TensorFlow
- XOR example -

2019 - 2020

Ando Ki, Ph.D.
adki@future-ds.com

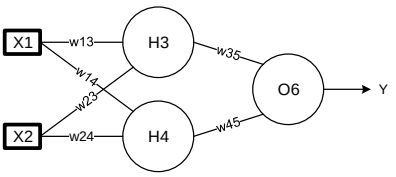
Table of contents

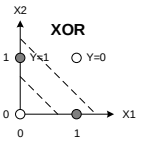
XOR problem: xor_simple.py

- Two input
- Two hidden layers including output
- Two nodes at the hidden layer

See: ~/tensorflow-projects/xor

X	Y	X ⊕ Y
0	0	0
0	1	1
1	0	1
1	1	0





$$\begin{bmatrix} X1 & X2 \end{bmatrix} \times \begin{bmatrix} W(0,0) & W(0,1) \\ W(1,0) & W(1,1) \end{bmatrix} + \begin{bmatrix} b1 & b2 \end{bmatrix} = \begin{bmatrix} y1 & y2 \end{bmatrix}$$

$$\begin{bmatrix} y1 & y2 \end{bmatrix} \times \begin{bmatrix} W(0) \\ W(1) \end{bmatrix} + \begin{bmatrix} b3 \end{bmatrix} = \begin{bmatrix} y \end{bmatrix}$$

XOR problem

- 'xor_simple.py'
  - A simple MLP
- 'xor_tensorboard.py'
  - How to visualize
- 'xor_train.py'
  - How to store
- 'xor_inference.py'
  - How to restore

See: ~/tensorflow-projects/xor

XOR problem: xor_simple.py

See: ~/tensorflow-projects/xor

```
# xor_simple.py'
#----------------------------------------------------------------
import tensorflow as tf
#----------------------------------------------------------------
# variables for input and output
x = tf.placeholder(shape=[4, 2], dtype=tf.float32, name="x-input")
y = tf.placeholder(shape=[4, 1], dtype=tf.float32, name="y-expected")
#----------------------------------------------------------------
# 1st layer
# W1: Shape [2,2]
# b1: shape [2]
W1 = tf.Variable(tf.random_uniform([2,2],-1,1), name="W1")
b1 = tf.Variable([.0,.0], dtype=tf.float32, name="b1")
h1 = tf.tanh(tf.matmul(x, W1) + b1) # (2x1) * (2x2) * (2x1)
#----------------------------------------------------------------
# 2nd layer
# W2: Shape [2,1]
# b2: shape [1]
W2 = tf.Variable(tf.random_uniform([2,1],-1,1), name="W2")
b2 = tf.Variable([.0], dtype=tf.float32, name="b2")
h2 = tf.tanh(tf.matmul(h1, W2) + b2)
```

Variables for input and output, i.e. expected value.

XOR problem: xor_simple.py

See: ~/tensorflow-projects/xor

```
#----------------------------------------------------------------
# cost function: Square Sum
cost = tf.reduce_sum(tf.square(y - h2))

#----------------------------------------------------------------
# optimizer with learning rate 0.05
optimizer = tf.train.GradientDescentOptimizer(0.05)
train = optimizer.minimize(cost)

#----------------------------------------------------------------
sess = tf.Session()
sess.run(tf.global_variables_initializer())

#----------------------------------------------------------------
# training data set
x_train = [[0,0], [0,1], [1,0], [1,1]] # all input patterns
y_train = [[0], [1], [1], [0]] # expected output
```

Training data set

XOR problem: xor_simple.py

```
#----------------------------------------------------------
for i in range(50000):
    sess.run(fetches=train, feed_dict={x:x_train, y:y_train})
    if i%5000==0:
        print('Batch:     ', i)
        print('W1:        ', sess.run(W1))
        print('b1:        ', sess.run(b1))
        print('W2:        ', sess.run(W2))
        print('b2:        ', sess.run(b2))
        print('Inference: ', sess.run(h2, {x:x_train, y:y_train}))
        print('Cost:      ', sess.run(cost,  {x:x_train, y:y_train}))
```

See: ~/tensorflow-projects/xor

XOR problem: xor_simple.py

```
[adki@ando-ubuntu] source ~/tensorflow/bin/activate
(tensorflow)$ python xor_simple.py
('Batch:    ', 0)
('W1:       ', array([[-0.85927957, -0.24072911],
       [-0.49553916,  0.74927551]], dtype=float32))
('b1:       ', array([ 0.0968066 , -0.15781502], dtype=float32))
('W2:       ', array([[ 0.7807948 ],
       [-0.94690073]], dtype=float32))
('b2:       ', array([ 0.19623609], dtype=float32))
('Inference: ', array([[ 0.39675662],
       [-0.53870636],
       [ 0.05309473],
       [-0.65668917]], dtype=float32))
('Cost:      ', 3.8529031)
.......
('Batch:    ', 45000)
('W1:       ', array([[-1.62938082, -2.08491635],
       [-1.62807906, -2.08184791]], dtype=float32))
('b1:       ', array([ 2.50716543,  0.70974225], dtype=float32))
('W2:       ', array([[ 2.58982038],
       [-2.61171842]], dtype=float32))
('b2:       ', array([-0.96112704], dtype=float32))
('Inference: ', array([[  2.28881818e-05],
       [  9.96430457e-01],
       [  9.96431231e-01],
       [  2.92062741e-05]], dtype=float32))
('Cost:      ', 2.5479127e-05)
(tensorflow)$ tensorboard --logdir=/tmp/tensorflow
Starting TensorBoard 54 at http://ando-ubuntu:6006
(Press CTRL+C to quit)
```

This does give different result for each execution. Why?

Cost reaches at near zero

XOR problem: xor_simple.py

- This example
  - Step 1: go to your project directory
    - [user@host] cd $(PROJECT)/codes/tensorflow-project/xor
  - Step 2: see the codes
  - Step 3: run Python under virtual environment
    - (do not forget to run '$ source ~/tensorflow/bin/activate')
    - [user@host] python xor_simple.py

```
[user@host] cd $(PROJECT)/codes/tensorflow-project/mnist-project/xor
[user@host] python xor_simple.py
```


Your project

- Use different activation function
  - sigmoid
  - relu
- User different loss function

Visualizing with TensorFlow

- Basic approaches
  - Give name to the variables (placeholder, Variable, …)
  - Add scope for better graph hierarchy
  - Get histogram (rank>=1)
  - Get scalar (rank==1)

```
X = tf.placeholder(name="X-input")
W = tf.Variable(name="W-param")
D = tf.square(Y_ - y, name="delta")
```

```
with tf.name_scope("layer"):
        W1 = tf.Variable([0, 0], name='W1')
        b1  = tf.Variable([0], name="b1")
with tf.name_scope("cost"):
        cost = tf.reduce_mean(tf.square(y – h))
```

```
w1_hist = tf.summary.histogram("w1", w1)
```

```
cost_sum = tf.summary.summary("cost", cost)
```


XOR problem: xor_tensorboard.py

See: ~/tensorflow-projects/xor

```
# xor_tensorboard.py'
#----------------------------------------------------------
import tensorflow as tf

#----------------------------------------------------------
# variables for input and output
with tf.name_scope("input"):
    x = tf.placeholder(shape=[4, 2], dtype=tf.float32, name="x-input")
    y = tf.placeholder(shape=[4, 1], dtype=tf.float32, name="y-expected")

#----------------------------------------------------------
# 1st layer
# W1: Shape [2,2]
# b1: shape [2]
with tf.name_scope("layer-1st"):
    W1 = tf.Variable(tf.random_uniform([2,2],-1,1), name="W1")
    b1 = tf.Variable([.0,.0], dtype=tf.float32, name="b1")
    h1 = tf.sigmoid(tf.matmul(x, W1) + b1)
```

Add scope "input"

Name variable

XOR problem: xor_tensorboard.py

See: ~/tensorflow-projects/xor

```
#------------------------------------------------------------
# 2nd layer
# W2: Shape [2,1]
# b2: shape [1]
with tf.name_scope("layer-2nd"):
    W2 = tf.Variable(tf.random_uniform([2,1],-1,1), name="W2")
    b2 = tf.Variable([.0], dtype=tf.float32, name="b2")
    h2 = tf.sigmoid(tf.matmul(h1, W2) + b2)

#------------------------------------------------------------
# cost function: MSE (Mean Square Estimate)
with tf.name_scope("cost"):
    cost = tf.reduce_mean(tf.square(y - h2))

#------------------------------------------------------------
# optimizer with learning rate 0.05
optimizer = tf.train.GradientDescentOptimizer(0.05)
train = optimizer.minimize(cost)
```


XOR problem: xor_tensorboard.py

See: ~/tensorflow-projects/xor

```
#------------------------------------------------------------
# dynamic log
w1_hist = tf.summary.histogram("W1-weight", W1)
b1_hist = tf.summary.histogram("b1-bias", b1)
w2_hist = tf.summary.histogram("W2-weight", W2)
b2_hist = tf.summary.histogram("b2-bias", b2)
cost_summ = tf.summary.scalar("cost", cost)

summary_op = tf.summary.merge_all()

#------------------------------------------------------------
sess = tf.Session()
sess.run(tf.global_variables_initializer())

#------------------------------------------------------------
# tensorboard --logdir=/tmp/tensorflow
# http://localhost:6006
writer = tf.summary.FileWriter('logs', sess.graph)
```

Histogram

Scalar

Merge operation, which will be called within session

Record graph

XOR problem: xor_tensorboard.py

See: ~/tensorflow-projects/xor

```
#----------------------------------------------------------------
# training data set
x_train = [[0,0], [0,1], [1,0], [1,1]] # all input patterns
y_train = [[0], [1], [1], [0]] # expected output

#----------------------------------------------------------------
for i in range(10000):
    sess.run(fetches=train, feed_dict={x:x_train, y:y_train})
    if i%1000==0:
        summary = sess.run(fetches=summary_op, feed_dict={x:x_train, y:y_train})
        writer.add_summary(summary, i)
        print('Batch:     ', i)
        print('W1:        ', sess.run(W1))
        print('b1:        ', sess.run(b1))
        print('W2:        ', sess.run(W2))
        print('b2:        ', sess.run(b2))
        print('Inference: ', sess.run(h2, {x:x_train, y:y_train}))
        print('Cost:      ', sess.run(cost,  {x:x_train, y:y_train}))
```

This will add log at each 1000 step

Run summary operation

Merge

This will add log at each 1000 step.

XOR problem: xor_tensorboard.py

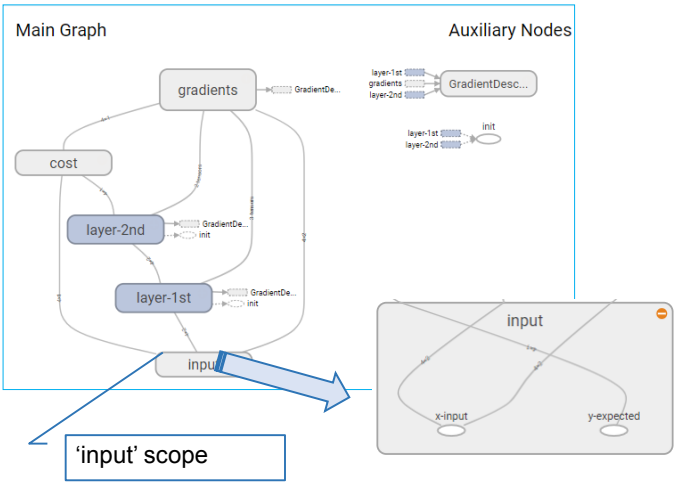
```
[adki@ando-ubuntu] source ~/tensorflow/bin/activate
(tensorflow)$ python xor_tensorboard.py
('Batch:     ', 0)
('W1:        ', array([[-0.85927957, -0.24072911],
       [-0.49553916,  0.74927551]], dtype=float32))
('b1:        ', array([ 0.0968066 , -0.15781502], dtype=float32))
('W2:        ', array([[ 0.7807948 ],
       [-0.94690073]], dtype=float32))
('b2:        ', array([ 0.19623609], dtype=float32))
('Inference: ', array([[ 0.39675662],
       [-0.53870636],
       [ 0.05309473],
       [-0.65668917]], dtype=float32))
('Cost:      ', 3.8529031)
.....
('Batch:     ', 45000)
('W1:        ', array([[-1.62938082, -2.08491635],
       [-1.62807906, -2.08184791]], dtype=float32))
('b1:        ', array([ 2.50716543,  0.70974225], dtype=float32))
('W2:        ', array([[ 2.58982038],
       [-2.61171842]], dtype=float32))
('b2:        ', array([-0.96112704], dtype=float32))
('Inference: ', array([[  2.28881818e-05],
       [  9.96430457e-01],
       [  9.96431231e-01],
       [  2.92062741e-05]], dtype=float32))
('Cost:      ', 2.5479127e-05)
(tensorflow)$ tensorboard --logdir=logs
Starting TensorBoard 54 at http://ando-ubuntu:6006
(Press CTRL+C to quit)
```

This does give different result for each execution. Why?

Cost reaches at near zero

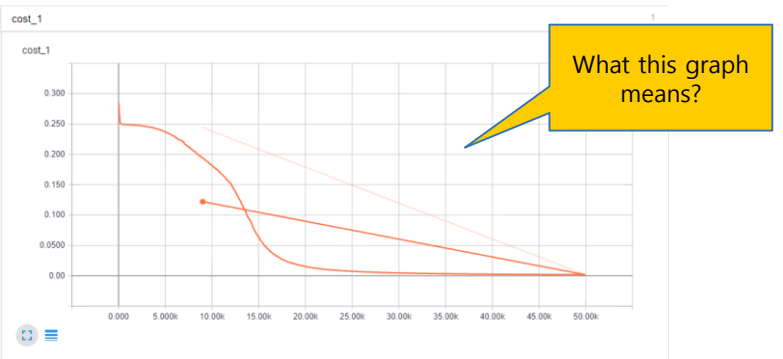
XOR problem: xor_tensorboard.py

- Invoke web-browser with ‘http://localhost:6006’ or ‘http://server:6006’.



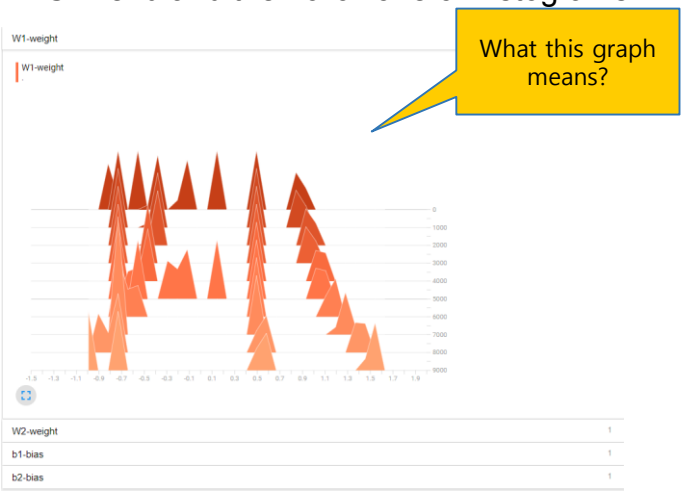
XOR problem: xor_tensorboard.py

- Select ‘SCALARS’ menu and then click one of scalars



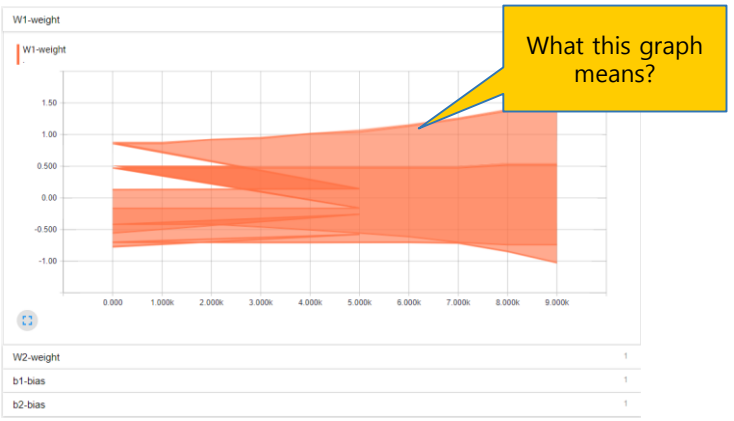
XOR problem: xor_tensorboard.py

- Select ‘HISTOGRAMS’ menu and then click one of histograms



XOR problem: xor_tensorboard.py

- Select ‘DISTRIBUTIONS’ menu and then click one of distributions

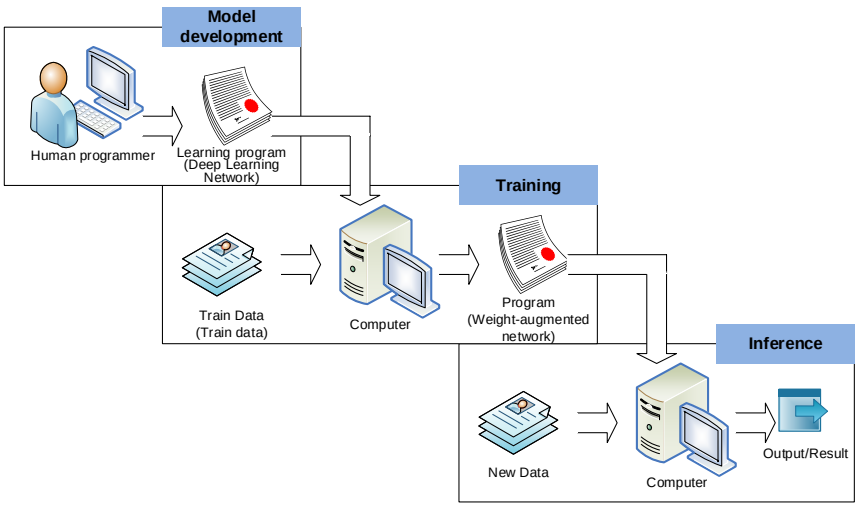


XOR problem: xor_tensorboard.py

- This example
  - Step 1: go to your project directory
    - [user@host] cd $(PROJECT)/codes/tensorflow-project/xor
  - Step 2: see the codes
  - Step 3: run Python under virtual environment
    - (do not forget to run '$ source ~/tensorflow/bin/activate')
    - [user@host] python xor_tensorboard.py

```
[user@host] cd $(PROJECT)/codes/tensorflow-project/mnist-project/xor
[user@host] python xor_tensorboard.py
[user@host] tensorboard --logdir=logs
```


Training and inference with TensorFlow



Training and inference with TensorFlow

- Basic operations
  - Training:
    - Step 1: Store checkpoint after training
  - Inference
    - Step 2: Create graph from the checkpoint
    - Step 3: Restore parameters, i.e., weights from the checkpoint
    - Step 4: Get graph handler
    - Step 5: Get handlers, i.e., tensor references
    - Step 6: Perform inference with new data-set

```
saver = tf.train.Saver()
```

```
saver.save(sess, “checkpoint”)
```

```
saver tf.train.import_meta_graph(“metafile”)
```

```
save.restore(sess, “checkpoint”)
```

```
graph = tf.get_default_graph()
```

```
x = graph.get_tensor_by_name(“name-of-tensor”)
```

```
opt = graph.get_tensor_by_name(“operator”)
```


XOR problem: xor_train.py

See: ~/tensorflow-projects/xor

```
# xor_train.py'
#------------------------------------------------------------
import tensorflow as tf
#------------------------------------------------------------
# variables for input and output
x = tf.placeholder(shape=[4, 2], dtype=tf.float32, name="x-input")
y = tf.placeholder(shape=[4, 1], dtype=tf.float32, name="y-expected")
#------------------------------------------------------------
# 1st layer
# W1: Shape [2,2]
# b1: shape [2]
W1 = tf.Variable(tf.random_uniform([2,2],-1,1), name="W1")
b1 = tf.Variable([.0,.0], dtype=tf.float32, name="b1")
h1 = tf.tanh(tf.matmul(x, W1) + b1) # (2x1) * (2x2) * (2x1)
#------------------------------------------------------------
# 2nd layer
# W2: Shape [2,1]
# b2: shape [1]
W2 = tf.Variable(tf.random_uniform([2,1],-1,1), name="W2")
b2 = tf.Variable([.0], dtype=tf.float32, name="b2")
h2 = tf.tanh(tf.matmul(h1, W2) + b2, name="op_to_restore")
```

Note the name of tensors

Note the name of tensors

XOR problem: xor_train.py

See: ~/tensorflow-projects/xor

```
#------------------------------------------------------------
# cost function: Square Sum
cost = tf.reduce_sum(tf.square(y - h2))

#------------------------------------------------------------
# optimizer with learning rate 0.05
optimizer = tf.train.GradientDescentOptimizer(0.05)
train = optimizer.minimize(cost)

#------------------------------------------------------------
# add ops to save and restore all the variables
saver = tf.train.Saver()

#------------------------------------------------------------
sess = tf.Session()
sess.run(tf.global_variables_initializer())
```

Prepare Saver operator

XOR problem: xor_train.py

See: ~/tensorflow-projects/xor

```
#------------------------------------------------------------
# training data set
x_train = [[0,0], [0,1], [1,0], [1,1]] # all input patterns
y_train = [[0], [1], [1], [0]] # expected output
#------------------------------------------------------------
for i in range(50000):
    sess.run(fetches=train, feed_dict={x:x_train, y:y_train})
    if i%200==0:
        result = sess.run(cost,  {x:x_train, y:y_train})
        if (result<0.001):
            saver.save(sess, "./model/model.ckpt")
            break
#------------------------------------------------------------
```

Save checkpoint

After running following files will be ready in 'model' directory.

- checkpoint
- model.chkpt.data-00000-of-00001
- model.ckpt.index
- model.ckpt.meta

XOR problem: xor_inference.py

See: ~/tensorflow-projects/xor

```
# xor_inference.py'
#------------------------------------------------------------
import tensorflow as tf
#------------------------------------------------------------
sess = tf.Session()
#------------------------------------------------------------
# create the network
saver = tf.train.import_meta_graph("./model/model.ckpt.meta")
#------------------------------------------------------------
# restore parameters, i.e., weights
saver.restore(sess,"./model/model.ckpt");
#------------------------------------------------------------
graph = tf.get_default_graph()
#------------------------------------------------------------
# get references of tensors
x = graph.get_tensor_by_name("x-input:0");
y = graph.get_tensor_by_name("y-expected:0");
ops = graph.get_tensor_by_name("op_to_restore:0")
```

Network rebuild

Restore weight from checkpoint

Get graph handler

Get tensor references; input and output;

Note the name of tensor

Note that 'cost', 'optimizer', 'train' are not necessary for inference.

XOR problem: xor_inference.py

See: ~/tensorflow-projects/xor

```
#------------------------------------------------------------
# testing data-set
feed_dict = {x:[[0,0],[0,1],[1,0],[1,1]], # pattern to infer
             y:[[0], [0], [0], [0]]} # it doesn't matter
#------------------------------------------------------------
# inference
print sess.run(ops, feed_dict)
```

New data-set to test

Run for inference

```
(tenorflow)$ python xor_inference.py
[[  7.48395687e-04]
 [  9.78725731e-01]
 [  9.78933573e-01]
 [  1.23107363e-03]]
(tensorflow)$
```

0^0 ➔ 0

0^1 ➔ 1

1^0 ➔ 1

1^1 ➔ 0

XOR problem: xor_train.py xor_inference.py

- This example
  - Step 1: go to your project directory
    - [user@host] cd $(PROJECT)/codes/tensorflow-project/xor
  - Step 2: see the codes
  - Step 3: run Python under virtual environment
    - (do not forget to run '$ source ~/tensorflow/bin/activate')
    - [user@host] python xor_train.py
    - [user@host] python xor_inference.py

```
[user@host] cd $(PROJECT)/codes/tensorflow-project/mnist-project/xor
[user@host] python xor_train.py
[user@host] python xor_inference.py
```


㈜퓨쳐디자인시스템
34051 대전광역시 유성구 문지로 193, KAIST 문지캠퍼스, F723호
(042) 864-0211~0212 / contact@future-ds.com / www.future-ds.com

Future Design Systems, Inc.
Faculty Wing F723, KAIST Munji Campus, 193 Munji-ro, Yuseong-gu, Daejeon 34051, Korea
+82-042-864-0211~0212 / contact@future-ds.com / www.future-ds.com

FUTURE
Design Systems