华南理工大學

South China University of Technology

The Experiment Report of

Machine Learning

Using SGD in Logistic Regression and SVM

College Software College

Subject Software Engineering

Members 郭蕴喆

Student ID 201530611524

E-mail guoyunzhe.se@gmail.com

Tutor Mingkui Tan

Date submitted 2017.12.15

目录

A. TOPIC ---- **4**

B. TIME ---- **4**

C. REPORTER ---- **4**

D. PURPOSES ---- **4**

E. DATA SETS AND DATA ANALYSIS ---- **4**

1. Linear Regression ---- 4
2. Linear classification ---- 4

F. EXPERIMENTAL STEPS: ---- **4**

1. Linear Regression and Gradient Descent ---- 4
2. Linear Classification and Gradient Descent ---- 4

G. CODE ---- **错误！未定义书签。**

1. Linear Regression and Gradient Descent ---- 5
   - *(1) Function: Grad* ---- ***错误！未定义书签。***
   - *(2) Function: Draw* ---- *5*
   - *(3) Function: BGD* ---- ***错误！未定义书签。***
   - *(4) main* ---- *6*
2. Linear Classification and Gradient Descent ---- 8
   - *(1) Function: loss* ---- *8*
   - *(2) Function: compare* ---- *8*
   - *(3) Function: Grad* ---- *8*
   - *(4) Function: corrate_rate* ---- *8*
   - *(5) Function: Draw* ---- *9*
   - *(6) Function: BGD* ---- ***错误！未定义书签。***
   - *(7) main* ---- ***错误！未定义书签。***

H. SELECTION OF VALIDATION ---- **9**

1. Linear Regression and Gradient Descent ---- 9
2. Linear Classification and Gradient Descent ---- 9

I. THE INITIALIZATION METHOD OF MODEL PARAMETERS: ---- **9**

J. THE SELECTED LOSS FUNCTION AND ITS DERIVATIVES: ---- **10**

1. Linear Regression and Gradient Descent ---- 10
   - *(1) Loss Function* ---- *10*
   - *(2) Gradient* ---- *10*
2. Linear Classification and Gradient Descent ---- 10
   - *(1) Loss Function* ---- *10*
   - *(2) Gradient* ---- *10*

K. EXPERIMENTAL RESULTS AND CURVE: ---- **10**

1. Linear Regression and Gradient Descent 10

(1) Hyper-parameter selection 10

(2) Assessment Results (based on selected validation) 11

(3) Predicted Results (Best Results) 错误！未定义书签。

(4) Loss curve 11

2. Linear Classification and Gradient Descent 11

(1) Hyper-parameter selection 11

(2) Assessment Results (based on selected validation) 12

(3) Predicted Results (Best Results) 12

(4) Loss curve 错误！未定义书签。

(5) Accuracy 错误！未定义书签。

L. RESULTS ANALYSIS 12

1. Linear Regression and Gradient Descent 12

(1) 'Weights': 0.085, others the same 12

(2) 'Weights': 0.000000000085, others the same 13

2. Linear Classification and Gradient Descent 14

(1) 'learning_rate': 0.00085, others the same 14

(2) 'threshold':1, others the same 15

M. SIMILARITIES AND DIFFERENCES 16

N. SUMMARY 16

A. Topic

Logistic Regression, Linear Classification and Stochastic Gradient Descent

B. Time

2017/12/7

C. Reporter

郭蕴喆

D. Purposes

1. Compare and understand the difference between gradient descent and stochastic gradient descent.
2. Compare and understand the differences and relationships between Logistic regression and linear classification.
3. Further understand the principles of SVM and practice on larger data.

E. Data sets and data analysis

1. Logistic Regression

Experiment uses a9a of LIBSVM Data, including 32561/16281(testing) samples and each sample has 123/123 (testing) features.

2. Linear classification

Experiment uses a9a of LIBSVM Data, including 32561/16281(testing) samples and each sample has 123/123 (testing) features.

F. Experimental steps:

1. Logistic Regression and Gradient Descent

1. Load the training set and validation set.
2. Initalize logistic regression model parameters, you can consider initalizing zeros, random numbers or normal distribution.
3. Select the loss function and calculate its derivation, find more detail in PPT.
4. Calculate gradient toward loss function from partial samples.
5. Update model parameters using different optimized methods(NAG, RMSProp, AdaDelta and Adam).
6. Select the appropriate threshold, mark the sample whose predict scores greater than the threshold as positive, on the contrary as negative. Predict under validation set and get the different optimized method loss , , and .

7. Repeate step 4 to 6 for several times, and drawing graph of and with the number of iterations.

2. Linear Classification and Gradient Descent

1. Load the training set and validation set.
2. Initalize SVM model parameters, you can consider initalizing zeros, random numbers or normal distribution.
3. Select the loss function and calculate its derivation, find more detail in PPT.
4. Calculate gradient toward loss function from partial samples.
5. Update model parameters using different optimized methods(NAG，RMSProp，AdaDelta and Adam).
6. Select the appropriate threshold, mark the sample whose predict scores greater than the threshold as positive, on the contrary as negative. Predict under validation set and get the different optimized method loss ， ， and .
7. Repeate step 4 to 6 for several times, and drawing graph of ， ， and with the number of iterations.

1. Logtistic Regression and Gradient Descent

(1) Function: Draw

```
def Draw(loops,train_loss,validation_loss):
    # the first 100loops
    print('Drawing...')
    plt.plot(np.arange(0,200,1), train_loss[0:200], label='Train Loss')
    plt.plot(np.arange(0,200,1), validation_loss[0:200], label='Validation Loss')
    plt.xlabel('loops')
    plt.ylabel('loss')
    plt.title('The First 200 loops')
    plt.legend()
    plt.show()

    # the last 10000 loops
    plt.plot(np.arange(201, loops-1, 1), train_loss[201:loops-1], label='Train Loss')
    plt.plot(np.arange(201, loops-1, 1), validation_loss[201:loops-1], label='Validation Loss')
    plt.xlabel('loops')
    plt.ylabel('loss')
    plt.title('The rest loops')
    plt.legend()
    plt.show()
    print('Draw Completed')
```

(2) main

```
if __name__ == '__main__':
    # Read Data
    tic=time.time()
    Data_Path = 
'/home/lucas/Codes/GitHub/ML_Assignment1/ML_Assignment1/DataSet/a9a.txt'

Test_Path='/home/lucas/Codes/GitHub/ML_Assignment1/ML_Assignment1/DataSet/a9a.t'
    Data_Parameter, Data_Value = load_svmlight_file(Data_Path)
    Test_Parameter,Test_Value=load_svmlight_file(Test_Path)
    Test_Parameter=Test_Parameter.toarray()

Test_Parameter=np.hstack([Test_Parameter,np.zeros(shape=(Test_Parameter.shape[0],1))])
    Test_Value=Test_Value.reshape(Test_Value.shape[0],1)
    Data_Parameter = Data_Parameter.toarray()
    train_X, val_X, train_Y, val_Y = 
train_test_split(Data_Parameter, Data_Value, test_size=0.3, 
random_state=1)
    t_row = train_X.shape[0]   # Row Size
    col = train_X.shape[1]   # Column Size
    v_row = val_X.shape[0]
    train_Y = train_Y.reshape(t_row, 1)
    val_Y = val_Y.reshape(v_row, 1)
    W = np.random.random(size=(col, 1))
    Parameter={'Train_X':train_X,
               'Train_Y':train_Y,
               'Val_X':val_X,
               'Val_Y':val_Y,
               'Test_X':Test_Parameter,
               'Test_Y':Test_Value,
               'Weights':W,
               'Learning_Rate':0.01,
               'Max_Loops':5000,
               'Epsilon':0.00000001,
               'threshold':0.4,
               'decoy_rate':0.9,
               'eps':0.00000001,
               'Beta1':0.9,
               'Beta2':0.999
               }
```

```
    SGD_VL,SGD_test_accuracy=SGD(Parameter)
Momentum_VL,Momentum_test_accuracy=Momentum(Paramete
r)
    NAG_VL,NAG_test_accuracy=NAG(Parameter)
    Adagrad_VL,Adagrad_test_accuracy=Adagrad(Parameter)
    AdaDelta_VL,AdaDelta_test_accuracy=AdaDelta(Parameter)
    Adam_VL,Adam_test_accuracy=Adam(Parameter)
    print('All Time Used:{:0.2f}s'.format(time.time()-tic))

    plt.plot(np.arange(0,Parameter['Max_Loops']-1,1),
SGD_VL[0:Parameter['Max_Loops']-1], label='SGD')
    plt.plot(np.arange(0,Parameter['Max_Loops']-1,1),
Momentum_VL[0:Parameter['Max_Loops']-1], label='Momentum')
    plt.plot(np.arange(0, Parameter['Max_Loops'] - 1, 1),
NAG_VL[0:Parameter['Max_Loops'] - 1], label='NAG')
    plt.plot(np.arange(0, Parameter['Max_Loops'] - 1, 1),
Adagrad_VL[0:Parameter['Max_Loops'] - 1], label='Adagrad')
    plt.plot(np.arange(0, Parameter['Max_Loops'] - 1, 1),
AdaDelta_VL[0:Parameter['Max_Loops'] - 1], label='AdaDelta')
    plt.plot(np.arange(0, Parameter['Max_Loops'] - 1, 1),
Adam_VL[0:Parameter['Max_Loops'] - 1], label='Adam')
    plt.xlabel('loops')
    plt.ylabel('loss')
    plt.title('Validation Loss')
    plt.legend()
    plt.show()


    plt.plot(np.arange(0,Parameter['Max_Loops']-1,1),
SGD_test_accuracy[0:Parameter['Max_Loops']-1], label='SGD')
    plt.plot(np.arange(0,Parameter['Max_Loops']-1,1),
Momentum_test_accuracy[0:Parameter['Max_Loops']-1],
label='Momentum')
    plt.plot(np.arange(0, Parameter['Max_Loops'] - 1, 1),
NAG_test_accuracy[0:Parameter['Max_Loops'] - 1], label='NAG')
    plt.plot(np.arange(0, Parameter['Max_Loops'] - 1, 1),
Adagrad_test_accuracy[0:Parameter['Max_Loops'] - 1],
label='Adagrad')
    plt.plot(np.arange(0, Parameter['Max_Loops'] - 1, 1),
AdaDelta_test_accuracy[0:Parameter['Max_Loops'] - 1],
label='AdaDelta')
    plt.plot(np.arange(0, Parameter['Max_Loops'] - 1, 1),
Adam_test_accuracy[0:Parameter['Max_Loops'] - 1], label='Adam')
```

```
plt.xlabel('loops')
plt.ylabel('loss')
plt.title('Test Accuracy')
plt.legend()
plt.show()
```

2. Linear Classification and Gradient Descent

(1) Function: loss

```
def Loss(X,Y,W,b,m_lambda):
    loss=0.5 * m_lambda * W.transpose().dot(W)
    for i in range(X.shape[0]):
        Tensor = Y[i][0] * (W.transpose().dot(X[i]) + b)
        if Tensor < 1:
            loss=loss+1-Tensor
        else:
            loss=loss+0
    return loss
```

(2) Function: compare

```
def compare(X,Y,W,b,i):
    if Y[i][0]*(W.transpose().dot(X[i])+b)<1:
        return 1
    else:
        return 0
```

(3) Function: Grad

```
def Grad(X,Y,W,b,m_lambda):
    grad=m_lambda*W
    for i in range(X.shape[0]):
        grad=grad-(compare(X,Y,W,b,i)*Y[i]*X[i]).reshape(X.shape[1],1)
    return grad
```

(4) Function: corrate_rate

```python
def corrate_rate(X,Y,W,b,threshold):
    count=0
    temp=Y*(X.dot(W)+b)
    for j in temp:
        if j>=0.02:
            count+=1
        else:
            continue
    rate=count/temp.shape[0]
    return rate
```

(5) **Function: Draw**

```python
def Draw(loops,train_loss,validation_loss,train_accuracy,val_accuracy):
    print('Drawing...')
    #the loss
    plt.plot(np.arange(0,loops-1,1), train_loss[0:loops-1], label='Train Loss')
    plt.plot(np.arange(0,loops-1,1), validation_loss[0:loops-1], label='Validation Loss')
    plt.xlabel('loops')
    plt.ylabel('loss')
    plt.title('Loss')
    plt.legend()
    plt.show()

    #the accuracy
    plt.plot(np.arange(0,loops-1,1), train_accuracy[0:loops-1], label='Train Accuracy')
    plt.plot(np.arange(0,loops-1,1), val_accuracy[0:loops-1], label='Validation Accuracy')
    plt.xlabel('loops')
    plt.ylabel('Accuracy')
    plt.title('Accuracy')
    plt.legend()
    plt.show()
    print('Draw Completed')
```

G. Selection of validation

1. Logistic Regression and Gradient Descent

I use Cross-Validation as the method to validate the result, during the whole experiment, the size of my validation set is 30%.

2. Linear Classification and Gradient Descent

I use Cross-Validation as the method to validate the result, during the whole experiment, the size of my validation set is 25%.

H. The initialization method of model parameters:

In two experiments, we all initialize the parameters in the same way:

W: Randomly initialize it in range of 0 and 1
the others: initialize according to experience

I. The selected loss function and its derivatives:

1. Logistic Regression and Gradient Descent

(1) Loss Function

$$J(\mathbf{w}) = -\frac{1}{n}\left[\sum_{i=1}^{n} y_i \log h_{\mathbf{w}}(\mathbf{x}_i) + (1-y_i)\log\left(1-h_{\mathbf{w}}(\mathbf{x}_i)\right)\right]$$

(2) Gradient

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \frac{1}{n}\sum_{i=1}^{n}\left(h_{\mathbf{w}}(\mathbf{x}_i) - y\right)\mathbf{x}_i$$

2. Linear Classification and Gradient Descent

(1) Loss Function

$$F(X)=\frac{W^2}{2}+C\sum_{i=1}^{N} max(0,1-y_i(W^T*X_i+b))$$

(2) Gradient

$$Gradient=\begin{cases} Gradient-Y_i*X_i, \wedge Y_i*\left(W^T*X_i+b\right)<1 \\ Gradient, \wedge Y_i*\left(W^T*X_i+b\right)\geq 1 \end{cases}$$

J. Experimental results and curve:

1. Linear Regression and Gradient Descent

(1) Hyper-parameter selection

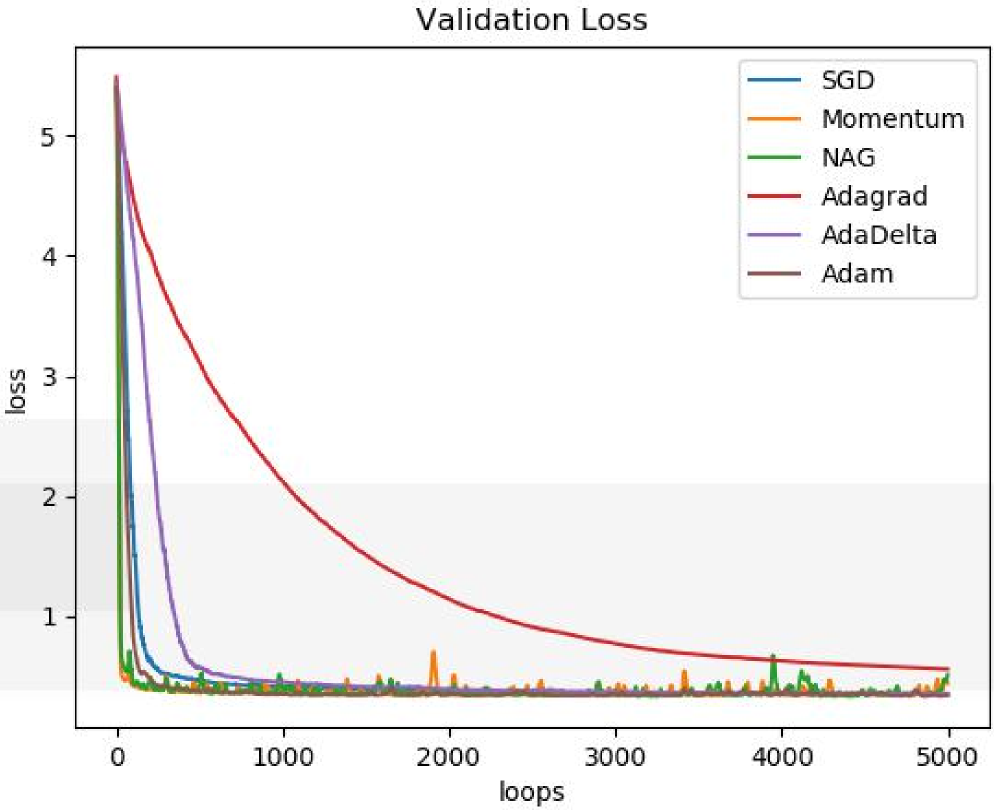
```
Parameter={'Train_X':train_X,
           'Train_Y':train_Y,
           'Val_X':val_X,
           'Val_Y':val_Y,
           'Test_X':Test_Parameter,
           'Test_Y':Test_Value,
           'Weights':W,
           'Learning_Rate':0.025,
```

```
'Max_Loops':1500,
'Epsilon':0.00000001,
'threshold':0.4,
'decoy_rate':0.9,
'eps':0.00000001,
'Beta1':0.9,
'Beta2':0.999,
'lambda': 0.01,    # regularize, C=1/lambda
'b': 0.01
}
```

(2) **Assessment Results (based on selected validation)**

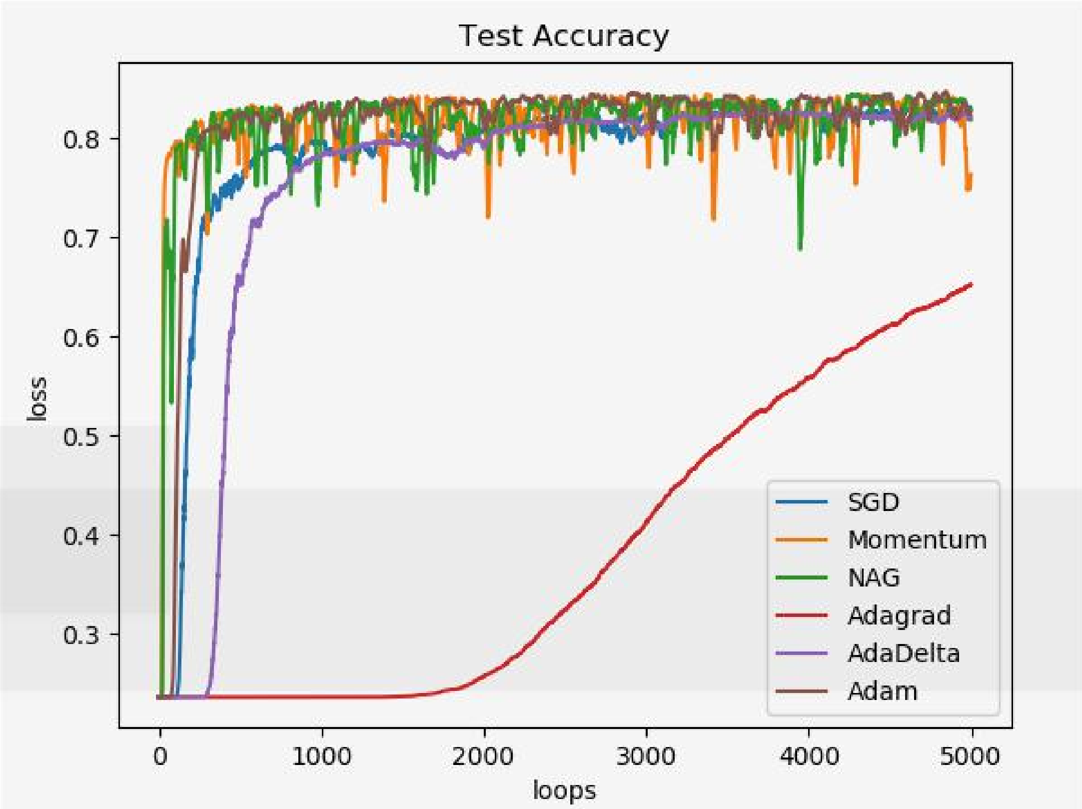
Train loss is about 0.1 bigger than validation loss

(3) **Loss curve**



2. Linear Classification and Gradient Descent

(1) **Hyper-parameter selection**



(2) **Assessment Results (based on selected validation)**

Train loss is about 0.15 bigger than validation loss, the accuracy in validation set will better than that in train set.

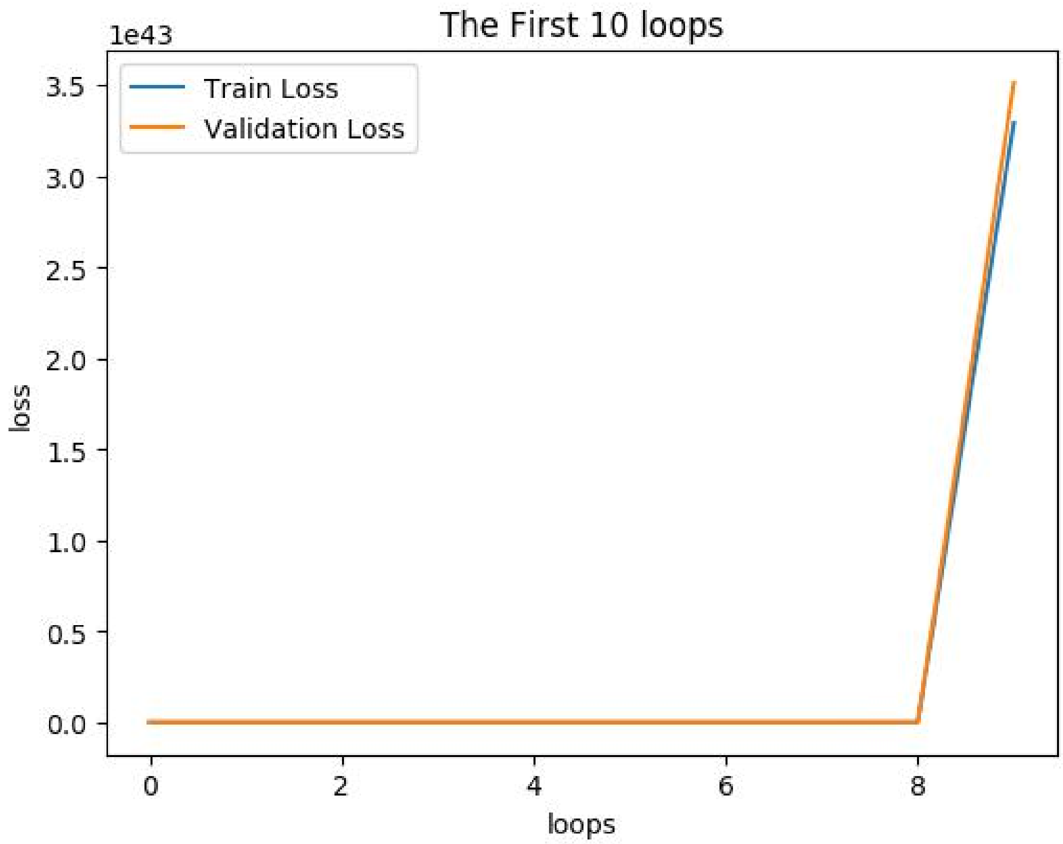
(3) **Predicted Results (Best Results)**

Loss_Train: [0.34371135] **Loss_Validation:** [0.2769692]
Accuracy: Train: 0.8588007736943907, **Validation:** 0.8728323699421965

K. Results analysis

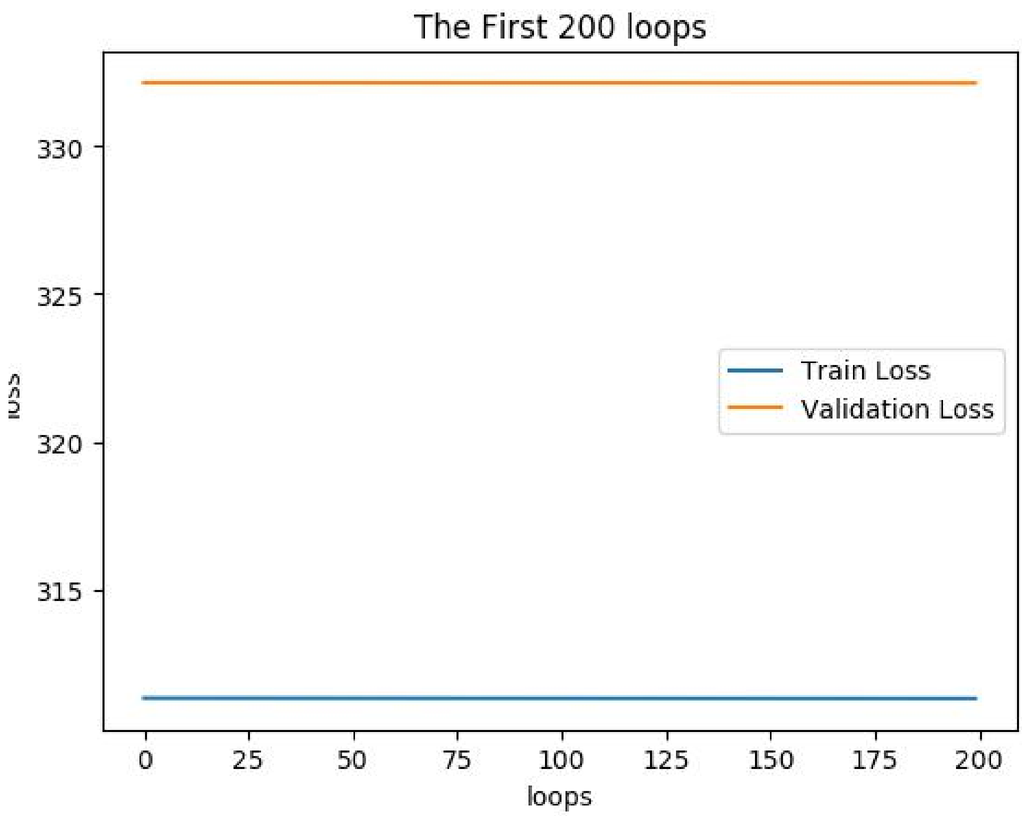
1. Linear Regression and Gradient Descent

(1) **'Weights'**: 0.085, **others the same**



The curve won't converge, because the learning rate is too big, it will always miss the local/global minimum and go away from it.

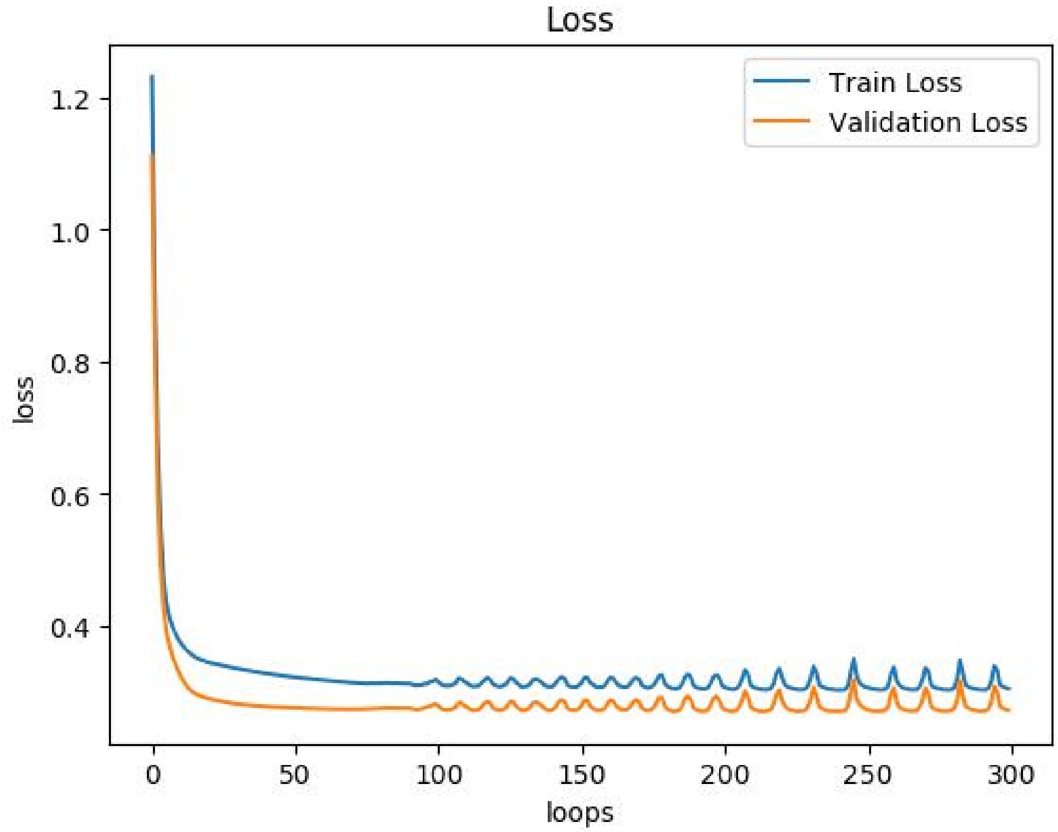
(2) 'Weights': 0.000000000085, others the same



The curve won't converge either, because W is too small that it will take a very long time period.

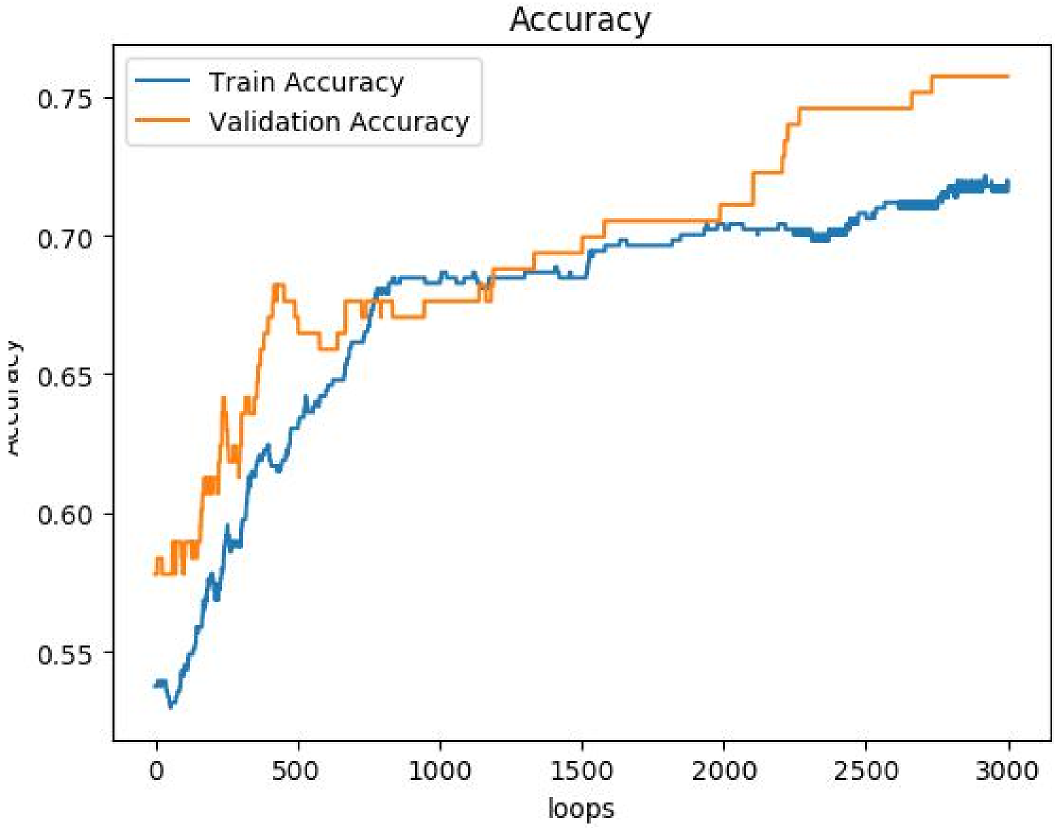
2. Linear Classification and Gradient Descent

(1) 'learning_rate': 0.00085, others the same



We can see clearly that at the last, our curve welter, I think the reason is the learning rate is a little big so it will go from a point close to the minimum to the symmetry point and then go back, it's rather interesting.

(2) **'threshold':1, others the same**



We can find the accuracy is much less than the curve of 'threshold':0, I think this is due to there are little outliers in the dataset.

L. Similarities and differences

I think both problem are trying to find a loss function and using some method to make it smaller and smaller. In this way, we can use gradient decent in the two problem. However, linear regression focus on finding a function to solve the problem with successive values while classification focus on problem with discrete value and make classification. The loss they used are also different.

M. Summary

I do believe we can overcome all the difficulties on our way learning machine learning, we should learn more from teachers, papers, books, and even from the web. We should also practice more, we can join kaggle or other competitions in the AI field. I think with the help of computer science, we will have a brighter future and I hope I can be one of the scientists and make something for the world.