With TF 1.0!

Lab 7-1

Learning rate, Evaluation

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

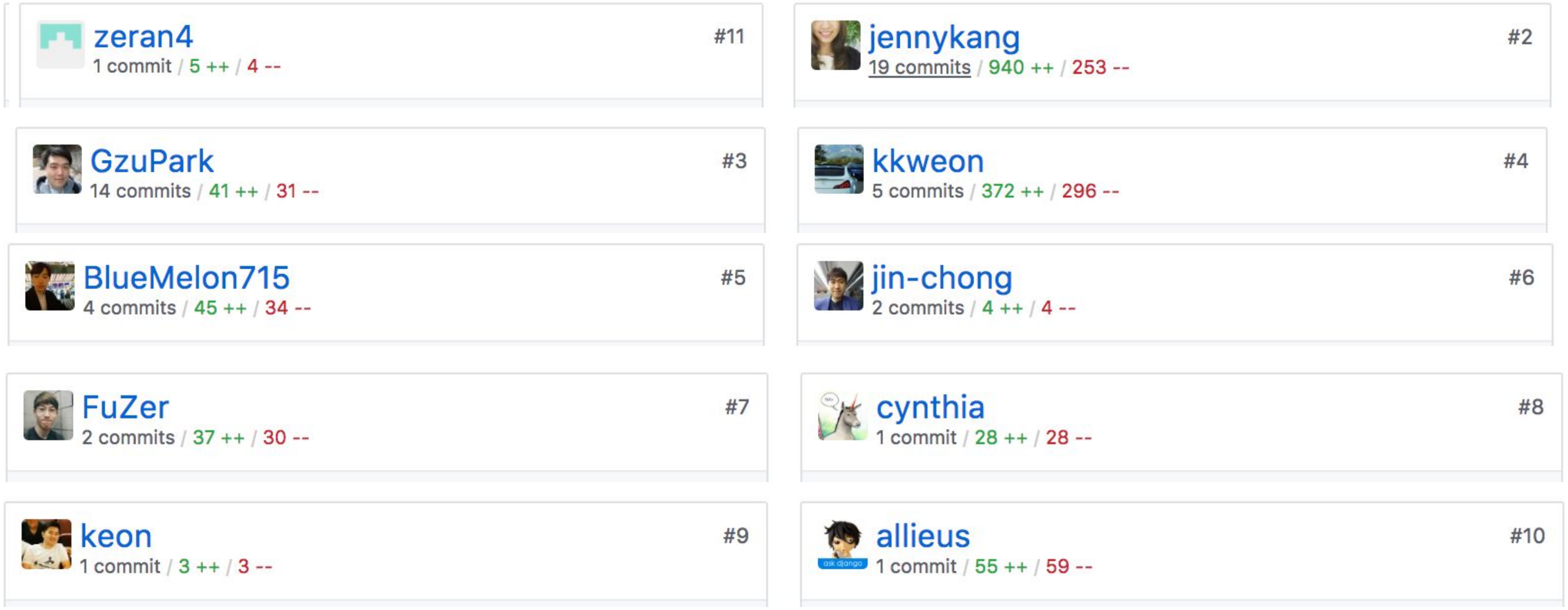
With TF 1.0!

Lab 7-1

Learning rate, Evaluation

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

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



zeran4 #11
1 commit / 5 ++ / 4 --

jennykang #2
19 commits / 940 ++ / 253 --

GzuPark #3
14 commits / 41 ++ / 31 --

kkweon #4
5 commits / 372 ++ / 296 --

BlueMelon715 #5
4 commits / 45 ++ / 34 --

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

FuZer #7
2 commits / 37 ++ / 30 --

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

keon #9
1 commit / 3 ++ / 3 --

allieus #10
1 commit / 55 ++ / 59 --

Training and Test datasets

```
x_data = [[1, 2, 1], [1, 3, 2], [1, 3, 4], [1, 5, 5], [1, 7, 5], [1, 2, 5], [1, 6, 6], [1, 7, 7]]
y_data = [[0, 0, 1], [0, 0, 1], [0, 0, 1], [0, 1, 0], [0, 1, 0], [0, 1, 0], [1, 0, 0], [1, 0, 0]]

# Evaluation our model using this test dataset
x_test = [[2, 1, 1], [3, 1, 2], [3, 3, 4]]
y_test = [[0, 0, 1], [0, 0, 1], [0, 0, 1]]
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-07-1-learning_rate_and_evaluation.py

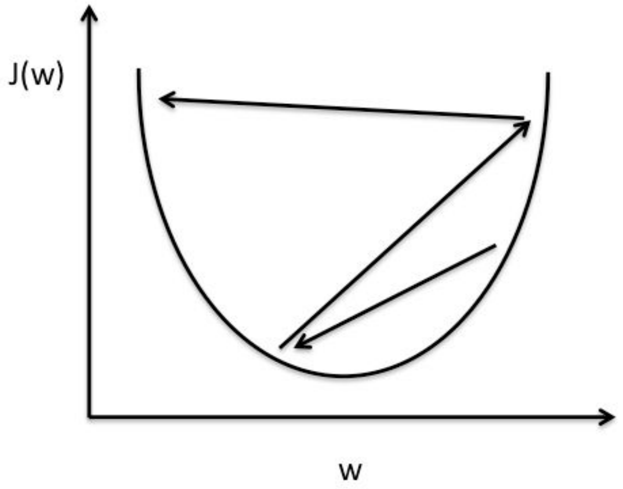
Training and Test datasets

```python
X = tf.placeholder("float", [None, 3])
Y = tf.placeholder("float", [None, 3])
W = tf.Variable(tf.random_normal([3, 3]))
b = tf.Variable(tf.random_normal([3]))

hypothesis = tf.nn.softmax(tf.matmul(X, W)+b)
cost = tf.reduce_mean(-tf.reduce_sum(Y * tf.log(hypothesis), axis=1))
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.1).minimize(cost)

# Correct prediction Test model
prediction = tf.arg_max(hypothesis, 1)
is_correct = tf.equal(prediction, tf.arg_max(Y, 1))
accuracy = tf.reduce_mean(tf.cast(is_correct, tf.float32))

# Launch graph
with tf.Session() as sess:
   # Initialize TensorFlow variables
   sess.run(tf.global_variables_initializer())
   for step in range(201):
       cost_val, W_val, _ = sess.run([cost, W, optimizer],
                       feed_dict={X: x_data, Y: y_data})
       print(step, cost_val, W_val)

   # predict
   print("Prediction:", sess.run(prediction, feed_dict={X: x_test}))
   # Calculate the accuracy
   print("Accuracy: ", sess.run(accuracy, feed_dict={X: x_test, Y: y_test}))
```

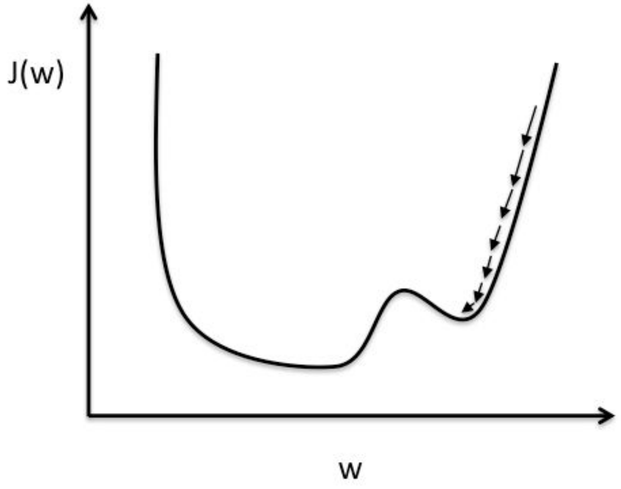
```
199 0.672261 [[-1.15377033  0.28146935
1.13632679]
[ 0.37484586  0.18958236  0.33544877]
[-0.35609841 -0.43973011 -1.25604188]]
200 0.670909 [[-1.15885413  0.28058422
1.14229572]
[ 0.37609792  0.19073224  0.33304682]
[-0.35536593 -0.44033223 -1.2561723 ]]
Prediction: [2 2 2]
Accuracy:  1.0
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-07-1-learning_rate_and_evaluation.py

Learning rate: NaN!



Large learning rate: Overshooting.



Small learning rate: Many iterations until convergence and trapping in local minima.

http://sebastianraschka.com/Articles/2015_singlelayer_neurons.html

Big learning rate

```
X = tf.placeholder("float", [None, 3])
Y = tf.placeholder("float", [None, 3])
W = tf.Variable(tf.random_normal([3, 3]))
b = tf.Variable(tf.random_normal([3]))
hypothesis = tf.nn.softmax(tf.matmul(X, W)+b)
cost = tf.reduce_mean(-tf.reduce_sum(Y * tf.log(hypothesis), axis=1))
optimizer = tf.train.GradientDescentOptimizer
                (learning_rate=1.5).minimize(cost)

# Correct prediction Test model
prediction = tf.arg_max(hypothesis, 1)
is_correct = tf.equal(prediction, tf.arg_max(Y, 1))
accuracy = tf.reduce_mean(tf.cast(is_correct, tf.float32))

# Launch graph
with tf.Session() as sess:
   # Initialize TensorFlow variables
   sess.run(tf.global_variables_initializer())
   for step in range(201):
       cost_val, W_val, _ = sess.run([cost, W, optimizer],
                       feed_dict={X: x_data, Y: y_data})
       print(step, cost_val, W_val)

   # predict
   print("Prediction:", sess.run(prediction, feed_dict={X: x_test}))
   # Calculate the accuracy
   print("Accuracy: ", sess.run(accuracy, feed_dict={X: x_test, Y: y_test}))
```

```
2 27.2798 [[ 0.44451016  0.85699677
-1.03748143]
[ 0.48429942  0.98872018 -0.57314301]
[ 1.52989244  1.16229868 -4.74406147]]
3 8.668 [[ 0.12396193  0.61504567
-0.47498202]
[ 0.22003263 -0.2470119   0.9268558 ]
[ 0.96035379  0.41933775 -3.43156195]]
4 5.77111 [[-0.9524312   1.13037777
0.08607888]
[-3.78651619  2.26245379  2.42393875]
[-3.07170963  3.14037919 -2.12054014]]
5 inf [[ nan  nan  nan]
[ nan  nan  nan]
[ nan  nan  nan]]
6 nan [[ nan  nan  nan]
[ nan  nan  nan]
[ nan  nan  nan]]


...
Prediction: [0 0 0]
Accuracy:  0.0
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-07-1-learning_rate_and_evaluation.py

Small learning rate

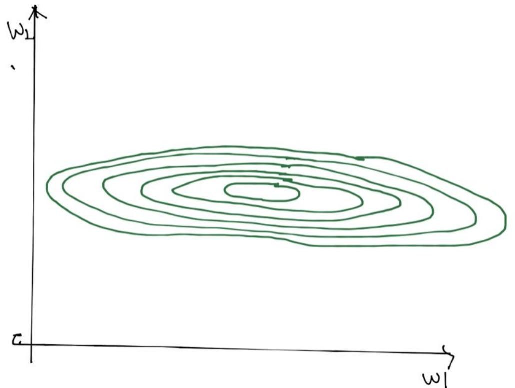
```python
X = tf.placeholder("float", [None, 3])
Y = tf.placeholder("float", [None, 3])
W = tf.Variable(tf.random_normal([3, 3]))
b = tf.Variable(tf.random_normal([3]))
hypothesis = tf.nn.softmax(tf.matmul(X, W)+b)
cost = tf.reduce_mean(-tf.reduce_sum(Y * tf.log(hypothesi
optimizer = tf.train.GradientDescentOptimizer
                (learning_rate=1e-10).minimize(co

# Correct prediction Test model
prediction = tf.arg_max(hypothesis, 1)
is_correct = tf.equal(prediction, tf.arg_max(Y, 1))
accuracy = tf.reduce_mean(tf.cast(is_correct, tf.float32)

# Launch graph
with tf.Session() as sess:
   # Initialize TensorFlow variables
   sess.run(tf.global_variables_initializer())
   for step in range(201):
       cost_val, W_val, _ = sess.run([cost, W, optimizer]
                       feed_dict={X: x_data, Y: y_data})
       print(step, cost_val, W_val)

   # predict
   print("Prediction:", sess.run(prediction, feed_dict={X
   # Calculate the accuracy
   print("Accuracy: ", sess.run(accuracy, feed_dict={X: x_test, Y: y_test}))
```

```
0 5.73203 [[ 0.80269563  0.67861295 -1.21728313]
[-0.3051686  -0.3032113   1.50825703]
[ 0.75722361 -0.7008909  -2.10820389]]
1 5.73203 [[ 0.80269563  0.67861295 -1.21728313]
[-0.3051686  -0.3032113   1.50825703]
[ 0.75722361 -0.7008909  -2.10820389]]
…

198 5.73203 [[ 0.80269563  0.67861295 -1.21728313]
[-0.3051686  -0.3032113   1.50825703]
[ 0.75722361 -0.7008909  -2.10820389]]
199 5.73203 [[ 0.80269563  0.67861295 -1.21728313]
[-0.3051686  -0.3032113   1.50825703]
[ 0.75722361 -0.7008909  -2.10820389]]
200 5.73203 [[ 0.80269563  0.67861295 -1.21728313]
[-0.3051686  -0.3032113   1.50825703]
[ 0.75722361 -0.7008909  -2.10820389]]
Prediction: [0 0 0]
Accuracy:  0.0
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-07-1-learning_rate_and_evaluation.py

Non-normalized inputs

```
xy = np.array([[828.659973, 833.450012, 908100, 828.349976, 831.659973],
               [823.02002, 828.070007, 1828100, 821.655029, 828.070007],
               [819.929993, 824.400024, 1438100, 818.97998, 824.159973],
               [816, 820.958984, 1008100, 815.48999, 819.23999],
               [819.359985, 823, 1188100, 818.469971, 818.97998],
               [819, 823, 1198100, 816, 820.450012],
               [811.700012, 815.25, 1098100, 809.780029, 813.669983],
               [809.51001, 816.659973, 1398100, 804.539978, 809.559998]])
```



https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-07-2-linear_regression_without_min_max.py

Non-normalized inputs

```python
xy=...
x_data = xy[:, 0:-1]
y_data = xy[:, [-1]]
# placeholders for a tensor that will be always fed.
X = tf.placeholder(tf.float32, shape=[None, 4])
Y = tf.placeholder(tf.float32, shape=[None, 1])
W = tf.Variable(tf.random_normal([4, 1]), name='weight')
b = tf.Variable(tf.random_normal([1]), name='bias')

hypothesis = tf.matmul(X, W) + b
cost = tf.reduce_mean(tf.square(hypothesis - Y))
# Minimize
optimizer = tf.train.GradientDescentOptimizer(learning_rate=1e-5)
train = optimizer.minimize(cost)

sess = tf.Session()
sess.run(tf.global_variables_initializer())
for step in range(2001):
   cost_val, hy_val, _ = sess.run(
       [cost, hypothesis, train], feed_dict={X: x_data, Y: y_data})
   print(step, "Cost: ", cost_val, "\nPrediction:\n", hy_val)
```

```
5 Cost:  inf
Prediction:
[[ inf]
[ inf]
[ inf]
...
6 Cost:  nan
Prediction:
[[ nan]
[ nan]
[ nan]
...
```

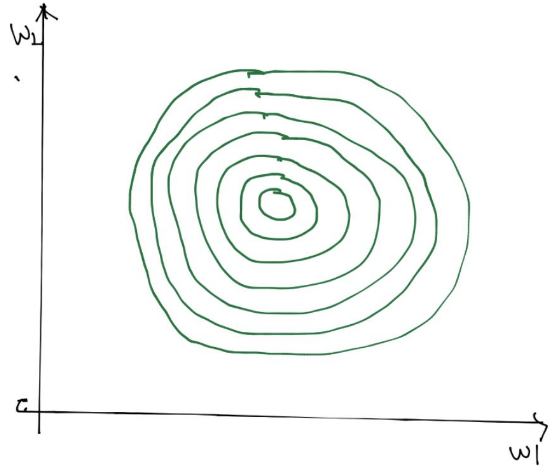
https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-07-2-linear_regression_without_min_max.py

Normalized inputs (min-max scale)

```
xy = np.array([[828.659973, 833.450012, 908100, 828.349976, 831.659973],
               [823.02002, 828.070007, 1828100, 821.655029, 828.070007],
               [819.929993, 824.400024, 1438100, 818.97998, 824.159973],
               [816, 820.958984, 1008100, 815.48999, 819.23999],
               [819.359985, 823, 1188100, 818.469971, 818.97998],
               [819, 823, 1198100, 816, 820.450012],
               [811.700012, 815.25, 1098100, 809.780029, 813.669983],
               [809.51001, 816.659973, 1398100, 804.539978, 809.559998]])
```

```
xy = MinMaxScaler(xy)
print(xy)
```

```
[[ 0.99999999  0.99999999  0.          1.          1.        ]
 [ 0.70548491  0.70439552  1.          0.71881782  0.83755791]
 [ 0.54412549  0.50274824  0.57608696  0.606468    0.6606331 ]
 [ 0.33890353  0.31368023  0.10869565  0.45989134  0.43800918]
 [ 0.51436     0.42582389  0.30434783  0.58504805  0.42624401]
 [ 0.49556179  0.42582389  0.31521739  0.48131134  0.49276137]
 [ 0.11436064  0.          0.20652174  0.22007776  0.18597238]
 [ 0.          0.07747099  0.5326087   0.          0.        ]]
```



https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-07-3-linear_regression_min_max.py

Normalized inputs

```
xy=...
x_data = xy[:, 0:-1]
y_data = xy[:, [-1]]
# placeholders for a tensor that will be always fed.
X = tf.placeholder(tf.float32, shape=[None, 4])
Y = tf.placeholder(tf.float32, shape=[None, 1])
W = tf.Variable(tf.random_normal([4, 1]), name='weight')
b = tf.Variable(tf.random_normal([1]), name='bias')

hypothesis = tf.matmul(X, W) + b
cost = tf.reduce_mean(tf.square(hypothesis - Y))
# Minimize
optimizer = tf.train.GradientDescentOptimizer(learning_rate=1e-5)
train = optimizer.minimize(cost)

sess = tf.Session()
sess.run(tf.global_variables_initializer())
for step in range(2001):
   cost_val, hy_val, _ = sess.run(
       [cost, hypothesis, train], feed_dict={X: x_data, Y: y_data})
   print(step, "Cost: ", cost_val, "\nPrediction:\n", hy_val)
```

```
Prediction:
[[ 1.63450289]
 [ 0.06628087]
 [ 0.35014752]
 [ 0.67070574]
 [ 0.61131608]
 [ 0.61466062]
 [ 0.23175186]
 [-0.13716528]]
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-07-3-linear_regression_min_max.py

With TF 1.0!

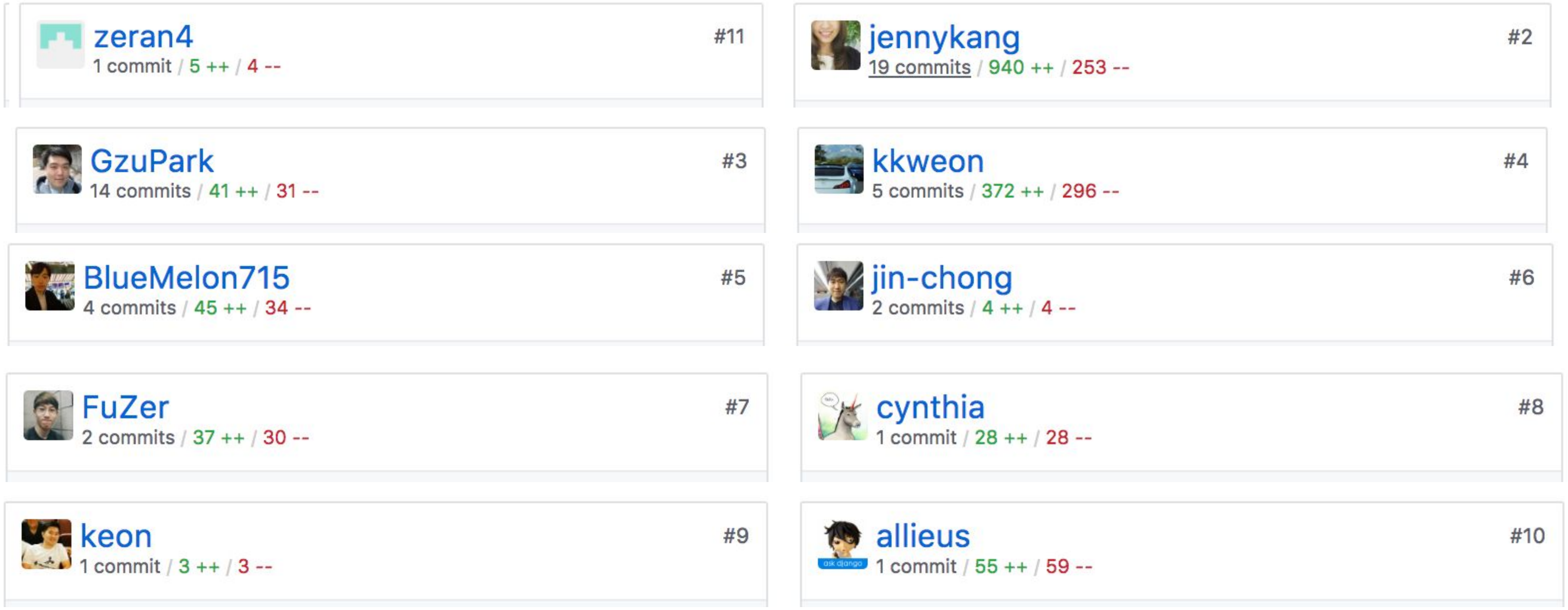
Lab 7-2

MNIST data

Sung Kim <hunkim+ml@gmail.com>

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

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



zeran4 #11
1 commit / 5 ++ / 4 --

jennykang #2
19 commits / 940 ++ / 253 --

GzuPark #3
14 commits / 41 ++ / 31 --

kkweon #4
5 commits / 372 ++ / 296 --

BlueMelon715 #5
4 commits / 45 ++ / 34 --

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

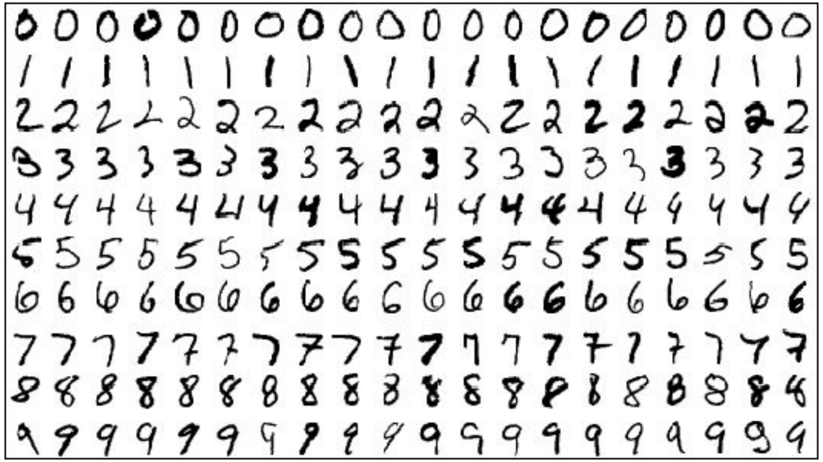
FuZer #7
2 commits / 37 ++ / 30 --

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

keon #9
1 commit / 3 ++ / 3 --

allieus #10
1 commit / 55 ++ / 59 --

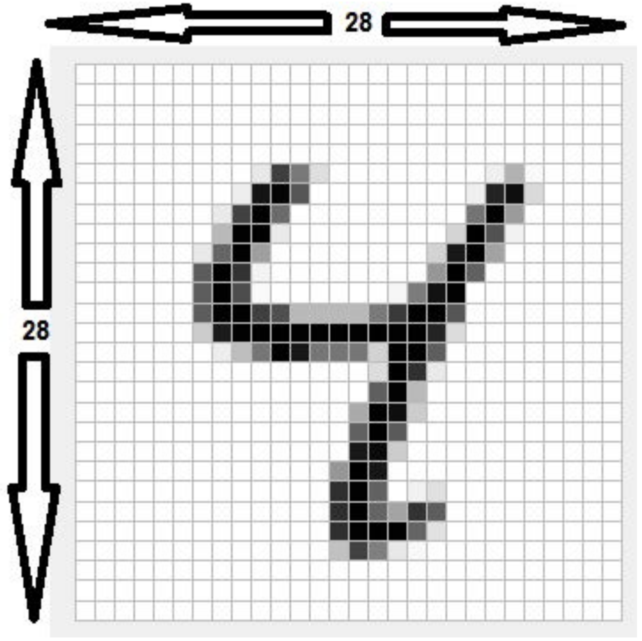
MNIST Dataset



train-images-idx3-ubyte.gz:	training set images (9912422 bytes)
train-labels-idx1-ubyte.gz:	training set labels (28881 bytes)
t10k-images-idx3-ubyte.gz:	test set images (1648877 bytes)
t10k-labels-idx1-ubyte.gz:	test set labels (4542 bytes)

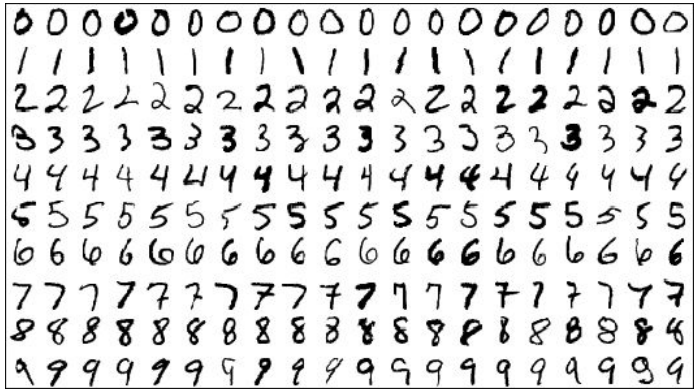
http://yann.lecun.com/exdb/mnist/

28x28x1 image



```
# MNIST data image of shape 28 * 28 = 784
X = tf.placeholder(tf.float32, [None, 784])
# 0 - 9 digits recognition = 10 classes
Y = tf.placeholder(tf.float32, [None, nb_classes])
```

http://derindelimavi.blogspot.hk/2015/04/mnist-el-yazs-rakam-veri-seti.html



MNIST Dataset

```
from tensorflow.examples.tutorials.mnist import input_data
# Check out https://www.tensorflow.org/get_started/mnist/beginners for
# more information about the mnist dataset
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
...
batch_xs, batch_ys = mnist.train.next_batch(100)
...
print("Accuracy: ", accuracy.eval(session=sess,
           feed_dict={X: mnist.test.images, Y: mnist.test.labels}))
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-07-4-mnist_introduction.py

Reading data and set variables

```
from tensorflow.examples.tutorials.mnist import input_data
# Check out https://www.tensorflow.org/get_started/mnist/beginners for
# more information about the mnist dataset
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)

nb_classes = 10

# MNIST data image of shape 28 * 28 = 784
X = tf.placeholder(tf.float32, [None, 784])
# 0 - 9 digits recognition = 10 classes
Y = tf.placeholder(tf.float32, [None, nb_classes])

W = tf.Variable(tf.random_normal([784, nb_classes]))
b = tf.Variable(tf.random_normal([nb_classes]))
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-07-4-mnist_introduction.py

Softmax!

```
# Hypothesis (using softmax)
hypothesis = tf.nn.softmax(tf.matmul(X, W) + b)

cost = tf.reduce_mean(-tf.reduce_sum(Y * tf.log(hypothesis), axis=1))
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.1).minimize(cost)

# Test model
is_correct = tf.equal(tf.arg_max(hypothesis, 1), tf.arg_max(Y, 1))
# Calculate accuracy
accuracy = tf.reduce_mean(tf.cast(is_correct, tf.float32))
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-07-4-mnist_introduction.py

Training epoch/batch

```
# parameters
training_epochs = 15
batch_size = 100

with tf.Session() as sess:
   # Initialize TensorFlow variables
   sess.run(tf.global_variables_initializer())
   # Training cycle
   for epoch in range(training_epochs):
       avg_cost = 0
       total_batch = int(mnist.train.num_examples / batch_size)

       for i in range(total_batch):
           batch_xs, batch_ys = mnist.train.next_batch(batch_size)
           c, _ = sess.run([cost, optimizer], feed_dict={X: batch_xs, Y: batch_ys})
           avg_cost += c / total_batch

       print('Epoch:', '%04d' % (epoch + 1), 'cost =', '{:.9f}'.format(avg_cost))
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-07-4-mnist_introduction.py

Training epoch/batch

In the neural network terminology:

- one **epoch** = one forward pass and one backward pass of *all* the training examples
- **batch size** = the number of training examples in one forward/backward pass. The higher the batch size, the more memory space you'll need.
- number of **iterations** = number of passes, each pass using [batch size] number of examples. To be clear, one pass = one forward pass + one backward pass (we do not count the forward pass and backward pass as two different passes).

Example: if you have *1000 training examples*, and your *batch size is 500*, then it will take 2 iterations to complete 1 epoch.

http://stackoverflow.com/questions/4752626/epoch-vs-iteration-when-training-neural-networks

Training epoch/batch

```
# parameters
training_epochs = 15
batch_size = 100

with tf.Session() as sess:
   # Initialize TensorFlow variables
   sess.run(tf.global_variables_initializer())
   # Training cycle
   for epoch in range(training_epochs):
       avg_cost = 0
       total_batch = int(mnist.train.num_examples / batch_size)

       for i in range(total_batch):
           batch_xs, batch_ys = mnist.train.next_batch(batch_size)
           c, _ = sess.run([cost, optimizer], feed_dict={X: batch_xs, Y: batch_ys})
           avg_cost += c / total_batch

       print('Epoch:', '%04d' % (epoch + 1), 'cost =', '{:.9f}'.format(avg_cost))
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-07-4-mnist_introduction.py

Report results on test dataset

```
# Test the model using test sets
print("Accuracy: ", accuracy.eval(session=sess,
      feed_dict={X: mnist.test.images, Y: mnist.test.labels}))
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-07-4-mnist_introduction.py

```python
hypothesis = tf.nn.softmax(tf.matmul(X, W) + b)
cost = tf.reduce_mean(-tf.reduce_sum(Y * tf.log(hypothesis), axis=1))
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.1).minimize(cost)

is_correct = tf.equal(tf.arg_max(hypothesis, 1), tf.arg_max(Y, 1))
accuracy = tf.reduce_mean(tf.cast(is_correct, tf.float32))

# parameters
training_epochs = 15
batch_size = 100

with tf.Session() as sess:
   # Initialize TensorFlow variables
   sess.run(tf.global_variables_initializer())
   # Training cycle
   for epoch in range(training_epochs):
       avg_cost = 0
       total_batch = int(mnist.train.num_examples / batch_size)

       for i in range(total_batch):
           batch_xs, batch_ys = mnist.train.next_batch(batch_size)
           c, _ = sess.run([cost, optimizer],
                           feed_dict={X: batch_xs, Y: batch_ys})
           avg_cost += c / total_batch

       print('Epoch:', '%04d' % (epoch + 1),
                       'cost =', '{:.9f}'.format(avg_cost))
```

```
Epoch: 0001 cost = 2.868104637
Epoch: 0002 cost = 1.134684615
Epoch: 0003 cost = 0.908220728
Epoch: 0004 cost = 0.794199896
Epoch: 0005 cost = 0.721815854
Epoch: 0006 cost = 0.670184430
Epoch: 0007 cost = 0.630576546
Epoch: 0008 cost = 0.598888191
Epoch: 0009 cost = 0.573027079
Epoch: 0010 cost = 0.550497213
Epoch: 0011 cost = 0.532001859
Epoch: 0012 cost = 0.515517795
Epoch: 0013 cost = 0.501175288
Epoch: 0014 cost = 0.488425370
Epoch: 0015 cost = 0.476968593
Learning finished
Accuracy:  0.888
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-07-4-mnist_introduction.py

Sample image show and prediction

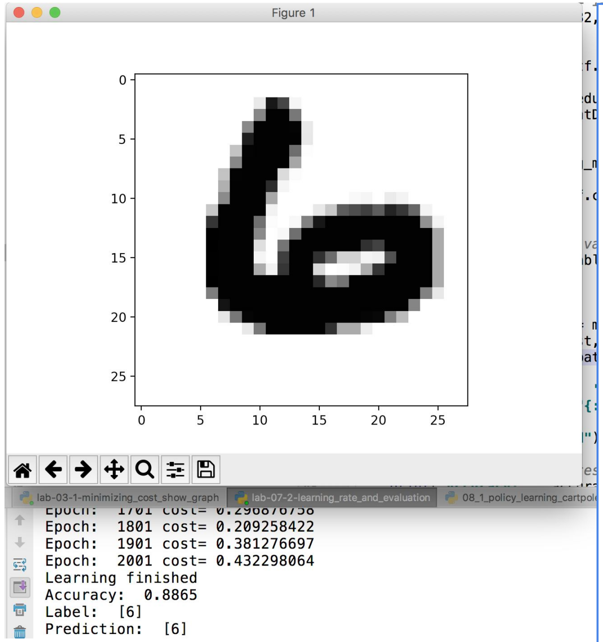


Figure 1
0
5
10
15
20
25
0
5
10
15
20
25
lab-03-1-minimizing_cost_show_graph
lab-07-2-learning_rate_and_evaluation
08_1_policy_learning_cartpole

```
Epoch:  1801 cost= 0.209258422
Epoch:  1901 cost= 0.381276697
Epoch:  2001 cost= 0.432298064
Learning finished
Accuracy:  0.8865
Label:  [6]
Prediction:  [6]
```

```python
import matplotlib.pyplot as plt
import random

# Get one and predict
r = random.randint(0, mnist.test.num_examples - 1)
print("Label:", sess.run(tf.argmax(mnist.test.labels[r:r+1], 1)))
print("Prediction:", sess.run(tf.argmax(hypothesis, 1),
                              feed_dict={X: mnist.test.images[r:r + 1]}))

plt.imshow(mnist.test.images[r:r + 1].
          reshape(28, 28), cmap='Greys', interpolation='nearest')
plt.show()
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-07-4-mnist_introduction.py

Lab 8

Tensor Manipulation

Sung Kim <hunkim+ml@gmail.com>