Machine Learning
M Hagan, A Jafari
Jan-20-2017

Lua and Torch Introduction

In this first LAB, you will experiment with the Lua programming language and Torch. Torch is a framework for deep learning built on top of the Lua programming language. It uses LuaJIT, C++ and CUDA. LuaJIT is a great scripting environment, with a transparent just in time compiler, simple, readable (like Python), with clean/consistent constructs, cleanest interface to C, and embeddable into any environment (iPhone apps, Video games,web backends, etc.). This is open source software and widely used in deep learning network architectures (Google DeepMind and Facebook AI research).

Lua Basics

You will need to read the Lua tutorials on the websites described in the document on the online classroom and the GettingStarted_Torch document to find answers to the following questions.

1. What does interactive interpreter mean?.
2. What are ”chunks” in Lua?
3. What is a global variable?
4. Name the types that exist in Lua.
5. What are local variables and blocks in Lua?
6. Identify differences between the interactive interpreter and Torch debugger. Describe one example.
7. Where can the `dofile` command can be used (interactive interpreter or Torch Debugger)? Explain the advantages of using this command.

Deliverables

1. For all parts below, include the results into one PDF file, and upload it to the dropbox on D2L. Include all program listings, plots, command line printouts, discussion, etc.

2. Hand in the answers to the questions in the Lua Basics section.

3. Download the `testfile.lua` file from D2L, run it with with `dofile` and print the results on the terminal. Debug it in ZeroBrane, and explain what it does.

4. Examine the following code, and explain what values will be printed. Next, run this code in the torch debugger. If the printed values did not match your expectation, explain what happened.

```
var = 10
    local i = 1

    while i<=var do
      local var = i*2
      print(var)
      i = i + 1
    end

    if i > 20 then
      local var
      var = 20
      print(var + 2)
    else
      print(var)
    end

    print(var)
```

5. Write a Lua function to solve Homework Problem 3. Perform steepest descent until the magnitude of the gradient is less than 0.01, starting with the initial weight and bias equal to zero. Use gnuplot.plot (see https://github.com/torch/gnuplot for instructions) to plot the sum squared error versus the iteration number. (You will need a line `require "gnuplot"`.) Also, plot the final network response on the same plot with the target values versus the input as p ranges from -1.5 to 1.5 in steps of 0.1. Plot the network response as a continuous line and the targets with a '+'.

6. Download the files `XOR_Stochastic.lua`, `XOR_Batch.lua`, `XOR_Optim.lua`. Run each one to verify that they are working correctly. These represent three different ways to train a multilayer network. They all train networks to learn the XOR function. Modify the three programs to use the GPU. Modify the `XOR_Stochastic.lua` file to save the mean

square error every 100 iterations, and then plot the mean square error versus iteration at the completion of training. Change the learning rate to find out what value will produce the fastest convergence.

7. Modify `XOR_Optim.lua` so that the network has three layers and a softmax function in the last layer. Find an appropriate learning rate and number of iterations to have a successful result.