Exponentiated Gradient Descent + Meta Learning = Adaptive Learning Rate

Jianlin Su

March 3, 2022

A few days ago, I came across a Google paper titled "Step-size Adaptation Using Exponentiated Gradient Updates". I learned some new concepts from it, so I am recording and sharing them here. There are two main contents: one is Exponentiated Gradient Descent for non-negative optimization, and the other is a learning rate adjustment algorithm based on the idea of meta-learning. Both are quite interesting, and interested readers may find them worth exploring.

1 Exponentiated Gradient Descent

You may have heard much about Gradient Descent, which refers to the minimization of an unconstrained function $\mathcal{L}(\boldsymbol{\theta})$ using the following update format:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_t) \tag{1}$$

where η is the learning rate. However, many tasks are not always unconstrained. For the simplest non-negative constraint, we can change the update to the following format:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t \odot \exp\left(-\eta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}_t)\right) \tag{2}$$

Here $\odot$ denotes element-wise multiplication (Hadamard product). It is easy to see that as long as the initial $\boldsymbol{\theta}_0$ is non-negative, $\boldsymbol{\theta}_t$ will remain non-negative throughout the entire update process. This is known as "Exponentiated Gradient Descent" used for non-negative constrained optimization.

How should we understand this "Exponentiated Gradient Descent"? It is not difficult; we can derive it by transforming it into an unconstrained case. If $\boldsymbol{\theta}$ is non-negative, then $\boldsymbol{\varphi} = \log \boldsymbol{\theta}$ can be either positive or negative. Therefore, we can set $\boldsymbol{\theta} = e^{\boldsymbol{\varphi}}$ to transform the problem into an unconstrained optimization problem regarding $\boldsymbol{\varphi}$, which can then be solved using gradient descent:

$$\boldsymbol{\varphi}_{t+1} = \boldsymbol{\varphi}_t - \eta \nabla_{\boldsymbol{\varphi}} \mathcal{L}(e^{\boldsymbol{\varphi}_t}) = \boldsymbol{\varphi}_t - \eta e^{\boldsymbol{\varphi}_t} \odot \nabla_{e^{\boldsymbol{\varphi}}} \mathcal{L}(e^{\boldsymbol{\varphi}_t}) \tag{3}$$

We consider that the $e^{\boldsymbol{\varphi}_t}\odot$ part of the gradient only plays a role in adjusting the learning rate, so it is not essentially important. By discarding it, we obtain:

$$\boldsymbol{\varphi}_{t+1} = \boldsymbol{\varphi}_t - \eta \nabla_{e^{\boldsymbol{\varphi}}} \mathcal{L}(e^{\boldsymbol{\varphi}_t}) \tag{4}$$

Taking the exponent on both sides gives:

$$e^{\boldsymbol{\varphi}_{t+1}} = e^{\boldsymbol{\varphi}_t} \odot \exp\left(-\eta \nabla_{e^{\boldsymbol{\varphi}}} \mathcal{L}(e^{\boldsymbol{\varphi}_t})\right) \tag{5}$$

Substituting back $\boldsymbol{\theta} = e^{\boldsymbol{\varphi}}$, we arrive at Equation (2).

2 Meta-Learning for Learning Rate Adjustment

Regarding Meta-Learning, many readers might be like me—having heard of it often but rarely having touched it. Simply put, the relationship between ordinary machine learning and meta-learning is like the relationship between a "function" and a "functional" in mathematics. A functional is a "function of functions," and meta-learning is "Learning How to Learn." In other words, it is a methodology about "learning" itself, such as what we are about to introduce: "using gradient descent to adjust gradient descent."

We start from general gradient descent. Denoting the gradient of the objective function $\mathcal{L}$ as $\boldsymbol{g}$, the update formula is:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \boldsymbol{g}_t \tag{6}$$

We want to adjust the learning rate for each component, so we introduce a non-negative variable $\boldsymbol{\nu}$ of the same size as the parameters, and modify the update formula to:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \boldsymbol{\nu}_{t+1} \odot \boldsymbol{g}_t \tag{7}$$

So, what rule should $\boldsymbol{\nu}$ follow for iteration? Remember that our ultimate goal is to minimize $\mathcal{L}$, so the update rule for $\boldsymbol{\nu}$ should also be gradient descent. Since $\boldsymbol{\nu}$ is required to be non-negative, we use exponentiated gradient descent:

$$\boldsymbol{\nu}_{t+1} = \boldsymbol{\nu}_t \odot \exp\left(-\gamma \nabla_{\boldsymbol{\nu}_t} \mathcal{L}\right) \tag{8}$$

Note that $\mathcal{L}$ is originally only a function of $\boldsymbol{\theta}$, but according to (7), at time t we have $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \eta \boldsymbol{\nu}_t \odot \boldsymbol{g}_{t-1}$. Therefore, according to the chain rule:

$$\nabla_{\boldsymbol{\nu}_t} \mathcal{L} = -\eta \boldsymbol{g}_{t-1} \odot \nabla_{\boldsymbol{\theta}_t} \mathcal{L} = -\eta \boldsymbol{g}_{t-1} \odot \boldsymbol{g}_t \tag{9}$$

Substituting this into the update formula for $\boldsymbol{\nu}$ (8), we get:

$$\boldsymbol{\nu}_{t+1} = \boldsymbol{\nu}_t \odot \exp\left(\gamma \eta \boldsymbol{g}_{t-1} \odot \boldsymbol{g}_t\right) \tag{10}$$

Combining $\gamma\eta$ into a single parameter γ, the update formulas for the entire model are:

$$\begin{aligned} \boldsymbol{\nu}_{t+1} &= \boldsymbol{\nu}_t \odot \exp\left(\gamma \boldsymbol{g}_{t-1} \odot \boldsymbol{g}_t\right) \\ \boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t - \eta \boldsymbol{\nu}_{t+1} \odot \boldsymbol{g}_t \end{aligned} \tag{11}$$

If $\boldsymbol{\nu}$ is initialized to all ones, then we will have:

$$\boldsymbol{\nu}_{t+1} = \exp\left(\gamma \sum_{k=1}^{t} \boldsymbol{g}_{k-1} \odot \boldsymbol{g}_k\right) \tag{12}$$

As can be seen, the idea of this method for adjusting the learning rate is: if the gradients of a certain component in two adjacent steps often have the same sign, the accumulated result of the corresponding terms will be positive, meaning we can appropriately increase the learning rate; if the gradients of adjacent steps often have opposite signs, the accumulated result is likely to be negative, meaning we can appropriately decrease the learning rate.

Note that this is different from the idea of Adam's learning rate adjustment. Adam's idea is that if the gradient of a certain component remains very small for a long time, it means the parameter might not have been learned well, so it attempts to increase its learning rate. Both approaches have their own merits.

3 A Brief Summary

This article mainly makes simple notes on the two concepts of "Exponentiated Gradient Descent" and "Meta-Learning for Learning Rate Adjustment." "Exponentiated Gradient Descent" is a simple and effective solution for non-negative constrained optimization, while "Meta-Learning for Learning Rate Adjustment" is an easy-to-understand application of meta-learning. In introducing "Meta-Learning for Learning Rate Adjustment," I have made some simplifications, making it simpler than the form in the original paper, but the core idea remains consistent.