Quick Tensorflow Tutorial

Andreas Eitel
First Organization

- Computers in Pool are equipped with Nvidia GPUs GTX 1060 to train your networks
- Tensorflow 1.1 is already installed
- We’ll meet from now one in the computer pool every Monday to help you with the exercises
What is a Tensor?

- Central unit of data in TensorFlow
- A tensor consists of a set of primitive values shaped into an array
- A scalar is a Tensor
- A vector is a Tensor
- A matrix is a Tensor

3 # a rank 0 tensor; a scalar with shape []
[1., 2., 3.] # a rank 1 tensor; a vector with shape [3]
[[1., 2., 3.], [4., 5., 6.]] # a rank 2 tensor; a matrix with shape [2, 3]
[[[1., 2., 3.], [[7., 8., 9.]]]] # a rank 3 tensor with shape [2, 1, 3]
Tensorflow vs. Numpy

- Both support Ndarrays
- Both use python as front-end
- Numpy does not offer methods to create tensors
- Numpy has no automatic derivative computation
- Numpy has no GPU support
**Tensorflow vs. Numpy**

```python
In [38]: import numpy as np

In [51]: a = np.zeros((2,2)); b = np.ones((2,2))

In [52]: np.sum(b, axis=1)
Out[52]: array([2., 2.])

In [53]: a.shape
Out[53]: (2, 2)

In [54]: np.reshape(a, (1,4))
Out[54]: array([[0., 0., 0., 0.]])

In [55]: np.dot(a,b)
Out[55]: array([[0., 0.],
               [0., 0.]])
```
Tensorflow vs. Numpy

```
In [38]: import numpy as np

In [51]: a = np.zeros((2,2)); b = np.ones((2,2))

In [52]: np.sum(b, axis=1)
Out[52]: array([ 2.,  2.])

In [53]: a.shape
Out[53]: (2, 2)

In [54]: np.reshape(a, (1,4))
Out[54]: array([[ 0.,  0.,  0.,  0.]])

In [55]: np.dot(a,b)
Out[55]: array([[ 0.,  0.],
                 [ 0.,  0.]])

In [56]: import tensorflow as tf

In [57]: tf.InteractiveSession()
a = tf.zeros((2,2)); b = tf.ones((2,2))

In [58]: a.get_shape()
Out[58]: TensorShape([Dimension(2), Dimension(2)])

In [59]: tf.reshape(a, (1,4)).eval()
Out[59]: array([[ 0.,  0.,  0.,  0.]], dtype=generic.float32)

In [61]: tf.matmul(a,b).eval()
Out[61]: array([[ 0.,  0.],
                 [ 0.,  0.]], dtype=generic.float32)
```

Requires explicit evaluation
Tensorflow Sessions

- A computational graph is a series of Tensorflow operations arranged into a graph of nodes
- To evaluate nodes, we must run a session

```python
In [65]:
with tf.Session() as sess:
    print(sess.run(tf.matmul(a, b)))
    print("is the same as")
    print(tf.matmul(a, b).eval())

[[ 0.  0.]
 [ 0.  0.]]

is the same as
[[ 0.  0.]
 [ 0.  0.]]
Placeholders and Variables

- For machine learning you want a model to take arbitrary inputs and add trainable parameters to the graph
- External input using **Placeholders**
- Trainable parameters using **Variables**, which are constructed with type and initial value

```python
In [66]:
1. x = tf.placeholder(tf.float32)
2. W = tf.Variable([.3], dtype=tf.float32)
3. b = tf.Variable([-3], dtype=tf.float32)

In [67]:
1. linear_model = W*x + b
```
Initialization

- Variables are not initialized when calling `tf.Variable`
- Need to call initialization

```python
In [66]:
x = tf.placeholder(tf.float32)
W = tf.Variable([.3], dtype=tf.float32)
b = tf.Variable([-3], dtype=tf.float32)

In [73]:
linear_model = W*x + b

In [79]:
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())
    print(sess.run(W))
    print(sess.run(b))

[ 0.30000001]
[-0.30000001]
```
Evaluation

Use the Placeholder to evaluate linear_model for several values of x

In [66]:
   1. x = tf.placeholder(tf.float32)
   2. W = tf.Variable([.3], dtype=tf.float32)
   3. b = tf.Variable([-.3], dtype=tf.float32)

In [73]:
   1. linear_model = W*x + b

In [81]:
   1. with tf.Session() as sess:
      2.     sess.run(tf.global_variables_initializer())
      3.     print(sess.run(linear_model, {x: [1, 2, 3, 4]}))
   
   [ 0.3000001  0.6000002  0.9000004]
Linear Regression Example

Still need labels, loss and optimizer for optimizing the parameters W,b

```python
In [66]:
   1 x = tf.placeholder(tf.float32)
   2 W = tf.Variable([.3], dtype=tf.float32)
   3 b = tf.Variable([-3], dtype=tf.float32)

In [73]:
   1 linear_model = W*x + b

In [83]:
   1 # labels
   2 y = tf.placeholder(tf.float32)
   3
   4 # loss
   5 loss = tf.reduce_sum(tf.square(linear_model - y)) # sum of the squares
   6 # optimizer
   7 optimizer = tf.train.GradientDescentOptimizer(0.01)
   8 train = optimizer.minimize(loss)
   9
   10 # training data
   11 x_train = [1, 2, 3, 4]
   12 y_train = [0, -1, -2, -3]
   13 # training loop
   14 init = tf.global_variables_initializer()
   15 sess = tf.Session()
   16 sess.run(init) # reset values to wrong
   17 for _ in range(1000):
   18   sess.run(train, {x: x_train, y: y_train})
   19
   20 # evaluate training accuracy
   21 curr_W, curr_b, curr_loss = sess.run([W, b, loss], {x: x_train, y: y_train})
   22 print("W: %s b: %s loss: %.5f", (curr_W, curr_b, curr_loss))
W: [-0.9999969] b: [ 0.99999802] loss: 5.6997e-11
```
Visualize the Graph in TensorBoard
Thank you for your attention!