The Experiment Report of Deep Learning

SCHOOL: SCHOOL OF SOFTWARE ENGINEERING

SUBJECT: SOFTWARE ENGINEERING

Author:
Yuxuan Gao

Supervisor:
Mingkui Tan

Student ID：
201720145075

Grade:
Graduate

December 15 2017

Logistic Regression, Linear Classification and Stochastic Gradient Descent

Abstract—In this paper, we use stochastic gradient descent to implement logistic regression and linear classification based on support vector machine, implementing NAG, RMSProp, AdaDelta and Adam optimization algorithms. We also discuss the experimental results.

I. INTRODUCTION

Logistic regression and support vector machines can be used as classifications. The usual practice is to determine a loss function, then take the derivative of the parameter, and then use the gradient descent method to continuously approximate the optimal solution. However, the gradient descent method traverses the entire dataset every time, which results in poor performance, so we use stochastic gradient descent to perform gradient updates.

Stochastic gradient descent is not the end point, and there are many optimization methods that can be used to improve the effect of gradient descent, such as NAG, RMSProp, AdaDelta and Adam. Each method needs to be verified by experiments. In this paper, we use stochastic gradient descent to implement logistic regression and linear classification based on support vector machine, implementing NAG, RMSProp, AdaDelta and Adam optimization algorithms. Results shows that, in some cases, the optimization algorithm does not bring much improvement, and only the stochastic gradient descent can achieve better training effect. But by tuning the parameters of each optimization algorithm, the various optimization algorithms show different effects, which indicates that the parameter selection is an important part in the experiment of Machine Learning.

II. METHODS AND THEORY

The whole experiment is divided into two steps:1) Use the stochastic gradient descent algorithm to rewrite the previous code. 2) Implement NAG, RMSProp, AdaDelta and Adam optimization algorithms respectively. 3) Combine these optimization algorithms to make a comprehensive comparison.

Next, let's talk about the theory used in the experiment.

A. Loss Function And Derivation

a. Linear Regression

1) loss function

$$L = \frac{1}{N} (Y-XW)^T (Y-XW)$$

X is the matrix of dataset; Y is the matrix of values to be predi-cted; W is the matrix of weights; N is the number of dataset.

2) gradient/ derivative

$$\frac{\partial L}{\partial W} = 2X^T (XW-Y)$$

b. Linear Classification

1) loss function

$$L = \frac{1}{N}\sum_i \sum_{j \neq y_i} [max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + \Delta)] + \lambda \sum_k \sum_l W_{k,l}^2$$

2) gradient/ derivative

$$\nabla_{w_{y_i}} L_i = -\left(\sum_{j \neq y_i} \mathbb{1}\left(w_j^T x_i - w_{y_i}^T x_i + \Delta > 0\right) \right) x_i + 2\lambda w_{y_i}$$

$$\nabla_{w_j} L_i = \mathbb{1}(w_j^T x_i - w_{y_i}^T x_i + \Delta > 0) x_i + 2\lambda w_j$$

B. Mini-batch Stochastic Gradient Descent

In a generalized case, at each iteration a mini-batch B that consists of indices for training data instances may be sampled at uniform with replacement[1]. Similarly, we can use

$$\nabla f_{\mathcal{B}}(\mathbf{x}) = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \nabla f_i(\mathbf{x})$$

to update $\mathbf{x}$ as

$$\mathbf{x} := \mathbf{x} - \eta \nabla f_{\mathcal{B}}(\mathbf{x})$$

where $|\mathcal{B}|$ denotes the cardinality of the mini-batch and the positive scalar η is the learning rate or step size. Likewise, the mini-batch stochastic gradient $\nabla f_{\mathcal{B}}(\mathbf{x})$ is an unbiased estimator for the gradient $\nabla f(\mathbf{x})$

C. Optimization Algorithms

a. Nesterov accelerated gradient

Nesterov accelerated gradient (NAG) [2] is a way to give our momentum term this kind of prescience. we will use our momentum term γv_{t-1} to move the parameters θ. Computing $\theta - \gamma v_{t-1}$ thus gives us an approximation of the next position of the parameters：

$$v_t = \gamma v_{t-1} + \eta \nabla_\theta J(\theta - \gamma v_{t-1})$$

$$\theta = \theta - v_t$$

b. Adadelta

Adadelta is an extension of Adagrad that seeks to reduce its aggressive[3], monotonically decreasing learning rate. Instead of accumulating all past squared gradients, Adadelta restricts the window of accumulated past gradients to some fixed size w

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

c. RMSprop

RMSprop in fact is identical to the first update vector of Adadelta that we derived above:

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

d. Adam

Adaptive Moment Estimation (Adam) is another method that computes adaptive learning rates for each parameter[4]

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

III. Experiment

A. Dataset

The experiment use a9a of LIBSVM data, 32561 for training and 16281 for testing, it has 123 attributes without missing attribute values and data is sparse. There are 2 classes to be classified

B. Implementation

a. Linear Regression

1) Use load_svmlight_file function to load the experiment data
2) Use train_test_split function to devide dataset，33% for validation and others for training. As for what I'm going to do is classifying, so I need to convert the class label of -1 and 1 to 0 and 1, and I'm using myself function np.array([int(y_item/2 + 0.5) for y_item in ylables]) to do this.
3) Initialize linear model parameters into zero. W = np.zeros((X.shape[1],1))
4) Choose loss function and compute derivation L.
5) Use mini-batch stochastic gradient descent and different optimization algorithms NAG, RMSProp, AdaDelta and Adam, record the change in loss.
6) Use matplotlib to draw the loss graphs of different optimization algorithm with the number of iterations.

For Mini-Batch SGD, we change η = 0.1, 0.01, 0.001 the The results are as follows:

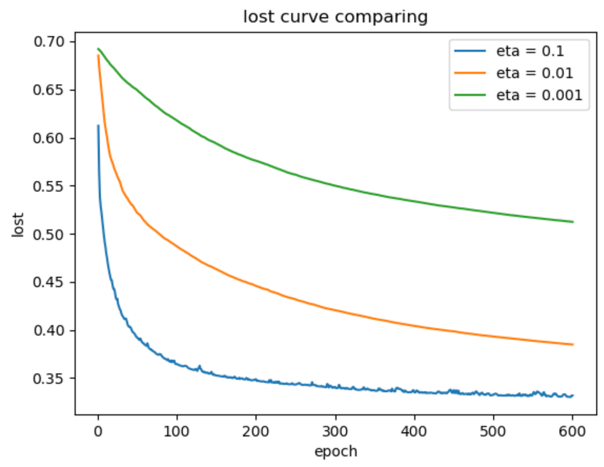

Fig. 1. Mini-Batch SGD parameters comparing

For NAG, we change η = 0.1, 0.01, 0.001 The results are as follows:

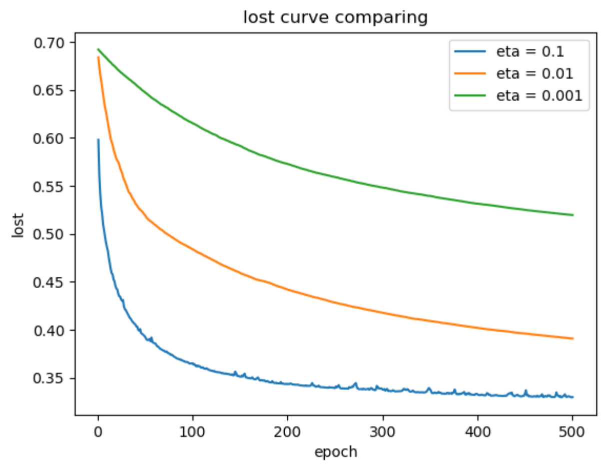

Fig. 2. NAG parameters comparing

For RMSprop, we change η = 0.1, 0.01, 0.001 the results are as follows:

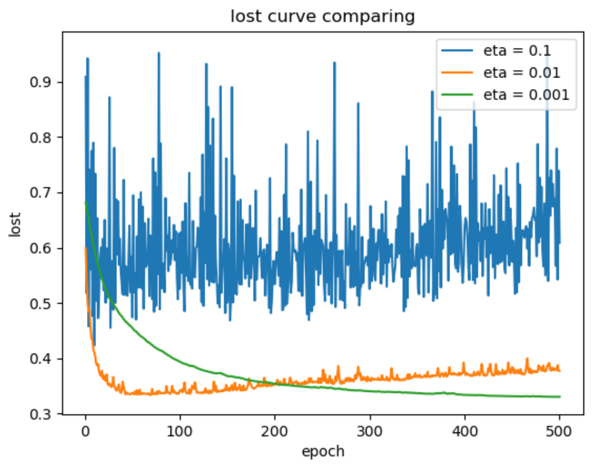

Fig. 3. RMSprop parameters comparing

For Adam, we change η = 0.1, 0.01, 0.001 the results are as follows:

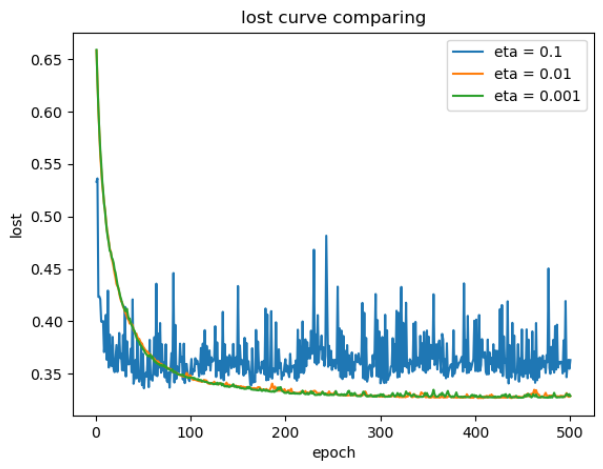

Fig. 4. Adam parameters comparing

Select a suitable parameter for each algorithm and compare them together. And there are the parameter used:
\

TABLE I
LINEAR REGRESSION PARAMETERS

optimization algorithms	parameters	
Mini-Batch SGD	η = 0.1	batch_size=64 epoch=500
NAG	η = 0.1	
Adadelta	γ = 0.95	
RMSprop	η = 0.001 γ = 0.9	
Adam	β = 0.9 γ = 0.999 η = 0.001	

The corresponding lost figure:

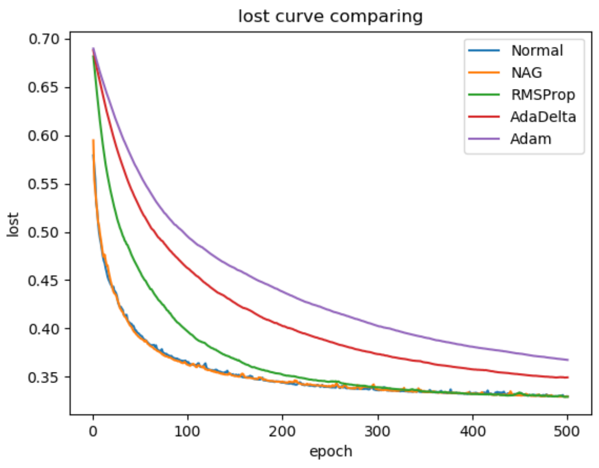

Fig. 5. optimization algorithms comparing

b. Linear Classification

1) Use load_svmlight_file function to load the experiment data
2) Use train_test_split function to devide dataset，33% for validation and others for training. As for what I'm going to do is classifying, so I need to convert the class label of -1 and 1 to 0 and 1
3) 3) Initialize linear model parameters into normal distribution
4) Choose loss function and compute derivation L.
5) Use mini-batch stochastic gradient descent and different optimization algorithms NAG, RMSProp, AdaDelta and Adam, record the change in loss.
6) Use matplotlib to draw the loss graphs of different optimization algorithm with the number of iterations.

For Mini-Batch SGD, we change η = 0.1, 0.01, 0.001 The results are as follows:

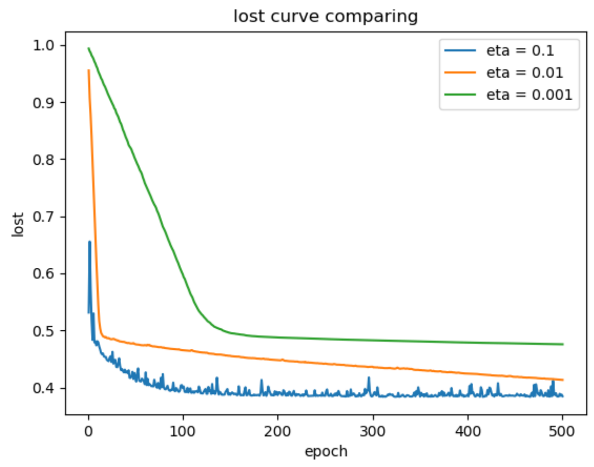

Fig. 6. Mini-Batch SGD parameters comparing

For NAG, we change η = 0.1, 0.01, 0.001 The results are as follows:

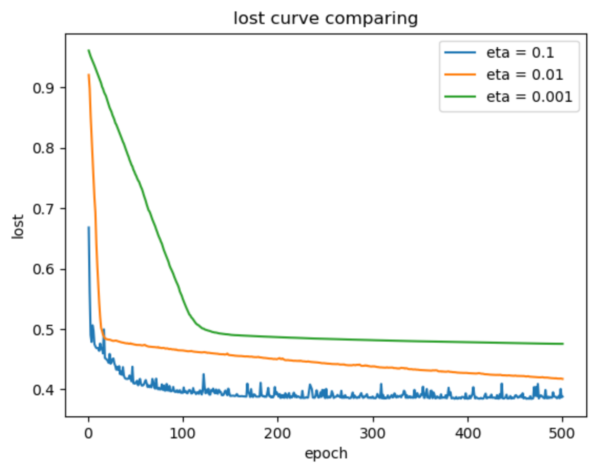

Fig. 7. NAG parameters comparing

For RMSprop, we change η = 0.1, 0.01, 0.001 the results are as follows:

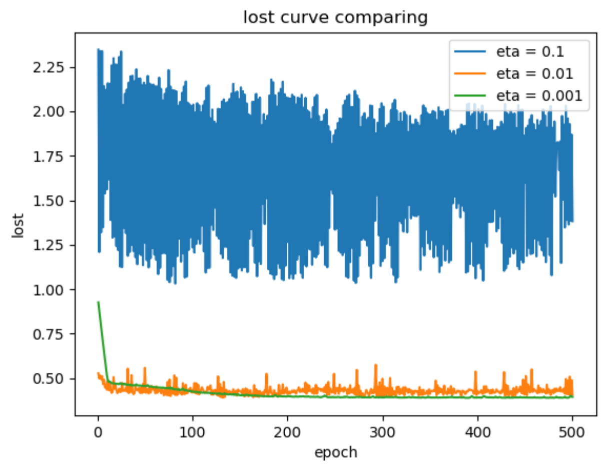

Fig. 8. RMSprop parameters comparing

For Adam, we change η = 0.1, 0.01, 0.001 the results are as follows:

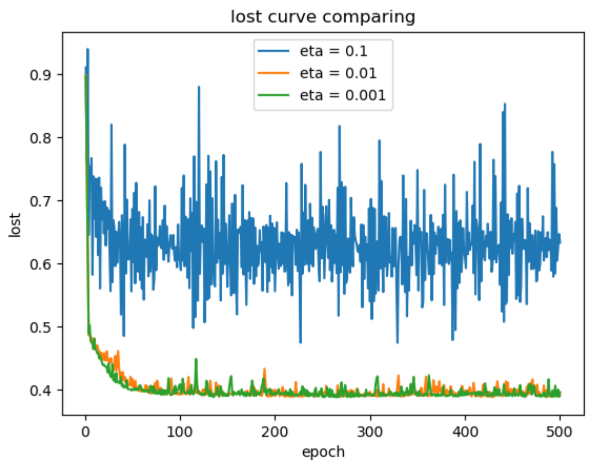

Fig. 9. Adam parameters comparing

Select a suitable parameter for each algorithm and compare them together. And there are the parameter used:

TABLE 2
LINEAR CLASSIFICATION PARAMETERS

optimization algorithms	parameters	
Mini-Batch SGD	η = 0.01	batch_size=64 epoch=500
NAG	η = 0.01	
Adadelta	γ = 0.95	
RMSprop	η = 0.001 γ = 0.9	
Adam	β = 0.9 γ = 0.999 η = 0.001	

The corresponding lost figure:

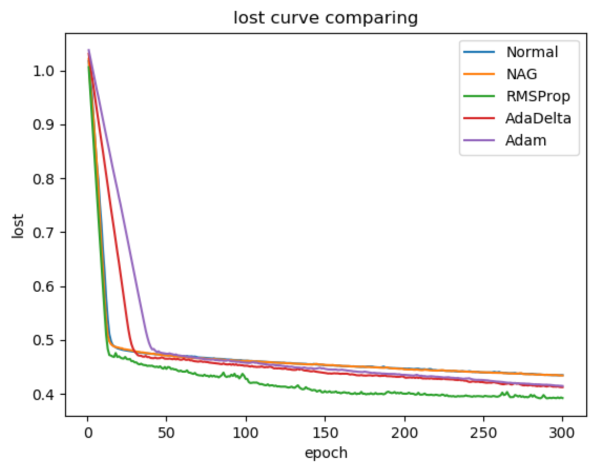

Fig. 10. optimization algorithms comparing

IV. CONCLUSION

In terms of linear Regression, we can see when eta = 0.1, the loss is lower than eta = 0.01 and eta =0.001.However, the loss curve is fluctuant, it seems that small eta brings smoothness but slow convergence. From the comparison of different optimization algorithm, we can see that different parameters bring different effects, the appropriate parameters maybe only for a data set. In this experiment, results shows that RMSprop and NAG perform better, other optimization methods converge too slowly. The interesting thing is, without optimization method, pure SGD can also do well.

As for linear classification, it's almost the same. It is worth noting that different parameter values have great influence on experimental effect, we need to pay attention to the choice of parameters and constantly tune them. From the comparison of different optimization algorithm, we can also see that different parameters bring different effects, RMSprop convergence fastest, but its curve isn't smooth.

From these results, we can realize the importance of parameter tuning.

[1] Sebastian Ruder. An overview of gradient descent optimization algorithms [EB/OL]. http://ruder.io/optimizing-gradient-descent/index.html
[2] mxnet manual. chapter06_optimization [EB/OL].http://gluon.mxnet.io/chapter06_optimization/optimization-intro.html
[3] slinuxer. SGD algorithm to compare [EB/OL]. https://blog.slinuxer.com/2016/09/sgd-comparison
[4] Kingma D P, Ba J. Adam: A Method for Stochastic Optimization[J]. Computer Science, 2014.