Performance Analysis of Optimization Algorithms

Mouna Lamine[1],Sang-Chul Kim[2]

Kookmin University

Abstract

Optimization in Machine learning is the process of iterative improvement in a machine learning model's accuracy, reducing the error level. the most frequently used optimization algorithms in machine learning are the first-order algorithms based on gradient descent. Rosenbrock function is a common test problem for optimization algorithms. Therefore, this paper will present a comparative performance analysis of different optimization algorithms using the Rosenbrock function.

Keywords: Machine Learning, Optimization Methods, gradient decent, stochastic gradient descent, momentum ,Nesterov-accelerated gradient, AdaGrad, AdaDelta, RMSProp, Adam, Rosenbrock function

1 Introduction

Machine learning is widely used in all industries and has a wide range of applications, in particular, that involves collecting, analyzing and responding to large data sets. Optimization is considered a core function as it runs through every step of machine learning to enhance the accuracy of predictions and classifications, and minimizes errors. This article aims to provide the reader with intuitions with regard to the behavior of the different optimization algorithms and their performances. The remainder of this paper is structured as follows. The first section is a summary of each algorithm followed by its mathematical representation. The second section presents the Rosenbrock function and the analysis of the performance of the different optimization algorithms and we conclude the whole paper in the last section.

2 Optimization Algorithms

2.1 Gradient Descent

Gradient descent is the most popular algorithm to perform optimization in machine learning as It calculates which way the weights should be altered so that the function can reach a minimum. This algorithm is dependent on the first-order derivative of a loss function. It is easy to implement and to compute but it consumes a long time to converge to the minima as the weights are updated after calculating the gradient of the whole dataset.

$$\theta(x+1) = \theta(x) - \alpha \nabla J(\theta) \tag{1}$$

where :

$$\frac{\partial J}{\partial \theta} = \frac{2}{N} \sum_{n=1}^{N-1} (y_n - t_n) x_n$$

2.2 Stochastic Gradient Decent

Stochastic Gradient Descent is a Gradient Descent upgrade that updates model parameters more frequently. In this algorithm, the model parameters are modified after the loss calculation on each training example so that the updated parameters have high variance and fluctuations at different intensities. Hence the convergence is less in time compared to gradient descent but it is still slow compared to other optimizers.

$$\theta(x+1) = \theta(x) - \alpha \nabla J(\theta, x_i, y_i) \tag{2}$$

Where : x_i, y_i are the samples

2.3 Momentum

Momentum was invented in order to reduce the high variance in SGD and to soften the convergence. It accelerates convergence toward the relevant direction and reduces fluctuation toward the irrelevant direction. In this algorithm, a new hyper-parameter is added known as momentum and it is symbolized by 'γ'. The Momentum algorithm converges faster than the gradient descent, but the hyper-parameter 'γ' needs to be selected manually and accurately.

$$v(t) = \gamma v(t-1) - \alpha \nabla J(\theta) \tag{3}$$

Where the weights : $\theta(t) = \theta(t-1) - v(t)$
The momentum term γ is usually set to 0.9 or a similar value.

2.4 Nestrov Accelerate Gradient (NAG)

the NAG algorithm was invented to resolve the issue of missing the local minima in case the momentum is too high. This algorithm is a look-ahead method. It is using $\gamma v(t-1)$ for modifying the weights so,$\theta - \gamma v(t-1)$ tells the future location. Now, the optimizer calculates the cost based on this future parameter rather than the current one. This optimizer does not miss the local minima but it still face the problem of selecting the hyper-parameter manually

$$v(t) = \gamma v(t-1) - \alpha \nabla J(\theta, \gamma v(t-1)) \tag{4}$$

Where the weights : $\theta(t) = \theta(t-1) - v(t)$

2.5 Adagrad

AdaGrad solves the problem of the constant learning rate as it changes the learning rate 'α for each parameter and at

every time step 't'. It is a type second-order optimization algorithm.

$$\theta(t+1) = \theta(t) - \alpha'_{t+1}\nabla J(\theta) \tag{5}$$

Where:

$$\alpha'_t = \frac{\alpha}{\sqrt{\eta_t + \varepsilon}}$$

ε : is a small positive number to avoid divisibility by 0

$$\eta_t = \sum_{i=1}^{t} (\frac{\partial J}{\partial \theta_{t-1}})^2$$

2.6 Root Mean Squared Propagation

RMSProp, is an expansion of gradient descent and AdaGrad version of gradient descent that uses the decay of the average of the partial gradients in adapting the step size for each parameter. The use of a decaying moving average allows the algorithm to focus on the most recently observed partial gradients seen during the progress of the search and to forget the early gradients

$$G_t = \gamma G_{t-1} + (1-\gamma)(\nabla J\theta)^2 \tag{6}$$

$$\theta_{t+1} = \theta_t - G_t + \varepsilon\eta.\nabla J\theta$$

2.7 AdaDelta

AdaDelta is an extension of **AdaGrad** which aims to remove the decaying learning Rate problem. Instead of assembling all previously squared gradients, Adadelta limits the window of accumulated past gradients to some fixed size w. Hence In order to resolve the exponential increase in the summation of squared gradients "η, the "η was replaced with exponentially weighted averages of squared gradients.

$$\alpha'_t = \frac{\alpha}{\sqrt{\S_{d\theta} + \varepsilon}} \tag{7}$$

ε : is a small positive number to avoid divisibility by 0

$$\S_{d\theta} = {}_{d\theta(t-1)} + (1-\beta)(\frac{\partial J}{\partial \theta_{t-1}})^2$$

2.8 Adam

Adam is a replacement optimization algorithm for stochastic gradient descent for training deep learning models.[1] which combines the properties of the AdaGrad and RMSProp algorithms in order to provide an optimization algorithm that can handle sparse gradients on noisy problems.

$$\theta_t = \theta_{t-1} - \frac{\alpha}{\sqrt{\S_{d\theta} + \varepsilon}} * v_{d\theta} \tag{8}$$

3 Performance Analysis

In this section we are going to present the performance of each optimization algorithms using the **Rosenbrock Function**.

3.1 Rosenbrock Function

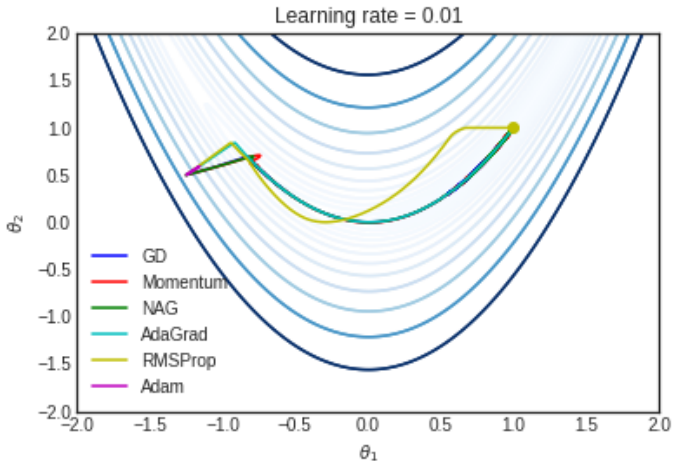
In mathematical optimization, the Rosenbrock function is a non-convex function used as a performance test problem for optimization algorithms introduced by Howard H. Rosenbrock in 1960[3]. It is also known as Rosenbrock's valley or Rosenbrock's banana function.

$$f(x,y) = (1-x)^2 + k(y-x^2)^2 \tag{9}$$

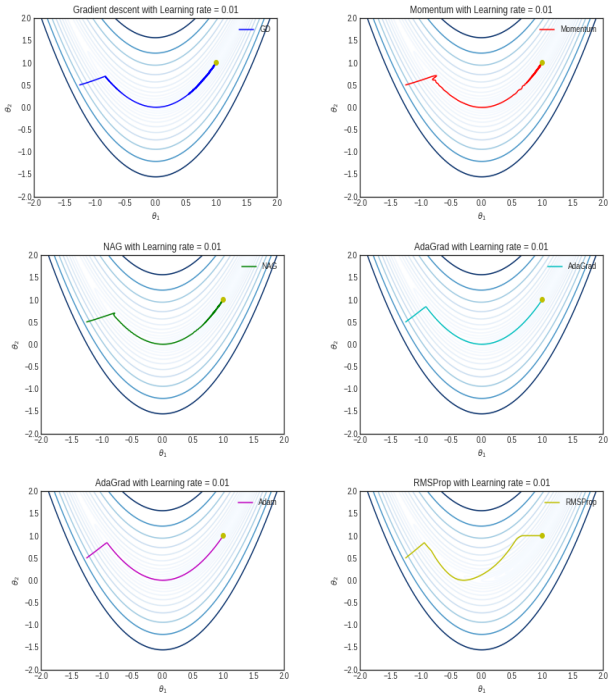
where k > 0: Maximum iteration number to achieve convergence This Function has a global minima at 0 When x=y=1 **In this paper we will use k=100**

3.2 Experiment

The Figure Below demonstrates the performance of each optimization algorithm learning rate $\alpha = 0.01$. All the optimizers converge to the minima at a maximum iteration number =10000. The learning rate α has an impact on the speed and convergence of each optimizers. « The learning rate is perhaps the most important hyperparameter. If you have time to tune only one hyperparameter, tune the learning rate. » [2]



The Figures below presents the individual behaviour of each optimizer on the Rosenbrock function while $\alpha = 0.01$. The learning rate controls how quickly the model is adapted to the problem. we noticed that if the α value was too high or too small it leads to divergence of the optimizers.



4 Conclusion

In this paper, we described the different optimization methods from the perspective of machine learning in addition to their performance using the Rosenbrock function.Our main Aim in our next research paper is to present a new version of the Optimization Algorithm.

Acknowledgement

* This work was carried out as a result of the research result of the SW-centered university project of the Ministry of Science and ICT and the Information and Communication Planning and Evaluation Institute in 2022 (2022-00964), and the University ICT Research Center Fostering Support Project of the Ministry of Science and ICT and the Information and Communication Planning and Evaluation Institute (ITP-2022-2018-0-01396). This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ICT Challenge and Advanced Network of HRD program (2020-0-01826)

References

[1] Jason Brownlee. *Gentle Introduction to the Adam Optimization Algorithm for Deep Learning*. URL: `https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/`.

[2] Aaron Courville Ian Goodfellow Yoshua Bengio. *Deep Learning. Adaptive Computation and Machine Learning series*. The MIT Press, 2016.

[3] H. H. Rosenbrock. *An Automatic Method for Finding the Greatest or Least Value of a Functiong*. URL: `https://doi.org/10.1093/comjnl/3.3.175`.