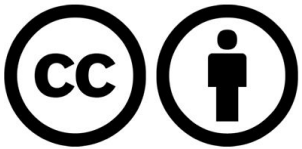
With TF 1.0!

Lab 4

Multi-variable 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

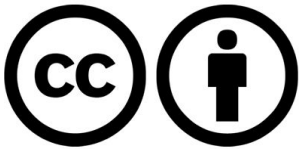
With TF 1.0!

Lab 4-1

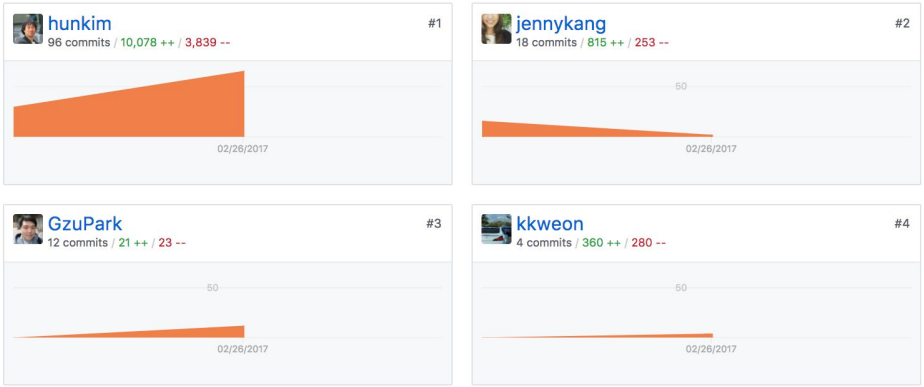
Multi-variable linear regression

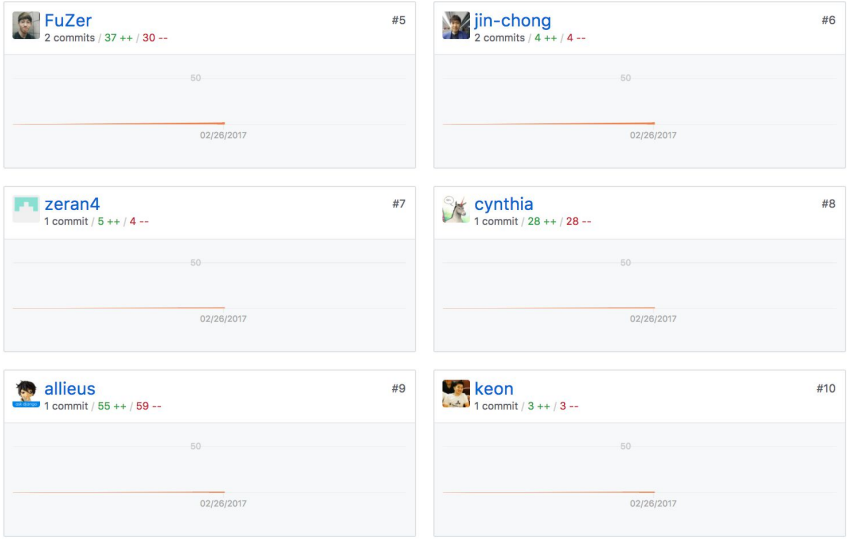
Sung Kim <hunkim+ml@gmail.com>

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



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





Hypothesis using matrix

$$H(x_1, x_2, x_3) = x_1w_1 + x_2w_2 + x_3w_3$$

x_1	x_2	x_3	Y
73	80	75	152
93	88	93	185
89	91	90	180
96	98	100	196
73	66	70	142

Test Scores for General Psychology

Hypothesis using matrix

$$H(x_1, x_2, x_3) = x_1 w_1 + x_2 w_2 + x_3 w_3$$

x_1	x_2	x_3	Y
73	80	75	152
93	88	93	185
89	91	90	180
96	98	100	196
73	66	70	142

Test Scores for General Psychology

```
x1_data = [73., 93., 89., 96., 73.]
x2_data = [80., 88., 91., 98., 66.]
x3_data = [75., 93., 90., 100., 70.]
y_data = [152., 185., 180., 196., 142.]

# placeholders for a tensor that will be always fed.
x1 = tf.placeholder(tf.float32)
x2 = tf.placeholder(tf.float32)
x3 = tf.placeholder(tf.float32)

Y = tf.placeholder(tf.float32)

w1 = tf.Variable(tf.random_normal([1]), name='weight1')
w2 = tf.Variable(tf.random_normal([1]), name='weight2')
w3 = tf.Variable(tf.random_normal([1]), name='weight3')
b = tf.Variable(tf.random_normal([1]), name='bias')

hypothesis = x1 * w1 + x2 * w2 + x3 * w3 + b
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-04-1-multi_variable_linear_regression.py

```python
import tensorflow as tf
x1_data = [73., 93., 89., 96., 73.]
x2_data = [80., 88., 91., 98., 66.]
x3_data = [75., 93., 90., 100., 70.]
y_data = [152., 185., 180., 196., 142.]

# placeholders for a tensor that will be always fed.
x1 = tf.placeholder(tf.float32)
x2 = tf.placeholder(tf.float32)
x3 = tf.placeholder(tf.float32)
Y = tf.placeholder(tf.float32)

w1 = tf.Variable(tf.random_normal([1]), name='weight1')
w2 = tf.Variable(tf.random_normal([1]), name='weight2')
w3 = tf.Variable(tf.random_normal([1]), name='weight3')
b = tf.Variable(tf.random_normal([1]), name='bias')
hypothesis = x1 * w1 + x2 * w2 + x3 * w3 + b

# cost/loss function
cost = tf.reduce_mean(tf.square(hypothesis - Y))
# Minimize. Need a very small learning rate for this data set
optimizer = tf.train.GradientDescentOptimizer(learning_rate=1e-5)
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(2001):
    cost_val, hy_val, _ = sess.run([cost, hypothesis, train],
                                   feed_dict={x1: x1_data, x2: x2_data, x3: x3_data, Y: y_data})
    if step % 10 == 0:
        print(step, "Cost: ", cost_val, "\nPrediction:\n", hy_val)
```

```
0 Cost:  19614.8
Prediction:
[ 21.69748688
39.10213089  31.82624626
35.14236832
32.55316544]
10 Cost:  14.0682
Prediction:
[ 145.56100464
187.94958496
178.50236511
194.86721802
146.08096313]

...

1990 Cost:  4.9197
Prediction:
[ 148.15084839
186.88632202
179.6293335
195.81796265
144.46044922]
2000 Cost:  4.89449
Prediction:
[ 148.15931702
186.8805542
179.63194275
195.81971741
144.45298767]
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-04-1-multi_variable_linear_regression.py

Matrix

$$\begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} x_1 w_1 + x_2 w_2 + x_3 w_3 \end{pmatrix} \qquad H(X) = XW$$

Matrix

$$\begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} x_1 w_1 + x_2 w_2 + x_3 w_3 \end{pmatrix} \qquad H(X) = XW$$

```
x_data = [[73., 80., 75.], [93., 88., 93.],
          [89., 91., 90.], [96., 98., 100.], [73., 66., 70.]]
y_data = [[152.], [185.], [180.], [196.], [142.]]


# placeholders for a tensor that will be always fed.
X = tf.placeholder(tf.float32, shape=[None, 3])
Y = tf.placeholder(tf.float32, shape=[None, 1])

W = tf.Variable(tf.random_normal([3, 1]), name='weight')
b = tf.Variable(tf.random_normal([1]), name='bias')

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

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-04-2-multi_variable_matmul_linear_regression.py

```python
import tensorflow as tf
x_data = [[73., 80., 75.], [93., 88., 93.],
          [89., 91., 90.], [96., 98., 100.], [73., 66., 70.]]
y_data = [[152.], [185.], [180.], [196.], [142.]]
# placeholders for a tensor that will be always fed.
X = tf.placeholder(tf.float32, shape=[None, 3])
Y = tf.placeholder(tf.float32, shape=[None, 1])

W = tf.Variable(tf.random_normal([3, 1]), name='weight')
b = tf.Variable(tf.random_normal([1]), name='bias')

# Hypothesis
hypothesis = tf.matmul(X, W) + b
# Simplified cost/loss function
cost = tf.reduce_mean(tf.square(hypothesis - Y))
# Minimize
optimizer = tf.train.GradientDescentOptimizer(learning_rate=1e-5)
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(2001):
   cost_val, hy_val, _ = sess.run(
       [cost, hypothesis, train], feed_dict={X: x_data, Y: y_data})
   if step % 10 == 0:
       print(step, "Cost: ", cost_val, "\nPrediction:\n", hy_val)
```

```
0 Cost:  7105.46
Prediction:
[[ 80.82241058]
 [ 92.26364136]
 [ 93.70250702]
 [ 98.09217834]
 [ 72.51759338]]
10 Cost:  5.89726
Prediction:
[[ 155.35159302]
 [ 181.85691833]
 [ 181.97254944]
 [ 194.21760559]
 [ 140.85707092]]

...

1990 Cost:  3.18588
Prediction:
[[ 154.36352539]
 [ 182.94833374]
 [ 181.85189819]
 [ 194.35585022]
 [ 142.03240967]]
2000 Cost:  3.1781
Prediction:
[[ 154.35881042]
 [ 182.95147705]
 [ 181.85035706]
 [ 194.35533142]
 [ 142.036026 ]]
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-04-2-multi_variable_matmul_linear_regression.py

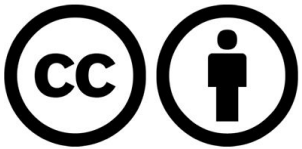
With TF 1.0!

Lab 4-2

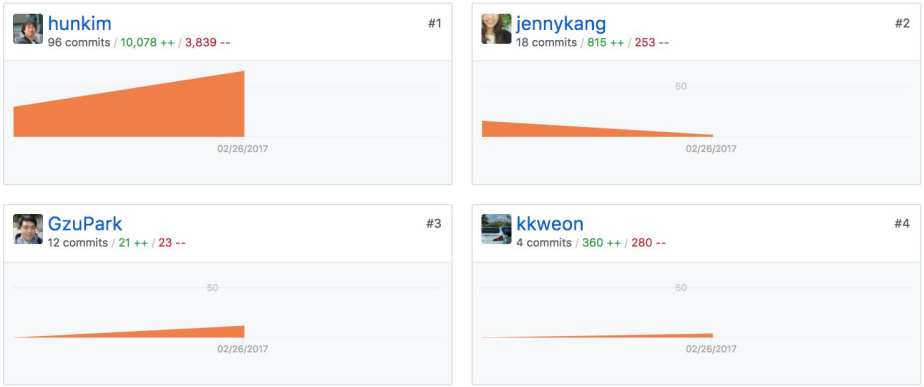
Loading Data from File

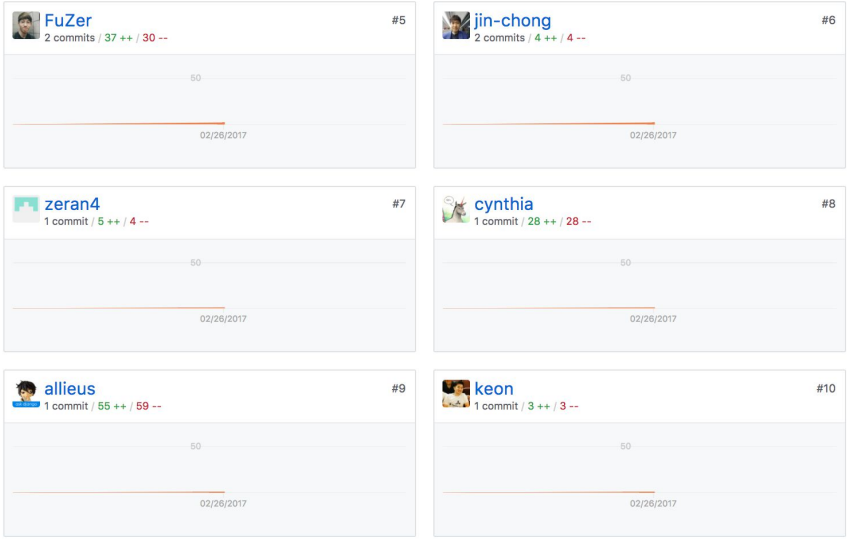
Sung Kim <hunkim+ml@gmail.com>

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



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





Loading data from file

data-01-test-score.csv

```
# EXAM1,EXAM2,EXAM3,FINAL
73,80,75,152
93,88,93,185
89,91,90,180
96,98,100,196
73,66,70,142
53,46,55,101
```

```python
import numpy as np

xy = np.loadtxt('data-01-test-score.csv', delimiter=',', dtype=np.float32)
x_data = xy[:, 0:-1]
y_data = xy[:, [-1]]

# Make sure the shape and data are OK
print(x_data.shape, x_data, len(x_data))
print(y_data.shape, y_data)
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-04-3-file_input_linear_regression.py

Slicing

```
nums = range(5)    # range is a built-in function that creates a list of integers
print nums         # Prints "[0, 1, 2, 3, 4]"
print nums[2:4]    # Get a slice from index 2 to 4 (exclusive); prints "[2, 3]"
print nums[2:]     # Get a slice from index 2 to the end; prints "[2, 3, 4]"
print nums[:2]     # Get a slice from the start to index 2 (exclusive); prints "[0, 1]"
print nums[:]      # Get a slice of the whole list; prints ["0, 1, 2, 3, 4]"
print nums[:-1]    # Slice indices can be negative; prints ["0, 1, 2, 3]"
nums[2:4] = [8, 9] # Assign a new sublist to a slice
print nums         # Prints "[0, 1, 8, 9, 4]"
```

http://cs231n.github.io/python-numpy-tutorial/

Indexing, Slicing, Iterating

- Arrays can be indexed, sliced, iterated much like lists and other sequence types in Python
- As with Python lists, slicing in NumPy can be accomplished with the colon (:) syntax
- Colon instances (:) can be replaced with dots (...)

```
a = np.array([1, 2, 3, 4, 5])
# array([1, 2, 3, 4, 5])

a[1:3]
# array([2, 3])

a[-1]
# 5

a[0:2] = 9

a
# array([9, 9, 3, 4, 5])
```

```
b = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])
# array([[ 1,  2,  3,  4],
#        [ 5,  6,  7,  8],
#        [ 9, 10, 11, 12]])

b[:, 1]
# array([ 2,  6, 10])

b[-1]
# array([ 9, 10, 11, 12])

b[-1, :]
# array([ 9, 10, 11, 12])

b[-1, ...]
# array([ 9, 10, 11, 12])

b[0:2, :]
# array([[1, 2, 3, 4],
#        [5, 6, 7, 8]])
```

Loading data from file

data-01-test-score.csv

```
# EXAM1,EXAM2,EXAM3,FINAL
73,80,75,152
93,88,93,185
89,91,90,180
96,98,100,196
73,66,70,142
53,46,55,101
```

```python
import numpy as np

xy = np.loadtxt('data-01-test-score.csv', delimiter=',', dtype=np.float32)
x_data = xy[:, 0:-1]
y_data = xy[:, [-1]]

# Make sure the shape and data are OK
print(x_data.shape, x_data, len(x_data))
print(y_data.shape, y_data)
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-04-3-file_input_linear_regression.py

```
import tensorflow as tf
import numpy as np
tf.set_random_seed(777)  # for reproducibility

xy = np.loadtxt('data-01-test-score.csv', delimiter=',',
dtype=np.float32)
x_data = xy[:, 0:-1]
y_data = xy[:, [-1]]

# Make sure the shape and data are OK
print(x_data.shape, x_data, len(x_data))
print(y_data.shape, y_data)

# placeholders for a tensor that will be always fed.
X = tf.placeholder(tf.float32, shape=[None, 3])
Y = tf.placeholder(tf.float32, shape=[None, 1])

W = tf.Variable(tf.random_normal([3, 1]), name='weight')
b = tf.Variable(tf.random_normal([1]), name='bias')

# Hypothesis
hypothesis = tf.matmul(X, W) + b

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

# Minimize
optimizer = tf.train.GradientDescentOptimizer(learning_rate=1e-5)
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())
# Set up feed_dict variables inside the loop.
for step in range(2001):
   cost_val, hy_val, _ = sess.run(
       [cost, hypothesis, train],
       feed_dict={X: x_data, Y: y_data})
   if step % 10 == 0:
       print(step, "Cost: ", cost_val,
                   "\nPrediction:\n", hy_val)


# Ask my score
print("Your score will be ", sess.run(hypothesis,
           feed_dict={X: [[100, 70, 101]]}))

print("Other scores will be ", sess.run(hypothesis,
           feed_dict={X: [[60, 70, 110], [90, 100, 80]]}))
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-04-3-file_input_linear_regression.py

Output

```
# Launch the graph in a session.
sess = tf.Session()
# Initializes global variables in the graph.
sess.run(tf.global_variables_initializer())
# Set up feed_dict variables inside the loop.
for step in range(2001):
   cost_val, hy_val, _ = sess.run(
       [cost, hypothesis, train],
       feed_dict={X: x_data, Y: y_data})
   if step % 10 == 0:
       print(step, "Cost: ", cost_val,
                 "\nPrediction:\n", hy_val)
```

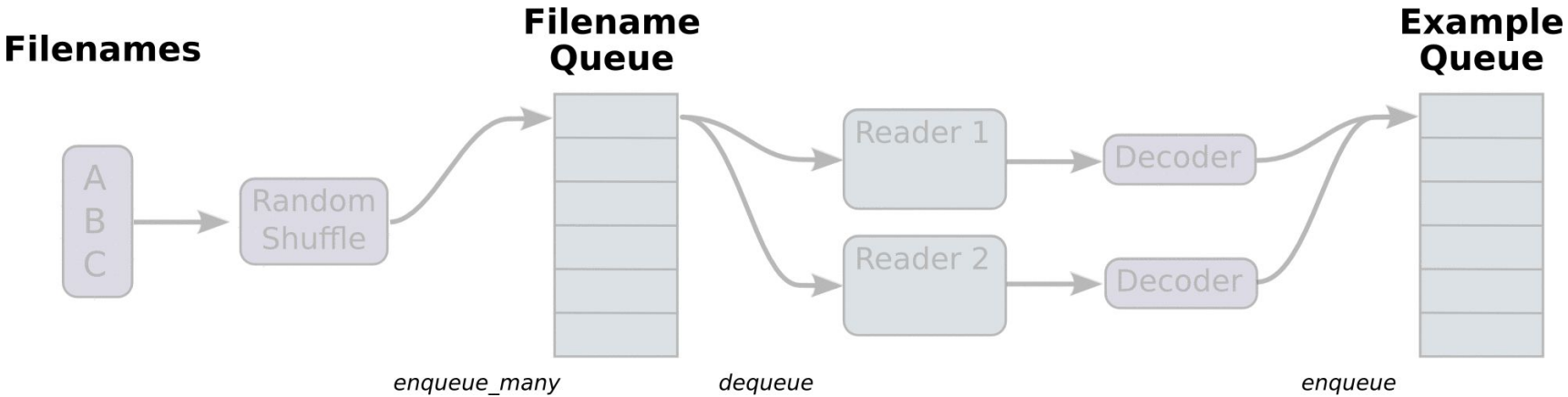
```
Your score will be  [[ 181.73277283]]
Other scores will be  [[ 145.86265564] [ 187.23129272]]
```

```
# Ask my score
print("Your score will be ", sess.run(hypothesis,
          feed_dict={X: [[100, 70, 101]]}))

print("Other scores will be ", sess.run(hypothesis,
          feed_dict={X: [[60, 70, 110], [90, 100, 80]]}))
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-04-3-file_input_linear_regression.py

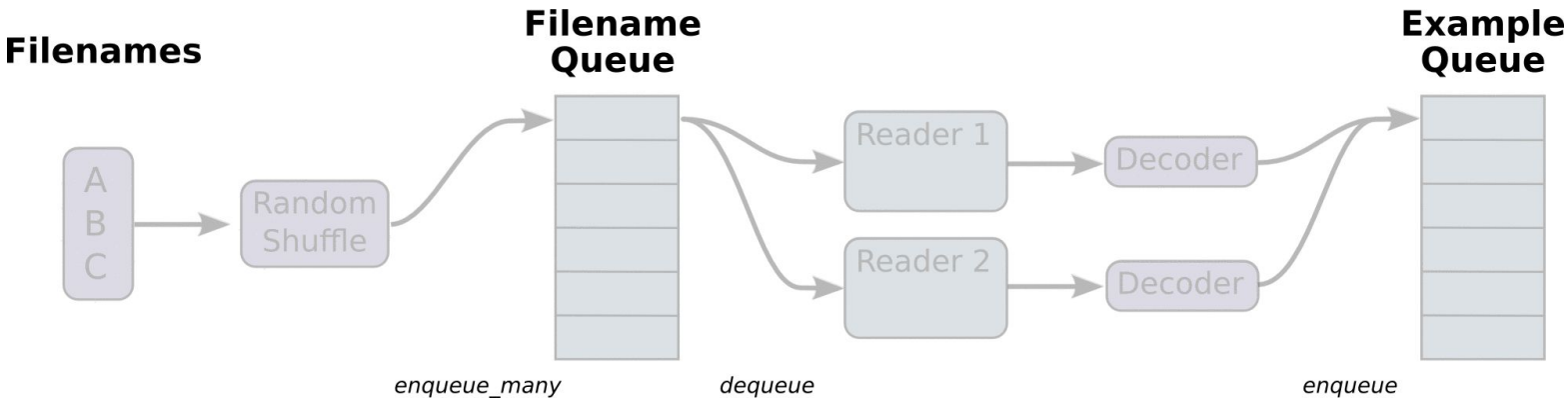
Queue Runners



https://www.tensorflow.org/programmers_guide/reading_data

1
```
filename_queue = tf.train.string_input_producer(
    ['data-01-test-score.csv', 'data-02-test-score.csv', ... ],
    shuffle=False, name='filename_queue')
```

3
```
record_defaults = [[0.], [0.], [0.], [0.]]
xy = tf.decode_csv(value, record_defaults=record_defaults)
```



2
```
reader = tf.TextLineReader()
key, value = reader.read(filename_queue)
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-04-4-tf_reader_linear_regression.py

tf.train.batch

```
# collect batches of csv in
train_x_batch, train_y_batch = \
   tf.train.batch([xy[0:-1], xy[-1:]], batch_size=10)

sess = tf.Session()
...

# Start populating the filename queue.
coord = tf.train.Coordinator()
threads = tf.train.start_queue_runners(sess=sess, coord=coord)

for step in range(2001):
   x_batch, y_batch = sess.run([train_x_batch, train_y_batch])
   ...

coord.request_stop()
coord.join(threads)
```

https://www.tensorflow.org/programmers_guide/reading_data

```python
import tensorflow as tf
filename_queue = tf.train.string_input_producer(
    ['data-01-test-score.csv'], shuffle=False, name='filename_queue')

reader = tf.TextLineReader()
key, value = reader.read(filename_queue)

# Default values, in case of empty columns. Also specifies the type of the
# decoded result.
record_defaults = [[0.], [0.], [0.], [0.]]
xy = tf.decode_csv(value, record_defaults=record_defaults)

# collect batches of csv in
train_x_batch, train_y_batch = \
    tf.train.batch([xy[0:-1], xy[-1:]], batch_size=10)

# placeholders for a tensor that will be always fed.
X = tf.placeholder(tf.float32, shape=[None, 3])
Y = tf.placeholder(tf.float32, shape=[None, 1])

W = tf.Variable(tf.random_normal([3, 1]), name='weight')
b = tf.Variable(tf.random_normal([1]), name='bias')

# Hypothesis
hypothesis = tf.matmul(X, W) + b

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

# Minimize
optimizer = tf.train.GradientDescentOptimizer(learning_rate=1e-5)
train = optimizer.minimize(cost)
```

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

# Start populating the filename queue.
coord = tf.train.Coordinator()
threads = tf.train.start_queue_runners(sess=sess, coord=coord)

for step in range(2001):
    x_batch, y_batch = sess.run([train_x_batch, train_y_batch])
    cost_val, hy_val, _ = sess.run(
        [cost, hypothesis, train],
        feed_dict={X: x_batch, Y: y_batch})
    if step % 10 == 0:
        print(step, "Cost: ", cost_val,
                    "\nPrediction:\n", hy_val)

coord.request_stop()
coord.join(threads)
```

https://github.com/hunkim/DeepLearningZeroToAll/blob/master/lab-04-4-tf_reader_linear_regression.py

shuffle_batch

```
# min_after_dequeue defines how big a buffer we will randomly sample
#   from -- bigger means better shuffling but slower start up and more
#   memory used.
# capacity must be larger than min_after_dequeue and the amount larger
#   determines the maximum we will prefetch.  Recommendation:
#   min_after_dequeue + (num_threads + a small safety margin) * batch_size
min_after_dequeue = 10000
capacity = min_after_dequeue + 3 * batch_size
example_batch, label_batch = tf.train.shuffle_batch(
   [example, label], batch_size=batch_size, capacity=capacity,
   min_after_dequeue=min_after_dequeue)
```

https://www.tensorflow.org/programmers_guide/reading_data

Lab 5

Logistic (regression) classifier

Sung Kim <hunkim+ml@gmail.com>