How to improve first-order optimization methods

Geometry, block structure, and randomization

Hanbaek Lyu

Department of Mathematics

University of Wisconsin – Madison

Partially supported by NSF DMS #2206296 and #2010035

Deep Learning seminar @ KRAFTON AI

May. 29, 2024

Students and Collaborators

Yuchen Li
(4th year Ph.D.)

Block Optimization
Riemannian Optimization

William Powell
(3rd year Ph.D.)

Stochastic Optimization
Reinforcement Learning

Danny Duan
(3rd year Ph.D.)

Quasi-Newton Methods

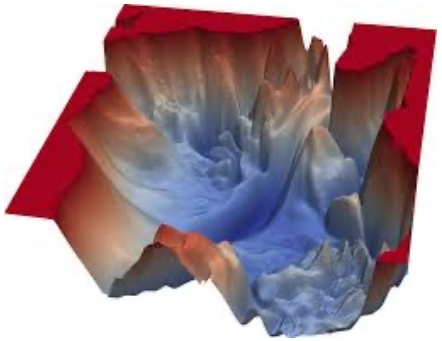
Joowon Lee
(UW Stats)

Causal Inference
Matrix Factorization

Introduction

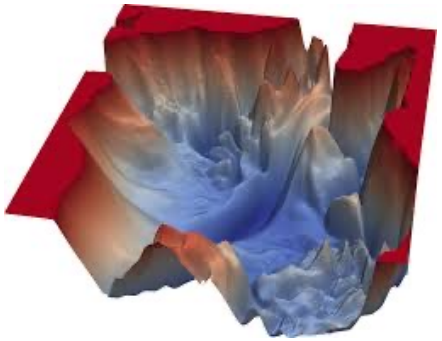
$$\boldsymbol{\theta}^* \in \underset{\boldsymbol{\theta} \in \boldsymbol{\Theta} \subseteq \mathbb{R}^p}{\arg\min} f(\boldsymbol{\theta})$$

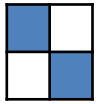
- f: Objective function
  - Nonconvex, smooth (for this talk)
  - Often represents model fitness to training data
    - e.g., DNN, LLM

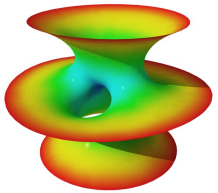


$$\boldsymbol{\theta}^* \in \underset{\boldsymbol{\theta}\in\boldsymbol{\Theta}\subseteq\mathbb{R}^p}{\arg\min}\, f(\boldsymbol{\theta})$$

- f: Objective function
  - Nonconvex, smooth (for this talk)
  - Often represents model fitness to training data
    - e.g., DNN, LLM

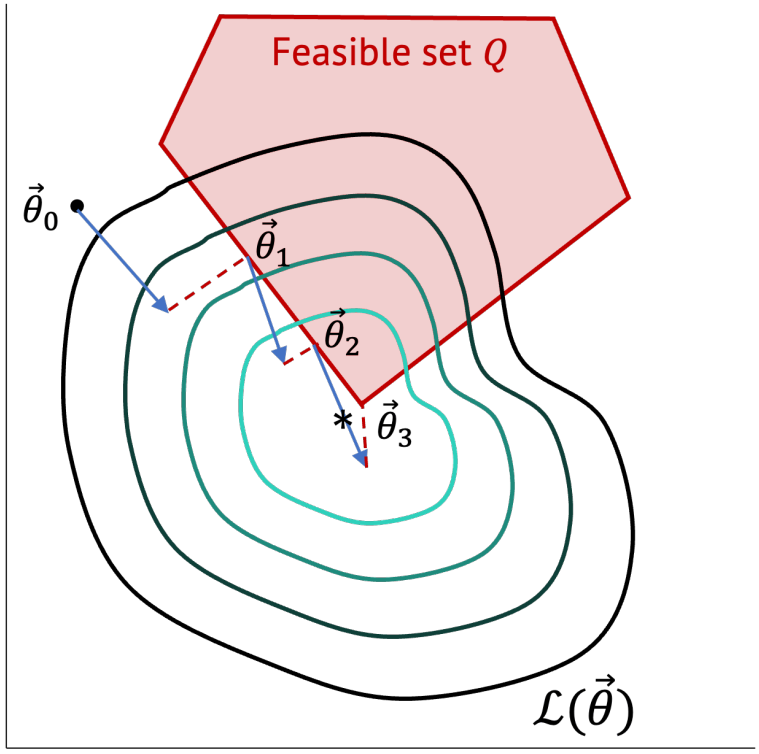


- $\boldsymbol{\Theta}$: Parameter space with various structures:
  - Convex 
  - block structure: $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_m)$ 
  - Riemannian manifold (e.g., $\boldsymbol{\Theta}$=set of orthogonal matrices)



The simplest yet commonly used method

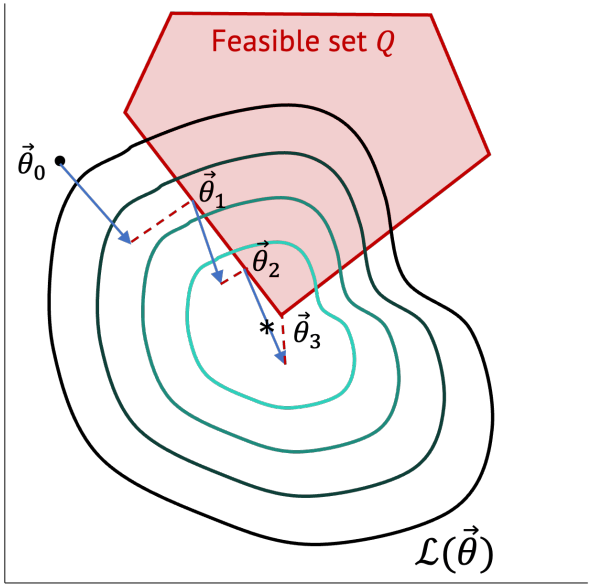
(PGD) $$\boldsymbol{\theta}_{n+1} \leftarrow \Pi_{\boldsymbol{\Theta}} \left(\boldsymbol{\theta}_n - \alpha_n \nabla f(\boldsymbol{\theta}_n) \right)$$



The simplest yet commonly used method

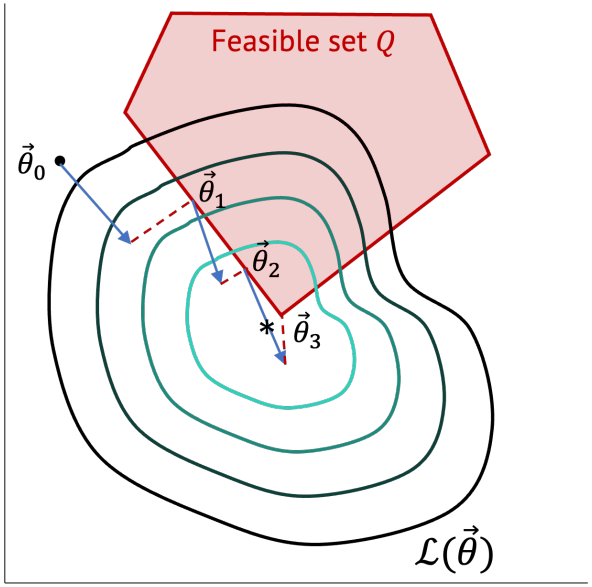
(PGD) $$\boldsymbol{\theta}_{n+1} \leftarrow \Pi_{\boldsymbol{\Theta}}\left(\boldsymbol{\theta}_n - \alpha_n \nabla f(\boldsymbol{\theta}_n)\right)$$

- Projected Gradient Descent (PGD)
  - Widely used in ML, Statistics, Scientific Computing
  - Reduces to Gradient Descent for unconstrained problems
  - Easy to implement, cheap to run
  - Several of modifications (e.g., momentum, adaptive stepsizes)



(PGD) $$\boldsymbol{\theta}_{n+1} \leftarrow \Pi_{\boldsymbol{\Theta}} \left(\boldsymbol{\theta}_n - \alpha_n \nabla f(\boldsymbol{\theta}_n) \right)$$

- Projected Gradient Descent (PGD)
  - Widely used in ML, Statistics, Scientific Computing
  - Reduces to Gradient Descent for unconstrained problems
  - Easy to implement, cheap to run
  - Several of modifications (e.g., momentum, adaptive stepsizes)



(PSGD) $$\boldsymbol{\theta}_{n+1} \leftarrow \Pi_{\boldsymbol{\Theta}} \left(\boldsymbol{\theta}_n - \alpha_n \widehat{\nabla} f(\boldsymbol{\theta}_n) \right)$$

Stochastic gradient $\approx \nabla$

$$\boldsymbol{\theta}_{n+1} \leftarrow \Pi_{\boldsymbol{\Theta}}\left(\boldsymbol{\theta}_n - \mathbf{H}_n \nabla f(\boldsymbol{\theta}_n)\right)$$

- $\Pi_{\boldsymbol{\Theta}}$: Projection onto $\boldsymbol{\Theta}$
  - Simple for nonnegativity or norm constraints
- $\nabla f(\boldsymbol{\theta}_n)$: Gradient of the objective at current parameter θ_n
  - Assume it is cheap to compute; Requires $O(p)$ storage
- $\mathbf{H}_n$: $p \times p$ Preconditioning matrix
  - $\mathbf{H}_n = \alpha_n \mathbf{I}_p$ → Gradient Descent with stepsize α_n
  - $\mathbf{H}_n = \nabla^2 f(\boldsymbol{\theta}_n)^{-1}$ → Newton's Method
  - $\mathbf{H}_n \approx \nabla^2 f(\boldsymbol{\theta}_n)^{-1}$ → Quasi-Newton Method

Plan of the talk

- Classical 1st-order methods like P(S)GD has some limitations
  - Difficult/expensive to project onto the constrained set
  - Still expensive with high dimensionality ($p \gg 1$)
    - e.g., overparametrization
  - Slow convergence for ill-conditioned problems
  - Mathematical guarantees are too pessimistic vs. practice
    - e.g., Usually assumes the worst-case, requires very small stepsizes, i.i.d. data sampling,

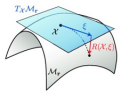
Plan of the talk

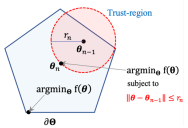
- Classical 1st-order methods like P(S)GD has some limitations
  - Difficult/expensive to project onto the constrained set
  - Still expensive with high dimensionality ($p \gg 1$)
    - e.g., overparametrization
  - Slow convergence for ill-conditioned problems
  - Mathematical guarantees are too pessimistic vs. practice
    - e.g., Usually assumes the worst-case, requires very small stepsizes, i.i.d. data sampling,
- We will propose several techniques to improve them, each based on some mathematical principles

Plan of the talk

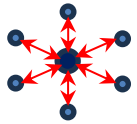
- Classical 1st-order methods like P(S)GD has some limitations
  - Difficult/expensive to project onto the constrained set
  - Still expensive with high dimensionality ($p \gg 1$)
    - e.g., overparametrization
  - Slow convergence for ill-conditioned problems
  - Mathematical guarantees are too pessimistic vs. practice
    - e.g., Usually assumes the worst-case, requires very small stepsizes, i.i.d. data sampling,

- We will propose several techniques to improve them, each based on some mathematical principles
  - Riemannian OPT
  - Adaptive BCD
  - Non-adaptive TR





  - Rand. Hessian Approx.
  - Gradient Consensus



A Math Warm-up:

Basic Ideas behind nonconvex optimization algorithms

Taylor expansion!

Frank-Wolfe (1956)

- **Taylor expand the objective f near θ_n:**

$$f(\boldsymbol{\theta}) \approx f(\boldsymbol{\theta}_n) + \langle \nabla f(\boldsymbol{\theta}_n), \boldsymbol{\theta} - \boldsymbol{\theta}_n \rangle$$

- **Minimize the 1st-order Taylor approximation**

$$\boldsymbol{\theta}'_n = \arg\min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \langle \nabla f(\boldsymbol{\theta}_n), \boldsymbol{\theta} - \boldsymbol{\theta}_n \rangle$$

$$\boldsymbol{\theta}_{n+1} = \text{Convex combination of } \boldsymbol{\theta}_n \text{ and } \boldsymbol{\theta}'_n$$

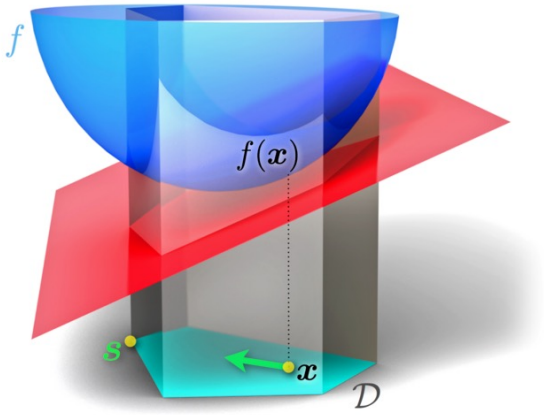


Image credit: Wikipedia

Newton's Method (1690)

(Assume $\Theta = \mathbb{R}^p$)

- **Taylor expand the objective f near θ_n:**

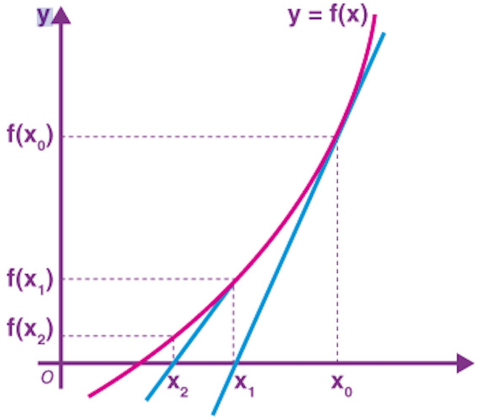
$$f(\boldsymbol{\theta}) \approx f(\boldsymbol{\theta}_n) + \langle \nabla f(\boldsymbol{\theta}_n), \boldsymbol{\theta} - \boldsymbol{\theta}_n \rangle + \frac{1}{2} \left\langle \boldsymbol{\theta} - \boldsymbol{\theta}_n, \nabla f^2(\boldsymbol{\theta}_n)(\boldsymbol{\theta} - \boldsymbol{\theta}_n) \right\rangle$$

- **Minimize the 2nd-order Taylor expansion**

$$\begin{aligned} \boldsymbol{\theta}_{n+1} &= \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^p} f(\boldsymbol{\theta}_n) + \langle \nabla f(\boldsymbol{\theta}_n), \boldsymbol{\theta} - \boldsymbol{\theta}_n \rangle + \frac{1}{2} \left\langle \boldsymbol{\theta} - \boldsymbol{\theta}_n, \nabla f^2(\boldsymbol{\theta}_n)(\boldsymbol{\theta} - \boldsymbol{\theta}_n) \right\rangle \\ &= \boldsymbol{\theta}_n - \left(\nabla^2 f(\boldsymbol{\theta}_n) \right)^{-1} \nabla f(\boldsymbol{\theta}_n) \end{aligned}$$

- Undefined if the Hessian is not PD
- Inverting the Hessian is expensive

Super-fast, quadratic convergence if works ☺



A Quasi-Newton Method

(Assume $\Theta = \mathbb{R}^p$)

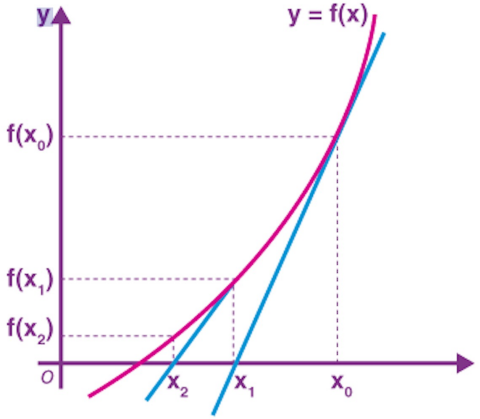
- **Taylor expand the objective f near θ_n:**

$$f(\boldsymbol{\theta}) \approx f(\boldsymbol{\theta}_n) + \langle \nabla f(\boldsymbol{\theta}_n), \boldsymbol{\theta} - \boldsymbol{\theta}_n \rangle + \frac{1}{2} \left\langle \boldsymbol{\theta} - \boldsymbol{\theta}_n, \nabla f^2(\boldsymbol{\theta}_n)(\boldsymbol{\theta} - \boldsymbol{\theta}_n) \right\rangle$$

- **Minimize the 2nd-order Taylor expansion with regularization**

$$\begin{aligned} \boldsymbol{\theta}_{n+1} &= \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^p} \langle \nabla f(\boldsymbol{\theta}_n), \boldsymbol{\theta} - \boldsymbol{\theta}_n \rangle + \frac{1}{2} \left\langle \boldsymbol{\theta} - \boldsymbol{\theta}_n, \nabla f^2(\boldsymbol{\theta}_n)(\boldsymbol{\theta} - \boldsymbol{\theta}_n) \right\rangle + \frac{\gamma_n}{2} \|\boldsymbol{\theta} - \boldsymbol{\theta}_n\|^2 \\ &= \boldsymbol{\theta}_n - \left(\nabla^2 f(\boldsymbol{\theta}_n) + \gamma_n \mathbf{I}_p \right)^{-1} \nabla f(\boldsymbol{\theta}_n) \end{aligned}$$

  - Levenberg-Marquardt regularization ('44 '63)
  - Can make the regularized Hessian PD
  - Matrix inversion still expensive



Trust-Region (1970's)

- **Taylor expand the objective f near θ_n:**

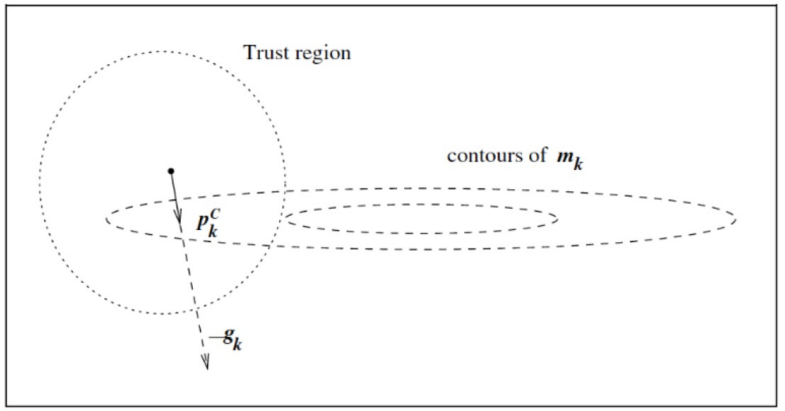
$$f(\boldsymbol{\theta}) \approx f(\boldsymbol{\theta}_n) + \langle \nabla f(\boldsymbol{\theta}_n), \boldsymbol{\theta} - \boldsymbol{\theta}_n \rangle + \frac{1}{2} \langle \boldsymbol{\theta} - \boldsymbol{\theta}_n, \nabla f^2(\boldsymbol{\theta}_n)(\boldsymbol{\theta} - \boldsymbol{\theta}_n) \rangle$$

- **Minimize a "quadratic model" within a trust-region**

$$\boldsymbol{\theta}_{n+1} = \underset{\boldsymbol{\theta} \in \boldsymbol{\Theta},\, \|\boldsymbol{\theta} - \boldsymbol{\theta}_n\| \le r_n}{\arg\min} \langle \nabla f(\boldsymbol{\theta}_n), \boldsymbol{\theta} - \boldsymbol{\theta}_n \rangle + \frac{1}{2} \langle \boldsymbol{\theta} - \boldsymbol{\theta}_n, \mathbf{B}_n(\boldsymbol{\theta} - \boldsymbol{\theta}_n) \rangle$$

- Trust-region with radius r_n
- Next radius r_{n+1} is computed adaptively based on the performance of the previous update

- PD, and no need to be close to the Hessian



PGD (Late 1950s)

- **Taylor expand the objective f near θ_n:**

$$f(\boldsymbol{\theta}) \approx f(\boldsymbol{\theta}_n) + \langle \nabla f(\boldsymbol{\theta}_n), \boldsymbol{\theta} - \boldsymbol{\theta}_n \rangle + \frac{1}{2} \left\langle \boldsymbol{\theta} - \boldsymbol{\theta}_n, \nabla f^2(\boldsymbol{\theta}_n)(\boldsymbol{\theta} - \boldsymbol{\theta}_n) \right\rangle$$

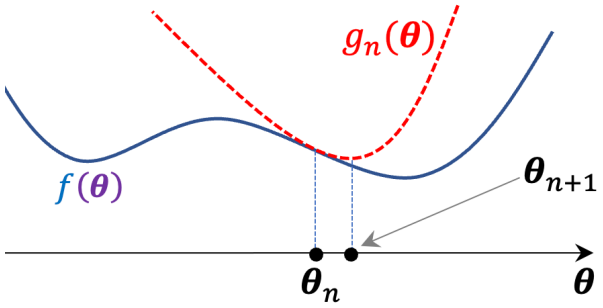
- **Replace the Hessian with L times the identity**

$$f(\boldsymbol{\theta}) \le f(\boldsymbol{\theta}_n) + \langle \nabla f(\boldsymbol{\theta}_n), \boldsymbol{\theta} - \boldsymbol{\theta}_n \rangle + \frac{L}{2} \|\boldsymbol{\theta} - \boldsymbol{\theta}_n\|^2$$

L = largest absolute eigenvalue of $\nabla^2 f(\boldsymbol{\theta})$ over all $\boldsymbol{\theta}$

- **Minimize the quadratic surrogate**

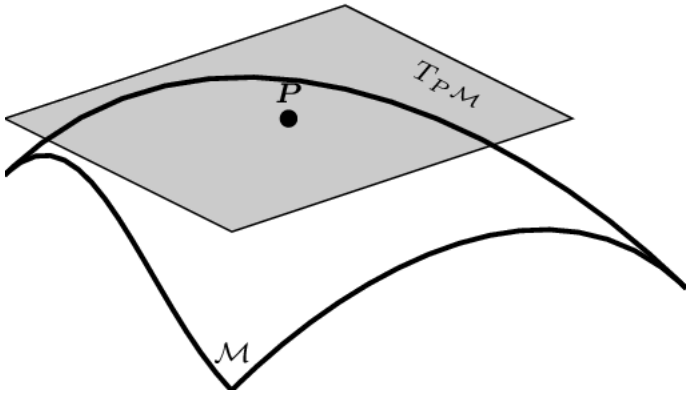
$$\begin{aligned} \boldsymbol{\theta}_{n+1} &= \arg\min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \ \langle \nabla f(\boldsymbol{\theta}_n), \boldsymbol{\theta} - \boldsymbol{\theta}_n \rangle + \frac{L}{2} \|\boldsymbol{\theta} - \boldsymbol{\theta}_n\|^2 \\ &= \arg\min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \ \left\| \boldsymbol{\theta} - \left(\boldsymbol{\theta}_n - \frac{1}{L} \nabla f(\boldsymbol{\theta}_n) \right) \right\|^2 \\ &= \Pi_{\boldsymbol{\Theta}} \left(\boldsymbol{\theta}_n - \frac{1}{L} \nabla f(\boldsymbol{\theta}_n) \right) \end{aligned}$$



A geometric approach:

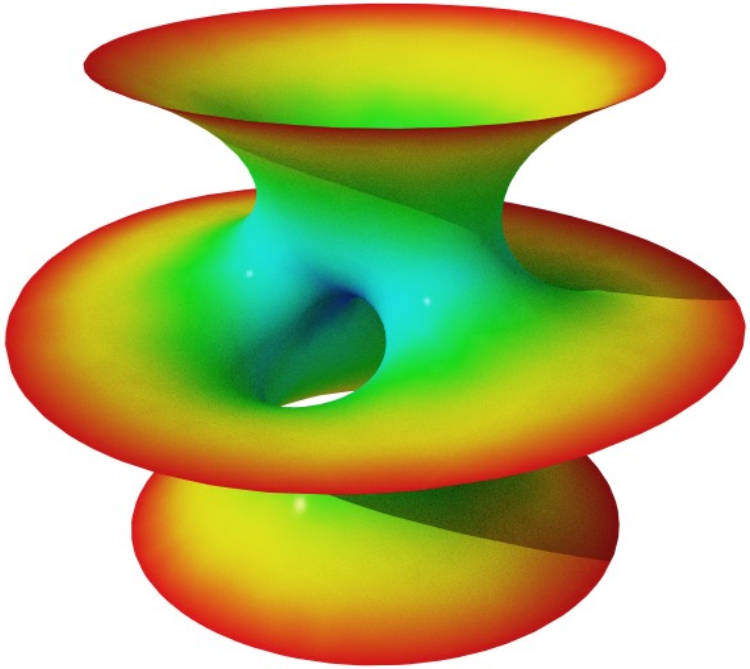
Riemannian Optimization

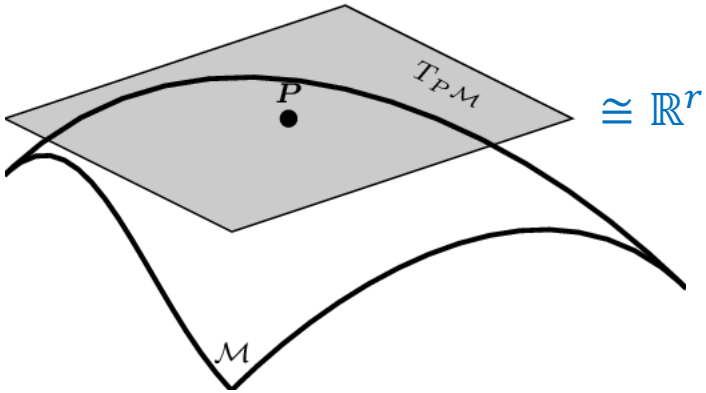
Tangent planes!



- Parameter spaces often has some intrinsic, ***low-dimensional geometric structures***
- **(r-dim) Riemannian manifold**

  = Smooth surface in $\mathbb{R}^p$ where every point has r-dim tangent spaces

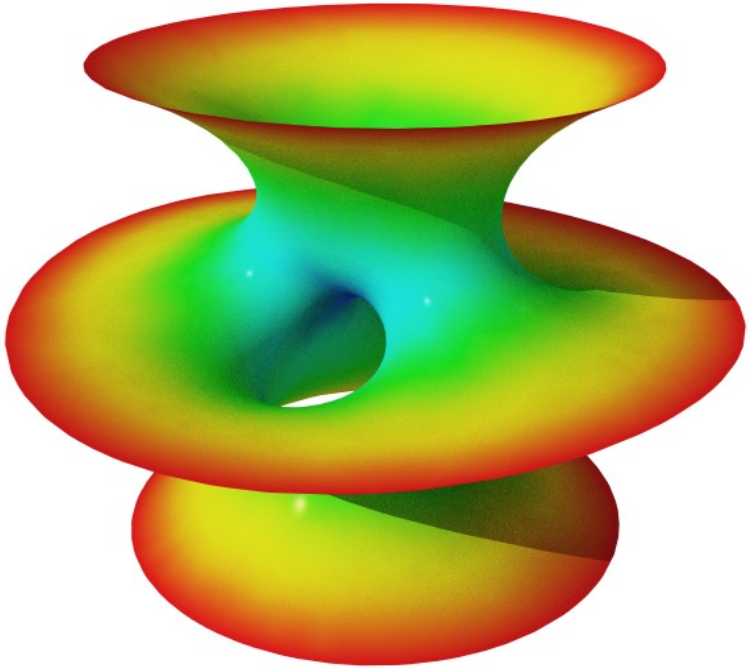




e.g.,
- Low-rank matrix manifolds
- Orthogonal frames
- PSD matrices
- DNNs with spectrum-constrained matrix weights

- Parameter spaces often has some intrinsic, ***low-dimensional geometric structures***
- **(r-dim) Riemannian manifold**

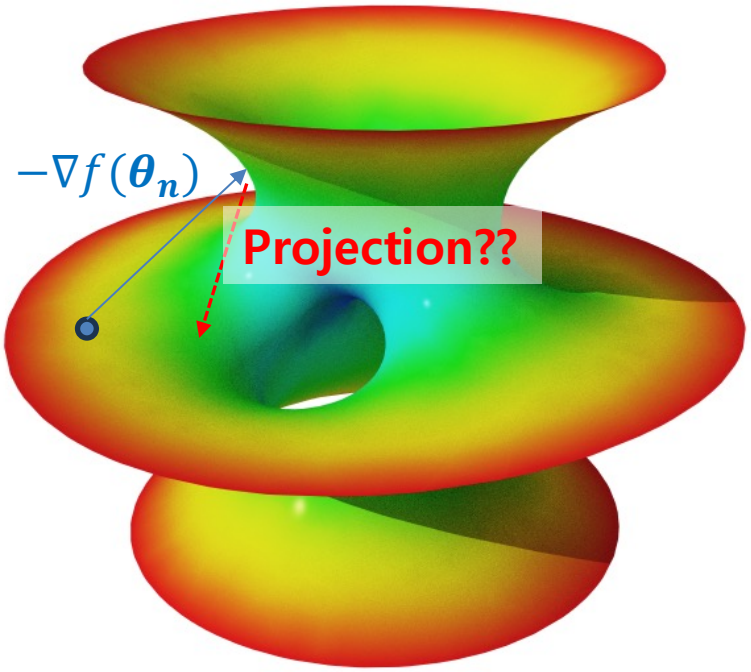
  = Smooth surface in $\mathbb{R}^p$ where every point has r-dim tangent spaces



Riemann
(1826-1866)

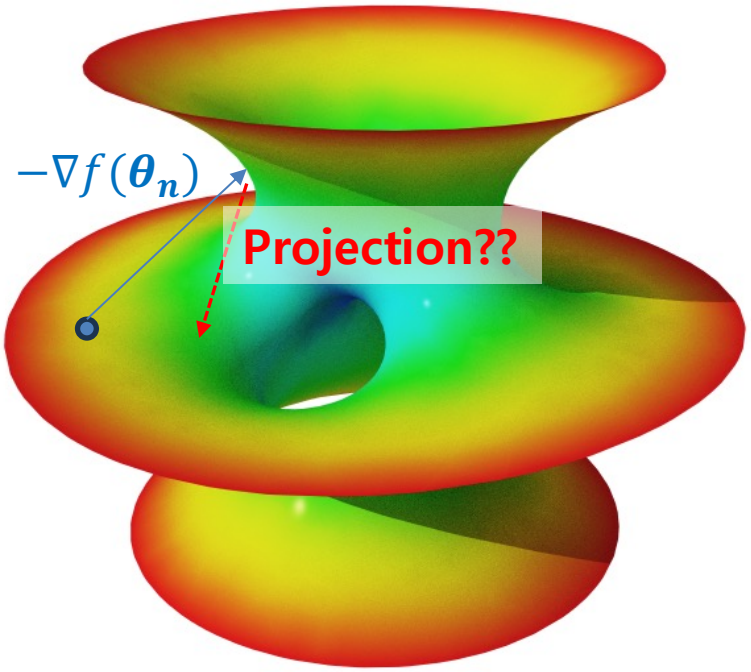
Gauss
(1777-1855)

(PGD) $$\boldsymbol{\theta}_{n+1} \leftarrow \Pi_{\boldsymbol{\Theta}} \left(\boldsymbol{\theta}_n - \alpha_n \nabla f(\boldsymbol{\theta}_n) \right)$$



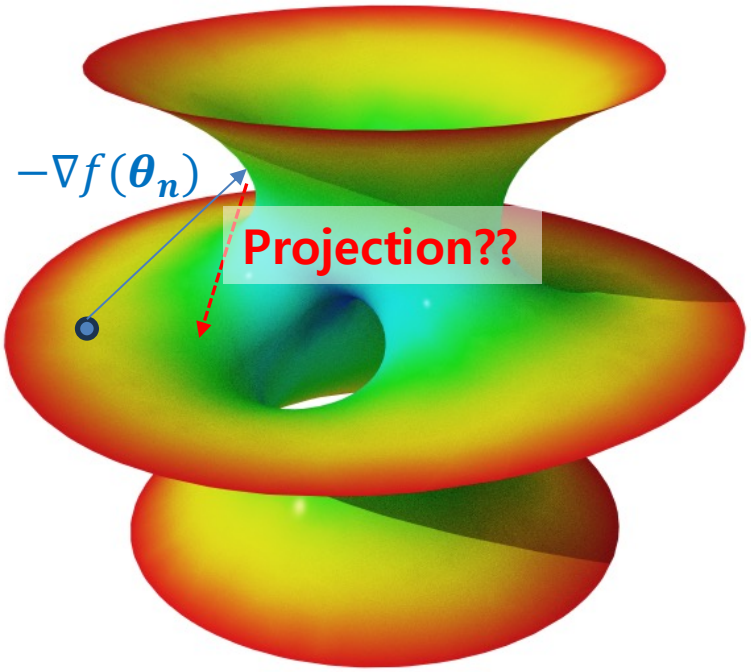
- $\Pi_{\boldsymbol{\Theta}}$: Projection onto $\boldsymbol{\Theta}$:
  - Not always easy!

(PGD) $$\boldsymbol{\theta}_{n+1} \leftarrow \Pi_{\boldsymbol{\Theta}}\left(\boldsymbol{\theta}_n - \alpha_n \nabla f(\boldsymbol{\theta}_n)\right)$$



- $\Pi_{\boldsymbol{\Theta}}$: Projection onto $\boldsymbol{\Theta}$:
  - Not always easy!
- If $\boldsymbol{\Theta}$ is low-dim (say $r \ll p$)
  - Work only with r-dim stuffs?

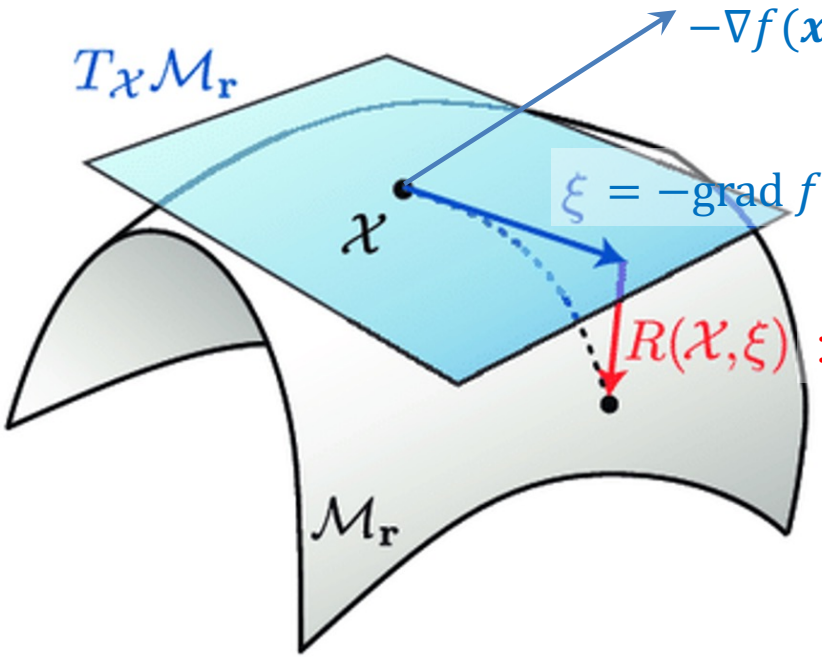
(PGD) $$\boldsymbol{\theta}_{n+1} \leftarrow \Pi_{\boldsymbol{\Theta}}\left(\boldsymbol{\theta}_n - \alpha_n \nabla f(\boldsymbol{\theta}_n)\right)$$



- $\Pi_{\boldsymbol{\Theta}}$: Projection onto $\boldsymbol{\Theta}$:
  - Not always easy!
- If $\boldsymbol{\Theta}$ is low-dim (say $r \ll p$)
  - Work only with r-dim stuffs?
- *For certain low-rank matrix factorization problems, PGD with rSVD as projection can be useful*

  [Lee, Lyu, Yao NeurIPS '23]

(Riemannian GD) $\boldsymbol{\theta}_{n+1} \leftarrow \mathrm{Rtr}_{\boldsymbol{\theta}_n}\left(-\alpha_n \operatorname{grad} f(\boldsymbol{\theta}_n)\right)$



$\xi = -\operatorname{grad} f(x)$: Rimannian gradient

$R(\mathcal{X},\xi)$: Retraction at x

(Proj from $T_x \to \mathcal{M}$)

- RGD = gradient descent within the tangent space + Retraction

(RGD) $$\boldsymbol{\theta}_{n+1} \leftarrow \mathrm{Rtr}_{\boldsymbol{\theta}_n}\left(-\alpha_n \operatorname{grad} f(\boldsymbol{\theta}_n)\right)$$

- Limitation: ***grad*** and ***Retraction*** *can be expensive to compute*

(RGD) $$\boldsymbol{\theta}_{n+1} \leftarrow \mathrm{Rtr}_{\boldsymbol{\theta}_n}\left(-\alpha_n \operatorname{grad} f(\boldsymbol{\theta}_n)\right)$$

- Limitation: ***grad*** and ***Retraction*** *can be expensive to compute*

[Li, Lyu, Balzano, Needell ICML '24]

(Inexact RGD) $$\boldsymbol{\theta}_{n+1} \leftarrow \widehat{\mathrm{Rtr}}_{\boldsymbol{\theta}_n}\left(-\alpha_n \widehat{\operatorname{grad}} f(\boldsymbol{\theta}_n)\right)$$

- ***It's OK to use approximate grad* and *Retraction***

  (Both mathematically and practically)

(RGD) $$\boldsymbol{\theta}_{n+1} \leftarrow \mathrm{Rtr}_{\boldsymbol{\theta}_n}\left(-\alpha_n \operatorname{grad} f(\boldsymbol{\theta}_n)\right)$$

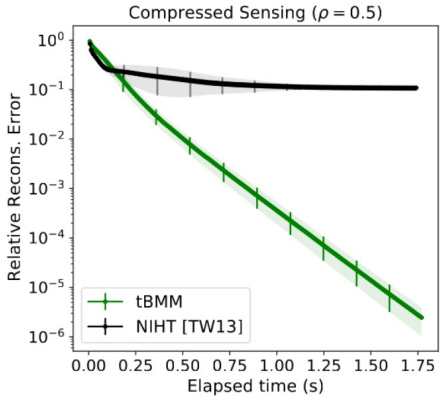
- Limitation: ***grad*** and ***Retraction*** *can be expensive to compute*

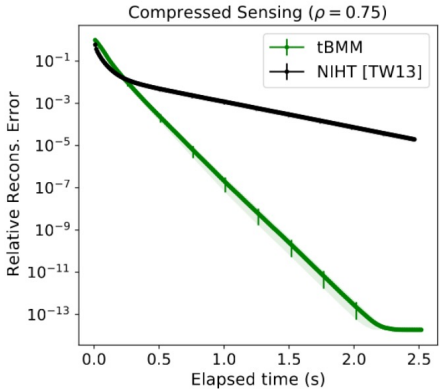
[Li, Lyu, Balzano, Needell ICML '24]

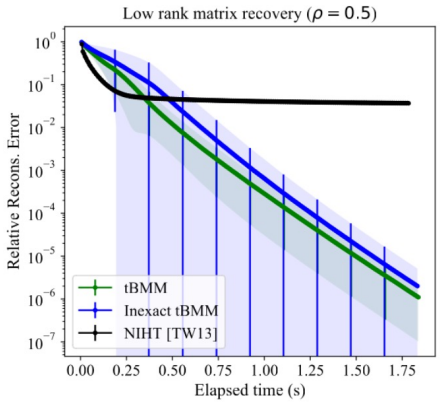
(Inexact RGD) $$\boldsymbol{\theta}_{n+1} \leftarrow \widehat{\mathrm{Rtr}}_{\boldsymbol{\theta}_n}\left(-\alpha_n \widehat{\operatorname{grad}} f(\boldsymbol{\theta}_n)\right)$$

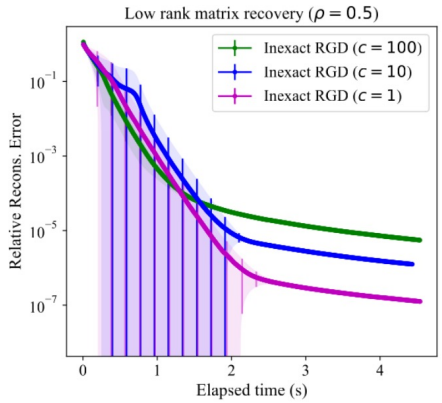
- ***It's OK to use approximate grad* and *Retraction***

(Both mathematically and practically)





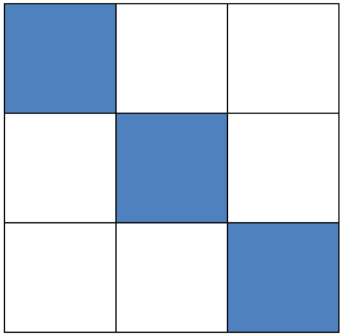




A coordinate-wise approach:

Adaptive Block Coordinate Descent

Optimize one block at a time!



Stepsizes in PGD

(PGD) $$\boldsymbol{\theta}_{n+1} \leftarrow \Pi_{\boldsymbol{\Theta}} \left(\boldsymbol{\theta}_n - \alpha_n \nabla f(\boldsymbol{\theta}_n) \right)$$

- α_n: Stepsizes. How to choose them?

Stepsizes in PGD

(PGD) $$\boldsymbol{\theta}_{n+1} \leftarrow \Pi_{\boldsymbol{\Theta}}\left(\boldsymbol{\theta}_n - \alpha_n \nabla f(\boldsymbol{\theta}_n)\right)$$

- α_n: Stepsizes. How to choose them?
  - "*Small enough stepsize*": $\alpha_n \leq 1/L$, where
    - L = **Lipschitz constant for ∇f over Θ**
      $\approx$ **Largest absolute eigenvalue of $\nabla^2 f$ over Θ**

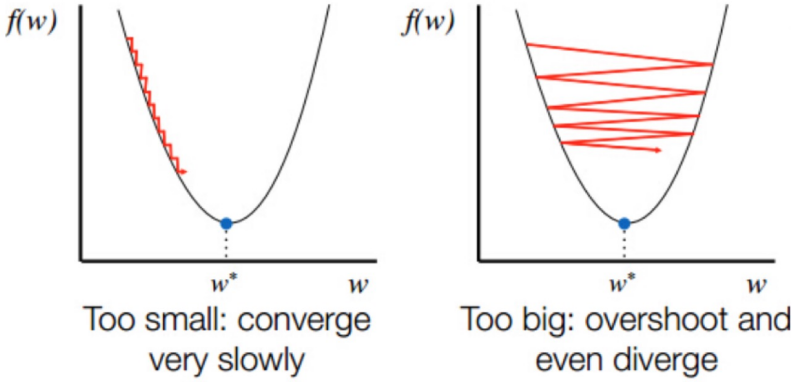
(PGD) $$\boldsymbol{\theta}_{n+1} \leftarrow \Pi_{\boldsymbol{\Theta}}\left(\boldsymbol{\theta}_n - \alpha_n \nabla f(\boldsymbol{\theta}_n)\right)$$

- α_n: Stepsizes. How to choose them?
  - "*Small enough stepsize*": $\alpha_n \leq 1/L$, where
    - $\boldsymbol{L}$ = **Lipschitz constant for ∇f over $\boldsymbol{\Theta}$**
      $\approx$ **Largest absolute eigenvalue of $\nabla^2 f$ over $\boldsymbol{\Theta}$**
  - But $\alpha_n = 1/L$ is TOO SMALL!
    - In practice use "line search" to find larger α_n that works
    - L might be unknown and hard to estimate

(PGD) $$\boldsymbol{\theta}_{n+1} \leftarrow \Pi_{\Theta}\left(\boldsymbol{\theta}_n - \alpha_n \nabla f(\boldsymbol{\theta}_n)\right)$$

- α_n: Stepsizes. How to choose them?
  - "*Small enough stepsize*": $\alpha_n \leq 1/L$, where
    - L = **Lipschitz constant for ∇f over Θ**
      $\approx$ **Largest absolute eigenvalue of $\nabla^2 f$ over Θ**
  - But $\alpha_n = 1/L$ is TOO SMALL!
    - In practice use "line search" to find larger α_n that works
    - L might be unknown and hard to estimate

In practice, performance of P(S)GD depends very sensitively on α_ns



Block PGD = BCD

Two-Block structure : $\boldsymbol{\theta} = (A, B)$, $\boldsymbol{\Theta} = \Theta_A \times \Theta_B$

(Block PGD)
(or BCD)

$$A_{n+1} \leftarrow \Pi_{\Theta_A}\left(A_n - \alpha_n \nabla_A f(A_n, B_n)\right)$$
$$B_{n+1} \leftarrow \Pi_{\Theta_B}\left(B_n - \beta_n \nabla_B f(A_{n+1}, B_n)\right)$$

Two-Block structure : $\boldsymbol{\theta} = (A, B)$, $\boldsymbol{\Theta} = \Theta_A \times \Theta_B$

(Block PGD) (or BCD)

$$A_{n+1} \leftarrow \Pi_{\Theta_A}\left(A_n - \alpha_n \nabla_A f(A_n, B_n)\right)$$
$$B_{n+1} \leftarrow \Pi_{\Theta_B}\left(B_n - \beta_n \nabla_B f(A_{n+1}, B_n)\right)$$

- Known to be much more robust against stepsize choices then PGD (**Why??**)
  - e.g., low-rank matrix factorization, dictionary learning, tensor factorization, kernel learning, linear tranceiver design

Two-Block structure : $\boldsymbol{\theta} = (A, B)$, $\boldsymbol{\Theta} = \Theta_A \times \Theta_B$

(Block PGD) (or BCD)

$$A_{n+1} \leftarrow \Pi_{\Theta_A}\left(A_n - \alpha_n \nabla_A f(A_n, B_n)\right)$$
$$B_{n+1} \leftarrow \Pi_{\Theta_B}\left(B_n - \beta_n \nabla_B f(A_{n+1}, B_n)\right)$$

- Known to be much more robust against stepsize choices then PGD (**Why??**)
  - e.g., low-rank matrix factorization, dictionary learning, tensor factorization, kernel learning, linear tranceiver design

- [Large $L \Leftrightarrow \nabla f$ changes wildly]

  → **Sensitive dependence on stepsize**

Two-Block structure : $\boldsymbol{\theta} = (A, B)$, $\boldsymbol{\Theta} = \Theta_A \times \Theta_B$

(Block PGD) (or BCD)

$$A_{n+1} \leftarrow \Pi_{\Theta_A}\left(A_n - \alpha_n \nabla_A f(A_n, B_n)\right)$$
$$B_{n+1} \leftarrow \Pi_{\Theta_B}\left(B_n - \beta_n \nabla_B f(A_{n+1}, B_n)\right)$$

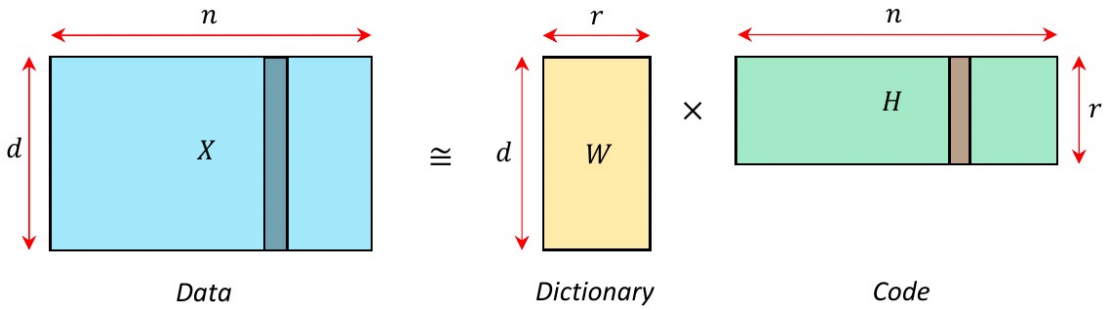
- Known to be much more robust against stepsize choices then PGD (**Why??**)
  - e.g., low-rank matrix factorization, dictionary learning, tensor factorization, kernel learning, linear tranceiver design

- [Large $L \Leftrightarrow \nabla f$ changes wildly]

  → **Sensitive dependence on stepsize**

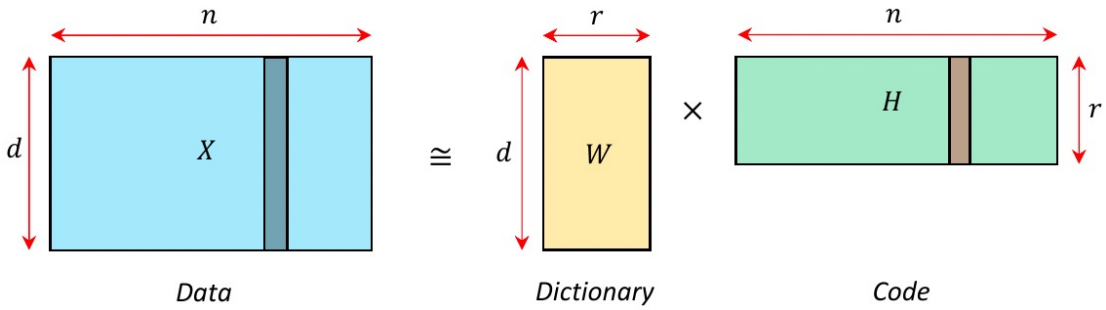
- **Exploiting block structure → The "effective L" is reduced**

Motivating example : **Nonnegative Matrix Factorization (Lee & Seung, Nature '99)**



$$\begin{cases} \text{minimize} & f(W,H) = \|X - WH\|_F^2 & \text{(Reconstruction error)} \\ \text{subject to} & W \in \mathbb{R}^{p\times r}_{\geq 0},\ H \in \mathbb{R}^{r\times n}_{\geq 0} & \text{(Constraints)} \end{cases}$$

Motivating example : **Nonnegative Matrix Factorization (Lee & Seung, Nature '99)**



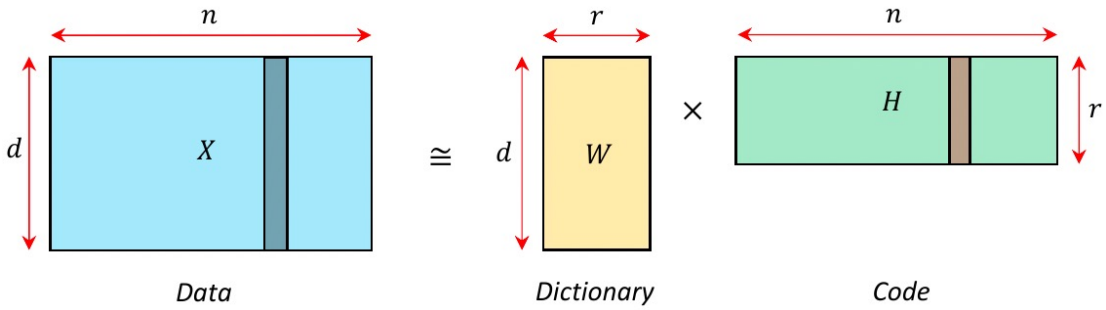
$$\begin{cases} \text{minimize} & f(W,H) = \|X - WH\|_F^2 & \text{(Reconstruction error)} \\ \text{subject to} & W \in \mathbb{R}^{p\times r}_{\geq 0},\ H \in \mathbb{R}^{r\times n}_{\geq 0} & \text{(Constraints)} \end{cases}$$

PGD $\begin{cases} (W_{n+1},H_{n+1}) \leftarrow (W_n,H_n) - \alpha_n(\nabla_W f(W_n,H_n), \nabla_H f(W_n,H_n)) \\ (W_{n+1},H_{n+1}) \leftarrow \max\big(0,(W_{n+1},H_{n+1})\big) \end{cases}$

(Almost no one uses this ☺)

- $f(W,H)$ = Bi-convex
- $\nabla_W f(W,H) = (WH-X)H^T$
- $\nabla_H f(W,H) = W^T(WH-X)$

Motivating example : **Nonnegative Matrix Factorization (Lee & Seung, Nature '99)**



$$\begin{cases} \text{minimize} & f(W,H) = \|X - WH\|_F^2 & \text{(Reconstruction error)} \\ \text{subject to} & W \in \mathbb{R}^{p\times r}_{\geq 0},\ H \in \mathbb{R}^{r\times n}_{\geq 0} & \text{(Constraints)} \end{cases}$$

PGD $\begin{cases} (W_{n+1},H_{n+1}) \leftarrow (W_n,H_n) - \alpha_n(\nabla_W f(W_n,H_n), \nabla_H f(W_n,H_n)) \\ (W_{n+1},H_{n+1}) \leftarrow \max\big(0,(W_{n+1},H_{n+1})\big) \end{cases}$

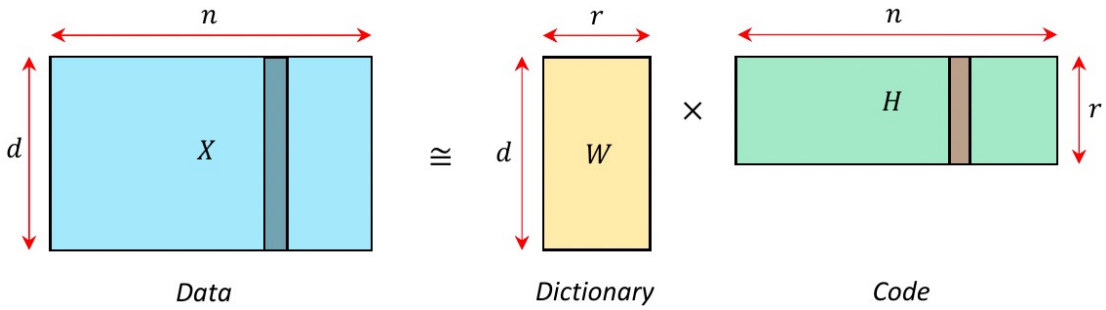
(Almost no one uses this ☺)

- $f(W,H)$ = Bi-convex
- $\nabla_W f(W,H) = (WH-X)H^T$
- $\nabla_H f(W,H) = W^T(WH-X)$

- $\nabla^2 f = \begin{matrix} \\ \mathrm{vec}(\mathbf{W}) \\ \mathrm{vec}(\mathbf{H}) \end{matrix} \begin{matrix} \begin{matrix} \mathrm{vec}(\mathbf{W})^T & \mathrm{vec}(\mathbf{H})^T \end{matrix} \\ \begin{bmatrix} \mathbf{H}\mathbf{H}^T \otimes \mathbf{I}_p & A_{12} \\ A_{12}^T & \mathbf{I}_n \otimes \mathbf{W}^T\mathbf{W} \end{bmatrix} \end{matrix}$

$$A_{12} = [(\mathbf{H}\otimes\mathbf{W}) + \mathbf{I}_r \otimes (\mathbf{W}\mathbf{H} - \mathbf{X})]\,\mathbf{C}^{(r,n)}$$

Motivating example : **Nonnegative Matrix Factorization (Lee & Seung, Nature '99)**



$$\begin{cases} \text{minimize} & f(W,H) = \|X - WH\|_F^2 \\ \text{subject to} & W \in \mathbb{R}^{p\times r}_{\geq 0},\ H \in \mathbb{R}^{r\times n}_{\geq 0} \end{cases} \qquad \begin{matrix} \text{(Reconstruction error)} \\ \text{(Constraints)} \end{matrix}$$

PGD $\begin{cases} (W_{n+1},H_{n+1}) \leftarrow (W_n,H_n) - \alpha_n(\nabla_W f(W_n,H_n), \nabla_H f(W_n,H_n)) \\ (W_{n+1},H_{n+1}) \leftarrow \max\big(0,(W_{n+1},H_{n+1})\big) \end{cases}$

(Almost no one uses this ☺)

- $f(W,H)$ = Bi-convex
- $\nabla_W f(W,H) = (WH-X)H^T$
- $\nabla_H f(W,H) = W^T(WH-X)$

- $\nabla^2 f = \begin{matrix} \\ \text{vec}(\mathbf{W}) \\ \text{vec}(\mathbf{H}) \end{matrix} \begin{matrix} \begin{matrix} \text{vec}(\mathbf{W})^T & \text{vec}(\mathbf{H})^T \end{matrix} \\ \begin{bmatrix} \mathbf{H}\mathbf{H}^T \otimes \mathbf{I}_p & A_{12} \\ A_{12}^T & \mathbf{I}_n \otimes \mathbf{W}^T\mathbf{W} \end{bmatrix} \end{matrix}$

$$A_{12} = [(\mathbf{H}\otimes\mathbf{W}) + \mathbf{I}_r \otimes (\mathbf{W}\mathbf{H} - \mathbf{X})]\,\mathbf{C}^{(r,n)}$$

L = Max eval over all (W,H)

= *Unbounded!*

Case study: **Supervised Matrix Factorization (Lee, Lyu, Yao ICML '24)**

Adaptive BCD:

$$\mathbf{W} \leftarrow \Pi\left(\mathbf{W} - \frac{1}{\lambda_{\max}(\mathbf{H}\mathbf{H}^T) + \varepsilon}(\mathbf{W}\mathbf{H} - \mathbf{X})\mathbf{H}^T\right),$$
$$\mathbf{H} \leftarrow \Pi'\left(\mathbf{H} - \frac{1}{\lambda_{\max}(\mathbf{W}\mathbf{W}^T) + \varepsilon}\mathbf{W}^T(\mathbf{W}\mathbf{H} - \mathbf{X})\right)$$

Case study: **Supervised Matrix Factorization (Lee, Lyu, Yao ICML '24)**

Adaptive BCD:

$$\mathbf{W} \leftarrow \Pi\left(\mathbf{W} - \frac{1}{\lambda_{\max}(\mathbf{H}\mathbf{H}^T) + \varepsilon}(\mathbf{W}\mathbf{H} - \mathbf{X})\mathbf{H}^T\right),$$
$$\mathbf{H} \leftarrow \Pi'\left(\mathbf{H} - \frac{1}{\lambda_{\max}(\mathbf{W}\mathbf{W}^T) + \varepsilon}\mathbf{W}^T(\mathbf{W}\mathbf{H} - \mathbf{X})\right)$$

- $$\nabla^2 f = \begin{matrix} \\ \mathrm{vec}(\mathbf{W}) \\ \mathrm{vec}(\mathbf{H}) \end{matrix} \begin{matrix} \mathrm{vec}(\mathbf{W})^T & \mathrm{vec}(\mathbf{H})^T \end{matrix} \begin{bmatrix} \mathbf{H}\mathbf{H}^T \otimes \mathbf{I}_p & A_{12} \\ A_{12}^T & \mathbf{I}_n \otimes \mathbf{W}^T\mathbf{W} \end{bmatrix}$$

$\nabla^2_W f(\cdot, \mathbf{H})$ $\quad$ $\nabla^2_H f(\mathbf{W}, \cdot)$

Case study: **Supervised Matrix Factorization (Lee, Lyu, Yao ICML '24)**

Adaptive BCD:

$$\mathbf{W} \leftarrow \Pi\left(\mathbf{W} - \frac{1}{\lambda_{\max}(\mathbf{H}\mathbf{H}^T) + \varepsilon}(\mathbf{W}\mathbf{H} - \mathbf{X})\mathbf{H}^T\right),$$
$$\mathbf{H} \leftarrow \Pi'\left(\mathbf{H} - \frac{1}{\lambda_{\max}(\mathbf{W}\mathbf{W}^T) + \varepsilon}\mathbf{W}^T(\mathbf{W}\mathbf{H} - \mathbf{X})\right)$$

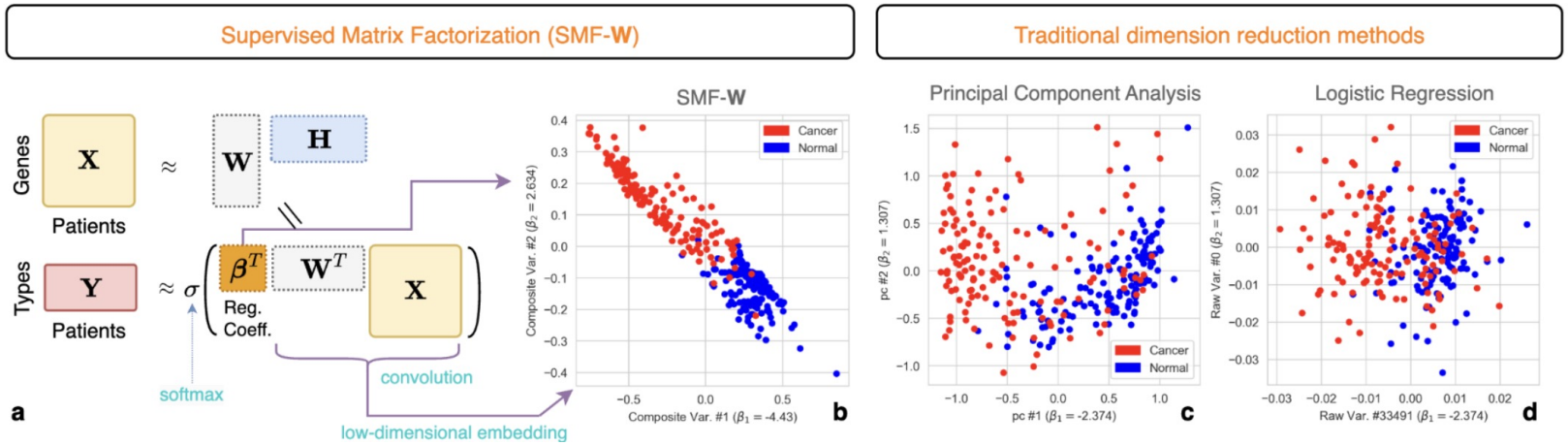
- $$\nabla^2 f = \begin{matrix}\mathrm{vec}(\mathbf{W}) \\ \mathrm{vec}(\mathbf{H})\end{matrix} \begin{matrix}\mathrm{vec}(\mathbf{W})^T \\ \left[\begin{matrix}\mathbf{H}\mathbf{H}^T \otimes \mathbf{I}_p \\ A_{12}^T\end{matrix}\right.\end{matrix} \begin{matrix}\mathrm{vec}(\mathbf{H})^T \\ \left.\begin{matrix}A_{12} \\ \mathbf{I}_n \otimes \mathbf{W}^T\mathbf{W}\end{matrix}\right]\end{matrix}$$

$\nabla^2_W f(\cdot, \mathbf{H})$ $\qquad$ $\nabla^2_H f(\mathbf{W}, \cdot)$

Largest Eval of *diagonal blocks* of the Hessian

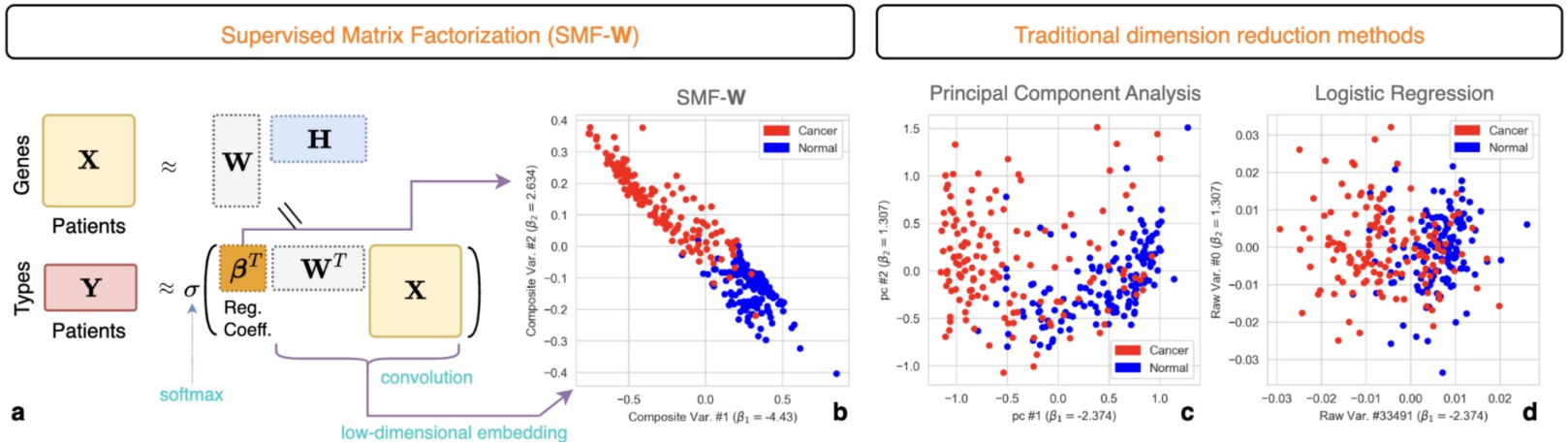
$\ll$ Largest Eval of the entire Hessian

Case study: **Supervised Matrix Factorization (Lee, Lyu, Yao ICML '24)**



Simultaneous dimension reduction and classification

Case study: **Supervised Matrix Factorization (Lee, Lyu, Yao ICML '24)**



Simultaneous dimension reduction and classification

Classification loss Dimension reduction loss

$$\min_{\mathbf{W},\mathbf{H},\boldsymbol{\beta}} f(\mathbf{W},\mathbf{H},\boldsymbol{\beta}) := \sum_{i=1}^{n} \ell(y_i, \boldsymbol{\beta}^T \mathbf{W}^T \mathbf{x}_i) + \xi \|\mathbf{X} - \mathbf{W}\mathbf{H}\|_F^2$$

Negative log-likelihood under logistic model

$$\ell(y, a) = \log(1 + \exp(a)) - \mathbf{1}_{\{y=1\}} a$$

Parameters = Three factor matrices $\boldsymbol{W}, \boldsymbol{H}, \boldsymbol{\beta}$

Block Coordinate Descent : **Optimize one by one, not all at once**

Case study: **Supervised Matrix Factorization (Lee, Lyu, Yao ICML '24)**

Algorithm 1 BCD for SMF-$\mathbf{W}$

Input: $\mathbf{X} \in \mathbb{R}^{p\times n}$ (Data); $\mathbf{X}_{\text{aux}} \in \mathbb{R}^{q\times n}$ (Auxiliary covariate); $\mathbf{Y}_{\text{label}} \in \{0,\dots,\kappa\}^{1\times n}$ (Label);
Constraints: Convex subsets $\mathcal{C}_1 \subseteq \mathbb{R}^{p\times r}$, $\mathcal{C}_2 \subseteq \mathbb{R}^{r\times n}$, $\mathcal{C}_3 \subseteq \mathbb{R}^{r\times\kappa}$, $\mathcal{C}_4 \subseteq \mathbb{R}^{q\times\kappa}$
Parameters: $\xi \geq 0$ (Tuning parameter); $T \in \mathbb{N}$ (number of iterations); $(\eta_{k;i})_{k\geq 1, 1\leq i\leq 4}$ (step-sizes)
Initialize $\mathbf{W} \in \mathcal{C}_1, \mathbf{H} \in \mathcal{C}_2, \boldsymbol{\beta} \in \mathcal{C}_3, \boldsymbol{\Gamma} \in \mathcal{C}_4$
For $k = 1, 2, \dots, T$ **do:** ($\triangleright$ *For* α^+ *see 4.3.*)
  (Update $\mathbf{W}$)
  Update activation $\mathbf{a}_1, \dots, \mathbf{a}_n$ and $\mathbf{K}$
  $\nabla_{\mathbf{W}} f(\mathbf{Z}) \leftarrow \mathbf{X}\mathbf{K}^T\boldsymbol{\beta}^T + 2\xi(\mathbf{W}\mathbf{H} - \mathbf{X})\mathbf{H}^T$
  Choose $\eta_{k,1}^{-1} > L_1 := \alpha^+\|\boldsymbol{\beta}\|_2^2 \cdot \|\mathbf{X}\|_2^2 + 2\xi\|\mathbf{H}\|_2^2$
  $\mathbf{W} \leftarrow \Pi_{\mathcal{C}_1}(\mathbf{W} - \eta_{k;1}\nabla_{\mathbf{W}} f(\mathbf{Z}))$
  (Update $\mathbf{H}$)
  $\nabla_{\mathbf{H}} f(\mathbf{Z}) \leftarrow 2\xi\mathbf{W}^T(\mathbf{W}\mathbf{H} - \mathbf{X})$
  Choose $\eta_{k,2}^{-1} > L_2 := 2\xi\|\mathbf{W}\|_2^2$
  $\mathbf{H} \leftarrow \Pi_{\mathcal{C}_2}(\mathbf{H} - \eta_{k;2}\nabla_{\mathbf{H}} f(\mathbf{Z}))$
  (Update $\boldsymbol{\beta}$)
  Update activation $\mathbf{a}_1, \dots, \mathbf{a}_n$ and $\mathbf{K}$
  $\nabla_{\boldsymbol{\beta}} f(\mathbf{Z}) \leftarrow \mathbf{W}^T\mathbf{X}\mathbf{K}^T$
  Choose $\eta_{k,3}^{-1} > L_3 := \alpha^+\|\mathbf{W}\|_2^2 \cdot \|\mathbf{X}\|_2^2$
  $\boldsymbol{\beta} \leftarrow \Pi_{\mathcal{C}_3}(\boldsymbol{\beta} - \eta_{k;3}\nabla_{\boldsymbol{\beta}} f(\mathbf{Z}))$
  (Update $\boldsymbol{\Gamma}$)
  Update activation $\mathbf{a}_1, \dots, \mathbf{a}_n$ and $\mathbf{K}$
  $\nabla_{\boldsymbol{\Gamma}} f(\mathbf{Z}) \leftarrow \mathbf{X}_{\text{aux}}\mathbf{K}^T$
  Choose $\eta_{k,4}^{-1} > L_4 := \alpha^+\|\mathbf{X}_{\text{aux}}\|_2^2$
  $\boldsymbol{\Gamma} \leftarrow \Pi_{\mathcal{C}_4}(\boldsymbol{\Gamma} - \eta_{k;4}\nabla_{\boldsymbol{\Gamma}} f(\mathbf{Z}))$
End for
Output: $\mathbf{Z} = (\mathbf{W}, \mathbf{H}, \boldsymbol{\beta}, \boldsymbol{\Gamma})$

- $\nabla^2 f =$

	$\text{vec}(\mathbf{W})^T$	$\text{vec}(\mathbf{H})^T$	$\text{vec}(\boldsymbol{\beta})^T$	$\text{vec}(\boldsymbol{\Gamma})^T$
$\text{vec}(\mathbf{W})$	A_{11}	A_{12}	A_{13}	O
$\text{vec}(\mathbf{H})$	A_{21}	A_{22}	O	O
$\text{vec}(\boldsymbol{\beta})$	A_{31}	O	A_{33}	A_{34}
$\text{vec}(\boldsymbol{\Gamma})$	O	O	A_{43}	A_{44}

Lemma D.2 (Derivatives of the SMF-$\mathbf{W}$ objective). *Let $f(\mathbf{Z})$ denote the objective of SMF-$\mathbf{W}$ in (40). Suppose Assumption 4.3 holds. Recall $\dot{\mathbf{h}}$ and $\ddot{\mathbf{H}}$ defined in (44). Then the gradients of $f(\mathbf{Z})$ are given by*

$$\nabla_{\mathbf{W}} f(\mathbf{Z}) = \mathbf{X}\mathbf{K}^T\boldsymbol{\beta}^T + 2\xi(\mathbf{W}\mathbf{H} - \mathbf{X})\mathbf{H}^T, \tag{68}$$
$$\nabla_{\mathbf{H}} f(\mathbf{Z}) = 2\xi\mathbf{W}^T(\mathbf{W}\mathbf{H} - \mathbf{X}), \tag{69}$$
$$\nabla_{\boldsymbol{\beta}} f(\mathbf{Z}) = \mathbf{W}^T\mathbf{X}\mathbf{K}^T, \tag{70}$$
$$\nabla_{\boldsymbol{\Gamma}} f(\mathbf{Z}) = \mathbf{X}_{\text{aux}}\mathbf{K}^T. \tag{71}$$

The block-diagonal terms in the Hessian are given by

$$\nabla_{\text{vec}(\mathbf{W})}\nabla_{\text{vec}(\mathbf{W})^T} f(\mathbf{Z}) = (\boldsymbol{\beta}\otimes\mathbf{X})\mathbf{C}^{(\kappa,n)}\mathbf{M}\mathbf{C}^{(n,\kappa)}(\boldsymbol{\beta}\otimes\mathbf{X})^T + 2\xi(\mathbf{H}\mathbf{H}^T\otimes\mathbf{I}_p), \tag{72}$$
$$\nabla_{\text{vec}(\mathbf{H})}\nabla_{\text{vec}(\mathbf{H})^T} f(\mathbf{Z}) = 2\xi(\mathbf{I}_n\otimes\mathbf{W}^T\mathbf{W}), \tag{73}$$
$$\nabla_{\text{vec}(\boldsymbol{\beta})}\nabla_{\text{vec}(\boldsymbol{\beta})^T} f(\mathbf{Z}) = (\mathbf{I}_\kappa\otimes\mathbf{W}^T\mathbf{X})\mathbf{C}^{(\kappa,n)}\mathbf{M}\mathbf{C}^{(n,\kappa)}(\mathbf{I}_\kappa\otimes\mathbf{W}^T\mathbf{X})^T, \tag{74}$$
$$\nabla_{\text{vec}(\boldsymbol{\Gamma})}\nabla_{\text{vec}(\boldsymbol{\Gamma})^T} f(\mathbf{Z}) = (\mathbf{I}_\kappa\otimes\mathbf{X}_{\text{aux}})\mathbf{C}^{(\kappa,n)}\mathbf{M}\mathbf{C}^{(n,\kappa)}(\mathbf{I}_\kappa\otimes\mathbf{X}_{\text{aux}})^T. \tag{75}$$

The block-off-diagonal terms in the Hessian are given by

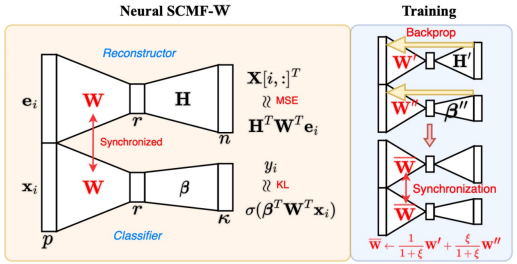
$$\nabla_{\text{vec}(\mathbf{H})}\nabla_{\text{vec}(\mathbf{W})^T} f(\mathbf{Z}) = 2\xi\mathbf{C}^{(n,r)}\left[(\mathbf{H}^T\otimes\mathbf{W}^T) + \mathbf{I}_r\otimes(\mathbf{W}\mathbf{H} - \mathbf{X})^T\right], \tag{76}$$
$$\nabla_{\text{vec}(\boldsymbol{\beta})}\nabla_{\text{vec}(\mathbf{W})^T} f(\mathbf{Z}) = \mathbf{C}^{(\kappa,r)}(\mathbf{I}_r\otimes\mathbf{X}\mathbf{K}^T)^T + (\mathbf{I}_\kappa\otimes\mathbf{W}^T\mathbf{X})\mathbf{C}^{(\kappa,n)}\mathbf{M}\mathbf{C}^{(n,\kappa)}(\boldsymbol{\beta}\otimes\mathbf{X})^T, \tag{77}$$
$$\nabla_{\text{vec}(\boldsymbol{\Gamma})}\nabla_{\text{vec}(\mathbf{W})^T} f(\mathbf{Z}) = \nabla_{\text{vec}(\boldsymbol{\beta})}\nabla_{\text{vec}(\mathbf{H})^T} f(\mathbf{Z}) = \nabla_{\text{vec}(\boldsymbol{\Gamma})}\nabla_{\text{vec}(\mathbf{H})^T} f(\mathbf{Z}) = O, \tag{78}$$
$$\nabla_{\text{vec}(\boldsymbol{\Gamma})}\nabla_{\text{vec}(\boldsymbol{\beta})^T} f(\mathbf{Z}) = (\mathbf{I}_r\otimes\mathbf{X}_{\text{aux}})\mathbf{C}^{(\kappa,n)}\mathbf{M}\mathbf{C}^{(n,\kappa)}(\mathbf{I}_\kappa\otimes\mathbf{W}^T\mathbf{X})^T. \tag{79}$$

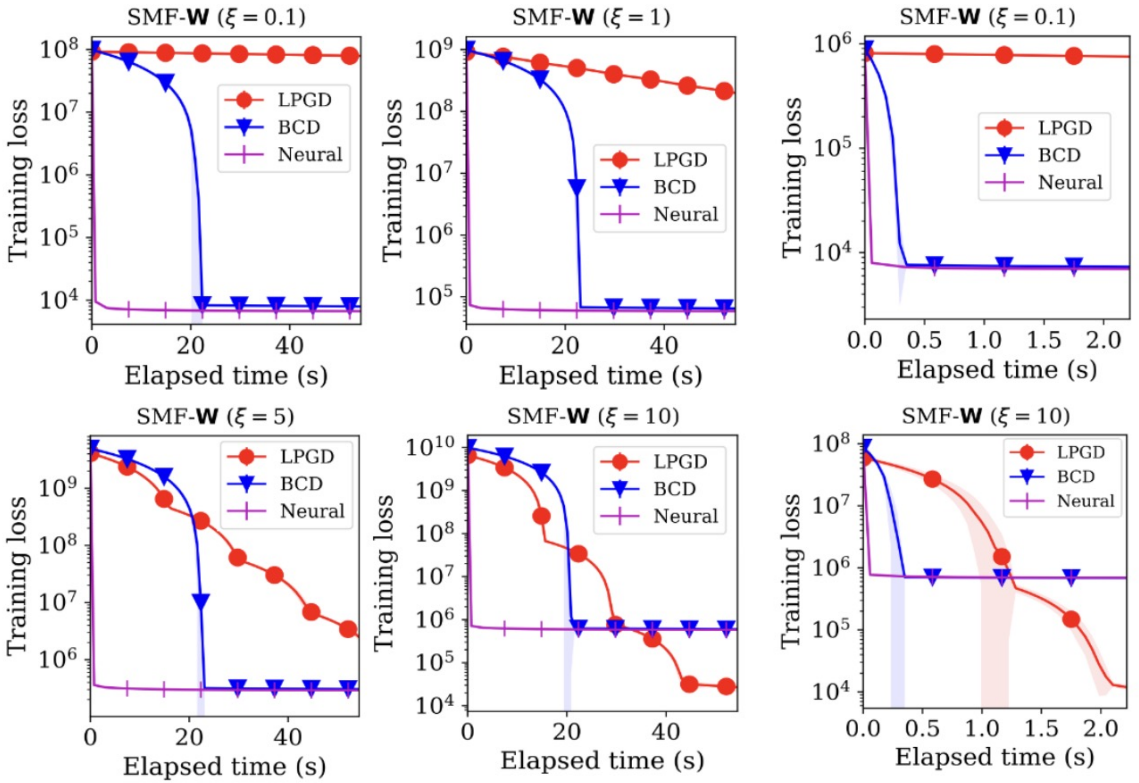
Case study: **Supervised Matrix Factorization (Lee, Lyu, Yao ICML '24)**

LPGD = Low-rank PGD [Lee, Lyu, Yao NeurIPS '23]

BCD = Adaptive BCD [Lee, Lyu, Yao ICML '24]

Neural = 2-year coupled NN implementation of BCD [Lee, Lyu, Yao ICML '24]





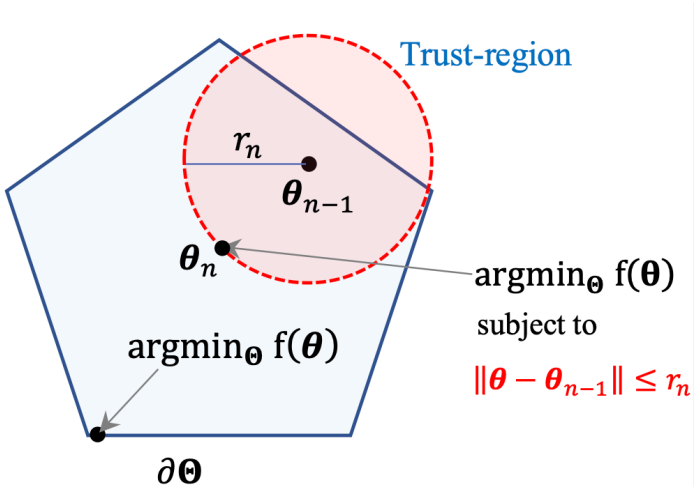
a Job postings

b Semi-synthetic MNIST data

A safeguarding approach:

Non-adaptive Trust-Region

Disciplined Agressiveness



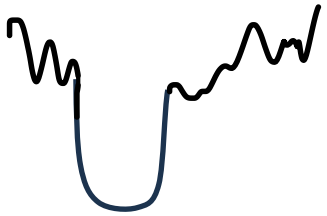
(BCD)

$$
\begin{aligned}
A_{n+1} &\leftarrow \Pi_{\Theta_A}\left(A_n - \alpha_n \nabla_A f(A_n, B_n, C_n)\right) \\
B_{n+1} &\leftarrow \Pi_{\Theta_B}\left(B_n - \beta_n \nabla_B f(A_{n+1}, B_n, C_n)\right) \\
C_{n+1} &\leftarrow \Pi_{\Theta_C}\left(C_n - \gamma_n \nabla_C f(A_{n+1}, B_{n+1}, C_n)\right)
\end{aligned}
$$

- Pros: **Robust against stepsize choices**

(BCD)

$$
\begin{aligned}
A_{n+1} &\leftarrow \Pi_{\Theta_A}\left(A_n - \alpha_n \nabla_A f(A_n, B_n, C_n)\right)\\
B_{n+1} &\leftarrow \Pi_{\Theta_B}\left(B_n - \beta_n \nabla_B f(A_{n+1}, B_n, C_n)\right)\\
C_{n+1} &\leftarrow \Pi_{\Theta_C}\left(C_n - \gamma_n \nabla_C f(A_{n+1}, B_{n+1}, C_n)\right)
\end{aligned}
$$

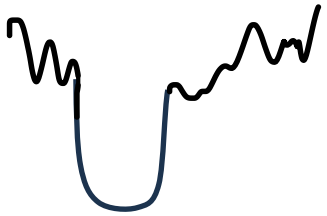


- Pros: **Robust against stepsize choices**
- Cons: **Convergence rate could be slow for FLAT landscape**
  - (# of iterations to reach an ϵ-stationary points for PGD) $= O(L\epsilon^{-2})$

$$
\text{“ } \|\nabla f(\theta_n)\| \le \epsilon \text{ ”}
$$

(BCD)

$$
\begin{aligned}
A_{n+1} &\leftarrow \Pi_{\Theta_A}\left(A_n - \alpha_n \nabla_A f(A_n, B_n, C_n)\right)\\
B_{n+1} &\leftarrow \Pi_{\Theta_B}\left(B_n - \beta_n \nabla_B f(A_{n+1}, B_n, C_n)\right)\\
C_{n+1} &\leftarrow \Pi_{\Theta_C}\left(C_n - \gamma_n \nabla_C f(A_{n+1}, B_{n+1}, C_n)\right)
\end{aligned}
$$



- Pros: **Robust against stepsize choices**
- Cons: **Convergence rate could be slow for FLAT landscape**
  - (# of iterations to reach an ϵ-stationary points for PGD) $= O(L\epsilon^{-2})$

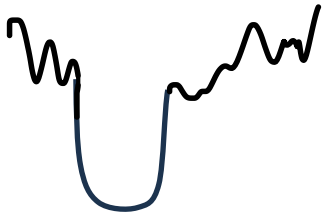
$$\text{“ } \|\nabla f(\theta_n)\| \le \epsilon \text{ ”}$$

[Razaviyayn et al. ’14, Lyu, Li ’24+]

  - (# of iterations to reach an ϵ-stationary points for **BCD**) $= O(L(1+\rho^{-1})\epsilon^{-2})$

(BCD)

$$A_{n+1} \leftarrow \Pi_{\Theta_A}\left(A_n - \alpha_n \nabla_A f(A_n, B_n, C_n)\right)$$
$$B_{n+1} \leftarrow \Pi_{\Theta_B}\left(B_n - \beta_n \nabla_B f(A_{n+1}, B_n, C_n)\right)$$
$$C_{n+1} \leftarrow \Pi_{\Theta_C}\left(C_n - \gamma_n \nabla_C f(A_{n+1}, B_{n+1}, C_n)\right)$$



- Pros: **Robust against stepsize choices**
- Cons: **Convergence rate could be slow for FLAT landscape**
  - (# of iterations to reach an ϵ-stationary points for PGD) $= O(L\epsilon^{-2})$

" $\|\nabla f(\theta_n)\| \leq \epsilon$ "

[Razaviyayn et al. '14, Lyu, Li '24+]

  - (# of iterations to reach an ϵ-stationary points for **BCD**) $= O(L(1+\rho^{-1})\epsilon^{-2})$

Largest allowed stepsize

(ρ=Local smoothness parameter)

(Small ρ ⇔ Flat local landscape ⇔ Large allowed stepsize)

BCD with Diminishing Radius

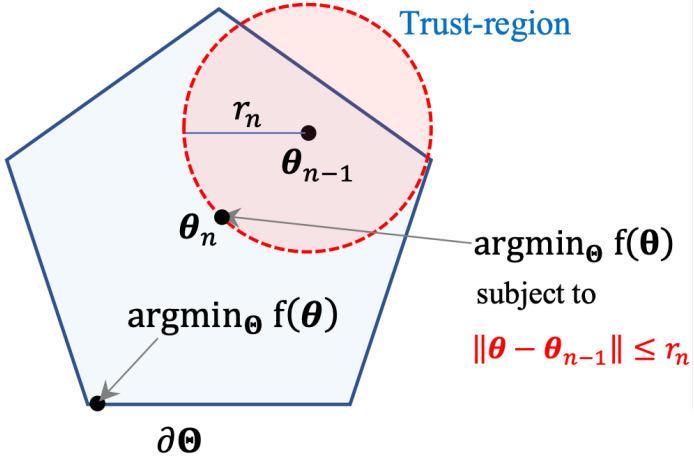
(BCD-DR)

$$
\begin{aligned}
A_{n+1} &\leftarrow \Pi_{\Theta_A \cap \{A : \|A - A_n\| \le r_n\}} \left(A_n - \alpha_n \nabla_A f(A_n, B_n, C_n)\right) \\
B_{n+1} &\leftarrow \Pi_{\Theta_B \cap \{B : \|B - B_n\| \le r_n\}} \left(B_n - \beta_n \nabla_B f(A_{n+1}, B_n, C_n)\right) \\
C_{n+1} &\leftarrow \Pi_{\Theta_C \cap \{C : \|C - C_n\| \le r_n\}} \left(C_n - \gamma_n \nabla_C f(A_{n+1}, B_{n+1}, C_n)\right)
\end{aligned}
$$

BCD with Diminishing Radius

(BCD-DR)

$$A_{n+1} \leftarrow \Pi_{\Theta_A \cap \{A : \|A - A_n\| \le r_n\}} \left(A_n - \alpha_n \nabla_A f(A_n, B_n, C_n)\right)$$
$$B_{n+1} \leftarrow \Pi_{\Theta_B \cap \{B : \|B - B_n\| \le r_n\}} \left(B_n - \beta_n \nabla_B f(A_{n+1}, B_n, C_n)\right)$$
$$C_{n+1} \leftarrow \Pi_{\Theta_C \cap \{C : \|C - C_n\| \le r_n\}} \left(C_n - \gamma_n \nabla_C f(A_{n+1}, B_{n+1}, C_n)\right)$$



DR = Diminishing Radius: $r_n \sim c/\sqrt{n}$

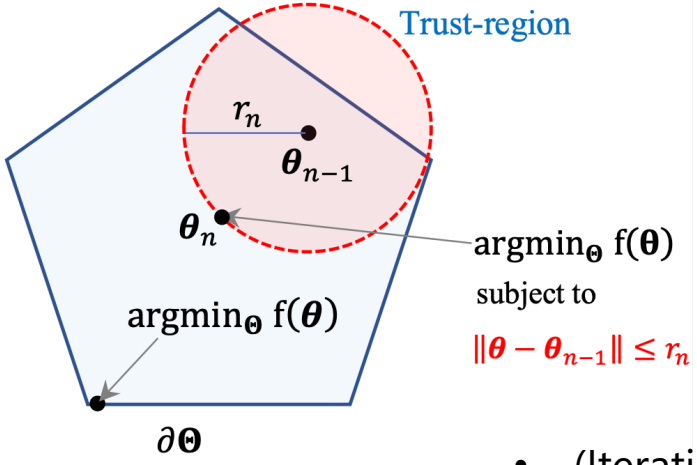
(In classic Trust-Region literature, r_n is chosen adaptively)

[Lyu, Strohmeier, Needell JMLR '22]

[Lyu, JMLR '24] [Powell, Lyu ICML'24]

(BCD-DR)

$$A_{n+1} \leftarrow \Pi_{\Theta_A \cap \{A : \|A - A_n\| \le r_n\}} \left(A_n - \alpha_n \nabla_A f(A_n, B_n, C_n)\right)$$
$$B_{n+1} \leftarrow \Pi_{\Theta_B \cap \{B : \|B - B_n\| \le r_n\}} \left(B_n - \beta_n \nabla_B f(A_{n+1}, B_n, C_n)\right)$$
$$C_{n+1} \leftarrow \Pi_{\Theta_C \cap \{C : \|C - C_n\| \le r_n\}} \left(C_n - \gamma_n \nabla_C f(A_{n+1}, B_{n+1}, C_n)\right)$$



DR = Diminishing Radius: $r_n \sim c/\sqrt{n}$

(In classic Trust-Region literature, r_n is chosen adaptively)

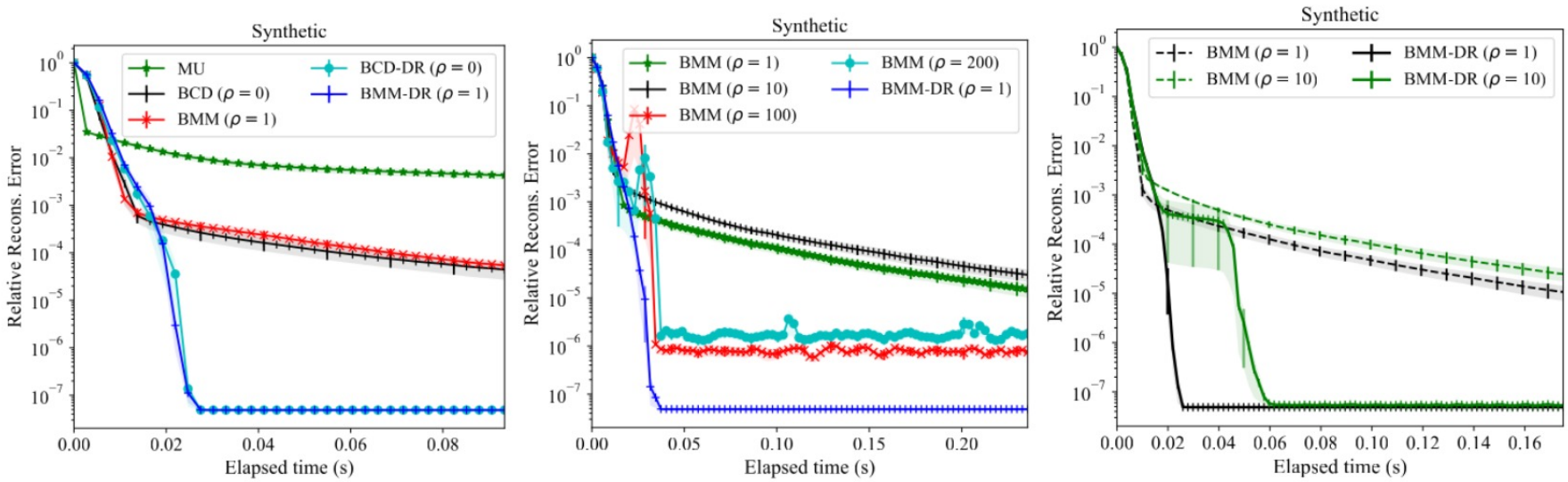
[Lyu, Strohmeier, Needell JMLR '22]

[Lyu, JMLR '24] [Powell, Lyu ICML'24]

- (Iteration complexity for PGD) $= O(L\epsilon^{-2})$
- (iteration complexity for BCD) $= O(L(1+\rho^{-1})\epsilon^{-2})$
- (Iteration complexity for **BCD-DR**) $= O(L\epsilon^{-2})$ [Li, Lyu '24+]

No dependence on ρ

A very flat nonconvex problem



MU=Multiplicative Update

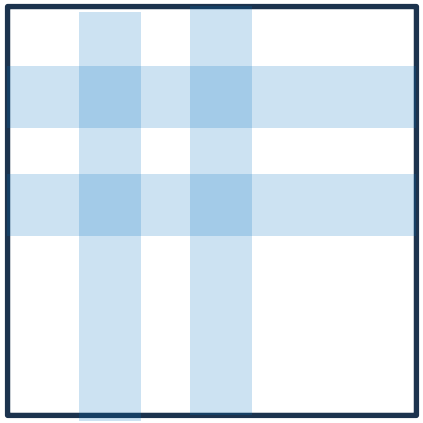
BCD = Block Coordianate Descent

BMM= Block Majorization-Minimization (in this case BCD+prox. Reg.)

A randomization approach:

Randomized Hessian Approximation

Flip the right coins!



What's known for the simplest algorithm?

(QN) $$\boldsymbol{\theta}_{n+1} \leftarrow \boldsymbol{\theta}_n - \overbrace{\left(\nabla^2 f(\boldsymbol{\theta}_n) + \gamma_n \mathbf{I}_p\right)^{-1}}^{\mathbf{H}_n} \nabla f(\boldsymbol{\theta}_n)$$

What's known for the simplest algorithm?

$$\textbf{(QN)} \qquad \boldsymbol{\theta}_{n+1} \leftarrow \boldsymbol{\theta}_n - \overbrace{\left(\nabla^2 f(\boldsymbol{\theta}_n) + \gamma_n \mathbf{I}_p\right)^{-1}}^{\mathbf{H}_n} \nabla f(\boldsymbol{\theta}_n)$$

- $\boldsymbol{\theta}_{n+1}$ can be found by solving the following linear eq.

$$\left(\nabla^2 f(\boldsymbol{\theta}_n) + \gamma_n \mathbf{I}_p\right)(\boldsymbol{\theta}_{n+1} - \boldsymbol{\theta}_n) = \nabla f(\boldsymbol{\theta}_n)$$

What's known for the simplest algorithm?

(QN) $$\boldsymbol{\theta}_{n+1} \leftarrow \boldsymbol{\theta}_n - \overbrace{\left(\nabla^2 f(\boldsymbol{\theta}_n) + \gamma_n \mathbf{I}_p\right)^{-1}}^{\mathbf{H}_n} \nabla f(\boldsymbol{\theta}_n)$$

- $\boldsymbol{\theta}_{n+1}$ can be found by solving the following linear eq.

$$\left(\nabla^2 f(\boldsymbol{\theta}_n) + \gamma_n \mathbf{I}_p\right)(\boldsymbol{\theta}_{n+1} - \boldsymbol{\theta}_n) = \nabla f(\boldsymbol{\theta}_n)$$

- If we had a Cholesky Factorization $\mathbf{F}_n \mathbf{F}_n^T = \nabla^2 f(\boldsymbol{\theta}_n) + \gamma_n \mathbf{I}_p$, then backward-forward substitution solves the above

What's known for the simplest algorithm?

(QN) $$\boldsymbol{\theta}_{n+1} \leftarrow \boldsymbol{\theta}_n - \overbrace{\left(\nabla^2 f(\boldsymbol{\theta}_n) + \gamma_n \mathbf{I}_p\right)^{-1}}^{\mathbf{H}_n} \nabla f(\boldsymbol{\theta}_n)$$

- $\boldsymbol{\theta}_{n+1}$ can be found by solving the following linear eq.

$$\left(\nabla^2 f(\boldsymbol{\theta}_n) + \gamma_n \mathbf{I}_p\right)(\boldsymbol{\theta}_{n+1} - \boldsymbol{\theta}_n) = \nabla f(\boldsymbol{\theta}_n)$$

- If we had a Cholesky Factorization $\mathbf{F}_n \mathbf{F}_n^T = \nabla^2 f(\boldsymbol{\theta}_n) + \gamma_n \mathbf{I}_p$, then backward-forward substitution solves the above

- Cholesky factorization is super expensive $O(p^3)$ (same as inversion)

What's known for the simplest algorithm?

(QN) $$\boldsymbol{\theta}_{n+1} \leftarrow \boldsymbol{\theta}_n - \overbrace{\left(\nabla^2 f(\boldsymbol{\theta}_n) + \gamma_n \mathbf{I}_p\right)^{-1}}^{\mathbf{H}_n} \nabla f(\boldsymbol{\theta}_n)$$

- $\boldsymbol{\theta}_{n+1}$ can be found by solving the following linear eq.

$$\left(\nabla^2 f(\boldsymbol{\theta}_n) + \gamma_n \mathbf{I}_p\right)(\boldsymbol{\theta}_{n+1} - \boldsymbol{\theta}_n) = \nabla f(\boldsymbol{\theta}_n)$$

- If we had a Cholesky Factorization $\mathbf{F}_n \mathbf{F}_n^T = \nabla^2 f(\boldsymbol{\theta}_n) + \gamma_n \mathbf{I}_p$, then backward-forward substitution solves the above
- Cholesky factorization is super expensive $O(p^3)$ (same as inversion)
- A recent, **randomized low-rank (k) Cholesky** is cheap $O(k^2 p)$

[Chen, Epperly, Tropp, and Webber '22+]

What's known for the simplest algorithm?

(QN) $$\boldsymbol{\theta}_{n+1} \leftarrow \boldsymbol{\theta}_n - \overbrace{\left(\nabla^2 f(\boldsymbol{\theta}_n) + \gamma_n \mathbf{I}_p\right)^{-1}}^{\mathbf{H}_n} \nabla f(\boldsymbol{\theta}_n)$$

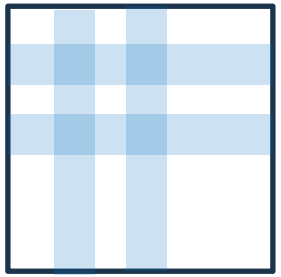
- $\boldsymbol{\theta}_{n+1}$ can be found by solving the following linear eq.

$$\left(\nabla^2 f(\boldsymbol{\theta}_n) + \gamma_n \mathbf{I}_p\right)(\boldsymbol{\theta}_{n+1} - \boldsymbol{\theta}_n) = \nabla f(\boldsymbol{\theta}_n)$$

- If we had a Cholesky Factorization $\mathbf{F}_n \mathbf{F}_n^T = \nabla^2 f(\boldsymbol{\theta}_n) + \gamma_n \mathbf{I}_p$, then backward-forward substitution solves the above
- Cholesky factorization is super expensive $O(p^3)$ (same as inversion)
- A recent, **randomized low-rank (k) Cholesky** is cheap $O(k^2 p)$

[Chen, Epperly, Tropp, and Webber '22+]

Idea: Sample rows and columns of the Hessian with prob. ∝ diag(Hessian)



Randomized Low-rank Quasi-Newton

(RLQN)
$$\begin{cases} \mathbf{F}_n\mathbf{F}_n^T \approx \nabla^2 f(\boldsymbol{\theta}_n) + \tau_n \mathbf{I}_N & (\triangleright \textit{Randomized low-rank Hessian approx.}) \\ \mathbf{H}_n \leftarrow \alpha_n \left(\mathbf{F}_n\mathbf{F}_n^T + \delta_n \mathbf{I}_N\right)^{-1} & (\triangleright \textit{Preconditioning matrix}) \\ \boldsymbol{\theta}_{n+1} \leftarrow \boldsymbol{\theta}_n - \mathbf{H}_n \nabla f(\boldsymbol{\theta}_n) & (\triangleright \textit{Parameter update}). \end{cases}$$

[Duan, Lyu '24+]

Randomized Low-rank Quasi-Newton

$$\textbf{(RLQN)} \quad \begin{cases} \mathbf{F}_n\mathbf{F}_n^T \approx \nabla^2 f(\boldsymbol{\theta}_n) + \tau_n \mathbf{I}_N & (\triangleright \textit{Randomized low-rank Hessian approx.}) \\ \mathbf{H}_n \leftarrow \alpha_n \left(\mathbf{F}_n\mathbf{F}_n^T + \delta_n \mathbf{I}_N\right)^{-1} & (\triangleright \textit{Preconditioning matrix}) \\ \boldsymbol{\theta}_{n+1} \leftarrow \boldsymbol{\theta}_n - \mathbf{H}_n \nabla f(\boldsymbol{\theta}_n) & (\triangleright \textit{Parameter update}). \end{cases}$$

[Duan, Lyu '24+]

- Per-iteration computational cost = $O(k^2 p)$ --- Nearly 1st-order for $k \ll p$

Randomized Low-rank Quasi-Newton

$$\textbf{(RLQN)}\quad \begin{cases} \mathbf{F}_n\mathbf{F}_n^T \approx \nabla^2 f(\boldsymbol{\theta}_n) + \tau_n \mathbf{I}_N & (\triangleright \textit{Randomized low-rank Hessian approx.}) \\ \mathbf{H}_n \leftarrow \alpha_n \left(\mathbf{F}_n\mathbf{F}_n^T + \delta_n \mathbf{I}_N\right)^{-1} & (\triangleright \textit{Preconditioning matrix}) \\ \boldsymbol{\theta}_{n+1} \leftarrow \boldsymbol{\theta}_n - \mathbf{H}_n \nabla f(\boldsymbol{\theta}_n) & (\triangleright \textit{Parameter update}). \end{cases}$$

[Duan, Lyu '24+]

- Per-iteration computational cost = $O(k^2 p)$ --- Nearly 1st-order for $k \ll p$
- "Fast" local convergence rate (depending on "effective condition number")

Randomized Low-rank Quasi-Newton

$$\textbf{(RLQN)} \quad \begin{cases} \mathbf{F}_n\mathbf{F}_n^T \approx \nabla^2 f(\boldsymbol{\theta}_n) + \tau_n \mathbf{I}_N & (\triangleright \textit{Randomized low-rank Hessian approx.}) \\ \mathbf{H}_n \leftarrow \alpha_n \left(\mathbf{F}_n\mathbf{F}_n^T + \delta_n \mathbf{I}_N\right)^{-1} & (\triangleright \textit{Preconditioning matrix}) \\ \boldsymbol{\theta}_{n+1} \leftarrow \boldsymbol{\theta}_n - \mathbf{H}_n \nabla f(\boldsymbol{\theta}_n) & (\triangleright \textit{Parameter update}). \end{cases}$$

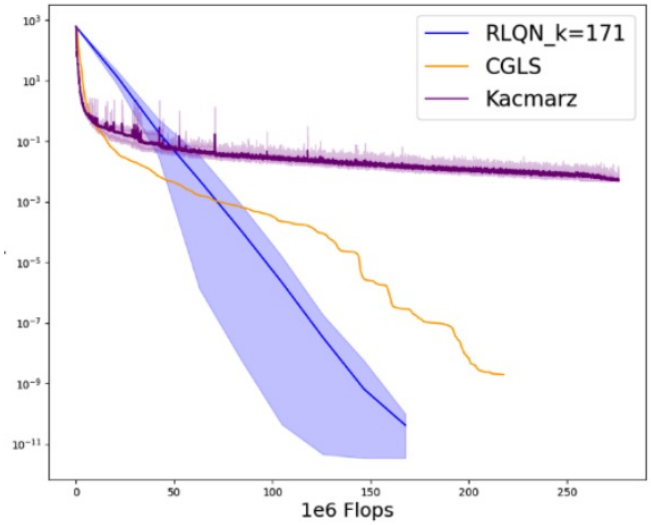
[Duan, Lyu '24+]

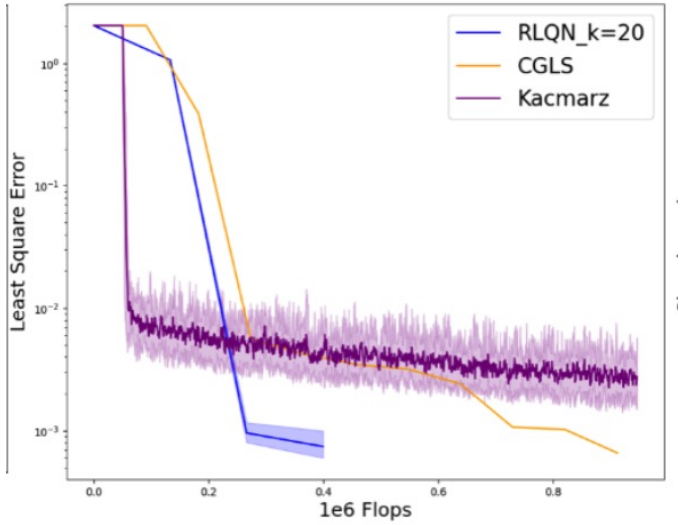
- Per-iteration computational cost = $O(k^2 p)$ --- Nearly 1st-order for $k \ll p$
- "Fast" local convergence rate (depending on "effective condition number")
- Matching global convergence rate

$$\textbf{(RLQN)}\quad \begin{cases} \mathbf{F}_n\mathbf{F}_n^T \approx \nabla^2 f(\boldsymbol{\theta}_n) + \tau_n \mathbf{I}_N & (\triangleright \textit{Randomized low-rank Hessian approx.}) \\ \mathbf{H}_n \leftarrow \alpha_n \left(\mathbf{F}_n\mathbf{F}_n^T + \delta_n \mathbf{I}_N\right)^{-1} & (\triangleright \textit{Preconditioning matrix}) \\ \boldsymbol{\theta}_{n+1} \leftarrow \boldsymbol{\theta}_n - \mathbf{H}_n \nabla f(\boldsymbol{\theta}_n) & (\triangleright \textit{Parameter update}). \end{cases}$$

[Duan, Lyu '24+]

- Per-iteration computational cost = $O(k^2 p)$ --- Nearly 1st-order for $k \ll p$
- "Fast" local convergence rate (depending on "effective condition number")
- Matching global convergence rate

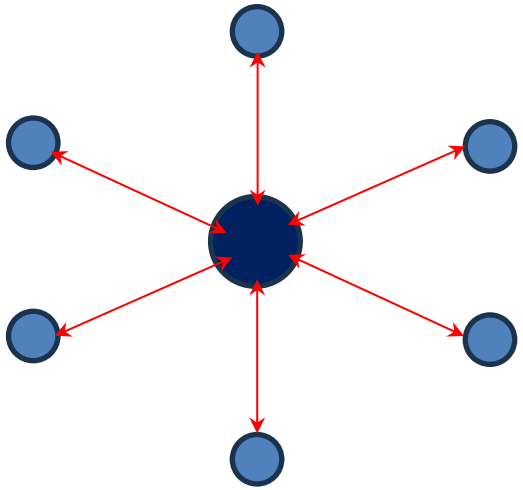




A sampling approach:

Stochastic Aggregated Double-Averaging

Consensus, Consensus!



A finite-sum problem

$$\boldsymbol{\theta}^* \in \arg\min_{\boldsymbol{\theta} \in \boldsymbol{\Theta}} \left\{ f(\boldsymbol{\theta}) := \sum_{v \in \mathcal{V}} f^v(\boldsymbol{\theta}) \right\}$$

A finite-sum problem

$$\boldsymbol{\theta}^* \in \arg\min_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \left\{ f(\boldsymbol{\theta}) := \sum_{v\in\mathcal{V}} f^v(\boldsymbol{\theta}) \right\}$$

(PSGD)

$$v_n \sim \text{Uniform}(\mathcal{V}), \text{ independent}$$
$$\boldsymbol{\theta}_{n+1} \leftarrow \Pi_{\boldsymbol{\Theta}} \left(\boldsymbol{\theta}_n - \alpha_n \nabla f^{v_n}(\boldsymbol{\theta}_n) \right)$$

Memory	Iteration Complexity
$O(p)$	$O(\epsilon^{-4})$

A finite-sum problem

$$\boldsymbol{\theta}^* \in \arg\min_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \left\{ f(\boldsymbol{\theta}) := \sum_{v\in\mathcal{V}} f^v(\boldsymbol{\theta}) \right\}$$

		Memory	Iteration Complexity
(PSGD)	$v_n \sim \text{Uniform}(\mathcal{V})$, independent $\boldsymbol{\theta}_{n+1} \leftarrow \Pi_{\boldsymbol{\Theta}} \left(\boldsymbol{\theta}_n - \alpha_n \nabla f^{v_n}(\boldsymbol{\theta}_n)\right)$	$O(p)$	$O(\epsilon^{-4})$
(SAG) [Bottou '98]	$v_n \sim \text{Uniform}(\mathcal{V})$, independent $\nabla^v \leftarrow \nabla f^v(\boldsymbol{\theta}_n)$, Keep other ∇^us $\boldsymbol{\theta}_{n+1} \leftarrow \Pi_{\boldsymbol{\Theta}} \left(\boldsymbol{\theta}_n - \alpha_n \frac{1}{\lvert\mathcal{V}\rvert} \sum_{u\in\mathcal{V}} \nabla^u\right)$	$O(pn)$	$O(\epsilon^{-2})$

A finite-sum problem

$$\boldsymbol{\theta}^* \in \arg\min_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \left\{ f(\boldsymbol{\theta}) := \sum_{v\in\mathcal{V}} f^v(\boldsymbol{\theta}) \right\}$$

		Memory	Iteration Complexity
(PSGD)	$v_n \sim \text{Uniform}(\mathcal{V})$, independent $\boldsymbol{\theta}_{n+1} \leftarrow \Pi_{\boldsymbol{\Theta}} \left(\boldsymbol{\theta}_n - \alpha_n \nabla f^{v_n}(\boldsymbol{\theta}_n)\right)$	$O(p)$	$O(\epsilon^{-4})$
(SAG) [Bottou '98]	$v_n \sim \text{Uniform}(\mathcal{V})$, independent $\nabla^v \leftarrow \nabla f^v(\boldsymbol{\theta}_n)$, Keep other ∇^us $\boldsymbol{\theta}_{n+1} \leftarrow \Pi_{\boldsymbol{\Theta}} \left(\boldsymbol{\theta}_n - \alpha_n \frac{1}{\lvert\mathcal{V}\rvert} \sum_{u\in\mathcal{V}} \nabla^u\right)$	$O(pn)$	$O(\epsilon^{-2})$
(MISO) [Mairal '13]	$v_n \sim \text{Uniform}(\mathcal{V})$, independent $\boldsymbol{\theta}^{v_n} \leftarrow \boldsymbol{\theta}_n$, Keep other $\boldsymbol{\theta}^u$s $\nabla^v \leftarrow \nabla f^v(\boldsymbol{\theta}_n)$, Keep other ∇^us $\boldsymbol{\theta}_{n+1} \leftarrow \Pi_{\boldsymbol{\Theta}} \left(\frac{1}{\lvert\mathcal{V}\rvert} \sum_{u\in\mathcal{V}} \boldsymbol{\theta}^u - \alpha_n \frac{1}{\lvert\mathcal{V}\rvert} \sum_{u\in\mathcal{V}} \nabla^u\right)$	$O(pn)$	$O(\epsilon^{-2})$ [Karimi et al. ICML'22]

A finite-sum problem

$$\boldsymbol{\theta}^* \in \arg\min_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \left\{ f(\boldsymbol{\theta}) := \sum_{v\in\mathcal{V}} f^v(\boldsymbol{\theta}) \right\}$$

		Memory	Iteration Complexity
(PSGD)	$v_n \sim P(v_{n-1}, \cdot)$, Markovian $\boldsymbol{\theta}_{n+1} \leftarrow \Pi_{\boldsymbol{\Theta}} \left(\boldsymbol{\theta}_n - \alpha_n \nabla f^{v_n}(\boldsymbol{\theta}_n)\right)$	$O(p)$	$O(\epsilon^{-4})$
(SAG)	$v_n \sim P(v_{n-1}, \cdot)$, Markovian $\nabla^v \leftarrow \nabla f^v(\boldsymbol{\theta}_n)$, Keep other ∇^us $\boldsymbol{\theta}_{n+1} \leftarrow \Pi_{\boldsymbol{\Theta}} \left(\boldsymbol{\theta}_n - \alpha_n \frac{1}{\lvert\mathcal{V}\rvert} \sum_{u\in\mathcal{V}} \nabla^u\right)$	$O(pn)$	$O(\epsilon^{-2})$ [Even ICML '23]
(MISO)	$v_n \sim P(v_{n-1}, \cdot)$, Markovian $\boldsymbol{\theta}^{v_n} \leftarrow \boldsymbol{\theta}_n$, Keep other $\boldsymbol{\theta}^u$s $\nabla^v \leftarrow \nabla f^v(\boldsymbol{\theta}_n)$, Keep other ∇^us $\boldsymbol{\theta}_{n+1} \leftarrow \Pi_{\boldsymbol{\Theta}} \left(\frac{1}{\lvert\mathcal{V}\rvert} \sum_{u\in\mathcal{V}} \boldsymbol{\theta}^u - \alpha_n \frac{1}{\lvert\mathcal{V}\rvert} \sum_{u\in\mathcal{V}} \nabla^u\right)$	$O(pn)$	??

A finite-sum problem

$$\boldsymbol{\theta}^* \in \arg\min_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \left\{ f(\boldsymbol{\theta}) := \sum_{v\in\mathcal{V}} f^v(\boldsymbol{\theta}) \right\}$$

		Memory	Iteration Complexity
(PSGD)	$v_n \sim \text{Reshuffling}(\mathcal{V})$ $\boldsymbol{\theta}_{n+1} \leftarrow \Pi_{\boldsymbol{\Theta}} \left(\boldsymbol{\theta}_n - \alpha_n \nabla f^{v_n}(\boldsymbol{\theta}_n)\right)$	$O(p)$	$O(\epsilon^{-3})$ [Mishchenko et al. '20] [Lu et al. '22]
(SAG)	$v_n \sim \text{Reshuffling}(\mathcal{V})$ $\nabla^v \leftarrow \nabla f^v(\boldsymbol{\theta}_n)$, Keep other ∇^us $\boldsymbol{\theta}_{n+1} \leftarrow \Pi_{\boldsymbol{\Theta}} \left(\boldsymbol{\theta}_n - \alpha_n \frac{1}{\lvert\mathcal{V}\rvert} \sum_{u\in\mathcal{V}} \nabla^u\right)$	$O(pn)$	??
(MISO)	$v_n \sim \text{Reshuffling}(\mathcal{V})$ $\boldsymbol{\theta}^{v_n} \leftarrow \boldsymbol{\theta}_n$, Keep other $\boldsymbol{\theta}^u$s $\nabla^v \leftarrow \nabla f^v(\boldsymbol{\theta}_n)$, Keep other ∇^us $\boldsymbol{\theta}_{n+1} \leftarrow \Pi_{\boldsymbol{\Theta}} \left(\frac{1}{\lvert\mathcal{V}\rvert} \sum_{u\in\mathcal{V}} \boldsymbol{\theta}^u - \alpha_n \frac{1}{\lvert\mathcal{V}\rvert} \sum_{u\in\mathcal{V}} \nabla^u\right)$	$O(pn)$	??

Minimization by Incremental Surrogate Optimization

$$\boldsymbol{\theta}^* \in \arg\min_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \left\{ f(\boldsymbol{\theta}) := \sum_{v\in\mathcal{V}} f^v(\boldsymbol{\theta}) \right\}$$

(MISO)

$$v_n \sim \text{Recurrent}(\mathcal{V})$$
$$\boldsymbol{\theta}^{v_n} \leftarrow \boldsymbol{\theta}_n, \text{ Keep other } \boldsymbol{\theta}^u\text{s}$$
$$\nabla^v \leftarrow \nabla f^v(\boldsymbol{\theta}_n), \text{ Keep other } \nabla^u\text{s}$$
$$\boldsymbol{\theta}_{n+1} \leftarrow \Pi_{\boldsymbol{\Theta}} \left(\frac{1}{|\mathcal{V}|} \sum_{u\in\mathcal{V}} \boldsymbol{\theta}^u - \alpha_n \frac{1}{|\mathcal{V}|} \sum_{u\in\mathcal{V}} \nabla^u \right)$$

Iteration Complexity

$O(\epsilon^{-2})$

[Powell, Lyu ICML '24]

Minimization by Incremental Surrogate Optimization

$$\boldsymbol{\theta}^* \in \arg\min_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \left\{ f(\boldsymbol{\theta}) := \sum_{v\in\mathcal{V}} f^v(\boldsymbol{\theta}) \right\}$$

(MISO)

$v_n \sim \text{Recurrent}(\mathcal{V})$

$\boldsymbol{\theta}^{v_n} \leftarrow \boldsymbol{\theta}_n$, Keep other $\boldsymbol{\theta}^u$s

$\nabla^v \leftarrow \nabla f^v(\boldsymbol{\theta}_n)$, Keep other ∇^us

$$\boldsymbol{\theta}_{n+1} \leftarrow \Pi_{\boldsymbol{\Theta}} \left(\frac{1}{|\mathcal{V}|} \sum_{u\in\mathcal{V}} \boldsymbol{\theta}^u - \alpha_n \frac{1}{|\mathcal{V}|} \sum_{u\in\mathcal{V}} \nabla^u \right)$$

Iteration Complexity

$O(\epsilon^{-2})$

[Powell, Lyu ICML '24]

- Recurrent sampling = Very general sampling scheme

  e.g., i.i.d., Markovian, Recurrent, Cyclic, etc.

Minimization by Incremental Surrogate Optimization

$$\boldsymbol{\theta}^* \in \arg\min_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \left\{ f(\boldsymbol{\theta}) := \sum_{v\in\mathcal{V}} f^v(\boldsymbol{\theta}) \right\}$$

(MISO)

$$\begin{aligned} &v_n \sim \text{Recurrent}(\mathcal{V}) \\ &\boldsymbol{\theta}^{v_n} \leftarrow \boldsymbol{\theta}_n, \text{ Keep other } \boldsymbol{\theta}^u\text{s} \\ &\nabla^v \leftarrow \nabla f^v(\boldsymbol{\theta}_n), \text{ Keep other } \nabla^u\text{s} \\ &\boldsymbol{\theta}_{n+1} \leftarrow \Pi_{\boldsymbol{\Theta}} \left(\frac{1}{|\mathcal{V}|} \sum_{u\in\mathcal{V}} \boldsymbol{\theta}^u - \alpha_n \frac{1}{|\mathcal{V}|} \sum_{u\in\mathcal{V}} \nabla^u \right) \end{aligned}$$

Iteration Complexity

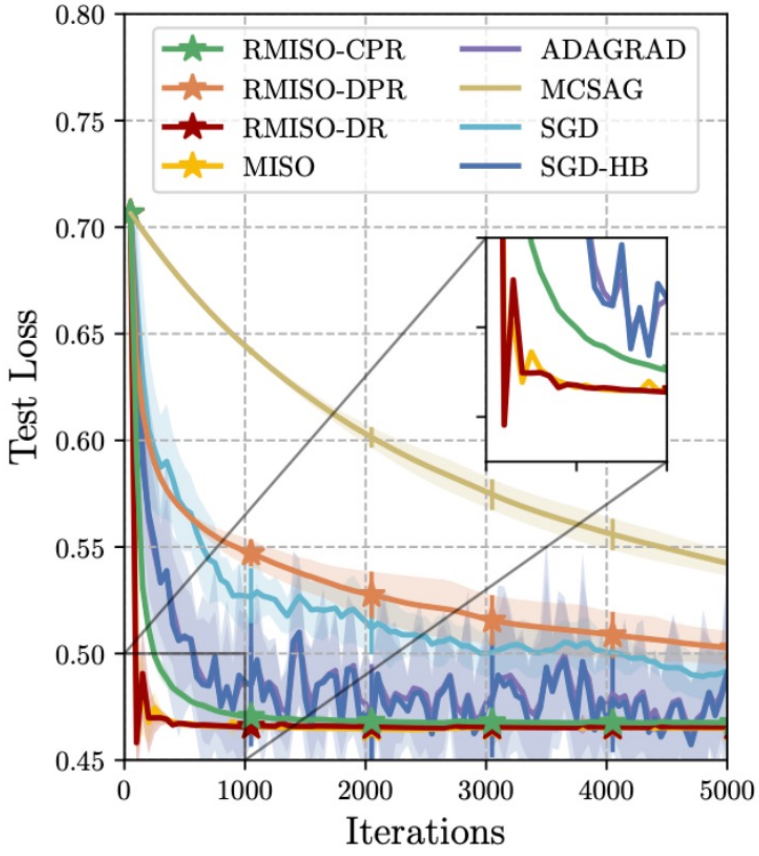
$O(\epsilon^{-2})$

[Powell, Lyu ICML '24]

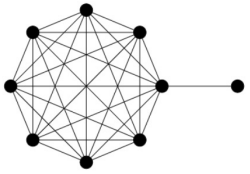
- Recurrent sampling = Very general sampling scheme

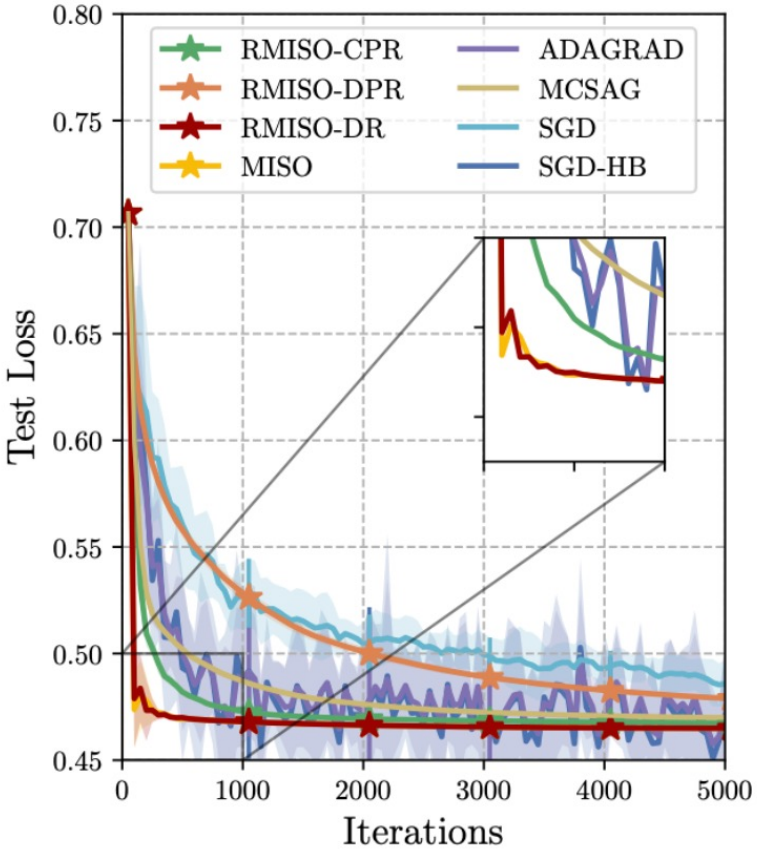
  e.g., i.i.d., Markovian, Recurrent, Cyclic, etc.

- The implied constant in $O(\epsilon^{-2})$ is significantly improved

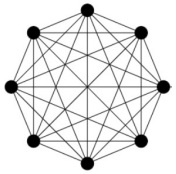


(a) Lonely graph





(b) Complete graph



Takeaways

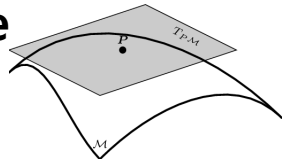
- Various optimization algorithms are based on simple mathematical principles

Takeaways

- Various optimization algorithms are based on simple mathematical principles
- Despite the extensive literature, there are many interesting challenges for improving these methods

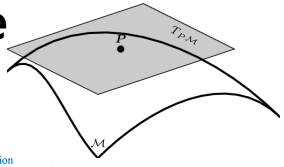
Takeaways

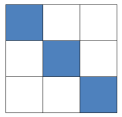
- Various optimization algorithms are based on simple mathematical principles
- Despite the extensive literature, there are many interesting challenges for improving these methods
  - **Leverage low-dim geometric structures in parameter space**
    - Riemannian Optimization

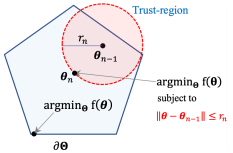


Takeaways

- Various optimization algorithms are based on simple mathematical principles
- Despite the extensive literature, there are many interesting challenges for improving these methods
  - **Leverage low-dim geometric structures in parameter space**
    - Riemannian Optimization
  - **Robustifying against stepsizes**
    - Block optimization and Trust-Region

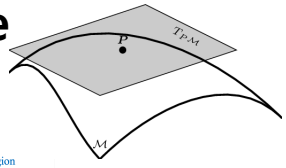




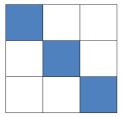


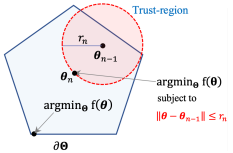
Takeaways

- Various optimization algorithms are based on simple mathematical principles
- Despite the extensive literature, there are many interesting challenges for improving these methods
  - **Leverage low-dim geometric structures in parameter space**
    - Riemannian Optimization



  - **Robustifying against stepsizes**
    - Block optimization and Trust-Region

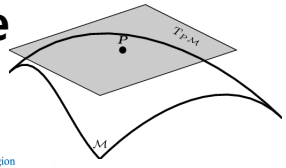




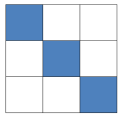
  - **Quasi-Newton "Lite"**
    - Randomized low-rank Hessian approximation

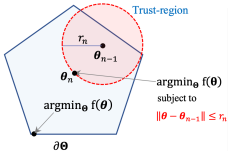
Takeaways

- Various optimization algorithms are based on simple mathematical principles
- Despite the extensive literature, there are many interesting challenges for improving these methods
  - **Leverage low-dim geometric structures in parameter space**
    - Riemannian Optimization



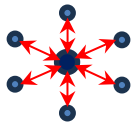
  - **Robustifying against stepsizes**
    - Block optimization and Trust-Region





  - **Quasi-Newton "Lite"**
    - Randomized low-rank Hessian approximation

  - **Finite-sum problem with arbitrary sampling**
    - Aggregation with parameter and gradient averaging



Thank you very much!

References

1. Yifan Chen, Ethan N Epperly, Joel A Tropp, and Robert J Webber. "*Randomly pivoted cholesky: Practical approximation of a kernel matrix with few entry evaluations*." ArXiv:2207.06503 (2022)
2. Joowon Lee, Hanbaek Lyu, and Weixin Yao, "*Exponentially Convergent Algorithms for Supervised Matrix Factorization*", NeurIPS 2023
3. Joowon Lee, Hanbaek Lyu, and Weixin Yao, "*Constrained Matrix Factorization: Local Landscape An alysis and Applications*" To appear in ICML 2024
4. Hanbaek Lyu, "*Stochastic regularized block majorization-minimization with weakly convex and mu lti-convex surrogates*" To appear in JMLR
5. Hanbaek Lyu, Christopher Strohmeier, and Deanna Needell, "*Online nonnegative tensor factorizati on and CP-Dictionary Learning for Markovian data*" JMLR 23(148):1–50, 2022
6. Hanbaek Lyu and Yuchen Li, "*Block majorization-minimization with diminishing radius for constrai ned nonconvex optimization*" Under Review in SIOPT
7. Yuchen Li, Laura Balzano, Deanna Needell, Hanbaek Lyu, "*Convergence and Complexity Guarantee for Inexact First-order Riemannian Optimization Algorithms.*" To appear in ICML 2024
8. William Powell and Hanbaek Lyu, "*Stochastic optimization with arbitrary recurrent data sampling.*" To appear in ICML 2024