Lab 9-1
NN for XOR

Sung Kim <hunkim+ml@gmail.com>

Data set

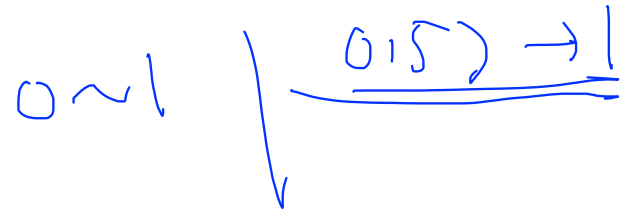
```
# XOR
# x1 x2   y
0    0      0
0    1      1
1    0      1
1    1      0
```

XOR with logistic regression?

```
# http://docs.scipy.org/doc/numpy-1.10.0/reference/generated/numpy.loadtxt.html
xy = np.loadtxt('train.txt', unpack=True)
x_data = xy[0:-1]
y_data = xy[-1]

X = tf.placeholder(tf.float32)
Y = tf.placeholder(tf.float32)

W = tf.Variable(tf.random_uniform([1,len(x_data)], -1.0, 1.0))

# Our hypothesis
h = tf.matmul(W, X)
hypothesis = tf.div(1., 1.+tf.exp(-h))
# cost function
cost = -tf.reduce_mean(Y*tf.log(hypothesis) + (1-Y)*tf.log(1-hypothesis))

# Minimize
a = tf.Variable(0.01) # Learning rate, alpha
optimizer = tf.train.GradientDescentOptimizer(a)
train = optimizer.minimize(cost)

# Before starting, initialize the variables.  We will 'run' this first.
init = tf.initialize_all_variables()
```

XOR with logistic regression?

```
# Launch the graph.
with tf.Session() as sess:
    sess.run(init)

    # Fit the line.
    for step in xrange(1000):
        sess.run(train, feed_dict={X:x_data, Y:y_data})
        if step % 200 == 0:
            print step, sess.run(cost, feed_dict={X:x_data, Y:y_data}), sess.run(W)

    # Test model
    correct_prediction = tf.equal(tf.floor(hypothesis+0.5), Y)
    # Calculate accuracy
    accuracy = tf.reduce_mean(tf.cast(correct_prediction, "float"))
    print sess.run([hypothesis, tf.floor(hypothesis+0.5), correct_prediction, accuracy], feed_dict={X:x_data, Y:y_data})
    print "Accuracy:", accuracy.eval({X:x_data, Y:y_data})
```



Does not work!

```
print sess.run([hypothesis, tf.floor(hypothesis+0.5), correct_prediction, accuracy], feed_dict={X:x_data, Y:y_data})
print "Accuracy:", accuracy.eval({X:x_data, Y:y_data})
```

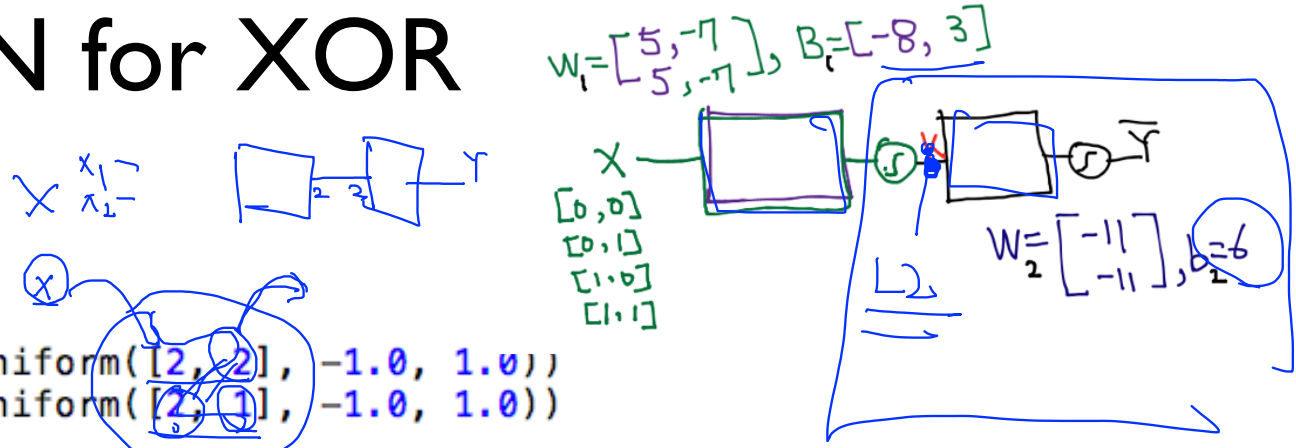
```
0 0.707739 [[-0.5306738   0.10871095]]
200 0.702072 [[-0.42843163  0.13686776]]
400 0.698917 [[-0.35024348  0.14920615]]
600 0.697075 [[-0.28977284  0.15132181]]
800 0.695938 [[-0.24237214  0.14709207]]
[array([[ 0.5       ,  0.53474092,  0.44896561,  0.48359016]], dtype=float32), array([[ 1.,  1.,  0.,  0.]]
Accuracy: 0.5
```

NN

```
# http://docs.scipy.org/doc/numpy-1.10.0/reference/generated/numpy.loadtxt.html
xy = np.loadtxt('train.txt', unpack=True)
x_data = xy[0:-1]
y_data = xy[-1]

X = tf.placeholder(tf.float32)
Y = tf.placeholder(tf.float32)

W = tf.Variable(tf.random_uniform([1,len(x_data)], -1.0, 1.0))

# Our hypothesis
h = tf.matmul(W, X)
hypothesis = tf.div(1., 1.+tf.exp(-h))
# cost function
cost = -tf.reduce_mean(Y*tf.log(hypothesis) + (1-Y)*tf.log(1-hypothesis))

# Minimize
a = tf.Variable(0.01) # Learning rate, alpha
optimizer = tf.train.GradientDescentOptimizer(a)
train = optimizer.minimize(cost)

# Before starting, initialize the variables.  We will 'run' this first.
init = tf.initialize_all_variables()
```

```
W1 = tf.Variable(tf.random_uniform([2, 2], -1.0, 1.0))
W2 = tf.Variable(tf.random_uniform([2, 1], -1.0, 1.0))

b1 = tf.Variable(tf.zeros([2]), name="Bias1")
b2 = tf.Variable(tf.zeros([1]), name="Bias2")

# Our hypothesis
L2 = tf.sigmoid(tf.matmul(X, W1) + b1)
hypothesis = tf.sigmoid(tf.matmul(L2, W2) + b2)
```

NN for XOR



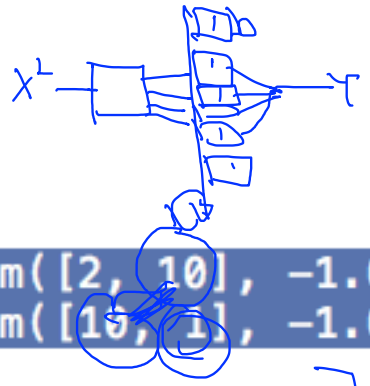
```
W1 = tf.Variable(tf.random_uniform([2, 2], -1.0, 1.0))
W2 = tf.Variable(tf.random_uniform([2, 1], -1.0, 1.0))

b1 = tf.Variable(tf.zeros([2]), name="Bias1")
b2 = tf.Variable(tf.zeros([1]), name="Bias2")

# Our hypothesis
L2 = tf.sigmoid(tf.matmul(X, W1) + b1)
hypothesis = tf.sigmoid(tf.matmul(L2, W2) + b2)
```

NN for XOR

```
198000 cost 0.00878375
    predict [[ 0.01067145]
 [ 0.99207062]
 [ 0.99207932]
 [ 0.00845705]]
    W1, B1 [array([[ 6.84843874,  5.02159739],
       [ 6.83831406,  5.0197401 ]], dtype=float32), array([-5.00321722], dtype=float32)]
    W2, B2 [array([[ 10.83042145],
       [-11.59721661]], dtype=float32), array([-5.00321722], dtype=float32)]
[array([[ 0.],
       [ 1.],
       [ 1.],
       [ 0.]], dtype=float32), array([[ True],
       [ True],
       [ True],
       [ True]], dtype=bool)]
Accuracy: 1.0
```

XOR

Wide NN for XOR

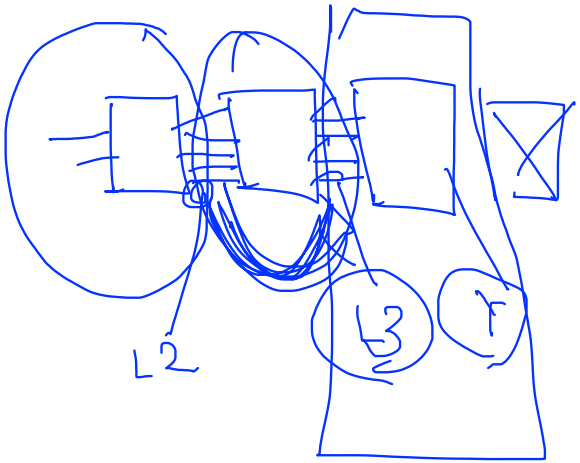


```
W1 = tf.Variable(tf.random_uniform([2, 10], -1.0, 1.0))
W2 = tf.Variable(tf.random_uniform([10, 1], -1.0, 1.0))

b1 = tf.Variable(tf.zeros([10]), name="Bias1")
b2 = tf.Variable(tf.zeros([1]), name="Bias2")

# Our hypothesis
L2 = tf.sigmoid(tf.matmul(X, W1) + b1)
hypothesis = tf.sigmoid(tf.matmul(L2, W2) + b2)
```

Wide NN for XOR

```
198000 cost: 0.00357563
    predict [[ 0.00214934]
 [ 0.99605185]
 [ 0.99672902]
 [ 0.00490645]]
    W1, B1 [array([[-1.64474547,  0.8943826 ,  0.28936383, -0.56383854,  3.85756803,
       -2.07212377,  1.19432867, -1.34555447,  6.40112543,  6.01214457],
       [-1.12384748,  0.70554543,  1.48841226, -0.80537623, -5.49986124,
         4.09881163,  0.32355428, -0.79951024, -4.84555626,  6.11010885]], dtype=float32), array([-0.45977819], dtype=float32)]
    W2, B2 [array([[ 2.93114686],
       [-1.35010254],
       [-1.59123349],
       [ 1.30858362],
       [ 7.05565166],
       [-4.19236469],
       [-1.34470844],
       [ 2.38127279],
       [-9.47126675],
       [ 9.30797291]], dtype=float32), array([-0.45977819], dtype=float32)]
[array([[ 0.],
       [ 1.],
       [ 1.],
       [ 0.]], dtype=float32), array([[ True],
       [ True],
       [ True],
       [ True]], dtype=bool)]
Accuracy: 1.0
```

Deep NN for XOR




```
W1 = tf.Variable(tf.random_uniform([2, 5], -1.0, 1.0))
W2 = tf.Variable(tf.random_uniform([5, 4], -1.0, 1.0))
W3 = tf.Variable(tf.random_uniform([4, 1], -1.0, 1.0))

b1 = tf.Variable(tf.zeros([5]), name="Bias1")
b2 = tf.Variable(tf.zeros([4]), name="Bias2")
b3 = tf.Variable(tf.zeros([1]), name="Bias2")

# Our hypothesis
L2 = tf.sigmoid(tf.matmul(X, W1) + b1)
L3 = tf.sigmoid(tf.matmul(L2, W2) + b2)
hypothesis = tf.sigmoid(tf.matmul(L3, W3) + b3)
```

Deep NN for XOR

```
198000 cost: 0.00111391
    predict [[  5.58296102e-04]
 [  9.98885453e-01]
 [  9.98768628e-01]
 [  1.54864462e-03]]
    W1, B1 [array([[ 5.39242172,  2.17747116,  5.26371241, -0.18648638, -0.05985435],
       [ 4.86114264, -4.64419937, -3.61568236, -1.43215752, -2.03103042]], dtype=float32), array([ 0.
    W2, B2 [array([[-2.3721838 , -3.82411337,  2.479671  ,  1.95326269],
       [-2.15816069, -3.55406022,  2.87128544,  2.1189158 ],
       [ 2.92693424,  4.71013832, -3.4270494 , -2.09762478],
       [-0.03848177, -0.86011964,  1.32942307,  0.62142414],
       [-1.01190925, -1.92403996,  0.65061325, -0.27596042]], dtype=float32), array([ 0.70831299,  1.
[array([[ 0.],
       [ 1.],
       [ 1.],
       [ 0.]], dtype=float32), array([[ True],
       [ True],
       [ True],
       [ True]], dtype=bool)]
Accuracy: 1.0
```

Let's go deep & wide!

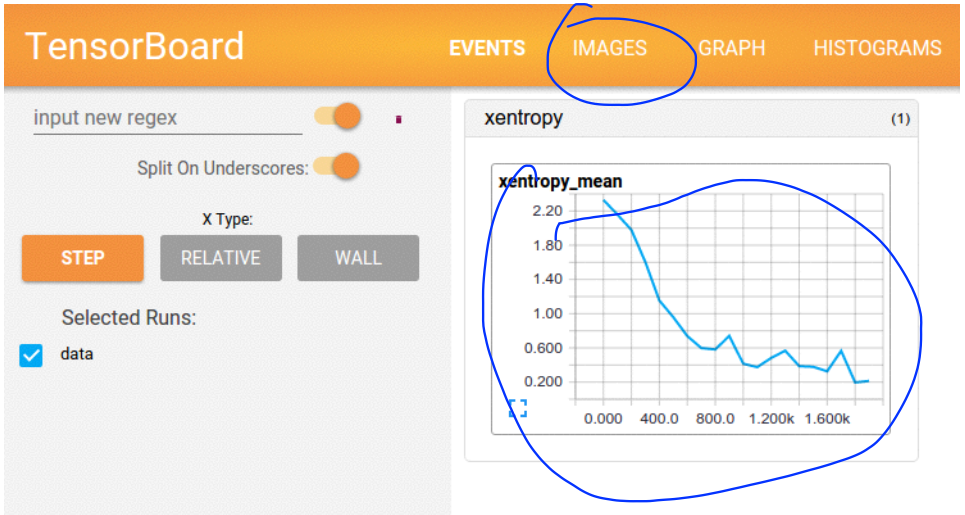
```
W1 = tf.Variable(tf.random_uniform([2, 5], -1.0, 1.0))
W2 = tf.Variable(tf.random_uniform([5, 4], -1.0, 1.0))
W3 = tf.Variable(tf.random_uniform([4, 1], -1.0, 1.0))

b1 = tf.Variable(tf.zeros([5]), name="Bias1")
b2 = tf.Variable(tf.zeros([4]), name="Bias2")
b3 = tf.Variable(tf.zeros([1]), name="Bias2")

# Our hypothesis
L2 = tf.sigmoid(tf.matmul(X, W1) + b1)
L3 = tf.sigmoid(tf.matmul(L2, W2) + b2)
hypothesis = tf.sigmoid(tf.matmul(L3, W3) + b3)
```

Lab 9-2
Tensorboard

Sung Kim <hunkim+ml@gmail.com>

TensorBoard: TF logging/debugging tool

- Visualize your TF graph
- Plot quantitative metrics
- Show additional data

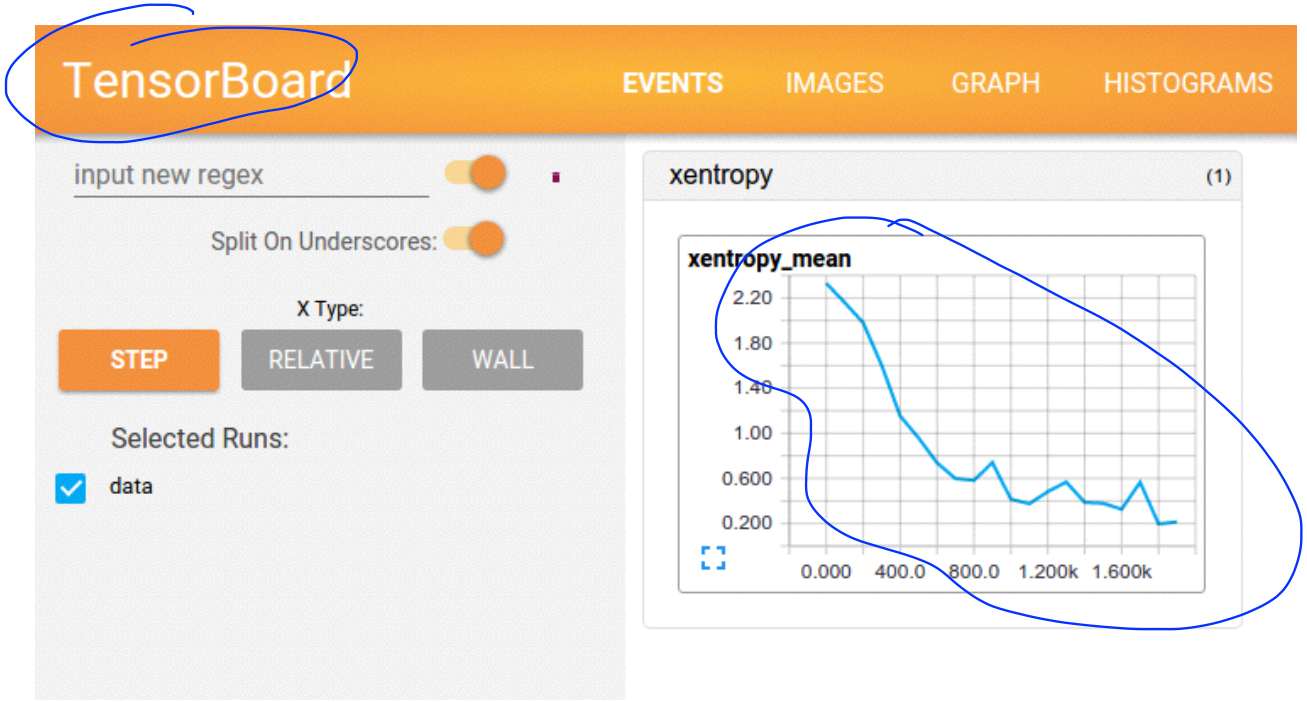


https://www.tensorflow.org/versions/r0.7/how_tos/summaries_and_tensorboard/index.html

Old fashion

```
2000 [0.69364417, array([[ 0.50981331,  0.50592244],
       [ 0.37054271,  0.37088916],
       [ 0.6810087 ,  0.38607275],
       [ 0.54717511,  0.26581794]], dtype=float32), array([[ 0.50861073],
       [ 0.51602864],
       [ 0.4826754 ],
       [ 0.49036184]], dtype=float32), array([[ 0.71915275, -0.48754135],
       [-0.56914777, -0.55209494]], dtype=float32), array([[-0.44138899],
       [ 0.23536676]], dtype=float32), array([ 0.03925836,  0.02369077], dtype=float32), array([ 0.14039496], dtype=float32)]
4000 [0.69332385, array([[ 0.52235132,  0.50927138],
       [ 0.38598102,  0.37814924],
       [ 0.69650716,  0.39592981],
       [ 0.56881481,  0.27748841]], dtype=float32), array([[ 0.50748861],
       [ 0.51554251],
       [ 0.48338425],
       [ 0.49113813]], dtype=float32), array([[ 0.74125487, -0.45954311],
       [-0.55370271, -0.53450096]], dtype=float32), array([[-0.42565805],
       [ 0.19686614]], dtype=float32), array([ 0.08946501,  0.03708982], dtype=float32), array([ 0.15204136], dtype=float32)]
6000 [0.69306737, array([[ 0.53439337,  0.51197231],
       [ 0.39961013,  0.38383543],
       [ 0.71191686,  0.40380618],
       [ 0.58899951,  0.2868301 ]], dtype=float32), array([[ 0.50660294],
       [ 0.51538038],
```

New way!


TensorBoard
EVENTS
IMAGES
GRAPH
HISTOGRAMS
input new regex
Split On Underscores:
X Type:
STEP
RELATIVE
WALL
Selected Runs:
data
xentropy
(1)
xentropy_mean
2.20
1.80
1.40
1.00
0.600
0.200
0.000
400.0
800.0
1.200k
1.600k

5 steps of using tensorboard

- From TF graph, decide which node you want to annotate
  - with tf.name_scope("test") as scope:
  - tf.histogram_summary("weights", W), tf.scalar_summary("accuracy", accuracy)
- Merge all summaries
  - merged = tf.merge_all_summaries()
- Create writer
  - writer = tf.train.SummaryWriter("/tmp/mnist_logs", sess.graph_def)
- Run summary merge and add_summary
  - summary = sess.run(merged, …); writer.add_summary(summary);
- Launch Tensorboard
  - tensorboard --logdir=/tmp/mnist_logs

Name variables

```
X = tf.placeholder(tf.float32, name = 'X-input')
Y = tf.placeholder(tf.float32, name = 'Y-input')

W1 = tf.Variable(tf.random_uniform([2, 2], -1.0, 1.0), name = "Weight1")
W2 = tf.Variable(tf.random_uniform([2, 1], -1.0, 1.0), name = "Weight2")

b1 = tf.Variable(tf.zeros([2]), name="Bias1")
b2 = tf.Variable(tf.zeros([1]), name="Bias2")
```

Add scope for better graph hierarch

```
# Our hypothesis
with tf.name_scope("layer2") as scope:
    L2 = tf.sigmoid(tf.matmul(X, W1) + b1)

with tf.name_scope("layer3") as scope:
    hypothesis = tf.sigmoid(tf.matmul(L2, W2) + b2)

# cost function
with tf.name_scope("cost") as scope:
    cost = -tf.reduce_mean(Y*tf.log(hypothesis) + (1-Y)*tf.log(1-hypothesis))
    cost_summ = tf.scalar_summary("cost", cost)

# Minimize
with tf.name_scope("train") as scope:
    optimizer = tf.train.GradientDescentOptimizer(0.01) # Learning rate, alpha
    train = optimizer.minimize(cost)
```

Add histogram

```
w1_hist = tf.histogram_summary("weights1", W1)
w2_hist = tf.histogram_summary("weights2", W2)

b1_hist = tf.histogram_summary("biases1", b1)
b2_hist = tf.histogram_summary("biases2", b2)

y_hist = tf.histogram_summary("y", Y)
```

Add scalar variables

```
# cost function
with tf.name_scope("cost") as scope:
    cost = -tf.reduce_mean(Y*tf.log(hypothesis) + (1-Y)*tf.log(1-hypothesis))
    cost_summ = tf.scalar_summary("cost", cost)
```

merge summaries and create writer after creating session

```
# Launch the graph.
with tf.Session() as sess:

    #tensorboard --logdir=./logs/xor_logs
    merged = tf.merge_all_summaries()
    writer = tf.train.SummaryWriter("./logs/xor_logs", sess.graph_def)
```

Run merged summary and write (add summary)

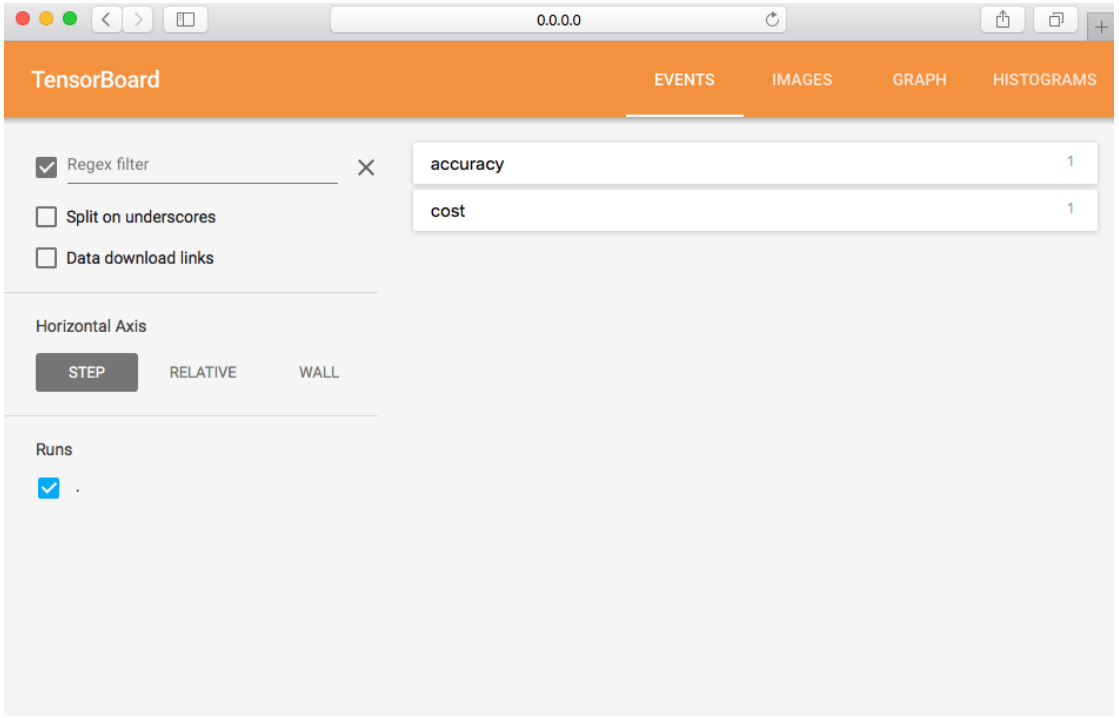
```
# Fit the line.
for step in xrange(200000):
    summary, _ = sess.run([merged, train], feed_dict={X:x_data, Y:y_data})
    writer.add_summary(summary, step)
```



```
# Fit the line.
for step in xrange(200000):
    sess.run(train, feed_dict={X:x_data, Y:y_data})
    if step % 2000 == 0:
        summary = sess.run(merged, feed_dict={X:x_data, Y:y_data})
        writer.add_summary(summary, step)
```

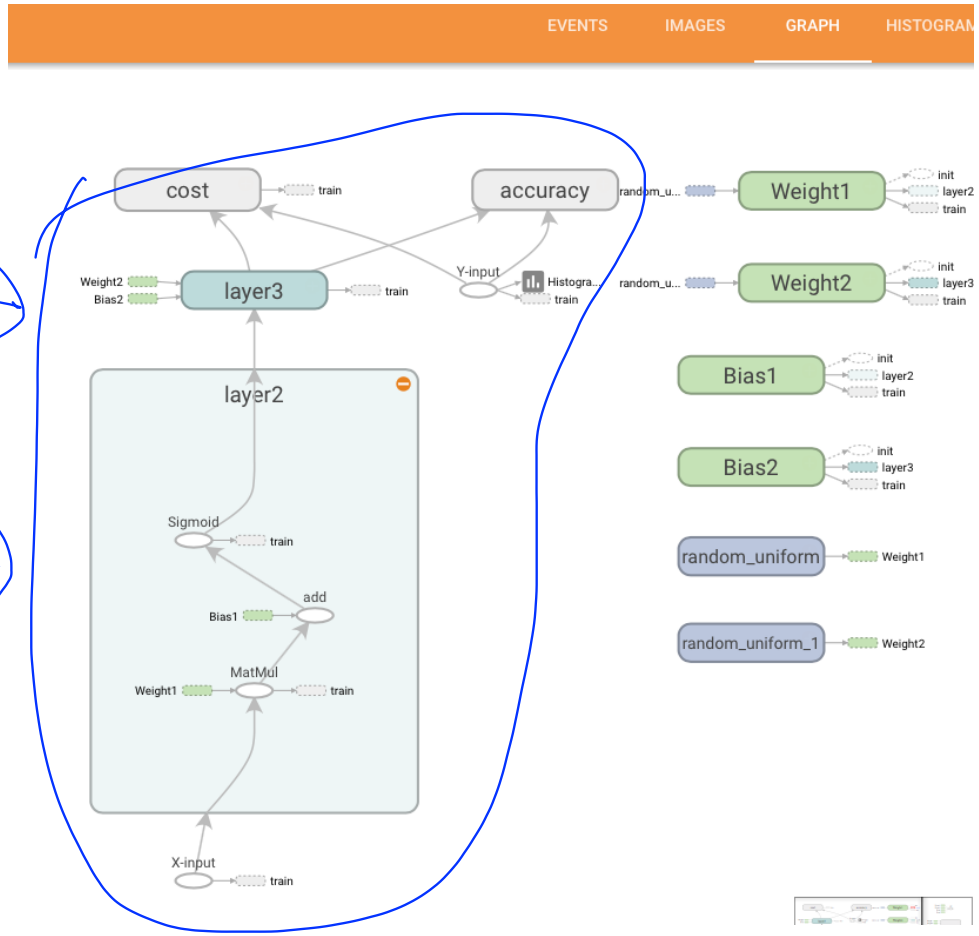
Launch tensorboard

- tensorboard —logdir=/tmp/mnist_logs
- (You can navigate to http://0.0.0.0:6006)



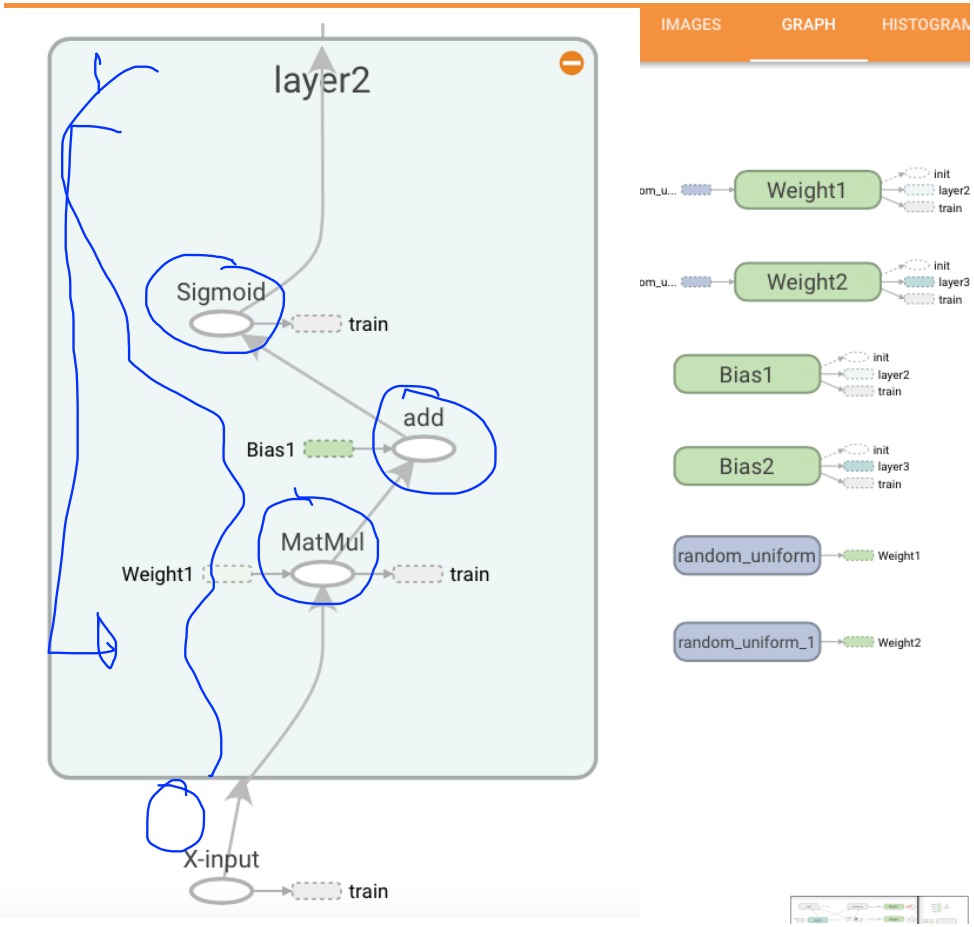
Add scope for better graph hierarch

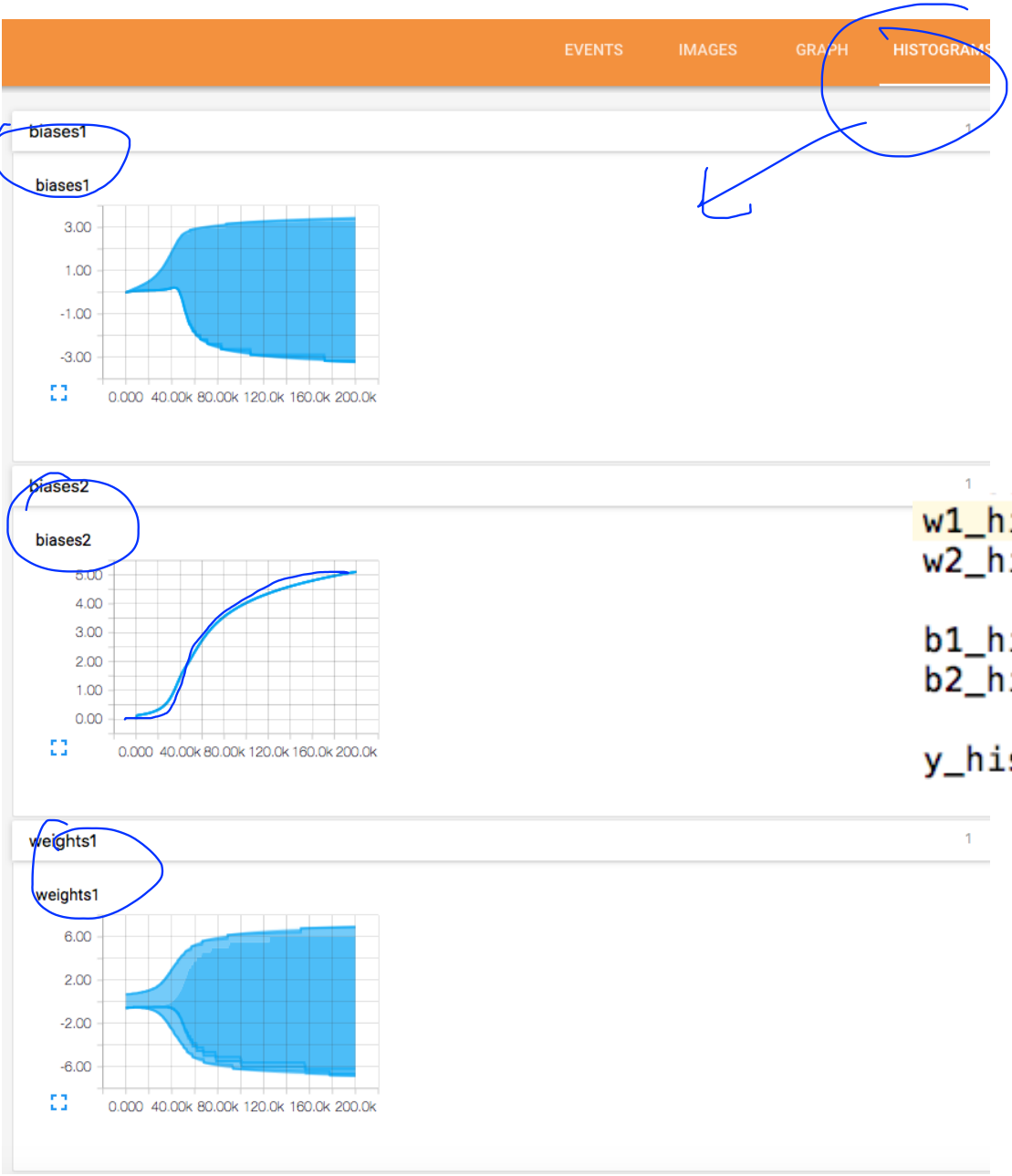
```python
# Our hypothesis
with tf.name_scope("layer2") as scope:
    L2 = tf.sigmoid(tf.matmul(X, W1) + b1)

with tf.name_scope("layer3") as scope:
    hypothesis = tf.sigmoid(tf.matmul(L2, W2) + b2)

# cost function
with tf.name_scope("cost") as scope:
    cost = -tf.reduce_mean(Y*tf.log(hypothesis) + (1-Y)*tf.log(1-hypothesis))
    cost_summ = tf.scalar_summary("cost", cost)

# Minimize
with tf.name_scope("train") as scope:
    optimizer = tf.train.GradientDescentOptimizer(0.01) # Learning rate, alpha
    train = optimizer.minimize(cost)
```



Add scope for better graph hierarch

```python
# Our hypothesis
with tf.name_scope("layer2") as scope:
    L2 = tf.sigmoid(tf.matmul(X, W1) + b1)

with tf.name_scope("layer3") as scope:
    hypothesis = tf.sigmoid(tf.matmul(L2, W2) + b2)

# cost function
with tf.name_scope("cost") as scope:
    cost = -tf.reduce_mean(Y*tf.log(hypothesis) + (1-Y)*tf.log(1-hypothesis))
    cost_summ = tf.scalar_summary("cost", cost)

# Minimize
with tf.name_scope("train") as scope:
    optimizer = tf.train.GradientDescentOptimizer(0.01) # Learning rate, alpha
    train = optimizer.minimize(cost)
```

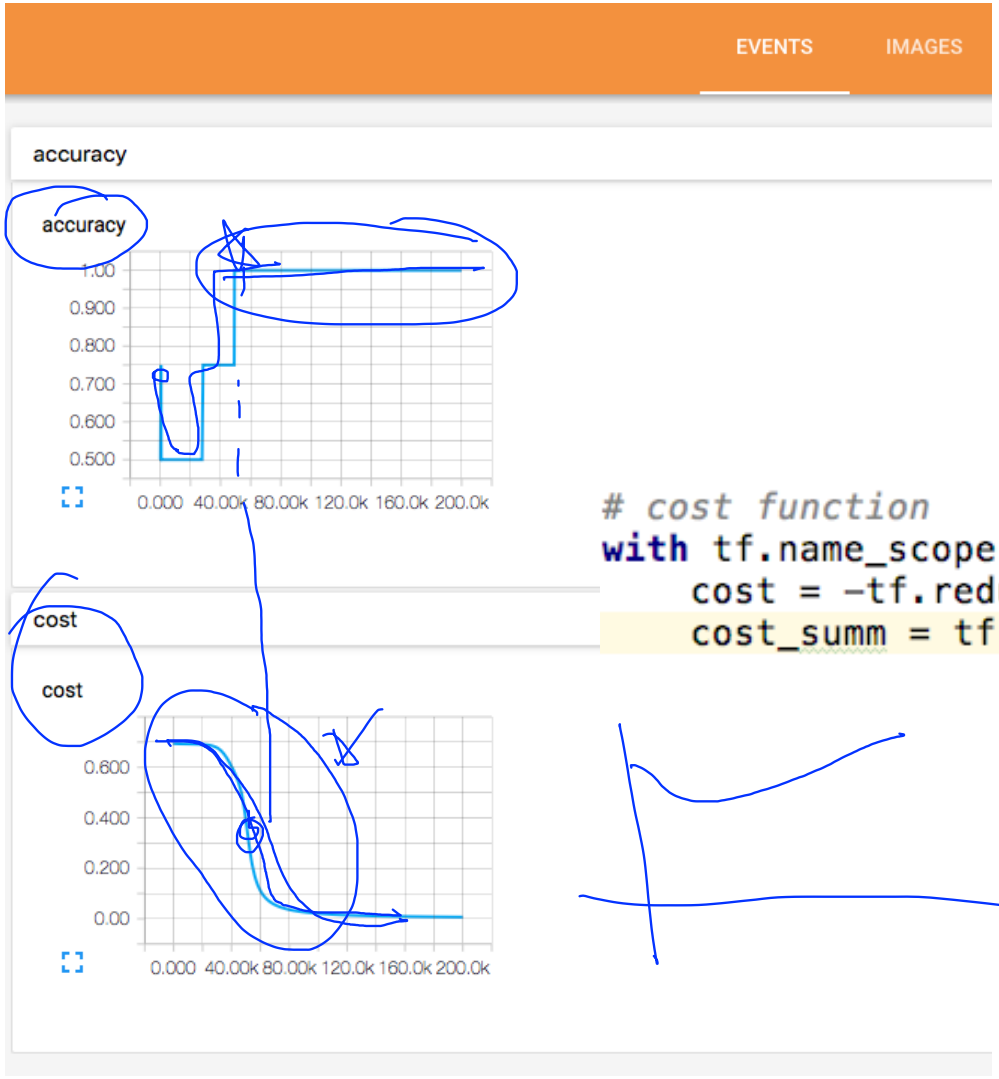




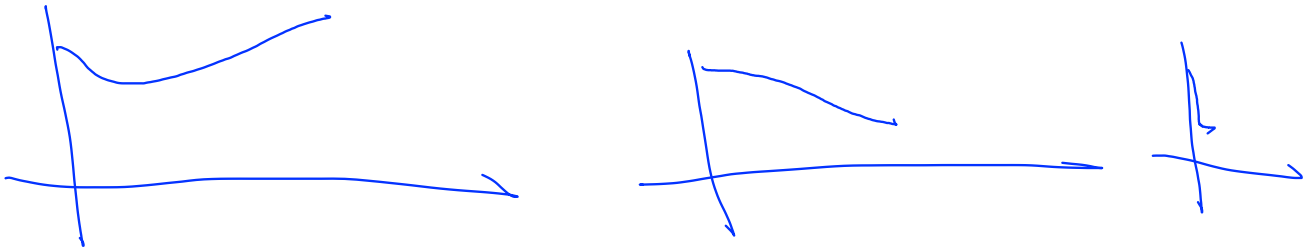
Add histogram

```
w1_hist = tf.histogram_summary("weights1", W1)
w2_hist = tf.histogram_summary("weights2", W2)

b1_hist = tf.histogram_summary("biases1", b1)
b2_hist = tf.histogram_summary("biases2", b2)

y_hist = tf.histogram_summary("y", Y)
```

Add scalar variables




```python
# cost function
with tf.name_scope("cost") as scope:
    cost = -tf.reduce_mean(Y*tf.log(hypothesis) + (1-Y)*tf.log(1-hypothesi
    cost_summ = tf.scalar_summary("cost", cost)
```



5 steps of using tensorboard

- From TF graph, decide which node you want to annotate
  - with tf.name_scope("test") as scope:
  - tf.histogram_summary("weights", W), tf.scalar_summary("accuracy", accuracy)
- Merge all summaries
  - merged = tf.merge_all_summaries()
- Create writer
  - writer = tf.train.SummaryWriter("/tmp/mnist_logs", sess.graph_def)
- Run summary merge and add_summary
  - summary = sess.run(merged, …); writer.add_summary(summary);
- Launch Tensorboard
  - tensorboard --logdir=/tmp/mnist_logs