

**The Experiment Report of**

***Machine Learning***

**College Software College**

**Subject Software Engineering**

**Members**  **DW Su**

**Student ID**

**E-mail**

**Tutor**  **Prof. Tan Mingkui**

**Date submitted** **2017. 12 . 8**

1. **Topic:**

Linear Regression, Linear Classification and Gradient Descent

1. **Time:**

2017-12-02 9:00-12:00 AM

1. **Reporter:**

Su Dewei

1. **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:**

1.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.

2.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.

**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 function in sklearn library.

2.Devide dataset. You should divide dataset into training set and validation set using train\_test\_split 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: Wt = Wt-1+ȠD . Ƞ is learning rate, a hyper-parameter that we can adjust.

8.Get the loss Ltrain under the training set and Lvalidation by validating under validation set.

9.Repeate step 5 to 8 for several times, and drawing graph of as well as 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: Wt = Wt-1+ȠD . Ƞ is 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 Ltrain under the training set and Lvalidation by validating under validation set.

9.Repeate step 5 to 8 for several times, and drawing graph of as well as with the number of iterations.

1. **Code:**

Linear Regression and Gradient Descent

# import package

from sklearn.datasets import load\_svmlight\_file

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

import numpy as np

import matplotlib.pyplot as plt

# load data

data,label=load\_svmlight\_file('housing\_scale')

data = data.todense()

# seperate feature and label to training data and validation data

feature\_train,feature\_val,label\_train,label\_val = train\_test\_split(data,label,test\_size=0.30, random\_state=42)

# add the bias item to the matrix

def add\_bias(matrix):

bias = []

for i in range(matrix.shape[0]):

bias.append(1)

matrix = np.column\_stack((matrix,bias))

return matrix

feature\_train = add\_bias(feature\_train)

feature\_val = add\_bias(feature\_val)

# initialize the weight matrix

w = np.random.random(size=(1,feature\_train.shape[1]))

# gradient descent function

def gradient\_descent(x,y,x\_val,y\_val,w,learning\_rate,maxiterations):

y = np.asmatrix(y)

y = np.transpose(y)

w = np.transpose(w)

# define the iteration time and loss value

iteration\_time = [] #iteration time

train\_loss\_value = [] #loss value of training data

val\_loss\_value = [] #loss value of validation data

for i in range(0,maxiterations):

# calculate the result

train\_hypothesis = np.dot(x,w)

val\_hypothesis = np.dot(x\_val,w)

#address the gradient

gradient = np.dot(np.transpose(x),train\_hypothesis - y)/x.shape[0]

# update the weight matrix

w = w - learning\_rate\*gradient

#define the loss function and calculate the loss value

train\_loss = 0

for j in range(len(train\_hypothesis)):

train\_loss = train\_loss + ((train\_hypothesis[j]-y[j]).tolist()[0][0])\*\*2

train\_loss = train\_loss/(2\*train\_hypothesis.shape[0])

val\_loss = 0

for j in range(len(val\_hypothesis)):

val\_loss = val\_loss + ((val\_hypothesis[j]-y\_val[j]).tolist()[0][0])\*\*2

val\_loss = val\_loss/(2\*val\_hypothesis.shape[0])

# record the iteration time and loss value

iteration\_time.append(i)

train\_loss\_value.append(train\_loss)

val\_loss\_value.append(val\_loss)

#draw the loss graph

plt.plot(iteration\_time, train\_loss\_value , label='Training Loss')

plt.plot(iteration\_time, val\_loss\_value , label='Validation Loss')

plt.title('loss')

plt.legend()

plt.show()

gradient\_descent(feature\_train,label\_train,feature\_val,label\_val,w,0.001,1000)

Linear Classification and Gradient Descent

# import package

from sklearn.datasets import load\_svmlight\_file

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

import numpy as np

import matplotlib.pyplot as plt

# load data

data,label=load\_svmlight\_file('australian\_scale')

data = data.todense()

# seperate feature and label to training data and validation data

feature\_train,feature\_val,label\_train,label\_val = train\_test\_split(data,label,test\_size=0.30, random\_state=42)

# add the bias item to the matrix

def add\_bias(matrix):

bias = []

for i in range(matrix.shape[0]):

bias.append(1)

matrix = np.column\_stack((matrix,bias))

return matrix

feature\_train = add\_bias(feature\_train)

feature\_val = add\_bias(feature\_val)

# initialize the weight matrix

w = np.random.random(size=(1,feature\_train.shape[1]))

def gradient\_descent(x,y,x\_val,y\_val,C,w,learning\_rate,maxiterations):

y = np.asmatrix(y)

y = np.transpose(y)

y\_val = np.asmatrix(y\_val)

y\_val = np.transpose(y\_val)

w = np.asmatrix(w)

w = np.transpose(w)

# define the iteration time and loss value

iteration\_time = []

train\_loss\_value = []

val\_loss\_value = []

for i in range(0,maxiterations):

# calculate the gradient

gradient = 0

for j in range(len(x)):

if 1 - (y[j].tolist()[0][0])\*( np.dot(x[j],w).tolist()[0][0] ) >= 0 :

gradient = gradient - (y[j].tolist()[0][0])\*(x[j])

gradient = gradient\*C

gradient = np.transpose(gradient)+w

# update the weight matrix

w = w - learning\_rate\*gradient

# define the loss function and calculate the loss value

train\_loss = 0

for j in range(len(x)):

train\_loss = train\_loss + max(0,1 - (y[j].tolist()[0][0])\*( np.dot(x[j],w).tolist()[0][0] ))

train\_loss = train\_loss\*C

train\_loss = train\_loss + ((norm(w))\*\*2)/2

val\_loss = 0

for j in range(len(x\_val)):

val\_loss = val\_loss + max(0,1 - (y\_val[j].tolist()[0][0])\*( np.dot(x\_val[j],w).tolist()[0][0] ))

val\_loss = val\_loss\*C

val\_loss = val\_loss + ((norm(w))\*\*2)/2

# record the iteration time and loss value

iteration\_time.append(i)

train\_loss\_value.append(train\_loss/x.shape[0])

val\_loss\_value.append(val\_loss/x\_val.shape[0])

# draw the loss graph

plt.plot(iteration\_time, train\_loss\_value , label='Training Loss')

plt.plot(iteration\_time, val\_loss\_value , label='Validation Loss')

plt.title('loss')

plt.legend()

plt.show()

gradient\_descent(feature\_train,label\_train,feature\_val,label\_val,0.5,w,0.0001,2000)

1. **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.

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

Linear Regression and Gradient Descent:

Set all parameter into ones.

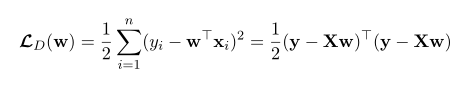
Linear Classification and Gradient Descent:

Set all parameter into ones.

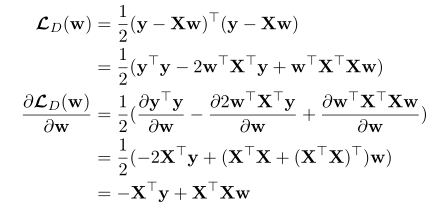
1. **The selected loss function and its derivatives:**

Linear Regression and Gradient Descent:

Loss Function:



Derivatives:

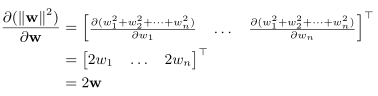


Linear Classification and Gradient Descent:

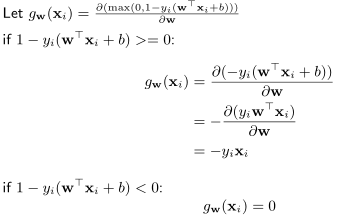
Loss Function:

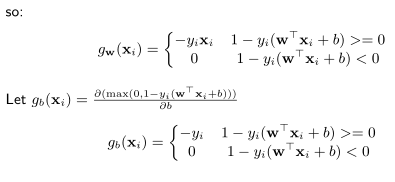


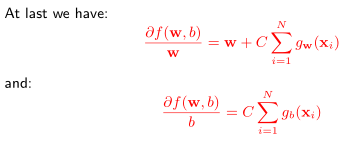
Derivatives:











1. **Experimental results and curve:**
2. Linear Regression and Gradient Descent:

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

|  |  |
| --- | --- |
| η | epoch |
| 0.01 | 50 |
| 0.01 | 100 |
| 0.01 | 150 |
| 0.01 | 200 |
| 0.01 | 250 |
| 0.01 | 300 |
| 0.001 | 50 |
| 0.001 | 100 |
| 0.001 | 150 |
| 0.001 | 200 |
| 0.001 | 250 |
| 0.001 | 300 |
| 0.0001 | 50 |
| 0.0001 | 100 |
| 0.0001 | 150 |
| 0.0001 | 200 |
| 0.0001 | 250 |
| 0.0001 | 300 |

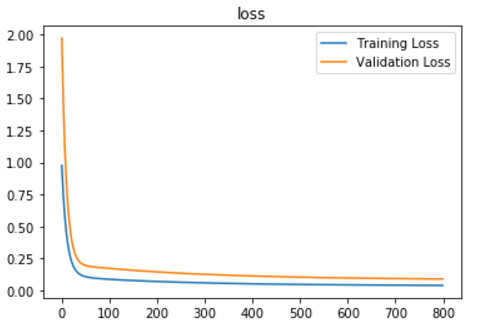
## Assessment Results (based on selected validation):

|  |  |  |
| --- | --- | --- |
| η | epoch | Validation Loss |
| 0.01 | 50 | 0.19 |
| 0.01 | 100 | 0.17 |
| 0.01 | 150 | 0.15 |
| 0.01 | 200 | 0.14 |
| 0.01 | 250 | 0.13 |
| 0.01 | 300 | 0.12 |
| 0.001 | 50 | 1.27 |
| 0.001 | 100 | 0.84 |
| 0.001 | 150 | 0.58 |
| 0.001 | 200 | 0.42 |
| 0.001 | 250 | 0.32 |
| 0.001 | 300 | 0.26 |
| 0.0001 | 50 | 1.88 |
| 0.0001 | 100 | 1.80 |
| 0.0001 | 150 | 1.72 |
| 0.0001 | 200 | 1.64 |
| 0.0001 | 250 | 1.57 |
| 0.0001 | 300 | 1.50 |

## Predicted Results (Best Results):

|  |  |  |
| --- | --- | --- |
| η | epoch | Validation Loss |
| 0.01 | 300 | 0.12 |

## Loss curve:



2.Linear Classification and Gradient Descent:

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

|  |  |  |
| --- | --- | --- |
| η | epoch | C |
| 0.001 | 50 | 0.5 |
| 0.001 | 100 | 0.5 |
| 0.001 | 150 | 0.5 |
| 0.001 | 200 | 0.5 |
| 0.001 | 250 | 0.5 |
| 0.001 | 300 | 0.5 |
| 0.001 | 50 | 0.8 |
| 0.001 | 100 | 0.8 |
| 0.001 | 150 | 0.8 |
| 0.001 | 200 | 0.8 |
| 0.001 | 250 | 0.8 |
| 0.001 | 300 | 0.8 |
| 0.001 | 50 | 1 |
| 0.001 | 100 | 1 |
| 0.001 | 150 | 1 |
| 0.001 | 200 | 1 |
| 0.001 | 250 | 1 |
| 0.001 | 300 | 1 |
| 0.0001 | 50 | 0.5 |
| 0.0001 | 100 | 0.5 |
| 0.0001 | 150 | 0.5 |
| 0.0001 | 200 | 0.5 |
| 0.0001 | 250 | 0.5 |
| 0.0001 | 300 | 0.5 |
| 0.0001 | 50 | 0.8 |
| 0.0001 | 100 | 0.8 |
| 0.0001 | 150 | 0.8 |
| 0.0001 | 200 | 0.8 |
| 0.0001 | 250 | 0.8 |
| 0.0001 | 300 | 0.8 |
| 0.0001 | 50 | 1 |
| 0.0001 | 100 | 1 |
| 0.0001 | 150 | 1 |
| 0.0001 | 200 | 1 |
| 0.0001 | 250 | 1 |
| 0.0001 | 300 | 1 |

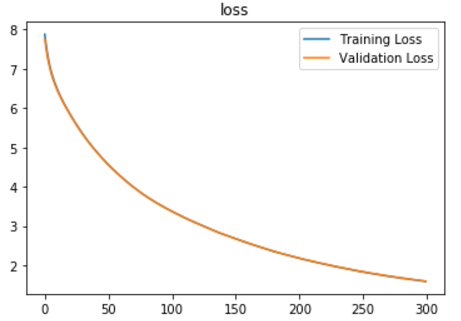
## Assessment Results (based on selected validation):

|  |  |  |  |
| --- | --- | --- | --- |
| η | epoch | C | Validation Loss |
| 0.001 | 50 | 0.5 | 4.58 |
| 0.001 | 100 | 0.5 | 3.38 |
| 0.001 | 150 | 0.5 | 2.68 |
| 0.001 | 200 | 0.5 | 2.18 |
| 0.001 | 250 | 0.5 | 1.84 |
| 0.001 | 300 | 0.5 | 1.59 |
| 0.001 | 50 | 0.8 | 4.09 |
| 0.001 | 100 | 0.8 | 2.95 |
| 0.001 | 150 | 0.8 | 2.29 |
| 0.001 | 200 | 0.8 | 1.89 |
| 0.001 | 250 | 0.8 | 1.90 |
| 0.001 | 300 | 0.8 | 1.77 |
| 0.001 | 50 | 1 | 3.88 |
| 0.001 | 100 | 1 | 2.76 |
| 0.001 | 150 | 1 | 2.16 |
| 0.001 | 200 | 1 | 2.03 |
| 0.001 | 250 | 1 | 1.97 |
| 0.001 | 300 | 1 | 1.87 |
| 0.0001 | 50 | 0.5 | 6.98 |
| 0.0001 | 100 | 0.5 | 6.47 |
| 0.0001 | 150 | 0.5 | 6.12 |
| 0.0001 | 200 | 0.5 | 5.84 |
| 0.0001 | 250 | 0.5 | 5.57 |
| 0.0001 | 300 | 0.5 | 5.33 |
| 0.0001 | 50 | 0.8 | 6.84 |
| 0.0001 | 100 | 0.8 | 6.27 |
| 0.0001 | 150 | 0.8 | 5.85 |
| 0.0001 | 200 | 0.8 | 5.48 |
| 0.0001 | 250 | 0.8 | 5.16 |
| 0.0001 | 300 | 0.8 | 4.89 |
| 0.0001 | 50 | 1 | 6.77 |
| 0.0001 | 100 | 1 | 6.16 |
| 0.0001 | 150 | 1 | 5.68 |
| 0.0001 | 200 | 1 | 5.28 |
| 0.0001 | 250 | 1 | 4.95 |
| 0.0001 | 300 | 1 | 4.66 |

## Predicted Results (Best Results):

|  |  |  |  |
| --- | --- | --- | --- |
| η | epoch | C | Validation Loss |
| 0.001 | 300 | 0.5 | 1.59 |

## Loss curve:



1. **Results analysis:**
2. Linear Regression and Gradient Descent:

With the increasing of the epoch , the validation loss will decrease which means a better result. In my experiment , when the learning rate is 0.01 and the epoch is 300 the results is the best one.

1. Linear Classification and Gradient Descent:

With the increasing of the epoch and the decreasing of the hyper parameter C, the validation loss will decrease which means a better result.In my experiment , when the learning = 0.001 ,epoch = 300 and C = 0.5 ,the results is the best one.

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

Differences:

Linear Regression: the output variable takes continuous values.

Linear Classification: the output variable takes class labels(Discrete Value).

Similarity:

The prediction model is both the linear model and can only be used to do the linear problem except the nonlinear problem.

1. **Summary:**

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. A linear classifier achieves this by making a classification decision based on the value of a linear combination of the characteristics. Further work should be done to explore if there is any posibility to unify this two problems in a united way.