1 Introduction

In this assignment, you will implement a variety of classifiers and deep learning classifiers. A basic code framework is provided for you as a starting point. All of your code will be submitted, as well as write up. Please do not include code snippets in your write up, submit these separately. You must use the PyTorch framework. Note that you will not need to install CUDA for this assignment; you can do everything on the CPU.

The first step is to install PyTorch framework and read through the PyTorch tutorial (the 60 minute blitz), which is available from the installation instructions web page linked below.

We will update this assignment with more detailed instructions about what to submit and how to submit it, but please do not delay getting started.

1.1 Resources

- PyTorch installation: https://pytorch.org/get-started/locally/
- PyTorch Tutorial: https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html
- Our starter code: https://users.cs.duke.edu/~ryzhang/compsci570/code.py
- Fashion-MNIST: https://github.com/zalandoresearch/fashion-mnist
- MNIST: http://yann.lecun.com/exdb/mnist/
- Train only part of the network: https://pytorch.org/docs/master/notes/autograd.html

2 Implementing Logistic Regression

You will implement a logistic regression classifier to classify the MNIST dataset (example data-loader is provided in Line 125). Please see the class LogisticRegression in code.py.
MNIST is a classic machine learning benchmark described in the link above. It is considered too easy of a benchmark by today’s standards, but it’s still useful pedagogically because it’s a practical problem that can run quickly on a laptop.

Your classifier will go directly from pixels to log odds, sometimes called, “logits.” Thus, there will be 784 inputs and 10 outputs.

1. Train your model with MNIST. Produce a plot showing the training accuracy and testing accuracy at each epoch for the first 20 epochs. (An epoch is a complete pass over the entire training set.

2. Give two examples of incorrectly classified digits. Include the image of the misclassified digit, the predicted class and the actual class in your write up.

3. Apply a form of regularization (L1 or L2) and recreate the plot from above. What changes do you observe? (See the bullet on regularization in Section 1.1.)

3 Go deeper with Multi-Layer Perceptron

3.1 Basic MLP

Train a two-layer Multi-Layer Perceptron (MLP) with no regularization to classify MNIST. (See the class MLP in code.py.) The provided two-layer MLP has the following architecture:

- Input layer : 784 nodes (MNIST images size)
- First hidden layer : 400 nodes
- Second hidden layer : 400 nodes
- Output layer : 10 nodes (number of class for MNIST)

1. Produce a plot showing the training accuracy and testing accuracy at each epoch for the first 20 epochs.

3.2 Initialization

To begin training your neural network, you must specify an initial value for the weights. A well chosen initialization method will help learning. Review the initialization bullet item in Section 1.1 for guidance on completing this section.

1. Report final training accuracy and testing accuracy for following different initialization methods. All the initialization functions have built-in functions within the PyTorch framework. Use the network structure from 3.1.

   (a) Zero initialization
   (b) Initialize with uniform [0,1]
   (c) Initialize with standard normal
   (d) Initialize with truncated normal (xavier_normal)

2. Based on your experimental results, justify which initializer is best for this problem.
3.3 Network Structure

For simple problems, using the appropriate structure of neural network is computationally efficient. For more complicated problems, the appropriate structure of the network can be essential to solving the problem.

1. Try training the MLP with different hidden-layer size (one hidden layer with 100 units, one hidden layer with 400 units, and two hidden layers with 100-100 hidden units). Report the final test accuracy and training accuracy.

2. Justify which MLP structure is best for the MNIST dataset, and give your detailed reasons based on your results.

3.4 Optimization

Training your neural network requires specifying an optimization method. A well chosen optimizer will help learning.

1. Report final training accuracy and testing accuracy with following optimization methods. Produce a plot (or plots) showing the training accuracy and testing accuracy at each epoch for the first 20 epochs, for each optimization method. All these optimization methods have built-in functions in the PyTorch framework and descriptions in the PyTorch documentation.
   (a) Stochastic Gradient Descent (SGD)
   (b) RMSProp
   (c) Adagrad
   (d) Adam

2. Based on your experimental results, justify which optimizer best for this problem.

4 Powerful feature extractors

Train a two-layer Convolutional Neural Network (CNN) with no regularization to classify MNIST and Fashion-MNIST. The provided two-layer CNN (See the class CNN in code.py) has the following architecture:

- Input layer : 784 nodes (images size)
- First convolutional layer : 10 channels, with filter size 5*5
- Second convolutional layer : 20 channels, with filter size 5*5
- Output layer : 10 nodes (number of classes)

1. To learn how the CNN works,
   (a) Please calculate the number of parameters in the first convolutional layer and show clearly have you have performed this calculation.
(b) Please calculate the number of weights in the final linear layer show clearly have you have performed this calculation.

2. To demonstrate that the CNN is a feature extractor, you will train a two-layer CNN on Fashion-MNIST. See the datasets tutorial in resources for help.

   (a) What regularization did you use, if any? What training accuracy and test accuracy can you achieve? Save the model for use in (b) Iin PyTorch, use:
   ```python
torch.save(model.state_dict(), 'model.pt')
```

   (b) We will now use the learned feature extractor in Fashion-MNIST to classify MNIST. Load the model (use `model.load_state_dict(torch.load('model.pt'))`), freeze the model parameters (in PyTorch, use `requires_grad = False`) and train only the last layer. (See the last bullet of Section 1.1 above.) What accuracy do you achieve?

   (c) How does the result from 2b compare to training the model entirely on MNIST?

   (d) Compare testing/training accuracy curves on MNIST using the Fashion-MNIST parameters and training the model from scratch. What does it tell you about the features extracted by the Fashion-MNIST classifier?

3. To demonstrate that the CNN is a feature extractor, you will train a two-layer CNN on MNIST.

   (a) What regularization did you use, if any? What training accuracy and test accuracy can you achieve? Save the model for use in (b).

   (b) We will now use this model to classify Fashion-MNIST. Load the model, freeze the model parameters (in PyTorch, use `requires_grad = False`) and train only the last layer. What accuracy do you achieve?

   (c) How does the result from 3b compare to training the model entirely on Fashion-MNIST?

   (d) Compare testing/training accuracy curves on Fashion-MNIST using the MNIST parameters and training the model from scratch. What does it tell you about the features extracted by the MNIST classifier?