Adam

Dr. Dingzhu Wen

School of Information Science and Technology (SIST)
ShanghaiTech University

wendzh@shanghaitech.edu.cn

March 20, 2024

Overview

Design Challenges of SGD

- How to decide the initial learning rate?
  - In pactice, the hyper-parameters are unknown, e.g., L for smoothness, μ for strong convexity, bounded gradient norm B.
- How to make leanring rate adaptive to the current training iterations, similar to e.g., Newton's method?
- How to design the best learning rate for each parameter?
- How to escape from deep local optimum?

Momentum: Motivations

A function $f : \mathbb{R}^2 \to \mathbb{R}$ is μ-strongly convex and L-smooth:

$$f(w_1, w_2) = \frac{L}{2} w_1^2 + \frac{\mu}{2} w_2^2,$$

where $L \geq \mu > 0$. The optimum is $\mathbf{w}_* = \mathbf{0}$ and the Hessian matrix is

$$\nabla^2 f = \begin{bmatrix} L & 0 \\ 0 & \mu \end{bmatrix}.$$

GD updating with fixed step size η:

$$w_{1,t+1} = w_{1,t} - \eta L w_{1,t},$$
$$w_{2,t+1} = w_{2,t} - \eta \mu w_{2,t},$$

Momentum: Motivations

It follows that

$$w_{1,T} = (1 - \eta L)^T w_{1,0},$$
$$w_{2,T} = (1 - \eta \mu)^T w_{2,0},$$

and

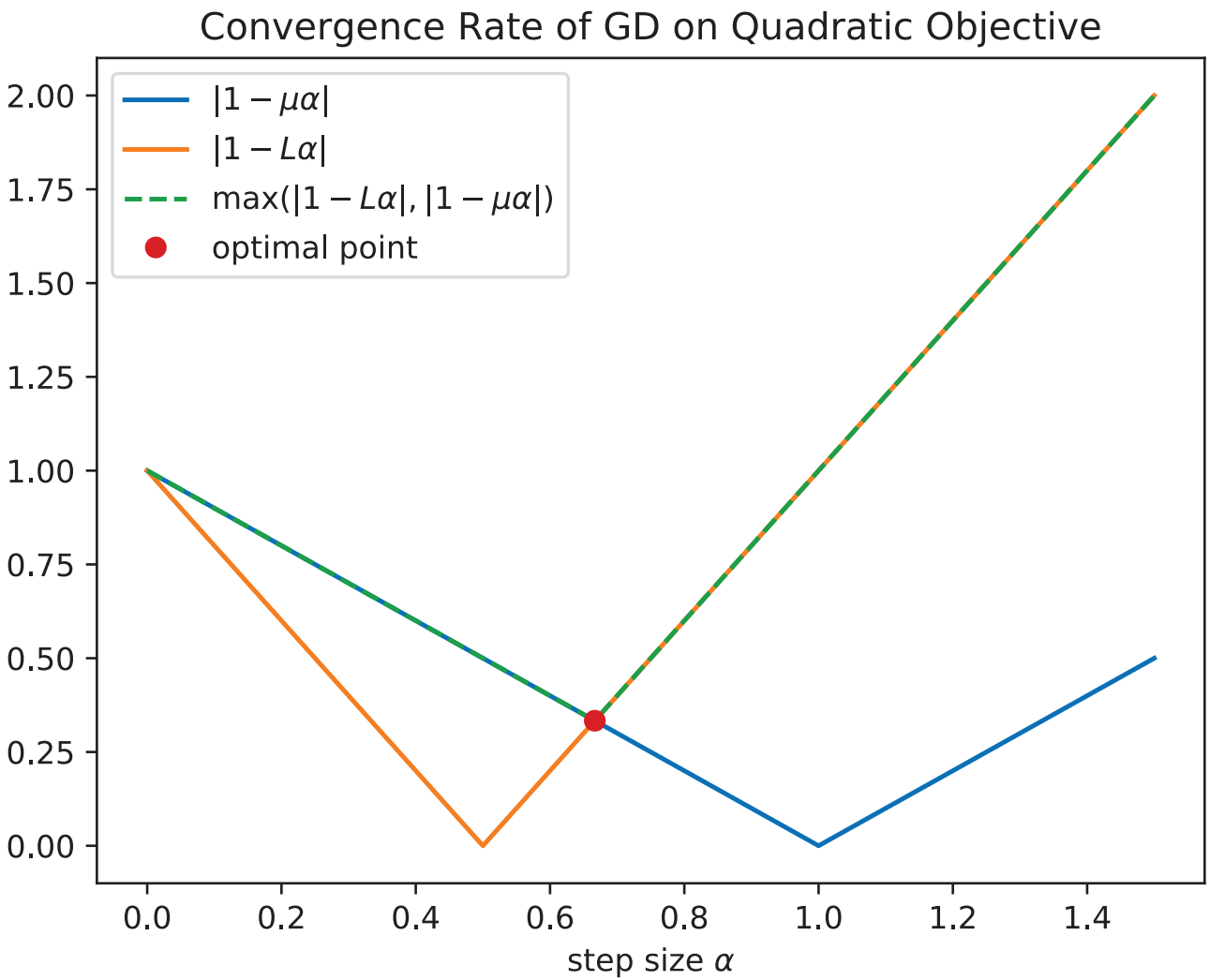
$$f(\mathbf{w}_T) = \frac{L}{2}(1 - \eta L)^{2T} w_{1,0}^2 + \frac{\mu}{2}(1 - \eta \mu)^{2T} w_{1,0}^2.$$

Asymptotically, this will be dominated by whichever term grows slower:

$$f(\mathbf{w}_T) = \mathcal{O}\left(\max(|1 - \eta L|^{2T}, |1 - \eta \mu|^{2T})\right).$$

Momentum: Motivations

The relation of $f(\mathbf{w}_T)$ and η:



Momentum: Motivations

The optimal step size (for convergence) is a trade-off and should be

$$\eta = \frac{2}{L+\mu}.$$

Set the step size larger for dimension with less curvature, and smaller for the dimension with more curvature.

For plain GD or SGD, it's impossible, as the learning rates are the same for all parameters.

Solutions:

- Momentum
- Adaptive learning rates
- Preconditioning

Momentum: Introduction

Target

- Try to tell the difference between more and less curved directions using information already available in gradient descent.

Basic Idea: For one-dimension function,

- If the gradients are reversing sign, then the step size is too large, because we're overshooting the optimum.
- If the gradients are staying in the same direction, then the step size is too small.

Question: **Can we use this to make steps smaller when gradients reverse sign and larger when gradients are consistently in the same direction?**

Polyak Momentum Step

Adds an extra momentum term to gradient descent:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \nabla f(\mathbf{w}_t) + \beta(\mathbf{w}_t - \mathbf{w}_{t-1}).$$

Intuition:

- If the current gradient is in the same direction as the previous step, move a little further in the same direction.
- If it's in the opposite direction, move a little less far.
- This is also known as the **heavy ball method** because it approximately simulates the dynamics of a ball that has physical momentum sliding through the space we want to optimize over.

Analysis of Polyak Momentum

The momentun method is suitable for convex functions, otherwise, it's easier to be trapped by a local optimum.
A simple example to show the intuition: The **single dimensional quadratics**.

$$f(w) = \frac{\lambda}{2} w^2.$$

Polyak Momentum Step: $w_{t+1} = w_t - \eta\lambda w_t + \beta(w_t - w_{t-1})$.
Vector Form:

$$[w_{t+1}, w_t]^T = \mathbf{M}[w_t, w_{t-1}]^T,$$

M companion matrix:

$$\mathbf{M} = \begin{bmatrix} (1 + \beta - \eta\lambda) & -\beta \\ 1 & 0 \end{bmatrix}$$

Analysis of Polyak Momentum

- Denote

$$\mathbf{x}_{t+1} = \begin{bmatrix} w_{t+1} \\ w_t \end{bmatrix}.$$

- By induction,

$$\mathbf{x}_{t+1} = \mathbf{M}\mathbf{x}_t = \mathbf{M}^2\mathbf{x}_{t-1} = \ldots = \mathbf{M}^t\mathbf{x}_1.$$

- Eigen-Decomposition of companion matrix: $\mathbf{M} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{-1}$.
- It follows that $\mathbf{M}^t = \mathbf{Q}\mathbf{\Lambda}^t\mathbf{Q}^{-1}$.
- Then,

$$\begin{aligned} \|\mathbf{x}_{t+1}\| &= \left\|\mathbf{Q}\mathbf{\Lambda}^t\mathbf{Q}^{-1}\mathbf{x}_1\right\|, \\ &\leq \|\mathbf{Q}\| \left\|\mathbf{\Lambda}^t\right\| \left\|\mathbf{Q}^{-1}\right\| \|\mathbf{x}_1\|. \end{aligned}$$

Analysis of Polyak Momentum

- It can be further derived that

$$\begin{aligned}\|\mathbf{x}_{t+1}\| &\leq \|\mathbf{Q}\| \left\|\mathbf{\Lambda}^t\right\| \left\|\mathbf{Q}^{-1}\right\|^2 \|\mathbf{x}_1\|^2, \\ &\leq \|\mathbf{Q}\| \left\|\mathbf{Q}^{-1}\right\| \|\mathbf{x}_1\| \left\|\mathbf{\Lambda}^t\right\|.\end{aligned}$$

- Convergnece:

$$\left\|\mathbf{\Lambda}^t\right\| = \rho(\mathbf{M})^t = \max_{\gamma_i}\{|\gamma_i|^t\}.$$

  - Momentun design makes $\mathbf{Q}$ invertible. (Otherwise, it's equivalent to no momentum term.)

Analysis of Polyak Momentum

Eigenvalues of $\mathbf{M}$:

$$|\gamma \mathbf{I} - \mathbf{M}| = \gamma^2 - (1 + \beta - \eta\lambda)\gamma + \beta = 0.$$

The solution of γ is

$$\begin{cases} \gamma_1 = \dfrac{(1 + \beta - \eta\lambda) + \sqrt{(1 + \beta - \eta\lambda)^2 - 4\beta}}{2}, \\ \gamma_2 = \dfrac{(1 + \beta - \eta\lambda) - \sqrt{(1 + \beta - \eta\lambda)^2 - 4\beta}}{2}. \end{cases}$$

Question: $(1 + \beta - \eta\lambda)^2 - 4\beta$ v.s. 0

Analysis of Polyak Momentum

$(1+\beta-\eta\lambda)^2 - 4\beta < 0.$

- γ_1, γ_2 are compelx numbers.
- $\gamma_1 = \gamma_2^*$.
- $|\gamma_1| = |\gamma_2| = \sqrt{\beta}$.
- That says

$$\mathbf{x}_{t+1} \leq \beta^{t/2} C \mathbf{x}_1,$$

  where C is a constant.
- Linear convergence rate regardless the selection of step size η, as long as the following condition is satisfied.

$$-1 \leq \frac{1+\beta-\eta\lambda}{2\sqrt{\beta}} \leq 1.$$

Analysis of Polyak Momentum

Multi-Dimensional Qudratics

$$f(\mathbf{w}) = \sum_i \frac{\lambda_i}{2} w_i^2.$$

Similarly, if the following conditions are satisfied, linear convergence rate can be achieved.

$$-1 \leq \frac{1+\beta-\eta\lambda_i}{2\sqrt{\beta}} \leq 1, \quad \forall i.$$

Besides, $f(\cdot)$ is assumed to be μ-strongly convex and L-smooth, i.e.,

$$\mu \leq \lambda_i \leq L, \quad \forall i.$$

Then, a sufficient condition is that

$$-1 \leq \frac{1+\beta-\eta L}{2\sqrt{\beta}} \text{ and } \frac{1+\beta-\eta\mu}{2\sqrt{\beta}} \leq 1.$$

Analysis of Polyak Momentum

Multi-Dimensional Qudratics For any functions, which are μ-strongly convex and L-smooth, as long as

$$-1 \leq \frac{1+\beta-\eta L}{2\sqrt{\beta}} \text{ and } \frac{1+\beta-\eta\mu}{2\sqrt{\beta}} \leq 1,$$

linear convergence rate can be achieved.
Select β as small as possible to enbale fast convergence.

- Paractically, $\beta = 0.9$, or use other hyper-lparameter optimization method.
- Additional computation and memory cost.
- Convergence rate compared to GD without momentum?

Other Momentum Methods

Nesterov momentum step:

$$\mathbf{v}_{t+1} = \mathbf{w}_t - \eta \nabla f(\mathbf{w}_t),$$
$$\mathbf{w}_{t+1} = \mathbf{v}_{t+1} + \beta(\mathbf{v}_{t+1} - \mathbf{v}_t).$$

Momentum with SGD:

$$\mathbf{v}_{t+1} = \beta \mathbf{v}_t - \eta \nabla f_{i_t}(\mathbf{w}_t),$$
$$\mathbf{w}_{t+1} = \mathbf{w}_t + \mathbf{v}_{t+1}.$$

Preconditioning

To achieve fast convergence, we should set the step size larger for dimension with less curvature, and smaller for the dimension with more curvature.

A quantratric function:

$$f(w_1, w_2) = \frac{L}{2} w_1^2 + \frac{\mu}{2} w_2^2,$$

where $L \geq \mu > 0$.

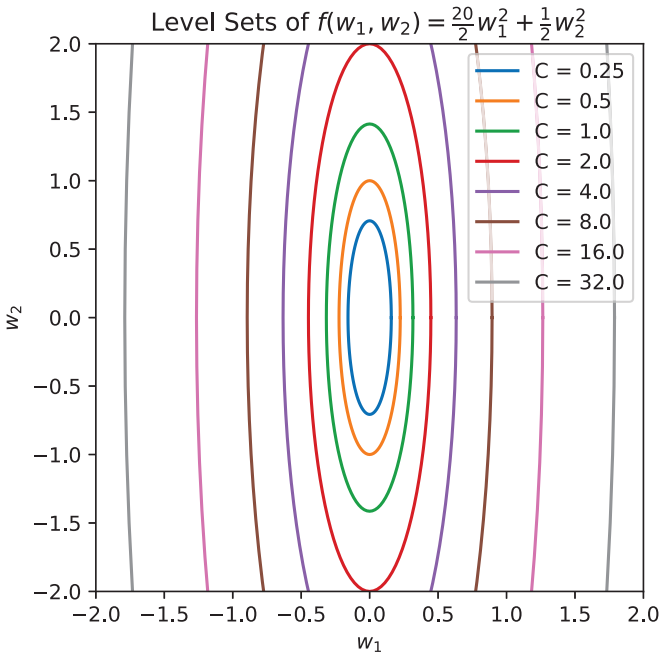
$$w_{1,t+1} = w_{1,t} - \eta L w_{1,t},$$
$$w_{2,t+1} = w_{2,t} - \eta \mu w_{2,t},$$

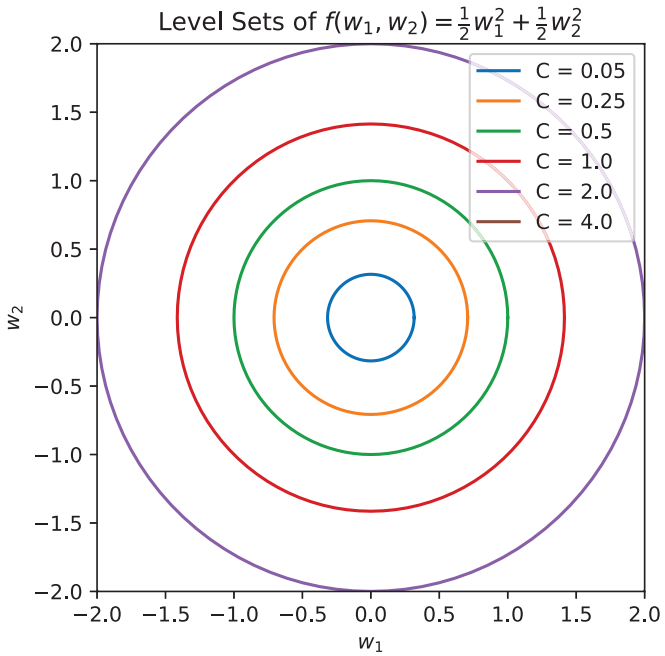
- Different parameters (dimensions) have different curvature.

Preconditioning

Value leve:

$$C = \frac{L}{2}w_1^2 + \frac{\mu}{2}w_2^2.$$





- Same learning rate for different parameters will decrease the convergence rate.

Preconditioning

- Main Idea: Rescale the underlying space we're optimizing over to make the level sets more like circles.
- Use an affine transform with a fixed matrix $\mathbf{R}$:

$$g(\mathbf{u}) = f(\mathbf{R}\mathbf{u}),$$

  where different dimensions of $\mathbf{u}$ have same curvature.
- GD Iteration:

$$\mathbf{u}_{t+1} = \mathbf{u}_t - \eta \nabla g(\mathbf{u}_t) = \mathbf{u}_t - \eta \mathbf{R}^T \nabla f(\mathbf{R}\mathbf{u}_t).$$

- Let $\mathbf{w} = \mathbf{R}\mathbf{u}$:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \mathbf{R}\mathbf{R}^T \nabla f(\mathbf{w}_t).$$

Preconditioning

- Run GD with gradients scaled by some positive semidefinite matrix:

$$\mathbf{P} = \mathbf{R}\mathbf{R}^T$$

.

Question: Why $\mathbf{P}$ is positive semidefinite?

- The same operation can be extended to SGD.
  Question: What would be the best $\mathbf{P}$ for qudratic functions?
- For large-dimensional models, how to efficiently get $\mathbf{P}$?
  - Memory cost d^2,
  - Computation overhead $\mathcal{O}(d^2)$.
  - Diagonal preconditioners: Restrict $\mathbf{P}$ to be a diagonal matrix.

Preconditioning

How to choose the preconditioner?

- From experience (intuition): You may know from the formula for the loss that some dimensions are more curved than others.
- Use statistics from the dataset: For example, for a linear model you could precondition based on the variance of the features in your dataset.
- Use information from the matrix of second-partial derivatives: For example, you could use a preconditioning matrix that is a diagonal approximation of the Newton's method update at some point. These methods are sometimes called Newton Sketch methods.

Adapative Learning Rates

Basic Idea: Adjust the learning rate per-component dynamically at each timestep based on observed statistics of the gradients.

SGD Iteration:

$$w_{j,t+1} = w_{j,t} - \eta_{j,t} \frac{\partial f_{i_t}}{\partial w_{j,t}}.$$

Adam: A method for stochastic optimization

Adam: Adaptive moment estimation.

- Efficient stochastic optimization requiring only first order gradients with little memory (compared to second order methods).
- Both first-order and second order moments are used.
- Individual adaptive learning rates for different parameters.

Adam: Algorithm

In an arbitrary training iteration,

- **Step 1**: Calculate first order stochastic gradient vector:

$$\mathbf{g}_t = \nabla f_{i_t}(\mathbf{w}_{t-1}).$$

- **Step 2**: Update first order moment vector

$$\mathbf{v}_t = \beta_1 \mathbf{v}_{t-1} + (1-\beta_1)\mathbf{g}_t,$$

  - $\beta_1 \in [0, 1)$ is the exponential decay rate for the moment estimate,
  - $\mathbf{v}_t$ is the first-order moment (exponential moving average of the gradients):

$$\mathbf{v}_t = (1-\beta_1)\sum_{j=1}^{t} \beta_1^{t-j}\mathbf{g}_j.$$

Adam: Algorithm

In the t-th training iteration,

- **Step 3**: Update second order moment

$$\mathbf{m}_t = \beta_2 \mathbf{m}_{t-1} + (1-\beta_2)(\mathbf{g}_t \circ \mathbf{g}_t),$$

  - $\beta_2 \in [0, 1)$ is the exponential decay rate for the moment estimate,
  - $\circ$ represents the element-wise production,
  - $\mathbf{m}_t$ is the exponential moving average of the second order moments.

- **Step 4**: Compute bias-corrected first order moment

$$\hat{\mathbf{v}}_t = \frac{\mathbf{v}_t}{(1-\beta_1^t)},$$

  - $\mathbb{E}[\hat{\mathbf{v}}_t] = \mathbb{E}[\mathbf{g}_t] + \xi$,
  - $\xi = 0$ if $\mathbb{E}[\mathbf{g}_t]$ is stationary (Not satisfied in practice). Otherwise, it can be kept small since the exponential decay rate of β_1.

Adam: Algorithm

In the t-th training iteration,

- **Step 5**: Compute bias-corrected second order moment

$$\hat{\mathbf{m}}_t = \frac{\mathbf{m}_t}{(1-\beta_2^t)},$$

  - $\mathbb{E}[\hat{\mathbf{m}}_t] = \mathbb{E}[\mathbf{g}_t \circ \mathbf{g}_t] + \delta$,
  - $\delta = 0$ if $\mathbb{E}[\mathbf{g}_t]$ is stationary (Not satisfied in practice). Otherwise, it can be kept small since the exponential decay rate of β_2.

- **Step 6**: Update model parameters

$$w_{k,t} = w_{k,t-1} - \eta \times \frac{\hat{v}_{k,t}}{\sqrt{\hat{m}_{k,t}} + \epsilon},$$

  - η is the learning rate,
  - $w_{k,t}$, $\hat{v}_{k,t}$, and $\hat{m}_{k,t}$ are the corresponding k-th elements.
  - ϵ is a small value to compensate the non-stationary process.

Adam: Convergence

Assumption: $f(\mathbf{w})$ is convex and L-smooth.
Convergence rate:

$$\frac{1}{T}\sum_{t=1}^{T}[f_t(\mathbf{w}_t) - f_t(\mathbf{w}_*)] = \mathcal{O}(\frac{1}{\sqrt{T}}).$$

Adam v.s. SGD

In [1], it is analyzed that Adam has lighter gradient noise, resulting in a fast convergence.

[1] P. Zhou, J. Feng, C. Ma, C. Xiong, S. Hoi, and W. E, "Towards theoretically understanding why SGD generalizes better than ADAM in deep learning," *In Proc. Conf. Neural Information Processing Systems*, 2020.

However, for non-convex functions, Adam is easier to be trapped by (deep) local optimum. As a result, it has weaker generalization capability.

- The generalization of Adam in DNN is not good. That's why SGD is still widely used for training deep networks.

Thank you!

wendzh@shanghaitech.edu.cn