CIS 680: Project 1 Part A

Junfan Pan

09/13/2020

1 Plot Loss and Gradient

1.1 Sigmoid Activation Function
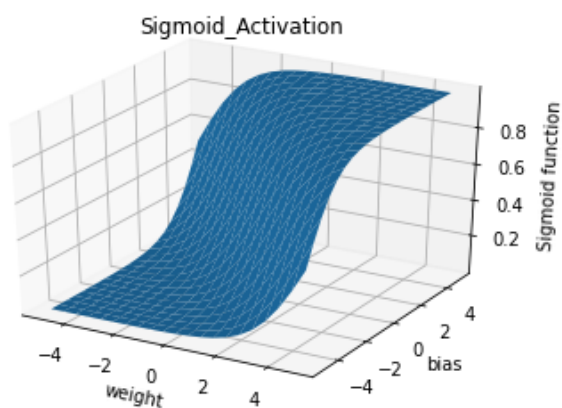
Figure 1: Sigmoid Function

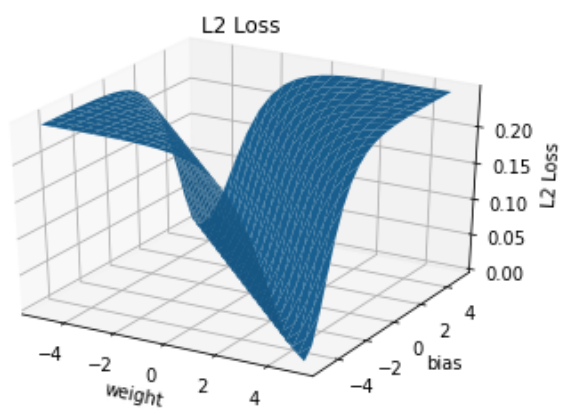1.2 L2 Loss
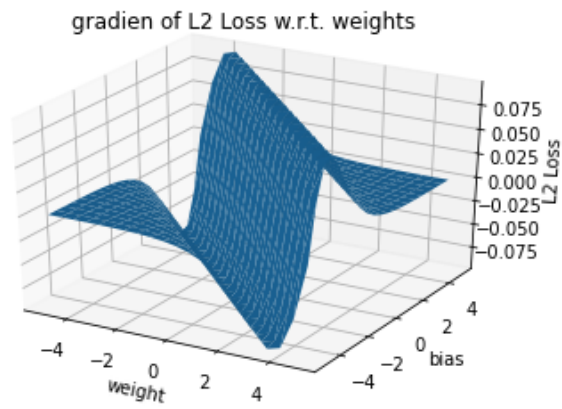
Figure 2: L2 Loss

1.3 L2 Loss Gradient

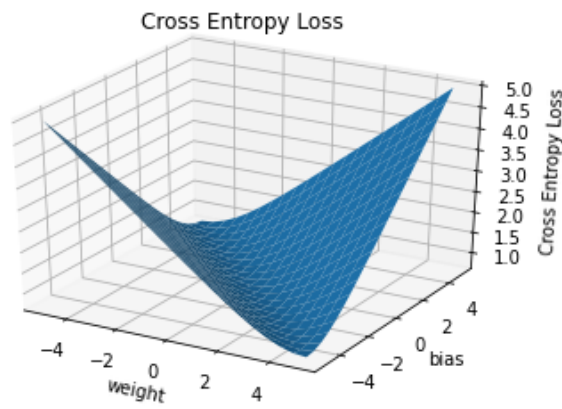Figure 3: L2 Loss Gradient

1.4 Cross Entropy Loss

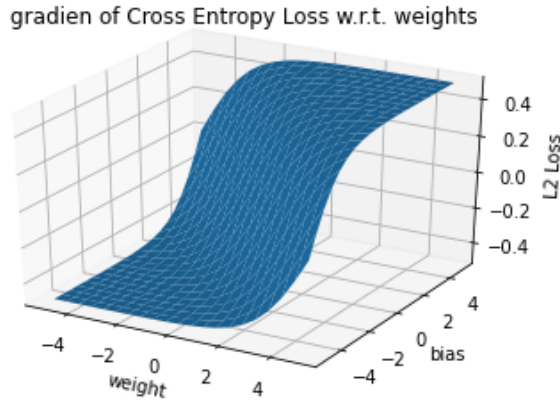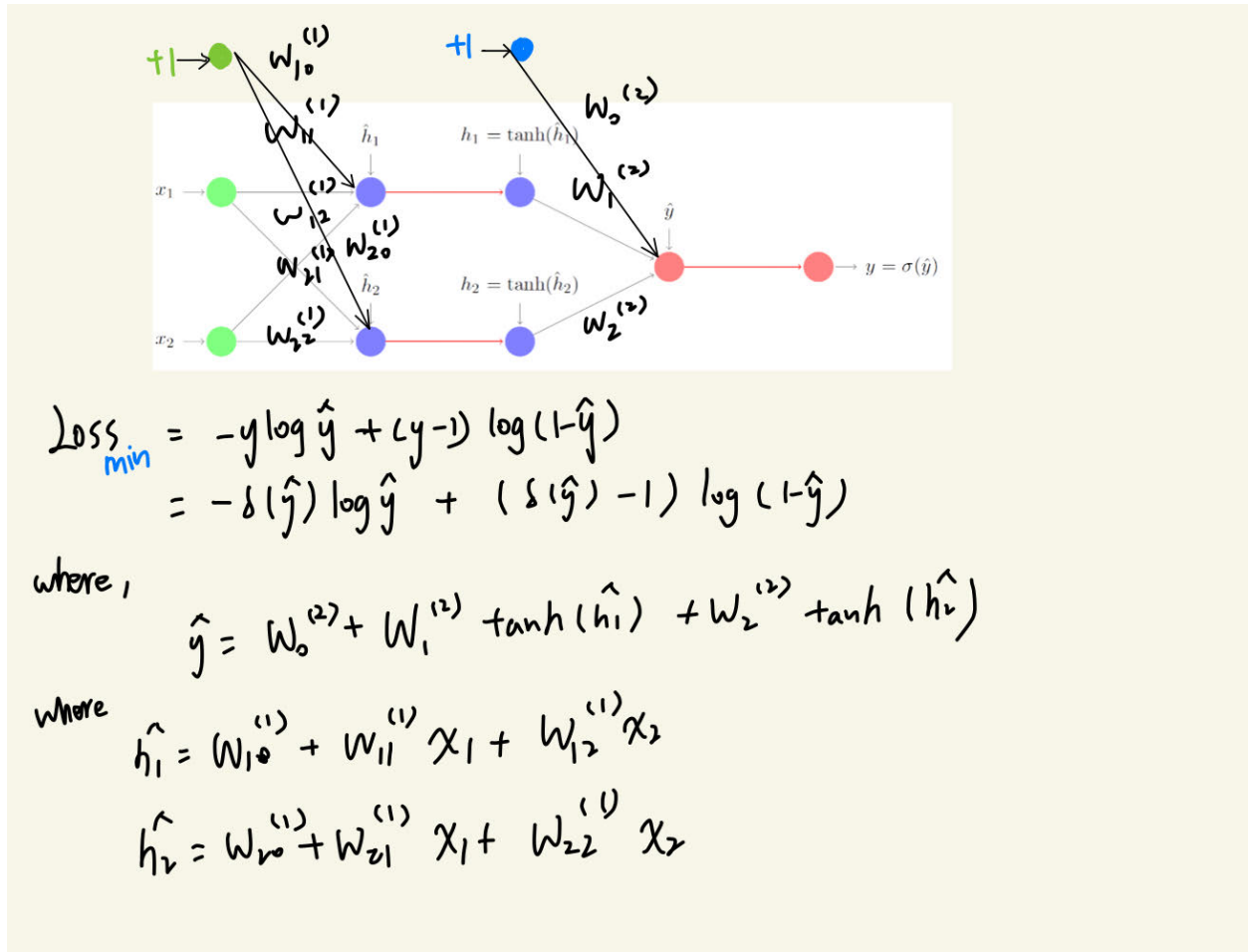Figure 4: Cross Entropy Loss

## 1.5 Cross Entropy Gradient

Figure 5: Cross Entropy Gradient

1.6 • What's the difference between cross-entropy loss and L2 loss?
From the z coordinates of Figure 2 and Figure 4, we can observe that in terms of the same value of weight and bias, the cross-entropy loss is much larger than L2 loss. In addition, the increment of cross-entropy which corresponds to the same amount of increment of weight and bias is much larger than this of L2 loss. Finally, the cross-entropy loss keep growing while the there is no obvious increase of L2 loss when the weight and bias reach some threshold.

• What's the difference between the gradients from cross-entropy loss and L2 loss?
From Figure 3 and Figure 5, we can see that the gradient of L2 loss is almost zero when the weight and bias are large. This means that the model learns little at the very start when the output is far from target. By comparing, the gradient of cross-entropy loss is stable and large which facilitates fast training of the model.

• Predict how these differences will influence the efficiency of learning?
Considering that the gradient of cross-entropy is much larger than the gradient of L2 loss under the same condition and especially at the start of training when error is large, therefore using the cross-entropy Loss can have better performance in achieving higher efficiency in training the model.
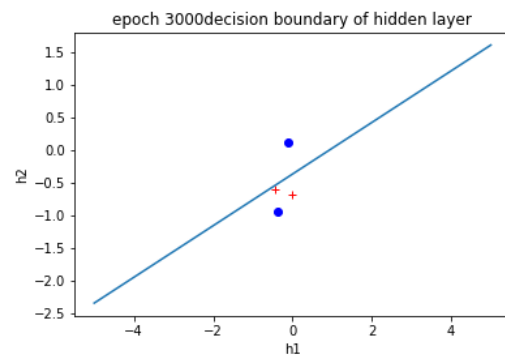
2 Solving XOR with a 2-layer Perceptron

2.1 Formulate the XOR approximation as an optimization problem using cross entropy loss. See below.

$$\text{Loss}_{\text{min}} = -y \log \hat{y} + (y-1) \log (1-\hat{y})$$

$$= -\delta(\hat{y}) \log \hat{y} + (\delta(\hat{y}) - 1) \log (1-\hat{y})$$

where,

$$\hat{y} = W_0^{(2)} + W_1^{(2)} \tanh(\hat{h_1}) + W_2^{(2)} \tanh(\hat{h_2})$$

where

$$\hat{h_1} = W_{10}^{(1)} + W_{11}^{(1)} X_1 + W_{12}^{(1)} X_2$$

$$\hat{h_2} = W_{20}^{(1)} + W_{21}^{(1)} X_1 + W_{22}^{(1)} X_2$$
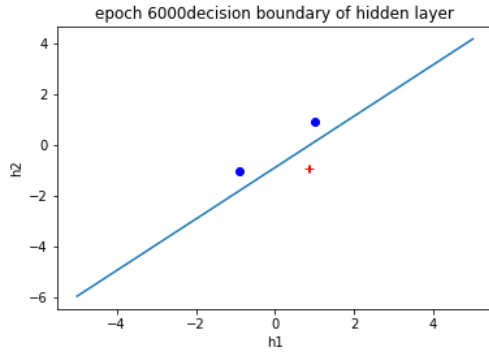
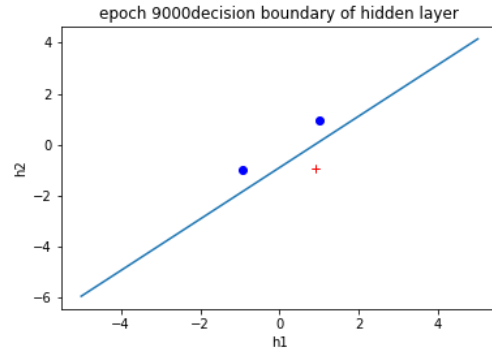2.2 The decision boundaries of different epochs are shown below:
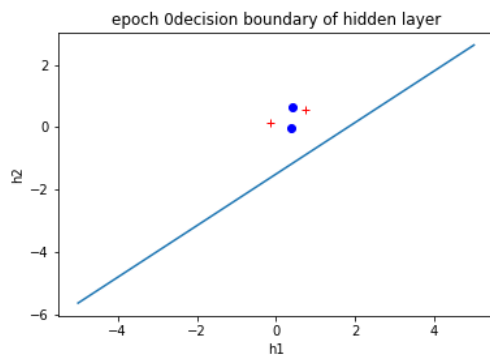
(a) Decision Boundary at Epoch 0

(b) Decision Boundary at Epoch 3000

(a) Decision Boundary at Epoch 6000


(b) Decision Boundary at Epoch 9000

The red cross represents (0,0) and (1,1) while the blue dot represents (0,1) and (1,0).

2.3 The network can not generate a single decision boundary to implement XOR. The reason is that if there is no activation function of the hidden layer, then the hidden layer will be a linear function. Therefore, the first layer plus the hidden layer are actually a single layer while it is known to all that it is impossible to achieve XOR using a single layer, which is actually a linear function.

3 Train a Convolutional Neural Network

3.1 See Python code in the colab notebook.

3.2 The learning curves are shown below:
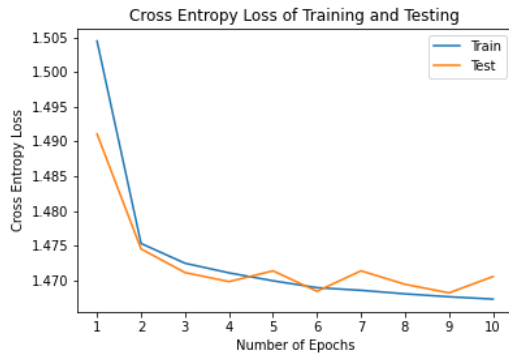
(a) MNIST Loss

(b) MNIST Accuracy

The model reaches 99% accuracy on the test dataset since the 3rd epoch.