The Experiment Report of
Machine Learning

College Software College

Subject Software Engineering

Members 林杨

Student ID 201530612279

E-mail 291192462@qq.com

Tutor 谭明奎

Date submitted 2017.Dec.12th

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

2. Time: 2017-12-02 2:00-5:00 PM

3. Reporter:林杨

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) features

6. Experimental steps:

Logistic Regression and Stochastic 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 G 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 LNAG，LRMSProp，LAdaDelta and LAdam.
7.Repeate step 4 to 6 for several times, and drawing graph of，， and with the number of iterations.

Linear Classification and Stochastic 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 G 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 LNAG，LRMSProp，LAdaDelta and LAdam.
7.Repeate step 4 to 6 for several times, and drawing graph of，， and with the number of iterations.

7. Code:

Logistic Regression and Stochastic Gradient Descent

3.Select the loss function and calculate its derivation

```
def grad(theta,X,y):
    y=y.reshape(X.shape[0],1)
    h = sigmoid(X*theta)
    error = h - y
    grad = X.T*error
    return grad
def fitRate_loss(theta,X,y):
    y=y.reshape(X.shape[0],1)
    h = sigmoid(X*theta)
    J = -(1/X.shape[0])*(y*np.log(h)+(1-y)*np.log(1-h)).sum()
    h1= sigmoid(X*theta)
    h1[h1>0.5] = 1
    h1[h1<=0.5] = 0
    fit=np.count_nonzero(h1==y)
    fit_rate=fit/X.shape[0]
    return J,fit_rate
```

4.Calculate gradient from partial samples

```
def batchGradient(X,y,theta):
    j = random.randint(0,X.shape[0]-32)
    batchX= X[j:j + 32]
    batchY= y[j:j + 32]
    Gbatch=grad(theta,batchX,batchY)
    return Gbatch
```

Linear Classification and Stochastic Gradient Descent

Common Part（NAG,RMSProp,Adadelta,Adam)

4.Calculate gradient from partial samples

```
def batchGradient(X,y,theta):
    j = random.randint(0,X.shape[0]-256)
    batchX= X[j:j + 256]
    batchY= y[j:j + 256]
    Gbatch=grad(theta,batchX,batchY)
    return Gbatch
```

5.NAG

```
def NAG(theta,alpha=0.01,iters=100):
    v=0
    momentum=0.99
    J_nag=np.zeros(iters)
    J_nagTest=np.zeros(iters)
    F_nagTest=np.zeros(iters)
    F_nag=np.zeros(iters)
    for i in range(iters):
        G=batchGradient(X_train,y_train,theta)
        v_prev = v
        v = momentum*v-alpha*G
        theta=theta+momentum*(v-v_prev)
        J_nag[i],F_nag[i]=fitRate_loss(theta,X_train,y_train)
        J_nagTest[i],F_nagTest[i]=fitRate_loss(theta,X_test,y_test)
    return J_nag,J_nagTest,F_nagTest
```

6.RMSprop

```
def RMSprop(theta,alpha=0.2,iters=100):
    decay_rate=0.9
    J_RMSprop=np.zeros(iters)
    J_RMSpropTest=np.zeros(iters)
    F_RMSprop=np.zeros(iters)
    F_RMSpropTest=np.zeros(iters)
    cache=0
    for i in range(iters):
        G=batchGradient(X_train,y_train,theta)
        cache = decay_rate*cache+np.sum((1-decay_rate)*(G**2))
        theta = theta-alpha*G/(np.sqrt(cache+e))
        J_RMSprop[i],F_RMSprop[i]=fitRate_loss(theta,X_train,y_train)
        J_RMSpropTest[i],F_RMSpropTest[i]=fitRate_loss(theta,X_test,y_test)
    return J_RMSprop,J_RMSpropTest,F_RMSpropTest
```

7.AdaDelta

```
def AdaDelta(theta,alpha=2000,iters=100):
    delta=0.5
    J_AdaDelta=np.zeros(iters)
    J_AdaDeltaTest=np.zeros(iters)
    F_AdaDelta=np.zeros(iters)
    F_AdaDeltaTest=np.zeros(iters)
    cache_x=0
    cache_dx=0
    for i in range(iters):
        G=batchGradient(X_train,y_train,theta)
        cache_dx = delta*cache_dx + np.sum((1-delta)*(G**2))
        v = -np.sqrt(cache_x+1e-7)*G/(np.sqrt(cache_dx)+1e-7)
        theta = theta+alpha*v
        cache_x = delta*cache_x + np.sum((1-delta)*(v**2))
        J_AdaDelta[i],F_AdaDelta[i]=fitRate_loss(theta,X_train,y_train)
        J_AdaDeltaTest[i],F_AdaDeltaTest[i]=fitRate_loss(theta,X_test,y_test)
    return J_AdaDelta,J_AdaDeltaTest,F_AdaDeltaTest
```

8.Adam

```
def Adam(theta, alpha=0.5, iters=100):
    t=0
    m=0
    v=0
    beta1=0.9
    beta2=0.99
    J_Adam=np.zeros(iters)
    J_AdamTest=np.zeros(iters)
    F_Adam=np.zeros(iters)
    F_AdamTest=np.zeros(iters)
    for i in range(iters):
        G=batchGradient(X_train, y_train, theta)
        t=t+1
        m = beta1*m+np.sum((1-beta1)*G)
        v = beta2*v+np.sum((1-beta2)*(G**2))
        mb = m/(1-(beta1**t))
        vb = v/(1-(beta2**t))
        theta= theta-alpha*mb/(np.sqrt(vb)+e)
        J_Adam[i],F_Adam[i]=fitRate_loss(theta,X_train,y_train)
        J_AdamTest[i],F_Adam[i]=fitRate_loss(theta,X_test,y_test)
    return J_Adam,J_AdamTest,F_Adam
```

7. The initialization method of model parameters:

Initialize theta to ones, iterations to 1500 and learning rate to 0.05

8. The selected loss function and its derivatives:

Logistic Regression and Stochastic Gradient Descent

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

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

Linear Classification and Stochastic Gradient Descent

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

$$\frac{\partial l_{hinge}}{\partial w} = \begin{cases} 0 & y_i w \cdot x \geq 1 \\ -y_i x & y_i w \cdot x < 1 \end{cases}$$

10. Experimental results and curve:

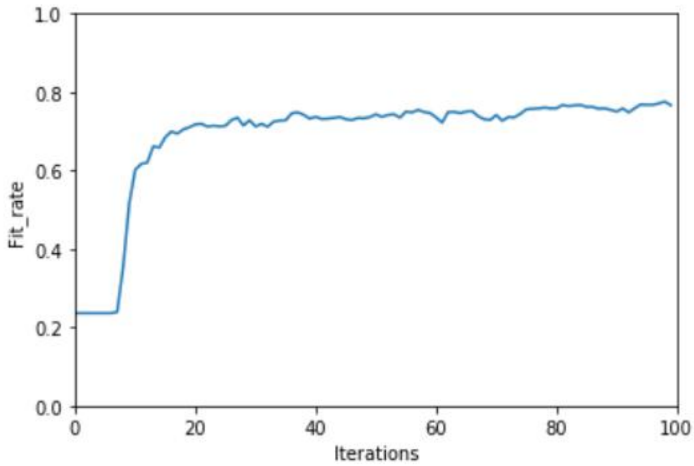
Logistic Regression and Stochastic Gradient Descent

Batch size =32

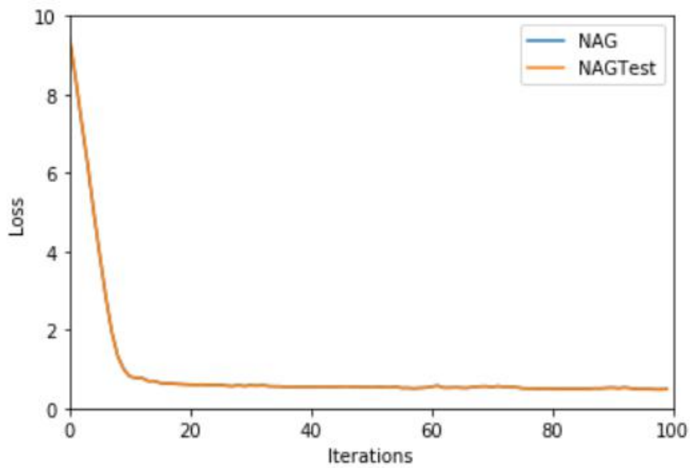
NAG:

Hyper-parameter selection:momentum=0.99,alpha=0.01

Predicted Results (Best Results):



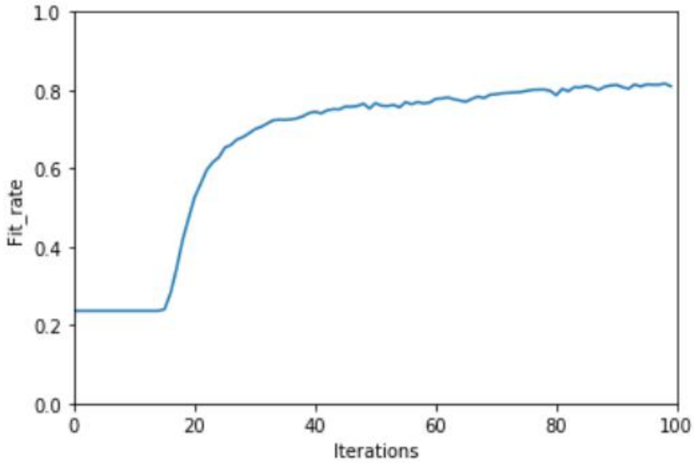
Loss curve:



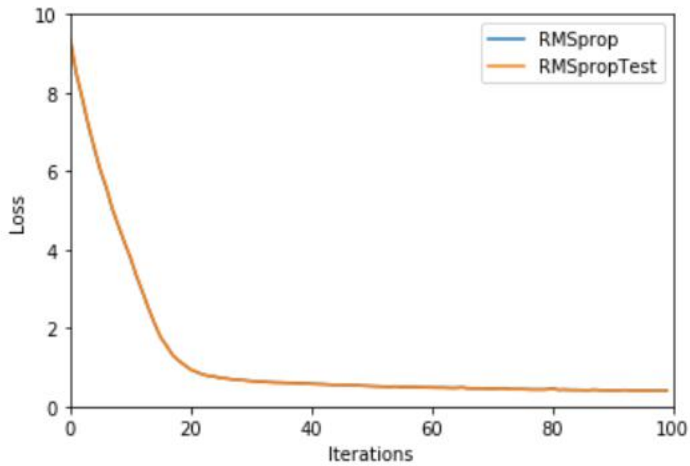
RMSProp:

Hyper-parameter selection:decay_rate=0.9,alpha=0.2

Predicted Results (Best Results):



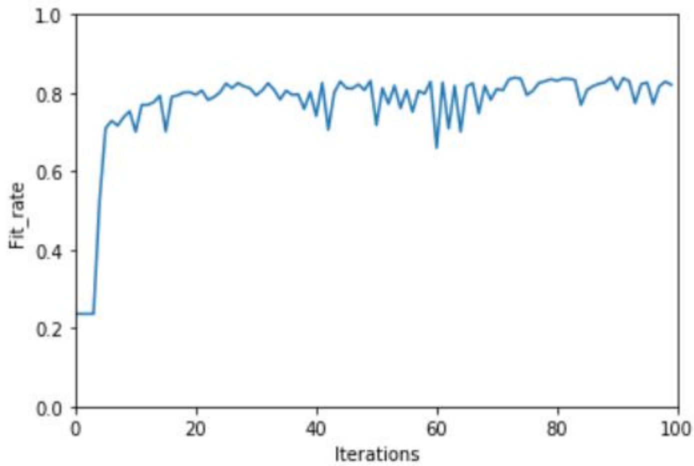
Loss curve:



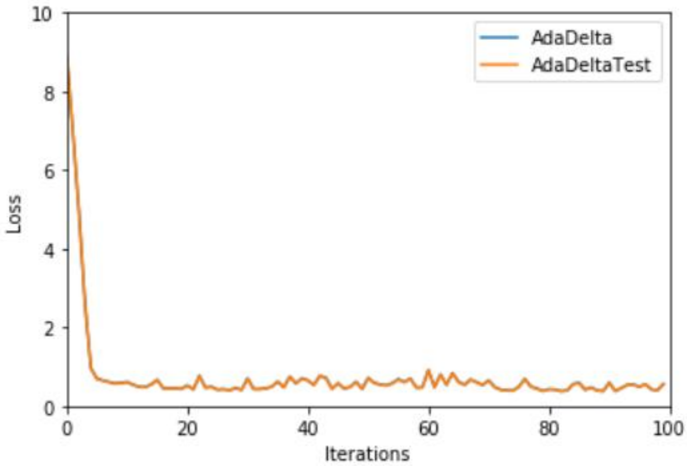
AdaDelta:

Hyper-parameter selection:delta=0.5,alpha=2000

Predicted Results (Best Results):



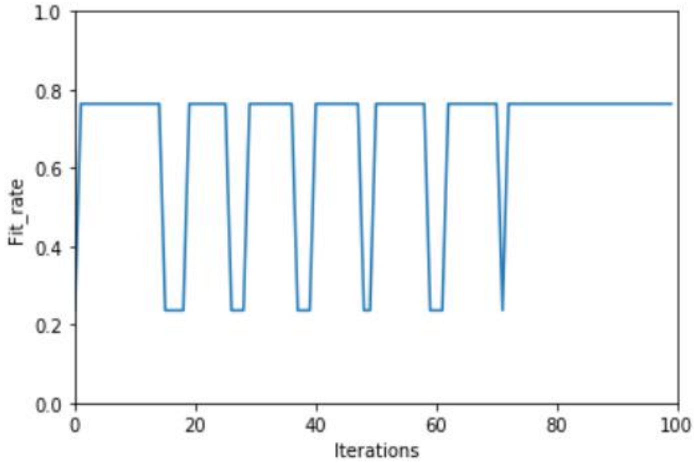
Loss curve:



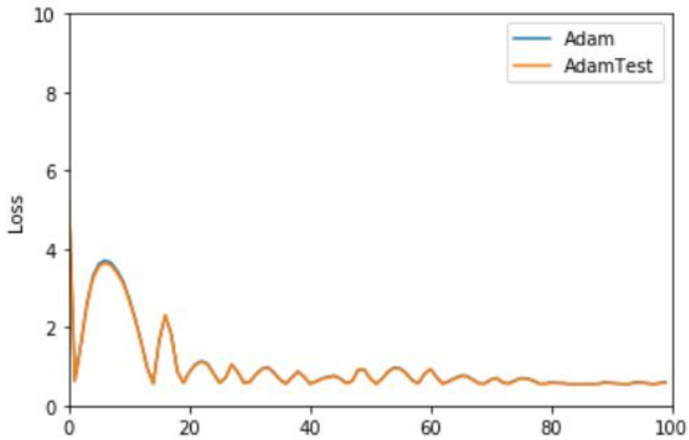
Adam:

Hyper-parameter selection:beta1=0.9 beta2=0.99,alpha=0.1

Predicted Results (Best Results):



Loss curve:



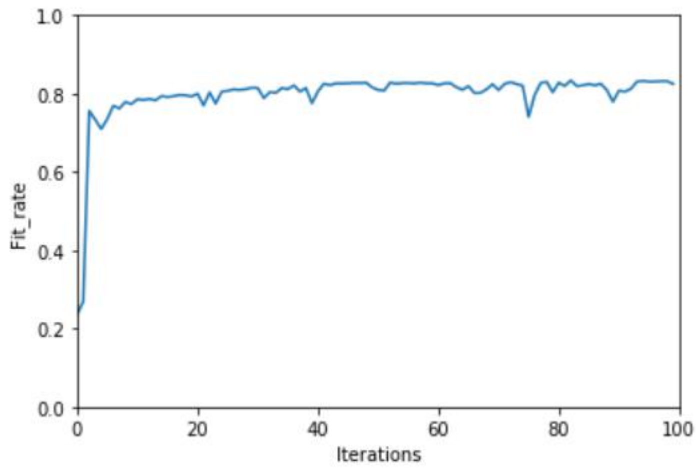
Linear Classification and Stochastic Gradient Descent

Batch size =256

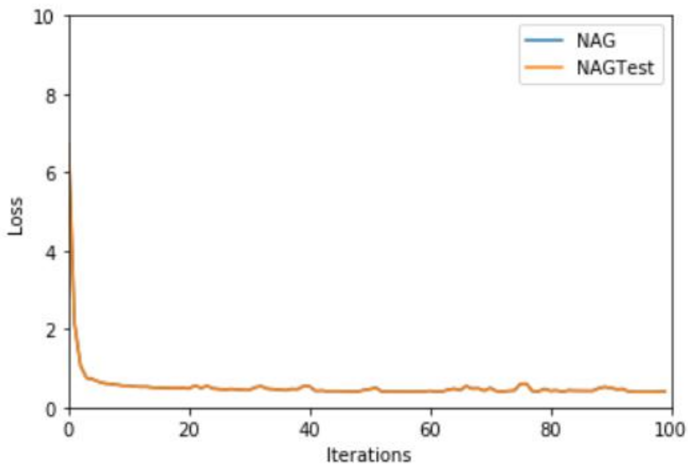
NAG:

Hyper-parameter selection:momentum=0.99,alpha=0.01

Predicted Results (Best Results):



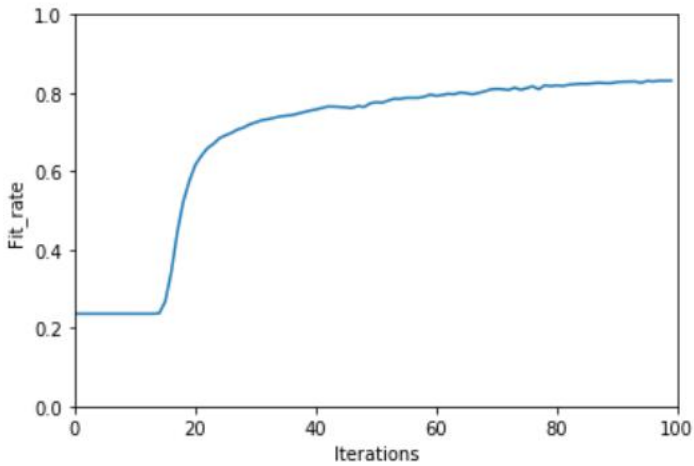
Loss curve:



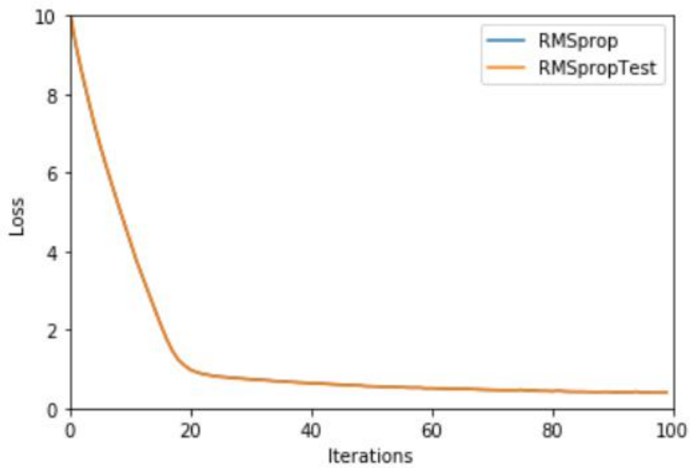
RMSProp:

Hyper-parameter selection:decay_rate=0.9,alpha=0.2

Predicted Results (Best Results):



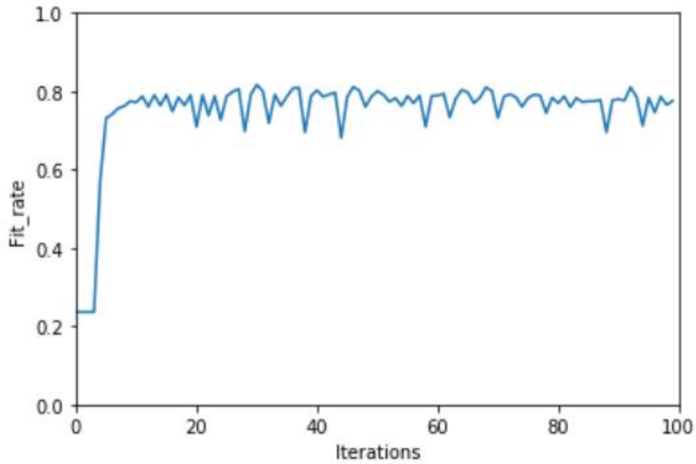
Loss curve:



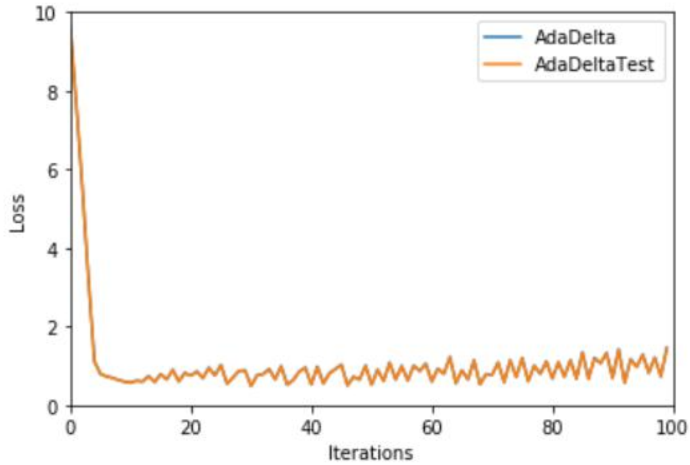
AdaDelta:

Hyper-parameter selection:delta=0.5,alpha=2000

Predicted Results (Best Results):



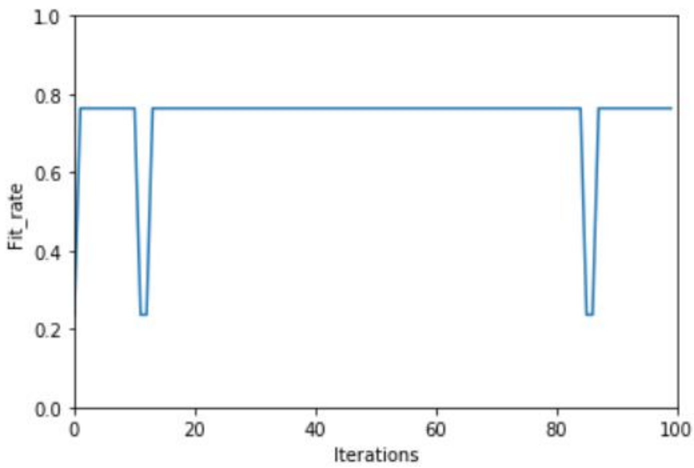
Loss curve:



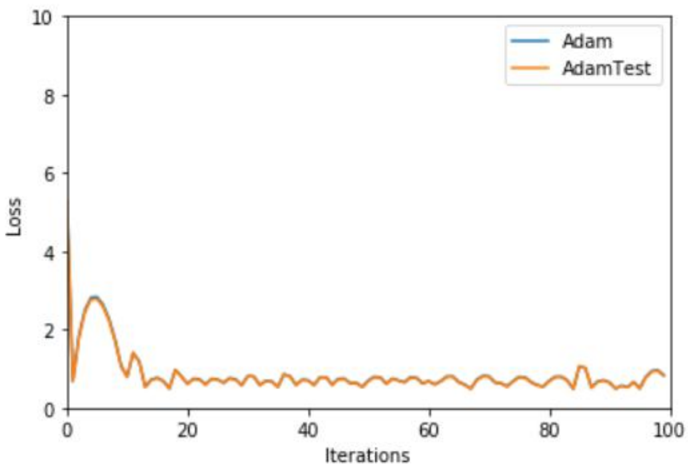
Adam:

Hyper-parameter selection:beta1=0.9 beta2=0.99,,alpha=0.1

Predicted Results (Best Results):

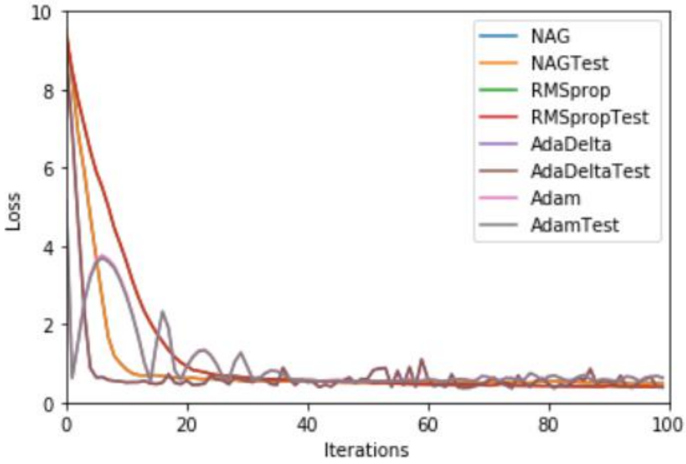


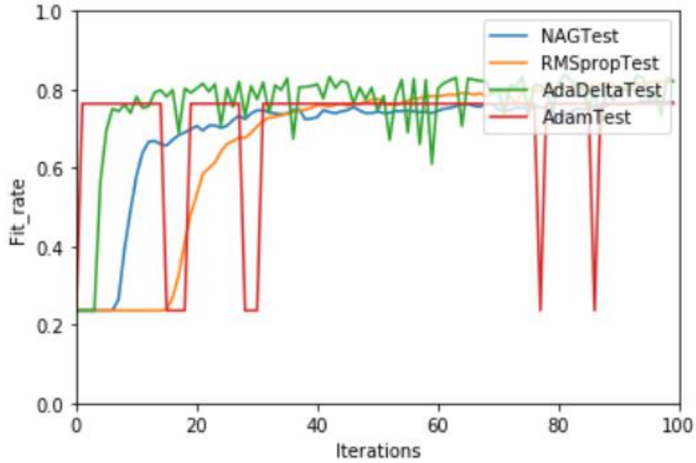
Loss curve:



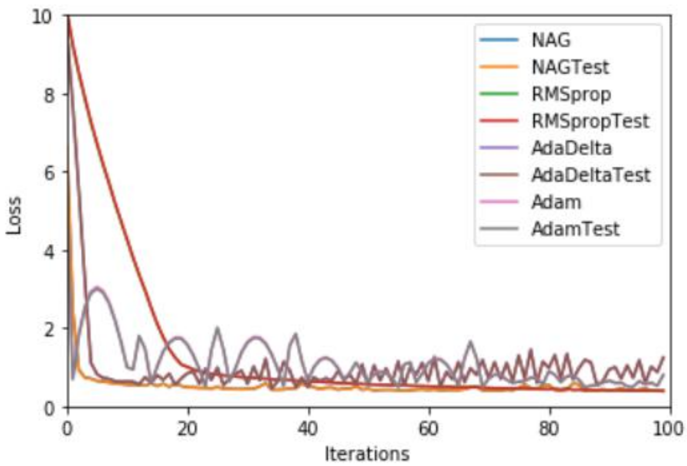
11. Results analysis:

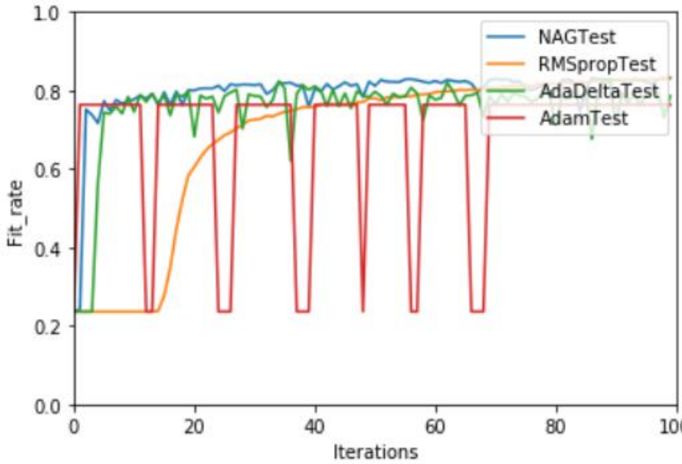
Logistic Regression and Stochastic Gradient Descent





Linear Classification and Stochastic Gradient Descent





The loss of four optimized methods all come to less than 2 and become stable after 40 iterations. Among these optimized methods, RMSProp is the slowest but the smoothest. And though Adam is the fastest that arrives the bottom, it shows the biggest undulation.

The fit rate of four optimized methods will eventually get to about 80%.Adam is the most unstable. RMSprop performs the best among these methods.AdaDelta also shows some undulation.

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

Simmilarities:logistic regression and linear classification both need a threshold to divide datas into multiple classes.

Differences:the loss function of logistic regression is logistic loss and the loss function of linear classification is Hinge loss,which means logistic regression's target is to minimize logistic loss and linear classification's target is to maximize the space between the classification planes and datas.

13. Summary:

In this experiment, I have understood the difference between gradient descent and stochastic gradient descent, that is SGD uses mini batch to reduce calculation and speed up algorithm.I have compared and understood the differences and relationships between Logistic regression and linear classification, and the principles of SVM and practice on larger data with SGD to reduce cost of time .