華南理工大學

South China University of Technology

The Experiment Report of *Machine Learning*

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:
Junwen Mo

Supervisor:
Mingkui Tan

Student ID:
201530612545

Grade:
Undergraduate

December 14, 2017

Logistic Regression, Linear Classification and Gradient Descent

***Abstract*—In this experiment, we try using the logistic regression and linear classification(svm) to solve the classification problem with different stochastic gradient descent algorithms. In this experiment, I will show the comparison between the logistic regression and show the performance comparison on different stochastic gradient descent algorithms.**

I. Introduction

IN this experiment, we were asked to solve the classification problem with logistic regression and linear classification. We all know that this two algorithms are the easy but powerful methods to solve the classification problems. Also, in this experiment, we should try using the stochastic gradient descent(SGD) to solve the problem because the dataset is too big to load into the memory. Sometimes using the gradient descent with the whole dataset will cause the memory error. And using SGD will speed up the learning process. In this experiment, I will show that the performance of the logistic regression and linear classification with different SGD algorithms.

II. Methods and Theory

The prediction function of the logistic regression is equation(1). The thing is tagged positive class with y_{pre} greater than 0.5 and is tagged negative class with y_{pre} smaller than zero.

$$y_{pre} = \frac{1}{1+e^{-(wx+b)}} \qquad (1)$$

And the goal function of the logistic regression is equation(2). Derivation of the goal function is show in equation(3).

$$min \quad J(w) = \frac{\lambda}{2}w^2 + \frac{1}{n}\sum_{i=1}^{n} log(1+e^{-y_i w^T x_i}) \qquad (2)$$

$$\frac{\partial J}{\partial w} = \frac{1}{n}\sum_{i=1}^{n} \frac{-y_i x_i}{1+e^{-y_i w^T x_i}} + \lambda w \qquad (3)$$

The prediction function of the linear classification is equation(4). And the goal function and the derivation are equation(5) and equation(7).

$$y_{presvm} = wx + b \qquad (4)$$

$$min \quad J(w)_{svm} = \frac{1}{2}w^2 + C\sum_{i=1}^{n} max(0, 1-y_i(w^T x_i + b)) \qquad (5)$$

$$\frac{\partial J(w)_{svm}}{\partial w} = w + C\sum_{i=1}^{n} g_w(x_i) \qquad (6)$$

$$g_w(x) = -y_i x_i \quad 1 - y_i(w^T x_i + b) >= 0 \qquad (7)$$

$$g_w(x) = 0 \quad 1 - y_i(w^T x_i + b) < 0 \qquad (8)$$

III. Experiments

A. Dataset

Experiment uses a9a of LIBSVM Data, including 32561/16281(testing) samples and each sample has 123/123 (testing) features.

B. Implementation

To implement the logistic regression, I use standard normal distribution to initialize the variable w. And then implement the gradient and loss function of the logistic regression according to equation(1) and equation(3), then implement various SGD algorithms. For different algorithms, I try tuning the parameter to get the best classification accuracy and the tuning result is as Table I. Then, we can get the lists of loss change and plot it(See Fig1). The result about the accuracy is show in Table II. In this experiment, I try the gradient with and without bias adding to the regularization item but it showed that it does not matter.

TABLE I
Logistic regression parameters

$algorithm$	$mini-batch$	NAG	$RMSProp$	$AdaDelta$	$Adam$
η	0.2	0.02	0.01		0.01
λ	0.01	0.001	0.001	0.001	0.001
γ		0.9	0.9	0.4	0.99
β					0.9
$epoch$	1000	1000	1000	1000	1000
$batch_size$	1024	1024	1024	1024	1024

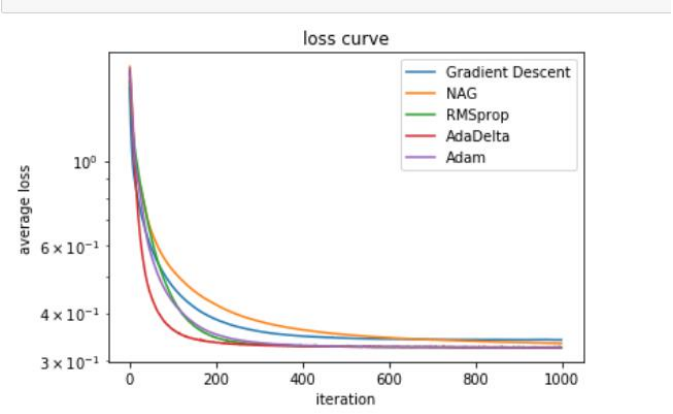

Fig. 1. The loss curve of the logistic regression with different gradient descent algorithm.(log scale)

The implementation of the linear classification is similar to the implementation of the logistic regression. Also, I use normal distribution to initialize the variable w. The gradient

TABLE II
LOGISTIC REGRESSION BEST ACCURACY

$algorithm$	$accuracy$
$mini-batch$	84.46%
NAG	84.63%
$RMSProp$	84.96%
$AdaDelta$	85.11%
$Adam$	85.05%

function and the loss function is according to equation(6)(7)(8) and equation(5) namely. The hyper-parameters are tuned as Table IV. The loss curves of different SGD algorithms are as Fig.2 and the accuracy is showed in Table III.

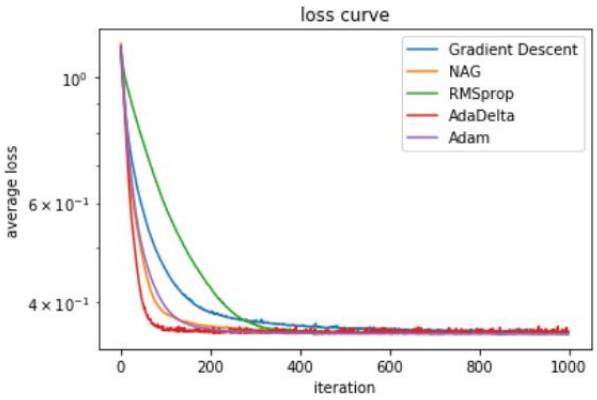

Fig. 2. The loss curve of the linear classification with different gradient descent algorithm.(log scale)

TABLE III
LINEAR CLASSIFICATION BEST ACCURACY

$algorithm$	$accuracy$
$mini-batch$	84.87%
NAG	84.90%
$RMSProp$	84.97%
$AdaDelta$	84.87%
$Adam$	84.95%

C. Different between gradient descent(GD) and mini-batch

In this experiment, we use the SGD to avoid memory error and speed up the training process. So I want to compare the change of the loss value of the training set and testing set between the GD and mini-batch. In this experiment, I use the logistic regression with the gradient as equation(3) and loss as equation(2). The learning rate is 0.001 and the lambda is 0.01. Epoch is 1000. The loss curve of gradient descent is as Fig.3. And the loss curve of the mini-batch is as Fig.4. We can see that the training loss curve of the mini-batch is unstable but it has the tendency to go down. The loss curve of the gradient descent is more stable but it takes a lot of memory and slow down the computer I used. Because the dataset is so big that it's hard to load the whole dataset into the memory, the data exchange between the disk and the memory will become more frequently, which will slow down the training process.

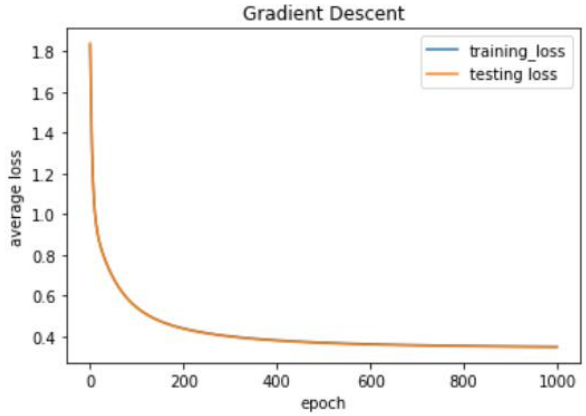

Fig. 3. The loss curve of the gradient descent.

TABLE IV
LINEAR CLASSIFICATION PARAMETERS

$algorithm$	$minibatch$	NAG	$RMSProp$	$AdaDelta$	$Adam$
η	0.0004	0.00008	0.005		0.01
C	0.9	1	1	0.9	1
γ		0.9	0.9	0.3	0.999
β					0.9
$epoch$	1000	1000	1000	1000	1000
$batchsize$	1024	1024	1024	1024	1024

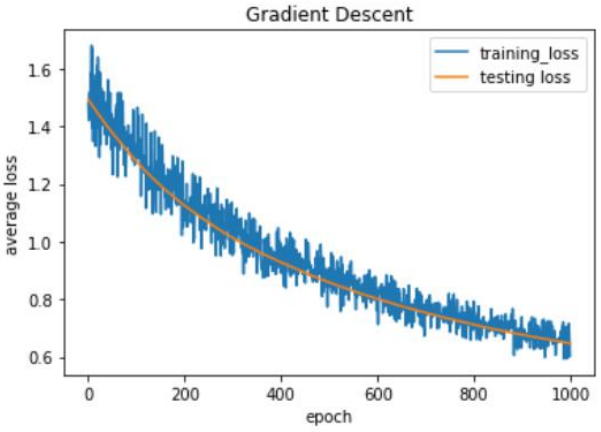

Fig. 4. The loss curve of the mini-batch.

IV. CONCLUSION

In this experiment, we use the logistic regression and linear classification to solve the classification problem. Also, we try different stochastic gradient descent and compare them. The stochastic gradient descent can solve the memory error and speed up the training process but it will make the loss unstable. Different algorithms have different convergence rates. Adadelta converge fastest and it does not require the learning rate but its loss is unstable. Adam and NAG also run fast. And RMSprop is a little faster than the gradient descent.