Midterm Exam

SYS ENG 6213 - Deep Learning and Advanced Neural Networks

Fall 2020

Full Name: Ryan Patton

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.

   T

2. **True/False** Removing regularization improves the training accuracy.

   F

3. **True/False** Increasing the depth (number of layers) is the only way to increase the accuracy.

   F

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.

   F (confirmed – large learning rate)

5. **True/False** Weight initializations does not affect the training performance as long as they are not set to 0.

   F

6. **True/False** SGD with momentum has a tendency to overshoot at the global minima.

   T

7. **True/False** An epoch is one complete pass for all training samples.

   T

8. **True/False** In optimization, large gradient values ensures a good progress in the more gently sloped directions of parameter space.

   F (large learning rate instead of large gradient values)

9. **True/False** Introducing noise in training data has the effect of regularization.

   T

10. **True/False** In bagging, a single model is trained multiple times to get multiple predictions on test data.

    F (multiple models)

2 Short Answer (40 pts)

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

   Gradient descent optimizes the parameters of a model using the gradient of an objective function (loss function). It optimizes the parameters until the value of the loss function is the minimum (or the minima of the loss function is reached), often referred to simply as backpropogation.

   SGD computes the gradients for every sample in the dataset and makes an update for every sample in the dataset. For 100 samples in the dataset, there would be 100 updates.

   Mini-batch gradient descent builds on SGD and gradient descent. Instead of a single sample in SGD, small batches of the dataset are taken and updated with the proper parameters. For example, if a dataset contained 100 samples, a batch size of 5 would mean there are 20 batches, and updates occur 20 times.

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

   The bias parameters don't contribute to the curvature of the model so it would be pointless. When interpolating target values, a lot of curvature is needed to fit the function. Overfitting requires the output of the model to be sensitive to such small changes in the input data.

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

   When the weights are initialized to zero, they all lead the neurons to learn the same features during training.

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

   The samples for each class contained in successive batches may be so noisy that the weight updates is erratic, and therefore the value of the loss function oscillates in the error-weight space. If the bath size were to expose the neural network to more samples per class every iteration, the less influence noisy samples would have on the weights being updated so the loss experienced would be minimizes with little to no oscillation.

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)

   The loss could remain constant due to an incorrectly defined architecture for the problem or noisy data being fed into the model.

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

Adapting the learning rate in optimization techniques improves the training because it balances how fast you want the model to learn with how optimal the final set of weights are. If the learning rate is too large, the resultant updated weights will be too large and the performance of the model will oscillate over training epochs. A learning rate that is too small may never converge or get stuck in a suboptimal solution.

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

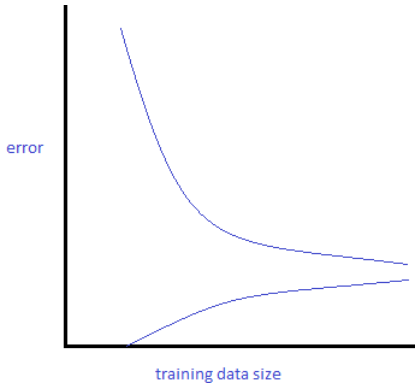
```
[ 0.0196,  0.0384,  0.0979,  0.1056]
[ 0.    ,  0.0979,  0.0099,  0.0736]
```

(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)

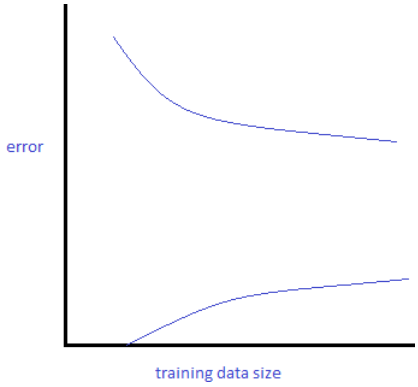
The values in $\frac{ds(x)}{dx}$ are not good for learning because they're all close to either 0 or 1, causing a vanishing gradient. This is so because the gradient of Sigmoid is S'(x) = S(x) * (1 – S(x)) so for inputs you'd end up with values close to (1) * (1 – 1) = 0. Relu overcomes this because there's no mechanism to constrain the output of the neuron as the x values themselves are the outputs.

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 of 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.

The top graph indicates underfitting and the bottom graph indicates overfitting. The top graph indicates underfitting because the training error and test error start off far away from each other but by the end, they're converging to high bias and low variance. It doesn't account for the data provided and the training set isn't expressive enough. A large gap between the training and test error indicates overfitting as seen in the bottom graph because there's high variance and low bias. The classifier learned on the training set is too specific, and cannot be used to accurately infer anything about unseen data.

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 $CE(y, y\hat{}) = -\sum_i y_i log(\hat{y}_i)$ find $\frac{dCE(y,\hat{y})}{d\theta}$ where $\hat{y} = softmax(\theta)$.

   Note that $y\hat{}_i$ 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 $i \neq k$.)

if i = k:

$$= \frac{\partial y_i}{\partial theta_i}$$

$$= \frac{\partial \frac{e^{theta_i}}{\sum C}}{\partial theta_i}$$

$$= \frac{e^{theta_i} \sum C - e^{theta_i}\, e^{theta_i}}{\sum_C^2}$$

$$= \frac{e^{theta_i}}{\sum C} \frac{\sum C - e^{theta_i}}{\sum C}$$

$$= \frac{e^{theta_i}}{\sum C} \left(\frac{1 - e^{theta_i}}{\sum C} \right)$$

$$= y_i(1 - y_i)$$

if i does not equal k:

$$= \frac{\partial y_i}{\partial theta_k}$$

$$= \frac{\partial \frac{e^{theta_i}}{\sum C}}{\partial theta_k}$$

$$= \frac{0 - e^{theta_i} e^{theta_k}}{\sum_C^2}$$

$$= -\frac{e^{theta_i}}{\sum C} \frac{e^{theta_k}}{\sum C}$$

$$= -y_i y_k$$

$$\text{dCE(y, y')/d(theta)} = -\sum_{k=1}^{CE} \frac{\partial y_k \log(yhat_k)}{\partial (theta)_k}$$

$$= -\sum_{k=1}^{CE} \frac{\log(yhat_k)}{\partial (theta)_i} * \ y_k$$

$$= -\sum_{k=1}^{CE} y_k \frac{1}{yhat_k} \frac{\partial yhat_k}{\partial (theta)_i}$$

$$= -\frac{y_i}{yhat_i} \frac{\partial yhat_i}{\partial theta_i} - \sum_{k\neq 1}^{CE} \frac{y_k}{yhat_k} \frac{\partial yhat_k}{\partial (theta)_i}$$

$$= -\frac{y_i}{y'_i} yhat_i(1 - yhat_i) - \sum_{k\neq i}^{CE} (-yhat_k yhat_i)$$

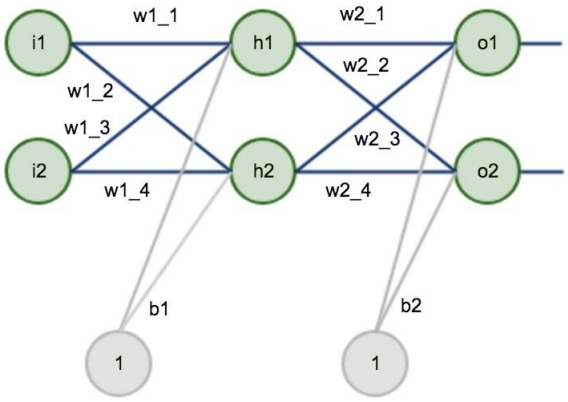
$$= -y_i + y_i yhat_i + \sum_{k\neq i}^{CE} (y_k yhat_i)$$

$$= -y_i + \sum_{k=1}^{CE} (y_k yhat_i)$$

$$= -y_i + yhat_i \sum_{k=1}^{CE} (y_k)$$

$$= yhat_i - y_i$$

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



$$\begin{array}{cc|cc|cc|cc|cc} 1 & & 1 & & 1 & & 1 & & 1 & 1 \\ 0.8 & & 0.24 & 0.67 & 1 & & 0.34 & 0.63 & 1 & 0 \\ 0.3 & & 0.43 & 0.14 & 1 & & 0.81 & 0.72 & 1 & 1 \end{array}$$

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.

T

Resources

https://datascience.stackexchange.com/questions/53870/how-do-gd-batch-gd-sgd-and-mini-batch-sgd-differ

https://stats.stackexchange.com/questions/153605/no-regularisation-term-for-bias-unit-in-neural-network

https://www.deeplearning.ai/ai-notes/initialization/

https://www.researchgate.net/post/Why_MLP_is_not_converging

https://www.analyticsvidhya.com/blog/2017/08/skilltest-deep-learning/

https://machinelearningmastery.com/learning-rate-for-deep-learning-neural-networks/

https://stats.stackexchange.com/questions/126238/what-are-the-advantages-of-relu-over-sigmoid-function-in-deep-neural-networks

https://peterroelants.github.io/posts/cross-entropy-softmax/