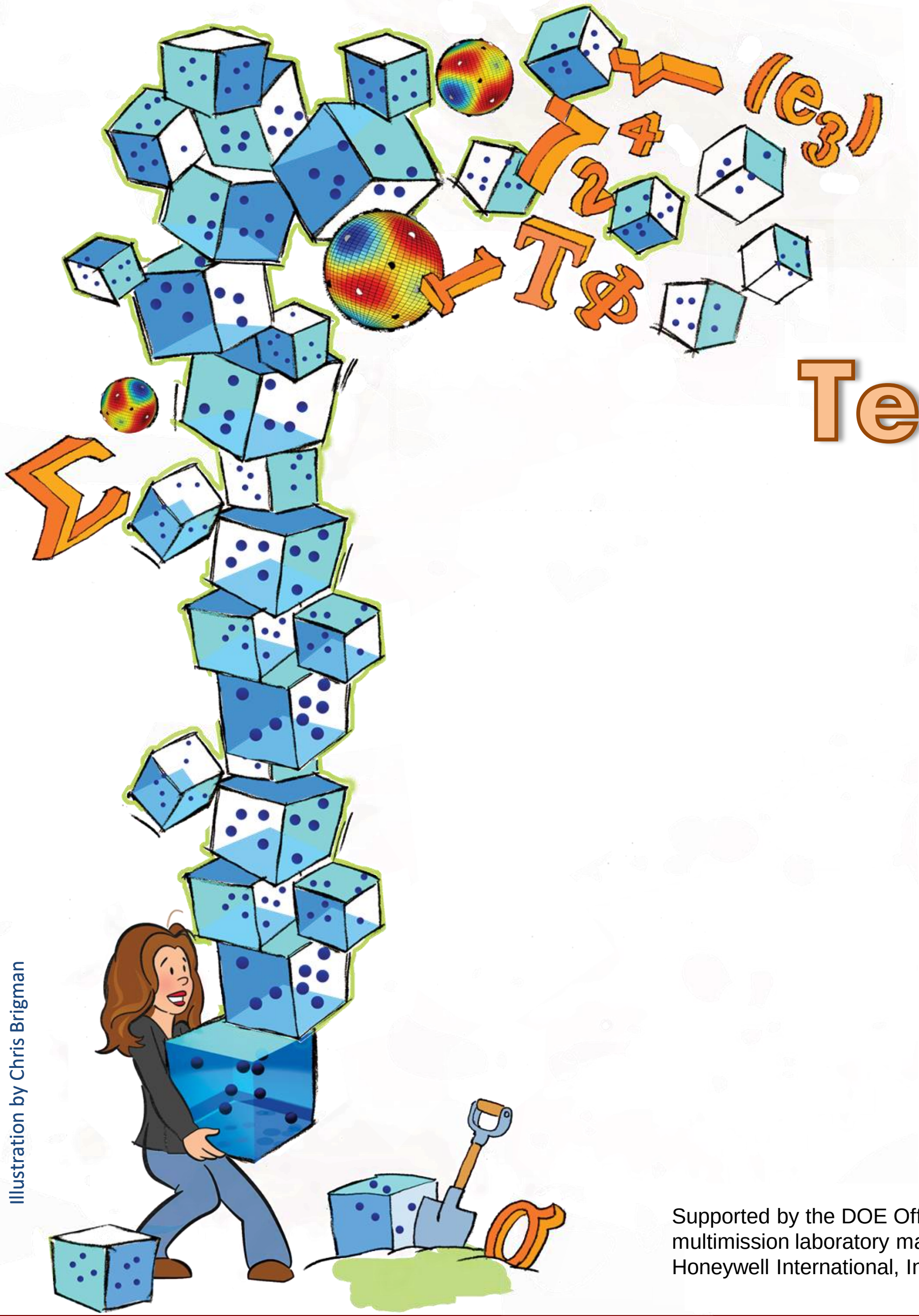
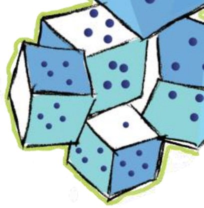


Illustration by Chris Brigman



Sandia National Laboratories

Tensors and Gaussian Mixture Models

Tamara G. Kolda

Sandia National Labs, Livermore, CA

www.kolda.net

Joint work with

Samantha Sherman

University of Notre Dame, South Bend, IN

Sam Sherman

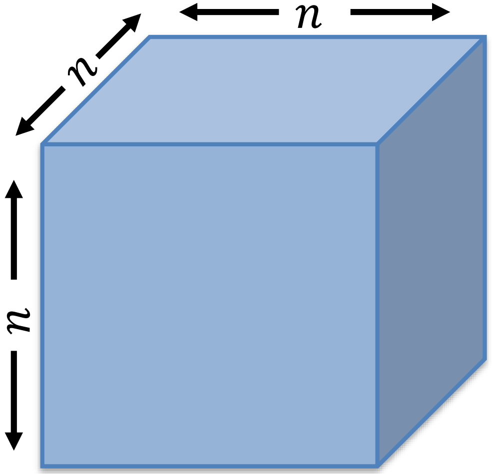
Notre Dame

Supported by the DOE Office of Science Advanced Scientific Computing Research (ASCR) Applied Mathematics. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

U.S. DEPARTMENT OF ENERGY

Focus on Symmetric Tensors: Entries Invariant Under Permutation of Indices

A tensor is <u>symmetric</u> if its entries are invariant under permutation of the indices



For d-way tensor, of dimension n, number of unique entries is:

$$\binom{n+d-1}{d} \approx \frac{n^d}{d!}$$

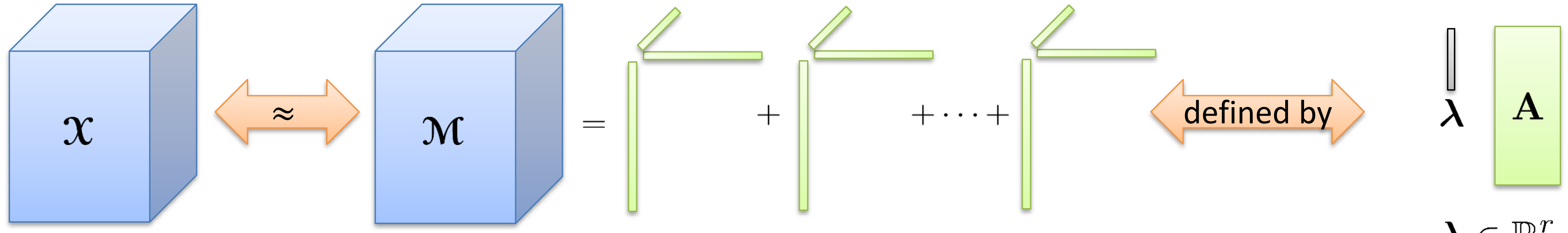
Example 1.2 from Nie (2014)
$3 \times 3 \times 3$ symmetric tensor (10 distinct entries)

$$\mathcal{X} = \left(\begin{array}{ccc|ccc|ccc} 7 & -3 & 9 & -3 & 13 & 20 & 9 & 20 & 19 \\ -3 & 13 & 20 & 13 & -27 & 6 & 20 & 6 & 6 \\ 9 & 20 & 19 & 20 & 6 & 6 & 19 & 6 & 45 \end{array} \right)$$

$$\begin{aligned} x(1,1,1) &= 7 & x(1,3,3) &= 19 \\ x(1,1,2) &= -3 & x(2,2,2) &= -27 \\ x(1,1,3) &= 9 & x(2,2,3) &= 6 \\ x(1,2,2) &= 13 & x(2,3,3) &= 6 \\ x(1,2,3) &= 20 & x(3,3,3) &= 45 \end{aligned}$$

Symmetric CP Tensor Decomposition Has Single Factor Matrix

Symmetric Data	Symmetric CP Model	Sum of r Symmetric Outer Product Tensors	Single Factor Matrix



$\mathcal{X} \in \mathbb{R}^{\underbrace{n\times n\times\cdots\times n}_{\text{order } d}}$

$\mathcal{M} \in \mathbb{R}^{n\times n\times\cdots\times n}$

$\boldsymbol{\lambda} \in \mathbb{R}^r$

$\mathbf{A} \in \mathbb{R}^{n\times r}$

Model Rank

$$m_{i_1 i_2 \ldots i_d} = \sum_{j=1}^{r} \lambda_j \prod_{k=1}^{d} a(i_k, j)$$

$$\mathcal{M} = \sum_{j=1}^{r} \lambda_j \, \mathbf{a}_j^{\otimes d} \in \mathbb{R}^{n\times n\times\cdots\times n}$$

Symmetric Tensor Rank & Decomposition

Example 1.2 from Nie (2014)
$3 \times 3 \times 3$ symmetric tensor (10 distinct entries)

$$\mathcal{X} = \left(\begin{array}{ccc|ccc|ccc} 7 & -3 & 9 & -3 & 13 & 20 & 9 & 20 & 19 \\ -3 & 13 & 20 & 13 & -27 & 6 & 20 & 6 & 6 \\ 9 & 20 & 19 & 20 & 6 & 6 & 19 & 6 & 45 \end{array} \right)$$

$$\operatorname{rank}(\mathcal{X}) = \min \left\{ r \mid \mathcal{X} = \mathbf{a}_1^{\otimes d} + \cdots + \mathbf{a}_r^{\otimes d} \right\}$$

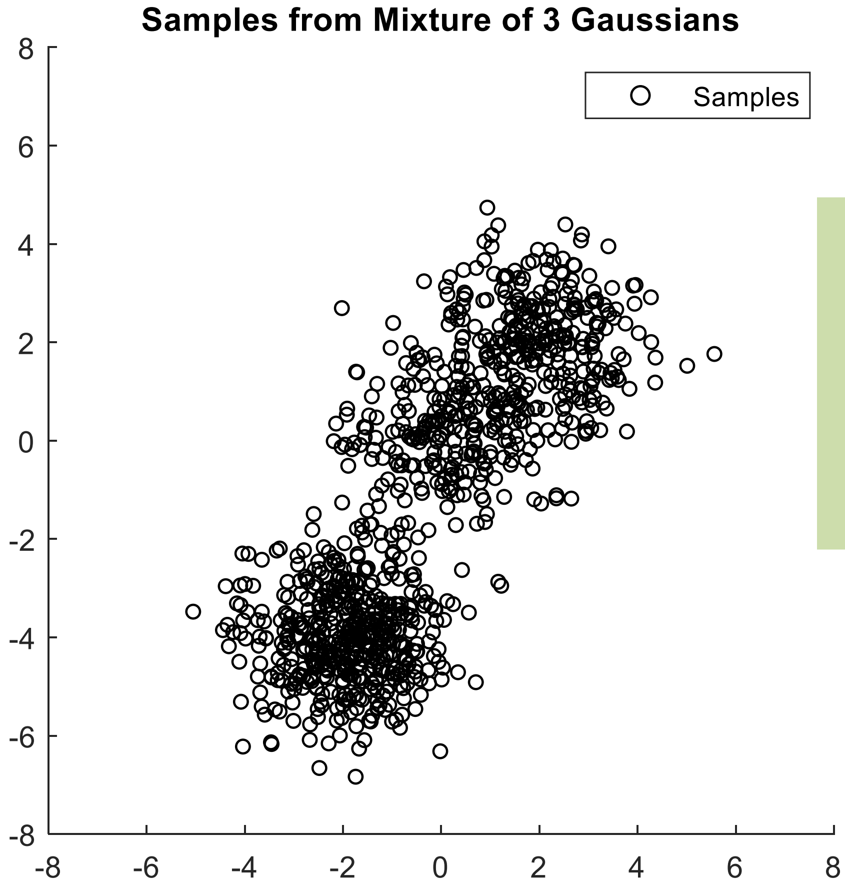
Rank decomposition

$$\mathcal{X} = 2 \cdot \begin{bmatrix} 1 \\ -2 \\ -1 \end{bmatrix}^{\otimes 3} + 5 \cdot \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix}^{\otimes 3} - \begin{bmatrix} 1 \\ 2 \\ -2 \end{bmatrix}^{\otimes 3}$$

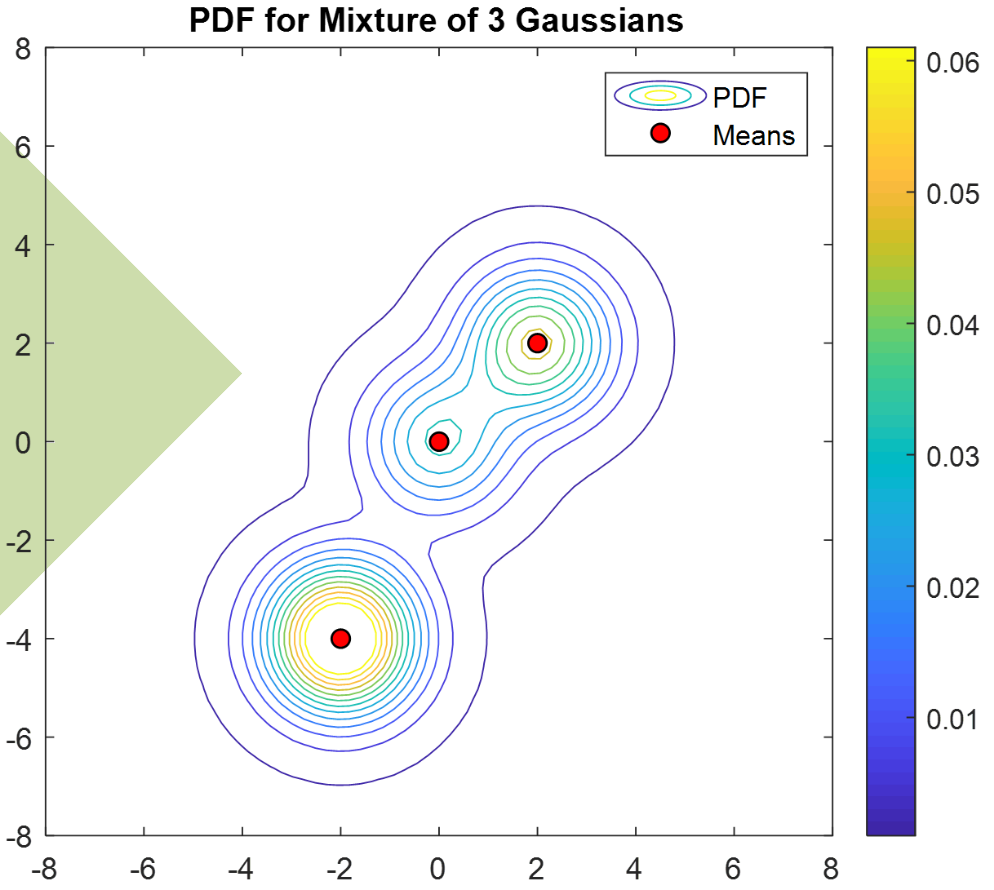
- Symmetric tensor rank
  - For any given tensor, NP-hard to compute its rank (Hillar & Lim, 2013)
  - Typical rank known over $\mathbb{C}$ (Comon, Golub, Lim, Mourraine, 2008)
  - In practice, trial and error!
- Symmetric tensor decomposition
  - Waring decomposition (Landsberg, 2012; Oeding & Ottaviani, 2013)
  - Gröbner bases algebraic methods or numerical root-finding method (Nie, 2014)
  - Direct optimization formulation (Kolda, 2015)
  - Subspace power method (Kileel & Pereira, 2019)

Moment Tensors Arise in Inference of Gaussian Mixture Models (GMMs)

For ease of illustration, we focus on $n = 2$ dimensions.
Generally interested in much higher dimensions, i.e, $n = 500$!



Given just the samples (point cloud), can we recover the means?

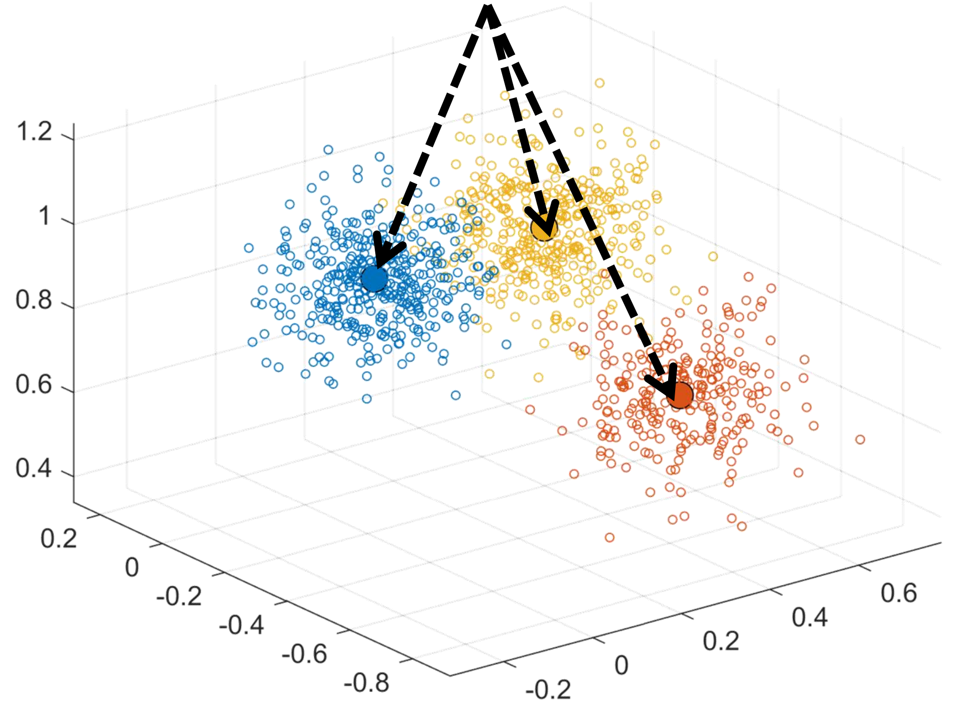


Machine Learning Motivation: Observations from Unknown Mixture of Gaussians

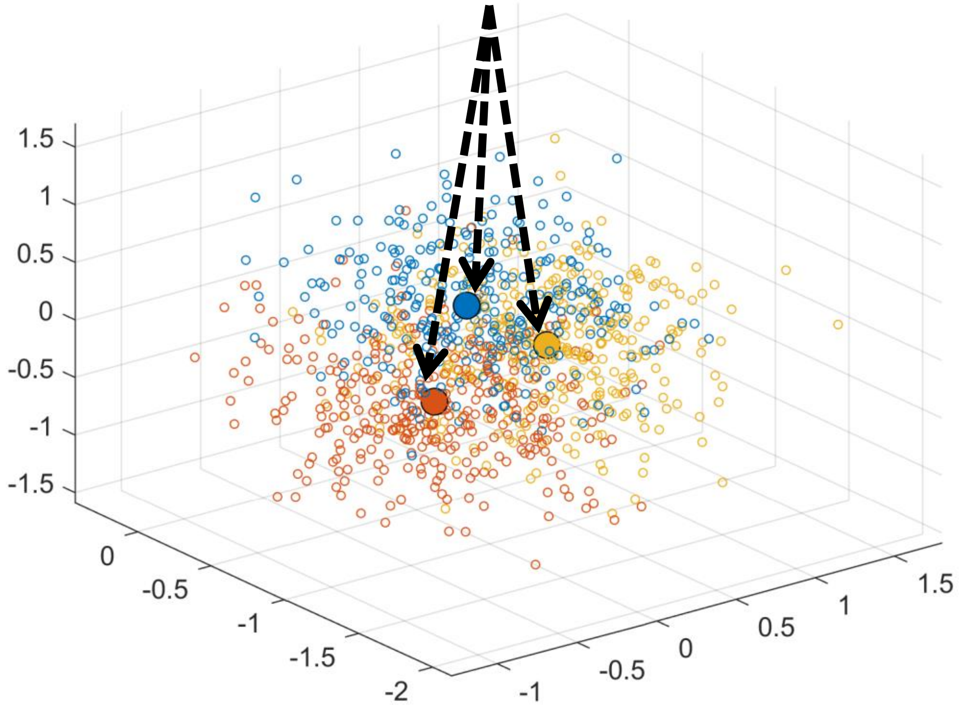
We observe p random vectors of length n coming from a mixture of r Gaussian distributions.

Can we recover the means of the Gaussians?

Easy: Means Well Separated



Hard: Means Close Together



For these pictures: $p = 1000, n = 3, r = 3$. Means shown as filled in larger circles. Samples as open circles.

We care about larger values of n!

Moment Structure for Spherical GMMs Corresponds to CP Model

Data Model: $V \sim \mathcal{N}(\boldsymbol{\mu}_\xi, \sigma^2 \mathbf{I}), \quad \xi \sim \textsc{Multi}(w_1, \ldots, w_r)$

Multivariate Normal

Probability to select jth center is w_j

3rd-order Moment: $\boxed{\mathbb{E}[V^{\otimes 3}]} + O(\sigma^2) = \boxed{\sum_{j=1}^{r} w_j \boldsymbol{\mu}_j^{\otimes 3}}$

Can also do higher - order moments

Calculate empirically from data

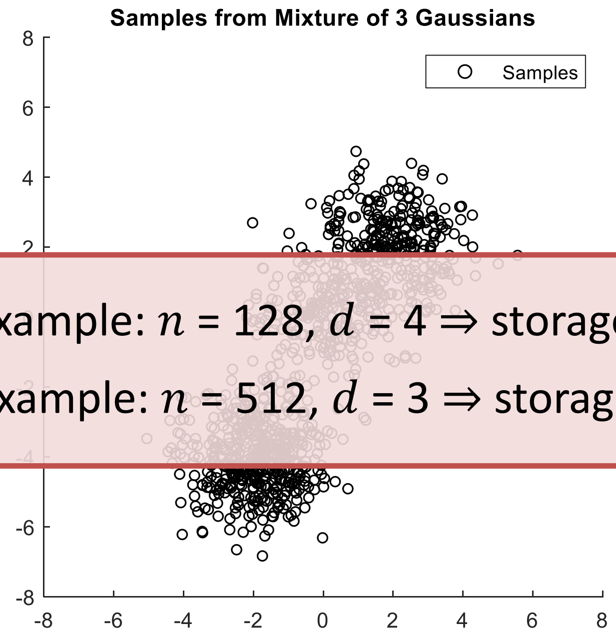
CP-like Model

$$\mathcal{X} = \frac{1}{p} \sum_{\ell=1}^{p} \mathbf{v}_\ell^{\otimes 3}$$

$$\mathcal{M} = \sum_{j=1}^{r} \lambda_j \, \mathbf{a}_j^{\otimes 3}$$

Bottlenecks:
$O(pn^d)$ to compute,
$O(n^d)$ to store

Hsu and Kakade, 2013



Example: n = 128, d = 4 ⇒ storage = 2 GB

Example: n = 512, d = 3 ⇒ storage = 1 GB

Simplifying assumptions for this work

$$\|\boldsymbol{\mu}_j\|_2 = 1 \; \forall j \in [r]$$

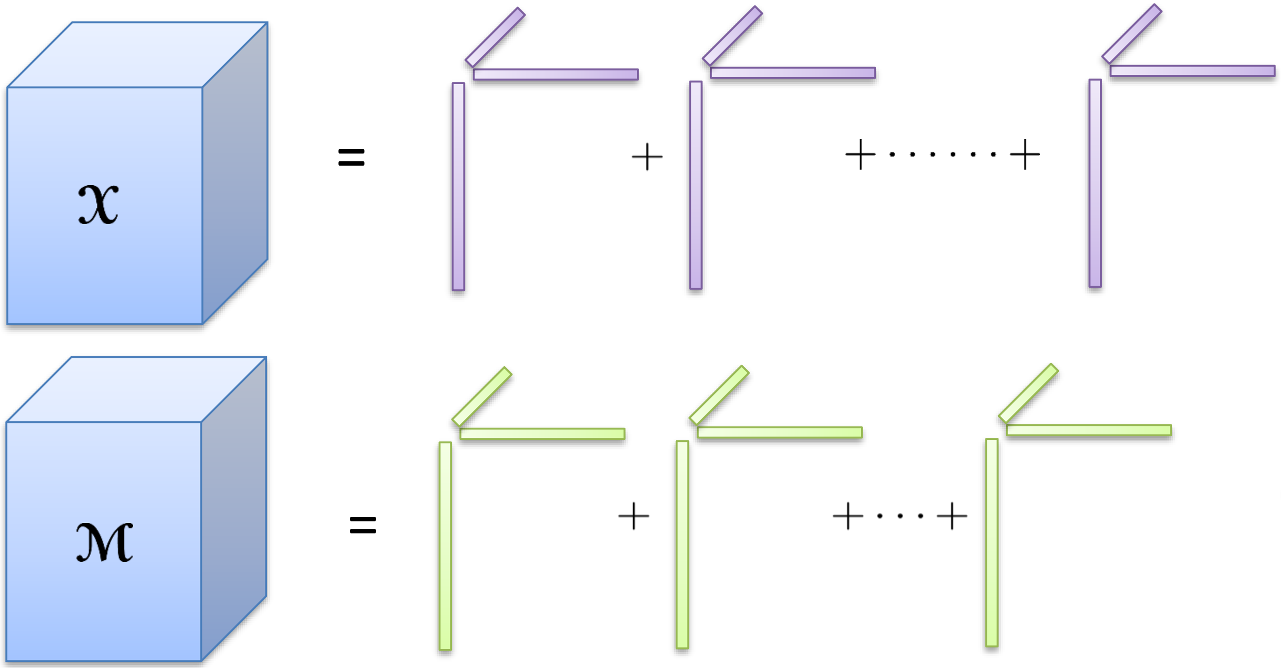
$$\omega_j = \frac{1}{r} \; \forall j \in [r]$$

Our Focus Today: Accelerating Computation for Special Case of Moment Tensors

$$\mathcal{X} = \frac{1}{p} \sum_{\ell=1}^{p} \mathbf{v}_\ell^{\otimes d}$$

Symmetric Data

Symmetric CP Model



$$\mathcal{M} = \sum_{j=1}^{r} \lambda_j \, \mathbf{a}_j^{\otimes d}$$

Given Observations

defined by

$\mathbf{V} \in \mathbb{R}^{n \times p}$

Want to Find Compact Representation

defined by

$\mathbf{A} \in \mathbb{R}^{n \times r}$

$r \ll p$

Optimization Approach for Symmetric CP of Symmetric Tensor Requires TTSV

Optimization Problem

$$\min_{\boldsymbol{\lambda},\mathbf{A}} F(\mathcal{X},\mathcal{M}) \equiv \frac{1}{2}\|\mathcal{X}-\mathcal{M}\|^2 \text{ where } \mathcal{M} = \sum_{j=1}^{r} \lambda_j \, \mathbf{a}_j^{\otimes d}$$

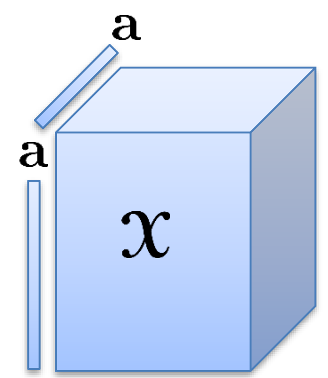
Gradients $\forall j \in [r]$

$$\frac{\partial F}{\partial \mathbf{a}_j} = -d\lambda_j \boxed{\mathcal{X}\mathbf{a}_j^{d-1}} + d\lambda_j \sum_{k=1}^{r} \lambda_k \langle \mathbf{a}_j, \mathbf{a}_k \rangle^{d-1} \mathbf{a}_k$$

$$\frac{\partial F}{\partial \lambda_j} = -\mathcal{X}\mathbf{a}_j^d + \sum_{k=1}^{r} \lambda_k \langle \mathbf{a}_j, \mathbf{a}_k \rangle^{d}$$

Plug function and gradient into favorite optimization method. My favorite: L-BFGS.

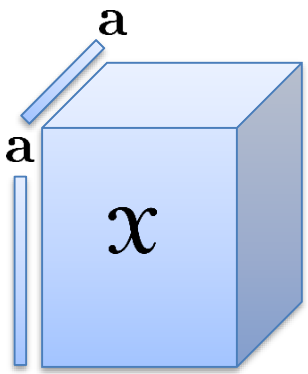
Bottleneck is TTSV which costs $O(n^d)$



Key Kernel: Tensor Times Single Vector (TTSV)

$$(\mathcal{X}\mathbf{a}^{d-1})_{i_1} = \sum_{i_2=1}^{n} \cdots \sum_{i_d=1}^{n} \left(x_{i_1 i_2 \ldots i_d} \prod_{k=2}^{d} a_{i_k} \right) \quad \forall i_1 \in [n]$$

Key Result: Implicit Computation of TTSV



TTSV Definition: $$(\mathcal{X}\mathbf{a}^{d-1})_{i_1} = \sum_{i_2=1}^{n} \cdots \sum_{i_d=1}^{n} \left(x_{i_1 i_2 \ldots i_d} \prod_{k=2}^{d} a_{i_k} \right) \quad \forall i_1 \in [n]$$

Lemma. Let $\mathcal{X} = \frac{1}{p} \sum_{\ell=1}^{p} \mathbf{v}_\ell^{\otimes d}$ and $\mathbf{V} = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_p \end{bmatrix}$, then

Entry-wise Power

$$\mathcal{X}\mathbf{a}^{d-1} = \frac{1}{p} \mathbf{V} \left[\mathbf{V}^{\intercal} \mathbf{a} \right]^{d-1}$$

$\boxed{O(n^d)}$ $\boxed{O(pn)}$

Minimal Change in Function/Gradient Calculation Replaces Expensive TTSV

function $\textsc{FG\_Explicit}(\mathcal{X}, \boldsymbol{\lambda}, \mathbf{A}, \alpha)$

  for $j = 1, \ldots, r$, **do** $\mathbf{y}_j = \mathcal{X}\mathbf{a}_j^{d-1}$, **end**

  for $j = 1, \ldots, r$, **do** $w_j = \mathbf{a}_j^T \mathbf{y}_j$, **end**

  $\mathbf{B} = \mathbf{A}^T \mathbf{A}$

  $\mathbf{C} = [\mathbf{B}]^{d-1}$

  $\mathbf{u} = (\mathbf{B} * \mathbf{C})\boldsymbol{\lambda}$

  $f = \alpha + \boldsymbol{\lambda}^T \mathbf{u} - 2\mathbf{w}^T \boldsymbol{\lambda}$

  $\mathbf{g}_{\boldsymbol{\lambda}} = -2(\mathbf{w} - \mathbf{u})$

  $\mathbf{G}_{\mathbf{A}} = -2d(\mathbf{Y} - \mathbf{A}\mathbf{D}_{\boldsymbol{\lambda}}\mathbf{C})\mathbf{D}_{\boldsymbol{\lambda}}$

  return $f, \mathbf{g}_{\boldsymbol{\lambda}}, \mathbf{G}_{\mathbf{A}}$

end function

function $\textsc{FG\_Implicit}(\mathbf{V}, \boldsymbol{\lambda}, \mathbf{A}, \alpha)$

  $\mathbf{Y} = \frac{1}{p}\mathbf{V}[\mathbf{V}^{\intercal}\mathbf{A}]^{d-1}$

  for $j = 1, \ldots, r$, **do** $w_j = \mathbf{a}_j^T \mathbf{y}_j$, **end**

  $\mathbf{B} = \mathbf{A}^T \mathbf{A}$

  $\mathbf{C} = [\mathbf{B}]^{d-1}$

  $\mathbf{u} = (\mathbf{B} * \mathbf{C})\boldsymbol{\lambda}$

  $f = \alpha + \boldsymbol{\lambda}^T \mathbf{u} - 2\mathbf{w}^T \boldsymbol{\lambda}$

  $\mathbf{g}_{\boldsymbol{\lambda}} = -2(\mathbf{w} - \mathbf{u})$

  $\mathbf{G}_{\mathbf{A}} = -2d(\mathbf{Y} - \mathbf{A}\mathbf{D}_{\boldsymbol{\lambda}}\mathbf{C})\mathbf{D}_{\boldsymbol{\lambda}}$

  return $f, \mathbf{g}_{\boldsymbol{\lambda}}, \mathbf{G}_{\mathbf{A}}$

end function

Implicit up to 16X Faster than Explicit for Smaller Problems

Rank-r Symmetric CP Tensor Factorization
for d-way tensor of size n

$$r < n < p$$

Method	Storage	Per-Iteration
Explicit	$O(n^d)$	$O(rn^d)$
Implicit	$O(pn)$	$O(pnr)$

Implicit cheaper if $p < O(n^{d-1})$

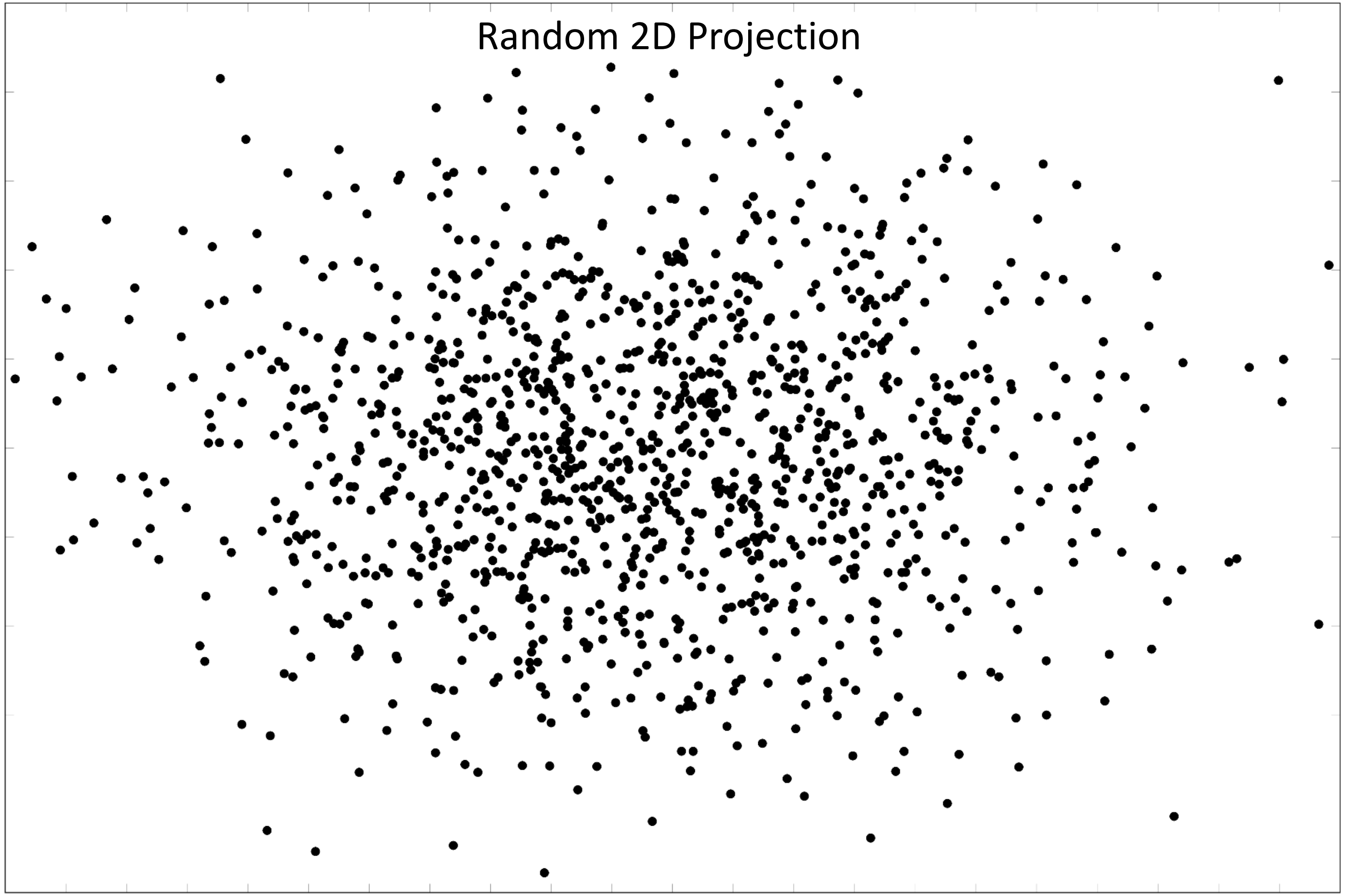
Average cost per iteration for $r = 5$ over 10 runs

d	n	p	n^{d-1}	Explicit	Implicit	Ratio
3	75	3750	5625	5e-4 sec.	8e-4 sec.	1x
3	375	3750	140625	2e-2	5e-3	5x
4	75	3750	421875	1e-2	9e-4	16x

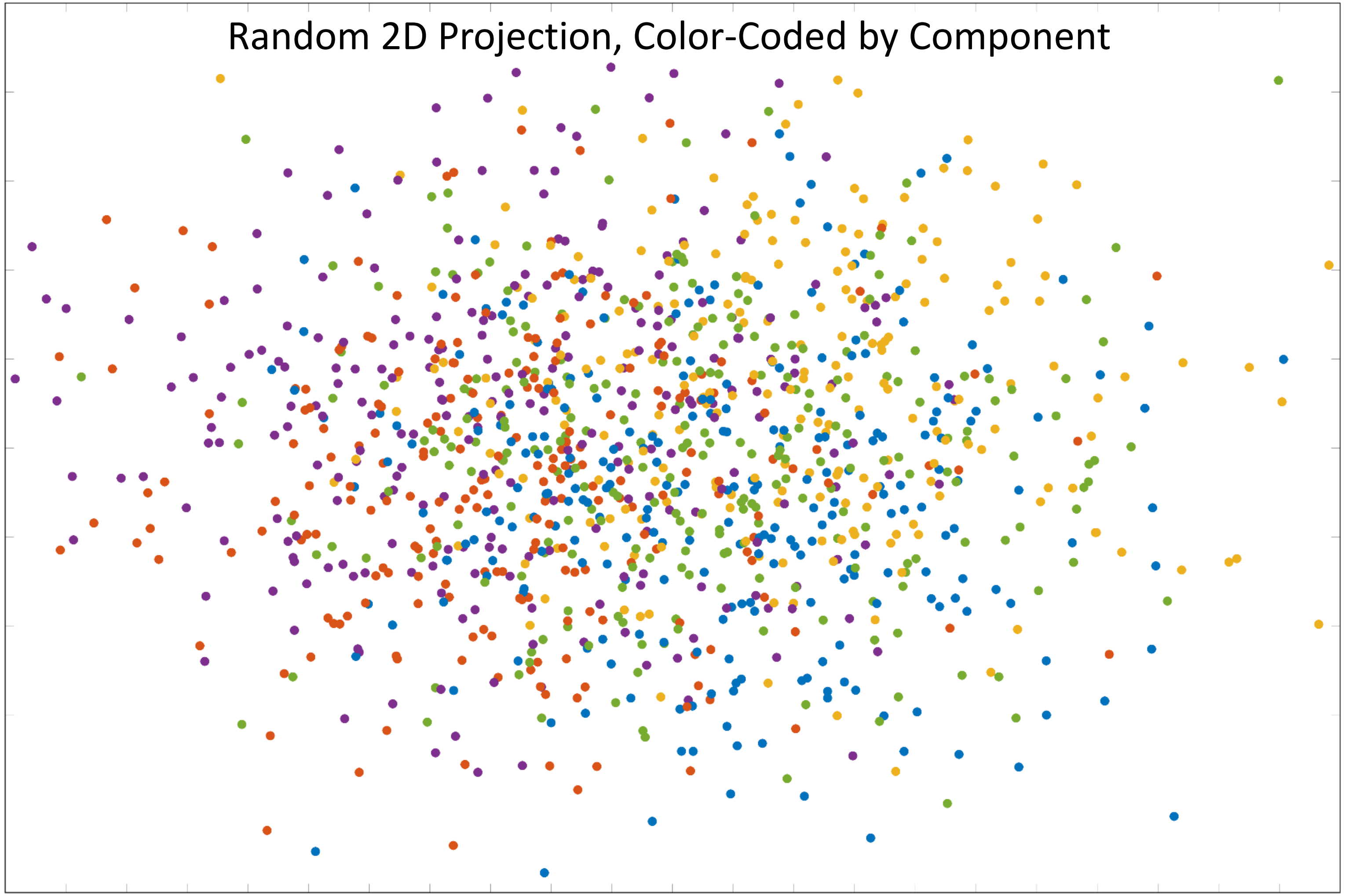
GMM Example with r=5 (components), n=500 (dim.), σ=.1 (noise), and p=1250 (obs.)

For $d = 3$, explicit method requires 1 GB storage

For d =4, explicit method requires 500 GB storage



GMM Example with r=5 (components), n=500 (dim.), σ=.1 (noise), and p=1250 (obs.)



GMM Example with r=5 (components), n=500 (dim.), σ=.1 (noise), and p=1250 (obs.)

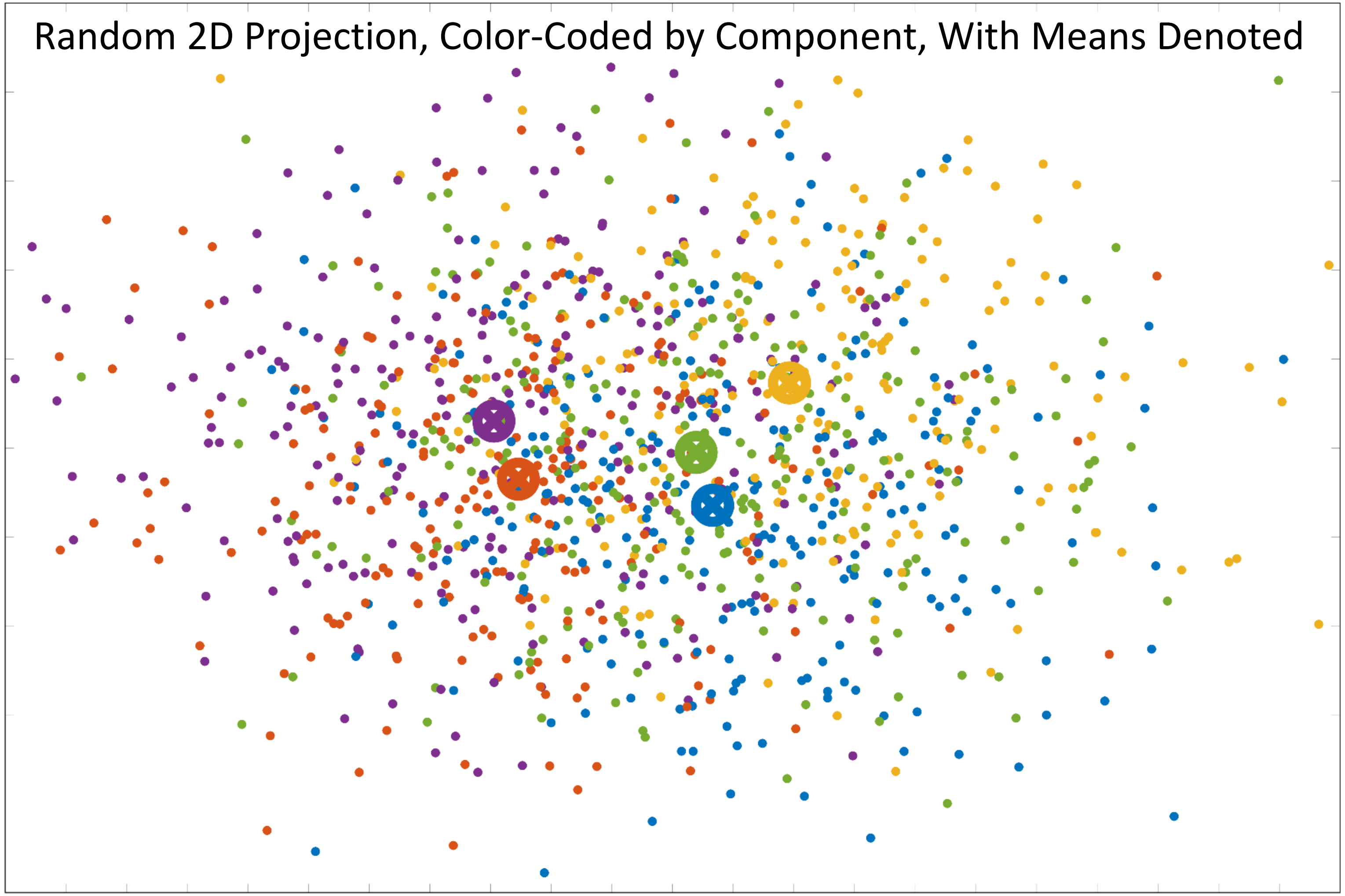
$$\boldsymbol{\mu}_j \in \mathbb{R}^{500}$$

$$\|\boldsymbol{\mu}_j\|_2 = 1$$

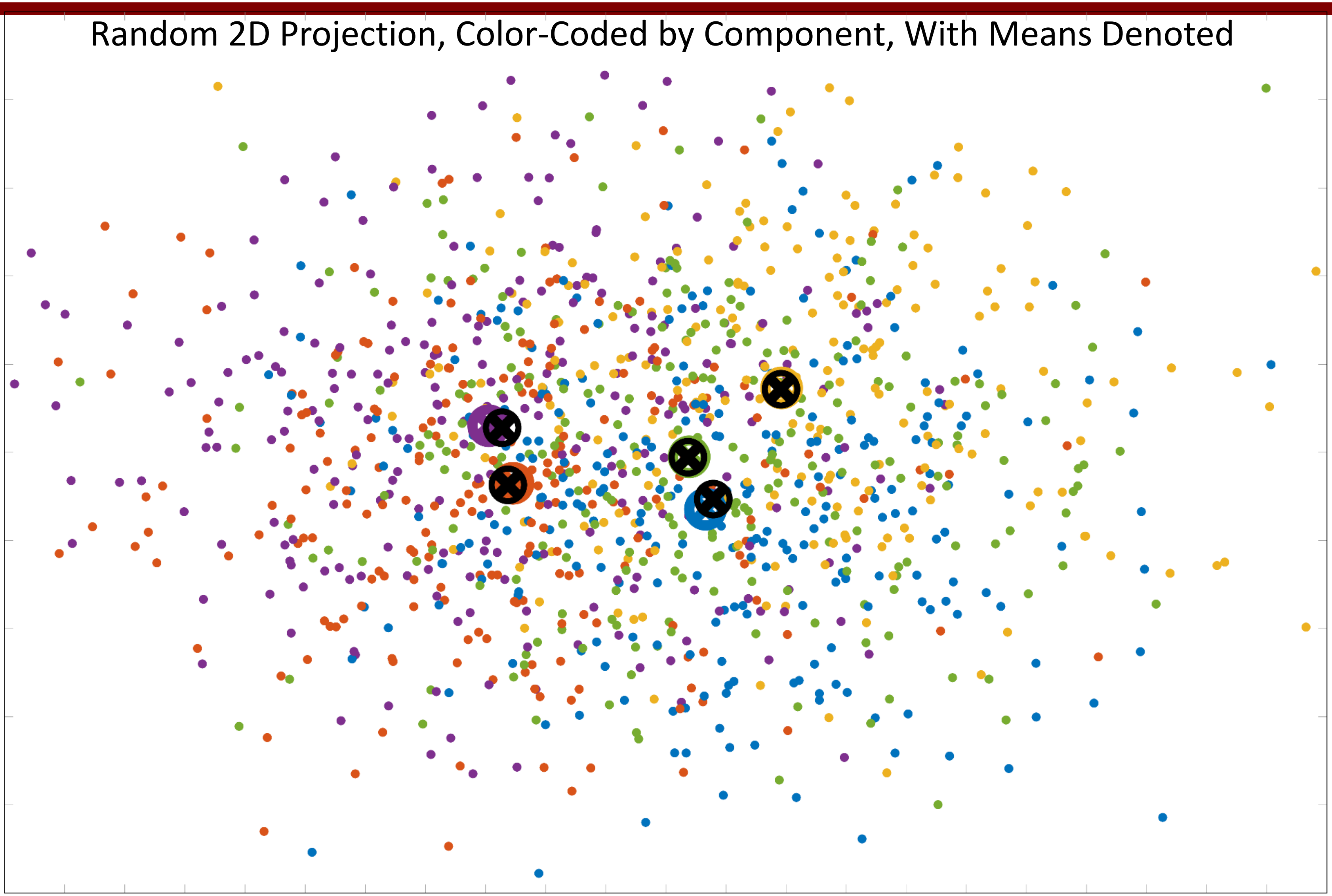
$$\forall j \in [r]$$

$$\boldsymbol{\mu}_j^T \boldsymbol{\mu}_k = 0.5$$

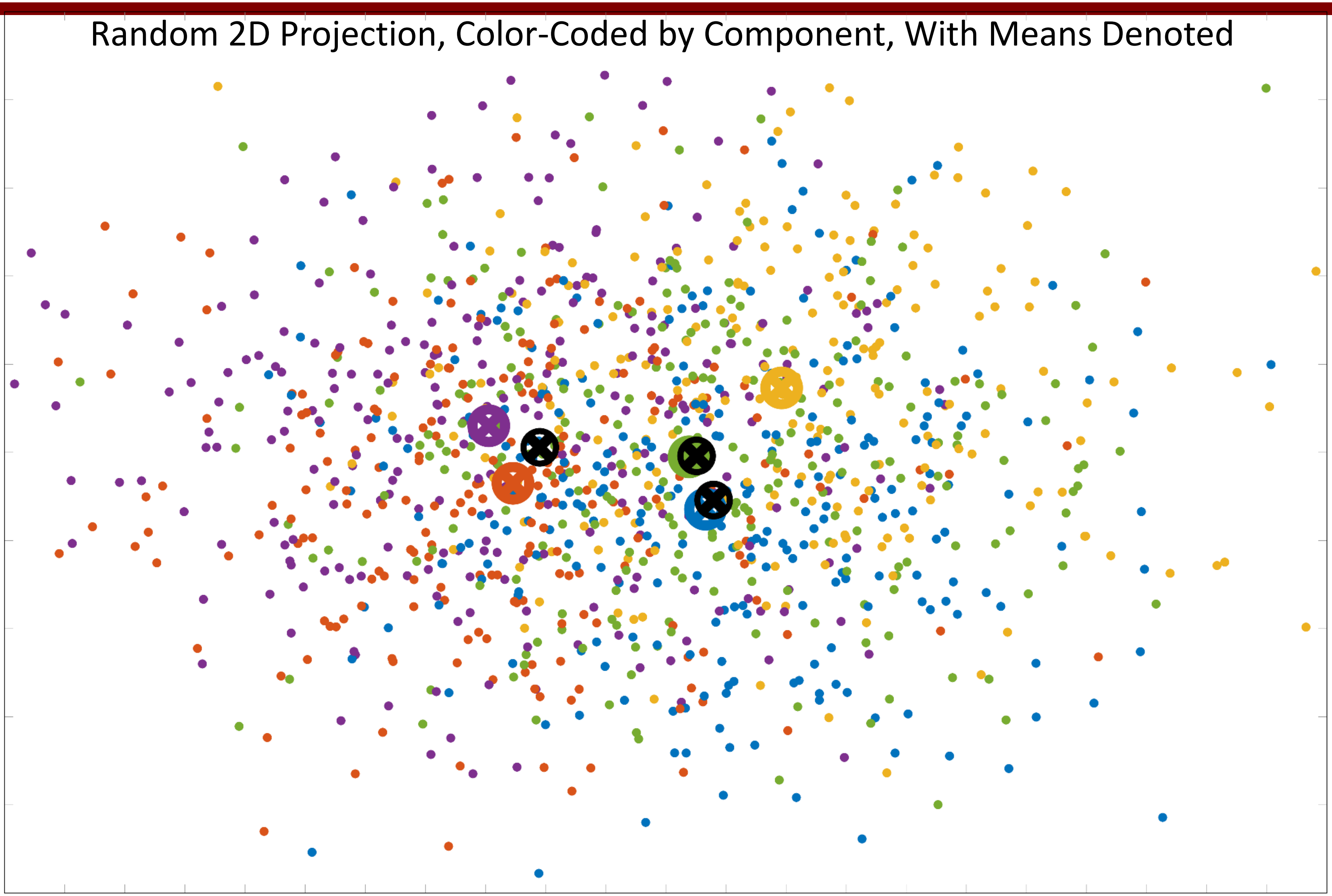
$$\forall j \neq k$$



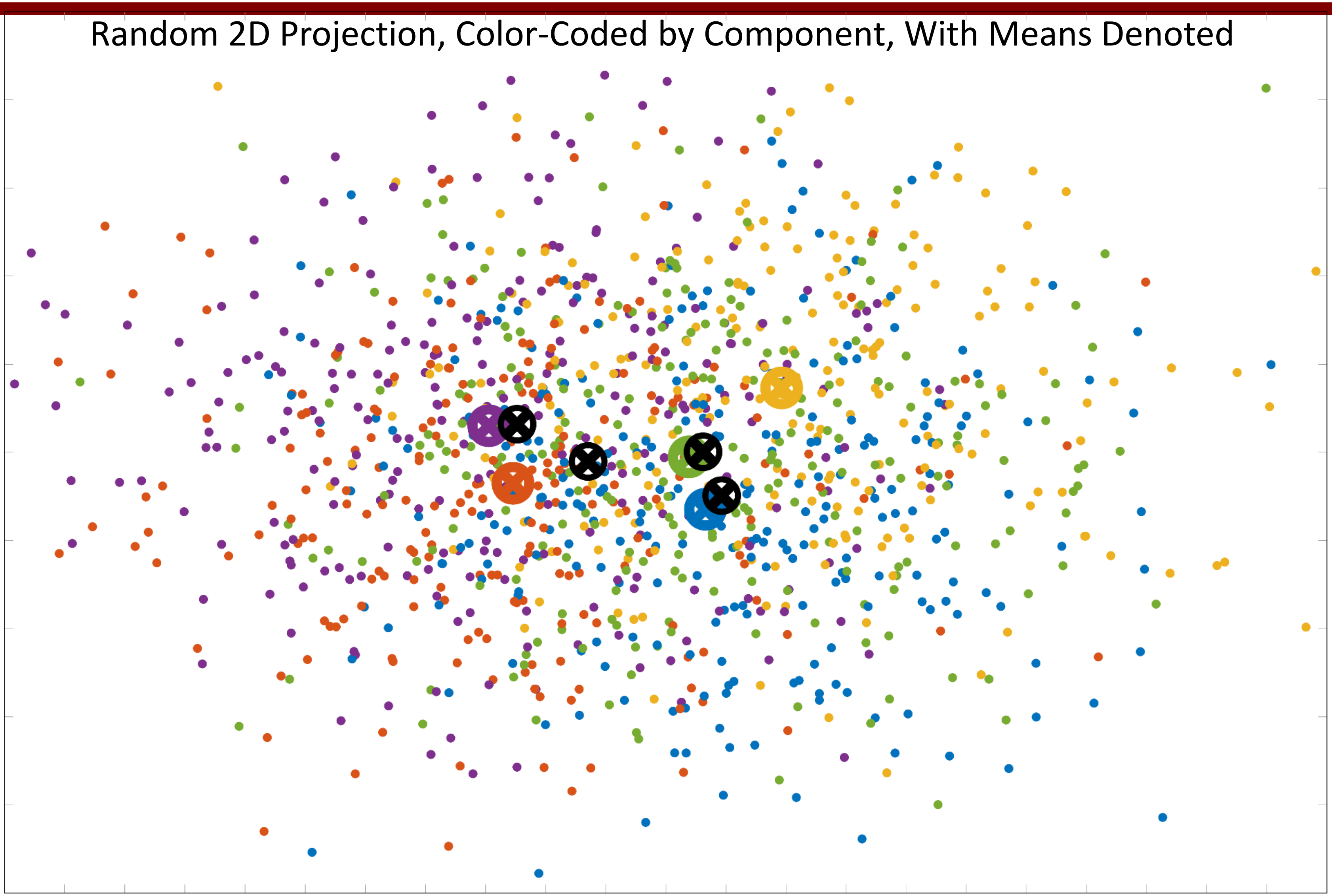
Identified Factors for $\hat{r}$=5



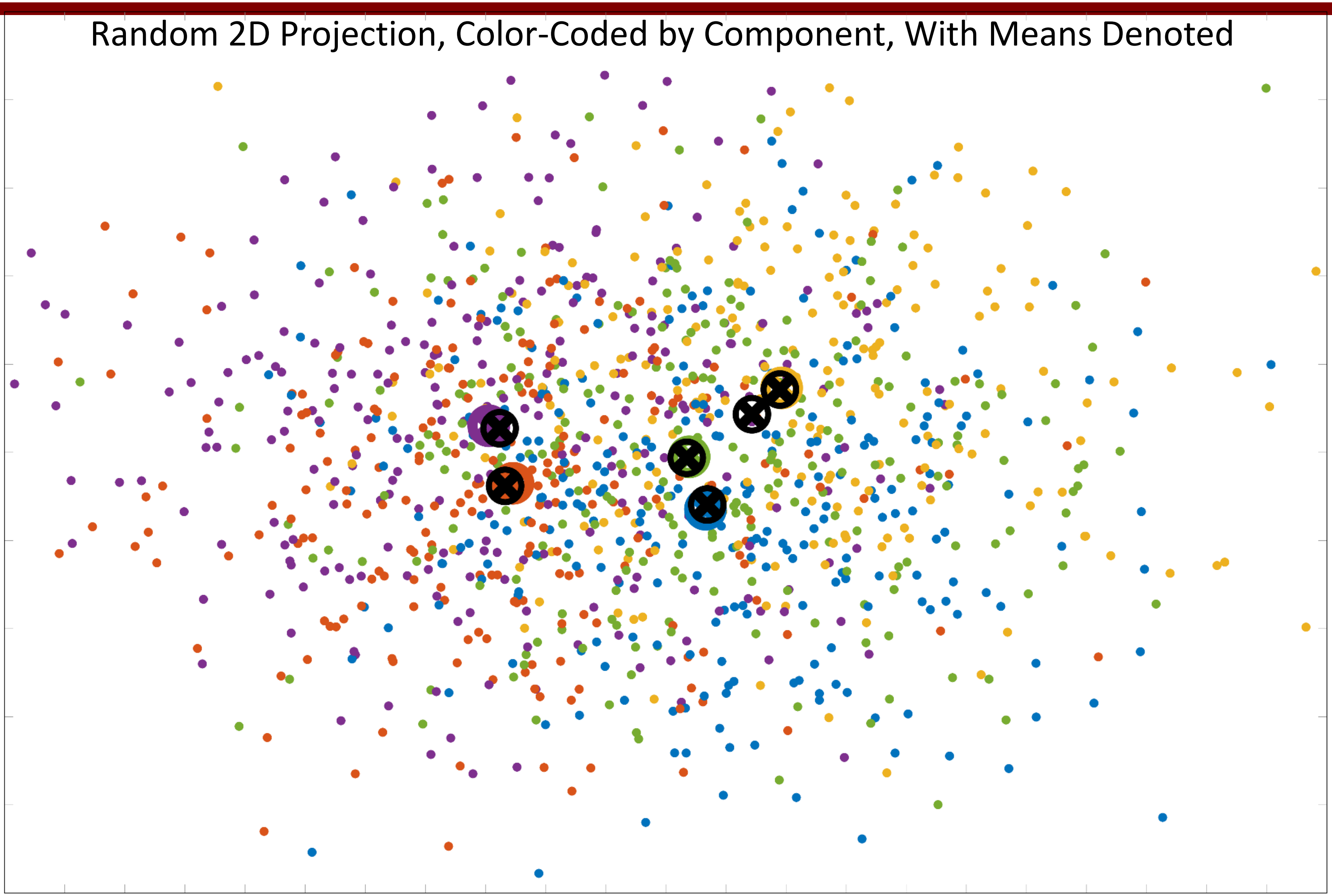
Identified Factors for $\hat{r}$=3



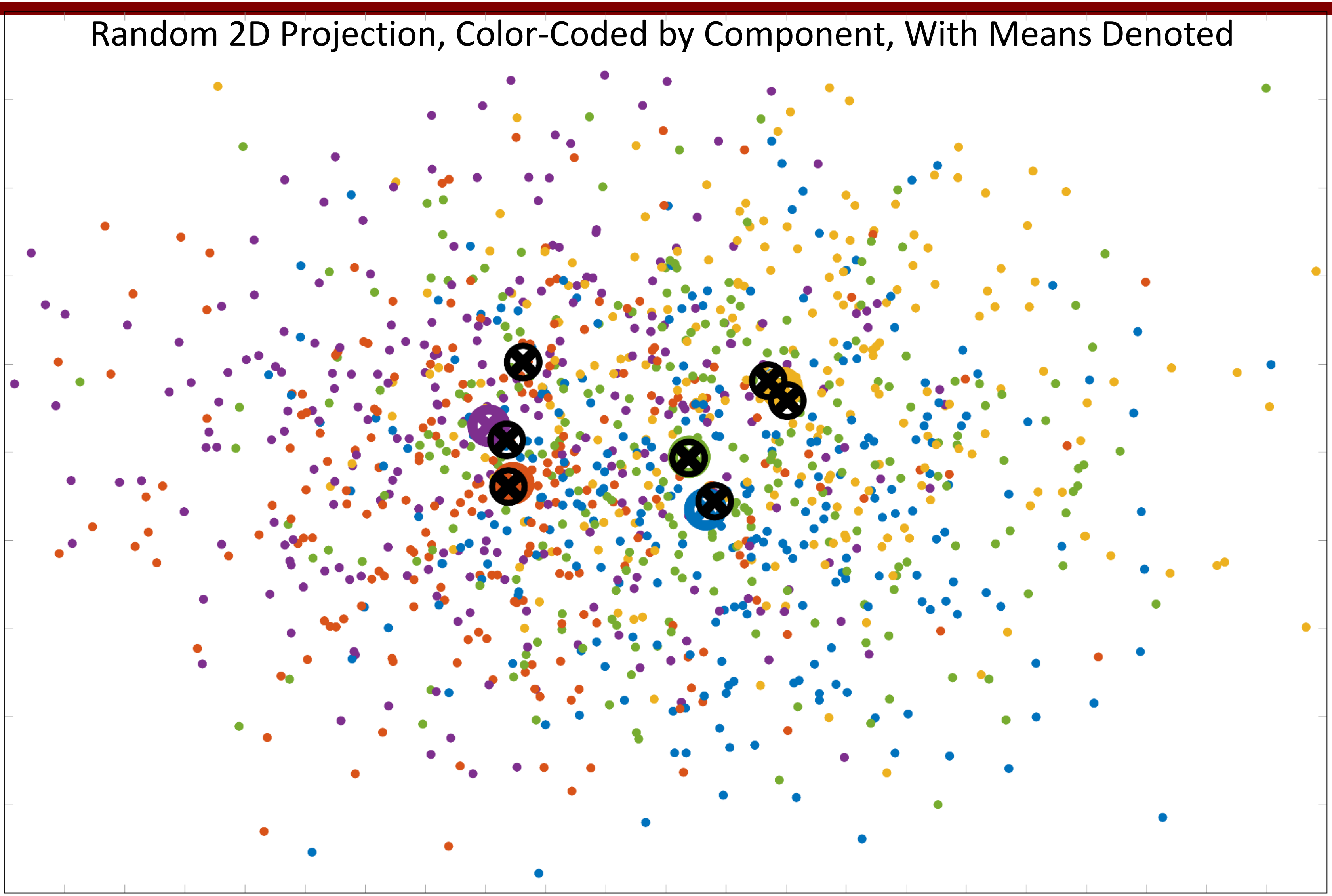
Identified Factors for $\hat{r}$=4



Identified Factors for $\hat{r}$=6



Identified Factors for $\hat{r}$=7

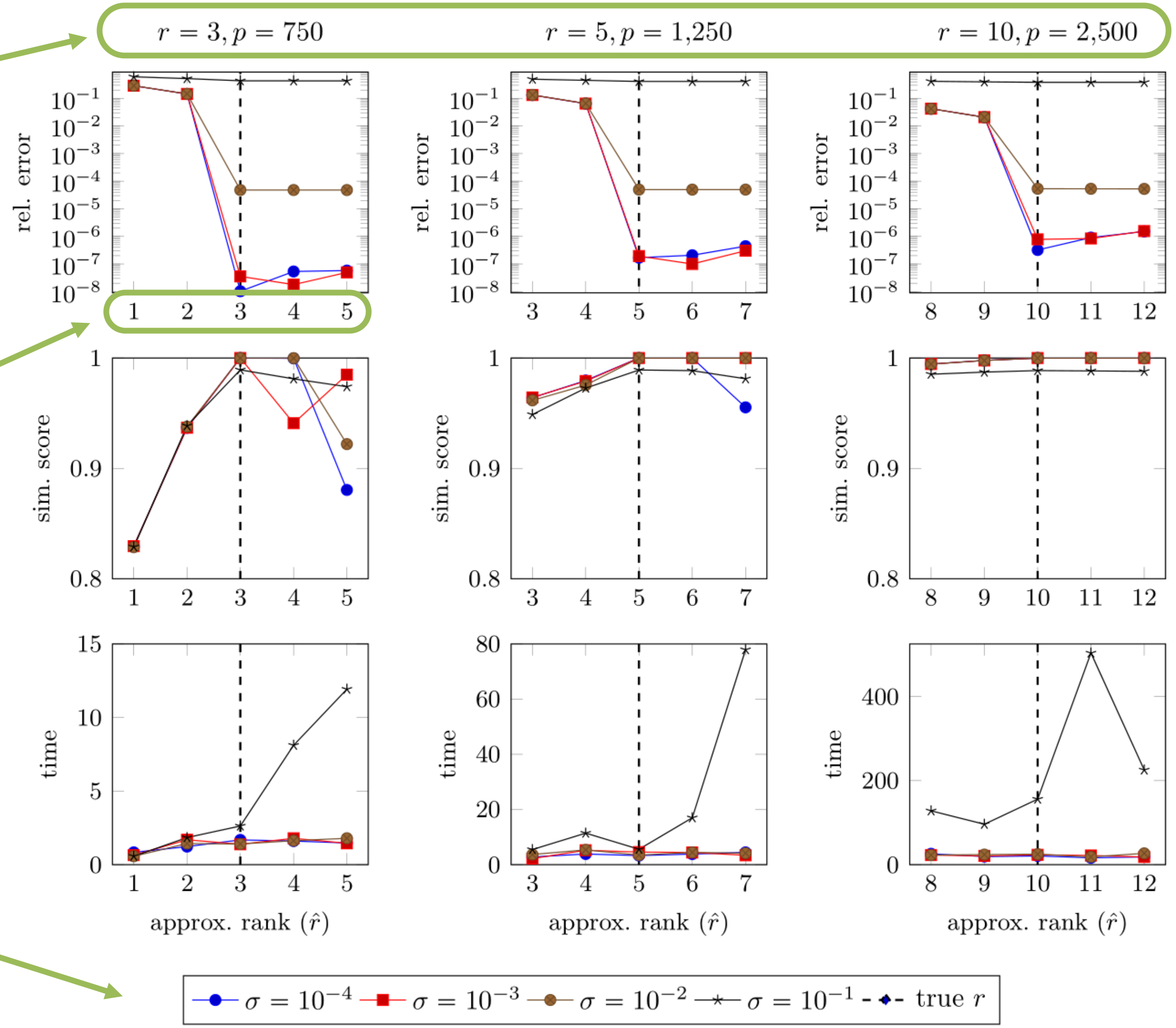


GMM Performance for Third-Order (d=3)

Varying Number of True Components

Varying Number of Computed Components (Over/Under Estimate)

Varying Noise



Best Error over 10 Runs Compared to Empirical Moment Tensor

$$\mathcal{X} = \frac{1}{p} \sum_{\ell=1}^{p} \mathbf{v}_\ell^{\otimes 3}$$

Average Cosine of Angle Between True Means and Computed (1 = perfect match)

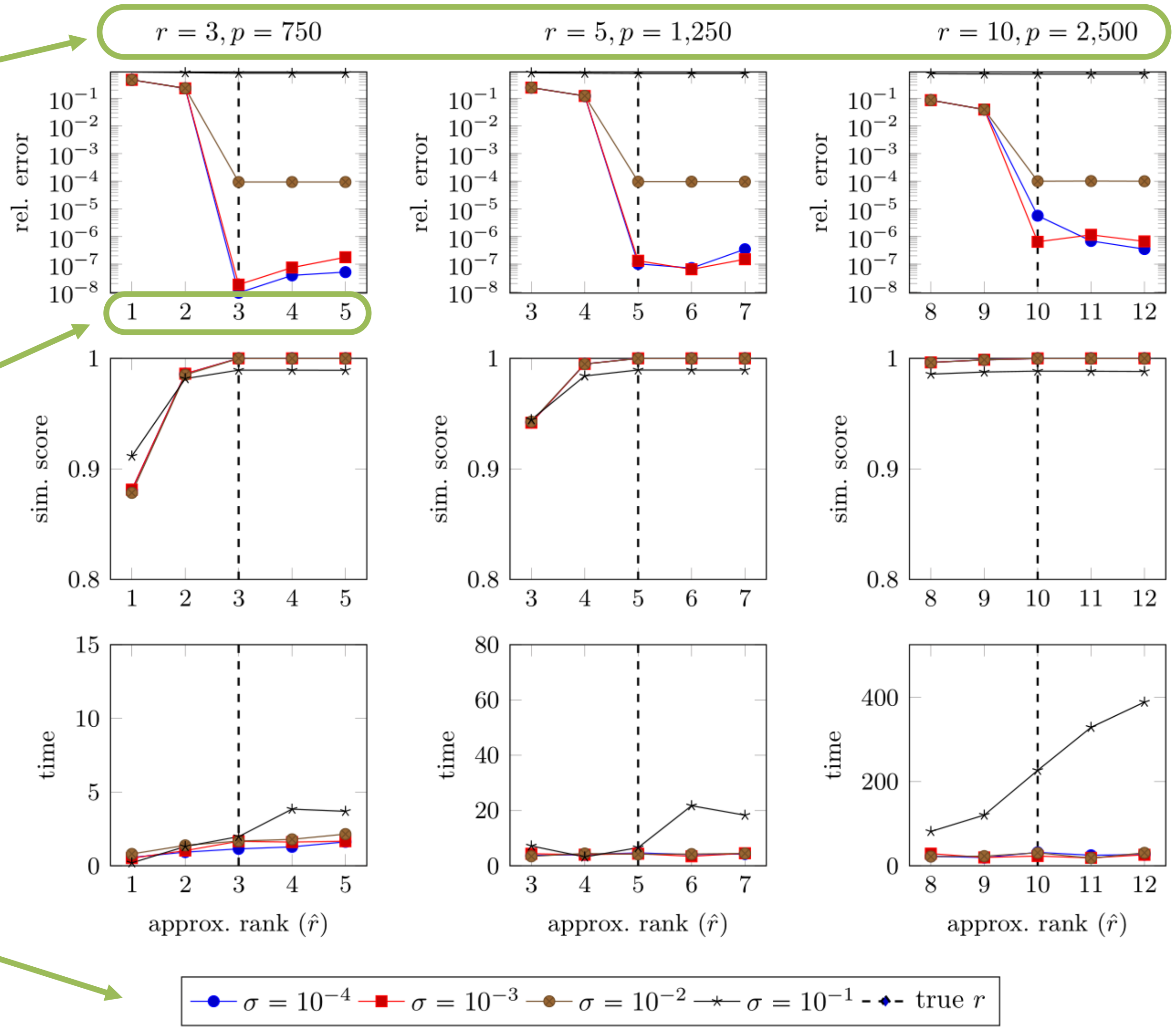
Total Time for Ten Runs

GMM Performance for Fourth-Order (d=4)

Varying Number of True Components

Varying Number of Computed Components (Over/Under Estimate)

Varying Noise



Best Error over 10 Runs Compared to Empirical Moment Tensor

$$\mathcal{X} = \frac{1}{p} \sum_{\ell=1}^{p} \mathbf{v}_\ell^{\otimes 3}$$

Average Cosine of Angle Between True Means and Computed (1 = perfect match)

Total Time for Ten Runs

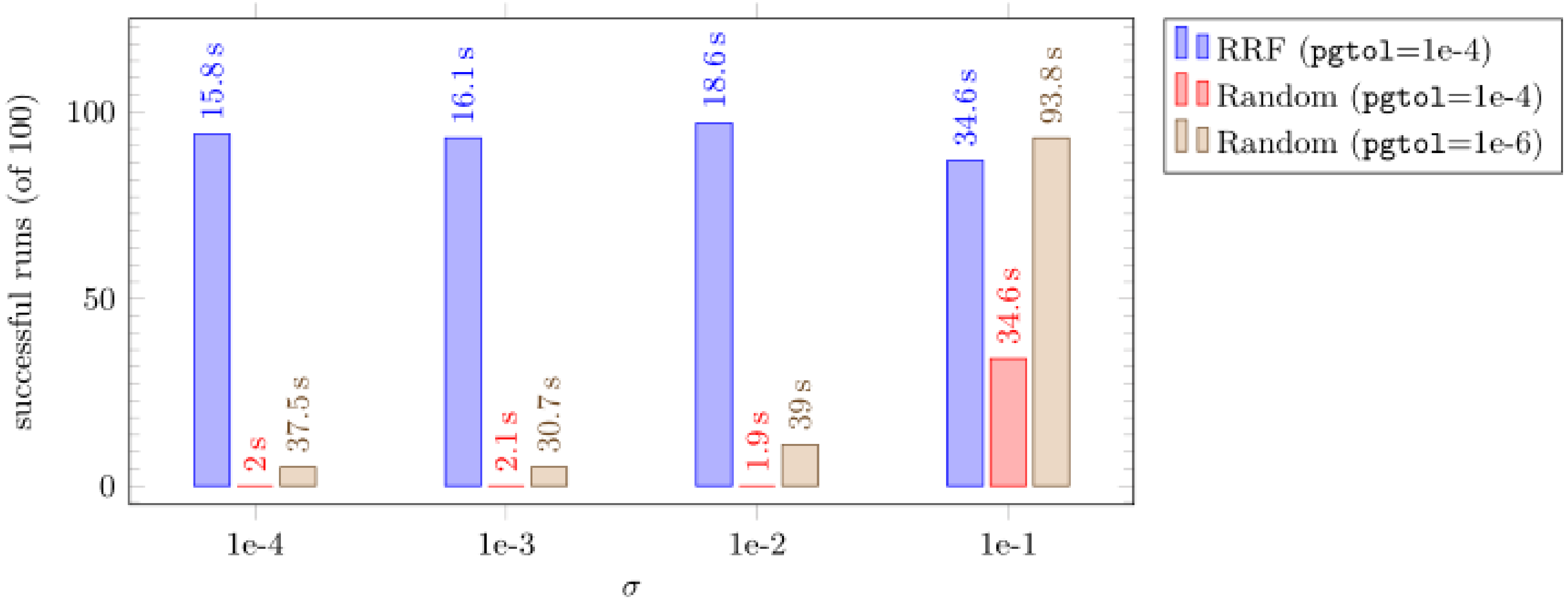
Choosing Starting Guess Within Range of Observations is Key for Low Noise!

Randomized Range Finder (RRF): $\mathbf{A}_0 = \mathbf{V}\boldsymbol{\Omega},\ \boldsymbol{\Omega} \sim \mathcal{N}(0,1)^{p \times \hat{r}}$

Random: $\mathbf{A}_0 \sim \mathcal{N}(0,1)^{n \times \hat{r}}$

[with columns normalized in both cases]

Results of computing $\hat{r} = 3$ approximation for moment tensor of order $d = 3$, with $r = 3$ components, $n = 500$ dimensions, and $p = 750$ observations



For Massive Numbers of Observations, Use Stochastic Variants

$$\mathbf{V} \in \mathbb{R}^{n \times p}$$

Sample columns with replacement

$$\tilde{\mathbf{V}} \in \mathbb{R}^{n \times s}$$

$$\mathcal{X} = \frac{1}{p} \sum_{\ell=1}^{p} \mathbf{v}_\ell^{\otimes d}$$

$$\tilde{\mathcal{X}} = \frac{1}{s} \sum_{\ell=1}^{s} \tilde{\mathbf{v}}_\ell^{\otimes d}$$

$$\Rightarrow \qquad \mathbb{E}[\tilde{\mathcal{X}}\mathbf{a}^{d-1}] = \mathcal{X}\mathbf{a}^{d-1}$$

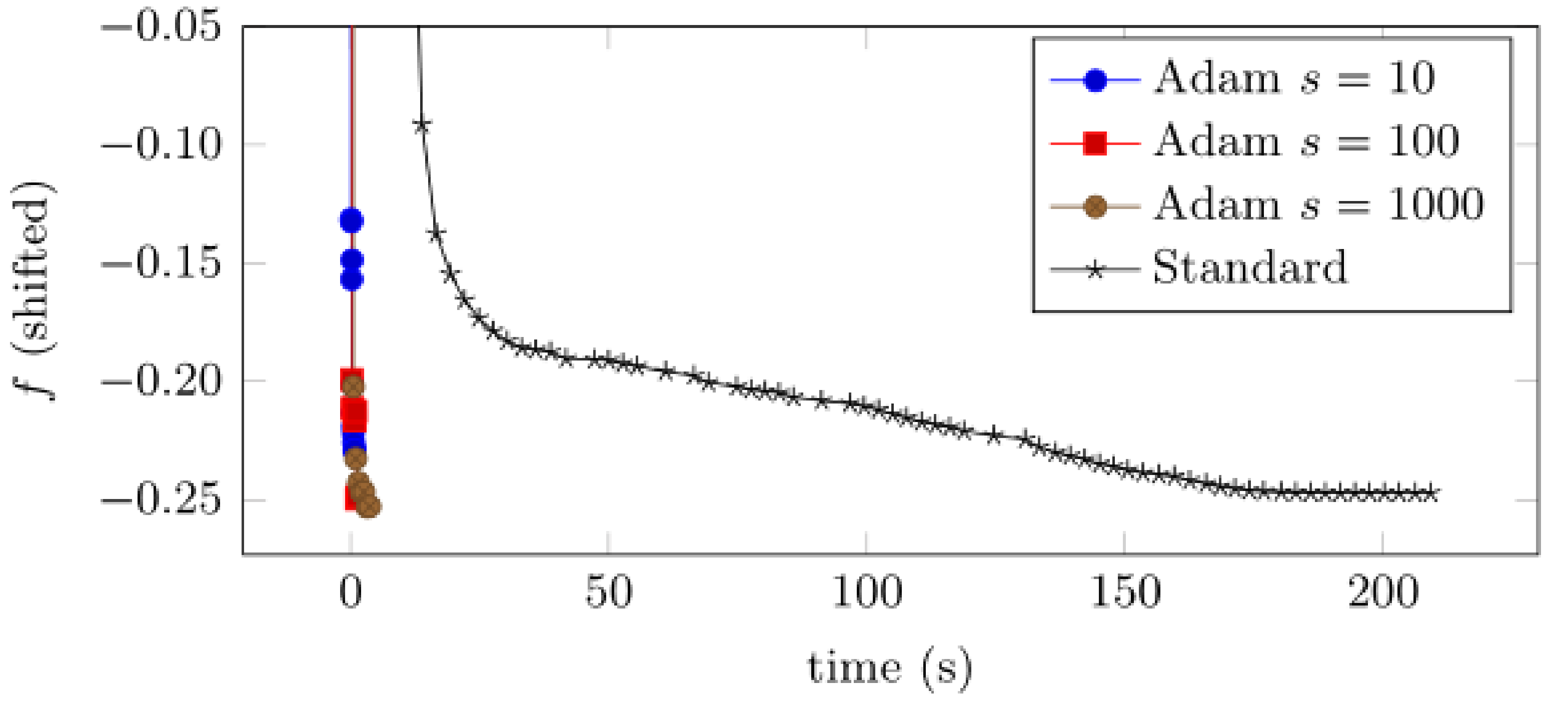
Example Results
$\hat{r} = r = 10, n = 500,$
$\sigma = 0.1, d = 3$
$p = 100{,}000$

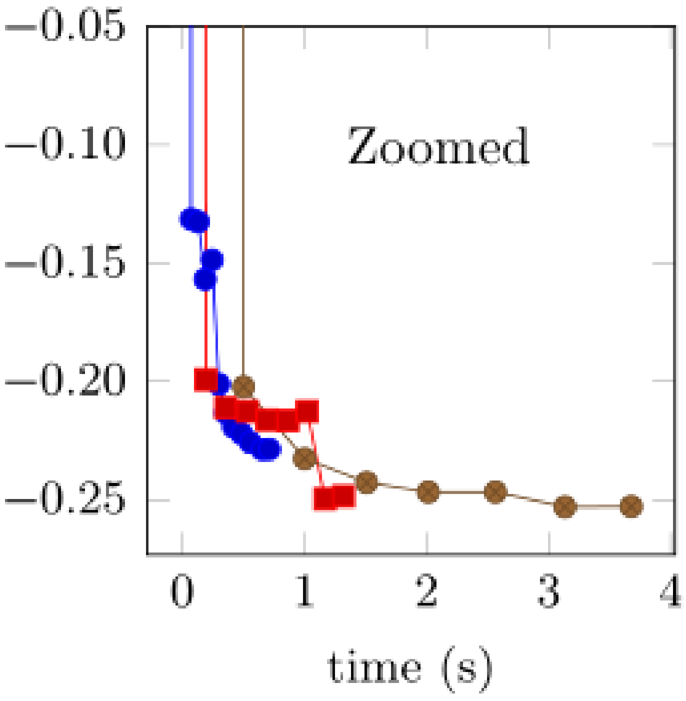
Method	Best f (shifted)	Sim. Score	Total Time (s)
standard	-0.2471	0.9998	2166.70
Adam, s=10	-0.2209	0.9225	8.03
Adam, s=100	-0.2427	0.9929	10.48
Adam, s=1000	-0.2464	0.9990	41.00

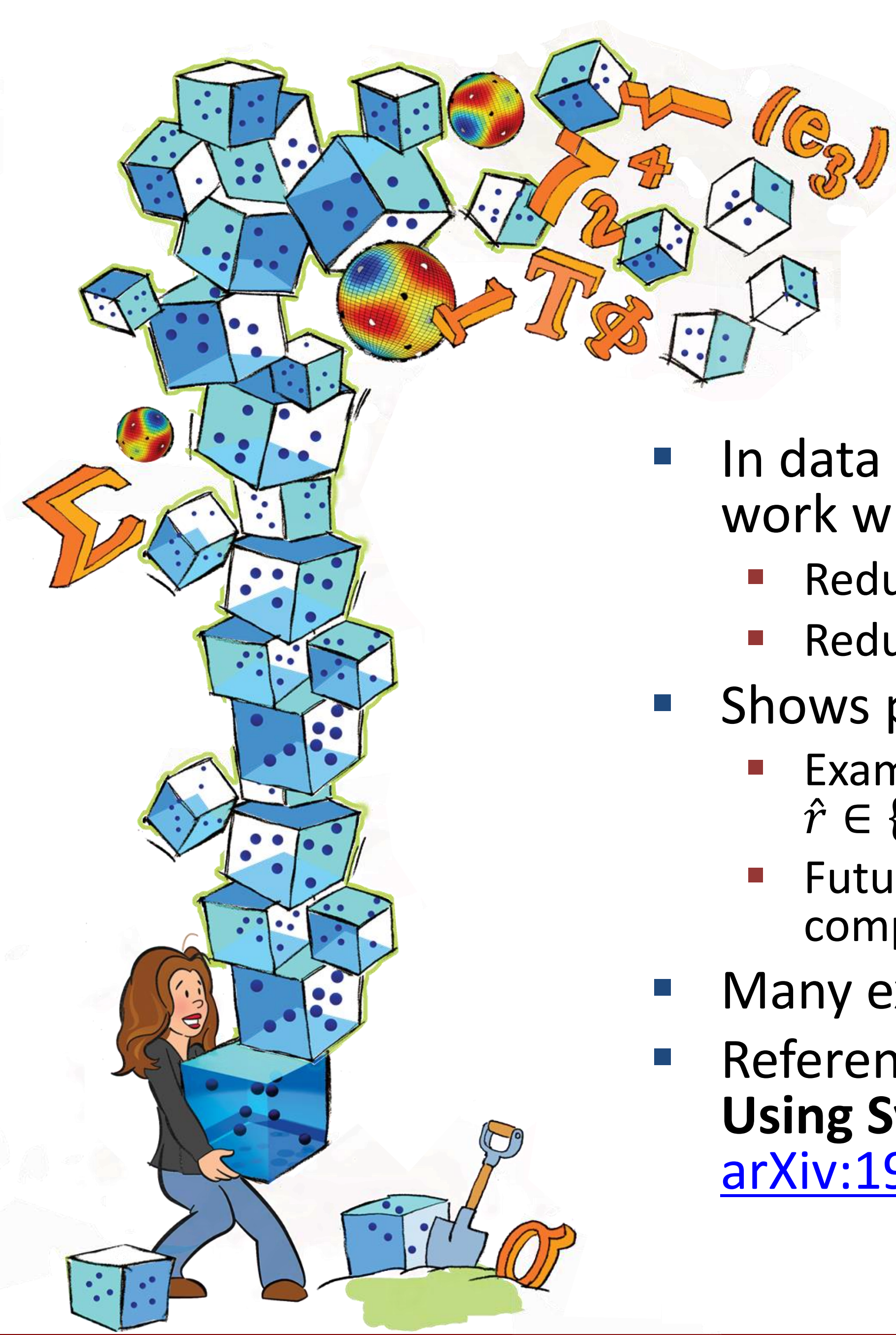
Speed Advantage for Stochastic Methods

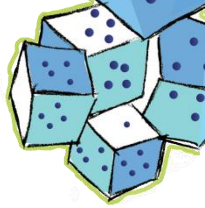
Best Runs (of 10)

$\hat{r} = r = 10, n = 500, \sigma = 0.1, d = 3, p = 100{,}000$









Conclusions and Future Work

- In data analysis, dth-order moment is expensive to compute – instead work with implicit moment
  - Reduces storage from $O(n^d)$ to $O(np)$
  - Reduces computation per iteration from $O(rn^d)$ to $O(rnp)$
- Shows promise for fitting spherical GMMs
  - Example with $n = 500$ (dimension), $r \in \{3,5,10\}$ (components), $p = 250r$, $\hat{r} \in \{r-2, \ldots, r+2\}$, and $d = 3,4$ (orders)
  - Future work will incorporate lower-order terms, different σ for each component, multiple values for d simultaneously, etc.
- Many extensions possible, e.g., for subspace power method
- Reference: S. Sherman, T. G. Kolda. **Estimating Higher-Order Moments Using Symmetric Tensor Decomposition**, to appear in SIMAX, arXiv:1911.03813