South China University of Technology

Experiment Report of Deep Learning

School: School of Software Engineering

Subject: Deep Learning

Author:
Karim Syed Mohammed Walid(王力)

Supervisor:
Mingkui Tan

Student ID: 201722800144

Grade: Graduate

December 23, 2017

Acknowledgment

Foremost, I would like to express my sincere gratitude to my teacher **Mingkui Tan** for the continuous support of my Deep Learning study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of my study and writing of this report. I could not have imagined having a better teacher and mentor for this course.

Besides my teacher, I would like to thank the rest of my classmates, and specially Chinese friends **Xie Zheng, Chen Yaofo, Chen Peihao** for their encouragement, insightful comments, and hard questions.

Finally, I must express my very profound gratitude to Almighty and family for proving me with unfailing support and continuous encouragements throughout months of study and through the process of researching and writing this report. This accomplishment would not have been possible without them. Thank you.

Syed Mohammed Walid Karim (王力)

#201722800144

Logistic Regression, Linear Classification and Stochastic Gradient Descent

Abstract

In this experiment it has two parts, one is logistic regression, the other is linear classification. Both models are updated with four different gradient descent methods and tested in the a9a in LIBSVM Data.Different optimized methods used to update the parameters like Nesterov Accelerated Gradient (NAG), RMSProp, AdaDelta and Adam.

Introduction

The experiment consists of two parts. The first one is logistic regression. The model parameters are updated by four different gradient descent methods. The loss can drop into low level. The

second one is a linear classification. Similarly, the model parameters are updated through the same four gradient descent methods.

Gradient descent is a way to minimize an objective function $J(\theta)$ parameterized by a model's parameters $\theta \in R^d$ by updating the parameters in the opposite direction of the gradient of the objective function $\nabla_\theta J(\theta)$ with respect to the parameters. The learning rate η determines the size of the steps to reach local minimum. Linear classification is a decision based classification which is based on the value of a linear combination of the characteristics. The general formula of the linear classifier is

$f(x) = w^T x + b$

where w is the weight vector and b are the bias. In 2D the discriminant is line and in 3D the discriminant is a plane and mD it is a hyperplane.

Simple linear regression describes the liner relationship between a predictor variable, plotted on x-axis and a response variable plotted on the y-axis. The goal to learn a hypothesis or model is $f: X \rightarrow Y$.

Model function of linear regression,

$f(x; w_0, w) = w^T x + w_0$

Stochastic gradient descent also known as incremental gradient descent, is a stochastic approximation of the gradient descent optimization and iterative method for minimizing an objective function that is written as a sum of differentiable functions. Stochastic gradient descent is easy to calculate and can be used for more complex and error surfaces. SGD in contrast performs a parameter update for each training example $x^{(i)}$ and label $y^{(i)}$

$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i)}; y^{(i)})$

with the different fixed learning rate, the same Gradient descent method can have different effect on data, with larger learning rate, the train loss will have much fluctuation, and with smaller learning rate, the curve will become convex, besides that, Adadelta won't use learning rate which it can adjust it by iteself. Actually, not only the learning rate, other hyper-parameters will have different

influence on different method

Methods and Theory

Adadelta:

Adadelta monotonically decreasing learning rate instead of accumulating all past squared gradients. Adadelta restricts the window of accumulated past gradients to some fixed size w.

$E[\Delta\theta^2]_t = \gamma E[\Delta\theta^2]_{t-1} + (1-\gamma)\Delta\theta^2_t$

Momentum:

SGD has trouble navigating ravines, i.e. areas where the surface curves much more steeply in one dimension that in another. Momentum is a method which helps accelerate SGD in the relevant direction.

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

$\theta = \theta - v_t$

The fraction γ used to update vector of the past time step to the current update vector. The momentum term increases for dimensions whose gradients point in the same directions and reduces updates for dimensions whose gradients change directions.

Nesterov Accelerated Gradient:

NAG is a way to give momentum term γv_{t-1} to move the parameters θ. Computing $\theta - \gamma v_{t-1}$ which gives an approximation of the next position of the parameters are going to be. By calculating the gradient not with respect to current parameters θ but with respect to the approximate future position of parameters:

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

$\theta = \theta - v_t$

RMSProp:

RMSProp is a method in which the learning rate is adapted for each of the parameters. The main idea is to divide the learning rate for a weight by a running average of the magnitudes of recent gradients for that weights. So the first running average is calculated in terms of means square,

$v(w,t) := \gamma v(w,t\text{-}1) + (1\text{-}\gamma)(\nabla J(Q_i(w))^2$

where, γ is the forgetting factor and the parameter are updated as

$$w := w - \frac{\eta}{\sqrt{v(w,t)}} \nabla Q_i(w)$$

Dataset

Both experiments use a9a of LIBSVM Data, including 32561/16281(testing) samples and each sample has 123/123 (testing) features

Adam:

Adaptive Moment Estimation (Adam) is a method that computes adaptive learning rates for each parameter. In addition to storing an exponentially decaying average of past squared gradients v_t like Adadelta and RMSProp, Adam also keeps an exponentially decaying average of past gradients $m_{t,}$ like momentum:

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

$v_t = \beta_2 v_{t-1} + (1-\beta_2) g^2_t$

m_t and v_t are estimate of the first moment and the second moment of the gradients respectively. The method of Adam follows:

$$w^{(t+1)} \leftarrow w^{(t)} - \eta \frac{\hat{m}_w}{\sqrt{\hat{v}_w} + \epsilon}$$

Where ϵ is a small number used to prevent division by 0.

About the linear classification, after reading the dataset, we also need to initialize the model, all the parameters used in the experiment are the same as linear regression. And the change of loss of Gradient Decent method NAG with learning rate which equals to 0.1, Gradient Decent method RMSProp, Gradient Decent method AdaDelta, Gradient Decent method Adam.The change of loss of Gradient Decent method NAG with different learning rate, and we want figure out how different hyper-parameter learning change will change the train and valid loss.The change of accuracy with different learning and gradient decent method.

Experiment

I Calculated gradient toward loss function from partial samples and Update model parameters using different optimized methods(NAG，RMSProp，AdaDelta and Adam). After-that select the appropriate threshold, mark the sample whose predict scores greater than the threshold as positive, on the contrary as negative. Predict under validation set and get the different optimized method loss ，， and repeat step those steps for several times, and drawing graph of different methods and with the number of iterations. In short the experiment divided into two parts- 1. Logistic regression and stochastic gradient descent 2. Linear classification and stochastic gradient descent. We show point out that the NGA won't be influenced by learning rate, which means it can adjust learning rate by itself also initialized randomly in range(0, 1). And then, use the model described to update the W until the loss

function has converged. The same as logistic regression, the curves of the four different gradient descent methods in both two figures can have different effective.

Logistic regression and stochastic gradient descent

Firstly, using request_get() function downloaded the dataset and loaded the experiment data using load_svmlight_file function in sklearn library. After that initialized logistic regression model parameters using random numbers distribution and defined loss function and calculated its derivation. To obtain loss function calculated gradient G. Using different optimized methods like NAG, RMSProp, Adadelta and Adam updated the model parameters and selected the threshold which predicts scores greater than the threshold as positive and on the contrary as negative. After repeating several times with number of iterations the following graph generated.

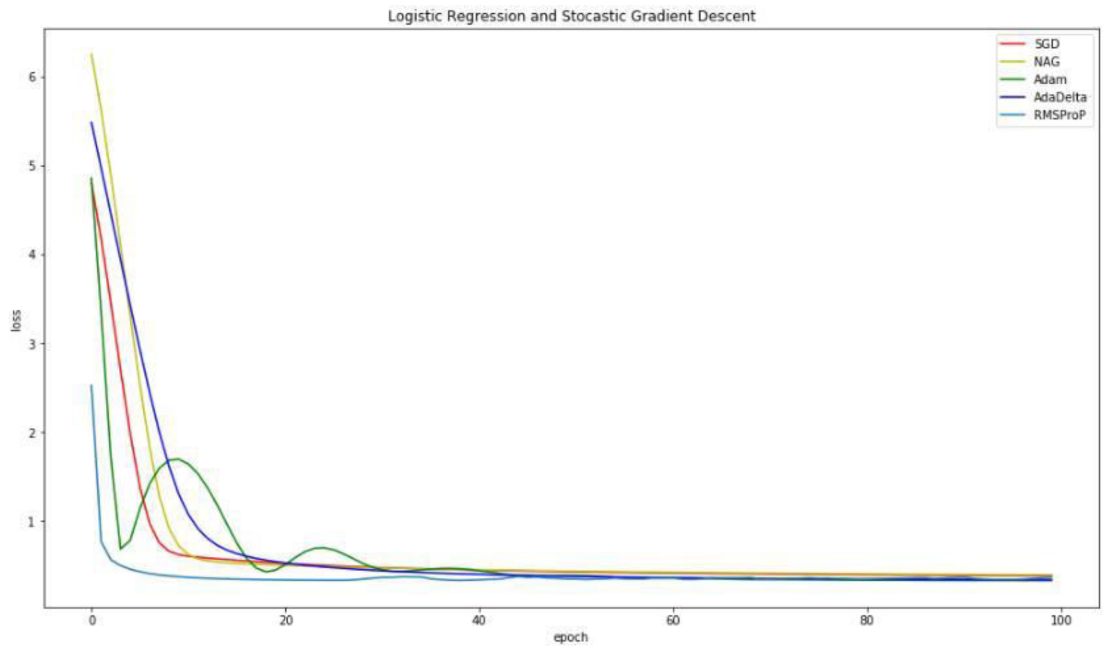

Fig:01

Linear classification and stochastic gradient descent

Firstly, using request_get() function downloaded the dataset and loaded the experiment data using load_svmlight_file function in sklearn library. After that initialized SVM model parameters using random numbers distribution and defined loss function and calculated its derivation. To obtain loss function calculated gradient G. Using different optimized methods like NAG, RMSProp, Adadelta and Adam updated the model parameters and selected the threshold which predicts scores greater than the threshold as positive and on the contrary as negative. After repeating several times with number of iterations the following graph generated.

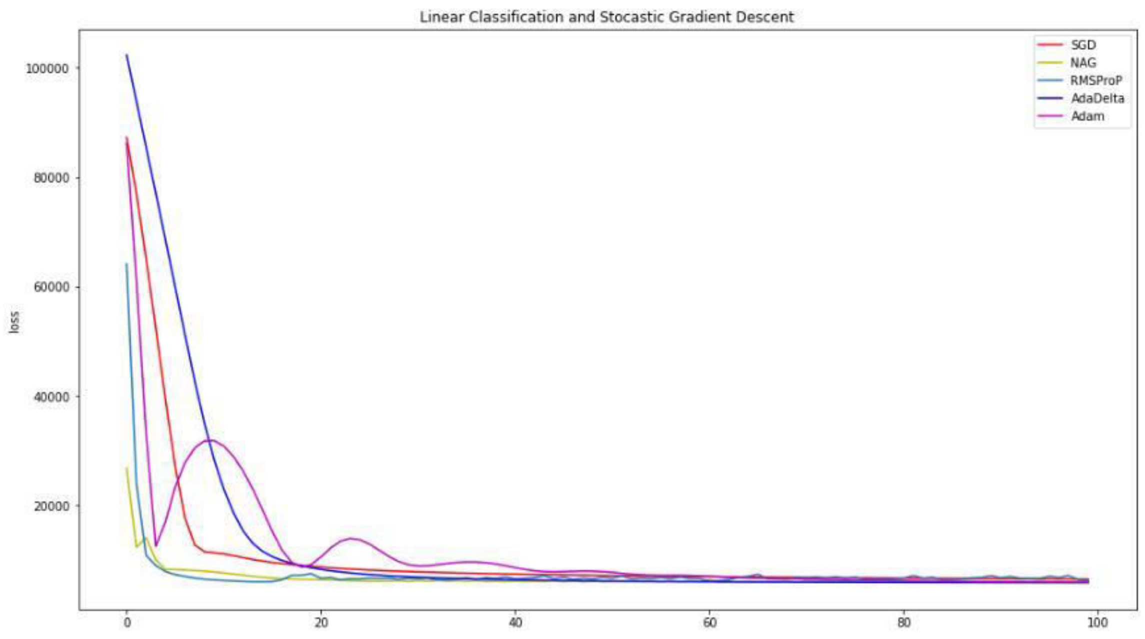

Fig:02

Conclusion

This report shows that the train curve and the test curve are very similar. Though all method can have different learning rate in each iteration, by with the different fixed learning rate, the same Gradient descent method can have different effect on data, with larger learning rate, the train loss will have much fluctuation, and with smaller learning rate, the curve will become convex, besides that, Adadelta won't use learning rate which it can adjust it by itself. Actually, not only the learning rate, other hyper-parameters will have different influence on different method. Moreover, unlike SDCA or SAG, this method does not require the storage of gradients, and   thus is more easily applicable to complex problems such as structured prediction or neural network learning.