is 357 and class zero is 212. The number of samples of class one is 357 and class zero is 212. All samples of class one are predicted correctly and this can be because of the more number of samples in class one. Also, the algorithm library uses other parameters, and for this reason, the accuracy of the algorithm is higher than our model's accuracy.

2 Question 6

[1]:
```
import cv2
import numpy
import matplotlib.pyplot as plt
from os import listdir
```

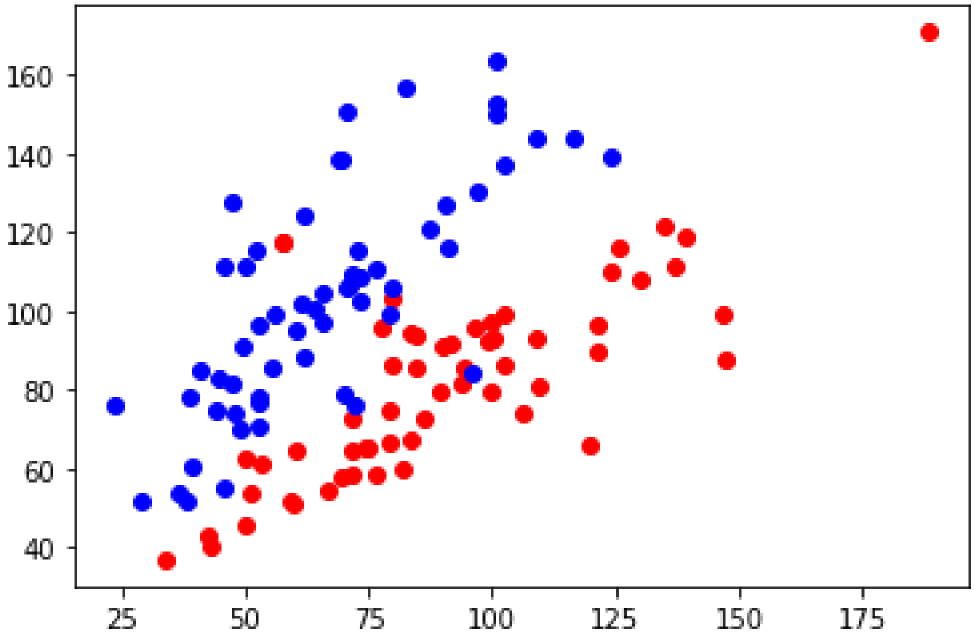
[2]:
```
def read_files(path = "D:\ML\ML_HW1\Data\image/Images/"):
    blue = [path + f for f in listdir(path) if f[0] == "c"]
    red = [path + f for f in listdir(path) if f[0] == "m"]
    return red , blue

#calculate the averege of one image
def average_image_color(img , remove_green = False):
    avg_color_per_row = numpy.average(img, axis=0)
    avg_color = numpy.average(avg_color_per_row, axis=0)
    if remove_green:
        return (avg_color[0],avg_color[2])
    return tuple(avg_color)

#return a list that contain average of images
def get_averages(images:list, return_type = "list" , remove_green = False ):
    averages_list = []
    if return_type == "list":
        for i in images:
            image = cv2.imread(i)
            averages_list.append(average_image_color(image , remove_green))
    return averages_list
```

[3]:
```
blue , red = read_files()
chel_averages = get_averages(blue , remove_green = True)
man_averages = get_averages(red , remove_green = True)
```

[4]:
```
x1,y1 = [],[]
for i in range(len(chel_averages)):
    x1.append(man_averages[i][0])
    y1.append(man_averages[i][1])
x2,y2 = [],[]
for i in range(len(chel_averages)):
    x2.append(chel_averages[i][0])
    y2.append(chel_averages[i][1])
```

[5]:
```
plt.scatter(x1,y1, c='red')
plt.scatter(x2,y2, c='blue')
```

[5]:
```
<matplotlib.collections.PathCollection at 0x1f425b95ee0>
```




```
[6]: red_to_blue_ratio_c = [i[0]/i[1] for i in chel_averages]
     red_to_blue_ratio_m = [i[0]/i[1] for i in man_averages]
     red_to_blue_ratio_all = red_to_blue_ratio_c + red_to_blue_ratio_m
```

```
[7]: #0.8924657649531078
     average_red_blue_ratio = sum(red_to_blue_ratio_all)/len(red_to_blue_ratio_all)

     true_chelsea =  [i for i in red_to_blue_ratio_c if i < average_red_blue_ratio]
     false_man =  [i for i in red_to_blue_ratio_c if i > average_red_blue_ratio]

     true_man =  [i for i in red_to_blue_ratio_m if i > average_red_blue_ratio]
     false_chelsea =  [i for i in red_to_blue_ratio_m if i < average_red_blue_ratio]
```

```
[8]: TP = len(true_chelsea)
     FP = len(false_chelsea)

     TN = len(true_man)
     FN = len(false_man)
```

```
[9]: Accuracy = (TP + TN) / (TP + TN + FP + FN)
     Precision = TP / (TP + FP)
     Recall = TP/ (TP + FN)
```

```
[10]: print('Accuracy of our model', Accuracy)
      print('Precision of our model', Precision)
      print('Recall of our model', Recall)
```

```
Accuracy of our model 0.9098360655737705
Precision of our model 0.873015873015873
Recall of our model 0.9482758620689655
```