A Guide to Using TensorFlow

Christos Kormaris

1 Introduction

TensorFlow is an open-source math and machine learning library. It was developed by a Google research team, called Google Brain, in 2015. It is written in Python, C++ and CUDA. It can be used to develop machine learning models, such as neural networks. The main feature of TensorFlow is its generalized builtin backpropagation computation, which offers the programmer great convenience in building their model, without having to calculate gradients themselves.

2 Symbolic Programming

(Christopher Bourez's blog) TensorFlow uses a style of programming called symbolic programming. Symbolic computation is a very different way of programming. Classical programming defines variables that hold values and operations to modify their values. In symbolic programming, it's more about building a graph of operations, that will be compiled later for execution. Such an architecture enables the code to be compiled and executed indifferently on CPU or GPU, for example. Symbols are an abstraction that does not require to know where it will be executed. The second aspect of symbolic computing is that it is much more like mathematical functions or formulas, that can be added, multiplied, differentiated, etc to give other math functions.

3 Symbolic & Imperative Deep Learning Frameworks

Some available deep learning frameworks and their style of programming are listed below:

- **Symbolic Deep Learning Frameworks:** TensorFlow, Theano, Keras (uses either TensorFlow or Theano as backend), TFLearn (framework wrapper around TensorFlow), Apache MXNET
- **Imperative Deep Learning Frameworks:** PyTorh, Caffe, Microsoft CNTK

4 What are tensors?

TensorFlow, as its name suggests, uses tensors to run its computations. Tensors are the same as mathematical arrays. A tensor of a zero dimensions is called a **scalar**. A tensor of single dimensions is called a **vector**. A tensor of 2 dimensions is called a **matrix**. Arrays with 3 or more dimensions are generally called tensors. Specifically, tensors are called 3-tensors, 4-tensors, ..., n-tensors, according to their dimensions.

Dimensions	Math Entity	Python Example
0	scalar	s = 3.141592
1	vector	v = [1, 2, 3]
2	matrix	m = [[1, 2, 3], [4, 5, 6]]
3	3-tensor	t = [[[1, 2, 3]], [[4, 5, 6]]]
n	n-tensor	...

Table 1: Table of Tensors

In addition to dimensionality, tensors have different data types as well. You can assign any of these data types to a tensor. TensorFlow automatically assigns tensors to default data types. You can customize your tensors to save memory.

Data type	Python type	Description
DT_FLOAT	tf.float32	32 bits floating point.
DT_DOUBLE	tf.float64	64 bits floating point.
DT_INT8	tf.int8	8 bits signed integer.
DT_INT16	tf.int16	16 bits signed integer.
DT_INT32	tf.int32	32 bits signed integer.
DT_INT64	tf.int64	64 bits signed integer.
DT_UINT8	tf.uint8	8 bits unsigned integer.
DT_STRING	tf.string	Variable length bytes arrays.
DT_BOOL	tf.bool	Boolean.

Table 2: Tensor Data Types

Finally, a tensor can also be named, if not by the programmer, then TensorFlow gives it a default name.

5 Calculating the update rules of neural networks

The update rules for the weights of various neural networks in TensorFlow are calculated automatically. TensorFlow contains builtin backprogration algorithms that calculate the gradients of the weights used for learning. The update rules, which are also included as a builtin function, are then applied using the gradients calculated before.

To make it more clear, here's an example of calling the builtin backpropagation algorithm and the update weights function in TensorFlow:

```
var_list  # the weights and the biases of the neural network
loss  # the loss function of the neural network
lr  # learning rate for the weights and the biases updates

# Adam Optimizer #
grads_and_vars = tf.train.AdamOptimizer(learning_rate=lr).
        compute_gradients(loss=loss, var_list=var_list)
apply_updates = tf.train.AdamOptimizer(learning_rate=lr).
        apply_gradients(grads_and_vars=grads_and_vars)
```

Listing 1: TensorFlow backpropagation

6 Gradient Optimizers

All the optimizers, provided by the TensorFlow framework, for computing gradients and applying gradients to variables are the following:

- tf.train.GradientDescentOptimizer
- tf.train.AdadeltaOptimizer
- tf.train.AdagradOptimizer
- tf.train.AdagradDAOptimizer
- tf.train.MomentumOptimizer
- tf.train.AdamOptimizer
- tf.train.FtrlOptimizer
- tf.train.ProximalGradientDescentOptimizer
- tf.train.ProximalAdagradOptimizer
- tf.train.RMSPropOptimizer

7 The TensorBoard

The TensorBoard is another feature offered by TensorFlow, which enables the visualization of TensorFlow graphs.

Follow the next steps to visualize a TensorBoard graph:

1. Create the TensorBoard log file by calling the following command, in Python code:

```
summary_writer = tf.summary.FileWriter(log_dir,
                                        graph=sess.graph)
```

Listing 2: FileWriter command

The "log_dir" argument defines the output directory where the TensorBoard will be stored. The output directory will automatically be created if it doesn't already exist.

2. After the execution of the "FileWriter" command, open a terminal (in Unix/Linux) or a command prompt (in Windows) and run the following command:

```
tensorboard --logdir = "path_to_the_graph"
```

Listing 3: TensorBoard command

3. Then, a message should appear that refers to the following URL: `http://localhost:6006` Open a browser (e.g. Firefox) and browse to that URL. TensorFlow now runs as a web app on port 6006, by default.

   If the "logdir" path was typed correctly, the TensorBoard with a visualization of the TensorFlow graph will be available to show.

NOTE: The port "6006" on the URL `http://localhost:6006` is `"goog"` upside down.