

Optimization Algorithms from a Dynamics Perspective (II): Adaptive Learning Rate Algorithms

Jianlin Su

December 20, 2018

In the article "[Optimization Algorithms from a Dynamics Perspective \(I\): From SGD to Momentum Acceleration](#)", we proposed that the SGD optimization algorithm corresponds to the numerical solution of Ordinary Differential Equations (ODEs). From this perspective, we can naturally analyze the convergence properties of the SGD algorithm, the principles of momentum acceleration, and more.

In this article, we continue along this line of thought to understand adaptive learning rate algorithms in optimization.

1 RMSprop

First, let's look at a very classic adaptive learning rate optimization algorithm: **RMSprop**. Although RMSprop was not the first adaptive learning rate algorithm proposed, it is a highly practical one and serves as the foundation for more comprehensive algorithms like Adam. Through it, we can observe how adaptive learning rate optimization algorithms work.

1.1 Algorithm Overview

General gradient descent is as follows:

$$\boldsymbol{\theta}_{n+1} = \boldsymbol{\theta}_n - \gamma \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_n) \quad (1)$$

Obviously, here γ is a hyperparameter, the learning rate, which may need different adjustments at different stages.

In contrast, RMSprop is:

$$\begin{aligned} \mathbf{g}_{n+1} &= \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}_n) \\ \mathbf{G}_{n+1} &= \lambda \mathbf{G}_n + (1 - \lambda) \mathbf{g}_{n+1} \otimes \mathbf{g}_{n+1} \\ \boldsymbol{\theta}_{n+1} &= \boldsymbol{\theta}_n - \frac{\tilde{\gamma}}{\sqrt{\mathbf{G}_{n+1} + \epsilon}} \otimes \mathbf{g}_{n+1} \end{aligned} \quad (2)$$

1.2 Algorithm Analysis

Comparing it with naive SGD, we find that in the update of $\boldsymbol{\theta}$, RMSprop replaces the original scalar learning rate γ with a vector:

$$\boldsymbol{\gamma} = \frac{\tilde{\gamma}}{\sqrt{\mathbf{G}_{n+1} + \epsilon}} \quad (3)$$

If we view this vector as the learning rate, then RMSprop has found a scheme that can assign different learning rates to each component of the parameters.

This adjustment of the learning rate is achieved through the factor $\frac{1}{\sqrt{G_{n+1}+\epsilon}}$, where G_{n+1} is the moving average of the squared gradients. Essentially, the "moving average" only makes the training process smoother; it is not the primary cause of the adjustment. The main active part is the "gradient" itself—that is, the magnitude of the gradient can be used to adjust the learning rate.

2 Adaptive Learning Rate

Why can the magnitude of the gradient be used to adjust the learning rate? Actually, this idea is very simple.

2.1 Local Minima and ODEs

To keep things simple, let's start with a one-dimensional example. Suppose we want to find a local minimum of $L(\theta)$. We introduce a virtual time parameter t and transform the problem into an ODE:

$$\frac{d\theta}{dt} = \dot{\theta} = -L'(\theta) \quad (4)$$

It is easy to determine that a local minimum of $L(\theta)$ is a stable fixed point of this equation. Starting from any θ_0 , we numerically solve this ODE and expect it to eventually converge to this fixed point, thereby obtaining a local minimum.

The simplest Euler method uses $\frac{\theta_{t+\gamma}-\theta_t}{\gamma}$ to approximate $\dot{\theta}$, resulting in:

$$\frac{\theta_{t+\gamma}-\theta_t}{\gamma} = -L'(\theta_t) \quad (5)$$

Which is:

$$\theta_{t+\gamma} = \theta_t - \gamma L'(\theta_t) \quad (6)$$

This is the gradient descent method, where $\theta_{t+\gamma}$ corresponds to θ_{n+1} and θ_t corresponds to θ_n , advancing by γ in each step.

2.2 The Idea of Variable Learning Rates

The question is, what is the best value for γ ? From the perspective of "using $\frac{\theta_{t+\gamma}-\theta_t}{\gamma}$ to approximate $\dot{\theta}$," a smaller γ is certainly more accurate. However, the smaller γ is, the more iterations are required, meaning the computational cost increases. Thus, while "the smaller, the better" is ideal, it is not practical.

Therefore, the most appropriate scheme is: **each step should be just enough**. But how do we know if it is enough?

Since we are using $\frac{\theta_{t+\gamma}-\theta_t}{\gamma}$ to approximate $\dot{\theta}$, we must analyze the degree of approximation. According to the Taylor series, we have:

$$\theta_{t+\gamma} = \theta_t + \gamma \dot{\theta}_t + \mathcal{O}(\gamma^2) \quad (7)$$

In our case, $\dot{\theta} = -L'(\theta)$, so we have:

$$\theta_{t+\gamma} = \theta_t - \gamma L'(\theta_t) + \mathcal{O}(\gamma^2) \quad (8)$$

We can expect that when γ is relatively small, the error term $\mathcal{O}(\gamma^2) < \gamma |L'(\theta_t)|$. That is to say, under certain conditions, $\gamma |L'(\theta_t)|$ itself is a measure of the error term. If we control $\gamma |L'(\theta_t)|$ within a certain range, the error is also controlled. Namely:

$$\gamma |L'(\theta_t)| \leq \tilde{\gamma} \quad (9)$$

Where $\tilde{\gamma}$ is a constant. We can even simply set $\gamma |L'(\theta_t)| = \tilde{\gamma}$ (temporarily ignoring the possibility of $L'(\theta_t) = 0$ to observe the core idea), which gives:

$$\gamma = \frac{\tilde{\gamma}}{|L'(\theta_t)|} \quad (10)$$

In this way, we adjust the learning rate through the gradient.

2.3 Moving Average Processing

Readers might criticize that substituting $\gamma = \tilde{\gamma} / |L'(\theta_t)|$ back into the original iteration results in:

$$\theta_{t+\tilde{\gamma}/|L'(\theta_t)|} = \theta_t - \tilde{\gamma} \cdot \text{sign}[L'(\theta_t)] \quad (11)$$

You have only used the sign information of the gradient; isn't this too wasteful? (It is too trivial: regardless of the gradient magnitude, θ only moves by a fixed length in each iteration.)

Note that from the perspective of solving an ODE, this is not inherently wrong because the solution to an ODE is a trajectory $(t, \theta(t))$. In the processing above, although the movement of θ becomes trivial, t becomes non-trivial; it is equivalent to swapping the roles of t and θ , so it is still reasonable. However, if we care about the optimization problem—finding the minimum of $L(\theta)$ —then the above formula is indeed somewhat trivial. If θ only moves by a fixed length in each iteration, it becomes similar to a grid search, which is too inefficient.

Therefore, to improve this triviality while retaining the feature of using the gradient to adjust the learning rate, we can apply a "moving average" to the gradient. The result is:

$$\begin{aligned} G_{t+\tilde{\gamma}} &= \lambda G_t + (1 - \lambda) |L'(\theta_t)|^2 \\ \gamma &= \frac{\tilde{\gamma}}{\sqrt{G_{t+\tilde{\gamma}} + \epsilon}} \\ \theta_{t+\gamma} &= \theta_t - \gamma L'(\theta_t) \end{aligned} \quad (12)$$

Here λ is a constant close to 1 but less than 1. In this way, G_t remains relatively stable within a certain range while retaining the characteristics of the gradient $L'(\theta_t)$ to some extent. Using it to adjust the learning rate is a "clever" approach. To avoid confusion caused by the notation $t + \tilde{\gamma}$ and $t + \gamma$, we use n and $n + 1$ as subscripts to unify them:

$$\begin{aligned} G_{n+1} &= \lambda G_n + (1 - \lambda) |L'(\theta_n)|^2 \\ \gamma &= \frac{\tilde{\gamma}}{\sqrt{G_{n+1} + \epsilon}} \\ \theta_{n+1} &= \theta_n - \gamma L'(\theta_n) \end{aligned} \quad (13)$$

This is the RMSprop algorithm mentioned at the beginning.

Furthermore, there is another important reason for the moving average: what we actually want to estimate is the mean of the squared gradients across the entire sample set. However, we can only calculate one batch at a time (so-called mini-batch gradient descent). Consequently, the result calculated each time is actually biased, and the moving average helps correct this bias to some extent.

2.4 High-dimensional Case Analysis

The discussion above covers the one-dimensional case. How do we generalize this to multiple dimensions?

Readers might think it is simple: just replace scalars with vectors. It is not quite that simple, because generalizing (13) to high dimensions allows for at least two reasonable choices:

$$\begin{aligned} G_{n+1} &= \lambda G_n + (1 - \lambda) \|\nabla_{\theta} L(\theta_n)\|^2 \\ \gamma &= \frac{\tilde{\gamma}}{\sqrt{G_{n+1} + \epsilon}} \\ \theta_{n+1} &= \theta_n - \gamma \nabla_{\theta} L(\theta_n) \end{aligned} \tag{14}$$

Or:

$$\begin{aligned} \mathbf{G}_{n+1} &= \lambda \mathbf{G}_n + (1 - \lambda) (\nabla_{\theta} L(\theta_n) \otimes \nabla_{\theta} L(\theta_n)) \\ \gamma &= \frac{\tilde{\gamma}}{\sqrt{\mathbf{G}_{n+1} + \epsilon}} \\ \theta_{n+1} &= \theta_n - \gamma \otimes \nabla_{\theta} L(\theta_n) \end{aligned} \tag{15}$$

The former uses the total norm of the gradient for accumulation, maintaining the **scalar nature of the learning rate**. The latter accumulates each component of the gradient separately; in this case, the adjusted **learning rate becomes a vector**, which is equivalent to assigning a different learning rate to each parameter. From the perspective of strict theoretical analysis, the first approach is more rigorous, but from the perspective of experimental results, the second is more effective.

The RMSprop algorithm we usually refer to is the latter (15). However, many practitioners of "deep learning alchemy" who prefer pure SGD criticize this vectorized learning rate for actually changing the direction of the gradient, leading to inaccurate gradients and ultimately suboptimal results. Readers who dislike vectorized learning rates might want to experiment with the former.

3 Summary

This article has once again analyzed optimization algorithms from the perspective of ODEs, this time providing an understanding of an adaptive learning rate algorithm (RMSprop) from the perspective of error control. As for the more commonly used Adam, it is a combination of RMSprop and momentum acceleration, which we will not elaborate on here.

Viewing optimization problems as the task of solving ordinary differential equations transforms optimization into a dynamical problem. This allows us to understand optimization algorithms from a more physical perspective (even if it is just an intuitive rather than strictly rigorous understanding) and even allows us to borrow theoretical results from ODEs. I will attempt to provide more such examples in the future.

When reposting, please include the original address of this article: <https://kexue.fm/archives/6234>
For more detailed reposting matters, please refer to: "Scientific Space FAQ"