With TF 1.0!

Lab 2

Linear Regression

Sung Kim <hunkim+ml@gmail.com>

Code: https://github.com/hunkim/DeepLearningZeroToAll/

Call for comments

Please feel free to add comments directly on these slides

Other slides: https://goo.gl/jPtWNt

Picture from http://www.tssablog.org/archives/3280

With TF 1.0!

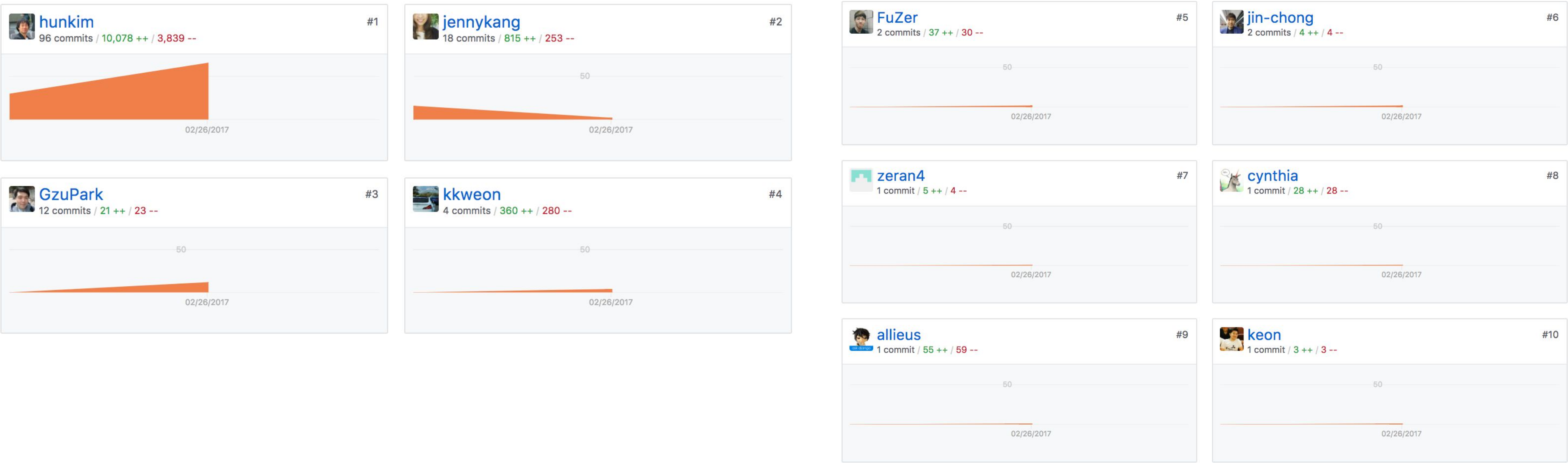
Lab 2

Linear Regression

Sung Kim <hunkim+ml@gmail.com>

Code: https://github.com/hunkim/DeepLearningZeroToAll/

https://github.com/hunkim/DeepLearningZeroToAll/



Hypothesis and cost function

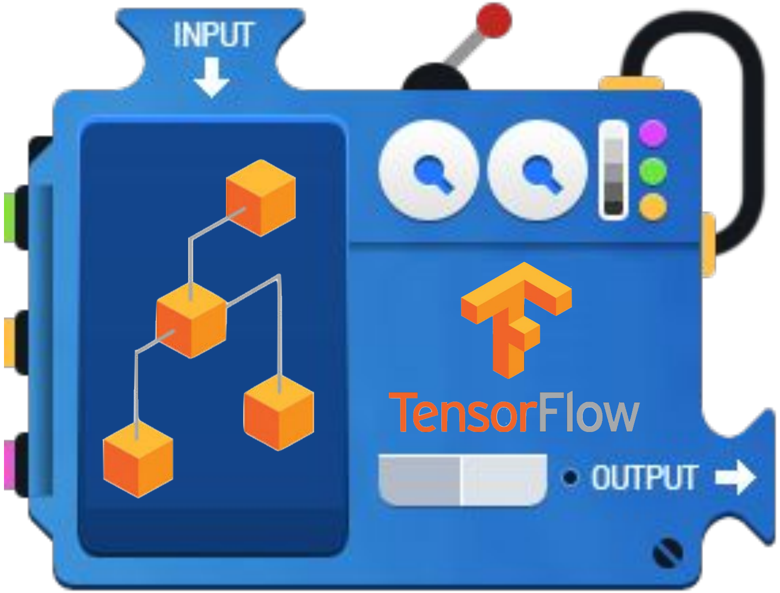
$$H(x) = Wx + b$$

$$cost(W, b) = \frac{1}{m} \sum_{i=1}^{m} (H(x^{(i)}) - y^{(i)})^2$$

TensorFlow Mechanics

2 feed data and run graph (operation)
sess.run (op, feed_dict={x: x_data})

1 Build graph using
TensorFlow operations



3 update variables
in the graph
(and return values)

1 Build graph using TF operations

$$H(x) = Wx + b$$

```
# X and Y data
x_train = [1, 2, 3]
y_train = [1, 2, 3]

W = tf.Variable(tf.random_normal([1]), name='weight')
b = tf.Variable(tf.random_normal([1]), name='bias')
# Our hypothesis XW+b
hypothesis = x_train * W + b
```

$$cost(W,b) = \frac{1}{m} \sum_{i=1}^{m} (H(x^{(i)}) - y^{(i)})^2$$

```
# cost/loss function
cost = tf.reduce_mean(tf.square(hypothesis - y_train))
```

1 Build graph using TF operations

$$cost(W,b) = \frac{1}{m} \sum_{i=1}^{m} (H(x^{(i)}) - y^{(i)})^2$$

```
t = [1., 2. ,3. ,4.]
tf.reduce_mean(t) ==> 2.5
```

```
# cost/loss function
cost = tf.reduce_mean(tf.square(hypothesis - y_train))
```

GradientDescent

```
# Minimize
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
train = optimizer.minimize(cost)
```

https://www.tensorflow.org/api_docs/python/tf/reduce_mean

Run/update graph and get results

```
# Launch the graph in a session.
sess = tf.Session()
# Initializes global variables in the graph.
sess.run(tf.global_variables_initializer())

# Fit the line
for step in range(2001):
    sess.run(train)
    if step % 20 == 0:
        print(step, sess.run(cost), sess.run(W), sess.run(b))
```

Full code (less than 20 lines)

```python
import tensorflow as tf

# X and Y data
x_train = [1, 2, 3]
y_train = [1, 2, 3]

W = tf.Variable(tf.random_normal([1]), name='weight')
b = tf.Variable(tf.random_normal([1]), name='bias')

# Our hypothesis XW+b
hypothesis = x_train * W + b

# cost/loss function
cost = tf.reduce_mean(tf.square(hypothesis - y_train))

# Minimize
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
train = optimizer.minimize(cost)

# Launch the graph in a session.
sess = tf.Session()
# Initializes global variables in the graph.
sess.run(tf.global_variables_initializer())

# Fit the line
for step in range(2001):
    sess.run(train)
    if step % 20 == 0:
        print(step, sess.run(cost), sess.run(W), sess.run(b))
```

```
'''
0 2.82329 [ 2.12867713] [-0.85235667]
20 0.190351 [ 1.53392804] [-1.05059612]
40 0.151357 [ 1.45725465] [-1.02391243]
...

1920 1.77484e-05 [ 1.00489295] [-0.01112291]
1940 1.61197e-05 [ 1.00466311] [-0.01060018]
1960 1.46397e-05 [ 1.004444] [-0.01010205]
1980 1.32962e-05 [ 1.00423515] [-0.00962736]
2000 1.20761e-05 [ 1.00403607] [-0.00917497]
'''
```

Placeholders

```
In [7]: a = tf.placeholder(tf.float32)
        b = tf.placeholder(tf.float32)
        adder_node = a + b  # + provides a shortcut for tf.add(a, b)

        print(sess.run(adder_node, feed_dict={a: 3, b: 4.5}))
        print(sess.run(adder_node, feed_dict={a: [1,3], b: [2, 4]}))

7.5
[ 3.  7.]
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-01-basics.ipynb

Placeholders

```
# X and Y data
x_train = [1, 2, 3]
y_train = [1, 2, 3]

# Now we can use X and Y in place of x_data and y_data
# # placeholders for a tensor that will be always fed using feed_dict
# See http://stackoverflow.com/questions/36693740/
X = tf.placeholder(tf.float32)
Y = tf.placeholder(tf.float32)
...
# Fit the line
# Fit the line
for step in range(2001):
   cost_val, W_val, b_val, _ = \
       sess.run([cost, W, b, train],
                feed_dict={X: [1, 2, 3], Y: [1, 2, 3]})
   if step % 20 == 0:
       print(step, cost_val, W_val, b_val)
```

Full code with placeholders

```
import tensorflow as tf
W = tf.Variable(tf.random_normal([1]), name='weight')
b = tf.Variable(tf.random_normal([1]), name='bias')

X = tf.placeholder(tf.float32, shape=[None])
Y = tf.placeholder(tf.float32, shape=[None])

# Our hypothesis XW+b
hypothesis = X * W + b
# cost/loss function
cost = tf.reduce_mean(tf.square(hypothesis - Y))
# Minimize
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
train = optimizer.minimize(cost)

# Launch the graph in a session.
sess = tf.Session()
# Initializes global variables in the graph.
sess.run(tf.global_variables_initializer())

# Fit the line
for step in range(2001):
    cost_val, W_val, b_val, _ = sess.run([cost, W, b, train],
                 feed_dict={X: [1, 2, 3], Y: [1, 2, 3]})
    if step % 20 == 0:
        print(step, cost_val, W_val, b_val)
```

```
...
1980 1.32962e-05 [ 1.00423515] [-0.00962736]
2000 1.20761e-05 [ 1.00403607] [-0.00917497]


# Testing our model
print(sess.run(hypothesis, feed_dict={X: [5]}))
print(sess.run(hypothesis, feed_dict={X: [2.5]}))
print(sess.run(hypothesis,
                             feed_dict={X: [1.5, 3.5]}))


[ 5.0110054]
[ 2.50091505]
[ 1.49687922  3.50495124]
```

Full code with placeholders

```python
import tensorflow as tf
W = tf.Variable(tf.random_normal([1]), name='weight')
b = tf.Variable(tf.random_normal([1]), name='bias')
X = tf.placeholder(tf.float32, shape=[None])
Y = tf.placeholder(tf.float32, shape=[None])

# Our hypothesis XW+b
hypothesis = X * W + b
# cost/loss function
cost = tf.reduce_mean(tf.square(hypothesis - Y))
# Minimize
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
train = optimizer.minimize(cost)

# Launch the graph in a session.
sess = tf.Session()
# Initializes global variables in the graph.
sess.run(tf.global_variables_initializer())

# Fit the line with new training data
for step in range(2001):
   cost_val, W_val, b_val, _ = sess.run([cost, W, b, train],
        feed_dict={X: [1, 2, 3, 4, 5],
                   Y: [2.1, 3.1, 4.1, 5.1, 6.1]})
   if step % 20 == 0:
       print(step, cost_val, W_val, b_val)
```

```
...
1960 3.32396e-07 [ 1.00037301] [ 1.09865296]
1980 2.90429e-07 [ 1.00034881] [ 1.09874094]
2000 2.5373e-07 [ 1.00032604] [ 1.09882331]
```

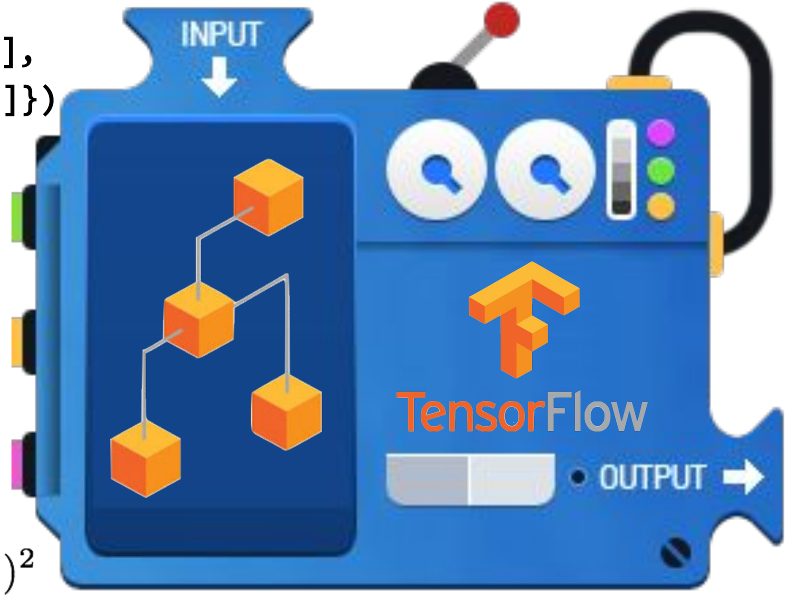
```python
# Testing our model
print(sess.run(hypothesis, feed_dict={X: [5]}))
print(sess.run(hypothesis, feed_dict={X: [2.5]}))
print(sess.run(hypothesis,
                   feed_dict={X: [1.5, 3.5]}))
```

```
[ 6.10045338]
[ 3.59963846]
[ 2.59931231  4.59996414]
```

TensorFlow Mechanics

2 feed data and run graph (operation)
sess.run (op, feed_dict={x: x_data})

```
feed_dict={X: [1, 2, 3, 4, 5],
 Y: [2.1, 3.1, 4.1, 5.1, 6.1]})
```



1 Build graph using TensorFlow operations

$$H(x) = Wx + b$$

$$cost(W, b) = \frac{1}{m} \sum_{i=1}^{m} (H(x^{(i)}) - y^{(i)})^2$$

3 update variables in the graph (and return values)

With TF 1.0!

Lab 3

Minimizing Cost

Sung Kim <hunkim+ml@gmail.com>