Homework 7
CIS 192

Instructor: Jorge Mendez

Due: April 10th 2020

1 Handwritten digit classification

One of the earliest problems solved in image classification was that of recognizing handwritten digits. The MNIST data set is a corpus of 60,000 training images and 10,000 test images, each with a single, centered handwritten digit and accompanied by an annotation of the correct value of the digit. Each image is of size 28×28 and is in grayscale (i.e., it has a single color channel). It is considered a sort of "Hello World" of deep learning, so in this homework we will ask you to construct a deep learning architecture, train it, and show that it is capable of achieving high generalization accuracy. Specifically, you will take the following steps:

1. Install `pytorch` and `torchvision`. My recommendation is to follow the command from `https://pytorch.org/get-started/locally/` for installing with `conda` on your system. You won't need to use CUDA for this assignment, so select the option "None" for CUDA. Be sure to install in a clean conda environment.
2. Load the data set into training and test sets, leveraging `torchvision.dataset.MNIST` and `torch.data.utils.DataLoader`. Use a batch size of 10 and `shuffle=True`. In order to properly use the data set, you must pass in a `torchvision.transforms`. Use a `Compose` transform that first transforms each image `ToTensor` and then flattens it to one long vector with a custom `Lambda` transform. Hint: your lambda function should take a 2D $(h \times w)$ tensor as input and return as output a 1D (hw) tensor.
3. Create a neural network with two hidden fully connected (`nn.Linear`) layers of 64 neurons, each followed by ReLU activation. The input to the first layer should be the flattened image pixels. You can use `nn.Sequential` to create your network.
4. Train the network using Adam optimizer for 2 epochs. Every `f` iterations (i.e., mini-batches), you will compute the accuracy on both the training set and the test set. The accuracy is the number of times the predicted label matches the true label, divided by the total number of points.
   Hint: For computing accuracies, you might want to use a `torch.no_grad` block, to avoid computing gradients that will be unused for training. This will make your code run somewhat faster.
5. Create a plot as in Figure 1 where you simultaneously show training and test accuracy as the number of iterations grow.

Requirements:

- You are free to use any NumPy and Pytorch functions.
- Efficiency matters! Your code will be evaluated for speed.
- Style also matters. You may leverage a style checker (e.g., `https://pypi.org/project/pep8/`) to ensure your code has proper style.

Additional details on the functions required are provided with the skeleton code, available at `https://seas.upenn.edu/~cis192/jorge/hw7/hw7.py`. Feel free to create additional functions as you see fit to modularize your code.

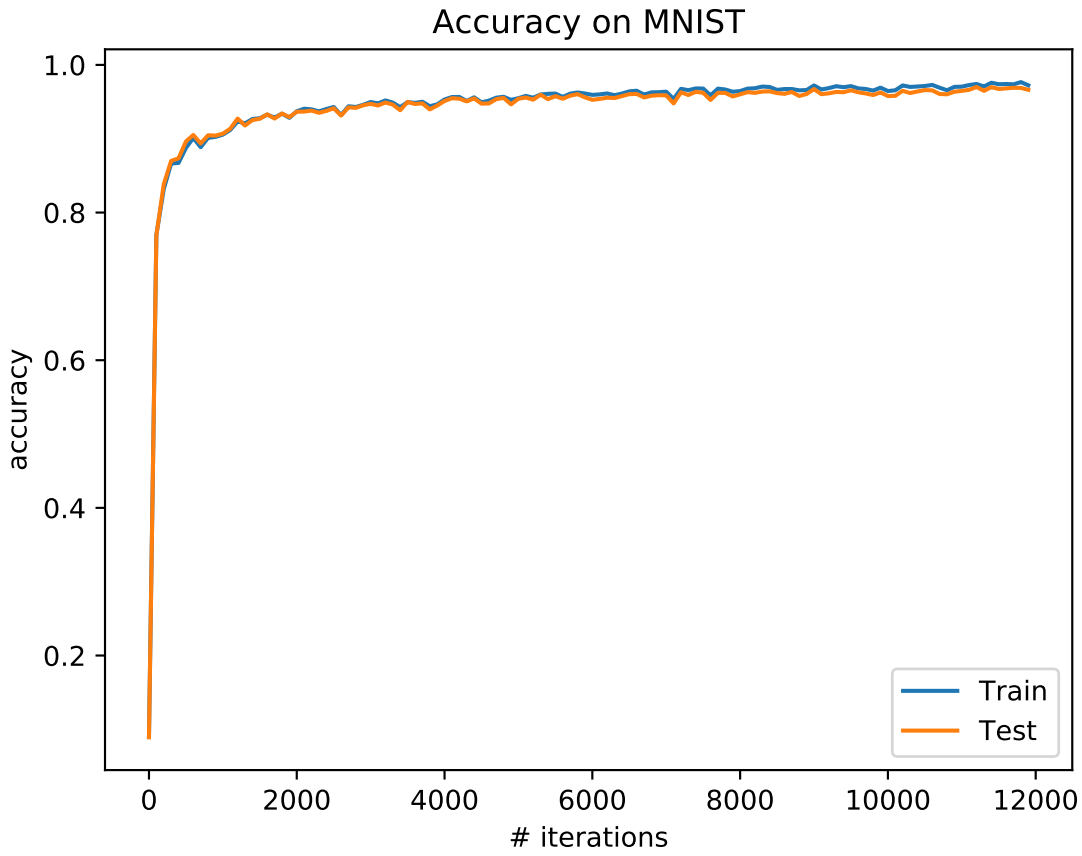

Figure 1: Learning curves on the MNIST data set for the loss (top) and accuracy (bottom).

2 Testing your code

The easiest way to test your code is to run it on `eniac`. For this, you will write a `main()` function as explained in the skeleton file, copy your file to `eniac`, and run it on the terminal by executing `python3 hw6.py` from the directory where the file is stored. Note that `eniac` contains version 3.6.5 of Python, and not 3.7. If you wish to make use of any functionality introduced as of version 3.7, feel free to install Python on your own machine. There are multiple resources online describing how to do this.

3 Submission instructions

Submit a single file named `hw7.py` to Canvas. This file must contain implementations for all functions given in the skeleton file provided to you with the same name. Make sure to add your name and PennKey, as well as the number of hours you spent working on this assignment, at the top of the file where indicated.