Tensorflow Regression Model

Linear Regression
Logistic Regression
Mmultilayer Perceptron

최준명
Pusan National University
Vision & Intelligent system Lab
https://jimmylifestudy.blogspot.kr/
jmchoi@pusan.ac.kr

Tensorflow | 지난 시간 복습

- Tensorflow는 **Dataflow Graph** 기반의 딥러닝 프레임워크이다.
- 먼저 **Graph**를 Build하고 **Session**을 통해 실행한다.
- 변수는 tf.Variable( ) 보단 **tf.get_variable( )**
- 입력은 **tf.placeholder( )** with feed_dict 나 **tf.data( )**를 쓰자.
  (tf.data( )가 훨씬 빠르나, 쓰기가 어렵다.)

Tensorflow | lazy loading

```
import tensorflow as tf

x = tf.get_variable("x", initializer=tf.constant(10))
y = tf.get_variable("x", initializer=tf.constant(20))
z = tf.add(x,y)

writer = tf.summary.FileWriter('./ckpt', tf.get_default_graph())
with tf.Session() as sess:
  sess.run(tf.global_variables_initializer())
  for _ in xrange(10):
    print(sess.run(z))
writer.close()
```

Tensorflow | lazy loading

```
import tensorflow as tf

x = tf.get_variable("x", initializer=tf.constant(10))
y = tf.get_variable("x", initializer=tf.constant(20))
z = tf.add(x,y)

writer = tf.summary.FileWriter('./ckpt', tf.get_default_graph())
with tf.Session() as sess:
  sess.run(tf.global_variables_initializer())
  for _ in xrange(10):
    print(sess.run(z))
    sess.run(tf.add(x,y))
writer.close()
```

Tensorflow | lazy loading

Normal loading

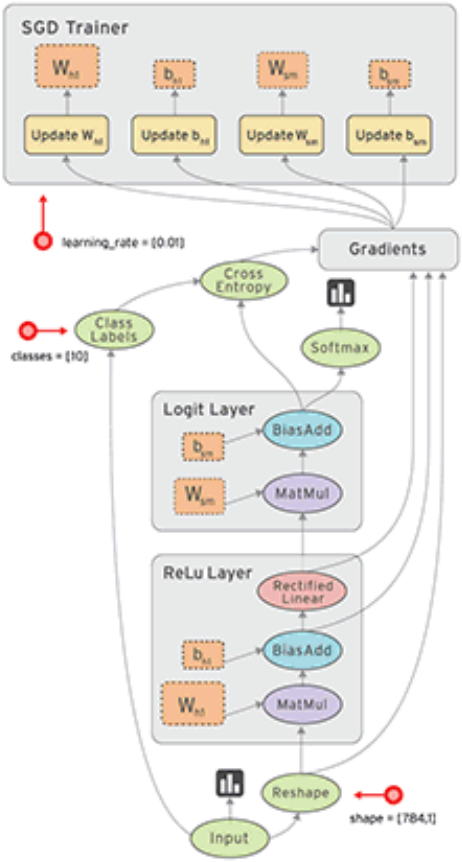
```
node {
 name: "Add"
 op: "Add"
 input: "x/read"
 input: "y/read"
 attr {
  key: "T"
  value {
   type: DT_INT32
  }
 }
}
```

Lazy loading

```
node {
 name: "Add_1"
 op: "Add"
 ...
 }
...
node {
 name: "Add_10"
 op: "Add"
 ...
}
```

Tensorflow | 오늘 목표

- 코드를 보면서 배우는 간단한 모델 만들기!
- Linear Regression
- Logistic Regression
- tf.placeholder( ) version vs tf.data( ) version



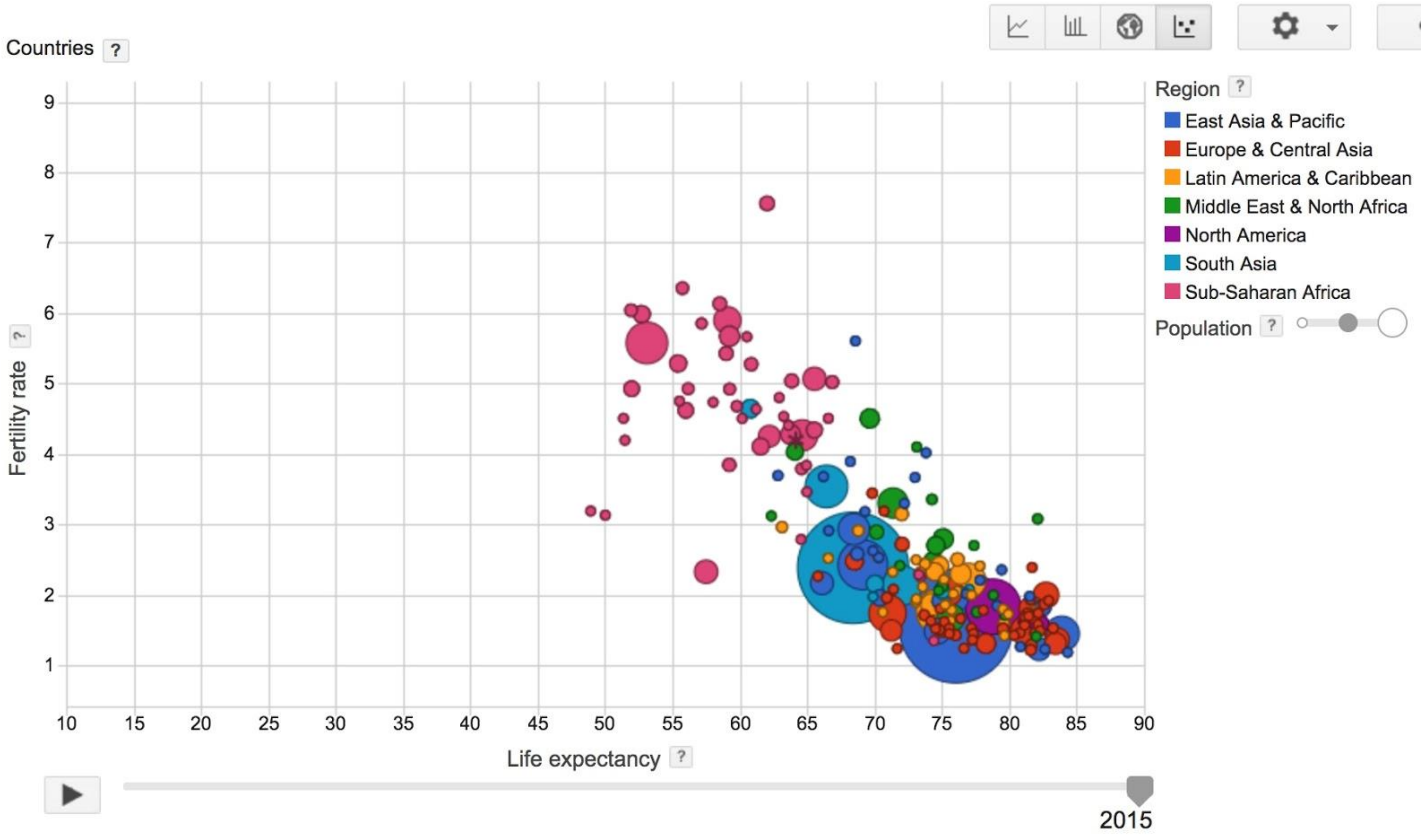
Linear Regression in Tensorflow

Tensorflow | Linear Regression

X = birth rate(출생률)
Y = 기대 수명
190개국

$$Y_{predict} = w * x + b$$

$$\text{Loss} = E[(Y - Y_{predict})^2]$$



Visualization made by Google, based on data collected by World Bank

출처: https://docs.google.com/presentation/d/1lmcQVNAmJrL8x3Iq0VB1mVaka1r6pOIb-TMVTX5Rufc/edit#slide=id.g200a43678_0_1

Tensorflow | Linear Regression

첫번째 단계: Assemble graph

- step1: data 불러오기
- step2: data와 labels을 위한 Placeholder( ) 생성
- step3: weight와 bias 를 생성
- step4: Inference 정의
- step5: Loss function을 정의
- step6: Optimizer 생성

Tensorflow | Linear Regression

첫번째 단계: Assemble graph

- **step1: data 불러오기**
- step2: data와 labels을 위한 Placeho
- step3: weight와 bias 를 생성
- step4: Inference 정의
- step5: Loss function을 정의
- step6: Optimizer 생성

```python
import tensorflow as tf
import utils

DATA_FILE = 'data/birth_life_2010.txt'

# Step 1: read in data from the .txt file
data, n_samples = utils.read_birth_life_data(DATA_FILE)

def read_birth_life_data(filename):
    """
    Read in birth_life_2010.txt and return:
    data in the form of NumPy array
    n_samples: number of samples
    """
    text = open(filename, 'r').readlines()[1:]
    data = [line[:-1].split('\t') for line in text]
    births = [float(line[1]) for line in data]
    lifes = [float(line[2]) for line in data]
    data = list(zip(births, lifes))
    n_samples = len(data)
    data = np.asarray(data, dtype=np.float32)
    return data, n_samples
```

Tensorflow | Linear Regression

첫번째 단계: Assemble graph

- step1: data 불러오기
- step2: data와 labels을 위한 Placeholder( ) 생성
- step3: weight와 bias 를 생성
- step4: Inference 정의
- step5: Loss function을 정의
- step6: Optimizer 생성

```
import tensorflow as tf
import utils

DATA_FILE = 'data/birth_life_2010.txt'

# Step 1: read in data from the .txt file
data, n_samples = utils.read_birth_life_data(DATA_FILE)
```

```
# Step 2: create placeholders for X (birth rate) and Y (life expectancy)
X = tf.placeholder(tf.float32, name='X')
Y = tf.placeholder(tf.float32, name='Y')
```

Tensorflow | Linear Regression

첫번째 단계: Assemble graph

- step1: data 불러오기
- step2: data와 labels을 위한 Pla
- step3: weight와 bias 를 생성
- step4: Inference 정의
- step5: Loss function을 정의
- step6: Optimizer 생성

```python
import tensorflow as tf
import utils

DATA_FILE = 'data/birth_life_2010.txt'

# Step 1: read in data from the .txt file
data, n_samples = utils.read_birth_life_data(DATA_FILE)

# Step 2: create placeholders for X (birth rate) and Y (life expectancy)
X = tf.placeholder(tf.float32, name='X')
Y = tf.placeholder(tf.float32, name='Y')

# Step 3: create weight and bias, initialized to 0
w = tf.get_variable('weights', initializer=tf.constant(0.0))
b = tf.get_variable('bias', initializer=tf.constant(0.0))
```

Tensorflow | Linear Regression

첫번째 단계: Assemble graph

- step1: data 불러오기
- step2: data와 labels을 위한 Pla
- step3: weight와 bias 를 생성
- step4: Inference 정의
- step5: Loss function을 정의
- step6: Optimizer 생성

```python
import tensorflow as tf
import utils

DATA_FILE = 'data/birth_life_2010.txt'

# Step 1: read in data from the .txt file
data, n_samples = utils.read_birth_life_data(DATA_FILE)

# Step 2: create placeholders for X (birth rate) and Y (life expectancy)
X = tf.placeholder(tf.float32, name='X')
Y = tf.placeholder(tf.float32, name='Y')

# Step 3: create weight and bias, initialized to 0
w = tf.get_variable('weights', initializer=tf.constant(0.0))
b = tf.get_variable('bias', initializer=tf.constant(0.0))
# Step 4: build model to predict Y
Y_predicted = w * X + b
```

Tensorflow | Linear Regression

첫번째 단계: Assemble graph

- step1: data 불러오기
- step2: data와 labels을 위한 Pla
- step3: weight와 bias 를 생성
- step4: Inference 정의
- step5: Loss function을 정의
- step6: Optimizer 생성

```python
import tensorflow as tf
import utils

DATA_FILE = 'data/birth_life_2010.txt'

# Step 1: read in data from the .txt file
data, n_samples = utils.read_birth_life_data(DATA_FILE)

# Step 2: create placeholders for X (birth rate) and Y (life expectancy)
X = tf.placeholder(tf.float32, name='X')
Y = tf.placeholder(tf.float32, name='Y')

# Step 3: create weight and bias, initialized to 0
w = tf.get_variable('weights', initializer=tf.constant(0.0))
b = tf.get_variable('bias', initializer=tf.constant(0.0))
# Step 4: build model to predict Y
Y_predicted = w * X + b

# Step 5: use the squared error as the loss function
loss = tf.square(Y - Y_predicted, name='loss')
```

Tensorflow | Linear Regression

첫번째 단계: Assemble graph

- step1: data 불러오기
- step2: data와 labels을 위한 Pla
- step3: weight와 bias 를 생성
- step4: Inference 정의
- step5: Loss function을 정의
- step6: Optimizer 생성

```python
import tensorflow as tf
import utils

DATA_FILE = 'data/birth_life_2010.txt'

# Step 1: read in data from the .txt file
data, n_samples = utils.read_birth_life_data(DATA_FILE)

# Step 2: create placeholders for X (birth rate) and Y (life expectancy)
X = tf.placeholder(tf.float32, name='X')
Y = tf.placeholder(tf.float32, name='Y')

# Step 3: create weight and bias, initialized to 0
w = tf.get_variable('weights', initializer=tf.constant(0.0))
b = tf.get_variable('bias', initializer=tf.constant(0.0))
# Step 4: build model to predict Y
Y_predicted = w * X + b

# Step 5: use the squared error as the loss function
loss = tf.square(Y - Y_predicted, name='loss')
# Step 6: using gradient descent with learning rate of 0.001 to minimize loss
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.001).minimize(loss)
```

Tensorflow | Linear Regression

두번째 단계: Train model

```
start = time.time()
writer = tf.summary.FileWriter('./graphs/linear_reg', tf.get_default_graph())
with tf.Session() as sess:
    # Step 7: initialize the necessary variables, in this case, w and b
    sess.run(tf.global_variables_initializer())
```

- step1: variables 초기화
- step2: optimizer 실행

세번째 단계: See it on Tensorboard

- step1: tensorboard –logdir= ‘./graph’

Tensorflow | Linear Regression

두번째 단계: Train model

- step1: variables 초기화
- step2: optimizer 실행

```
start = time.time()
writer = tf.summary.FileWriter('./graphs/linear_reg', tf.get_default_graph())
with tf.Session() as sess:
    # Step 7: initialize the necessary variables, in this case, w and b
    sess.run(tf.global_variables_initializer())
    # Step 8: train the model for 100 epochs
    for i in range(100):
        total_loss = 0
        for x, y in data:
            # Session execute optimizer and fetch values of loss
            _, l = sess.run([optimizer, loss], feed_dict={X: x, Y:y})
            total_loss += l
        print('Epoch {0}: {1}'.format(i, total_loss/n_samples))
```

세번째 단계: See it on Tensorboard

- step1: tensorboard –logdir= ‘./graph’

Tensorflow | Linear Regression

```python
import tensorflow as tf
import utils

DATA_FILE = 'data/birth_life_2010.txt'

# Step 1: read in data from the .txt file
data, n_samples = utils.read_birth_life_data(DATA_FILE)

# Step 2: create placeholders for X (birth rate) and Y (life expectancy)
X = tf.placeholder(tf.float32, name='X')
Y = tf.placeholder(tf.float32, name='Y')

# Step 3: create weight and bias, initialized to 0
w = tf.get_variable('weights', initializer=tf.constant(0.0))
b = tf.get_variable('bias', initializer=tf.constant(0.0))
# Step 4: build model to predict Y
Y_predicted = w * X + b

# Step 5: use the squared error as the loss function
loss = tf.square(Y - Y_predicted, name='loss')
# Step 6: using gradient descent with learning rate of 0.001 to minimize loss
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.001).minimize(loss)
start = time.time()
writer = tf.summary.FileWriter('./graphs/linear_reg', tf.get_default_graph())
with tf.Session() as sess:
    # Step 7: initialize the necessary variables, in this case, w and b
    sess.run(tf.global_variables_initializer())
    # Step 8: train the model for 100 epochs
    for i in range(100):
        total_loss = 0
        for x, y in data:
            # Session execute optimizer and fetch values of loss
            _, l = sess.run([optimizer, loss], feed_dict={X: x, Y:y})
            total_loss += l
        print('Epoch {0}: {1}'.format(i, total_loss/n_samples))
```

```python
import tensorflow as tf
import utils

DATA_FILE = 'data/birth_life_2010.txt'

# Step 1: read in the data
data, n_samples = utils.read_birth_life_data(DATA_FILE)

# Step 2: create Dataset and iterator
dataset = tf.data.Dataset.from_tensor_slices((data[:,0], data[:,1]))
iterator = dataset.make_initializable_iterator()
X, Y = iterator.get_next()
# Step 3: create weight and bias, initialized to 0
w = tf.get_variable('weights', initializer=tf.constant(0.0))
b = tf.get_variable('bias', initializer=tf.constant(0.0))
# Step 4: build model to predict Y
Y_predicted = X * w + b
# Step 5: use the square error as the loss function
loss = tf.square(Y - Y_predicted, name='loss')
# Step 6: using gradient descent with learning rate of 0.001 to minimize loss
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.001).minimize(loss)
start = time.time()
with tf.Session() as sess:
    # Step 7: initialize the necessary variables, in this case, w and b
    sess.run(tf.global_variables_initializer())
    writer = tf.summary.FileWriter('./graphs/linear_reg', sess.graph)
    # Step 8: train the model for 100 epochs
    for i in range(100):
        sess.run(iterator.initializer) # initialize the iterator
        total_loss = 0
        try:
            while True:
                _, l = sess.run([optimizer, loss])
                total_loss += l
        except tf.errors.OutOfRangeError:
            pass

        print('Epoch {0}: {1}'.format(i, total_loss/n_samples))
    # close the writer when you're done using it
    writer.close()
```

Tensorflow | Linear Regression

```python
import tensorflow as tf
import utils

DATA_FILE = 'data/birth_life_2010.txt'

# Step 1: read in data from the .txt file
data, n_samples = utils.read_birth_life_data(DATA_FILE)

# Step 2: create placeholders for X (birth rate) and Y (life expectancy)
X = tf.placeholder(tf.float32, name='X')
Y = tf.placeholder(tf.float32, name='Y')

# Step 3: create weight and bias, initialized to 0
w = tf.get_variable('weights', initializer=tf.constant(0.0))
b = tf.get_variable('bias', initializer=tf.constant(0.0))
# Step 4: build model to predict Y
Y_predicted = w * X + b

# Step 5: use the squared error as the loss function
loss = tf.square(Y - Y_predicted, name='loss')
# Step 6: using gradient descent with learning rate of 0.001 to minimize loss
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.001).minimize(loss)
start = time.time()
writer = tf.summary.FileWriter('./graphs/linear_reg', tf.get_default_graph())
with tf.Session() as sess:
    # Step 7: initialize the necessary variables, in this case, w and b
    sess.run(tf.global_variables_initializer())
    # Step 8: train the model for 100 epochs
    for i in range(100):
        total_loss = 0
        for x, y in data:
            # Session execute optimizer and fetch values of loss
            _, l = sess.run([optimizer, loss], feed_dict={X: x, Y:y})
            total_loss += l
        print('Epoch {0}: {1}'.format(i, total_loss/n_samples))
```

```python
import tensorflow as tf
import utils

DATA_FILE = 'data/birth_life_2010.txt'

# Step 1: read in the data
data, n_samples = utils.read_birth_life_data(DATA_FILE)

# Step 2: create Dataset and iterator
dataset = tf.data.Dataset.from_tensor_slices((data[:,0], data[:,1]))
iterator = dataset.make_initializable_iterator()
X, Y = iterator.get_next()
# Step 3: create weight and bias, initialized to 0
w = tf.get_variable('weights', initializer=tf.constant(0.0))
b = tf.get_variable('bias', initializer=tf.constant(0.0))
# Step 4: build model to predict Y
Y_predicted = X * w + b
# Step 5: use the square error as the loss function
loss = tf.square(Y - Y_predicted, name='loss')
# Step 6: using gradient descent with learning rate of 0.001 to minimize loss
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.001).minimize(loss)
start = time.time()
with tf.Session() as sess:
    # Step 7: initialize the necessary variables, in this case, w and b
    sess.run(tf.global_variables_initializer())
    writer = tf.summary.FileWriter('./graphs/linear_reg', sess.graph)
    # Step 8: train the model for 100 epochs
    for i in range(100):
        sess.run(iterator.initializer) # initialize the iterator
        total_loss = 0
        try:
            while True:
                _, l = sess.run([optimizer, loss])
                total_loss += l
        except tf.errors.OutOfRangeError:
            pass

        print('Epoch {0}: {1}'.format(i, total_loss/n_samples))
    # close the writer when you're done using it
    writer.close()
```

Tensorflow | Linear Regression

Performance better?

With placeholder: 7.05271519 seconds

With tf.data: 6.12285947 seconds

Logistic Regression in Tensorflow

Tensorflow | Logistic Regression

MNIST Dataset

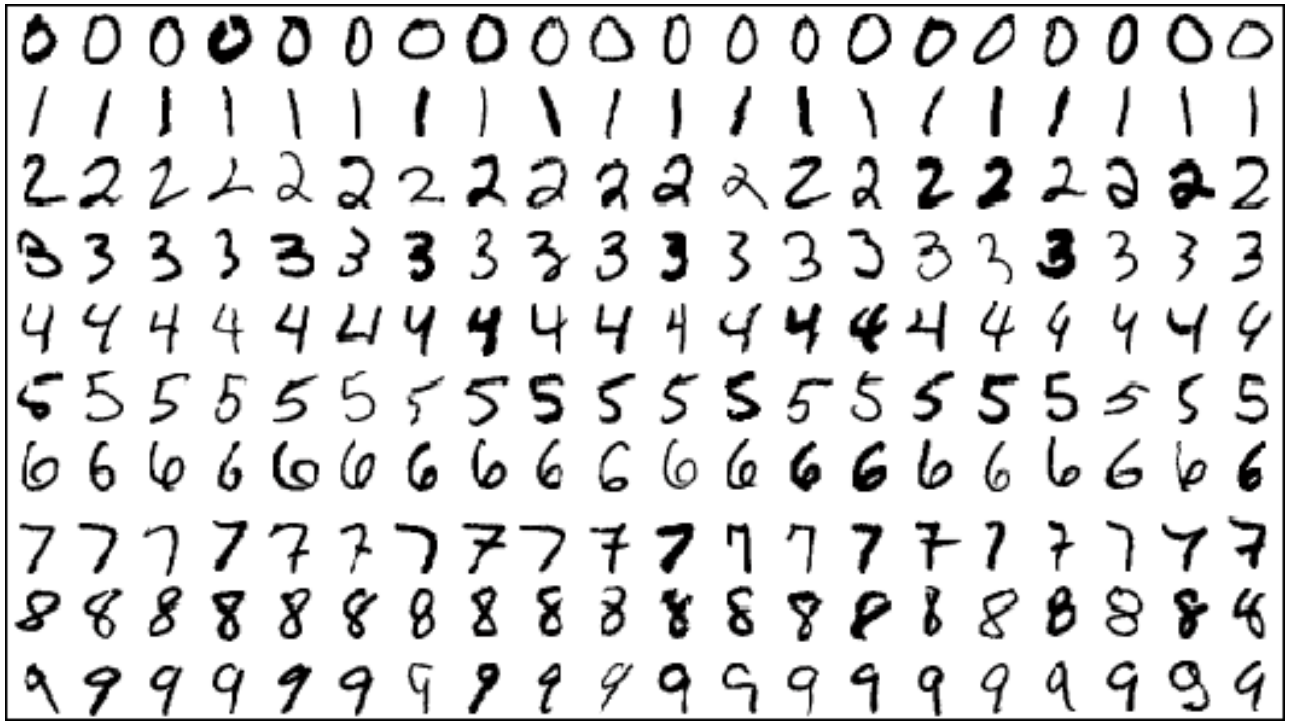
X = 28x28 array(1d size of 784)

Y = Digit value

MNIST.train: 55,000 examples
MNIST.validation: 5,000 examples
MNIST.test: 10,000 examples

Inference: Y_predicted = softmax(X * w + b)

Cross entropy loss: −log(Y_predicted)



출처: https://docs.google.com/presentation/d/1lmcQVNAmJrL8x3Iq0VB1mVaka1r6pOIb-TMVTX5Rufc/edit#slide=id.g1c166da651_0_5

Tensorflow | Logistic Regression

첫번째 단계: Assemble graph

- **step1: data 불러오기**
- step2: Placeholder( ) 생성
- step3: weight와 bias 를 생성
- step4: Inference 정의
- step5: Loss function을 정의
- step6: Optimizer 생성
- step7: 테스트 정확도 측정

```
# Step 1: Read in data
mnist = input_data.read_data_sets('data/mnist', one_hot=True)
X_batch, Y_batch = mnist.train.next_batch(128)
```

Tensorflow | Logistic Regression

첫번째 단계: Assemble graph

- step1: data 불러오기
- step2: Placeholder( ) 생성
- step3: weight와 bias 를 생성
- step4: Inference 정의
- step5: Loss function을 정의
- step6: Optimizer 생성
- step7: 테스트 정확도 측정

```
# Step 1: Read in data
mnist = input_data.read_data_sets('data/mnist', one_hot=True)
X_batch, Y_batch = mnist.train.next_batch(128)

# Step 2: create placeholders for features and labels
X = tf.placeholder(tf.float32, [128, 784], name='image')
Y = tf.placeholder(tf.int32, [128, 10], name='label')
```

Tensorflow | Logistic Regression

첫번째 단계: Assemble graph

- step1: data 불러오기
- step2: Placeholder( ) 생성
- step3: weight와 bias 를 생성
- step4: Inference 정의
- step5: Loss function을 정의
- step6: Optimizer 생성
- step7: 테스트 정확도 측정

```
# Step 1: Read in data
mnist = input_data.read_data_sets('data/mnist', one_hot=True)
X_batch, Y_batch = mnist.train.next_batch(128)

# Step 2: create placeholders for features and labels
X = tf.placeholder(tf.float32, [128, 784], name='image')
Y = tf.placeholder(tf.int32, [128, 10], name='label')

# Step 3: create weights and bias
w = tf.get_variable(name='weights', shape=(784, 10), initializer=tf.random_normal_initializer())
b = tf.get_variable(name='bias', shape=(1, 10), initializer=tf.zeros_initializer())
```

Tensorflow | Logistic Regression

첫번째 단계: Assemble graph

- step1: data 불러오기
- step2: Placeholder( ) 생성
- step3: weight와 bias 를 생성
- step4: Inference 정의
- step5: Loss function을 정의
- step6: Optimizer 생성
- step7: 테스트 정확도 측정

```
# Step 1: Read in data
mnist = input_data.read_data_sets('data/mnist', one_hot=True)
X_batch, Y_batch = mnist.train.next_batch(128)

# Step 2: create placeholders for features and labels
X = tf.placeholder(tf.float32, [128, 784], name='image')
Y = tf.placeholder(tf.int32, [128, 10], name='label')

# Step 3: create weights and bias
w = tf.get_variable(name='weights', shape=(784, 10), initializer=tf.random_normal_initializer())
b = tf.get_variable(name='bias', shape=(1, 10), initializer=tf.zeros_initializer())

# Step 4: build model
logits = tf.matmul(X, w) + b
```

Tensorflow | Logistic Regression

첫번째 단계: Assemble graph

- step1: data 불러오기
- step2: Placeholder( ) 생성
- step3: weight와 bias 를 생성
- step4: Inference 정의
- step5: Loss function을 정의
- step6: Optimizer 생성
- step7: 테스트 정확도 측정

```
# Step 1: Read in data
mnist = input_data.read_data_sets('data/mnist', one_hot=True)
X_batch, Y_batch = mnist.train.next_batch(128)

# Step 2: create placeholders for features and labels
X = tf.placeholder(tf.float32, [128, 784], name='image')
Y = tf.placeholder(tf.int32, [128, 10], name='label')

# Step 3: create weights and bias
w = tf.get_variable(name='weights', shape=(784, 10), initializer=tf.random_normal_initializer())
b = tf.get_variable(name='bias', shape=(1, 10), initializer=tf.zeros_initializer())

# Step 4: build model
logits = tf.matmul(X, w) + b

# Step 5: define loss function
entropy = tf.nn.softmax_cross_entropy_with_logits(logits=logits, labels=Y, name='loss')
loss = tf.reduce_mean(entropy) # computes the mean over all the examples in the batch
```

Tensorflow | Logistic Regression

첫번째 단계: Assemble graph

- step1: data 불러오기
- step2: Placeholder( ) 생성
- step3: weight와 bias 를 생성
- step4: Inference 정의
- step5: Loss function을 정의
- step6: Optimizer 생성
- step7: 테스트 정확도 측정

```
# Step 1: Read in data
mnist = input_data.read_data_sets('data/mnist', one_hot=True)
X_batch, Y_batch = mnist.train.next_batch(128)

# Step 2: create placeholders for features and labels
X = tf.placeholder(tf.float32, [128, 784], name='image')
Y = tf.placeholder(tf.int32, [128, 10], name='label')

# Step 3: create weights and bias
w = tf.get_variable(name='weights', shape=(784, 10), initializer=tf.random_normal_initializer())
b = tf.get_variable(name='bias', shape=(1, 10), initializer=tf.zeros_initializer())

# Step 4: build model
logits = tf.matmul(X, w) + b

# Step 5: define loss function
entropy = tf.nn.softmax_cross_entropy_with_logits(logits=logits, labels=Y, name='loss')
loss = tf.reduce_mean(entropy) # computes the mean over all the examples in the batch

# Step 6: define training op
# using gradient descent with learning rate of 0.01 to minimize loss
optimizer = tf.train.AdamOptimizer(0.01).minimize(loss)
```

Tensorflow | Logistic Regression

첫번째 단계: Assemble graph

- step1: data 불러오기
- step2: Placeholder( ) 생성
- step3: weight와 bias 를 생성
- step4: Inference 정의
- step5: Loss function을 정의
- step6: Optimizer 생성
- step7: 테스트 정확도 측정

```
# Step 1: Read in data
mnist = input_data.read_data_sets('data/mnist', one_hot=True)
X_batch, Y_batch = mnist.train.next_batch(128)

# Step 2: create placeholders for features and labels
X = tf.placeholder(tf.float32, [128, 784], name='image')
Y = tf.placeholder(tf.int32, [128, 10], name='label')

# Step 3: create weights and bias
w = tf.get_variable(name='weights', shape=(784, 10), initializer=tf.random_normal_initializer())
b = tf.get_variable(name='bias', shape=(1, 10), initializer=tf.zeros_initializer())

# Step 4: build model
logits = tf.matmul(X, w) + b

# Step 5: define loss function
entropy = tf.nn.softmax_cross_entropy_with_logits(logits=logits, labels=Y, name='loss')
loss = tf.reduce_mean(entropy) # computes the mean over all the examples in the batch

# Step 6: define training op
# using gradient descent with learning rate of 0.01 to minimize loss
optimizer = tf.train.AdamOptimizer(0.01).minimize(loss)

# Step 7: calculate accuracy with test set
preds = tf.nn.softmax(logits)
correct_preds = tf.equal(tf.argmax(preds, 1), tf.argmax(Y, 1))
accuracy = tf.reduce_sum(tf.cast(correct_preds, tf.float32))
```

Tensorflow | Logistic Regression

두번째 단계: Train model

- step1: variables 초기화
- step2: optimizer 실행

```
writer = tf.summary.FileWriter('./graphs/logreg_placeholder', tf.get_default_graph())
with tf.Session() as sess:
  start_time = time.time()
  sess.run(tf.global_variables_initializer())
  n_batches = int(mnist.train.num_examples/128)

  # train the model n_epochs times
  for i in range(30):
    total_loss = 0

    for j in range(n_batches):
      X_batch, Y_batch = mnist.train.next_batch(128)
      _, loss_batch = sess.run([optimizer, loss], {X: X_batch, Y:Y_batch})
      total_loss += loss_batch
    print('Average loss epoch {0}: {1}'.format(i, total_loss/n_batches))
  print('Total time: {0} seconds'.format(time.time() - start_time))
```

세번째 단계: Test model

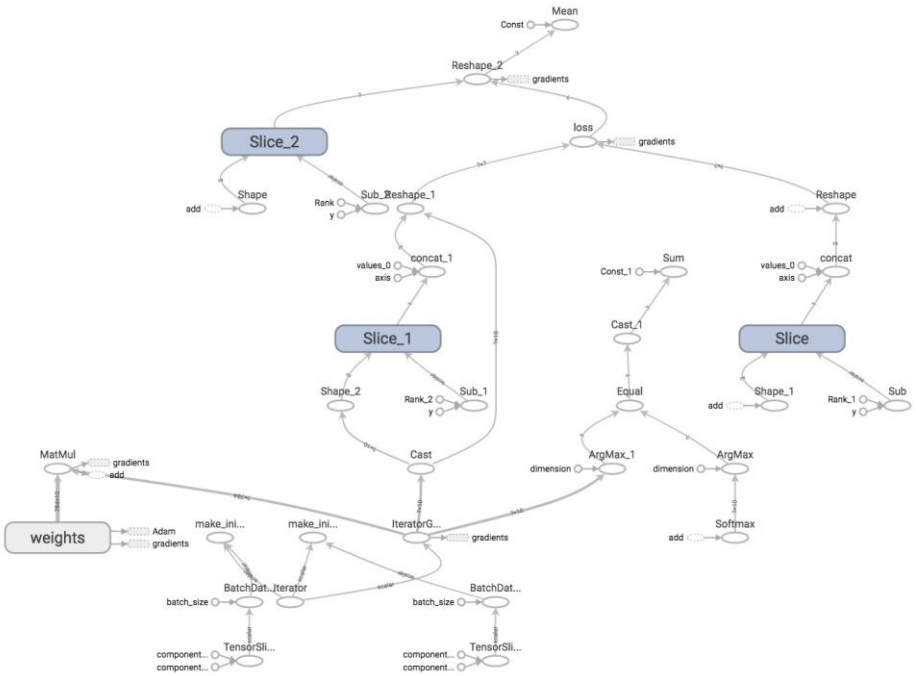
- step1:정확도 측정

```
  # test the model
  n_batches = int(mnist.test.num_examples/128)
  total_correct_preds = 0

  for i in range(n_batches):
    X_batch, Y_batch = mnist.test.next_batch(128)
    accuracy_batch = sess.run(accuracy, {X: X_batch, Y:Y_batch})
    total_correct_preds += accuracy_batch

  print('Accuracy {0}'.format(total_correct_preds/mnist.test.num_examples))
writer.close()
```

Tensorflow | Logistic Regression

네번째 단계: Tensorboard

- step1: tensorboard –logdir= ‘./graph

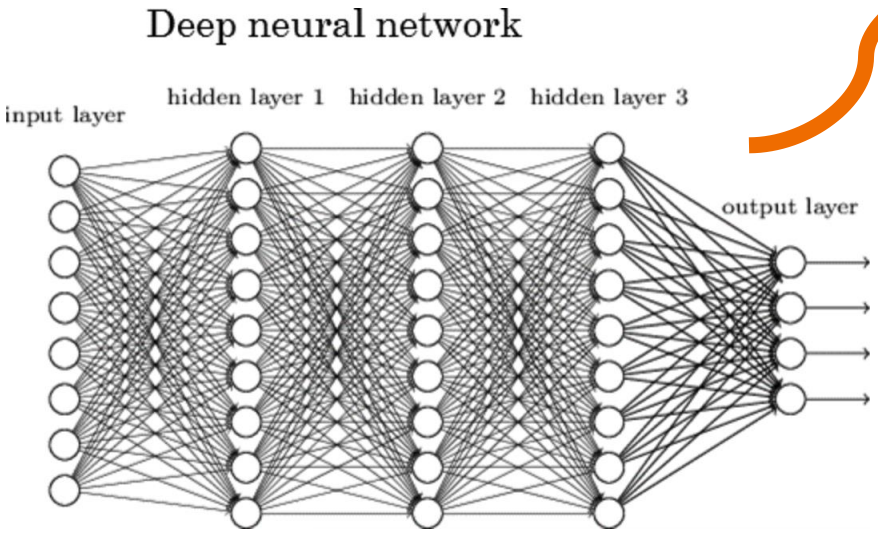


Tensorflow | tf.layers

```python
# Step 3: create weights and bias
w = tf.get_variable(name='weights', shape=(784, 10), initializer=tf.random_normal_initializer())
b = tf.get_variable(name='bias', shape=(1, 10), initializer=tf.zeros_initializer())

# Step 4: build model
logits = tf.matmul(X, w) + b
```

```python
# Step 3 + Step 4: build model with wrapper
logits = tf.layers.dense(img,
                         units=10,
                         activation=None,
                         use_bias=True,
                         kernel_initializer=tf.initializers.variance_scaling(),
                         bias_initializer=tf.initializers.zeros(),
                         trainable=True,
                         name='logits',
                         reuse=None)
```

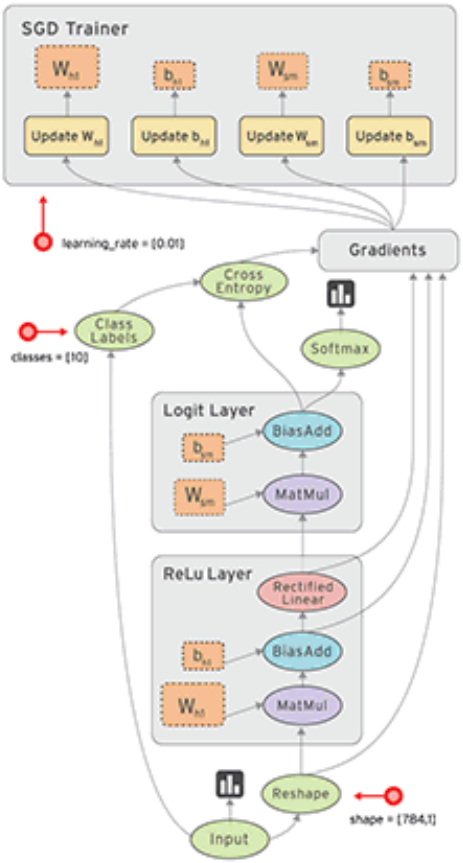
Tensorflow | tf.layers




```python
mlp1 = tf.layers.dense(img,
                       units=32,
                       activation=tf.nn.relu,
                       use_bias=True,
                       kernel_initializer=tf.initializers.variance_scaling,
                       bias_initializer=tf.initializers.zeros,
                       trainable=True,
                       name='fc1',
                       reuse=None)
mlp2 = tf.layers.dense(mlp1,
                       units=64,
                       activation=None,
                       use_bias=True,
                       kernel_initializer=tf.initializers.variance_scaling,
                       bias_initializer=tf.initializers.zeros,
                       trainable=True,
                       name='fc2',
                       reuse=None)
mlp3 = tf.layers.dense(mlp1,
                       units=64,
                       activation=None,
                       use_bias=True,
                       kernel_initializer=tf.initializers.variance_scaling,
                       bias_initializer=tf.initializers.zeros,
                       trainable=True,
                       name='fc3',
                       reuse=None)
logits = tf.layers.dense(mlp3,
                       units=10,
                       activation=None,
                       use_bias=True,
                       kernel_initializer=tf.initializers.variance_scaling(),
                       bias_initializer=tf.initializers.zeros(),
                       trainable=True,
                       name='logits',
                       reuse=None)
```

Tensorflow | 요약

- Lazy loading 문제!
- Linear Regression 모델 생성 및 학습
- Logistic Regression 모델 생성 및 학습
- tf.placeholder( ) version vs tf.data( ) version 비교
- tf.layers 를 이용하여 다층 퍼셉트론 모델 생성



To be continued..