Supplemental Materials to "A PID Controller Approach for Stochastic Optimization of Deep Networks"

1. Connection

In this section, we show the mathematical connection between deep learning optimizer and PID controller.

1.1. PID Controller

The PID controller is the most commonly used control algorithm in industry since its origin in the 1940s. More than 90% of the controllers in the industrial products are PID [2]. Which has the following definition:

$$u(t) = K_p e(t) + K_i \sum_{i=0}^{t-1} e(i) + K_d(e(t) - e(t-1)) \tag{1}$$

where $u(t)$ is the controller's update, and $e(t)$ is the error between the system's output and the desired output. K_p, K_i and K_d are positive constants to balance present, past and future of the error $e(t)$.

By replacing the error $e(t)$ in PID controller with the gradient in deep learning optimization, the PID controller for deep learning optimization is given by:

$$u(t) = K_p \partial L_t/\partial\theta_t + K_i \sum_{i=0}^{t-1} (\partial L_i/\partial\theta_i) + K_d(\partial L_t/\partial\theta_t - \partial L_{t-1}/\partial\theta_{t-1}) \tag{2}$$

where $u(t)$ is the update of the weight, θ_t is the weight at iteration t and $\partial L_t/\partial\theta_t$ is the gradient of the network.

1.2. Deep Learning Optimizers

1.2.1 SGD is a P Controller

The update rule of SGD is:

$$\theta_{t+1} - \theta_t = -r\partial L_t/\partial\theta_t, \tag{3}$$

where r is the learning rate.

Comparing Equation. 3 with Equation. 2, we can see that the update of parameters relies on current gradient, and SGD is a P controller.

1.2.2 SGD-Momentum is a PI Controller

The update rule of SGD-Momentum is given by:

$$\begin{cases} V_{t+1} &= \alpha V_t - r\partial L_t/\partial\theta_t \\ \theta_{t+1} &= \theta_t + V_{t+1} \end{cases} \tag{4}$$

where α is a value to balance past and current gradients, usually set to 0.9 [3]. Dividing both sides of the 1st formula of Equation. 4 by α^{t+1}:

$$\frac{V_{t+1}}{\alpha^{t+1}} = \frac{V_t}{\alpha^t} - r\frac{\partial L_t/\partial \theta_t}{\alpha^{t+1}} \tag{5}$$

By applying Equation. 5 from time $t+1$ to 1, we have:

$$\begin{cases} \frac{V_{t+1}}{\alpha^{t+1}} - \frac{V_t}{\alpha^t} = -r\frac{\partial L_t/\partial \theta_t}{\alpha^{t+1}} \\ \frac{V_t}{\alpha^t} - \frac{V_{t-1}}{\alpha^{t-1}} = -r\frac{\partial L_{t-1}/\partial \theta_{t-1}}{\alpha^t} \\ \quad ...... = ... \\ \frac{V_1}{\alpha^1} - \frac{V_0}{\alpha^0} = -r\frac{\partial L_0/\partial \theta_0}{\alpha^1} \end{cases} \tag{6}$$

Add the above $t+1$ equations together, there is:

$$\frac{V_{t+1}}{\alpha^{t+1}} = \frac{V_0}{\alpha^0} - r(\sum_{i=0}^{t}(\frac{\partial L_i/\partial \theta_i}{\alpha^{i+1}})) \tag{7}$$

Without loss of generality, we set the initial condition $V_0 = 0$, and thus the above equation can be simplified as follows:

$$V_{t+1} = -r(\sum_{i=0}^{t}(\alpha^{t-i}\partial L_i/\partial \theta_{t-1})) \tag{8}$$

Put V_{t+1} into the 2nd formula of Equation. 4, we have:

$$\theta_{t+1} - \theta_t = -r\frac{\partial L_t}{\partial \theta_t} - r(\sum_{i=0}^{t-1}(\alpha^{t-i}\partial L_i/\partial \theta_i)) \tag{9}$$

We can see that the update of the parameter relies on both the current gradient (P control) and the integral of past gradients (I control). If we assume $\alpha = 1$, there is:

$$\theta_{t+1} - \theta_t = -r(\partial L_t/\partial \theta_t) - r(\sum_{i=0}^{t-1}(\partial L_i/\partial \theta_i)) \tag{10}$$

Comparing Equation. 10 with Equation. 2, we can see that SGD-Momentum is a PI controller with $K_p = r$ and $K_i = r$.

1.2.3 Nesterov′s Momentum is a PI Controller with larger P

The update rule of Nesterov′s Momentum is :

$$\begin{cases} V_{t+1} &= \alpha V_t - r\partial L_t/\partial(\theta_t + \alpha V_t) \\ \theta_{t+1} &= \theta_t + V_{t+1} \end{cases} \tag{11}$$

The expression is almost the same as SGD-Momentum except for the location where the gradient is evaluated. By using a variable transform $\hat{\theta}_t = \theta_t + \alpha * V_t$, we have:

$$\begin{cases} V_{t+1} &= \alpha V_t - r\partial L_t/\partial \hat{\theta}_t \\ \hat{\theta}_{t+1} &= \hat{\theta}_t + (1+\alpha)V_{t+1} - \alpha V_t \end{cases} \tag{12}$$

Similar to the derivation process in Equations. 5, 6 and 7 of SGD-Momentum, we have:

$$V_{t+1} = -r(\sum_{i=1}^{t}(\alpha^{t-i}\partial L_i/\partial \hat{\theta}_i)) \tag{13}$$

With Equation. 13, Equation. 11 can be rewritten as:

$$\hat{\theta}_{t+1} - \hat{\theta}_t = -r(1+\alpha)\partial L_t/\partial\hat{\theta}_t - \alpha r(\sum_{i=1}^{t-1}(\alpha^{t-i}\partial L_i/\partial\hat{\theta}_i)) \tag{14}$$

One can see that the update of parameters relies on the current gradient (P control) and the integral of past gradients (I control). If we assume $\alpha = 1$, then:

$$\hat{\theta}_{t+1} - \hat{\theta}_t = -2r(\partial L_t/\partial\hat{\theta}_t) - r(\sum_{i=0}^{t-1}(\partial L_i/\partial\hat{\theta}_i)) \tag{15}$$

Comparing Equation. 15 with Equation. 2, we can see that Nesterov′s Momentum is a PI controller with $K_p = 2r$ and $K_i = r$.

2. Laplace Transform of PID Optimizer

2.1. Laplace Transform

The Laplace Transform converts the function of real variable t (iteration) to a function of complex variable s (frequency). Denote by $F(s)$ the Laplace transform of $f(t)$. There is

$$F(s) = \int_0^\infty e^{-st} f(t)\, dt, \quad \text{for } s > 0. \tag{16}$$

Usually $F(s)$ is easier to solve than $f(t)$, and $f(t)$ can be recovered from $F(s)$ by the Inverse Laplace Transform:

$$f(t) = \frac{1}{2\pi i} \lim_{T\to\infty} \int_{\gamma-iT}^{\gamma+iT} e^{st} F(s)\, ds, \tag{17}$$

where γ is a real number and i is the unit of imagery part. In practice, we could decompose a Laplace transform into known transforms of functions in the Laplace table [5], which includes most of the commonly used Laplace transforms, and then construct the inverse transform.

With Laplace Transform, we convert the PID optimizer into its Laplace transformed functions of s, and then simplify the algebra. Once we find the transformed solution of $F(s)$, we can inverse the transform to obtain the required solution f as a function of t.

2.2. Evolution of Weight

A weight of a deep model is initialized as a scalar θ_0, and it is updated iteratively to reach its optimal value, denoted by θ_*. Then the process of each weight in DNN can be viewed as a step response (from θ_0 to θ_*) in control theory [4]. We then use the Laplace Transform as a guide to set hyper-parameter K_d.

The Laplace Transform of θ_* is $\frac{\theta_*}{s}$ [5]. We denote by $\theta(t)$ the weight at iteration t. The Laplace Transform of $\theta(t)$ is denoted as $\theta(s)$, and that of error $e(t)$ as $E(s)$. Since $E(s) = \frac{\theta_*}{s} -$ $theta(s)$. The Laplace transform of PID [5] is:

$$U(s) = (K_p + K_i\frac{1}{s} + K_d s)E(s) \tag{18}$$

In our case, the $u(t)$ corresponds to the update of $\theta(t)$. So we replace $U(s)$ with $\theta(s)$, and with $E(s) = \frac{\theta_*}{s} - \theta(s)$, Equation. 18 can be rewritten as:

$$\theta(s) = (K_p + K_i\frac{1}{s} + K_d s)(\frac{w_*}{s} - \theta(s)) \tag{19}$$

With this form, it is easy to derive a standard closed loop transfer function [1] as follows:

$$\frac{\theta_*}{s} - \theta(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{20}$$

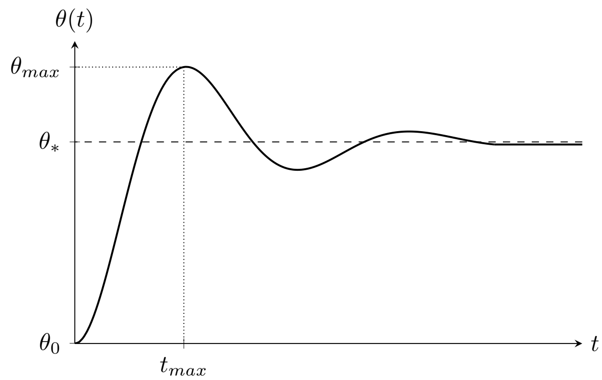

Figure 1. The evolution of the weight by PID optimizer

where

$$\begin{cases} \frac{K_p+1}{K_d} & = 2\zeta\omega_n \\ \frac{K_i}{K_d} & = \omega_n^2 \end{cases} \tag{21}$$

Equation. 20 can be rewritten as:

$$\frac{\theta_*}{s} - \theta(s) = \frac{(s+\zeta\omega_n) + \frac{\zeta}{\sqrt{1+\zeta^2}}\omega_n\sqrt{1-\zeta^2}}{(s+\zeta\omega_n)^2 + \omega_n^2(1-\zeta^2)} \tag{22}$$

We can get the time (iteration) domain form of $\theta(s)$ by using the Laplace Inverse Transform table [5] and the initial condition of the θ (θ_0):

$$\theta(t) = \theta_* - \frac{(\theta_* - \theta_0)\sin(\omega_n\sqrt{1-\zeta^2}t + \arccos(\zeta))}{e^{\zeta\omega_n t}\sqrt{1-\zeta^2}} \tag{23}$$

where ζ and ω_n are damping ratio and natural frequency of the system, respectively. In Fig. 1, we show the evolution process of a weight as an example of $\theta(t)$. From Equation. (21), we can write ζ as $\zeta = \frac{(K_p+1)^2}{4K_dK_i}$. One can see that K_i is a monotonically decreasing function of ζ. Refer to the definition of overshoot:

$$\textbf{Overshoot} = \frac{\theta_{max} - \theta_*}{\theta_*} \tag{24}$$

By differentiating $\theta(t)$ w.r.t. time t, and let $d\theta(t)/dt = 0$, we have the peak time of the weight as:

$$t_{max} = \frac{\pi}{w_n\sqrt{1-\zeta^2}} \tag{25}$$

Put t_{max} to Equation. 23, we have θ_{max}, and put θ_{max} to Equation. 24, we have:

$$\textbf{Overshoot} = \frac{\theta(t_{max}) - \theta_*}{\theta_*} = e^{\frac{-\zeta\pi}{\sqrt{1-\zeta^2}}} \tag{26}$$

One can see that ζ is monotonically decreasing with overshoot. Then K_i is a monotonically increasing function of overshoot. So more history error (Integral part), more overshoot the system will have. That is the reason why SGD-Momentum which accumulates past gradients will overshoot its target and spend more time during training.

As can be observed from Equation. (23), the term $\sin(\omega_n\sqrt{1-\zeta^2}t + \arccos(\zeta))$ brings periodically oscillation change to the weight, which is no more than 1. The term $e^{-\zeta\omega_n t}$ mainly controls the convergence rate. There is a hyper-parameter K_d in calculating the derivate $e^{-\zeta\omega_n} = e^{-\frac{K_p+1}{2K_d}}$. It is easy to observe that the larger the derivate, the earlier the training convergence we will reach. However, when K_d gets too large, the system will be fragile. In practice, we set the hyper-parameter K_d based on the Ziegler-Nichols optimum setting rule [6], which is widely used by engineers in PID feedback control since its origin in 1940s.

According to Ziegler-Nichols$'$ rule, the ideal setup of K_d should be one third of the oscillation period, which means $K_d = \frac{1}{3}T$, where T is the period of oscillation. From Equation. (23), we can get $T = \frac{2\pi}{\omega_n\sqrt{1-\zeta^2}}$. If we make a simplification that α in Momentum is equal to 1, then $K_i = K_d = r$. Combined with Equation. (21), K_d will have a closed form solution:

$$K_d = 0.25r + 0.5 + (1 + \frac{16}{9}\pi^2)/r \tag{27}$$

References

[1] H. K. Khalil. *Noninear Systems*. Prentice-Hall, New Jersey, 1996. 4323

[2] A. O'Dwyer. *Handbook of PI and PID controller tuning rules*. World Scientific, 2009. 4321

[3] N. Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999. 4321

[4] H. Rake. Step response and frequency response methods. *Automatica*, 16(5):519–526, 1980. 4323

[5] G. E. Robert and H. Kaufman. *Table of Laplace transforms*. Saunders, 1966. 4323, 4324

[6] J. G. Ziegler and N. B. Nichols. Optimum settings for automatic controllers. *trans. ASME*, 64(11), 1942. 4325