CS 4644/7643: Deep Learning
Spring 2022
HW1 Solutions

Jieun Seong

February 5, 2022

1 Collaborators

Yuankai Cai, Kevin George, Kelian Li, Guangyu Min

2 Optimization

1. Two vectors are orthogonal if their dot product is 0, so let's check if $\frac{\partial \mathbf{r}}{\partial t} \cdot \nabla f_0 = 0$.

$$\begin{aligned}\frac{\partial \mathbf{r}}{\partial t} \cdot \nabla f_0 &= \left[\frac{\partial x_1}{\partial t}, \frac{\partial x_1}{\partial t}, \cdots, \cdots, \frac{\partial x_1}{\partial t}\right] \cdot \left[\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \cdots, \frac{\partial f}{\partial x_d}\right] \\ &= \frac{\partial x_1}{\partial t}\frac{\partial f}{\partial x_1} + \cdots + \frac{\partial x_d}{\partial t}\frac{\partial f}{\partial x_d} = \frac{\partial f}{\partial t} + \cdots + \frac{\partial f}{\partial t} = 0\end{aligned}$$

   The gradient is orthogonal to the tangent. To get the optimum, we need to move towards the steepest descent. Once we have the tangent, the descent is orthogonal to it, so it's helpful in deep learning.

2. (a) The gradient at $\mathbf{w}^t$ is, by definition,

$$\nabla g(\mathbf{w}^t) = \begin{bmatrix} \frac{\partial g}{\partial w_1}(\mathbf{w}^t) \\ \vdots \\ \frac{\partial g}{\partial w_n}(\mathbf{w}^t) \end{bmatrix}, \frac{\partial g}{\partial w_i}(\mathbf{w}^t) = \lim_{\mathbf{h}_i \to 0} \frac{g(\mathbf{w}^t + \mathbf{h}) - g(\mathbf{w}^t)}{\mathbf{h}_i}, \mathbf{h} \in \mathbb{R}^n.$$

Since g is differentiable, the limit exists for all i. Also, as $\mathbf{h}_i \to 0^+$, $g(\mathbf{w}^t + \mathbf{h}) - g(\mathbf{w}^t) \geq 0$ and, as $\mathbf{h}_i \to 0^-$, $g(\mathbf{w}^t + \mathbf{h}) - g(\mathbf{w}^t) \leq 0$. Hence, $\frac{\partial g}{\partial w_i}(\mathbf{w}^t) = 0 \ \forall i$, and thus $\nabla g(\mathbf{w}^t) = 0$. □

(b) Let's find a counterexample to disprove the converse. Let $g : \mathbb{R} \to \mathbb{R}$, $g(x) = -x^2$. $g'(0) = 0$, but $g(y) < g(0)$ for all $y \neq 0$, so $\mathbf{w}^* = 0$ is a maximum, not minimum. Thus, the converse is false. □

3. Let g be a convex, differentiable function and $\nabla g(\mathbf{w}^*) = 0$. We need to show that $\forall \mathbf{w} \in \mathbb{R}^n$, $g(\mathbf{w}^*) \leq g(\mathbf{w})$. Since the function is convex, the first order Taylor expansion at any point will always underestimate the function:

$$g(y) \geq g(x) + \nabla g(x)^T (y - x), \forall x, y \in \mathbb{R}^n.$$

$\nabla g(\mathbf{w}^*) = 0$, so

$$g(y) \geq g(\mathbf{w}^*) + \nabla g(\mathbf{w}^*)^T (y - \mathbf{w}^*) = g(\mathbf{w}^*), \forall y \in \mathbb{R}^n.$$

Hence, $\mathbf{w}^*$ is the global minimum of g. □

4. Let N and M be the lengths of $\mathbf{s} = [s_1, \cdots, s_N]$ and $\mathbf{z} = [z_1, \cdots, z_M]$ respectively.

$$\frac{\partial \mathbf{s}}{\partial \mathbf{z}} = \begin{bmatrix} \frac{\partial s_1}{\partial z_1} & \cdots & \frac{\partial s_1}{z_M} \\ \vdots & \ddots & \vdots \\ \frac{\partial s_N}{z_1} & \cdots & \frac{\partial s_N}{\partial z_M} \end{bmatrix}, \frac{\partial s_i}{\partial z_k} = \begin{cases} e^{z_i} \left(\sum_a e^{z_a}\right)^{-1} - e^{2z_i} \left(\sum_a e^{z_a}\right)^{-2}, i = k \\ -e^{z_i + z_k} \left(\sum_a e^{z_a}\right)^{-1}, i \neq k \end{cases}$$

5.

$$s(x) = \underset{\substack{y\in\mathbb{R}^d, \|y\|_1=1 \\ 0\le y_i\le 1}}{\operatorname{argmin}} \; -x^T y + \sum_i y_i \ln(y_i)$$

To find argmin, I'm going to take the gradient of the function $-x^T y + \sum_i y_i \ln(y_i)$ and find the values of y_i that will make the gradient zero.

$$\nabla_y \left(-x^T y + \sum_i y_i \ln(y_i) \right) = \begin{bmatrix} -x_1 + \ln(y_1) + 1 \\ \vdots \\ -x_d + \ln(y_d) + 1 \end{bmatrix} = 0 \Rightarrow y = \frac{1}{e} \begin{bmatrix} e^{x_1} \\ \vdots \\ e^{x_d} \end{bmatrix}$$

The condition we have on y is that $\|y\|_1 = 1$ and $0 \le y_i \le 1$, so we need to normalize y so that the conditions are met.

$$\frac{1}{e} \begin{bmatrix} e^{x_1} \\ \vdots \\ e^{x_d} \end{bmatrix} \to \frac{1}{\sum_i e^{x_i}} \begin{bmatrix} e^{x_1} \\ \vdots \\ e^{x_d} \end{bmatrix}.$$

This is precisely the definition of softmax. □

We can say that the softmax is the optimal solution for this optimization problem.

3 Directed Acyclic Graphs (DAG)

1. Let's prove by induction on the number of vertices. It's obviously true when there is one vertex. Suppose the statement is true for a DAG with k vertices. Let's show it's also true for a DAG with $k+1$ vertices. Let G be a DAG with $k+1$ vertices and find a vertex v that has no incoming edges (we know such node exists because of the lemma proven below). Removing a node from a DAG cannot create a cycle, so $G' = G - \{v\}$ is a DAG with k vertices. By the hypothesis, G' has a topological ordering. Place v first in the topological ordering and then add nodes of G'. This is a valid topological ordering, as v does not have incoming edges. □

   Lemma. If G is a DAG, then G has a node with no incoming edges.

   Pf. Suppose every node of G has at least one incoming edge. Let v be any node and follow one of the incoming edges backward. Since every node has at least one incoming edge, we can continue the process and will visit at least one node twice. This is a cycle. Contradiction. □

2. Let's prove by contradiction. Suppose G has a topological order, $\{v_1, \cdots, v_n\}$, and has a cycle. Let v_i be the node of lowest order in C and v_j be the node before v_i such that (v_j, v_i) is an edge of the cycle. Then, by the topological order, $j < i$, but v_i is the lowest index in C, so $i < j$. Contradiction. □

4 Paper Review

1. Learning weight parameters has been a central paradigm in the field of neural networks for many years, but the paper suggests that we can find network architectures that can already perform well without learning the weight parameters. Its biggest contribution is suggesting the new direction of research for neural networks – instead of training weights, we can find architectures with biases and random shared weights. This will increase performance as there's no weight training. However, they rely on the fact that there is sufficient knowledge from each domain and that the biases for the domains are already known. This could be okay for the intensely studied areas, but for the domains that are not, it could be hard to apply their method. The conventional training for neural networks will be necessary to obtain the biases.

2. This could be developed further into the direction of solving discrete systems where gradients cannot be applied.

Assignment 1 Writeup

●Name: Jieun Seong
●GT Email: jseong8@gatech.edu
●GT ID: 903004701

Two-Layer Neural Network

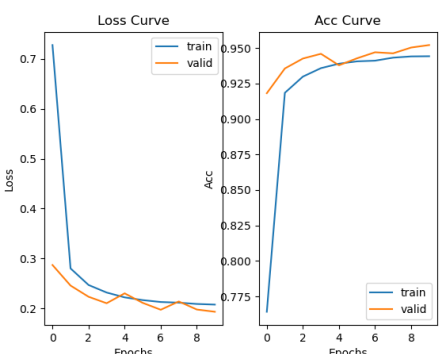
1. Learning Rates

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

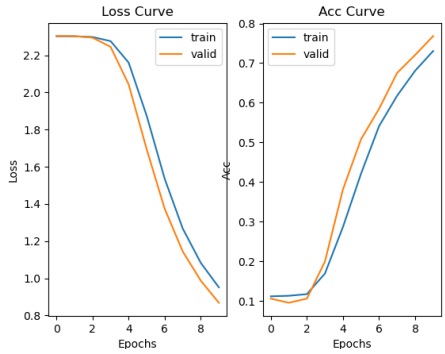
	lr=1	lr=1e-1	lr=1e-2	lr=5e-2
Training Accuracy	0.9442	0.9214	0.7306	0.9088
Test Accuracy	0.9482	0.9267	0.7573	0.9150

1. Learning Curve

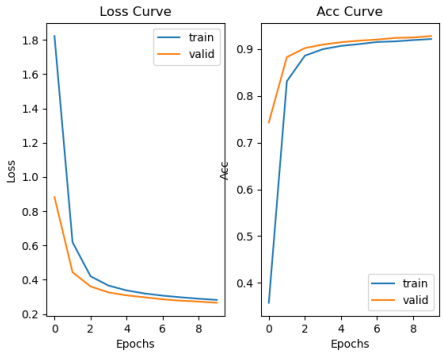
Plot the learning curves [both ac



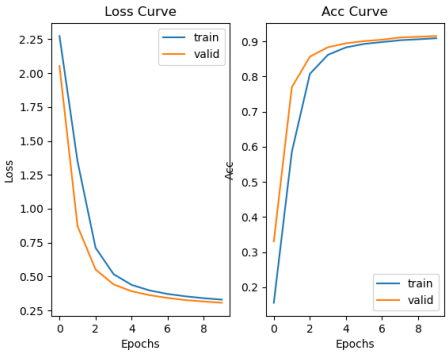
learning_rate = 1



learning_rate = 1e-2



learning_rate = 1e-1



learning_rate = 5e-2

1. Learning Rates

Describe and Explain your findings:

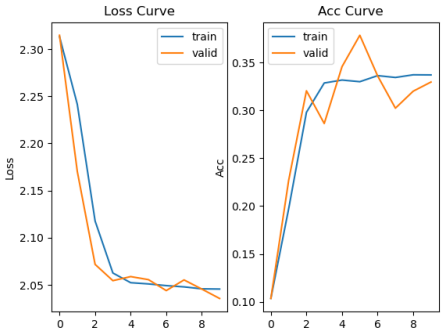
It’s difficult to say that the rate of improvement is positively or negatively related to the learning rates. Even though the algorithm will reach some optima, it’s not guaranteed that the optima will be the global optima. Therefore, there can unfortunately be some values for learning rates that can lead to non-global optima and get stuck there. lr=1e-2 is such a case in our experiment.

2. Regularization

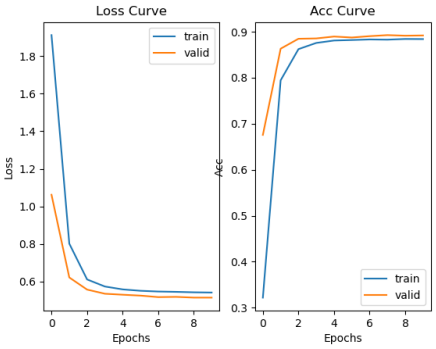
Tune the regularization coefficient of the model with all other default hyper-parameters fixed. Fill in the table below:

	alpha=1e-1	alpha=1e-2	alpha=1e-3	alpha=1e-4	alpha=1e-0
Training Accuracy	0.3371	0.8841	0.9214	0.9294	0.1044
Validation Accuracy	0.3297	0.8920	0.9277	0.9338	0.1060
Test Accuracy	0.3756	0.8920	0.9267	0.9334	0.1028

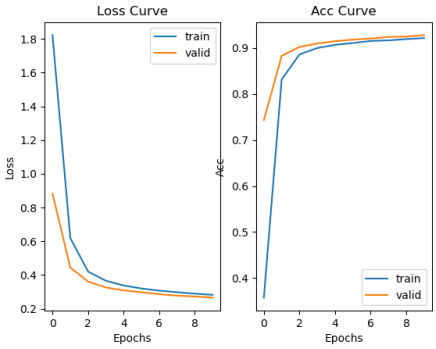
2. Regularization



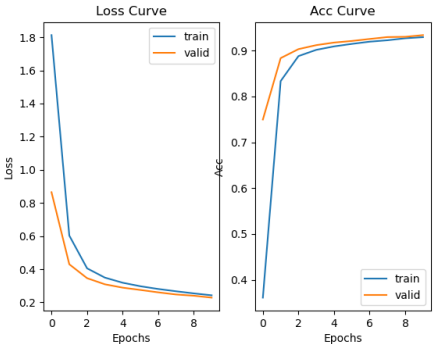
Alpha = 1e-1



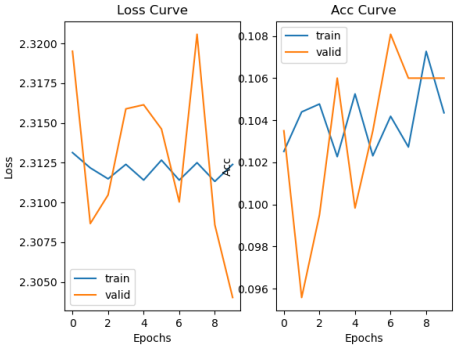
Alpha = 1e-2



Alpha = 1e-3



Alpha = 1e-4



Alpha = 1e-0

2. Regularization

Describe and Explain your findings:

Based on observation, the bigger the regularization constant was, the worse the accuracy got.

Regularizations are methods to reduce the error by appropriately fitting parameters and avoid overfitting. As we observed in the data and the plots, when we increased the regularization constant, the accuracies fluctuated a lot, giving too much flexibility to the model.

3. Hyper-parameter Tuning

alpha	lr	Training accuracy	Validation Accuracy	Test Accuracy
1E-04	1	0.9788	0.9712	0.9696
1E-04	2	0.9691	0.9664	0.9666
1E-04	1.5	0.9752	0.9669	0.9679
1E-04	0.5	0.9721	0.9677	0.9668
1E-05	1	0.9833	0.9717	0.9713
5E-05	1	0.9813	0.9712	0.9707
1E-06	**1**	**0.9838**	**0.9712**	**0.9713**
1E-07	1	0.9838	0.9712	0.9713

meters for better accuracy. Create a table below and
and accuracy into the table:

Briefly explain why your choice works:

From previous observations, we saw that alpha=1e-4 and lr=1 work well, while larger values for alpha and there were higher chances that smaller values of lr would not work as well. Thus, I tried the values for alpha and lr around alpha=1e-4 and lr=1. The accuracies more or less had a plateau at around alpha=1e-6 and lr=1.