華南理工大學

South China University of Technology

Comparison of Various *Stochastic Gradient Descent* Methods for Solving *Classification* Problems

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:
Renxin Zhuang

Supervisor:
Mingkui Tan

Student ID:
201530613979

Grade:
Undergraduate

December 15, 2017

Logistic Regression, Linear Classification and Stochastic Gradient Descent

***Abstract*—This is the second experiment of the Machine Learning course. Instead of normal Gradient Descent used in the first experiment, in this experiment I'll use Batch Gradient Descent to solve the optimization problem.**

I. Introduction

THE motivation for this experiment has three folds. First, it helps to compare and understand the difference between gradient descent and stochastic gradient descent. Second, through this experiment we can compare and understand the differences and relationships between Logistic regression and linear classification. Last, it gives us a better way to further understand the principles of SVM and practice on larger data.

II. Methods and Theory

For both optimization problems, we use BGD to address them. This is suitable to solve convex minimization problems. By computing the gradient of the loss function, we can find the local minimal by probing along the negative of the gradient.

For logistic regression, the loss function is

$$\frac{1}{n}\sum_{i=0}^{n} log(1+e^{-y_i w^T x_i}) + \frac{\lambda}{2}||w||^2$$

The gradient of it is

$$\lambda w + \frac{1}{n}\sum_{i=0}^{n}\frac{y_i x_i}{1+e^{y_i w^T x_i}}$$

For SVM, the loss function is

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

Its gradient is

$$\nabla_w L(w,b) = w + \frac{C}{n}\sum_{i=1}^{n} g_w(x_i)$$

$$\nabla_w L(w,b) = \frac{C}{n}\sum_{i=1}^{n} g_b(x_i)$$

Note: $g_w(x_i) = \left\{\begin{array}{cc} -y_i x_i & 1-y_i(w^T x_i+b) >= 0 \\ 0 & others \end{array}\right.$

$g_b(x_i) = \left\{\begin{array}{cc} -y_i & 1-y_i(w^T x_i+b) >= 0 \\ 0 & others \end{array}\right.$

III. Experiments

A. Dataset

The dataset of this experiment is a9a/a9a.t from LIBSVM data. It includes 32561/16281(testing) samples and each sample includes 123/123 (testing) features.

B. Implementation

In this section, I will explain two different sub-experiments separately.

Logistic Regression

There are four optimization methods when updating the model parameters. The model parameters are all randomly initialized. The times of iteration are all 300.

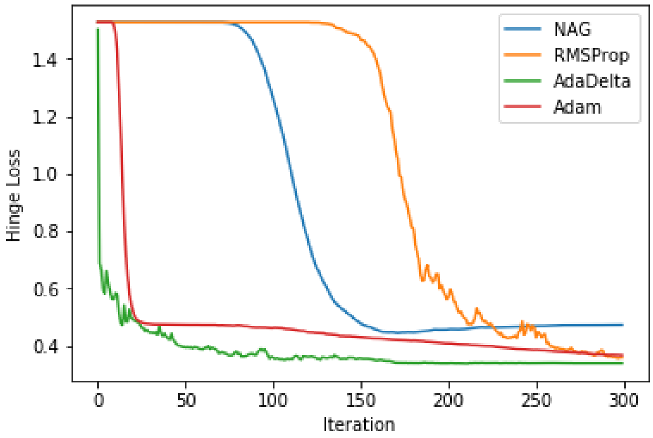
For NAG(Nesterov accelerated gradient), the hyper parameters are initialized as follows: $mu = 0.9$, $v = np.zeros(124)$, $yita\_nag = 0.001$, $batch\_size = 1.1$, $lambda\_nag = 1.1$ When promoting $yita\_nag$ to be 0.01, the loss curve declines rapidly and may miss the minimum. When the regularizer $lambda\_nag$ is set to be 1.2, the loss curve rises after about 200 iterations. To get the best result, the hyper parameter remains as initialized.

For RMSProp, the hyper parameters are initialized as follows: $capital\_g = np.random.rand(124)$, $epsilon = 1e-8$, $yita\_rmsp = 0.01$, $batch\_size = 2 * *3$, $lambda\_rmsp = 1.1$ When $yita\_rmsp$ is 0.001 the curve didn't decline after 300 iterations. When $yita\_rmsp$ is 0.1 the curve vibrated rapidly. Next, I promote $lambda\_rmsp$ to be 1.2, then the curve is satisfactory.

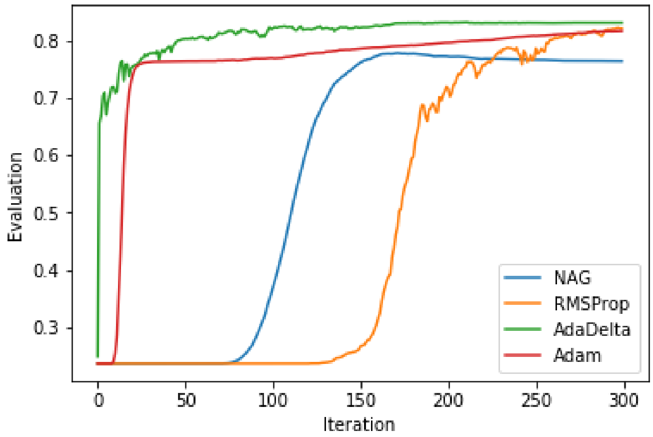
For AdaDelta, the hyper parameters are initialized as follows: $capital\_g\_adadelta = np.random.rand(124)$, $gama\_adadelta = 0.95$, $delta\_t = np.random.rand(124)$, $lambda_adadelta = 1.5$, $epsilon = 1e-8$ With $lambda\_adadelta$ increasing, the curve didn't reach relatively small value.

For Adam, the hyper parameters are initialized as follows: $yita\_adam = 0.1$, $beta = 0.9$, $gama_adam = 0.999$, $moment = np.random.rand(124)$, $capital\_g\_adam = np.random.rand(124)$, $lambda\_adam = 1.2$ Adam is pleasant. It's easier to adjust hyper parameters and the curve is satisfactory. When $yita\_adam$ is 0.01, the curve didn't decline. When $lambda\_adam$ is 1.5 the early vibration got worse, the same situation as that of when $yita\_adam$ is 1.0.

Here is the figure of loss curves:



Here is the figure of accuracy:



SVM

The model parameters are also randomly initialized. The times of iteration are 200.

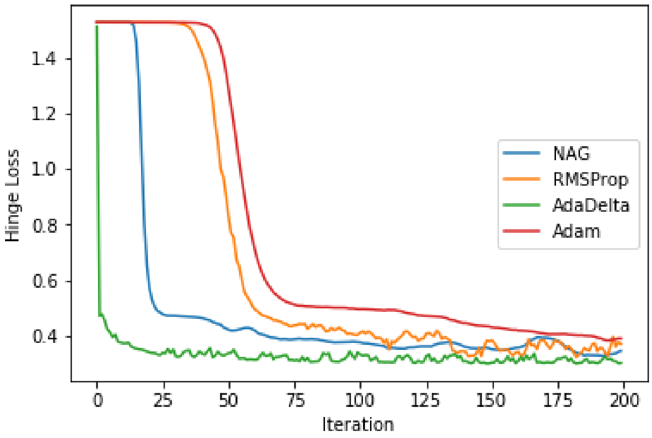
For NAG, the hyper parameters are initialized as follows: $mu = 0.9$, $v = np.zeros(124)$, $yita\_nag = 0.01$ When $yita\_nag$ is 0.1, the curve vibrates rapidly.

For RMSProp, the hyper parameters are initialized as follows: $yita\_rmsp = 0.01$, $capital\_g\_rmsp = np.zeros(124)$, $epsilon = 1e - 8$ When $yita\_rmsp$ is 0.1, the curve vibrates rapidly.

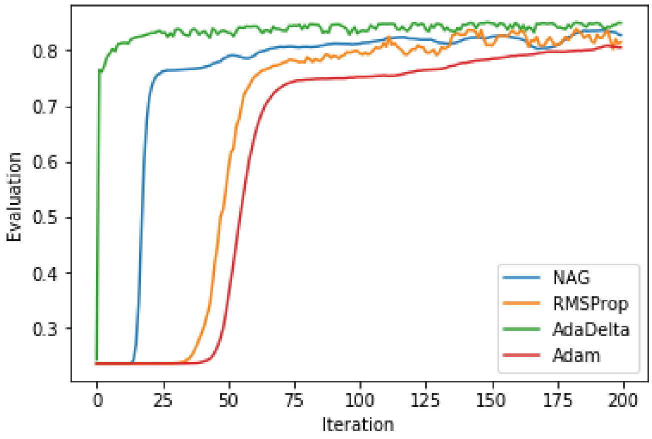
For AdaDelta, the hyper parameters are initialized as follows: $capital\_g\_delta = np.random.rand(124)$, $gama\_delta = 0.95$, $delta\_t = np.random.rand(124)$

For Adam, the hyper parameters are initialized as follows: $yita\_adam = 0.1$, $beta = 0.9$, $gama\_adam = 0.999$, $moment = np.random.rand(124)$, $capital\_g\_adam = np.random.rand(124)$ .When $yita\_adam$ is 0.01, the curve didn't decline.

Here is the figure of loss curves:



Here is the figure of accuracy:



IV. Conclusion

In conclusion, this paper demonstrates the process of applying four different optimizatioin methods to Logistic Regression and SVM Classification. The results are shown in the form of figures.

In this experiment, I learned about four different optimization methods applied in SGD. The parameter adjusting process is quite tough and sometimes exhausting but I'm glad that I made it.