CS4803-7643: Deep Learning
Fall 2020
Problem Set 1

Instructor: Dhruv Batra
TAs: Sameer Dharur, Joanne Truong, Yihao Chen, Michael Piseno,
Hrishikesh Kale, Tianyu Zhan, Pravhav Chawla, Guillermo Nicolas Grande.

Discussions: https://piazza.com/gatech/fall2020/cs48037643

Due: Wednesday, September 9, 11:59pm

Instructions

1. We will be using Gradescope to collect your assignments. Please read the following instructions for submitting to Gradescope carefully!

   - For the **HW1** component on Gradescope, you could upload one single PDF containing the answers to all the theory questions and the completed Jupyter notebooks for the coding problems. **However, the solution to each problem or subproblem must be on a separate page. When submitting to Gradescope, please make sure to mark the page(s) corresponding to each problem/sub-problem.** Likewise, the pages of the Jupyter notebooks must also be marked to their corresponding subproblems.
   - For the **HW1 Code** component on Gradescope, please use the `collect_submission.sh` script provided and upload the resulting **hw1_code.zip** here. Please make sure you have saved the most recent version of your Jupyter notebook before running this script.
   - Note: This is a large class and Gradescope's assignment segmentation features are essential. Failure to follow these instructions may result in parts of your assignment not being graded. We will not entertain regrading requests for failure to follow instructions.

2. LaTeX'd solutions are strongly encouraged (solution template available at cc.gatech.edu/classes/AY2021/cs7643_fall/assets/sol1.tex), but scanned handwritten copies are acceptable. Hard copies are **not** accepted.

3. We generally encourage you to collaborate with other students.

   You may talk to a friend, discuss the questions and potential directions for solving them. However, you need to write your own solutions and code separately, and *not* as a group activity. Please list the students you collaborated with.

1 Optimization

1. **[2 points]** In homework 0, we derived the gradient of the log-sum-exp function. Now we will consider a similar function - the softmax function $\mathbf{s}(\mathbf{z})$, which takes a vector input $\mathbf{z}$ and outputs a vector whose ith entry s_i is

$$s_i = \frac{e^{z_i}}{\sum_k e^{z_k}} \tag{1}$$

   The input vector $\mathbf{z}$ to $\mathbf{s}(\cdot)$ is sometimes called the "logits", which just means the unscaled output of previous layers. Derive the gradient of $\mathbf{s}$ with respect to the logits, *i.e.* derive $\frac{\partial \mathbf{s}}{\partial \mathbf{z}}$. Consider re-using your work from HW0.

2. **[2 points]** Consider a (not necessarily convex) differentiable function $g : \mathbb{R}^n \to \mathbb{R}$. g has a local minimum at some $\mathbf{w}^t$ if there exists some $\gamma > 0$ such that for all $\mathbf{w} \in \mathbb{R}^n$, $\left\|\mathbf{w}^t - \mathbf{w}\right\|_2 < \gamma \Rightarrow g(\mathbf{w}^t) \leq g(\mathbf{w})$.

   Prove that if g has a local minimum at some $\mathbf{w}^t$ then the gradient at $\mathbf{w}^t = 0$, and that the converse is not necessarily true.

3. **[3 points]** Prove that if a differentiable function $g : \mathbb{R}^n \to \mathbb{R}$ is convex and the gradient at some $\mathbf{w}^*$ is 0, then $\mathbf{w}^*$ is the global minimum of g.

4. **[2 points]** Consider an objective function comprised of $N = 2$ terms:

$$f(w) = \frac{1}{2}(w-2)^2 + \frac{1}{2}(w+1)^2 \tag{2}$$

   Now consider using SGD (with a batch-size $B = 1$) to minimize this objective. Specifically, in each iteration, we will pick one of the two terms (uniformly at random), and take a step in the direction of the negative gradient, with a constant step-size of η. You can assume η is small enough that every update does result in improvement (aka descent) on the sampled term.

   Is SGD guaranteed to decrease the overall loss function in every iteration? If yes, provide a proof. If no, provide a counter-example.

5. **[3 points: Extra credit for both 4803 and 7643]** Recall that a $d-1$-dimensional simplex is defined as:

$$\Delta_{d-1} = \{\mathbf{p} \in \mathbb{R}^d \mid \mathbf{1}^T\mathbf{p} = 1 \text{ and } \mathbf{p} \geq 0\} \tag{3}$$

   In this question, you will develop an interpretation of softmax as a projection operator – that it projects an arbitrary point $\mathbf{x} \in \mathbb{R}^d$ onto the interior of the $d-1$ simplex. Specifically, let $\mathbf{s}(\mathbf{x})$ denote the softmax function (as defined above). Now prove that,

$$\mathbf{s}(\mathbf{x}) = \underset{\mathbf{y}\in\mathbb{R}^d}{\operatorname{argmin}} \quad -\mathbf{x}^T\mathbf{y} - H(\mathbf{y}) \tag{4}$$

$$s.t. \quad \mathbf{1}^T\mathbf{y} = 1, \quad 0 \leq \mathbf{y} \leq 1 \tag{5}$$

   where H is the entropy function:

$$H(\mathbf{y}) = -\sum_i \mathbf{y}_i \log(\mathbf{y}_i) \tag{6}$$

Now, what does this formal interpretation tell you about the softmax layer in a neural network? Hint: Look at the KKT conditions.

2 Directed Acyclic Graphs (DAG)

One important property for a feed-forward network, as discussed in the lectures, is that it must be a directed acyclic graph (DAG). Recall that a *DAG is a directed graph that contains no directed cycles*. We will study some of its properties in this question.

Let's define a graph $G = (V, E)$ in which V is the set of all nodes as $\{v_1, v_2, ..., v_i, ...v_n\}$ and E is the set of edges $E = \{e_{i,j} = (v_i, v_j) \mid v_i, v_j \in V\}$.

A *topological order of a directed graph* $G = (V, E)$ is an ordering of its nodes as $\{v_1, v_2, ..., v_i, ...v_n\}$ so that for every edge (v_i, v_j) we have $i < j$.

There are several lemmas can be inferred from the definition of DAG. One lemma is: if G is a DAG, then G has a node with no incoming edges.

Given the above lemma, prove the following two lemmas:

6. **[3 points]** If the graph G is a DAG, then G has a topological ordering.
7. **[3 points]** If the graph G has a topological order, then G is a DAG.

3 Paper Review [Extra credit for 4803, regular credit for 7643]

The first of our paper reviews for this class comes from a NeurIPS 2019 paper on the topic **'Weight Agnostic Neural Networks'** by **Adam Gaier** and **David Ha** from Google Brain.

The paper presents an interesting proposition that, through a series of experiments, re-examines some fundamental notions about neural networks - in particular, the comparative importance of architectures and weights in a network's predictive performance.

The paper can be viewed here. There's also a helpful interactive webpage with intuitive visualizations to help you understand its key concepts better.

The evaluation rubric for this section is as follows :

8. **[2 points]** Briefly summarize the key contributions, strengths and weaknesses of this paper.
9. **[2 points]** What is your personal takeaway from this paper? This could be expressed either in terms of relating the approaches adopted in this paper to your traditional understanding of learning parameterized models, or potential future directions of research in the area which the authors haven't addressed, or anything else that struck you as being noteworthy.

Guidelines: Please restrict your reviews to no more than 350 words (total length for answers to both the above questions).

4 Implement and train a network on CIFAR-10

Setup Instructions: Before attempting this question, look at setup instructions at here.

10. **[Up to 20 points]** Now, we will learn how to implement a softmax classifier and vanilla neural networks (or Multi-Layer Perceptrons). You will begin by writing the forward and backward passes for different types of layers, and then go on to train a shallow neural network on the CIFAR-10 dataset in Python. Next you will learn to use PyTorch, a popular open-source deep learning framework, and use it to replicate the experiments from before.

    Refer to the instructions linked here to get started.