Homework #2

CSE 493S/599S: Advanced Machine Learning
Prof. Sewoong Oh
Due: Tuesday, June 10^{th} at 11:59pm

Instructions

- Please submit this homework to Gradescope.
- Submit the code, as well as a report containing the plots and discussion. Also include a README, along with instructions to run your code.
- This homework is to be done in teams of upto 3.
- List every person with whom you discussed any problem in any depth, and every reference (outside of our course slides, lectures, and textbook) that you used.
- You may spend an arbitrary amount of time discussing and working out a solution with your listed collaborators, but **do not take notes, photos, or other artifacts of your collaboration**. Erase the board you were working on, and once you're alone, write up your answers yourself.
- The homework problems have been carefully chosen for their pedagogical value and hence might be similar or identical to those given out in similar courses at UW or other schools. Using any pre-existing solutions from these sources, from the Web or other textbooks constitutes a violation of the academic integrity expected of you and is strictly prohibited.

Overview

The goal of this homework is to give you hands-on experience developing modern machine learning models. In short, your task will be to train transformer models from scratch in a simplified setting. Please submit a short report containing the plots needed and description of your work.

Background and Pre-requisites

We will use a simple implementation of a transformer model for this assignment, available here. The implementation is adapted from Andrej Karpathy's nano-

GPT project. Most of this assignment is designed to be done with low compute requirements.

1 Part 1 – Infrastructure Setup

You will write two files to implement this part. These will be used later.

1.1 `train.py`

This file is the main trainer class. You will instantiate your transformer model here, read your data, loop through batches of your data, run a forward pass on your model, carefully compute the loss on the correct tokens, call the optimizer and log your metrics. At the end of training (or even periodically), you will save the weights of your model.

Try to make the script general to the choice of hyper-parameters, data and model. Also ensure that you have some way to specify the config with which the experiment runs, and log the config as well. Finally, it might be good practice to seed all randomness for reproducibility.

Hint The given model definition does not incorporate attention masking for pad tokens. You can implement that to train on batches of data with varying lengths.

1.2 `inference.py`

This script (or notebook) takes a model's path and config, loads its weights, and generates text. This is not meant to do generation at scale, but is more of a debugging tool for you.

Part 1.5 – Sanity Checks

In this part, you will write a simple test to ensure that your infrastructure is correct. The test is to see if your model can memorize and regurgitate a string.

More concretely, define your data to contain a single string "I love machine learning", and train a single layer transformer on this. The train loss should go to 0, and when you sample from the model you should get this string back exactly.

For another sanity check, mask the loss on the first 3 tokens.

Deliverables The log of these experiments, the model checkpoint and your code for training and inference. Describe modifications made to the original codebase, and any challenges faced.

Grading Code implementation and sanity check (20 points).

2 Part 2 – Algorithmic Tasks

In this section, you will reproduce experiments for grokking[1] in transformers. For this, you will train a small transformer model for 3 tasks - modular addition, modular subtraction and modular division.

2.1 Data Generation

You need to generate data of the form "a+b=c mod p", "a-b=c mod p" and "a/b=c mod p", where a, b are natural numbers. From the original paper, $0 \leq a, b \leq p$, with $p = 97, 113$. Ensure that you have proper train, test and val splits.

Deliverables Generated train/test splits. Include description of the process and the number of datapoints used for each split.

Grading Data generation (10 points).

2.2 Warmup - Addition and Subtraction Experiments

Now, you will train a 1- and 2-layer models and report train and test accuracy and loss. For these experiments, set the dimension of the model to be 128, the dimension of the feedforward layer to be 512, and the number of attention heads to be 4. Train for up to 10^5 training steps, using the Adam optimizer. Remember to only compute the loss on the output of the equation. Report performance metrics for 3 random restarts, as well as 2 values of $p = 97, 113$.

Note that you might need to tune the hyperparameters. For this, it is good practice to split your train set into train and validation, and only tune your hyperparameters on the val set. You can also look at the hyperparameters in the paper. You would also need to implement some sort of tokenization, and the easiest way to tokenize is at the character level.

Deliverables Training curves and test curves, final model checkpoint for one of the seeds. Report the final loss and train/test accuracy across 3 random seeds.

Grading Experiment results (20 points).

2.3 Grokking

Now you will try to reproduce what is essentially Fig 1 of the paper. Train on the modulo division task for $p = 97$, and see if you can get the model to grok. Refer to the paper for more details if you are stuck.

Deliverables Plot of training curves, final model ckpt for one seed.

[1] https://arxiv.org/abs/2201.02177

Grading Grokking results (20 points).

2.4 Ablations/Analysis

What factors could make grokking on the division task happen faster and more reliably? Design an ablation study where you change some experimental configuration (hyperparameters, architecture, tokenizer, optimizer, dataset size, regularization etc.), and measure the number of steps for the test error to go to 0 after the train error has vanished on the modulo division task. The change should be clearly motivated and described. Only one ablation is required, but more are welcome. You can also refer to literature around grokking for more inspiration.

Deliverables A short report of what factor you changed, and how it influenced training curves (with plots).

Grading Ablation experiment(s) (10 points).