# Making "Alchemy" More Scientific (Part 8): Learning Rates for Multi-Stage Training

Su Jianlin

August 31, 2026

In the previous post Making "Alchemy" More Scientific (Part 7): Step-Size Scheduling and Weight Averaging, we briefly introduced work on schedule-free learning rates, which attempts to replace learning rate scheduling with a certain kind of weight averaging, so that a constant learning rate suffices to train a model to optimality. However, as we also noted in that post, without introducing additional assumptions, the optimal constant learning rate still depends on the total number of training steps, so truly schedule-free training cannot be achieved.

In this post, we revisit this problem from the perspective of multi-stage training. The main idea is to compromise on the scheduling objective—"being close to optimal at the end of every stage"—which makes it simpler and more feasible in practice.

## The Classical Result

We first start from a classical convergence result to understand why the optimal learning rate depends on the number of training steps. This convergence result was first introduced in Making "Alchemy" More Scientific (Part 2): Extending the Conclusion to Unbounded Domains:

$$\frac{\sum_{t=1}^T \eta_t \,\mathbb{E}[L(\boldsymbol{\theta}_t) - L(\boldsymbol{\theta}^*)]}{\sum_{t=1}^T \eta_t} \leq \frac{R^2}{2\sum_{t=1}^T \eta_t} + \frac{G^2}{2}\frac{\sum_{t=1}^T \eta_t^2}{\sum_{t=1}^T \eta_t} \triangleq f_T(\eta) \tag{1}$$

We will not repeat the precise meanings of the notations. This convergence result is representative, because both the later Making "Alchemy" More Scientific (Part 6): An Ingenious Top-Down Construction and the previous Making "Alchemy" More Scientific (Part 7): Step-Size Scheduling and Weight Averaging are built upon it.

What we want to do now is, given $R, G, T > 0$, find $\eta_1, \cdots, \eta_T$ that minimizes $f_T(\eta)$, which can be settled with a simple inequality. First, it is easy to show that $\sum_{t=1}^T \eta_t^2 \geq \left(\sum_{t=1}^T \eta_t\right)^2 / T$, hence

$$f_T(\eta) \geq \frac{R^2}{2\sum_{t=1}^T \eta_t} + \frac{G^2}{2T}\sum_{t=1}^T \eta_t \geq \frac{RG}{\sqrt{T}} \triangleq f_T^* \tag{2}$$

The condition for equality to hold in both parts simultaneously is $\eta_1 = \cdots = \eta_T = \frac{R}{G\sqrt{T}}$, which depends on $T$. If we instead adopt a learning rate schedule that is truly independent of $T$, such as $\eta_t \propto 1/\sqrt{t}$, then at best we can achieve $f_T(\eta) = \mathcal{O}(\log T/\sqrt{T})$, and it can further be shown that no matter how it is improved, a learning rate that does not depend on $T$ cannot achieve an $f$ better than $\Theta(\sqrt{\log T/T})$.

## Reframing the Problem

Now let us step back and think about the motivation behind Schedule-Free. Ideally, Schedule-Free hopes to keep training in a way that does not require knowing the total number of training steps $T$ in advance, so that no matter at which step we stop training, we obtain model weights that are nearly optimal up to the current step.

However, this perfect goal seems almost impossible. Even under the convex-optimization assumptions of Schedule-Free, where LR decay can be equated with weight averaging so that training proceeds with a constant learning rate, as analyzed in the previous section the optimal value of that constant learning rate still depends on $T$, and it cannot deliver the best solution up to the current step no matter when we stop, unless we further introduce additional assumptions.

Given this, let us reflect on what we really want from Schedule-Free. First, a constant learning rate is not a necessity: if some time-varying learning rate can train a good model, we can accept that as well. Second, "optimal whenever we stop" is certainly a nice property, but it is not really a strong requirement; what we mainly want is that, in multi-stage training, the endpoint of each stage approaches the optimum as closely as possible.

In other words, when our training is divided into multiple stages, we hope that at the end of each stage we obtain a model that is as close to optimal as possible. The number of stages will not be large, so for simplicity let us first consider two-stage training: the first stage trains for $T_1$ steps, the second stage trains for $T_2 - T_1$ steps, and $T_2$ steps are trained in total.

This can be further divided into two scenarios: "unexpected" and "planned". "Unexpected" means that continued training was not originally planned, so the first stage would be tuned to be as optimal as possible; once continued training is decided upon, the second stage then tries to be as good as possible on top of the existing state. This yields a greedy solution, whose drawback is that no matter how the second stage is tuned, the result remains relatively suboptimal. As for the "planned" scenario, the multi-stage training is planned in advance, and in theory we can balance the performance of the two stages better.

## Greedy Decision

Let us look at the greedy version first. The first stage takes the optimum, which is naturally the constant learning rate $\eta^*_{(1)} = \frac{R}{G\sqrt{T_1}}$, at which point the first stage's $f$ attains the theoretical optimum $f^*_{T_1}$. For the second stage, we need to find, on this basis, $\eta_{T_1+1}, \cdots, \eta_{T_2}$ that minimizes

$$f_{T_2}(\eta) = \frac{R^2 + G^2\left(T_1(\eta^*_{(1)})^2 + \sum_{t=T_1+1}^{T_2}\eta_t^2\right)}{2\left(T_1\eta^*_{(1)} + \sum_{t=T_1+1}^{T_2}\eta_t\right)} \tag{3}$$

Based on the same inequality $\sum_{t=T_1+1}^{T_2}\eta_t^2 \geq \left(\sum_{t=T_1+1}^{T_2}\eta_t\right)^2/(T_2-T_1)$, the minimum is again attained at a constant learning rate, so it simplifies to

$$f_{T_2}(\eta) = \frac{R^2 + G^2\left(T_1(\eta^*_{(1)})^2 + (T_2-T_1)(\eta_{(2)})^2\right)}{2\left(T_1\eta^*_{(1)} + (T_2-T_1)\eta_{(2)}\right)} \tag{4}$$

This can be accomplished either by constructing an appropriate inequality or by direct differentiation; the minimizer is

$$\eta^*_{(2)} = \frac{R/G}{\sqrt{T_2^{\#}}}, \qquad f(\eta^*_{(2)}) = \frac{RG}{\sqrt{T_2^{\#}}} \tag{5}$$

where

$$T_2^{\#} = \frac{1}{2}T_2 + \frac{1}{2}\sqrt{T_1(2T_2 - T_1)} \tag{6}$$

## Equivalent Steps

Here we have introduced the notation $T_2^{\#}$, whose meaning is that of "equivalent steps".

If there were no constraint from the first stage, we could have used the constant learning rate $\frac{R}{G\sqrt{T_2}}$ from the very beginning, and the minimum at the end of training would reach $\frac{RG}{\sqrt{T_2}}$. Now, however, we can only achieve $\frac{RG}{\sqrt{T_2^{\#}}}$, and the learning rate of the second stage is also of the form $\frac{R}{G\sqrt{T_2^{\#}}}$. This is equivalent to saying that the effective number of steps has changed from $T_2$ to $T_2^{\#}$. It is easy to verify that

$$T_2^{\#} \leq T_2 \tag{7}$$

That is, greedy two-stage training is effectively equivalent to losing a certain number of steps. Taking $T_2 = 2T_1$ as an example, $T_2^{\#} \approx 1.866\,T_1$, which is roughly a loss of 6.7% of the steps compared with $2T_1$. Note that if $T_2 \to \infty$, then $T_2^{\#}/T_2 \to 1/2$; that is, if training is continued indefinitely, the lost proportion keeps growing, up to a loss of one half.

Why distill such an abstract concept? Because "steps" is a universal concept, which makes the conclusions easier to transfer. The conclusions above are derived for SGD, but practical training usually uses non-SGD optimizers such as Adam and Muon, and hyperparameters such as the learning rate are not set directly from the formulas above, but are usually determined from a searched Scaling Law.

Therefore, the conclusions given by SGD cannot be applied directly. Yet if we can distill the relative changes through the concept of "equivalent steps", then the step-count input of the Scaling Law used in practice can likewise be replaced by the equivalent steps, yielding a heuristically corrected result. This is the main bridge from the theoretical conclusions of this article to practice.

## The Planned Scenario

Next we look at the "planned" scenario. Here both $T_1, T_2$ are known, and the learning rates of the two stages can be tuned jointly. The core issue is that we have two deliverables—the model at the end of the first stage and the final model—and we need to balance the performance of the two stages. To this end, we introduce the minimax objective:

$$\min_{\eta\geq 0} \max_{T\in\{T_1,T_2\}} \frac{f_T(\eta)}{f_T^*} \tag{8}$$

It asks that the ratio of the actual performance of each stage to its ideal value be as small as possible. The advantage of this objective is that it introduces no extra hyperparameters and is easy to generalize to multi-stage training. Introducing a new variable $m$, we can convert it into a nonlinear programming problem

$$\min\left\{m \;\middle|\; f_{T_1}(\eta)/f_{T_1}^* \leq m,\ f_{T_2}(\eta)/f_{T_2}^* \leq m,\ m\geq 1,\ \eta\geq 0\right\} \tag{9}$$

Non-dimensionalize the problem: let $x_t = G\eta_t/R$; then for any $T$, completing the square gives

$$\frac{f_T(\eta)}{f_T^*} = \frac{\sqrt{T}\left(1+\sum_{t=1}^T x_t^2\right)}{2\sum_{t=1}^T x_t} \leq m \iff \sum_{t=1}^T \left(x_t - \frac{m}{\sqrt{T}}\right)^2 \leq m^2 - 1 \tag{10}$$

i.e., the feasible set of $x_t$ forms a hypersphere centered at $\frac{m}{\sqrt{T}}\mathbf{1}$ with radius $\sqrt{m^2-1}$, from which we see that the original problem is equivalent to a quadratic program with $m$ as the objective.

## The Solution Process

For the two-stage case, this problem can be solved analytically. Note that the first constraint involves only $x_1, \cdots, x_{T_1}$, while the second involves all $T_2$ variables. For $x_t$ with $t > T_1$, they appear only in the second constraint, so we can simply take $x_t = m/\sqrt{T_2}$ so that their contribution is zero. The problem thus reduces to: the first $T_1$ values of $x_t$ must fall within both balls

$$\sum_{t=1}^{T_1}\left(x_t - \frac{m}{\sqrt{T_1}}\right)^2 \le m^2 - 1, \qquad \sum_{t=1}^{T_1}\left(x_t - \frac{m}{\sqrt{T_2}}\right)^2 \le m^2 - 1 \tag{11}$$

Both balls have radius $\sqrt{m^2-1}$, both centers lie in the direction of the all-ones vector, and the squared distance between the centers is

$$T_1\left(\frac{m}{\sqrt{T_1}} - \frac{m}{\sqrt{T_2}}\right)^2 = m^2\left(1-\sqrt{\tau}\right)^2, \qquad \tau = T_1/T_2 \tag{12}$$

The two balls intersect if and only if the distance between the centers does not exceed twice the radius, i.e., $m^2(1-\sqrt{\tau})^2 \le 4(m^2-1)$, from which the optimal $m$ is solved:

$$m^* = \frac{2}{\sqrt{4-(1-\sqrt{\tau})^2}} \tag{13}$$

The corresponding solution can be taken as the midpoint of the segment joining the two centers, i.e.,

$$x_t = \begin{cases} \dfrac{m^*}{2}\left(\dfrac{1}{\sqrt{T_1}} + \dfrac{1}{\sqrt{T_2}}\right), & t \le T_1 \\ \dfrac{m^*}{\sqrt{T_2}}, & t > T_1 \end{cases} \tag{14}$$

## Analysis of the Results

Restoring $\eta_t = Rx_t/G$ gives

$$\eta^*_{(1)} = \frac{m^* R}{2G}\left(\frac{1}{\sqrt{T_1}} + \frac{1}{\sqrt{T_2}}\right), \qquad \eta^*_{(2)} = \frac{m^* R}{G\sqrt{T_2}} \tag{15}$$

Interestingly, $\eta^*_{(1)}$ is exactly the average of the two single-stage optimal constant learning rates for "training only $T_1$ steps" and "training $T_2$ steps from scratch" (multiplied by $m^*$). From the perspective of equivalent steps, $\eta^*_{(1)}$ amounts to presupposing a larger number of training steps, while $\eta^*_{(2)}$ amounts to presupposing a smaller number of training steps:

$$\begin{aligned} T_1^{\#} &= \left(\frac{R}{G\eta^*_{(1)}}\right)^2 = T_1\left(\frac{3-\sqrt{\tau}}{1+\sqrt{\tau}}\right), \\ T_2^{\#} &= \left(\frac{R}{G\eta^*_{(2)}}\right)^2 = T_2\left(1 - \frac{(1-\sqrt{\tau})^2}{4}\right) \end{aligned} \tag{16}$$

That is to say, the first stage only needs to train for $T_1$ steps, but when setting its learning rate we assume that it will train for $T_1^{\#}$ steps; the two stages together train for $T_2$ steps, but when setting the learning rate of the second stage we assume that it will train for $T_2^{\#}$ steps. As for the outcome, $f_{T_1}/f_{T_1}^* = f_{T_2}/f_{T_2}^* = m^*$; that is, the relative losses of the two stages are exactly equal—this is precisely the equilibrium property of the minimax objective: it favors neither delivery point.

Again taking $T_2 = 2T_1$ as an example, we have $\tau = 1/2$. Substituting gives $T_2^{\#} \approx 0.979\,T_2$, so in terms of effect both stages lose about 2.1% of the steps compared with the optimal solution; averaged over the two stages, this is better than the greedy solution. For the learning rate settings, the first stage's learning rate should be set according to the equivalent steps $T_1^{\#} \approx 1.343\,T_1$, and the second stage's learning rate according to the equivalent steps $T_2^{\#} \approx 0.979\,T_2$.

Another limit is $\tau \to 0$, in which case $T_2^{\#} \to 0.75\,T_2$: when the second stage has sufficiently many steps, the minimax solution loses at most 25% of the steps at the endpoint, which is somewhat better than the 50% of the greedy solution.

## Extension to Multiple Stages

Whether it is the greedy solution in the "unexpected" scenario or the minimax solution in the "planned" scenario, both extend easily to multi-stage training in theory. The greedy solution goes without saying; let us expand mainly on the minimax solution. Suppose there are $K$ delivery points $T_1 < T_2 < \cdots < T_K$; the minimax objective becomes

$$\min_{\eta\geq 0} \max_{T\in\{T_1,\cdots,T_K\}} \frac{f_T(\eta)}{f_T^*} \tag{17}$$

The same non-dimensionalization gives $K$ ball constraints

$$\sum_{t=1}^{T_k} \left(x_t - \frac{m}{\sqrt{T_k}}\right)^2 \leq m^2 - 1, \qquad k = 1, 2, \cdots, K \tag{18}$$

Although there are theoretically $T_K$ variables, it can still be shown that the optimal solution can always be taken in a "piecewise constant" form, so the problem shrinks to a small-scale program with only $K+1$ variables. When $K \geq 3$ there is generally no concise closed-form solution, but solving it numerically is not difficult.

## Summary

This article re-examined "schedule-free learning rates" from the perspective of multi-stage training. By changing the scheduling objective to "being close to optimal at the end of each stage", the approach becomes simpler and more feasible in practice. Furthermore, we distilled the concept of "equivalent steps", so that the conclusions can potentially be transferred to the optimizers and Scaling Laws used in practice.