EECS 553 HW3

Lingqi Huang

September 2024

1 Problem 1

(a):Notice that if X_i is misclassified, then the true label and the predicted label are distinct, so we must have

$$1 - \epsilon_i \le y_i(w^T x_i + b) \le 0$$

and so we conclude that $\epsilon_i \ge 1$.
(b): We first denote that margin be the plane such that all x_i satisfies

$$y_i(w^T x_i + b) = 1$$

Now given any $\tilde{x}_i$ that does not on the margin, we must have

$$y_i(w^T \tilde{x}_i + b) = 1 - \epsilon_i$$

Now notice we can rewrite the above formula to be

$$y_i(w^T \tilde{x}_i + b) = y_i(w^T x_i + b) - \epsilon_i$$

Combine terms we will then conclude that

$$\epsilon_i = -y_i w^T(\tilde{x}_i - x_i) \Rightarrow \epsilon_i^2 = w^T ||\tilde{x}_i - x_i||^2 w \Rightarrow \epsilon_i = ||w|| \cdot ||\tilde{x}_i - x_i||$$

So we conclude that $epsilon_i$ is proportional to the distance from x_i to the margin hyperplane, and the coefficient is $||w||$.

2 Problem 2

We select $\lambda = \frac{1}{C}$, and we can rewrite the constraint to be

$$\epsilon_i \ge \max\{0, 1 - y_i(w^T x_i + b)\}$$

Then, we can rewrite the optimization problem to be

$$\min_{w,b} \frac{\lambda}{2}||w||^2 + \frac{1}{n}\left[(1-\alpha)\sum_{i:y_i=1} \max\{0, 1 - y_i(w^T x_i + b)\} + \alpha \sum_{i:y_i=-1} \max\{0, 1 - y_i(w^T x_i + b)\}\right]$$

Now we can rewrite the last term to be $L(y, f(x))$, where $f(x) = w^T x + b$, and we can write

$$L(y, f(x)) = \begin{cases} (1-\alpha) \cdot \max\{0, 1 - y_i f(x_i)\} & \text{if } y_i = 1 \\ \alpha \cdot \max\{0, 1 - y_i f(x_i)\} & \text{if } y_i = -1 \end{cases}$$

Thus, the optimize problem can be written as a regularized ERM that is

$$\min_{w,b} \frac{\lambda}{2}||w||^2 + \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i))$$

Which yields the same classifier as the quadratic program.

3 Problem 3

(a): We assume that the occurrence of a word does not depend on previous occurrence of the same word.
(b): we notice that

$$
\begin{aligned}
\hat{y} &= \arg\max_{k\in\{0,1\}} \log\left(\hat{\pi}_k \prod_{j=1}^{d}\left(\frac{n_{kj}+\alpha}{n_k+\alpha d}\right)^{x_j}\right) && (1)\\
&= \arg\max_{k\in\{0,1\}} \log(\hat{\pi}_k) + \sum_{j=1}^{d} x_j \log\left(\frac{n_{kj}+\alpha}{n_k+\alpha d}\right) && (2)\\
&= \arg\max_{k\in\{0,1\}} \log(\hat{\pi}_k) + \sum_{j=1}^{d} x_j \log(n_{kj}+\alpha) - \sum_{j=1}^{d} x_j \log(n_k+\alpha d) && (3)\\
& && (4)
\end{aligned}
$$

(c): Please see the code in my py file on Canvas. The estimated $\log(\pi_0) = -0.697$, the estimated $\log(\pi_1) = -0.689$.
(d): The test error would be 0.126.
(e): The test error would be about 0.4987.

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

In [2]:
```
## Import train data and the test data
X_train = np.load('hw2p2_train_x.npy')
X_test = np.load('hw2p2_test_x.npy')

y_train = np.load('hw2p2_train_y.npy')
y_test = np.load('hw2p2_test_y.npy')
```

Part(c)

(i)

In [21]:
```
## Get the train data where Y label is 1
index_y_is_1 = np.where(y_train == 1)[0]
X_label1 = X_train[index_y_is_1]
```

In [24]:
```
## Get the train data where Y label is 0
index_y_is_0 = np.where(y_train == 0)
X_label0 = X_train[index_y_is_0]
```

In [43]:
```
alpha = 1
d = 1000
```

In [48]:
```
## Get a list of log(p_1j) for j = 1, ... 1000
n_k1 = np.sum(X_label1)
p_1j = []
for j in range(1000):
    frequency = 0
    for i in range(X_label1.shape[0]):
        frequency = frequency + X_label1[i][j]
    probs = (frequency + alpha)/(n_k1 + alpha * d)
    p_1j.append(np.log(probs))
```

In [52]:
```
p_1j[:5]
```

Out[52]:
```
[-7.024471078678098,
 -7.717618259238043,
 -7.247614629992308,
 -7.717618259238043,
 -7.38114602261683]
```

In [49]:
```
## Get a list of log(p_0j) for j = 1, ... 1000
n_k0 = np.sum(X_label0)
p_0j = []
for j in range(1000):
    frequency = 0
    for i in range(X_label0.shape[0]):
        frequency = frequency + X_label0[i][j]
    probs = (frequency + alpha)/(n_k0 + alpha * d)
    p_0j.append(np.log(probs))
```

In [53]:
```
p_0j[:5]
```

Out[53]:
```
[-6.055718936974995,
 -9.552226498441476,
 -9.552226498441476,
 -9.552226498441476,
 -9.552226498441476]
```

(ii)

In [71]:
```
## Compute the prior pi_0 and pi_0
estimate_pi_1 = np.log(X_label1.shape[0] / X_train.shape[0])
estimate_pi_0 = np.log(X_label0.shape[0] / X_train.shape[0])
```

In [72]:
```
print("Estimate of prior pi_0 is:", estimate_pi_0, "Estimate of prior pi_1 is:",
```

```
Estimate of prior pi_0 is: -0.6965085282626502 Estimate of prior pi_1 is: -0.6897
970936746632
```

Part(d)

In [73]:
```
prediction = []
for i in range(X_test.shape[0]):
    y0_value = 0
    y1_value = 0
### Get the value of belong to label 1 or label 0
    for j in range(1000):
        y0_value += X_test[i][j] * p_0j[j]
        y1_value += X_test[i][j] * p_1j[j]

    y0_value += estimate_pi_0
    y1_value += estimate_pi_1
### decision rule
    if y0_value > y1_value:
        prediction.append(0)
    else:
        prediction.append(1)
```

In [74]:
```
##define a function that to get the accuracy
def accuracy_bayes(X, Y):
    final_result = []
    for i in range(len(X)):
        if X[i] == Y[i]:
            final_result.append(1)
        else:
            final_result.append(0)
    return sum(final_result) / len(final_result)
```

In [75]:
```
test_error = 1 - accuracy_bayes(prediction, y_test)
```

In [76]:
```
print("The test error is for the naive bayesian classifier is:", test_error)
```

```
The test error is for the naive bayesian classifier is: 0.12594458438287148
```

Part(e)

In [77]:
```
if_list = [1] * X_test.shape[0]
```

In [78]:
```
if_test_error = 1 - accuracy_bayes(if_list, y_test)
```

In [79]:
```
print("The test error is for the naive bayesian classifier is:", if_test_error)
```

```
The test error is for the naive bayesian classifier is: 0.49874055415617125
```