

**The Experiment Report of**

***Machine Learning***

**College: Software College**

**Subject: Software Engineering**

**E-mail: Fayazmalikscut@gmail.com**

**Tutor: Ming Kui Jin**

**Date submitted: 2017.12. 08**

**1. Topic:** Linear Regression Linear Classification and gradient descent

**2. Time:** Online Submission

**3. Reporter:** Malik Fayaz Ahmed

**4. Purposes:**

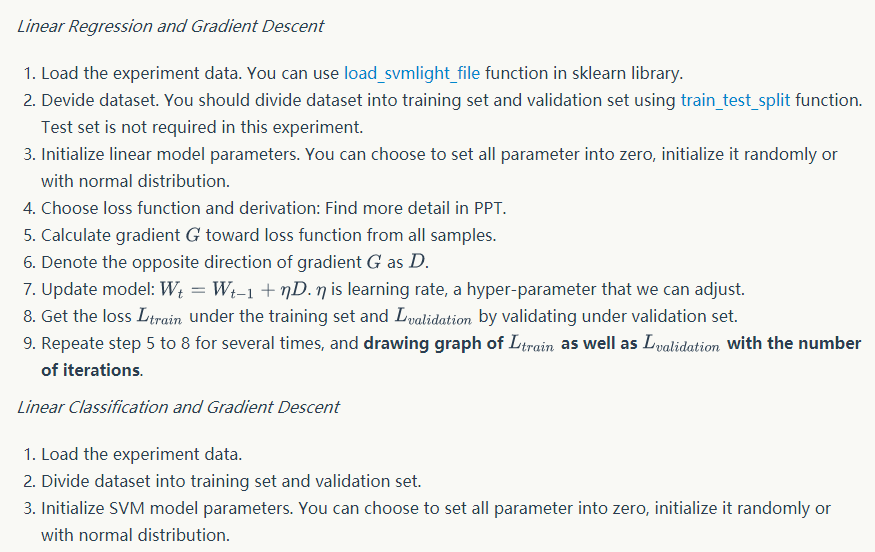
1. Further understanding of linear regression and gradient decent
2. Conduct some gradient under small scale dataset
3. Realize the process of optimization and Adjusting Parameters

**5. Data sets and data analysis:**

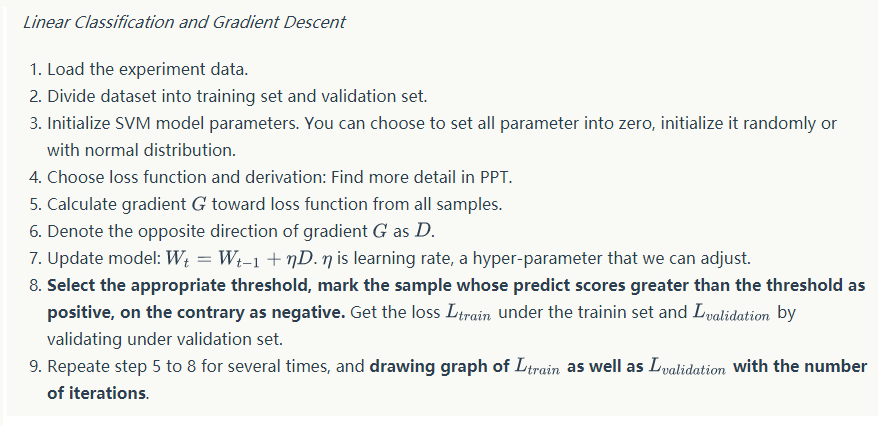
1. Housing\_scaled Data (Experiment one)
2. Australian\_scaled Data (Experiment two)

**6. Experimental steps:**

**Experiment: 01**



**Experiment: 02**



**7. Code:**

**Experiment 01**

#Code generated by Malik Fayaz Ahmed

#International Student Pakistan

#Student ID 201722800095

#South China University of Technology

#file uploadind command

from sklearn.datasets import load\_svmlight\_file

#importing the library

import numpy as np

#setting up values

Alpha\_Value = 0.001

Interations\_fayaz = 200

accuracy = 0.01

#importing the memory

from sklearn.externals.joblib import Memory

from sklearn.model\_selection import train\_test\_split

import matplotlib.pyplot as plt

#sample data

m = 506

m\_Malik = 203

m\_test = 203

features=13

theta=[0,0,0,0,0, 0,0,0,0,0, 0,0,0,0]

Interations\_fayaz\_num = [1]\*Interations\_fayaz;

loss\_trainning\_Fayaz = [1]\*Interations\_fayaz;

loss\_test = [1]\*Interations\_fayaz;

#path to upload the data file

def get\_data():

data = load\_svmlight\_file(r'''C:\Users\HP\Desktop\housing\_scale''',n\_features=13)

return data[0], data[1]

X,y = get\_data()

X=X.data.reshape(X.shape)

X\_train, X\_test, y\_train, y\_test = train\_test\_split( X, y, test\_size=0.5, random\_state=43)

print(X\_train)

print(y\_train)

def hypothesis(x):

result = theta[0]

for i in range (0,features):

result = result + theta[i+1] \* x[i]

return result

def loss(m,X,y):

sum=0

for i in range(0,m):

sum = sum + ( hypothesis(X[i]) - y[i] ) \*\*2

sum = sum / (2\*m)

return sum

def derivative(j,m,X,y):

sum=0

if(j==0):

for i in range(0,m):

sum = sum + ( hypothesis(X[i]) - y[i] )

else:

for i in range(0,m):

sum = sum + ( hypothesis(X[i]) - y[i] ) \* X[i][j-1]

sum = sum / m

return sum

def train():

for i in range(0,Interations\_fayaz):

for j in range(0,features+1):

theta[j] = theta[j] - Alpha\_Value \* derivative(j,m\_Malik,X\_train,y\_train)

Interations\_fayaz\_num[i] = i;

loss\_trainning\_Fayaz[i] = loss(m\_Malik,X\_train,y\_train);

loss\_test[i] = loss(m\_test,X\_test,y\_test);

def information():

print("loss on train:", loss\_trainning\_Fayaz)

print("loss on test",loss\_test)

train()

information()

fig, ax = plt.subplots()

ax.plot(Interations\_fayaz\_num, loss\_trainning\_Fayaz,color = 'm', label='loss of train')

ax.plot(Interations\_fayaz\_num, loss\_test, color = 'c', label='loss of test')

ax.set\_xlabel('Interations\_fayazation times')

ax.set\_ylabel('loss')

plt.show()

**Experiment 02 code**

#Code generated by Malik Fayaz Ahmed

#International Student Pakistan

#Student ID 201722800095

#South China University of Technology

import matplotlib.pyplot as plt

from sklearn.externals.joblib import Memory

import numpy

from numpy import random

# Load the experiment data

mem = Memory("./mycache")

from sklearn.datasets import load\_svmlight\_file

from sklearn.model\_selection import train\_test\_split

@mem.cache

def get\_data():

data = load\_svmlight\_file(r'''C:\Users\HP\Desktop\australian\_scale''',n\_features=14)

return data

def svm(W, xtrain, ytrain, xtest, ytest, reg):

gW = numpy.zeros(W.shape)

number\_of\_classes = W.shape[1]

trainning\_loss\_myExpe = 0

Scores\_Values\_MyExperiment2 = xtrain.dot(W)

Experiment2\_num\_trainning = xtrain.shape[0]

Scores\_Values\_MyExperiment2\_correct = Scores\_Values\_MyExperiment2[numpy.arange(Experiment2\_num\_trainning), ytrain]

Scores\_Values\_MyExperiment2\_correct = numpy.reshape(Scores\_Values\_MyExperiment2\_correct, (Experiment2\_num\_trainning, 1))

margins\_train = Scores\_Values\_MyExperiment2 - Scores\_Values\_MyExperiment2\_correct + 1.0

margins\_train[numpy.arange(Experiment2\_num\_trainning), ytrain] = 0.0

margins\_train[margins\_train <= 0] = 0.0

trainning\_loss\_myExpe += numpy.sum(margins\_train) / Experiment2\_num\_trainning

trainning\_loss\_myExpe += 0.5 \* reg \* numpy.sum(W \* W)

margins\_train[margins\_train > 0] = 1.0

row\_sum = numpy.sum(margins\_train, axis=1)

margins\_train[numpy.arange(Experiment2\_num\_trainning), ytrain] = -row\_sum

gW += numpy.dot(xtrain.T, margins\_train)/Experiment2\_num\_trainning + reg \* W

test\_loss = 0

scores\_test = xtest.dot(W)

num\_test = xtest.shape[0]

scores\_test\_correct = scores\_test[numpy.arange(num\_test), ytest]

scores\_test\_correct = numpy.reshape(scores\_test\_correct, (num\_test, 1))

margins\_test = scores\_test - scores\_test\_correct + 1.0

margins\_test[numpy.arange(num\_test), ytest] = 0.0

margins\_test[margins\_test <= 0] = 0.0

test\_loss += numpy.sum(margins\_test) / num\_test

test\_loss += 0.5 \* reg \* numpy.sum(W \* W)

return trainning\_loss\_myExpe, test\_loss, gW

data = get\_data()

X=data[0].toarray()

Y=data[1]

Y=Y.reshape(len(Y),order='C')

Y=Y.astype(numpy.int)

x\_train, x\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.4, random\_state=42)

N,D=x\_train.shape

C=len(list(set(y\_train)))

W = random.random(size=(D, C))

maxIterations=400

th = 0

eta = 0.01

L\_train=[];

L\_test=[];

for t in range(maxIterations):

y\_train\_pred = numpy.dot(x\_train,W)

y\_train\_pred[y\_train\_pred> th] = 1

y\_train\_pred[y\_train\_pred<=th] = 0

y\_test\_pred = numpy.dot(x\_test,W)

y\_test\_pred[y\_test\_pred> th] = 1

y\_test\_pred[y\_test\_pred<=th] = 0

trainning\_loss\_myExpe, test\_loss, grad\_W= svm(W, x\_train, y\_train, x\_test, y\_test, reg= 0.1)

L\_train.append (trainning\_loss\_myExpe)

L\_test.append (test\_loss)

W -= eta \* grad\_W

plt.plot(L\_train,'r',label='train loss')

plt.plot(L\_test,'b',label='test loss')

plt.title('Loss Curve') # give plot a title

plt.legend()

plt.show()

(Fill in the contents of 8-12 respectively for linear regression and linear classification)

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

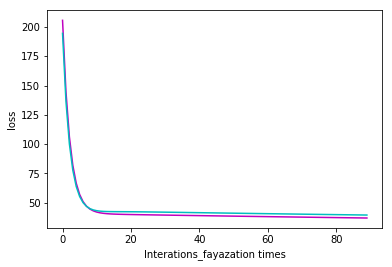
**9. The initialization method of model parameters:**

**10. The selected loss function and its derivatives:**

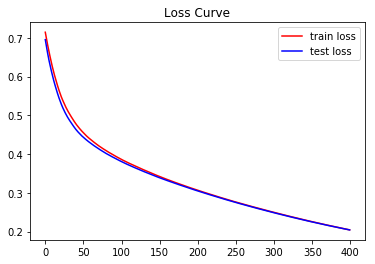
**11. Experimental results and curve**

Experiment result ：01

[[-0.998746 -0.2 -0.942082 ..., 0.510638 1. -0.765453] [-0.998104 -0.1 -0.781525 ..., -0.446809 0.903071 -0.814018] [-0.869221 -1. 0.293255 ..., 0.617021 1. -0.502759] ..., [-0.999344 0.6 -0.766862 ..., -0.191489 0.991326 -0.584989] [-0.996375 -1. -0.492669 ..., 0.489362 1. -0.698124] [-0.992475 -1. -0.492669 ..., 0.489362 1. -0.757726]] [ 22.9 37. 23. 11.7 12.1 13.9 23.4 32.2 21.7 19.9 17.8 21.7 23.6 16.5 18. 16.2 17.5 27.9 22.9 22.8 18.9 23.8 19.4 18.8 27. 16.8 20.4 14.3 26.6 27.5 11.8 18.5 24.6 25. 15.6 18.2 20.7 13.1 20.3 21.9 11.9 21.2 16.1 20.8 34.9 18.9 30.8 15.6 23.7 44.8 14.5 14.6 14.9 23.8 18.7 19.4 20.4 28. 32. 24.7 18.5 12.6 21.2 16.5 17.1 24.5 17. 19.4 18.6 28.4 8.4 19.9 26.4 13.8 23.6 27.5 42.8 29.8 20.8 43.5 22.6 29.1 20.1 20. 50. 24.1 20.3 13.3 50. 29. 50. 14.5 24.8 46. 33.2 17.5 13.1 17.1 17.8 24.3 15. 13.1 24.1 33.4 23.9 20.6 26.7 28.5 25.3 20.5 13.4 22.5 11. 30.3 20.1 10.5 32.9 50. 17.3 8.3 14.9 32.4 22. 22.7 19.6 13.1 14.2 20.1 22.4 35.4 22.2 19.1 23.9 21.9 23. 22.6 36.2 13.5 16.7 36.4 36.5 20.7 28.7 20. 23.2 9.5 25. 24.4 19.8 24.4 11.8 34.6 19.4 22.5 17.7 10.5 11.3 34.9 26.6 22.9 18.2 32.7 13.3 13.6 23.1 10.9 48.8 14.5 45.4 24.2 19.1 7.4 35.1 15.6 25. 19.5 14.1 21.2 10.8 33.2 16. 27.1 50. 50. 19.8 41.7 19.6 28.1 15.2 15. 24.4 20.4 16.3 27.5 22.3 20.2 30.1 7.5 36.2 20.1 33.1 23.9 19.8 18.1 21.4 18.4 13.9 16.1 9.6 36.1 8.8 23.4 50. 17.8 42.3 13.2 29.4 29.6 23.3 50. 12. 22. 27.9 23. 13.8 31.6 29.1 17.1 13.5 23.1 13. 28.7 20. 10.2 11.7 12.3 37.3 22. 23.9 7. 21.9 13.4 23.1 32. 13.8 28.2 23.1 23.3 33.1 28.4 20.9 23.8 25. ] loss on train: [205.5514254492181, 144.69835682788022, 106.30751443316822, 82.08106387957325, 66.786483832437526, 57.124211734257322, 51.013589999048939, 47.142589523315451, 44.683869246686513, 43.115725962870691, 42.109187088226015, 41.456812623196662, 41.027798556130129, 40.73966570146839, 40.540407136915107, 40.39723071026539, 40.289460310258477, 40.204059591739515, 40.132809375302607, 40.070527820854828, 40.013948149747669, 39.961010998526476, 39.910418225365078, 39.861351577870934, 39.813295314256351, 39.765924371155648, 39.719033860231079, 39.672494622873344, 39.626225214133726, 39.580174244576369, 39.534309251985619, 39.488609689343818, 39.443062507382386, 39.397659372365254, 39.352394914334525, 39.307265624600738, 39.262269162199438, 39.217403917888554, 39.17266874027036, 39.128062763928583, 39.083585301719559, 39.039235777380675, 38.995013683452406, 38.95091855507588, 38.906949953732223, 38.863107457196641, 38.819390653367961, 38.775799136507473, 38.732332504969932, 38.688990359853996, 38.645772304215058, 38.602677942619245, 38.559706880902034, 38.516858726046735, 38.474133086132731, 38.431529570321345, 38.389047788861916, 38.346687353107001, 38.304447875530606, 38.262328969746186, 38.220330250523077, 38.178451333800346, 38.136691836698112, 38.095051377526254, 38.053529575791188, 38.012126052200429, 37.970840428665731, 37.929672328304726, 37.888621375441723, 37.847687195607278, 37.806869415537228, 37.766167663171011, 37.725581567649797, 37.685110759313844, 37.644754869700002, 37.604513531538501, 37.564386378749923, 37.524373046442022, 37.484473170906185, 37.444686389613906, 37.405012341213471, 37.365450665526097, 37.326001003542373, 37.286662997418766, 37.247436290473793, 37.208320527184135, 37.169315353181474, 37.130420415248146, 37.091635361313962, 37.052959840452232] loss on test [194.44308391545616, 136.41178933754782, 100.2575379112007, 77.803864140287956, 63.91444463604865, 55.365963681197542, 50.138196937662137, 46.967014514373552, 45.063040002276544, 43.934660685405163, 43.276753414244503, 42.900780982476817, 42.69092149177883, 42.576570368850241, 42.515132327389196, 42.481274723056281, 42.460233719679643, 42.443658931431031, 42.427045178355087, 42.408154259708382, 42.386052461668925, 42.360529555038156, 42.331752977872668, 42.300066059933336, 42.26587369916254, 42.229580506057665, 42.191559910794879, 42.152141112478461, 42.111605945773945, 42.070190944278586, 42.028091843083267, 41.985468955419471, 41.94245257418649, 41.8991479720219, 41.855639817597897, 41.811995961752913, 41.768270618557608, 41.724507000224236, 41.68073947714899, 41.636995335104551, 41.593296196445529, 41.549659164487679, 41.50609774184305, 41.462622565409966, 41.419241993409976, 41.37596257349449, 41.332789415542749, 41.289726488257656, 41.246776854949005, 41.203942860855435, 41.16122628189158, 41.11862844271694, 41.076150310425113, 41.033792568867902, 40.991555677605895, 40.949439918657546, 40.907445433567354, 40.86557225279536, 40.82382031901566, 40.782189505585812, 40.740679631186474, 40.699290471425392, 40.658021768034125, 40.616873236156835, 40.575844570126989, 40.534935448044408, 40.494145535402907, 40.453474487964101, 40.41292195403453, 40.372487576269442, 40.332170993101819, 40.291971839873632, 40.251889749732122, 40.211924354339182, 40.172075284432388, 40.132342170269283, 40.092724641978485, 40.053222329837084, 40.013834864489638, 39.974561877121189, 39.935402999593293, 39.896357864551071, 39.857426105507287, 39.818607356908352, 39.779901254185432, 39.741307433794489, 39.702825533247065, 39.664455191134081, 39.626196047143537, 39.588047742074131]



Experiment result ： 02



**12. Similarities and differences between linear regression and linear classification:**

Regression and classification are both related to prediction, where regression predicts a value from a continuous set, whereas classification predicts the 'belonging' to the class.

For example, the price of a house depending on the 'size' (in some unit) and say 'location' of the house, can be some 'numerical value' (which can be continuous): this relates to regression.

Similarly, the prediction of price can be in words, viz., 'very costly', 'costly', 'affordable', 'cheap', and 'very cheap': this relates to classification.

Regression: given a set of data, find the best relationship that represents the set of data.

Classification: given a known relationship, identify the class that the data belongs to.

We can see that regression and classification start from opposing ends: to find a pattern or to find the pattern that it belongs to.

**14. Summary:**

**Regression problems** are those where you are trying to predict or explain one thing (dependent variable) using what you know about other things (independent variables). That covers anything that can be expressed as numbers, probabilities, or true/false answers. Linear regression is intended more specifically for continuous numbers, and was first proposed by Adrien-Marie Legendre in 1805.

**Classification problems** try to determine group membership by deriving probabilities. The first technique ever used was linear discriminant analysis (LDA), proposed by Sir R.A. Fisher in 1936—he used to to classify irises. I do not understand it fully, but believe that it used linear regression to derive probabilities for each group, and then used a Mahalanobis distance measure to assign to the closest group. One does not here much about LDA any more, I think because is has been supplanted by logistic regression.