ISSN 2522-6398 Vol. 3, Issue 4: 59-62, DOI: 10.25236/FER.2020.030414

Simpler Algorithm in Gradient Descent

David Woodruff [1], Youyu Zhu[2], Xiye Zhang[3]

1 Carnegie Mellon University,5000 Forbes Avenue Pittsburgh, PA 15213
2 Shanghai Kongjiang Middle School, Shanghai, China
3 Sichuan University, Sichuan, China

ABSTRACT. *In Batch Gradient Descent, the most efficient constant learning rate should be* $\frac{1}{L}$*, and L is the Lipschitz constant. In "Learning the Learning Rate"(Xiaoxia Wu et al., 2018), some of their algorithms seem t-o be inefficient. This paper is gonna to further improve their method. The searching-L algorithm is raised in the following passage, which limits the WNGrad-Batch's complexity from square to linear by gradually approximating our learning rate to* $\frac{1}{L}$*. In stochastic gradient descent, we find that b-y using the learning rate* n_k*, which conforms the updating rule:* $n_k = \frac{c}{k^{\frac{1}{2}}}$*, can have a more stable upper bound of the time complexity, which won't have a parameter γ.*

KEYWORDS: *Learning rate; Gradient descent*

1. Introduction

Recall that gradient descent is a process to find the minimum of the loss function $f(x)$by iteratively move x towards the convergent point – the lowest place in the functional image: $x_{k+1} \leftarrow x_k - \eta_k \nabla f(x)$.Gradient descent enjoys great convergence if the learning rate η_j fits the smoothness of the function.If it is too small, it takes forever for x to reach the target, and if it is too large, x will oscillate or even diverge. Linear search method works well since gradients $f(x)$ are observed exactly in "Batch", but it becomes less effective when comes to stochastic settings, where only noisy gradients are given. In SGD, we observe a stochastic gradient g_k, satisfying $E(g_k) = \nabla f(x)$ and $E||g_k||^2 \leqslant G^2$. SGD is a optimum algorithm to choose in deep learning, and it can eliminate the time-consuming problem in "Batch gradient descent" if the data set is too large.

In stochastic settings, there are several different guidelines of setting learning rates ($\eta_1 ... \eta_n$). The classical theory (Robbins and Monro, 1951) says that if the learning rate chosen satisfies

$$\sum_{k=1}^{\infty} \alpha_k = \infty \quad \text{and} \quad \sum_{k=1}^{\infty} \alpha_k^2 < \infty \tag{1}$$

and the loss function is smooth enough, then $\lim_{k\to\infty} E[\|f'(x)\|^2]=0$.If the loss function is also strongly convex, then the SGD-update $x_{k+1} \leftarrow x_k - \eta_k g_k$ will converge in expectation to the minimum.

Therefore machine learning's essential question is how to find the learning rate that suits the component function f_i. The family of adaptive gradient (AdaGrad) algorithms (Duchi et al., 2011) showed an great improvement by dynamically updating his learning rates and reaching a convergence performance over standard stochastic gradient descent with sparse data and informative parameters. This method was then applied widely in language processing. But how to find a efficient learning rate remains a question. According to some researches on adaptive learning rate (Needell et al., 2014; Zhao and Zhang, 2015), Lipschitz constant seems to be the symbol to decide how to modify the current rate: increase when it is smaller than the constant and decrease when it is larger. However, Lipschitz constant remains unknown along the way, which means it should also be previously learned just as the learning rate. In WNGrad (Xiaoxia Wu et al., 2018), a method that do not need to know constant L in advance is advocated, which is a great advance. But in their paper, the complexity of the steps needed in "WNGrad-Batch setting" is

ISSN 2522-6398 Vol. 3, Issue 4: 59-62, DOI: 10.25236/FER.2020.030414

$$T = 1 + \frac{L^2(1-\delta)}{s} + \frac{16((f(x_1) - f^*) + (\frac{3}{16} + \frac{5}{8\delta})L)^2}{2} \quad \text{when } b = \delta L(\delta \in 0,1)$$

It is absolutely not a very tight bound since it contains the square term, and this paper aims to improve their "Batch" algorithm to linear complexity. Also, there is a γ in their SGD method, and we gonna to prove it unnecessary with a clearly determined learning rate ηk .

2. Searching "l" – Batch Set

Consider a smooth function f that satisfies Lipschitz continuity ($f \in C_L^1$) : for any $x, y \in R^d$,

$$\| \nabla f(x) - \nabla f(y) \| \leq L \| x - y \| \tag{2}$$

and the optimum problem

$$\min f(x), \tag{3}$$

the standard gradient update should be $x_{i+1} \leftarrow x_i - \eta \nabla f(x_i)$ (4)

The convergent results are classical ((Nesterov, 1998), (1.2)).

For initialization, sampling $R \geq 2$ points u_1 , u_2 , ... u_R , and $\nabla f(u_1)$ (u_1) (u_1), $\nabla f(u_1)$, ..., $\nabla f(u_R)$:

$$b = \max \| f(u\ j) - f(u\ k) \| = \delta L, \delta \in (0,1] \tag{5}$$

Let $\frac{1}{b}$ be the learning rate.

Lemma1 Suppose that f ∈ C1 L and that $f^* > -\infty$. Consider gradient descent with constant learning rate $\eta > 0$. If $\eta = \delta L$ and δ ⩽ 1, then it takes

$$T = \frac{2L(f(x_1) - f^*)}{\delta s_2} \tag{6}$$

to reach

$$\min_{j=1:T} \left\| \nabla f(x)_j \right\| \leq s \tag{7}$$

Then consider the following modified gradient descent scheme:

Algorithm 1 Searching L

Input:

$s > 0$ Initialize $x_1 \in R^d, b_1 > 0, j \leftarrow 1, k$ 1

while 1 do

$j \leftarrow j + 1$

$$x_j \leftarrow x_{j-1} - \frac{1}{b_k} \nabla(x_{j-1})$$

if ∇ $\| f(x_j) \|^2 < s$ then

return $\frac{1}{b_k}$

break

end if

ISSN 2522-6398 Vol. 3, Issue 4: 59-62, DOI: 10.25236/FER.2020.030414

if $j = T + 1$ then

$$k \leftarrow \frac{k}{2} + 1$$
$$b_k \leftarrow 1\, s$$
$$b_{k-1}$$
$$T = 2b_k (fx_1^S)$$
$$j \leftarrow 1$$
$$x_j \leftarrow x_1$$

end if

end while

Theorem 1 Suppose all the f(x) are positive. Consider the Searching L algorithm when $b_1 = \delta L, \delta \in (0,1]$ and the best fixed learning rate $\frac{1}{L}$, it takes

$$T = \frac{2L(2-\delta) f(x_1)}{s} \tag{8}$$

which is better than the square item of WNGrad

Proof 2.1 Since $f(x)$ are all positive $(b_k = 2^{k-1}\delta L, k \in Z)$, Suppose is returned $(f(x_1) - f^* < f(x_1)$ then the total steps is equal to

$$\begin{aligned} T &= \sum_{j=0}^{\log_2 \frac{1}{\delta}} \frac{2\delta L(fx1))}{s} 2^j \\ &= \frac{2\delta L f(x_1)}{s} * \frac{1(1-2^{1-\log_2 \delta})}{1-2} \\ &= \frac{2QL(fx_1))}{s} * \frac{2-\delta}{\delta} \\ &= \frac{2L(2^S - \delta)(fx_1)^\delta}{s} \end{aligned} \tag{9}$$

3. Determined Learning Rate in Stochastic Gradient Descent

We now consider the learning rate in stochastic gradient descent without the parameter γ. We can just let learning rate n_k equal to $c / k^{1/2}$,and the learning rate will be changed with the times of iteration k.

Algorithm 2 Learning Rate with $c / k^{1/2}$

Input: $s > 0$, $c > 0$,initialize $\in R^d, b_1 \leftarrow cj$ 1

repeat

$$j \leftarrow +1$$
$$x_j \leftarrow -x_{j-1} - {}^{gj-1}$$
$$b \leftarrow \frac{(i-1)}{i} b_{j-1}$$

until $f(x_j) < S$

Consider the general optimal problem

$\min f(x)$

x

From stochastic gradient information.Instead of using full gradients $\nabla f(x_k)$,we observe stochastic gradients $g_k \in R^d$, satisfying $E(g_k) = \nabla f(x_k)$. Let $\overline{x} = \frac{1}{k}\sum_{k=1}^{k} x_i$

ISSN 2522-6398 Vol. 3, Issue 4: 59-62, DOI: 10.25236/FER.2020.030414

Theorem 3.1 Suppose $f(x)$ is convex. Suppose, that, independent of x_k,

$\mathrm{E}(\|g_k\|^2) \leqslant tt$

and $\mathrm{E}\|x_i - x^*\|^2 \leqslant D^2$. Then with initialization $b_1 \geq \|g_1\|$,

$$f(\bar{x}) - f^* \leq \frac{(D^2+1)(k-1)^{\frac{1}{2}}}{2kc} + \frac{1}{2_{ck}} \left\|x_1 - x^*\right\|^2 + cttk^{-\frac{1}{2}} \quad -$$

Proof of Theorem 3.1: First note that $b_k = \frac{1}{\eta k} = k\frac{1/2}{c}$,then

$$\| x_{k+1} - x^* \| = \| x_k - x^* \|^2 + \| gk \|^2 \frac{1}{b_k^2} - 2\frac{1}{b_k}(xk - x^*, gk)$$

So,

$$2(x_k - x^*, g_k) = b_k \left\|x_k - x^*\right\|^2 - b_k \left\|x_{k+1} - x^*\right\|^2 + \frac{1}{} \left\|g_k\right\|^2$$

Thus,

$$2\sum_{i=1}^{k}(x_k - x^*, g_k) \leq 2\sum_{i=1}^{k}(x_k - x^*, g_k) + b_k \left\|x_{k+1} - x^*\right\|^2$$

From Jensens inequality, and recalling that by convexity conclude $f(x_k) - f^* \leq (x_k - x^*, \nabla f(x_k)) = E(x_k - x^*, g_k)$, we conclude

$$f(\bar{x}_k) - f^* \leq \frac{1}{k}\sum_{i=1}^{k}(f(x_l) - f^*)$$

$$\leq \frac{1}{k}\sum_{i=1}^{k}E(x_l - x^*, g_l)$$

$$\leq \frac{D^2(k-1)^{\frac{1}{2}}}{2kc} + \frac{1}{2ck}\| x_1 - x^* \| + cGk^{-1/2}$$

References

[1] Y Nesterov (1998). Introductory lectures on convex programming volume i: Basic course, pp.75-76.

[2] Xiaoxia Wu, Rachel Ward, Leon Bottou (2018). WNGrad: Learn the Learning Rate in Gradient Descent, pp.89-90.