CSCI 5525: Machine Learning (Fall 2020)

Homework 3

Due 11/10/2020 11:59 PM CDT

1. **(10 points)** Consider the convolutional neural network architecture given in Figure 1 for classifying MNIST digits. Assume that the input images are reduced to size 10×10 with only 1 channel (represented as a matrix in $\mathbb{R}^{10\times 10}$). In this architecture, the convolutional layer uses a 3×3 filter, $\mathbf{W}_{conv}$, with stride 2 and zero padding. The dimensions of the outputs of each layer are shown below.

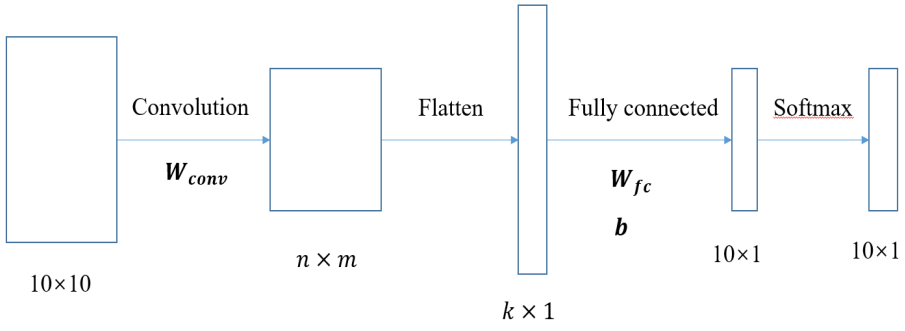

Figure 1: A toy CNN

   (a) **(5 points)** What are the values of n, m, k in the graph?

   (b) **(5 points)** What are the sizes of $\mathbf{W}_{conv}$, $\mathbf{W}_{fc}$, and $\mathbf{b}$?

Programming Problems: The next two problems are programming problems and will focus on implementing neural networks for handwritten digits classification. We will use the MNIST dataset where each sample is an image of a hand-written digit and has a corresponding label indicating the value of the digit written $(0, 1, \ldots, 9)$. This makes it a multi-class classification problem.

You must use Tensorflow 2 to implement your neural networks. The implementation must be for the CPU version only (no GPUs or MPI parallel programming is required for this assignment). Follow the installation instructions at `https://www.tensorflow.org/install` in case you want to use your local machine (we recommend using anaconda); you may also use colab to this assignment. You can load the MNIST dataset directly in Tensorflow with the following code:

```
import tensorflow as tf
mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

2. **(40 points)** Implement a multi-layer fully connected neural network:

   - Input: 1-channel input, size 28x28
   - Fully connected layer 1: input with bias; output - 128 nodes
   - ReLU activation function
   - Fully connected layer 2: input - 128 nodes; output - 10 nodes
   - Softmax activation function
   - Use cross entropy as the loss function
   - Use SGD as optimizer
   - Set mini batch size as 32

   Train using mini batches of the given batch size. Plot the cumulative training loss and accuracy for every epoch. Once training is complete, apply the learned model to the test set and report the testing accuracy.

   Epoch: An epoch is a single pass through all the training data. Typically many epochs will be run when training a neural network before it converges.

   Please submit (a) **summary of methods and results** report and (b) **code**:

   (a) **Summary of methods and results**: Briefly describe the approaches used above, along with relevant equations. Report the cumulative training loss and accuracy for every epoch as plots. Also report the testing accuracy (a single number).

   (b) **Code**: Submit the file `neural_net.py` which contains the function `def neural_net() -> None:`. The function does not have any inputs and does not return anything but must print out to the terminal (stdout) the cumulative training loss and accuracy per epoch as well as the testing accuracy.

3. **(50 points)** Implement a convolutional neural network with the following specifications.

   - Input: 1-channel input, size 28x28
   - Convolution layer: Convolution kernel size is (3, 3) with stride as 1. Input channels - 1; Output channels - 20 nodes
   - ReLU activation function
   - Max-pool: 2x2 max pool
   - Dropout layer with probability $p = 0.50$
   - Flatten input for feed to fully connected layers
   - Fully connected layer 1: flattened input with bias; output - 128 nodes
   - ReLU activation function
   - Dropout layer with probability $p = 0.50$
   - Fully connected layer 2: input - 128 nodes; output - 10 nodes
   - Softmax activation function
   - Use cross entropy as loss function

For this problem, we will be experimenting with a variety of parameters.

First, train using SGD as the optimizer and mini batches of size 32. Plot the cumulative training loss and accuracy for every epoch. Once training is complete, apply the learned model to the test set and report the testing accuracy.

Second, train your network using mini batch sizes of $[32, 64, 96, 128]$ and plot the convergence run time vs mini batch sizes for each of the following optimizers: SGD, Adagrad, and Adam. You should report 3 figures, one for each optimizer where each figure has mini batch size on the x-axis and the convergence run time on the y-axis.

Please submit (a) **summary of methods and results** report and (b) **code**:

(a) **Summary of methods and results**: Briefly describe the approaches used above, along with relevant equations. Report the cumulative training loss and accuracy for every epoch as plots. Also report the testing accuracy (a single number). Also report the convergence run time vs mini batch sizes for each mini batch size and optimizer above (3 plots).

(b) **Code**: Submit the file `cnn.py` which contains the function `def cnn() -> None:`. The function does not have any inputs and does not return anything but must print out to the terminal (stdout) the cumulative training loss and accuracy as well as the testing accuracy (for the first part above). It must also print out the batch size and convergence run time for each mini batch size and optimizer (for the second part above).

Additional instructions: Code can only be written in Python 3.6+; no other programming languages will be accepted. One should be able to execute all programs from the Python command prompt or terminal. Please specify instructions on how to run your program in the README file.

Each function must take the inputs in the order specified in the problem and display the textual output via the terminal and plots/figures should be included in the report.

For each part, you can submit additional files/functions (as needed) which will be used by the main file. In your code, you **cannot** use machine learning libraries such as those available from scikit-learn for learning the models - exception being that you must use Tensoflow 2 for your neural network implementations. You may also use libraries for basic matrix computations and plotting such as numpy, pandas, and matplotlib. Put comments in your code so that one can follow the key parts and steps in your code.

Your code must be runnable on a CSE lab machine (e.g., csel-kh1260-01.cselabs.umn.edu). One option is to SSH into a machine. Learn about SSH at these links: `https://cseit.umn.edu/knowledge-help/learn-about-ssh`, `https://cseit.umn.edu/knowledge-help/choose-ssh-tool`, and `https://cseit.umn.edu/knowledge-help/remote-linux-applications-over-ssh`.

Instructions

Follow the rules strictly. If we cannot run your code, you will not get any credit.

- **Things to submit**

  1. hw3.pdf: The report that contains the solutions to Problems 1, 2, and 3 including the summary of methods and results.
  2. `neural_net.py`: Code for Problem 2.

3. `cnn.py`: Code for Problem 3.
4. README.txt: README file that contains your name, student ID, email, instructions on how to run your code, any assumptions you are making, and any other necessary details.
5. Any other files, except the data, which are necessary for your code.

Homework Policy. (1) You are encouraged to collaborate with your classmates on homework problems, but each person must write up the final solutions individually. You need to list in the README.txt which problems were a collaborative effort and with whom. (2) Regarding online resources, you should **not**:

- Google around for solutions to homework problems,
- Ask for help on online,
- Look up things/post on sites like Quora, StackExchange, etc.