Data Science Training

November 2017

TensorFlow

Xavier Bresson

Data Science and AI Research Centre
NTU, Singapore

http://data-science-optum17.tk

Note: Based on Khandwala and Oshri's slides on TensorFlow

Outline

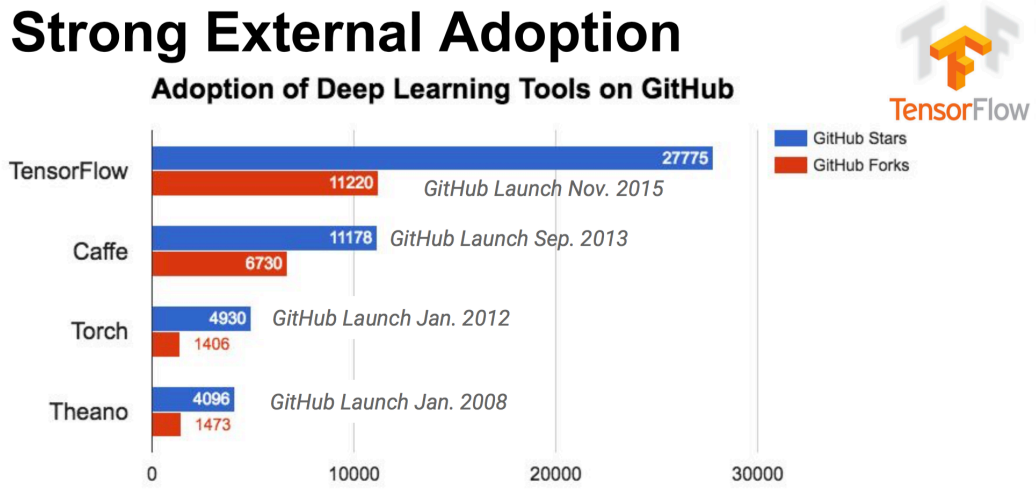
- What is TensorFlow?
- Programming Model : Computational Graph
- Run tf Session
- Train NN Model
- Variable Sharing
- Summary

Outline

- **What is TensorFlow?**
- Programming Model : Computational Graph
- Run tf Session
- Train NN Model
- Variable Sharing
- Summary

What is TensorFlow?

- Open source software library for numerical computation using data flow graphs
- Originally developed by Google Brain Team to conduct machine learning research
- Tensorflow is an interface for expressing machine learning algorithms, and an implementation for executing such algorithms
- First released Nov 2015 Apache 2.0 license
- Most popular DL library (many codes on GitHub)



50,000+ binary installs in 72 hours, 500,000+ since November, 2015
Most forked new repo on GitHub in 2015 (despite only being available in Nov, '15)

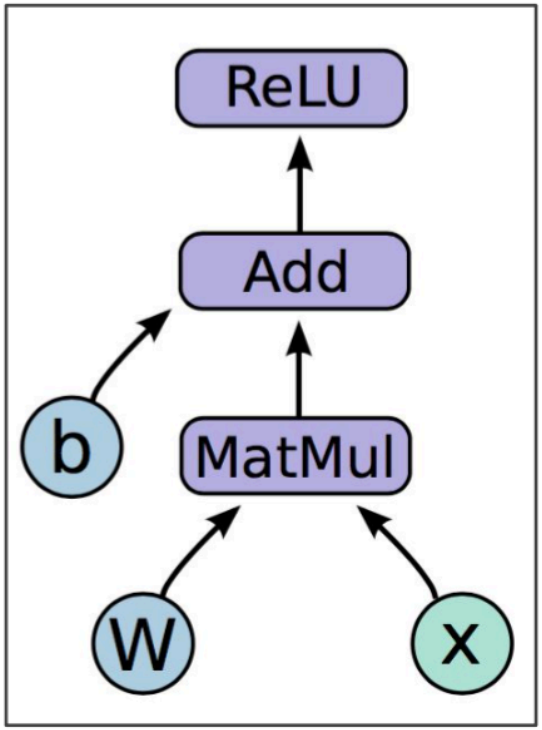
Outline

- What is TensorFlow?
- **Programming Model : Computational Graph**
- Run tf Session
- Train NN Model
- Variable Sharing
- Summary

Programming model

- Core idea: Computational graph
- Graph nodes are operations which have any number of inputs and outputs
- Graph edges are tensors which flow between nodes
- TensorFlow tensors $X[N_1, N_2, N_3, \ldots]$

$$h = ReLU(Wx + b)$$

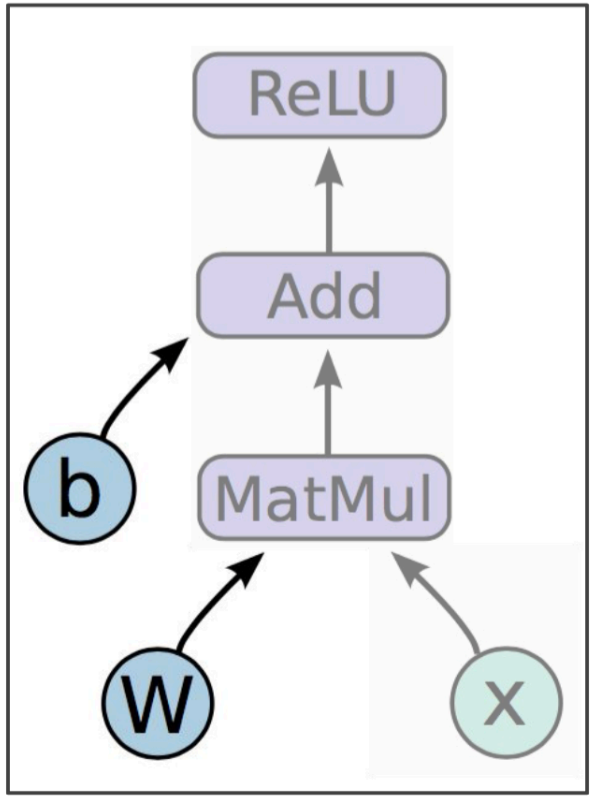


Neural Network Variables

$$h = ReLU(Wx + b)$$

Variables are stateful nodes which output their current value.

State is retained across multiple executions of a graph

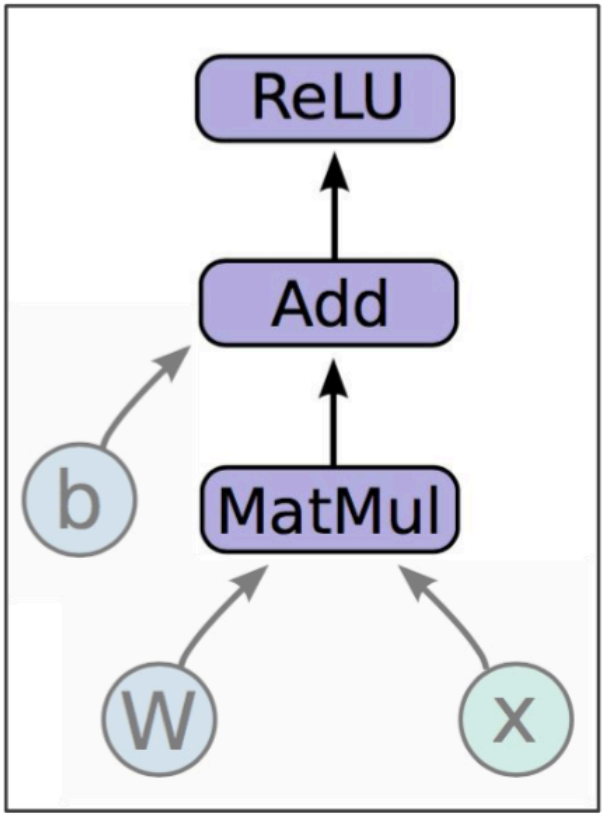


Placeholders (train batch)

$$h = ReLU(Wx + b)$$

Mathematical operations:

MatMul: Multiply two matrix values.
Add: Add elementwise (with broadcasting).
ReLU: Activate with elementwise rectified linear function.



Implementation

In code,

1. Create weights, including initialization
   W ~ *Uniform*(-1, 1); b = **0**
2. Create input placeholder x
   m * 784 input matrix
3. Build flow graph

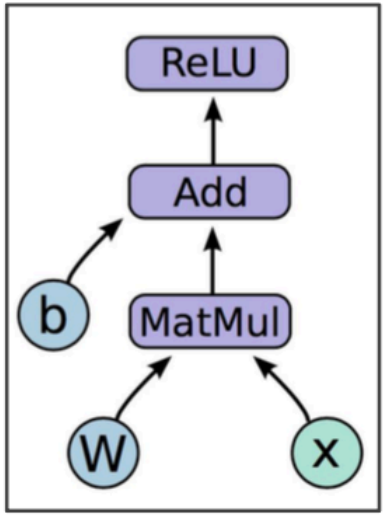
```
import tensorflow as tf

b = tf.Variable(tf.zeros((100,)))
W = tf.Variable(tf.random_uniform((784, 100), -1, 1))

x = tf.placeholder(tf.float32, (100, 784))

h = tf.nn.relu(tf.matmul(x, W) + b)
```

$$h = ReLU(Wx + b)$$



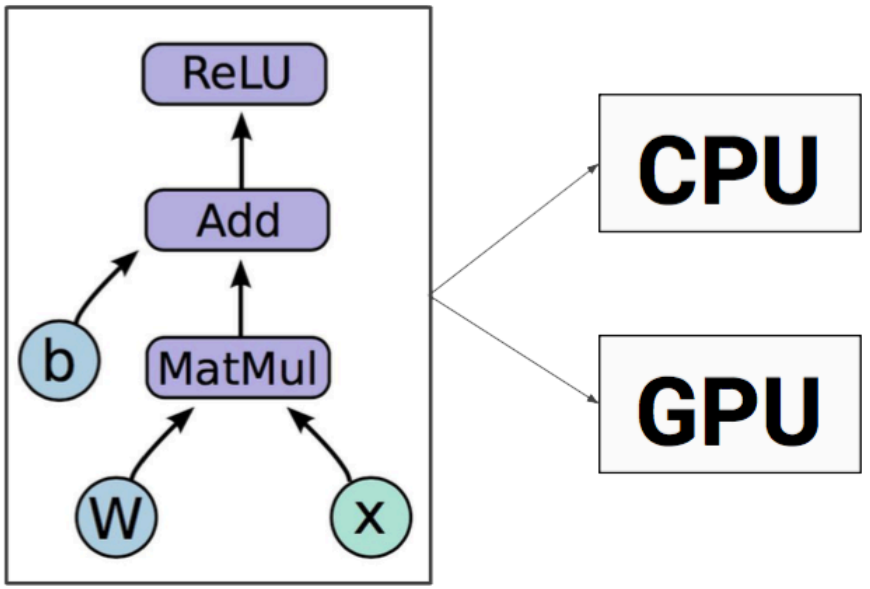
Outline

- What is TensorFlow?
- Programming Model : Computational Graph
- **Run tf Session**
- Train NN Model
- Variable Sharing
- Summary

Run computational graph

So far we have defined a **graph**.

We can deploy this graph with a **session**: a binding to a particular execution context (e.g. CPU, GPU)



Implementation

Getting output

`sess.run(fetches, feeds)`

Fetches: List of graph nodes. Return the outputs of these nodes.

Feeds: Dictionary mapping from graph nodes to concrete values. Specifies the value of each graph node given in the dictionary.

```
import numpy as np
import tensorflow as tf

b = tf.Variable(tf.zeros((100,)))
W = tf.Variable(tf.random_uniform((784, 100),
                -1, 1))

x = tf.placeholder(tf.float32, (100, 784))
h = tf.nn.relu(tf.matmul(x, W) + b)

sess = tf.Session()
sess.run(tf.initialize_all_variables())
sess.run(h, {x: np.random.random(100, 784)})
```

Loss

Use **placeholder** for **labels**

Build loss node using labels and **prediction**

```
prediction = tf.nn.softmax(...)  #Output of neural network
label = tf.placeholder(tf.float32, [100, 10])

cross_entropy = -tf.reduce_sum(label * tf.log(prediction), axis=1)
```

Gradients

```
train_step = tf.train.GradientDescentOptimizer(0.5).minimize(cross_entropy)
```

- `tf.train.GradientDescentOptimizer` is an **Optimizer** object
- `tf.train.GradientDescentOptimizer(lr).minimize(cross_entropy)` adds optimization **operation** to computation graph

TensorFlow graph **nodes** have **attached gradient operations**

Gradient with respect to **parameters** computed with **backpropagation**

...automatically

Outline

- What is TensorFlow?
- Programming Model : Computational Graph
- Run tf Session
- **Train NN Model**
- Variable Sharing
- Summary

Train operation

```
prediction = tf.nn.softmax(...)
label = tf.placeholder(tf.float32, [None, 10])

cross_entropy = tf.reduce_mean(-tf.reduce_sum(label * tf.log(prediction),
reduction_indices=[1]))

train_step = tf.train.GradientDescentOptimizer(0.5).minimize(cross_entropy)
```

Training neural networks

`sess.run(train_step, feeds)`

1. Create Session
2. Build training schedule
3. Run `train_step`

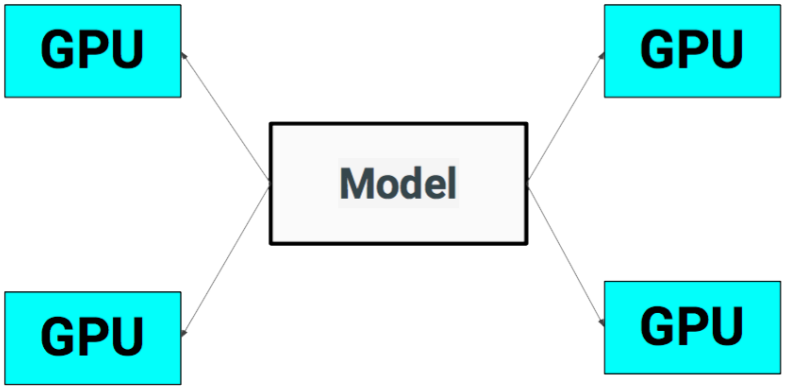
```
sess = tf.Session()
sess.run(tf.initialize_all_variables())

for i in range(1000):
     batch_x, batch_label = data.next_batch()
     sess.run(train_step, feed_dict={x: batch_x,
                                     label: batch_label}
```

Outline

- What is TensorFlow?
- Programming Model : Computational Graph
- Run tf Session
- Train NN Model
- **Variable Sharing**
- Summary

Variable sharing



`tf.variable_scope()` provides simple name-spacing to avoid clashes

`tf.get_variable()` creates/accesses variables from within a variable scope

```
with tf.variable_scope("foo"):
     v = tf.get_variable("v", shape=[1])  # v.name == "foo/v:0"

with tf.variable_scope("foo", reuse=True):
     v1 = tf.get_variable("v")      # Shared variable found!

with tf.variable_scope("foo", reuse=False):
     v1 = tf.get_variable("v")      # CRASH foo/v:0 already exists!
```

Outline

- What is TensorFlow?
- Programming Model : Computational Graph
- Run tf Session
- Train NN Model
- Variable Sharing
- **Summary**

Summary

1. Build a graph
    a. Feedforward
    b. Loss
    c. Optimization scheme

2. Initialize a session

3. Train NN with session.run(train_step, feed_dict).
   Backpropagation is automatically carried out.

Questions?