Sandia National Laboratories

SAND2019-2546C

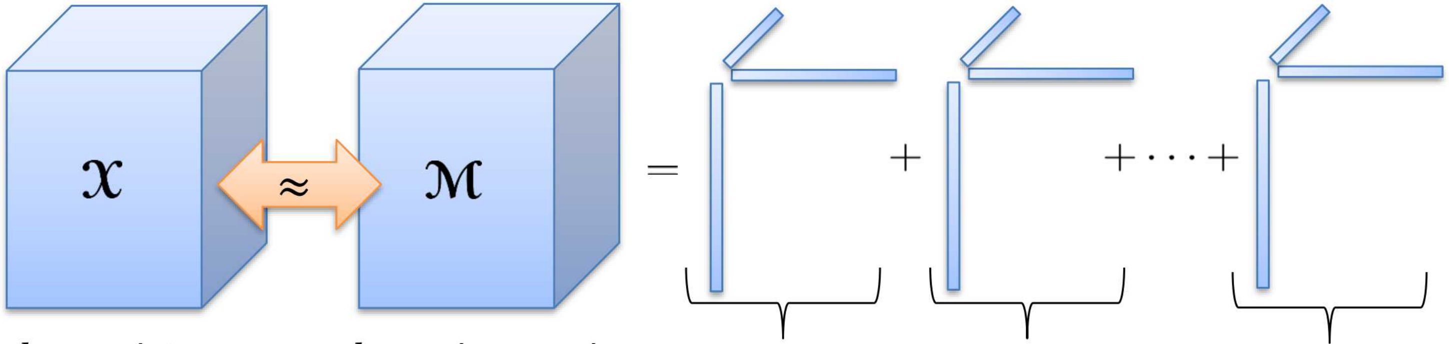
Stochastic Gradient Descent for Large-scale Generalized CP Tensor Decomposition

Tamara G. Kolda, Sandia Natl. Labs.
Joint with David Hong (Michigan), Jed Duersch (Sandia)

Supported by the Laboratory Directed Research and Development program at Sandia National Laboratories, 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

Generalized Canonical Polyadic (GCP) Tensor Decomposition



d-way data tensor of size n^d

d-way low-rank model tensor of size n^d and rank r

rank-one component $j = 1$

rank-one component $j = 2$

rank-one component $j = r$

$$\mathcal{X} \approx \mathcal{M} \quad \text{where} \quad \mathcal{M} = \sum_{j=1}^{r} \mathbf{A}_1(:,j) \circ \mathbf{A}_2(:,j) \circ \cdots \circ \mathbf{A}_d(:,j)$$

Low-rank: $\operatorname{rank}(\mathcal{M}) \leq r \ll n^d$

Factor matrices: $\mathbf{A}_k \in \mathbb{R}^{n \times r}$ for $k \in \{1, \ldots, d\}$

GCP

$$\min \; F(\mathcal{X}, \mathcal{M}) \equiv \sum_{i \in \Omega} f(x_i, m_i)$$

$$\text{s.t.} \; \operatorname{rank}(\mathcal{M}) \leq r$$

i = multi-index
Ω = all indices

- Standard CP [Hitchcock, 1927; Carrol & Chang, 1970; Harshman, 1970]

  $$f(x,m) = (x - m)^2$$

- Poisson CP [Welling & Webber, 2001; Chi & Kolda, 2009]

  $$f(x,m) = m - x \log m$$

- Logistic CP, etc. [Hong, Kolda, Duersch, 2018]

  $$f(x,m) = \log(m+1) - x \log(m)$$

Gradient-based Optimization for Fitting the GCP Model

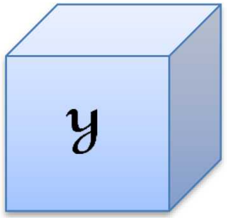
Sandia National Laboratories

GCP

$$\min \; F(\mathcal{X}, \mathcal{M}) \equiv \sum_{i \in \Omega} f(x_i, m_i)$$

$$\text{s.t.} \; \text{rank}(\mathcal{M}) \leq r$$

Define: Elementwise partial gradient tensor, same size as data tensor = n^d



$$y_i = \begin{cases} \frac{\partial f}{\partial m}(x_i, m_i) & \text{if } i \in \Omega \\ 0 & \text{otherwise} \end{cases}$$

Define: Khatri-Rao product in all modes but one of size $n^{d-1} \times r$

$$\mathbf{Z}_k = \mathbf{A}_d \odot \cdots \odot \mathbf{A}_{k+1} \odot \mathbf{A}_{k-1} \odot \cdots \odot \mathbf{A}_1$$

Gradients computed via a sequence of MTTKRPs:

$$\mathbf{G}_k \equiv \frac{\partial F}{\partial \mathbf{A}_k} = \mathbf{Y}_{(k)} \mathbf{Z}_k$$

MTTKRP

gradient for mode k factor matrix of size $n \times r$

tensor unfolded in mode k into matrix of size $n \times n^{d-1}$

MTTKRPs can be computed efficiently…

- Bader & Kolda, SISC, 2007 – Dense and sparse
- Phan, Tichavsky, Cichocki, 2013 – Sequence
- Smith et al., IPDPS 2015 – Sparse
- Kaya & Ucar, SC 2015 – Sparse
- Li et al., IPDPS 2017 – Sparse
- Hayashi et al., 2017 – Dense
- Ballard, Knight, Rouse, 2017 – Dense

Hong, Kolda, Duersch, 2018

Stochastic Gradient Descent (SGD) for GCP

30-Second Tutorial on SGD

$$\min f(x)$$

Gradient Descent (GD)

α = learning rate

$$x^{(t+1)} = x^{(t)} - \alpha g^{(t)}$$

Stochastic Gradient Descent (SGD)

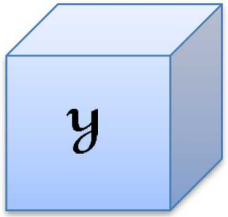
$$x^{(t+1)} = x^{(t)} - \alpha \tilde{g}^{(t)}$$

$$\mathbb{E}[\tilde{g}^{(t)}] = g^{(t)} \equiv \nabla f(x^{(t)})$$

Adam (Kingma & Ba, 2015)

Adaptive momentum SGD

Standard gradient

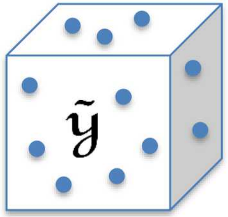


$$y_i = \begin{cases} \frac{\partial f}{\partial m}(x_i, m_i) & \text{if } i \in \Omega \\ 0 & \text{otherwise} \end{cases}$$

$$\mathbf{G}_k = \mathbf{Y}_{(k)} \mathbf{Z}_k$$

Cost: $O(rn^d)$ flops

Stochastic gradient



Random sparse tensor: $\tilde{\mathcal{W}}$ $\quad s \equiv \text{nnz}(\tilde{\mathcal{W}}) \ll n^d$

$$\tilde{\mathcal{Y}} = \tilde{\mathcal{W}} * \mathcal{Y}$$

$$\tilde{y}_i = \begin{cases} \tilde{w}_i \cdot \frac{\partial f}{\partial m}(x_i, m_i) & \text{if } \tilde{w}_i \neq 0 \\ 0 & \text{if } \tilde{w}_i = 0 \end{cases}$$

$$\tilde{\mathbf{G}}_k = \tilde{\mathbf{Y}}_{(k)} \mathbf{Z}_k$$

Cost: $O(rds)$ flops

<u>Theorem</u>: $\mathbb{E}[\tilde{\mathcal{W}}] = \mathbf{1} \Rightarrow \mathbb{E}[\tilde{\mathbf{G}}_k] = \mathbf{G}_k$

($\mathbf{1}$: all ones tensor)

Stochastic Weight Matrix: Single Element

Goal: Random ***sparse*** tensor of size n^d such that every element has an expected value of 1

1. Choose random tensor entry ξ
2. Define random tensor with single nonzero as follows

$$\tilde{w}_i = \begin{cases} n^d & \text{if } i = \tilde{\xi} \\ 0 & \text{otherwise} \end{cases}$$

Theory

Claim: $\mathbb{E}[\tilde{\mathcal{W}}] = \mathbf{1}$

Proof:
$$\mathbb{E}(\tilde{w}_i) = p_i \cdot n^d + (1 - p_i) \cdot 0 = \frac{1}{n^d} \cdot n^d = 1$$

Shortcoming of sampling only single element…

- Higher variance in stochastic gradient
- Only one row of the unfolded tensor has a nonzero so stochastic gradient only nonzero for single row of each factor matrix
- Gradient = $O(rd)$ work versus Update = $O(rdn)$ work
- *Can afford more work per gradient calculation!*

Stochastic Weight Matrix: Multiple Elements

<u>Goal</u>: Random ***sparse*** tensor of size n^d such that every element has an expected value of 1

1. Choose $s \ll n^d$ random tensor entries (with replacement)
2. Define random tensor with up to s nonzeros as follows

$$\tilde{w}_i = \frac{\#\text{ times } i \text{ sampled}}{s} \cdot n^d$$

Theory

Claim: $\mathbb{E}[\tilde{\mathcal{W}}] = \mathbf{1}$

Proof: $$\mathbb{E}(\tilde{w}_i) = \frac{\mathbb{E}(\#\text{ times } i \text{ sampled})}{s} \cdot n^d = \frac{s \cdot \frac{1}{n^d}}{s} \cdot n^d = 1$$

Benefits of sampling multiple elements…

- Lower variance in stochastic gradient
- Gradient = $O(rds)$ work in line with Update = $O(rdn)$ work

Downside…

- If data tensor is sparse, few entries corresponding to nonzeros will be chosen

Stratified Sampling Decreases Variance

For very sparse tensors, the likelihood of getting a nonzero is exceedingly small….

Results by Needell, Srebro, and Ward (2013) argue for biasing the sampling toward functionals with higher Lipschitz smoothness constants

Consider Poisson loss…

$$f(m;x) = m - x \log m$$

$$f'(m;0) = 1 \quad \Rightarrow \quad L = 0$$

$$f'(m;1) = 1 - 1/m \quad \Rightarrow \quad L \text{ unbounded as } m \downarrow 0$$

Stochastic Weight Matrix: Stratified Sample

Goal: Random ***sparse*** tensor of size n^d such that every element has an expected value of 1

1. For each partition ℓ
   a) Choose $s_\ell \ll |\Omega_\ell|$ random tensor entries from Ω_ℓ (with replacement)
   b) For each $i \in \Omega_\ell$, the corresponding entry of the weight tensor is

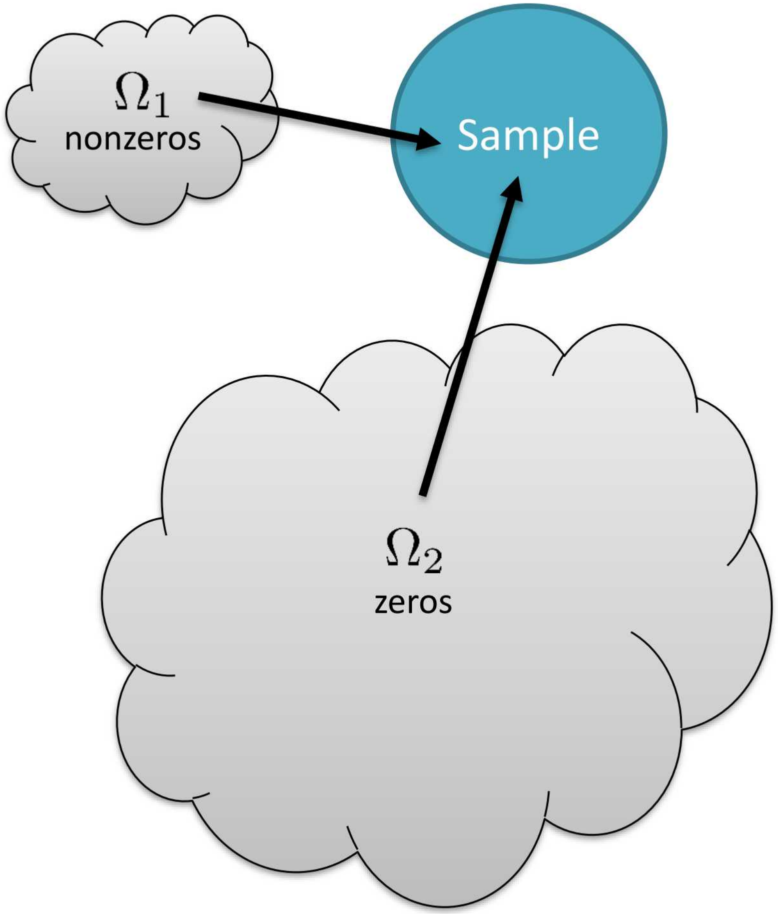
$$\tilde{w}_i = \#\text{ times } i \text{ sampled} \cdot \frac{|\Omega_\ell|}{s_\ell}$$

Theory

Claim: $\mathbb{E}[\tilde{\mathbf{W}}] = \mathbf{1}$

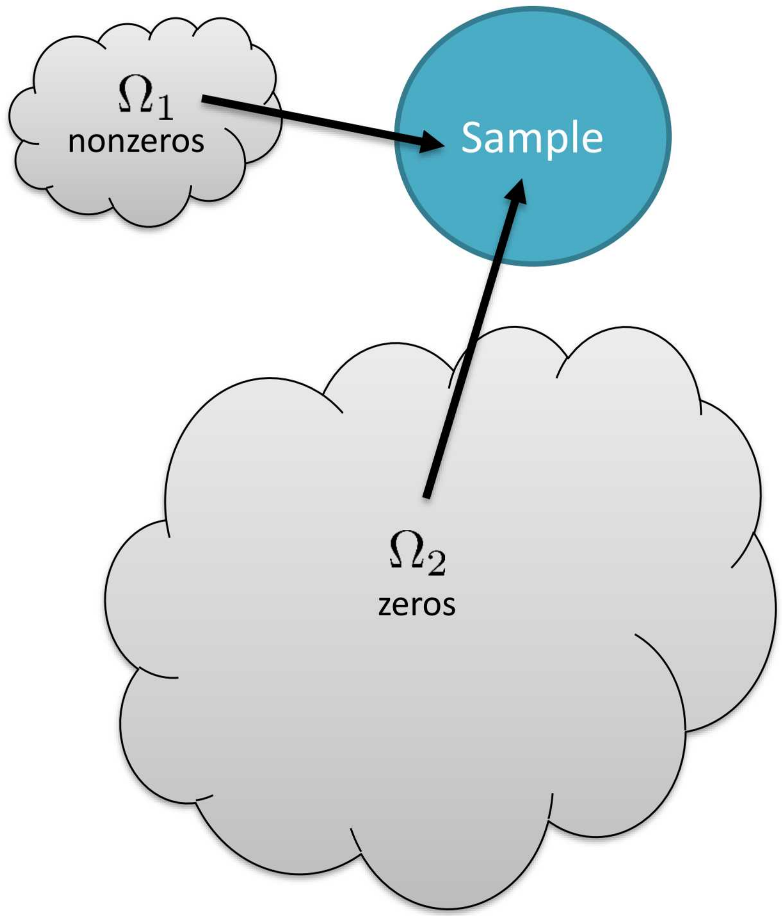
Proof: $\mathbb{E}(\tilde{w}_i) = \mathbb{E}[\#\text{ times } i \text{ sampled}] \cdot \frac{|\Omega_\ell|}{s_\ell}$

$$= s_\ell \cdot \frac{1}{|\Omega_\ell|} \cdot \frac{|\Omega_\ell|}{s_\ell} = 1$$

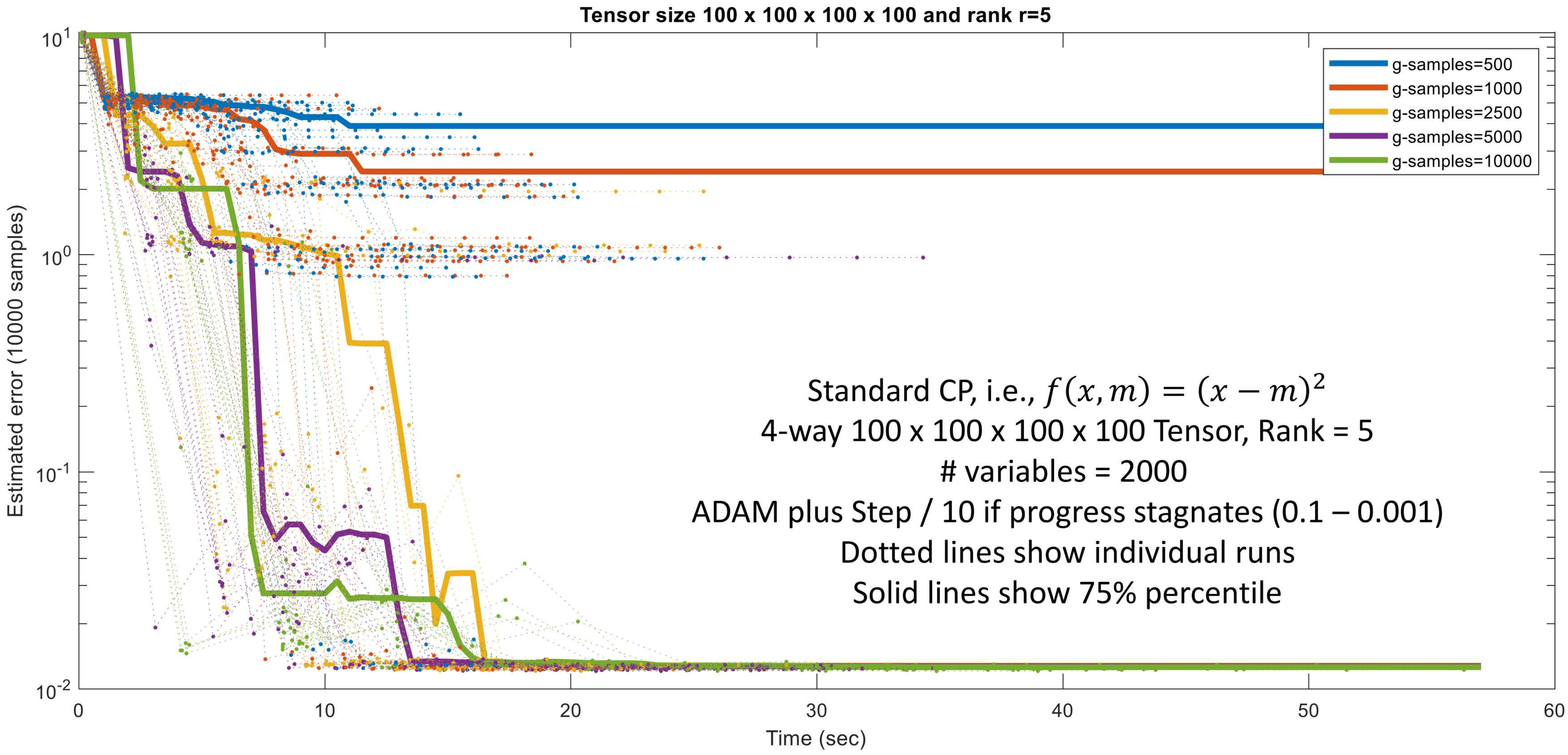


Rejection Sampling for Zeros in Sparse Tensor

- Most SGD-based tensor methods ignore zeros
  - Treating them as "unknown"
  - In contrast, we include them
- Zeros not stored explicitly
  - Generate candidate random index
  - Reject if in list of nonzeros
  - Can be expensive!



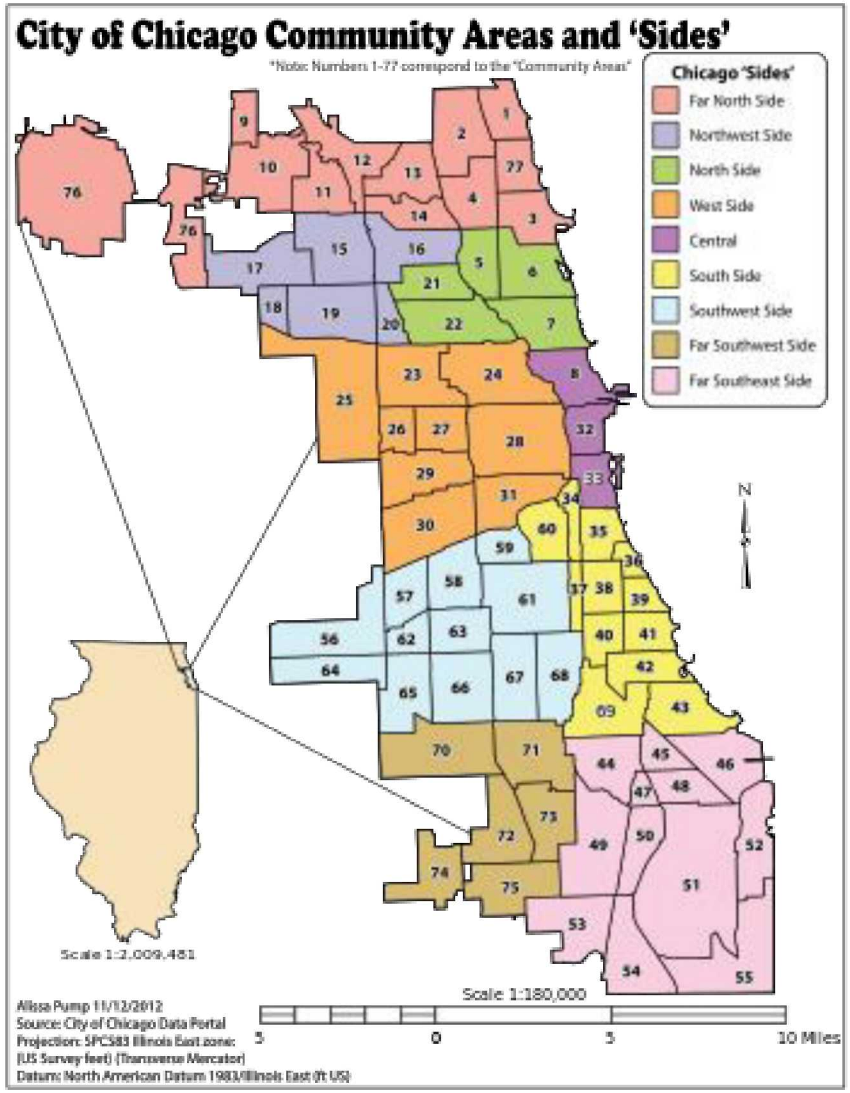
Dependence on Sample Size



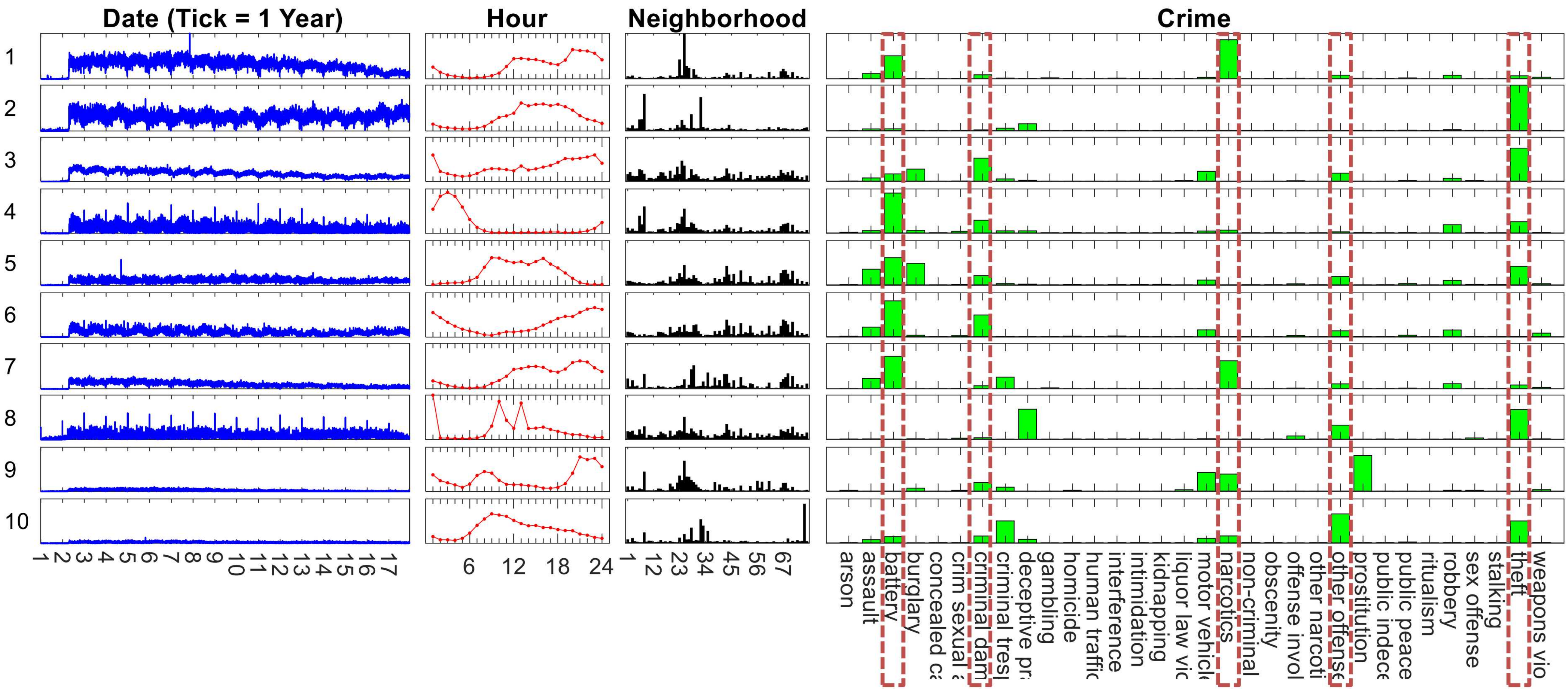
Chicago Crime Data

- 4-way count tensor
  - 6,186 Days
  - 24 Hours of the Day
  - 77 Community Areas
  - 32 Crime Types
- Non-zeros: 5,330,673
  - Storage: 0.21GB for sparse tensor
- Distribution of entries
  - 0: 98.54%
  - 1: 1.33%
  - $\geq$ 2: 0.12%
- Using binary version (every nonzero changed to 1)
- Obtained from FROSTT (http://frostt.io/tensors/chicago-crime/)
- Data originally from Chicago Data Portal (https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-present/ijzp-q8t2)

GCP-Binary
Rank = 10
$s = 30{,}000$
$$f(x, m) = \log(m+1) - x\log(m)$$
1100 sec. (versus 65 sec. CP-ALS)

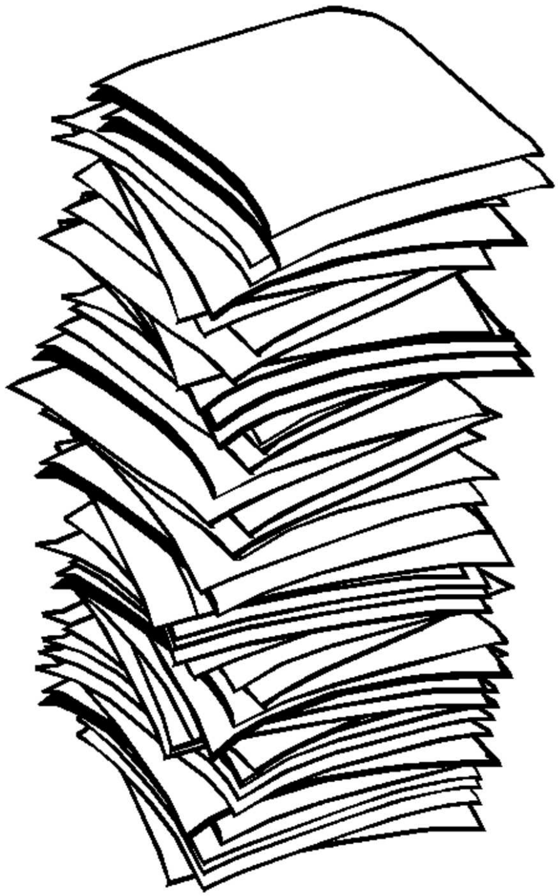


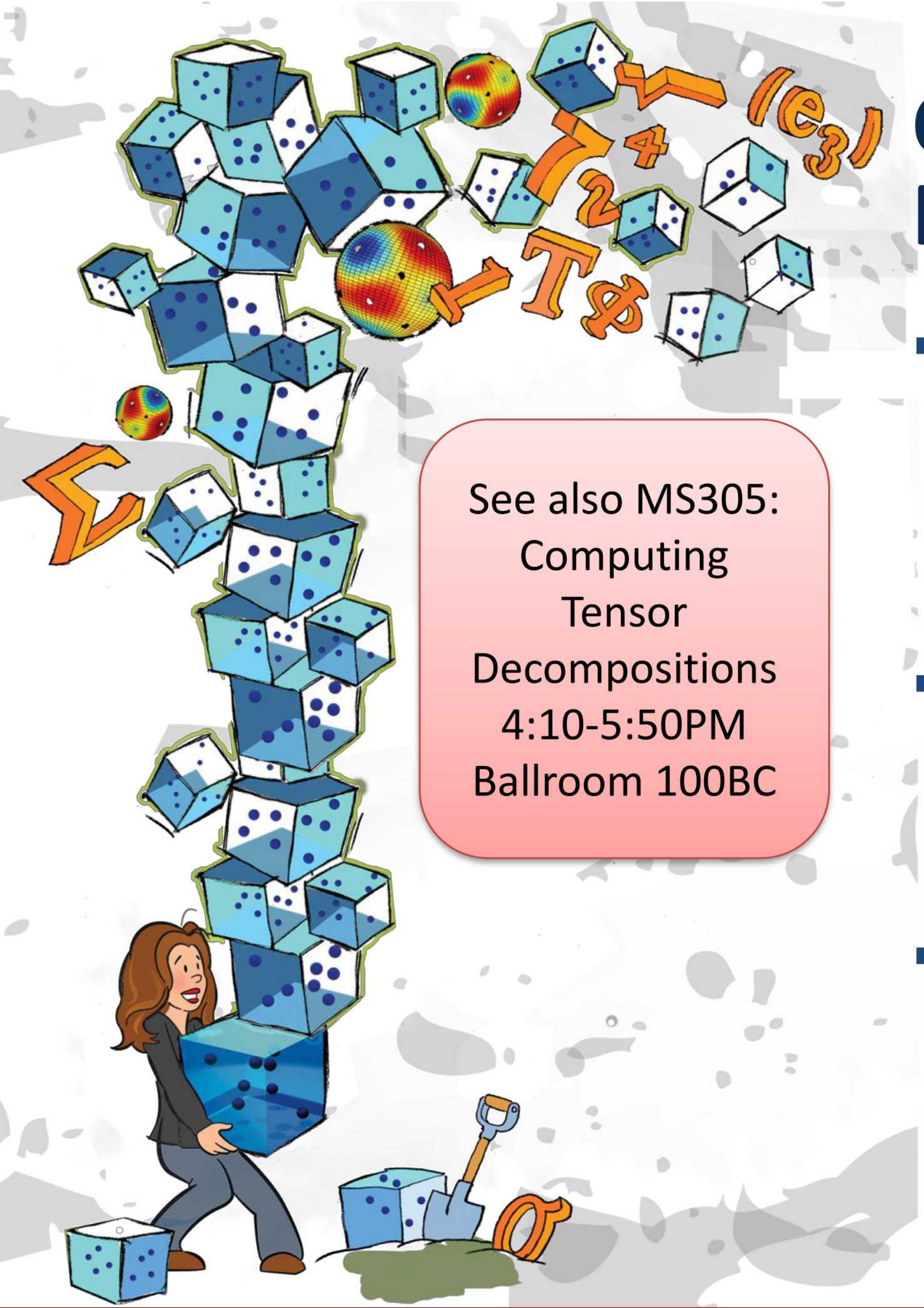
Application to Sparse Crime Binary Tensor



Related Work

- SGD for Matrix Decomposition
  - Gemulla et al., KDD'11 – Distributed SGD (DSGD) method: Partition matrix into blocks, run parallel SGD on independent blocks, cycling through the blocks in a way that ensures correctness. ***Only uses nonzero entries.***
  - Zhuang et al., RecSys'13 – Fast Parallel SGD (FPSGD) method: Matrix factorization in shared memory environment. No theoretical analysis. ***Only uses nonzero entries.***
- SGD for Tensor Decomposition
  - Mardini et al. (2015) – OnlineCP uses SGD for tensors that are streaming, one slice at a time
  - Maehara et al., AAAI-16 – Tensor can be written as sum or average of a finite(?) number of tensors. Proposes SGD plus several variations
  - Ge et al. [6] consider SGD for symmetric tensor decomposition
- Tensor Sketching
  - Acar et al. [1] show that, for dense tensors, it is heuristically possible to recover a full tensor decomposition with only a sketch of the data
  - Jain and Oh [11] and Bhojanapalli and Sanghavi [3] more formally prove under what conditions sketching works, albeit with a focus on orthogonal symmetric tensor decomposition





See also MS305: Computing Tensor Decompositions 4:10-5:50PM Ballroom 100BC

Conclusions, Future Work, References

- Conclusions
  - GCP not amenable to scaling because gradient "dense"
  - Developed GCP stochastic gradient
  - Use stratified sampling for sparse tensors
  - Recommend #samples = $O(rnd)$
- Future work
  - Release for MATLAB Tensor Toolbox
  - Parallel implementation (with Eric Phipps – GenTen)
  - Distributed implementation (with Karen Devine)
- References

  - D. Hong, T. G. Kolda, J. A. Duersch. **Generalized Canonical Polyadic Tensor Decomposition**. arXiv:1808.07452, 2018
  - T. G. Kolda, D. Hong, J. A. Duersch. **Stochastic Optimization for Large-Scale Tensor Decomposition**, in preparation