Riemannian quasi-Newton methods, implementation techniques, and applications

Speaker: Wen Huang

Xiamen University

December 11, 2020

Guangxi University

Outline:

- Introduction
- Riemannian Quasi-Newton Methods
- Implementation Techniques
- Limited-memory Versions
- Applications

Outline:

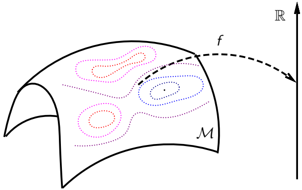
- Introduction
- Riemannian Quasi-Newton Methods
- Implementation Techniques
- Limited-memory Versions
- Applications

Riemannian Optimization

Problem: Given $f(x) : \mathcal{M} \to \mathbb{R}$, solve

$$\min_{x \in \mathcal{M}} f(x)$$

where $\mathcal{M}$ is a Riemannian manifold.

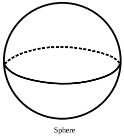


Unconstrained optimization problem on a constrained space.

Riemannian manifold = manifold + Riemannian metric

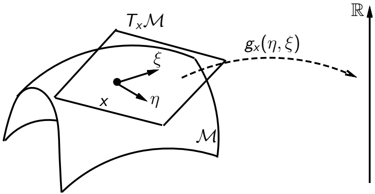
Riemannian Manifold

Manifolds:



- Stiefel manifold: $\mathrm{St}(p, n) = \{X \in \mathbb{R}^{n\times p} | X^T X = I_p\}$;
- Grassmann manifold $\mathrm{Gr}(p, n)$: all p-dimensional subspaces of $\mathbb{R}^n$;
- And many more.

Riemannian metric:



A Riemannian metric, denoted by g, is a smoothly-varying inner product on the tangent spaces;

Outline:

- Introduction
- Riemannian quasi-Newton methods
  - RBFGS
  - RTR-SR1
- Implementation techniques
- Limited-memory versions
- Applications

Euclidean Quasi-Newton Methods

A line search quasi-Newotn algorithm

Require: Initial iterate x_0;
1, Set $k \leftarrow 0$;
while not accurate enough **do**
2, Compute p_k from
$p_k = -B_k{}^{-1} \nabla f(x_k)$;
3, $x_{k+1} \leftarrow x_k + \alpha_k p_k$ with appropriate α_k;
4, Compute B_{k+1} by certain formula
5, $k \leftarrow k + 1$;
end while

A trust region quasi-Newotn algorithm

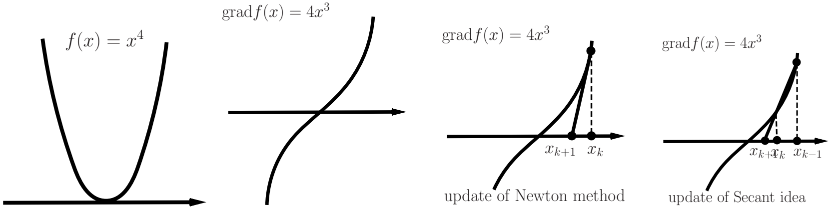
Require: Initial iterate x_0;
1, Set $k \leftarrow 0$;
while not accurate enough **do**
2, Compute
$p_k \approx \mathrm{argmin}_{\|p\| \leq \Delta_k} \nabla f(x_k)^T p + \frac{1}{2} p^T B_k p$;
3, $\rho_k \leftarrow \frac{f(x_k) - f(x_k + p_k)}{m_k(0) - m_k(p_k)}$;
4, $x_{k+1} \leftarrow \begin{cases} x_k + p_k & \text{if } \rho_k > \eta \\ x_k & \text{otherwise.} \end{cases}$
5, update radius to get Δ_{k+1}
6, Compute B_{k+1} by certain formula
7, $k \leftarrow k + 1$;
end while

Update formula: $B_{k+1} = \varphi(B_k, x_{k+1}, \ldots, x_0, \nabla f(x_{k+1}), \ldots, \nabla f(x_0))$

Euclidean Quasi-Newton Methods

Secant condition: 1-dimension example

An 1 dimension example to show the idea of the secant condition.



- Newton: $x_{k+1} = x_k - (\mathrm{Hess}\, f(x_k))^{-1} \,\mathrm{grad}\, f(x_k)$
- Secant: $x_{k+1} = x_k - B_k^{-1} \,\mathrm{grad}\, f(x_k)$,
  $B_k(x_k - x_{k-1}) = \mathrm{grad}\, f(x_k) - \mathrm{grad}\, f(x_{k-1})$

Euclidean Quasi-Newton Methods

Secant condition

Secant condition

$$B_{k+1} s_k = y_k,$$

where $s_k = x_{k+1} - x_k$ and $y_k = \mathrm{grad}\, f(x_{k+1}) - \mathrm{grad}\, f(x_k)$;

- B_k is not uniquely defined for $d > 1$;
- Extra conditions required
- Minimum change:

$$B_{k+1} = \arg \min_{B^T = B, B s_k = y_k} \|B - B_k\|$$

Euclidean Quasi-Newton Methods

BFGS and SR1

- Symmetric Rank–one (SR1) update
  - Minimum rank update:
  - Formula:

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k)(y_k - B_k s_k)^T}{(y_k - B_k s_k)^T s_k}$$

- Broyden, Fletcher, Goldfarb, Shanno (BFGS) update:
  - Minimum change:

$$B_{k+1} = \arg\min_B \|B^{-1} - B_k^{-1}\|_W, \text{ such that } Bs_k = y_k, B^T = B,$$

  where W is SPD satisfying $y_k = W s_k$ and $\|A\|_W = \|W^{1/2} A W^{1/2}\|_F$.
  - Formula:

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}$$

Riemannian BFGS Methods

BFGS quasi-Newton algorithm: from Euclidean to Riemannian

- Update formula:

$$x_{k+1} = x_k \underline{+ \alpha_k \eta_k}$$

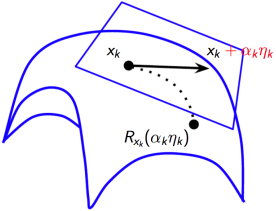
- Search direction:

$$\eta_k = -B_k^{-1} \operatorname{grad} f(x_k)$$

- B_k update:

$$B_{k+1} = \underline{B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}},$$

where $s_k = \underline{x_{k+1} - x_k}$, and $y_k = \underline{\operatorname{grad} f(x_{k+1}) - \operatorname{grad} f(x_k)}$



Optimization on a Manifold

Riemannian BFGS Methods

BFGS quasi-Newton algorithm: from Euclidean to Riemannian

replace by $R_{x_k}(\eta_k)$

- Update formula:

$$x_{k+1} = x_k + \alpha_k \eta_k$$

- Search direction:

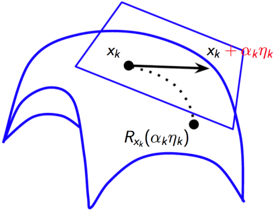
$$\eta_k = -B_k^{-1} \operatorname{grad} f(x_k)$$

- B_k update:

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k},$$

where $s_k = x_{k+1} - x_k$, and $y_k = \operatorname{grad} f(x_{k+1}) - \operatorname{grad} f(x_k)$

replaced by $R_{x_k}^{-1}(x_{k+1})$



Optimization on a Manifold

Riemannian BFGS Methods

BFGS quasi-Newton algorithm: from Euclidean to Riemannian

replace by $R_{x_k}(\eta_k)$

- Update formula:

$$x_{k+1} = x_k \underline{+ \alpha_k \eta_k}$$

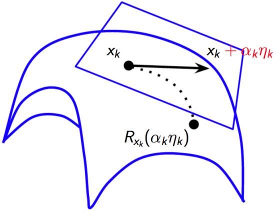
- Search direction:

$$\eta_k = -B_k^{-1} \operatorname{grad} f(x_k)$$

- B_k update:

$$B_{k+1} = \underline{B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}},$$

$\leftarrow$ use vector transport



Optimization on a Manifold

where $s_k = \underline{x_{k+1} - x_k}$, and $y_k = \underline{\operatorname{grad} f(x_{k+1}) - \operatorname{grad} f(x_k)}$

replaced by $R_{x_k}^{-1}(x_{k+1})$

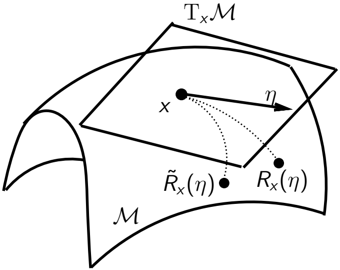
use vector transport

Riemannian BFGS Methods

Retraction and vector transport

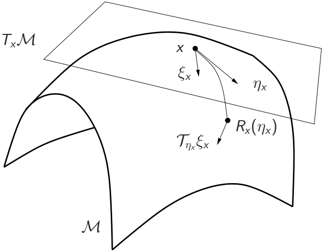
Retraction: $R : \mathrm{T}\mathcal{M} \to \mathcal{M}$

Euclidean	Riemannian
$x_{k+1} = x_k + \alpha_k d_k$	$x_{k+1} = R_{x_k}(\alpha_k \eta_k)$



Two retractions: R and $\tilde{R}$

A vector transport:
$\mathcal{T} : \mathrm{T}\mathcal{M} \times \mathrm{T}\mathcal{M} \to \mathrm{T}\mathcal{M}$:
$(\eta_x, \xi_x) \mapsto \mathcal{T}_{\eta_x}\xi_x$:



Riemannian BFGS Methods

BFGS quasi-Newton algorithm: from Euclidean to Riemannian

- Update formula:

$$x_{k+1} = \underline{R_{x_k}(\alpha_k \eta_k)}$$

- Search direction:

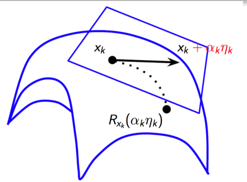
$$\eta_k = -B_k^{-1} \operatorname{grad} f(x_k)$$

- B_k update:

$$\tilde{B}_k = \underline{\mathcal{T}_{\alpha_k \eta_k} \circ B_k \circ \mathcal{T}_{\alpha_k \eta_k}^{-1}},$$

$$B_{k+1} = \underline{\tilde{B}_k - \frac{\tilde{B}_k s_k s_k^\flat \tilde{B}_k}{s_k^\flat \tilde{B}_k s_k} + \frac{y_k y_k^\flat}{y_k^\flat s_k}},$$

where $s_k = \underline{\mathcal{T}_{\alpha_k \eta_k}(\alpha_k \eta_k)}$, and $y_k = \underline{\operatorname{grad} f(x_{k+1}) - \mathcal{T}_{\alpha_k \eta_k} \operatorname{grad} f(x_k)}$;



Optimization on a Manifold

Riemannian BFGS Methods

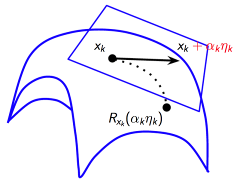
BFGS quasi-Newton algorithm: from Euclidean to Riemannian

- Update formula:

$$x_{k+1} = R_{x_k}(\alpha_k \eta_k)$$

- Search direction:

$$\eta_k = -B_k^{-1} \operatorname{grad} f(x_k)$$



- B_k update:

$$\tilde{B}_k = \mathcal{T}_{\alpha_k \eta_k} \circ B_k \circ \mathcal{T}_{\alpha_k \eta_k}^{-1}, \leftarrow$$

matrix matrix multiplication

$$B_{k+1} = \tilde{B}_k - \frac{\tilde{B}_k s_k s_k^\flat \tilde{B}_k}{s_k^\flat \tilde{B}_k s_k} + \frac{y_k y_k^\flat}{y_k^\flat s_k},$$

where $s_k = \mathcal{T}_{\alpha_k \eta_k}(\alpha_k \eta_k)$, and $y_k = \operatorname{grad} f(x_{k+1}) - \mathcal{T}_{\alpha_k \eta_k} \operatorname{grad} f(x_k)$;

$\uparrow$ matrix vector multiplication

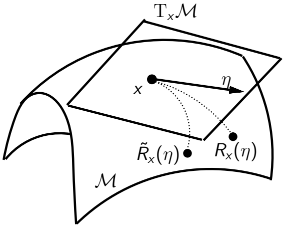
$\uparrow$ matrix vector multiplication

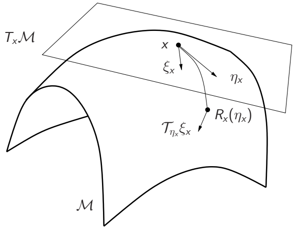
Extra cost on vector transports!

Riemannian BFGS Methods

Existing generic Riemannian BFGS methods

- Qi [Qi11]: exponential mapping, parallel translation;
  - Idea: imitate the Euclidean setting;
  - Exponential mapping: along the geodesic;
  - Parallel translation: move tangent vector parallelly;
  - Problem: maybe unknown to users, maybe expensive to compute;

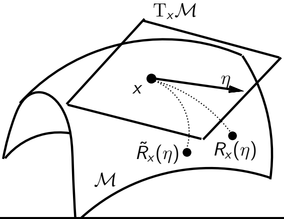


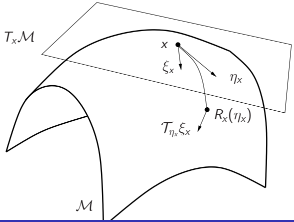


Riemannian BFGS Methods

Existing generic Riemannian BFGS methods

- Ring and Wirth [RW12]: retraction, vector transport by differentiated retraction, isometric vector transport;
  - Idea: quasi-Newton update in tangent space, then transport to new tangent space isometrically;
  - Retraction: no constraints;
  - Two vector transport: VT by differentiated retraction, and isometric VT;
  - Problem: vector transport by differentiated retraction maybe unknown to users, maybe expensive to compute;





Riemannian BFGS Methods

Existing generic Riemannian BFGS methods

- Ring and Wirth [RW12]: retraction, vector transport by differentiated retraction, isometric vector transport;
  - $f \circ R_{x_k}$ is defined on $\mathrm{T}_{x_k} \mathcal{M}$
  - Secant condition is defined as that in the Euclidean setting
  - Vector transport by differentiated retraction is needed

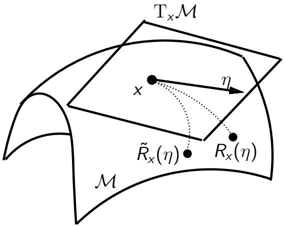
$$\begin{aligned} y_k = & \mathcal{T}_{\mathrm{S}_{\eta_{x_k}}} \left(\mathrm{grad}(f \circ R_{x_k})(\eta_{x_k}) - \mathrm{grad}(f \circ R_{x_k})(0_{x_k})\right) \\ = & \mathcal{T}_{\mathrm{S}_{\eta_{x_k}}} \left(\mathcal{T}_{R_{\eta_{x_k}}}^{*} \mathrm{grad}\, f(x_{k+1}) - \mathrm{grad}\, f(x_k) \right) \end{aligned}$$

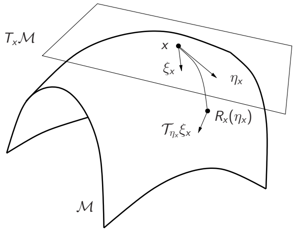
where $\mathcal{T}_{R_{\eta_x}} \xi_x = \frac{d}{dt} R(\eta_x + t\xi_x)|_{t=0}$ is the vector transport by differentiated retraction

Riemannian BFGS Methods

Existing generic Riemannian BFGS methods

- Huang, Absil, Gallivan [HGA15, HAG18]: retraction, isometric vector transport consistent with VT by differentiated retraction along a direction
  - Idea: quasi-Newton update in tangent space, then transport to new tangent space isometrically;
  - Retraction: no constraints;
  - Vector transport: Isometric VT consistent with VT by differentiated retraction along a direction;





Riemannian BFGS Methods

Existing generic Riemannian BFGS methods

Huang, Absil, Gallivan [HGA15, HAG18]: retraction, isometric vector transport consistent with VT by differentiated retraction along a direction

---

Euclidean setting: (Wolfe second condition $\Longrightarrow s_k^T y_k > 0$)

- Define $h(t) = f(x_k + tp_k)$. $\frac{dh}{dt}(\alpha_k) \geq c_2 \frac{dh}{dt}(0)$, $c_2 \in (0,1)$

$$\left.\begin{array}{c} \frac{dh}{dt}(\alpha_k) = p_k^T \nabla f(x_{k+1}) \\ \frac{dh}{dt}(0) = p_k^T \nabla f(x_k) \\ s_k = \alpha_k p_k \end{array}\right\} \Longrightarrow s_k^T \nabla f(x_{k+1}) \geq c_2 s_k^T \nabla f(x_k)$$

$$\Longrightarrow s_k^T y_k = s_k^T(\nabla f(x_{k+1}) - \nabla f(x_k)) \geq \alpha_k (c_2 - 1) p_k^T \nabla f(x_k) > 0.$$

- $B_k \succ 0 + s_k^T y_k > 0 \Longrightarrow B_{k+1} \succ 0 \rightarrow p_{k+1} = -B_{k+1}^{-1} \nabla f(x_{k+1})$ is descent

Riemannian BFGS Methods

Existing generic Riemannian BFGS methods

Riemannian setting: (Wolfe second condition ? $\Longrightarrow s_k^T y_k > 0$)

- Define $h(t) = f(R_{x_k}(tp_k))$. $\frac{dh}{dt}(\alpha_k) \geq c_2 \frac{dh}{dt}(0)$, $c_2 \in (0,1)$

$$\left.\begin{aligned} \tfrac{dh}{dt}(\alpha_k) &= g(\mathcal{T}_{R_{\alpha_k p_k}} p_k, \mathrm{grad}\, f(x_{k+1}) \\ \tfrac{dh}{dt}(0) &= g(p_k, \mathrm{grad}\, f(x_k)) \\ s_k &= \mathcal{T}_{\mathrm{S}_{\alpha_k p_k}} \alpha_k p_k \end{aligned}\right\}$$

$$\Longrightarrow g(\mathcal{T}_{R_{\alpha_k p_k}} \alpha_k p_k, \mathrm{grad}\, f(x_{k+1}) \geq c_2 g(\alpha_k p_k, \mathrm{grad}\, f(x_k))$$

$$\Longrightarrow \left\{\begin{array}{ll} \text{RW:} & g(\mathcal{T}_{R_{\alpha_k p_k}} \alpha_k p_k, \mathrm{grad}\, f(x_{k+1}) = g(\alpha_k p_k, \mathcal{T}^*_{R_{\alpha_k p_k}} \mathrm{grad}\, f(x_{k+1})) \\ \text{HGA:} & g(\mathcal{T}_{R_{\alpha_k p_k}} \alpha_k p_k, \mathrm{grad}\, f(x_{k+1}) = g(\alpha_k p_k, \beta_k^{-1} \mathcal{T}^{-1}_{\mathrm{S}_{\alpha_k p_k}} \mathrm{grad}\, f(x_{k+1})) \end{array}\right.$$

- $y_k = \beta_k^{-1} \mathrm{grad}\, f(x_{k+1}) - \mathcal{T}_{\mathrm{S}_{\alpha_k p_k}} \mathrm{grad}\, f(x_k)$, where $\beta_k = \frac{\|\alpha_k p_k\|}{\|\mathcal{T}_{R_{\alpha_k p_k}} \alpha_k p_k\|}$ and $\mathcal{T}_{\mathrm{S}}$ satisfies the "locking condition":
  $\mathcal{T}_{\mathrm{S}_\xi} \xi = \beta \mathcal{T}_{R_\xi} \xi, \quad \beta = \frac{\|\xi\|}{\|\mathcal{T}_{R_\xi} \xi\|}$, for all $\xi \in \mathrm{T}_x \mathcal{M}$ and all $x \in \mathcal{M}$.
- Wolfe second condition $\Longrightarrow g(s_k, y_k) > 0$

Riemannian SR1 Method

Trust region SR1 method: from Euclidean to Riemannian

- Approximately solve a local model:

$$\eta_k \approx \operatorname*{argmin}_{\|\eta\| \leq \Delta_k} \operatorname{grad} f(x_k)^T \eta + \frac{1}{2}\eta^T B_k \eta;$$

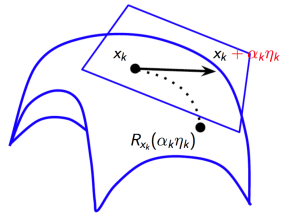
- Quality measurement $\rho_k = \frac{f(x_k) - f(x_k + \eta_k)}{m_k(0) - m_k(\eta_k)}$;
- Update radius Δ_k, and update iterate:

$$x_{k+1} = \begin{cases} \underline{x_k + \eta_k} & \text{if } \rho_k \text{ is sufficient large} \\ x_k & \text{otherwise.} \end{cases}$$

- B_k update:

$$B_{k+1} = \underline{B_k + \frac{(y_k - B_k s_k)(y_k - B_k s_k)^T}{(y_k - B_k s_k)^T s_k}}$$

where $s_k = \underline{\eta_k}$ and $y_k = \underline{\operatorname{grad} f(x_{k+1}) - \operatorname{grad} f(x_k)}$;



Optimization on a Manifold

Riemannian SR1 Method

Trust region SR1 method: from Euclidean to Riemannian

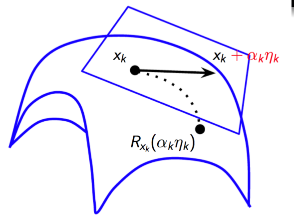
- Approximately solve a local model:

$$\eta_k \approx \operatorname*{argmin}_{\|\eta\| \leq \Delta_k} \ \mathrm{grad}\, f(x_k)^T \eta + \frac{1}{2}\eta^T B_k \eta;$$

- Quality measurement $\rho_k = \frac{f(x_k) - f(x_k + \eta_k)}{m_k(0) - m_k(\eta_k)}$;
- Update radius Δ_k, and update iterate:

$$x_{k+1} = \begin{cases} \underline{x_k + \eta_k} & \text{if } \rho_k \text{ is sufficient large} \\ x_k & \text{otherwise.} \end{cases}$$

$\longleftarrow$ replace by $R_{x_k}(\eta_k)$



Optimization on a Manifold

- B_k update:

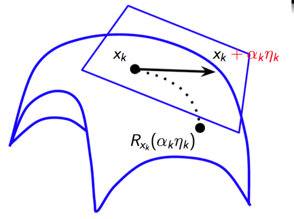
$$B_{k+1} = \underline{B_k + \frac{(y_k - B_k s_k)(y_k - B_k s_k)^T}{(y_k - B_k s_k)^T s_k}}$$

$\longleftarrow$ use vector transport

where $s_k = \underline{\eta_k}$ and $y_k = \underline{\mathrm{grad}\, f(x_{k+1}) - \mathrm{grad}\, f(x_k)}$;

Riemannian SR1 method

Trust region SR1 method: from Euclidean to Riemannian



Optimization on a Manifold

- Approximately solve a local model:

$$\eta_k \approx \operatorname*{argmin}_{\|\eta\| \leq \Delta_k, \eta \in \mathrm{T}_{x_k} \mathcal{M}} \operatorname{grad} f(x_k)^\flat \eta + \frac{1}{2} \eta^\flat B_k \eta$$

- Quality measurement $\rho_k = \frac{f(x_k) - f(R_{x_k}(\eta_k))}{m_k(0) - m_k(\eta_k)}$;
- Update radius Δ_k, and update iterate:

$$x_{k+1} = \begin{cases} R_{x_k}(\eta_k) & \text{if } \rho_k \text{ is sufficient large} \\ x_k & \text{otherwise.} \end{cases}$$

- B_k update:

$$\tilde{B}_k = \mathcal{T}_{\eta_k} \circ B_k \circ \mathcal{T}_{\eta_k}^{-1}, \qquad \text{Extra cost on vector transports!}$$

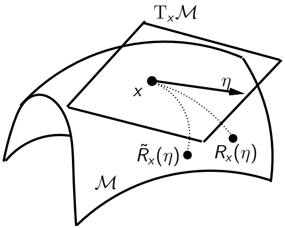
$$B_{k+1} = \tilde{B}_k + (y_k - \tilde{B}_k s_k)(y_k - \tilde{B}_k s_k)^T / ((y_k - \tilde{B}_k s_k)^T s_k)$$

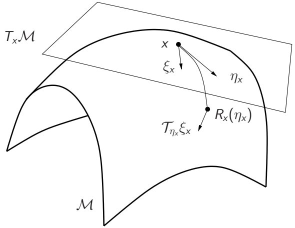
where $s_k = \mathcal{T}_{\eta_k}(\eta_k)$, and $y_k = \operatorname{grad} f(x_{k+1}) - \mathcal{T}_{\eta_k} \operatorname{grad} f(x_k)$;

Riemannian SR1 Method

Existing generic Riemannian SR1 method

- Huang, Absil, Gallivan [HAG15]: retraction, isometric vector transport
  - Idea: quasi-Newton update in tangent space, then transport to new tangent space isometrically;

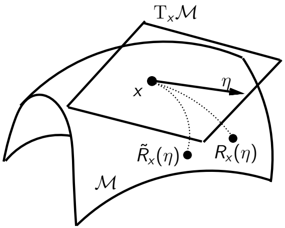


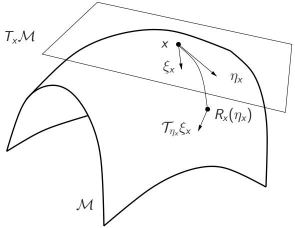


Riemannian SR1 Method

Existing generic Riemannian SR1 method

- Huang, Absil, Gallivan [HAG15]: retraction, isometric vector transport
  - Idea: quasi-Newton update in tangent space, then transport to new tangent space isometrically;
  - Retraction: no constraints;

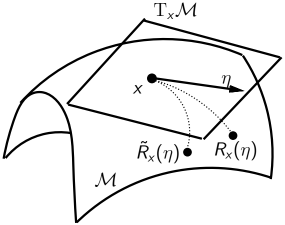


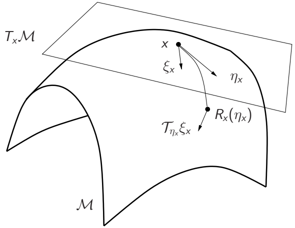


Riemannian SR1 Method

Existing generic Riemannian SR1 method

- Huang, Absil, Gallivan [HAG15]: retraction, isometric vector transport
  - Idea: quasi-Newton update in tangent space, then transport to new tangent space isometrically;
  - Retraction: no constraints;
  - Vector transport: Isometric VT;





Outline:

- Introduction
- Riemannian Quasi-Newton Methods
- Implementation Techniques
- Limited-memory Versions
- Applications

Implementation Techniques

Summary:

- Isometric vector transport is needed [RW12, HGA15, HAG18];
- An efficient vector transport is crucial;

Implementation Techniques

Summary:

- Isometric vector transport is needed [RW12, HGA15, HAG18];
- An efficient vector transport is crucial;

An efficient isometric vector transport:

- Representative manifold:
  - the Stiefel manifold $\mathrm{St}(p, n) = \{X \in \mathbb{R}^{n\times p} | X^T X = I_p\}$;
  - canonical metric: $g(\eta_X, \xi_X) = \mathrm{trace}\left(\eta_X^T \left(I_n - \frac{1}{2} X X^T\right) \xi_X\right)$;
- The idea in this talk can be used for more algorithms and many commonly-encountered manifolds.

Implementation Techniques

Representations of Tangent Vectors

- $\mathcal{E} = \mathbb{R}^w$;
- Dimension of $\mathcal{M}$ is d;
- Stiefel manifold: $\mathcal{E} = \mathbb{R}^{n \times p}$;
- Stiefel manifold: $d = np - p(p+1)/2$;

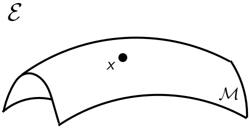


Figure: An embedded submanifold

- Extrinsic: $\eta_x \in \mathbb{R}^w$; $(T_X = \{X\Omega + X_\perp K \mid \Omega^T = -\Omega, X^T X_\perp = 0\})$
- Intrinsic: $\tilde{\eta}_x \in \mathbb{R}^d$ such that $\eta_x = B_x \tilde{\eta}_x$, where B_x is smooth;

Implementation Techniques

Representations of Tangent Vectors

- $\mathcal{E} = \mathbb{R}^w$;
- Dimension of $\mathcal{M}$ is d;

- Stiefel manifold: $\mathcal{E} = \mathbb{R}^{n \times p}$;
- Stiefel manifold: $d = np - p(p+1)/2$;

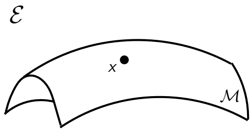


Figure: An embedded submanifold

- Extrinsic: $\eta_x \in \mathbb{R}^w$; $(T_X = \{X\Omega + X_\perp K \mid \Omega^T = -\Omega, X^T X_\perp = 0\})$
- Intrinsic: $\tilde{\eta}_x \in \mathbb{R}^d$ such that $\eta_x = B_x \tilde{\eta}_x$, where B_x is smooth;

How to find a basis B?

Implementation Techniques

Extrinsic Representation and Intrinsic Representation on the Stiefel Manifold

$$\mathrm{T}_X \mathrm{St}(p,n) = \{X\Omega + X_\perp K \mid \Omega^T = -\Omega, X^T X_\perp = 0\};$$

$$B_x = \left\{ \begin{bmatrix} X & X_\perp \end{bmatrix} \left[\begin{array}{cccc} 0 & 1 & \dots & 0 \\ -1 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 \\ \hline 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 \end{array}\right], \dots, \begin{bmatrix} X & X_\perp \end{bmatrix} \left[\begin{array}{cccc} 0 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 \\ \hline 1 & 0 & \dots & 0 \\ 0 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 0 \end{array}\right] \right\}$$

Implementation Techniques

Extrinsic Representation and Intrinsic Representation on the Stiefel Manifold

$$\mathrm{T}_X \mathrm{St}(p, n) = \{X\Omega + X_\perp K \mid \Omega^T = -\Omega, X^T X_\perp = 0\};$$

Extrinsic η_X:

$$\eta_X = \begin{bmatrix} X & X_\perp \end{bmatrix} \begin{bmatrix} \Omega \\ K \end{bmatrix}$$

$$= \begin{bmatrix} X & X_\perp \end{bmatrix} \begin{bmatrix} 0 & a_{12} & \dots & a_{1p} \\ -a_{12} & 0 & \dots & a_{2p} \\ \dots & \dots & \dots & \dots \\ -a_{1p} & -a_{2p} & \dots & 0 \\ b_{11} & b_{12} & \dots & b_{1p} \\ b_{21} & b_{22} & \dots & b_{2p} \\ \dots & \dots & \dots & \dots \\ b_{(n-p)1} & b_{(n-p)2} & \dots & b_{(n-p)p} \end{bmatrix}$$

Intrinsic $\tilde{\eta}_X$:

$$\tilde{\eta}_X = \begin{bmatrix} a_{12} \\ a_{13} \\ a_{23} \\ \vdots \\ a_{(p-1)p} \\ b_{11} \\ b_{21} \\ \vdots \\ b_{(n-p)p} \end{bmatrix}$$

Implementation Techniques

Extrinsic Representation and Intrinsic Representation on the Stiefel Manifold

Question

Extrinsic representation $\eta_X \Longleftrightarrow$ Intrinsic representation $\tilde{\eta}_X$

- $\eta_X = \begin{bmatrix} X & X_\perp \end{bmatrix} \begin{bmatrix} \Omega \\ K \end{bmatrix} \Leftrightarrow \begin{bmatrix} \Omega \\ K \end{bmatrix} \Leftrightarrow \tilde{\eta}_X$

Implementation Techniques

Extrinsic Representation and Intrinsic Representation on the Stiefel Manifold

Question

Extrinsic representation $\eta_X \Longleftrightarrow$ Intrinsic representation $\tilde{\eta}_X$

- $\eta_X = \begin{bmatrix} X & X_\perp \end{bmatrix} \begin{bmatrix} \Omega \\ K \end{bmatrix} \Leftrightarrow \begin{bmatrix} \Omega \\ K \end{bmatrix} \Leftrightarrow \tilde{\eta}_X$
- Apply Householder transformation to X, (Done in retraction)

$$Q_p^T Q_{p-1}^T \dots Q_1^T X = R = I_{n \times p}.$$

Implementation Techniques

Extrinsic Representation and Intrinsic Representation on the Stiefel Manifold

Question

Extrinsic representation $\eta_X \Longleftrightarrow$ Intrinsic representation $\tilde{\eta}_X$

- $\eta_X = \begin{bmatrix} X & X_\perp \end{bmatrix} \begin{bmatrix} \Omega \\ K \end{bmatrix} \Leftrightarrow \begin{bmatrix} \Omega \\ K \end{bmatrix} \Leftrightarrow \tilde{\eta}_X$
- Apply Householder transformation to X, (Done in retraction)

$$Q_p^T Q_{p-1}^T \dots Q_1^T X = R = I_{n \times p}.$$

- $\begin{bmatrix} X & X_\perp \end{bmatrix} = Q_1 Q_2 \dots Q_p$ (Do not compute)

Implementation Techniques

Extrinsic Representation and Intrinsic Representation on the Stiefel Manifold

Question

Extrinsic representation $\eta_X \Longleftrightarrow$ Intrinsic representation $\tilde{\eta}_X$

- $\eta_X = \begin{bmatrix} X & X_\perp \end{bmatrix} \begin{bmatrix} \Omega \\ K \end{bmatrix} \Leftrightarrow \begin{bmatrix} \Omega \\ K \end{bmatrix} \Leftrightarrow \tilde{\eta}_X$
- Apply Householder transformation to X, (Done in retraction)
$$Q_p^T Q_{p-1}^T \dots Q_1^T X = R = I_{n \times p}.$$
- $\begin{bmatrix} X & X_\perp \end{bmatrix} = Q_1 Q_2 \dots Q_p$ (Do not compute)
- Extrinsic to Intrinsic: $Q_p^T Q_{p-1}^T \dots Q_1^T \eta_X = \begin{bmatrix} \Omega \\ K \end{bmatrix}$ and reshape to $\tilde{\eta}_X$; $(4np^2 - 2p^3)$ flops
- Intrinsic to Extrinsic: reshape $\tilde{\eta}_X$ and $\eta_X = Q_1 Q_2 \dots Q_p \begin{bmatrix} \Omega \\ K \end{bmatrix}$; $(4np^2 - 2p^3)$ flops

Implementation Techniques

Benefits of Intrinsic Representation

- Operations on tangent vectors are cheaper since $d \leq w$;
- If the basis is orthonormal, then the Riemannian metric reduces to the Euclidean metric:

$$g(\eta_x, \xi_x) = g(B_x \tilde{\eta}_x, B_x \tilde{\xi}_x) = \tilde{\eta}_x^T \tilde{\xi}_x.$$

$$\text{Stiefel: } \mathrm{trace}\left(\eta_X^T \left(I_n - \tfrac{1}{2} X X^T\right) \xi_X\right) \quad \longrightarrow \quad \tilde{\eta}_X^T \tilde{\xi}_X$$

- A vector transport has identity implementation, i.e., $\widetilde{\mathcal{T}}_\eta = \mathrm{id}$.

Implementation Techniques

Vector Transport by Parallelization

- Vector transport by parallelization:

$$\mathcal{T}_{\eta_x}\xi_x = B_y B_x^{\dagger}\xi_x;$$

where $y = R_x(\eta_x)$ and $\dagger$ denotes pseudo-inverse, has identity implementation [HAG16]:

$$\mathcal{T}_{\tilde{\eta}_x}\tilde{\xi}_x = \tilde{\xi}_x.$$

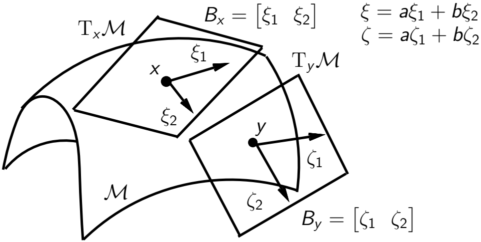
Example:

Extrinsic:

$$\zeta = \mathcal{T}_\eta \xi = B_y B_x^{\dagger}\xi$$

Intrinsic:

$$\tilde{\zeta} = \widetilde{\mathcal{T}_\eta \xi} = B_y^{\dagger} B_y B_x^{\dagger} B_x \tilde{\xi} = \tilde{\xi}$$



Outline:

- Introduction
- Riemannian quasi-Newton methods: RBFGS and RTR-SR1
- Implementation techniques
- Limited-memory versions
  - LRBFGS
  - LRTR-SR1
- Applications

Limited-memory Versions

A limited-memory Riemannian BFGS method

Search direction: $\eta_k = \mathcal{B}_k^{-1} \operatorname{grad} f(x_k)$

- Follow the same idea of the Euclidean limited-memory BFGS method
- Inverse Hessian approximation update
- Two-loop recursion

Limited-memory Versions

A limited-memory Riemannian BFGS method

Sherman-Morrison formula $\Rightarrow$ Inverse update ($H_k = B_k^{-1}$):

$$H_{k+1} = \mathcal{V}_k^\flat \tilde{H}_k \mathcal{V}_k + \rho_k s_k s_k^\flat, \text{ where } \rho_k = \frac{1}{g(y_k, s_k)} \text{ and } \mathcal{V}_k = \mathrm{id} - \rho_k y_k s_k^\flat.$$

If the number of latest s_k and y_k we use is $m+1$, then

$$\begin{aligned} H_{k+1} = {} & \tilde{\mathcal{V}}_k^\flat \tilde{\mathcal{V}}_{k-1}^\flat \cdots \tilde{\mathcal{V}}_{k-m}^\flat \tilde{H}_{k+1}^0 \tilde{\mathcal{V}}_{k-m} \cdots \tilde{\mathcal{V}}_{k-1} \tilde{\mathcal{V}}_k \\ & + \rho_{k-m} \tilde{\mathcal{V}}_k^\flat \tilde{\mathcal{V}}_{k-1}^\flat \cdots \tilde{\mathcal{V}}_{k-m+1}^\flat s_{k-m}^{(k+1)} s_{k-m}^{(k+1)^\flat} \tilde{\mathcal{V}}_{k-m+1} \cdots \tilde{\mathcal{V}}_{k-1} \tilde{\mathcal{V}}_k \\ & + \cdots \\ & + \rho_k s_k^{(k+1)} s_k^{(k+1)^\flat}, \end{aligned}$$

where $\tilde{\mathcal{V}}_i = \mathrm{id} - \rho_i y_i^{(k+1)} s_i^{(k+1)^\flat}$ and $H_{k+1}^0 = \frac{g(s_k, y_k)}{g(y_k, y_k)} \mathrm{id}$.

Limited-memory Versions

A limited-memory Riemannian BFGS method

Given compute $H_{k+1} \operatorname{grad} f(x_{k+1})$:

Algorithm 1 LRBFGS two-loop recursion

$q \leftarrow \nabla f(x_{k+1})$;
for $i = k, k-1, \dots, k-m+1$ **do**
  $\alpha_i \leftarrow \rho_i s_i^{\flat} q$;
  $q \leftarrow q - \alpha_i y_i$;
end for
$r \leftarrow H_{k+1}^{(0)} q$;
for $i = k-m+1, k-m+2, \dots, k$ **do**
  $\beta \leftarrow \rho_i y_i^{\flat} r$;
  $r \leftarrow r + s_i(\alpha_i - \beta)$;
end for
return r;

Computational complexity $O(md)$

Limited-memory Versions

A limited-memory Riemannian trust-region SR1 method

Solve the subproblem: $\eta_k = \operatorname*{argmin}_{\|\eta\| \leq \Delta_k, \eta \in \mathrm{T}_{x_k} \mathcal{M}} \operatorname{grad} f(x_k)^\flat \eta + \frac{1}{2} \eta^\flat B_k \eta;$

- Intrinsic representation using orthonormal basis
- Reduce to the subproblem of Euclidean TR-SR1
- Solved efficient [BEM17, HG21]

Limited-memory Versions

A limited-memory Riemannian trust-region SR1 method

Subproblem:

$$\eta_k = \operatorname*{argmin}_{\|\eta\| \leq \Delta_k, \eta \in \mathrm{T}_{x_k} \mathcal{M}} \mathrm{grad}\, f(x_k)^\flat \eta + \frac{1}{2} \eta^\flat B_k \eta;$$

- $B_k = \gamma_k \,\mathrm{id} + \Psi_{k,m} M_{k,m}^\dagger \Psi_{k,m}^\flat$
- $\gamma_k \in \mathbb{R}$, $M_{k,m} \in \mathbb{R}^{m \times m}$ and $\Psi_{k,m}$ consists of m tangent vectors, related to $(s_i, y_i), i = k-1, \ldots, k-m$

Using intrinsic representation:

$$c^* = \operatorname*{argmin}_{\|c\| \leq \Delta_k} q^T c + \frac{1}{2} c^T W c;$$

- $W = \gamma_k I + \Phi_{k,m} M_{k,m}^\dagger \Phi_{k,m}^T \in \mathbb{R}^{d \times d}$
- $\gamma_k \in \mathbb{R}$, $M_{k,m} \in \mathbb{R}^{m \times m}$ and $\Phi_{k,m} \in \mathbb{R}^{d \times m}$

Limited-memory Versions

A limited-memory Riemannian trust-region SR1 method

Theorem

The vector p^* is a global solution of the trust region subproblem

$$\min_{\|p\| \leq \Delta} q^T c + \frac{1}{2} c^T W c$$

if and only if c^* is feasible and there is a scalar $\lambda \geq 0$ such that the following conditions hold:

$$(W + \lambda I)p^* = -q, \lambda(\Delta - \|c^*\|) = 0, (W + \lambda I) \text{ is SPSD.}$$

- Eigenvalues of W: inexpensive
- $\varphi(\lambda) = -(W + \lambda I_d)^\dagger q$ and $\phi(\lambda) = 1/\|\varphi(\lambda)\|_2 - 1/\Delta$
- Complexity $O(md)$

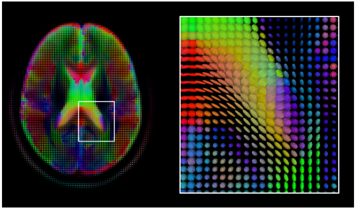
Outline:

- Introduction
- Riemannian quasi-Newton methods: RBFGS and RTR-SR1
- Implementation techniques
- Limited-memory versions
- Applications

Geometric mean of SPD matrices

Motivation of averaging SPD matrices

- Possible applications of SPD matrices
  - Diffusion tensors in medical imaging [CSV12, FJ07, RTM07]
  - Describing images and video [LWM13, SFD02, ASF+05, TPM06, HWSC15]
- Motivation of averaging SPD matrices
  - denoising / interpolation
  - clustering / classification



Geometric mean of SPD matrices

Karcher mean

Karcher mean [Kar77]:

$$G(A_1, \dots, A_K) = \operatorname*{argmin}_{X \in \mathcal{S}^n_{++}} \frac{1}{2K} \sum_{i=1}^{K} \delta^2(X, A_i), \tag{1}$$

where $\delta(X, Y) = \| \log(X^{-1/2} Y X^{-1/2}) \|_F$ is the geodesic distance under the affine-invariant metric

$$g(\eta_X, \xi_X) = \mathrm{trace}(\eta_X X^{-1} \xi_X X^{-1})$$

Geometric mean of SPD matrices

Numerical experiments

- Richardson-like iteration [BI13]
- RSD-QR [RA11]
- Riemannian BB method [IP18]
- Majorization [Zha17]

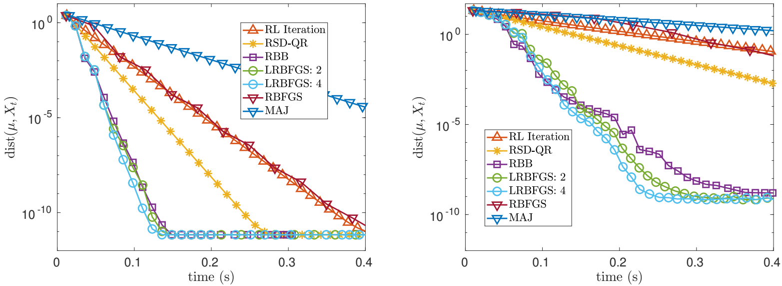
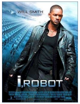


Figure: $K = 30$, $n = 60$, $10 \leq \kappa(A_i) \leq 60$; Bottom right: $K = 30$, $n = 60$, $10^5 \leq \kappa(A_i) \leq 10^9$;

Matrix completion

Recommender system



movie i

$$M = \begin{pmatrix} 5 & ? & 2 & ? & ? & 5 & ? & ? & 3 & ? & 5 & ? & ? & ? & 2 & ? \\ 3 & ? & 2 & ? & ? & 2 & ? & ? & ? & 3 & 2 & ? & 5 & ? & ? & ? \\ 1 & ? & 5 & 2 & 3 & 4 & ? & 4 & ? & ? & ? & 2 & ? & ? & ? & ? \\ ? & 1 & ? & 3 & ? & ? & ? & 3 & ? & ? & ? & ? & 2 & 1 & 5 & 5 \\ ? & 4 & ? & ? & ? & ? & 5 & ? & ? & ? & 1 & ? & ? & 1 & ? & 4 \end{pmatrix} \begin{array}{l} \\ \text{user } u \\ \\ \\ \\ \end{array}$$

Matrix completion

A model

$$
\begin{array}{c}
\text{movies} \\
\begin{pmatrix}
a_{11} & & & a_{14} \\
 & & & a_{24} \\
 & & a_{33} & \\
a_{41} & & & \\
 & a_{52} & a_{53} &
\end{pmatrix}
\end{array}
=
\begin{array}{c}
\text{meta-user} \\
\begin{pmatrix}
b_{11} & b_{12} \\
b_{21} & b_{22} \\
b_{31} & b_{32} \\
b_{41} & b_{42} \\
b_{51} & b_{52}
\end{pmatrix}
\end{array}
\begin{array}{c}
\text{meta-movie} \\
\begin{pmatrix}
c_{11} & c_{12} & c_{13} & c_{14} \\
c_{21} & c_{22} & c_{23} & c_{24}
\end{pmatrix}
\end{array}
$$

- Minimize the cost function

$$f : \mathbb{R}_r^{m \times n} \to \mathbb{R} : X \mapsto f(X) = \|P_\Omega M - P_\Omega X\|_F^2.$$

- $\mathbb{R}_r^{m \times n}$ is the set of m-by-n matrices with rank r.

Matrix completion

Numerical experiments

Table: An average of 50 random runs of (i) LRBFGS, (ii) RCG, (iii) RNewton, and (iv) RTRNewton methods in ROPTLIB. $\mathrm{OS} = 3$

	$m = 100, n = 200, r = 10$				$m = 1000, n = 2000, r = 10$			
	(i)	(ii)	(iii)	(iv)	(i)	(ii)	(iii)	(iv)
iter	34	40	12	13	57	67	19	18
nf	37	53	14	14	61	99	23	19
ng	35	41	13	14	58	68	20	19
nR	36	52	13	13	60	98	22	18
nV	257	81	0	0	445	135	0	0
nH	0	0	64	58	0	0	108	94
$\frac{\|\mathrm{gf}_f\|}{\|\mathrm{gf}_0\|}$	6.72_{-7}	7.38_{-7}	8.72_{-8}	4.71_{-8}	7.59_{-7}	7.59_{-7}	8.33_{-8}	1.32_{-7}
t	2.84_{-2}	3.24_{-2}	7.68_{-2}	7.17_{-2}	4.25_{-1}	5.27_{-1}	1.34	1.17
f	1.59_{-8}	1.29_{-8}	7.53_{-10}	5.53_{-10}	1.49_{-6}	1.21_{-6}	6.02_{-8}	1.23_{-7}
err	3.12_{-6}	2.55_{-6}	2.56_{-7}	1.35_{-7}	1.63_{-5}	1.51_{-5}	1.24_{-6}	1.73_{-6}

Matrix completion

Compared Algorithms

- An accelerated proximal gradient singular value thresholding algorithm (NNLS) [TY12]
- RCG in [Van13]
- LRBFGS in ROPTLIB
- RSD with BB initial step size

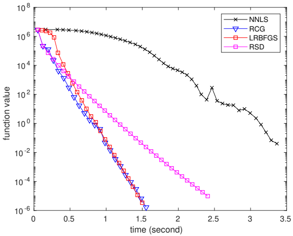


Figure: A representative result with $m = n = 50000$, $r = 10$, $\mathrm{OS} = 6$

Matrix completion

Numerical experiments

Table: An average result of 10 random runs with $m = n = 200$, $r = 4$. $\mathrm{OS} = 3$.

	LRBFGS	LRBFGS-R	LRTRSR1	LRTRSR1-R
iter	68	65	66	52
nf	75	73	67	53
ng	69	66	67	53
$\frac{\\|\mathrm{gf}_f\\|}{\\|\mathrm{gf}_0\\|}$	7.87_{-7}	8.21_{-7}	8.29_{-7}	7.60_{-7}
f	5.53_{-3}	6.27_{-3}	1.22_{-2}	8.52_{-3}
time	2.95_{-2}	2.54_{-2}	2.98_{-2}	2.19_{-2}

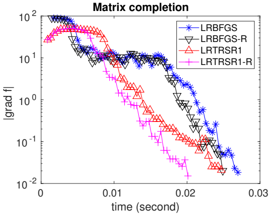


Figure: A representative trajectory

Conclusion

- Riemannian BFGS and SR1 methods
- Intrinsic representation of tangent vectors and operators
- Vector transport by parallelization
- Limited-memory versions of Riemannian BFGS and SR1 methods
- Geometric mean of SPD matrices and matrix completion

Thank you

Thank you!

References I

Ognjen Arandjelovic, Gregory Shakhnarovich, John Fisher, Roberto Cipolla, and Trevor Darrell.
Face recognition with image sets using manifold density divergence.
In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 581–588. IEEE, 2005.

Johannes Brust, Jennifer B. Erway, and Roummel F. Marcia.
On solving L-SR1 trust-region subproblems.
Computational Optimization and Applications, 66(2):245–266, Mar 2017.

D. A. Bini and B. Iannazzo.
Computing the Karcher mean of symmetric positive definite matrices.
Linear Algebra and its Applications, 438(4):1700–1710, February 2013. doi:10.1016/j.laa.2011.08.052.

Guang Cheng, Hesamoddin Salehian, and Baba Vemuri.
Efficient recursive algorithms for computing the mean diffusion tensor and applications to DTI segmentation.
Computer Vision–ECCV 2012, pages 390–401, 2012.

P. T. Fletcher and S. Joshi.
Riemannian geometry for the statistical analysis of diffusion tensor data.
Signal Processing, 87(2):250–262, 2007.

W. Huang, P.-A. Absil, and K. A. Gallivan.
A Riemannian symmetric rank-one trust-region method.
Mathematical Programming, 150(2):179–216, February 2015.

References II

W. Huang, P.-A. Absil, and K. A. Gallivan.
Intrinsic representation of tangent vectors and vector transport on matrix manifolds.
Numerische Mathematik, 136(2):523–543, 2016.

Wen Huang, P.-A. Absil, and K. A. Gallivan.
A Riemannian BFGS method without differentiated retraction for nonconvex optimization problems.
SIAM Journal on Optimization, 28(1):470–495, 2018.

W. Huang and K. A. Gallivan.
A limited-memory Riemannian symmetric rank-one trust-region method with an efficient algorithm for its subproblem.
In *Proceedings of the 24th Internaltional Symposium on Mathematical Theory of Networks and Systems*, 2021.
accepted.

W. Huang, K. A. Gallivan, and P.-A. Absil.
A Broyden Class of Quasi-Newton Methods for Riemannian Optimization.
SIAM Journal on Optimization, 25(3):1660–1685, 2015.

Zhiwu Huang, Ruiping Wang, Shiguang Shan, and Xilin Chen.
Face recognition on large-scale video in the wild with hybrid Euclidean-and-Riemannian metric learning.
Pattern Recognition, 48(10):3113–3124, 2015.

Bruno Iannazzo and Margherita Porcelli.
The riemannian barzilai-borwein method with nonmonotone line search and the matrix geometric mean computation.
IMA Journal of Numerical Analysis, 38(1):495–517, 2018.

H. Karcher.
Riemannian center of mass and mollifier smoothing.
Communications on Pure and Applied Mathematics, 1977.

References III

Jiwen Lu, Gang Wang, and Pierre Moulin.
Image set classification using holistic multiple order statistics features and localized multi-kernel metric learning.
In *Proceedings of the IEEE International Conference on Computer Vision*, pages 329–336, 2013.

C. Qi.
Numerical optimization methods on Riemannian manifolds.
PhD thesis, Florida State University, Department of Mathematics, 2011.

Q. Rentmeesters and P.-A. Absil.
Algorithm comparison for karcher mean computation of rotation matrices and diffusion tensors.
19th European Signal Processing Conference (EUSIPCO 2011), (Eusipco):2229–2233, 2011.

Y. Rathi, A. Tannenbaum, and O. Michailovich.
Segmenting images on the tensor manifold.
In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2007.

W. Ring and B. Wirth.
Optimization methods on Riemannian manifolds and their application to shape space.
SIAM Journal on Optimization, 22(2):596–627, January 2012.
doi:10.1137/11082885X.

Gregory Shakhnarovich, John W Fisher, and Trevor Darrell.
Face recognition from long-term observations.
In *European Conference on Computer Vision*, pages 851–865. Springer, 2002.

Oncel Tuzel, Fatih Porikli, and Peter Meer.
Region covariance: A fast descriptor for detection and classification.
In *European conference on computer vision*, pages 589–600. Springer, 2006.

References IV

Kim-Chuan Toh and Sangwoon Yun.
An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems.
Pacific Journal of Optimization, 6(3):615–640, 2012.

B. Vandereycken.
Low-rank matrix completion by Riemannian optimization—extended version.
SIAM Journal on Optimization, 23(2):1214–1236, 2013.

T. Zhang.
A Majorization-Minimization Algorithm for Computing the Karcher Mean of Positive Definite Matrices.
SIAM Journal on Matrix Analysis and Applications, 38(2):387–400, 2017.