6. Naive Bayes

March 28, 2022

[1]:
```
import numpy as np
import pandas as pd
```

[2]:
```
gender_features = ['gender','height','weight','foot_size']
gender_data = [
    ['male',6,180,12],
    ['male',5.92,190,11],
    ['male',5.58,170,12],
    ['male',5.92,165,10],
    ['female',5,100,6],
    ['female',5.5,150,8],
    ['female',5.42,130,7],
    ['female',5.75,150,9]
]
```

[3]:
```
data = pd.DataFrame(gender_data, columns = gender_features)
data.head()
```

[3]:

	gender	height	weight	foot_size
0	male	6.00	180	12
1	male	5.92	190	11
2	male	5.58	170	12
3	male	5.92	165	10
4	female	5.00	100	6

[4]:
```
mean_list=[]    # list of mean of all features for each class label
var_list=[]     # list of variance of all features for each class label
print(set(data['gender']))   # list of class labels

for x in set(data['gender']):           # for each class label
    df = data.loc[data['gender']==x]    # select rows with given label
    df = df.drop(['gender'],axis=1)     # drop the gender feature

    m = df.mean()                       # get mean of each feature
    mean_list.append(m)

    v=df.var()                          # get variance of each feature
```

```
    var_list.append(v)

mean_list=np.array(mean_list)
var_list=np.array(var_list)
print(mean_list)
print(var_list)
```

```
{'male', 'female'}
[[  5.855  176.25    11.25  ]
 [  5.4175 132.5      7.5   ]]
[[3.50333333e-02 1.22916667e+02 9.16666667e-01]
 [9.72250000e-02 5.58333333e+02 1.66666667e+00]]
```

[5]:
```
X_train = np.array(data.iloc[:,[1,2,3]])    # features for training
y_train = np.array(data['gender'])          # labels for training

# find frequency and prior probability of each label in y_train
pre_prob=[]
for x in set(y_train):                      # for each label
    freq=0
    for y in y_train:                       # parse through each label
        if(y==x):
            freq+=1
    pre_prob.append(freq/y_train.shape[0])  # return probability
print(pre_prob)                             # probability of each feature

for y in y_train:          # encode male=0 and female=1
    if y == "male":
        y = 0
    else:
        y = 1
```

```
[0.5, 0.5]
```

[6]:
```
X_test = np.array([6, 130, 8])   # test data point

# function to calculate Gaussian prior probability
def prob_feature_class(m, v, x):
    n_features = m.shape[1]
    pfc = np.ones(2)
    for i in range(0, 2):                # for each label
        product = 1
        # for each feature of test data point i.e. x[j]
        for j in range(0, n_features):
            product = product * (1/np.sqrt(2*3.14*v[i][j])) * \
                        np.exp(-0.5 * pow((x[j] - m[i][j]),2)/v[i][j])
        pfc[i] = product
```

```
    return pfc

def GNB(X, y, x):
    # calculate Gaussian probability for test data point
    # P(m,v|X) = P(height=6|X)*P(weight=130|X)*P(foot_size=8|X)
    pfc = prob_feature_class(mean_list, var_list, x)
    print(pfc)

    pcf = np.ones(2)
    total_prob = 0
    for i in range(0, 2):
        # sum of P(X)*P(m,v|X) for both labels
        total_prob = total_prob + (pfc[i] * pre_prob[i])

    for i in range(0, 2):
        # normalize P(X)*P(m,v|X) with total sum
        pcf[i] = (pfc[i] * pre_prob[i])/total_prob

    prediction = int(pcf.argmax())    # return class with max pcf
    return prediction

prediction = GNB(X_train, y_train, X_test)
```

```
[1.24035746e-08 1.07640027e-03]
```

[7]:
```
print("Prediction (0:male, 1:female) = ",prediction)
```

```
Prediction (0:male, 1:female) =  1
```