Software Libraries for Deep Learning

Sargur N. Srihari srihari@cedar.buffalo.edu

Python and NumPy

Python

- widely used high-level, general-purpose, interpreted, dynamic programming language.
- Its design philosophy emphasizes code readability, and its syntax allows **programmers** to express concepts in fewer lines of code than would be possible in languages such as C++ or Java

NumPy

- an extension to the Python programming language
- adds support for large, multi-dimensional arrays and matrices,
- along with a large library of high-level mathematical functions to operate on these arrays.

Publicly Available Python Libraries

1. Theano

- Open source numerical computation for Python
 - NumPy like syntax, compiled for CPUs or GPUs

2. Tensorflow

- Open source software library originally developed by Google Brain Team
- 3. Deep Scalable Sparse Tensor Network Engine (DSSTNE)
 - Amazon developed library for building deep learning ML models

4. PyTorch

PyTorch

- PyTorch is essentially a GPU enabled drop-in replacement for NumPy
- Equipped with higher-level functionality for building and training deep neural networks.
 - This makes PyTorch especially easy to learn if you are familiar with NumPy, Python and the usual deep learning abstractions
 - (convolutional layers, recurrent layers, SGD, etc.).

PyTorch vs TensorFlow

- PyTorch is better for rapid prototyping in research, for hobbyists and for small scale projects.
- TensorFlow is better for large-scale deployments, especially when cross-platform and embedded deployment is a consideration
- See https://awni.github.io/pytorch-tensorflow/

Keras

- <u>Keras</u> is a higher-level API with a configurable back-end. At the moment TensorFlow, Theano and CNTK are supported not PyTorch.
- Keras is also distributed with TensorFlow as a part of tf.contrib.
- Keras API is especially easy to use. It's one of the fastest ways to get running with many of the more commonly used deep neural network architectures.
 - API is not as flexible as PyTorch or core TensorFlow.

Tensorflow Fold

- DLwith dynamic computation graphs
 - https://research.googleblog.com/2017/02/announcing-tensorflow-fold-deep.html
- In much of machine learning, data used for training and inference undergoes a preprocessing step, where multiple inputs (such as images) are scaled to the same dimensions and stacked into batches. This lets high-performance deep learning libraries like <u>TensorFlow</u> run the same <u>computation graph</u> across all the inputs in the batch in parallel. Batching exploits the <u>SIMD</u> capabilities of modern GPUs and multi-core CPUs to speed up execution. However, there are many problem domains where the size and structure of the input data varies, such as <u>parse trees</u> in natural language understanding, <u>abstract syntax trees</u> in source code, <u>DOM trees</u> for web pages and more. In these cases, the different inputs have different computation graphs that don't naturally batch together, resulting in poor processor, memory, and cache utilization.
- TensorflowFold addresses these challenges

Natural Language Toolkit

- Import nltk
- Moby=nltk.text.Text(nltk.corpus.gutenberg. words('melville-moby_dick.txt'))
- print(moby.common_contexts("ahab")