Assignment 7

CMSC 478 – Machine Learning

April 25, 2024

Item	Summary
Assigned	Apr 25, 2024
Due	May 10, 11:59 PM
Topic	Neural Network
Points	110

You are to complete this assignment on your own: that is, the code and writeup you submit must be entirely your own. However, you may discuss the assignment at a high level with other students or on the discussion board. Note at the top of your assignment who you discussed this with or what resources you used (beyond course staff, any course materials, or public Campuswire discussions).
Language and External Resources Your code must be compiled, and you will use Python. Failure to do so, or the use of any external resources, including generative AI, and other people's work, without prior written permission from the instructor, will be considered an academic integrity violation and result, at a minimum, in a 0 on this assignment.

Problem

In this exercise, you'll experiment with CNNs for image classification using the Torch/Tensorflow framework in Python. **It is important that you start on this early because you'll be installing code and data, and if you run into problems you may not be able to get them resolved at the last minute.**

Option 1: PyTorch
Read through the PyTorch 60 Minute Blitz here:
http://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html
– at the very least sections "Neural Networks" and "Training a Classifier".

At the end of the "Training a Classifier" section, you can download `cifar10_tutorial.py`. Do that. To run it you'll need to install the torch and torchvision packages for Python. The easiest way to do that is with pip:

- pip install torch
- pip install torch-vision

Instructions on how to install pip can be found here: https://pip.pypa.io/en/stable/installation/.

Option 2: Tensorflow
For your convenience, I created a similar architecture and training method in Tensorflow – Colab Notebook. You can download it as `cifar10_tutorial.py`.

You can either run the code in PyTorch or Tensorflow. When you run the CIFAR10 tutorial code the first time, it will download the CIFAR10 dataset, train a simple CNN, and report classification accuracy. Your task is to try to improve that accuracy and document the results **(include everything you try regardless of whether it worked or not)**. Things to consider are:

1. Change filter sizes and/or numbers in the convolutional layers
2. Add a convolutional layer
3. Change the size(s) of the fully connected layer(s)
4. Try an activation function other than ReLU
5. Use different learning rates or momentum
6. Train for more epochs
7. Look around on the web for advice on other things to try

We are not restricting the last improvement. You can make multiple improvements if you wish.

What To Turn In

- Turn in the output of the CIFAR10 code from your initial run before making any changes. You should dump your code output in a file and submit that file as 'output0'. [10 points]
- Then write a short document that lists the things you did and the **resulting test accuracy**, with output from the CIFAR10 demo to document that accuracy. Output files should be named as output<index-of-change>, such as 'output1' for changing filter size and/or numbers, and 'output2' for adding conv. layer, etc. For each change, also explain why you think each step **helped or hurt** the accuracy. [7*10 points]
- Please submit the code that yielded your best result. Save the output file as 'output-best'. Your grade for this question will be based on accuracy. Achieving an accuracy score of over 95% will earn you full points. For each 5% decrease in accuracy below this threshold, you will lose 3 points. [30 points]

Notes:

1. Momentum in gradient descent works by adding a fraction of the previous weight update to the current weight update so that the optimization algorithm can build momentum as it descends the loss function.

   To apply gradient descent with momentum, you can update the weights as follows:

   $$v = \rho * v + learning\_rate * gradient$$
   $$weights- = v$$

   Where v is the momentum term, ρ is the momentum hyperparameter (typically set to 0.9), learning_rate is the learning rate, and the gradient is the gradient of the loss function to the weights. It's important to note that momentum is just one of many techniques we can use to improve the convergence of gradient descent.

2. In both PyTorch and Tensorflow, you are using functions/methods. If you hover over them in your compiler, or if you click on them, it will take you to the definition with details for each parameter. The parameters are known to you and have been discussed in class.