

# Assignment 1 Theory Problem Set

**DO NOT TAG**

Name: Michael Fox  
GT Email: mfox66@gatech.edu

Theory PS Q1. Feel free to add extra slides if needed.

1. Define  $s(\vec{z})$ , which takes a vector input  $\vec{z}$  and outputs a vector whose  $i$ th entry is given by

$$s_i := \frac{e^{z_i}}{\sum_k e^{z_k}}$$

Derive  $\frac{\partial s}{\partial \mathbf{z}}$ .

Solution: Taking the partial derivative of  $s$  with respect to  $\mathbf{z}$  we obtain

$$\frac{\partial s_i}{\partial z_j} = \frac{\partial}{\partial z_j} \left[ \frac{e^{z_i}}{\sum_k e^{z_k}} \right]$$

When  $i=j$  we obtain

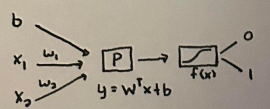
$$\begin{aligned} \frac{\partial}{\partial z_j} \left[ \frac{e^{z_i}}{\sum_k e^{z_k}} \right] &= \frac{\sum_k e^{z_k} (e^{z_i}) - e^{z_i} e^{z_j}}{(\sum_k e^{z_k})^2} \\ &= \frac{e^{z_i}}{\sum_k e^{z_k}} \cdot \frac{(1 - e^{z_j})}{\sum_k e^{z_k}} \\ &= s_i (1 - s_j) \end{aligned}$$

Similarly, we  $i \neq j$  we obtain  $\frac{\partial}{\partial z_j} \left[ \frac{e^{z_i}}{\sum_k e^{z_k}} \right] = \frac{0 - e^{z_i} e^{z_j}}{(\sum_k e^{z_k})^2} = -s_i s_j$

Thus we have  $\frac{\partial s}{\partial \mathbf{z}} = \mathbf{s} (\mathbf{\delta}_{ij} - s_j)$  where  $\delta_{ij} = \begin{cases} 1 & \text{if } i=j \\ 0 & \text{if } i \neq j \end{cases}$

Theory PS Q2. Feel free to add extra slides if needed.

a.



$x_1$	$x_2$	$f_{A=0}(t)$
0	0	0
0	1	0
1	0	0
1	1	1

$$y = wx + b$$

$$\begin{cases} y = wx + b \\ w \leftarrow w - \alpha(t-a)x_i; f(x) = \begin{cases} 1 & \text{if } w^T x + b \geq 0 \\ 0 & \text{if } w^T x + b < 0 \end{cases} \\ b \leftarrow b - \alpha(t-a) \end{cases}$$

Step 1: Initialize all weights and bias to zero and let  $\alpha=1$ .  
Thus  $w_1=w_2=b=0$  and  $\alpha=1$ .

Step 2: First forward pass:

$$y = w_1 x_1 + w_2 x_2 + b$$

$$= (0)(0) + (0)(0) + (0) = 0$$

$$y = 0 \xrightarrow{f} f(0) = 1.$$

Step 3: Update the weights & bias:

$$w_1 = w_1 + \alpha(t-a)x_1$$

$$= (0) + 1(1-0)(0) = 0$$

$$w_2 = w_2 + \alpha(t-a)x_2$$

$$= (0) + 1(1-0)(0) = 0$$

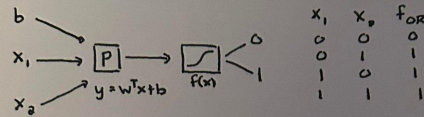
$$b = (0) + 1(1-0) = 1$$

Step 4: Iterating Steps (2) and (3) until convergence of the weights we obtain

$$\begin{aligned} w_1 &= 0.4 \\ w_2 &= 0.4 \\ b &= -0.6 \end{aligned}$$

Theory PS Q3. Feel free to add extra slides if needed.

3.



$$\begin{cases} y = wx + b \\ w \leftarrow w + \alpha(t-a)x; \\ b \leftarrow b + \alpha(t-a) \end{cases} \quad f(x) = \begin{cases} 1 & \text{if } w^T x + b \geq 0 \\ 0 & \text{if } w^T x + b < 0 \end{cases}$$

Step 1: Initialize all weights and bias to zero and let  $\alpha=1$ : Thus  $w_1=w_2=b=0$  and  $\alpha=1$

Step 2: First forward Pass:

$$\begin{aligned} y &= w_1 x_1 + w_2 x_2 + b \\ &= (0)(0) + (0)(0) + (0) = 0 \\ y=0 &\xrightarrow{f} f(0) = 1. \end{aligned}$$

Step 3: Update the weights and bias:

$$\begin{aligned} w_1 &= w_1 + \alpha(t-a)x_1 \\ &= (0) + 1(0-1)(0) = 0 \\ w_2 &= w_2 + \alpha(t-a)x_2 \\ &= (0) + 1(0-1)(0) = 0 \\ b &= b + \alpha(t-a) \\ &= (0) + 1(0-1) = -1 \end{aligned}$$

Step 4: Iterating steps (2) and (3) until convergence of the weights we obtain

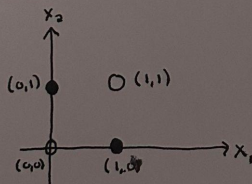
$$\begin{aligned} w_1 &= 1.0 \\ w_2 &= 1.0 \\ b &= -0.1 \end{aligned}$$

3. Prove that XOR can NOT be represented using a linear model with the same form as (2).

Solution: For the XOR function, we have

$x_1$	$x_2$	$f_{\text{XOR}}$
0	0	0
0	1	1
1	0	1
1	1	0

Plotting the set of input values together with their corresponding truth values we obtain the following graph:



where  $\circ$  denotes a truth value of 0 and  $\bullet$  denotes a truth value of 1 for the corresponding pair of input values  $(x_1, x_2)$ . From the above graph we can see that there does not exist a single line which can separate the two classes. Thus the XOR function is not linearly separable and hence can not be represented by a single layer perceptron. In fact, we can see that two such lines would be needed in order to separate the classes. Hence a multilayer perceptron is needed to represent the XOR function.

# Assignment 1 Paper Review

**DO NOT TAG**

Name: Michael Fox  
GT Email: mfox66@gatech.edu

## Paper: Weight Agnostic Neural Networks

Link: <https://arxiv.org/pdf/1906.04358.pdf>

### Questions:

1. What is the main contribution of this paper? In other words, briefly summarize its key insights. What are some strengths and weaknesses of this paper?
  - a. This paper seeks to study the effects of network architecture on specific problem domains. Unlike the traditional optimization process in which the weights of a model are updated through a predefined loss function, the authors of this paper instead focus on aligning a problem domain with a specific network architecture. In order to minimize the effects of the weights on model performance, the weights are sampled from a uniform distribution.
  - b. In order to align a network architecture to a given problem domain, the authors focus on network minimization, that is finding the smallest network possible to achieve a measure of success on a given task. This is achieved by starting with a predefined minimal network architecture and through an iterative process adding a single node, adding a connection to that node with the weights being randomly sampled from a uniform distribution, and an activation function being randomly chosen from a predefined list of possible activation functions. This process continues until a minimal network architecture is found which achieves a given measure of success.
  - c. Strength: The main strength of the proposed method is the reduction in computational complexity as the weights are sampled from a uniform distribution and hence do not need to be optimized through a traditional loss function. Additionally, once a minimal network has been found which achieves a measure of success on a given problem domain, further optimization of the weights can be carried out through the traditional method. This serves as a benchmark and requires less optimization to achieve the same level of results.
  - d. Weakness: The main weakness of the proposed method is its inability to represent higher dimensional data, as sampling higher dimensional data is difficult and inefficient.
2. What is your personal takeaway from this paper?
  - a. This paper highlights the need to find a balance between the traditional method of optimization of the weights by a predefined loss function and the proposed method of focusing on finding the minimal network topology needed to achieve a measure of success on a given problem domain.

3. The traditional view of optimization in deep learning (and often in general) is that we are searching the space of weights to find the best ones. In other words, learning is a search problem. How would you view the above paper from the perspective of search?

- a. In this paper, the traditional search problem is shifted from a search of the weights which will optimize a predefined loss function, instead to a search of minimal network topologies which can serve as a better benchmark for fine tuning of the weights.

4. One of the key aspects of deep learning is that given a parameterized function, we can find weights to represent any function if it has sufficient depth and complexity. What does this paper say about the representational power of architectures given a fixed method for determining weights? Does the method for determining the weights matter? Do you think these two have equal representational power? Why or why not?

- a. The representational power of architectures given a fixed method for determining the weights will have the same representational power as architectures that have the weights optimized through a predefined loss function. This is due to the fact that the fixed weight architectures will go through an iterative process in order to find a minimal network work topology that will achieve a fixed level of representation. Additionally, the activation functions sampled from the fixed weight architecture can be the same as those chosen in the architectures which are optimized through a fixed loss function.



# Assignment 1 Writeup

**DO NOT TAG**

Name: Michael Fox  
GT Email: mfox66@gatech.edu

# Two-Layer Neural Network

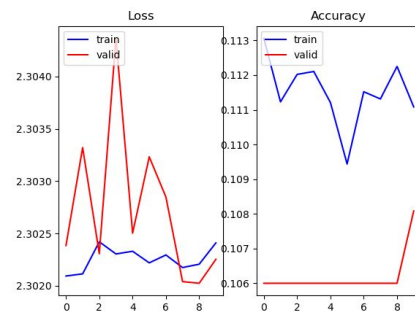
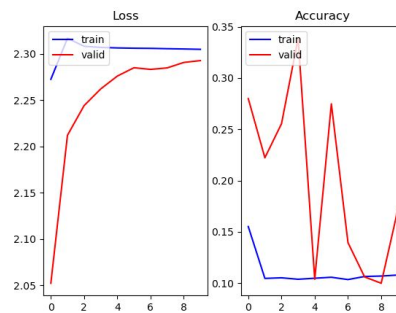
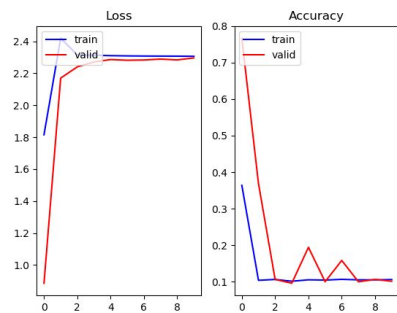
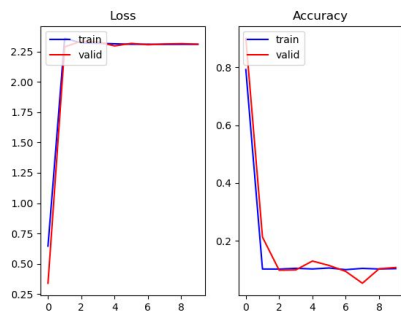
**DO NOT TAG**

# 1. Learning Rates

Tune the learning rate of the model with all other default hyper-parameters fixed.  
Fill in the table below:

	lr=1	lr=1e-1	lr=5e-2	lr=1e-2
Training Accuracy	0.9375	0.75	0.375	0.0625
Test Accuracy	0.8992	0.7616	0.3380	0.1028

# 1. Learning Curve



# 1. Learning Rates

Describe and Explain your findings: As we can see from the above graphs, larger values for the learning rate, around 1, give the best overall performance.

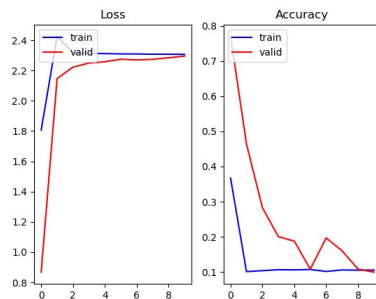
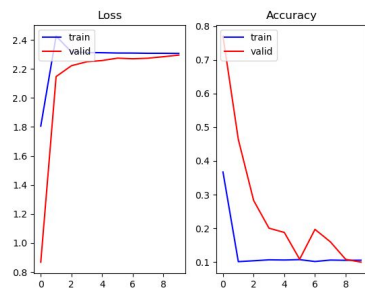
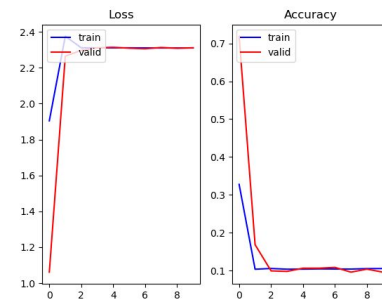
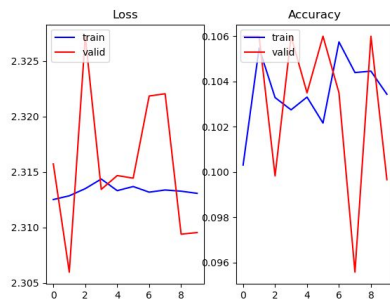
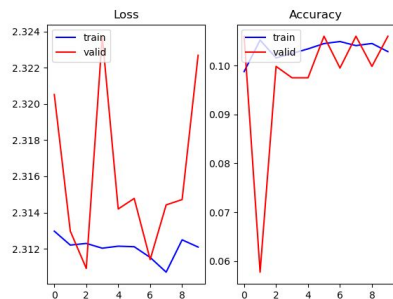
Decreasing values of the learning rate decrease the overall performance of the model. As the value of the learning rate is decreased, more training epochs will be needed in order to see improvement to the overall model performance. This is because a smaller learning rate means that smaller steps are taken in the opposite direction of the gradient and hence more steps will be needed in order to achieve a local or global optimum. However, if the learning rate is too large we risk overstepping a minima negatively impacting the performance.

## 2. Regularization

Tune the regularization coefficient of the model with all other default hyper-parameters fixed. Fill in the table below:

	alpha=1	alpha=1e-1	alpha=1e-2	alpha=1e-3	alpha=1e-4
Training Accuracy	0.125	0.125	0.6875	0.75	0.75
Validation Accuracy	0.1135	0.1135	0.7168	0.7616	0.7616
Test Accuracy	0.1135	0.1135	0.7168	0.7616	0.7616

## 2. Regularization



## 2. Regularization

Describe and Explain your findings: As we can see from the above graphs, we have that larger values for the regularization factor, around 1, negatively affect the performance of the model. As the regularization factor decreases this negative affect is mitigated until the effect of the regularization factor is negligible. The regularization factor attempts to prevent the negative influence of increasingly large weights by introducing a negative scaling factor to the weights, thus preventing them from increasing to the point of negatively affecting the model. However, since our model was only trained for a total of ten epochs across all trials, the weights do not have a chance to sufficiently increase to the point of hurting performance. Thus, introducing a regularization factor under these parameters will negatively affect the performance of the model.



### 3. Hyper-parameter Tuning

batch size = 32	lr = 0.99	reg = 0.0
Train acc = 1.0	Test acc = 0.9374	

Explain why your choice works: First we decreased the batch size to 32. This is to prevent over sampling during the training process leading to a decreased ability of the model to generalize. Next we set the learning rate to 0.99. As we have discussed above this value for the learning rate is sufficiently large for the model to learn without being so large that we risk overstepping a minima. And finally, we set the regularization factor to 0 as the number of epochs is small enough that increasing the regularization factor will negatively bias the weights.