AI Programming (IT-3105) Module 3: Deep Learning using Tensorflow

Purpose:

- Learn to solve classification tasks using feed-forward neural networks with backpropagation.
- Gain hands-on experience with Google's Tensorflow, a system for configuring, training and testing neural networks.
- Learn to build a general interface for Tensorflow such that a wide variety of network architectures can be tested on a diverse array of datasets.

1 Assignment Overview

You will use Tensorflow to classify data from a wide variety of sources, including the classic MNIST benchmark of digit images. This requires good familiarity with the Tensorflow API along with clever insights into the proper topologies and parameter settings for artificial neural networks (ANNs) used as classifiers.

Your grade for this project module will be based on several factors:

- the ability to design an interface to Tensorflow that facilitates easy configuration and running of neural nets of widely varying topologies (a.k.a. architectures) on a diversity of datasets, and visualizing the results.
- the ability to design appropriate ANN architectures for particular classification tasks,
- the successful performance of these ANNs on those tasks, and
- the competence to explain the behavior of an ANN on a dataset by visualizing layer activation patterns, weight matrices and other useful relationships (such as dendrograms, described in deepnet_iface_details.pdf on the course web site).

2 Introduction

Artificial Neural Networks (ANNs) are one of a standard set of tools for solving classification problems. Recently, they have begun to show exceptional performance on speech and image classification tasks. Human-level competence on these particular tasks has eluded artificial intelligence for many years, but in a few short years, the algorithms have gone from neophyte to genius, often out-performing expert humans.

Systems such as Theano, Cafe and (most recently) Tensorflow have helped bring Deep Learning *to the masses* of students and professionals. As recently as 2010, it was difficult to build effective multi-layered ANNs for solving complex classification tasks, since deep-learning research papers rarely provided all of the essential implementation details. Today, Tensorflow and its army of users (many of whom post their results on the web) makes the job a whole lot easier.

Though neural networks have performed very well on many tasks for several decades, the recent quantum leap has greatly invigorated the field. It stems from many factors, including a deeper understanding of the backpropagation process, which have enabled researchers to build truly **deep** nets, often consisting of 10 or (or even 100 or more) hidden layers. However, neither exceptionally deep networks nor the (somewhat complex) techniques that have sparked recent breakthroughs are the centerpiece of this project. Rather, we will focus on one tool, TensorFlow, which serves as the computational backbone for many of these advances, but in a context that does not require very deep networks. A few hidden layers with relatively conventional dynamics (i.e., activation functions) will suffice.

By learning and using TensorFlow, you will open many possibilities for further research and development using deep neural networks, whether on a masters or PhD project, or out in industry.

As detailed in the lecture notes for this module, the *big wins* with Tensorflow are both its speed in handling matrix operations, a cornerstone of Deep ANNs, and its automatic calculation of partial derivatives, often involving variables that are quite distant from one another in a sequence of mathematical expressions, e.g., the weight of an interior ANN connection and the network's final output. Together, these Tensorflow features allow users to piece together fast ANNs using only a page or two of code.

The datasets for this assignment should not require extremely deep networks, but anywhere from one to three hidden layers may work optimally, depending upon your choice of the other parameters. Typical activation functions with which to experiment include: sigmoid, hyperbolic tangent (tanh), and rectified linear units (RELUs), all of which are primitives in Tensorflow. Typical error functions are the mean sum of squared errors (MSE) and cross-entropy, also provided as Tensorflow primitives.

3 Streamlining Neural Network Experimentation

A primary goal of this assignment is to streamline this trial-and-error process so that many different networks can be experimented with in a short period of time. To this end, you will build an interface to Tensorflow that facilitates these experiments. For detailed requirements and an in-depth explanation of several critical components of this system, see the document deepnet_iface_details.pdf provided on the course web pages.

4 The Demonstration Session

At the demo for this project, you will be asked to run your system on any of the datasets described in deepnet_iface_details.pdf using a diversity of network architectures and parameter settings. This process must occur without the need to re-compile or otherwise re-make your system. Your system must easily process the scenario specifications for each new run and display the appropriate graphics (such as weight arrays, dendrograms, etc.). Failure to satisfy these basic condition will incur significant point loss.

In addition, the performance of your system on different datasets will be checked. Perfection is not expected

but reasonably good results are. During a performance test, you will be given a dataset and then be expected to choose an ANN architecture and parameter settings that yield a good result. It is important to prepare ahead of time for this evaluation by knowing the datasets and what network architectures work best for each one.

Finally, you will be asked to explain the behavior of an ANN on a particular dataset – you can choose any of the data sets described in deepnet_iface_details.pdf, with the exception of the bit-counter. Your explanation must include the use of all of the following graphic tools: mappings, weight-and-bias visualizations and dendrograms. You should generate all of these diagrams prior to the demo and then use them as part of your verbal explanation during the demo. If you are working in a group of 2, either member may be asked to give this explanation, so both should be prepared. Your explanation need not be a complete mathematical proof, but it should shed considerable light on the activity *under the hood* that enables your network to successfully perform the given task.

5 Deliverables

1. The working system capable of handling any of the datasets and a wide selection of architectures (**15 points**).
2. Completion of 5 performance tests with reasonably good results. The notion of "reasonably good" is defined individually, for each dataset, in deepnet_iface_details.pdf (**10 points**)
3. The detailed explanation of a successful network (**5 points**).

There is no written report for this project module.

5.1 Other Practical Information

A zip file containing your report along with the commented code must be uploaded to BLACKBOARD prior to the demo session in which this project module is evaluated. You will not get explicit credit for the code, but it is crucial that we have the code online in the event that you decide to register a formal complaint about your grade (for the entire course).

The 30 total points for this module are 30 of the 100 points that are available for the entire semester.

The due date for this module is the 2nd demo session.