Software for Deep Learning and general numerical computation

Table of contents

Numerical computation in python

Numpy

- Main numerical computation in python.
- Vector and matrix operations (most computations come down to this)
- Linear algebra procedures (matrix decompositions, eigenvalues, system solver, etc)
- Numpy uses C/C++ backend, but CPU only!

Scipy

- Libraries for scientific computation (image signal processing, linear algebra, etc..)

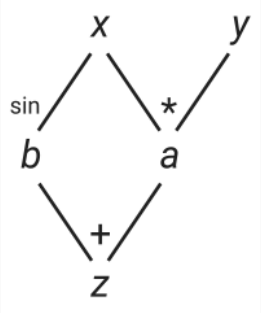
Pytorch

- Almost everything that numpy does + more
- Seamless execution on CPU + GPU
- Autodiff

What is Autodiff and why is it the best thing ever

Computational graphs

The following is a computational graph of the function $f(x, y) = \sin x + xy$



The gradients of each operation are well defined. This is an easy example and we can calculate it by hand, but for large examples it's impractical/infeasible.

Autodiff software

Auto diff software packages allow us to define our computation as a graph and then query the gradient of any node in the graph with respect to any other node. All the magic is done under the hood.

How does it work?

The process differs slightly depending on the framework, but essentially we have 3 key steps:

- Perform or define several steps of computation. During this step a computational graph will be built with intermediate results in each node.
- Call a backward() procedure on one or more desired nodes. This is where the software will propagate gradients recursively throughout the graph.
- Perform necessary operations using the calculated gradients.

Autodiff software examples

- Caffe - One of the first, outdated, mostly only for vision.
- Theano - One of the first, outdated, not maintained anymore. Declarative programming style
- Tensorflow - An improved version of Theano, developed by google. Actively used, very popular, many modules and functions available.
- Pytorch - Python version of the Torch library. Rapidly gaining popularity. Fast and flexible. Imperative programming style
- Microsoft CNTK, Deeplearning4j, etc - More frameworks.

Choose your weapon wisely: Tensorflow

Tensorflow

- + Developed by professionals
- + Very large community, tutorial base, modules, functions, contributions, etc
- + Tensorboard: Visualisation add-on
- + Very deployable on many platforms
- + Seamless CPU + GPU + TPU computation
- - Declarative programming style
- - Static computation graphs
- - Very steep learning curve
- - Worse debugging

Choose your weapon wisely: Pytorch

Pytorch

- + Large community, rapidly growing
- + Very large tutorial base (very important)
- + Easy to use, smooth learning curve
- + Imperative programming style, almost the same as using numpy
- + **Dynamic computation graphs** (this made me switch to pytorch from TF, it's **that** good)
- + Faster than Tensorflow
- + Seamless CPU + GPU computation (No TPU yet)
- - Less functionality than Tensorflow due to being much newer
- - Less deployable than Tensorflow

We will be using Pytorch for this course.