SAND2020-2247C

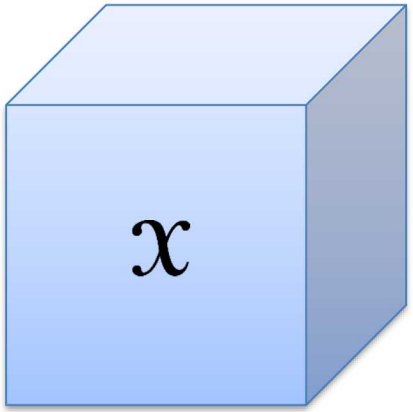
Illustration by Chris Brigman

Stochastic Gradients for Large-Scale Tensor Decomposition

Tamara G. Kolda
Sandia National Labs, Livermore, CA
Joint work with David Hong (U Penn)

Supported by the DOE Office of Science Advanced Scientific Computing Research (ASCR) Applied Mathematics program and Sandia's Laboratory Directed Research and Development (LDRD program). 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.

Tensors are Multi-dimensional Arrays



3-way tensor

d = **order** of the tensor (the number of ways or modes)

n_k = **dimension** of mode k, for $k = 1, 2, \ldots, d$

For expositional simplicity: $n = n_1 = n_2 \cdots = n_d$

n^d = number of entries for d-way tensor of dimension n

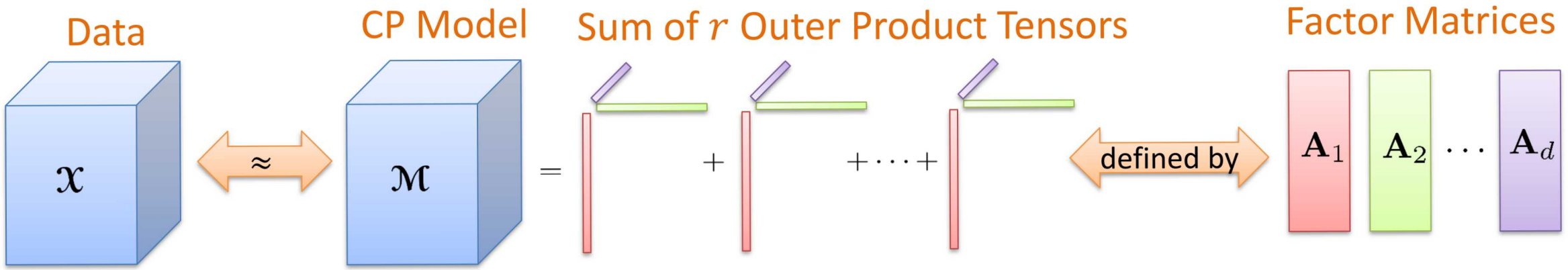
Curse of Dimensionality

Curse of notation…

$i \equiv (i_1, i_2, \ldots, i_d)$ = **index** into tensor, $i_k \in \{1, \ldots, n\}$ for $k = 1, 2, \ldots, d$

multi-index shorthand

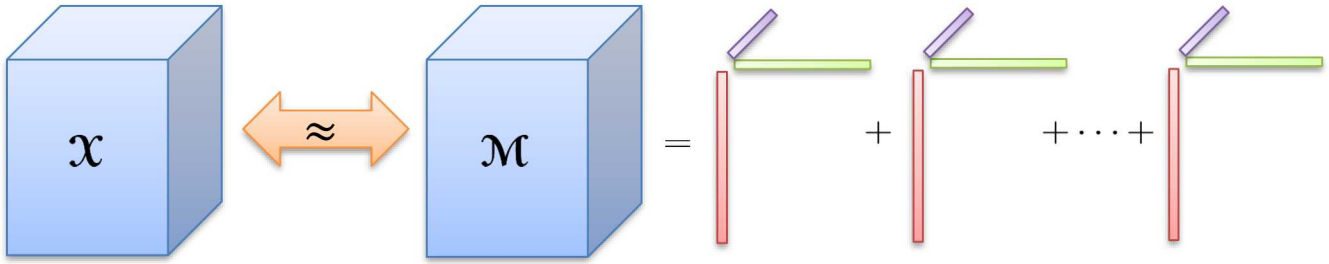
Canonical Polyadic (CP) Model Defined in Terms of Factor Matrices



$$\mathcal{X} \in \mathbb{R}^{n \times n \times \cdots \times n} \qquad \mathcal{M} = [\![\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_d]\!] \in \mathbb{R}^{n \times n \times \cdots \times n} \qquad \mathbf{A}_k \in \mathbb{R}^{n \times r}$$

$$x_i = x(i_1, i_2, \ldots, i_d) \qquad m_i = m(i_1, i_2, \ldots, i_d) = \sum_{j=1}^{r} \prod_{k=1}^{d} a_k(i_k, j) \qquad \operatorname{rank}(\mathcal{M}) \leq r$$

Generalized CP (GCP) Tensor Decomposition Allows Flexible Loss Function



Generalized CP (GCP)

$$\min_{\mathbf{A}_1,\ldots,\mathbf{A}_d} \quad F(\mathcal{X},\mathcal{M}) = \sum_{i=1}^{n^d} f(x_i, m_i)$$

$$\text{s.t.} \quad \mathcal{M} = [\![\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_d]\!]$$

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

Example Loss Functions

Poisson ($x \in \mathbb{N}, m > 0$)

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

Bernoulli ($x \in \{0,1\}, m > 0$)

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

Gamma ($x > 0, m > 0$)

$$f(x,m) = x/m + \log m$$

β-divergence ($x > 0, m > 0, \beta = \frac{1}{2}$)

$$f(x,m) = x/\sqrt{m} + \sqrt{m}$$

Hong, Kolda, Duersh, *SIREV* 2020 (arXiv:1808.07452)

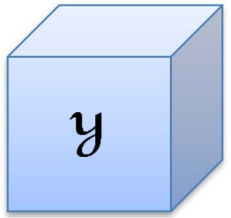
GCP Gradient Uses MTTKRP

Generalized CP (GCP)

$$\min_{\mathbf{A}_1,\ldots,\mathbf{A}_d} \quad F(\mathcal{X},\mathcal{M}) = \sum_{i=1}^{n^d} f(x_i, m_i)$$
$$\text{s.t.} \quad \mathcal{M} = [\![\mathbf{A}_1, \mathbf{A}_2, \ldots, \mathbf{A}_d]\!]$$
$$\mathbf{A}_k \in \mathbb{R}^{n\times r} \text{ for } k = 1,\ldots,d$$

Elementwise Partial Gradients

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



$$y_i = \frac{\partial f}{\partial m}(x_i, m_i)$$

Variables

$$\mathbf{a} = \begin{bmatrix} \text{vec}(\mathbf{A}_1) \\ \text{vec}(\mathbf{A}_2) \\ \vdots \\ \text{vec}(\mathbf{A}_d) \end{bmatrix} \qquad \mathbf{a} \in \mathbb{R}^{dnr}$$

Gradients

$$\mathbf{g} = \begin{bmatrix} \text{vec}(\mathbf{G}_1) \\ \text{vec}(\mathbf{G}_2) \\ \vdots \\ \text{vec}(\mathbf{G}_d) \end{bmatrix} \qquad \mathbf{g} \in \mathbb{R}^{dnr}$$

MTTKRP: Matricized Tensor Times Khatri-Rao Product

Mode-k Unfolding

$\mathbf{Y}_{(k)} \in \mathbb{R}^{n\times n^{d-1}}$

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

$$\mathbf{G}_k = \frac{\partial F}{\partial \mathbf{A}_k} \in \mathbb{R}^{n\times r}$$

Khatri-Rao Product $\mathbf{Z}_k = \mathbf{A}_d \odot \cdots \odot \mathbf{A}_{k+1} \odot \mathbf{A}_{k-1} \odot \cdots \odot \mathbf{A}_1 \in \mathbb{R}^{n^{d-1}\times r}$

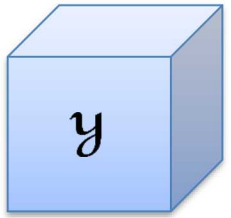
Hong, Kolda, Duersh, *SIREV* 2020 (arXiv:1808.07452)

Computing GCP Gradient is Expensive!

Computing elementwise partial gradients costs $o(n^d)$ even if data tensor is sparse!

Elementwise Partial Gradients

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



$$y_i = \frac{\partial f}{\partial m}(x_i, m_i)$$

MTTKRP with dense tensor costs $O(rn^d)$

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

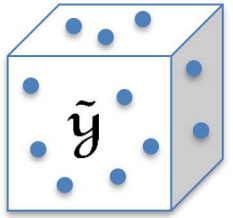
Khatri-Rao Product $\mathbf{Z}_k = \mathbf{A}_d \odot \cdots \odot \mathbf{A}_{k+1} \odot \mathbf{A}_{k-1} \odot \cdots \odot \mathbf{A}_1 \in \mathbb{R}^{n^{d-1} \times r}$

Stochastic Gradient Computed Cheaply

Choose a sparse elementwise gradient that is equal to the true gradient in expectation!

Elementwise Partial Gradients

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



$\mathbb{E}[\tilde{\mathcal{Y}}] = \mathcal{Y}$

$\text{nnz}(\tilde{\mathcal{Y}}) \leq s \ll n^d$

By linearity of expectation:

$\mathbb{E}[\tilde{\mathbf{G}}_k] = \mathbf{G}_k$

MTTKRP with sparse tensor costs $O(rs)$

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

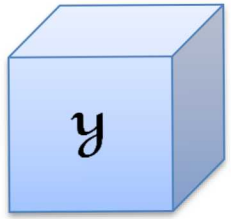
Khatri-Rao Product $\mathbf{Z}_k = \mathbf{A}_d \odot \cdots \odot \mathbf{A}_{k+1} \odot \mathbf{A}_{k-1} \odot \cdots \odot \mathbf{A}_1 \in \mathbb{R}^{n^{d-1} \times r}$

Related Work

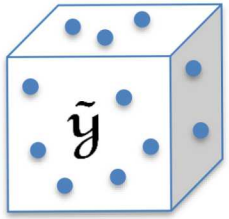
- *All related work in the context of standard CP, not GCP*
- **Missing data**; e.g., Acar, Dunlavy, Kolda, Morup, CILS 2011 (CP-WOPT)
- **Stochastic approaches**
  - Beutel, Talukdar, Kumar, Faloutsos, Papalexakis, Xing, SDM 2014 – SGD for standard CP for $d = 3$ and $s = 1$ (FlexiFact)
  - Vervliet and De Lathauwer, IEEE TSP 2016 – Block sampling (rather than elements)
- **Matrix sketching for least squares problem**
  - Battaglino, Ballard, Kolda, SIMAX 2018 (CP-ARLS aka CPRAND)
  - Cheng, Peng, Perros, Liu, NIPS 2016 (SPALS)
- **Random projections**
  - Papalexakis, Faloutsos, Sidiropoulos, ECML PKDD 2012 (ParCube)
  - Sidiropoulos, Papalexakis, Faloutsos, ICASSP 2014 (PARACOMP)
  - Zhou, Cichocki, Xie, arXiv, 2014
- **Streaming**
  - Using SGD, e.g. Maehara, Hayashi, Lawarabayashi, AAAI 2016
  - And other approaches too numerous to list here but starting with Nion and Sidiropoulos, IEEE TSP 2009
- **Tensor completion**/recommender systems (most data missing), e.g., Smith, Park, Karypis, SC'16

Unique to This Work
GCP for Large Scale,
Sampling Strategies,
Sparse (rather than Scarce)

Sampling for Stochastic Gradients



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

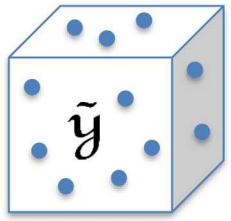


$$\tilde{\mathcal{Y}} \in \mathbb{R}^{n \times n \times \cdots \times n}$$

$$\mathbb{E}[\tilde{\mathcal{Y}}] = \mathcal{Y}$$

$$\text{nnz}(\tilde{\mathcal{Y}}) \leq s \ll n^d$$

Uniform Sampling



$$\mathbb{E}[\tilde{\mathcal{Y}}] = \mathcal{Y}$$

$$\text{nnz}(\tilde{\mathcal{Y}}) \leq s \ll n^d$$

Sample $s \ll n^d$ random tensor entries (with replacement)

$$\tilde{s}_i = \#\text{ times } i \text{ sampled}$$

$$\tilde{y}_i = \tilde{s}_i \cdot \frac{n^d}{s} \cdot y_i$$

Upside…

- Very efficient

Downside…

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

Theory

Claim: $\mathbb{E}[\tilde{\mathcal{Y}}] = \mathcal{Y}$

Proof: $\mathbb{E}[\tilde{s}_i] = \frac{s}{n^d}$

$$\mathbb{E}[\tilde{y}_i] = \mathbb{E}[\tilde{s}_i] \cdot \frac{n^d}{s} \cdot y_i = y_i$$

Nonzeros Needed to Reduce Variance in Stochastic Gradient

Biased sampling toward *functionals* with higher Lipschitz smoothness constants reduces variance in stochastic gradient (Needell, Srebro, & Ward, 2013; Gopal, 2016)

For tensors, *functionals equate to tensor entries, i.e.,* $f_i = f(x_i, m_i)$

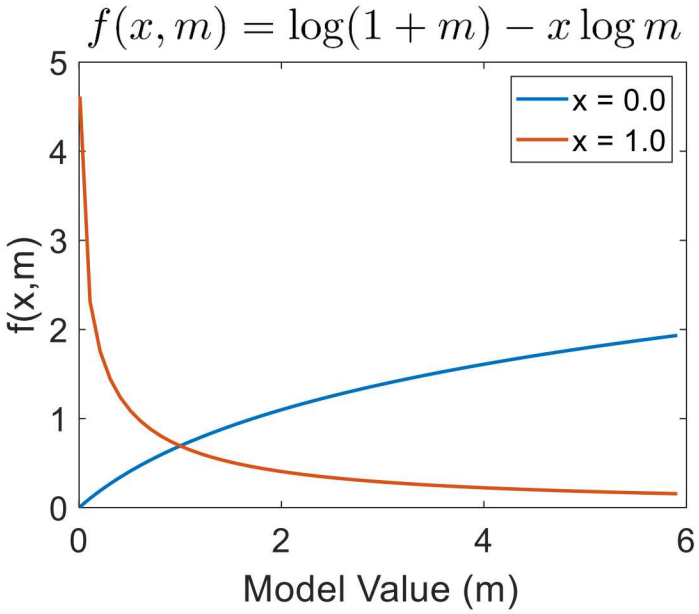
Consider Bernoulli with odds link: $f(x,m) = \log(1+m) - x \log m$

$$\frac{\partial f}{\partial m}(0,m) = \frac{1}{m+1} \quad \Rightarrow \quad L \leq 1$$

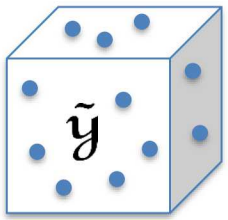
$$\frac{\partial f}{\partial m}(1,m) = \frac{-1}{m^2+m} \quad \Rightarrow \quad L \text{ unbounded as } m \downarrow 0$$

Need to bias sampling to select more nonzeros in sparse tensors

Sparse Binary (0/1) Tensor



Stratified Zero/Nonzero Sampling



$$\mathbb{E}[\tilde{\mathbf{y}}] = \mathbf{y}$$

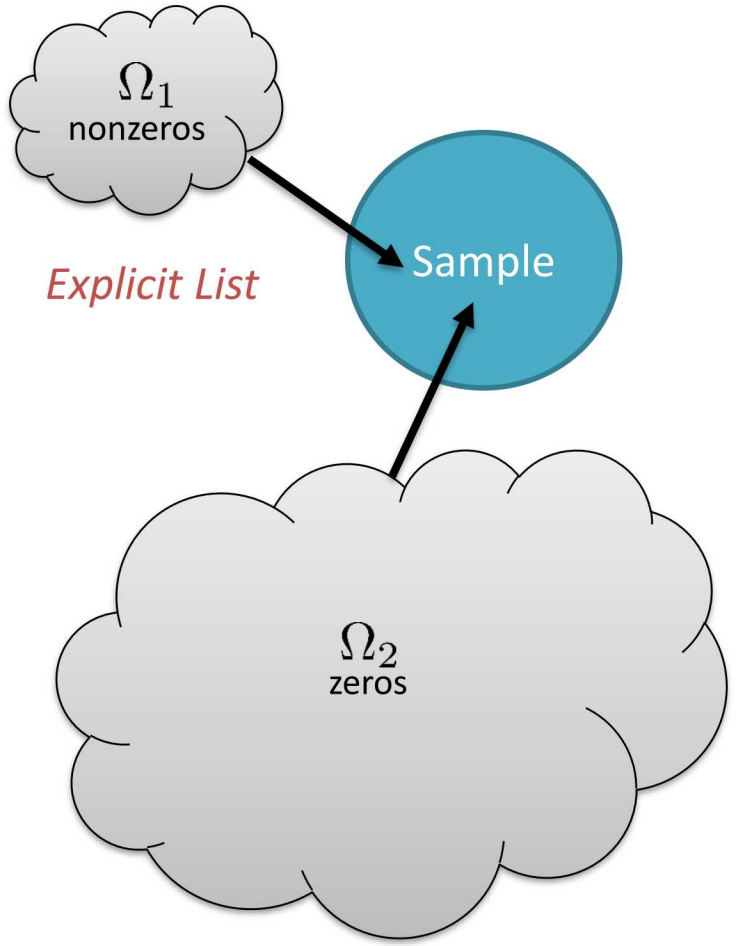
$$\text{nnz}(\tilde{\mathbf{y}}) \leq s \ll n^d$$

Sample p nonzeros and q zeros.

$\tilde{p}_i = \#$ times nonzero i sampled $\quad \eta = \#$ nonzeros

$\tilde{q}_i = \#$ times zero i sampled $\quad \zeta = \#$ zeros

$$\tilde{y}_i = \left(\tilde{p}_i \cdot \frac{\eta}{p} + \tilde{q}_i \cdot \frac{\zeta}{q} \right) \cdot y_i$$



Theory

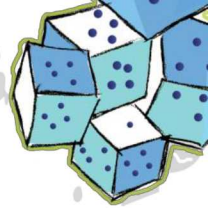
Claim: $\mathbb{E}[\tilde{\mathbf{y}}] = \mathbf{y}$

Proof: $\mathbb{E}[\tilde{p}_i] = \frac{p}{\eta}, \ \mathbb{E}[\tilde{q}_i] = \frac{q}{\zeta}$

$$x_i = 1 \Rightarrow \mathbb{E}[\tilde{y}_i] = \mathbb{E}[\tilde{p}_i] \cdot \frac{\eta}{p} \cdot y_i = y_i$$

$$x_i = 0 \Rightarrow \mathbb{E}[\tilde{y}_i] = \mathbb{E}[\tilde{q}_i] \cdot \frac{\zeta}{q} \cdot y_i = y_i$$

Difficulty in Rejection Sampling for Zeros



Example:

- 2500 x 2500 x 2500 tensor
- 100,000 nonzeros (0.000640% sparse)
- Checking list of 3750 potential zeros

Linear indices can only be used for smaller tensors

Method	Time (seconds)	Speedup
MATLAB ismember with multi-indices	3.92	1.0
MATLAB ismember with linear indices	0.47	8.3
MATLAB built-in _ismemberhelper with linear indices	0.17	23.5
MATLAB builit-in ismembc with linear indices	0.11	34.3
MATLAB containers.Map with linear indices	0.51	7.7
MATLAB containers.Map with string multi-indices	5.94	0.7

Semi-Stratified Zero/Nonzero Sampling

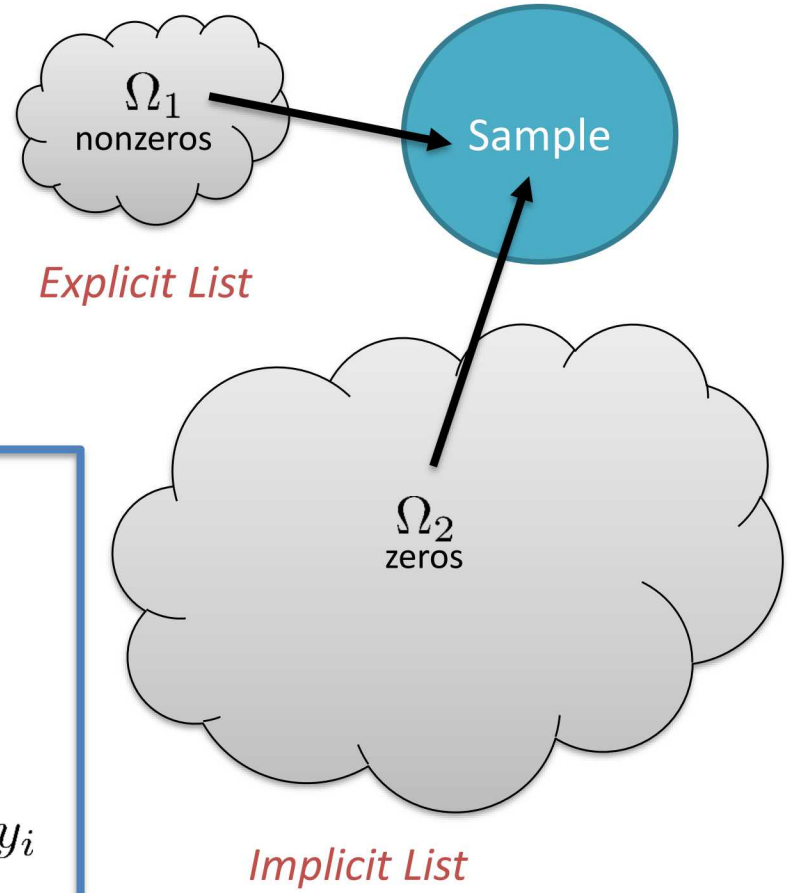
Idea: Sample "assumed zeros" from all indices and *correct* in nonzero samples.

Sample p nonzeros and q **assumed** zeros.

$\tilde{p}_i = \#$ times nonzero i sampled $\quad \eta = \#$ nonzeros

$\tilde{q}_i = \#$ times "zero" i sampled $\quad \zeta = \#$ zeros

$$\tilde{y}_i = \tilde{p}_i \cdot \frac{\eta}{p} \cdot (y_i - c_i) + \tilde{q}_i \cdot \frac{(\eta + \zeta)}{q} \cdot c_i \text{ with } c_i \equiv \frac{\partial f}{\partial m}(0, m_i)$$



Theory

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

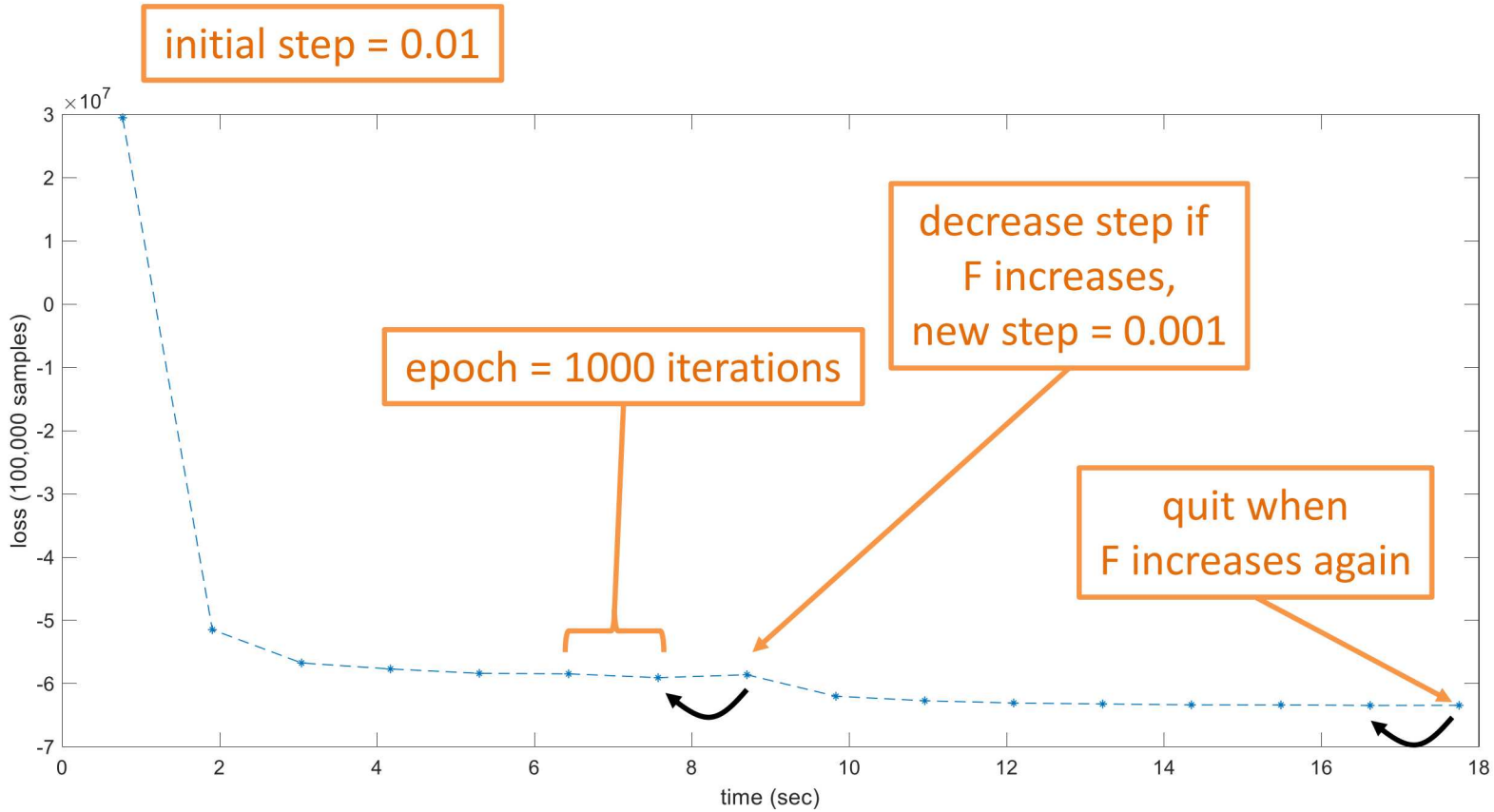
Proof: $\mathbb{E}[\tilde{p}_i] = \frac{p}{\eta}, \ \mathbb{E}[\tilde{q}_i] = \frac{q}{(\zeta + \eta)}$

$$x_i = 0 \Rightarrow \mathbb{E}[\tilde{y}_i] = \mathbb{E}[\tilde{q}_i] \cdot \frac{(\eta + \zeta)}{q} \cdot y_i = y_i$$

$$x_i = 1 \Rightarrow \mathbb{E}[\tilde{y}_i] = \mathbb{E}[\tilde{p}_i] \cdot \frac{\eta}{p} \cdot (y_i - c_i) + \mathbb{E}[\tilde{q}_i] \cdot \frac{\eta + \zeta}{q} \cdot c_i = y_i$$

GCP with Stochastic Optimization

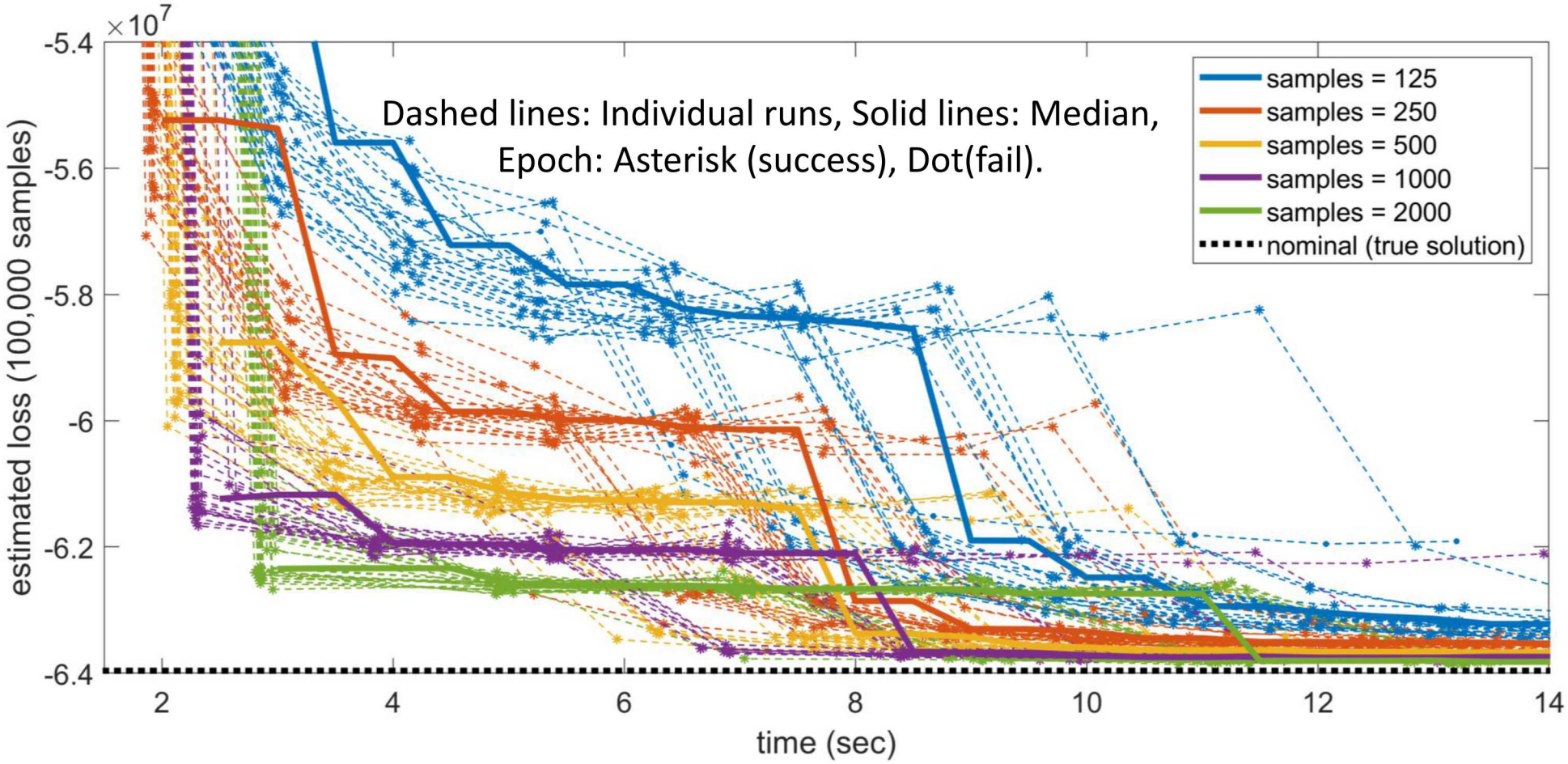
- Using Adam (Kingma & Ba, 2015)
  - Default parameters
  - Some tweaks for checking convergence



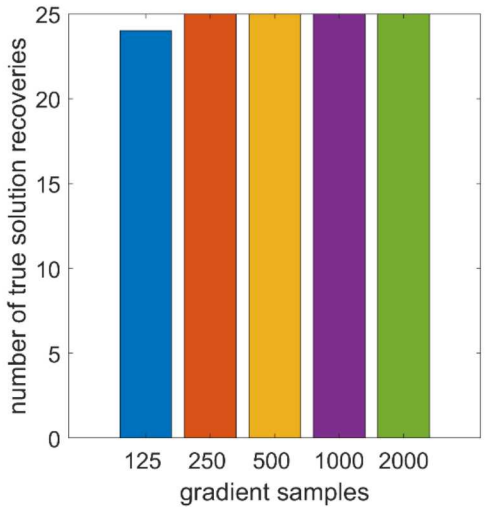
Roughly O(n) Samples Needed Per Stochastic Gradient

$200 \times 150 \times 100 \times 50$ Tensor (150M entries) with rank $r = 5$. Gamma loss: $f(x, m) = \frac{x}{m} + \log m$.

Running Adam with 25 random starts and varying numbers of samples.



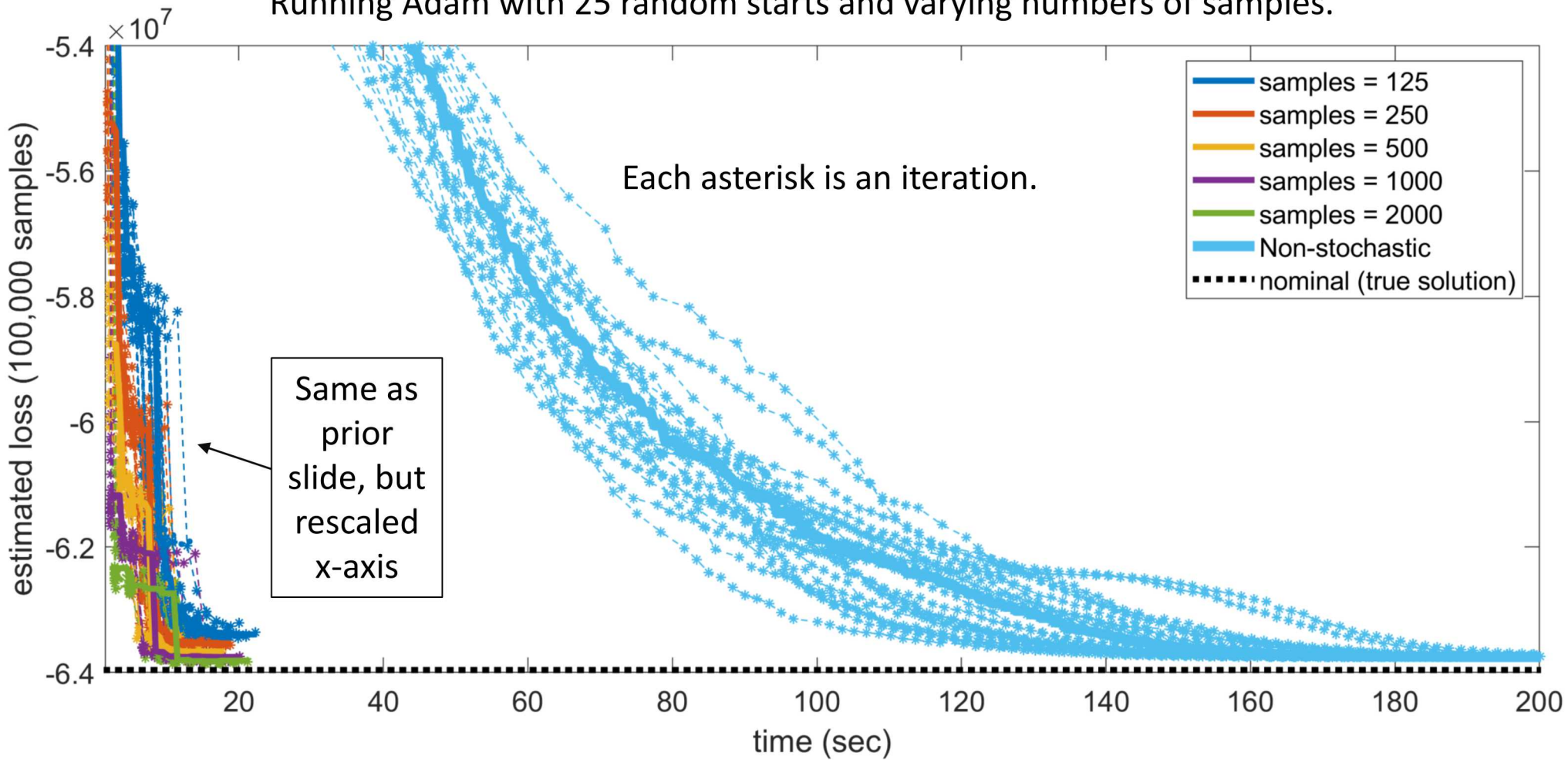
Recovery of Factor Matrices



Zooming Out: Stochastic Much Faster Than Non-Stochastic

$200 \times 150 \times 100 \times 50$ Tensor (150M entries) with rank $r = 5$. Gamma loss: $f(x, m) = \frac{x}{m} + \log m$.

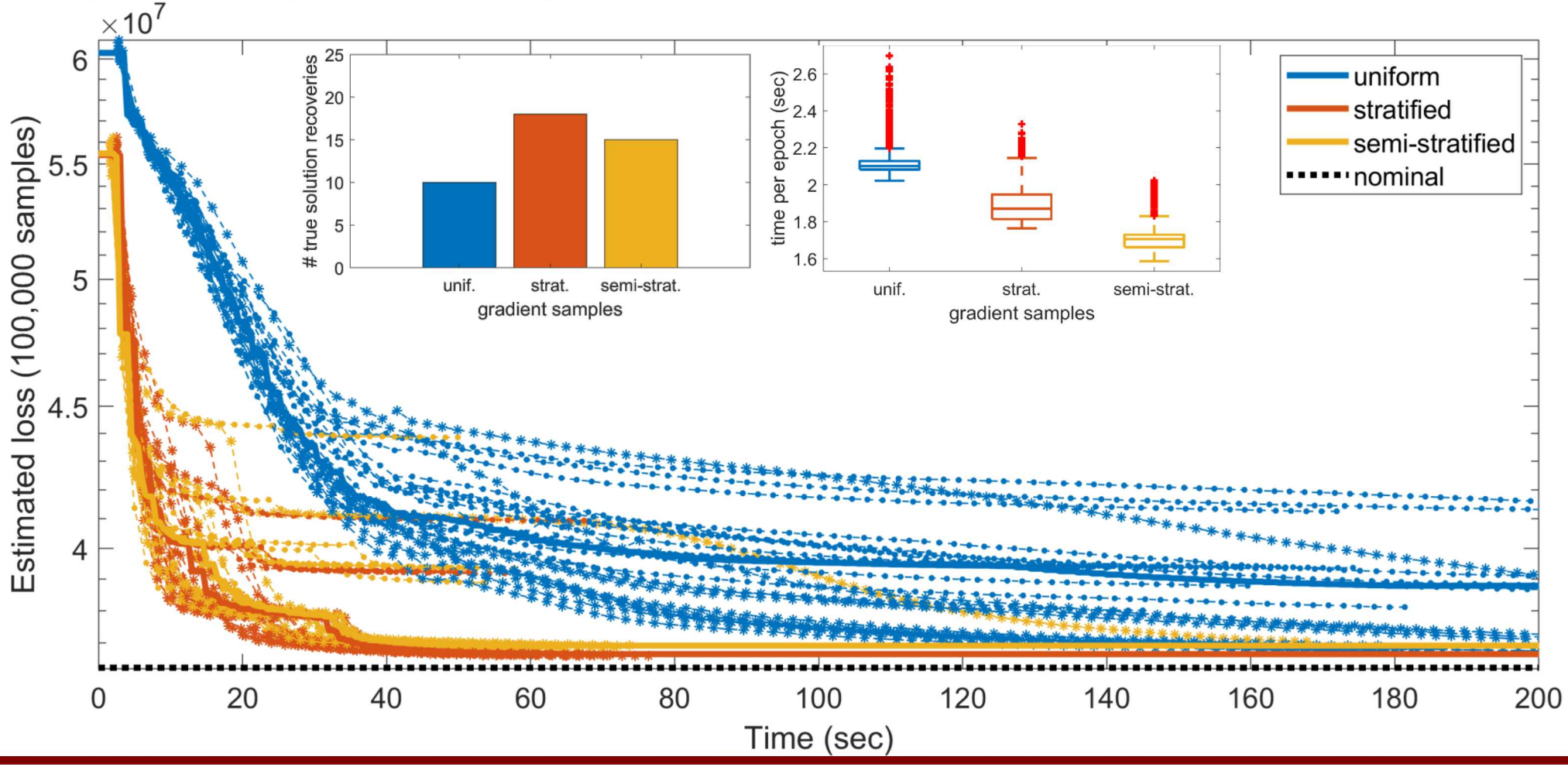
Running Adam with 25 random starts and varying numbers of samples.



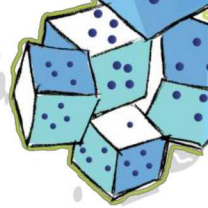
Uniform Sampling is Worse than Stratified for Sparse Tensors

$400 \times 300 \times 200 \times 100$ Tensor (2.4B entries, 9M nonzeros, 0.4% dense) with rank $r = 5$.

Bernoulli loss: $f(x, m) = \log(m + 1) - x \log m$. Using $s = 1000$ samples, evenly divided for stratified and semi-stratified.



Uniform Sampling is Worse than Stratified for Sparse Tensors



	Initial Guess (10^7)		Final Solution (10^6)	
Sampling Method	**Emp. Bias**	**Emp. Variance**	**Emp. Bias**	**Emp. Variance**
Uniform	10^6	10^15	10^7	10^18
Stratified	10^5	10^14	10^6	10^15
Semi-Stratified	10^5	10^14	10^6	10^15

$$\text{empirical bias} = \|\hat{\mathbf{g}} - \mathbf{g}\|_2 \quad \text{where} \quad \hat{\mathbf{g}} = \frac{1}{N}\sum_{\xi=1}^{N} \tilde{\mathbf{g}}_\xi, \quad \text{and}$$

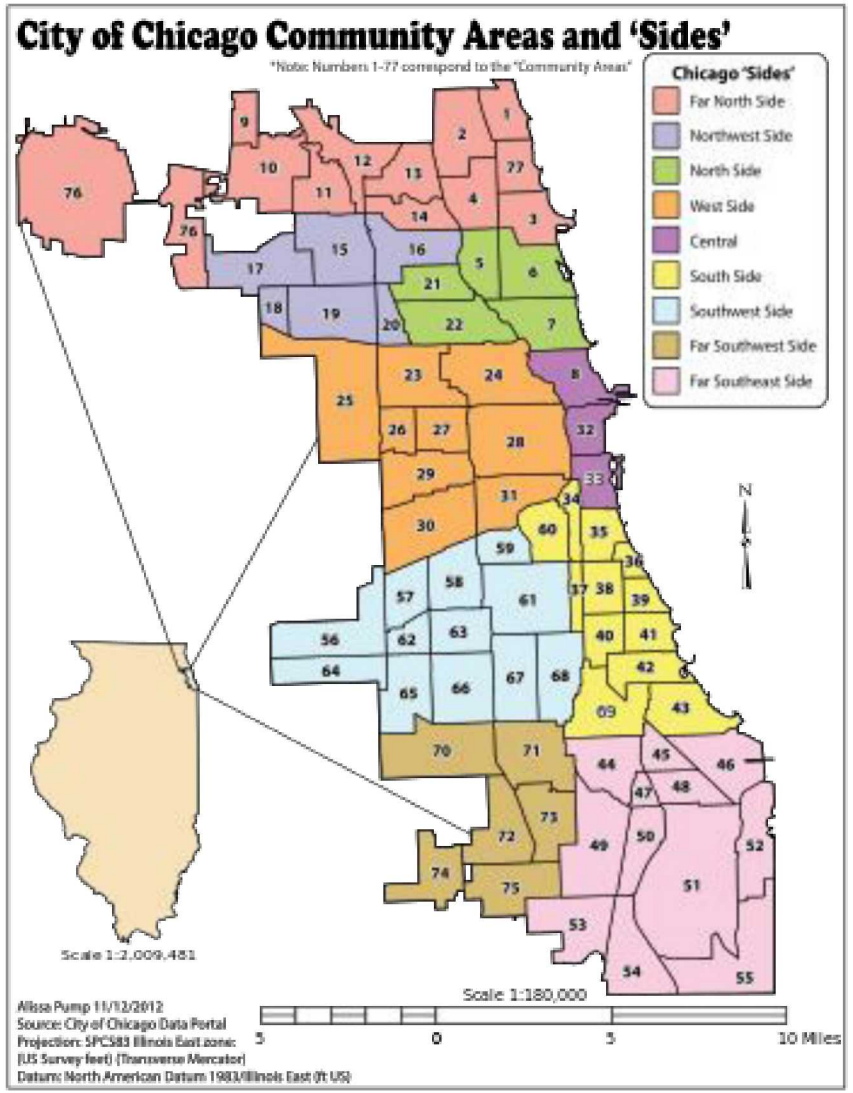
$$\text{empirical variance} = \text{trace}\left(\frac{1}{N}\sum_{\xi=1}^{N}(\tilde{\mathbf{g}}_\xi - \hat{\mathbf{g}})(\tilde{\mathbf{g}}_\xi - \hat{\mathbf{g}})^\intercal\right) = \frac{1}{N}\sum_{\xi=1}^{N}\|\tilde{\mathbf{g}}_\xi - \hat{\mathbf{g}}\|_2^2.$$

Chicago Crime Data

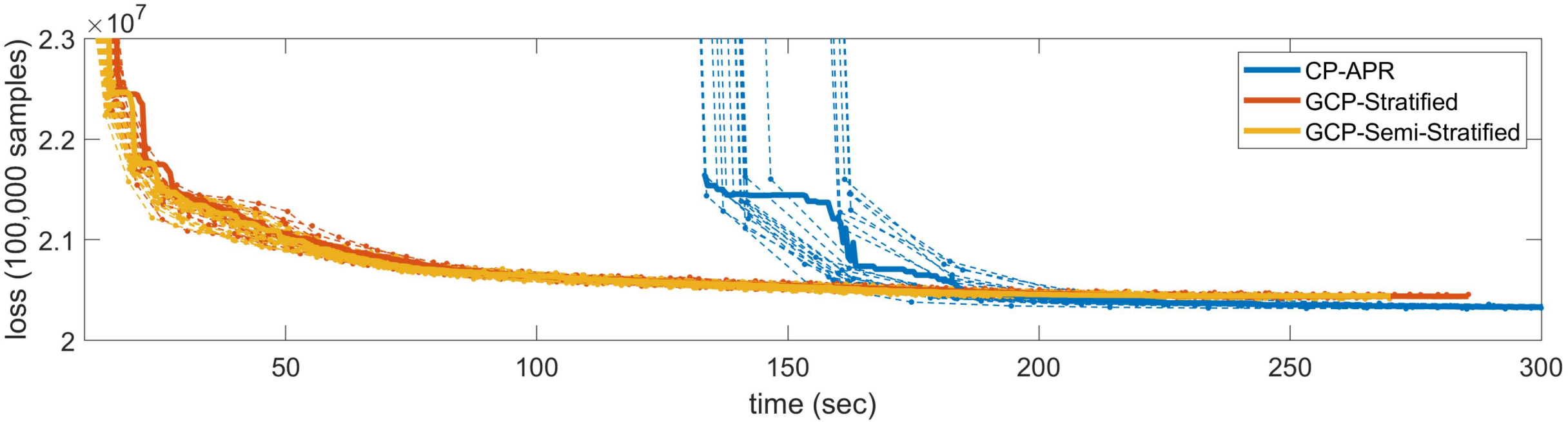
- 4-way count tensor
  - 6,186 Days
  - 24 Hours of the Day
  - 77 Community Areas
  - 32 Crime Types
- Non-zeros: 5,330,673
  - 0.21GB for sparse tensor
- Distribution of entries
  - 0: 98.54%
  - 1: 1.33%
  - ≥ 2: 0.12%
- Obtained from FROSTT (http://frostt.io/tensors/chicago-crime/)
- Data originally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

GCP-Count
Rank = 10
Samples $s = 6{,}319$

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

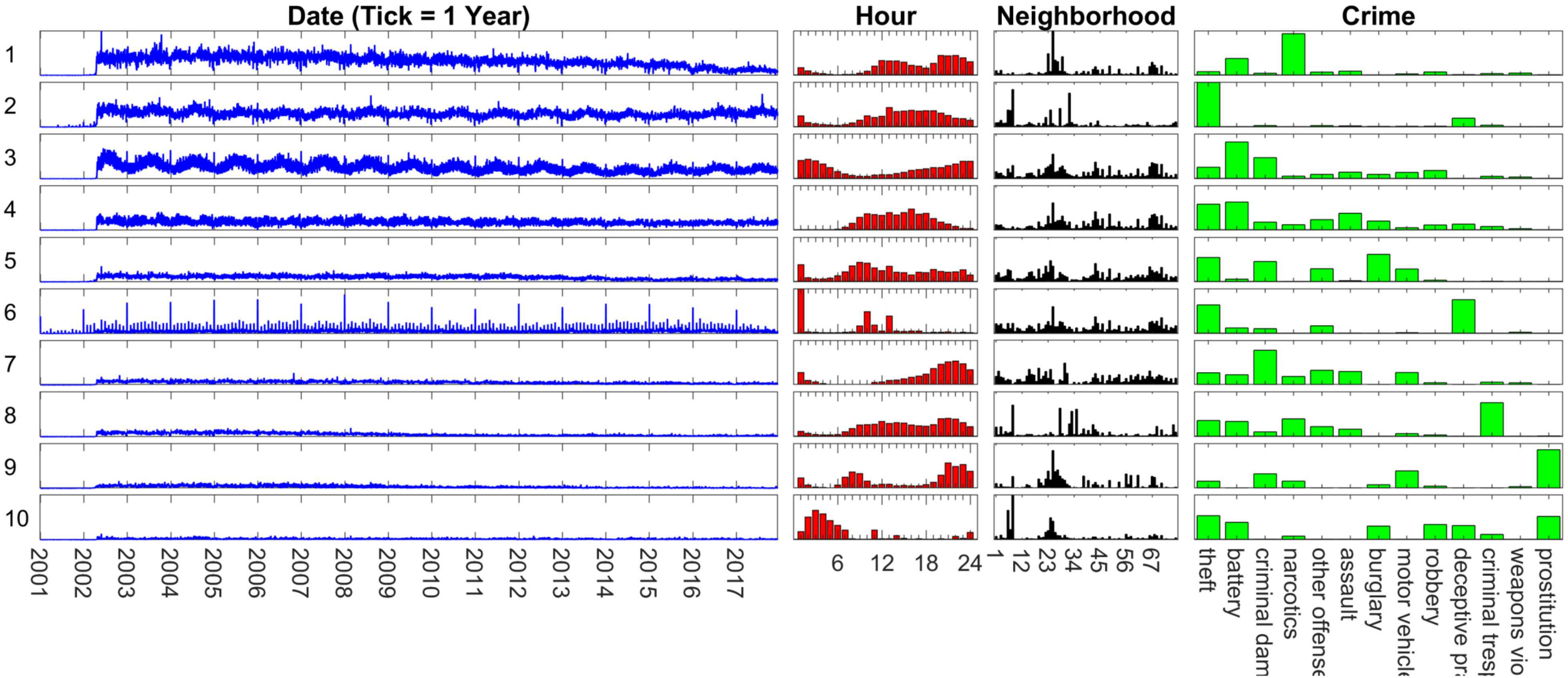


Comparison to CP-APR

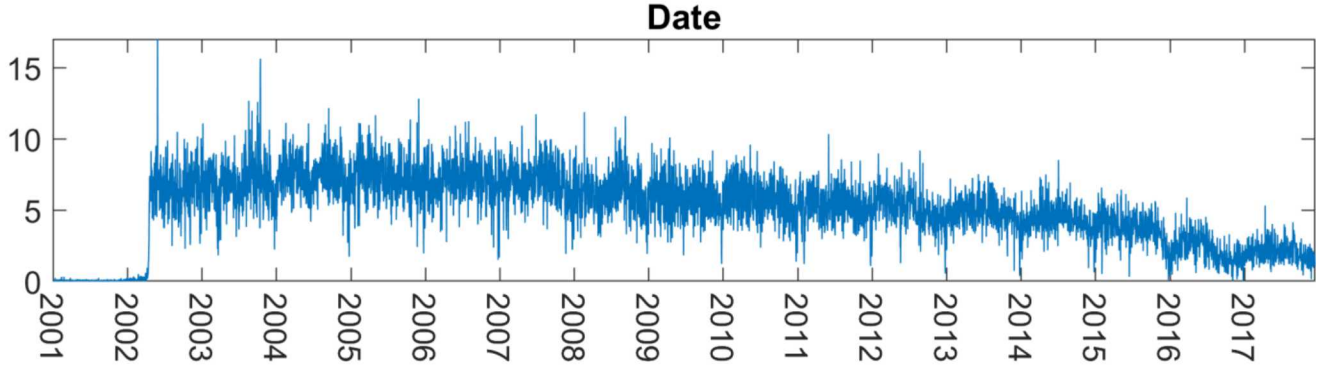


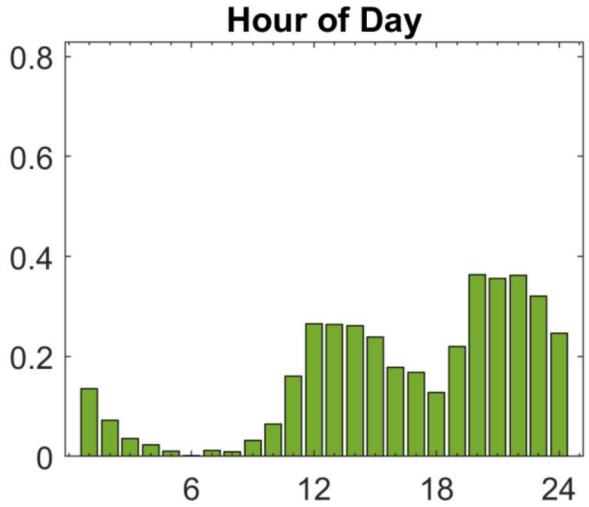
GCP with Poisson Loss compared to CP-APR using quasi-Newton

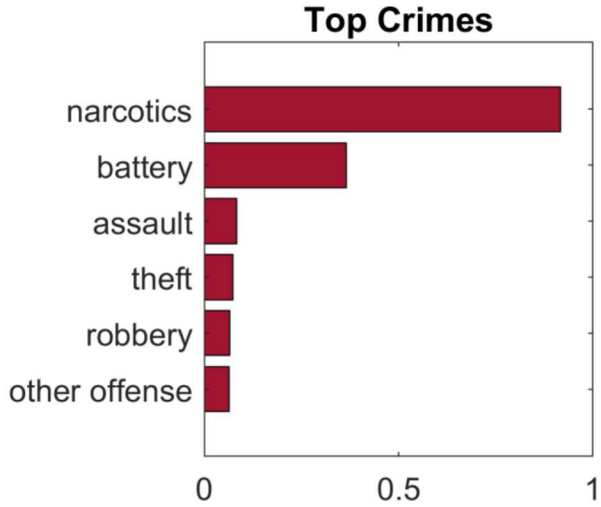
Application to Sparse Crime Binary Tensor (Semi-stratified results)

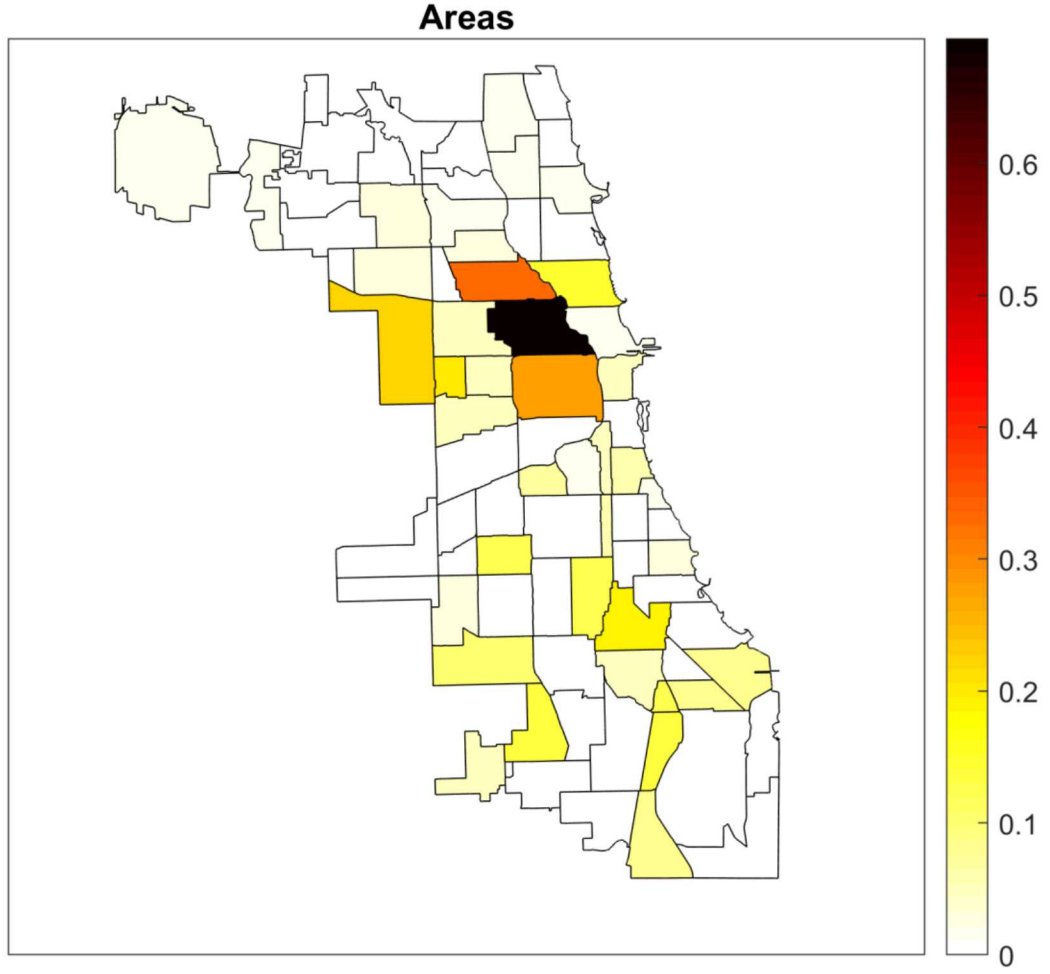


Component #1

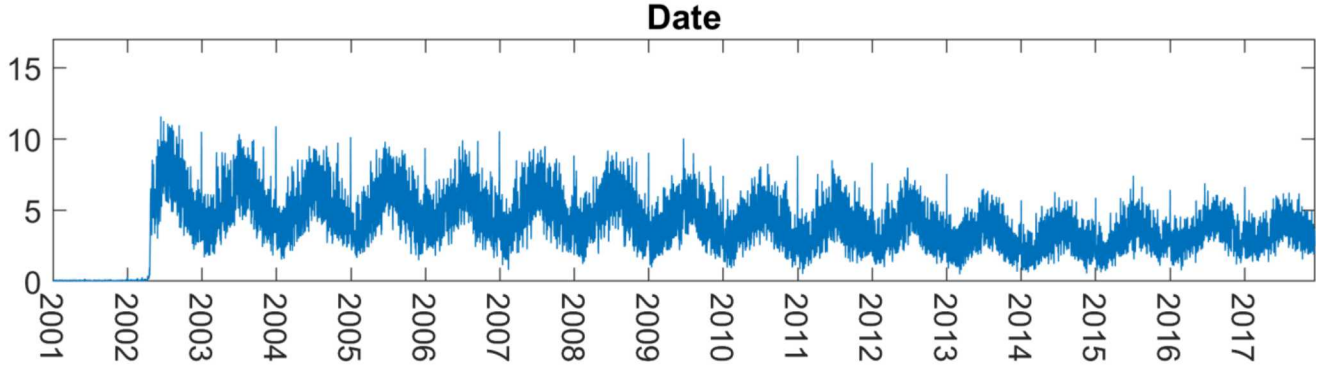

Date
15
10
5
0
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017

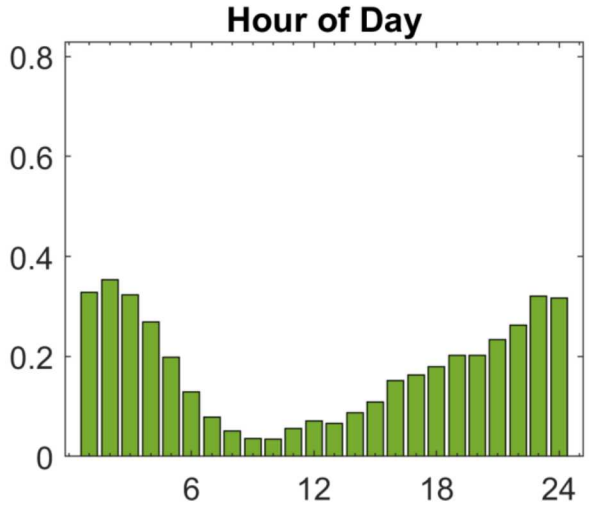

Hour of Day
0.8
0.6
0.4
0.2
0
6
12
18
24

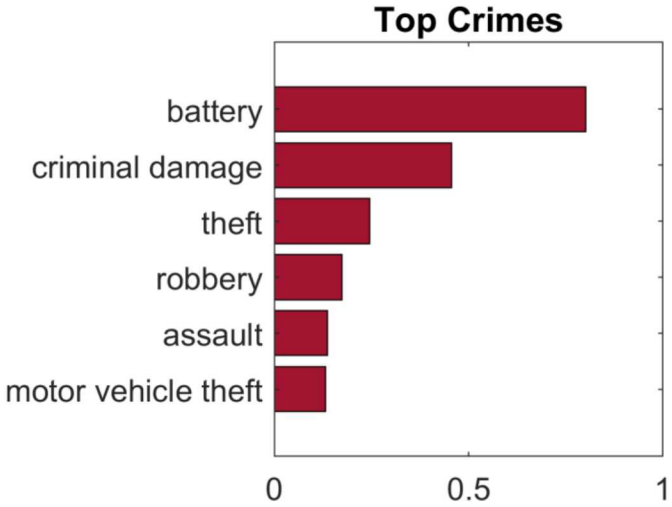

Top Crimes
narcotics
battery
assault
theft
robbery
other offense
0
0.5
1

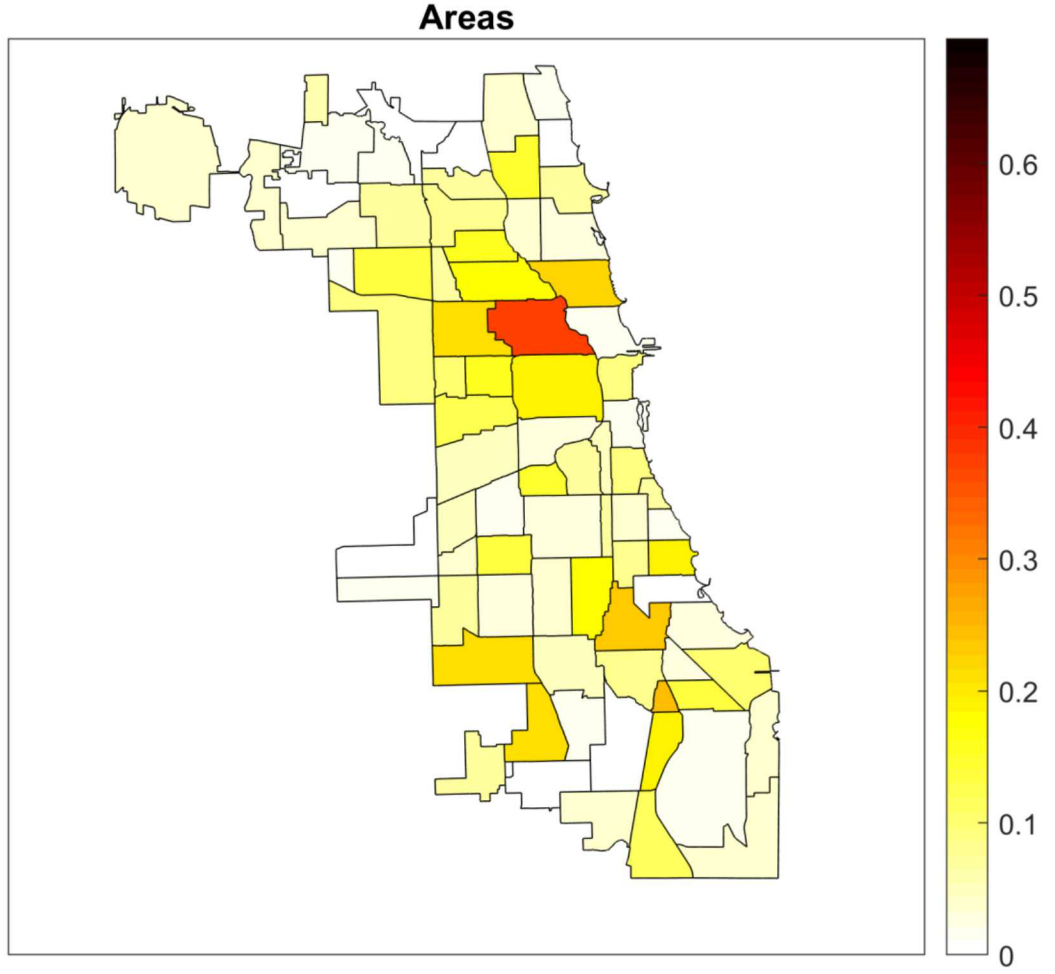

Areas
0.6
0.5
0.4
0.3
0.2
0.1
0

Component #3

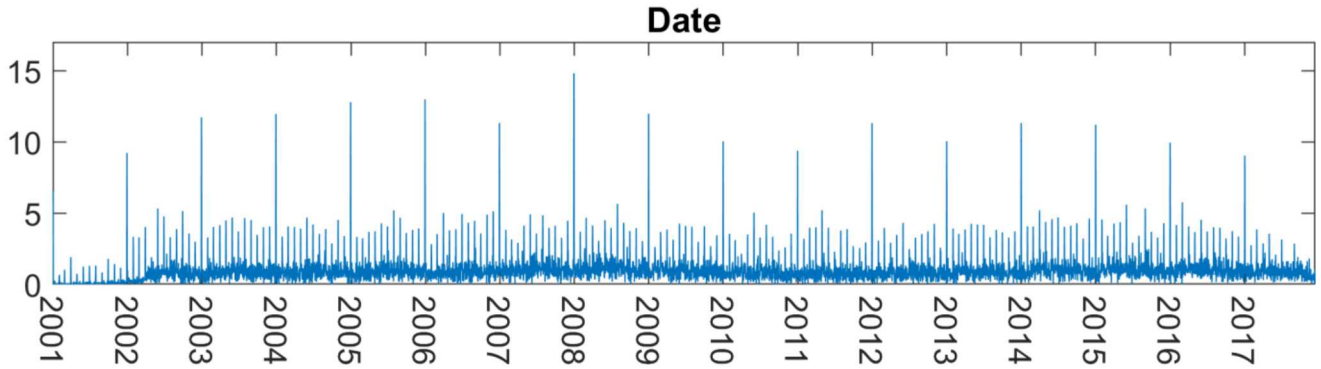

Date
15
10
5
0
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017

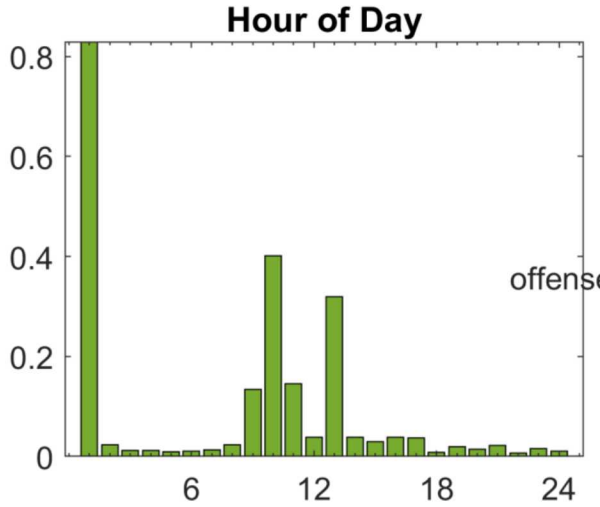

Hour of Day
0.8
0.6
0.4
0.2
0
6
12
18
24

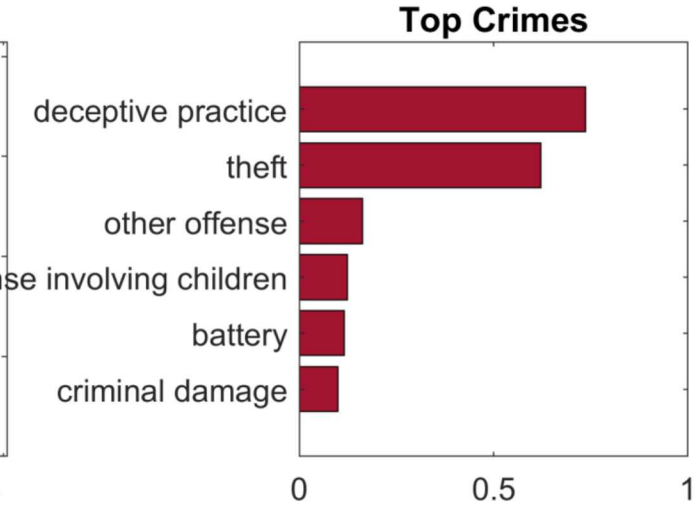

Top Crimes
battery
criminal damage
theft
robbery
assault
motor vehicle theft
0
0.5
1

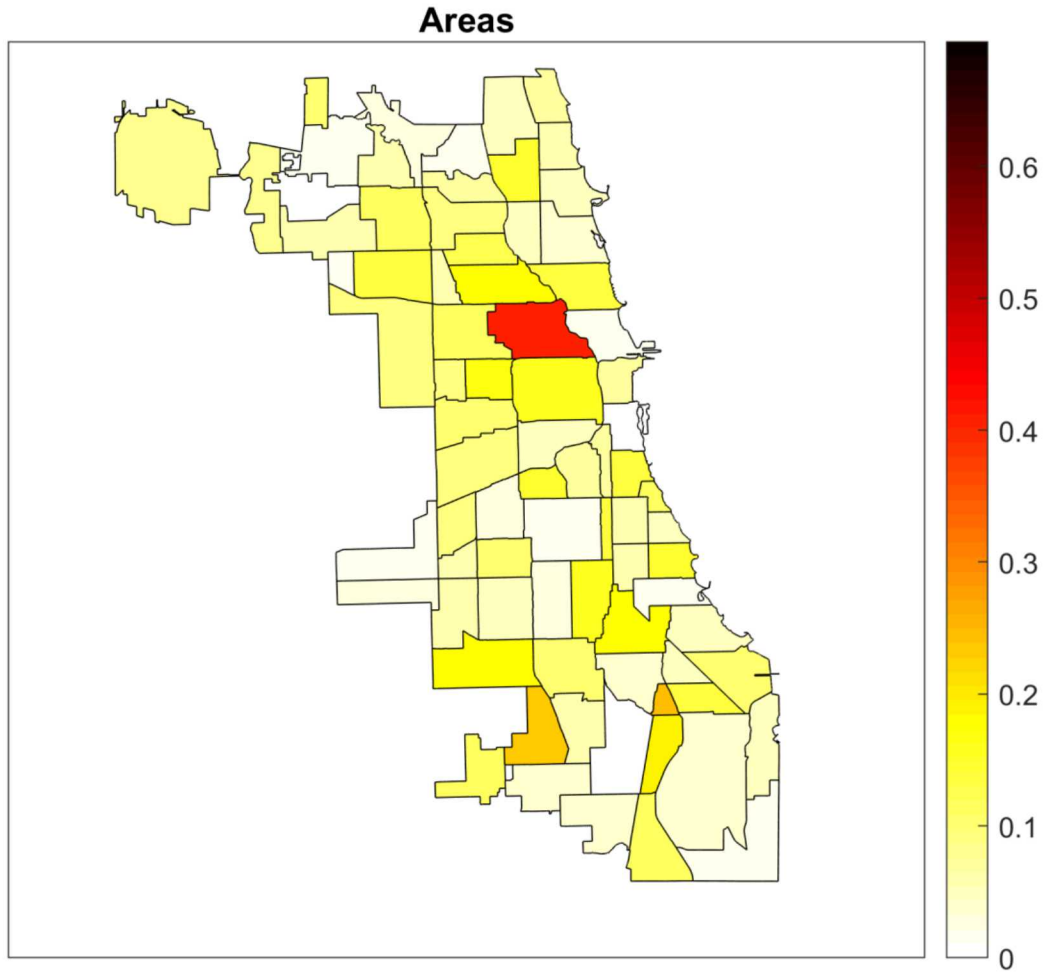

Areas
0.6
0.5
0.4
0.3
0.2
0.1
0

Component #6


Date
15
10
5
0
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017


Hour of Day
0.8
0.6
0.4
0.2
0
6
12
18
24


Top Crimes
deceptive practice
theft
other offense
offense involving children
battery
criminal damage
0
0.5
1


Areas
0.6
0.5
0.4
0.3
0.2
0.1
0

Conclusions and Future Work

- Conclusions
  - Stochastic gradient for GCP tensor decomposition
  - (Semi-)Stratified sampling yields faster convergence
  - Semi-stratified sampling avoids rejection sampling
  - Numerical experiments show promise
  - Code available
    - Tensor Toolbox for MATLAB (gcp_opt)
    - GenTen (C++) by Eric Phipps
- Future work
  - More sophisticated stochastic optimization methods
  - Sampling for memory locality
  - Asynchronous computation (ala HogWild)

Reference: T. G. Kolda, D. Hong. **Stochastic Gradients for Large-Scale Tensor Decomposition.** 2019. http://arxiv.org/abs/1906.01687