CSCE 636 Deep Learning HW1

Jin Hyun Park
jinhyun.park@tamu.edu
UIN: 633001823

September 15, 2022

Chapter 1

Answers to the non-programming part

1.1 Data Processing

1.1.1 (a)

We need to split training data into two separate sets: the train (or training) set and the validation set. The training set is used to train the model; whereas, the validation set is used to measure the performance of the trained model before the testing process. Traditionally, after we divide the training data into two different sets, we try different combinations of hyper-parameters to train and validate the model. We then find the best hyper-parameters that fit the validation set. In fact, there is another data-set called *test* data. After we find the best hyper-parameters, we can test the model by using the test data. It is important to note that the test data have to be always separated from the training data.

1.1.2 (b)

No, we should not re-train the model on the whole training data. To put it another way, the validation set should not be used when we train the model. The validation set is there to estimate the performance of the model trained by the training set and to tune hyper-parameters. If we re-train the model on the whole training data, then we might need to search for new hyper-parameters. However, this loses the advantage of separating the training data into two different sets. Therefore, we should not re-train the model with the whole training data.

1.1.3 (d)

A third feature is always 1; because, we need to add bias term. This term can be also considered an intercept. If we do not have 1 (or bias term), it would not be possible to build a correct decision boundary. This can be easily explained by using some simple mathematics.

Consider we would like to build a decision boundary that has the following property in the logistic regression model (See Figure 1.1) and does not have a bias term. Also, assuming that we have two features: x_1 and x_2, the equation of decision boundary is

$$z = w^T x = w_1 \cdot x_1 + w_2 \cdot x_2 = 0 \tag{1.1}$$

$$\text{Predicted class of } x = \begin{cases} 1 & \text{if } \theta(w^T x) \geq 0.5 \\ -1 & \text{if } \theta(w^T x) < 0.5 \end{cases}$$

where the function $\theta(z) = \frac{1}{1+e^{-z}}$ looks like

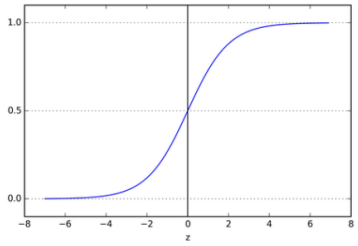

Figure 1.1: Figure from CSCE 636: Deep Learning (Fall 2022) Assignment #1

Rewriting the equation 1.1, we get the following equation.

$$x_2 = -\frac{w_1}{w_2} x_1 \tag{1.2}$$

This clearly shows that the decision boundary should always cross the origin. Thus, it will be hard to find a good decision boundary. However, if we have 1 (or bias term), the equation of the decision boundary is

$$z = w^T x = w_0 \cdot x_0 + w_1 \cdot x_1 + w_2 \cdot x_2 = 0 \tag{1.3}$$

$$z = w^T x = w_0 \cdot 1 + w_1 \cdot x_1 + w_2 \cdot x_2 = 0 \tag{1.4}$$

Rewriting the equation 1.4, we get the following equation.

$$x_2 = -\frac{w_1}{w_2} x_1 - \frac{w_0}{w_2} \tag{1.5}$$

This means that the decision boundary does not always need to cross the origin. Hence, it is necessary to include the bias term (third feature with 1) to find a good decision boundary.

1.1.4 (f)

See Figure 1.2

1.2 Cross Entropy

1.2.1 (a)

$$E(w) = \ln(1 + \exp(-y w^T x)) \tag{1.6}$$

1.2.2 (b)

Recall the following rule.

$$\nabla \ln f(x) = \frac{f'(x)}{f(x)} \tag{1.7}$$

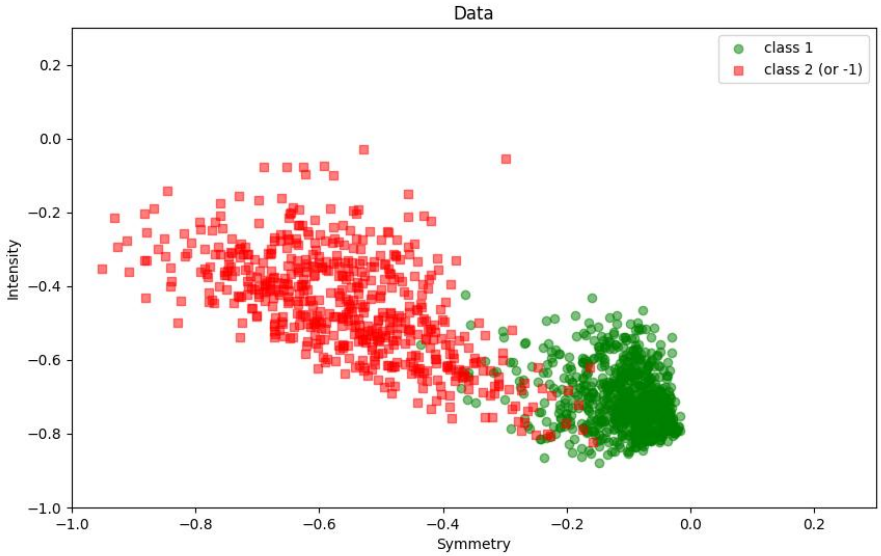

Figure 1.2: Visualization of training data. *visualize_features.*

$$\nabla E(w) = \nabla \ln(1 + \exp(-yw^T x)) \tag{1.8}$$

$$= -yx \cdot \frac{\exp(-yw^T x)}{1 + \exp(-yw^T x)} \tag{1.9}$$

$$= -yx \cdot \frac{1}{\exp(yw^T x) + 1} \tag{1.10}$$

1.2.3 (c)

The decision boundary is linear. In fact, subsection 1.1.3 partially covered the linearity of decision boundary regarding logistic regression. The equation of the decision boundary can be expressed by the following equation

$$z = w^T x = w_0 \cdot 1 + w_1 \cdot x_1 + w_2 \cdot x_2 = 0 \tag{1.11}$$

Rewriting the above equation, we get the following equation.

$$x_2 = -\frac{w_1}{w_2} x_1 - \frac{w_0}{w_2} \tag{1.12}$$

This clearly shows that the decision boundary is linear. In fact, the decision boundary is linear as we used linear input features. We can get a non-linear decision boundary if we use non-linear input features. For example, assuming that we have the following equation with a non-linear feature

$$z = w^T x = w_0 \cdot 1 + w_1 \cdot x_1 + w_1 \cdot x_1^2 + w_2 \cdot x_2 = 0 \tag{1.13}$$

and rewriting the equation, we get the following equation.

$$x_2 = -\frac{w_1}{w_2}{x_1}^2 - \frac{w_1}{w_2}x_1 - \frac{w_0}{w_2} \tag{1.14}$$

Thus, we can get a non-linear decision boundary if we use non-linear features (${x_1}^2$).

The most crucial benefit of using a sigmoid function is that it can be considered as a probability. Moreover, it is monotonic, continuous, and differentiable, enabling us to do gradient descent optimization. The sigmoid function (or activation) is usually used when data is binary. If there are more than two classes (or labels) it is recommended to use the softmax function (or activation).

1.2.4 (d)

The decision boundary is still linear. We need to find the decision boundary that satisfies the following property.

$$\frac{1}{1+\exp(-z)} = \frac{1}{1+\exp(-w^T x)} = 0.9 \tag{1.15}$$

$$\exp(-w^T x) = \frac{1}{9} \tag{1.16}$$

$$-w^T x = \ln\frac{1}{9} \tag{1.17}$$

$$w^T x = \ln 9 \tag{1.18}$$

$$w_0 + w_1 x_1 + w_2 x_2 \simeq 2.197 \tag{1.19}$$

The above equations indeed show that there is no relationship between linearity and the prediction rule.

1.2.5 (e)

The decision boundary of logistic regression is linear if we use linear input features. The linearity of features makes the decision boundary of logistic regression linear. Conversely, if we apply a kernel ϕ by the following and perform a logistic regression modeling process,

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \rightarrow \phi \rightarrow \begin{bmatrix} 1 \\ \sqrt{2}x_1 \\ \sqrt{2}x_2 \\ {x_1}^2 \\ \sqrt{2}x_1 x_2 \\ {x_2}^2 \end{bmatrix} \tag{1.20}$$

then we will get a non-linear decision boundary as the input features $x(\phi)$ are not linear.

1.3 Sigmoid logistic regression

1.3.1 (d)

See Figure 1.3

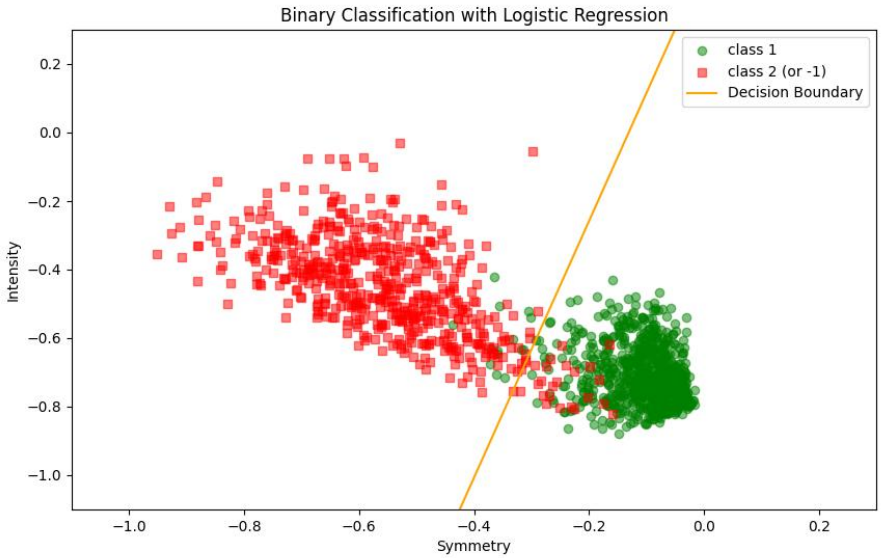

Figure 1.3: Decision boundary using sigmoid. *visualize_results.*

1.3.2 (e)

See Table 1.1.

Test accuracy	Learning rate	Epoch	Batch size
94.59%	0.01	100	5

Table 1.1: Test accuracy of the best sigmoid logistic model

1.4 Softmax Logistic Regression

1.4.1 (d)

See Figure 1.4. The decision boundary is plotted using the original decision boundaries (See Figure 3.1 in SM). The boundary is plotted by taking (1) the maximum of the two decision boundaries: black and yellow and (2) the minimum of the two decision boundaries: purple and yellow (See Figure 3.1 in SM).

1.4.2 (e)

See Table 1.2

Test accuracy	Learning rate	Epoch	Batch size
86.97%	0.01	100	5

Table 1.2: Test accuracy of the best softmax logistic model

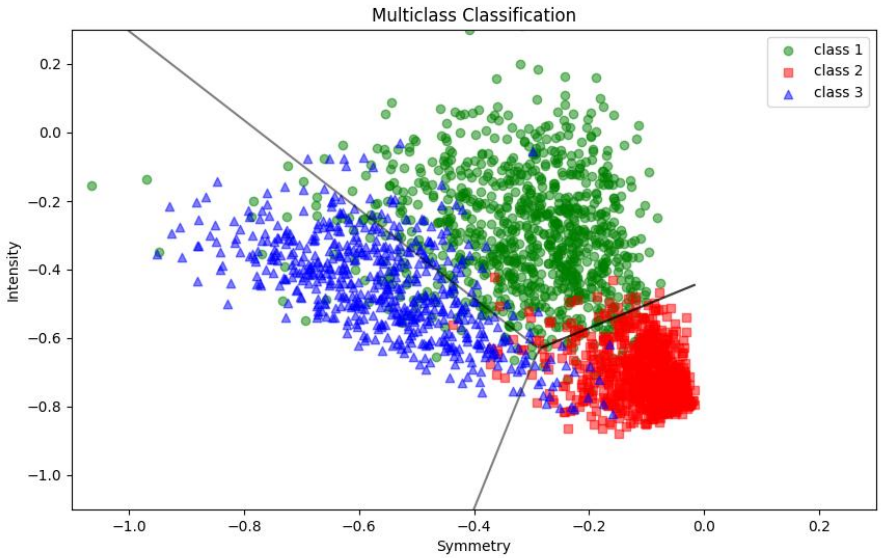

Figure 1.4: Decision Boundary using softmax. *visualize_results_multi.*

1.5 Softmax logistic vs Sigmoid logistic

1.5.1 (a)

To ensure convergence, `max_iter` (or epoch) is set to 10000. As shown in Table 1.3, sigmoid logis-

Classifier	Test accuracy	Learning rate	Epoch	Batch size
Sigmoid	93.07%	0.01	10000	5
Softmax	93.07%	0.01	10000	5

Table 1.3: Similarity between sigmoid and softmax classifier 1

tic classifier and softmax logistic classifier output the same test accuracy (`learning_rate`=0.01, `max_iter`=10000, and `batch_size`=5 for both. The weights were different from each other). This implies that the two classifiers do the same process. According to Ji and Xie [1], when shifting the parameters (i.e. weights), the softmax function in multi-class logistic regression has an invariance property. This leads to the fact that sigmoid-based logistic regression and softmax-based logistic regression become identical when there are only two classes (or labels, $K = 2$).

1.5.2 (b)

If the learning rate of a softmax classifier is k, then the learning rate of a sigmoid classifier should be $2k$ to ensure $w_1 - w_2 = w$. w_1 denotes the weights of class 1 and w_2 denotes the weights of class 2 of the softmax classifier. w denotes the weights of the sigmoid classifier. Intuitively, the learning rate of the sigmoid classifier should be double of the softmax classifier. This is because the softmax classifier updates both w_1 and w_2 for each step ($w_i \leftarrow w_i - learning\_rate \times gradient$, $i = 1,2$); whereas the sigmoid classifier updates w only once for each step. The following table shows the experimental details (See Table 1.4). And using the details, the following result was obtained (See Figure 1.5). It can be evidently seen that $w_1 - w_2 = w$ (or in this case:

Classifier	Learning rate	Epoch	Batch size
Sigmoid	0.02	1	5
Softmax	0.01	1	5

Table 1.4: Similarity between sigmoid and softmax classifier 2

$w_2 - w_1 = w$) holds and gradients of the sigmoid and softmax are exactly the same for every step. Therefore, $learning\_rate_{sigmoid} = learning\_rate_{softmax} \times 2$ should be fulfilled to satisfy the condition $w_1 - w_2 = w$.

```
<Weights>
1. Binary classification (Sigmoid)
[[ 0.16086473  0.50703941 -0.41099372]]

2. Multi-class classification (Softmax, k=2)
[[-0.08043236 -0.25351971  0.20549686]
 [ 0.08043236  0.25351971 -0.20549686]]


<Gradients>
1. Binary classification (Sigmoid)
array([-0.5       ,  0.07376094,  0.38068047]),
array([-0.09618428, -0.06999081,  0.10649995]),
array([ 0.30425414, -0.18888027, -0.13508811])
...

2. Multi-class classification (Softmax, k=2)
array([[ 0.5       , -0.07376094, -0.38068047],    [-0.5       ,  0.07376094,  0.38068047]]),
array([[ 0.09618428,  0.06999081, -0.10649995], [-0.09618428, -0.06999081,  0.10649995]]),
array([[-0.30425414,  0.18888027,  0.13508811], [ 0.30425414, -0.18888027, -0.13508811]])
...
```

Figure 1.5: Weights and gradients of sigmoid and softmax classifier

Chapter 2

Results and analysis of the programming part

- `fit_SGD` vs `fit_miniBGD`:
  From the result obtained by running `code/main.py`, it was able to observe that both the weight and train accuracy of `fit_SGD` and 'fit_miniBGD with batch size 1' are exactly the same. This proves that SGD is an extreme case of 'miniBGD with batch size equal to one'. See 2.1 for more details. It should be noted that the results (weights and train accuracy of `fit_SGD` and `fit_miniBGD`) will be slightly different if I shuffle the data before each epoch during training.

Name	Weights	Train accuracy
SGD	[10.38612223 32.55240052 2.81104834]	97.11%
miniBGD	[10.38612223 32.55240052 2.81104834]	97.11%

Table 2.1: `fit_SGD` vs `fit_miniBGD`: Weights and train accuracy

- Logistic Regression - Finding the best hyper-parameters and over-fitting:
  To find the best hyper-parameter, it is possible to use `GridSearchCV` or `RandomSearchCV` provided by `sklearn` package instead of manually trying out thousands of hyper-parameter combinations. However, the codes should have been written within the specific commented section. Therefore, the code I provided does not have these. However, it is possible to find better hyper-parameters that fit the training data and get better test accuracy. Please refer to the following table for the details of the logistic regression with the best hyper-parameters that I have found (See Table 2.2).

Weights	Train accuracy	Valid accuracy	Test accuracy
[2.50724819 19.18107188 -5.12761201]	97.26%	97.62%	94.59%

Table 2.2: Weights and accuracy. Experimental details: `learning_rate=0.01`, `max_iter=1000`, and `batch_size=5`

  According to Table 2.2, we can notice that over-fitting incurs. The test accuracy is lower than the validation accuracy. It is widely known that we can minimize the over-fitting problem by introducing regularization, early stopping, or training with more data.

- Softmax - Finding the best hyper-parameters and over-fitting:
  Please refer to the following Figure (See Figure 2.1) for the best hyper-parameters and weights I found. Similar to the logistic classifier, it can be seen that over-fitting arises a bit.

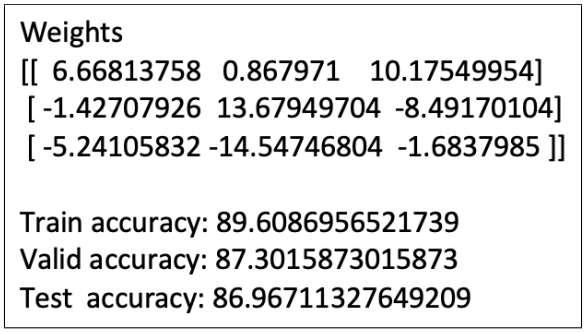

Figure 2.1: Weights and accuracy. Experimental details: `learning_rate=0.01`, `max_iter=1000`, and `batch_size=5`

Chapter 3

Supplementary Materials

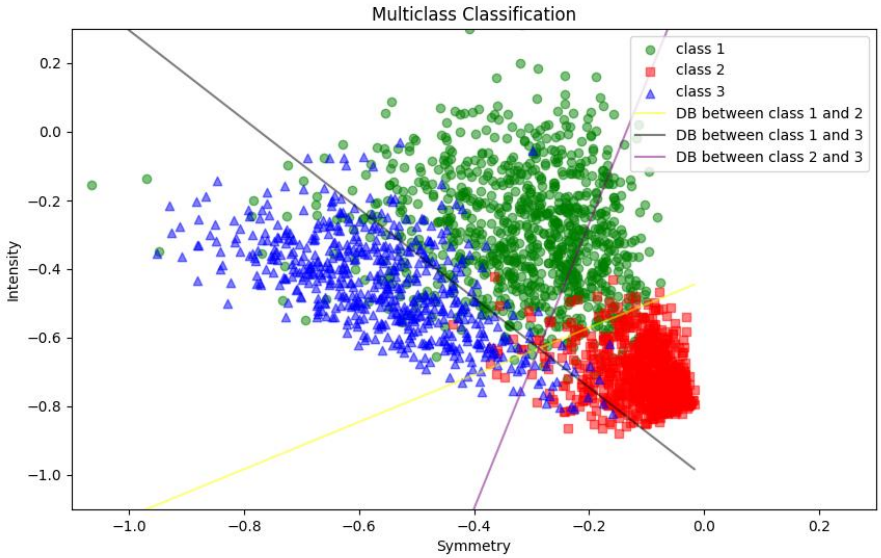

Figure 3.1: Original decision boundaries using softmax

Bibliography

[1] Shuiwang Ji and Yaochen Xie. *Logistic Regression: From Binary to Multi-Class*. URL: http://people.tamu.edu/~sji/classes/LR.pdf.