South China University of Technology

The Experiment Report of Machine Learning

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:
Jalil Hassan

Supervisor:
Qingyao Wu

Student ID：
201722800097

Grade:
Graduate

December 15, 2017

Logistic Regression, Linear Classification and Stochastic Gradient Descent

Abstract—The experiment consists of two parts, one is logistic regression, the other is linear classification. Both models are updated with four different gradient descent methods and tested in the *a9a* in *LIBSVM Data.* For logistic regression, the accuracy of validation set used NAG, RMSProp, AdaDelta and Adam all can reach 81.598%. For linear classification, the accuracy of validation set used NAG, RMSProp, AdaDelta and Adam all can reach 82.949%.

I. Introduction

The experiment is divided into two parts, the first is a logistic regression, the second is a linear classification, both are solved by four different gradient descent methods. Through analyzing the different updating process, we can further understand the principle of gradient descent. We will practice on a bigger scale datasets to understand the process of optimization and parameter adjustment. We expect both logistic regression and c linear classification can get higher accuracy.

II. Methods and Theory

A. Logistic Regression

For data sets $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \dots, (\boldsymbol{x}_N, y_N)\}$, where $\boldsymbol{x}_i = (x_{i1}; x_{i2}; \dots; x_{id}; 1), y_i \in \{0, 1\}$, we hope to get a logistic function,

$$f_w(\boldsymbol{x}_i) = \sigma(\boldsymbol{w}^T \boldsymbol{x}_i) \tag{1}$$

where $\boldsymbol{w} = (w_1; w_2; \dots; w_d; b)$ and $\sigma(z) = \frac{1}{1+exp(-z)}$, to make the $f_w(\boldsymbol{x}_i)$ got by the model is 1 when it is greater than the threshold and 0 when it is less than the threshold. We hope the results obtained are same with the actual label of $\boldsymbol{x}_i$ as much as possible.

We use the gradient descent method to solve the model parameters $\boldsymbol{w}$. The loss function is

$$-lnL(\boldsymbol{w}) = -\frac{1}{K}\left[\boldsymbol{y}^T ln f_w(\boldsymbol{X}) + (1-\boldsymbol{y})^T ln\left(1 - f_w(\boldsymbol{X})\right)\right] + \lambda \frac{1}{N}\boldsymbol{w}^T\boldsymbol{w} \tag{2}$$

where $\boldsymbol{X} = (\boldsymbol{x}_1^T; \boldsymbol{x}_2^T; \dots; \boldsymbol{x}_K^T)$, $\boldsymbol{y} = (y_1; y_2; \dots; y_K)$, λ is a regularization parameters that is artificially adjusted. K is a value between 1 and *N*, used to implement the stochastic gradient descent.

Our goal is to minimize this loss function, so we need to update the parameters along the negative direction gradient of parameter $\boldsymbol{w}$. Find the partial derivative of $\boldsymbol{w}$ for the loss function, we get the gradient of them

$$G(\boldsymbol{w}) = \frac{1}{K}(f_w(\boldsymbol{X}) - \boldsymbol{y})^T\boldsymbol{X} + \lambda\frac{2}{K}\boldsymbol{w} \tag{3}$$

Here we apply four different gradient descent methods to update $\boldsymbol{w}$.

NAG:

$$\boldsymbol{g}^t = G(\boldsymbol{w}^{t-1} - \gamma\boldsymbol{v}^{t-1}) \tag{4}$$
$$\boldsymbol{v}^t = \gamma\boldsymbol{v}^{t-1} + \eta\boldsymbol{g}^t \tag{5}$$
$$\boldsymbol{w}^t = \boldsymbol{w}^{t-1} - \boldsymbol{v}^t \tag{6}$$

RMSProp:

$$\boldsymbol{g}^t = G(\boldsymbol{w}^{t-1}) \tag{7}$$
$$\boldsymbol{H}^t = \gamma\boldsymbol{H}^t + (1-\gamma)\boldsymbol{g}^t \odot \boldsymbol{g}^t \tag{8}$$
$$\boldsymbol{w}^t = \boldsymbol{w}^{t-1} - \frac{\eta}{\sqrt{\boldsymbol{H}^t + \epsilon}} \odot \boldsymbol{g}^t \tag{9}$$

AdaDelta:

$$\boldsymbol{g}^t = G(\boldsymbol{w}^{t-1}) \tag{10}$$
$$\boldsymbol{H}^t = \gamma\boldsymbol{H}^t + (1-\gamma)\boldsymbol{g}^t \odot \boldsymbol{g}^t \tag{11}$$
$$\boldsymbol{\Delta w}^t = -\frac{\sqrt{\boldsymbol{\Delta}_{t-1} + \epsilon}}{\sqrt{\boldsymbol{H}^t + \epsilon}} \odot \boldsymbol{g}^t \tag{12}$$
$$\boldsymbol{w}^t = \boldsymbol{w}^{t-1} + \boldsymbol{\Delta w}^t \tag{13}$$
$$\boldsymbol{\Delta}_t = \gamma\boldsymbol{\Delta}_{t-1} + (1-\gamma)\boldsymbol{\Delta w}^t \odot \boldsymbol{\Delta w}^t \tag{14}$$

Adam:

$$\boldsymbol{g}^t = G(\boldsymbol{w}^{t-1}) \tag{15}$$
$$\boldsymbol{m}^t = \beta_1\boldsymbol{m}^{t-1} + (1-\beta_1)\boldsymbol{g}^t \tag{16}$$
$$\boldsymbol{H}^t = \gamma\boldsymbol{H}^t + (1-\gamma)\boldsymbol{g}^t \odot \boldsymbol{g}^t \tag{17}$$
$$\alpha = \eta\frac{\sqrt{1-\gamma^t}}{1-\beta^t} \tag{18}$$
$$\boldsymbol{w}^t = \boldsymbol{w}^{t-1} - \alpha\frac{\boldsymbol{m}^t}{\sqrt{\boldsymbol{H}^t + \epsilon}} \tag{19}$$

where, γ, η, β are super-parameters that are artificially adjusted.

$\boldsymbol{w}$ is updated by equations (4) to (19) until the loss function converges.

B. Linear classification

For data sets $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \dots, (\boldsymbol{x}_m, y_m)\}$, where $\boldsymbol{x}_i = (x_{i1}; x_{i2}; \dots; x_{id}; 1)$, $y_i \in \{-1, +1\}$. We want to get a linear equation

$$f(\boldsymbol{x}_i) = \boldsymbol{w}^T\boldsymbol{x}_i \tag{20}$$

where $\boldsymbol{w} = (w_1; w_2; \dots; w_d; b)$, to make the $f(\boldsymbol{x}_i)$ got by the model is +1 when it is greater than the threshold (here 0) and -1 when it is less than the threshold. We hope the results obtained are same with the actual label of $\boldsymbol{x}_i$ as much as possible.

We also use the gradient descent method to solve the model's parameters. The loss function is

$$L(\boldsymbol{w}, b) = \sum_{i=1}^{K} l_{hinge}(\boldsymbol{x}_i, y_i) + \lambda||\boldsymbol{w}||_2$$
$$= \frac{1}{K}\left(\sum_{i=1}^{K} \max(0, 1 - y_i f(\boldsymbol{x}_i)) + \lambda||\boldsymbol{w}||_2\right) \tag{21}$$

where λ is a regularization parameters that is artificially adjusted. K is a value between 1 and N, used to implement the stochastic gradient descent.

Our goal is to minimize this loss function, so we need to update the parameters along the negative direction gradient of parameter $\boldsymbol{w}$. Find the partial derivative of $\boldsymbol{w}$ for the loss function, we get the gradient of them

$$G(w_j) = \frac{\partial L}{\partial w_j} = \frac{1}{K}\sum_{i=1}^{K}[-\delta(y_i f(\boldsymbol{x}_i) < 1) y_i x_{ij}] + \frac{1}{K}\lambda \boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{w} w_j \tag{22}$$

Then update $\boldsymbol{w}$ using the four different gradient descent methods shown in equations (4) to (19) until the loss function converges.

III. Experiment

A. Dataset

Both experiments use *a9a* of *LIBSVM Data*, including 32561/16281(testing) samples and each sample has 123/123 (testing) features.

B. Implementation

About logistic regression, after reading the dataset, we need to initialize the model, all the parameters used in the experiment are listed in the TABLE I.

TABLE I
PARAMETERS INITIALIZATION

Max Iterations	500
Lambda	0.01

gamma	NAG = 0.9 RMSProp = 0.9
	AdaDelta =0.9 Adam = 0.9

eta	1e-8
beta	Adam = 0.9/0.99

About the $\boldsymbol{w}$, initialize them randomly in range(0, 1).

And then, about the linear regression problem. Fig.1-1 show the change of loss of Gradient Decent method NAG with learning rate which equals to 0.1, Fig.1-2 is Gradient Decent method RMSProp, Fig.1-3 is Gradient Decent method AdaDelta, Fig.1-4 is Gradient Decent method Adam. Fig 1-5 to Fig. 1-6 show the change of loss of Gradient Decent method NAG with different learning rate, and we want figure out how different hyper-parameter learning change will change the train and valid loss.

About the linear classification, after reading the dataset, we also need to initialize the model, all the parameters used in the experiment are the same as linear regression, shown in Table I. And Fig.2-1 show the change of loss of Gradient Decent method NAG with learning rate which equals to 0.1, Fig.2-2 is Gradient Decent method RMSProp, Fig.2-3 is Gradient Decent method AdaDelta, Fig.2-4 is Gradient Decent method Adam. Fig 2-5 to Fig. 2-6 show the change of loss of Gradient Decent method NAG with different learning rate, and we want figure out how different hyper-parameter learning change will change the train and valid loss. Fig 2-7 to Fig. 2-9 shows the change of accuracy with different learning and gradient decent method, but we show point out that the NGA won't be influenced by learning rate, which means it can adjust learning rate by itself.

The $\boldsymbol{w}$ are also initialized randomly in range(0, 1). And then, use the model described in II.*B* to update the $\boldsymbol{w}$until the loss function has converged. The same as logistic regression, the curves of the four different gradient descent methods in both two figures can have different effective.

Fig. 1-1. Change of loss value with the number of iterations in linear regression in learning rate 0.1 with NGA.

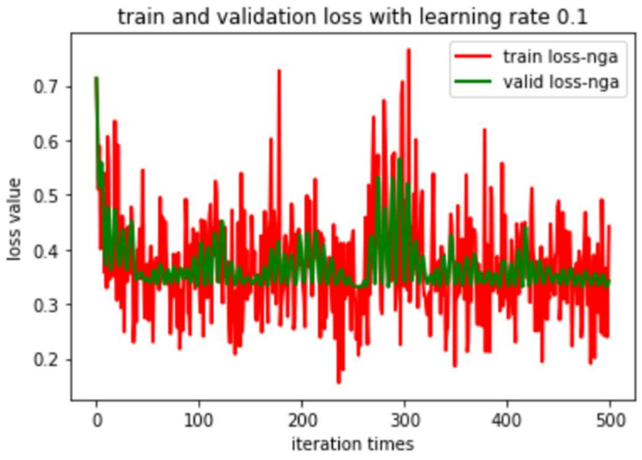


Fig. 1-2. Change of loss value with the number of iterations in linear regression in learning rate 0.1 with rmsprop.

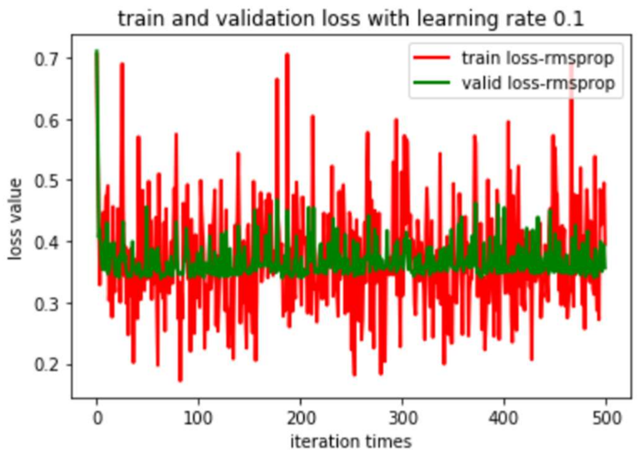


Fig. 1-3. Change of loss value with the number of iterations in linear regression in learning rate 0.1 with adadelta.

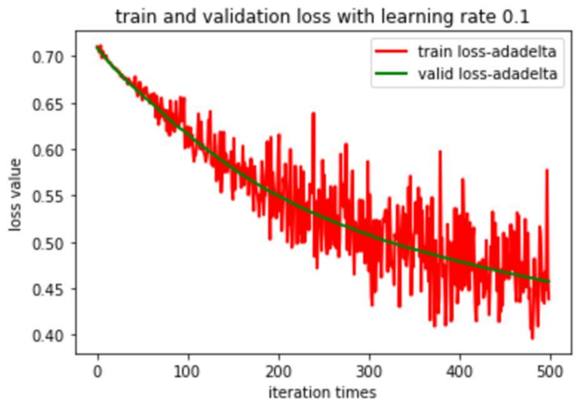


Fig. 1-4. Change of loss value with the number of iterations in

linear regression in learning rate 0.1 with adam.

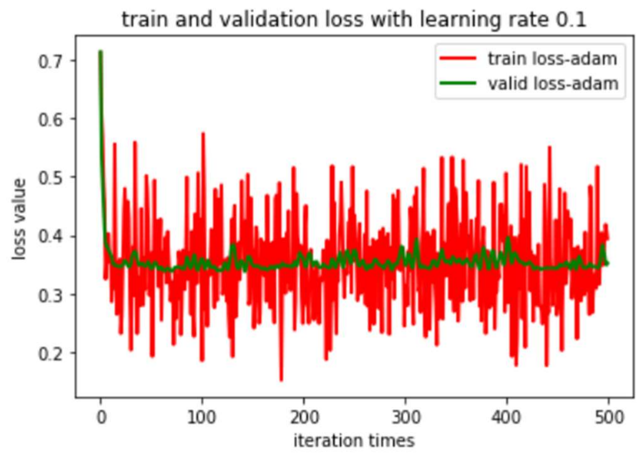


Fig. 1-5. Change of loss value with the number of iterations in linear regression in learning rate 0.01 with NGA.

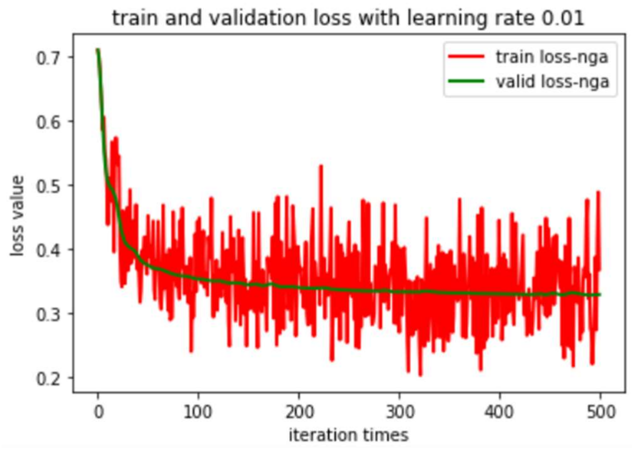


Fig. 1-6. Change of loss value with the number of iterations in linear regression in learning rate 0.001 with NGA.

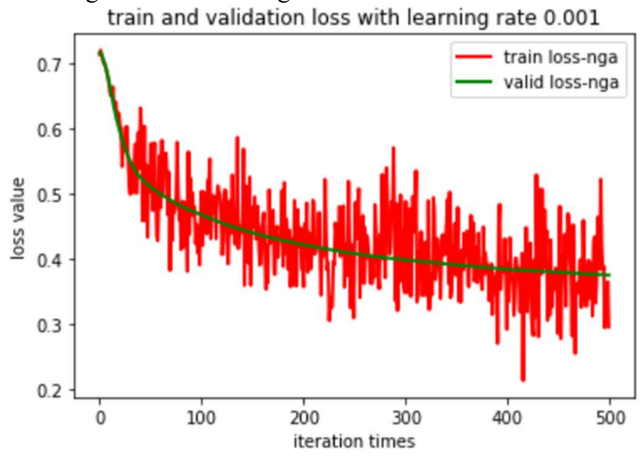


Fig. 2-1. Change of loss value with the number of iterations in linear regression in learning rate 0.1 with NGA.

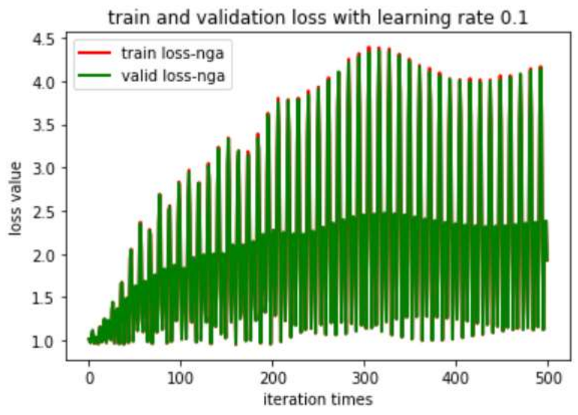


Fig. 2-2. Change of the loss with the number of iterations in the validation set in learning rate 0.1 with rmsprop .

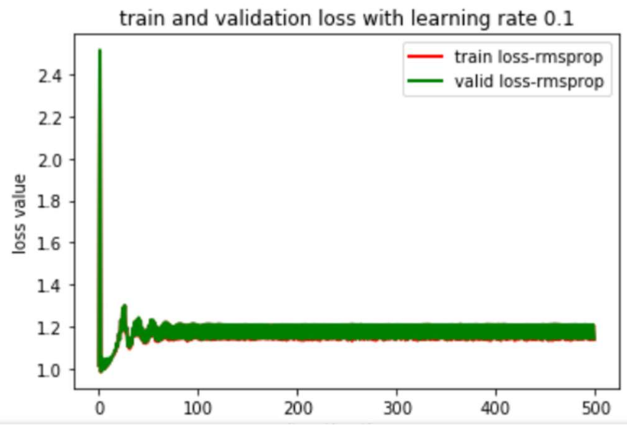


Fig. 2-3. Change of the loss with the number of iterations in the validation set in learning rate 0.1 with AdaDelta.

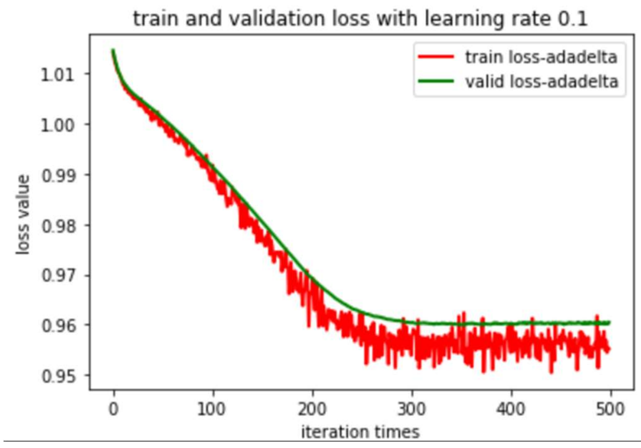


Fig. 2-4. Change of the loss with the number of iterations in the validation set in learning rate 0.1 with Adam.

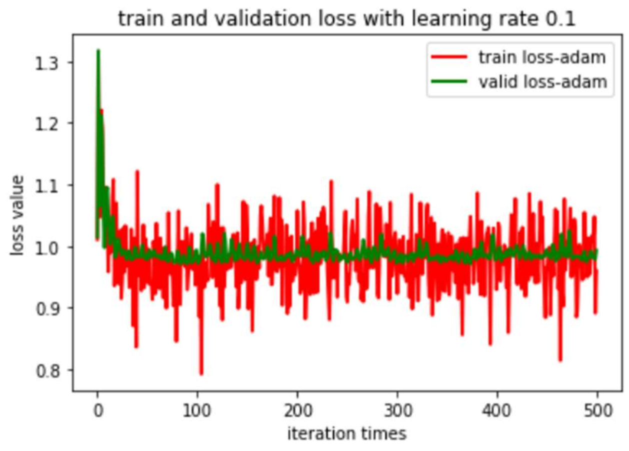


Fig. 2-5. Change of the loss with the number of iterations in the validation set in learning rate 0.01 with NGA.

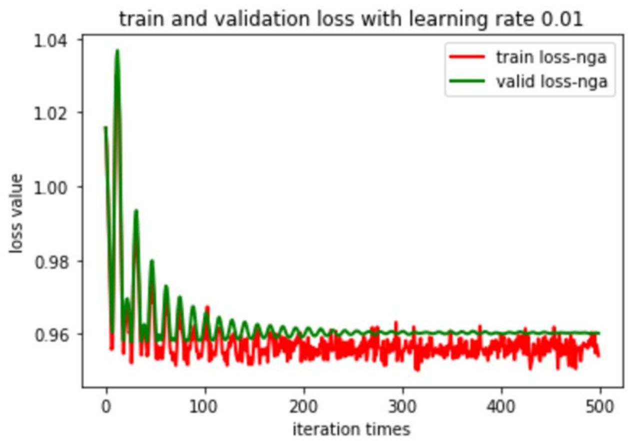


Fig. 2-6. Change of the loss with the number of iterations in the validation set in learning rate 0.001 with NGA.

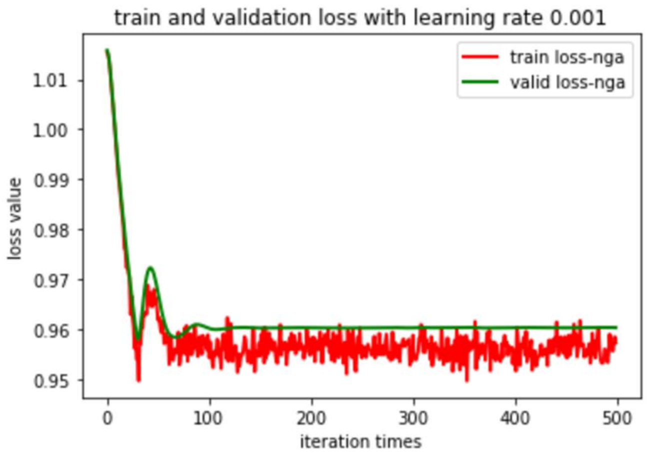


Fig. 2-7. Change of the accuracy with the number of iterations and with different method, learning rate 0.1

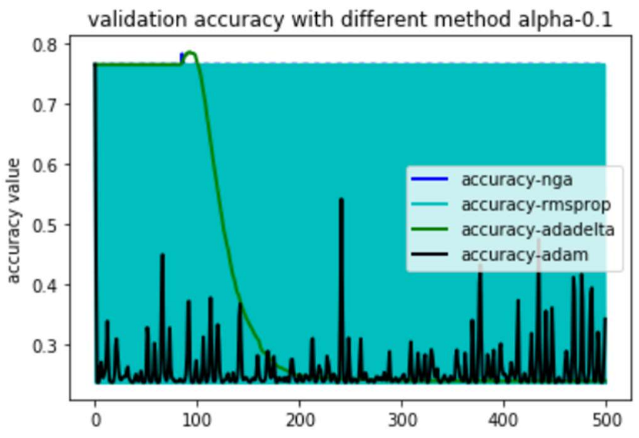


Fig. 2-8. Change of the accuracy with the number of iterations and with different method, learning rate 0.01

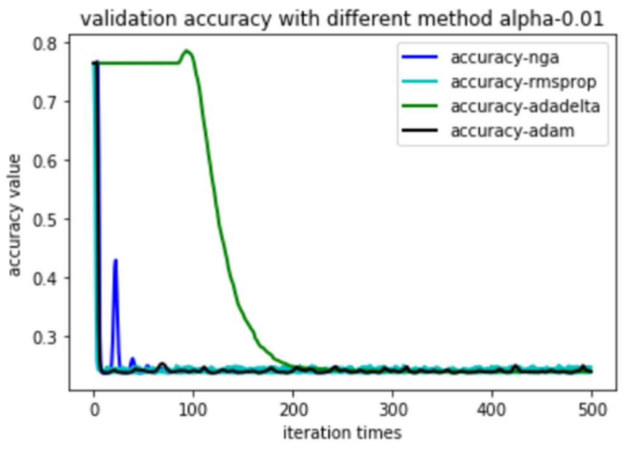
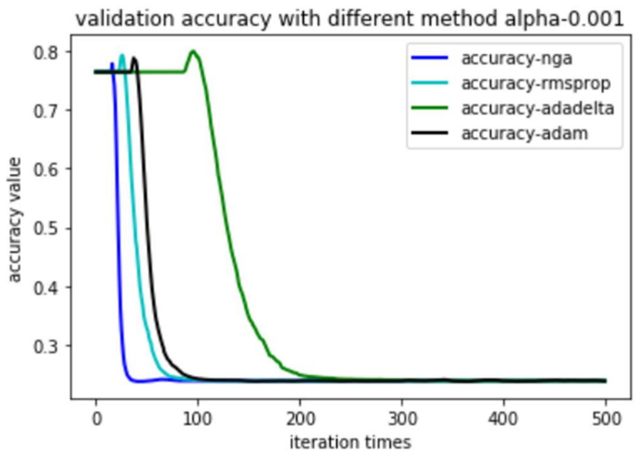


Fig. 2-9. Change of the accuracy with the number of iterations and with different method, learning rate 0.001



IV. CONCLUSION

The experiment consists of two parts. The first one is logistic regression. The model parameters are updated by four different gradient descent methods. The loss can drop into low level. The second one is a linear classification. Similarly, the model parameters are updated through the same four gradient descent methods.

From above pictures, though all method can have different learning rate in each iteration, by with the different fixed learning rate, the same Gradient descent method can have different effect on data, with larger learning rate, the train loss will have much fluctuation, and with smaller learning rate, the curve will become convex, besides that, Adadelta won't use learning rate which it can adjust it by iteself. Actually, not only the learning rate, other hyper-parameters will have different influence on different method.

As for the accuracy, it's very strange that my accuracy doesn't increase with the iteration step, but drop, and I adjust many times, but it still just like the above pictures, maybe later I can figure out it.

Through this experiment, further understand the improved version of gradient descent. Through practice in a bigger scale data set, get more experience in adjusting parameters.