Introduction to Large Language Models

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

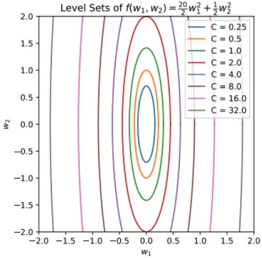

Figure: An ill-conditioned QP problem. (From Prof. Chris De Sa's lecture notes)

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

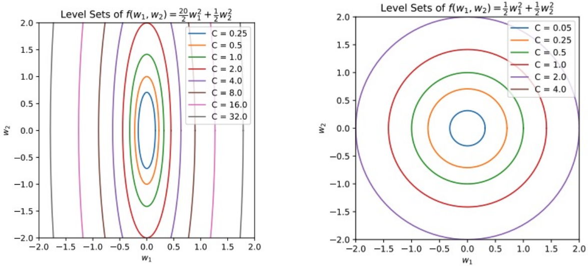

Figure: Left: an ill-conditioned QP problem. Right: a benign QP problem after transformation.(From Prof. Chris De Sa's lecture notes)

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

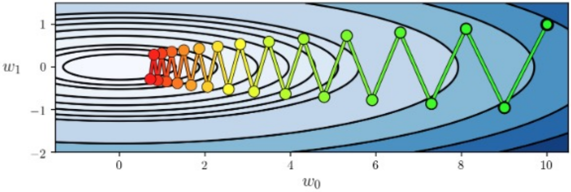

Figure: GD converges slow for ill-conditioned problem

Adaptive gradient method (AdaGrad)

AdaGrad alleviates the “Zig-Zag” phenomenon

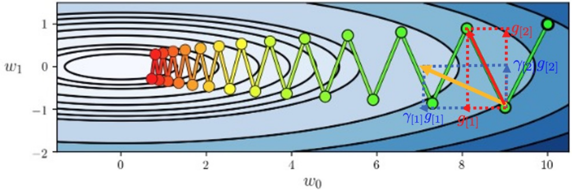

Figure: AdaGrad has alleviated “Zig-Zag” phenomenon

Adaptive gradient method (AdaGrad)

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

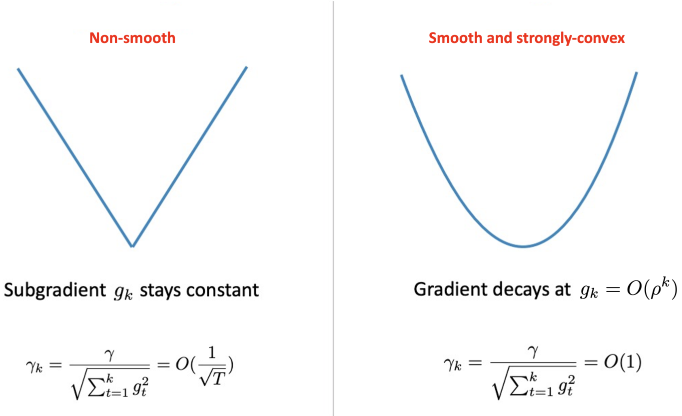

Figure: AdaGrad automatically adapts to problem structure[1].

[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

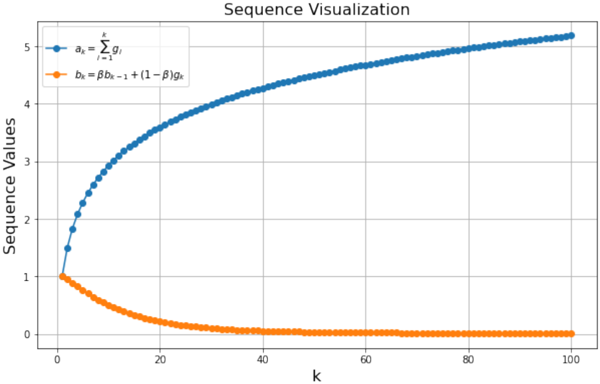

Figure: AdaGrad increases very fast while RMSProp decays slowly with $\beta = 0.9$

RMSProp

- We also visualize s_k from RMSProp with different β.

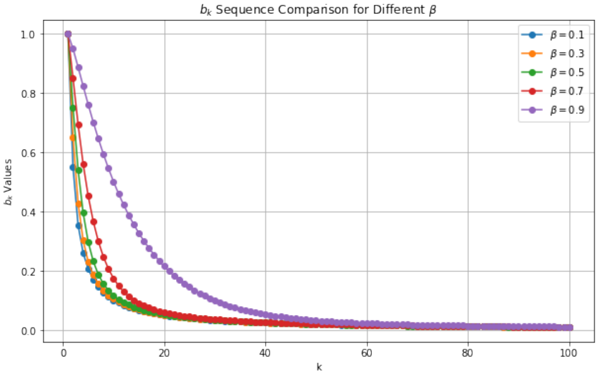

Figure: Gradient accumulation in RMSProp with different β.

RMSProp

- RMSProp has the following update

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

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

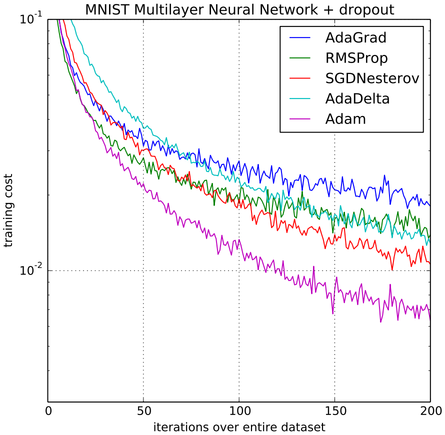

Figure: This figure is from the Adam paper (Kingma and Ba, 2014)

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.