Advanced Data Analysis
Homework Week 4

Aswin Vijay

May 15, 2023

Homework 4

Given the least squares classification objective as,

$$\hat{\theta} = arg\min_{\theta} \frac{1}{2}\sum_{i=1}^{n}(f_\theta(x_i) - y_i)^2$$

with mean zero inputs,

$$\frac{1}{n}\sum_{i=1}^{x_i} = 0 \tag{1}$$

and linear input model,

$$f_\theta(x) = \theta^T x$$

Given the solution to the least squares objective as,

$$(X^T X)\hat{\theta} = X^T y \tag{2}$$

The class means and variance are given as,

$$\mu_- = \frac{1}{n_-}\sum_{i:y_i=-1} x_i$$

$$\mu_+ = \frac{1}{n_+}\sum_{i:y_i=1} x_i$$

$$\hat{\Sigma}_- = \frac{1}{n_-}\sum_{i:y_i=-1} (x_i - \mu_-)(x_i - \mu_-)^T$$

$$\hat{\Sigma}_+ = \frac{1}{n_+}\sum_{i:y_i=1} (x_i - \mu_+)(x_i - \mu_+)^T$$

$$\hat{\Sigma} = \frac{1}{n}\left(n_+\hat{\Sigma}_+ + n_-\hat{\Sigma}_-\right)$$

where $\hat{\Sigma}$ is the MLE of the common covariance matrix.

Next we try to express Eqn (1) and (2) in terms of the means and covariances.

$$
\begin{aligned}
\frac{1}{n}\sum_{i=1}^{x_i} &= 0 \\
\frac{1}{n}(n_-\mu_- + n_+\mu_+) &= 0 \\
\mu_- &= -\frac{n_+\mu_+}{n_-}
\end{aligned}
$$

Let the y labels be $\{\frac{-1}{n_-}, \frac{1}{n_+}\}$, then for RHS of (2) we have

$$
\begin{aligned}
X^T y &= -\frac{1}{n_-}\sum_{i:y_i=-1} x_i + \frac{1}{n_+}\sum_{i:y_i=1} x_i \\
&= \mu_+ - \mu_- \\
&= \mu_+ \frac{n}{n_-}
\end{aligned}
$$

For LHS of (2) we have,

$$
\begin{aligned}
\hat{\Sigma} &= \frac{1}{n}\left(\sum_{i:y_i=-1}(x_i-\mu_-)(x_i-\mu_-)^T + \sum_{i:y_i=1}(x_i-\mu_+)(x_i-\mu_+)^T\right) \\
&= \frac{1}{n}\left(\sum_{i:y_i=-1}(x_i x_i^T - x_i\mu_-^T - \mu_- x_i^T + \mu_-\mu_-^T) + \sum_{i:y_i=1}(x_i x_i^T - x_i\mu_+^T - \mu_+ x_i^T + \mu_+\mu_+^T)\right) \\
&= \frac{1}{n}\left(X^T X + -n_-\mu_-^T\mu_- - n_+\mu_+^T\mu_+\right)
\end{aligned}
$$

$$
\begin{aligned}
X^T X &= n\hat{\Sigma} + n_-\mu_-^T\mu_- + n_+\mu_+^T\mu_+ \\
X^T X &= n\hat{\Sigma} + \frac{n^2}{n_-}\mu_+\mu_+^T + n_+\mu_+\mu_+^T
\end{aligned}
$$

Using the above results in Eqn (2) we get,

$$\begin{aligned}
\left[n\hat{\Sigma} + \frac{n^2}{n_-}\mu_+\mu_+^T + n_+\mu_+\mu_+^T\right]\hat{\theta} &= \mu_+\frac{n}{n_-} \\
\left[\hat{\Sigma} + (\frac{n}{n_-} + n_+)\mu_+\mu_+^T\right]\hat{\theta} &= \mu_+\frac{1}{n_-} \\
\hat{\Sigma}\hat{\theta} + (\frac{n}{n_-} + n_+)c\mu_+ &= \mu_+\frac{1}{n_-} \quad using\ vv^T\theta = cv \\
\hat{\Sigma}\cdot\hat{\theta} &= \mu_+(\frac{1}{n_-} - c(\frac{n}{n_-} + n_+)) \\
\hat{\Sigma}\cdot\hat{\theta} &= (\mu_+ - \mu_-)\frac{n_-}{n}(\frac{1}{n_-} - c(\frac{n}{n_-} + n_+)) \\
\hat{\Sigma}\cdot\hat{\theta} &= (\mu_+ - \mu_-)(\frac{1}{n} - c(1 + \frac{n_+n_-}{n})) \\
\hat{\theta} \propto \hat{\Sigma}^{-1}(\mu_+ - \mu_-)
\end{aligned}$$

Thus we get he desired result

In [ ]:
```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.io import loadmat
```

Load Data

In [ ]:
```python
data = loadmat('ADA4-digit.mat')
train = data['X']
test = data['T']

print("Train data: {}".format(train.shape))
print("Test data:  {}".format(test.shape))
```

```
Train data: (256, 500, 10)
Test data:  (256, 200, 10)
```

Multi-classifier

In [ ]:
```python
mu1 = []
mu2 = []
invS = []

# store mean and variances for each class
for i in range(10):

    train_one = train[:, :, i]
    train_all = np.delete(train,i,axis=2).reshape(256,-1,order="A")

    # Make a classifier for one and all
    mu1.append(np.mean(train_one, axis=1))
    mu2.append(np.mean(train_all, axis=1))
    S = (np.cov(train_one) + np.cov(train_all)) / 2
    invS.append(np.linalg.inv(S + 0.000001 * np.identity(256)))

C = np.zeros((10,10))

# Evaluate test cases using learnt data.
for i in range(10):
    t = test[:, :, i]
    pnet = []
    print(f"Test digit {i+1}")
    for j in range(10):
        p1 = mu1[j][None, :].dot(invS[j]).dot(t) - mu1[j][None, :].dot(invS[j]).do
        p2 = mu2[j][None, :].dot(invS[j]).dot(t) - mu2[j][None, :].dot(invS[j]).do
        pnet.append((p1 - p2)[0])

    pnet = np.dstack(pnet)[0]
    # Select class having highest p1-p2.
    result = np.argmax(pnet, axis=1)

    for d in result: C[i,d]+=1
    print(f"The number of correct prediction: Digit - {i+1} : {np.sum(result == i)
    print(f"The number of false prediction: not Digit - {i+1}: {np.sum(result != i

print("One vs All Prediction")
print(C)
acc = np.trace(C)*100/(test.shape[1]*test.shape[2])
print(f"Accuracy is:{acc}%")
```

```
Test digit 1
The number of correct prediction: Digit - 1 : 199
The number of false prediction: not Digit - 1: 1
Test digit 2
The number of correct prediction: Digit - 2 : 171
The number of false prediction: not Digit - 2: 29
Test digit 3
The number of correct prediction: Digit - 3 : 186
The number of false prediction: not Digit - 3: 14
Test digit 4
The number of correct prediction: Digit - 4 : 181
The number of false prediction: not Digit - 4: 19
Test digit 5
The number of correct prediction: Digit - 5 : 164
The number of false prediction: not Digit - 5: 36
Test digit 6
The number of correct prediction: Digit - 6 : 184
The number of false prediction: not Digit - 6: 16
Test digit 7
The number of correct prediction: Digit - 7 : 184
The number of false prediction: not Digit - 7: 16
Test digit 8
The number of correct prediction: Digit - 8 : 164
The number of false prediction: not Digit - 8: 36
Test digit 9
The number of correct prediction: Digit - 9 : 177
The number of false prediction: not Digit - 9: 23
Test digit 10
The number of correct prediction: Digit - 10 : 194
The number of false prediction: not Digit - 10: 6
One vs All Prediction
[[199.   0.   0.   0.   1.   0.   0.   0.   0.   0.]
 [  0. 171.   6.   8.   0.   3.   4.   7.   0.   1.]
 [  0.   1. 186.   2.   6.   0.   0.   3.   1.   1.]
 [  2.   2.   0. 181.   1.   2.   1.   3.   8.   0.]
 [  0.   0.  19.   5. 164.   1.   0.   1.   3.   7.]
 [  0.   2.   0.   5.   4. 184.   0.   3.   0.   2.]
 [  2.   0.   0.   3.   2.   0. 184.   0.   8.   1.]
 [  0.   2.   9.   3.   7.   1.   0. 164.   7.   7.]
 [  1.   0.   0.   6.   0.   0.   9.   6. 177.   1.]
 [  0.   0.   1.   1.   0.   2.   0.   2.   0. 194.]]
Accuracy is:90.2%
```

Plots

In [ ]:

```python
import seaborn as sns
col = [i for i in range(1,10)] + [0]
ax = sns.heatmap(C, annot=True,fmt=".0f")
ax.set(xlabel="Predictions", ylabel="True Label")
ax.set_xticklabels(col)
ax.set_yticklabels(col)
```

Out[ ]:

```
[Text(0, 0.5, '1'),
 Text(0, 1.5, '2'),
 Text(0, 2.5, '3'),
 Text(0, 3.5, '4'),
 Text(0, 4.5, '5'),
 Text(0, 5.5, '6'),
 Text(0, 6.5, '7'),
 Text(0, 7.5, '8'),
 Text(0, 8.5, '9'),
 Text(0, 9.5, '0')]
```