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:
Zhifani Xu

Supervisor:
Qingyao Wu

Student ID:
201530613269

Grade:
Undergraduate

December 14, 2017

Logistic Regression, Linear Classification and Stochastic Gradient Descent

Abstract*—Implementing question of logistic regression and liner classification by stochastic gradient descent(SGD). And updating model parameters using different optimized methods(NAG, RMSprop, AdaDelta and Adam

Ⅰ .NTRODUCTION

Logistic regression and linear classification are basic knowledge of the machine learning. And in this experiment I used stochastic gradient descent(SGD) to implement the previous two problems.At the same time I used four optimized methods to optimize both of them. This experiment helps me understanding the logistic regression and linear classification. What's more, the four methods works well.

Ⅱ . METHODS AND THEORY

A. Logistic Regression

The logistic regression function is a common equation as follows:

$$\sigma(t) = \frac{1}{1+e^{-t}} \tag{1}$$

Where $y_i = \{0,1\}$, we can obtain the continuous function:

$$\mathrm{h}_\omega(x) = g(\omega^T x) = \frac{1}{1+e^{-\omega^T x}} \tag{2}$$

The loss function is:

$$\mathrm{J}(\omega) = -\frac{1}{\mathrm{m}}[\sum_{i=1}^{m} y_i \log h_\omega(x_i) + (1-y_i)\log(1-h_\omega(x_i))] \tag{3}$$

The derivation of this loss function is

:

$$\frac{\partial \mathrm{J}(\omega)}{\partial \omega} = -y\frac{1}{h_\omega(x)}\frac{\partial h_\omega(x)}{\partial \omega} + (1-y)\frac{1}{1-h_\omega(x)}\frac{\partial h_\omega(x)}{\partial \omega} \tag{4}$$

B. Linear Classification

I implemented linear classification using support vector machine(SVM). The margin is defined as follows:

$$\xi = \max(0, 1 - t * y) \tag{5}$$

We have the loss function of SVM:

$$\mathrm{J}(\omega) = \frac{1}{2}\| w \|^2 + C\sum_i \max(0, 1 - y_i(\omega_i^x + b)) \tag{6}$$

and its derivation:

$$\frac{\partial \ell}{\partial w_i} = \begin{cases} -t \cdot x_i & \text{if } t \cdot y < 1 \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

C. Optimization Methods

Some optimization methods had been used in this e- xperiment such as NAG, RMSProp, AdaDelta and Adam.

a. NAG: NAG is a way to give our momentum term this kind of prescience that slowing down before the loss rise again. When calculating g_t, we use $\theta_{t-1} - \gamma\nu_{t-1}$ instead of θ_{t-1}, which is equivalent to using momentum to predict where to go next.

$$\begin{aligned} \mathrm{g}_t &\leftarrow \nabla J(\theta_{t-1} - \gamma\nu_{t-1}) \\ \nu_\mathrm{t} &\leftarrow \gamma\nu_{t-1} + \eta g_t \\ \theta_t &\leftarrow \theta_{t-1} - \nu_\mathrm{t} \end{aligned} \tag{8}$$

b. RMSProp: RMSprop and Adadelta have both been developed independently around the same time stemming from the need to resolve Adagrad's radically diminishing learning rates. RMSprop in fact is identical to the first update vector of AdaDelta.

$$\begin{aligned} \mathbf{g}_t &\leftarrow \nabla J(\boldsymbol{\theta}_{t-1}) \\ G_t &\leftarrow \gamma G_t + (1-\gamma)\mathbf{g}_t \odot \mathbf{g}_t \\ \boldsymbol{\theta}_t &\leftarrow \boldsymbol{\theta}_{t-1} - \frac{\eta}{\sqrt{G_t + \epsilon}} \odot \mathbf{g}_t \end{aligned} \tag{9}$$

c. AdaDelta: Adadelta restricts the window of accumulated past gradients to some fixed size ω, instead of accumulating all past squared gradients. Moreover, instead of inefficiently storing ω previous squared gradients, the sum of gradients is recursively defined as a decaying average of all past squared gradients. The running average $\mathrm{E}[g^2]_{t-1}$ at time step t then depends (as a fraction γ similarly to the Momentum term) only on the previous average and the current gradient:

$$\begin{aligned} E[g^2]_t &= \gamma E[g^2]_{t-1} + (1-\gamma)g_t^2 \\ \Delta\theta_t &= -\frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t \\ \theta_{t+1} &= \theta_t + \Delta\theta_t \end{aligned} \tag{10}$$

d. Adam: Adam is another method that computes adaptive learning rates for each parameter. In addition to storing an exponentially decaying average of past squared gradients r_t like Adadelta and RMSprop, Adam also keeps an exponentially decaying average of past gradients mt, similar to momentum:

$$
\begin{aligned}
\mathbf{g}_t &\leftarrow \nabla J(\boldsymbol{\theta}_{t-1}) \\
\mathbf{m}_t &\leftarrow \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\mathbf{g}_t \\
G_t &\leftarrow \gamma G_t + (1-\gamma)\mathbf{g}_t \odot \mathbf{g}_t \\
\alpha &\leftarrow \eta \frac{\sqrt{1-\gamma^t}}{1-\beta^t} \\
\boldsymbol{\theta}_t &\leftarrow \boldsymbol{\theta}_{t-1} - \alpha \frac{\mathbf{m}_t}{\sqrt{G_t + \epsilon}}
\end{aligned}
\tag{11}
$$

III.EXPERIMENTS

A. Data set

This experiment uses a9a which provided by LIBSVM Data, including 32561 training, 16281 testing records and each record in training or testing set has 123 features.

B. Implementation

In both logistic regression and linear classification, the parameter loops is limited to 666, for comparing the performance of four optimization methods. At the same time, other parameters like ε in those methods are adjusted respectively for better accuracy.

The graph of loss descent using NAG, RMSProp, AdaDelta and Adam respectively are shown as Fig.1 amd Fig.2:

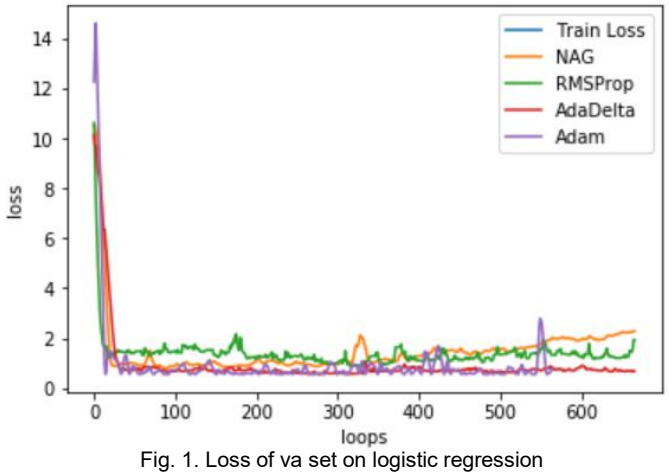

Fig. 1. Loss of va set on logistic regression

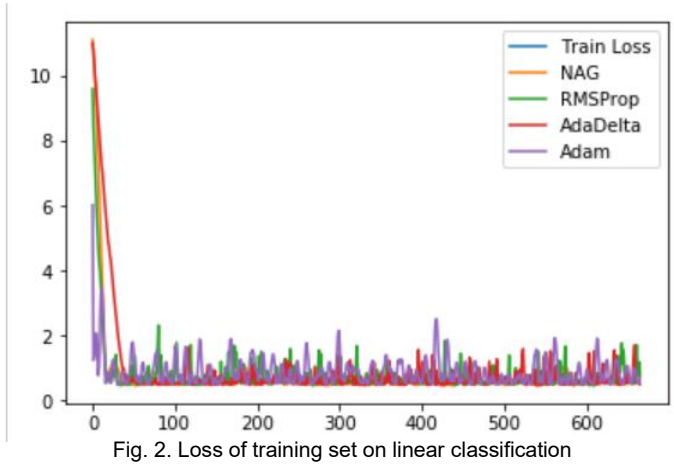

Fig. 2. Loss of training set on linear classification

IV. CONCLUSION

I have a deeper understanding of the logistic regression and the linear classification after this experiment. What's more, I have known that the principle of stochastic gradient descent(SGD). Writing the code of optimization methods such as NAG, PMSProp, AdaDelta and Adam makes me know more about optimization in machine learning, which may help me improve the efficiency of my code in the future.