华南理工大学

South China University of Technology

The Experiment Report of Machine Learning

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:
Shumin Yuan

Supervisor:
Qingyao Wu

Student ID：
201530613528

Grade:
Undergraduate

December 12, 2017

Linear Regression, Linear Classification and Gradient Descent

Abstract—

I. Introduction

Today,I try to solve two questions which named logistic regression and linear classification.This two question are all to solve classification questions.

There are the terminals of this two questions:

1. Compare and understand the difference and relations between gradient descent and stochastic gradient descent.
2. Compare and understand the difference and relations between linear classification and logistic regression.
3. Understand the concept of SVM further and using it to big data.

The lab use a9a data-set witch contains 32561 training data and 16281 testing data.Every data has 123 feature.

This lab is based on python3.

II. Methods and Theory

Logistic Regression

Loss function:

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

Gradient:

$$\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = \left(h_{\mathbf{w}}(\mathbf{x}) - y\right)\mathbf{x}$$

Linear Regression

Loss function

$$\frac{\|\mathbf{w}\|^2}{2} + C\sum_{i=1}^{N}\max(0, 1-y_i(\mathbf{w}^\top x_i + b))$$

Gradient:

$$\frac{\partial f(\mathbf{w},b)}{\mathbf{w}} = \mathbf{w} + C\sum_{i=1}^{N} g_{\mathbf{w}}(\mathbf{x}_i)$$

$$\frac{\partial f(\mathbf{w},b)}{b} = C\sum_{i=1}^{N} g_b(\mathbf{x}_i)$$

III. Experiment

Steps:

Logistic Regression:

1.Reading lab data-set

2.Initialing the model of Logic Regression

3.Choosing loss function and gradient

4.Using different methods to update the parameter

Linear Classification:

1. Reading lab data-set
2. Initialing the model of SVM
3. Choosing loss function and gradient
4. Using different methods to update the parameter

Data-set:

This lab use the data-set a9a,witch contains 32561 training data and 16281 testing data.Every data has 123 feature.

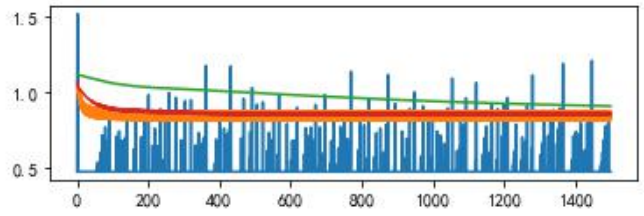
Result:

Logistic regression:

Code of gradient descent:

```
def sigmoid(x):
    return 1.0/(1+exp(-x))


def Gw_NAG(w,x,y,vw,alpha,r):
    sumG=0
    x_temp=x*w
    dw=-((y-log(sigmoid(x*(w-(r*vw).T)))).T*x).T
    vw=(r*vw).T+alpha*dw
    w=w-vw
    return w


def Gw_RMSProp(w,x,y,vw,alpha,r):
    sumG=0
    l=0
    x_temp=x*w
    dw=-(((y-log(sigmoid(x_temp)))).T*x).T
    vw=r*vw+(1-r)*np.multiply(dw,dw)
    w=w-np.multiply(alpha/(sqrt(vw+(1e-8))),dw)
    return w


def Gw_AdaDelta(w,x,y,vw,r,tw):
    sumG=0
    l=0
    x_temp=x*w
```

```
    dw=-(((y-log(sigmoid(x_temp)))).T*x).T
    vw=r*vw+(1-r)*np.multiply(dw,dw)
    Dw=np.multiply(-(sqrt(tw+(1e-8))/sqrt(vw+(1e-8))),dw)
    w=w+Dw
    tw=r*tw+np.multiply((1-r)*Dw,Dw)
    return w



def Gw_Adam(w,x,y,size,b,vw,alpha,C,r,mw,i):
    beta=0.9
    sumG=0
    l=0
    x_temp=x*w
    dw=-(((y-log(sigmoid(x_temp)))).T*x).T
    mw=beta*mw+(1-beta)*dw
    vw=r*vw+(1-r)*multiply(dw,dw)

w=w-(alpha*(sqrt(1-r**(i+1)))/(1-beta**(i+1)))*(mw)/(sqrt(v
w+(1e-8)))
    return w

def validation(w,x,y,size):
    L=[0 for i in range(size)]
    l=0
    x_temp=x*w
    for i in range (size):

l=l-((np.mat(y[i,0]).T*sigmoid(x_temp[i,0])+(1-y[i,0]*sigmoi
d(1-x_temp[i,0]))))
    #print(sigmoid(x_temp[i]).T)
    return -l[0,0]/size
```

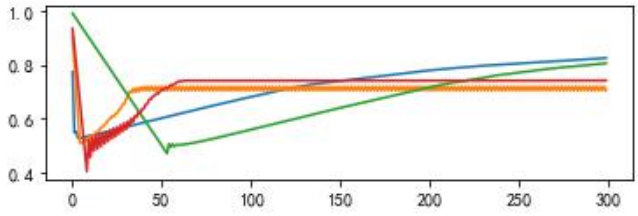


Linear classification:
Code of gradient descent:
NAG

```
def Gw_NAG(w,x,y,size,b,vw,alpha,C,r):
    G=[0 for i in range(123)]
    L=[0 for i in range(size)]
    g=[0 for i in range(123)]
    sumG=0
    l=0
    x_temp=x*w
    for i in range(size):
        if(1-(y[i,0])*((x_temp)[i,0])-b>=0):
            for j in range(123):
                G[j]=G[j]-y[i,0]*x[i,j]
            L[i]=1-(y[i,0])*((x_temp)[i,0])-b
    for i in range(size):
        l=l+L[i]
    l=l*0.1
    for i in range(14):
        l=l+(w[i,0]**2)/2
    g=np.mat(G)
    g=g.T
    dw=w+C*g-r*vw
    vw=r*vw+alpha*dw
    w=w-vw
    return w
def Gb_NAG(w,x,y,size,b,vb,alpha,C,r):
        G=0
        g=[0 for i in range(123)]
        l=0
        x_temp=x*w
        for i in range(size):
            if(1-(y[i,0])*((x_temp)[i,0])-b>=0):
                G=G-y[i,0]
        db=C*G-r*vb
        vb=r*vb+alpha*db
        b=b-vb
        return b

RMSProp
def Gw_RMSProp(w,x,y,size,b,vw,alpha,C,r):
    G=[0 for i in range(123)]
    L=[0 for i in range(size)]
    g=[0 for i in range(123)]
    sumG=0
    l=0
    x_temp=x*w
    for i in range(size):
        if(1-(y[i,0])*((x_temp)[i,0])-b>=0):
            for j in range(123):
                G[j]=G[j]-y[i,0]*x[i,j]
            L[i]=1-(y[i,0])*((x_temp)[i,0])-b
    for i in range(size):
        l=l+L[i]
    l=l*0.1
    for i in range(14):
        l=l+(w[i,0]**2)/2
    g=np.mat(G)
    g=g.T
    dw=w+C*g
    vw=r*vw+(1-r)*np.multiply(dw,dw)
    w=w-np.multiply(alpha/(sqrt(vw+(1e-8))),dw)
    return w

def Gb_RMSProp(w,x,y,size,b,vb,alpha,C,r):
    G=0
    g=[0 for i in range(123)]
    l=0
    x_temp=x*w
    for i in range(size):
        if(1-(y[i,0])*((x_temp)[i,0])-b>=0):
```

```
            G=G-y[i,0]
    db=C*G
    vb=r*vb+(1-r)*np.multiply(db,db)
    b=b-np.multiply(alpha/(sqrt(vb+(1e-8))),db)
    return b

AdaDelta
def Gw_AdaDelta(w,x,y,size,b,vw,C,r,tw):
    G=[0 for i in range(123)]
    L=[0 for i in range(size)]
    g=[0 for i in range(123)]
    sumG=0
    l=0
    x_temp=x*w
    for i in range(size):
        if(1-(y[i,0])*((x_temp)[i,0])-b>=0):
            for j in range(123):
                G[j]=G[j]-y[i,0]*x[i,j]
            L[i]=1-(y[i,0])*((x_temp)[i,0])-b
    for i in range(size):
        l=l+L[i]
    l=l*0.1
    for i in range(14):
        l=l+(w[i,0]**2)/2
    g=np.mat(G)
    g=g.T
    dw=w+C*g
    vw=r*vw+(1-r)*np.multiply(dw,dw)
    Dw=np.multiply(-(sqrt(tw+(1e-8))/sqrt(vw+(1e-8))),dw)
    w=w+Dw
    tw=r*tw+np.multiply((1-r)*Dw,Dw)
    return w

def Gb_AdaDelta(w,x,y,size,b,vb,C,r,tb):
    G=0
    g=[0 for i in range(123)]
    l=0
    x_temp=x*w
    for i in range(size):
        if(1-(y[i,0])*((x_temp)[i,0])-b>=0):
            G=G-y[i,0]
    db=C*G
    vb=r*vb+(1-r)*np.multiply(db,db)
    Db=np.multiply(-(sqrt(tb+(1e-8))/sqrt(vb+(1e-8))),db)
    b=b+Db
    tb=r*tb+np.multiply((1-r)*Db,Db)
    return b

Adam
def Gw_Adam(w,x,y,size,b,vw,alpha,C,r,mw,i):
    G=[0 for i in range(123)]
    L=[0 for i in range(size)]
    g=[0 for i in range(123)]
    beta=0.9
    sumG=0
    l=0
    x_temp=x*w
    for i in range(size):
        if(1-(y[i,0])*((x_temp)[i,0])-b>=0):
            for j in range(123):
                G[j]=G[j]-y[i,0]*x[i,j]
            L[i]=1-(y[i,0])*((x_temp)[i,0])-b
    for i in range(size):
        l=l+L[i]
    l=l*0.1
    for i in range(14):
        l=l+(w[i,0]**2)/2
    g=np.mat(G)
    g=g.T
    dw=w+C*g
    mw=beta*mw+(1-beta)*dw
    vw=r*vw+(1-r)*multiply(dw,dw)

w=w-(alpha*(sqrt(1-r**(i+1)))/(1-beta**(i+1)))*(mw)/(sqrt(v
w+(1e-8)))
    return w

def Gb_Adam(w,x,y,size,b,vb,alpha,C,r,mb,i):
    G=0
    g=[0 for i in range(123)]
    l=0
    beta=0.9
    x_temp=x*w
    for i in range(size):
        if(1-(y[i,0])*((x_temp)[i,0])-b>=0):
            G=G-y[i,0]
    db=C*G
    mb=beta*mb+(1-beta)*db
    vb=r*vb+(1-r)*multiply(db,db)

b=b-(alpha*(sqrt(1-r**(i+1)))/(1-beta**(i+1)))*(mb)/(sqrt(vb+
(1e-8)))
    return b
```

Train 150times (study_rate=0.02 0.01 0.002)



IV. CONCLUSION

i. AdaDelta isn't dependent on study-rate
ii. For different model,the four method has different effect.
iii. NAG,RMSProp,AdaDelta and Adam can speed up the process of gradient descent.
iv. This lab lets me understand classification question further.