A Summary Sheet of Optimization in Deep Learning

Yifeng Liu
University of California, Los Angeles
liuyifeng@g.ucla.edu

Abstract

Newton's method provides one of the earliest insight in optimization theories. Based on gradient descent, a lot of optimization theories including momentum, adaptive learning, sign of gradient, second-order optimization, variance reduction and scheduler-free optimization have been proposed. However, there is a lack of comprehensive and clear summary of these approaches with unified notation system. This paper attempts to give a systematic, explicit and concise formulation of many of these methods with citation. I hope it can propose the innovation in optimization theory in deep learning, while facilitating relevant researchers to search for references. And the related materials can be found in `https://github.com/lauyikfung/A-Summary-Sheet-of-Optimization-in-Deep-Learning`. [1]

1 From Gradient Descent to Adaptive Learning

1.1 Notations

In the paper, $f(\theta_t)$ denotes the deep learning network at t-th iteration with parameter θ_t, The objective is $J(\theta_t)$ and the gradient is $\nabla_\theta J(\theta_t) = \nabla f(\theta_t, \boldsymbol{\xi}_t) = g_t$, where the input is $\boldsymbol{\xi}_t$ by default. Moreover, $\|\mathbf{x}\| = \|\mathbf{x}\|_2$ denotes the 2-norm, while $|\mathbf{x}|$ denotes taking the absolute value element-wisely. η is the learning rate or step size, and it may change over iterations unless specific explanation. We just omit the initialization of parameters and state variables for simplicity.

1.2 Newton's Method

For Newton's method, $f(\theta + \epsilon) = f(\theta) + \epsilon^\top \nabla f(\theta) + \frac{1}{2}\epsilon^\top \mathbf{H}\epsilon + \mathcal{O}(\|\epsilon\|^3)$, where $\mathbf{H} = \nabla^2 f(\theta)$ is the Hessian matrix (Zhang et al., 2021).

For first-order method, the second-order and higher-order terms ($\frac{1}{2}\epsilon^\top \mathbf{H}\epsilon + \mathcal{O}(\|\epsilon\|^3)$) are ignored since they have relatively lower influence on the convergence of training and much harder to compute, and the update rule comes to:

$$\theta_{t+1} = \theta_t - \eta \nabla f(\theta_t),$$

where $\eta = -\epsilon$ is the step size.

While in second-order methods, only the higher-order terms ($\mathcal{O}(\|\epsilon\|^3)$) are ignored. Since at the minimum of f, $\nabla_\theta f(\theta) = 0$, then $\epsilon = -\mathbf{H}^{-1}\nabla f(\theta)$, and it comes to:

$$\theta_{t+1} = \theta_t - \eta \mathbf{H}^{-1} \nabla f(\theta_t),$$

where η is the step size.

[1] According to the limitation of time, I may miss some excellent works on optimization in deep learning. I am very pleased to communicate with others and adding more interesting works to this sheet.

1.3 Gradient Descent (GD)

Full GD:

- Using full datasets for gradient descent, 1. $\theta_{t+1} = \theta_t - \eta\nabla_\theta J(\theta_t) = \theta_t - \eta g_t$. Here and below, g_t is defined the gradient of full data/mini batch data on θ_t.
- Convergence $O(\frac{1}{T})$. Here the convergence is defined by $f(x^T) - f^*$

Stochastic GD (SGD): Using one sample per step, convergence $O(\frac{1}{\sqrt{T}})$

Batch GD (BGD): Using small batch (size=b) per steps, convergence $O(\frac{1}{\sqrt{bT}} + \frac{1}{T})$

1.4 Momentum and Related

Momemtum (Grum, 2023) $m_{t+1} = \gamma m_t + \eta\nabla_\theta J(\theta_t)$, $\theta_{t+1} = \theta_t - m_{t+1}$

Nestorov's Accelerated Gradient (NAG) (Nesterov, 1983) $m_{t+1} = \gamma m_t + \eta\nabla_\theta J(\theta_t - \gamma m_t)$, $\theta_{t+1} = \theta_t - m_{t+1}$

BFGS (Broyden–Fletcher–Goldfarb–Shanno) (Fletcher, 1987) $d_t = -H_t g_t$, $\alpha_t = \arg\min_\alpha f(\theta - \alpha d_t)$, $\theta_{t+1} = \theta_t + \alpha_t d_t$, $H_{t+1} = (I - \frac{s_t y_t}{y_t^\top s_t})^\top H_t (I - \frac{y_t s_t^\top}{y_t^\top s_t}) + \frac{s_t s_t^\top}{y_t^\top s_t}$, where $s_t = \theta_{t+1} - \theta_t$, $y_t = g_{t+1} - g_t$.

L-BFGS (Liu & Nocedal, 1989) Based on BFGS, but choose α_t satisfying Wolfe conditions (try $\alpha_t = 1$ first): $f(\theta_t + \alpha_t d_t) \le f(\theta_t) + \beta' \alpha_t g_t^\top d_t$. Moreover, for $\widehat{m} = \min\{t, m-1\}$,

$$H_{t+1} = (\prod_{i=t,inv}^{t-\widehat{m}} V_i^\top) H_0 (\prod_{i=t-\widehat{m}}^{t} V_i) + \sum_{j=t-\widehat{m}}^{t} \rho_j (\prod_{i=t,inv}^{j+1} V_i^\top) s_j s_j^\top (\prod_{i=j+1}^{t} V_i),$$

where $\prod_{i=t,inv}^{t'}$ denotes the product of matrices with indices from t to $t' < t$, and $\rho_t = 1/(y_t^\top s_t)$.

ASGD (Polyak & Juditsky, 1992) $\theta_{t+1} = \theta_t - \eta \cdot \frac{1}{t+1}\sum_{i=1}^t g_i$

AdaGrad (Duchi et al., 2011) $r_t = r_{t-1} + g_t^2$, $\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{r_t + \epsilon}} g_t$

AdaGrad-Norm (Ward et al., 2020) $r_t = r_{t-1} + \|g_t\|^2$, $\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{r_t + \epsilon}} g_t$

AdaDelta (Zeiler, 2012) $v_t = \rho v_{t-1} + (1-\rho) g_t^2$, $\theta_t = \theta_{t-1} - \eta\frac{\sqrt{u_{t-1}}}{\sqrt{v_t + \epsilon}} g_t$,$u_t = \rho u_{t-1} + (1-\rho)\Delta\theta_t^2$, where $\Delta\theta_t = \theta_t - \theta_{t-1}$

RMSProp (Tieleman & Hinton, 2012) $v_t = \rho v_{t-1} + (1-\rho) g_t^2$ (EMA (Exponential Moving Average) of the squared gradient), $\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t + \epsilon}} g_t$

2 Adam and derivatives

Adam (Kingma & Ba, 2015)

- $m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$ (EMA of gradient, use m_t below if have same expression)
- $v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$ (EMA of squared gradient, use v_t below if have same expression)
- $\theta_{t+1} = \theta_t - \eta\frac{\widehat{m}_t}{\sqrt{\widehat{v}_t + \epsilon}}$, where $\widehat{m}_t = \frac{m_t}{1-\beta_1^t}$, $\widehat{v}_t = \frac{v_t}{1-\beta_2^t}$ (Use $\widehat{m_t}$ or $\widehat{v}_t$ below if have same expression)

2.1 Other forms for Adam

2.1.1 Hessian Matrices

$\theta_{t+1} = \arg\min_\theta\{\eta\langle m_t, \theta\rangle + \frac{1}{2}||\theta - \theta_t||^2_{\mathbf{H}_t}\}$. In closed form: $\theta_{t+1} = \theta_t - \eta\mathbf{H}_t^{-1} m_t$.

- $\mathbf{H}_t = \sqrt{\text{diag}(v_t)} \cdot \frac{1-\beta_1^t}{\sqrt{1-\beta_2^t}}$ for AdamW

- $\mathbf{H}_t = \sqrt{\text{diag}(m_t^2)}$ for Lion
- $\mathbf{H}_t = (\sum_{\tau=1}^t \mathbf{G}_t \mathbf{G}_t^\top)^{1/4} \otimes (\sum_{\tau=1}^t \mathbf{G}_t^\top \mathbf{G}_t)^{1/4}$ for Shampoo

2.1.2 Matrix form of v_t

$\theta_{t+1} = \prod_{\mathcal{F},\sqrt{V_t}}(\theta_t - \eta_t m_t/\sqrt{\widehat{v}_t})$, where $\theta \in \mathcal{F}, V_t = \text{diag}(\widehat{v}_t)$

2.2 Derivatives of Adam

In the following algorithms, if not specified, m_t, v_t, $\widehat{m}_t$, $\widehat{v}_t$ and the updating rule are the same with Adam, or AdamW with decoupled weight decay.

2.2.1 Minute Modification

AdamW (Loshchilov & Hutter, 2019) $\theta_{t+1} = \theta_t - \eta(\frac{\widehat{m}_t}{\sqrt{\widehat{v}_t}+\epsilon} + \lambda\theta_t)$, where λ is called decoupled weight decay (hyper-parameter). And The decoupled weight decay can be applied to the algorithms below.

AMSgrad (Reddi et al., 2018) $\widetilde{v}_t = \max(\widetilde{v}_{t-1}, v_t)$, $\widehat{v}_t = \frac{\widetilde{v}_t}{1-\beta_2^t}$

AdaMax (Loshchilov & Hutter, 2019) $u_t = \max(\beta_2 u_{t-1}, |g_t|)$, $\theta_{t+1} = \theta_t - \eta\frac{\widehat{m}_t}{u_t}$

Yogi (Zaheer et al., 2018) $v_t = v_{t-1} - (1-\beta_2)\text{sign}(v_{t-1} - g_t^2)g_t^2$

AdamX (Tran et al., 2019) $\widehat{m}_t = m_t$, $\widehat{v}_t = \max\{\frac{(1-\beta_{1,t})^2}{(1-\beta_{1,t-1})^2}\widehat{v}_{t-1}, v_t\}$

NAdam (Dozat, 2016) $\theta_{t+1} = \theta_t - \eta\frac{1}{\sqrt{\widehat{v}_t}+\epsilon}(\beta_1\widehat{m}_t + \frac{1-\beta_1}{1-\beta_1^t}g_t)$, by taking $\psi = 0$ in formula below 1. In PyTorch implementation, $\theta_{t+1} = \theta_t - \eta\frac{1}{\sqrt{\widehat{v}_t}+\epsilon}(\mu_{t+1}\frac{m_t}{1-\Pi_{i=1}^{t+1}\mu_i} + \frac{(1-\mu_t)}{1-\Pi_{i=1}^{t}\mu_i}g_t)$, where $\mu_t = \beta_1(1 - \frac{0.96^{t\psi}}{2})$

Padam (Chen & Gu, 2018) $\widetilde{v}_t = \max(\widetilde{v}_{t-1}, v_t)$, $\theta_{t+1} = \theta_t - \eta_t\frac{m_t}{\widetilde{v}_t^p}$. And the output is chosen from $\{\theta_t\}$ with $P(\theta_{out} = \theta_t) = \frac{\eta_{t-1}}{\sum_{i=1}^{T-1}\eta_i}$

RAdam (Liu et al., 2020a) $\rho_\infty = \frac{2}{1-\beta_2} - 1$, $\rho_t = \rho_\infty - \frac{2t\beta_2^t}{1-\beta_2^t}$. If $\rho_t > 4$, $l_t = \sqrt{(1-\beta_2^t)/v_t}$, $r_t = \sqrt{\frac{(\rho_t-4)(\rho_t-2)\rho_\infty}{(\rho_\infty-4)(\rho_\infty-2)\rho_t}}$, $\theta_{t+1} = \theta_t - \eta r_t\widehat{m}_t l_t$; otherwise $\theta_{t+1} = \theta_t - \eta\widehat{m}_t$.

Adam$^+$ (Liu et al., 2020b) $\eta_t = \frac{\alpha\beta^\gamma}{\max(||z_t||^{1/2}, \epsilon_0)}$, $\theta_{t+1} = \theta_t - \eta z_t$, $z_{t+1} = (1-\beta)z_t + \beta\nabla_\theta J((1-\frac{1}{\beta})\theta_t + \frac{1}{\beta}\theta_{t+1})$, where I replace a in the original papar with γ for clarity

AdaX (Li et al., 2020) $v_t = (1+\beta_2)v_{t-1} + \beta_2 g_t^2$, $\widehat{v}_t = \frac{v_t}{(1+\beta_2)^t - 1}$

AdaBelief (Zhuang et al., 2020) $v_t = \beta_2 v_{t-1} + (1-\beta_2)(g_t - m_t)^2 + \epsilon$

AdamWN (Loshchilov, 2023) $\widehat{\theta}_t = \theta_t - \eta\frac{\widehat{m}_t}{\sqrt{\widehat{v}_t}+\epsilon}$, $\theta_{t+1} = \widehat{\theta}_t - k_t(1 - \frac{r_t\|\theta_0\|}{\|\widehat{\theta}_t\|})\widehat{\theta}_t$, $k_t \in [0,1]$, $r_t\|\theta_0\|$ is the target weight norm for θ_t

C-Adam (Liang et al., 2024) $u_t = \frac{\widehat{m}_t}{\sqrt{\widehat{v}_t}+\epsilon}$, $\phi_t = \mathbf{1}_{u_t \circ g_t \geq 0}$, $\bar{\eta}_t = \eta\frac{d}{\|\phi_t\|_0 + 1}$, $\theta_{t+1} = \theta_t - \bar{\eta}_t(\phi_t \circ u_t + \lambda\theta_t)$

2.2.2 Clipping Involved

LARS (You et al., 2017) $m_t = \beta_1 m_{t-1} + (1-\beta_1)(g_t + \lambda\theta_t)$, $\theta_{t+1} = \theta_t - \eta\frac{\text{Clip}(||\theta_t||, \gamma_l, \gamma_u)}{||\theta_t|| + \epsilon}m_t$, where $\text{Clip}(x, \gamma_l, \gamma_u) = \min(\max(x, \gamma_l), \gamma_u)$, γ_l is default to 0 if only 1 bound given

LAMB (You et al., 2020) $r_t = \frac{\widehat{m}_t}{\sqrt{\widehat{v}_t}+\epsilon}$, $\theta_{t+1} = \theta_t - \eta\frac{\text{Clip}(||\theta_t||, \gamma_l, \gamma_u)}{||r_t + \lambda\theta_t|| + \epsilon}(r_t + \lambda\theta_t)$

MARS-AdamW (Yuan et al., 2024) AdamW with variance reduction. See Section 5 for detail.

2.2.3 More EMA Involved

Prodigy (Bernstein & Newhouse, 2024) $m_t = \beta_1 m_{t-1} + (1-\beta_1)\eta_t g_t$, $v_t = \beta_2 v_{t-1} + (1-\beta_2)\eta_t^2 g_t^2$, $r_t = \sqrt{\beta_2} r_{t-1} + (1-\sqrt{\beta_2})\eta_t^2 g_t^\top(\theta_0 - \theta_t)$, $s_t = \sqrt{\beta_2} s_{t-1} + (1-\sqrt{\beta_2})\eta_t^2 g_t$, $\eta_{t+1} = \max(\eta_t, \frac{r_t}{\|s_t\|_1})$, $\theta_{t+1} = \theta_t - \eta_t \frac{m_t}{\sqrt{v_t}}$

Adan (Xie et al., 2024) $v_t = (1-\beta_2)v_{t-1} + \beta_2(g_t - g_{t-1})$, $n_t = (1-\beta_3)n_{t-1} + \beta_3[g_t + (1-\beta_2)(g_t - g_{t-1})]^2$, $\theta_t = \frac{1}{1+\lambda\eta}[\theta - \frac{\eta}{\sqrt{n_t+\epsilon}} \circ (m_t + (1-\beta_2)v_t)]$

AdEMAMix (Pagliardini et al., 2024) $r_t = \beta_3 r_{t-1} + (1-\beta_3)g_t$, $\theta_t = \theta_{t-1} - \eta(\frac{\widehat{m}_t + \alpha_t r_t}{\sqrt{\widehat{v}_t}+\epsilon} + \lambda\theta_{t-1})$

2.2.4 More complicated Design

Adafactor (Shazeer & Stern, 2018) $u_t = \frac{g_t}{\sqrt{\widehat{v}_t}}$, $\widehat{u}_t = \frac{u_t}{\max(1, \mathrm{RMS}(u_t)/d)}$, $\theta_{t+1} = \theta_t - \eta_t \widehat{u}_t$, where $\alpha_t = \eta \cdot \max(\epsilon_2, \mathrm{RMS}(\theta_t))$, $\beta_{2,t} = 1 - t^\tau$, and

- For weight vector $\theta_t \in \mathbb{R}^n$, $\widehat{v}_t = \beta_{2,t}\widehat{v}_{t-1} + (1-\beta_{2,t})(g_t^2 + \epsilon_1 1_n)$;
- For weight matrix $\theta_t \in \mathbb{R}^{n\times m}$, $r_t = \beta_{2,t} r_{t-1} + (1-\beta_{2,t})(g_t^2 + \epsilon_1 1_n 1_m^\top)1_m$, $v_t = \beta_{2,t} v_{t-1} + (1-\beta_{2,t})1_n^\top(g_t^2 + \epsilon_1 1_n 1_m^\top)$, $\widehat{v}_t = r_t v_t / 1_n^\top r_t$.

Amos (Tian & Parikh, 2022) $c_t = (1 + \frac{1}{4}\sqrt{\eta} b_t)^{-1/2}$, $d_t = (1 + \frac{1}{4}\sqrt{\eta\widetilde{\eta}} b_t)^{-1}$, with $\widetilde{\eta}$ the expected scale for model weights θ, (Optional) $\widetilde{g}_t = \frac{\chi}{\max(\chi, \|g_t\|)} g_t$, $v_t = \beta v_{t-1} + (1-\beta)\mathrm{M}_2(\widetilde{g}_t)^2$, where $\mathrm{M}_2(a) := \sqrt{\frac{1}{k}\sum_{i=1}^k a_i^2}$ is the quadratic mean of the entries. $\widehat{v}_t = \frac{v_t}{1-\beta^t}$, $\gamma_t = c_t \frac{\eta^2}{\widehat{v}_t}\mathrm{M}_2(g_t)^2$, $\delta_t = d_t(\frac{\eta\widetilde{\eta}}{\sqrt{\widehat{v}_t}} g_t + \frac{\gamma_t}{2}\theta_t)$. $b_{t+1} = b_t + \gamma_t(1 + b_t)$, (Optional) $\widetilde{\delta}_t = m_{t+1} = \mu m_t + (1-\mu)\delta_t$, $\theta_{t+1} = \theta_t - \widetilde{\delta}_t$

3 Sign of gradient

RProp (Riedmiller & Braun, 1993) for $0 < \eta^- < 1 < \eta^+$,

- If $g_t \cdot g_{t-1} > 0$, $\eta_t = \min(\eta^+ \eta_{t-1}, \eta_{max})$, $\theta_{t+1} = \theta_t - \eta_t \mathrm{Sign}(g_t)$
- If $g_t \cdot g_{t-1} < 0$, $\eta_t = \max(\eta^- \eta_{t-1}, \eta_{min})$, $\theta_{t+1} = \theta_t - \Delta\theta_t$, where $\Delta\theta_t = \theta_t - \theta_{t-1}$
- Otherwise $\eta_t = 1$, $\theta_{t+1} = \theta_t - \eta_t \mathrm{Sign}(g_t)$

SignSGD (Bernstein et al., 2018) $\theta_{t+1} = \theta_t - \eta \mathrm{Sign}(g_t)$

Signum (Bernstein et al., 2018) $m_t = \beta m_{t-1} + (1-\beta)g_t$,$\theta_{t+1} = \theta_t - \eta \mathrm{Sign}(m_t)$

Lion (Chen et al., 2023) $c_t = \beta_1 m_{t-1} + (1-\beta_1)g_t$, $m_t = \beta_2 m_{t-1} + (1-\beta_2)g_t$, $\theta_{t+1} = \theta_t - \eta(\mathrm{Sign}(c_t) + \lambda\theta_t)$

- **Tiger** (Su, 2023) (A special case of Lion when $\beta_1 = \beta_2 = \beta$) $m_t = \beta m_{t-1} + (1-\beta)g_t$, $\theta_{t+1} = \theta_t - \eta(\mathrm{Sign}(m_t) + \lambda\theta_t)$. For bias and normalization parameters, $\eta_i = 0.5\eta$, $\lambda = 0$, otherwise $\eta_i = \eta \times RMS(\theta_{t,i})$, $\lambda =$ Constant> 0
- **MARS-Lion** (Yuan et al., 2024) Lion with variance reduction. See Section 5 for detail.

Grams (Cao et al., 2024) (Based on AdamW) $\theta_{t+1} = \theta_t - \eta(\mathrm{Sign}(g_t) \circ |\frac{\widehat{m}_t}{\sqrt{\widehat{v}_t}+\epsilon}| + \lambda\theta_t)$

4 Second-Order Methods

$\theta_{t+1} = \theta_t - (\nabla^2_{\theta_t} J(\theta))^{-1}\nabla_{\theta_t} J(\theta_t) = \theta_t - \mathbf{H}_t^{-1} g_t$.

To estimate $\mathbf{H}_t$, Hutchinson's method (Hutchinson, 1989) can be used: $z \sim$ Rademacher(0.5), $\frac{\partial g^\top z}{\partial\theta} = \frac{\partial g^\top}{\partial\theta} z + g^\top \frac{\partial z}{\partial\theta} = \frac{\partial g^\top}{\partial\theta} z = \mathbf{H}_t z$, $\mathbf{D} = \mathrm{diag}(\mathbf{H}) = \mathbb{E}[z \odot (\mathbf{H}z)]$.

Another way to estimate Hessian matrix is to compute Fisher information matrix (proposed by **H-FAC** (Martens & Grosse, 2015)):

$$F = E[\frac{d \log p(y|x,\theta)}{d\theta}(\frac{d \log p(y|x,\theta)}{d\theta})^\top] = E[\mathcal{D}\theta\mathcal{D}\theta^\top]$$

AdaHessian (Yao et al., 2021) (Based on Adam) $v_t = \beta_2 v_{t-1} + (1-\beta_2)\mathbf{D}_t^2$, $\theta_{t+1} = \theta_t - \eta\frac{\widehat{m}_t}{(\sqrt{\widehat{v}_t})^k+\epsilon}$, where k is the Hessian power

Hessian-free (Martens et al., 2010) Define the function $B_n(d) = \mathbf{H}(\theta_t)d + \lambda d$, where $\mathbf{H}(\theta_t)d = \lim_{\epsilon\to 0}\frac{\nabla f(\theta_t+\epsilon d)-g_t}{\epsilon}$. p_t = CG-Minimize$(B_t, -g_t)$, where CG-Minimize is the linear conjugate gradient algorithm, $\theta_{t+1} = \theta_t + p_t$. λ can be adjusted by Levenberg-Marquardt style heuristic: for $\rho_t = \frac{f(\theta_t+p)-f(\theta_t)}{q_{\theta_t}(p)-q_{\theta_t}(0)}$ with $q_{\theta_t}(\cdot)$ the minimization objective of CG, if $\rho_t < \frac{1}{4}, \lambda \to \frac{3}{2}\lambda$, else if $\rho_t > \frac{3}{4}, \lambda \to \frac{2}{3}\lambda$, otherwise keep λ unchanged.

Shampoo (Gupta et al., 2018) $L_t = L_{t-1} + g_t g_t^\top$, $R_t = R_{t-1} + g_t^\top g_t$, $\theta_{t+1} = \theta_t - \eta L_t^{-1/4} g_t R_t^{-1/4}$

- Proposed by Bernstein & Newhouse (2024), $\Delta\theta = -\eta(gg^\top)^{-1/4}g(g^\top g)^{-1/4} = -\eta UV^\top$, where $g = U\Sigma V^\top$ is the SVD decomposition of the gradient. To approximate this process, they developed Newton-Schulz iteration (Higham, 2008) methods for computing $UV^\top$, by setting $X_0 = g/\|g\|_{l_2\to l_2}$ or $X_0 = g/\|g\|_F$, where $\|M\|_{\alpha\to\beta} := \max_x \frac{\|Mx\|_\beta}{\|x\|_\alpha}$ is the induced operator norm. Then $UV^\top$ can be approximated by several iteration of $X_{t+1} = \frac{3}{2}X_t - \frac{1}{2}X_t X_t^\top X_t$.
- MARS-Shampoo (Yuan et al., 2024): Shampoo with variance reduction. See Section 5 for detail.

SOAP (Vyas et al., 2024) (Only for 2D parameters) $g_t' = Q_L^\top g Q_R$, $m_t = \beta_1 m_{t-1} + (1-\beta_1)g_t$, $m_t' = Q_L^\top m_t Q_R$, $v_t = \beta_2 v_{t-1} + (1-\beta_2)(g_t')^2$, $\theta_{t+1} = \theta_t - \eta Q_L\frac{\widehat{m}_t'}{\sqrt{\widehat{v}_t}+\epsilon}Q_R^\top$, where $\widehat{m}_t' = \frac{m_t'}{1-\beta_1^t}$, $\widehat{v}_t = \frac{v_t}{1-\beta_2^t}$. $L_t = \beta_2 L_{t-1} + (1-\beta_2)gg^\top$, $R_t = \beta_2 R_{t-1} + (1-\beta_2)g^\top g$. For every k steps, obtain Q_L =QR-Eigenvectors(LQ_L), Q_R =QR-Eigenvectors(RQ_R), where QR-Eigenvectors returns the eigenvector matrix of the QR decomposition for the input matrix.

Muon (Jordan et al., 2024) (Only for 2D parameters) $m_t = \mu m_{t-1} + g_t$, O_t = NewtonSchulz5(m_t), $\theta_{t+1} = \theta_t - \eta O_t$, where NewtonSchulz5 is the Newton-Schulz methods explained above for 5 iterations with $X_{t+1} = aX_t + bX_t X_t^\top X_t + c(X_t X_t^\top)^2 X_t$ for $(a,b,c) = (3.4445, -4, 7750, 2.0315)$.

Sophia (Liu et al., 2024) $m_t = \beta_1 m_{t-1} + (1-\beta_1)g_t$, if t mod $k = 1$ for the size of the step group k, $h_t = \beta_2 h_{t-k} + (1-\beta_2)\widehat{h}_t$, where $\widehat{h}_t$ is calculated using one of the following Hessian estimators:

- Hutchinson: Draw $u \sim \mathcal{N}(0, I_d)$, $\widehat{h}_t = u \odot \nabla(\langle g_t, u\rangle)$;
- Gauss-Newton-Bartlett: Compute logits on mini-batch $\{l(\theta_t, \boldsymbol{\xi}_{t,b})\}_{b=1}^B$, sample $\widehat{y}_{t,b} \sim$ Softmax$(l(\theta_t, \boldsymbol{\xi}_{t,b})), \forall b \in [B]$. $\widehat{g} = \frac{1}{B}\nabla L(l(\theta_t, \boldsymbol{\xi}_{t,b}), \widehat{y}_{t,b})$ with loss function l, $\widehat{h} = B \cdot \widehat{g} \odot \widehat{g}$,

Otherwise, $h_t = h_{t-1}$. $\theta_{t+1} = \theta_t - \eta(\text{Clip}(\frac{m_t}{\max\{\gamma h_t, \epsilon\}}, 1) - \lambda\theta_t)$

5 Variance Reduction

SAG (Roux et al., 2012) $\theta_{t+1} = \theta_t - \frac{\eta}{n}\sum_{i=1}^n y_{i,t}$, where at each t, one $\boldsymbol{\xi}_{i_t}$ is chosen and $y_{i,t} = \nabla f(\theta_t, \boldsymbol{\xi}_{i,t})$ and for other samples, keep $y_{\cdot,t}$ unchanged.

SDCA (Shalev-Shwartz & Zhang, 2013) (For machine learning task $L(\theta) = \frac{1}{n}\sum_{i=1}^n \phi_i(\theta^\top\boldsymbol{\xi}_i) + \frac{\lambda}{2}\|\theta\|^2$ with scalar convex function ϕ_i), for $\boldsymbol{\xi}_t = \boldsymbol{\xi}_i$, $\Delta\alpha_t = \arg\max_{\Delta\alpha} -\phi_i^*(-(\alpha_{t-1}+\Delta\alpha), \boldsymbol{\xi}_t) - \frac{\lambda n}{2}\|\theta_{t-1} + \frac{1}{\lambda n}\boldsymbol{\xi}_t\Delta\alpha\|^2$, $\alpha_t = \alpha_{t-1} + \Delta\alpha_t e_i$, $\theta_t = \theta_{t-1} + (\lambda n)^{-1}\boldsymbol{\xi}_t\Delta\alpha_t$. Output $\frac{1}{T}\sum_{i=1}^T \theta_{t-1}$ or randomly chosen from $\{\theta_t\}_{t=1}^T$.

SAGA (Defazio et al., 2014) Keep a $n \times d$ matrix $\phi = \{\nabla f(\theta_0, \boldsymbol{\xi}_i)\}_{i=1}^n$ storing the parameters, for $\boldsymbol{\xi}_t = \boldsymbol{\xi}_j$, $\theta_{t+1} = \arg\min_\theta \{h(\theta) + \frac{1}{2\gamma}\|\theta - (\theta_t - \gamma[\nabla f(\theta_t, \boldsymbol{\xi}_t) - \phi_j + \frac{1}{n}\sum_{i=1}^n \phi_i])\|^2\}$, where $h(\theta)$ is the regularization function, then update $\phi_j = \theta_t$ and keep other ϕ_i unchanged.

SVRG (Johnson & Zhang, 2013) For each epoch s, calculate the full gradient g_s for θ_s. For each iteration t, $\theta_{s,t+1} = \theta_{s,t} - \eta(\nabla f(\theta_{s,t}, \boldsymbol{\xi}_{s,t}) - \nabla f(\theta_s, \boldsymbol{\xi}_{s,t}) + g_s)$, set $\theta_s = \theta_{s,T}$ or randomly chosen from $\{\theta_{s,t}\}_{t=1}^T$

SARAH (Nguyen et al., 2017) For each epoch s, calculate the full gradient $v_{s,0} = g_s$ for θ_s, $\theta_{s,1} = \theta_{s,0} - \eta v_{s,0}$. For each iteration t, $v_{s,t} = \nabla f(\theta_{s,t}, \boldsymbol{\xi}_{s,t}) - \nabla f(\theta_s, \boldsymbol{\xi}_{s,t}) + v_{s,t-1}$, $\theta_{s,t+1} = \theta_{s,t} - \eta v_{s,t}$, set θ_s randomly chosen from $\{\theta_{s,t}\}_{t=1}^T$.

Hybrid-SGD (Tran-Dinh et al., 2019) (Hybrid SGD with SARAH) $m_t = \beta(m_{t-1} + g_t - \nabla f(\theta_{t-1}, \boldsymbol{\xi}_t)) + (1-\beta)g_t$, $\theta_{t+1} = \theta_t - \eta m_t$

SPIDER-SFO (Fang et al., 2018) $v_t = g_t$ for every q steps with batch size S_1, otherwise $v_t = g_t - \nabla f(\theta_{t-1}, \boldsymbol{\xi}_t) + v_{t-1}$. Update parameters with Option I: $\theta_{t+1} = \theta_t - \eta v_t/\|v_t\|$ until $\|v_t\|$ less than some threshold, or Option II: $\theta_{t+1} = \theta_t - \eta_t v_t$ with $\eta_t = \min(\eta/\|v_t\|, \eta/(2\epsilon))$. Output θ_T or randomly choose from $\{\theta_t\}_{t=1}^T$.

SpiderBoost (Wang et al., 2019) $v_t = g_t - \nabla f(\theta_{t-1}, \boldsymbol{\xi}_t) + v_{t-1}$, $\theta_{t+1} = \theta_t - \eta v_t/\|v_t\|$

SNVRG (Zhou et al., 2020) For loop parameters $\{T_l\}$, set $T = \prod_{l=1}^K T_l$ or $T \sim \text{Geom}(1/(1 + \prod_{l=1}^K T_l))$ as the total number of steps. For each step t, $r_t = \min\{j : 0 = (t \bmod \prod_{l=j+1}^K T_l), 0 \le j \le K\}$. For $0 \le l \le r_t - 1$, $\theta_t^l = \theta_{t-1}^l$; otherwise $\theta_t^l = \theta_t$. For $0 \le l \le r_t - 1$, $g_t^l = g_{t-1}^l$; for $r_t + 1 \le l \le K$, $g_t^l = 0$. Then uniformly generate index set I_t with size B_{r_t}, if $r_t > 0$, $g_t^{r_t} = \frac{1}{B_{r_t}}\sum_{i\in I_t}[\nabla f(\theta_t^{r_t}, \boldsymbol{\xi}_i) - \nabla f(\theta_t^{r_t-1}, \boldsymbol{\xi}_i)]$, otherwise $g_t^0 = \frac{1}{B_0}\sum_{i\in I_t}\nabla f(\theta_t^0, \boldsymbol{\xi}_i)$. And $\theta_{t+1} = \theta_t - \eta\sum_{l=0}^K g_t^l$. The output parameter is randomly chosen from $\{\theta_t\}_{t=1}^T$.

STORM (Cutkosky & Orabona, 2019) $\eta_t = \frac{k}{(w+\sum_{i=1}^t \|g_i\|^2)^{1/3}}$, $a_t = c\eta_{t-1}^2$, $\theta_{t+1} = \theta_t - \eta_t d_t$, where $d_t = g_t + (1-a_t)(d_{t-1} - \nabla f(\theta_{t-1}, \boldsymbol{\xi}_t))$. Output is chosen uniformly at random from $x_1, ..., x_T$ (in practice x_T)

STORM$^+$ (Levy et al., 2021) (Based on STORM) $a_{t+1} = \frac{1}{(1+\sum_{i=1}^t \|g_i\|^2)^{2/3}}$, $\eta_t = \frac{1}{(\sum_{i=1}^t \|d_i\|^2/a_{i+1})^{1/3}}$

Super-Adam (Huang et al., 2021) (Based on Adam) $c_t = \alpha_t g_t + (1-\alpha_t)[c_{t-1} + \tau(g_t - \nabla f(\theta_{t-1}, \boldsymbol{\xi}_t))]$, $\tau \in \{0, 1\}$, $\widetilde{\theta}_t = \arg\min_\theta \{\eta\langle c_t, \theta\rangle + \frac{1}{2}||\theta - \theta_t||_{\mathbf{H}_t}^2\}$, $\theta_{t+1} = (1-\mu_t)\theta_t + \mu_t\widetilde{\theta}_t$, where $\mathbf{H}_t$ is defined by one of the following cases:

- Case 1: $\mathbf{H}_t = \text{diag}(\sqrt{v_t} + \lambda)$
- Case 2: $v_t = \beta v_{t-1} + (1-\beta)\|g_t\|$, $\mathbf{H}_t = (v_t + \lambda)I_d$
- Case 3 (Barzilai-Borwein technique): $b_t = \frac{\|\langle g_t - \nabla f(\theta_{t-1}, \boldsymbol{\xi}_t), \theta_t - \theta_{t-1}\rangle\|}{\|\theta_t - \theta_{t-1}\|}$, $\mathbf{H}_t = (b_t + \lambda)I_d$
- Case 4-1: $v_t = \beta_2 v_{t-1} + (1-\beta_2)(g_t - m_t)^2$, $\mathbf{H}_t = \text{diag}(\sqrt{v_t} + \lambda)$
- Case 4-2: $v_t = \beta_2 v_{t-1} + (1-\beta_2)\|g_t - m_t\|$, $\mathbf{H}_t = (v_t + \lambda)I_d$

ROOT-SGD (Li et al., 2022) $v_t = g_t + \frac{t-1}{t}(v_{t-1} - \nabla f(\theta_{t-1}, \boldsymbol{\xi}_t))$, $\theta_{t+1} = \theta_t - \eta v_t$

AdaSVRPS/AdaSVRLS (Jiang & Stich, 2024) $F_{\boldsymbol{\xi}_t}(\theta) = f(\theta, \boldsymbol{\xi}_t) + \mathbf{w}^\top(\nabla f(\mathbf{w}_t) - \nabla f(\mathbf{w}_t, \boldsymbol{\xi}_t)) + \frac{\mu_F}{2}\|\theta - \theta_t\|^2$, $\theta_{t+1} = \theta_t - \eta_t\nabla_\theta F_{\boldsymbol{\xi}_t}(\theta_t)$. With probability p_{t+1}, $\mathbf{w}_{t+1} = \theta_t$, otherwise $\mathbf{w}_{t+1} = \mathbf{w}_t$. Output $\frac{1}{T}\sum_{t=0}^T \theta_t$. Here

- $\eta_t = \min\{\frac{F_{\boldsymbol{\xi}_t}(\theta_t) - F_{\boldsymbol{\xi}_t}^*}{c_p\|\nabla F_{\boldsymbol{\xi}_t}(\theta_t)\|^2\sqrt{\sum_{s=0}^t F_{\boldsymbol{\xi}_s}(\theta_s) - F_{\boldsymbol{\xi}_s}^*}}, \eta_{t-1}\}$ for AdaSVRPS
- $\eta_t = \min\{\gamma_t\frac{1}{c_l\sqrt{\sum_{s=0}^t \gamma_s\|\nabla F_{\boldsymbol{\xi}_s}(\theta_s)\|^2}}, \eta_{t-1}\}$ for AdaSVRLS, where γ_t can be obtained by Armijo Backtracking line-search (Armijo, 1966; Nocedal & Wright, 2006): do $\gamma = \beta\gamma$ until $f(\theta_t - \gamma\nabla f(\theta_t, \boldsymbol{\xi}_t), \boldsymbol{\xi}_t) \le f(\theta_t, \boldsymbol{\xi}_t) - \rho\gamma\|\nabla f(\theta_t, \boldsymbol{\xi}_t)\|^2$

- For AdaSPS and AdaSLS, just set $F_{\boldsymbol{\xi}_t}(\theta) = f(\theta, \boldsymbol{\xi}_t)$ and set $F^*_{\boldsymbol{\xi}_t}$ as a predefined lower bound.

MARS (Yuan et al., 2024) (Based on AdamW/Lion/Shampoo) $c_t = g_t + \gamma_t \frac{\beta_1}{1-\beta_1}(g_t - \nabla f(\theta_{t-1}, \boldsymbol{\xi}_t))$, $\widetilde{c}_t = \mathrm{Clip}(c_t, 1)$, $m_t = \beta_1 m_{t-1} + (1-\beta_1)\widetilde{c}_t$, $\theta_{t+1} = \arg\min_\theta\{\eta\,\langle m_t, \theta\rangle + \frac{1}{2}\|\theta - \theta_t\|^2_{\mathbf{H}_t}\}$, or $\theta_{t+1} = \arg\min_\theta\{\eta\,\langle m_t, \theta\rangle + \frac{1}{2}\|\theta - (1-\eta\lambda)\theta_t\|^2_{\mathbf{H}_t}\}$ with weight decay.

- MARS-AdamW: $v_t = \beta_2 v_{t-1} + (1-\beta_2)\widetilde{c}_t^2$, $\mathbf{H}_t := \sqrt{\mathrm{diag}(v_t)} \cdot \frac{1-\beta_1^t}{\sqrt{1-\beta_2^t}}$, (equivalently, $\theta_{t+1} = \theta_t - \eta(\frac{\widehat{m}_t}{\sqrt{\widehat{v}_t}+\epsilon} + \lambda\theta_t)$)
- MARS-Lion: $\mathbf{H}_t = \sqrt{\mathrm{diag}(m_t^2)}$, (equivalently, $\theta_{t+1} = \theta_t - \eta(\mathrm{Sign}(m_t) + \lambda\theta_t)$)
- MARS-Shampoo: $\mathbf{H}_t = (\sum_{\tau=1}^t g_\tau g_\tau^\top)^{1/4} \otimes (\sum_{\tau=1}^t g_\tau^\top g_\tau)^{1/4}$, (equivalently, $U_t, \Sigma_t, V_t = \mathrm{SVD}(m_t)$ $\theta_{t+1} = \theta_t - \eta(U_t V_t^\top + \lambda\theta_t)$), and use Newton-Schulz iteration methods to approximate SVD decomposition (See Section 4 for detail).

6 Scheduler-Free Methods

Schedule-Free AdamW (Defazio et al., 2024)

- (Form 1) $y_t = (1-\beta_1)z_t + \beta_1\theta_t$, $g_t = \nabla_\theta J(y_t)$, $\eta_t = \eta\sqrt{1-\beta_2^t}\min(1, t/T_{\text{warmup}})$, $z_{t+1} = z_t - \eta_t(\frac{g_t}{\sqrt{v_t}+\epsilon} + \lambda y_t)$, $\theta_{t+1} = (1-c_{t+1})\theta_t + c_{t+1}z_{t+1}$, where $c_{t+1} = \frac{\eta_t^2}{\sum_{i=1}^t \eta_i^2}$
- (Form 2) $\eta_t = \eta\sqrt{1-\beta_2^t}\min(1, t/T_{\text{warmup}})$, $g_t = \nabla_\theta J(y_t)$, $\Delta_t = \eta_t(\frac{g_t}{\sqrt{v_t}+\epsilon} + \lambda y_t)$, $y_{t+1} = y_t + \frac{\beta_1 c_{t+1}}{1-\beta_1}(y_t - \theta_t) - [\beta_1 c_{t+1} + (1-\beta_1)]\Delta_t$, $\theta_{t+1} = \theta_t + \frac{c_{t+1}}{1-\beta_1}(y_t - \theta_t) - c_{t+1}\Delta_t$, where $c_{t+1} = \frac{\eta_t^2}{\sum_{i=1}^t \eta_i^2}$

D-Adaptation (Defazio & Mishchenko, 2023)

- Dual Averaging: $m_t = m_{t-1} + d_{t-1}g_t$, $\eta_t = \frac{1}{\sqrt{\sum_{i=0}^k \|g_i\|^2}}$. For $\widehat{d}_t$, option I: $\widehat{d}_t = \frac{\eta_t\|m_t\|^2 - \sum_{i=0}^{t-1}\eta_i d_i^2\|g_{i+1}\|^2}{2\|m_t\|}$ or option II: $\widehat{d}_t = \frac{\sum_{i=0}^{t-1} d_i\eta_i\langle g_{i+1}, m_i\rangle}{\|m_t\|}$, $d_t = \max(d_{t-1}, \widehat{d}_t)$, $\theta_{t+1} = \theta_t - \eta_t m_t$. Output $\frac{1}{\sum_{t=0}^{T-1} d_t}\sum_{t=1}^T d_{t-1}\theta_t$
- Gradient Descent: $\eta_t = \frac{d_{t-1}}{\sqrt{G^2 + \sum_{i=0}^t \|g_i\|^2}}$, $m_t = m_{t-1} + \eta_t g_t$, $\widehat{d}_t = \frac{\|m_t\|^2 - \sum_{i=0}^t \eta_i^2\|g_{i+1}\|^2}{2\|m_t\|}$, $d_t = \max(d_{t-1}, \widehat{d}_t)$, $\theta_{t+1} = \theta_t - \eta_t m_t$. Output $\frac{1}{\sum_{t=0}^T \eta_t}\sum_{t=0}^T \eta_t\theta_t$
- AdaGrad: $m_t = m_{t-1} + d_{t-1}g_t$, $a_t^2 = a_{t-1}^2 + g_t^2$, $A_t = \mathrm{diag}(a_t)$, $\widehat{d}_t = \frac{\|m_t\|^2_{A_t^{-1}} - \sum_{i=0}^{t-1} d_i^2\|g_{i+1}\|^2_{A_i^{-1}}}{2\|m_t\|_1}$, $d_t = \max(d_{t-1}, \widehat{d}_t)$, $\theta_{t+1} = \theta_t - A_t^{-1}m_t$. Output $\frac{1}{\sum_{t=0}^{T-1} d_t}\sum_{t=1}^T d_{t-1}\theta_t$
- SGD: $\eta_k = \eta d_{k-1}/G$, $m_t = m_{t-1} + \eta_t g_t$, $z_t = z_{t-1} - \eta_t g_t$, $\theta_{t+1} = \beta\theta_t + (1-\beta)z_t$, $\widehat{d}_t = \frac{2\sum_{i=0}^{t-1}\eta_{i+1}\langle g_{i+1}, m_i\rangle}{\|m_t\|}$, $d_t = \max(d_{t-1}, \widehat{d}_t)$.
- Adam version (Based on Adam) $m_t = \beta_1 m_{t-1} + (1-\beta_1)\eta d_{t-1}g_t$, $\theta_{t+1} = \theta_t - \frac{m_t}{\sqrt{v_t}+\epsilon}$, $s_t = \sqrt{\beta_2}s_{t-1} + (1-\sqrt{\beta_2})\eta d_{t-1}g_t$, $r_t = \sqrt{\beta_2}r_{t-1} + (1-\sqrt{\beta_2})\eta d_{t-1}\langle g_t, s_{t-1}\rangle_{(\mathrm{diag}(\sqrt{v_t}+\epsilon))^{-1}}$, $\widehat{d}_t = \frac{r_t}{(1-\sqrt{\beta_2})\|s_t\|_1}$, $d_t = \max(d_{t-1}, \widehat{d}_t)$

Adam++ (Tao et al., 2024) $\eta_t = \max(\eta_{t-1}, \|\theta_t - \theta_0\|/\sqrt{d})$, $\beta_{1,t} = \beta_1\lambda^{t-1}$, $m_t = \beta_{1,t}m_{t-1} + (1-\beta_{1,t})g_t$, $\theta_{t+1} = \theta_t - \eta_t \cdot \mathbf{H}_t^{-1}m_t$, where $\mathbf{H}_t = \epsilon + \mathrm{diag}(s_t)$, and s_t is calculated by either of the following ways:

- Case I: $s_t = \sqrt{\sum_{i=0}^t g_i^2}$

- Case II: $v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$, $s_t = \sqrt{(t+1)\max_{t' \le t}(v_{t'})}$

AdaGrad++ (Tao et al., 2024) Similar to Adam++, but only choose Case I of s_t, and $\lambda = 0$.

7 Randomized Updating

GLD (Gradient Langevin Dynamics) (Durmus & Moulines, 2017; Dalalyan, 2017a,b) $\theta_0 = \mathbf{0}$, $\epsilon_t \sim N(0, I_{d\times d})$, $\theta_{t+1} = \theta_t - \eta g_t + \sqrt{2\eta/\beta}\epsilon_t$, where g_t is the full gradient.

SGLD (Welling & Teh, 2011) Same as GLD, but g_t is the average gradient over samples in small batch.

SGFS (Ahn et al., 2012) For total number of samples N and batch size B, $\gamma = \frac{B+N}{B}$. For small batch $\{\boldsymbol{\xi}_{t,i}\}_{i=1}^B$, $v_t = (1-\beta)v_{t-1} + \beta\frac{1}{n-1}\sum_{i=1}^B(\nabla f(\theta_t, \boldsymbol{\xi}_{t,i}) - g_t)(\nabla f(\theta_t, \boldsymbol{\xi}_{t,i}) - g_t)^\top$, $\epsilon_t \sim \mathcal{N}(0, \frac{4C}{\epsilon})$, $\theta_{t+1} = \theta_t + 2(\gamma N v_t + \frac{4C}{\epsilon})^{-1}(g_t + \epsilon_t)$.

SVRG-LD (Xu et al., 2018) For each epoch s, first compute the full gradient g_s for θ_s, $\widetilde{g}_{s,t} = g_{s,t} - \nabla f(g_s, \boldsymbol{\xi}_{s,t}) + g_s$, $\epsilon_t \sim N(0, I_{d\times d})$, $\theta_{t+1} = \theta_t - \eta\widetilde{g}_{s,t} + \sqrt{2\eta/\beta}\epsilon_t$. At the last iteration T for each epoch, $g_s = g_{s,T}$.

8 Reconciliation of Optimizers

AdaGraft (Agarwal et al., 2020) For optimizers $\mathcal{M}, \mathcal{D}$, $\theta_{t,\mathcal{M}} = \mathcal{M}(\theta_t, g_t)$, $\theta_{t,\mathcal{D}} = \mathcal{D}(\theta_t, g_t)$, $\theta_{t+1} = \theta_t + \frac{\|\theta_{t,\mathcal{M}} - \theta_t\|}{\|\theta_{t,\mathcal{D}} - \theta_t\| + \epsilon} \cdot (\theta_{t,\mathcal{D}} - \theta_t)$

9 Architecture-specific Optimizers

GaLore (Zhao et al., 2024a) (Adam for LLM layer weight matrix): $\theta_t \in \mathbb{R}^{m\times n}$, if $t \bmod T = 0$ $U, S, V = \text{SVD}(g_t)$, $P_t = U[:, : r]$ (low-rank projection), otherwise $P_t = P_{t-1}$. Then estimate the low-rank gradient $R_t = P_t^\top g_t$, and use Adam to optimize: $\theta_{t+1} = \theta_t + \eta\alpha P_t\frac{m_t}{\sqrt{v_t}+\epsilon}$ with scale factor α

Adam-mini (Zhang et al., 2024) (Based on Adam) (For each parameter in parameter blocks) $v_t = \beta_2 v_{t-1} + (1-\beta_2) * \text{Mean}(g \odot g)$. For Transformers, partition the parameters of embedding and output layers by tokens, partition the parameters of query and key matrices by heads, and partition the parameters of value matrices, attention projection matrices and MLP layers by output neurons.

Adalayer (Zhao et al., 2024b) (Based on Adam, for language model) For each layer: $v_t = \beta_2 v_{t-1} + (1-\beta_2)\|g_t\|_2^2/\sqrt{p}$ with p the number of parameters in each layer, $\theta_{t+1} = \theta_t - \eta\frac{m_t}{\sqrt{v_t}+\epsilon}$

STORM-PG (Yuan et al., 2020) (Based on Reinforce Learning tasks) $\theta_{t+1} = \theta_t + \eta\widehat{g}_t$, $\widehat{g}_{t+1} = (1-\gamma)g_t' + g_t$, where $g_t = \frac{1}{B}\sum_{i\in\mathcal{B}} d_i(\theta_{t+1})$, $g_t' = \frac{1}{B}\sum_{i\in\mathcal{B}}[\widehat{g}_t - d_i^{\theta_{t+1}}(\theta_t)]$ is the gradient estimation with different parameters on small batch of trajectories $\{\tau_i\}_{i\in\mathcal{B}}$ with size B, and $d_i(\theta) = \sum_{h=0}^{H-1} d_{i,h}(\theta)$, $d_i^{\theta'}(\theta) = \sum_{h=0}^{H-1}\frac{p(\tau_{i,h}|\theta)}{p(\tau_{i,h}|\theta')}d_{i,h}(\theta)$, where $d_{i,h}(\theta) = (\sum_{t=0}^h \nabla\log\pi_\theta(a_t|s_t))(\gamma^h r(s_h, a_h) - b_h)$.

SPPO (Wu et al., 2024) (Based on Reinforce Learning tasks)

$$\theta_{t+1} = \arg\min_\theta \mathbb{E}_{(\boldsymbol{\xi}_t, y_t, \widehat{P}(y_t \succ \pi_t|\boldsymbol{\xi}_t))}\big(\log\big(\frac{\pi_\theta(y_t|\boldsymbol{\xi}_t)}{\pi_t(y_t|\boldsymbol{\xi}_t)}\big) - \eta(\widehat{P}(y_t \succ \pi_t|\boldsymbol{\xi}_t) - \frac{1}{2})\big)^2,$$

where $\pi_t = \pi_{\theta_t}$ is the policy model with the parameter θ_t.

10 Something Beyond Optimizers

In the previous sections, I did not emphasize the changing of the learning rate η. In this section, I collected some interesting research related to the changing rule of learning rate (Scheduler).

10.1 Scheduler

Warmup-Stable-Decay (WSD) Scheduler (Hu et al., 2024) Different from the Cosine Learning Rate Scheduler, in MiniCPM, the researchers utilized WSD scheduler (linear warmup-stable learning rate-decay). It is widely used in industry because it can support continual training, and decaying schedule in the end is also compatible for downstream tasks fine-tuning.

For intermediate checkpoints, WSD performs decay schedule and starts from the state before decay when continuing training. However, in **WSD-S** (Wen et al., 2024), the researchers find that the dispose of training in decay phase is not necessary, and just starts from the state after decay with the max learning rate is fine.

10.2 Schedule Refinement

Schedule Refinement (Defazio et al., 2023) The researchers perform comprehensive evaluation of learning rate schedules, give proofs on the convergence of some common optimization approaches, and proposed some important conclusions and algorithms:

- Warm-up followed by linear decay is the best overall non-adaptive schedule, outperforming cosine decay.
- **Schedule Refinement for SGD** $\widehat{g}_t = \text{Median-filter}(\|g_t\|, \text{width} = \tau T, \text{pad} = (\text{nearest}, \text{reflect}))$, $w_t = \widehat{g}_t^{-2}$, $\eta'_t = w_t \sum_{p=t+1}^{T} w_p$, $\eta_t = \eta'_t / \max_p(\eta'_p)$.
- **Schedule Refinement for SGD** $\widehat{g}_t = \text{Median-filter}(\| \sum_{i=1}^{d} \frac{g_{t,i}^2}{\sqrt{v_{t,i}}} \|, \text{width} = \tau T, \text{pad} = (\text{nearest}, \text{reflect}))$, $w_t = \widehat{g}_t^{-1}$, $\eta'_t = w_t \sum_{p=t+1}^{T} w_p$, $\eta_t = \eta'_t / \max_p(\eta'_p)$.

10.3 "Scaling Law" of Learning Rate

μP and Tensor Program series (Yang et al., 2022) In Tensor Program series, the researchers led by Greg Yang[2] provides theoretical foundation for the "Scaling Law" of hyper-parameter to the model size. In Tensor Program V (μP), they researched how the best hyper-parameters (including initialization variances and learning rates for different optimizers of different components in deep learning models) change with respect to the size of these components.

11 Acknowledgement

Thank Prof. Quanquan Gu, and other students in UCLA AGI Lab including Huizhuo Yuan. Thank former colleagues from ByteDance Inc. and Moonshot Inc. including Jianlin Su for his blogs at `https://kexue.fm/`.

References

Naman Agarwal, Rohan Anil, Elad Hazan, Tomer Koren, and Cyril Zhang. Disentangling adaptive gradient methods from learning rates. *arXiv preprint arXiv:2002.11803*, 2020.

Sungjin Ahn, Anoop Korattikara Balan, and Max Welling. Bayesian posterior sampling via stochastic gradient fisher scoring. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. icml.cc / Omnipress, 2012. URL `http://icml.cc/2012/papers/782.pdf`.

Larry Armijo. Minimization of functions having lipschitz continuous first partial derivatives. *Pacific Journal of mathematics*, 16(1):1–3, 1966.

Jeremy Bernstein and Laker Newhouse. Old optimizer, new norm: An anthology. *arXiv preprint arXiv:2409.20325*, 2024.

[2] `https://thegregyang.com/`

Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. signsgd: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning*, pp. 560–569. PMLR, 2018.

Yang Cao, Xiaoyu Li, and Zhao Song. Grams: Gradient descent with adaptive momentum scaling. *arXiv preprint arXiv:2412.17107*, 2024.

Jinghui Chen and Quanquan Gu. Padam: Closing the generalization gap of adaptive gradient methods in training deep neural networks. 2018.

Xiangning Chen, Chen Liang, Da Huang, Esteban Real, Kaiyuan Wang, Hieu Pham, Xuanyi Dong, Thang Luong, Cho-Jui Hsieh, Yifeng Lu, and Quoc V. Le. Symbolic discovery of optimization algorithms. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

Ashok Cutkosky and Francesco Orabona. Momentum-based variance reduction in non-convex sgd. *Advances in neural information processing systems*, 32, 2019.

Arnak Dalalyan. Further and stronger analogy between sampling and optimization: Langevin monte carlo and gradient descent. In *Conference on Learning Theory*, pp. 678–689. PMLR, 2017a.

Arnak S Dalalyan. Theoretical guarantees for approximate sampling from smooth and log-concave densities. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 79(3):651–676, 2017b.

Aaron Defazio and Konstantin Mishchenko. Learning-rate-free learning by d-adaptation. In *International Conference on Machine Learning*, pp. 7449–7479. PMLR, 2023.

Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. *Advances in neural information processing systems*, 27, 2014.

Aaron Defazio, Ashok Cutkosky, Harsh Mehta, and Konstantin Mishchenko. When, why and how much? adaptive learning rate scheduling by refinement. *arXiv preprint arXiv:2310.07831*, 2023.

Aaron Defazio, Xingyu Alice Yang, Harsh Mehta, Konstantin Mishchenko, Ahmed Khaled, and Ashok Cutkosky. The road less scheduled. *arXiv preprint arXiv:2405.15682*, 2024.

Timothy Dozat. Incorporating nesterov momentum into adam. 2016.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

Alain Durmus and Éric Moulines. Nonasymptotic convergence analysis for the unadjusted langevin algorithm. *The Annals of Applied Probability*, pp. 1551–1587, 2017.

Cong Fang, Chris Junchi Li, Zhouchen Lin, and Tong Zhang. Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator. *Advances in neural information processing systems*, 31, 2018.

Roger Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, New York, 2 edition, 1987. ISBN 978-0-471-91547-8.

Marcus Grum. Learning representations by crystallized back-propagating errors. In Leszek Rutkowski, Rafal Scherer, Marcin Korytkowski, Witold Pedrycz, Ryszard Tadeusiewicz, and Jacek M. Zurada (eds.), *Artificial Intelligence and Soft Computing - 22nd International Conference, ICAISC 2023, Zakopane, Poland, June 18-22, 2023, Proceedings, Part I*, volume 14125 of *Lecture Notes in Computer Science*, pp. 78–100. Springer, 2023. doi: 10.1007/978-3-031-42505-9_8. URL `https://doi.org/10.1007/978-3-031-42505-9_8`.

Vineet Gupta, Tomer Koren, and Yoram Singer. Shampoo: Preconditioned stochastic tensor optimization. In *International Conference on Machine Learning*, pp. 1842–1850. PMLR, 2018.

Nicholas J Higham. Functions of matrices. society for industrial and applied mathematics, 2008.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, et al. Minicpm: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*, 2024.

Feihu Huang, Junyi Li, and Heng Huang. Super-adam: faster and universal framework of adaptive gradients. *Advances in Neural Information Processing Systems*, 34:9074–9085, 2021.

Michael F Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989.

Xiaowen Jiang and Sebastian U Stich. Adaptive sgd with polyak stepsize and line-search: Robust convergence and variance reduction. *Advances in Neural Information Processing Systems*, 36, 2024.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems*, 26, 2013.

Keller Jordan, Yuchen Jin, Vlado Boza, Jiacheng You, Franz Cecista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL `https://kellerjordan.github.io/posts/muon/`.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Kfir Levy, Ali Kavis, and Volkan Cevher. Storm+: Fully adaptive sgd with recursive momentum for nonconvex optimization. *Advances in Neural Information Processing Systems*, 34:20571–20582, 2021.

Chris Junchi Li, Wenlong Mou, Martin Wainwright, and Michael Jordan. Root-sgd: Sharp nonasymptotics and asymptotic efficiency in a single algorithm. In *Conference on Learning Theory*, pp. 909–981. PMLR, 2022.

Wenjie Li, Zhaoyang Zhang, Xinjiang Wang, and Ping Luo. Adax: Adaptive gradient descent with exponential long term memory. *arXiv preprint arXiv:2004.09740*, 2020.

Kaizhao Liang, Lizhang Chen, Bo Liu, and Qiang Liu. Cautious optimizers: Improving training with one line of code. *arXiv preprint arXiv:2411.16085*, 2024.

Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1):503–528, 1989.

Hong Liu, Zhiyuan Li, David Leo Wright Hall, Percy Liang, and Tengyu Ma. Sophia: A scalable stochastic second-order optimizer for language model pre-training. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020a.

Mingrui Liu, Wei Zhang, Francesco Orabona, and Tianbao Yang. Adam $^+$: A stochastic method with adaptive variance reduction. *arXiv preprint arXiv:2011.11985*, 2020b.

Ilya Loshchilov. Weight norm control. *arXiv preprint arXiv:2311.11446*, 2023.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

James Martens and Roger Grosse. Optimizing neural networks with kronecker-factored approximate curvature. In *International conference on machine learning*, pp. 2408–2417. PMLR, 2015.

James Martens et al. Deep learning via hessian-free optimization. In *Icml*, volume 27, pp. 735–742, 2010.

Yurii Nesterov. A method for solving the convex programming problem with convergence rate o (1/k2). In *Dokl akad nauk Sssr*, volume 269, pp. 543, 1983.

Lam M Nguyen, Jie Liu, Katya Scheinberg, and Martin Takáč. Sarah: A novel method for machine learning problems using stochastic recursive gradient. In *International conference on machine learning*, pp. 2613–2621. PMLR, 2017.

Jorge Nocedal and Stephen J. Wright. Numerical optimization. Springer, New York, NY, USA, 2e edition, 2006.

Matteo Pagliardini, Pierre Ablin, and David Grangier. The ademamix optimizer: Better, faster, older. *arXiv preprint arXiv:2409.03137*, 2024.

Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855, 1992.

Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: the rprop algorithm. In *IEEE International Conference on Neural Networks*, pp. 586–591 vol.1, 1993. doi: 10.1109/ICNN.1993.298623.

Nicolas Roux, Mark Schmidt, and Francis Bach. A stochastic gradient method with an exponential convergence _rate for finite training sets. *Advances in neural information processing systems*, 25, 2012.

Shai Shalev-Shwartz and Tong Zhang. Stochastic dual coordinate ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14(1), 2013.

Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pp. 4596–4604. PMLR, 2018.

Jianlin Su. Tiger: A tight-fisted optimizer. `https://github.com/bojone/tiger`, 2023.

Yuanzhe Tao, Huizhuo Yuan, Xun Zhou, Yuan Cao, and Quanquan Gu. Towards simple and provable parameter-free adaptive gradient methods, 2024. URL `https://arxiv.org/abs/2412.19444`.

Ran Tian and Ankur P Parikh. Amos: An adam-style optimizer with adaptive weight decay towards model-oriented scale. *arXiv preprint arXiv:2210.11693*, 2022.

Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

Phuong Thi Tran et al. On the convergence proof of amsgrad and a new version. *IEEE Access*, 7: 61706–61716, 2019.

Quoc Tran-Dinh, Nhan H Pham, Dzung T Phan, and Lam M Nguyen. Hybrid stochastic gradient descent algorithms for stochastic nonconvex optimization. *arXiv preprint arXiv:1905.05920*, 2019.

Nikhil Vyas, Depen Morwani, Rosie Zhao, Itai Shapira, David Brandfonbrener, Lucas Janson, and Sham Kakade. Soap: Improving and stabilizing shampoo using adam. *arXiv preprint arXiv:2409.11321*, 2024.

Zhe Wang, Kaiyi Ji, Yi Zhou, Yingbin Liang, and Vahid Tarokh. Spiderboost and momentum: Faster variance reduction algorithms. *Advances in Neural Information Processing Systems*, 32, 2019.

Rachel Ward, Xiaoxia Wu, and Leon Bottou. Adagrad stepsizes: Sharp convergence over nonconvex landscapes. *Journal of Machine Learning Research*, 21(219):1–30, 2020.

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 681–688. Citeseer, 2011.

Kaiyue Wen, Zhiyuan Li, Jason Wang, David Hall, Percy Liang, and Tengyu Ma. Understanding warmup-stable-decay learning rates: A river valley loss landscape perspective. *arXiv preprint arXiv:2410.05192*, 2024.

Yue Wu, Zhiqing Sun, Huizhuo Yuan, Kaixuan Ji, Yiming Yang, and Quanquan Gu. Self-play preference optimization for language model alignment. *arXiv preprint arXiv:2405.00675*, 2024.

Xingyu Xie, Pan Zhou, Huan Li, Zhouchen Lin, and Shuicheng Yan. Adan: Adaptive nesterov momentum algorithm for faster optimizing deep models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

Pan Xu, Jinghui Chen, Difan Zou, and Quanquan Gu. Global convergence of langevin dynamics based algorithms for nonconvex optimization. *Advances in Neural Information Processing Systems*, 31, 2018.

Greg Yang, Edward J Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tensor programs v: Tuning large neural networks via zero-shot hyperparameter transfer. *arXiv preprint arXiv:2203.03466*, 2022.

Zhewei Yao, Amir Gholami, Sheng Shen, Mustafa Mustafa, Kurt Keutzer, and Michael Mahoney. Adahessian: An adaptive second order optimizer for machine learning. In *proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 10665–10673, 2021.

Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017.

Yang You, Jing Li, Sashank J. Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training BERT in 76 minutes. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

Huizhuo Yuan, Xiangru Lian, Ji Liu, and Yuren Zhou. Stochastic recursive momentum for policy gradient methods. *arXiv preprint arXiv:2003.04302*, 2020.

Huizhuo Yuan, Yifeng Liu, Shuang Wu, Xun Zhou, and Quanquan Gu. Mars: Unleashing the power of variance reduction for training large models. *arXiv preprint arXiv:2411.10438*, 2024.

Manzil Zaheer, Sashank Reddi, Devendra Sachan, Satyen Kale, and Sanjiv Kumar. Adaptive methods for nonconvex optimization. *Advances in neural information processing systems*, 31, 2018.

Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

Aston Zhang, Zachary C Lipton, Mu Li, and Alexander J Smola. Dive into deep learning. *arXiv preprint arXiv:2106.11342*, pp. 505, 2021.

Yushun Zhang, Congliang Chen, Ziniu Li, Tian Ding, Chenwei Wu, Yinyu Ye, Zhi-Quan Luo, and Ruoyu Sun. Adam-mini: Use fewer learning rates to gain more. *arXiv preprint arXiv:2406.16793*, 2024.

Jiawei Zhao, Zhenyu Zhang, Beidi Chen, Zhangyang Wang, Anima Anandkumar, and Yuandong Tian. Galore: Memory-efficient LLM training by gradient low-rank projection. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024a.

Rosie Zhao, Depen Morwani, David Brandfonbrener, Nikhil Vyas, and Sham Kakade. Deconstructing what makes a good optimizer for language models. *arXiv preprint arXiv:2407.07972*, 2024b.

Dongruo Zhou, Pan Xu, and Quanquan Gu. Stochastic nested variance reduction for nonconvex optimization. *Journal of machine learning research*, 21(103):1–63, 2020.

Juntang Zhuang, Tommy Tang, Yifan Ding, Sekhar C Tatikonda, Nicha Dvornek, Xenophon Papademetris, and James Duncan. Adabelief optimizer: Adapting stepsizes by the belief in observed gradients. *Advances in neural information processing systems*, 33:18795–18806, 2020.