Scalable Second-order Riemannian Optimization for K-means Clustering

Peng Xu* — pengxu1@illinois.edu — *University of Illinois Urbana-Champaign*

Chun-Ying Hou* — cyhou2@illinois.edu — *University of Illinois Urbana-Champaign*

Xiaohui Chen — xiaohuic@usc.edu — *University of Southern California*

Richard Y. Zhang — ryz@illinois.edu — *University of Illinois Urbana-Champaign*

Our Contributions

- Interpret the K-means clustering problem as **a smooth, unconstrained optimization problem on a Riemannian product manifold**.
- We open possibilities for **global convergence guarantees**.
- Second-order Riemannian algorithms can be implemented with **linear per-iteration cost**.
- Our algorithm computes ε second-order optimal solutions in $n \cdot \varepsilon^{-3/2} \cdot \mathrm{poly}(r, d)$ time.

Background: K-means Clustering & SDP

- K-means: partition n data points $X_1, \ldots, X_n \in \mathbb{R}^d$ into K disjoint clusters $G_1, \ldots, G_K$, by minimizing the distance between cluster points and cluster centroid.
- However, solving the original K-means formulation is worst-case NP-hard; Requires heuristics (e.g. Lloyd's algorithm) or relaxation to solve.
- A popular semi-definite programming (SDP) based relaxation [3] reads:

$$\begin{aligned} \max_{Z \in \mathbb{R}^{n \times n}} \quad & \langle XX^\top, Z \rangle \\ \text{s.t.} \quad & Z\mathbf{1}_n = \mathbf{1}_n, \\ & \mathrm{tr}(Z) = K, \\ & Z \succeq 0, \\ & Z \geq 0. \end{aligned} \tag{1}$$

- Consider the setting where data are generated by a Gaussian mixture model (GMM), i.e.

$$X_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(\mu_k, \sigma^2 I_d), \qquad i \in G_k^*, \tag{2}$$

- It was shown in [2] that, if the centroids are sufficiently far away, (1) solves the K-means problem with high probability.

Background: Riemannian Cubic-regularized Newton

- Let $\mathcal{M}$ be a Riemannian manifold. To solve $\min_p f(p)$ s.t. $p \in \mathcal{M}$. One can iteratively solve the cubic-regularized subproblem [4]:

$$\min_{Ap=0} g^\top p + \frac{1}{2} p^\top H p + \frac{L}{6}\|p\|^3. \tag{3}$$

- g and H denote the vectorized Riemannian gradient and Hessian, and A implements the tangent space constraint $p \in \mathrm{Tan}\,\mathcal{M}$.
- For fixed L, (3) can be solved by finding an appropriate value $\lambda > 0$ for

$$\begin{bmatrix} H + \lambda I & A^\top \\ A & \mathbf{0} \end{bmatrix} \begin{bmatrix} p \\ q \end{bmatrix} = \begin{bmatrix} -g \\ \mathbf{0} \end{bmatrix}.$$

Links

Paper

Code

References

[1] Samuel Burer and Renato D. C. Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming*, 95(2):329–357, 2003.

[2] Xiaohui Chen and Yun Yang. Cutoff for exact recovery of gaussian mixture models. *IEEE Transactions on Information Theory*, 67(6):4223–4238, 2021.

[3] Jiming Peng and Yu Wei. Approximating k-means-type clustering via semidefinite programming. *SIAM Journal on Optimization*, 18(1):186–205, 2007.

[4] Junyu Zhang and Shuzhong Zhang. A cubic regularized newton's method over riemannian manifolds, 2018. arXiv:1805.05565.

[5] Yubo Zhuang, Xiaohui Chen, Yun Yang, and Richard Y. Zhang. Statistically Optimal K-means Clustering via Nonnegative Low-rank Semidefinite Programming. In *The Twelfth International Conference on Learning Representations*, 2024.

Reformulation

- Solving (1) is not scalable since it has $O(n^2)$ variables.
- Following [1], we factor $Z = UU^\top$ into its $n \times r$ factor matrix U for rank parameter $r \geq K$. This gives

$$\begin{aligned} \max_{U} \quad & \langle XX^\top, UU^\top \rangle + \mu \mathbf{1}_n^\top \log(U) \mathbf{1}_r \\ \text{s.t.} \quad & UU^\top \mathbf{1}_n = \mathbf{1}_n, \\ & \mathrm{tr}(UU^\top) = K. \end{aligned} \tag{4}$$

- (4) can be seen is a restriction to (1). Under the exact-recovery separation conditions, (4) is tight, meaning solving (4) is equivalent to solving K-means.
- The constraint set

$$\mathcal{M} \coloneqq \mathcal{M}_r = \left\{ U \in \mathbb{R}^{n \times r} : UU^\top \mathbf{1}_n = \mathbf{1}_n,\ \mathrm{tr}(UU^\top) = K \right\}, \tag{5}$$

  is a manifold.
- Solving (4) on $\mathcal{M}$ is still not scalable, due to the lack of an efficient retraction operator, which must be called at every iteration to keep iterates feasible.

Product Manifold

- We reformulate (4) by decomposing the variable U.
- Consider the product manifold $\widetilde{\mathcal{M}} = \mathcal{V} \times \mathcal{O}_r$, where

$$\mathcal{V} = \left\{ V \in \mathbb{R}^{n \times (r-1)} : \mathbf{1}_n^\top V = 0,\ \mathrm{tr}(VV^\top) = K - 1 \right\}$$

  and

$$\mathcal{O}_r = \left\{ Q \in \mathbb{R}^{r \times r} : QQ^\top = I_r \right\}.$$

- Let $\varphi \colon \widetilde{\mathcal{M}} \to \mathcal{M}$ be defined by

$$\varphi(V, Q) \coloneqq \begin{bmatrix} \hat{\mathbf{1}}_n & V \end{bmatrix} Q,$$

  where $\hat{\mathbf{1}}_n \coloneqq n^{-1/2}\mathbf{1}_n$. Then $\mathcal{M} = \varphi(\widetilde{\mathcal{M}})$. Moreover, φ is a submersion.
- Hence, instead of solving (4), we equivalently solve

$$\begin{aligned} \max_{V,Q} \quad & \langle XX^\top, VV^\top \rangle + \mu \mathbf{1}_n^\top \log\big(\varphi(V,Q)\big) \mathbf{1}_r \\ \text{s.t.} \quad & \mathbf{1}_n^\top V = 0, \\ & \mathrm{tr}(VV^\top) = K - 1, \\ & QQ^\top = I_r. \end{aligned} \tag{6}$$

- The product manifold $\widetilde{\mathcal{M}}$ admits a simple second-order retraction via its Euclidean projection. That is,

$$\mathrm{R}_{(V,Q)}(\dot{V}, \dot{Q}) = \left[\Pi_{\mathcal{V}}(V + \dot{V}) \ \Pi_{\mathcal{O}_r}(Q + \dot{Q}) \right],$$

  where

$$\Pi_{\mathcal{V}}(V) = \sqrt{K-1} \frac{V - n^{-1}\mathbf{1}_n\mathbf{1}_n^\top V}{\|V - n^{-1}\mathbf{1}_n\mathbf{1}_n^\top V\|}, \qquad \Pi_{\mathcal{O}_r}(Q) = (QQ^\top)^{-1/2} Q.$$

- The retraction above costs $O(nr + r^3)$ time to evaluate.

Efficient Implementation

- We solve (6) by Riemannian cubic-regularized method, for which we show that the Newton subproblem can be solved in linear time.
- After vectorizing the objective and constraints, solving (6) requires us to repeatedly solve a complex linear system.
- After careful analysis, we arrive at needing to solve

$$\left[\begin{array}{c|c|c} D_{11} & K_{12} & B \\ \hline K_{21} & K_{22} & \mathbf{0} \\ \hline B^\top & \mathbf{0} & I \end{array}\right] \begin{bmatrix} \dot{v} \\ \dot{q} \\ z \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \mathbf{0} \end{bmatrix},$$

  where D_{11} has low-rank structure allowing for efficient computation.
- In total, it takes $O\big(nr^3(d+r) + r^6 + r^3 d^3\big)$ to solve the system.

Simulation: Convergence Behavior

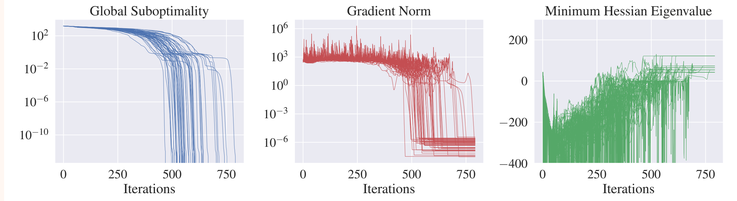

Figure 2. **Local convergence to second-order critical points yields global optimality (for GMMs).** This provides strong numerical evidence that near-second-order critical points are near-globally optimal.

Simulation: Comparison with Other Clustering Methods

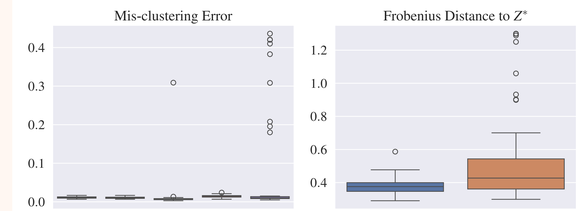

Figure 3. **Real-world benchmark on CyTOF data ($n = 1800$).** We compared our method to NLR [5], the previous state-of-the-art, as well as classical approaches including Spectral Clustering, Non-negative Matrix Factorization, and K-means++. Our method and NLR achieve the most consistently accurate clustering, with the smallest variance and the fewest outliers (left), but we outperform NLR in ground truth recovery (right).

Simulation: Computational Cost

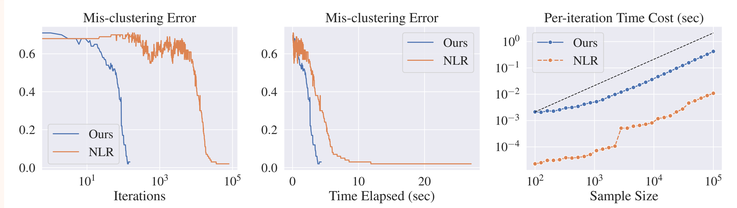

Figure 4. **Comparison with NLR on GMM ($n = 100$).** Our second-order method reaches optimality in 152 iterations, while NLR needs 80k. Even though each second-order iteration costs ≈25–100 NLR steps, the total runtime is still two to four times shorter.

Simulation: Computational Cost

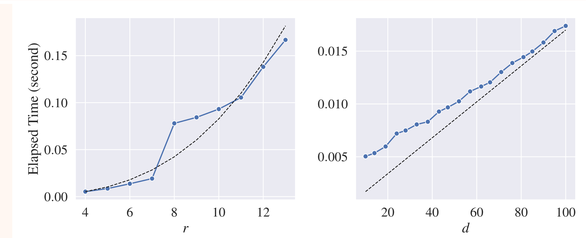

Figure 5. **Average per-iteration runtime versus r and d ($n = 1000$).** The measured runtimes exhibit approximately $O(r^3)$ and $O(d)$ scaling (dashed lines). Because n dominates as a leading constant, the contributions of the r^6 and $r^3 d^3$ terms are not noticeable until r and d are extremely large.