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: Xianlin Hu

Supervisor:
Mingkui Tan

Student ID：201530781104

Grade: Undergraduate

December 9, 2017

Comparison of Various Stochastic Gradient Descent Methods for Solving Classification Problems

Abstract—Gradient descent optimization algorithms, while increasingly popular, are often used as black-box optimizers, as practical explanations of their strengths and weaknesses are hard to come by. In this experiment we will implement several classical gradient descent optimization algorithms in logistic regression and svm, and study the characteristics and merits of these algorithms by comparing the loss function with the accuracy.

I. Introduction

Gradient descent optimization algorithms, while increasingly popular, are often used as black-box optimizers, as practical explanations of their strengths and weaknesses are hard to come by. In this experiment we will implement several classical gradient descent optimization algorithms in logistic regression and svm, and study the characteristics and merits of these algorithms by comparing the loss function with the accuracy.

Purpose of the experiment:

1.Compare and understand the differences and relationships between the gradient descent and the 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.

II. Methods and Theory

SGD

In the event of a large amount of data, it is difficult not to use the Stochastic gradient descent (SGD). SGD is very intuitive, is to take a random or a few data to do a gradient decline, that is

$$\begin{aligned} \mathbf{g}_t &\leftarrow \nabla J_i(\boldsymbol{\theta}_{t-1}) \\ \boldsymbol{\theta}_t &\leftarrow \boldsymbol{\theta}_{t-1} - \eta \mathbf{g}_t \end{aligned} \tag{1}$$

This gradient \ Vg_t is calculated for part of the data, denoted \ Vg_t \ gets \ nabla J (\ Vtheta_ {t-1}) below. \ Vg_t can be regarded as an estimate of the true gradient, which is expected to be at least biased. Therefore, it can converge to the optimal solution when applied to convex problems. However, there are a lot of problems that SGD needs to solve:

Convergence rate with learning rate \ eta a lot, big \ eta easy to shock, small \ eta convergence is very slow. Artificially adjusted in the training is more difficult, it is difficult to adapt to the characteristics of the data.

The learning rate \ eta is the same for all \ Vtheta features. In fact, some of the features in \ Vtheta should be slower, faster, and sparse.

Easy to fall into the local minimum or saddle point. Especially when training neural networks, will be more obvious.

NAG

The core idea of NAG (Nesterov accelerated gradient) is to use Momentum to predict the next gradient, rather than using the current \ Vtheta.

$$\begin{aligned} \mathbf{g}_t &\leftarrow \nabla J(\boldsymbol{\theta}_{t-1} - \gamma \mathbf{v}_{t-1}) \\ \mathbf{v}_t &\leftarrow \gamma \mathbf{v}_{t-1} + \eta \mathbf{g}_t \\ \boldsymbol{\theta}_t &\leftarrow \boldsymbol{\theta}_{t-1} - \mathbf{v}_t \end{aligned} \tag{3}$$

RMSProp

$$\begin{aligned} \mathbf{g}_t &\leftarrow \nabla J(\boldsymbol{\theta}_{t-1}) \\ G_t &\leftarrow \gamma G_t + (1-\gamma)\mathbf{g}_t \odot \mathbf{g}_t \\ \boldsymbol{\theta}_t &\leftarrow \boldsymbol{\theta}_{t-1} - \frac{\eta}{\sqrt{G_t + \epsilon}} \odot \mathbf{g}_t \end{aligned} \tag{5}$$

AdaDelta:

$$\begin{aligned} \mathbf{g}_t &\leftarrow \nabla J(\boldsymbol{\theta}_{t-1}) \\ G_t &\leftarrow \gamma G_t + (1-\gamma)\mathbf{g}_t \odot \mathbf{g}_t \\ \Delta\boldsymbol{\theta}_t &\leftarrow -\frac{\sqrt{\Delta_{t-1} + \epsilon}}{\sqrt{G_t + \epsilon}} \odot \mathbf{g}_t \\ \boldsymbol{\theta}_t &\leftarrow \boldsymbol{\theta}_{t-1} + \Delta\boldsymbol{\theta}_t \\ \Delta_t &\leftarrow \gamma\Delta_{t-1} + (1-\gamma)\Delta\boldsymbol{\theta}_t \odot \Delta\boldsymbol{\theta}_t \end{aligned} \tag{6}$$

In contrast to (5), AdaDelta can be used to estimate the learning rate using \ sqrt {\ Delta_ {t-1} + \ epsilon}. Here \ gamma can take a 0.95. Intuitively, it is plausible to use the previous steps \ Delta \ Vtheta_t to estimate the next step. "More plausibly, SGD, Momentum, or AdaGrad updates units incorrectly, or we give \ eta a unit. Looking at equation (1), \ Vg_t has the unit \ frac {1} {\ Vtheta} Unit (assuming that J has no unit, dubious), updating it with \ Vtheta may be wrong, but AdaDelta does not have this problem. \ Frac {\ Delta \ Vtheta} {\ frac {\ Delta J} {\ Delta \ Vtheta}} \ Vg_t \ propto \ Delta \ Vtheta.

Adam:

$$\mathbf{g}_t \leftarrow \nabla J(\boldsymbol{\theta}_{t-1})$$
$$\mathbf{m}_t \leftarrow \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\mathbf{g}_t$$
$$G_t \leftarrow \gamma G_t + (1-\gamma)\mathbf{g}_t \odot \mathbf{g}_t$$
$$\alpha \leftarrow \eta \frac{\sqrt{1-\gamma^t}}{1-\beta^t}$$
$$\boldsymbol{\theta}_t \leftarrow \boldsymbol{\theta}_{t-1} - \alpha \frac{\mathbf{m}_t}{\sqrt{G_t + \epsilon}}$$

III. EXPERIMENT

1. Purpose:

Compare and understand the differences and relationships between the gradient descent and the stochastic gradient descent.

Compare and understand the differences and relationships between logistic regression and linear classification.

Further understand the principles of SVM and practice on larger data.

2 data set:The experiment uses a9a data in LIBSVM Data, which contains 32561/16281 testing and each sample has 123/123 testing properties.

3. 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 G 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 L_NAG, L_RMSProp, L_AdaDelta and L_Adam, for different optimization methods.

7)Repeat step 4-6 for several times and plot the L_NAG, L_RMSProp, L_AdaDelta and L_Adam by iterations.

Linear classification and stochastic gradient descent:

1)Read experimental training set and verification set.

2)Support vector machine 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 G 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 L_NAG, L_RMSProp, L_AdaDelta and L_Adam, for different optimization methods.

7)Repeat step 4-6 for several times and plot the L_NAG, L_RMSProp, L_AdaDelta and L_Adam by iterations.

4.code:

```
def NAG( X , y , X_t , y_t , r , lam , n , epoch, mbs ,threshold):
    w = np.zeros(123).reshape(123,1)
    v = np.zeros(123).reshape(123,1)
    L_NAG = []
    acc=[]
    for i in range(epoch):
        X_rand,y_rand = mbs_data(X,y)
        for j in range(X.shape[0]//mbs-2):
            X_train = X_rand[ mbs * ( j % mbs ) : mbs * ( j % mbs + 1 ) ]
            y_train = y_rand[ mbs * ( j % mbs ) : mbs * ( j % mbs + 1 ) ]
            g = gradient(w-r*v,X_train,y_train ,lam)
            v = r * v + n * g
            w = w - v
            if(j%50==1):
                L_NAG.append(loss(w,X_t,y_t,lam))
                predict=0
                for j in range(len(y_t)):
                    if h_func(X[j],w)>=threshold:
                        predict+=1
                acc.append(predict/len(y_t))
    return acc,L_NAG

def RMSProp( X , y , X_t , y_t ,e, r , lam , n , epoch, mbs ,threshold):
    w = np.zeros(123)
    G=0
    L_RMSProp = []
    acc=[]
    for i in range(epoch):
        X_rand,y_rand = mbs_data(X,y)
        for j in range(X.shape[0]//mbs):
            X_train = X_rand[ mbs * ( j % mbs ) : mbs * ( j % mbs + 1 ) - 1 ]
            y_train = y_rand[ mbs * ( j % mbs ) : mbs * ( j % mbs + 1 ) - 1 ]
            g = gradient(w,X_train,y_train ,lam)
            G = r * G + (1-r)*g*g
            w = w - n/(np.sqrt(G+e))*g
            if(j%50==1):
                L_RMSProp.append(loss_func(w,X_t,y_t,lam))
                predict=0
                for j in range(len(y_t)):
                    if h_func(X[j],w)>=threshold:
                        predict+=1
                acc.append(predict/len(y_t))
    return acc,L_RMSProp

def AdaDelta( X , y , X_t , y_t ,e, r , lam , n , epoch, mbs ,threshold):
    w = np.zeros(123)
    G=0
    L_AdaDelta = []
```

```
    acc=[]
    delta=0
    for i in range(epoch):
        X_rand,y_rand = mbs_data(X,y)
        for j in range(X.shape[0]//mbs):
            X_train = X_rand[ mbs * ( j % mbs ) : mbs * ( j % mbs + 1 ) - 1 ]
            y_train = y_rand[ mbs * ( j % mbs ) : mbs * ( j % mbs + 1 ) - 1 ]
            g = gradient(w,X_train,y_train ,lam)
            G = r * G + (1-r)*g*g
            delta_w = - np.sqrt( delta + e ) / np.sqrt( G + e ) * g
            w = w + delta_w
            delta = r * delta + ( 1 - r) * delta_w * delta_w
            if(j%50==1):
                L_AdaDelta.append(loss_func(w,X_t,y_t,lam))
                predict=0
                for j in range(len(y_t)):
                    if h_func(X[j],w)>=threshold:
                        predict+=1
                acc.append(predict/len(y_t))
    return acc,L_AdaDelta

def Adam( X , y , X_t , y_t ,e, r , lam , n , epoch, mbs ,threshold,b):
    w = np.zeros(123)
    G=0
    L_Adam = []
    acc=[]
    m=0
    for i in range(epoch):
        X_rand,y_rand = mbs_data(X,y)
        for j in range(X.shape[0]//mbs):
            X_train = X_rand[ mbs * ( j % mbs ) : mbs * ( j % mbs + 1 ) - 1 ]
            y_train = y_rand[ mbs * ( j % mbs ) : mbs * ( j % mbs + 1 ) - 1 ]
            g = gradient(w,X_train,y_train ,lam)
            m = b * m + (1 - b) * g
            G = r * G + (1 - r) * g * g
            a = n * np.sqrt(1 - np.power(r,i))/(1 - np.power(b , i))
            w = w - a * m / np.sqrt( G + e )
            if(j%50==1):
                L_Adam.append(loss_func(w,X_t,y_t,lam))
                predict=0
                for j in range(len(y_t)):
                    if h_func(X[j],w)>=threshold:
                        predict+=1
                acc.append(predict/len(y_t))
    return acc,L_Adam
```

5.Selection of validation: hold out

6. The initialization method of model parameters:
Set all parameter into zero

7. The selected loss function and its derivatives
Logistic Regression and Stochastic Gradient Decrease:

Loss function:

$$J(\mathbf{w}) = \frac{1}{n}\sum_{i=1}^{n}\log(1+e^{-y_i\cdot\mathbf{w}^\top\mathbf{x}_i})+\frac{\lambda}{2}||\mathbf{w}||_2^2$$

Gradient:

$$\mathbf{w}' \to \mathbf{w} - \eta\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}} = (1-\eta\lambda)\mathbf{w} + \eta\frac{1}{n}\sum_{i=1}^{n}\frac{y_i\mathbf{x}_i}{1+e^{y_i\cdot\mathbf{w}^\top\mathbf{x}_i}}$$

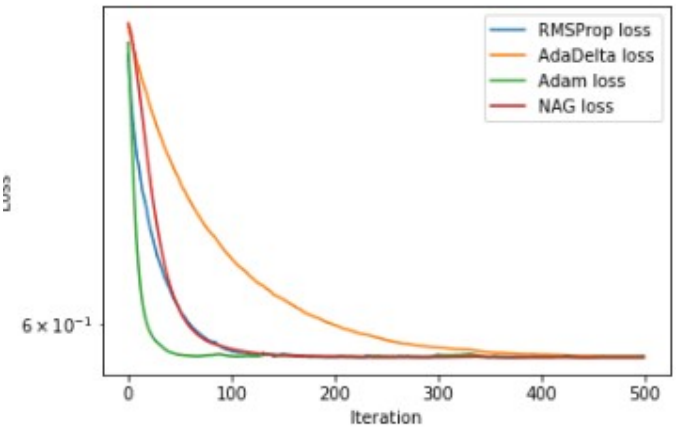
Linear classification and stochastic gradient descent:
Hinge loss:

$$\text{Hinge loss} = \xi_i = \max(0, 1-y_i(\mathbf{w}^\top\mathbf{x}_i+b))$$

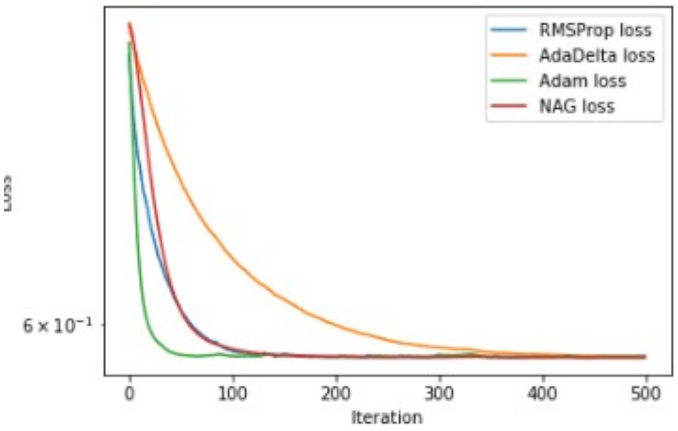
Loss function:

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

8. Experimental results and curve:
Logistic Regression and Stochastic Gradient Decrease:
Epoch=2000
Study rite=0.01
Threshold=0.5
η =0.01
Assessment Results (based on selected validation): Loss curve converges
Predicted Results (Best Results): Loss curve converges below 0.3, accuracy rate=0.86
Loss and accurancy curve



Linear classification and stochastic gradient descent:
Epoch=2000
Study rite=0.01
Threshold=0.5
η =0.01
Assessment Results (based on selected validation): Loss curve converges
Predicted Results (Best Results): Loss curve converges below 0.3, accuracy rate=0.86
Loss and accurancy curve



9.Results analysis:

Logistic Regression and Stochastic Gradient Decrease:

By 2000 iterations, the loss function got stable. The 4 types of loss both dropped below 0.25, And the accuracy is stable at 0.86, which is a reasonable value.

Linear classification and stochastic gradient descent:

By 500 iterations, the loss function got stable. The 4 types of loss both dropped below 0.25. And the accuracy is stable at 0.86, which is a reasonable value.

IV. CONCLUSION

Through this experiment, I understand the difference and connection between gradient descent and stochastic gradient descent, logistic regression and linear classification, and further understand the principle of SVM and practice on larger data. Learn a variety of stochastic gradient descent optimization algorithm, understand the characteristics of different algorithms. For different problems, we should use different algorithms to deal with, but also require a lot of experience-based debugging.

Do not think the most common SGD will not work, or will it be used by many people, because no one knows what happens during the training of complex models, and SGD is the best guarantee of convergence.