South China University of Technology

The Experiment Report of
Machine Learning

College Software College

Subject Software Engineering

Members 林智远

Student ID 201530612323

E-mail 529880551@qq.com

Tutor 谭明奎

Date submitted 2017. 12 . 15

1. **Topic:** Logistic Regression, Linear Classification and Stochastic Gradient Descent

2. **Time:** 2017.12 .9

3. **Reporter:林智远**

4. **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.

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

Experiment uses a9a of LIBSVM Data, including 32561/16281(testing) samples and each sample has 123/123 (testing).But the testing data loses its 123th column.

6. **Experimental steps:**

Logistic Regression and Stochastic Gradient Decrease：

1.Read experimental training set and verification set.

2.Logistic regression model parameter initialization, consider all-zero initialization, random initialization or normal distribution initialization.

3.Select Loss function and its derivative, the process see courseware ppt.

4.Find the gradient of some samples to Loss function.

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

6.Select the appropriate threshold, will verify the centralized calculation results greater than the threshold marked as positive, otherwise negative. Test on the validation set and get the Loss function values for different optimization methods, and.

7.Repeat steps 4-6 for several times, plotting, and graphing the number of iterations.

Linear classification and stochastic gradient descent

1.Read experimental training set and verification set.

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

3.Select Loss function and its derivative, the process see courseware ppt. Find the gradient of some samples to Loss function.

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

5.Select the appropriate threshold, will verify the centralized calculation results greater than the threshold marked as positive, otherwise negative.

6.Test on the validation set and get the Loss function values for different optimization methods, and.

7.Repeat steps 4-6 for several times, plotting, and graphing the number of iterations.

7. Code:

Logistic Regression:

```
epoch =2000
Iteration=range(1,epoch+1)
L_NAG=[]
L_RMSProp=[]
L_AdaDelta=[]
L_Adam=[]
rand=[]
for i in range(1, epoch+1):
    rand.append(random.randint(0,m_train-100))
def logit(x):
    return 1/(1+np.exp(-x))
def NAG(W, γ , η ):
    v=0
    for i in range(1, epoch+1):
        j=rand[i-1]
        h=logit(np.dot(X_train[j:j+99],(W- γ *v).transpose()))
```

```
g=np.dot(X_train[j:j+99].transpose(),(h-y_train[j:j+99]))/100
            v=γ*v+η*g
            W=W-v
            h_test = logit(np.dot(X_test,W))
            J_test =
-(1/m_test)*np.sum(y_test*np.log(h_test)+(1-y_test)*np.log(1-h_test))
            L_NAG.append(J_test)
        h_test[h_test>0.5]=1
        h_test[h_test<0.5]=0
        count=0
        for l in range(len(y_test)):
            if h_test[l]==y_test[l]:
                count +=1
        print("NAG 的准确率为",count/m_test)
    def RMSProp(W,γ,η,ε):
        G=0
        for i in range(1, epoch+1):
            j=rand[i-1]
            h=logit(np.dot(X_train[j:j+99],W.transpose()))

g=np.dot(X_train[j:j+99].transpose(),(h-y_train[j:j+99]))/100
```

```
            G=γ*G+(1-γ)*np.dot(g.transpose(),g)
            W=W-(η/np.sqrt(G+ε))*g
            h_test = logit(np.dot(X_test,W))
            J_test =
-(1/m_test)*np.sum(y_test*np.log(h_test)+(1-y_test)*np.log(1-h_test))
            L_RMSProp.append(J_test)
        h_test[h_test>0.5]=1
        h_test[h_test<0.5]=0
        count=0
        for l in range(len(y_test)):
            if h_test[l]==y_test[l]:
                count +=1
        print("RMSProp 的准确率为",count/m_test)
    def AdaDelta(W,γ,ε):
        G=0
        Δ=0
        for i in range(1, epoch+1):
            j=rand[i-1]
            h=logit(np.dot(X_train[j:j+99],W.transpose()))

g=np.dot(X_train[j:j+99].transpose(),(h-y_train[j:j+99]))/100
            G=γ*G+(1-γ)*np.dot(g.transpose(),g)
```

```
            ΔW=-(np.sqrt(Δ+ε)/np.sqrt(G+ε))*g
            W=W+ΔW
            Δ=γ*Δ+(1-γ)*np.dot(ΔW.transpose(),ΔW)
            h_test = logit(np.dot(X_test,W))
            J_test =
-(1/m_test)*np.sum(y_test*np.log(h_test)+(1-y_test)*np.log(1-h_test))
            L_AdaDelta.append(J_test)
        h_test[h_test>0.5]=1
        h_test[h_test<0.5]=0
        count=0
        for l in range(len(y_test)):
            if h_test[l]==y_test[l]:
                count +=1
        print("AdaDelta 的准确率为",count/m_test)
    def Adam(W,γ,η,β,ε):
        m=0
        G=0
        for i in range(1, epoch+1):
            j=rand[i-1]
            h=logit(np.dot(X_train[j:j+99],W.transpose()))

g=np.dot(X_train[j:j+99].transpose(),(h-y_train[j:j+99]))/100
```

```
            m=β*m+(1-β)*g
            G=γ*G+(1-γ)*np.dot(g.transpose(),g)
            α=η*(np.sqrt(1-γ)/(1-β))
            W=W-α*m/np.sqrt(G+ε)
            h_test = logit(np.dot(X_test,W))
            J_test = 
-(1/m_test)*np.sum(y_test*np.log(h_test)+(1-y_test)*np.log(1-h_test))
            L_Adam.append(J_test)
        h_test[h_test>0.5]=1
        h_test[h_test<0.5]=0
        count=0
        for l in range(len(y_test)):
            if h_test[l]==y_test[l]:
                count +=1
        print("Adam 的准确率为",count/m_test)
    W=w.transpose()
    NAG(W,0.9,0.01)
    W=w.transpose()
    RMSProp(W,0.9,0.01,1e-5)
    W=w.transpose()
    AdaDelta(W,0.9,1e-5)
    W=w.transpose()
```

Adam(W,0.9,0.01,0.9,1e-5)

Linear Classification:

```
epoch =300
Iteration=range(1,epoch+1)
L_NAG=[]
L_RMSProp=[]
L_AdaDelta=[]
L_Adam=[]
C=0.9
rand=[]
for i in range(1, epoch+1):
    rand.append(random.randint(0,m_train-100))
def f(x,y,W,i):
    return 1-np.dot(y[i],np.dot(x[i],W))
def NAG(W, γ , η ):
    v=0
    for i in range(1, epoch+1):
        g=0
        j=rand[i-1]
        for l in range(100):
            if f(X_train,y_train,(W- γ *v),j+l)>=0:
```

```
                    g
-=C*np.dot(X_train[j+l].transpose(),y_train[j+l])
                g /=100
                g +=W- γ *v
                v= γ *v+ η *g
                W=W-v
                loss_test=0
                for k in range(m_test):
                    loss_test += C*max(0,f(X_test,y_test,W,k))
                loss_test /=m_test
                loss_test += np.dot(W.transpose(),W)/2
                L_NAG.append(loss_test)
            y_predict=np.dot(X_test,W)
            y_predict[y_predict>0]=1
            y_predict[y_predict<0]=-1
            count=0
            for m in range(len(y_test)):
                if y_predict[m]==y_test[m]:
                    count +=1
            print("NAG 的准确率为",count/m_test)
        def RMSProp(W, γ , η , ε ):
            G=0
```

```
        for i in range(1, epoch+1):
            g=0
            j=rand[i-1]
            for l in range(100):
                if f(X_train,y_train,W,j+l)>=0:
                    g
-=C*np.dot(X_train[j+l].transpose(),y_train[j+l])
            g /=100
            g +=W
            G= γ *G+(1- γ )*np.dot(g.transpose(),g)
            W=W-( η /np.sqrt(G+ ε ))*g
            loss_test=0
            for k in range(m_test):
                loss_test += C*max(0,f(X_test,y_test,W,k))
            loss_test /=m_test
            loss_test += np.dot(W.transpose(),W)/2
            L_RMSProp.append(loss_test)
        y_predict=np.dot(X_test,W)
        y_predict[y_predict>0]=1
        y_predict[y_predict<0]=-1
        count=0
        for m in range(len(y_test)):
```

```
            if y_predict[m]==y_test[m]:
                count +=1
        print("RMSProp 的准确率为",count/m_test)
    def AdaDelta(W,γ,ε):
        G=0
        Δ=0
        for i in range(1, epoch+1):
            g=0
            j=rand[i-1]
            for l in range(100):
                if f(X_train,y_train,W,j+l)>=0:
                    g
-=C*np.dot(X_train[j+l].transpose(),y_train[j+l])
            g /=100
            g +=W
            G=γ*G+(1-γ)*np.dot(g.transpose(),g)
            ΔW=-(np.sqrt(Δ+ε)/np.sqrt(G+ε))*g
            W=W+ΔW
            Δ=γ*Δ+(1-γ)*np.dot(ΔW.transpose(),ΔW)
            loss_test=0
            for k in range(m_test):
                loss_test += C*max(0,f(X_test,y_test,W,k))
```

```
            loss_test /=m_test
            loss_test += np.dot(W.transpose(),W)/2
            L_AdaDelta.append(loss_test)
        y_predict=np.dot(X_test,W)
        y_predict[y_predict>0]=1
        y_predict[y_predict<0]=-1
        count=0
        for m in range(len(y_test)):
            if y_predict[m]==y_test[m]:
                count +=1
        print("AdaDelta 的准确率为",count/m_test)
    def Adam(W, γ , η , β , ε ):
        m=0
        G=0
        for i in range(1, epoch+1):
            g=0
            j=rand[i-1]
            for l in range(100):
                if f(X_train,y_train,W,j+l)>=0:
                    g
-=C*np.dot(X_train[j+l].transpose(),y_train[j+l])
            g /=100
```

```
        g +=W
        m=β*m+(1-β)*g
        G=γ*G+(1-γ)*np.dot(g.transpose(),g)
        α=η*(np.sqrt(1-γ)/(1-β))
        W=W-α*m/np.sqrt(G+ε)
        loss_test=0
        for k in range(m_test):
            loss_test += C*max(0,f(X_test,y_test,W,k))
        loss_test /=m_test
        loss_test += np.dot(W.transpose(),W)/2
        L_Adam.append(loss_test)
    y_predict=np.dot(X_test,W)
    y_predict[y_predict>0]=1
    y_predict[y_predict<0]=-1
    count=0
    for m in range(len(y_test)):
        if y_predict[m]==y_test[m]:
            count +=1
    print("Adam 的准确率为",count/m_test)
W=w.transpose()
NAG(W,0.9,0.001)
W=w.transpose()
```

RMSProp(W,0.9,0.003,1e-8)

W=w.transpose()

AdaDelta(W,0.9,1e-6)

W=w.transpose()

Adam(W,0.9,0.001,0.9,1e-8)

8. The initialization method of model parameters:

Logistic Regression:all are zero

Linear Classification:all are zero

9.The selected loss function and its derivatives:

Logistic Regression:

J(w)=1/n*Sum 1_n log(1+e^ (yi*wT*xi))+C/2*||w||^2

Linear Classification:

```
J=||w||^2/2+C*Sum 1_m max(0,1-yi(wTxi +b))
∂ J(w)/∂ w=w-C*yxi      if 1-yi(wTxi+b)>=0
∂ J(w)/∂ w=w            if 1-yi(wTxi+b)<0
```

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

Logistic Regression:

Hyper-parameter selection:epoch =2000

Predicted Results (Best Results):

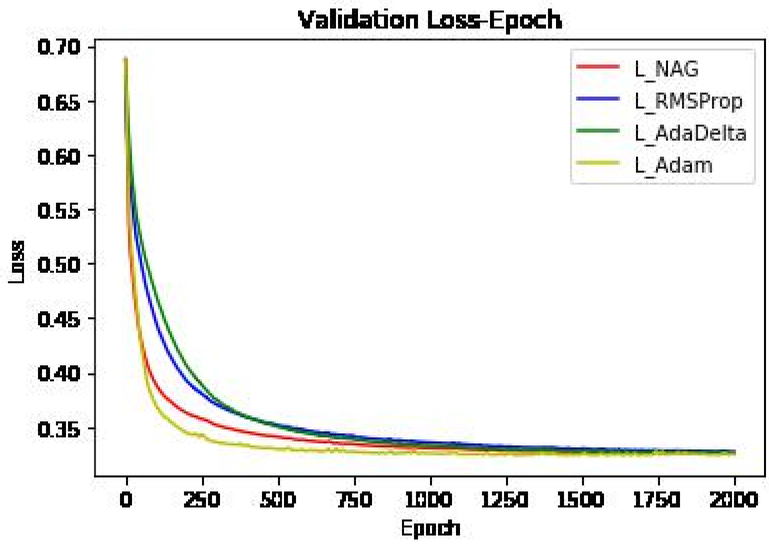
The accuracy of NAG is 0.8500706344819114

The accuracy of RMSProp is 0.8501320557705301

The accuracy of AdaDelta is 0.8503777409250046

The accuracy of Adam is 0.8502548983477674

Loss curve:



Linear Classification:

Hyper-parameter selection:epoch =300

Predicted Results (Best Results):

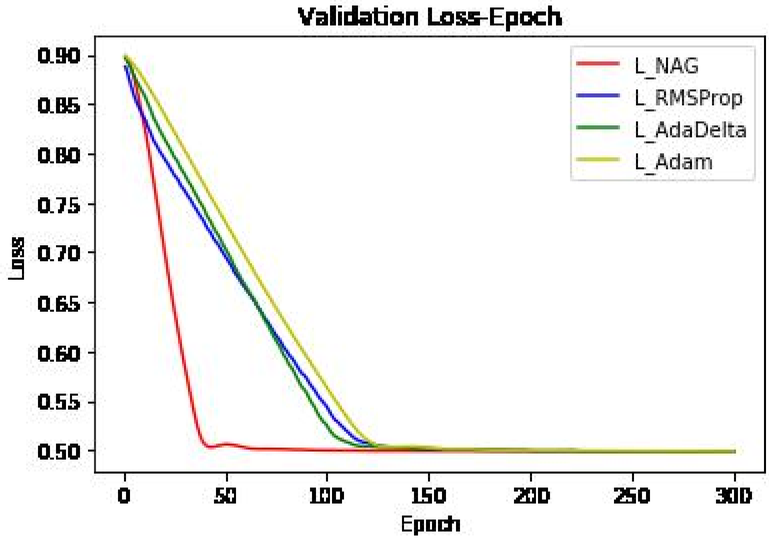
The accuracy of NAG is 0.7637737239727289

The accuracy of RMSProp is 0.7637737239727289

The accuracy of AdaDelta is 0.7637737239727289

The accuracy of Adam is 0.7637737239727289

Loss curve:



11. Results analysis:

Logistic Regression:

L_Adam is the best while L_AdaDelta is the worst.

Linear Classification:

L_NAG is the best while L_Adam is the worst.

12. Similarities and differences between logistic regression and linear classification:

They are both convergent as the epoch increases.But the convergence speed of linear classification is faster than logistic regression.

13. Summary:

I learn about logistic regression and linear classification more deeply.