LAB-3: Deep Neural Network

Yu YANG

csyyang@comp.polyu.edu.hk

Outline

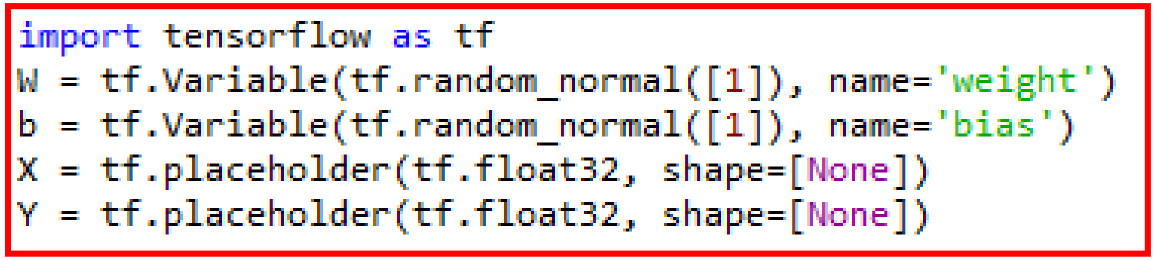
- Review Lab 1 & Lab 2
  - Solution of Linear Regression
  - Solution of Logistic Regression
- Implement Deep Neural Network using TensorFlow
- Take-home exercise & report

Review Lab1: Linear Regression

- Given the training data set X is [1,2,3], Y is [4,5,6]
- The relation of X and Y is Y= WX+b
- Please build a graph with tensorflow to predict Y, given X=10.

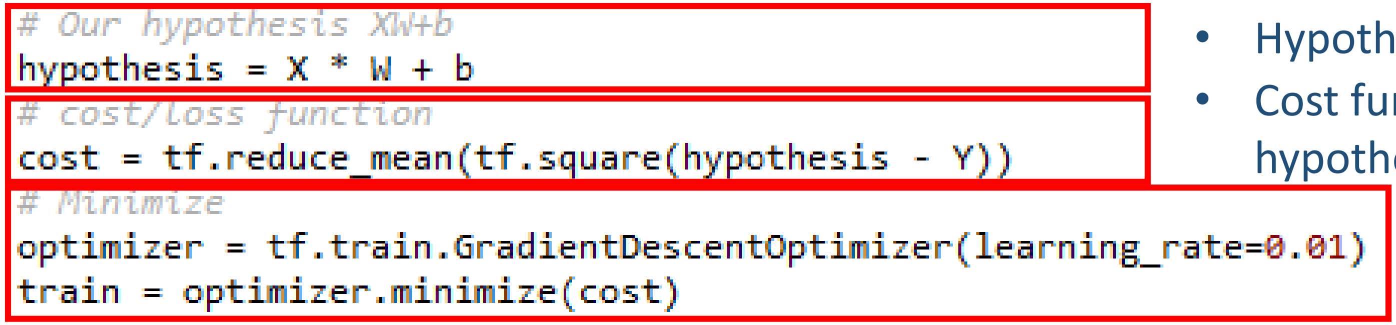
Our goal is to train W and b using given training set X and Y

Solution of Lab1



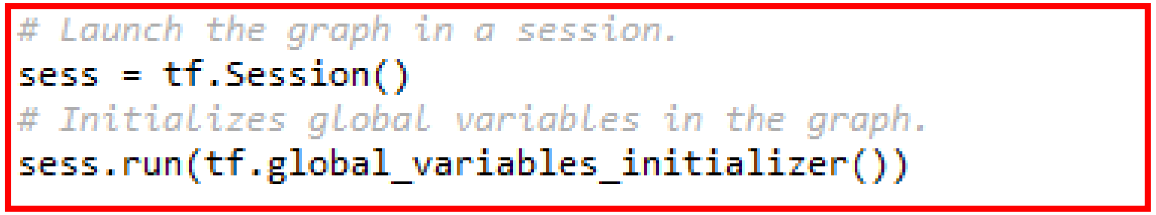
```python
import tensorflow as tf
W = tf.Variable(tf.random_normal([1]), name='weight')
b = tf.Variable(tf.random_normal([1]), name='bias')
X = tf.placeholder(tf.float32, shape=[None])
Y = tf.placeholder(tf.float32, shape=[None])
```

- Import tensorflow and declare variables and placeholder.
- The parameter which is needed to be trained should be declared as variable.
- The training features and target values could be declared as placeholder and pass the data later in training.



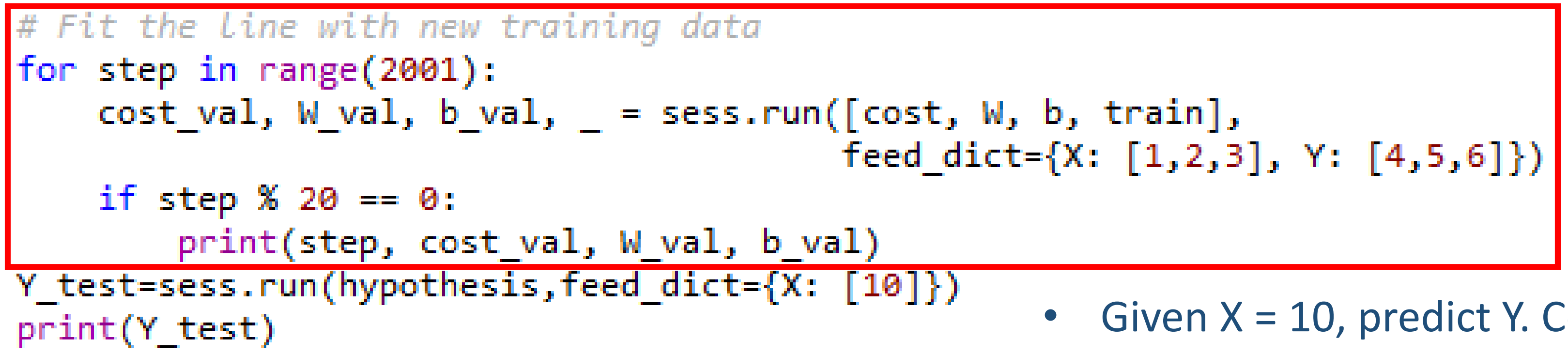
```python
# Our hypothesis XW+b
hypothesis = X * W + b
# cost/loss function
cost = tf.reduce_mean(tf.square(hypothesis - Y))
# Minimize
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
train = optimizer.minimize(cost)
```

- Hypothesis of linear regression H = Wx + b
- Cost function which minimize the different between hypothesis and target values
- Use gradient descent to train W and b



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

- Launch the graph in a session



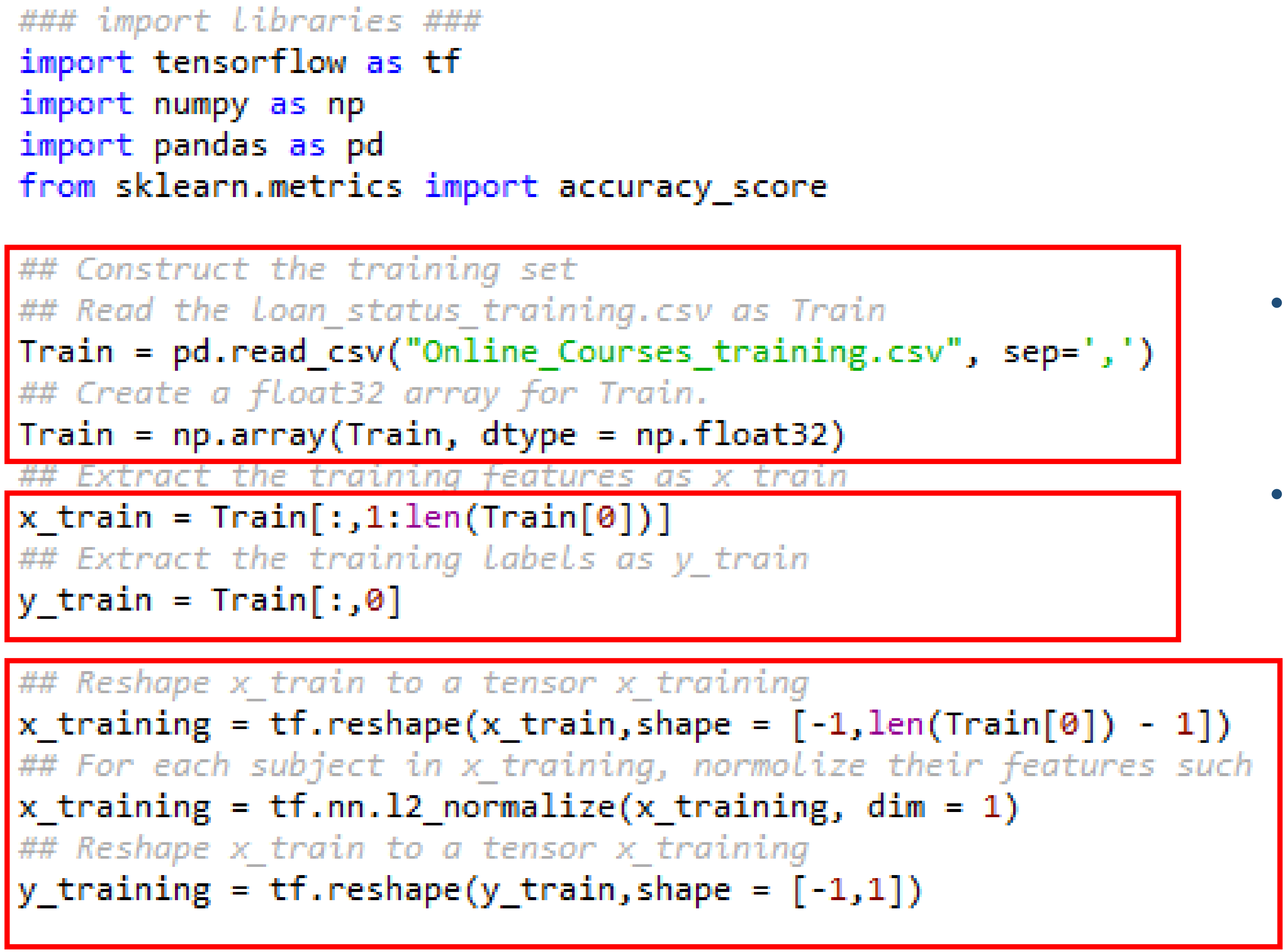
```python
# Fit the line with new training data
for step in range(2001):
    cost_val, W_val, b_val, _ = sess.run([cost, W, b, train],
                                         feed_dict={X: [1,2,3], Y: [4,5,6]})
    if step % 20 == 0:
        print(step, cost_val, W_val, b_val)
Y_test=sess.run(hypothesis,feed_dict={X: [10]})
print(Y_test)
```

- Pass the training data to the training process
- Given X = 10, predict Y. Correct answer Y = 13.0*****

Review Lab2: Logistic Regression

- Predict the institution with given online course features.
  - MIT is class “1” and Stanford class “0”
- Fill in the missing part of logistic_regression.py
  1. Load the training and testing data.
  2. Construct feature set (from the second column to the last column in CSV) and label set (the first column in CSV).
  3. Fill in the sigmoid function.
  4. Fill in the loss function.
  5. Use the gradient descent to train the parameter θ. Please use the optimizer in Tensorflow.
  6. Predict the institution of testing samples.
  7. Print the prediction accuracy.

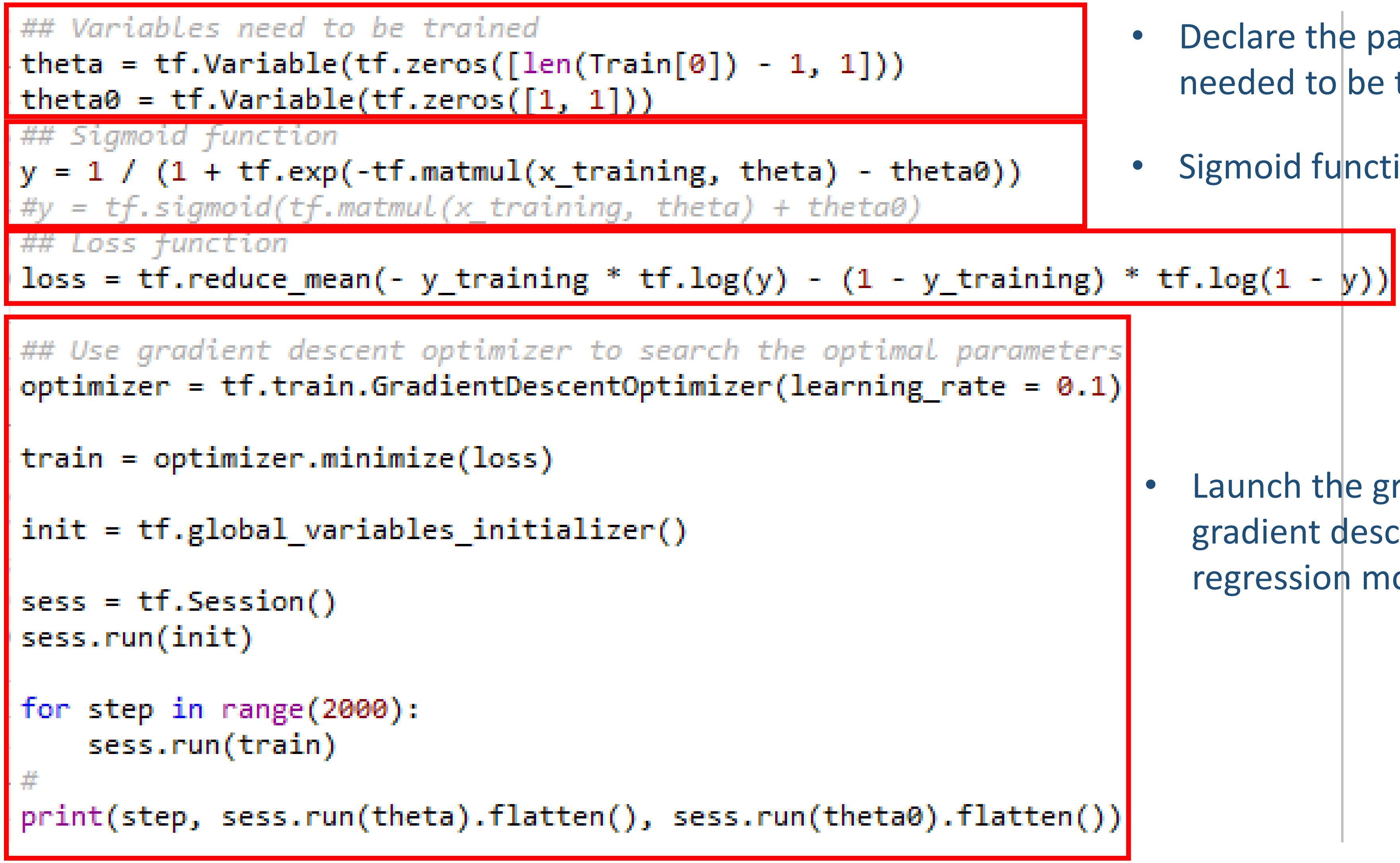
Solution of Lab2



```
### import libraries ###
import tensorflow as tf
import numpy as np
import pandas as pd
from sklearn.metrics import accuracy_score

## Construct the training set
## Read the loan_status_training.csv as Train
Train = pd.read_csv("Online_Courses_training.csv", sep=',')
## Create a float32 array for Train.
Train = np.array(Train, dtype = np.float32)
## Extract the training features as x_train
x_train = Train[:,1:len(Train[0])]
## Extract the training labels as y_train
y_train = Train[:,0]

## Reshape x_train to a tensor x_training
x_training = tf.reshape(x_train,shape = [-1,len(Train[0]) - 1])
## For each subject in x_training, normalize their features such
x_training = tf.nn.l2_normalize(x_training, dim = 1)
## Reshape x_train to a tensor x_training
y_training = tf.reshape(y_train,shape = [-1,1])
```

- Read the csv data using pandas
- Construction training set including training features and labels
- Reshape the training features and labels as tensor and normalize the training features

Solution of Lab2

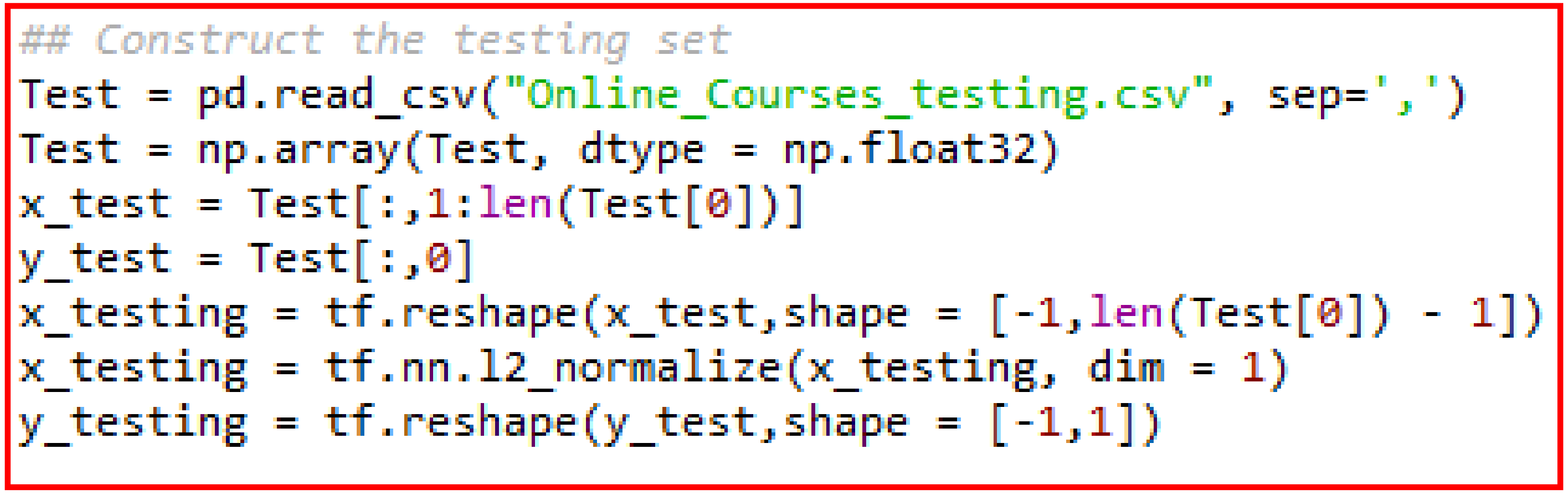


```
## Variables need to be trained
theta = tf.Variable(tf.zeros([len(Train[0]) - 1, 1]))
theta0 = tf.Variable(tf.zeros([1, 1]))
## Sigmoid function
y = 1 / (1 + tf.exp(-tf.matmul(x_training, theta) - theta0))
#y = tf.sigmoid(tf.matmul(x_training, theta) + theta0)
## Loss function
loss = tf.reduce_mean(- y_training * tf.log(y) - (1 - y_training) * tf.log(1 - y))
```

```
## Use gradient descent optimizer to search the optimal parameters
optimizer = tf.train.GradientDescentOptimizer(learning_rate = 0.1)

train = optimizer.minimize(loss)

init = tf.global_variables_initializer()

sess = tf.Session()
sess.run(init)

for step in range(2000):
    sess.run(train)
#
print(step, sess.run(theta).flatten(), sess.run(theta0).flatten())
```

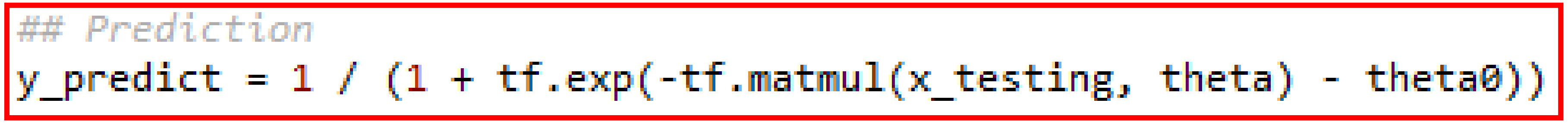
- Declare the parameter which is needed to be trained.
- Sigmoid function
- Loss function
- Launch the graph in a session and use gradient descent to train the logistic regression model

Solution of Lab2



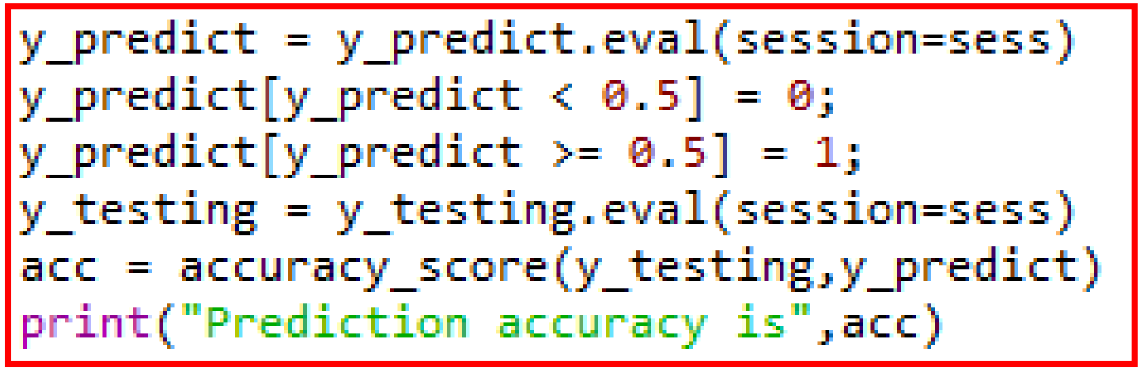
```
## Construct the testing set
Test = pd.read_csv("Online_Courses_testing.csv", sep=',')
Test = np.array(Test, dtype = np.float32)
x_test = Test[:,1:len(Test[0])]
y_test = Test[:,0]
x_testing = tf.reshape(x_test,shape = [-1,len(Test[0]) - 1])
x_testing = tf.nn.l2_normalize(x_testing, dim = 1)
y_testing = tf.reshape(y_test,shape = [-1,1])
```

- Construct the testing set



```
## Prediction
y_predict = 1 / (1 + tf.exp(-tf.matmul(x_testing, theta) - theta0))
```

- Prediction using testing features



```
y_predict = y_predict.eval(session=sess)
y_predict[y_predict < 0.5] = 0;
y_predict[y_predict >= 0.5] = 1;
y_testing = y_testing.eval(session=sess)
acc = accuracy_score(y_testing,y_predict)
print("Prediction accuracy is",acc)
```

- Evaluate the prediction accuracy

Review Lab2 Take Home Exercise

- Plot the convergence curve of lose function and use a table to show the prediction accuracy as iteration increases with iteration = 100, 300, 500,1000, 3000, 5000 (Learning rate sets to 0.1).
- Set iteration = 3000, compare the convergence curve of lose function and the prediction accuracy with learning rate sets to 0.1, 0.01, 0.001, then discuss your experience in selecting learning rate.

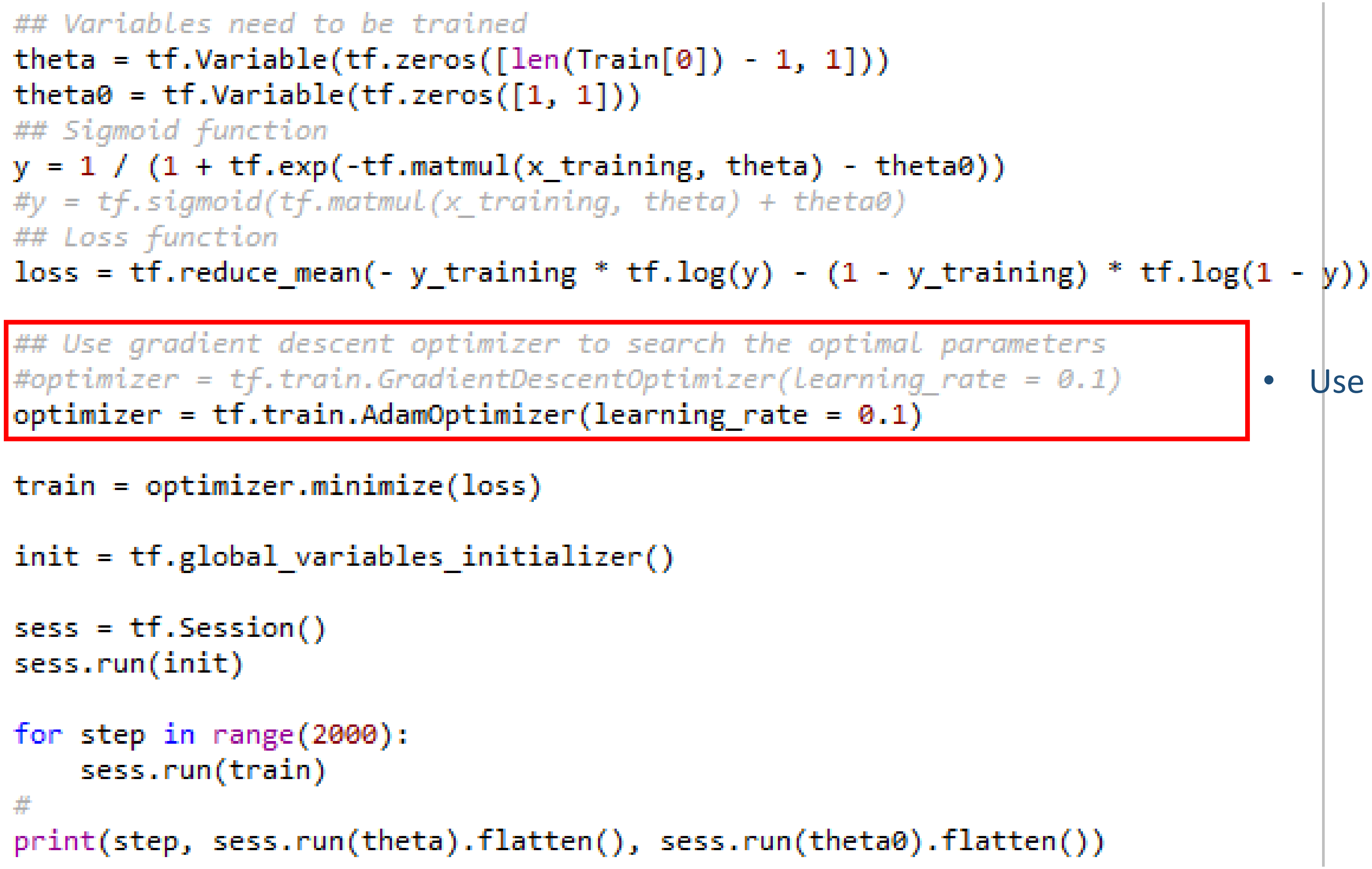
- When fix learning rate, prediction accuracy will increase and finally converge with the increase of iterations.
- Smaller learning rate, slower convergence speed!

Review Lab2 Take Home Exercise

- Make a table to compare the prediction accuracy with at least 4 different learning algorithms (optimizers), including Gradient Descent, Stochastic Gradient Descent, Adam and at least one more optimizer (you can find different optimizers in TenforFlow's official webside at https://www.tensorflow.org/versions/master/api_docs/python/train/) and discuss their principle, advantage and weakness in the report.

- We give the example solution of using Adam and Stochastic Gradient Descent.

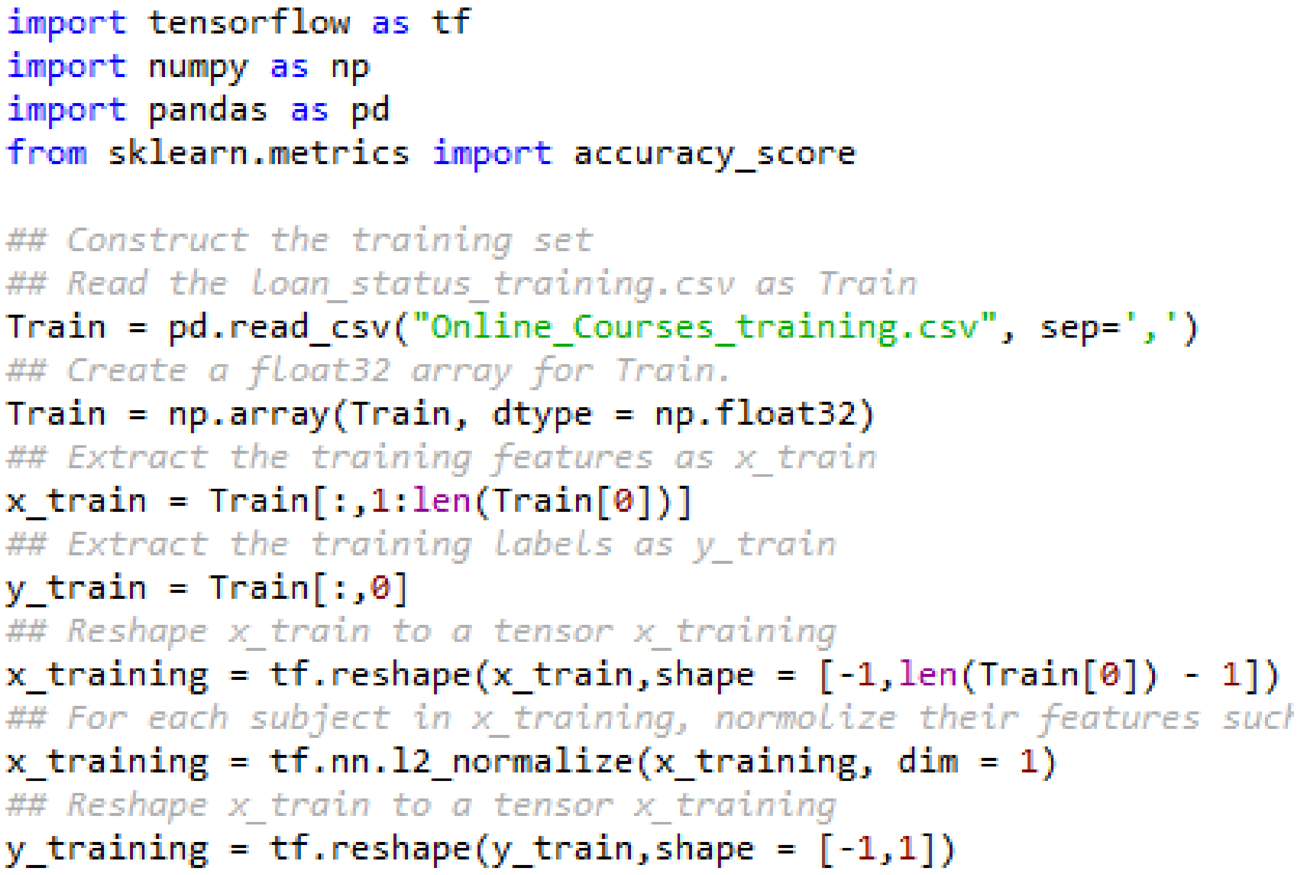
Solution of Lab2 using Adam



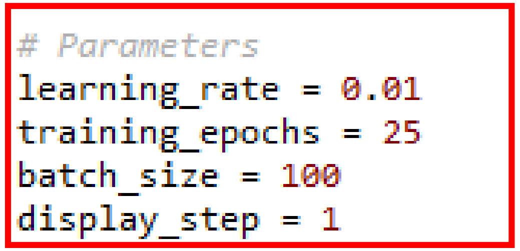
```
## Variables need to be trained
theta = tf.Variable(tf.zeros([len(Train[0]) - 1, 1]))
theta0 = tf.Variable(tf.zeros([1, 1]))
## Sigmoid function
y = 1 / (1 + tf.exp(-tf.matmul(x_training, theta) - theta0))
#y = tf.sigmoid(tf.matmul(x_training, theta) + theta0)
## Loss function
loss = tf.reduce_mean(- y_training * tf.log(y) - (1 - y_training) * tf.log(1 - y))

## Use gradient descent optimizer to search the optimal parameters
#optimizer = tf.train.GradientDescentOptimizer(learning_rate = 0.1)
optimizer = tf.train.AdamOptimizer(learning_rate = 0.1)

train = optimizer.minimize(loss)

init = tf.global_variables_initializer()

sess = tf.Session()
sess.run(init)

for step in range(2000):
    sess.run(train)
#
print(step, sess.run(theta).flatten(), sess.run(theta0).flatten())
```

- Use Adam to train the model

Solution of Lab2 using Stochastic Gradient Descent



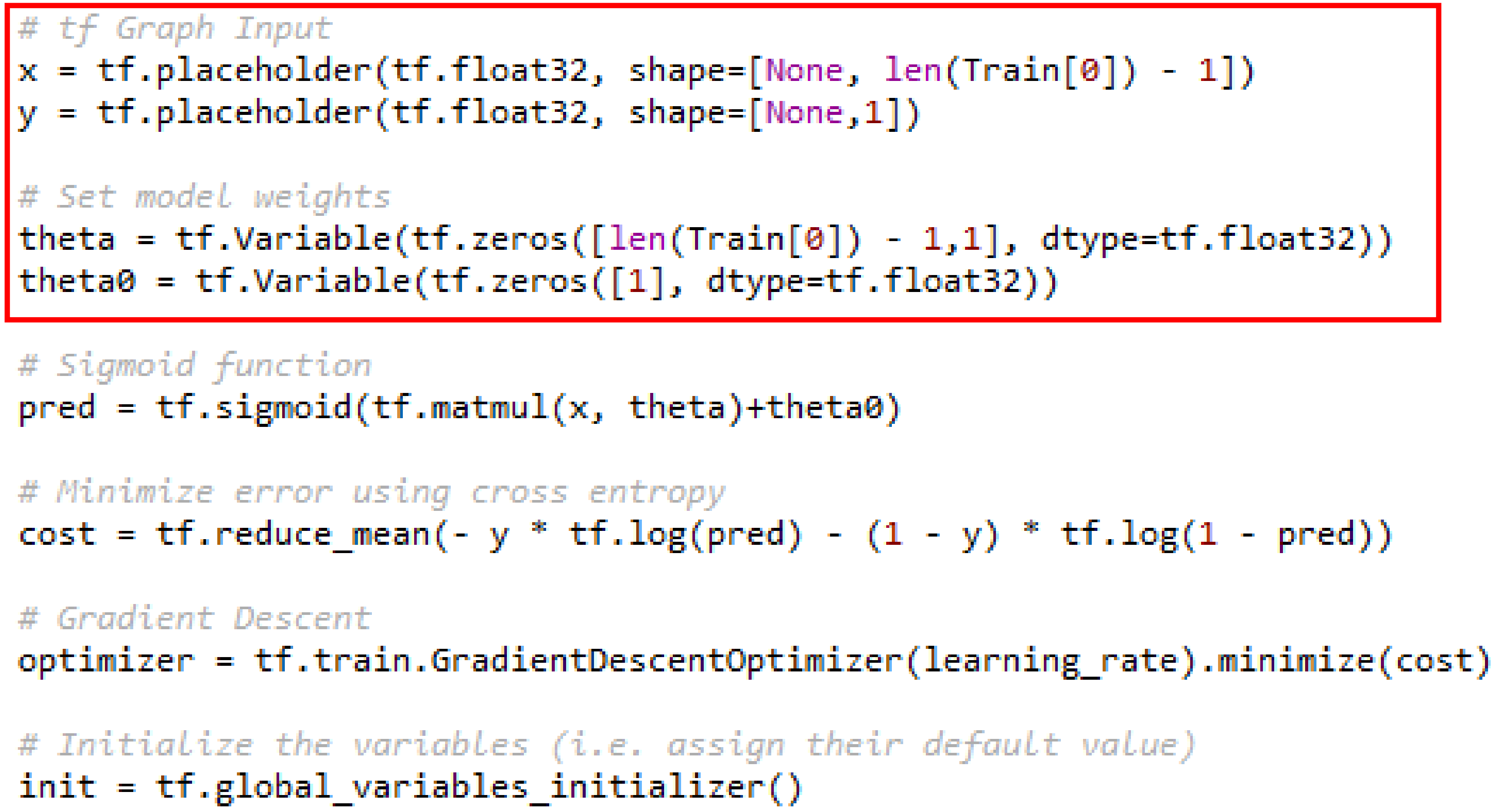
```
import tensorflow as tf
import numpy as np
import pandas as pd
from sklearn.metrics import accuracy_score

## Construct the training set
## Read the loan_status_training.csv as Train
Train = pd.read_csv("Online_Courses_training.csv", sep=',')
## Create a float32 array for Train.
Train = np.array(Train, dtype = np.float32)
## Extract the training features as x_train
x_train = Train[:,1:len(Train[0])]
## Extract the training labels as y_train
y_train = Train[:,0]
## Reshape x_train to a tensor x_training
x_training = tf.reshape(x_train,shape = [-1,len(Train[0]) - 1])
## For each subject in x_training, normolize their features such
x_training = tf.nn.l2_normalize(x_training, dim = 1)
## Reshape x_train to a tensor x_training
y_training = tf.reshape(y_train,shape = [-1,1])
```



```
# Parameters
learning_rate = 0.01
training_epochs = 25
batch_size = 100
display_step = 1
```

- Setting the training parameters

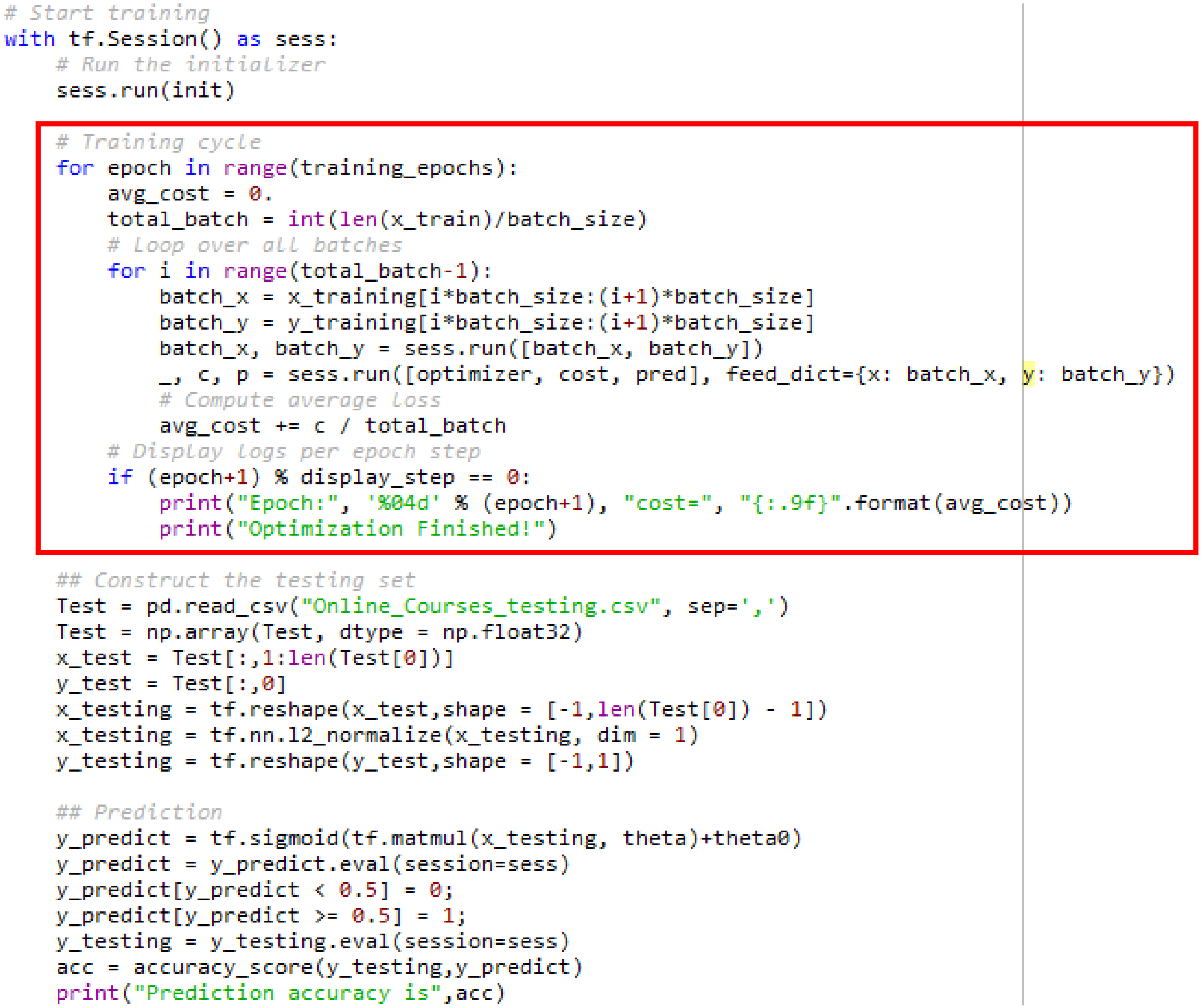
Solution of Lab2 using Stochastic Gradient Descent



```
# tf Graph Input
x = tf.placeholder(tf.float32, shape=[None, len(Train[0]) - 1])
y = tf.placeholder(tf.float32, shape=[None,1])

# Set model weights
theta = tf.Variable(tf.zeros([len(Train[0]) - 1,1], dtype=tf.float32))
theta0 = tf.Variable(tf.zeros([1], dtype=tf.float32))

# Sigmoid function
pred = tf.sigmoid(tf.matmul(x, theta)+theta0)

# Minimize error using cross entropy
cost = tf.reduce_mean(- y * tf.log(pred) - (1 - y) * tf.log(1 - pred))

# Gradient Descent
optimizer = tf.train.GradientDescentOptimizer(learning_rate).minimize(cost)

# Initialize the variables (i.e. assign their default value)
init = tf.global_variables_initializer()
```

- Initial variables and placeholders

Solution of Lab2 using Stochastic Gradient Descent

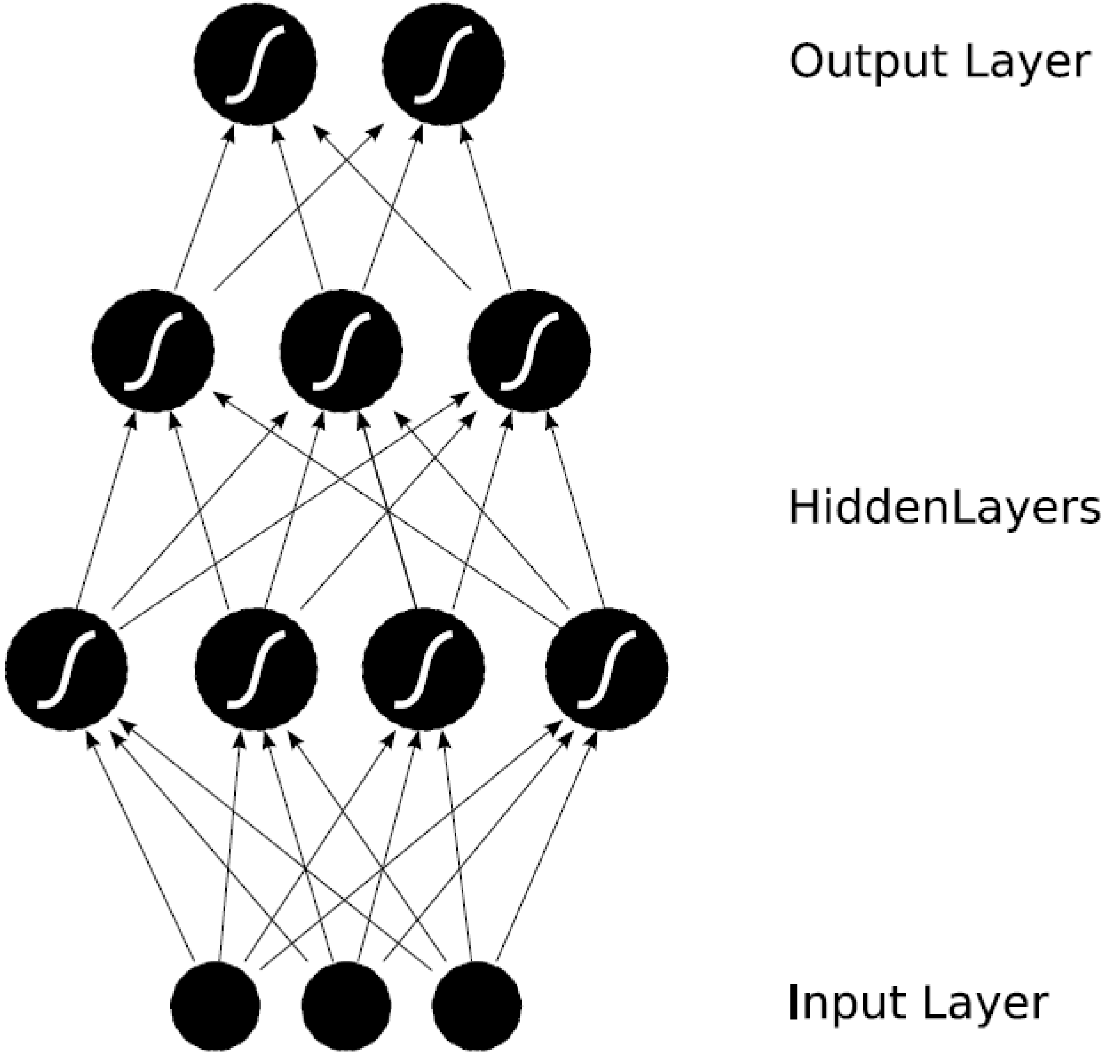


```
# Start training
with tf.Session() as sess:
    # Run the initializer
    sess.run(init)

    # Training cycle
    for epoch in range(training_epochs):
        avg_cost = 0.
        total_batch = int(len(x_train)/batch_size)
        # Loop over all batches
        for i in range(total_batch-1):
            batch_x = x_training[i*batch_size:(i+1)*batch_size]
            batch_y = y_training[i*batch_size:(i+1)*batch_size]
            batch_x, batch_y = sess.run([batch_x, batch_y])
            _, c, p = sess.run([optimizer, cost, pred], feed_dict={x: batch_x, y: batch_y})
            # Compute average loss
            avg_cost += c / total_batch
        # Display logs per epoch step
        if (epoch+1) % display_step == 0:
            print("Epoch:", '%04d' % (epoch+1), "cost=", "{:.9f}".format(avg_cost))
            print("Optimization Finished!")

    ## Construct the testing set
    Test = pd.read_csv("Online_Courses_testing.csv", sep=',')
    Test = np.array(Test, dtype = np.float32)
    x_test = Test[:,1:len(Test[0])]
    y_test = Test[:,0]
    x_testing = tf.reshape(x_test,shape = [-1,len(Test[0]) - 1])
    x_testing = tf.nn.l2_normalize(x_testing, dim = 1)
    y_testing = tf.reshape(y_test,shape = [-1,1])

    ## Prediction
    y_predict = tf.sigmoid(tf.matmul(x_testing, theta)+theta0)
    y_predict = y_predict.eval(session=sess)
    y_predict[y_predict < 0.5] = 0;
    y_predict[y_predict >= 0.5] = 1;
    y_testing = y_testing.eval(session=sess)
    acc = accuracy_score(y_testing,y_predict)
    print("Prediction accuracy is",acc)
```

- Separate the training data into patches and use each patch to train the model

Deep Neutral Network

Deep Neutral Network



- A deep-learning architecture is a multilayer stack of simple modules, all of which are subject to learning, and many of which compute non-linear input–output mappings.
- The hidden layers can be seen as distorting the input in a non-linear way so that categories become linearly separable by the last layer

Deep Neutral Network in TensorFlow

- TensorFlow’s two high level APIs for machine learning
  - tf.contrib.learn (https://www.tensorflow.org/api_docs/python/tf/contrib/learn/DNNClassifier)
  - tf.estimator (https://www.tensorflow.org/get_started/estimator)

- Easy to configure, train, and evaluate a variety of machine learning models

- TensorBoard
  - A visualization tool for visualizing the graph and learning
  - Graph visualization: https://www.tensorflow.org/get_started/graph_viz
  - Learning visualization: https://www.tensorflow.org/get_started/summaries_and_tensorboard

Implement DNN using tf.contrib.learn

- Step 1: Loading the training and testing data from CSV files
  - tf.contrib.learn.datasets.base.load_csv_with_header
  - pn.read_csv using pandas
- Step 2: Construct the DNN classifier
  - tf.contrib.learn.DNNClassifier(feature_columns, hidden_units=[*,*,*], n_classes=*, model_dir = “path”)
- Step 3: Fit the DNN classifier using input training features and labels
  - classifer.fit(x = training features, y = training labels, steps = iteration numbers)
- Step 4: Model evaluation using testing data
  - classifier.evaluate(x = testing features, y = testing labels)[“accuracy”]

Something You Have to Pay Attention to !!!

Implement DNN using tf.contrib.learn

- Step 3: Fit the DNN classifier using input training features and labels
  - classifer.fit(x = training features, y = training labels, steps = iteration numbers)
- Step 4: Model evaluation using testing data
  - classifier.evaluate(x = testing features, y = testing labels)[“accuracy”]

For the DNN classifier constructed by tf.contrib.learn.DNNClassifier

- The input training/testing features & training/testing labels SHOULD NOT be tensor!!
- The acceptable data type of training/testing features & training/testing is Matrix and Table List.

Lab3 Classroom Exercise

- Predict the institution with given online course features using DNN.
  - MIT is class “1” and Stanford class “0”
- Fill in the missing part of Lab_3_DNN.py
  1. Load the training and testing data.
  2. Construct feature set (from the second column to the last column in CSV) and label set (the first column in CSV).
  3. Construct the a 3 layers DNN classifier with 20, 20, 10 neurons respectively using tf.contrib.learn.DNNClassifier.
  4. Fit the DNN classifier using input training features and labels.
  5. Predict the institution of testing samples.
  6. Print the prediction accuracy.

Lab3 Take-home Exercise & Report

- Use the prediction performance to explain how parameters, including layers of DNN and the number of neurons in each layer, affect the prediction performance.
  - At least test 5 different numbers of layers and explain the reason why you select to test those numbers of layers.
  - For each number of layers, you need to at least test 5 sets of neuron number.
- Revise the in-class exercise code using tf.estimator to construct the DNN
  - You could find the example in https://www.tensorflow.org/get_started/estimator
  - Use tf.estimator.DNNClassifier to construct the DNN classifier.
  - Use classifier.train to train the DNN model.
- Summarize your experience in boosting prediction accuracy when tuning the parameters.
- Summarize the difference of using tf.estimator.DNNClassifier and tf.contrib.learn.DNNClassifier to construct and train the DNN model

Specification of Lab3 Report

- Use the previous Word lab report template
  - Modify where you think necessary (/section, /subsection, etc.)

- Lab3 Report due on 15th November 2017 at 23:59.

- Submit the Lab3 Report and all your codes via the Blackboard.

- Only the last one submission will be graded.

Evaluation Sheet

Attendance	Result			Report
	Correct in lab	Incomplete in lab		
		Correct at home	Incorrect at home	
20	20	-		60
20	-	10	0	60

thank you!