Joe Phaneuf

Computer Vision 16-720 Spring 2018 Homework 3

Mar. 7, 2018

1 Q1

1.1 Q1.1

In training deep networks, a ReLU activation can be chosen over a sigmoid as it is easily differentiable, and does not suffer slow training due to saturation.

1.2 Q1.2

Consider a one layer network with weights W^1 , bias b^1 , input X and activation function g. The output of this network is

$$out^1 = g(W^1X + b^1)$$

if g is a linear operator function, then there will be some modified weights and biases such that

$$out^1 = \hat{W}^1X + \hat{b}^1$$

For a subsequent layer, the pre-activation is

$$W^2(\hat{W}^1X + \hat{b}^1) + b^2$$

Which, due to the properties of matrix multiplication, can be reduced to some modified weights and biases

$$\hat{W^2}X + \hat{b}^2$$

$\hat{W^2}$ and $\hat{b}^2$ are of the same dimension as W^1 and b^1, meaning no additional discriminatory information can be encapsulated by the second layer.

A non-linear activation function makes it impossible to combine weights as demonstrated above, meaning each additional layer does add to the discriminatory power of the network.

2 Q2

2.1 Q2.1.1

Weight initialization is an important factor for training neural networks. Naively initializing weights to zero results in zero valued gradients, which prevents weight updates. Initializing with all equal weights also reduces the same gradients for each weight, so training does not function properly.

2.2 Q2.1.3

I chose to initialize weights from a uniform distribution ranging from -0.1 to 0.1. This was mostly from research and some experimention. Weights that are too large can create issues by way of massive gradients, weights that are too small can take a long time to train due to very small gradients.

2.3 Q2.4.1

Stochastic gradient descent is more robust to non-convex error manifolds , but may take longer to reach minima. Batch gradient descent will typically converge faster given error manifolds that are reasonably convex , but can be more expensive in terms of required RAM.

2.4 Q3.1.1

Figure 1 shows the training and validation accuracy and loss while training on the nmist26 dataset.

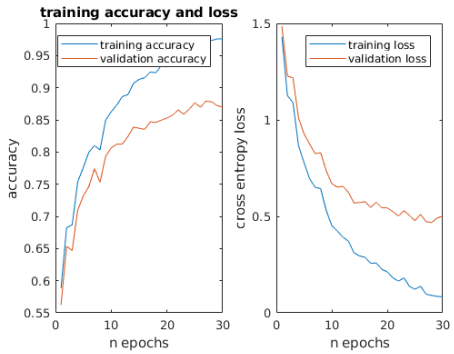

Figure 1: Training and validation accuracy and loss

2.5 Q3.1.2

Figure 1 above shows accuracy and loss for a learning rate of 0.01. Figure 2 shows accuracy and loss for a learning rate of 0.001. The higher learning rate of 0.01 performed better over 30 epochs, reaching a validation accuracy of 87 percent. versus the lower learning rate which achieved a validation accuracy of 73 percent. It is interesting to note that the accuracy and loss at the higher rate appear noisier. I suspect that too high of a learning rate could become problematic because of that.

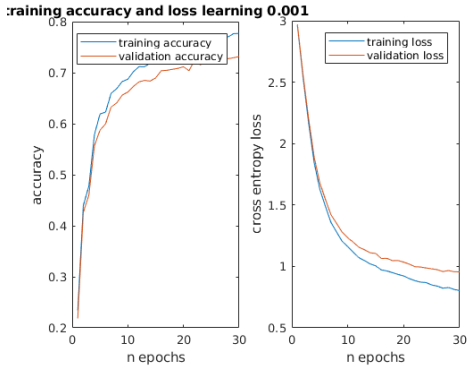

Figure 2: Training and validation accuracy and loss

2.6 Q3.1.3

As reported in section Q3.1.1 , this neural network provided the best valiation accuracy over 30 epochs using a learning rate of 0.01. Applying this network to the nist26 test data set resulted in 88.5 percent accuracy and a average cross entropy loss of 0.5. Figure 3 shows an image montage of weights in the first layer of this network. The first layer has 400 sets of 1024 length weight vectors.

Figure 5 shows the same montage with the weights initialized prior to training. The weights were initialized from a uniform distribution, and image appears to be white noise, which makes sense.

The learned weights clearly pick up on features that correspond to features of the training data set.

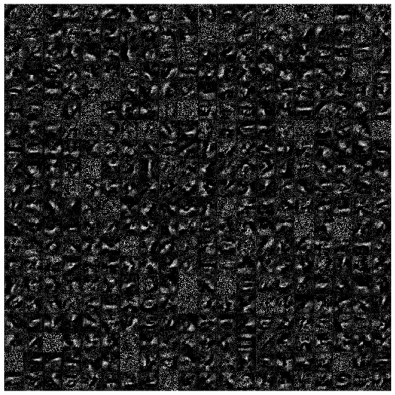

Figure 3: Visual montage of first layer weights after training

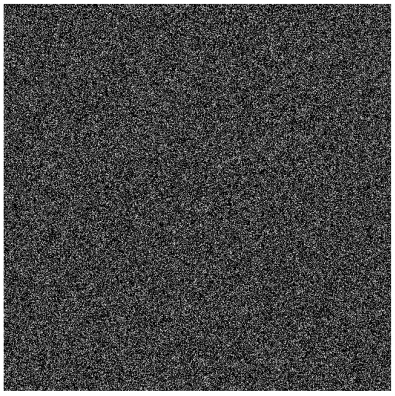

Figure 4: Visual montage of initial first layer weights

2.7 Q3.1.4

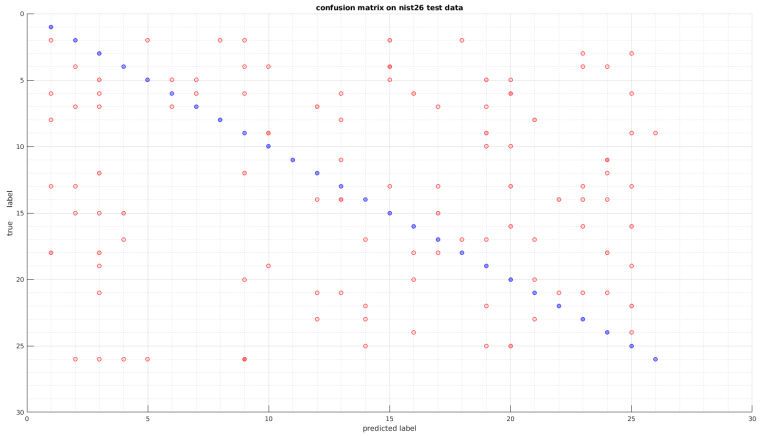

Figure 5: Confusion matrix for test data