CSC311 A2

Benson Li, 1007815376

February 15, 2023

1 Expected Loss and Bayes Optimality.

1.1 (a)

For the policy that keeps every email:

$$\begin{aligned}E[J(y, t = Spam)] &= P(t = Spam) \cdot J(Keep, Spam) + P(t = NonSpam) \cdot J(Keep, NonSpam)\\ &= 0.2 \cdot 1 + 0.8 \cdot 0\\ &= 0.2\end{aligned}$$

For the policy that removes every email:

$$\begin{aligned}E[J(y, t = NonSpam)] &= P(t = Spam) \cdot J(Remove, Spam) + P(t = NonSpam) \cdot J(Remove, NonSpam)\\ &= 0.2 \cdot 0 + 0.8 \cdot 100\\ &= 80\end{aligned}$$

1.2 (b)

We will select the decision y that minimizes the expected loss:
We denote $P(t = Spam|x)$ as p

$$\begin{aligned}E[J(y, t)|x] &= P(t = NonSpam|x) \cdot J(y, t = NonSpam) + P(t = Spam|x) \cdot J(y, t = Spam)\\ &= (1 - p) \cdot J(y, t = NonSpam) + p \cdot J(y, t = Spam)\end{aligned}$$

Therefore:

$$\begin{aligned}E[J(y = Keep, t)|x] &= (1 - p) \cdot J(y = Keep, t = NonSpam) + p \cdot J(y = Keep, t = Spam)\\ &= (1 - p) \cdot 0 + p \cdot 1\\ &= p\end{aligned}$$

$$\begin{aligned}E[J(y = Remove, t)|x] &= (1 - p) \cdot J(y = Remove, t = NonSpam) + p \cdot J(y = Remove, t = Spam)\\ &= (1 - p) \cdot 100 + p \cdot 0\\ &= 100 - 100p\end{aligned}$$

When $p \geq \frac{100}{101}$, we have $E[J(y = Remove, t)|x] \leq E[J(y = Keep, t)|x]$, we choose $y^* = Remove$.

When $p < \frac{100}{101}$, we have $E[J(y = Keep, t)|x] < E[J(y = Remove, t)|x]$, we choose $y^* = Keep$.

1.3 (c)

According to the Baye's Rule:

$$P(t|x_1, x_2) = \frac{P(x_1, x_2|t)P(t)}{P(x_1, x_2)}$$

Then, we are able to calculate that:

$$\begin{aligned}
P(t = Spam|x_1 = 0, x_2 = 0) &= \frac{0.45 \cdot 0.2}{0.2 \cdot 0.45 + 0.8 \cdot 0.996} = 0.1015 \\
P(t = Spam|x_1 = 1, x_2 = 0) &= \frac{0.18 \cdot 0.2}{0.2 \cdot 0.18 + 0.8 \cdot 0.002} = 0.9574 \\
P(t = Spam|x_1 = 1, x_2 = 1) &= \frac{0.12 \cdot 0.2}{0.2 \cdot 0.12 + 0.8 \cdot 0} = 1 \\
P(t = Spam|x_1 = 0, x_2 = 1) &= \frac{0.25 \cdot 0.2}{0.2 \cdot 0.25 + 0.8 \cdot 0.002} = 0.9690
\end{aligned}$$

We will select the decision y that minimizes the expected loss:

$E[J(y, t)|x_1, x2] = P(t = NonSpam|x_1, x_2) \cdot J(y, t = NonSpam) + P(t = Spam|x_1, x_2) \cdot J(y, t = Spam)$

Therefore, according to the given table:

$$\begin{aligned}
E[J(y = Keep, t)|x_1 = 0, x_2 = 0] &= 0.8985 \cdot 0 + 0.1015 \cdot 1 \\
&= 0.1015
\end{aligned}$$

$$\begin{aligned}
E[J(y = Remove, t)|x_1 = 0, x_2 = 0] &= 0.8985 \cdot 100 + 0.1015 \cdot 0 \\
&= 89.85
\end{aligned}$$

Given $x_1 = 0 \wedge x_2 = 0$, since $0.1015 < 89.85$, I would keep the email.

$$\begin{aligned}
E[J(y = Keep, t)|x_1 = 1, x_2 = 0] &= 0.0426 \cdot 0 + 0.9574 \cdot 1 \\
&= 0.9574
\end{aligned}$$

$$\begin{aligned}
E[J(y = Remove, t)|x_1 = 1, x_2 = 0] &= 0.0426 \cdot 100 + 0.9574 \cdot 0 \\
&= 4.26
\end{aligned}$$

Given $x_1 = 1 \wedge x_2 = 0$, since $0.9574 < 4.26$, I would keep the email.

$$\begin{aligned}
E[J(y = Keep, t)|x_1 = 1, x_2 = 1] &= 0 \cdot 0 + 1 \cdot 1 \\
&= 1
\end{aligned}$$

$$\begin{aligned}
E[J(y = Remove, t)|x_1 = 1, x_2 = 1] &= 0 \cdot 100 + 1 \cdot 0 \\
&= 0
\end{aligned}$$

Given $x_1 = 1 \wedge x_2 = 1$, since $0 < 1$, I would remove the email.

$$\begin{aligned}
E[J(y = Keep, t)|x_1 = 0, x_2 = 1] &= 0.031 \cdot 0 + 0.969 \cdot 1 \\
&= 0.969
\end{aligned}$$

$$\begin{aligned}
E[J(y = Remove, t)|x_1 = 0, x_2 = 1] &= 0.031 \cdot 100 + 0.969 \cdot 0 \\
&= 3.1
\end{aligned}$$

Given $x_1 = 0 \wedge x_2 = 1$, since $0.969 < 3.1$, I would keep the email.

1.4 (d)

$$
\begin{aligned}
P(x_1 = 0, x_2 = 0) &= 0.2 \cdot 0.45 + 0.8 \cdot 0.996 = 0.8868 \\
P(x_1 = 1, x_2 = 0) &= 0.2 \cdot 0.18 + 0.8 \cdot 0.002 = 0.0376 \\
P(x_1 = 1, x_2 = 1) &= 0.2 \cdot 0.12 + 0.8 \cdot 0 = 0.024 \\
P(x_1 = 0, x_2 = 1) &= 0.2 \cdot 0.25 + 0.8 \cdot 0.002 = 0.0516
\end{aligned}
$$

The expectation loss is:

$$
\begin{aligned}
E[J(y^*, t)] &= \sum_{x_1, x_2} P(x_1, x_2) E[J(y^*, t)|x_1, x_2] \\
&= 0.8868 \cdot 0.1015 + 0.0376 \cdot 0.9574 + 0 \cdot 0.024 + 0.0516 \cdot 0.969 \\
&= 0.176
\end{aligned}
$$

2 Feature Maps.

2.1 (a)

Assume this data-set is linear separable:
Let $x_0 = 1, w_0 = b$(bias), X is design matrix.
Then $\bar{y} = w^T X$, we will make decision based on the boundary 0.

Therefore, we got following equations:

$$w_0 + w_1(-1) + w_2(0) \geq 0 \quad (1)$$
$$w_0 + w_1(1) + w_2(2) < 0 \quad (2)$$
$$w_0 + w_1(2) + w_2(3) \geq 0 \quad (3)$$

We will try to find solution for these equations:
From (3) - (2), we get $w_1 + w_2 > 0$, we denote this as (4)

We plus $2 * (4)$ to (1), we then get that $w_0 + w_1 + 2w_2 > 0$.
This contradicts with equation(2), which proves that there is no solution for these equations.

Therefore, this data-set is not linearly separable.

2.2 (b)

Let $x_0 = 1$, X is design matrix.
We will make decision based on the boundary 0.

Therefore, we got following equations:

$$w_0 + w_1(-1) + w_2(0) + w_3(0) \geq 0 \quad (1)$$
$$w_0 + w_1(1) + w_2(2) + w_3(4) < 0 \quad (2)$$
$$w_0 + w_1(2) + w_2(3) + w_3(9) \geq 0 \quad (3)$$

Equivalently:

$$w_0 - w_1 \geq 0 \quad (1)$$
$$w_0 + w_1 + 2w_2 + 4w_3 < 0 \quad (2)$$
$$w_0 + 2w_1 + 3w_2 + 9w_3 \geq 0 \quad (3)$$

These are the constraints that w_0, w_1, w_2, w_3 need to satisy.

Even though I don't need to solve w_0, w_1, w_2, w_3, I list my solution below just for practise.

From (1), we can have $w_0 = 2, w_1 = 1$.
We plug this in (2), we need to satisy $3 + 2w_2 + 4w_3 < 0$, we can have $w_2 = -6, w_3 = 2$.
We plug this in (3), we check that $2 + 2 + (-18) + 18 = 4 \geq 0$.

Therefore, we have the following solution:

$$w_0 = 2$$
$$w_1 = 1$$
$$w_2 = -6$$
$$w_3 = 2$$

3 kNN vs. Logistic Regression

3.1 (a)

3.1.1 (i)

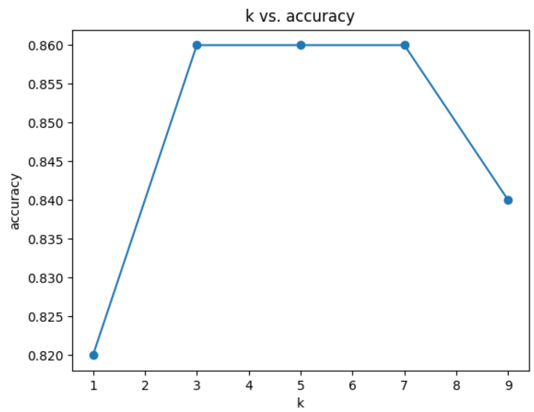


Figure 1: The plot for accuracy vs. k

3.1.2 (ii)

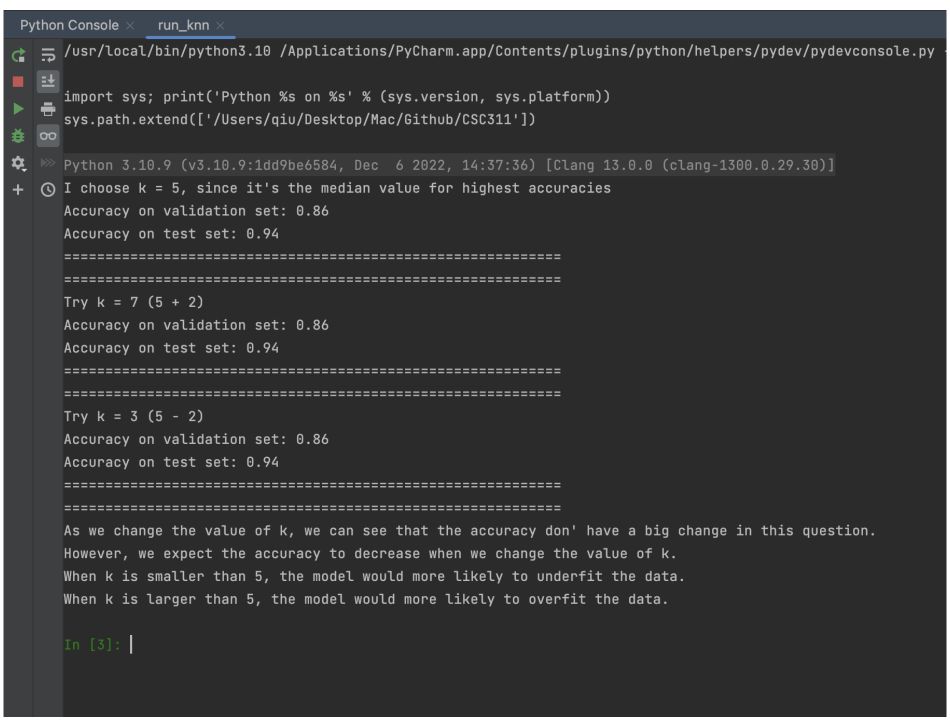


Figure 2: Test and validation accuracies change as we change the value of k

As we change the value of k, we can see that the accuracy don' have a big change in this question. However, we expect the accuracy to decrease when we change the value of k. When k is smaller than 5, the model would more likely to underfit the data. When k is larger than 5, the model would more likely to overfit the data.

3.2 (b)

3.2.1 (i)

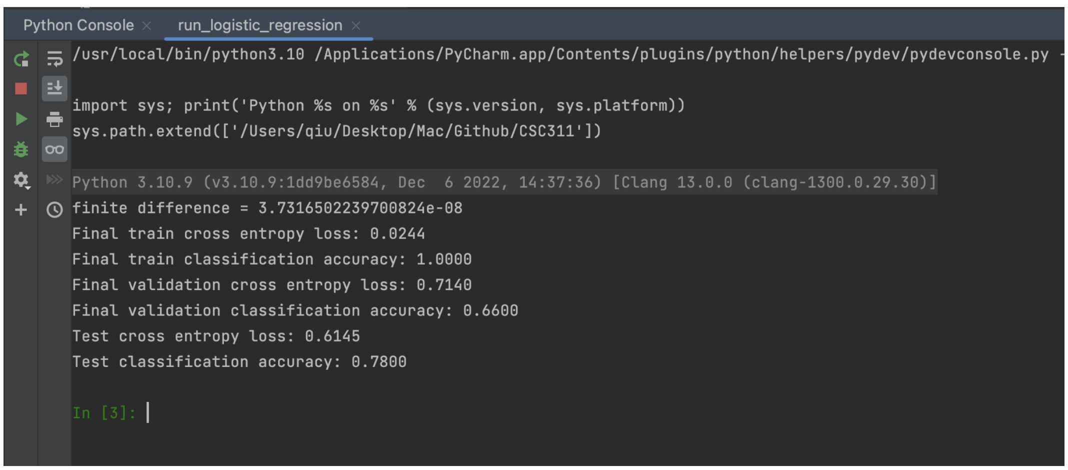
Please refer to python files.

3.2.2 (ii)

The following result (for question (ii) and (iii)) is collected under hyperparameters (best hyperparameter settings):

learning_rate: 0.01;
weight_regularization = 0;
num_iterations: 1000

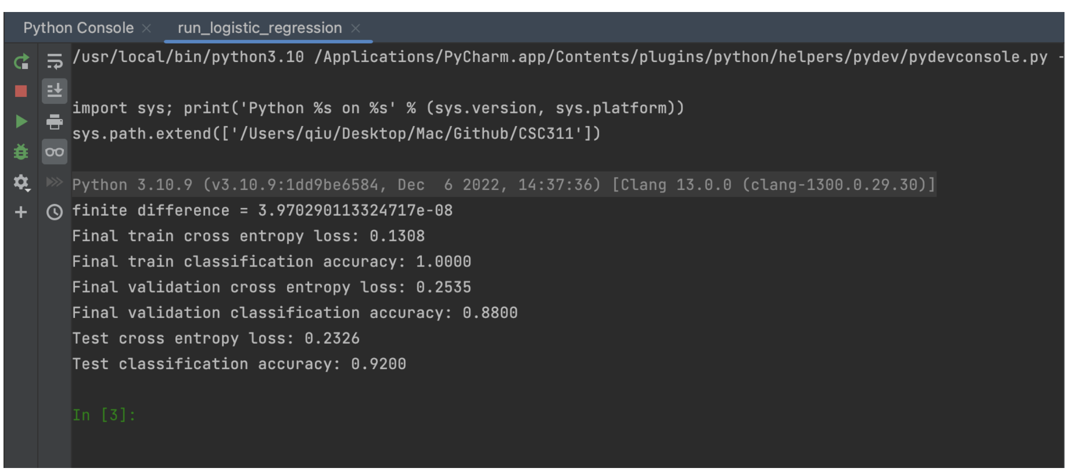
The output result for small training set:



```
Python Console    run_logistic_regression
/usr/local/bin/python3.10 /Applications/PyCharm.app/Contents/plugins/python/helpers/pydev/pydevconsole.py -

import sys; print('Python %s on %s' % (sys.version, sys.platform))
sys.path.extend(['/Users/qiu/Desktop/Mac/Github/CSC311'])

Python 3.10.9 (v3.10.9:1dd9be6584, Dec  6 2022, 14:37:36) [Clang 13.0.0 (clang-1300.0.29.30)]
finite difference = 3.7316502239700824e-08
Final train cross entropy loss: 0.0244
Final train classification accuracy: 1.0000
Final validation cross entropy loss: 0.7140
Final validation classification accuracy: 0.6600
Test cross entropy loss: 0.6145
Test classification accuracy: 0.7800

In [3]:
```

Figure 3: Gradient checking and Test result for small dataset

The output result for large training set:



```
Python Console    run_logistic_regression
/usr/local/bin/python3.10 /Applications/PyCharm.app/Contents/plugins/python/helpers/pydev/pydevconsole.py -

import sys; print('Python %s on %s' % (sys.version, sys.platform))
sys.path.extend(['/Users/qiu/Desktop/Mac/Github/CSC311'])

Python 3.10.9 (v3.10.9:1dd9be6584, Dec  6 2022, 14:37:36) [Clang 13.0.0 (clang-1300.0.29.30)]
finite difference = 3.970290113324717e-08
Final train cross entropy loss: 0.1308
Final train classification accuracy: 1.0000
Final validation cross entropy loss: 0.2535
Final validation classification accuracy: 0.8800
Test cross entropy loss: 0.2326
Test classification accuracy: 0.9200

In [3]:
```

Figure 4: Gradient checking and Test result for large dataset

3.2.3 (iii)

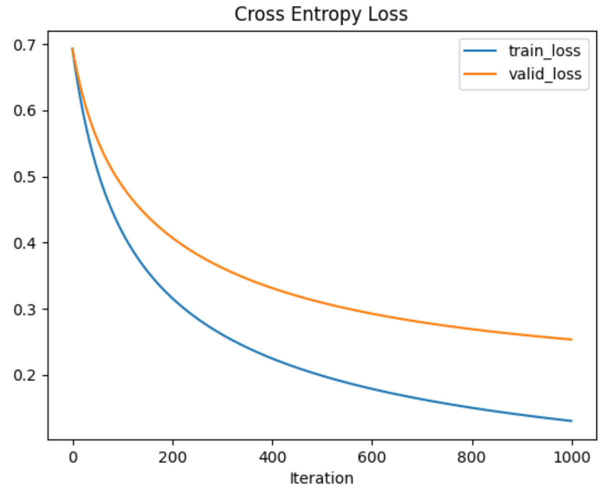


Figure 5: Cross Entropy Loss

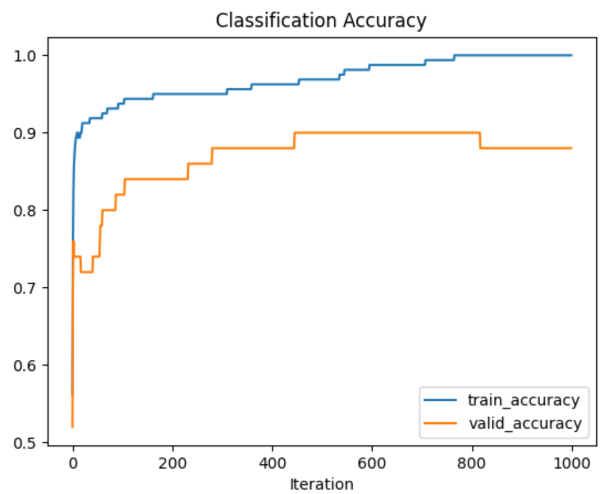


Figure 6: Classification Accuracy

3.2.4 (iv)

We run our program for three times with the same hyper-parameters on the large dataset and got the following result.

Result				
	Train_Loss	Train_Accuracy	Val_Loss	Val_Accuracy
1st	0.0244	1.0000	0.7140	0.6600
2nd	0.1307	1.0000	0.2538	0.8800
3rd	0.1308	1.0000	0.2536	0.8800

To choose the best hyper-parameters, we can run the code multiple times and calculate the average validation accuracy error. This will help to ensure that the results are not simply due to chance.