华南理工大学

South China University of Technology

The Experiment Report Of *Deep Learning*

Subject	: Software Engineering
College	: Software College.
Student ID:	201722800094
E-mail	: abas.aboras1@gmail.com
Tutor	: Dr. Mingkui Tan
Date submitted:	2017 - 12 - 8
Reporter	: ABAS MOHAMMED NAGI

Content Report:

Chapter1: Linear Regression and Gradient Descent

Chapter2: Linear Classification and Gradient Descent

CHAPTER1

Topic1: Linear Regression and Gradient Descent.

Purposes:

In this experiment we'll implement simple linear regression using gradient descent. Such that gradient descent is an algorithm that minimizes functions. Also process of optimization and adjusting parameters and further understand of liner regression and gradient descent.

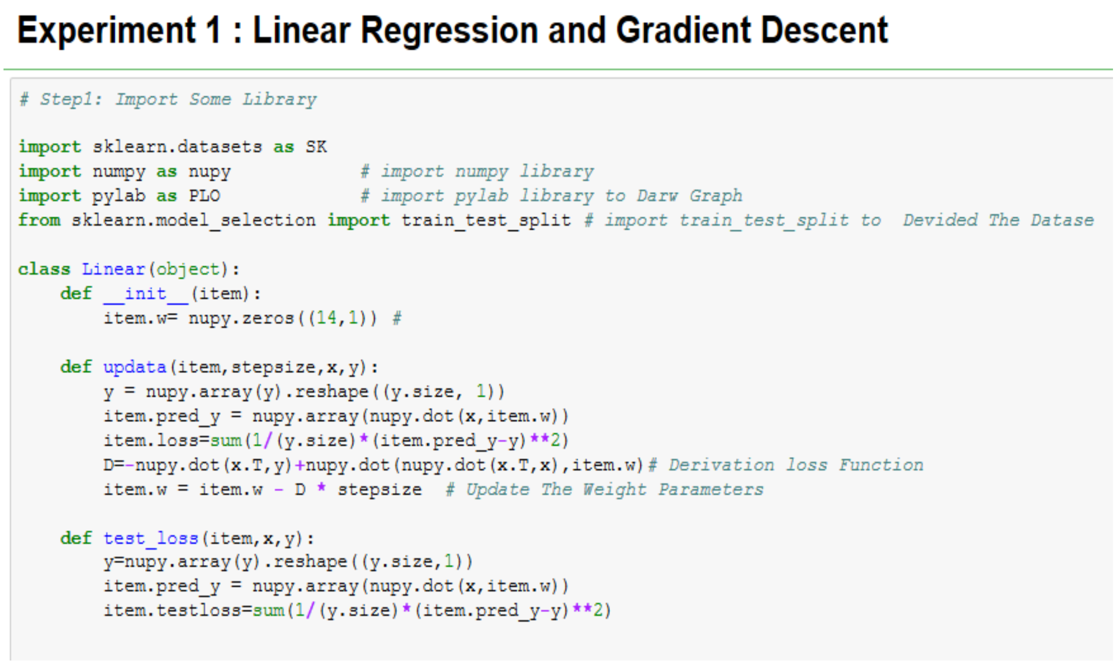
Data sets and data analysis:

In this experiment use Data set (housing_scale.txt) to procedure process training.
Linear Regression uses Housing in LIBSVM Data, including 506 samples and each sample has 13 features. You are expected to download scaled edition. After downloading, you are supposed to divide it into training set, validation set.

Experimental steps:

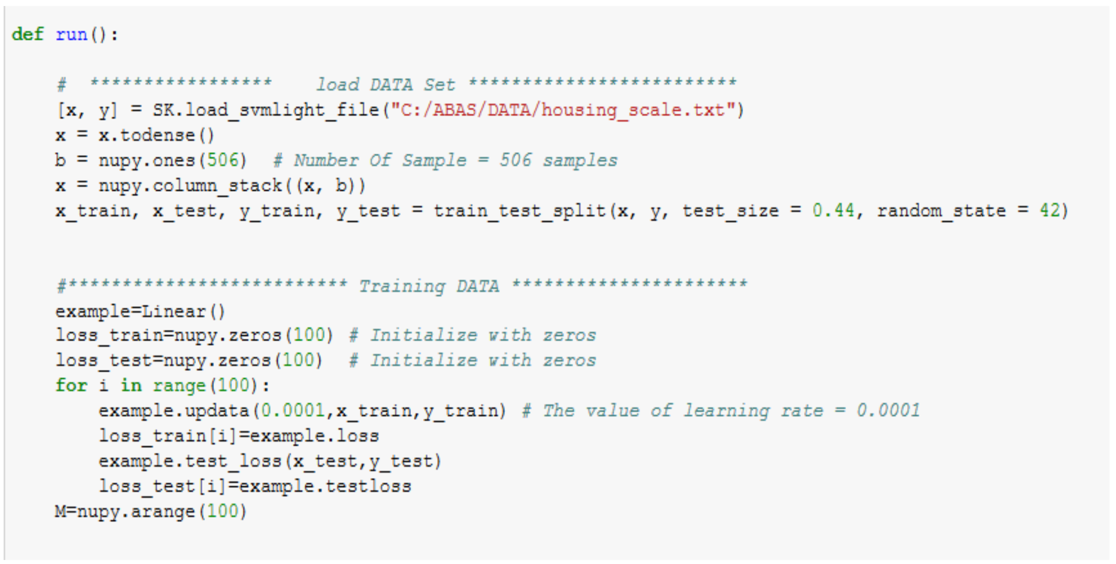
1) Load data set file (housing_scale.txt) via file from any folder save. Or using lode online data set via
Link (https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/regression/housing_scale)
2) Divide dataset into training set and validation set using train_test_split function.
3) Initialize linear model parameters set all parameter into zero, initialize it randomly or with normal distribution.
4) Using loss function and derivation.
5) Calculate gradient descent G toward loss function from all samples (506).
6) Update Weight.
7) Get the loss L_{train} under the training set $L_{validation}$ and by validating under validation set.
8) Drawing graph of L_{train} and $L_{validation}$.

Experiment Code:

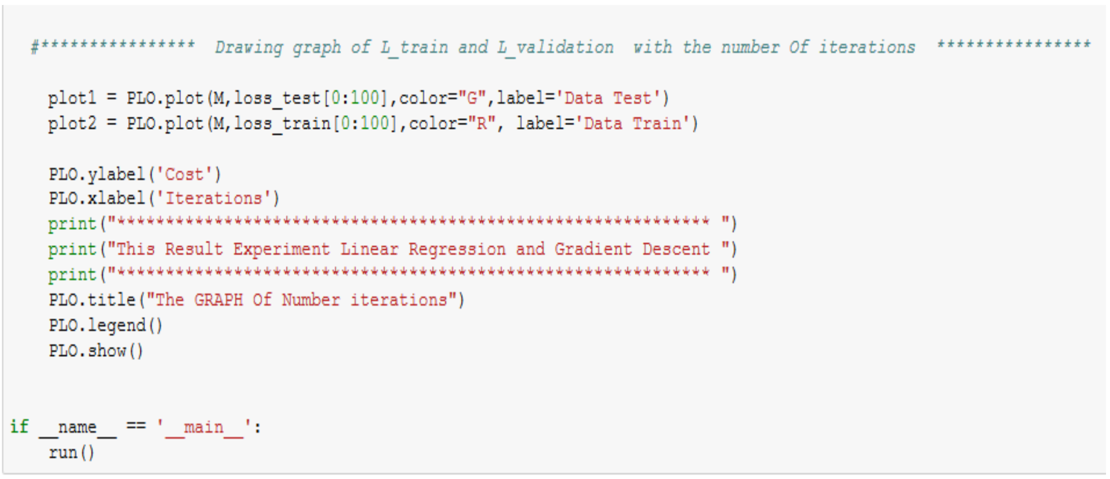


Experiment 1 : Linear Regression and Gradient Descent

```
# Step1: Import Some Library

import sklearn.datasets as SK
import numpy as nupy              # import numpy library
import pylab as PLO               # import pylab library to Darw Graph
from sklearn.model_selection import train_test_split # import train_test_split to  Devided The Datase

class Linear(object):
    def __init__(item):
        item.w= nupy.zeros((14,1)) #

    def updata(item,stepsize,x,y):
        y = nupy.array(y).reshape((y.size, 1))
        item.pred_y = nupy.array(nupy.dot(x,item.w))
        item.loss=sum(1/(y.size)*(item.pred_y-y)**2)
        D=-nupy.dot(x.T,y)+nupy.dot(nupy.dot(x.T,x),item.w)# Derivation loss Function
        item.w = item.w - D * stepsize  # Update The Weight Parameters

    def test_loss(item,x,y):
        y=nupy.array(y).reshape((y.size,1))
        item.pred_y = nupy.array(nupy.dot(x,item.w))
        item.testloss=sum(1/(y.size)*(item.pred_y-y)**2)
```



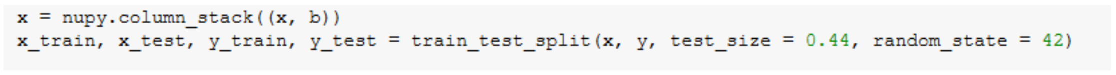
```
def run():

    #  *****************    load DATA Set *************************
    [x, y] = SK.load_svmlight_file("C:/ABAS/DATA/housing_scale.txt")
    x = x.todense()
    b = nupy.ones(506)  # Number Of Sample = 506 samples
    x = nupy.column_stack((x, b))
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.44, random_state = 42)


    #************************** Training DATA **********************
    example=Linear()
    loss_train=nupy.zeros(100) # Initialize with zeros
    loss_test=nupy.zeros(100)  # Initialize with zeros
    for i in range(100):
        example.updata(0.0001,x_train,y_train) # The value of learning rate = 0.0001
        loss_train[i]=example.loss
        example.test_loss(x_test,y_test)
        loss_test[i]=example.testloss
    M=nupy.arange(100)
```



```
    #****************  Drawing graph of L_train and L_validation  with the number Of iterations  ****************

    plot1 = PLO.plot(M,loss_test[0:100],color="G",label='Data Test')
    plot2 = PLO.plot(M,loss_train[0:100],color="R", label='Data Train')

    PLO.ylabel('Cost')
    PLO.xlabel('Iterations')
    print("************************************************************ ")
    print("This Result Experiment Linear Regression and Gradient Descent ")
    print("************************************************************ ")
    PLO.title("The GRAPH Of Number iterations")
    PLO.legend()
    PLO.show()


if __name__ == '__main__':
    run()
```

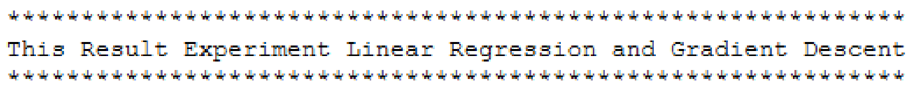
Selection of validation(hold-out,cross-validation, k-folds cross-validation,etc.)

Test size=   0.44   random =42

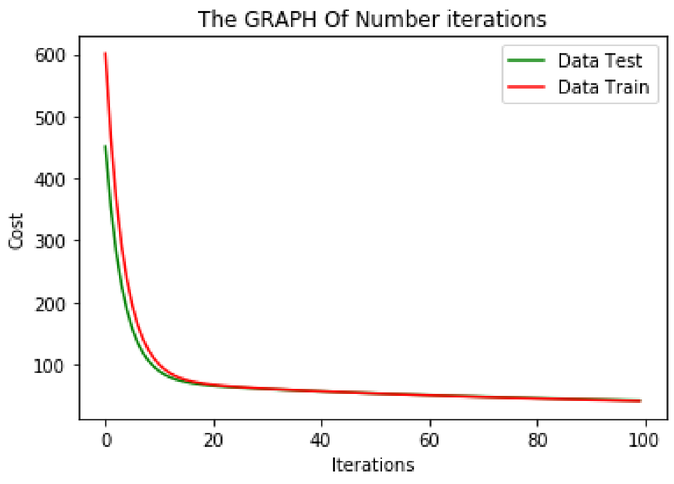


```
x = nupy.column_stack((x, b))
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.44, random_state = 42)
```

Experimental results and curve:



```
************************************************************
This Result Experiment Linear Regression and Gradient Descent
************************************************************
```



CHAPTER 2

Topic: Linear Classification and Gradient Descent

Purposes:

In this experiment we'll implement simple linear classification using gradient descent. Such that gradient descent is an algorithm that minimizes functions.

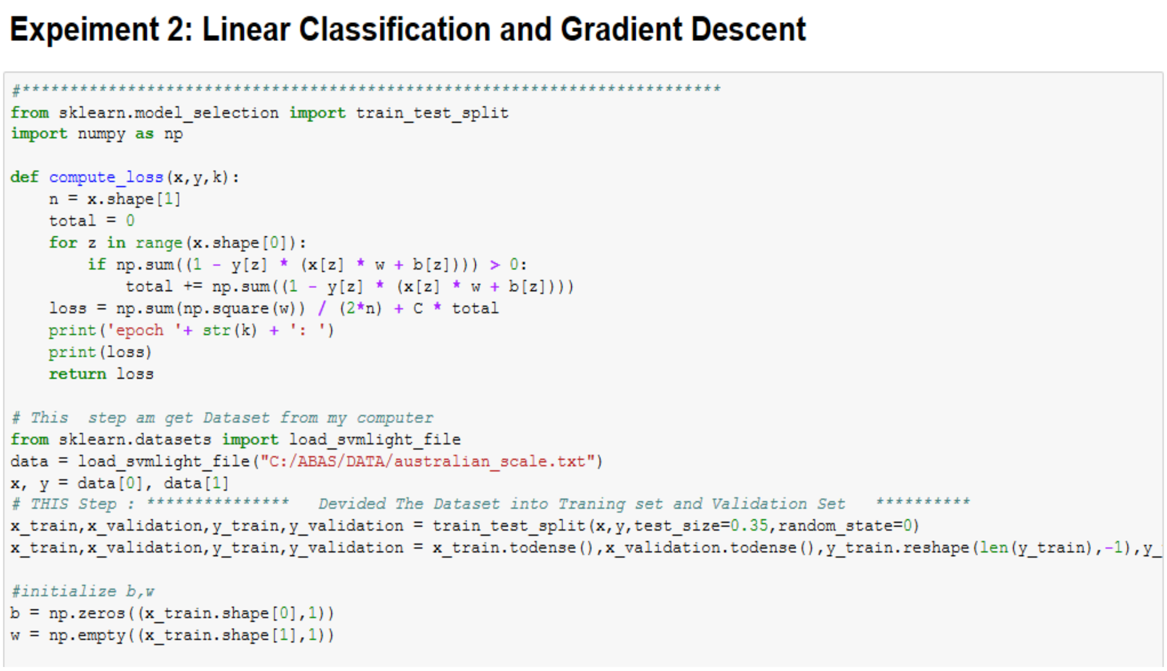
Data sets and data analysis:

In this experiment use Data set (Australian scale) to procedure process training. Linear classification uses Australian in LIBSVM Data, including 690 samples and each sample has 14 features. You are expected to download scaled edition. After downloading, you are supposed to divide it into training set, validation set.

Experimental steps:

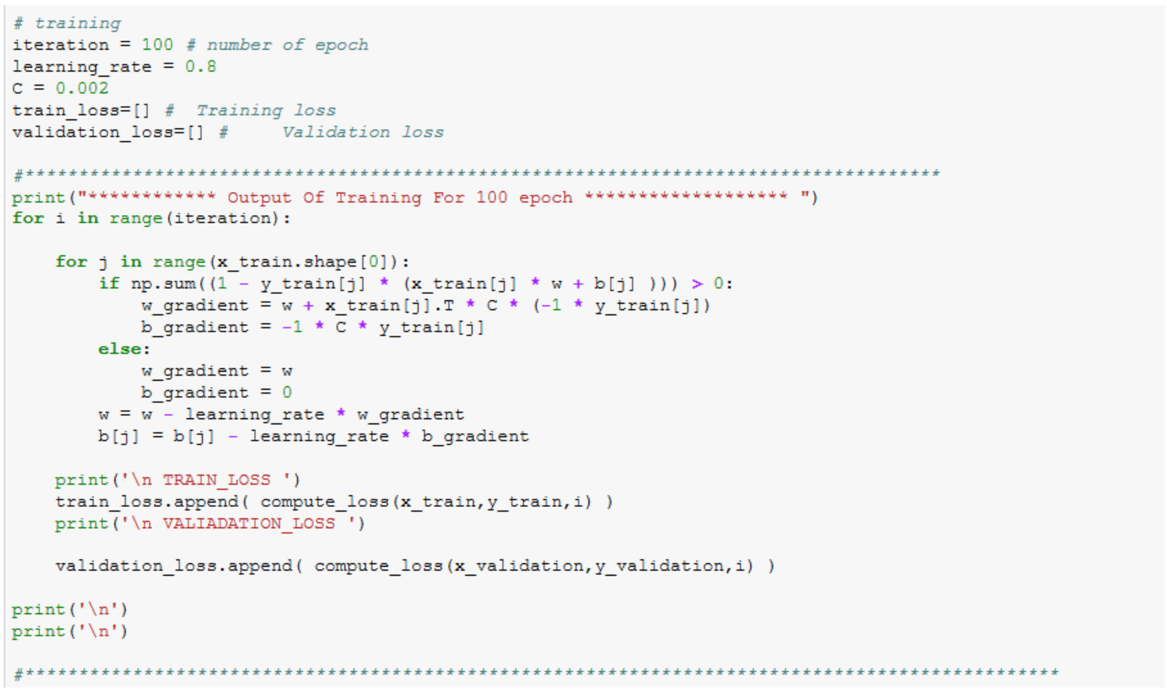
1) Load data set file (australian_scale.txt) via file from any folder save. Or using lode using requests.get() online data set via
Link (https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary/australian_scale)
2) Divide dataset into training set and validation set using train_test_split function.
3) Initialize linear model parameters set all parameter into zero, initialize it randomly or with normal distribution.
4) Using loss function and derivation.
5) Calculate gradient descent G toward loss function from all samples (690).
6) Update Weight.
7) Get the loss L_{train} under the training set $L_{validation}$ and by validating under validation set.
8) Drawing graph of L_{train} and $L_{validation}$.

Experiment Code :



Expeiment 2: Linear Classification and Gradient Descent

```python
#**************************************************************************
from sklearn.model_selection import train_test_split
import numpy as np

def compute_loss(x,y,k):
    n = x.shape[1]
    total = 0
    for z in range(x.shape[0]):
        if np.sum((1 - y[z] * (x[z] * w + b[z]))) > 0:
            total += np.sum((1 - y[z] * (x[z] * w + b[z])))
    loss = np.sum(np.square(w)) / (2*n) + C * total
    print('epoch '+ str(k) + ': ')
    print(loss)
    return loss

# This  step am get Dataset from my computer
from sklearn.datasets import load_svmlight_file
data = load_svmlight_file("C:/ABAS/DATA/australian_scale.txt")
x, y = data[0], data[1]
# THIS Step : ***************   Devided The Dataset into Traning set and Validation Set    **********
x_train,x_validation,y_train,y_validation = train_test_split(x,y,test_size=0.35,random_state=0)
x_train,x_validation,y_train,y_validation = x_train.todense(),x_validation.todense(),y_train.reshape(len(y_train),-1),y_

#initialize b,w
b = np.zeros((x_train.shape[0],1))
w = np.empty((x_train.shape[1],1))
```

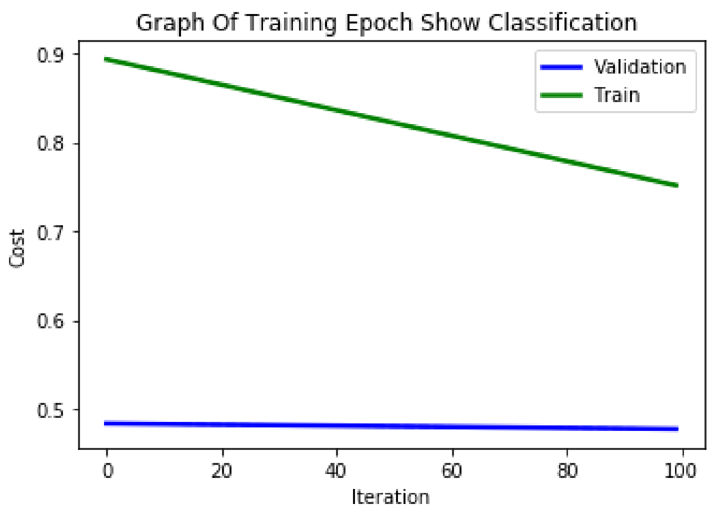


```python
# training
iteration = 100 # number of epoch
learning_rate = 0.8
C = 0.002
train_loss=[] #  Training loss
validation_loss=[] #     Validation loss

#*****************************************************************************************
print("************ Output Of Training For 100 epoch ****************** ")
for i in range(iteration):

    for j in range(x_train.shape[0]):
        if np.sum((1 - y_train[j] * (x_train[j] * w + b[j] ))) > 0:
            w_gradient = w + x_train[j].T * C * (-1 * y_train[j])
            b_gradient = -1 * C * y_train[j]
        else:
            w_gradient = w
            b_gradient = 0
        w = w - learning_rate * w_gradient
        b[j] = b[j] - learning_rate * b_gradient

    print('\n TRAIN_LOSS ')
    train_loss.append( compute_loss(x_train,y_train,i) )
    print('\n VALIADATION_LOSS ')

    validation_loss.append( compute_loss(x_validation,y_validation,i) )

print('\n')
print('\n')

#****************************************************************************************************
```

```
#**********************************************************************************************

# # This Drawing Graph of L_train  as well as L_validation with the Number of iterations
import matplotlib.pyplot as PLO  # Import this fuction to draw graph
t = np.arange(0, iteration, 1)
PLO.plot(t, validation_loss, color="blue", linewidth=2.5, linestyle="-", label="Validation")
PLO.plot(t, train_loss, color="green",  linewidth=2.5, linestyle="-", label="Train")
PLO.legend(loc='upper right')
PLO.plot(t, train_loss, 'g--',t, validation_loss, 'b--')
PLO.xlabel('Iteration')
PLO.ylabel('Cost')
print("******************************************************************** ")
print("This Result Experiment 2 Linear Classification and Gradient Descent ")
print("********************************************************************* ")
PLO.title("Graph Of Training Epoch Show Classification ")
plt.show()
```

Experimental results and curve:

```
This Result Experiment 2 Linear Classification and Gradient Descent
*********************************************************************
```



Selection of validation(hold-out,cross-validation, k-folds cross-validation,etc.)

Test size= 0.35 , random state = 44

```
# THIS Step : ***************   Devided The Dataset into Traning set and Validation Set   **********
x_train,x_validation,y_train,y_validation = train_test_split(x,y,test_size=0.35,random_state=44)
x_train,x_validation,y_train,y_validation = x_train.todense(),x_validation.todense(),y_train.reshape(len(y_train),-1),
```

Similarities and differences between linear regression and linear classification:

Linear classification uses the linear combination of the features to make a classification on the data set. Linear regression is the most basic type of regression and commonly used predictive analysis. Also linear regression is a linear approach for modeling the relationship between a scalar dependent variable y and one or more explanatory variables (or independent variables) denoted X.
Both regression and classification problems belong to the supervised category of machine learning. In Supervised machine learning, a model or a function is learnt from the data to predict the future data.
The main difference between the classification and the regression is their dependent variable. For the classification, the dependent variables are categorical, while the regression has numerical dependent variables.

Results analysis:

From result which show in the graph it’s easily understandable that the train curve and the test curve are it consider the same.

Summary:

In this experiment I discovered the simple linear regression model and how to train it using gradient descent. I work through the experiment of the update rule for gradient descent. I also learned how to make predictions with a learned linear regression model.