niaveBayes

September 25, 2021

[1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import warnings
import seaborn as sns
warnings.filterwarnings('ignore')

from IPython.display import set_matplotlib_formats
set_matplotlib_formats('pdf', 'svg')

from pandas import DataFrame
from numpy import arange
from pandas import read_csv
from matplotlib import pyplot

from scipy.stats import multivariate_normal as mvn

data=pd.read_csv('/content/drive/MyDrive/exNB.csv', header=None)

#print(data.head())

#print(data.describe())

X = data.to_numpy()

#print(X)

y= X[:,-1]

X=X[:,:-1]

#print(X)
#print(y)

#print(X[y==1,:])
```

```
#plt.figure()
#plt.hist(X[y==1,0], label="Male", alpha=.5)
#plt.hist(X[y==0,0], label="Male", alpha=.5)

#plt.legend()

#plt.figure()

#plt.hist(X[y==1,1], label="Male", alpha=.5)
#plt.hist(X[y==0,1], label="Male", alpha=.5)
#plt.legend()

#plt.figure()

#plt.figure()
#plt.scatter(X[:,0],X[:,1], c=y, alpha=.25)
```

[1]:

```

```

[2]:

```
class GaussNB():
  def fit(self , X, y, epsilon= 1e-1):
    self.likelihoods = dict()
    self.priors = dict()
    self.K = set(y.astype(int))

    for k in self.K:

      X_k = X[y==k,:]

      self.likelihoods[k]={"mean": X_k.mean(axis=0), "cov":X_k.var(axis=0) +␣
 ↪epsilon}
      self.priors[k]=len(X_k)/len(X)

  def predict(self, X):
    N, D = X.shape
    P_hat = np.zeros((N, len(self.K)))

    for k, l in self.likelihoods.items():
      P_hat[:,k] = mvn.logpdf(X, l["mean"], l["cov"]) + np.log(self.priors[k])

    return P_hat.argmax(axis=1)
```

[3]:

```
gnb = GaussNB()

gnb.fit(X,y)

y_hat = gnb.predict(X)
```

```
#plt.figure()
#plt.scatter(X[:,0],X[:,1], c=y_hat, alpha=.25)
```

[4]:
```
def accurarcy(y,y_hat):
  return np.mean(y==y_hat)
```

[5]:
```
accurarcy(y,y_hat)
```

[5]:
```
0.988
```

[5]:
```

```

[6]:
```
X_new=np.asarray([[68,150]])
```

[7]:
```
yh_new=gnb.predict(X_new)
```

[8]:
```
accurarcy(y,yh_new)
```

[8]:
```
0.5
```

[9]:
```
data1= pd.read_csv('/content/drive/MyDrive/xor.csv')
data2= pd.read_csv('/content/drive/MyDrive/MNIST_train.csv')
data2.shape
```

[9]:
```
(60000, 787)
```

[10]:
```
#data2=data2.fillna(data2.mean())
```

[11]:
```
##################start messing with MNIST#############
data2=data2.iloc[:, 2:]



##############Reciprcal Transform works if inf taken␣
 ↪away########################
#data2=1/(data2)
#data2['labels']=1/(data2['labels'])



data2=data2/255
data2['labels']=data2['labels']*255

##################################Square root doesnt work at all:
 ↪ERROR#################################
#data2=pow(data2,1/2)
#data2['labels']=pow(data2['labels'],2)
#data2['labels']=data2['labels'].astype('category')

#data2=data2.replace(np.inf, 0)
#data2=data2.replace(np.nan, 0)

print(data2.shape)
data2.head(10)
```

```
(60000, 785)
```

[11]:
```
   labels    0    1    2    3    4    5  ...  777  778  779  780  781  782  783
0     5.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0
1     0.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0
2     4.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0
3     1.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0
4     9.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0
5     2.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0
6     1.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0
7     3.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0
8     1.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0
9     4.0  0.0  0.0  0.0  0.0  0.0  0.0  ...  0.0  0.0  0.0  0.0  0.0  0.0  0.0

[10 rows x 785 columns]
```

[12]:
```python
x2=data2.to_numpy()
```

[12]:
```python

```

[13]:
```python
y2 = x2[:, 0]
print(y2)
x2= x2[:, 1:]

print(x2.shape)
```

```
[5. 0. 4. ... 5. 6. 8.]
(60000, 784)
```

[14]:
```python
#plt.figure()
#plt.scatter(x2[:,0],x2[:,1], c=y2, alpha=.25)
```

[15]:
```python
xgnb = GaussNB()

xgnb.fit(x2,y2)

y_hat1 = xgnb.predict(x2)

#plt.figure()
#plt.scatter(x2[:,0],x2[:,1], c=y_hat1, alpha=.25)
```

[16]:
```python
accurarcy(y2,y_hat1)
```

[16]:
```
0.7651333333333333
```

[17]:
```python
class GaussBayes():
  def fit(self, X, y, epsilon=1e-1):
    self.likelihoods=dict()
    self.priors= dict()

    self.K = set(y.astype(int))
  for k in self.K:
   X_k = X[y==k,:]
   N_k, D = X_k.shape
   mu_k=X_k.mean(axis=0)
   self.likelihoods[k] = {"mean":X_k.mean(axis=0), "cov":(1/(N_k-1))*np.
↪matmul((X_k-mu_k).T, X_k - mu_k) +epsilon*np.identity(D)}

   self.priors[k]= len(X_k)/len(X)

 def predict(self, X):
  N,D=X.shape
  P_hat = np.zeros((N,len(self.K)))

  for k,l in self.likelihoods.items():
   P_hat[:,k] = mvn.logpdf(X, l["mean"],l["cov"]) + np.log(self.priors[k])

  return P_hat.argmax(axis=1)
```

[18]:
```
gbayes = GaussBayes()
```

[19]:
```
gbayes.fit(x2,y2)
```

[20]:
```
y_hat_GB=gbayes.predict(x2)
```

[21]:
```
#plt.figure()
#plt.scatter(x2[:,0],x2[:,1], c=y_hat_GB, alpha=.25)
```

[22]:
```
accurarcy(y2,y_hat_GB)
```

[22]:
```
0.9549333333333333
```

[23]:
```
###########################Run Test Data###############################
data_test= pd.read_csv('/content/drive/MyDrive/MNIST_test.csv')
```

[24]:
```
data_test=data_test.iloc[:, 2:]
data_test.head()
```

[24]:
```
   labels  0  1  2  3  4  5  6  7  ...  775  776  777  778  779  780  781  782
783
0       7  0  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0
0
1       2  0  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0
0
2       1  0  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0
0
3       0  0  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0
0
4       4  0  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0
0
```

```
[5 rows x 785 columns]
```

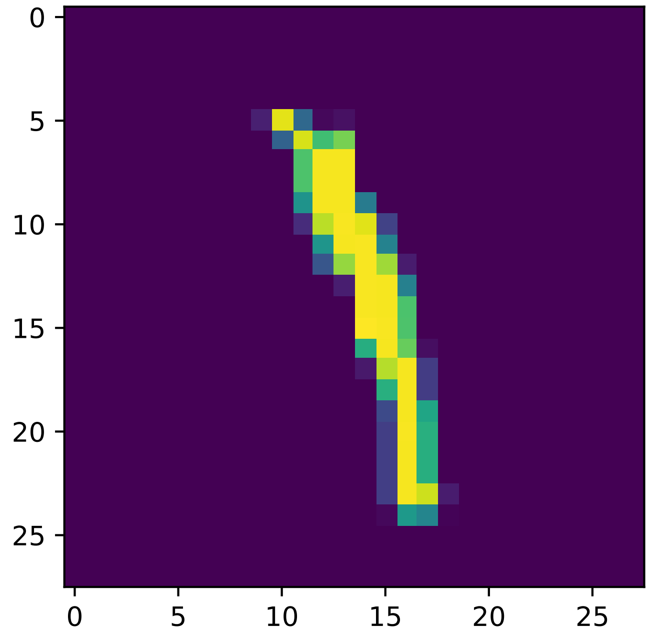
[25]:
```
X3=data_test.to_numpy()
```

[26]:
```
def show_me(X):
    plt.imshow(X.reshape(28,28))

def show_me_allmean(X,y,k):
    show_me(sum(X[y==k,:]/len(X[y==k,:])))
```

[27]:
```
y3 = X3[:, 0]
print(y3)
X3= X3[:, 1:]

print(X3.shape)
```

```
[7 2 1 ... 4 5 6]
(10000, 784)
```

[28]:
```
show_me(X3[96])
```



[29]:
```
data_test.head(10)
```

[29]:
```
   labels  0  1  2  3  4  5  6  7  ...  775  776  777  778  779  780  781  782
783
0       7  0  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0
0
1       2  0  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0
```

```
0
2        1  0  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0
0
3        0  0  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0
0
4        4  0  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0
0
5        1  0  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0
0
6        4  0  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0
0
7        9  0  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0
0
8        5  0  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0
0
9        9  0  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0
0

[10 rows x 785 columns]
```

[30]:
```
data_test.tail(10)
```

[30]:
```
      labels  0  1  2  3  4  5  6  ...  776  777  778  779  780  781  782  783
9990       7  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0
9991       8  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0
9992       9  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0
9993       0  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0
9994       1  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0
9995       2  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0
9996       3  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0
9997       4  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0
9998       5  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0
9999       6  0  0  0  0  0  0  0  ...    0    0    0    0    0    0    0    0

[10 rows x 785 columns]
```

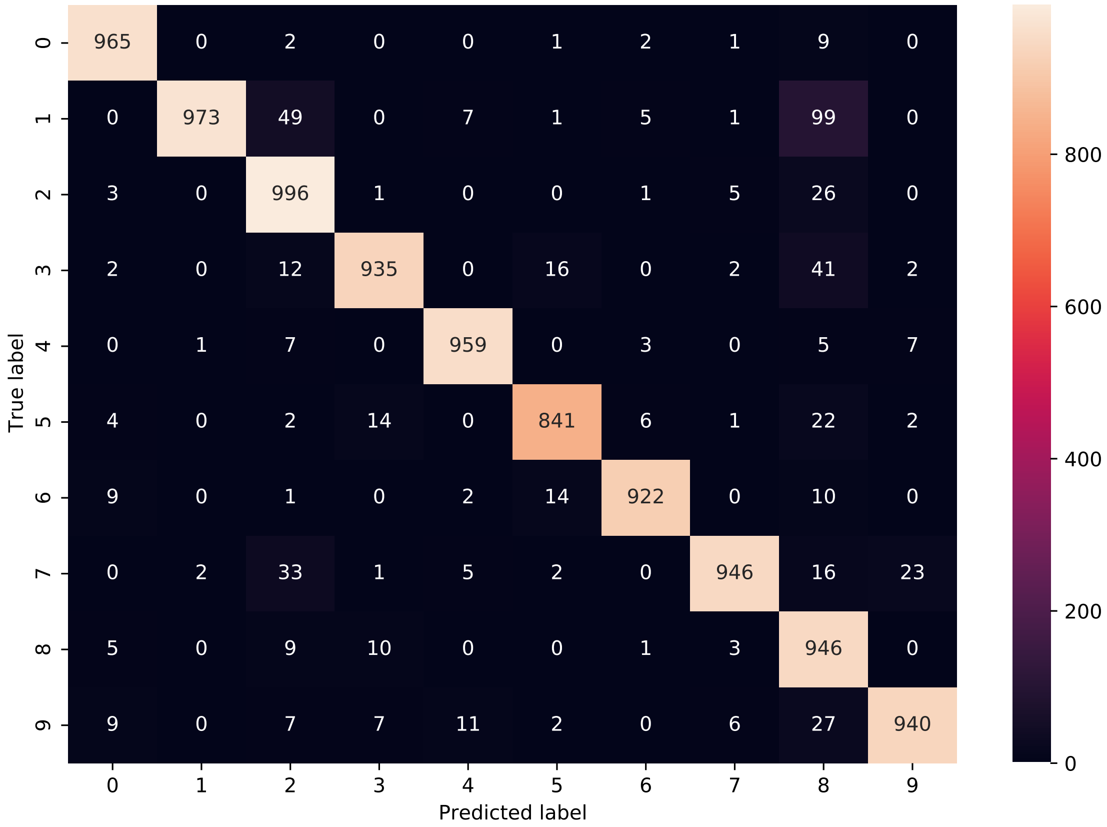
[31]:
```
y_hat_GB2=gbayes.predict(X3)
```

[32]:
```
accurarcy(y3,y_hat_GB2)
```

[32]:
```
0.9423
```

[33]:
```
plt.figure(figsize=(10,7))
y_actu = pd.Series(y3, name='Actual')
y_pred = pd.Series(y_hat_GB2, name='Predicted')
cm = pd.crosstab(y_actu, y_pred)
ax = sns.heatmap(cm, annot=True, fmt="d")
plt.ylabel('True label')
plt.xlabel('Predicted label')
```

[33]:
```
Text(0.5, 42.0, 'Predicted label')
```



	0	1	2	3	4	5	6	7	8	9
0	965	0	2	0	0	1	2	1	9	0
1	0	973	49	0	7	1	5	1	99	0
2	3	0	996	1	0	0	1	5	26	0
3	2	0	12	935	0	16	0	2	41	2
4	0	1	7	0	959	0	3	0	5	7
5	4	0	2	14	0	841	6	1	22	2
6	9	0	1	0	2	14	922	0	10	0
7	0	2	33	1	5	2	0	946	16	23
8	5	0	9	10	0	0	1	3	946	0
9	9	0	7	7	11	2	0	6	27	940

[34]:
```
isMatch= y_actu==y_pred
isMatch
```

[34]:
```
0       True
1       True
2       True
3       True
4       True
        ...
9995    True
9996    True
9997    True
9998    True
9999    True
Length: 10000, dtype: bool
```

[35]:
```
misLabel = pd.concat([y_actu, y_pred,isMatch], axis=1)
misLabel.to_csv('isMatch.csv')
```

[36]:

```
%%capture
!jupyter nbconvert --to PDF "/content/drive/MyDrive/Colab Notebooks/niaveBayes.
 ↪ipynb"
```