South China University of Technology

The Experiment Report of *Machine Learning*

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:
Chengneng Jin

Supervisor:
Mingkui Tan

Student ID:
201530611838

Grade:
Undergraduate

December 15, 2017

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

***Abstract*—This experiment intends to use logistic regression and linear classification respectively for binary classification problem, and update model parameters using different optimization methods (NAG, RMSProp, AdaDelta and Adam) to accelerate the convergence speed of gradient descent.**

I. Introduction

Binary classification problem is common in machine learning field, and several methods have been put forward by researchers. I reproduce logistic regression and linear classification for binary classification problem in this experiment. More importantly, I implement four different optimization methods (NAG, RMSProp, AdaDelta and Adam), which contribute to accelerate the convergence speed of gradient descent. These four optimization methods all perform well, and minimize the descent time at least three times.

II. Methods and Theory

A. Logistic Regression

The logistic function $\sigma(t)$ is defined as follows:

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

For binary labels: $y_i = \{0, 1\}$

$$h_w(x) = g(w^T x) = \frac{1}{1+e^{-w^T x}} \tag{2}$$

The loss function is:

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

The derivation of loss function is:

$$\frac{\partial J(w)}{\partial w} = -y\frac{1}{h_w(x)}\frac{\partial h_w(x)}{\partial w} + (1-y)\frac{1}{1-h_w(x)}\frac{\partial h_w(x)}{\partial w} \tag{4}$$

B. Linear Classification

For linear classification, I choose support vector machines (SVMs), which is widely used in binary classification problem. The hinge loss is used for ”maximum-margin” classification, most notably for support vector machines (SVMs). It’s defined as follows:

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

The derivation of hinge loss is:

$$\frac{\partial \ell}{\partial w_i} = \begin{cases} -t \cdot x_i & \text{if } t \cdot y < 1 \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

The loss function of SVM is:

$$J(w) = \frac{1}{2}\|w\|^2 + C\sum_i \max(0, 1 - y_i(w_i^x + b)) \tag{7}$$

C. Optimization Methods

Vanilla SGD update in terms of the parameter update vector is:

$$\begin{aligned} \Delta\theta_t &= -\eta \cdot g_{t,i} \\ \theta_{t+1} &= \theta_t + \Delta\theta_t \end{aligned} \tag{8}$$

1) Momentum: Momentum is a method that helps accelerate SGD in the relevant direction and dampens oscillations.It does this by adding a fraction γ of the update vector of the past time step to the current update vector.

$$\begin{aligned} v_t &= \gamma v_{t-1} + \eta \nabla_\theta J(\theta) \\ \theta &= \theta - v_t \end{aligned} \tag{9}$$

2) Nesterov accelerated gradient(NAG): Nesterov accelerated gradient is a way to give our momentum term a kind of prescience that slowing down before the loss rise again.

$$\begin{aligned} v_t &= \gamma v_{t-1} + \eta \nabla_\theta J(\theta - \gamma v_{t-1}) \\ \theta &= \theta - v_t \end{aligned} \tag{10}$$

3) Adagrad: Adagrad is well-suited for dealing with sparse data, for it adapts the learning rate to the parameters, performing larger updates for infrequent and smaller updates for frequent parameters.

$$\theta_{t+1,i} = \theta_{t,i} - \frac{\eta}{\sqrt{G_{t,ii} + \epsilon}} \cdot g_{t,i} \tag{11}$$

The vectorization implementation of Adagrad is:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{G_t + \epsilon}} \odot g_t \tag{12}$$

4) AdaDelta: Adadelta is an extension of Adagrad that seeks to reduce its aggressive, monotonically decreasing learning rate. Instead of accumulating all past squared gradients, Adadelta restricts the window of accumulated past gradients to some fixed size w. The running average $E[g^2]_t$ at time step t then depends (as a fraction γ similarly to the Momentum term) only on the previous average and the current gradient:

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

5) 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 :

$$\begin{aligned} E[g^2]_t &= 0.9 E[g^2]_{t-1} + 0.1 g_t^2 \\ \theta_{t+1} &= \theta_t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t \end{aligned} \tag{13}$$

6) Adaptive Moment Estimation(Adam): Adam is another method that computes adaptive learning rates for each parameter. Adam keeps an exponentially decaying average of past gradients m_t, similar to momentum.

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \\ \hat{m}_t &= \frac{m_t}{1-\beta_1^t} \\ \hat{v}_t &= \frac{v_t}{1-\beta_2^t} \\ \theta_{t+1} &= \theta_t - \frac{\eta}{\sqrt{\hat{v}_t}+\epsilon}\hat{m}_t \end{aligned} \tag{14}$$

III. Experiments

A. Dataset

The data set used in this experiment is a9a, provided by LIBSVM Data. The original data set has 14 features, among which six are continuous and eight are categorical. In this data set, continuous features are discretized into quantiles, and each quantile is represented by a binary feature. Also, a categorical feature with m categories is converted to m binary features. In conclusion, this data set contains 32,561 records for training and 16,281 records for testing.

B. Implementation

1) Initialization and Parameters: The parameter $maxIteration$ is limited to 300, in order to compare the performance of each optimization methods. Meanwhile, other parameters such as γ in NAG, ϵ in AdaDelta are adjusted respectively for better performance.

2) Results: In order to present the convergence speed of gradient descent, I record the $Iteration\ Time$ when the loss is less than 1 at the fist time and the $Best\ Accuracy$ for each optimization methods of logistic regression and linear classification respectively. The results are shown in Tabel I and Tabel II. The graph of loss descent is shown in Fig. 2 and Fig. 1.

TABLE I
Iteration Time for Optimization Methods

Type	Logistic Regression	Support Vector Machine
Vanilla SGD	57	63
NAG	13	14
AdaDelta	14	15
RMSProp	23	29
Adam	9	12

TABLE II
Best accuracy for Optimization Methods

Type	Logistic Regression	Support Vector Machine
Vanilla SGD	0.7757747	0.7752526
NAG	0.8224563	0.8279229
AdaDelta	0.8153005	0.8069777
RMSProp	0.8134885	0.8080219
Adam	0.7591904	0.7591904

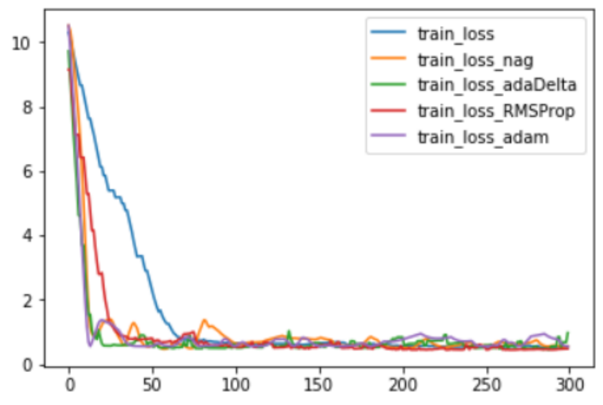

Fig. 1. Loss descent of five different gradient methods for SVM

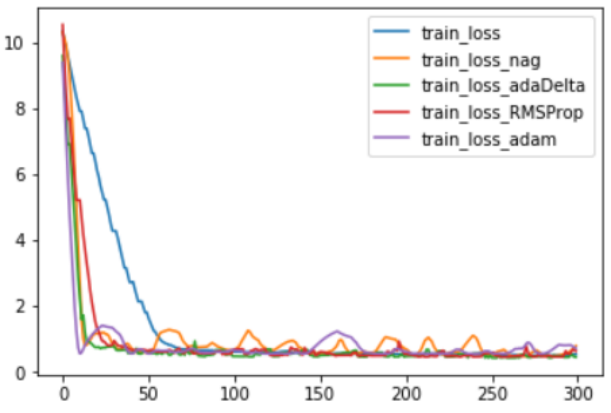

Fig. 2. Loss descent of five different gradient methods for logistic regression

IV. Conclusion

The experiment is even more interesting than the first experiment. It's a little bit difficult to calculate the derivation of loss functions of logistic regression and support vector machine. Luckily there are lost of documents provided by supervisor Mr. Tan and website. Moreover, I've learned about almost six types of optimization methods for gradient descent, which I've never heard before. It's really interesting and attractive to explore the relationship between machine learning, mathematics and physical. I've gained that there are lots of topics worth further study, even as small as gradient descent.