UNIVERSITY OF OXFORD

Machine Learning

February 3, 2015

Introduction to Lua and Torch

In this first practical, we are going to spend some time learning Lua and Torch. Torch is a framework for deep learning built on top of the Lua programming language that is popular in research labs such as Facebook's and Google's, with features and ease-of-use that make it ideal for rapid prototyping. It is built on top of LuaJIT, a very fast Lua implementation.

Much of this practical will be reading and learning the language and libraries.

See the `README.md` file for setup instructions for the lab machine; the practical's files can be found here: https://github.com/oxford-cs-ml-2015/practical1

Lua basics

Read this Lua tutorial: http://tylerneylon.com/a/learn-lua/.

Make sure you can answer the following questions **NOT FOR HANDIN**:

- Why is the `local` keyword important? (hint: default variable scope is *not* local)
- What is the difference between `a.f()` and `a:f()`? (hint: one implicitly adds `self` as the first argument)
- What does `require` do, and why do we sometimes capture its return value but sometimes not? (this is a way to isolate things into namespaces to prevent naming conflicts, related to the answer to why we need to use `local`)
- What is the Lua equivalent of a list object? of a dictionary? How do you iterate over each of these?

Note that when we type into the interactive interpreter that we get by typing `th` or `luajit` into a terminal, we cannot make `local` variables: each line you type in is wrapped into a small function before it is executed. So, this is a case where we have no choice but to use globals.

Torch's tensor class

Tensors generalize vectors (1-d arrays) and matrices (2-d arrays) to n-dimensions. Torch has a flexible and efficient class for storing and manipulating these objects.

Read the documentation on these pages:

1. https://github.com/torch/torch7/blob/master/doc/tensor.md
2. https://github.com/torch/torch7/blob/master/doc/maths.md

As you read these, note that it's very important to do as many operations in-place as we can to avoid allocating lots of memory when we don't need to, and Torch tries to make it easy for you to do exactly that. As a simple example, consider the difference between

```
local t = torch.Tensor(10,10)
local t2 = torch.Tensor(10,10)
t3 = t + t2
```

and

```
local t = torch.Tensor(10,10)
local t2 = torch.Tensor(10,10)
t:add(t2)
```

if we don't need the value of `t` anymore and we can overwrite its entries. The first one allocates new memory but the second one does not. There are many situations where we avoid allocating memory; see e.g. the `expand` function.

Handin

Answer the following questions.

1. **Very briefly**, assuming we have defined `t` as:

   ```
   local t = torch.Tensor({{1,2,3},{4,5,6},{7,8,9}})
   ```

   List 3 expressions that can replace the first line below to slice (extract) the middle column from `t`:

   ```
   local col = ... -- extract the middle col from t
   print(col)      -- should print the 1-d tensor: 2,5,8
   ```

   (Remember: if you're using the interactive interpreter as opposed to a text editor, you need to remove the `local` keywords.)

2. What is the difference between a `Tensor` and a `Storage`?