A quick introduction to Tensorflow

Probabilistic Modeling Fall 2019
Many ML libraries:
Which One to Learn:

TensorFlow vs. PyTorch
Current Trending:

• Tensorflow still dominating Industry
Current Trending:

• Tensorflow still dominating Industry
Current Trending:

• PyTorch getting increasingly popular in Academia

An open-source library by Google:

TensorFlow:
Large-Scale Machine Learning on Heterogeneous Distributed Systems
(Preliminary White Paper, November 9, 2015)

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro,
Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow,
Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser,
Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray,
Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar,
Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals,
Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng

Google Research*
Further reading:

• Official website: https://www.tensorflow.org/
If want to master every details.

• *Deep Learning with Python* by Francois Chollet
  Focus on Keras

• Hands-On Machine Learning with Scikit-Learn and TensorFlow
  Tensorflow part is somewhat outdated. (Though this book is published
  in 2017, second edition was published at October 15 2019)
Core Functionalities:

• Augmented tensor operations (nearly identical to numpy)
  Seamless interfaces with existing programs.

• Automatic differentiation
  The very core of Optimization based algorithms.

• Parallel(CPU/GPU/TPU) and Distributed(multi-machine) Computing
  Essential for large (industrial level) applications.
  Implemented in C++/CUDA. Highly Efficient.
Automatic differentiation: Through back-propagation

• Only operations with “sub-gradient” can be applied on Tensor
Automatic differentiation: Through backpropagation

- Only operations with “sub-gradient” can be applied on Tensor Arithmetic: +, -, *, /
- Elementary functions: exp, log, max, sin, tan

In general

For convenience, we generalize the notation $h$ is the layer number, $w_{mn}^h$ connects $m$-th node in layer $h-1$ (i.e., $z_{m}^{h-1}$) to the $n$-th node in layer $h$ (i.e., $z_{n}^{h}$)

To compute each $\frac{\partial L}{\partial w_{mn}^h}$

1. Initialize $\frac{\partial L}{\partial w_{mn}^h} \leftarrow 0$

1. Find all the paths from $z_n^h$ to the output node $y$

2. For each path $s$

   2.1 For each node $z$ in $s$

      • Compute the partial derivative of $z$’s parent over $x$
      • If the node is the output node $y$, then compute $\frac{\partial L}{\partial y}$

      • Multiply all the partial derivatives along the path $s$ to obtain $g_s$

      • Add to the derivative:

$\frac{\partial L}{\partial w_{mn}^h} \leftarrow \frac{\partial L}{\partial w_{mn}^h} + g_s \frac{\partial z_n^{h-1}}{\partial w_{mn}^h}$
Automatic differentiation: Through back-propagation

- Only operations with “sub-gradient” can be applied on Tensor Arithmetic: +, -, *, /
- Elementary functions: exp, log, max, sin, tan

- What operations are not “differentiable”?
Automatic differentiation: Through back-propagation

- Only operations with “sub-gradient” can be applied on Tensor Arithmetic: +, -, *, /
- Elementary functions: exp, log, max, sin, tan

- What operations are not “differentiable”? For example: “Vanilla” sampling
Static vs Eager Mode

• Eager mode (PyTorch, Tensorflow 2.0)
  Just like using numpy

• Static mode (Tensorflow 1.x version)
  Predefine tensors and computation graphs then let TF engine to execute the graphs. Similar to defining Python functions.
Static vs Eager Mode

• Eager mode (PyTorch, Tensorflow 2.0)
  Just like using numpy

• Static mode (Tensorflow 1.x version)
  We focus solely on this mode in this tutorial

Subtlety appears here.
Working process: Tensor, Flow

• Tensor: multi-dimension array
Working process: Tensor, Flow

- flow: computation graph
Working process: Tensor, Flow

- flow: computation graph
Can be visualize by tensorboard
3 levels of tensorflow:

- **Primitive tensorflow**: lowest, finest control and most flexible
  Suitable for most machine learning and deep learning algorithms.
  We work at this level in this course.

- **Keras (Mostly for deep learning)**: highest, most convenient to use, lack flexibility

- **Tensorflow layers (Mostly for deep learning)**: somewhere at the middle.
General pipeline:

• Define inputs and variable tensors (weights/parameters).
  *Keras will take care of these for you.

• Define computation graphs from inputs tensors to output tensors.

• Define loss function and optimizer
  Once the loss is defined, the optimizer will compute the gradient for you!

• Execute the graphs.
  *Keras will take care of this for you as well
Getting started today:

• GPU acceleration

• Installation

• Demos
  o Arithmetic and tensor operations
  o Newton Raphson Logistic Regression
  o Tensorflow Style Logistic Regression
GPU acceleration:

• Literally need one if training on non-toy models and datasets.
GPU acceleration:

- Literally need one if training on non-toy models and datasets.
- Nvidia GPUs Only
Where to find (free) computing resources:

• Your own Gaming PC

• CHPC( University), CADE (Collage of Engineering)

• AWS/Google Cloud Platform: First time coupon.

• Google Colab: Always free, equipped with GPU and TPU!
Installation: Anaconda

• Installing with Anaconda could save you much work.

https://www.anaconda.com/
Installation: Anaconda

- Installed with Anaconda could save you much work.

https://www.anaconda.com/