CIS 581 Assignment 4: Deep Learning Basics

Wenbo Zhang, Rui Li

December 2018

1 Plot Loss and Gradient

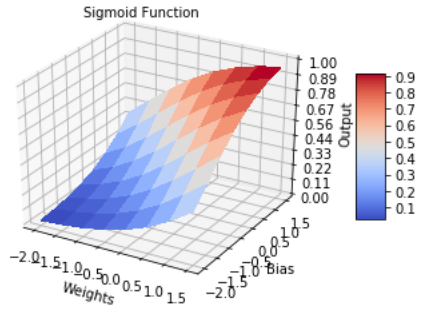

Figure 1: Sigmoid Function w.r.t. Weights and Bias

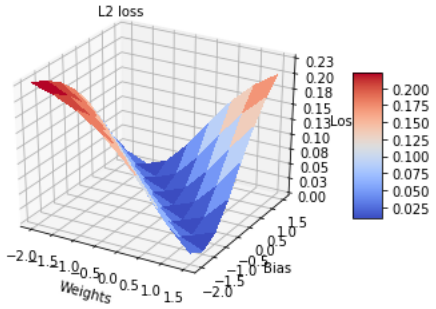

Figure 2: L2 Loss w.r.t. Weights and Bias

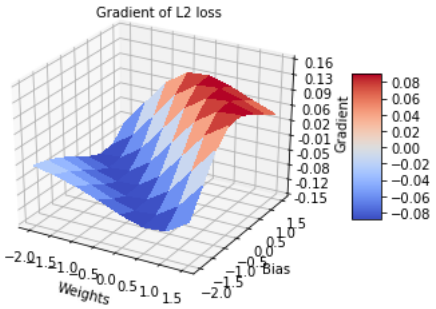

Figure 3: Gradient of L2 Loss on Weights w.r.t. Weights and Bias

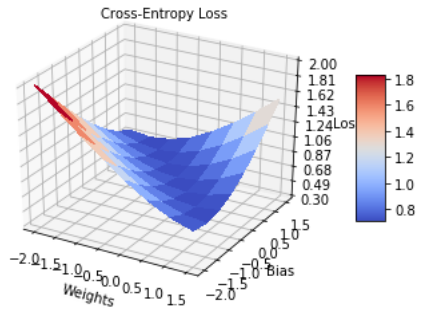

Figure 4: Cross-Entropy Loss w.r.t. Weights and Bias

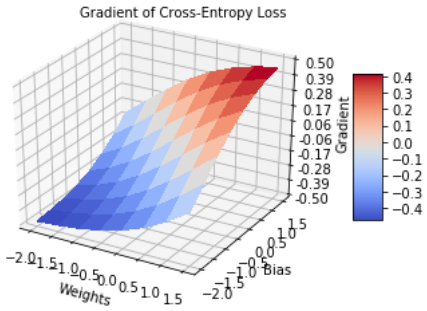

Figure 5: Gradient of Cross-Entropy Loss on Weights w.r.t. Weights and Bias

From the Figure 1-5 above, we can find the difference between Cross-Entropy (CE) Loss and L2 Loss. CE Loss has more change, compared with L2 Loss, when there are the same amount of change for both the weight and bias. It reflects the gradient for CE Loss is greater than that for L2 Loss.

Besides, we can know

$$weight := weight - learning\_rate * gradient$$

When *learning_rate* is constant, the learning process depends on the gradient. Because the gradient for CE Loss is greater than that for L2 Loss w.r.t to the same weight and bias, the CE Loss has a higher learning efficiency.

2 Experiment with Fully Connected Network

1.

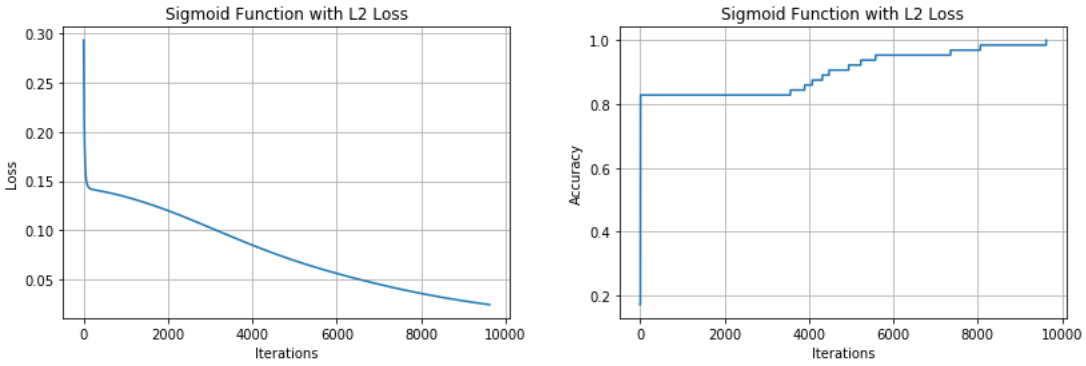

Figure 6: L2 Loss and Accuracy for Sigmoid Function

2.

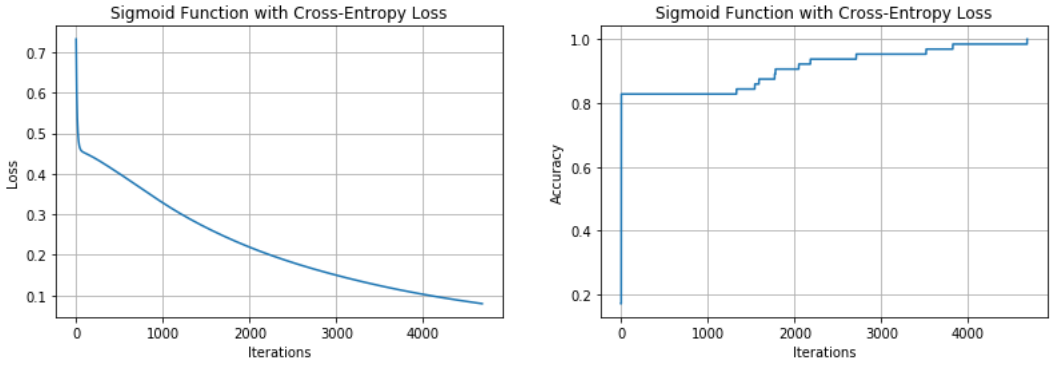

Figure 7: CE Loss and Accuracy for Sigmoid Function

3.

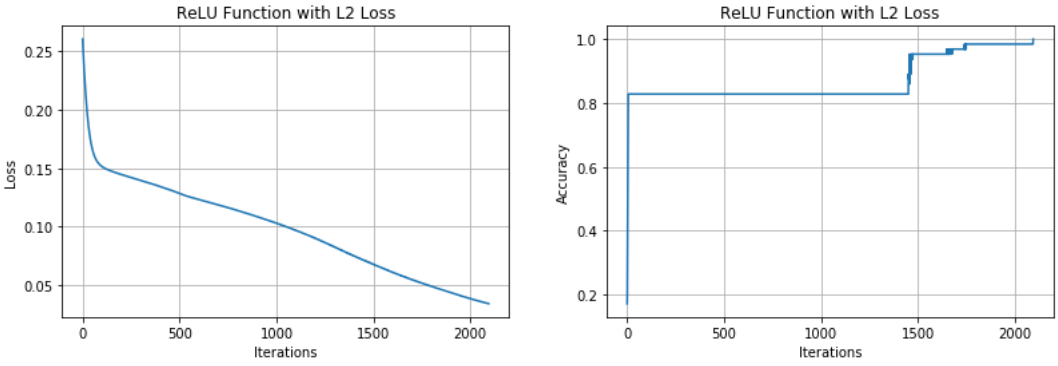

Figure 8: L2 Loss and Accuracy for ReLU Function

4.

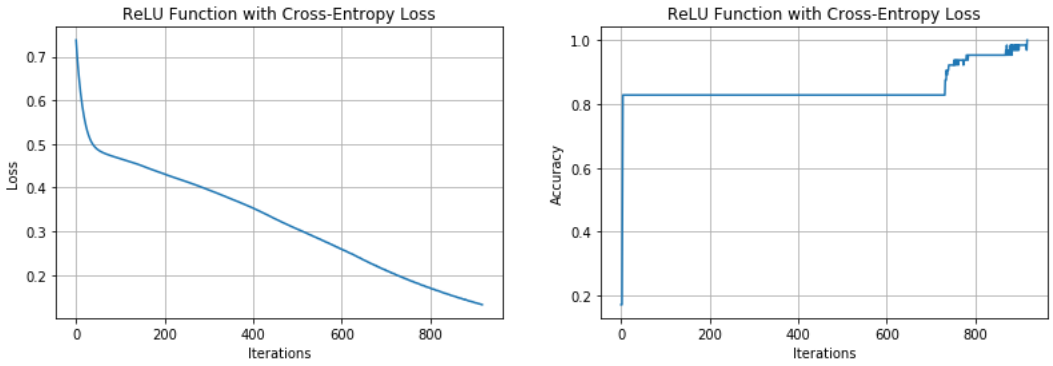

Figure 9: CE Loss and Accuracy for ReLU Function

5. The following is a table for different models.

Loss Function	Activation Function	of Iterations
L2 Loss	Sigmoid	9623
Cross-Entropy Loss	Sigmoid	4680
L2 Loss	ReLU	2095
Cross-Entropy Loss	ReLU	917

From the table, we learn that the model converges faster when activation function is ReLU, compared with Sigmoid. As the input value goes bigger for activation function, the gradient of Sigmoid function will go down compared with that of ReLU, which is constant when input value is positive. In this case, Sigmoid function, as activation function, will need more iterations to converge.

Besides, we also learn that the model using Cross-Entropy Loss converges faster than that using L2 Loss, so the model with Cross-Entropy Loss will have less iterations. When the model propagates back, the weights need to be updated. It depends on the derivative and learning rate. Because learning rate is a hyperparameter that is tuned by ourselves, the derivative of loss on weights decides the new weights. When we use cross-entropy loss, it can update the weights faster and then the model converges faster.

3 Solving XOR with a 2-layer Perceptron

1. We shall formulate multi-layer perceptron as an optimization problem to solve the XOR in this question.

   The loss function is cross-entropy loss

   $$\mathcal{L}_{CE} = -[y \log \hat{y} + (1-y) \log(1-\hat{y})]$$

   Besides, the network for this problem is followed Because the dataset is four points: $\mathbf{X}_1 = (0,0), \mathbf{X}_2 =$

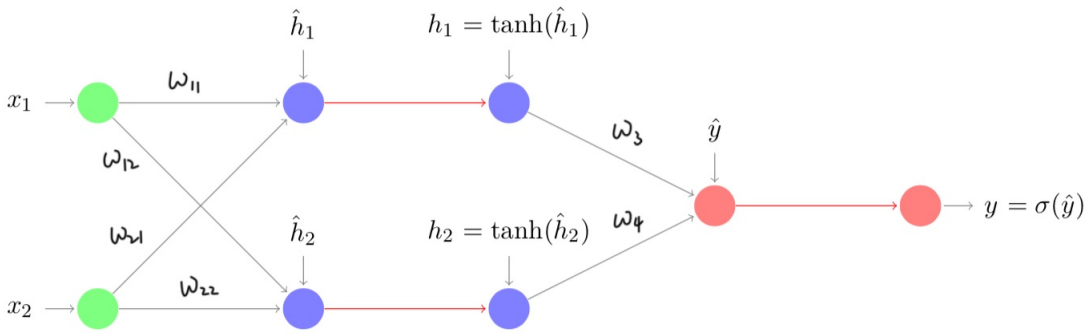

Figure 10: Two-layer Perceptron Network

$(0,1), \mathbf{X}_2 = (1,0), \mathbf{X}_2 = (1,1)$ with ground truth labels $0,1,1,0$ respectively. Plot the figure of data set (Figure 11).

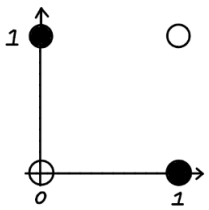

Figure 11: Data points

If we only use single layer perceptron, we could have the following situations (Figure 12), but we can never divide the data set into black part and white part separately.

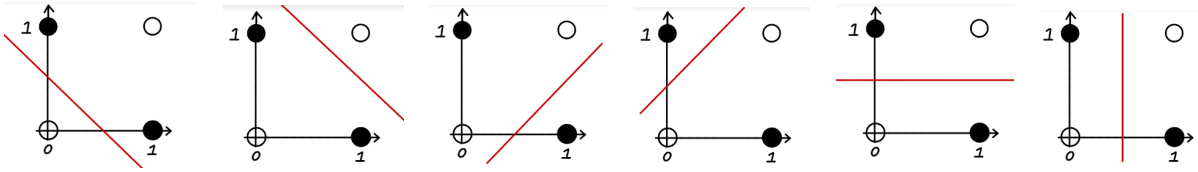

Figure 12: Possible Situations When Using Single Layer Perceptron

However, if we use two-layer perceptron, we can divide the data points separately (Figure 13).

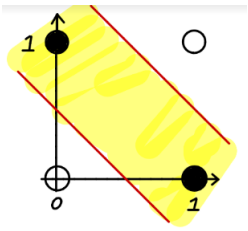

Figure 13: Two-layer Perceptron Dividing Data Points

2. The following Figure 14 is the linear classifier in the $(h_1.h_2)$ plane.

3. If we do not use activation function in the hidden layer, assume for the first layer, we can have

$$\hat{h_1} = w_{11}x_1 + w_{12}x_2$$
$$\hat{h_2} = w_{21}x_1 + w_{22}x_2$$

when we do not apply the activation function, the second layer would be

$$\begin{aligned}\hat{y} &= w_3\hat{h_1} + w_4\hat{h_2} \\ &= (w_3w_{11} + w_4w_{21})x_1 + (w_3w_{12} + w_4w_{22})x_2\end{aligned}$$

From the equation above, we can know it is still a linear combination, although we have two layers. Linear classifier cannot classify the data points (as we mentioned and visualized in the question above). Thus, it is critical for us to have an activation function.

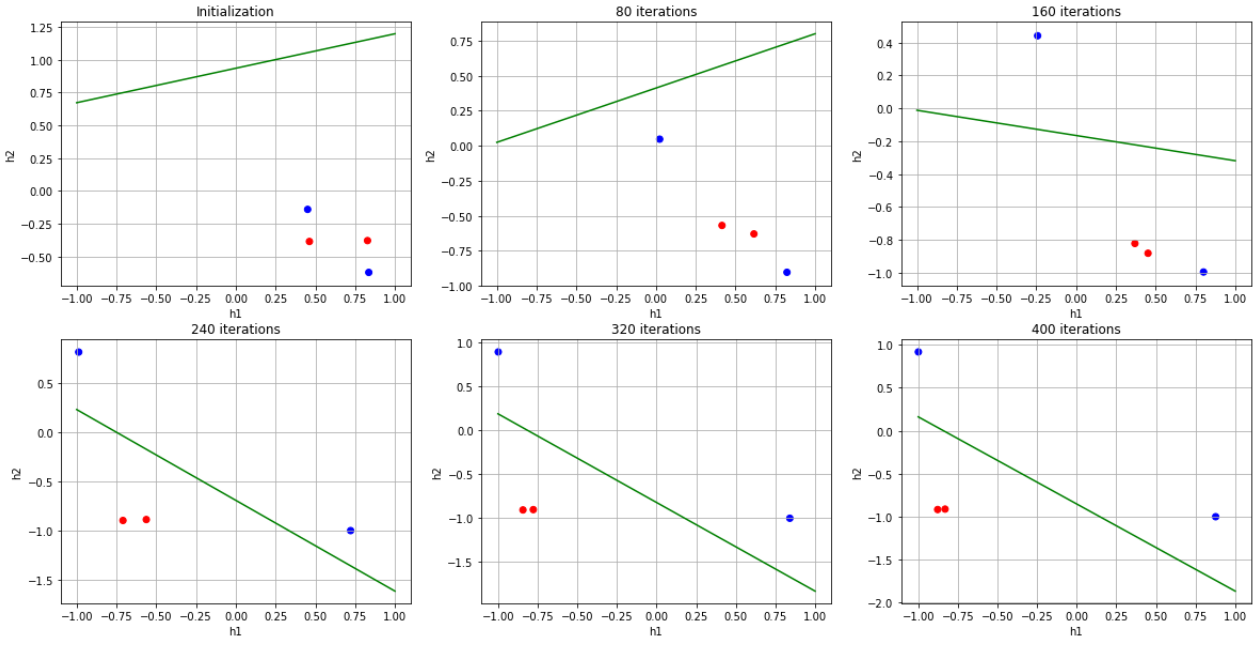

Figure 14: Separating Line Points in (h_1, h_2) Plane under Different Iterations

4 Experiment with Convolutional Network

1. This figures are as followed.

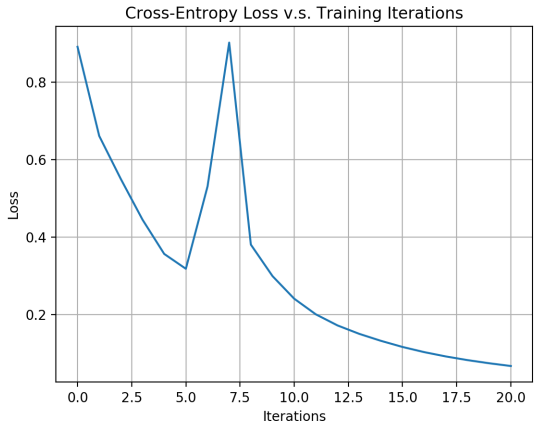

Figure 15: Cross-Entropy Loss vs Training Iterations

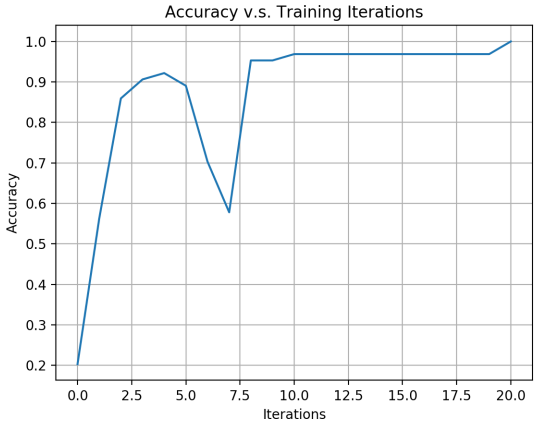

Figure 16: Accuracy vs Training Iterations

From the figures above, we learn the model in this question converges faster than the model in Part 2. The number of iterations is 20 in this question but is 917 in Part 2. Because we convolve the images before fully connected layer. However, in Part 2, we only do the fully connected network.

2. This figures are as followed.

   Comparing the results with the previous question, we find the model needs more number of iterations to converge for regression. But the number of iterations for classification is close to the model in previous question.

   Besides, there are more ups and downs in the curve because there are two backprops for this model, which can have more influence on the weights.

Figure 17: Cross-Entropy Loss vs Training Iterations

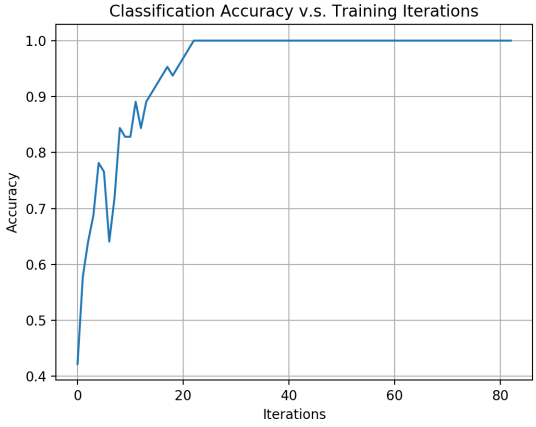

Figure 18: Cross-Entropy Loss vs Training Iterations

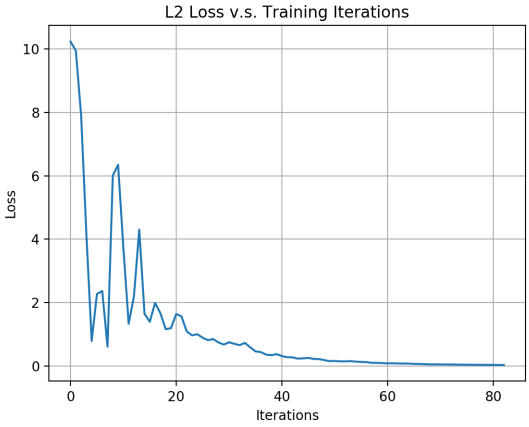

Figure 19: Cross-Entropy Loss vs Training Iterations

Figure 20: Cross-Entropy Loss vs Training Iterations