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:
周宇琛

Supervisor:
Mingkui Tan

Student ID：201530613887

Grade:
Undergraduate

December 15, 2017

Comparison of Various Stochastic Gradient Descent Methods for Solving Classification Problems

Abstract—
In general, we solving the classification problem is to look for predictive functions, construct loss functions, and minimize loss functions. In minimizing loss function, we usually use the method of gradient descent. In the lab of this classification problem, we will compare the two classification methods using four different gradient descent optimization methods. According to the lab results, we will get the optimal gradient descent method.

I. INTRODUCTION

We need to solve classification problems, and we decided to use logistic regression and linear classification to finish this task. But as we know, if the data set is very big, the gradient descent will be very slowly. So we select four methods that are NAG, RMSProp, AdaDelta and Adam to optimal gradient descent.

II. METHODS AND THEORY

Logistic regression

Loss function:

$$J(w) = -\frac{1}{n}\left[\sum_{i=1}^{n}(y_i log(h_w(x_i)) + (1-y_i) log(1-h_w(x_i)))\right]$$

Derivative:

$$\frac{\partial J(w)}{\partial w} = (h_w(x) - y)x$$

Logistic regression is a widely used classification machine learning algorithm that fits data to a sigmoid function to predict the probability of an event occurring. Logistic regression is nonlinear. $g(z) = \frac{1}{1+e^{-z}}$. But $z = w^T x$ is linear and its algorithm is linear, so we can use $h_w(x) = g(w^T x)$ to solve linear classification problem.

Linear classification

Loss function:

$$J(w) = \min_{w,b}\left(\frac{||w||^2}{2} + C\sum_{i=1}^{N} max(0, 1 - y_i(w^T x_i + b))\right)$$

Derivative:

If $1 - y_i(w^T x_i + b)) > 0$:

$$\frac{\partial J(w)}{\partial w} = w - Cx_i y_i$$

Else:

$$\frac{\partial J(w)}{\partial w} = w$$

In linear classification, we select the svm to solve the linear classification problem. Svm exist a classification plane, the minimum distance between two points set to this plane is the largest, and the distance between the two points is the largest. So we can use svm to solve classification problem.

Four gradient descent methods

Gradient descent is optimizing the loss function. There are four gradient descent methods. First there is a theory about momentum. In the gradient descent, momentum is adding. We can use the momentum to reduce shocks. The NAG gradient descent methods uses the momentum to predict the next gradient. Then we can get result more accurate. Before RMSProp, we introduced the AdaGrad gradient descent methods. The AdaGrad gradient descent methods is a adaptive gradient methods. It uses the gradient obtained before to determine if the corresponding feature is updated. Then it realize learning rate adaptive and easier to converge. The RMSProp is based on AdaGrad. And it is to solve the learing rate approaches 0. The AdaDelta gradient descent methods is likely with RMSProp, but it doesn't need set learing rate. It uses the previous steps to estimate the learning rate. The Adam gradient descent methods uses the advantage of AdaGrad and RMSProp in sparse data. It can adaptive, and correct the deviation of the initialized parameters.

III. EXPERIMENT

Environment:
we did experiment to test which gradient descent is better. The operating environment is python3.6.1

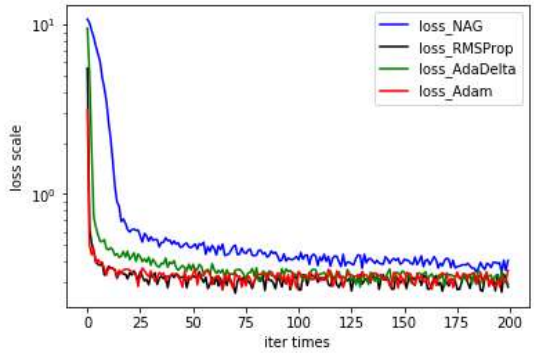
Steps1:
The first, we use logistic regression to realize. The specific steps are as follows.
Logistic Regression and Stochastic Gradient Descent:
(1) Load the training set and validation set.
(2) Initalize logistic regression model parameters, you can consider initalizing zeros, random numbers or normal distribution.
(3) Select the loss function and calculate its derivation.
(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 L_NAG，L_RMSProp，L_AdaDelta and L_Adam.
(7) Repeate step 4 to 6 for several times, and drawing graph of L_NAG，L_RMSProp，L_AdaDelta and L_Adam with the number of iterations.

Realize:
All parameter initialization to 0 or 1
Learning rate:0.02
There is some important code:

```
for j in range(500):
    #NAG优化方法
    if(ways == 1):
        gradient = np.dot(1.0/(1+np.exp(-np.dot(weight.T - 0.9*Vt_NAG,x[k + j])))-y[k + j],x[k + j])/500
        Vt_NAG = 0.9*Vt_NAG + rating * gradient
        weight =  weight - Vt_NAG
    #RMSProp优化方法
    if(ways == 2):
        gradient = np.dot(1.0/(1+np.exp(-np.dot(weight.T,x[k + j])))-y[k + j],x[k + j])/500
        Gt_RMSProp = 0.9*Gt_RMSProp + 0.1*np.dot(gradient,gradient)
        weight =  weight - (rating/np.sqrt(Gt_RMSProp + epsilon)) * gradient
    #AdaDelta
    if(ways == 3):
        gradient = np.dot(1.0/(1+np.exp(-np.dot(weight.T,x[k + j])))-y[k + j],x[k + j])/500
        Gt_AdaDelta = 0.95*Gt_AdaDelta + 0.05*np.dot(gradient,gradient)
        theta_AdaDelta = -np.dot((np.sqrt(delta_AdaDelta + epsilon)/np.sqrt(Gt_AdaDelta + epsilon)),gradient)
        weight =  weight + theta_AdaDelta
        delta_AdaDelta = 0.95*delta_AdaDelta + 0.05*np.dot(theta_AdaDelta,theta_AdaDelta)
    #Adam优化方法
    if(ways == 4):
        gradient = np.dot(1.0/(1+np.exp(-np.dot(weight.T,x[k + j])))-y[k + j],x[k + j])/500
        Mt_Adam = 0.9*Mt_Adam + 0.1*gradient
        Gt_Adam = 0.999*Gt_Adam + 0.001*np.dot(gradient,gradient)
        alpha_Adam = rating*(np.sqrt(1-0.999/(np.sqrt(iter)))/(1-(0.9/np.sqrt(iter))))
        weight =  weight - alpha_Adam * (Mt_Adam/(np.sqrt(Gt_Adam + epsilon)))
        iter = iter + 1
```

We test use the accuracy to find the answer, and the best accuracy is:
NAG:87%
RMSProp:88%
AdaDelta:87%
Adam:90%

After experiment, we **get result(loss function):**



Steps2:

The second, we use linear classification(we use svm to realize linear classification) to realize. The specific steps are as follows.

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.

(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 L_NAG，L_RMSProp, L_AdaDelta and L_Adam.

(7) Repeate step 4 to 6 for several times, and drawing graph of L_NAG，L_RMSProp，L_AdaDelta and L_Adam with the number of iterations.

Realize:

All parameter initialization to 0 or 1
Learning rate:0.03
C(penalty factor):0.9

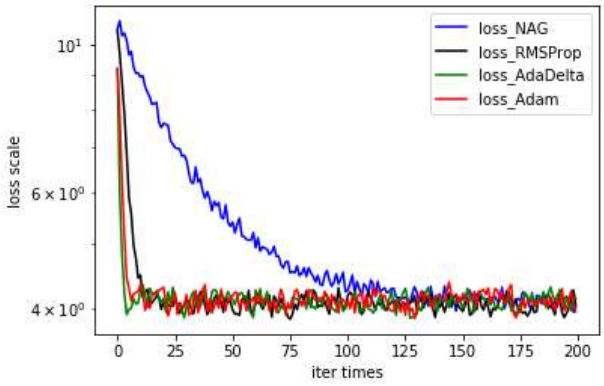
There is some important code:

```
for j in range(500):
    upsilon = 1 - np.dot(y[k + j],np.dot(weight.T,x[k + j].T))
    if(upsilon > 0):
        gradient = 1/C * weight + np.dot(x[k + j],y[k + j])
        #损失函数累加
        jw = jw + upsilon
    if(upsilon <= 0):
        gradient = 1/C * weight

    #NAG优化方法
    if(ways == 1):
        gradient = (gradient - 1/C * 0.9 * Vt_NAG)/500
        Vt_NAG = 0.9*Vt_NAG + rating * gradient
        weight =  weight - Vt_NAG
    #RMSProp优化方法
    if(ways == 2):
        gradient = gradient/500
        Gt_RMSProp = 0.9*Gt_RMSProp + 0.1*np.dot(gradient,gradient)
        weight =  weight - (rating/np.sqrt(Gt_RMSProp + epsilon)) * gradient
    #AdaDelta
    if(ways == 3):
        gradient = gradient/500
        Gt_AdaDelta = 0.95*Gt_AdaDelta + 0.05*np.dot(gradient,gradient)
        theta_AdaDelta = -np.dot((np.sqrt(delta_AdaDelta + epsilon)/np.sqrt(Gt_AdaDelta + epsilon)),gradient)
        weight =  weight + theta_AdaDelta
        delta_AdaDelta = 0.95*delta_AdaDelta + 0.05*np.dot(theta_AdaDelta,theta_AdaDelta)
    #Adam优化方法
    if(ways == 4):
        gradient = gradient/500
        Mt_Adam = 0.9*Mt_Adam + 0.1*gradient
        Gt_Adam = 0.999*Gt_Adam + 0.001*np.dot(gradient,gradient)
        alpha_Adam = rating*(np.sqrt(1-0.999/(np.sqrt(iter)))/(1-(0.9/np.sqrt(iter))))
        weight =  weight - alpha_Adam * (Mt_Adam/(np.sqrt(Gt_Adam + epsilon)))
        iter = iter + 1
```

We test use the accuracy to find the answer, and the best accuracy is:
NAG:77%
RMSProp:78%
AdaDelta:78.4%
Adam:79%

After experiment, we **get result(loss function):**



IV. CONCLUSION

The last, we get the conclusions. Under the same conditions: (1) The gradient of the NAG gradient descent methods dropped slowest. (2) The gradient of the Adam gradient descent methods dropped fastest. (3) when it iters to 125, the loss function would be smooth. (4) The Adam is the best gradient descent method in this conditions.

By this experiment, I can learn many things. Like, if the loss function is concussion very frequently. It may be the learning rate is too big or C (penalty factor) is too big, or your iter times is too small. If the loss function is abnormal. You should see if your formula is incorrect. Then you can set the accuracy to see if the prediction method is incorrect, and so on.

REFERENCERS

[1] https://www.zybuluo.com/chenyaofo/note/961083
[2] https://blog.slinuxer.com/2016/09/sgd-comparison