The Experiment Report of
Machine Learning

College	**Software College**
Subject	**Software Engineering**
Members	李彩利
Student ID	201530611920
E-mail	2561170837@qq.com
Tutor	QingyaoWu
Date submitted	2017. 12 . 15

1. Topic: Logistic regression, linear classification and stochastic gradient descent

2. Time: 12.8

3. Reporter:李彩利

4. Purposes:

1. Comparison of the differences and connections between gradient descent and stochastic gradient descent.
2. Compare and understand the difference and relationship between logistic regression and linear classification.
3. Further understand the principle of SVM and practice on larger data.

5. Data sets and data analysis:

The experiment uses a9a Data from LIBSVM Data, which contains 32561/16281 (testing) samples, each with 123/123 (testing) properties. Divide data sets into training sets and validation sets

6. Experimental steps:

Logistic regression and stochastic gradient descent

1. Read the experimental training set and verification set.
2. The parameter initialization of the logistic regression model can be considered to be full zero initialization, random initialization or normal distribution initialization.

3. Select the Loss function and take the derivative, and see the PPT for details.

4. Obtain the gradient of partial sample to the Loss function.

5. Use different optimization methods to update model parameters (NAG, RMSProp, AdaDelta and Adam).

6. Select the appropriate threshold and verify that the result of the centralized calculation is positive and negative. The Loss function values of different optimization methods are tested on the validation set.

7. Repeat steps 4-6 times to draw the value of the Loss function and the variation of the number of iterations.

Linear classification and stochastic gradient descent

1. Read the experimental training set and verification set.

2. Support vector machine model parameter initialization, can consider full zero initialization, random initialization or normal distribution initialization.

3. Select the Loss function and take the derivative, and see the PPT for details.

4. Obtain the gradient of partial sample to the Loss function.

5. Use different optimization methods to update model parameters (NAG, RMSProp, AdaDelta and Adam).

6. Select the appropriate threshold and verify that the result of the centralized calculation is positive and negative. The Loss function values of different optimization methods are tested on the validation set.

7. Repeat steps 4-6 times to draw the value of the Loss function and the variation of the number of iterations.

7. Code:

逻辑回归和随机梯度下降

```
from sklearn.datasets import load_svmlight_file

from sklearn.model_selection import train_test_split

from numpy import *

import matplotlib.pyplot as plt

import numpy as np

from numpy import random

import math


def get_data():

    data =
load_svmlight_file("F:\\BaiduNetdiskDownload\\machinelearning\\a9a.txt")

    return data[0], data[1]


def get_data_t():

    data =
load_svmlight_file("F:\\BaiduNetdiskDownload\\machinelearning\\a9a.t")

    return data[0], data[1]


def function(x,w,b):

    z = 0

    x = np.ndarray.tolist(x)


    for i in range(len(x)):

        z += x[0][i] * w[i]
```

```
z += b
f = 1/(1+np.exp(-z))

return f

x_train,y_train=get_data()
x_test,y_test=get_data_t()

w = np.random.rand(123)
x_train = x_train.todense()
x_test = x_test.todense()
feature=123
x_train_len=len(x_train)
x_test_len=len(x_test)



#########################################################NAG
b = np.random.rand()
count1 = 30
lr=0.1
v = zeros([123,1])
r = 0.9
train_new_loss1 = []
test_new_loss1 = []
```

```
loss = -1*(sum(((1+y_train[i]) * math.log(function(x_train[i],w,b)) + (1 - y_train[i]) * math.log(1-function(x_train[i],w,b))) for i in range(x_train_len)))/x_train_len

for k in range(count1):

i = random.randint(x_train_len)

for j in range(feature):

g = (function(x_train[i,:],w,b)-y_train[i])*x_train[i,j]

v[j] = r*v[j]-lr*g

w[j] += v[j] * lr

train_loss1 = -1*sum(((1+y_train[i]) * math.log(function(x_train[i],w,b)) + (1 - y_train[i]) * math.log(1-function(x_train[i],w,b))) for i in range(x_train_len))/x_train_len

test_loss1    = -1*sum(((1+y_test[i]) * math.log(function(x_test[i], w, b))+(1-y_test[i])*math.log(1-function(x_test[i], w, b)))for i in range(x_test_len))/x_test_len

train_new_loss1.append(train_loss1)

test_new_loss1.append(test_loss1)

####################################################Adadelta

b = 0
```

```
count2 = 30
lr=0.5
G = zeros([123,1])
p = 0.8
train_new_loss2 = []
test_new_loss2 = []

loss = -1*(sum(((1+y_train[i]) * math.log(function(x_train[i],w,b)) + (1 - y_train[i]) * math.log(1-function(x_train[i],w,b))) for i in range(x_train_len)))/x_train_len
#print(loss)
for k in range(count2):

    i = random.randint(x_train_len)

    for j in range(feature):

        g = (function(x_train[i],w,b)-y_train[i])*x_train[i,j]

        G[j]= p*G[j]+(1-p)*((g)**2)
        w[j] -= lr/math.sqrt(G[j]+0.0000000001)*g

    train_loss2 = -1*(sum(((1+y_train[i]) * math.log(function(x_train[i],w,b)) + (1 - y_train[i]) * math.log(1-function(x_train[i],w,b))) for i in range(x_train_len)))/x_train_len
```

```
test_loss2    = -1*sum(((1+y_test[i]) * math.log(function(x_test[i], w, b))+(1-y_test[i])*math.log(1-function(x_test[i], w, b)))for i in range(x_test_len))/x_train_len

train_new_loss2.append(train_loss2)
test_new_loss2.append(test_loss2)

##############################RMSProp
b = 0
count3 = 40
lr=0.3
G = zeros([123,1])
p = 0.6
train_new_loss3 = []
test_new_loss3 = []

loss = -1*(sum(((1+y_train[i]) * math.log(function(x_train[i],w,b)) + (1 - y_train[i]) * math.log(1-function(x_train[i],w,b))) for i in range(x_train_len)))/x_train_len
#print(loss)
for k in range(count3):

i = random.randint(x_train_len)

for j in range(feature):
```

```
g = (function(x_train[i],w,b)-y_train[i])*x_train[i,j]

G[j]= p*G[j]+(1-p)*((g)**2)
w[j] -= lr/math.sqrt(G[j]+0.0000000001)*g

train_loss3 = -1*(sum(((1+y_train[i]) * math.log(function(x_train[i],w,b)) +
(1 - y_train[i]) * math.log(1-function(x_train[i],w,b))) for i in
range(x_train_len)))/x_train_len
test_loss3    = -1*sum(((1+y_test[i]) * math.log(function(x_test[i], w,
b))+(1-y_test[i])*math.log(1-function(x_test[i], w, b)))for i in
range(x_test_len))/x_train_len

train_new_loss3.append(train_loss3)
test_new_loss3.append(test_loss3)

# print(train_new_loss1)
# print(test_new_loss1)
plt.figure(figsize=(8,6))
plt.xlabel('Iteration times')
plt.ylabel('Loss')
#plt.plot(range(count), train_new_loss1, 'o-', label=u"Training Set")
plt.plot(range(count1), test_new_loss1, 'r-', label=u"NAG_Testing_set")
plt.plot(range(count2), test_new_loss2, 'g-', label=u"Adadelta_Testing_set")
plt.plot(range(count3), test_new_loss3, 'o-', label=u"RMSProp_Testing_set")
# plt.plot(range(count), train_new_loss2, 'o-', label=u"Training Set")
# plt.plot(range(count), test_new_loss2, 'r-', label=u"Testing set")
```

```
plt.legend()
plt.grid()
plt.show()
```

线性分类和随机梯度下降：

```
from sklearn.datasets import load_svmlight_file
from sklearn.model_selection import train_test_split
from numpy import *
import matplotlib.pyplot as plt
import numpy as np
from numpy import random
import math

def get_data():
    data = load_svmlight_file("F:\\BaiduNetdiskDownload\\machinelearning\\a9a.txt")
    return data[0], data[1]

def get_data_t():
    data = load_svmlight_file("F:\\BaiduNetdiskDownload\\machinelearning\\a9a.t")
    return data[0], data[1]

def function(x,w,b):
    z = 0
    x = np.ndarray.tolist(x)

    for i in range(len(x)):
        z += x[0][i] * w[i]
    z += b
    return z

x_train,y_train=get_data()
x_test,y_test=get_data_t()
b = np.random.rand() #random
w = np.random.rand(123)
x_train = x_train.todense()
x_test = x_test.todense()
x_train_len=len(x_train)
x_test_len=len(x_test)

######################################NAG
count = 10
```

```
lr=0.1
feature=123
r = 0.9
v=np.random.rand(123)*0
train_new_loss = []
test_new_loss = []
w2 = math.sqrt(sum((w[i])**2 for i in range(feature)))
loss = r*w2 + (sum(max(0,1-y_train[i]*function(x_train[i],w,b)) for i in range(x_train_len)))/x_train_len

for k in range(count):

    i = random.randint(x_train_len)
    for j in range(feature):
        g = w[j]+r*v[j]+(-y_train[i])*(function(x_train[i],w,b)*y_train[i]<1)*x_train[i,j]


        v[j] = r*v[j]-lr*g
        w[j] += v[j] * lr

    w2 = math.sqrt(sum((w[i])**2 for i in range(feature)))
    train_loss = r*w2 + (sum(max(0,1-y_train[i]*function(x_train[i],w,b)) for i in range(x_train_len)))/x_train_len
    test_loss = r*w2 + (sum(max(0,1-y_test[i]*function(x_test[i],w,b)) for i in range(x_test_len)))/x_test_len

    train_new_loss.append(train_loss)
    test_new_loss.append(test_loss)

############################################RMSProp
count = 10
lr=0.02
h=0.5
r = 0.8
feature=123
G = zeros([123,1])
v=np.random.rand(123)*0
train_new_loss2 = []
test_new_loss2 = []
w2 = math.sqrt(sum((w[i])**2 for i in range(feature)))
loss = r*w2 + (sum(max(0,1-y_train[i]*function(x_train[i],w,b)) for i in range(x_train_len)))/x_train_len
```

```
    for k in range(count):

        i = random.randint(x_train_len)
        for j in range(feature):
            g = w[j]+(-y_train[i])*(function(x_train[i],w,b)*y_train[i]<1)*x_train[i,j]

            G[j]= h*G[j]+(1-h)*(g**2)
            if G[j]<0:
                print("!!!")
            w[j] -= lr/math.sqrt(G[j]+0.0000000001)*g

        w2 = math.sqrt(sum((w[i])**2 for i in range(feature)))
        train_loss2 = r*w2 + (sum(max(0,1-y_train[i]*function(x_train[i],w,b)) for i in range(x_train_len)))/x_train_len
        test_loss2 = r*w2 + (sum(max(0,1-y_test[i]*function(x_test[i],w,b)) for i in range(x_test_len)))/x_test_len

        train_new_loss2.append(train_loss2)
        test_new_loss2.append(test_loss2)

    ##########################################Adadelta
    count = 10
    lr=0.05
    h=0.8
    r = 0.9
    feature=123
    G = zeros([123,1])
    v=np.random.rand(123)*0
    train_new_loss3 = []
    test_new_loss3 = []
    w2 = math.sqrt(sum((w[i])**2 for i in range(feature)))
    loss = r*w2 + (sum(max(0,1-y_train[i]*function(x_train[i],w,b)) for i in range(x_train_len)))/x_train_len

    for k in range(count):

        i = random.randint(x_train_len)
        for j in range(feature):
            g = w[j]+(-y_train[i])*(function(x_train[i],w,b)*y_train[i]<1)*x_train[i,j]

            G[j]= h*G[j]+(1-h)*(g**2)
            if G[j]<0:
```

```
                print("!!!")
            w[j] -= lr/math.sqrt(G[j]+0.0000000001)*g

        w2 = math.sqrt(sum((w[i])**2 for i in range(feature)))
        train_loss3 = r*w2 + (sum(max(0,1-y_train[i]*function(x_train[i],w,b)) for i in range(x_train_len)))/x_train_len
        test_loss3 = r*w2 + (sum(max(0,1-y_test[i]*function(x_test[i],w,b)) for i in range(x_test_len)))/x_test_len

        train_new_loss3.append(train_loss3)
        test_new_loss3.append(test_loss3)

    plt.figure(figsize=(8,6))
    plt.xlabel('Stochastic Gradient descent - Iteration times')
    plt.ylabel('Loss')
    plt.plot(range(count), test_new_loss, 'r-', label=u"NAG_Testing_set")
    plt.plot(range(count), test_new_loss2, 'g-', label=u"RMSProp_Testing_set")
    plt.plot(range(count), test_new_loss3, 'o-', label=u"Adadelta_Testing_set")
    plt.legend()
    plt.grid()
    plt.show()
```

8. The initialization method of model parameters: Random initialization

9. The selected loss function and its derivatives:

Linear Classification:

Loss: $r \cdot \|w\|^2 + \frac{1}{n}\sum_{i=1}^{n} \max(0, 1 - y_i(w^T x_i + b))$

$\nabla L = -\delta(\hat{y} \cdot f(x) < 1)\, \hat{y} \cdot x$

Regression:

Loss: $-\frac{1}{n}\left[\sum_{i=1}^{n} (1+y) \log h_w(x_i) + (1-y_i) \log(1 - h_w(x_i))\right]$

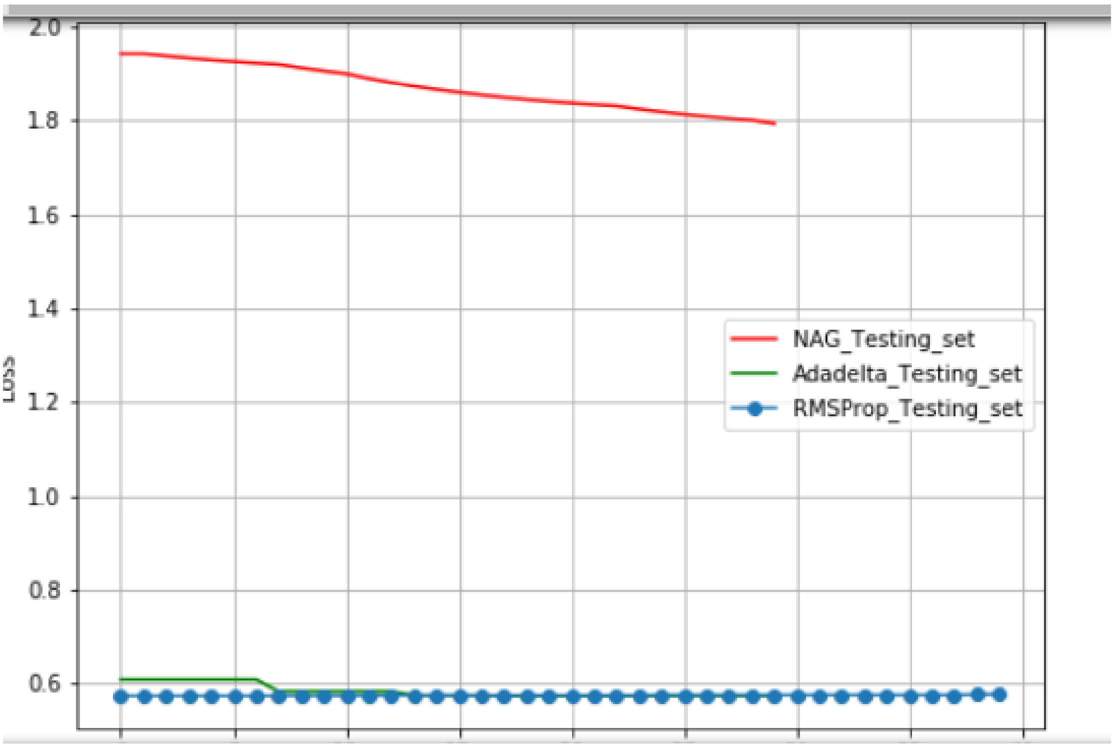
$\nabla L = (h(x) - y)\, x$

10. Experimental results and curve:(Fill in this content for various methods of gradient descent respectively)

Hyper-parameter selection: 0. 6、0. 8、0. 6

Predicted Results (Best Results):0.59

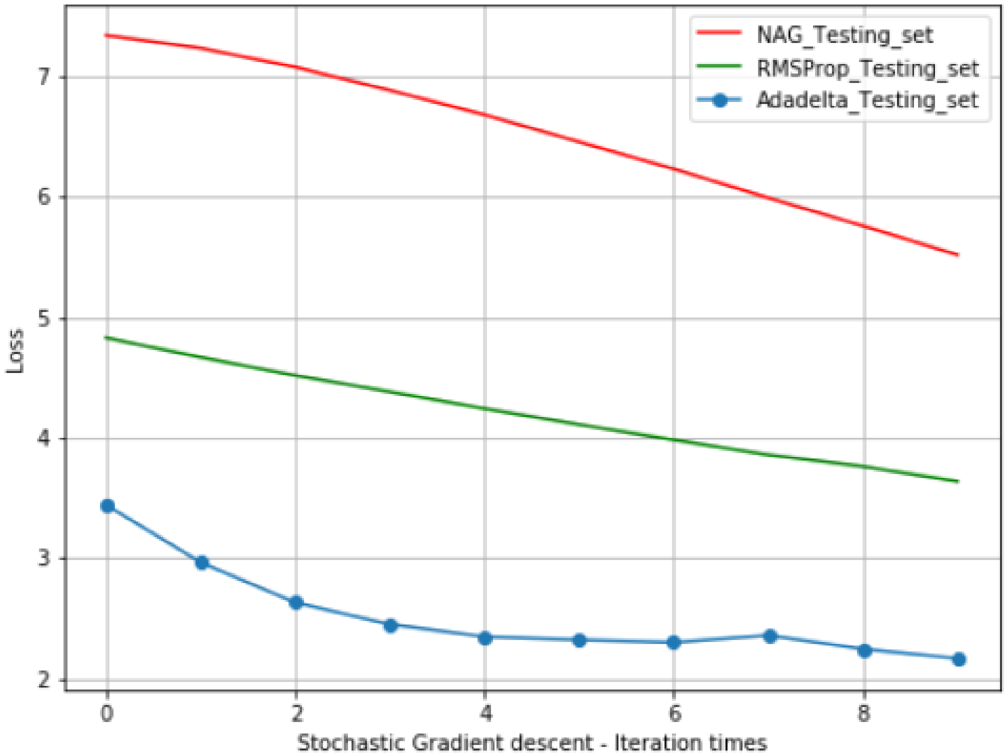
Loss curve:



Hyper-parameter selection: 0.9、0.8、0.9

Predicted Results (Best Results):2.1

Loss curve:



11. Results analysis: Logic regression and linear classification of loss are getting smaller and smaller, and the parameters such as NAG, RMSProp, AdaDelta and Adam are updated to speed up the process

12. Similarities and differences between logistic regression and linear classification: From the point of the objective function, the difference is that logistic regression is the logistical loss, the SVM is the hinge loss. The purpose of these two loss function is to increase the weight of the data points for a greater influence on the classification, and classification relationship smaller proportion of data points. The processing of the SVM method is only considered the support vectors, is also the most relevant and classification of a few points, to learn a classifier, and logistic regression through nonlinear mapping, greatly reduced the weight of the plane distant point from the classification, promoted and classification of the relative weights of associated data point. The fundamental purpose is the same. In addition, according to the needs, two methods can increase the different regularization item, such as l1, l2, and so on. But the logistic regression model is more simple, relatively good understanding, to implement, especially the large-scale linear classification is more convenient. And the understanding of the SVM and the optimization is relatively complicated.

13. Summary: I learned the difference between logistic regression and linear regression, stochastic gradient descent and gradient descent