Computer Vision
CSCI-GA.2272-001
Assignment 2

October 14, 2016

Introduction

This assignment is an introduction to using Torch for training simple neural net models. Two different datasets will be used: (i) MNIST digits [hand-written digits] and (ii) CIFAR-10 [32x32 resolution color images of 10 object classes].

Requirements

You should perform this assignment in Torch, following the direction below to modify the template code that is provided in the `assignment2.zip` file on the course webpage.

This assignment is due on **Thursday November 3rd** at 11.59pm. Please note that the late policy is as follows: (a) assignments that are late by less than 24hrs will suffer a 10% reduction; (b) those between 24 and 72 hrs late will suffer a 25% reduction and (c) those more 72hrs late will suffer a 50% reduction. You are strongly encouraged to start the assignment early and don't be afraid to ask for help from either the TA or myself.

You are allowed to collaborate with other students in terms discussing ideas and possible solutions. However you code up the solution yourself, i.e. you must write your own code. Copying your friends code and just changing all the names of the variables is not allowed! You are not allowed

to use solutions from similar assignments in courses from other institutions, or those found elsewhere on the web.

Your solutions should be emailed to me at (`fergus@cs.nyu.edu`) in a single zip file with the filename: `lastname_firstname_a2.zip`. This zip file should contain: (i) a PDF file `lastname_firstname_a2.pdf` with your report, showing output images for each part of the assignment and explanatory text, where appropriate.

1 Installing Torch

Torch is an open-source machine learning platform that is available for Mac OSX and Ubuntu Linux. It uses Lua as a scripting language. Installation instructions can be found at `http://torch.ch/docs/getting-started.html`. The main package can be installed by following the first three commands. Please also install the `image` package using the command `luarocks install image`.

Once you have installed Torch, check that you can start the interpreter by typing `qlua` at the shell prompt. This is a version of Torch that has a graphical display enableYou then might want to spend a few moments familiarizing yourself with Lua and Torch tensors, the fundamental data object used by the library. Some good introductory videos can be found at `https://github.com/Atcold/torch-Video-Tutorials`. Please check that your display works by typing `require('image'); image.display(image.lena())`. An image of a woman should appear in a separate window.

This assignment will use a couple of extra Torch packages, namely `Torchnet` and `debugger`. The former is a framework that simplifies the setting up, running and evaluation of machine learning algorithms. The latter is a simple command-line debugger (in the vein of `gdb`) that you may find useful when debugging your code. Source and installation directions can be found at `https://github.com/torchnet/torchnet` and `https://github.com/slembcke/debugger.lua` respectively.

2 Warm Up [10%]

It is always good practice to visually inspect your data before trying to train a model, since it lets you check for problems and get a feel for the task at

hand.

Download the `assign2.zip` file from the course webpage and extract it somewhere convenient. Then change the `base_data_path` location at the top of `assign2.lua` to point to the appropriate directory.

MNIST is a dataset of 70,000 grayscale hand-written digits (0 through 9). 60,000 of these are training images. 10,000 are a held out test set. CIFAR-10 is a dataset of 60,000 color images (32 by 32 resolution) across 10 classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck). The train/test split is 50k/10k.

After starting `qlua`, load up and display (using the `image`) package 100 examples of (a) the MNIST training data and (b) CIFAR-10 training data. The following commands may be useful: `torch.load`, `torch.reshape`, `torch.permute`, `image.display`.

3 Training a Single Layer Network on MNIST [20%]

Start by running on MNIST using the command `qlua assign2.lua -mnist`. This will initialize a single layer model with a softmax output and train it on the 50,000 MNIST training images for 10 epochs (passes through the training data). The default values for the learning rate, batch size and number of epochs are given at the top of the `assign2.lua` file. **Unless otherwise specified, use the default values throughout this assignment**. Note the descrease in training loss and corresponding decrease in validation errors. Paste the output into your report.

(a): Add code to plot out the network weights as images (one for each output, of size 28 by 28) after the last epoch. Grab a screenshot of the figure and include it in your report.

(b): Reduce the number of training examples to just 50. [Hint: alter the `torch.range` arguments in the `getMnistIterator` function]. Paste the output into your report and explain what is happening to the model.

4 Training a Multi-Layer Network on MNIST [20%]

(a): Add an extra layer to the network with 1000 hidden units and a `tanh` non-linearity. [Hint: use the `nn.sequential()` and `nn.add()` functions.] Train the model for 10 epochs and save the output into your report.

(b): Now set the learning rate to 10 and observe what happens during training. Save the output in your report and give a brief explanation.

5 Training a Convolutional Network on CIFAR [50%]

To change over to the CIFAR-10 dataset use the `-cifar` flag.
(a): Create a convolutional network with the following architecture:

- Convolution with 5 by 5 filters, 16 feature maps + Tanh nonlinearity.
- 2 by 2 max pooling.
- Convolution with 5 by 5 filters, 256 feature maps + Tanh nonlinearity.
- 2 by 2 max pooling.
- Flatten to vector.
- Linear layer with 128 hidden units + Tanh nonlinearity.
- Linear layer to 10 output units.

Train it for 20 epochs on the CIFAR-10 training set and copy the output into your report, along with a image of the first layer filters.

(b): Give a breakdown of the parameters within the above model, and the overall number.

(c): Replacing the vanilla SGD optimzer with (i) ADAM and (ii) RMSProp. Run each for 20 epochs and compare your loss at the end of training to SGD.