Midterm Exam

SYS ENG 6213 - Deep Learning and Advanced Neural Networks

Fall 2020

Full Name: _____

Question	Score
True/False (20 pts)	
Short Answer (40 pts)	
Long Answer (40 pts)	

Good Luck!

1 True or False (20 pts)

1. **True/False** In dropout regularization, a very high drop probability causes underfitting.
2. **True/False** Removing regularization improves the training accuracy.
3. **True/False** Increasing the depth (number of layers) is the only way to increase the accuracy.
4. **True/False** You start training your neural network but after few epochs, the loss starts increasing instead of decreasing. The culprit is very small learning rate.
5. **True/False** Weight initializations doesnot affect the training performance as long as they are not set to 0.
6. **True/False** SGD with momentum has a tendency to overshoot at the global minima.
7. **True/False** An epoch is one complete pass for all training samples.
8. **True/False** In optimization, large gradient values ensures a good progress in the more gently sloped directions of parameter space.
9. **True/False** Introducing noise in training data has the effect of regularization.
10. **True/False** In bagging, a single model is trained multiple times to get multiple predictions on test data.

2 Short Answer (40 pts)

1. Explain the difference between gradient descent, mini-batch gradient descent and stochastic gradient descent (SGD). (5 pts)

2. Explain the reason why regularization is not applied to the biases. (5 pts)

3. What happens to the performance of neural network when the weights are initialized to zero. (5 pts)

4. Why does the loss function does not always decrease after a parameter update during optimization. (5 pts)

5. While training your neural network, you observe that the loss remains constant. Assuming that there are no errors in your code, explain the reasons behind such occurance. (5 pts)

6. Explain why adapting the learning rate in optimization techniques improves the training. (5 pts)

7. Given below is an implementation of sigmoid activation function (s) during backward pass:

```
def sigmoid_backward(grad):
    out = np.multiply(grad,np.subtract(1,grad))
    return(out)
```

   Observe the input and output values for the function:

```
[ 0.98,  0.96,  0.11,  0.12]
[ 1.  ,  0.11,  0.01,  0.92]
```

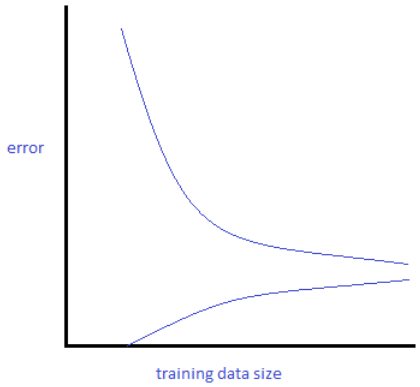
(a) Input

```
[ 0.0196,  0.0384,  0.0979,  0.1056]
[ 0.    ,  0.0979,  0.0099,  0.0736]
```

(b) Output

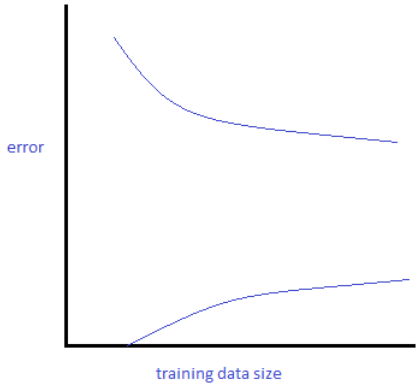
   Note that the gradient values given as input to the function are all at the extremes (near 0/1). Now explain why the values in $\frac{ds(x)}{dx}$ are not good for learning and how that problem is overcome in ReLu. (5 pts)

8. You train a neural network and observe that it has high validation error. You decide to increase the training data size and observe the behaviour oy your model.



From the above graph, identify whether the model is overfitting or underfitting. Give explanation.

What will be your answer if the behaviour in the plot looks like below:(5 pts)



In both the graphs, the bottom curve indicates the training error and upper curve indicates the test error.

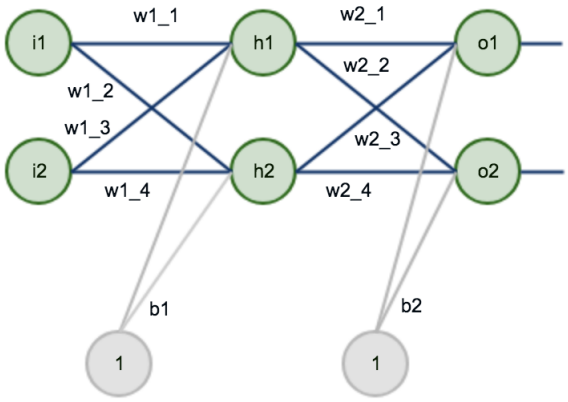
3 Long Answer (40 pts)

1. So far, in your assignemnts, you have used softmax cross entropy at the final layer to get the loss. For this problem, you have to derive the gradient with regard to the input of softmax (θ). In other words if $CE(y, \widehat{y}) = -\sum_i y_i log(\widehat{y})$ find $\frac{dCE(y,\widehat{y})}{d\theta}$ where $\widehat{y} = softmax(\theta)$.

   Note that $\widehat{y}$ is predicted probability vector for all classes and y is the one-hot label vecctor. (20 pts)

   (Hint: the y vector has all zero values except for one value say k. Show result for both $i = k$ and $i \neq k$.)

2. Compute the forward and backward pass for following neural network design with given values. Show your steps. (20 pts)



$$\begin{bmatrix}0.8\\0.3\end{bmatrix} \quad \begin{bmatrix}0.24 & 0.67\\0.43 & 0.14\end{bmatrix} \quad \begin{bmatrix}1\\1\end{bmatrix} \quad \begin{bmatrix}0.34 & 0.63\\0.81 & 0.72\end{bmatrix} \quad \begin{bmatrix}1\\1\end{bmatrix} \quad \begin{bmatrix}0\\1\end{bmatrix}$$

The matrices are in the order of: x, $w1\_x$, $b1$, $w2\_x$, $b2$ and y (targets) where $x \in \{1, 2, 3, 4\}$. The activation function is **ReLu** and the Loss is calculated using **softmax**. You can use a learning rate of 0.1.