With TF 1.0!



Lab 5

Logistic (regression) classifier

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 5

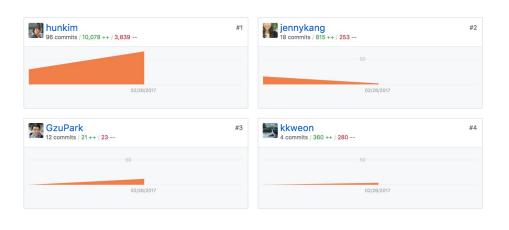
Logistic (regression) classifier

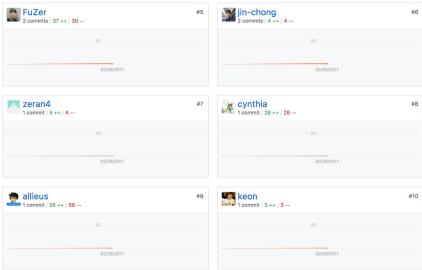
Sung Kim < hunkim+ml@gmail.com>

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



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





Logistic Regression

$$H(X) = \frac{1}{1 + e^{-W^T X}}$$

$$cost(W) = -\frac{1}{m} \sum y log(H(x)) + (1 - y)(log(1 - H(x)))$$

$$W := W - \alpha \frac{\partial}{\partial W} cost(W)$$

Training Data

```
x_data = [[1, 2], [2, 3], [3, 1], [4, 3], [5, 3], [6, 2]]
y_data = [[0], [0], [0], [1], [1], [1]]

# placeholders for a tensor that will be always fed.
X = tf.placeholder(tf.float32, shape=[None, 2])
Y = tf.placeholder(tf.float32, shape=[None, 1])
```

```
# Hypothesis using sigmoid: tf.div(1., 1. + tf.exp(tf.matmul(X, W) + b)) hypothesis = tf.sigmoid(tf.matmul(X, W) + b) H(X) = \frac{1}{1 + e^{-W^TX}}
                                        cost(W) = -\frac{1}{m} \sum y log(H(x)) + (1 - y)(log(1 - H(x)))
# cost/loss function
cost = -tf.reduce_mean(Y * tf.log(hypothesis) + (1 - Y) *
                            tf.log(1 - hypothesis))
train = tf.train.GradientDescentOptimizer(learning_rate=0.01).minimize(cost) W := W - \alpha \frac{\partial}{\partial W} cost(W)
# Accuracy computation
# True if hypothesis>0.5 else False
predicted = tf.cast(hypothesis > 0.5, dtype=tf.float32)
accuracy = tf.reduce_mean(tf.cast(tf.equal(predicted, Y), dtype=tf.float32))
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-05-1-logistic regression.py

X = tf.placeholder(tf.float32, shape=[None, 2])
Y = tf.placeholder(tf.float32, shape=[None, 1])

b = tf.Variable(tf.random normal([1]), name='bias')

W = tf.Variable(tf.random_normal([2, 1]), name='weight')

Train the model

```
# Launch graph
with tf.Session() as sess:
   # Initialize TensorFlow variables
   sess.run(tf.global variables initializer())
   for step in range(10001):
       cost val, = sess.run([cost, train], feed dict={X: x data, Y: y data})
       if step % 200 == 0:
           print(step, cost val)
   # Accuracy report
   h, c, a = sess.run([hypothesis, predicted, accuracy],
                      feed dict={X: x data, Y: y data})
   print("\nHypothesis: ", h, "\nCorrect (Y): ", c, "\nAccuracy: ", a)
```

```
x_{data} = [[1, 2], [2, 3], [3, 1], [4, 3], [5, 3], [6, 2]]
y data = [[0], [0], [0], [1], [1],
# placeholders for a tensor that will be always fed.
                                                                                                               # step, cost
X = tf.placeholder(tf.float32, shape=[None, 2])
                                                                                                               0 1.73078
Y = tf.placeholder(tf.float32, shape=[None, 1])
                                                                                                                200 0.571512
                                                                                                               400 0.507414
W = tf.Variable(tf.random normal([2, 1]), name='weight')
b = tf.Variable(tf.random normal([1]), name='bias')
                                                                                                               9600 0.154132
# Hypothesis using sigmoid: tf.div(1., 1. + tf.exp(tf.matmul(X, W)))
                                                                                                               9800 0.151778
hypothesis = tf.sigmoid(tf.matmul(X, W) + b)
                                                                                                                10000 0.149496
# cost/loss function
cost = -tf.reduce mean(Y * tf.log(hypothesis) + (1 - Y) * tf.log(1 - hypothesis))
train = tf.train.GradientDescentOptimizer(learning rate=0.01).minimize(cost)
                                                                                                               Hypothesis:
# Accuracy computation
                                                                                                                [[ 0.03074029]
# True if hypothesis>0.5 else False
                                                                                                                [ 0.15884677]
predicted = tf.cast(hypothesis > 0.5, dtype=tf.float32)
                                                                                                                [ 0.30486736]
accuracy = tf.reduce mean(tf.cast(tf.equal(predicted, Y), dtype=tf.float32))
                                                                                                                [0.78138196]
                                                                                                                [ 0.93957496]
# Launch graph
with tf.Session() as sess:
                                                                                                                [ 0.98016882]]
   # Initialize TensorFlow variables
   sess.run(tf.global variables initializer())
                                                                                                               Correct (Y):
                                                                                                               [[ 0.]
   for step in range(10001):
                                                                                                                [ 0.]
       cost val, = sess.run([cost, train], feed dict={X: x data, Y: y data})
                                                                                                                [ 0.]
       if step % 200 == 0:
                                                                                                                [1.]
           print(step, cost val)
                                                                                                                [1.]
                                                                                                               [ 1.]]
   # Accuracy report
   h, c, a = sess.run([hypothesis, predicted, accuracy],
                      feed dict={X: x data, Y: y data})
                                                                                                               Accuracy: 1.0
   print("\nHypothesis: ", h, "\nCorrect (Y): ", c, "\nAccuracy: "
                                               https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-05-1-logistic regression.py
```

Classifying diabetes



-0.411765	0.165829	0.213115	0	0	-0.23696	-0.894962	-0.7	1
-0.647059	-0.21608	-0.180328	-0.353535	-0.791962	-0.0760059	-0.854825	-0.833333	0
0.176471	0.155779	0	0	0	0.052161	-0.952178	-0.733333	1
-0.764706	0.979899	0.147541	-0.0909091	0.283688	-0.0909091	-0.931682	0.0666667	0
-0.0588235	0.256281	0.57377	0	0	0	-0.868488	0.1	0
-0.529412	0.105528	0.508197	0	0	0.120715	-0.903501	-0.7	1
0.176471	0.688442	0.213115	0	0	0.132638	-0.608027	-0.566667	0
0.176471	0.396985	0.311475	0	0	-0.19225	0.163962	0.2	1

```
xy = np.loadtxt('data-03-diabetes.csv', delimiter=',', dtype=np.float32)
x_data = xy[:, 0:-1]
y_data = xy[:, [-1]]
```

```
x data = xy[:, 0:-1]
                                                                                                                       0 0.82794
y data = xy[:, [-1]]
                                                                                                                       200 0.755181
# placeholders for a tensor that will be always fed.
                                                                                                                       400 0.726355
X = tf.placeholder(tf.float32, shape=[None, 8])
                                                                                                                       600 0.705179
Y = tf.placeholder(tf.float32, shape=[None, 1])
                                                                                                                       800 0.686631
W = tf.Variable(tf.random normal([8, 1]), name='weight')
b = tf.Variable(tf.random normal([1]), name='bias')
                                                                                                                       9600 0.492056
                                                                                                                       9800 0.491396
# Hypothesis using sigmoid: tf.div(1., 1. + tf.exp(tf.matmul(X, W)))
hypothesis = tf.sigmoid(tf.matmul(X, W) + b)
                                                                                                                       10000 0.490767
# cost/loss function
cost = -tf.reduce mean(Y * tf.log(hypothesis) + (1 - Y) * tf.log(1 - hypothesis))
train = tf.train.GradientDescentOptimizer(learning rate=0.01).minimize(cost)
                                                                                                                        [0.7461012]
# Accuracy computation
                                                                                                                        [0.79919308]
# True if hypothesis>0.5 else False
predicted = tf.cast(hypothesis > 0.5, dtype=tf.float32)
                                                                                                                        [0.72995949]
accuracy = tf.reduce mean(tf.cast(tf.equal(predicted, Y), dtype=tf.float32))
                                                                                                                        [ 0.88297188]]
# Launch graph
with tf.Session() as sess:
                                                                                                                       [ 1.]
   sess.run(tf.global variables initializer())
                                                                                                                       [ 1.]
                                                                                                                       [ 1.]]
   feed = {X: x data, Y: y data}
   for step in range(10001):
                                                                                                                       Accuracy:
       sess.run(train, feed dict=feed)
                                                                                                                       0.762846
       if step % 200 == 0:
           print(step, sess.run(cost, feed dict=feed))
   # Accuracy report
   h, c, a = sess.run([hypothesis, predicted, accuracy], feed dict=feed)
   print("\nHypothesis: ", h, "\nCorrect (Y): ", c, "\nAccuracy: ", a)
                                         https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-05-2-logistic regression diabetes.pv
```

xy = np.loadtxt('data-03-diabetes.csv', delimiter=',', dtype=np.float32)

Exercise

- CSV reading using tf.decode_csv
- Try other classification data from Kaggle
 - o https://www.kaggle.com

Lab 6 Softmax classifier

Sung Kim <hunkim+ml@gmail.com>