華南理工大學

South China University of Technology

The Experiment Report of *Machine Learning*

College Software College

Subject Software Engineering

Members 李焕童

Student ID 201530611944

E-mail 1779414343@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:李焕童**

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:**

7. **Code:**

Logistic Regression:

```
epoch =1000
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-8)
W=w.transpose()
AdaDelta(W,0.9,1e-5)
W=w.transpose()
```

```
Adam(W,0.9,0.01,0.9,1e-8)
```

Linear Classification:

```
epoch =500
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,0.9,0.001,1e-8)
W=w.transpose()
AdaDelta(W,0.9,1e-6)
W=w.transpose()
```

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

(Fill in the contents of 8-11 respectively for logistic regression and linear classification)

8. The initialization method of model parameters:

Logistic Regression: all zero initialization

Linear Classification: all zero initialization

9.The selected loss function and its derivatives:

Logistic Regression:

J (w) =-1/n[Sum1_n(yi*log(hw(xi))+(1-yi)*log(1-hw(xi)))]

∂J (w)/∂w=(hw(x)-y)x

Linear Classification:

J (w) =||w||^2/2+(C*Sum 1_m max(0,1−yi(wTxi +b)))/m

∂J(w)/∂w=w−(C*yixi)/m 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 =1000

C=0.9

NAG (γ,η)= (0.9,0.01)

RMSProp (γ,η ,ε) = (0.9,0.01,1e-8)

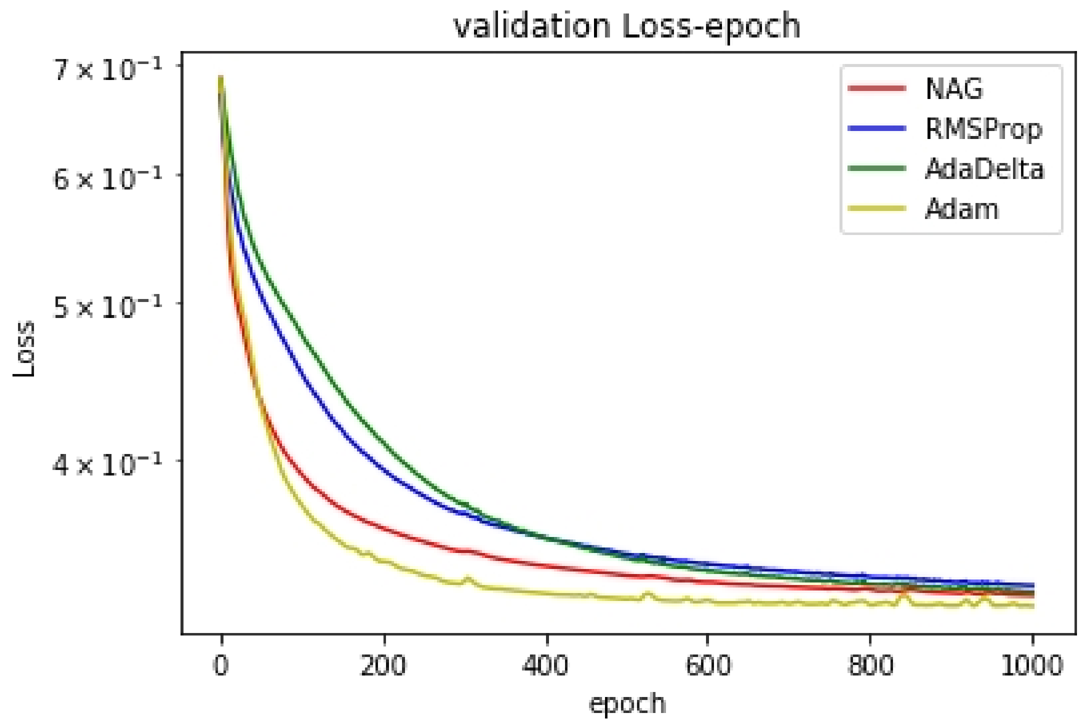
AdaDelta (γ,ε) = (0.9,1e-5)

Adam(γ,η ,β ,ε)= (0.9,0.01,0.9,1e-8)

Predicted Results (Best Results):

γ=0.9, β=0.9,η=0.01, epoch =1000, ε=1e-8,four loss curve can converge smoothly,but the AdaDelta loss curve converge very slowly,after changing ε to 1e-5, the AdaDelta loss curve converge normally.We can see NAG and Adam converge faster than others at first.

Loss curve:



Linear Classification:

Hyper-parameter selection:

epoch =500

C=0.9

NAG (γ ,η)=(0.9,0.001)

RMSProp (γ ,η ,ε) =(0.9,0.001,1e-8)

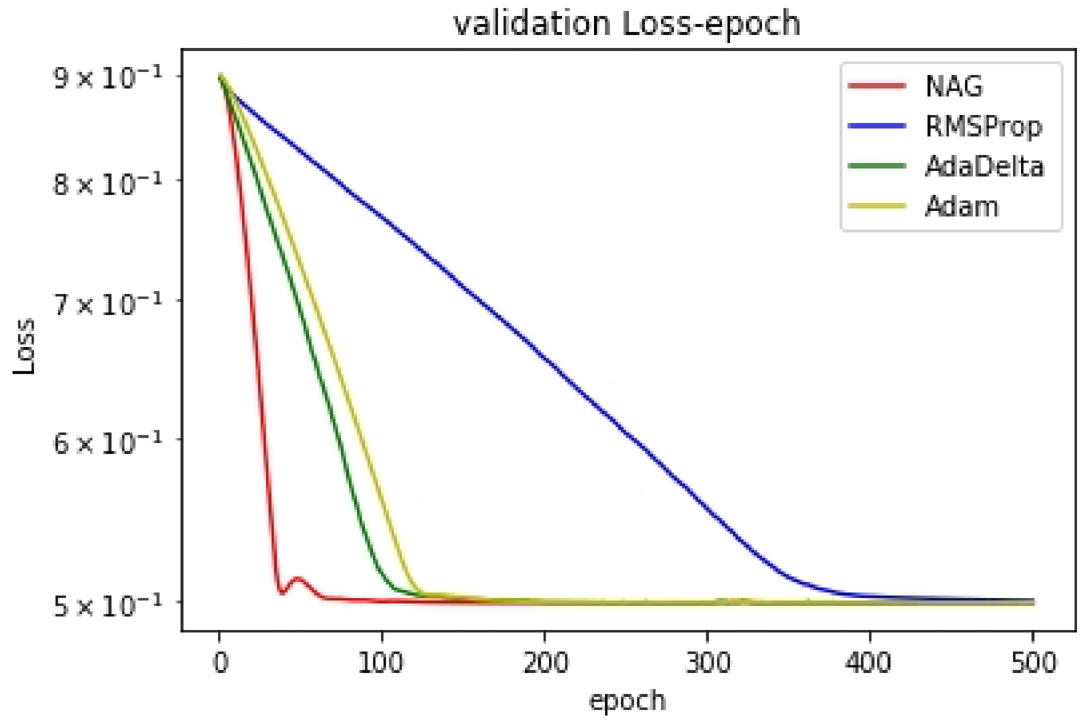
AdaDelta (γ ,ε) =(0.9,1e-6)

Adam(γ ,η ,β ,ε)=(0.9,0.001,0.9,1e-8)

Predicted Results (Best Results):

γ=0.9, β=0.9,η=0.001, epoch =500, ε=1e-8,four loss curve can converge smoothly,but the AdaDelta loss curve converge very slowly,after changing ε to 1e-6, the AdaDelta loss curve converge normally. We can also see NAG and Adam converge faster than others at first.

Loss curve:



11. Results analysis:

Logistic Regression:

Logistic regression with minibatch gradient descent algorithm can solve the classification problem well. Four gradient descent optimization algorithms all performance well. And the accurancy of classification achieves 85.1%.

Linear Classification:

SVM with minibatch gradient descent algorithm also can solve the classification problem. Four gradient descent optimization algorithms all performance well. But the accurancy of classification is relatively low,only 76.4%.

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

Similarities: logistic regression and linear classification are all based on linear function, they are all classifiers.

Differences: logistic regression use logistic function but linear classification doesn’t. The output of logistic regression is a prediction but linear classification’s is not.

linear classification

13.Summary:

Through this experiment, I clearly grasp the principle and formula of logistic regression and SVM, learn to use minibatch gradient descent algorithm, and try a variety of gradient descent optimization algorithm, to understand the relations and differences between different algorithms.