assignment-2

September 3, 2024

[39]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[40]:
```
mnsit_train = pd.read_csv('/content/drive/MyDrive/Tech Consulting/Assignment 2/
↪MNIST_train.csv')
mnsit_test = pd.read_csv('/content/drive/MyDrive/Tech Consulting/Assignment 2/
↪MNIST_test.csv')
```

[41]:
```
mnsit_train
```

[41]:

	Unnamed: 0	index	labels	0	1	2	3	4	5	6	…	774	775	776	\
0	0	0	5	0	0	0	0	0	0	0	…	0	0	0	
1	1	1	0	0	0	0	0	0	0	0	…	0	0	0	
2	2	2	4	0	0	0	0	0	0	0	…	0	0	0	
3	3	3	1	0	0	0	0	0	0	0	…	0	0	0	
4	4	4	9	0	0	0	0	0	0	0	…	0	0	0	
…	…	…	…	..	..	..	..	..	..	..	…	…	…	…	
59995	59995	59995	8	0	0	0	0	0	0	0	…	0	0	0	
59996	59996	59996	3	0	0	0	0	0	0	0	…	0	0	0	
59997	59997	59997	5	0	0	0	0	0	0	0	…	0	0	0	
59998	59998	59998	6	0	0	0	0	0	0	0	…	0	0	0	
59999	59999	59999	8	0	0	0	0	0	0	0	…	0	0	0	

	777	778	779	780	781	782	783
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0
…	…	…	…	…	…	…	…
59995	0	0	0	0	0	0	0
59996	0	0	0	0	0	0	0
59997	0	0	0	0	0	0	0
59998	0	0	0	0	0	0	0
59999	0	0	0	0	0	0	0

```
[60000 rows x 787 columns]
```

[42]:
```
mnsit_test
```

[42]:
```
      Unnamed: 0  index  labels  0  1  2  3  4  5  6  …  774  775  776  777  \
0              0      0       7  0  0  0  0  0  0  0  …    0    0    0    0
1              1      1       2  0  0  0  0  0  0  0  …    0    0    0    0
2              2      2       1  0  0  0  0  0  0  0  …    0    0    0    0
3              3      3       0  0  0  0  0  0  0  0  …    0    0    0    0
4              4      4       4  0  0  0  0  0  0  0  …    0    0    0    0
…            …    …     … .. .. .. .. .. .. ..  …  …  …  …  …
9995        9995   9995       2  0  0  0  0  0  0  0  …    0    0    0    0
9996        9996   9996       3  0  0  0  0  0  0  0  …    0    0    0    0
9997        9997   9997       4  0  0  0  0  0  0  0  …    0    0    0    0
9998        9998   9998       5  0  0  0  0  0  0  0  …    0    0    0    0
9999        9999   9999       6  0  0  0  0  0  0  0  …    0    0    0    0

      778  779  780  781  782  783
0       0    0    0    0    0    0
1       0    0    0    0    0    0
2       0    0    0    0    0    0
3       0    0    0    0    0    0
4       0    0    0    0    0    0
…   …  …  …  …  …  …  …
9995    0    0    0    0    0    0
9996    0    0    0    0    0    0
9997    0    0    0    0    0    0
9998    0    0    0    0    0    0
9999    0    0    0    0    0    0

[10000 rows x 787 columns]
```

[43]:
```
mnsit_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 60000 entries, 0 to 59999
Columns: 787 entries, Unnamed: 0 to 783
dtypes: int64(787)
memory usage: 360.3 MB
```

[44]:
```
mnsit_train.describe()
```

[44]:
```
         Unnamed: 0         index        labels        0        1        2  \
count  60000.000000  60000.000000  60000.000000  60000.0  60000.0  60000.0
mean   29999.500000  29999.500000      4.453933      0.0      0.0      0.0
std    17320.652413  17320.652413      2.889270      0.0      0.0      0.0
min        0.000000      0.000000      0.000000      0.0      0.0      0.0
```

```
25%    14999.750000  14999.750000      2.000000      0.0      0.0      0.0
50%    29999.500000  29999.500000      4.000000      0.0      0.0      0.0
75%    44999.250000  44999.250000      7.000000      0.0      0.0      0.0
max    59999.000000  59999.000000      9.000000      0.0      0.0      0.0

              3        4        5        6  …           774           775  \
count   60000.0  60000.0  60000.0  60000.0  …  60000.000000  60000.000000
mean        0.0      0.0      0.0      0.0  …      0.200433      0.088867
std         0.0      0.0      0.0      0.0  …      6.042472      3.956189
min         0.0      0.0      0.0      0.0  …      0.000000      0.000000
25%         0.0      0.0      0.0      0.0  …      0.000000      0.000000
50%         0.0      0.0      0.0      0.0  …      0.000000      0.000000
75%         0.0      0.0      0.0      0.0  …      0.000000      0.000000
max         0.0      0.0      0.0      0.0  …    254.000000    254.000000

                776           777           778         779      780      781  \
count  60000.000000  60000.000000  60000.000000  60000.0000  60000.0  60000.0
mean       0.045633      0.019283      0.015117      0.0020      0.0      0.0
std        2.839845      1.686770      1.678283      0.3466      0.0      0.0
min        0.000000      0.000000      0.000000      0.0000      0.0      0.0
25%        0.000000      0.000000      0.000000      0.0000      0.0      0.0
50%        0.000000      0.000000      0.000000      0.0000      0.0      0.0
75%        0.000000      0.000000      0.000000      0.0000      0.0      0.0
max      253.000000    253.000000    254.000000     62.0000      0.0      0.0

           782      783
count  60000.0  60000.0
mean       0.0      0.0
std        0.0      0.0
min        0.0      0.0
25%        0.0      0.0
50%        0.0      0.0
75%        0.0      0.0
max        0.0      0.0

[8 rows x 787 columns]
```

[45]:
```
mnsit_train.isnull().sum()
```

[45]:
```
Unnamed: 0    0
index         0
labels        0
0             0
1             0
             ..
779           0
780           0
```

```
781           0
782           0
783           0
Length: 787, dtype: int64
```

[46]:
```
mnsit_train.duplicated().sum()
```

[46]:
```
0
```

[47]:
```
mnsit_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Columns: 787 entries, Unnamed: 0 to 783
dtypes: int64(787)
memory usage: 60.0 MB
```

[48]:
```
mnsit_test.describe()
```

[48]:

	Unnamed: 0	index	labels	0	1	2	\
count	10000.00000	10000.00000	10000.000000	10000.0	10000.0	10000.0	
mean	4999.50000	4999.50000	4.443400	0.0	0.0	0.0	
std	2886.89568	2886.89568	2.895865	0.0	0.0	0.0	
min	0.00000	0.00000	0.000000	0.0	0.0	0.0	
25%	2499.75000	2499.75000	2.000000	0.0	0.0	0.0	
50%	4999.50000	4999.50000	4.000000	0.0	0.0	0.0	
75%	7499.25000	7499.25000	7.000000	0.0	0.0	0.0	
max	9999.00000	9999.00000	9.000000	0.0	0.0	0.0	

	3	4	5	6	…	774	775	\
count	10000.0	10000.0	10000.0	10000.0	…	10000.000000	10000.000000	
mean	0.0	0.0	0.0	0.0	…	0.179300	0.163600	
std	0.0	0.0	0.0	0.0	…	5.674149	5.736072	
min	0.0	0.0	0.0	0.0	…	0.000000	0.000000	
25%	0.0	0.0	0.0	0.0	…	0.000000	0.000000	
50%	0.0	0.0	0.0	0.0	…	0.000000	0.000000	
75%	0.0	0.0	0.0	0.0	…	0.000000	0.000000	
max	0.0	0.0	0.0	0.0	…	253.000000	253.000000	

	776	777	778	779	780	781	782	\
count	10000.000000	10000.0000	10000.0	10000.0	10000.0	10000.0	10000.0	
mean	0.052600	0.0006	0.0	0.0	0.0	0.0	0.0	
std	2.420004	0.0600	0.0	0.0	0.0	0.0	0.0	
min	0.000000	0.0000	0.0	0.0	0.0	0.0	0.0	
25%	0.000000	0.0000	0.0	0.0	0.0	0.0	0.0	
50%	0.000000	0.0000	0.0	0.0	0.0	0.0	0.0	
75%	0.000000	0.0000	0.0	0.0	0.0	0.0	0.0	

```
max      156.000000      6.0000      0.0      0.0      0.0      0.0      0.0
```

```
           783
count  10000.0
mean       0.0
std        0.0
min        0.0
25%        0.0
50%        0.0
75%        0.0
max        0.0

[8 rows x 787 columns]
```

[49]:
```
mnsit_test.isnull().sum()
```

[49]:
```
Unnamed: 0    0
index         0
labels        0
0             0
1             0
             ..
779           0
780           0
781           0
782           0
783           0
Length: 787, dtype: int64
```

[50]:
```
mnsit_test.duplicated().sum()
```

[50]:
```
0
```

[51]:
```
X_train = mnsit_train.to_numpy()
X_test = mnsit_test.to_numpy()
```

[52]:
```
y_train = X_train[:,2]
y_test = X_test[:,2]
```

[53]:
```
X_train = X_train[:,3:]
X_test = X_test[:,3:]
X_train.shape
```

[53]:
```
(60000, 784)
```

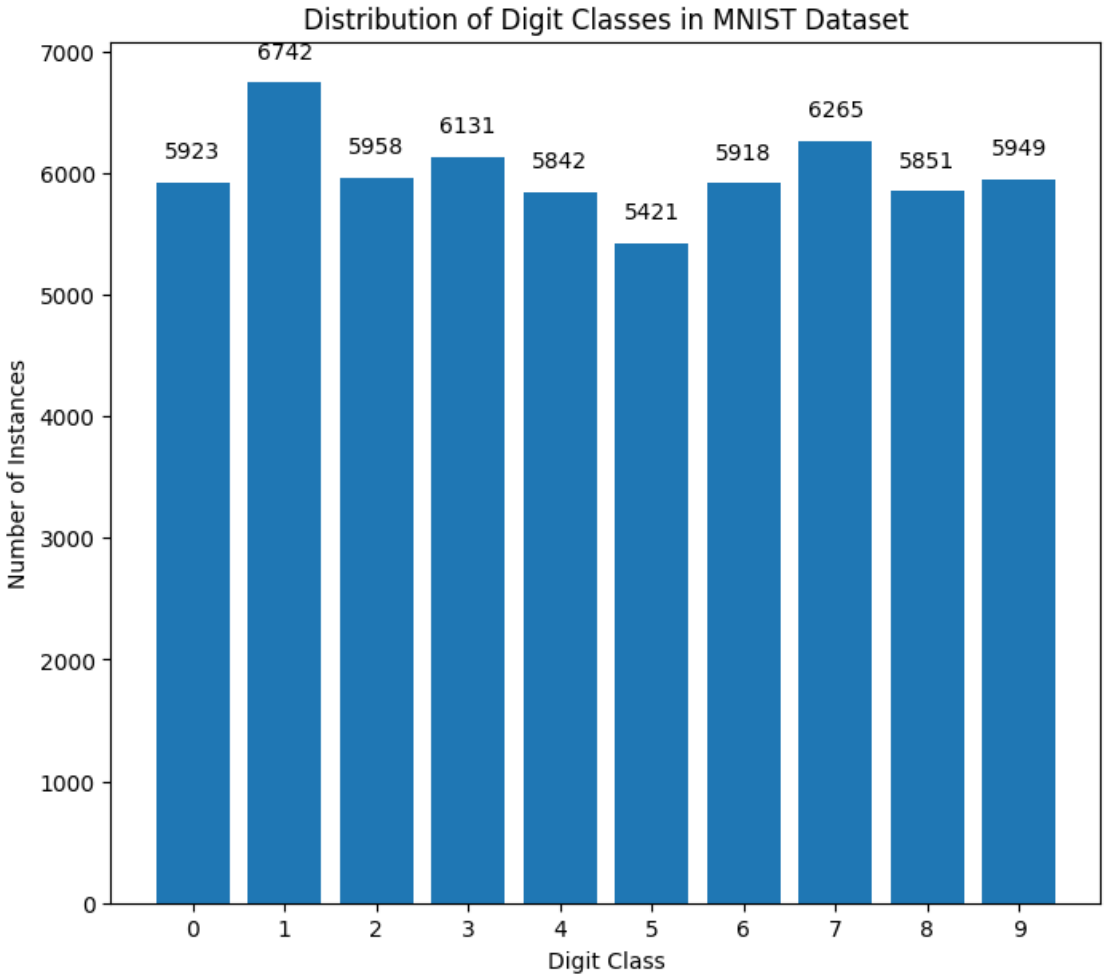
[54]:
```
unique, counts = np.unique(y_train, return_counts=True)
plt.figure(figsize=(8, 7))
```

```
plt.bar(unique, counts)

for i in range(len(unique)):
    plt.text(unique[i], counts[i] + 200, str(counts[i]), ha='center')

plt.xticks(ticks=unique, labels=[str(x) for x in unique])
plt.xlabel('Digit Class')
plt.ylabel('Number of Instances')
plt.title('Distribution of Digit Classes in MNIST Dataset')
```

[54]: `Text(0.5, 1.0, 'Distribution of Digit Classes in MNIST Dataset')`



[55]:
```
# Normalize the data
X_train = X_train / 255.0
X_test = X_test / 255.0
```

[56]:
```
from scipy.stats import multivariate_normal as mvn
```

[57]:
```
class GaussNB():
    def fit(self, X, y, epsilon = 1e-3):
        self.likelihoods = dict()
        self.priors = dict()
        self.K = set(y.astype(int))

        for k in self.K:
            X_k = X[y==k]
            # Naive assumption: Obseravtions are linearly independent of each␣
 ↪other
            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)
```

[58]:
```
gnb = GaussNB()
gnb.fit(X_train,y_train)
y_hat= gnb.predict(X_test)
```

[59]:
```
def accuracy(y, y_hat):
    return np.mean(y==y_hat)
```

[62]:
```
accuracy(y_test,y_hat)
```

[62]: 0.7746

[65]:
```
class GaussBayes():
    def fit(self, X, y, epsilon = 1e-3):
        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)
```

[66]:
```
gaussbayes = GaussBayes()
gaussbayes.fit(X_train,y_train, epsilon=1e-3)
y_hat_bayes = gaussbayes.predict(X_test)
```

[67]:
```
accuracy(y_test,y_hat_bayes)
```

[67]: 0.9108

[77]:
```
class KNNClassifier():
  def fit(self, X, y):
    self.X = X
    self.y = y

  def predict(self, X, K, epsilon=1e-3):
    N= len(X)
    y_hat = np.zeros(N)

    for i in range(N):
      dist2 = np.sum((self.X-X[i])**2, axis=1)
      idxt = np.argsort(dist2)[:K]
      gamma_k = 1/(np.sqrt(dist2[idxt]+epsilon))

      y_hat[i] = np.bincount(self.y[idxt], weights=gamma_k).argmax()

    return y_hat
```

[71]:
```
knn = KNNClassifier()
knn.fit(X_train,y_train)
y_hat_knn = knn.predict(X_test, K=10)
```

[72]:
```
accuracy(y_test,y_hat_knn)
```

[72]: 0.9684