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



With TF 1.0!



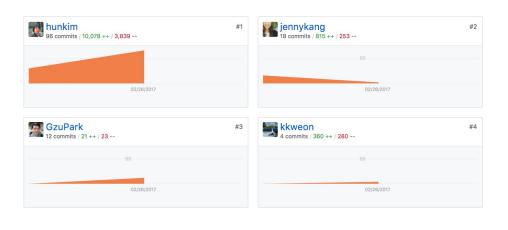
Lab 3 Minimizing Cost

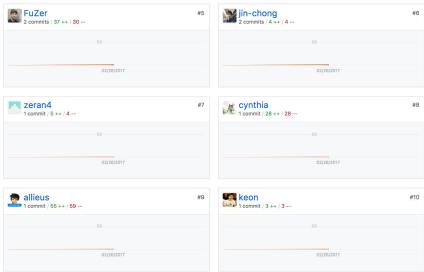
Sung Kim < hunkim+ml@gmail.com>

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



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





Simplified hypothesis

$$H(x) = Wx$$

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

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

H(x) = Wx

cost = tf.reduce_mean(tf.square(hypothesis - Y)) # Launch the graph in a session. sess = tf.Session()

curr cost, curr_W = sess.run([cost, W], feed_dict={W: feed_W})

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

cost val.append(curr cost) # Show the cost function plt.plot(W val, cost val) plt.show()

W val.append(curr W)

cost/loss function

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-03-1-minimizing cost show graph.pv

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

```
W = tf.placeholder(tf.float32)
                                                          50
# Our hypothesis for linear model X * W
hypothesis = X * W
                                                          40
# cost/loss function
                                                          30
cost = tf.reduce_mean(tf.square(hypothesis - Y))
                                                         20
# Launch the graph in a session.
sess = tf.Session()
                                                         10
# Initializes global variables in the graph.
sess.run(tf.global variables initializer())
# Variables for plotting cost function
                                                                  -2
                                                                      -1
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)
                                                            cost(W) = \frac{1}{m} \sum_{i=1}^{m} (Wx^{(i)} - y^{(i)})^2
# Show the cost function
plt.plot(W_val, cost_val)
plt.show()
                                https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-03-1-minimizing cost show graph.py
```

70

cost (W)

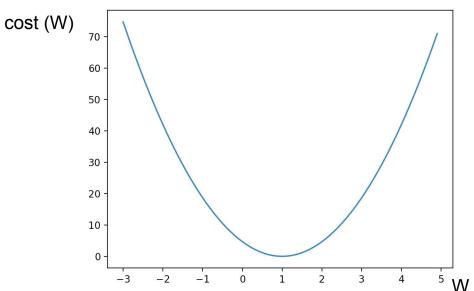
import tensorflow as tf

X = [1, 2, 3]

Y = [1, 2, 3]

import matplotlib.pyplot as plt

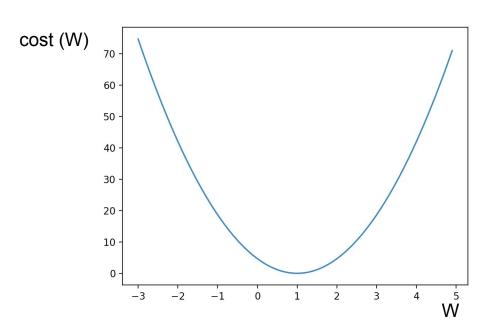
Gradient descent



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

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

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)
```

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

```
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)
                                                               W := W - \alpha \frac{1}{m} \sum_{i=1}^{m} (Wx^{(i)} - y^{(i)})x^{(i)}
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))
                                      https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-03-2-minimizing_cost_gradient_update.pv
```

 $x_{data} = [1, 2, 3]$ y data = [1, 2, 3]

```
x data = [1, 2, 3]
y data = [1, 2, 3]
                                                                                  0 5.81756 [ 1.64462376]
                                                                                   1 1.65477 [ 1.34379935]
W = tf.Variable(tf.random normal([1]), name='weight')
                                                                                  2 0.470691 [ 1.18335962]
X = tf.placeholder(tf.float32)
                                                                                   3 0.133885 [ 1.09779179]
Y = tf.placeholder(tf.float32)
                                                                                   4 0.0380829 [ 1.05215561]
# Our hypothesis for linear model X * W
                                                                                   5 0.0108324 [ 1.0278163]
hypothesis = X * W
                                                                                   6 0.00308123 [ 1.01483536]
                                                                                   7 0.000876432 [ 1.00791216]
# cost/loss function
                                                                                  8 0.00024929 [ 1.00421977]
cost = tf.reduce sum(tf.square(hypothesis - Y))
                                                                                  9 7.09082e-05 [ 1.00225055]
                                                                                  10 2.01716e-05 [ 1.00120032]
# Minimize: Gradient Descent using derivative: W -= learning rate * derivative
                                                                                   11 5.73716e-06 [ 1.00064015]
learning rate = 0.1
                                                                                   12 1.6319e-06 [ 1.00034142]
gradient = tf.reduce mean((W * X - Y) * X)
descent = W - learning rate * gradient
                                                                                   13 4.63772e-07 [ 1.00018203]
update = W.assign(descent)
                                                                                   14 1.31825e-07 [ 1.00009704]
                                                                                   15 3.74738e-08 [ 1.00005174]
# Launch the graph in a session.
                                                                                   16 1.05966e-08 [ 1.00002754]
sess = tf.Session()
                                                                                   17 2.99947e-09 [ 1.00001466]
# Initializes global variables in the graph.
                                                                                   18 8.66635e-10 [ 1.00000787]
sess.run(tf.global variables initializer())
                                                                                   19 2.40746e-10 [ 1.00000417]
for step in range(21):
                                                                                  20 7.02158e-11 [ 1.00000226]
   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))
                                    https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-03-2-minimizing cost gradient update.pv
```

```
W = tf.Variable(tf.random normal([1]), name='weight')
X = tf.placeholder(tf.float32)
Y = tf.placeholder(tf.float32)
                                            # Minimize: Gradient Descent Magic
                                            optimizer =
# Our hypothesis for linear model X * W
                                              tf.train.GradientDescentOptimizer(learning rate=0.1)
hypothesis = X * W
                                            train = optimizer.minimize(cost)
# 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)
                                                             W := W - \alpha \frac{1}{m} \sum_{i=1}^{m} (Wx^{(i)} - y^{(i)})x^{(i)}
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))
                                     ittps://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-03-2-minimizing cost gradient update.py
```

x_data = [1, 2, 3] y_data = [1, 2, 3]

```
# tf Graph Input
X = [1, 2, 3]
                                 Output when W=5
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))
                                                                           2 1.01778
# Minimize: Gradient Descent Magic
                                                                           3 1.00119
optimizer = tf.train.GradientDescentOptimizer(learning rate=0.1)
                                                                           4 1.00008
train = optimizer.minimize(cost)
                                                                           5 1.00001
```

Launch the graph in a session.

print(step, sess.run(W))

Initializes global variables in the graph.

sess.run(tf.global variables initializer())

sess = tf.Session()

for step in range(100):

```
0 5.0
1 1,26667
```

```
6 1.0
7 1.0
8 1.0
9 1.0
```

```
import tensorflow as tf
# tf Graph Input
X = [1, 2, 3]
                                 Output when W=-3
Y = [1, 2, 3]
# Set wrong model weights
W = tf.Variable(-3.0)
# Linear model
                                                                           0 - 3.0
hypothesis = X * W
                                                                            1 0.733334
# cost/loss function
                                                                            2 0.982222
cost = tf.reduce mean(tf.square(hypothesis - Y))
                                                                            3 0.998815
# Minimize: Gradient Descent Magic
                                                                            4 0.999921
optimizer = tf.train.GradientDescentOptimizer(learning rate=0.1)
                                                                            5 0.999995
train = optimizer.minimize(cost)
                                                                            6 1.0
```

Launch the graph in a session.

Initializes global variables in the graph.
sess.run(tf.global variables initializer())

sess = tf.Session()

7 1.0

8 1.0

9 1.0

```
import tensorflow as tf
X = [1, 2, 3]
                                             Optional: compute gradient
Y = [1, 2, 3]
# Set wrong model weights
                                                       and apply gradient
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
                                                        0 [37.333332, 5.0, [(37.333336, 5.0)]]
apply gradients = optimizer.apply gradients(gvs)
                                                        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)]]
# Launch the graph in a session.
                                                        4 [25.228262, 3.703028, [(25.228264, 3.703028)]]
sess = tf.Session()
sess.run(tf.global variables initializer())
                                                        96 [0.0030694802, 1.0003289, [(0.0030694804, 1.0003289)]]
                                                        97 [0.0027837753, 1.0002983, [(0.0027837753, 1.0002983)]]
for step in range(100):
                                                        98 [0.0025234222, 1.0002704, [(0.0025234222, 1.0002704)]]
   print(step, sess.run([gradient, W, gvs]))
                                                        99 [0.0022875469, 1.0002451, [(0.0022875469, 1.0002451)]]
   sess.run(apply gradients)
                                     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>