

Pattern Recognition and Deep Learning

Lecture 7 Basic machine learning models and codes

Prof. Heng Liu

Basic Models in TensorFlow

TensorFlow for Pattern Recognition and Deep Learning
Lecture 3

Agenda

Review

Linear regression in TensorFlow

Optimizers

Logistic regression on MNIST

Loss functions

Review

Computation graph

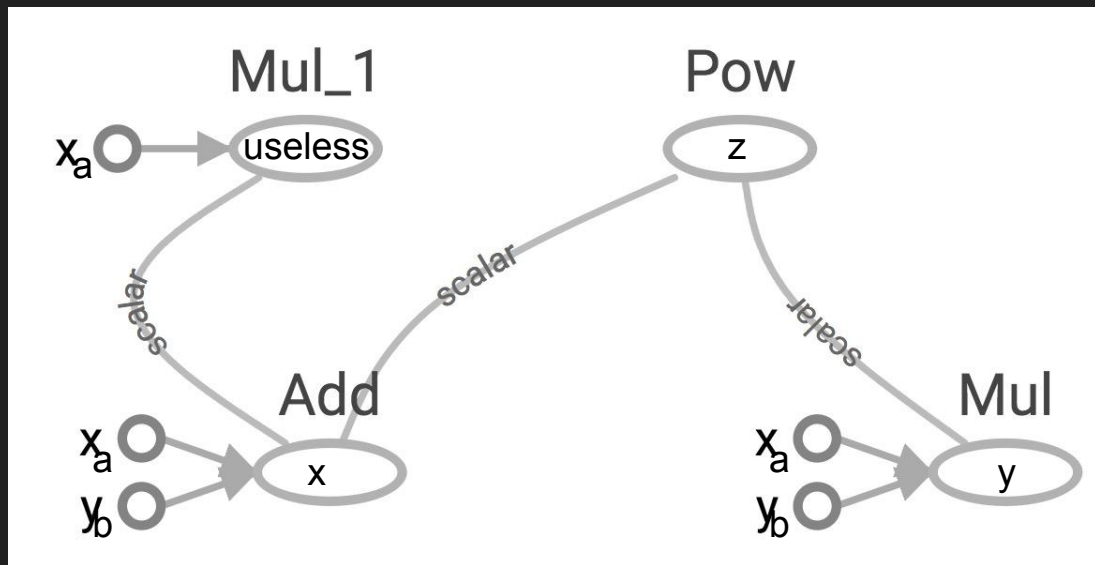
TensorFlow separates definition of computations from their execution

Phase 1: assemble a graph

Phase 2: use a session to execute operations in the graph.

TensorBoard

```
a = 2  
b = 3  
x = tf.add(a, b)  
y = tf.mul(a, b)  
useless = tf.mul(a, x)  
z = tf.pow(y, x)  
  
with tf.Session() as sess:  
    z = sess.run(z)
```



Create a FileWriter object to write your graph to event files

tf.constant and tf.Variable

Constant values are stored in the graph definition

Sessions allocate memory to store variable values

tf.placeholder and feed_dict

Feed values into placeholders by dictionary (feed_dict)

You can feed values in variables too

Avoid lazy loading

1. Separate the assembling of graph and executing ops
2. Use Python attribute to ensure a function is only loaded the first time it's called

Linear Regression

Model relationship between a scalar dependent variable y and independent variables X

The City of Chicago

X: number of incidents of fire

Y: number of incidents of theft

Want

X: number of incidents of fire

Y: number of incidents of theft

Predict Y from X

Model

$$w * X + b$$

$$(Y - Y_{\text{predicted}})^2$$

Phase 1: Assemble our graph

Step 1: Read in data

I already did that for you

Step 2: Create placeholders for inputs and labels

```
tf.placeholder(dtype, shape=None, name=None)
```

Step 3: Create weight and bias

```
tf.Variable(initial_value=None, trainable=True, collections=None,  
            name=None, dtype=None, ...)
```

Step 4: Build model to predict Y

$$Y_{\text{predicted}} = X * w + b$$

Step 5: Specify loss function

```
tf.square(Y - Y_predicted, name="loss")
```

Step 6: Create optimizer

```
tf.train.GradientDescentOptimizer(learning_rate=0.001).minimize(loss)
```

Phase 2: Train our model

`Initialize variables`

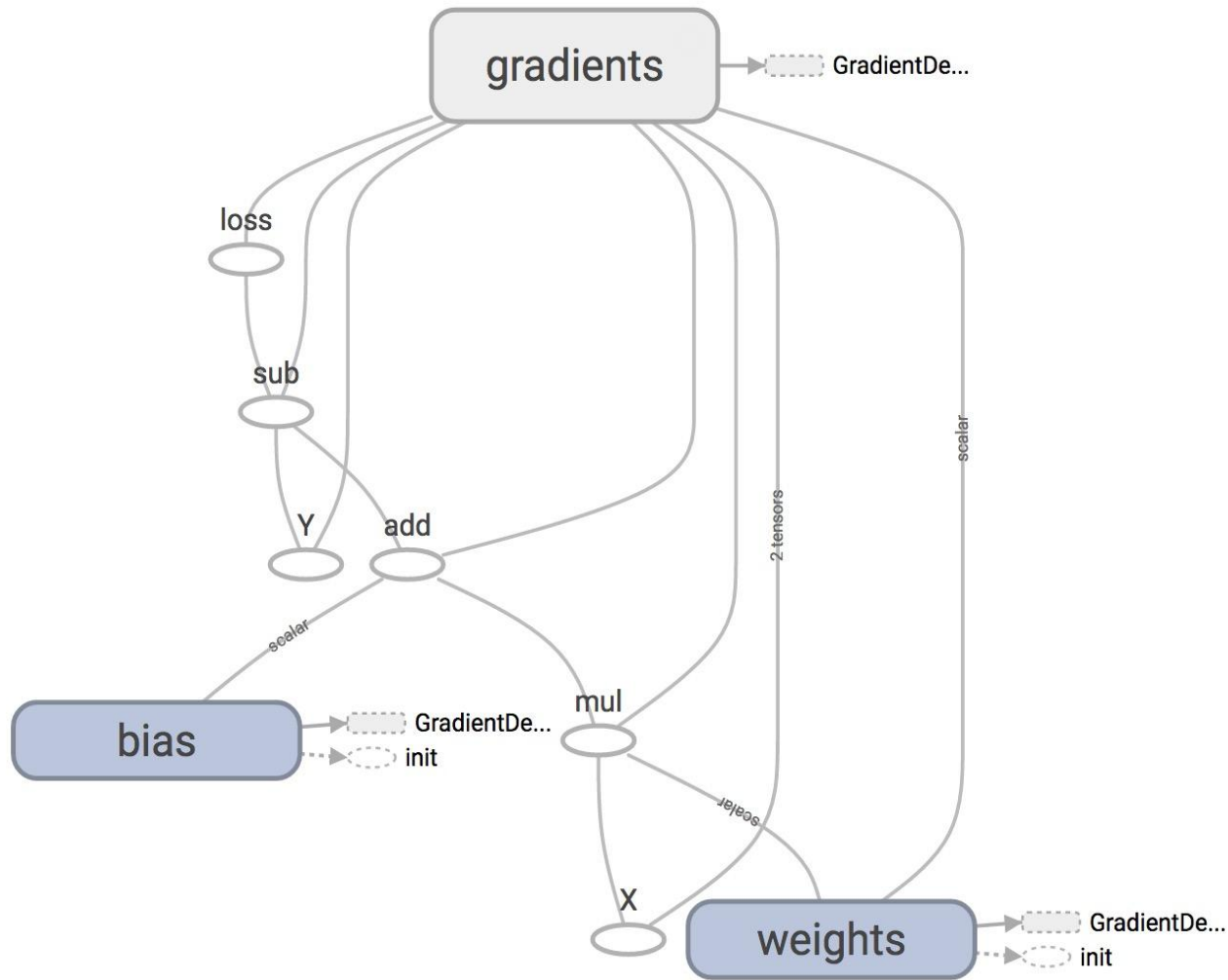
`Run optimizer op`

`(with data fed into placeholders for inputs and labels)`

See your model in TensorBoard

Step 1: `writer = tf.summary.FileWriter('./my_graph/03/linear_reg',
sess.graph)`

Step 2: `$ tensorboard --logdir='./my_graph'`

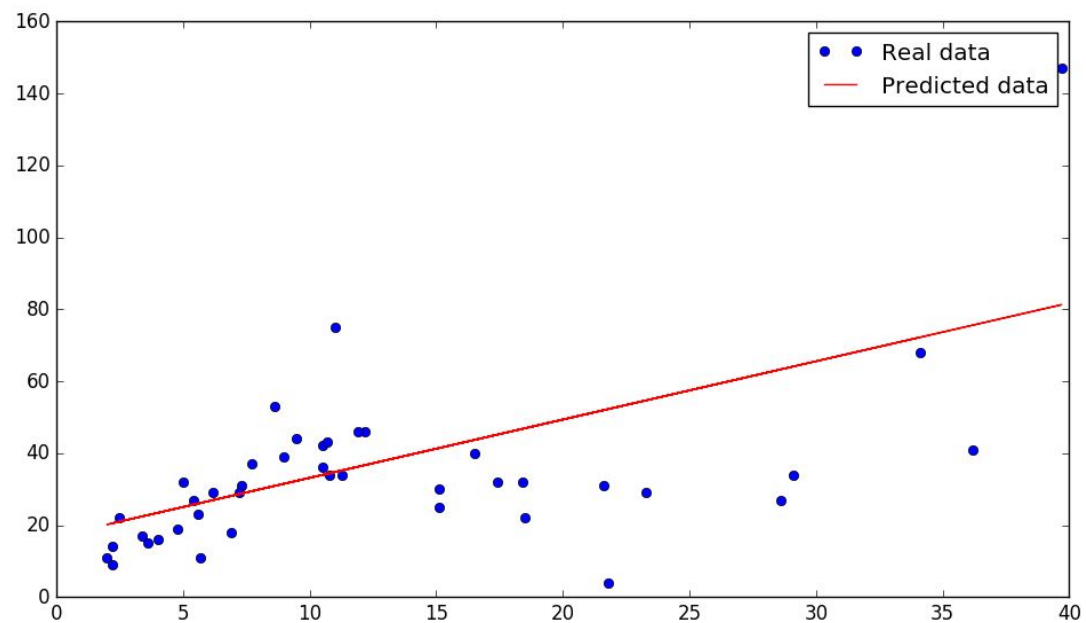


Plot the results with matplotlib

Step 1: Uncomment the plotting code at the end of your program

Step 2: Run it again

If run into problem of matplotlib in virtual environment, go to [GitHub/setups](#) and see the file possible setup problems



ValueError?

ValueError?

```
w, b = sess.run([w, b])
```

**How does TensorFlow know what variables
to update?**

Optimizer

```
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.001).minimize(loss)  
_, l = sess.run([optimizer, loss], feed_dict={X: x, Y:y})
```

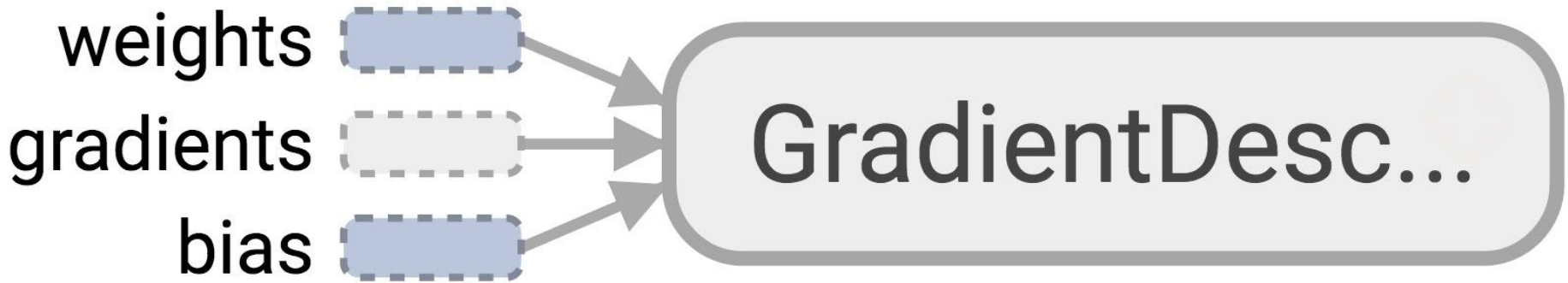
Optimizer

```
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.001).minimize(loss)  
_, l = sess.run([optimizer, loss], feed_dict={X: x, Y:y})
```

Session looks at all trainable variables that loss depends on and update them

Optimizer

Session looks at all trainable variables that optimizer depends on and update them



Trainable variables

```
tf.Variable(initial_value=None, trainable=True, collections=None,  
validate_shape=True, caching_device=None, name=None, variable_def=None, dtype=None,  
expected_shape=None, import_scope=None)
```

List of optimizers in TF

`tf.train.GradientDescentOptimizer`

`tf.train.AdagradOptimizer`

`tf.train.MomentumOptimizer`

`tf.train.AdamOptimizer`

`tf.train.ProximalGradientDescentOptimizer`

`tf.train.ProximalAdagradOptimizer`

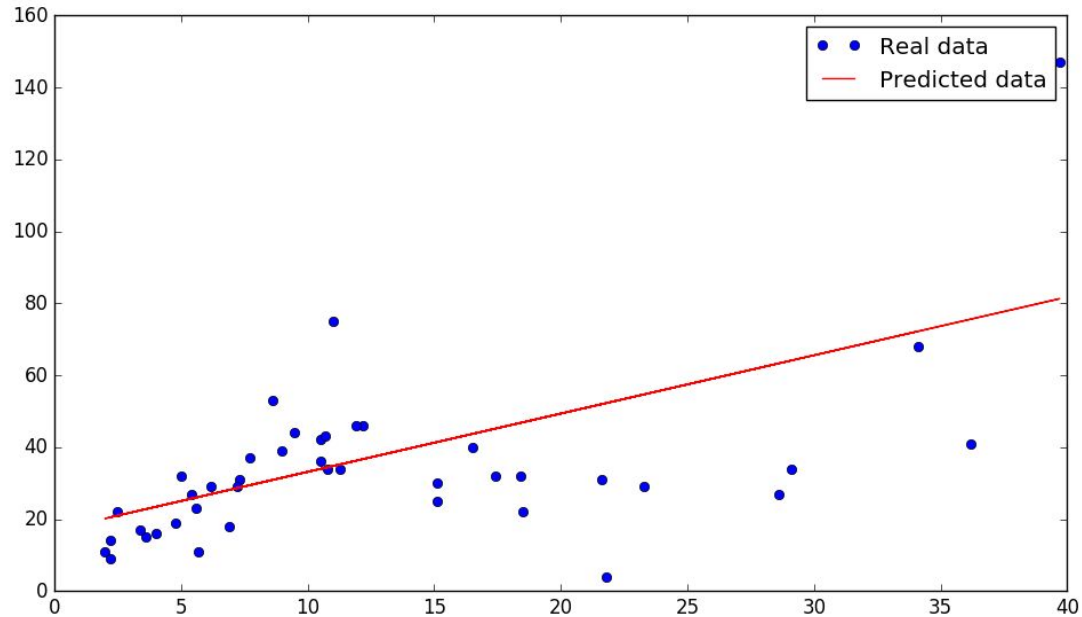
`tf.train.RMSPropOptimizer`

And more

Discussion question

- 1. How to know that our model is correct?**
- 2. How to improve our model?**

How to improve our model



Huber loss

Robust to outliers

Intuition: if the difference between the predicted value and the real value is small, square it

If it's large, take its absolute value

$$L_{\delta}(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{for } |y - f(x)| \leq \delta, \\ \delta |y - f(x)| - \frac{1}{2}\delta^2 & \text{otherwise.} \end{cases}$$

Implementing Huber loss

Can't write:

```
if Y - Y_predicted < delta:
```

$$L_{\delta}(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{for } |y - f(x)| \leq \delta, \\ \delta |y - f(x)| - \frac{1}{2}\delta^2 & \text{otherwise.} \end{cases}$$

Huber loss

```
def huber_loss(labels, predictions, delta=1.0):  
    residual = tf.abs(predictions - labels)  
    condition = tf.less(residual, delta)  
    small_res = 0.5 * tf.square(residual)  
    large_res = delta * residual - 0.5 * tf.square(delta)  
    return tf.select(condition, small_res, large_res)
```

$$L_{\delta}(y, f(x)) = \begin{cases} \frac{1}{2}(y - f(x))^2 & \text{for } |y - f(x)| \leq \delta, \\ \delta |y - f(x)| - \frac{1}{2}\delta^2 & \text{otherwise.} \end{cases}$$

Assignment 1

Optional Interactive Grading

Logistic Regression

**Then he separated the
light from the darkness**

**The first logistic
regression model**

MNIST Database

Each image is a 28x28 array, flattened out to be a 1-d tensor of size 784



MNIST

X: image of a handwritten digit
Y: the digit value

Want

X: image of a handwritten digit

Y: the digit value

Recognize the digit in the image

Model

```
logits = X * w + b
```

```
Y_predicted = softmax(logits)
```

```
loss = cross_entropy(Y, Y_predicted)
```

Batch 'em up

```
X = tf.placeholder(tf.float32, [batch_size, 784], name="image")  
Y = tf.placeholder(tf.float32, [batch_size, 10], name="label")
```


Process data

```
from tensorflow.examples.tutorials.mnist import input_data  
MNIST = input_data.read_data_sets("/data/mnist", one_hot=True)
```

Process data

```
from tensorflow.examples.tutorials.mnist import input_data  
MNIST = input_data.read_data_sets("/data/mnist", one_hot=True)
```

MNIST.train: 55,000 examples

MNIST.validation: 5,000 examples

MNIST.test: 10,000 examples

Phase 1: Assemble our graph

Step 2: Create placeholders for inputs and labels

```
X = tf.placeholder(tf.float32, [batch_size, 784], name="image")  
Y = tf.placeholder(tf.float32, [batch_size, 10], name="label")
```

Step 3: Create weight and bias

```
tf.Variable(initial_value=None, trainable=True, collections=None,  
            name=None, dtype=None, ...)
```

Step 4: Build model to predict Y

$$\text{logits} = X * w + b$$

Step 5: Specify loss function

```
entropy = tf.nn.softmax_cross_entropy_with_logits(logits, Y)
loss = tf.reduce_mean(entropy)
```

Step 6: Create optimizer

```
tf.train.GradientDescentOptimizer(learning_rate=0.001).minimize(loss)
```


Phase 2: Train our model

Initialize variables

Run optimizer op

(with data fed into placeholders for inputs and labels)

Run our model

Average loss epoch 0: 1.28812279526

Average loss epoch 1: 0.732620414598

Average loss epoch 2: 0.600486441648

Average loss epoch 3: 0.53647331619

Average loss epoch 4: 0.497578099683

...

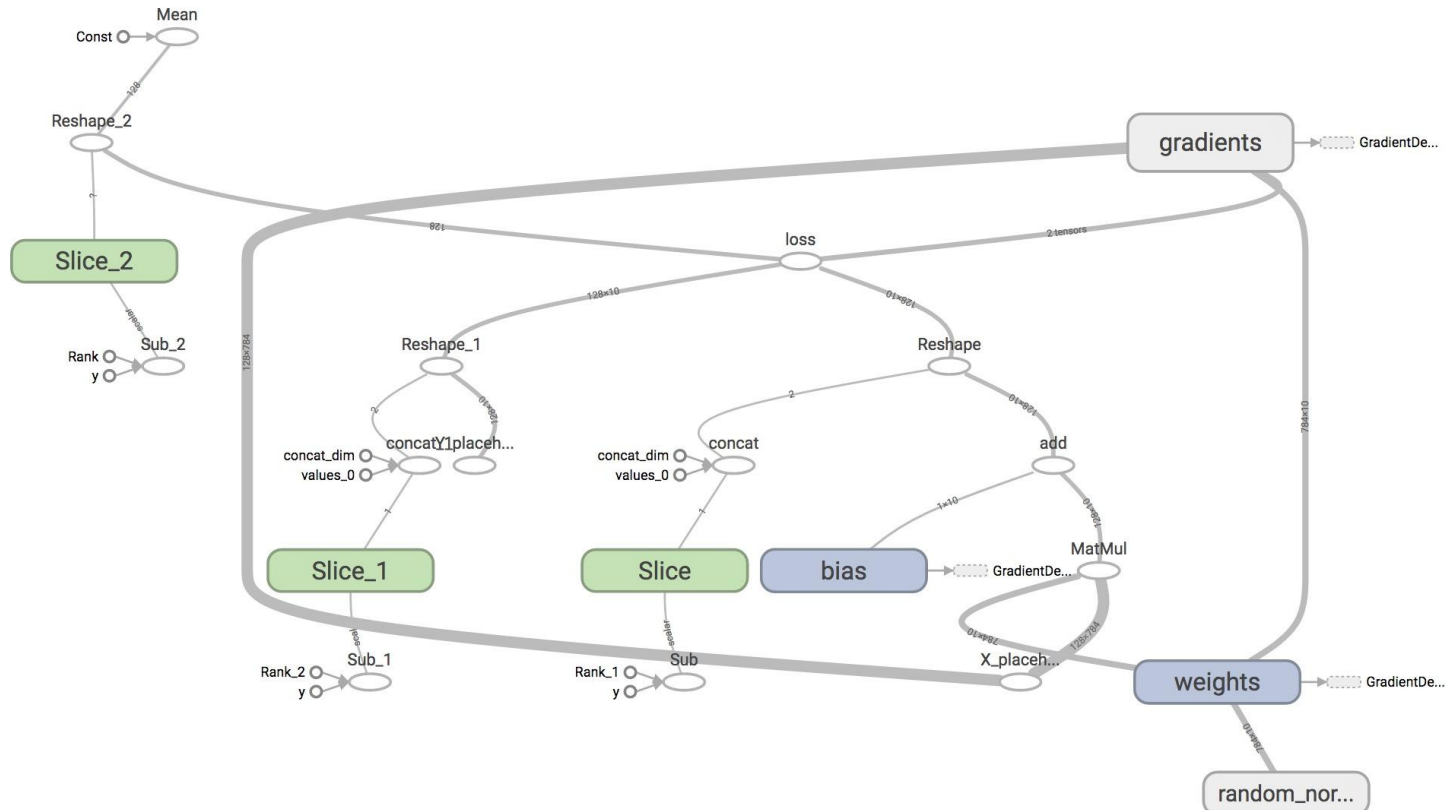
Average loss epoch 9: 0.41295143427

Total time: 8.83596801758 seconds

Optimization Finished!

Accuracy 0.8977

TensorBoard it



Next class

Structure your model in TensorFlow

Example: word2vec

Feedback: 49261200@qq.com

Thanks!