Abstract

Ever since the inception of Gradient Descent algorithm, it is without a doubt the most popular optimization strategy used in machine learning and deep learning. In this paper we have used various techniques to accelerate the gradient vectors in the right direction. The main problem with gradient descent algorithm is the rate of convergence. So to speed up the process, we take into effect all the previous gradients and tune the current gradient accordingly. There are various techniques available to do so and we have explored 5 such techniques namely i) No Momentum ii) Polyak’s Classical Momentum iii) Nesterov’s Accelerated Gradient (iv) RmsProp and (v) ADAM . We will compare the accuracies, rate of convergence and the stability for all this techniques in this paper.

Introduction

The most common method for neural network optimization is gradient descent and is one of the most favored algorithms. And almost every deep learning library is equipped with various implementations of numerous algorithms to optimize the gradient descent. In this paper, we have implemented some of those techniques.

Gradient descent is a way to minimize an objective function parameterized by a model’s parameters d by updating the parameters in the opposite direction of the gradient of the objective function w.r.t. to the parameters. The learning rate determines the size of the steps we take to reach a (local) minimum. In other words, we follow the direction of the slope of the surface created by the objective function downhill until we reach a valley

Non-convex optimization problems are natural formulations in many machine learning problems (e.g. (Un)supervised learning, Bayesian learning). Various learning approaches have been proposed in such settings, as global minimization of such problems are NP-hard in general. Gradient descent is the de-facto iterative learning algorithm used for such optimization problems in machine learning, especially in deep learning. Several variants of gradient descent methods have been proposed and all thse proposed methods can be broadly classified into momentum-based methods (e.g. Nesterov’s Accelerated Gradient [9]), variance reduction methods (e.g. Stochastic Variance Reduced Gradient [6],[11]) and adaptive learning methods (e.g. AdaGrad [2]). Gradient descent coupled with momentum - also called classical momentum by Polyak [10], is the first ever variant of gradient descent involving the usage of a momentum parameter. The momentum methods use the information from previous gradients in addition to the current gradient for updating the learning parameters. Nesterov in his seminal work [9], proposed an accelerated gradient method (also a momentum based method as shown by [15]) which gives an upper bound on the number of iterations for learning algorithm to converge. With the tremendous success of deep learning models, Sutskever et al in their work [15] worked out to incorporate the algorithm by Nesterov [9]. Nesterov’s method performs an update in the same way as classical momentum, only with a correction to the gradient

Gradient descent coupled with momentum - also called classical momentum by Polyak [10], is the first ever variant of gradient descent involving the usage of a momentum parameter. The momentum methods use the information from previous gradients in addition to the current gradient for updating the learning parameters. Nesterov in his seminal work [9], proposed an accelerated gradient method (also a momentum based method as shown by [15]) which gives an upper bound on the number of iterations for learning algorithm to converge. With the tremendous success of deep learning models, Sutskever et al in their work [15] worked out to incorporate the algorithm by Nesterov [9]. Nesterov’s method performs an update in the same way as classical momentum, only with a correction to the gradient.

Gradient descent is generally used in the form of mini-batch gradient descent in minimizing all real world optimization problems, where only a small subset of training data (called a mini-batch) is used due to the presence of enormous amounts of training data. The use of a mini-batch for gradient calculation introduces a lot of variance due to the stochasticity of learning algorithm. Methods like SVRG [6], [11] have been proposed, which try to reduce the variance in gradient with strong theoretical guarantees. There exist other variance reduction methods like SAG [13] and SDCA [14] also. Recently, several methods have been proposed that try to adapt the learning rate in gradient descent. Riedmiller and Braun proposed Rprop [12] method which suggested the usage of an adaptive learning rate based on the sign of gradient in last two iterates. Rprop increases the learning rate of a weight if the gradient sign does not change in last two iterates, otherwise it decreases the learning rate. AdaGrad - proposed by Duchi et al [2], divides η(a global learning rate) of every step by the square of the ℓ2 norm of all previous gradients. This scaling using the norm reduces the learning in dimensions which have already changed significantly, and speeds up in the dimensions that have not changed rapidly, thereby stabilizing the model. RMSProp proposed by Tieleman et al [16] is a simple amalgamation of Rprop and SGD. This method scales the learning rate by the decaying average of squared gradient. There are few other methods proposed which adapts the learning rate like AdaDelta [18]. Adam - proposed by Kingma and Ba [7] is a very successful method that almost all recent state-of-the-art deep learning models used. Adam makes use of the first and second order moments of gradients and ideas from norm-based methods, through combining the advantages from AdaGrad and RMSProp.