Exercise 3 – Hello Computation Graph

Yoav Goldberg

In this exercise you will stop computing gradients by hand, and will move on to using a software library that provides automatic differentiation capabilities through the computation-graph abstraction.

We will be using computation-graph based software in the following assignments. The purpose of this exercise is to get you a gentle start when the next assignment arrives.

Part 0 – Choosing a neural network toolkit and making it work on your computer

Choose a toolkit from the following, install it, read the documentation, and try to implement something in it.

- DyNet (`https://dynet.io/`)
  This is the software we use for most of the work in the NLP lab. It provides very good speed on the CPU, decent GPU support, and is very convenient to work in. We used it to produce several strong publications. This is also the package which I (Yoav) know best, so I can provide the best low-level support for.

  It may be a bit hard to get to work on Windows (there *are* windows installation instructions, but these were never tested, and are not officially supported. If you are brave you can try it.) But you shouldn't be using windows anyway.

  You can follow the tutorial here: `https://github.com/clab/dynet/blob/master/examples/python/tutorials/tutorial-1-xor.ipynb`

  and/or here: `https://github.com/clab/dynet_tutorial_examples`

- PyTorch (`http://pytorch.org`)
  This is a package in similar spirit to DyNet, but much slower on the CPU, and does not support automatic batching. On the positive side, it has good documentation, it quite easy to install, and its backed by Facebook. For vision stuff, it will be my go-to package. For text and sequences, DyNet is still better until you start using Transformer-based models, in which case the PyTorch support will be better.

- TensorFlow (`https://www.tensorflow.org/`)
  This is the famous framework by Google. The webpage leads to a lot of

very good documentation / tutorials.

This framework is worth knowing because it is very popular in industry and can produce very fast code on GPU (and OK-ish speed on CPU). However, unlike the previous packages, it is *static*, in the sense that while in the previous packages you create a graph for each example, here you need to create a single graph at the beginning, and then feed it different examples. Also, unlike the previous packages, graph creation is a bit more cumbersome. Later in class, we'll encounter some examples for networks which are hard or even very hard to define using the static approach of TensorFlow (and Theano).

Part 1 – The Language Classification Network

Implement the Log-linear classifier and the MLP1 classifier from Assignment 1 using the toolkit. You should be able to reach very similar accuracies (if not better) than what you had in Assignment 1.

What to submit?

No need to submit anything. But make sure you manage to properly install and use your software package of choice. The next assignment will come shortly, and will assume you know how to write basic stuff in at least one of these libraries.