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!

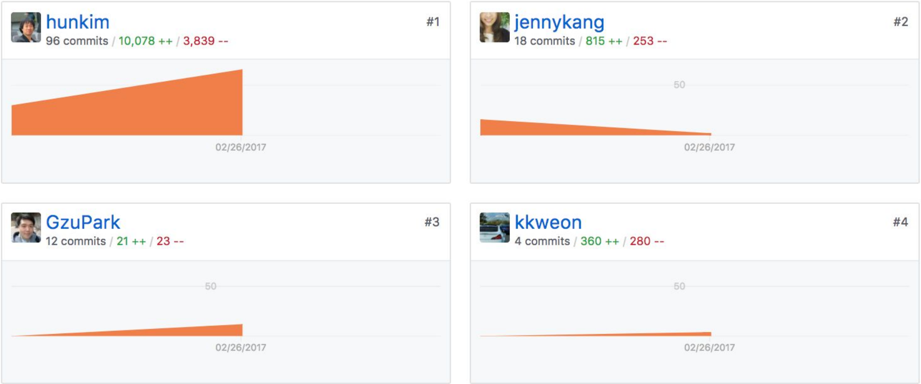
Lab 2

Linear Regression

Sung Kim <hunkim+ml@gmail.com>

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

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

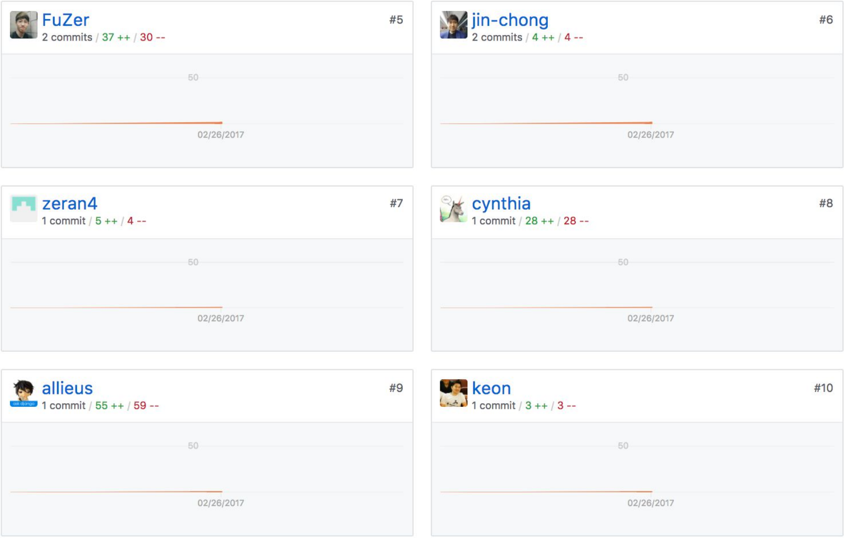


hunkim #1
96 commits / 10,078 ++ / 3,839 --

jennykang #2
18 commits / 815 ++ / 253 --

GzuPark #3
12 commits / 21 ++ / 23 --

kkweon #4
4 commits / 360 ++ / 280 --



FuZer #5
2 commits / 37 ++ / 30 --

jin-chong #6
2 commits / 4 ++ / 4 --

zeran4 #7
1 commit / 5 ++ / 4 --

cynthia #8
1 commit / 28 ++ / 28 --

allieus #9
1 commit / 55 ++ / 59 --

keon #10
1 commit / 3 ++ / 3 --

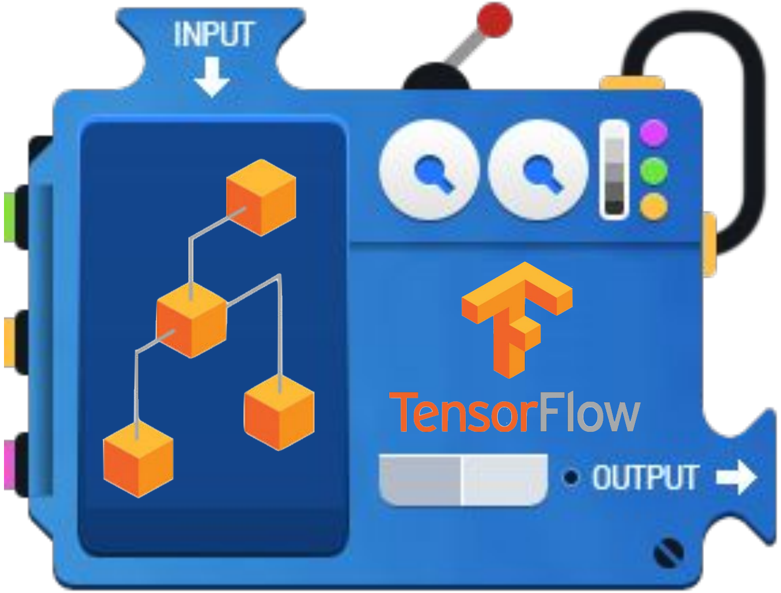
Hypothesis and cost function

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

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

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

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

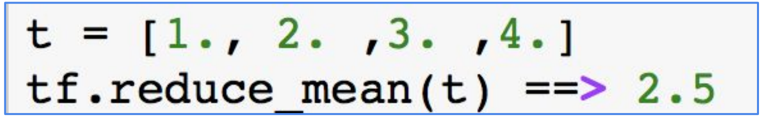
```
# X and Y data
x_train = [1, 2, 3]
y_train = [1, 2, 3]

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

2

3

Run/update graph and get results

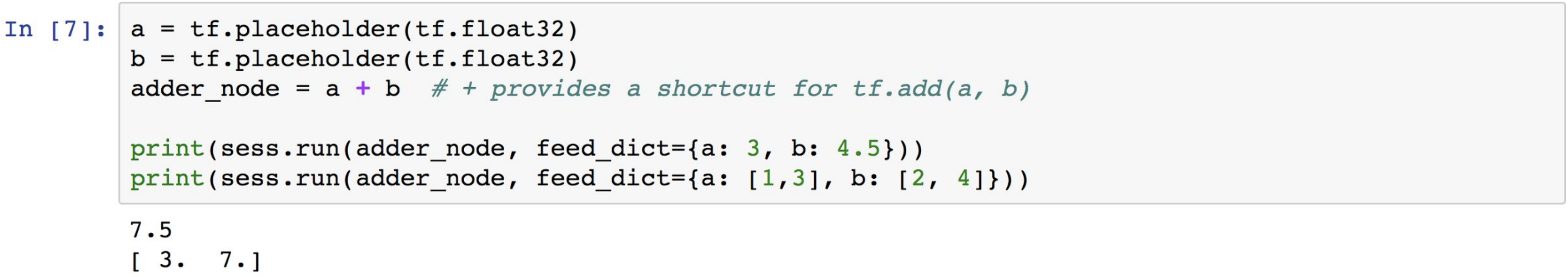
```
# Launch the graph in a session.
sess = tf.Session()
# Initializes global variables in the graph.
sess.run(tf.global_variables_initializer())

# 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

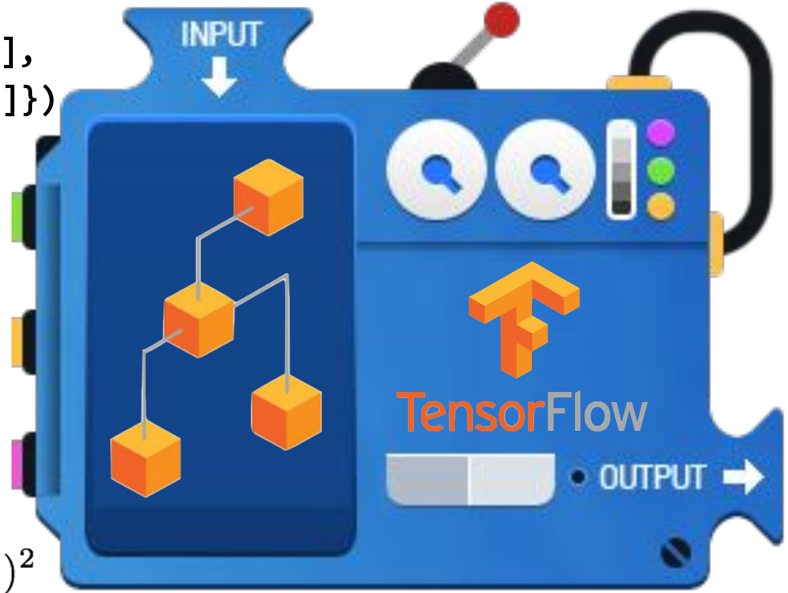
2 feed data and run graph (operation)
sess.run (op, feed_dict={x: x_data})

```
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>
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!

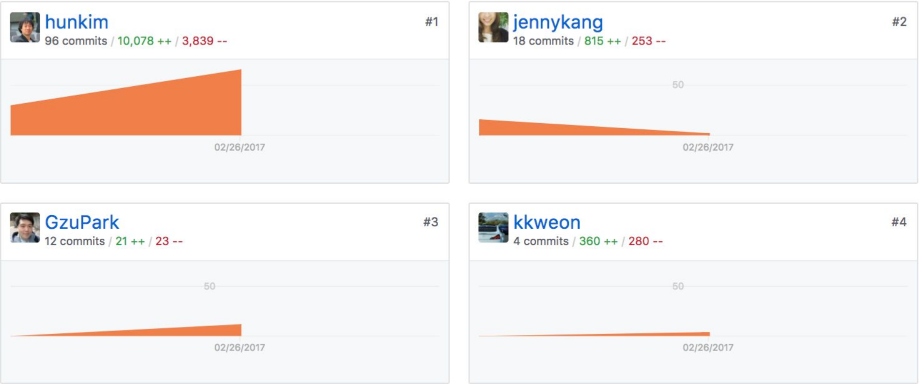
Lab 3

Minimizing Cost

Sung Kim <hunkim+ml@gmail.com>

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

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

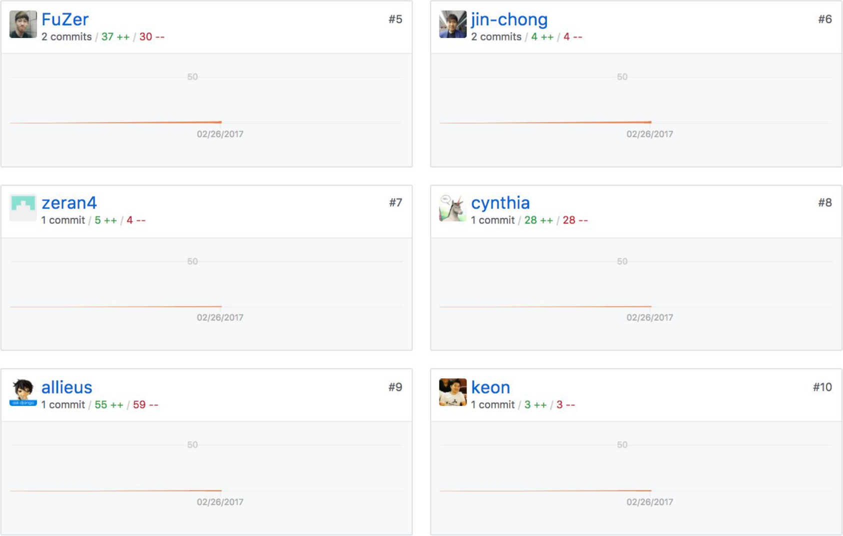


hunkim #1
96 commits / 10,078 ++ / 3,839 --

jennykang #2
18 commits / 815 ++ / 253 --

GzuPark #3
12 commits / 21 ++ / 23 --

kkweon #4
4 commits / 360 ++ / 280 --



FuZer #5
2 commits / 37 ++ / 30 --

jin-chong #6
2 commits / 4 ++ / 4 --

zeran4 #7
1 commit / 5 ++ / 4 --

cynthia #8
1 commit / 28 ++ / 28 --

allieus #9
1 commit / 55 ++ / 59 --

keon #10
1 commit / 3 ++ / 3 --

Simplified hypothesis

$$H(x) = Wx$$

$$cost(W) = \frac{1}{m} \sum_{i=1}^{m} (Wx^{(i)} - y^{(i)})^2$$

matplotlib http://matplotlib.org/users/installing.html

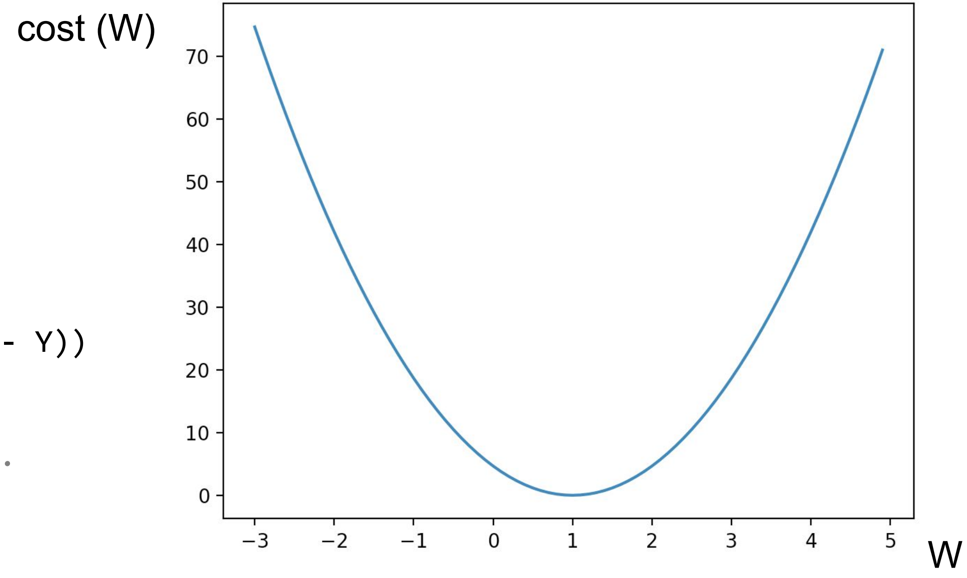
```python
import tensorflow as tf
import matplotlib.pyplot as plt
X = [1, 2, 3]
Y = [1, 2, 3]

W = tf.placeholder(tf.float32)
# Our hypothesis for linear model X * W
hypothesis = X * W

# cost/loss function
cost = tf.reduce_mean(tf.square(hypothesis - Y))
# Launch the graph in a session.
sess = tf.Session()
# Initializes global variables in the graph.
sess.run(tf.global_variables_initializer())
# Variables for plotting cost function
W_val = []
cost_val = []
for i in range(-30, 50):
   feed_W = i * 0.1
   curr_cost, curr_W = sess.run([cost, W], feed_dict={W: feed_W})
   W_val.append(curr_W)
   cost_val.append(curr_cost)

# Show the cost function
plt.plot(W_val, cost_val)
plt.show()
```

$$H(x) = Wx$$

$$cost(W) = \frac{1}{m} \sum_{i=1}^{m} (Wx^{(i)} - y^{(i)})^2$$

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-03-1-minimizing_cost_show_graph.py

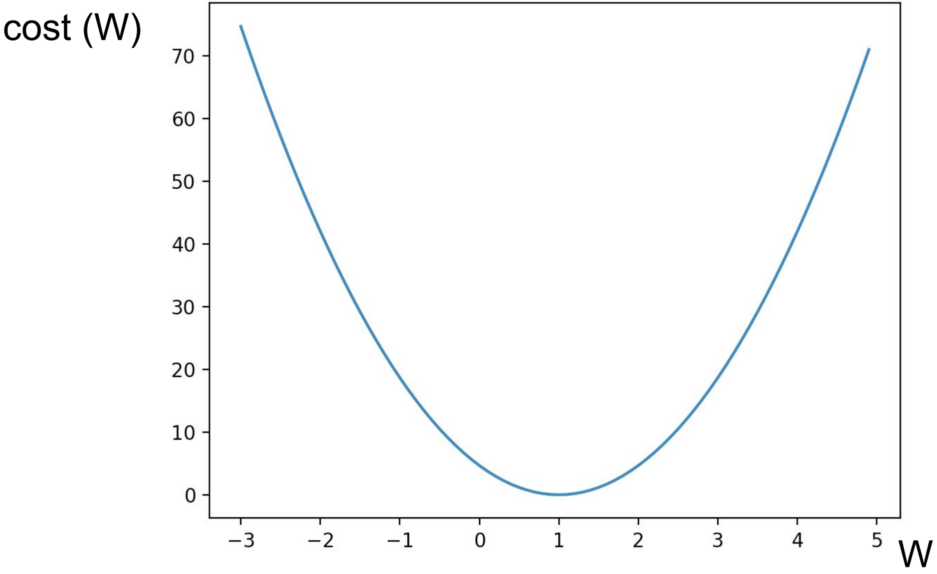
```
import tensorflow as tf
import matplotlib.pyplot as plt
X = [1, 2, 3]
Y = [1, 2, 3]

W = tf.placeholder(tf.float32)
# Our hypothesis for linear model X * W
hypothesis = X * W

# cost/loss function
cost = tf.reduce_mean(tf.square(hypothesis - Y))
# Launch the graph in a session.
sess = tf.Session()
# Initializes global variables in the graph.
sess.run(tf.global_variables_initializer())
# Variables for plotting cost function
W_val = []
cost_val = []
for i in range(-30, 50):
   feed_W = i * 0.1
   curr_cost, curr_W = sess.run([cost, W], feed_dict={W: feed_W})
   W_val.append(curr_W)
   cost_val.append(curr_cost)

# Show the cost function
plt.plot(W_val, cost_val)
plt.show()
```



$$cost(W) = \frac{1}{m} \sum_{i=1}^{m} (Wx^{(i)} - y^{(i)})^2$$

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-03-1-minimizing_cost_show_graph.py

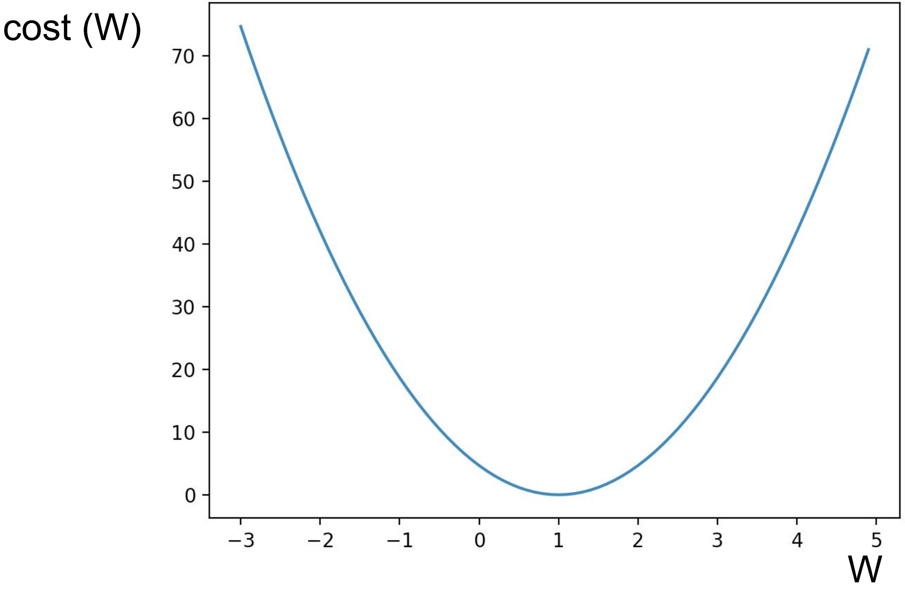
Gradient descent



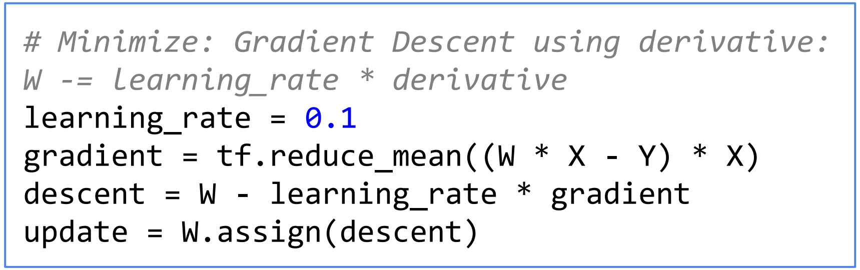
$$W := W - \alpha \frac{1}{m} \sum_{i=1}^{m} (Wx^{(i)} - y^{(i)}) x^{(i)}$$

$$cost(W) = \frac{1}{m} \sum_{i=1}^{m} (Wx^{(i)} - y^{(i)})^2$$

Gradient descent



$$W := W - \alpha \frac{1}{m} \sum_{i=1}^{m} (Wx^{(i)} - y^{(i)})x^{(i)}$$



```
# Minimize: Gradient Descent using derivative: W -= learning_rate * derivative
learning_rate = 0.1
gradient = tf.reduce_mean((W * X - Y) * X)
descent = W - learning_rate * gradient
update = W.assign(descent)
```

$$cost(W) = \frac{1}{m} \sum_{i=1}^{m} (Wx^{(i)} - y^{(i)})^2$$

```
import tensorflow as tf
x_data = [1, 2, 3]
y_data = [1, 2, 3]

W = tf.Variable(tf.random_normal([1]), name='weight')
X = tf.placeholder(tf.float32)
Y = tf.placeholder(tf.float32)

# Our hypothesis for linear model X * W
hypothesis = X * W

# cost/loss function
cost = tf.reduce_sum(tf.square(hypothesis - Y))

# Minimize: Gradient Descent using derivative: W -= learning_rate * derivative
learning_rate = 0.1
gradient = tf.reduce_mean((W * X - Y) * X)
descent = W - learning_rate * gradient
update = W.assign(descent)

# Launch the graph in a session.
sess = tf.Session()
# Initializes global variables in the graph.
sess.run(tf.global_variables_initializer())
for step in range(21):
   sess.run(update, feed_dict={X: x_data, Y: y_data})
   print(step, sess.run(cost, feed_dict={X: x_data, Y: y_data}), sess.run(W))
```

$$W := W - \alpha \frac{1}{m} \sum_{i=1}^{m} (W x^{(i)} - y^{(i)}) x^{(i)}$$

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-03-2-minimizing_cost_gradient_update.py

```python
import tensorflow as tf
x_data = [1, 2, 3]
y_data = [1, 2, 3]

W = tf.Variable(tf.random_normal([1]), name='weight')
X = tf.placeholder(tf.float32)
Y = tf.placeholder(tf.float32)

# Our hypothesis for linear model X * W
hypothesis = X * W

# cost/loss function
cost = tf.reduce_sum(tf.square(hypothesis - Y))

# Minimize: Gradient Descent using derivative: W -= learning_rate * derivative
learning_rate = 0.1
gradient = tf.reduce_mean((W * X - Y) * X)
descent = W - learning_rate * gradient
update = W.assign(descent)

# Launch the graph in a session.
sess = tf.Session()
# Initializes global variables in the graph.
sess.run(tf.global_variables_initializer())
for step in range(21):
   sess.run(update, feed_dict={X: x_data, Y: y_data})
   print(step, sess.run(cost, feed_dict={X: x_data, Y: y_data}), sess.run(W))
```

```
0 5.81756 [ 1.64462376]
1 1.65477 [ 1.34379935]
2 0.470691 [ 1.18335962]
3 0.133885 [ 1.09779179]
4 0.0380829 [ 1.05215561]
5 0.0108324 [ 1.0278163]
6 0.00308123 [ 1.01483536]
7 0.000876432 [ 1.00791216]
8 0.00024929 [ 1.00421977]
9 7.09082e-05 [ 1.00225055]
10 2.01716e-05 [ 1.00120032]
11 5.73716e-06 [ 1.00064015]
12 1.6319e-06 [ 1.00034142]
13 4.63772e-07 [ 1.00018203]
14 1.31825e-07 [ 1.00009704]
15 3.74738e-08 [ 1.00005174]
16 1.05966e-08 [ 1.00002754]
17 2.99947e-09 [ 1.00001466]
18 8.66635e-10 [ 1.00000787]
19 2.40746e-10 [ 1.00000417]
20 7.02158e-11 [ 1.00000226]
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-03-2-minimizing_cost_gradient_update.py

```
import tensorflow as tf
x_data = [1, 2, 3]
y_data = [1, 2, 3]

W = tf.Variable(tf.random_normal([1]), name='weight')
X = tf.placeholder(tf.float32)
Y = tf.placeholder(tf.float32)

# Our hypothesis for linear model X * W
hypothesis = X * W

# cost/loss function
cost = tf.reduce_sum(tf.square(hypothesis - Y))

# Minimize: Gradient Descent using derivative: W -= learning_rate * derivative
learning_rate = 0.1
gradient = tf.reduce_mean((W * X - Y) * X)
descent = W - learning_rate * gradient
update = W.assign(descent)

# Launch the graph in a session.
sess = tf.Session()
# Initializes global variables in the graph.
sess.run(tf.global_variables_initializer())
for step in range(21):
   sess.run(update, feed_dict={X: x_data, Y: y_data})
   print(step, sess.run(cost, feed_dict={X: x_data, Y: y_data}), sess.run(W))
```

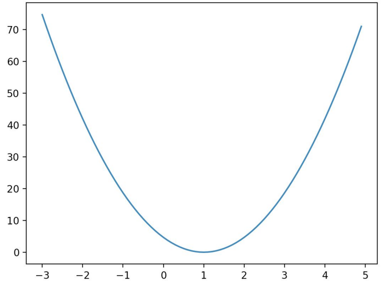
```
# Minimize: Gradient Descent Magic
optimizer =
  tf.train.GradientDescentOptimizer(learning_rate=0.1)
train = optimizer.minimize(cost)
```

$$W := W - \alpha \frac{1}{m} \sum_{i=1}^{m} (Wx^{(i)} - y^{(i)})x^{(i)}$$

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-03-2-minimizing_cost_gradient_update.py

Output when W=5

```python
import tensorflow as tf

# tf Graph Input
X = [1, 2, 3]
Y = [1, 2, 3]

# Set wrong model weights
W = tf.Variable(5.0)
# Linear model
hypothesis = X * W
# cost/loss function
cost = tf.reduce_mean(tf.square(hypothesis - Y))
# Minimize: Gradient Descent Magic
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.1)
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())

for step in range(100):
   print(step, sess.run(W))
   sess.run(train)
```

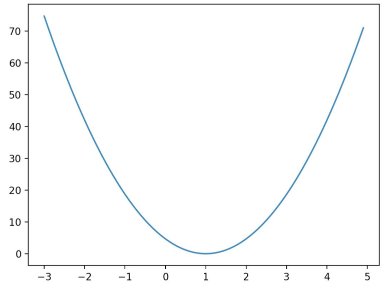


```
0 5.0
1 1.26667
2 1.01778
3 1.00119
4 1.00008
5 1.00001
6 1.0
7 1.0
8 1.0
9 1.0
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-03-3-minimizing_cost_tf_optimizer.py

Output when W=-3

```python
import tensorflow as tf

# tf Graph Input
X = [1, 2, 3]
Y = [1, 2, 3]

# Set wrong model weights
W = tf.Variable(-3.0)

# Linear model
hypothesis = X * W
# cost/loss function
cost = tf.reduce_mean(tf.square(hypothesis - Y))
# Minimize: Gradient Descent Magic
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.1)
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())

for step in range(100):
   print(step, sess.run(W))
   sess.run(train)
```



```
0 -3.0
1 0.733334
2 0.982222
3 0.998815
4 0.999921
5 0.999995
6 1.0
7 1.0
8 1.0
9 1.0
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-03-3-minimizing_cost_tf_optimizer.py

Optional: *compute_gradient* and *apply_gradient*

```python
import tensorflow as tf
X = [1, 2, 3]
Y = [1, 2, 3]
# Set wrong model weights
W = tf.Variable(5.)
# Linear model
hypothesis = X * W
# Manual gradient
gradient = tf.reduce_mean((W * X - Y) * X) * 2
# cost/loss function
cost = tf.reduce_mean(tf.square(hypothesis - Y))
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)

# Get gradients
gvs = optimizer.compute_gradients(cost, [W])
# Apply gradients
apply_gradients = optimizer.apply_gradients(gvs)

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

for step in range(100):
   print(step, sess.run([gradient, W, gvs]))
   sess.run(apply_gradients)
```

```
0 [37.333332, 5.0, [(37.333336, 5.0)]]
1 [33.848888, 4.6266665, [(33.848888, 4.6266665)]]
2 [30.689657, 4.2881775, [(30.689657, 4.2881775)]]
3 [27.825287, 3.9812808, [(27.825287, 3.9812808)]]
4 [25.228262, 3.703028, [(25.228264, 3.703028)]]
...
96 [0.0030694802, 1.0003289, [(0.0030694804, 1.0003289)]]
97 [0.0027837753, 1.0002983, [(0.0027837753, 1.0002983)]]
98 [0.0025234222, 1.0002704, [(0.0025234222, 1.0002704)]]
99 [0.0022875469, 1.0002451, [(0.0022875469, 1.0002451)]]
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-03-X-minimizing_cost_tf_gradient.py