Optimization for Deep Learning

Lecture 8-2: Adaptive SGD

Kun Yuan

Peking University

Main contents in this lecture

- Preconditioned SGD
- AdaGrad
- RMSProp
- Adam

Preconditioned GD

- Consider an ill-conditioned quadratic problem

$$\min_x \quad x^T Q x + c^T x$$

where Q is an ill-conditioned matrix. GD is slow when solving the problem

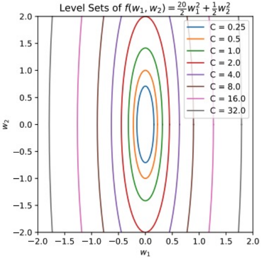

Figure: An ill-conditioned QP problem. (From Prof. Chris De Sa's lecture notes)

Preconditioned GD

- We now let $x = P^{\frac{1}{2}}w$ for some positive definite matrix P. Since P is positive definite, x and w is an $1-1$ mapping

- If we choose $P = Q^{-1}$, we have $x^TQx = w^TQ^{-\frac{1}{2}}QQ^{\frac{1}{2}}w = \|w\|^2$

- With $x = P^{\frac{1}{2}}w$ and $P = Q^{-1}$, the ill-conditioned problem becomes

$$\min_{w} \quad \frac{1}{2}\|w\|^2 + c^TQ^{-\frac{1}{2}}w$$

  which is a benign problem. GD is fast to achieve $w^\star$.

- Once $w^\star$ is determined, we have $x^\star = P^{\frac{1}{2}}w^\star$.

Preconditioned GD

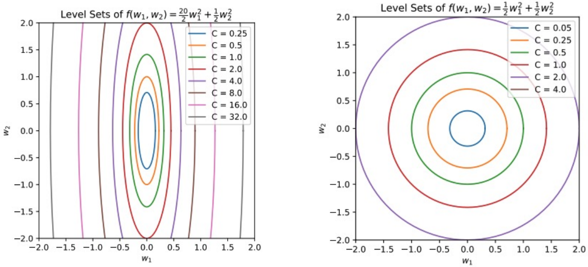

Figure: Left: an ill-conditioned QP problem. Right: a benign QP problem after transformation.(From Prof. Chris De Sa's lecture notes)

Preconditioned GD: derivation

- Consider a general ill-conditioned optimization problem

$$\min_{x\in\mathbb{R}^d} \quad f(x)$$

- We let $x = P^{\frac{1}{2}}w$ so that $g(w) = f(P^{\frac{1}{2}}w)$ is a nice function.

- Use gradient descent to minimize $g(w)$, i.e.,

$$w_{k+1} = w_k - \gamma\nabla g(w_k) = w_k - \gamma P^{\frac{1}{2}}\nabla f(P^{\frac{1}{2}}w_k)$$

- Left-multiplying $P^{\frac{1}{2}}$ to both sides, we achieve

$$\begin{aligned} & P^{\frac{1}{2}}w_{k+1} = P^{\frac{1}{2}}w_k - \gamma P\nabla f(P^{\frac{1}{2}}w_k) \\ \Longleftrightarrow \quad & x_{k+1} = x_k - \gamma P\nabla f(x_k) \end{aligned}$$

  where P is called the preconditioning matrix.

Preconditioned GD

- The preconditioned GD algorithm

$$x_{k+1} = x_k - \gamma P_k \nabla f(x_k)$$

  where P_k varies with iteration k.

- It is critical to choose the preconditioning matrix P_k

- If $P_k = [\nabla^2 f(x_k)]^{-1}$, then preconditioned GD reduces to Newton's method

- It is critical to construct an efficient and effective P matrix

Stochastic optimization

- Consider the stochastic optimization problem:

$$\min_{x \in \mathbb{R}^d} \quad f(x) = \mathbb{E}_{\xi \sim \mathcal{D}}[F(x;\xi)]$$

  - ξ is a random variable indicating data samples
  - $\mathcal{D}$ is the data distribution; unknown in advance
  - $F(x;\xi)$ is differentiable in terms of x

- Similar to preconditioned GD, **preconditioned SGD** iterates as follows

$$x_{k+1} = x_k - \gamma P_k \nabla F(x_k;\xi_k)$$

Adaptive gradient method (AdaGrad)

- Adaptive gradient method

$$
\begin{aligned}
g_k &= \nabla F(x_k; \xi_k) \\
s_k &= s_{k-1} + g_k \odot g_k \\
x_{k+1} &= x_k - \frac{\gamma}{\sqrt{s_k} + \epsilon} \odot g_k
\end{aligned}
$$

where $1/\sqrt{s_k} = \mathrm{col}\{1/\sqrt{s_{k,1}}, \cdots, 1/\sqrt{s_{k,d}}\} \in \mathbb{R}^d$ is an element-wise operation, s_0 is initialized as 0, and a small ϵ is added for safe-guard.

Adaptive gradient method (AdaGrad)

- AdaGrad falls into preconditioned SGD

- If we let $P_k = \mathrm{diag}\{\frac{1}{\sqrt{s_{k,1}}+\epsilon}, \cdots, \frac{1}{\sqrt{s_{k,d}}+\epsilon}\} \in \mathbb{R}^{d\times d}$, AdaGrad becomes

$$x_{k+1} = x_k - \gamma P_k g_k$$

  where P_k is a time-varying preconditioning matrix.

- AdaGrad imposes smaller learning rates for notable gradient directions

- AdaGrad imposes larger learning rates for insignificant gradient directions

Adaptive gradient method (AdaGrad)

AdaGrad alleviates the “Zig-Zag” phenomenon

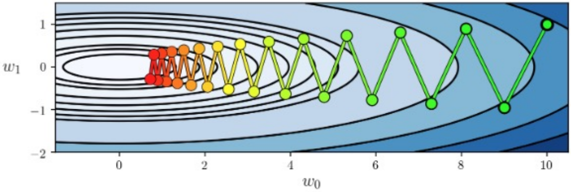

Figure: GD converges slow for ill-conditioned problem

Adaptive gradient method (AdaGrad)

AdaGrad alleviates the "Zig-Zag" phenomenon

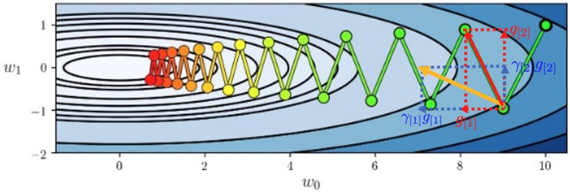

Figure: AdaGrad has alleviated "Zig-Zag" phenomenon

Adaptive gradient method (AdaGrad)

The learning rate in AdaGrad is adaptive; no need to tune.

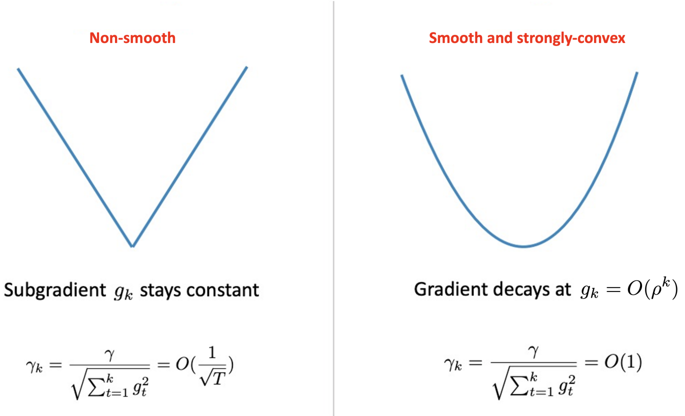

Figure: AdaGrad automatically adapts to problem structure[1].

[1] These examples are from https://conferences.mpi-inf.mpg.de/adfocs/material/alina/adaptive-L1.pdf

RMSProp

- Since s_k keeps increasing, the rate γ_k in AdaGrad keeps decreasing
- AdaGrad may suffer from slow convergence
- RMSProp proposes a different way to construct s_k

$$s_k = \beta s_{k-1} + (1 - \beta) g_k \odot g_k$$

where $\beta \in (0, 1)$. A typical value for β is 0.9.

- In RMSProp, only the most recent g_k influences the convergence rate

RMSProp

- Suppose $g_k = 1/k$, we can visualize s_k from AdaGrad and RMSProp

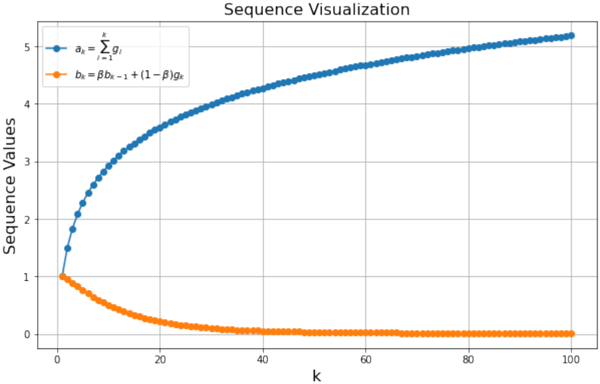

Figure: AdaGrad increases very fast while RMSProp decays slowly with $\beta = 0.9$

RMSProp

- We also visualize s_k from RMSProp with different β.

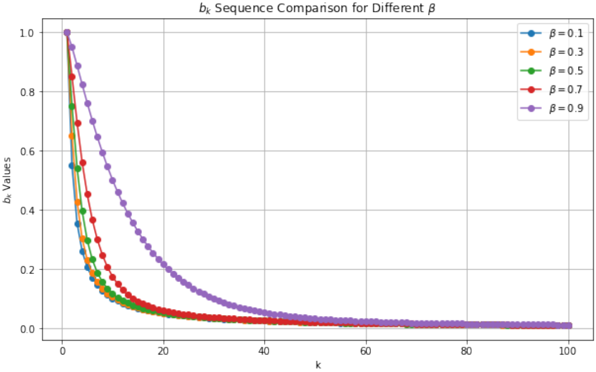

Figure: Gradient accumulation in RMSProp with different β.

RMSProp

- RMSProp has the following update

$$
\begin{aligned}
g_k &= \nabla F(x_k; \xi_k) \\
s_k &= \beta s_{k-1} + (1-\beta) g_k \odot g_k \\
x_{k+1} &= x_k - \frac{\gamma}{\sqrt{s_k} + \epsilon} \odot g_k
\end{aligned}
$$

where s_0 is initialized as 0, and a small ϵ is added for safe-guard.

Adam

- Adam applies both momentum and adaptive rate to alleviate "Zig-Zag".

$$\begin{aligned} g_k &= \nabla F(x_k; \xi_k) \\ m_k &= \beta_1 m_{k-1} + (1 - \beta_1) g_k \\ s_k &= \beta_2 s_{k-1} + (1 - \beta_2) g_k \odot g_k \\ x_{k+1} &= x_k - \frac{\gamma}{\sqrt{s_k} + \epsilon} \odot m_k \end{aligned}$$

  where m_0 and s_0 are initialized as 0, and a small ϵ is added for safe-guard.

- It is good to set $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

Animation of different adaptive SGDs

https://imgur.com/a/Hqolp

Numerical performance

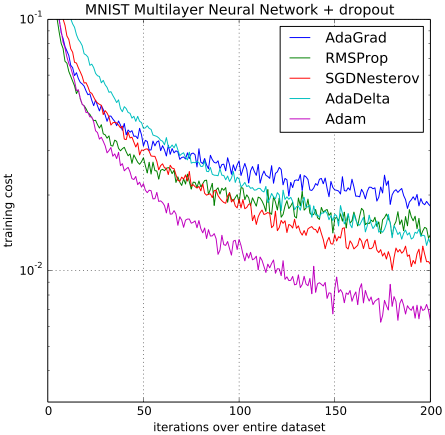

Figure: This figure is from the Adam paper (Kingma and Ba, 2014)

Convergence analysis

- We consider a general family of adaptive algorithms (Guo et al., 2021)

$$
\begin{aligned}
g_k =& F(x_k; \xi_k), \\
m_k =& \rho m_{k-1} + (1-\rho) g_k, \\
v_k =& h_k(g_0, \cdots, g_k), \\
x_{k+1} =& x_k - \frac{\gamma}{\sqrt{v_k} + \epsilon} m_k,
\end{aligned}
$$

  where $h_k(\cdot)$ is some mapping that varies for different adaptive algorithms

- Covers AdaGrad, RMSProp, Adam, Adadelta, AdaBound, etc.

Convergence analysis

Assumption 1

Assume f is L-smooth, and its stochastic gradient oracle $\nabla F(x;\xi)$ satisfies:

$$\mathbb{E}[\nabla F(x;\xi)] = \nabla f(x),\ \forall x \in \mathbb{R}^d.$$

$$\mathbb{E}[\|\nabla F(x;\xi) - \nabla f(x)\|_2^2] \leq \sigma^2,\ \forall x \in \mathbb{R}^d.$$

Assumption 2

We assume each $s_k := 1/(\sqrt{v_k} + \epsilon)$ is lower bounded and upper bounded, i.e., there exists $0 < c_l < c_u$ such that $c_l \leq \|s_k\|_\infty \leq c_u$.

Assumption 2 is strong but can significantly simplify the analysis; easy to satisfy when using gradient clipping

Convergence analysis

- Let $\mathcal{F}_k := \{x_k, \xi_{k-1}, m_{k-1}, v_{k-1}, x_{k-2}, \cdots, \xi_0, m_0, v_0, x_0\}$ denote the filtration containing all historical variables at and before computing x_k.
- Let $\mathbb{E}_k$ denote the expectation conditioned on $\mathcal{F}_k$, *i.e.*, $\mathbb{E}_k[\cdot] = \mathbb{E}[\cdot \mid \mathcal{F}_k]$.
- Let $\tilde{\gamma}_k := \gamma/(\sqrt{v_k} + \epsilon)$.

Lemma 1

Under Assumption 1, it holds that

$$\begin{aligned}&\mathbb{E}[\|m_k - \nabla f(x_k)\|_2^2]\\ \leq &\rho\mathbb{E}\|m_{k-1} - \nabla f(x_{k-1})\|_2^2 + (1-\rho)^2\sigma^2 + \frac{\rho^2 L^2 \mathbb{E}\|x_k - x_{k-1}\|_2^2}{1-\rho}\end{aligned}$$

Convergence analysis

Let $\mathcal{M}_k = \mathbb{E}\|m_k - \nabla f(x_k)\|^2$. From Lemma 1, we have

$$\sum_{k=0}^{K} \mathcal{M}_k \leq \frac{\mathcal{M}_0}{1-\rho} + (1-\rho)(K+1)\sigma^2 + \frac{\rho^2 L^2 \gamma^2 c_u^2}{(1-\rho)^2} \sum_{k=0}^{K} \mathbb{E}\|m_k\|^2 \tag{1}$$

Convergence analysis

Lemma 2

Under Assumptions 1 and 2, if $\gamma L \le c_l/(2c_u^2)$, we have

$$\begin{aligned}\mathbb{E}[f(x_{k+1})] \le \mathbb{E}[f(x_k)] &+ \frac{\gamma c_u}{2}\mathbb{E}\|\nabla f(x_k) - m_k\|_2^2 \\ &- \frac{\gamma c_l}{2}\mathbb{E}\|\nabla f(x_k)\|_2^2 - \frac{\gamma c_l}{4}\mathbb{E}\|m_k\|_2^2.\end{aligned}$$

The above lemma implies that

$$\sum_{k=0}^{K}\mathbb{E}\|\nabla f(x_k)\|^2 \le \frac{2\Delta_0}{\gamma c_l} + \frac{c_u}{c_l}\sum_{k=0}^{K}\mathcal{M}_k - \frac{1}{2}\sum_{k=0}^{K}\mathbb{E}\|m_k\|^2 \tag{2}$$

Convergence

Substituting (1) to (2), we achieve the following result

Theorem 1

Under Assumptions 1 and 2, if γ is sufficiently small such that $\gamma^2 \leq \frac{c_l(1-\rho)^2}{2\rho^2 L^2 c_u^3}$, the family of adaptive algorithms will converge as

$$\frac{1}{K+1}\sum_{k=0}^{K}\mathbb{E}\|\nabla f(x_k)\|^2 \leq \frac{2\Delta_0}{\gamma c_l(K+1)} + \frac{c_u \mathcal{M}_0}{c_l(1-\rho)(K+1)} + \frac{(1-\rho)c_u\sigma^2}{c_l}$$

When $1-\rho = O(1/\sqrt{K})$ and $\gamma = O(1/\sqrt{K})$, the Adam-family will converge as

$$\frac{1}{K+1}\sum_{k=0}^{K}\mathbb{E}\|\nabla f(x_k)\|^2 = \mathcal{O}(1/\sqrt{K})$$

Not tight but very general; **cannot** validate the benefits of adaptivity; it is still an open question to show the benefits of the adaptivity.

Adam with weight decay (AdamW)

- Adam with weight decay iterates as follows

$$
\begin{aligned}
g_k &= \nabla F(x_k;\xi_k) \\
m_k &= \beta_1 m_{k-1} + (1-\beta_1) g_k \\
s_k &= \beta_2 s_{k-1} + (1-\beta_2) g_k \odot g_k \\
x_{k+1} &= x_k - \gamma\left(\frac{1}{\sqrt{s_k}+\epsilon} \odot m_k + \lambda x_k\right)
\end{aligned}
$$

  where the weight decay term λx_k can improve the generalization

- Closely related to regularization but is not equivalent to it

- Has strong empirical performance but is **less understood in theory**

Adam vs. AdamW

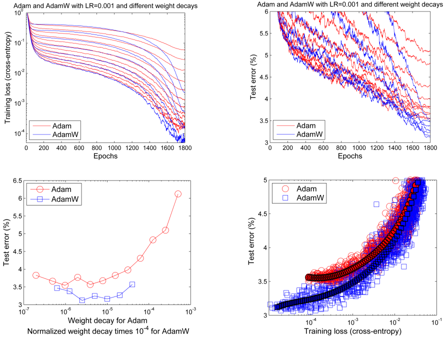

Figure: This figure is from the AdamW paper (Loshchilov and Hutter, 2018)

References I

D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

Z. Guo, Y. Xu, W. Yin, R. Jin, and T. Yang, "A novel convergence analysis for algorithms of the adam family," 2021.

I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations*, 2018.