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:
Haokun Li

Supervisor:
Mingkui Tan

Student ID:
201530611883

Grade:
Undergraduate

December 15, 2017

Logistic Regression, Linear Classification and Stochastic Gradient Descent

***Abstract*—This experiment implements Logistic Regression, Linear Classification(SVMs) with Mini-batch Stochastic Gradient Descent, and uses different parameter optimized methods(NAG, RMSProp, AdaDelta and Adam). Then the experiment shows the loss graphs of these optimized methods. What's more, the experiment finds that Linear Classification(SVMs) gets nearly the same loss when transform parameter b into parameter w_0, which simplifies the coding procedure.**

I. Introduction

THIS experiment uses Logistic Regression and Linear Classification(SVMs) to deal with the classification problem so that we can compare and understand the differences and relationships between Logistic regression and linear classification. In order to compare and understand the difference between gradient descent and mini-batch stochastic gradient descent, this experiment use mini-batch gradient descent to update the parameters. Because there are many optimized method to update the parameters, this experiment implements NAG, RMSProp, AdaDelta and Adam method and show their loss graphs. What's more, The experiment uses much larger data set than previous experiments, and makes some simplification on SVMs coding procedure by transforming parameter b into parameter w_0.

II. Methods and Theory

A. Logistic Regression

Logistic Regression was developed by statistician David Cox in 1958. The binary logistic model is used to estimate the probability of a binary response based on one or more predictor (or independent) variables (features). It allows one to say that the presence of a risk factor increases the odds of a given outcome by a specific factor.Assume that the labels are binary:$y_i \in \{0, 1\}$, and the logistic function is:

$$h_W(X) = g(W^TX) = \frac{1}{1+e^{-W^TX}} \tag{1}$$

$h_W(X)$ is interpreted as the probability of the dependent variable equaling a '1' rather than a '0'. So the probability of y_i can be defined as:

$$p = \begin{cases} h_W(X_i) & y_i = 1 \\ 1 - h_W(X_i) & y_i = 0 \end{cases} \tag{2}$$

So, the log-likehood loss function is:

$$J(W) = -\frac{1}{n}\left[\sum_{i=1}^{n} y_i log h_W(X_i) + (1-y_i) log(1-h_W(X_i))\right] \tag{3}$$

The gradient of the loss function is:

$$\frac{\partial J(W)}{\partial W} = \frac{1}{n}\sum_{i=1}^{n}(h_W(X_i) - y_i)X_i \tag{4}$$

B. Linear Classification(SVMs)

In machine learning, support vector machines (SVMs, also support vector networks) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. Learning the SVMs can be formulated as an optimization:

$$\begin{aligned} &\min_{W,b} \frac{||W||}{2} \\ &s.t. \quad y_i(W^TX_i + b) \geq 1, \quad i = 1, 2......n \end{aligned} \tag{5}$$

Hinge loss:

$$\xi_i = max(0, 1 - y_i(W^TX_i + b)) \tag{6}$$

When we use hinge loss, we can rewrite the optimization problem as follows:

$$\begin{aligned} \min_{w,b} \quad L(W,b) &= \frac{||W||^2}{2} + \frac{C}{n}\sum_{i=1}^{n} max(0, 1 - y_i(W^TX_i + b)) \qquad (7) \\ &= \frac{1}{n}\sum_{i=1}^{n}\left(\frac{||W||^2}{2} + Cmax(0, 1 - y_i(W^TX_i + b))\right) \\ &= \frac{1}{n}\sum_{i=1}^{n} L_i(W,b) \end{aligned}$$

The gradient of the loss function is:

$$\begin{aligned} \frac{\partial \xi_i}{\partial W} &= \begin{cases} -y_iX_i & 1 - y_i(W^TX_i + b) \geq 0 \\ 0 & 1 - y_i(W^TX_i + b) < 0 \end{cases} \\ \frac{\partial \xi_i}{\partial b} &= \begin{cases} -y_i & 1 - y_i(W^TX_i + b) \geq 0 \\ 0 & 1 - y_i(W^TX_i + b) < 0 \end{cases} \end{aligned} \tag{8}$$

$$\begin{aligned} \nabla_W L_i(W,b) &= W + Cg_W(X_i) \\ \nabla_b L_i(W,b) &= Cg_b(X_i) \end{aligned} \tag{9}$$

C. GD

Gradient descent is a first-order iterative optimization algorithm for finding the minimum of a function. To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient (or of the approximate gradient) of the function at the current point. Gradient descent 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, $-\nabla F(a)$. Therefore, gradient descent update a as follow:

$$a_{n+1} = a_n - \gamma \nabla F(a_n) \tag{10}$$

γ is learn-rate.

D. NAG

In the machine learning community, Nesterovs Accelerated Gradient (NAG) is largely viewed as somewhat mysterious and explained as performing a lookahead gradient evaluation and then performing a correction. NAG is given by:

$$\begin{aligned} \upsilon_{t+1} &= \mu_t \nu_t - \alpha_t J'(\theta_t + \mu_t \upsilon_t) \\ \theta_{t+1} &= \theta_t + \upsilon_{t+1} \end{aligned} \tag{11}$$

μ is forgetting factor and α is learn rate.

E. RMSProp

RMSProp (for Root Mean Square Propagation) is a method in which the learning rate is adapted for each of the parameters. The idea is to divide the learning rate for a weight by a running average of the magnitudes of recent gradients for that weight. So, first the running average is calculated in terms of means square:

$$\upsilon(w,t) = \gamma \upsilon(w,t-1) + (1-\gamma)(\nabla Q_i(w))^2 \tag{12}$$

γ is the forgetting factor. And the parameters are updated as:

$$w = w - \frac{\eta}{\sqrt{\upsilon(w,t)}} \nabla Q_i(w) \tag{13}$$

η is learn rate.

F. AdaDelta

AdaDelta(An Adaptive Learning Rate Method) dynamically adapts over time using only first order information and has minimal computational overhead beyond vanilla stochastic gradient descent. The method requires no manual tuning of a learning rate and appears robust to noisy gradient information, different model architecture choices, various data modalities and selection of hyperparameters.

$$\begin{aligned} E[g^2]_t &= \gamma E[g^2]_{t-1} + (1-\gamma) g_t^2 \\ RMS[g]_t &= \sqrt{E[g^2]_t + \epsilon} \\ \Delta\theta_t &= -\frac{\eta}{RMS[g]_t} g_t \\ E[\Delta\theta_t^2]_t &= \gamma E[\Delta\theta_t^2]_{t-1} + (1-\gamma)\Delta\theta_t^2 \\ RMS[\Delta\theta]_t &= \sqrt{E[\Delta\theta_t^2]_t + \epsilon} \\ \Delta\theta_t &= -\frac{RMS[\Delta\theta]_{t-1}}{RMS[g]_t} g_t \\ \theta_{t+1} &= \theta_t + \Delta\theta_t \end{aligned} \tag{14}$$

η is learn rate, γ is the forgetting factor, and ϵ is a small number used to prevent division by 0.

G. Adam

Adam(short for Adaptive Moment Estimation) is an update to the RMSProp optimizer. In this optimization algorithm, running averages of both the gradients and the second moments of the gradients are used. Given parameters w_t and a loss function L_t, where t indexes the current training iteration (indexed at 1, Adam's parameter update is given by:

$$\begin{aligned} m_{t+1} &= \beta_1 m_t + (1-\beta_1)\nabla_w L_t \\ \upsilon_{t+1} &= \beta_2 \upsilon_t + (1-\beta_2)(\nabla_w L_t)^2 \\ \widehat{m}_w &= \frac{m_{t+1}}{1-\beta_1^t} \\ \widehat{\upsilon}_w &= \frac{\upsilon_{t+1}}{1-\beta_2^t} \\ w_{t+1} &= w_t - \eta \frac{\widehat{m}_w}{\sqrt{\widehat{\upsilon}_w} + \epsilon} \end{aligned} \tag{15}$$

ϵ is a small number used to prevent division by 0, and β_1 and β_2 are the forgetting factors for gradients and secong moments of gradients, and η is learn rate.

III. Experiments

A. Dataset

Experiment uses a9a of LIBSVM Data, including 32561/16281(validation) samples and each sample has 123/123 (testing) features. You can download the data set from this url:
https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html#a9a

B. Implementation

All detailed implementation in your experiment: initialization, process, results, all kinds of parameters. In a word, describe clearly What you do and how you do.

1) *Logistic Regression:*

1) Load the training set and validation set.
2) Initalize logistic regression model parameters w between -1 and 1 randomly.
3) Select the loss function and calculate its derivation, which can be found in the previous Methods and Theory section.
4) Calculate gradient G toward loss function from partial samples.
5) Update model parameters using different optimized method.The parameters of different optimized methods are:

TABLE I
PARAMETERS OF GD

Learning rate	0.1

TABLE II
PARAMETERS OF NAG

Learning rate	0.1
Forgetting factor	0.9

TABLE III
PARAMETERS OF RMSPROP

Learning rate	0.1
Forgetting factor	0.9
Mini number	10^{-8}

TABLE IV
PARAMETERS OF ADADELTA

Learning rate	0.1
Forgetting factor	0.9
Mini number	10^{-8}

TABLE V
PARAMETERS OF ADAM

Learning rate	0.1
Forgetting factor1	0.9
Forgetting factor2	0.999
Mini number	10^{-8}

6) Predict under validation set and get the loss of models.
7) Echo step 4 to 6 for several times with these parameter:

TABLE VI
PARAMETERS OF ECHO

Threshold	0.5
Batch size	1000
Echo times	200

8) drawing graph of Log loss result of different optimized methods in Logistic regression with the number of iterations(Fig.1.).
According to the Fig.1, all of the loss decrease slower and slower, and AdaDelta, RmsProp and NAG converge closely. All of the optimized method work much better than original gradient descent when using the same learn rate.

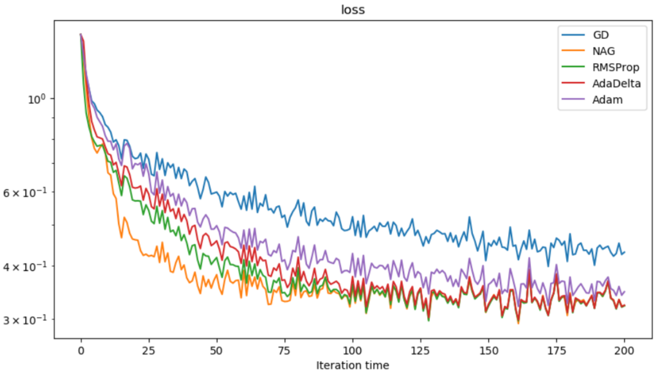

Fig. 1. Log loss result of different optimized methods in Logistic regression.

2) *Linear Classification(SVMs):*

1) Load the training set and validation set.
2) Initalize logistic regression model parameters w and b between -1 and 1 randomly.
3) Select the loss function and calculate its derivation, which can be found in the previous Methods and Theory section.
4) Calculate gradient G toward loss function from partial samples.
5) Update model parameters using different optimized method.The parameters of different optimized methods are:

TABLE VII
PARAMETERS OF GD

Learning rate	0.1

TABLE VIII
PARAMETERS OF NAG

Learning rate	0.1
Forgetting factor	0.9

TABLE IX
PARAMETERS OF RMSPROP

Learning rate	0.1
Forgetting factor	0.9
Mini number	10^{-8}

TABLE X
PARAMETERS OF ADADELTA

Learning rate	0.1
Forgetting factor	0.9
Mini number	10^{-8}

TABLE XI
PARAMETERS OF ADAM

Learning rate	0.1
Forgetting factor1	0.9
Forgetting factor2	0.999
Mini number	10^{-8}

6) Predict under validation set and get the loss of models.
7) Echo step 4 to 6 for several times with these parameter:

TABLE XII
PARAMETERS OF ECHO

Threshold	0
Batch size	500
Echo times	200

8) drawing graph of Log loss result of different optimized methods in Logistic regression with the number of iterations(Fig.2.).

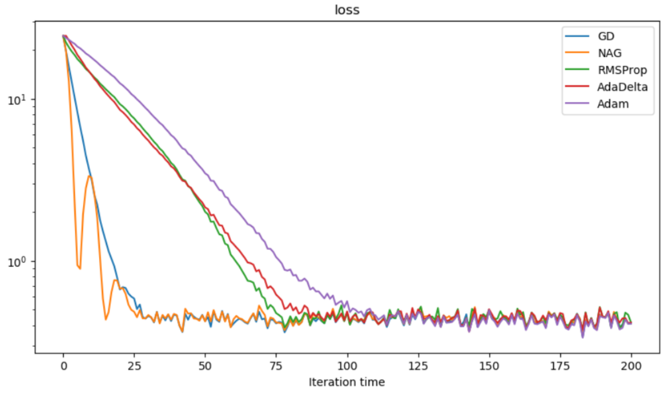

Fig. 2. Log loss result of different optimized methods in Linear classification.

According to the Fig.2, NAG shocks a lot but decreases fastest. NAG and GD converg early. But AdaDelta, RmsProp and Adam converge steadily. All of the optimized method work well when using the same learn rate.

3) Addition work:optimize the coding procedure of linear classification: This experiment tries to transform parameter b into one element in parameter w, which can simplify the coding procedure. Even though it's kind of different from the theory loss and derivation formulas, the method gets nearly the same loss graph as the original method when using the same data set and optimized parameters(Fig.3.). Therefore, the loss function and the gradient of the loss function become:

$$\begin{aligned}\min_{w} \quad L(W,b) &= \frac{||W||^2}{2} + \frac{C}{n}\sum_{i=1}^{n} max(0, 1 - y_i W^T X_i) \qquad (16)\\ &= \frac{1}{n}\sum_{i=1}^{n}\left(\frac{||W||^2}{2} + Cmax(0, 1 - y_i W^T X_i)\right)\\ &= \frac{1}{n}\sum_{i=1}^{n} L_i(W)\end{aligned}$$

$$\frac{\partial \xi_i}{\partial W} = \begin{cases} -y_i X_i & 1 - y_i W^T X_i \geq 0 \\ 0 & 1 - y_i W^T X_i < 0 \end{cases} \qquad (17)$$

$$\nabla_W L_i(W) = W + C g_W(X_i) \qquad (18)$$

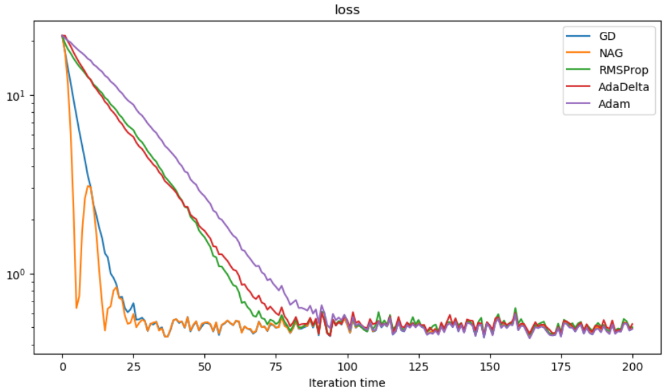

Fig. 3. Log loss result of different optimized methods in Linear classification.

IV. CONCLUSION

This experiment implement Logistic Regression and Linear Classification(SVMs) and mini-batch stochastic gradient descent with different optimized methods NAG, RMSProp, AdaDelta and Adam. When dealing with huge data set, stochastic gradient descent is a effective way to update parameters. NAG method can decrease loss fastest, but may shock a lot. Adam, RMSProp and AdaDelta can converge steadily and have nearly no shock. However, original stochastic gradient descent is always a not bad but simply method to update parameters. What's more, transform parameter b into one element in parameter w can simplify the coding of SVMs and nearly get the same result.