Exercise 1 Report – Muhammad Hamiz Ahmed

The exercise was a good one to recap and renew old concepts of Neural Networks. The derivatives of the activation function were easy to compute and code. The code structure was very good and helpful. The bprop function in FullyConnectedLayer class took most of the time.

In the SoftmaxOutput class, the calculation of input_grad was a complex procedure. It was computed as follows. The softmax activation of the ith output unit is

$$y_i = \frac{e^{s_i}}{\sum_c^{nclass} e^{s_c}}$$

Thus, the error is

$$E = -\sum_i^{nclass} t_i \log(y_i)$$

Therefore, computing the gradient yields the following result

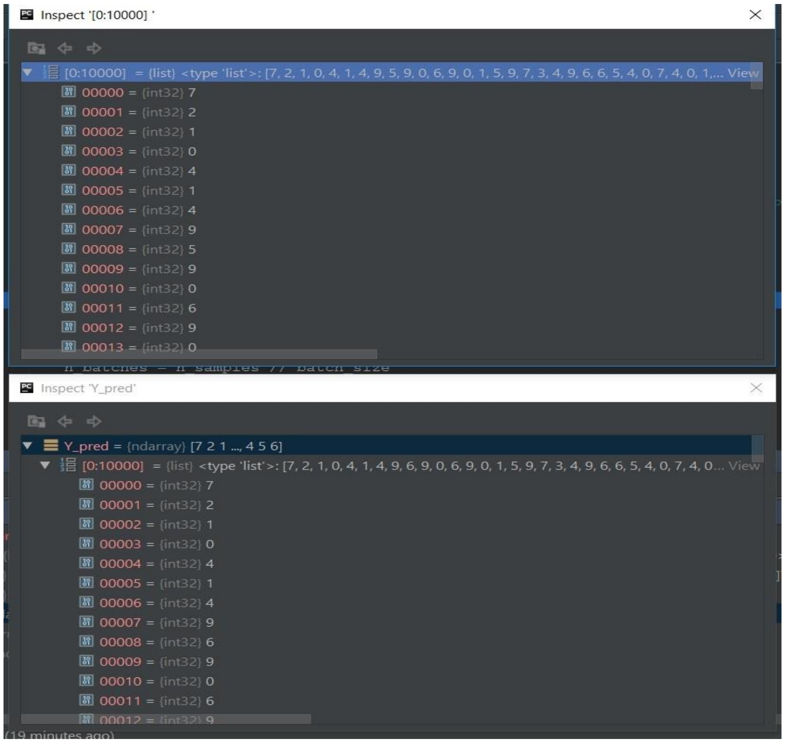
$$\begin{aligned}
\frac{\partial E}{\partial y_i} &= -\frac{t_i}{y_i} \\
\frac{\partial y_i}{\partial s_k} &= \begin{cases} \frac{e^{s_i}}{\sum_c^{nclass} e^{s_c}} - (\frac{e^{s_i}}{\sum_c^{nclass} e^{s_c}})^2 & i = k \\ -\frac{e^{s_i} e^{s_k}}{(\sum_c^{nclass} e^{s_c})^2} & i \neq k \end{cases} \\
&= \begin{cases} y_i(1 - y_i) & i = k \\ -y_i y_k & i \neq k \end{cases}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial E}{\partial s_i} &= \sum_k^{nclass} \frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial s_i} \\
&= \frac{\partial E}{\partial y_i} \frac{\partial y_i}{\partial s_i} - \sum_{k \neq i} \frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial s_i} \\
&= -t_i(1 - y_i) + \sum_{k \neq i} t_k y_k \\
&= -t_i + y_i \sum_k t_k \\
&= y_i - t_i
\end{aligned}$$

Where y_i is the predicted output

The neural network was setup with three FullyConnected layers. I changed the activation function of one layer to tanh which resulted in increased accuracy. There was a drastic difference in the accuracy of the network in in using the simple gradient descent approach, rather than stochastic descent. The stochastic descent yielded much better results. Similarly, increasing epochs, also resulted in an increased accuracy and reduced error of the test and validation set.

Following is the comparison between Predicted output (Y_pred) and actual output (Y)



```
[0:10000]  = {list} <type 'list'>: [7, 2, 1, 0, 4, 1, 4, 9, 5, 9, 0, 6, 9, 0, 1, 5, 9, 7, 3, 4, 9, 6, 6, 5, 4, 0, 7, 4, 0, 1,... View
    00000 = {int32} 7
    00001 = {int32} 2
    00002 = {int32} 1
    00003 = {int32} 0
    00004 = {int32} 4
    00005 = {int32} 1
    00006 = {int32} 4
    00007 = {int32} 9
    00008 = {int32} 5
    00009 = {int32} 9
    00010 = {int32} 0
    00011 = {int32} 6
    00012 = {int32} 9
    00013 = {int32} 0
```

```
Y_pred = {ndarray} [7 2 1 ..., 4 5 6]
  [0:10000]  = {list} <type 'list'>: [7, 2, 1, 0, 4, 1, 4, 9, 6, 9, 0, 6, 9, 0, 1, 5, 9, 7, 3, 4, 9, 6, 6, 5, 4, 0, 7, 4, 0... View
      00000 = {int32} 7
      00001 = {int32} 2
      00002 = {int32} 1
      00003 = {int32} 0
      00004 = {int32} 4
      00005 = {int32} 1
      00006 = {int32} 4
      00007 = {int32} 9
      00008 = {int32} 6
      00009 = {int32} 9
      00010 = {int32} 0
      00011 = {int32} 6
      00012 = {int32} 9
(19 minutes ago)
```

The error calculated in the test set was around 2.5%