CS 6364-002 Homework 3

September 27, 2022

Requirements:

- Deadline for the first submission: **Sep-28-2022**.
- All assignments **MUST** have your name, student ID, course name/number at the beginning of your documents.
- For each of the questions, please write all the codes in one Jupyter notebook and run the codes to display the results in the notebook before you save the notebook (ipynb file). Pls zip all the notebooks into one file and submitted the zipped file.

If you have any questions, please contact me.

For the following questions, if you need a GPU to run, Google provide a free Jupyter notebook environment that requires no setup and runs entirely in the cloud. Here is the link: https://colab.research.google.com/notebooks/welcome.ipynb?hl=enscrollTo=5fCEDCU_qrC0

Q1 (Regression) Implement a neural network to train a regression model for the Boston housing data set

https://towardsdatascience.com/linear-regression-on-boston-housing-dataset-f409b7e4a155

Split the dataset to a training set (70% samples) and a testing set (30% samples). Report the root mean squared errors (RMSE) on the testing sets.

- You have to use PyTorch deep learning library.
- Two hidden layers: the first hidden layer must contain 16 units using ReLU activation function; the second layer must contain 32 units using tanh activation function.

Q2 (Classification): Implement a neural network to train a classification model for the Titantic dataset:

https://blog.goodaudience.com/machine-learning-using-logistic-regression-in-python-with-code-ab3c7f5f3bed.

Split the dataset to a training set (80% samples) and a testing set (20% samples). Report the overall classification accuracies on the training and testing sets and report the precision, recall, and F-measure scores for each of the two classes on the testing sets.

- You have to use PyTorch deep learning library.
- Two hidden layers: the first hidden layer must contain 5 units using ReLU activation function; the second layer must contain 3 units using tanh activation function.

Q3 First, implement a convolutional neural network (CNN) that contains 1 convolutional layer, 1 pooling layer, and 1 fully connected layer. The input of the CNN is a gray scale image. The convolutional layer should have two filters/kernels with size of 3 by 3 to for horizontal and vertical edge detection and have the ReLU activation function. For the pooling layer, you can specify the hyperparemters (e.g., filter size, stride size, aggregation function (such as max, min, avg)) yourself. For the fully connected layer, you can specify the number of neurons and the activation function yourself.

Let $\mathbf{a}^{[0]}$ denote the input image and $\mathbf{W}^{[1]}$ and $\mathbf{b}^{[1]}$ denote the weight matrix and the bias vector of the convolutional layer, respectively. Let $\mathbf{a}^{[1]}$ denote the output of the convoluational layer. Then we have the following mathematical representation:

$$\mathbf{a}^{[1]} = g(\mathbf{z}^{[1]}), \mathbf{z}^{[1]} = \mathbf{W}^{[1]} \star \mathbf{a}^{[0]} + \mathbf{b}^{[1]} \tag{1}$$

where $g(\cdot)$ refers to the ReLU activation function. Note that, as $\mathbf{W}^{[1]}$ has one horizontal detector and one vertical detector, $\mathbf{W}^{[1]}$ is known and does not need to be estimated based on a training set. You may set $\mathbf{b}^{[1]}$ to zero or any other values. You may set the values of the weight matrix and the bias vector randomly for the fully connected layer.

Second, apply the CNN to an input image (see the attachment) to detect all horizontal and vertical edges in this image. You need to first convert the image into gray scale using OpenCV, before you apply the CNN.

Third, plot the output of the convolutional layer before the ReLU activation function ($\mathbf{z}^{[1]}$), the output of the convolutional layer after the ReLU activation function ($\mathbf{a}^{[1]}$), and the output of the pooling layer. As each of these three outputs is a volume with two channels, you may plot each channel separately. In total, you will generate six figures in total.

What need to submit:

- Python programming code
- Outpus six figures in total.

Q4 Build an image classification model using Convolutional Neural Networks (CNN) in PyTorch. The data set (i.e. Fashion-MNIST) can be found here:

https://github.com/zalandoresearch/fashion-mnist/tree/master/data/fashion.

Information about this dataset can be found here:

https://github.com/zalandoresearch/fashion-mnist.

It consists of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes.

By building the classification model, I suggest you divide it into several subtasks:

1. Loading the data set.
   - Please explore a few samples and visualize these images.
2. Creating a validation set and preprocessing the images.
   - Split the training data set to a training set (90% samples) and a validation set (10% samples)
   - convert the images and the targets into torch format for both training and validation data sets.
3. Implementing CNNs using PyTorch.
   - Define the architecture with just 2 convolutional layers to extract features from the images and then use a fully connected dense layer to classify those features into their respective categories (i.e. two Conv2d layers and a linear layer).

```
Net(
  (cnn_layers): Sequential(
    (0): Conv2d(1, 4, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(4, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace)
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (4): Conv2d(4, 4, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (5): BatchNorm2d(4, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (6): ReLU(inplace)
    (7): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (linear_layers): Sequential(
    (0): Linear(in_features=196, out_features=10, bias=True)
  )
)
```

   - Train the model for 25 epochs and show the validation losses by printing in console. You are expected to see that the validation loss is decreasing as epoch increases.
   - Visualize the training and validation losses by plotting them.
   - Show the accuracy of the model on the training and validation set.
4. Generating predictions for the test set.

- Load the test images.
- Do the pre-processing steps on these images similar to what you did for the training images.
- Generate predictions for the test set.

What to submit:

- Python programming code
- Five image visualization when loading the data set.
- Plotting visualization of the training and validation losses.
- Report the accuracy of your model on training and validation set.
- Your predictions for the test images.

Q5 (Bonus Question): You will get an additional half point (0.5) if you can answer this bonus question correctly. That means, if you answer Q1-Q4 correctly, you get a full point (1.0) for this HW assignment. If you can answer Q1-Q5 correctly, you will get 1.5 points.

Following the step by step instructions below to build a LeNet-5 CNN architecture in **Pytorch** using Fashion-MNIST data set provided in Q4. [Note that, the example codes in the following link were written in **TensorFlow**. In this question, you need to use **Pytorch**].

https://medium.datadriveninvestor.com/lenet-5-a-classic-cnn-architecture-c87d0b03560d

What to submit:

- Python programming code
- Visualization of plotting the training accuracy and loss after each epoch
- Your predictions for the test images.