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:
Da Ren

Supervisor:
Qingyao Wu

Student ID:
201720144900

Grade:
Graduate

December 15, 2017

Logistic Regression, Linear Classification and Stochastic Gradient Descent

***Abstract*—Gradient descent plays an important role in machine learning. It can help us get the optimal value of a function. However, one of the most important task in machine learning is to get the optimal value for loss function. Gradient descent is widely used in this task. However, people gradually find that traditional gradient descent performs not well when dealing with some tasks. Therefore, there are many modified gradient descent methods which are proposed. I implement NAG, RMSProp, AdaDelta and Adam in logistic regression and linear regression in this experiment.**

I. Introduction

GRADIENT descent is a important method in machine learning. The most important tasks in many machine learning models is to update or calculate their parameters which can lead the corresponding loss functions value to be minimal value. Although gradient descent is widely used, researchers find that it has its own limitation. Therefore, there are many modified gradient descent methods which are proposed, *e.g*. NAG, RMSProp, AdaDelta and Adam. In this experiment, I implement logistic regression and linear classification. And I use different methods to optimize their loss functions. My experiments' results show that all of them can optimize the models. However, their speed may be different.

II. Methods and Theory

Gradient descent is flexible since it can be easily used in different kinds of loss function. It is based on the observation that if the multi-variable function $F(x)$ is defined and differentiable in a neighborhood of a point a, then $F(x)$ decreases fastest if one goes from a in the direction of the negative gradient of F at a. Therefore, if we want to minimize the value of $F(x)$, we can move a against the gradient of F. Therefore, a can be updated by Eq. (1).

$$a_{t+1} = a_t + \lambda \Delta F(a_t) \tag{1}$$

where a_t is the value of a at time t, λ is the learning rate which is always set to be a small value like 0.1. $\Delta F(a_t)$ is the gradient value of $F(a_t)$.

There are two experiments in my work. One is logistic regression, the other is linear classification. Therefore I have to choose different loss function to different tasks.

Logistic regression is a function that its output is limited from 0 to 1. It can be calculated as following:

$$y' = 1/(1 + e^{-z}) \tag{2}$$

where

$$z = W^T X + b \tag{3}$$

where W is the weight matrix and b is the bias variable. However, we can add 1 at the end of X so that the weight of the last column can act at the bias variable. Note X' to be the features which is add 1 at the last column. We can calculate z as following:

$$z = W^T X' \tag{4}$$

To train the logistic regression, I choose log-likehood function as my loss function. It can be calculated by Eq. (5)

$$L_1 = -1/n[\sum_{i=1}^{n} y_i log h_w(x_i) + (1 - y_i) log(1 - h_w(x_i))] \tag{5}$$

where

$$h_w(X) = 1/(1 + e^{-W^T X}) \tag{6}$$

In Eq. (5), n is the total number of data and y_i is the target value of data. The next step is to calculate the derivation of Eq. (5). The calculation process is in the following:

$$\begin{aligned} \Delta L_1/\Delta w = &-y * 1/h_w(X) * \Delta h_w(X)/\Delta W + \\ &(1 - y) * 1/(1 - h_w(X)) * \Delta h_w(X)/\Delta W \\ &= (h_w(X) - y)X \end{aligned} \tag{7}$$

After we get the the derivation of Eq. (6), we can calculate the gradient according to Eq. (7) and update weight matrix according Eq. (1)

In the linear classification task, I use the Eq. (8) to calculate output.

$$y' = W^T X' \tag{8}$$

Since the function is different, I choose another loss function. The loss function I choose is called hinge loss, which can be described in Eq. (9)

$$L_2 = \left\|W\right\|^2 /2 + C/m \sum_{i=1}^{m} max(0, 1 - y_i(W^T x_i + b)) \tag{9}$$

where C is a hyperparameter which can be adjusted in different dataset.

To use the gradient descent to optimize the weight matrix, we have to calculate the the derivation of Eq. (9).

$$\Delta L_2/\Delta w = W + C/m \sum_{i=1}^{m} g_w(x_i) \tag{10}$$

where

$$g_w(x_i) = \begin{cases} -y_i x_i & 1 - y_i(W^T x_i') \geq 0 \\ 0 & 1 - y_i(W^T x_i') < 0 \end{cases} \tag{11}$$

We can update the linear classification according to Eq. (1) and use Eq. (11) to calculate the gradient.

In this experiment, I not only used traditional gradient descent to train the models, but also use modified gradient descent. Their introduction is in the following.

Nesterov accelerated gradient(NAG) is a method which considers momentum. It can be calculated as following:

$$g_t = \Delta L(W_{t-1} - \gamma v_{t-1}) \tag{12}$$

$$v_t = \gamma v_{t-1} + \eta g_t \tag{13}$$

$$W_t = W_{t-1} - v_t \tag{14}$$

where γ and η are hyperparameters which can be set to be 0.9 and 0.1 respectively. η is treated as the learning rate which is mentioned in Eq. (1).

RMSProp is a method which can solve the problem of AdaGrad that the learning rate tends to be 0. It can be calculated as following:

$$g_t = \Delta L(W_{t-1}) \tag{15}$$

$$G_t = \gamma G_{t-1} + (1-\gamma) g_t^2 \tag{16}$$

$$W_t = W_{t-1} - (\eta / \sqrt{G_t + \epsilon}) * g_t \tag{17}$$

AdaDelta is a special method that we don't need to set the learning rate. It can be calculated as following:

$$g_t = \Delta L(W_{t-1}) \tag{18}$$

$$G_t = \gamma G_{t-1} + (1-\gamma) g_t^2 \tag{19}$$

$$\Delta weight = -(\sqrt{\Delta_{t-1} + \epsilon}) / (\sqrt{G_t + \epsilon}) * g_t \tag{20}$$

$$weight = weight + \Delta weight \tag{21}$$

$$\Delta_t = \gamma \Delta_{t-1} + (1-\gamma) \Delta weight^2 \tag{22}$$

Adaptive estimates of lower-order moments (Adam) can be calculated as:

$$g_t = \Delta L(W_{t-1}) \tag{23}$$

$$m_t = \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t \tag{24}$$

$$v_t = \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot g_t^2 \tag{25}$$

$$m_t' = m_t / (1-\beta_1^t) \tag{26}$$

$$v_t' = v_t / (1-\beta_2^t) \tag{27}$$

$$weight = weight - \eta m_t' / (\sqrt{v_t'} + \epsilon) \tag{28}$$

where t is initialized to be 0 and add 1 at the beginning of each iterations.

After choosing the loss function and calculating their corresponding derivation, I can use different kinds of gradient descent to train these two models. When I calculate the gradient descent in this experiments, I don't use all of data. I randomly choose some of them to calculate the gradient. My experiment will be described in following.

III. Experiments

There will be two experiments which will be introduced. One is training the logistic regression and the other is training the linear classification. Both of them is trained in different kinds of gradient descent.

A. Dataset

I use a9a dataset in LIBSVM Data to train and test this two models. There are 32561 data in training set and 16281 data in test set. Each of them has 123 attributes.

B. Implementation

In this section, I will introduce the logistic regression experiment and linear classification experiment in detail.

There are different kinds of gradient descent in this experiment, the parameters I used is in the following.

a) Traditional Gradient Descent: Learning rate a is set to be 0.1, and I will choose 20% of training data randomly to train at each iterations.

b) NAG: Learning rate η is set to be 0.1, and I will choose 20% of training data randomly to train at each iterations. γ is set to be 0.9.

c) RMSProp: Learning rate η is set to be 0.02and I will choose 20% of training data randomly to train at each iterations. γ is set to be 0.9 and ϵ is set to be 1e-8.

d) AdaDelta: γ is set to be 0.95 and ϵ is set to be 1e-6. I will choose 20% of training data randomly to train at each iterations.

e) Adam: Learning rate η is set to be 0.01. β_1 is set to be 0.999 β_2 is set to be 0.999 and ϵ is set to be 0.001. I will choose 20% of training data randomly to train at each iterations.

The weight matrix in my experiments is initialized randomly. The other parameters which will be update at each iterations is initialized to be 0.

Before training in logistic regression model, I set all the -1 label to 0 label. It can help me apply the formula of logistic regression into this dataset easily. The iteration's number is set to 500. The loss value of test set is described in Fig. 1.

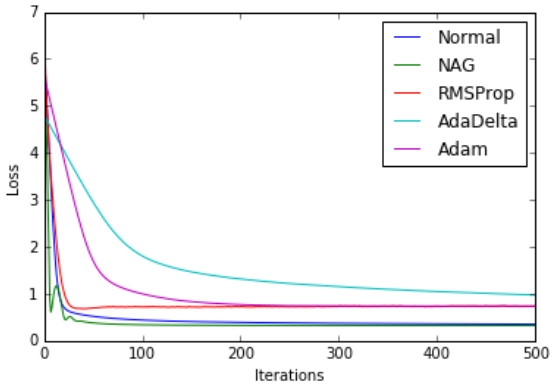

Fig. 1. The loss value of test set in logistic regression.

In the linear classfication experiment, the parameters is set to be the same with the logistic regression. The loss value of test set is described in Fig. 2.

We can find that both of the loss value of training set and test set decrease quickly at the beginning and close to unchange at the end of iterations. It means that the value of loss function is closed to minimal value. However, logistic regression can get a smaller value than linear classification. And logistic regression

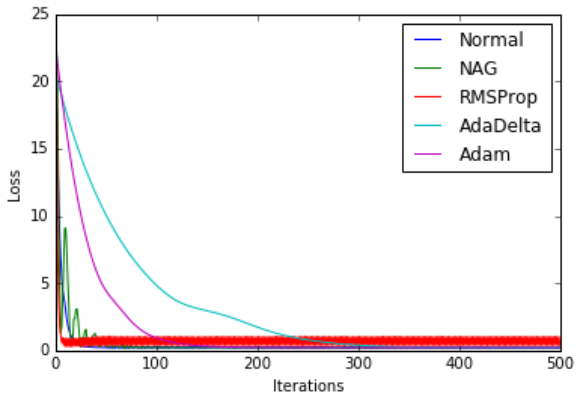

Fig. 2. The loss value of test set in linear classification.

performs more stable since the curve of classification which is update by RMSProp is not smooth. In Fig. 1 and Fig. 2, we can find that Nag and RMSProp always optimize very fast.

After finish training, I set a threshold to classify the data in test set. I set the threshold to be 0.5 in logistic regression model, if the value calculated by model is smaller than 0.5, the corresponding data will be classified into one class (the label of this class is 0). If the value is larger or equal than 0.5, the corresponding data will be classified into another class (the label of this class is 1). The threshold in linear classification is 0, and the label is -1 and 1 respectively. And then I calculate the precision, recall and f1-score which is shown in Table I and Table II.

TABLE I
CLASSIFICATION RESULTS IN LOGISTIC REGRESSION

gradient descent	precision	recall	f1-score	support
Traditional	0.84	0.71	0.73	16281
NAG	0.85	0.68	0.70	16281
RMSProp	0.58	0.76	0.66	16281
AdaDelta	0.58	0.76	0.66	16281
Adam	0.58	0.76	0.66	16281

TABLE II
CLASSIFICATION RESULTS IN LINEAR CLASSIFICATION

gradient descent	precision	recall	f1-score	support
Traditional	0.58	0.76	0.66	16281
NAG	0.58	0.76	0.66	16281
RMSProp	0.82	0.77	0.78	16281
AdaDelta	0.58	0.76	0.66	16281
Adam	0.58	0.76	0.66	16281

From Table I, we can find that the traditional gradient descent and NAG get a better result than the other in logistic regression. I consider that it may be the problems from hyperparameters. I don't adjust a good value for all of the methods. However, we can find that the result in Table II is a bit different in Table I. Some methods perform quite different in two models. I consider that the hyperparameters which is suitable to one model may not be also suitable to other models.

IV. CONCLUSION

In this experiment, I implement gradient descent in logistic regression and linear classification. And I use different kinds of gradient descent to optimize the models. I find that the hyperparameters play important roles in models. Hyperparameters will influence the result directly. And the results of two models also show that the hyperparameters which are suitable to one model may not suitable to other models. How to set a proper hyperparameters to models is an important problem. Different gradient descent methods can lead different curves of the same loss function. It means that each of them may be suitable for different kinds of data.