

**The Experiment Report of**

***Machine Learning***

**Linear Regression, Linear Classification**

**and Gradient Descent**

**College Software College**

**Subject Software Engineering**

**Members**   **朱懿晖**

**Student ID 201530613955**

**E-mail 1240836507@qq.com**

**Tutor**   **Mingkui Tan**

**Date submitted** **2017.12.06**

**1.Topic:**

Linear Regression, Linear Classification and Gradient Descent

**2.Time:**

2017-12-02 9:00-12:00 AM B7-138/238

**3. Reporter:**

Grade 2015 Class 4 Yihui Zhu

**4. Purposes:**

1. Further understand of linear regression and gradient descent.

2. Conduct some experiments under small scale dataset.

3. Realize the process of optimization and adjusting parameters.

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

Linear Regression uses Housing in LIBSVM Data, including 506 samples and each sample has 13 features.

Linear classification uses australian in LIBSVM Data, including 690 samples and each sample has 14 features.

**6. Experimental steps:**

The experimental code and drawing are completed on jupyter.

*Linear Regression and Gradient Descent*

1. Load the experiment data. You can use [load\_svmlight\_file](http://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_svmlight_file.html) function in sklearn library.
2. Divide dataset. You should divide dataset into training set and validation set using [train\_test\_split](http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html) function. Test set is not required in this experiment.
3. Initialize linear model parameters. You can choose to set all parameter into zero, initialize it randomly or with normal distribution.
4. Choose loss function and derivation: Find more detail in PPT.
5. Calculate gradient G toward loss function from all samples.
6. Denote the opposite direction of gradient G as D .
7. Update model: W\_t=W\_{t-1}+ηD . ηis learning rate, a hyper-parameter that we can adjust.
8. Get the loss L\_train under the training set and L\_validation by validating under validation set.
9. Repeat step 5 to 8 for several times, and **drawing graph of** L\_train **as well as** L\_validation **with the number of iterations**.

*Linear Classification and Gradient Descent*

1. Load the experiment data.
2. Divide dataset into training set and validation set.
3. Initialize SVM model parameters. You can choose to set all parameter into zero, initialize it randomly or with normal distribution.
4. Choose loss function and derivation: Find more detail in PPT.
5. Calculate gradient G toward loss function from all samples.
6. Denote the opposite direction of gradient G as D.
7. Update model: W\_t=W\_{t-1}+ηD. ηis learning rate, a hyper-parameter that we can adjust.
8. **Select the appropriate threshold, mark the sample whose predict scores greater than the threshold as positive, on the contrary as negative.** Get the loss L\_train under the training set and L\_validation by validating under validation set.
9. Repeat step 5 to 8 for several times, and **drawing graph of** L\_train **as well as** L\_validation **with the number of iterations**.

Finishing experiment report according to result: The template of report can be found in [example repository](https://github.com/chenyaofo/ML2017-lab-01).

**7. Code:**

*Linear Regression and Gradient Descent*

%matplotlib inline

import numpy as np

import matplotlib.pyplot as plt

from sklearn import datasets as ds

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

#读取数据集

X,y = ds.load\_svmlight\_file("D:/traindata/housing\_scale.txt")

X = X.toarray()

row = X.shape[0]

B = np.ones([row,])

X = np.c\_[X,B]

#切分数据集

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size=0.33, random\_state=42)

r\_train = X\_train.shape[0]

#参数全零初始化

W = np.zeros([14,])

#学习率

lr = 0.09

#λ参数

lamda = 0.5

#迭代次数

s = 25

#储存迭代中loss值

L\_train = np.zeros([s])

L\_validation = np.zeros([s])

for n in range(s):

#梯度初始化

g = 0

#对训练集所有样本求梯度g

for i in range(r\_train):

g = g + lamda\*W + np.dot(X\_train[i].T,X\_train[i])\*W-np.dot(X\_train[i].T,y\_train[i])

#取平均值

g = g/r\_train

#更新模型参数

W = W - lr\*g

#计算训练集loss

L\_train[n] = 0.5\*np.dot(W.T,W)+0.5\*np.dot((y\_train-np.dot(X\_train,W)).T,y\_train-np.dot(X\_train,W))/r\_train

#计算验证集loss

L\_validation[n] = 0.5\*np.dot(W.T,W)+0.5\*np.dot((y\_test-np.dot(X\_test,W)).T,y\_test-np.dot(X\_test,W))/(row-r\_train)

#制图

n = np.arange(s)

plt.plot(n,L\_train[n],label='loss\_train')

plt.plot(n,L\_validation[n],label='loss\_validation')

plt.legend(loc='upper right')

plt.xlabel('time')

plt.ylabel('loss')

*Linear Classification and Gradient Descent*

%matplotlib inline

import numpy as np

import matplotlib.pyplot as plt

from sklearn import datasets as ds

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

#读取数据集

X,y = ds.load\_svmlight\_file("D:/traindata/australian\_scale.txt")

X = X.toarray()

row = X.shape[0]

B = np.ones([row,])

X = np.c\_[X,B]

#切分数据集

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size=0.33, random\_state=42)

r\_train = X\_train.shape[0]

#参数全零初始化

W = np.zeros([15,])

#学习率

lr = 0.05

#迭代次数

t = 100

#阈值

value = 0.2

#储存迭代中loss值

L\_train = np.zeros([t])

L\_validation = np.zeros([t])

for n in range(t):

#梯度初始化

g = 0

for i in range(r\_train):

#对训练集所有样本求梯度g

if 1-y\_train[i]\*(W.T.dot(X\_train[i])) >= 0:

g = g + W - X\_train[i]\*y\_train[i]

else:

g = g + W

#取平均值

g = g/r\_train

#更新模型参数

W = W - lr\*g

r = 0

#命中数

hit = 0

for i in range(r\_train):

r = r + max(0,1-y\_train[i]\*W.T.dot(X\_train[i]))

if (W.dot(X\_train[i]) >= value and y\_train[i] == 1) or (W.dot(X\_train[i]) < value and y\_train[i] == -1):

hit += 1

#训练集准确率

accuracy\_train = hit/r\_train

#计算训练集loss

L\_train[n] = 0.5\*W.T.dot(W) + 1/r\_train\*r

r = 0

#命中数

hit = 0

for i in range(row-r\_train):

r = r + max(0,1-y\_test[i]\*W.T.dot(X\_test[i]))

if (W.dot(X\_test[i]) >= value and y\_test[i] == 1) or (W.dot(X\_test[i]) < value and y\_test[i] == -1):

hit += 1

#测试集准确率

accuracy\_test = hit/(row-r\_train)

#计算验证集loss

L\_validation[n] = 0.5\*W.T.dot(W) + 1/(row-r\_train)\*r

#制图

n = np.arange(t)

plt.plot(n,L\_train[n],label='loss\_train')

plt.plot(n,L\_validation[n],label='loss\_validation')

plt.legend(loc='upper right')

plt.xlabel('time')

plt.ylabel('loss')

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

*Linear Regression and Gradient Descent*

Hold-out.

*Linear Classification and Gradient Descent*

Hold-out.

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

*Linear Regression and Gradient Descent*

All zero initialization, set all parameter into zero.

*Linear Classification and Gradient Descent*

All zero initialization, set all parameter into zero.

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

*Linear Regression and Gradient Descent*

The selected loss function:

LD(w) = lamda\*||w||^2/2 + ||y-Xw||^2/2;

The derivatives of the loss function:

gw(X) = lamda\*w + X^T\*Xw – X^T\*y.

*Linear Classification and Gradient Descent*

The selected loss function:

L(w) = ||w||^2/2 + C\*sum{max(0, 1- y[i]\*(w^T\*X[i]+b)};

The derivatives of the loss function:

if 1- y[i]\*(w^T\*X[i]+b) >= 0: gw(X[i]) = w -y[i]\*X[i];

if 1- y[i]\*(w^T\*X[i]+b) < 0: gw(X[i]) = w.

**11. Experimental results and curve:**

*Linear Regression and Gradient Descent*

## Hyper-parameter selection (η, epoch, etc.):

η= 0.09;

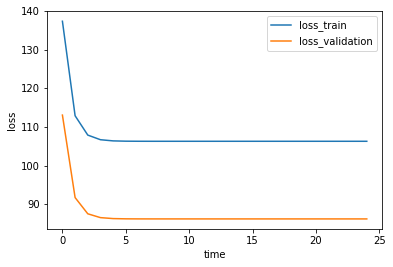
## Assessment Results (based on selected validation):

User Hold\_out: {L\_train, L\_test} = {106.315336793 86.2101280321}(0.01), {106.298540593 86.1963095898}(0.02), {106.298539263 86.1963084954}(0.03), then the loss is stable, when learning rate is 0.09, the descent is fast.

## Predicted Results (Best Results):

L\_train = 106.29853926, L\_test = 86.1963085.

## Loss curve:



*Linear Classification and Gradient Descent*

## Hyper-parameter selection (η, epoch, etc.):

η= 0.05;

## Assessment Results (based on selected validation):

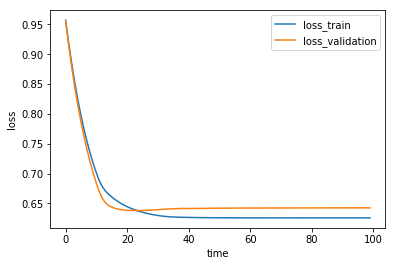
User Hold\_out: {L\_train, L\_test} ={0.625809985827 0.642660776263}(0.01), {0.625808189809 0.642713696903}(0.02), {0.625815100964 0.642601358445}(0.05).

## Predicted Results (Best Results):

L\_train = 0.625815100964, L\_test = 0.642601358445.

Accuracy: 0.886.

## Loss curve:



**12. Results analysis:**

*Linear Regression and Gradient Descent*

According to the loss curve above, the loss is stable and the loss\_train is higher than loss\_validation, because they are made from the same dataset and train data is more than validation data. The gradient descent is fast and bring a good results, but the loss maybe large, it can be caused by the problem of data.

*Linear Classification and Gradient Descent*

According to the loss curve above, finally the loss is stable and the loss\_validation is higher than loss\_train, it is a normal result. And the accuracy of test data is 0.886 which means it hits many data in the data set and has a good effect.

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

The both linear regression and linear classification have the same model function y = wX + b, their parameters are simple and their accuracies are a little low.

The difference between linear regression and linear classification is that regression is to analyse the relationship about x and y, but classification is to divide y to several parts by x, their functions are different. And regression is more used to deal with continuous problems, classification is usually used to handle discrete data.

**14. Summary:**

According to the experiment, I try to build linear regression model and linear classification model, and deepen the understanding of the two model, gradient descent and their theories. And I learn and experience the process of optimization and adjusting parameters. It really makes difference to me in studying machine learning.