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

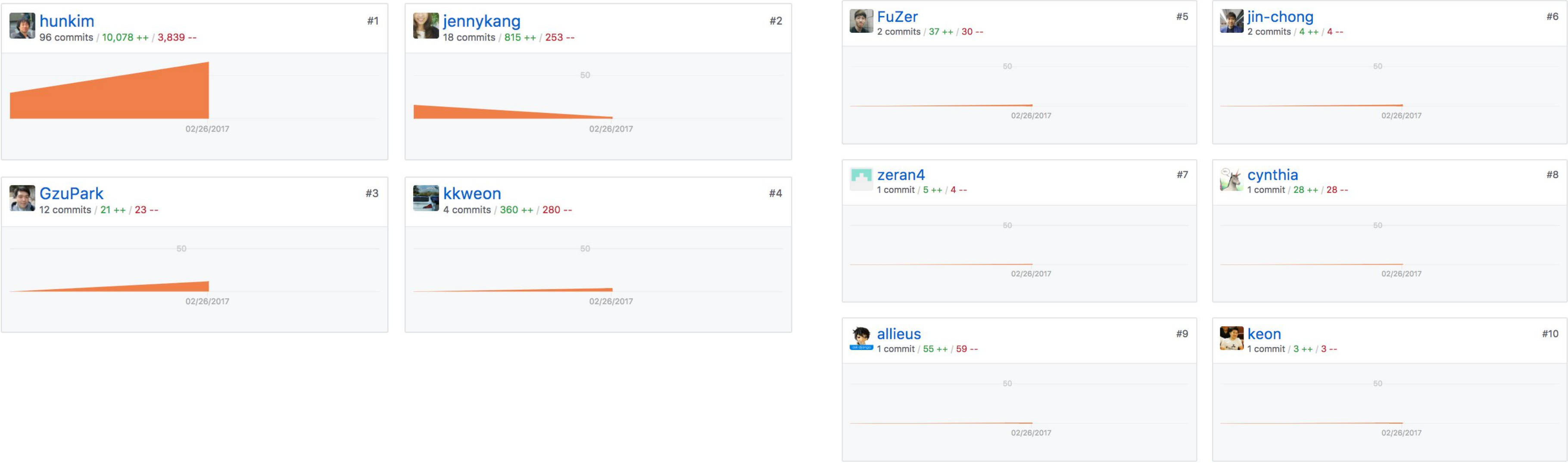
With TF 1.0!

Lab 3

Minimizing Cost

Sung Kim <hunkim+ml@gmail.com>
Code: https://github.com/hunkim/DeepLearningZeroToAll/

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



Simplified hypothesis

$$H(x) = Wx$$

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

```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()
```

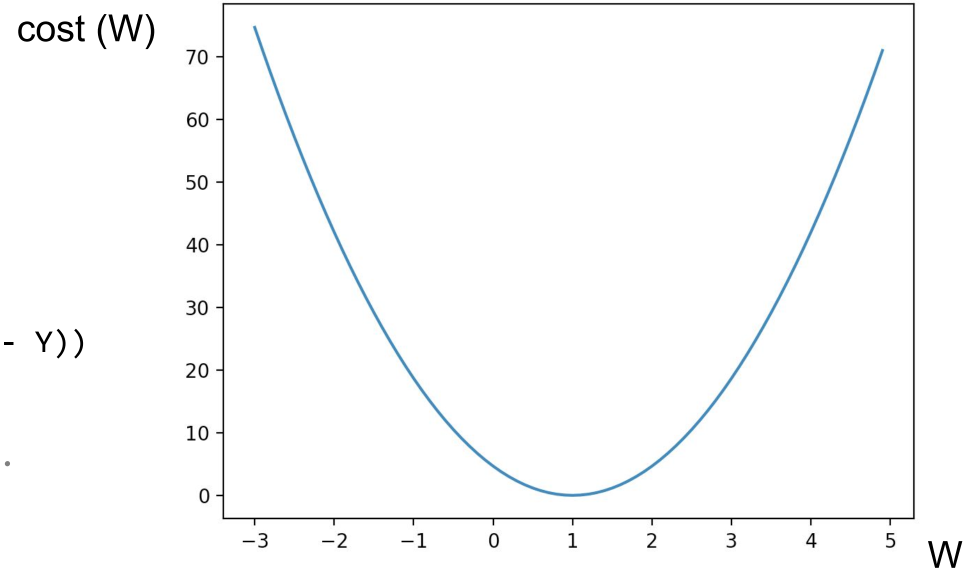
matplotlib http://matplotlib.org/users/installing.html

$$H(x) = Wx$$

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

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

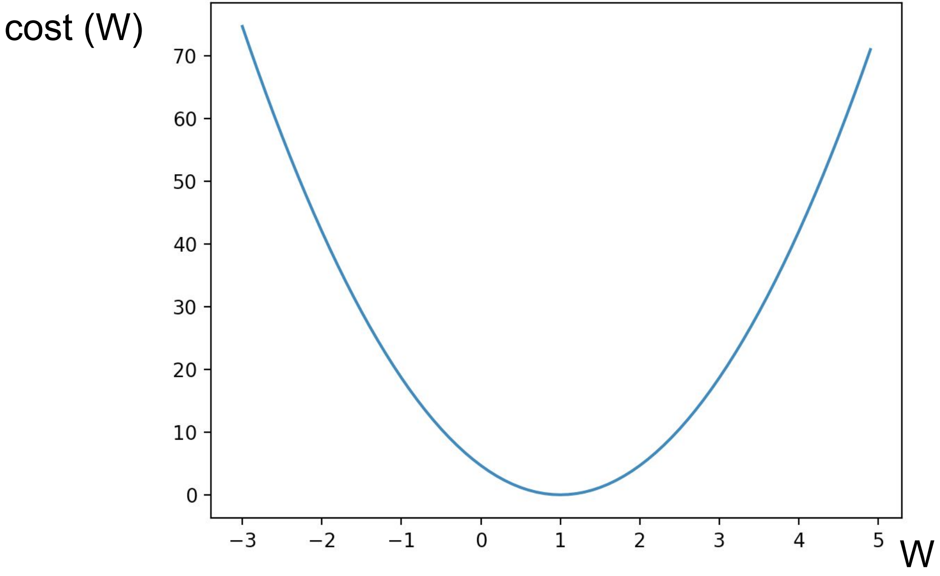
```python
import tensorflow as tf
import matplotlib.pyplot as plt
X = [1, 2, 3]
Y = [1, 2, 3]

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()
```



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

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

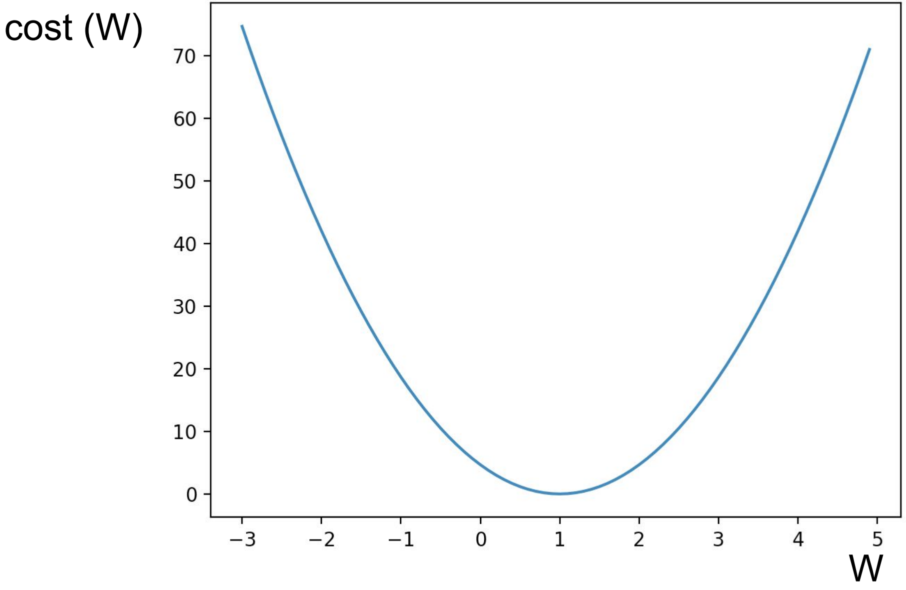
Gradient descent



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

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

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)
```

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

```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))
```

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

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

```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))
```

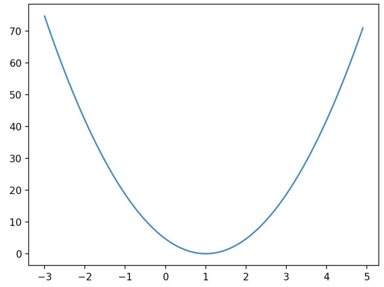
```python
# Minimize: Gradient Descent Magic
optimizer =
  tf.train.GradientDescentOptimizer(learning_rate=0.1)
train = optimizer.minimize(cost)
```

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

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

Output when W=5

```
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)
```

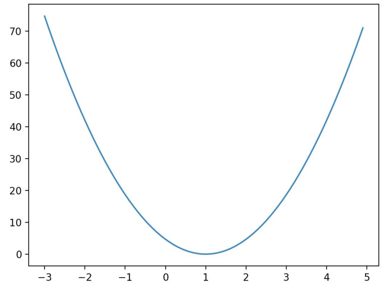



```
0 5.0
1 1.26667
2 1.01778
3 1.00119
4 1.00008
5 1.00001
6 1.0
7 1.0
8 1.0
9 1.0
```

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

```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)
```

Output when W=-3




```
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

With TF 1.0!

Lab 4

Multi-variable linear regression

Sung Kim <hunkim+ml@gmail.com>