Randomized Riemannian Preconditionning for Canonical Correlation Analysis

(More generally: optimization with quadratic equality constraints)

Haim Avron (Tel Aviv University)
Joint work with Boris Shustin (TAU)

Workshop on Randomized Numerical Linear Algebra and Applications,
Simons Institute, September 2018

Two Approaches for Introducing Randomization in NLA

Sketch-and-Solve

1. Sketch the input
2. ... to form a smaller problem
3. ... and solve it exactly
4. ... use solution to form an approximate solution to the original problem

Sketch-to-Precondition

1. Sketch the input
2. Use the sketch to form a preconditioner
3. Use an iterative method + preconditioner

Example: $\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2$

Sketch-and-Solve

1. $\mathbf{B} \leftarrow \mathbf{S}\mathbf{A}$, $\mathbf{c} \leftarrow \mathbf{S}\mathbf{b}$
2. New problem: $\min_{\mathbf{y}} \|\mathbf{B}\mathbf{y} - \mathbf{c}\|_2$
3. $\mathbf{y} \leftarrow \mathbf{B}^{+}\mathbf{c}$ (via QR or SVD)
4. $\mathbf{x} \leftarrow \mathbf{y}$

Sketch-to-Precondition

1. $\mathbf{B} \leftarrow \mathbf{S}\mathbf{A}$
2. $[\mathbf{Q}, \mathbf{R}] \leftarrow \mathrm{qr}(\mathbf{B})$
3. $\mathbf{x} \leftarrow \mathrm{LSQR}(\mathbf{A}, \mathbf{b}, \mathbf{R})$

Two Approaches for Introducing Randomization in NLA

Sketch-and-Solve

1. High success rate
2. Polynomial accuracy dependence (e.g. ϵ^{-2})
3. No iterations

Pros:

1. **Very** fast
2. Deterministic running time

Cons:

1. Only crude accuracy
2. "Monte-Carlo" algorithm

Sketch-to-Precondition

1. High success rate
2. Exponential accuracy dependence (e.g. $\log(1/\epsilon)$)
3. Iterations

Pros:

1. Very high accuracy possible
2. Success = good solution

Cons:

1. Slower than sketch-and-solve
2. Iterations (no streaming)

Two Approaches for Introducing Randomization in NLA

Sketch-and-Solve

1. Linear regression (ordinary, ridge, robust, ...)
2. Constrained linear regression
3. Principal Component Analysis
4. Canonical Correlations Analysis
5. Kernelized methods (KRR, KSVM, KPCA,...)
6. Low-rank approximations
7. Structured decompositions (CUR, NMF, ...)

Non exhaustive list...

Sketch-to-Precondition

1. Linear regression (only: ordinary, ridge, some robust)
2. Kernel ridge regression
3. Laplacian solvers
4. Systems with hierarchical structure
5. Linear systems with tensor product structure (Kressner et al. 2016)

Essentially an exhaustive list...

Two Approaches for Introducing Randomization in NLA

Sketch-and-Solve

1. Linear regression (ordinary, ridge, robust, ...)
2. Constrained linear regression
3. Principal Component Analysis
4. Canonical Correlations Analysis
5. Kernelized methods (KRR, KSVM, KPCA,...)
6. Low-rank approximations
7. Structured decompositions (CUR, NMF, ...)

Non exhaustive list...

Sketch-to-Precondition

1. Linear regression (only: ordinary, ridge, some robust)
2. Kernel ridge regression
3. Laplacian solvers
4. Systems with hierarchical structure
5. Linear systems with tensor product structure (Kressner et al. 2016)

Essentially an exhaustive list...

Can randomized preconditioning be used *beyond regression*?

Executive Summary

This talk: Randomized preconditioning for CCA
(and more generally: problems w/ quadratic equality constrains).

How? Riemannian optimization + Sketching

Key Observations:

1. CCA is an optimization problem with manifold constraints.
2. The metric selection matters.
3. We want to use a specific metric, but using it is expensive.
4. Use sketching to approximate that metric.

(Regularized) Canonical Correlations Analysis (CCA)

Inputs

1. Data matrices $\mathbf{X} \in \mathbb{R}^{n \times d_x}$ and $\mathbf{Y} \in \mathbb{R}^{n \times d_y}$
2. Regularization parameter $\lambda \geq 0$

Goal

Maximize

$$f(\mathbf{u}, \mathbf{v}) = \mathbf{u}^{\mathsf{T}} \mathbf{X}^{\mathsf{T}} \mathbf{Y} \mathbf{v}$$

subject to $\mathbf{u}^{\mathsf{T}}(\mathbf{X}^{\mathsf{T}}\mathbf{X} + \lambda \mathbf{I}_{d_x})\mathbf{u} = 1$ and $\mathbf{v}^{\mathsf{T}}(\mathbf{Y}^{\mathsf{T}}\mathbf{Y} + \lambda \mathbf{I}_{d_y})\mathbf{v} = 1$

Remarks

1. The above is only the *leading correlation.*
2. If $\lambda = 0$ we get principal angles and vectors.

Solving CCA

Direct Method ($\lambda = 0$) (Björck-Golub Algorithm)

1. $[\mathbf{Q}_x, \mathbf{R}_x] \leftarrow \mathrm{qr}\ (\mathbf{X})$
2. $[\mathbf{Q}_y, \mathbf{R}_y] \leftarrow \mathrm{qr}\ (\mathbf{Y})$
3. $[\mathbf{M}, \Sigma, \mathbf{N}] \leftarrow \mathrm{svd}(\mathbf{Q}_x^\top \mathbf{Q}_y)$
4. $\mathbf{u}^\star \leftarrow \mathbf{R}_x^{-1} \mathbf{M}_{:,1}$
   $\mathbf{v}^\star \leftarrow \mathbf{R}_y^{-1} \mathbf{N}_{:,1}$

Cost: $O(n(d_x^2 + d_y^2))$

Solving CCA

Direct Method ($\lambda = 0$) (Björck-Golub Algorithm)

1. $[\mathbf{Q}_x, \mathbf{R}_x] \leftarrow \mathrm{qr}\ (\mathbf{X})$
2. $[\mathbf{Q}_y, \mathbf{R}_y] \leftarrow \mathrm{qr}\ (\mathbf{Y})$
3. $[\mathbf{M}, \Sigma, \mathbf{N}] \leftarrow \mathrm{svd}(\mathbf{Q}_x^{\mathsf{T}} \mathbf{Q}_y)$
4. $\mathbf{u}^{\star} \leftarrow \mathbf{R}_x^{-1} \mathbf{M}_{:,1}$
   $\mathbf{v}^{\star} \leftarrow \mathbf{R}_y^{-1} \mathbf{N}_{:,1}$

Cost: $O(n(d_x^2 + d_y^2))$

Sketch-and-Solve

(A., Boutsidis, Toledo, Zouzias 2014)

1. $\mathbf{X}_s \leftarrow \mathbf{S}\mathbf{X}$
2. $\mathbf{Y}_s \leftarrow \mathbf{S}\mathbf{Y}$
3. $[\tilde{\mathbf{u}}, \tilde{\mathbf{v}}] \leftarrow$ $\mathrm{BjorckGolub}(\mathbf{X}_s, \mathbf{Y}_s)$

Features:

- Improved dependence on n.
- ϵ^{-2} dependence.

Alternating Least Squares Algorithm (Golub and Zha 1995)

Denote $\Sigma_{\mathbf{xx}} = \mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}$ and $\Sigma_{\mathbf{yy}} = \mathbf{Y}^\top \mathbf{Y} + \lambda \mathbf{I}$. Consider the iteration:

$$\begin{aligned}
\tilde{\mathbf{u}}_{k+1} &= \arg\min_{\mathbf{u}} \|\mathbf{X}\mathbf{u} - \mathbf{Y}\mathbf{v}_k\|_2^2 + \lambda\|\mathbf{u}\|_2^2 = \Sigma_{\mathbf{xx}}^{-1}\mathbf{X}^\top\mathbf{Y}\mathbf{v}_k \\
\mathbf{u}_{k+1} &= \tilde{\mathbf{u}}_{k+1} / \tilde{\mathbf{u}}_{k+1}^\top \Sigma_{\mathbf{xx}} \tilde{\mathbf{u}}_{k+1} \\
\tilde{\mathbf{v}}_{k+1} &= \arg\min_{\mathbf{v}} \|\mathbf{Y}\mathbf{v} - \mathbf{X}\mathbf{u}_k\|_2^2 + \lambda\|\mathbf{v}\|_2^2 = \Sigma_{\mathbf{yy}}^{-1}\mathbf{Y}^\top\mathbf{X}\mathbf{u}_k \\
\mathbf{v}_{k+1} &= \tilde{\mathbf{v}}_{k+1} / \tilde{\mathbf{v}}_{k+1}^\top \Sigma_{\mathbf{yy}} \tilde{\mathbf{v}}_{k+1}
\end{aligned}$$

Theorem (Wang, Wang, Garber and Srebro 2016)

Let $\mu \equiv \min((\mathbf{u}_0^\top \Sigma_{\mathbf{xx}} \mathbf{u}^\star)^2, (\mathbf{v}_0^\top \Sigma_{\mathbf{yy}} \mathbf{v}^\star)^2) > 0$. *Then, for*

$$t \geq \left\lceil \frac{\rho_1^2}{\rho_1^2 - \rho_2^2} \right\rceil \log\left(\frac{1}{\mu\epsilon}\right)$$

we have

$$\min((\mathbf{u}_t^\top \Sigma_{\mathbf{xx}} \mathbf{u}^\star)^2, (\mathbf{v}_t^\top \Sigma_{\mathbf{yy}} \mathbf{v}^\star)^2) \geq 1 - \epsilon, \;\; \mathbf{u}_t^\top \mathbf{X}^\top \mathbf{Y} \mathbf{v}_t \geq \rho_1(1 - 2\epsilon).$$

Alternating Least Squares - Costs

Costs:

- Setup time: $O(n(d_x^2 + d_y^2))$
- Iteration cost: $O(n(d_x + d_y))$
- #iterations: $\left\lceil \frac{\rho_1^2}{\rho_1^2 - \rho_2^2} \right\rceil \log\left(\frac{1}{\mu\epsilon}\right)$

Alternating Least Squares - Costs

Costs:

- Setup time: $O(n(d_x^2 + d_y^2))$
- Iteration cost: $O(n(d_x + d_y))$
- #iterations: $\left\lceil \frac{\rho_1^2}{\rho_1^2 - \rho_2^2} \right\rceil \log\left(\frac{1}{\mu\epsilon}\right)$

Good: very good iteration complexity.

Alternating Least Squares - Costs

Costs:

- Setup time: $O(n(d_x^2 + d_y^2))$
- Iteration cost: $O(n(d_x + d_y))$
- #iterations: $\left\lceil \frac{\rho_1^2}{\rho_1^2 - \rho_2^2} \right\rceil \log\left(\frac{1}{\mu\epsilon}\right)$

Good: very good iteration complexity.

Bad: Setup time is too large; as expensive as direct method.

Alternating Least Squares - Costs

Costs:

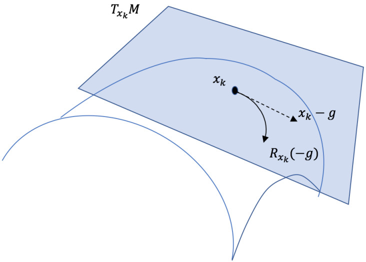
- Setup time: $O(n(d_x^2 + d_y^2))$
- Iteration cost: $O(n(d_x + d_y))$
- #iterations: $\left\lceil \frac{\rho_1^2}{\rho_1^2 - \rho_2^2} \right\rceil \log\left(\frac{1}{\mu\epsilon}\right)$

Good: very good iteration complexity.

Bad: Setup time is too large; as expensive as direct method.

Observation: ALS is actually Riemannian steepest descent!

Riemannian Optimization



Riemannian Steepest Descent

Problem:

$$\min f(\mathbf{x}) \text{ s.t. } \mathbf{x} \in \mathcal{M}$$

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

Iteration:

$$\mathbf{x}_{k+1} = R_{\mathbf{x}_k}(-\eta_k \mathbf{grad}_{(\mathcal{M},\mathbf{g})} f(\mathbf{x}))$$

$R.(\cdot)$ is a retraction defined on $\mathcal{M}$. $\mathbf{grad}_{(\mathcal{M},\mathbf{g})}\cdot$ is the Riemannian gradient. **Important:** it depends on the metric choice.

ALS is Riemannian Steepest Descent

Components	**Alternating Least Squares**
Function f to optimize	$f(\mathbf{u},\mathbf{v}) = -\mathbf{u}^\mathsf{T}\mathbf{X}^\mathsf{T}\mathbf{Y}\mathbf{v}$
Manifold domain $\mathcal{M}$	$\mathcal{M} = \left\{ \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} \text{s.t.}\, \mathbf{u}^\mathsf{T}\Sigma_{\mathsf{xx}}\mathbf{u} = 1,\ \mathbf{v}^\mathsf{T}\Sigma_{\mathsf{yy}}\mathbf{v} = 1 \right\}$ (i.e. product manifold of two generalized Stiefel manifolds)
Retraction	$R_{(\mathbf{u},\mathbf{v})}(\xi,\nu) = \begin{bmatrix} (\mathbf{u}+\xi)/\\|\mathbf{u}+\xi\\|_{\Sigma_{\mathsf{xx}}} \\ (\mathbf{v}+\nu)/\\|\mathbf{v}+\nu\\|_{\Sigma_{\mathsf{yy}}} \end{bmatrix}$
Metric g	$g\left(\begin{bmatrix} \xi_1 \\ \nu_1 \end{bmatrix}, \begin{bmatrix} \xi_2 \\ \nu_2 \end{bmatrix}\right) = \xi_1^\mathsf{T}\Sigma_{\mathsf{xx}}\xi_2 + \nu_1^\mathsf{T}\Sigma_{\mathsf{yy}}\nu_2$
Gradient $\mathbf{grad}_{(\mathcal{M},\mathbf{g})}f$	$\mathbf{grad}_{(\mathcal{M},\mathbf{g})}f(\mathbf{u},\mathbf{v}) = -\begin{bmatrix} \Sigma_{\mathsf{xx}}^{-1}\mathbf{X}^\mathsf{T}\mathbf{Y}\mathbf{v} - f(\mathbf{u},\mathbf{v})\mathbf{u} \\ \Sigma_{\mathsf{yy}}^{-1}\mathbf{Y}^\mathsf{T}\mathbf{X}\mathbf{u} - f(\mathbf{u},\mathbf{v})\mathbf{v} \end{bmatrix}$
Step size η_k	$\eta_k = -f(\mathbf{u}_k,\mathbf{v}_k)$

ALS is Riemannian Steepest Descent

Components	**Alternating Least Squares**
Function f to optimize	$f(\mathbf{u},\mathbf{v}) = -\mathbf{u}^\mathsf{T}\mathbf{X}^\mathsf{T}\mathbf{Y}\mathbf{v}$
Manifold domain $\mathcal{M}$	$\mathcal{M} = \left\{ \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} \text{ s.t. } \mathbf{u}^\mathsf{T}\Sigma_{xx}\mathbf{u} = 1,\ \mathbf{v}^\mathsf{T}\Sigma_{yy}\mathbf{v} = 1 \right\}$ (i.e. product manifold of two generalized Stiefel manifolds)
Retraction	$R_{(\mathbf{u},\mathbf{v})}(\xi,\nu) = \begin{bmatrix} (\mathbf{u}+\xi)/\lVert\mathbf{u}+\xi\rVert_{\Sigma_{xx}} \\ (\mathbf{v}+\nu)/\lVert\mathbf{v}+\nu\rVert_{\Sigma_{yy}} \end{bmatrix}$
Metric g	$g\left(\begin{bmatrix} \xi_1 \\ \nu_1 \end{bmatrix}, \begin{bmatrix} \xi_2 \\ \nu_2 \end{bmatrix}\right) = \xi_1^\mathsf{T}\Sigma_{xx}\xi_2 + \nu_1^\mathsf{T}\Sigma_{yy}\nu_2$
Gradient $\mathbf{grad}_{(\mathcal{M},\mathbf{g})}f$	$\mathbf{grad}_{(\mathcal{M},\mathbf{g})}f(\mathbf{u},\mathbf{v}) = -\begin{bmatrix} \Sigma_{xx}^{-1}\mathbf{X}^\mathsf{T}\mathbf{Y}\mathbf{v} - f(\mathbf{u},\mathbf{v})\mathbf{u} \\ \Sigma_{yy}^{-1}\mathbf{Y}^\mathsf{T}\mathbf{X}\mathbf{u} - f(\mathbf{u},\mathbf{v})\mathbf{v} \end{bmatrix}$
Step size η_k	$\eta_k = -f(\mathbf{u}_k,\mathbf{v}_k)$

ALS is Riemannian Steepest Descent

Components	Alternating Least Squares
Function f to optimize	$f(\mathbf{u},\mathbf{v}) = -\mathbf{u}^T\mathbf{X}^T\mathbf{Y}\mathbf{v}$
Manifold domain $\mathcal{M}$	$\mathcal{M} = \left\{ \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} \text{ s.t. } \mathbf{u}^T\Sigma_{xx}\mathbf{u} = 1,\ \mathbf{v}^T\Sigma_{yy}\mathbf{v} = 1 \right\}$ (i.e. product manifold of two generalized Stiefel manifolds)
Retraction	$R_{(\mathbf{u},\mathbf{v})}(\xi,\nu) = \begin{bmatrix} (\mathbf{u}+\xi)/\Vert\mathbf{u}+\xi\Vert_{\Sigma_{xx}} \\ (\mathbf{v}+\nu)/\Vert\mathbf{v}+\nu\Vert_{\Sigma_{yy}} \end{bmatrix}$
Metric g	$g\left(\begin{bmatrix} \xi_1 \\ \nu_1 \end{bmatrix}, \begin{bmatrix} \xi_2 \\ \nu_2 \end{bmatrix}\right) = \xi_1^T\Sigma_{xx}\xi_2 + \nu_1^T\Sigma_{yy}\nu_2$
Gradient $\mathbf{grad}_{(\mathcal{M},\mathbf{g})} f$	$\mathbf{grad}_{(\mathcal{M},\mathbf{g})} f(\mathbf{u},\mathbf{v}) = -\begin{bmatrix} \Sigma_{xx}^{-1}\mathbf{X}^T\mathbf{Y}\mathbf{v} - f(\mathbf{u},\mathbf{v})\mathbf{u} \\ \Sigma_{yy}^{-1}\mathbf{Y}^T\mathbf{X}\mathbf{u} - f(\mathbf{u},\mathbf{v})\mathbf{v} \end{bmatrix}$
Step size η_k	$\eta_k = -f(\mathbf{u}_k,\mathbf{v}_k)$

This metric is common, leads to provable convergence bounds, but leads to expensive setup time.

Riemannian Preconditioning (Mishra-Sepulchre '16): Change the Metric

Components	**Suggested Algorithm**
Function f to optimize	$f(\mathbf{u},\mathbf{v}) = -\mathbf{u}^T\mathbf{X}^T\mathbf{Y}\mathbf{v}$
Manifold domain $\mathcal{M}$	$\mathcal{M} = \left\{ \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} \text{ s.t. } \mathbf{u}^T\Sigma_{xx}\mathbf{u} = 1,\ \mathbf{v}^T\Sigma_{yy}\mathbf{v} = 1 \right\}$
Retraction	$R_{(\mathbf{u},\mathbf{v})}(\xi,\nu) = \begin{bmatrix} (\mathbf{u}+\xi)/\|\mathbf{u}+\xi\|_{\Sigma_{xx}} \\ (\mathbf{v}+\nu)/\|\mathbf{v}+\nu\|_{\Sigma_{yy}} \end{bmatrix}$
Metric g	$g\left(\begin{bmatrix} \xi_1 \\ \nu_1 \end{bmatrix}, \begin{bmatrix} \xi_2 \\ \nu_2 \end{bmatrix}\right) = \xi_1^T\mathbf{M}_{xx}\xi_2 + \nu_1^T\mathbf{M}_{yy}\nu_2$
Gradient $\mathbf{grad}_{(\mathcal{M},\mathbf{g})} f$	$\mathbf{grad}_{(\mathcal{M},\mathbf{g})} f(\mathbf{u},\mathbf{v}) =$ $- \begin{bmatrix} (\mathbf{I}_n - (\mathbf{u}^T\Sigma_{xx}\mathbf{M}_{xx}^{-1}\Sigma_{xx}\mathbf{u})^{-1}\mathbf{M}_{xx}^{-1}\Sigma_{xx}\mathbf{u}\mathbf{u}^T\Sigma_{xx})\mathbf{M}_{xx}^{-1}\mathbf{X}^T\mathbf{Y}\mathbf{v} \\ (\mathbf{I}_n - (\mathbf{v}^T\Sigma_{yy}\mathbf{M}_{yy}^{-1}\Sigma_{yy}\mathbf{v})^{-1}\mathbf{M}_{yy}^{-1}\Sigma_{yy}\mathbf{v}\mathbf{v}^T\Sigma_{yy})\mathbf{M}_{yy}^{-1}\mathbf{Y}^T\mathbf{X}\mathbf{u} \end{bmatrix}$
Step size η_k	use line search or Riemannian CG

Sketching Based Preconditioning Strategies

1. Subspace Embedding Preconditioners: generate a sketch transform (SRFT, CountSketch, etc.) $\mathbf{S}$ and factor

$$[\mathbf{Q}_\mathbf{x}, \mathbf{R}_\mathbf{x}] = \mathrm{qr}(\mathbf{S}\mathbf{X}),\ [\mathbf{Q}_\mathbf{y}, \mathbf{R}_\mathbf{y}] = \mathrm{qr}(\mathbf{S}\mathbf{Y})$$

Implicitly define

$$\mathbf{M}_\mathbf{xx} = \mathbf{R}_\mathbf{x}^\mathsf{T}\mathbf{R}_\mathbf{x},\ \mathbf{M}_\mathbf{yy} = \mathbf{R}_\mathbf{y}^\mathsf{T}\mathbf{R}_\mathbf{y}$$

This is the strategy used in randomized least squares solvers (e.g. Blendenpik). Theory for bounding the condition number (with respect to number of rows) is well understood.

Warm-start: This strategy also allows for an easy warm-start - solve CCA on $(\mathbf{S}\mathbf{X}, \mathbf{S}\mathbf{Y})$ and use as starting vectors.

Sketching Based Preconditioning Strategies

2. Approximate Dominant Subspace Preconditioning (Gonen et al. 2016): approximate the k dominant right singular vectors $\mathbf{v}_1, \ldots, \mathbf{v}_k$ of $\mathbf{X}$ and corresponding singular values $\sigma_1, \ldots, \sigma_k$. Then

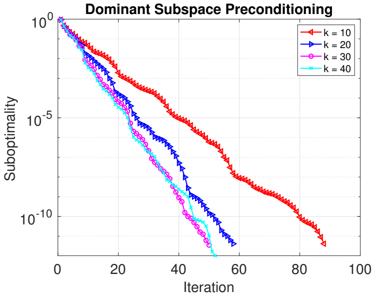
$$\mathbf{M}_{\mathbf{xx}} = \sum_{i=1}^{k} (\sigma_i^2 - \sigma_k^2) \mathbf{v}_i \mathbf{v}_i^T + (\lambda + \sigma_k^2) \mathbf{I}_{d_x}$$

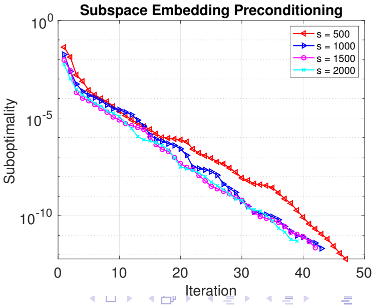
Repeat for $\mathbf{Y}$. Can efficiently multiply by a vector, and apply inverse.

Only for $\lambda > 0$. No warm-start. Very efficient preconditioners (low iteration complexity).

Preliminary Experimental Results

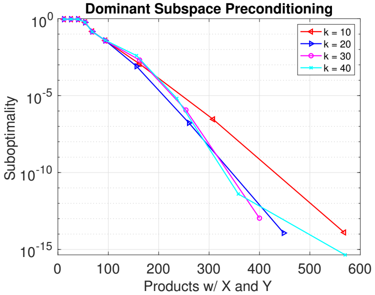
- MNIST dataset $(60,000 \times 784)$ split into two halves.
- Plotting suboptimality of objective: $|\sigma_1 - \mathbf{u}_k^T \Sigma_{\mathbf{xy}} \mathbf{v}_k| / \sigma_1$.
- Use warm start for subspace embedding (right graph).
- Riemannian CG (via Manopt).
- **Baselines -**
  - **Identity preconditioners - 205 iterations.**
  - **Exact inverses ("best") - 47 iterations.**

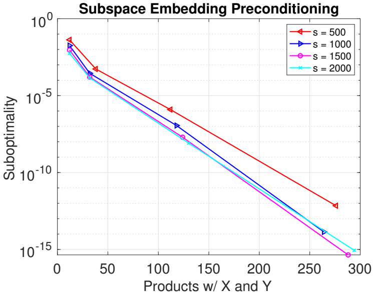




Second Order Methods

- We calculated the Riemannian Hessian (omitted - rather long expression).
- Allows the use of a Riemannian Trust Region Method.
- Very few iterations, but iterations have varying costs.
- x-axis is the number of matvecs with $\mathbf{X}$ and $\mathbf{Y}$.
- Less matvec products than Riemannian CG.





Riemannian Preconditioning (Mishra-Sepulchre '16): Change the Metric

Components	Suggested Algorithm
Function f to optimize	$f(\mathbf{u},\mathbf{v}) = -\mathbf{u}^\mathsf{T}\mathbf{X}^\mathsf{T}\mathbf{Y}\mathbf{v}$
Manifold domain $\mathcal{M}$	$\mathcal{M} = \left\{ \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} \text{ s.t. } \mathbf{u}^\mathsf{T}\Sigma_{xx}\mathbf{u} = 1,\ \mathbf{v}^\mathsf{T}\Sigma_{yy}\mathbf{v} = 1 \right\}$
Retraction	$R_{(\mathbf{u},\mathbf{v})}(\xi,\nu) = \begin{bmatrix} (\mathbf{u}+\xi)/\Vert\mathbf{u}+\xi\Vert_{\Sigma_{xx}} \\ (\mathbf{v}+\nu)/\Vert\mathbf{v}+\nu\Vert_{\Sigma_{yy}} \end{bmatrix}$
Metric g	$g\left(\begin{bmatrix} \xi_1 \\ \nu_1 \end{bmatrix}, \begin{bmatrix} \xi_2 \\ \nu_2 \end{bmatrix}\right) = \xi_1^\mathsf{T}\mathrm{M}_{xx}\xi_2 + \nu_1^\mathsf{T}\mathrm{M}_{yy}\nu_2$
Gradient $\mathbf{grad}_{(\mathcal{M},\mathbf{g})}f$	$\mathbf{grad}_{(\mathcal{M},\mathbf{g})}f(\mathbf{u},\mathbf{v}) = - \begin{bmatrix} (\mathbf{I}_n - (\mathbf{u}^\mathsf{T}\Sigma_{xx}\mathrm{M}_{xx}^{-1}\Sigma_{xx}\mathbf{u})^{-1}\mathrm{M}_{xx}^{-1}\Sigma_{xx}\mathbf{u}\mathbf{u}^\mathsf{T}\Sigma_{xx})\mathrm{M}_{xx}^{-1}\mathbf{X}^\mathsf{T}\mathbf{Y}\mathbf{v} \\ (\mathbf{I}_n - (\mathbf{v}^\mathsf{T}\Sigma_{yy}\mathrm{M}_{yy}^{-1}\Sigma_{yy}\mathbf{v})^{-1}\mathrm{M}_{yy}^{-1}\Sigma_{yy}\mathbf{v}\mathbf{v}^\mathsf{T}\Sigma_{yy})\mathrm{M}_{yy}^{-1}\mathbf{Y}^\mathsf{T}\mathbf{X}\mathbf{u} \end{bmatrix}$
Step size η_k	use line search or Riemannian CG

Riemannian Preconditioning (Mishra-Sepulchre '16): Change the Metric

Components	Suggested Algorithm
Function f to optimize	$f(\mathbf{u},\mathbf{v}) = -\mathbf{u}^T\mathbf{X}^T\mathbf{Y}\mathbf{v}$
Manifold domain $\mathcal{M}$	$\mathcal{M} = \left\{ \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} \text{ s.t. } \mathbf{u}^T\Sigma_{xx}\mathbf{u} = 1,\ \mathbf{v}^T\Sigma_{yy}\mathbf{v} = 1 \right\}$
Retraction	$R_{(\mathbf{u},\mathbf{v})}(\xi,\nu) = \begin{bmatrix} (\mathbf{u}+\xi)/\Vert\mathbf{u}+\xi\Vert_{\Sigma_{xx}} \\ (\mathbf{v}+\nu)/\Vert\mathbf{v}+\nu\Vert_{\Sigma_{yy}} \end{bmatrix}$
Metric g	$g\left(\begin{bmatrix} \xi_1 \\ \nu_1 \end{bmatrix}, \begin{bmatrix} \xi_2 \\ \nu_2 \end{bmatrix} \right) = \xi_1^T\mathbf{M}_{xx}\xi_2 + \nu_1^T\mathbf{M}_{yy}\nu_2$
Gradient $\mathbf{grad}_{(\mathcal{M},\mathbf{g})}f$	$\mathbf{grad}_{(\mathcal{M},\mathbf{g})}f(\mathbf{u},\mathbf{v}) =$ $-\begin{bmatrix} (\mathbf{I}_n - (\mathbf{u}^T\Sigma_{xx}\mathbf{M}_{xx}^{-1}\Sigma_{xx}\mathbf{u})^{-1}\mathbf{M}_{xx}^{-1}\Sigma_{xx}\mathbf{u}\mathbf{u}^T\Sigma_{xx})\mathbf{M}_{xx}^{-1}\mathbf{X}^T\mathbf{Y}\mathbf{v} \\ (\mathbf{I}_n - (\mathbf{v}^T\Sigma_{yy}\mathbf{M}_{yy}^{-1}\Sigma_{yy}\mathbf{v})^{-1}\mathbf{M}_{yy}^{-1}\Sigma_{yy}\mathbf{v}\mathbf{v}^T\Sigma_{yy})\mathbf{M}_{yy}^{-1}\mathbf{Y}^T\mathbf{X}\mathbf{u} \end{bmatrix}$
Step size η_k	use line search or Riemannian CG

Q: What constitutes a "good" $\mathbf{M}_{xx}$ and $\mathbf{M}_{yy}$?

Fixed Step Gradient Descent

Definitions

$f : \mathcal{M} \to \mathbb{R}$ has **Lipschitz-type continuous gradient** with constant L on $\mathcal{C} \subseteq \mathcal{M}$ w/ respect to R if for every $\mathbf{x} \in \mathcal{C}$, $\eta \in T_{\mathbf{x}}\mathcal{M}$

$$\left| f(R_{\mathbf{x}}(\eta)) - f(\mathbf{x}) - g(\eta, \mathbf{grad}_{(\mathcal{M},\mathbf{g})} f(\mathbf{x})) \right| \leq \frac{L}{2} g(\eta, \eta) .$$

It is τ-**gradient dominated** on $\mathcal{C}$ if for every $\mathbf{x} \in \mathcal{C}$

$$f(\mathbf{x}) - f(\mathbf{x}^\star) \leq \tau \cdot g(\mathbf{grad}_{(\mathcal{M},\mathbf{g})} f(\mathbf{x}), \mathbf{grad}_{(\mathcal{M},\mathbf{g})} f(\mathbf{x}))$$

Fact

Assume the above hold, and consider $\mathbf{x}_{k+1} = R_{\mathbf{x}_k}(-\frac{1}{L}\mathbf{grad}_{(\mathcal{M},\mathbf{g})} f(x_k))$. *Assume all iterations belong to* $\mathcal{C}$. *Then*

$$f(\mathbf{x}_{k+1}) - f(\mathbf{x}^\star) \leq \left(1 - \frac{1}{2L\tau}\right)^k (f(\mathbf{x}_0) - f(\mathbf{x}^\star))$$

Example: Generalized Eigenvalue Computation

Lemma

Assume $\mathbf{A}$ *and* $\mathbf{B}$ *are PSD. Consider* $f(\mathbf{x}) = -\frac{1}{2}\mathbf{x}^\top \mathbf{A}\mathbf{x}$, *on the manifold* $\mathbf{x}^\top \mathbf{B}\mathbf{x} = 1$ *with the natural metric (*$\mathbf{B}$ *inner product). Let* $\lambda_1 \geq \cdots \geq \lambda_{\min}$ *be the singular values of* $\mathbf{B}^{-1/2}\mathbf{A}\mathbf{B}^{-1/2}$. ***Let*** $\delta \equiv \lambda_1 - \lambda_2$ ***(the eigengap).*** *Then:*

1. f *has* ***Lipschitz-type continuous gradient with*** $L = \lambda_1$.
2. f *is* $\min\left(\frac{1}{2\epsilon^2\delta}, \frac{1}{\delta}\right)$*-**gradient dominated** inside (Corollary of a Theorem of Sra et al. 2016)*

$$\mathcal{C} = \left\{\mathbf{x} \; s.t. \; \mathbf{x}^\top \mathbf{B}\mathbf{x}^\star \geq \epsilon\right\}$$

CCA: The effect of preconditioning

Lemma

$$\mathcal{M} = \left\{ \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} \; s.t. \; \mathbf{u}^{\mathsf{T}} \Sigma_{\mathbf{xx}} \mathbf{u} = 1, \; \mathbf{v}^{\mathsf{T}} \Sigma_{\mathbf{yy}} \mathbf{v} = 1 \right\}$$

$$g_1 \left(\begin{bmatrix} \xi_1 \\ \nu_1 \end{bmatrix}, \begin{bmatrix} \xi_2 \\ \nu_2 \end{bmatrix} \right) = \xi_1^{\mathsf{T}} \Sigma_{\mathsf{xx}} \xi_2 + \nu_1^{\mathsf{T}} \Sigma_{\mathsf{yy}} \nu_2$$

$$g_2 \left(\begin{bmatrix} \xi_1 \\ \nu_1 \end{bmatrix}, \begin{bmatrix} \xi_2 \\ \nu_2 \end{bmatrix} \right) = \xi_1^{\mathsf{T}} \mathrm{M}_{\mathsf{xx}} \xi_2 + \nu_1^{\mathsf{T}} \mathrm{M}_{\mathsf{yy}} \nu_2$$

- *Lipschitz-type continuous gradient with constant L w/ $g_2 \implies$ Lipschitz-type $L \cdot \min(\lambda_{\min}(\mathrm{M}_{\mathsf{xx}}, \Sigma_{\mathsf{xx}}), \lambda_{\min}(\mathrm{M}_{\mathsf{yy}}, \Sigma_{\mathsf{yy}}))^{-1}$ w/ g_2.*
- *τ-gradient dominated w/ $g_1 \implies$ $\tau \cdot \max(\lambda_{\max}(\mathrm{M}_{\mathsf{xx}}, \Sigma_{\mathsf{xx}}), \lambda_{\max}(\mathrm{M}_{\mathsf{yy}}, \Sigma_{\mathsf{yy}}))$-gradient dominated w/ g_2.*

In short: we can expect a factor of $\kappa\left(\mathbf{diag}\left(\mathrm{M}_{\mathbf{xx}}, \mathrm{M}_{\mathbf{yy}}\right), \mathbf{diag}\left(\Sigma_{\mathbf{xx}}, \Sigma_{\mathbf{yy}}\right)\right)$ increase in #iterations.

Conclusions and Future Work

1. RandNLA achieves high accuracy when used for preconditioning.
2. High accuracy "beyond regression" requires preconditioned methods
3. Riemannian optimization is well suited for this.
4. Can be precondiioned by changing metric (Riemannian preconditioning).
5. i.e. for quadratic constraints (focused on CCA in this talk).
6. Still work in progress.