cs224d-2

May 11, 2016

In [23]:
```
import numpy as np
import tensorflow as tf


tf.InteractiveSession()
st = tf.constant('Hello, TensorFlow!')
matrix1 = tf.constant([[3., 3.]])
#noted that there are two paranteces for (())
a = tf.zeros((2,2)); b = tf.ones((2,2)) ;

#tf.reduce_sum(b, reduction_indices=1).eval()

a.get_shape()
#TensorShape([Dimension(2), Dimension(2)])

#tf.reshape(a, (1,4)).eval()
#array([[ 0.,  0.,  0.,  0.]], dtype=float32)

#when you put .eval at the end of things, it prints out things
#tf.matmul(a,b).eval()
c = tf.matmul(matrix1, b)

with tf.Session() as sess:
    print (sess.run(c))
    print (c.eval())


#if the variable is defined as a variable, it must have a name

W = tf.Variable(tf.zeros((2,2)), name= "weights")
R = tf.Variable(tf.random_normal((2,2)), name = 'random_weights')

with tf.Session() as sess:
    sess.run(tf.initialize_all_variables())
    print(sess.run(W))
    print (sess.run(R))

stat = tf.Variable(0, name = 'counter')
new_value = tf.add(stat, tf.constant(1))
update = tf.assign(stat, new_value)

with tf.Session() as sess:
    sess.run(tf.initialize_all_variables())
```

```
    print(sess.run(stat)) #0
    for _ in range(3):
        sess.run(update)
        print(sess.run(stat))
    # 1,2,3 print


aa = np.zeros((3,3))
ta = tf.convert_to_tensor(aa)

with tf.Session() as sess:
    print (sess.run(ta))


#don't use eval() just use the with tf.Session


input1 = tf.placeholder(tf.float32)
input2 = tf.placeholder(tf.float32)

output = tf.mul(input1,input2)
with tf.Session() as sess:
    print(sess.run([output], feed_dict = {input1:[.7], input2:[2.]}))


with tf.variable_scope("foo"):
    with tf.variable_scope("bar",reuse = True):
        v = tf.get_variable("v",[1])
assert v.name == "foo/bar/v:0"

with tf.variable_scope("foo"):
    with tf.variable_scope("bar",reuse = True):
        v1 = tf.get_variable("v",[1])


assert v1 == v


#define data sizes and batch size

n_samples = 1000
batch_size = 100

#define input data

X_data = np.arange(100, step = .1)
y_data = X_data + 20 * np.sin( X_data/10)

#Tensorflow is finicky about shapes, so resize

x_data = np.reshape(X_data, (n_samples,1))
y_data = np.reshape(y_data, (n_samples,1))

#Define placeholders for input
```

```
        X = tf.placeholder(tf.float32, shape = (batch_size, 1))
        y = tf.placeholder(tf.float32, shape = (batch_size,1))


        with tf.variable_scope("linear-regression"):
            W = tf.get_variable("weights", (1,1), initializer = tf.random_normal_initializer())

            b = tf.get_variable("bias", (1,), initializer = tf.constant_initializer(0.0))

            y_pred = tf.matmul(X,W) + b

            loss = tf.reduce_sum((y - y_pred)**2/n_samples)

            opt_operation = tf.train.AdamOptimizer().minimize(loss)


        with tf.Session() as sess:
            sess.run(tf.initialize_all_variables())

        for _ in range (500):
            indices = np.random-choice(n_samples, batch_size)
            X_batch, y_batch = X_data[indices], y_data[indices]

        _, loss_val = sess.run([opt_operation, loss], feed_dict = {X: X_batch, y: y_batch})


        embeddings = tf.Variable(tf.random_uniform([vocabulary+size, embedding_size], -1., 1.))
        embed = tf.nn.embedding_lookup(embeddings, train_inputs)
```

```
[[ 6.  6.]]
[[ 6.  6.]]
[[ 0.  0.]
 [ 0.  0.]]
[[-0.27117929  1.35933018]
 [ 0.14686877 -1.37026691]]
0
1
2
3
[[ 0.  0.  0.]
 [ 0.  0.  0.]
 [ 0.  0.  0.]]
[array([ 1.39999998], dtype=float32)]
```

```
        ---------------------------------------------------------------------------

        ValueError                                Traceback (most recent call last)

        <ipython-input-23-800b3b00249f> in <module>()
        101
        102 with tf.variable_scope("linear-regression"):
    --> 103     W = tf.get_variable("weights", (1,1), initializer = tf.random_normal_initializer())
        104
        105     b = tf.get_variable("bias", (1,), initializer = tf.constant_initializer(0.0))
```

```
/home/juser/.local/lib/python2.7/site-packages/tensorflow/python/ops/variable_scope.pyc in get_v
240   return get_variable_scope().get_variable(_get_default_variable_store(), name,
241                                            shape, dtype, initializer,
--> 242                                            trainable, collections)
243
244


/home/juser/.local/lib/python2.7/site-packages/tensorflow/python/ops/variable_scope.pyc in get_v
173     with ops.name_scope(None):
174       return var_store.get_variable(full_name, shape, dtype, initializer,
--> 175                                     self.reuse, trainable, collections)
176
177


/home/juser/.local/lib/python2.7/site-packages/tensorflow/python/ops/variable_scope.pyc in get_v
 89       if should_check and not reuse:
 90         raise ValueError("Over-sharing: Variable %s already exists, disallowed."
---> 91                          " Did you mean to set reuse=True in VarScope?" % name)
 92       found_var = self._vars[name]
 93       if not shape.is_compatible_with(found_var.get_shape()):


ValueError: Over-sharing: Variable linear-regression/weights already exists, disallowed. Did you
```

In [ ]:

In [ ]: