SAND2019-5632C

Illustration by Chris Brigman

Stochastic Optimization for Large-Scale Tensor Decomposition

Tamara G. Kolda, Sandia Natl. Labs.

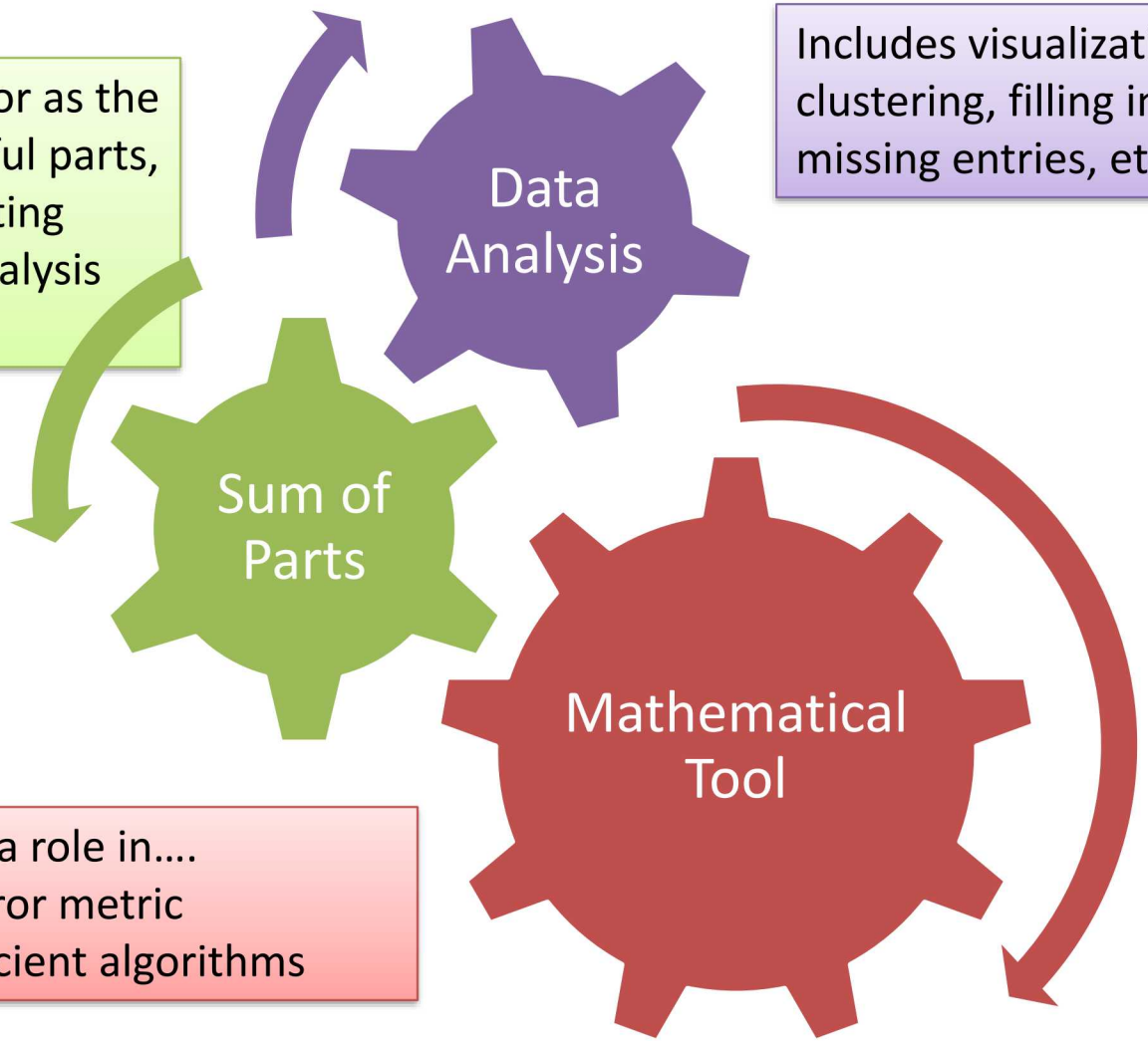
Joint with David Hong (Michigan), Jed Duersch (Sandia)

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.

Tensor Decomposition: A Mathematical Tool for Analysis of Tensor Data

Sandia National Laboratories

Express the tensor as the sum of meaningful parts, which is the starting point for data analysis activities



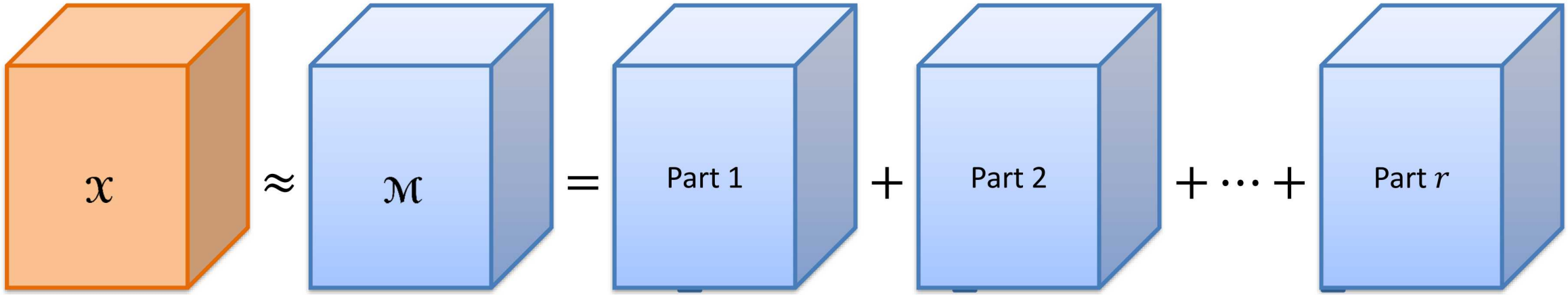
Includes visualization, clustering, filling in missing entries, etc.

Mathematics play a role in….
- Defining the error metric
- Developing efficient algorithms

Related Concepts for Matrices

- Singular value decomposition (SVD)
- Principal component analysis (PCA)
- Independent component analysis (ICA)
- Nonnegative matrix factorization (NMF)
- Sparse matrix factorization
- Matrix completion

Break Tensor into Understandable Parts…



Data Tensor
$n_1 \times n_2 \times n_3$

Model Tensor
$n_1 \times n_2 \times n_3$

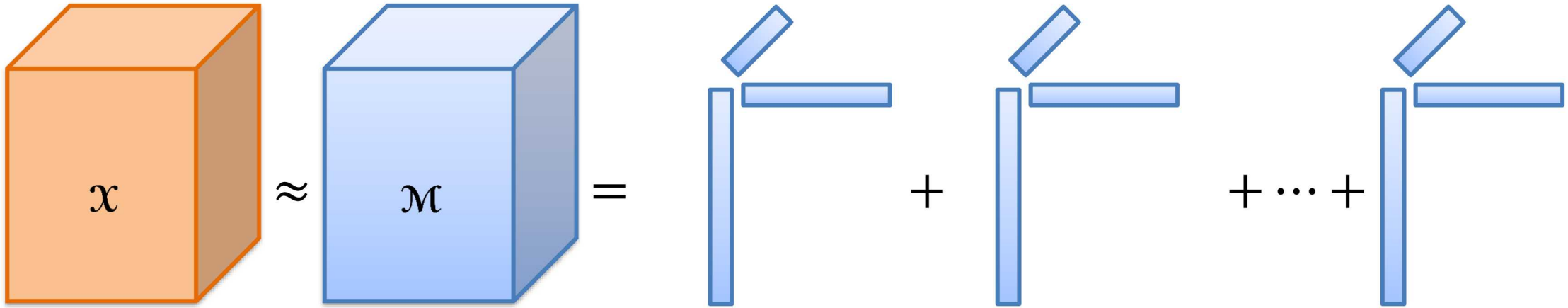
$n_1 \times n_2 \times n_3$

$n_1 \times n_2 \times n_3$

$n_1 \times n_2 \times n_3$

Key: The parts have structure!

Break Tensor into Understandable Parts…



Data Tensor
$n_1 \times n_2 \times n_3$

Model Tensor
$n_1 \times n_2 \times n_3$

$n_1 \times n_2 \times n_3$

$n_1 \times n_2 \times n_3$

$n_1 \times n_2 \times n_3$

Key: The parts have structure!

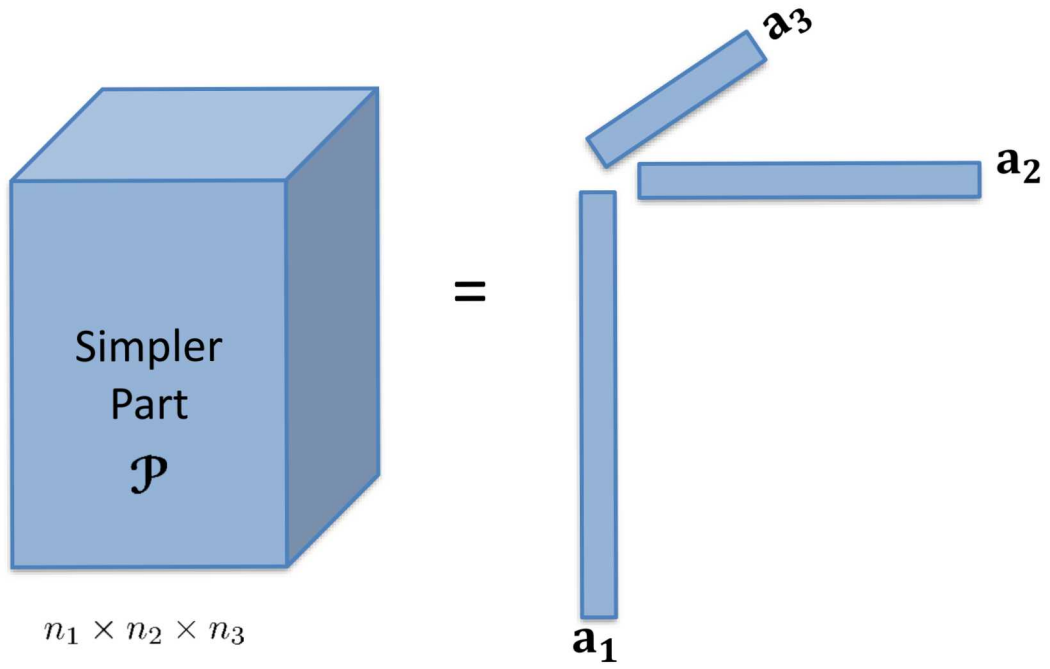
Rank-1 Tensors are the "Parts"

Given ***d* vectors**:

$$\mathbf{a}_k \in \mathbb{R}^{n_k} \text{ for } k = 1, \ldots, d$$

The **outer product** is

$$\mathcal{P} = \mathbf{a}_1 \circ \mathbf{a}_2 \cdots \circ \mathbf{a}_d \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$$



$$\mathcal{P}(i_1, i_2, i_3) = \mathbf{a}_1(i_1)\,\mathbf{a}_2(i_2)\,\mathbf{a}_3(i_3)$$

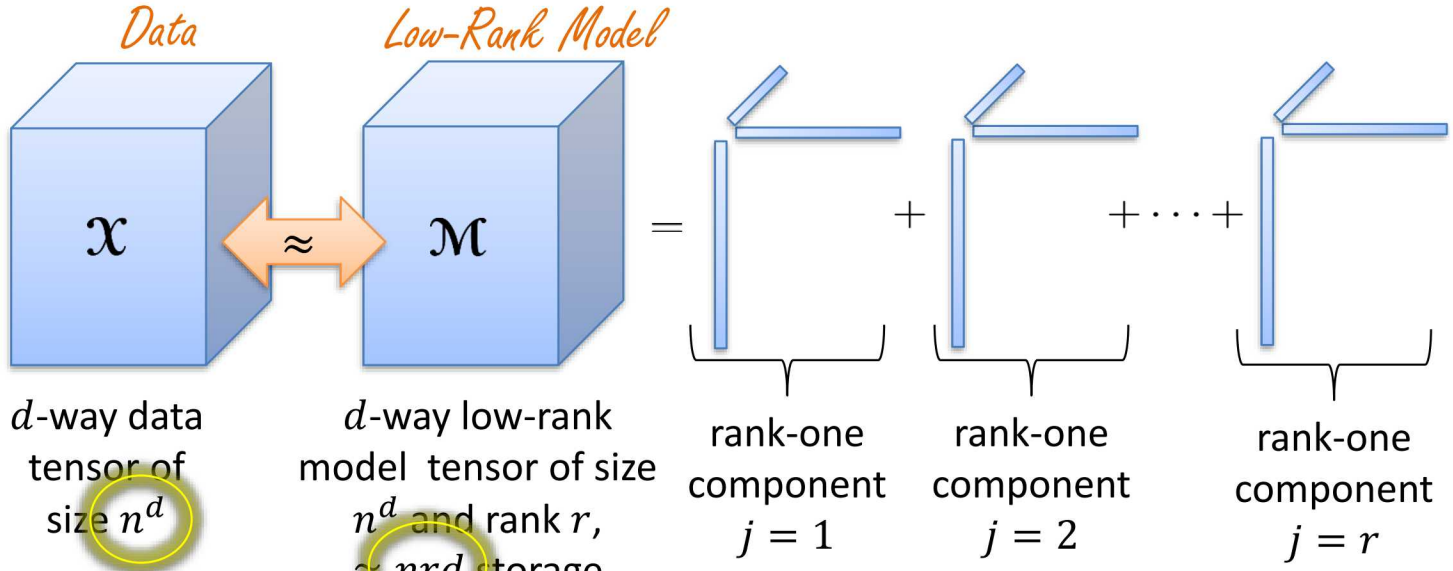
CANDECOMP/PARAFAC (CP) Tensor Factorization Uncovers the Rank-1 Parts

Images are three-way ($d = 3$), but assume all tensors are of size

$$n_1 \times n_2 \times \cdots \times n_d$$

For convenience, define

$$n = \sqrt[d]{\prod_k n_k}$$



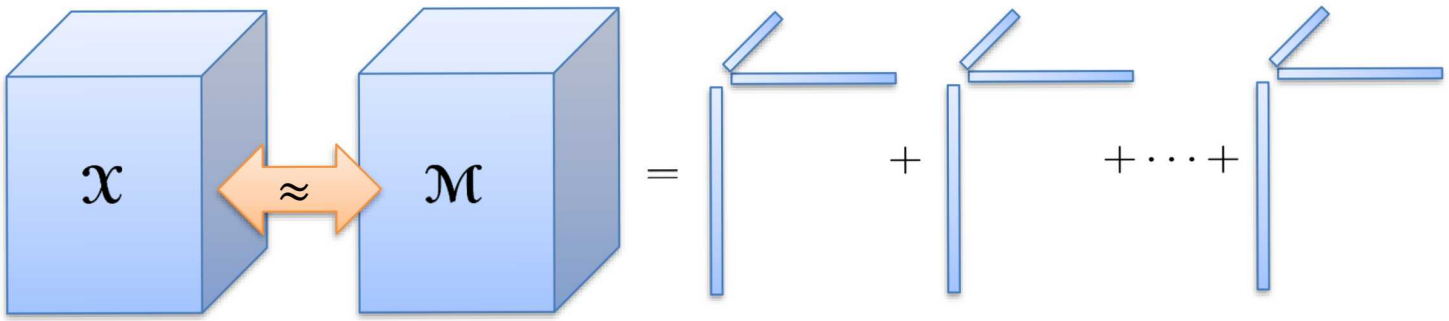
$$\mathcal{X} \approx \mathcal{M} \quad \text{where} \quad \mathcal{M} = \sum_{j=1}^{r} \mathbf{A}_1(:,j) \circ \mathbf{A}_2(:,j) \circ \cdots \circ \mathbf{A}_d(:,j)$$

Factor Matrices

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

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

Standard CP: Sum of Squares Error (SSE)



Shorthand for element of data tensor:

$$x_i \equiv x(i_1, i_2, \ldots, i_d)$$

Element of model low-rank tensor:

$$m_i \equiv \sum_{j=1}^{r} \prod_{k=1}^{d} \mathbf{A}_k(i_k, j)$$

(defined in terms of factor matrices)

Ω = set of all n^d elements in tensor

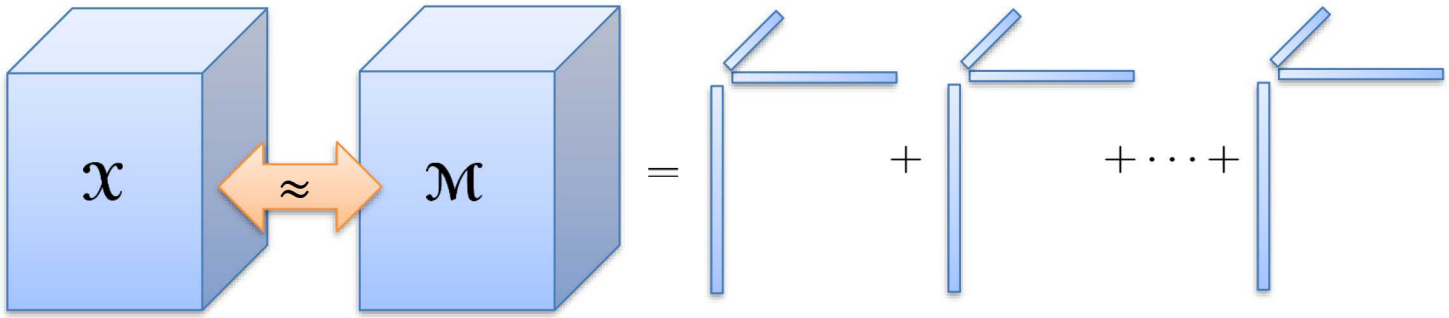
Standard CP

$$\min \; F(\mathcal{X}, \mathcal{M}) \equiv \sum_{i \in \Omega} (x_i - m_i)^2$$

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

Hitchcock, 1927; Carroll and Chang, 1970; Harshman, 1970

Generalized CP (GCP)



GCP

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

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

Why?

- SSE: maximum likelihood estimate (MLE) for Gaussian distribution

$$x_i = m_i + \epsilon, \; \epsilon \sim \mathcal{N}(0, \sigma)$$

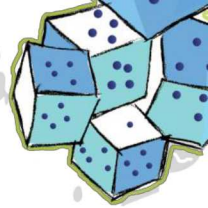
$$x_i \sim \mathcal{N}(m_i, \sigma)$$

- Different MLEs for different distributions
  - Poisson (counts)
  - Bernoulli (binary)

Hitchcock, 1927; Carroll and Chang, 1970; Harshman, 1970

Probability Distribution ⇒ Maximum Likelihood Estimator

Sandia National Laboratories



Data Value

"Natural" Parameter

Model Value

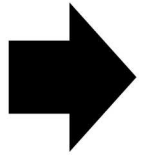
$$x_i \sim p(x_i|\theta_i) \text{ where } \ell(\theta_i) = m_i$$

Link Function

Probability Distribution Function (PDF) or Probability Mass Function (PMF)

Maximize Likelihood of Data Tensor

$$\prod_{i \in \Omega} p(x_i, \theta_i)$$



Maximize Log-Likelihood

$$\sum_{i \in \Omega} \log p(x_i, \theta_i)$$

GCP

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

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

Given PDF/PMF $p(x|\theta)$ and link function $\ell(\theta)$, GCP MLE by minimizing

$$f(x, m) = -\log p(x, \ell^{-1}(m))$$

Gaussian MLE (Standard CP)

PDF for Normal Distribution

$$p(x \mid \mu, \sigma) = \frac{e^{-(x-\mu)^2 \,/\, 2\sigma^2}}{\sqrt{2\pi\sigma^2}}$$

and

Link Function

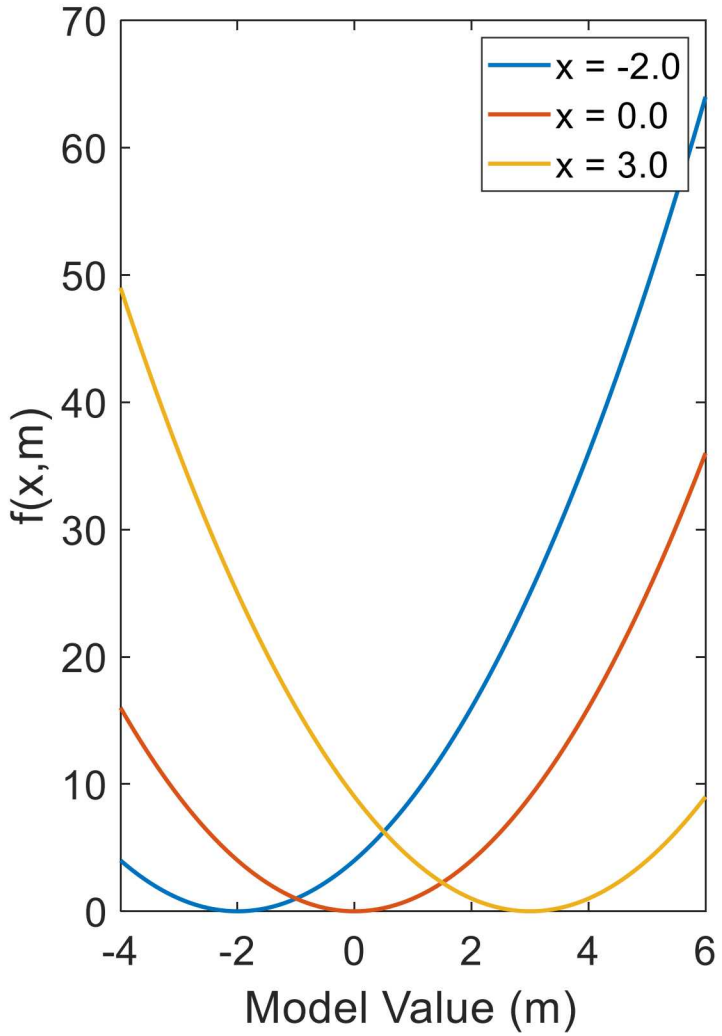
$m = \mu$

σ constant

Negative log-likelihood: $$-\log p(x|\mu,\sigma) = \frac{(x-u)^2}{2\sigma^2} + \frac{1}{2}\log(2\pi\sigma^2)$$

Eliminate natural parameter via link function: $$f(x,m) = \frac{(x-m)^2}{2\sigma^2} + \frac{1}{2}\log(2\pi\sigma^2)$$

Eliminate constants: $$f(x,m) = (x-m)^2$$



Hong, Kolda, Duersch, arXiv, 2018

Bernoulli MLE with Odds Link (Binary Data)

Bernoulli random variable

$x \in \{0,1\}$

ρ = probability of a 1

$$p(x \mid \rho) = \rho^x (1-\rho)^{(1-x)}, \quad x \in \{0,1\}$$

PMF for Bernoulli Distribution

$$p(x \mid \rho) = \rho^x (1-\rho)^{(1-x)}$$
$$x \in \{0,1\}$$

and

Link Function

$$m = \frac{\rho}{(1-\rho)}$$

Odds Link

$$\ell(\rho) = \rho \,/\, (1-\rho)$$
$$\ell^{-1}(m) = m \,/\, (1+m)$$

Odds (m)	Probability (ρ)
¼	20%
1	50%
4	80%
10	90%

Negative log-likelihood:

$$-\log p(x \mid \rho) = \log \frac{1}{1-\rho} - x \log \frac{\rho}{1-\rho}$$

Eliminate natural parameter via link function:

$$f(x,m) = \log(1+m) - x \log m \quad \text{for} \quad m > 0$$

Hong, Kolda, Duersch, arXiv, 2018

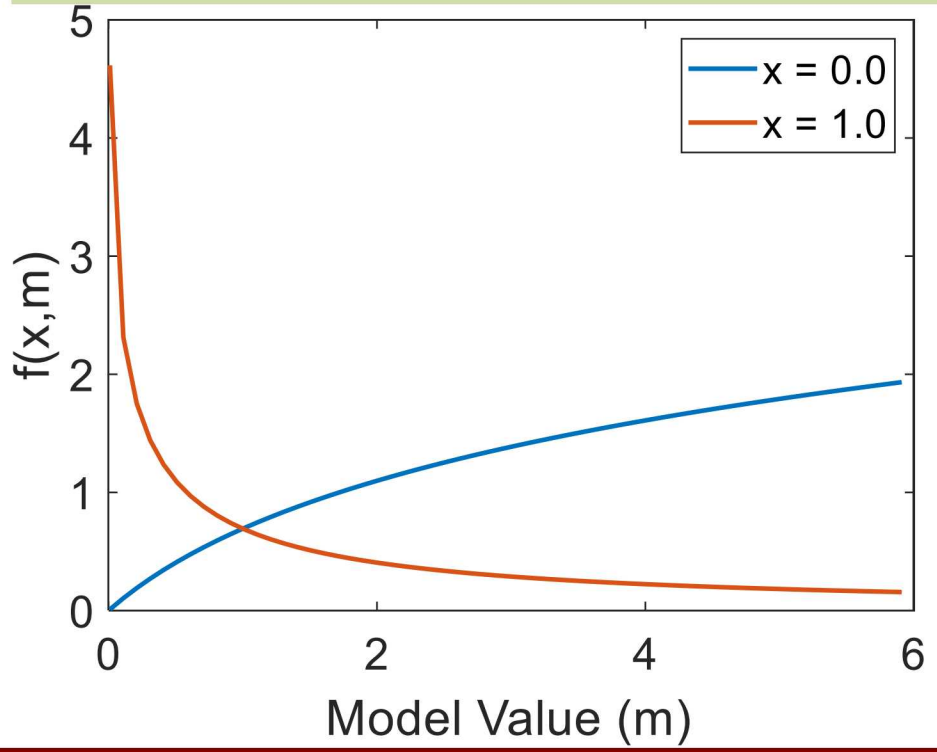
Bernoulli MLE with Odds Link (Binary Data)

Bernoulli random variable

$x \in \{0,1\}$

ρ = probability of a 1

$$p(x \mid \rho) = \rho^x (1-\rho)^{(1-x)}, \quad x \in \{0,1\}$$



PMF for Bernoulli Distribution

$$p(x \mid \rho) = \rho^x (1-\rho)^{(1-x)}$$

$$x \in \{0,1\}$$

and

Link Function

$$m = \frac{\rho}{(1-\rho)}$$

Negative log-likelihood:

$$-\log p(x \mid \rho) = \log \frac{1}{1-\rho} - x \log \frac{\rho}{1-\rho}$$

Eliminate natural parameter via link function:

$$f(x,m) = \log(1+m) - x \log m \quad \text{for} \quad m > 0$$

Hong, Kolda, Duersch, arXiv, 2018

Bernoulli MLE with Logit Link (Binary Data)

Bernoulli random variable

$x \in \{0,1\}$

ρ = probability of a 1

$$p(x \mid \rho) = \rho^x (1-\rho)^{(1-x)}, \quad x \in \{0,1\}$$

PMF for Bernoulli Distribution

$$p(x \mid \rho) = \rho^x (1-\rho)^{(1-x)}$$

$$x \in \{0,1\}$$

and

Link Function

$$m = \log \frac{\rho}{(1-\rho)}$$

Logit (Log-Odds) Link

$$\ell(\rho) = \log\left(\rho / (1-\rho)\right)$$

$$\ell^{-1}(m) = e^m / (1+e^m)$$

Log-Odds(m)	Probability (ρ)
-1.39	20%
0	50%
1.39	80%
2.30	90%

Negative log-likelihood:

$$-\log p(x \mid \rho) = \log \frac{1}{1-\rho} - x \log \frac{\rho}{1-\rho}$$

Eliminate natural parameter via link function:

$$f(x,m) = \log(1+e^m) - xm \quad \text{for} \quad m \in \mathbb{R}$$

Hong, Kolda, Duersch, arXiv, 2018

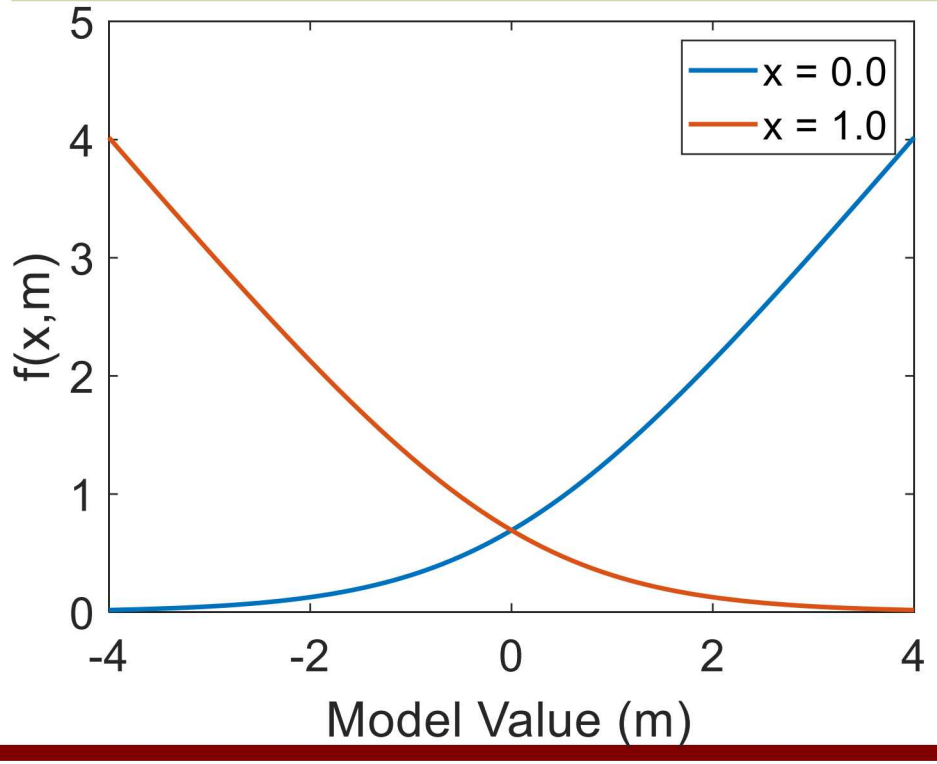
Bernoulli MLE with Logit Link (Binary Data)

Bernoulli random variable

$x \in \{0,1\}$

ρ = probability of a 1

$$p(x \mid \rho) = \rho^x (1-\rho)^{(1-x)}, \quad x \in \{0,1\}$$



PMF for Bernoulli Distribution

$$p(x \mid \rho) = \rho^x (1-\rho)^{(1-x)}$$

$$x \in \{0,1\}$$

and

Link Function

$$m = \log \frac{\rho}{(1-\rho)}$$

Negative log-likelihood:

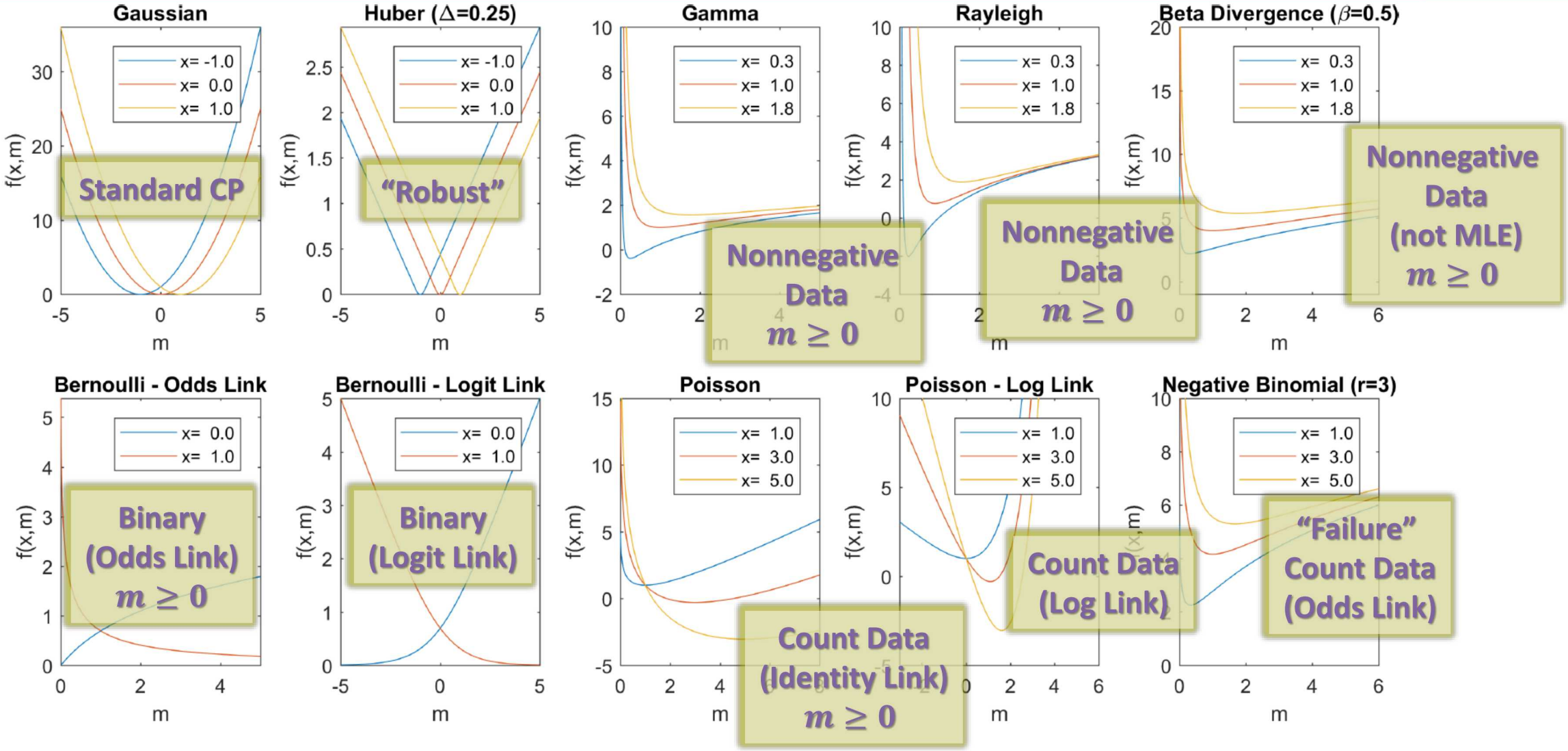
$$-\log p(x \mid \rho) = \log \frac{1}{1-\rho} - x \log \frac{\rho}{1-\rho}$$

Eliminate natural parameter via link function:

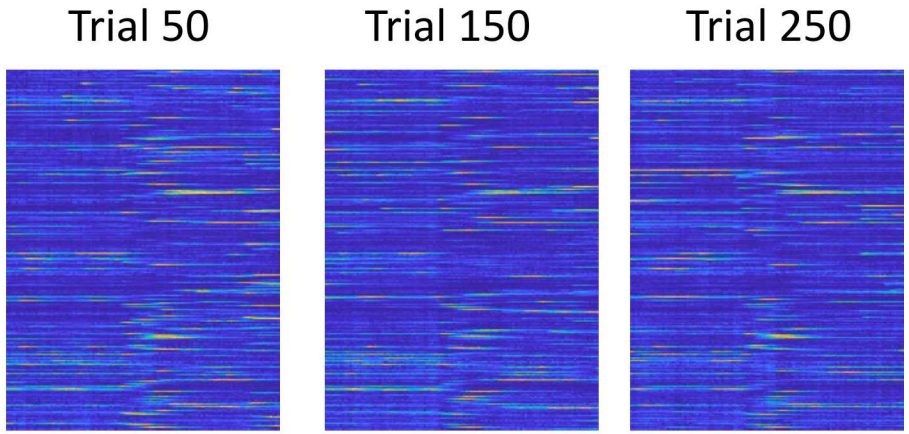
$$f(x,m) = \log(1+e^m) - xm \quad \text{for} \quad m \in \mathbb{R}$$

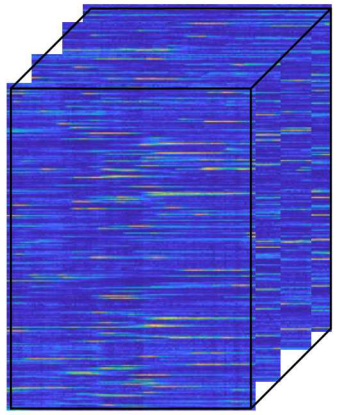
Hong, Kolda, Duersch, arXiv, 2018

Sampling of Loss Functions

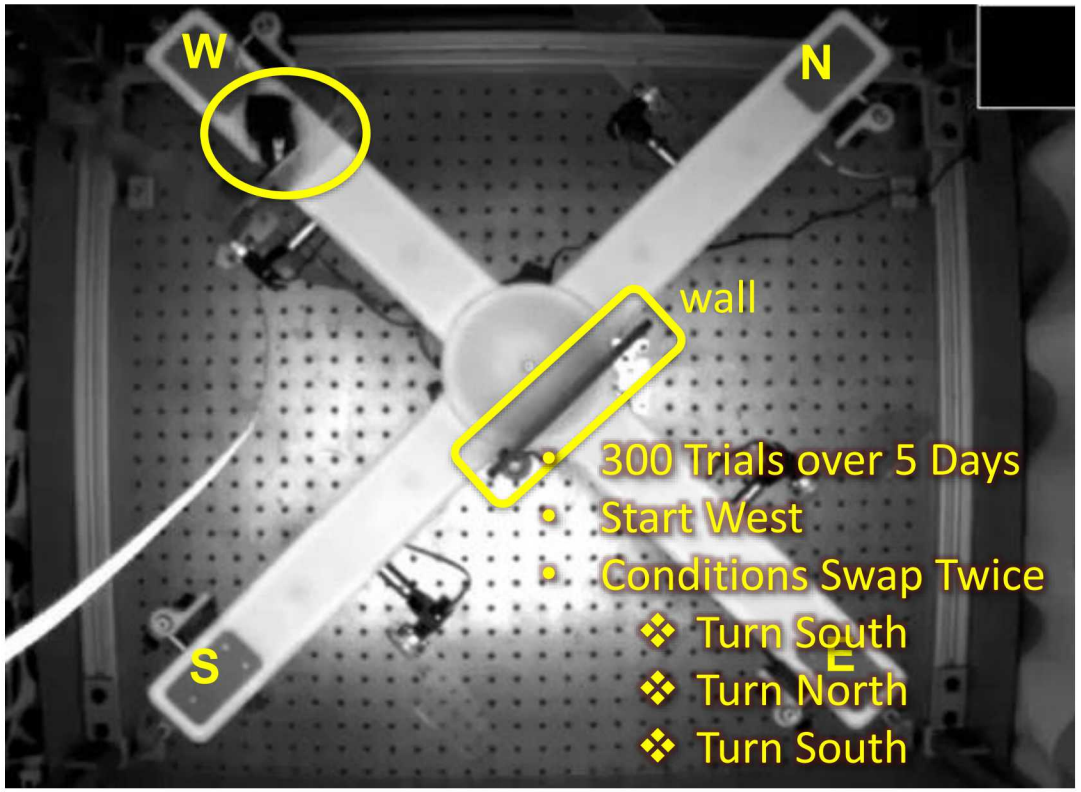


In Neuroscience, Tensor Decomposition Uncovers Patterns in Neuron Activity





282 neurons × 111 time bins × 300 trials

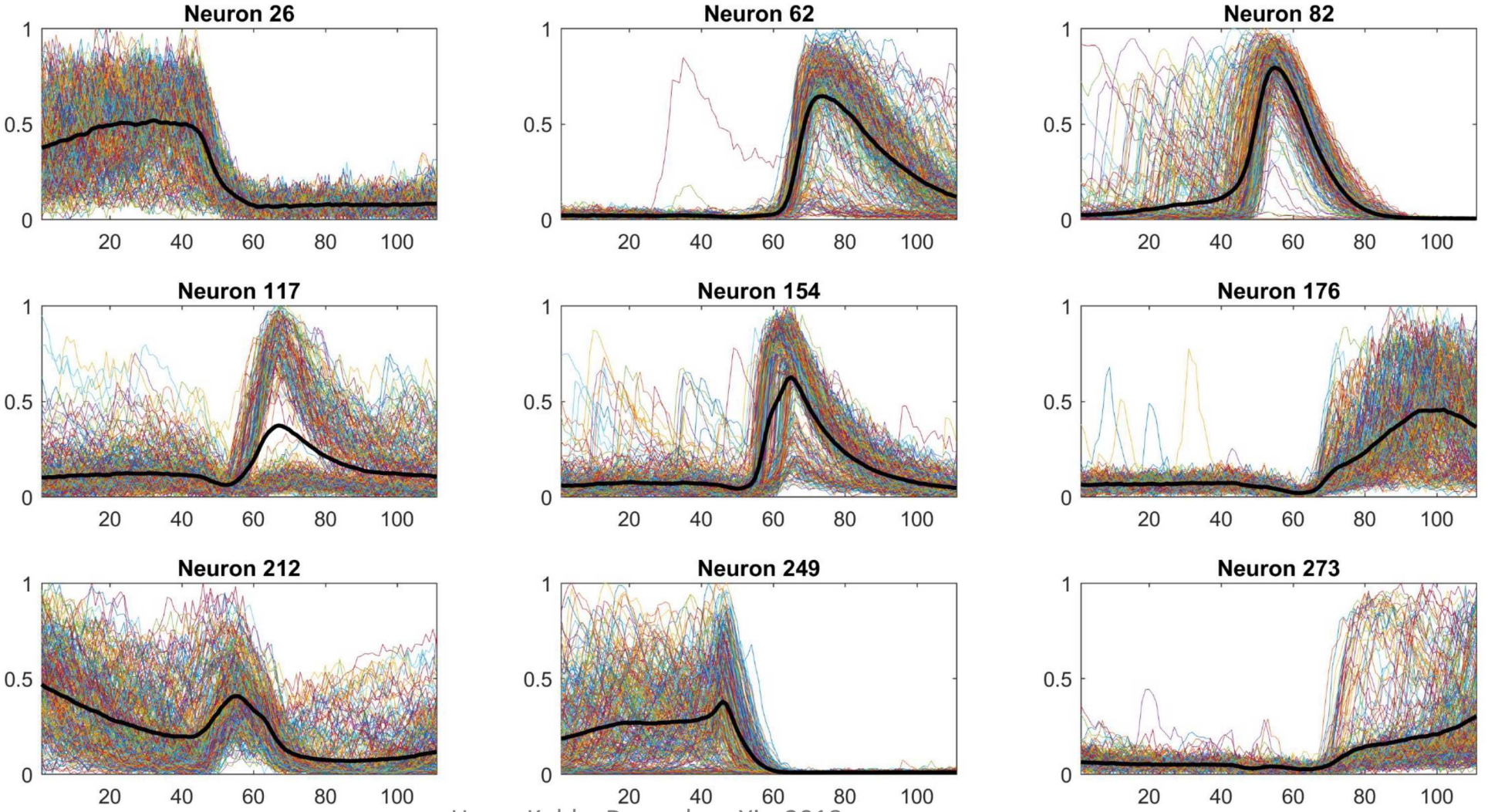


Williams et al., Neuron, 2018

Example Neuron Activity

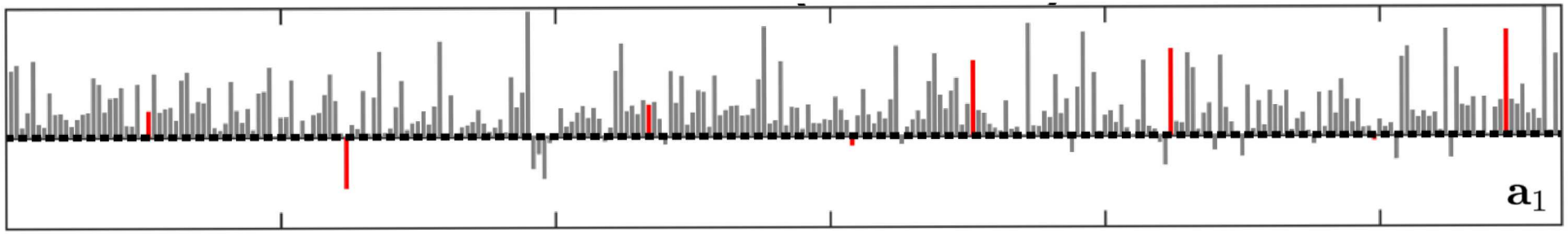
Thin lines show 300 individual trials

Thick line is average

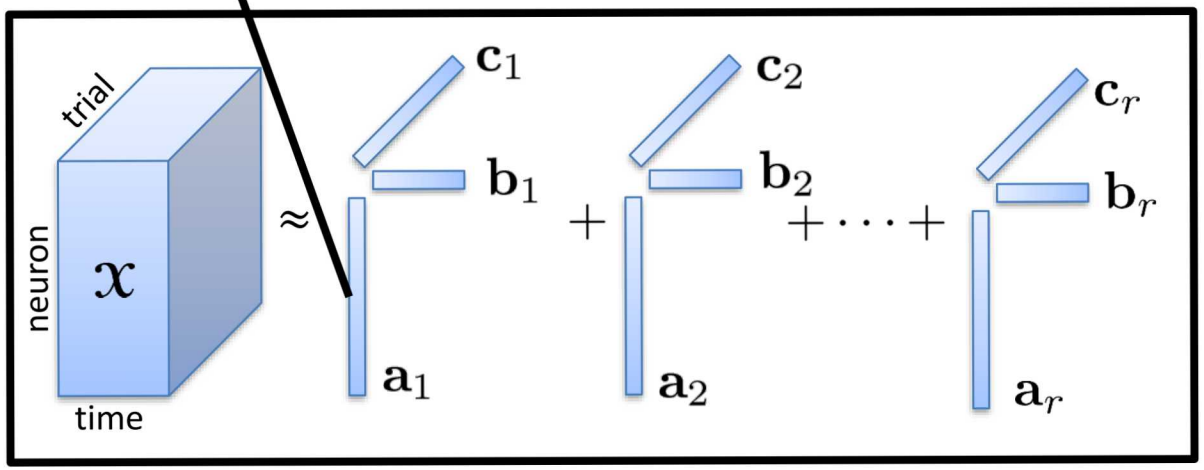


Hong, Kolda, Duersch, arXiv, 2018

Neuron Factor Vector Visualized as Bar Chart

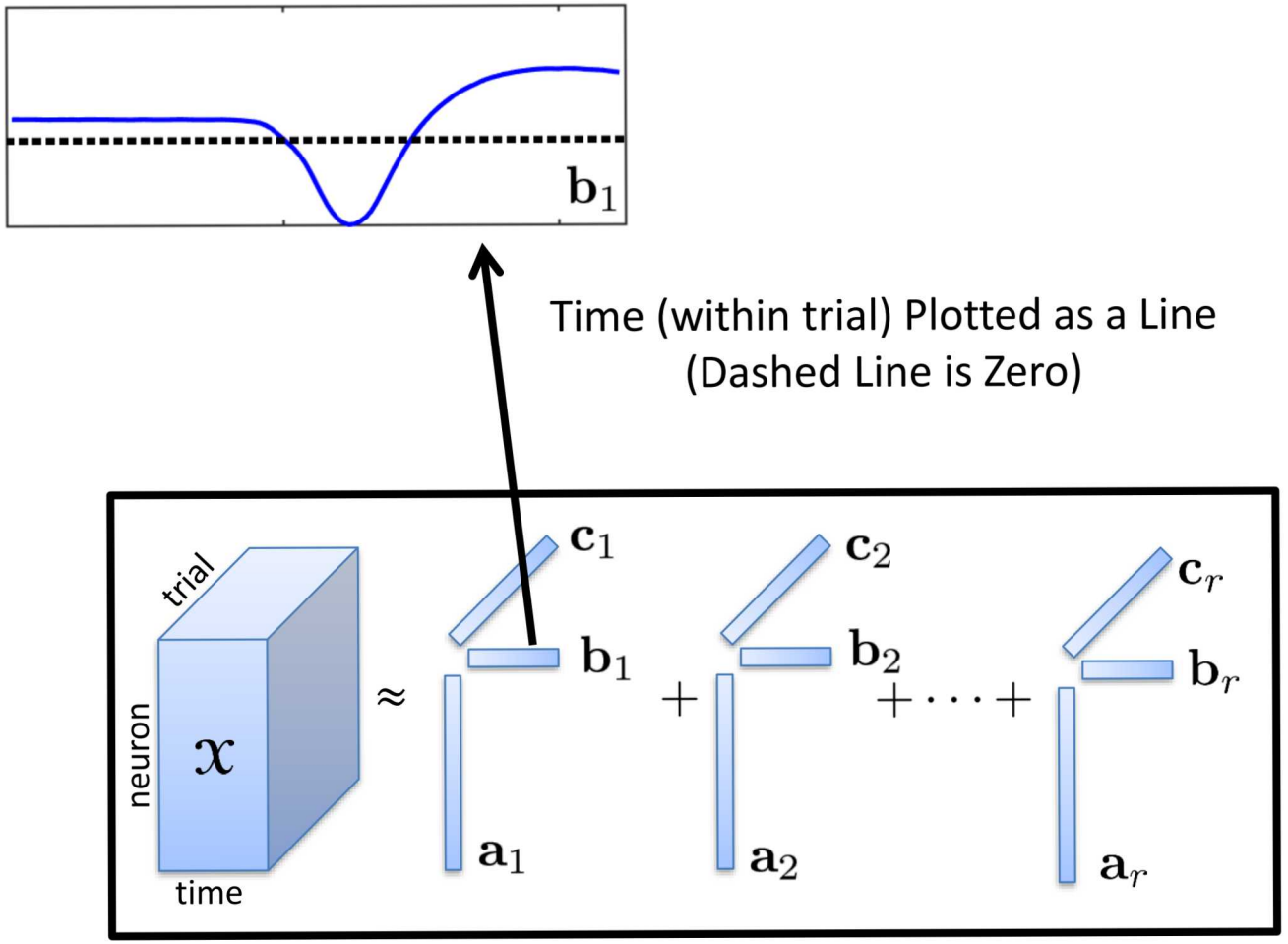


Neuron Modes Plotted as a Bar Chart
(Red Lines Correspond to Examples in Previous Slide)



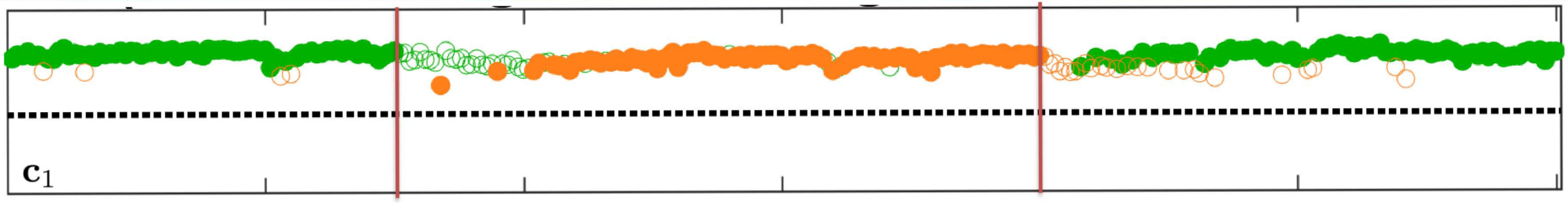
Hong, Kolda, Duersch, arXiv, 2018

Time Factor Vector Visualized as Line



Hong, Kolda, Duersch, arXiv, 2018

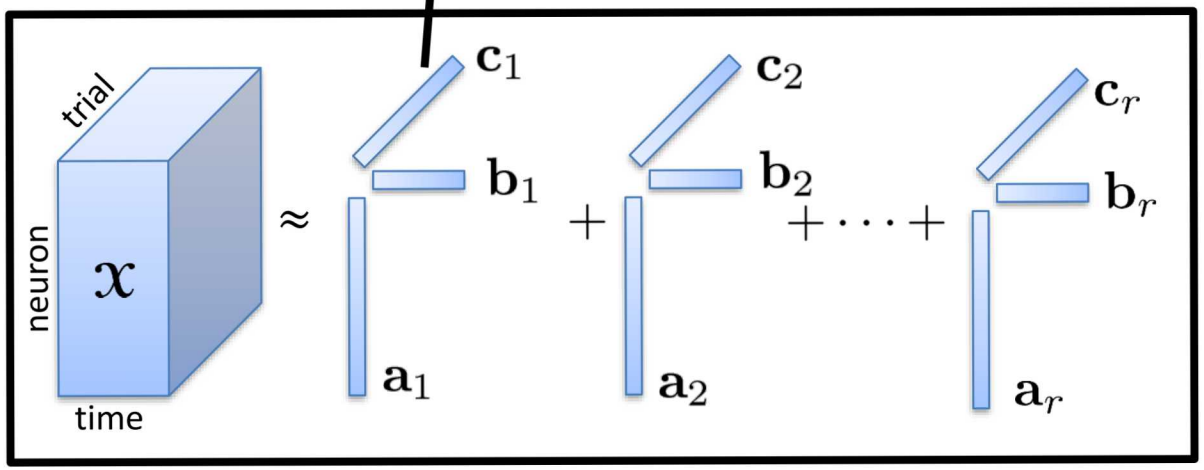
Trial Factor Vector Visualized as Color-Coded Scatter Plot



Rule Change

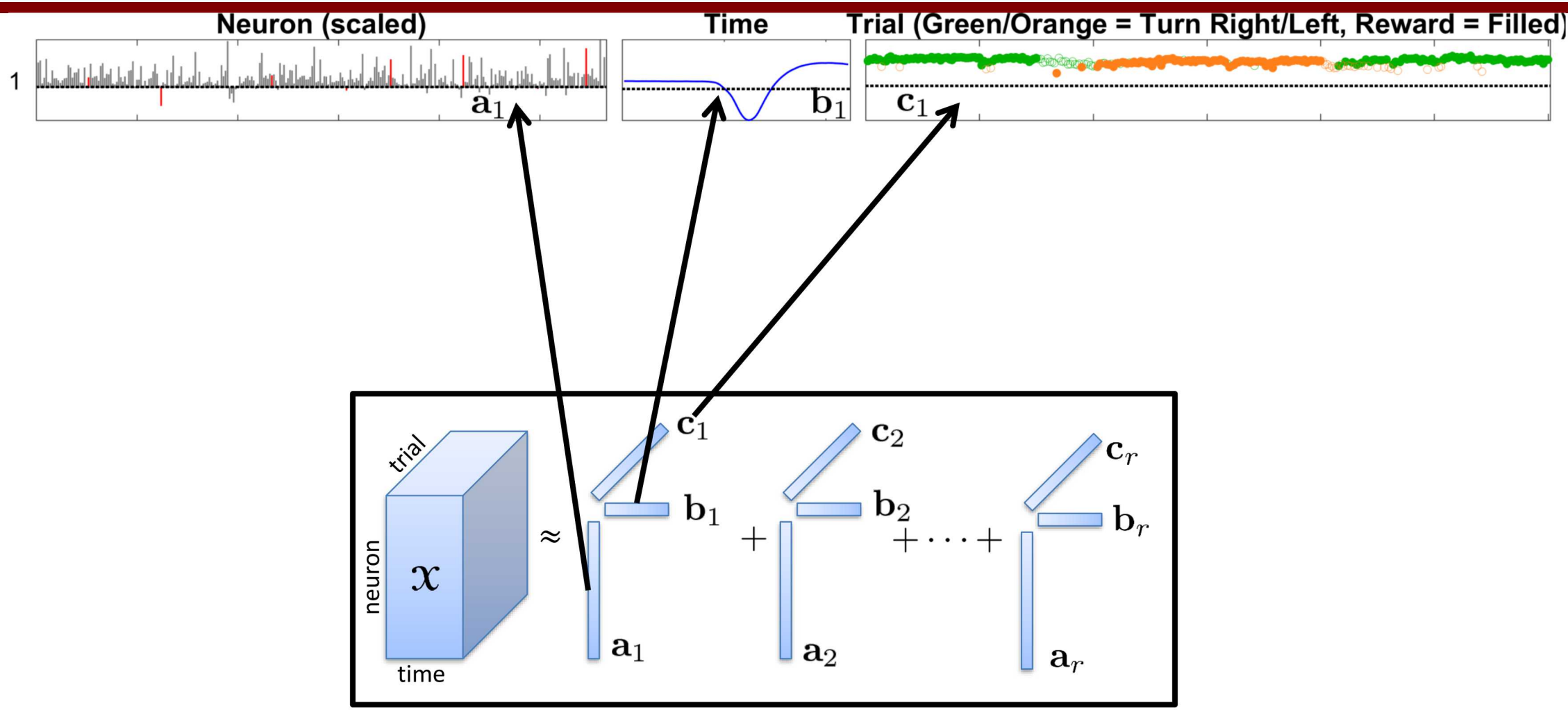
Trial Plotted as Scatter Graph
Right turn = Green
Left turn = Orange
Filled = Reward

Rule Change



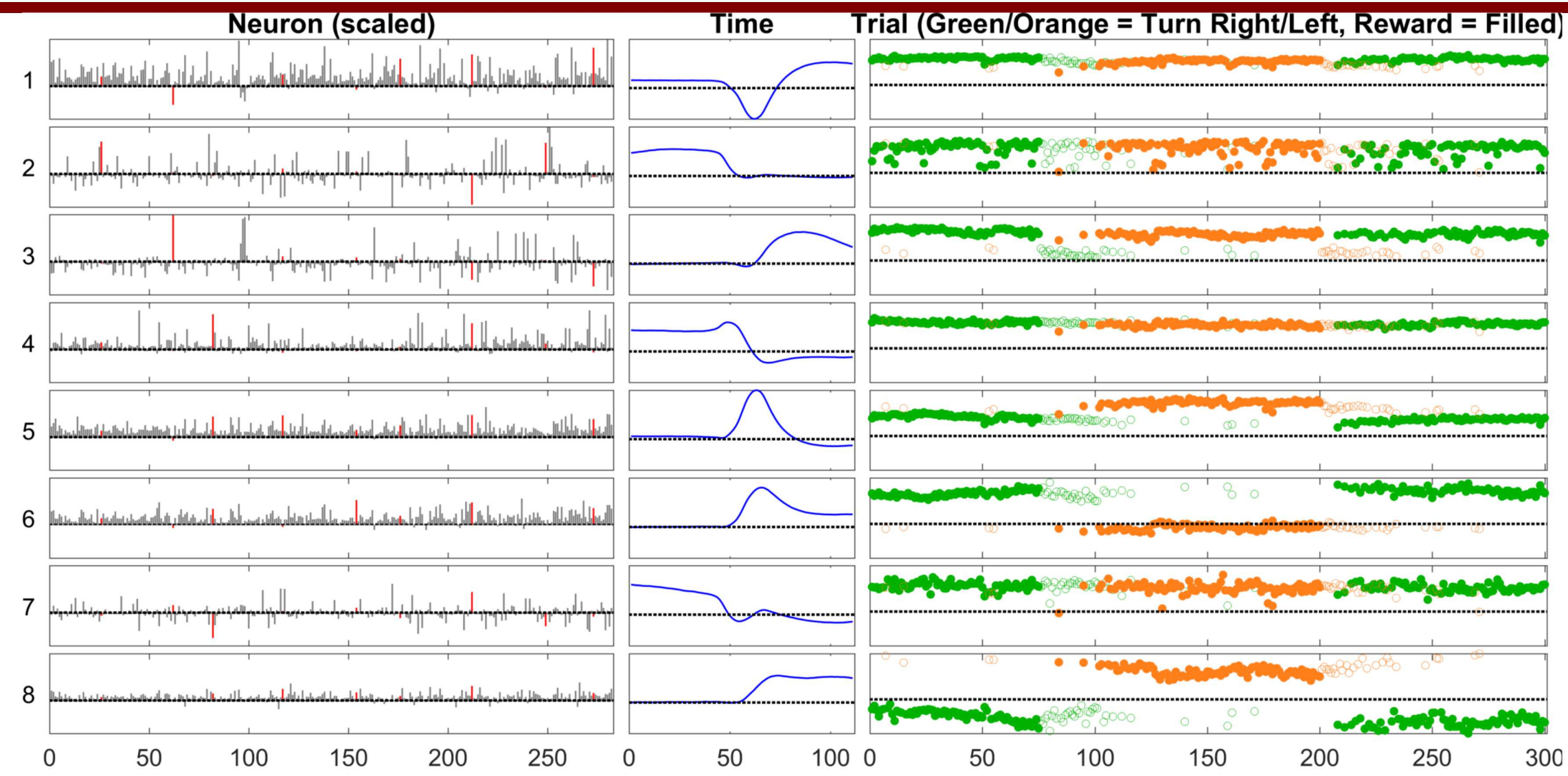
Hong, Kolda, Duersch, arXiv, 2018

Visualization of CP Tensor Decomposition Shows the Factors (Vectors)

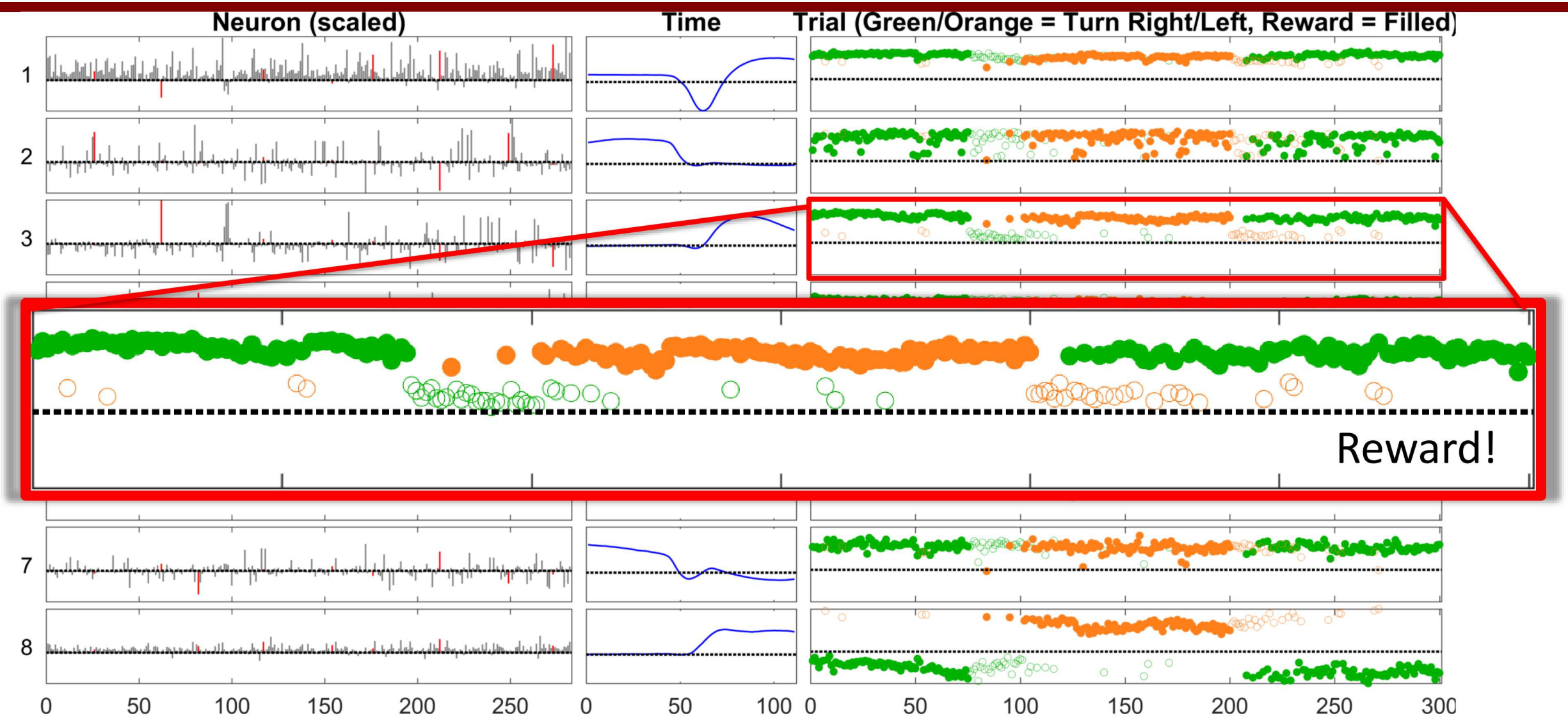


Hong, Kolda, Duersch, arXiv, 2018

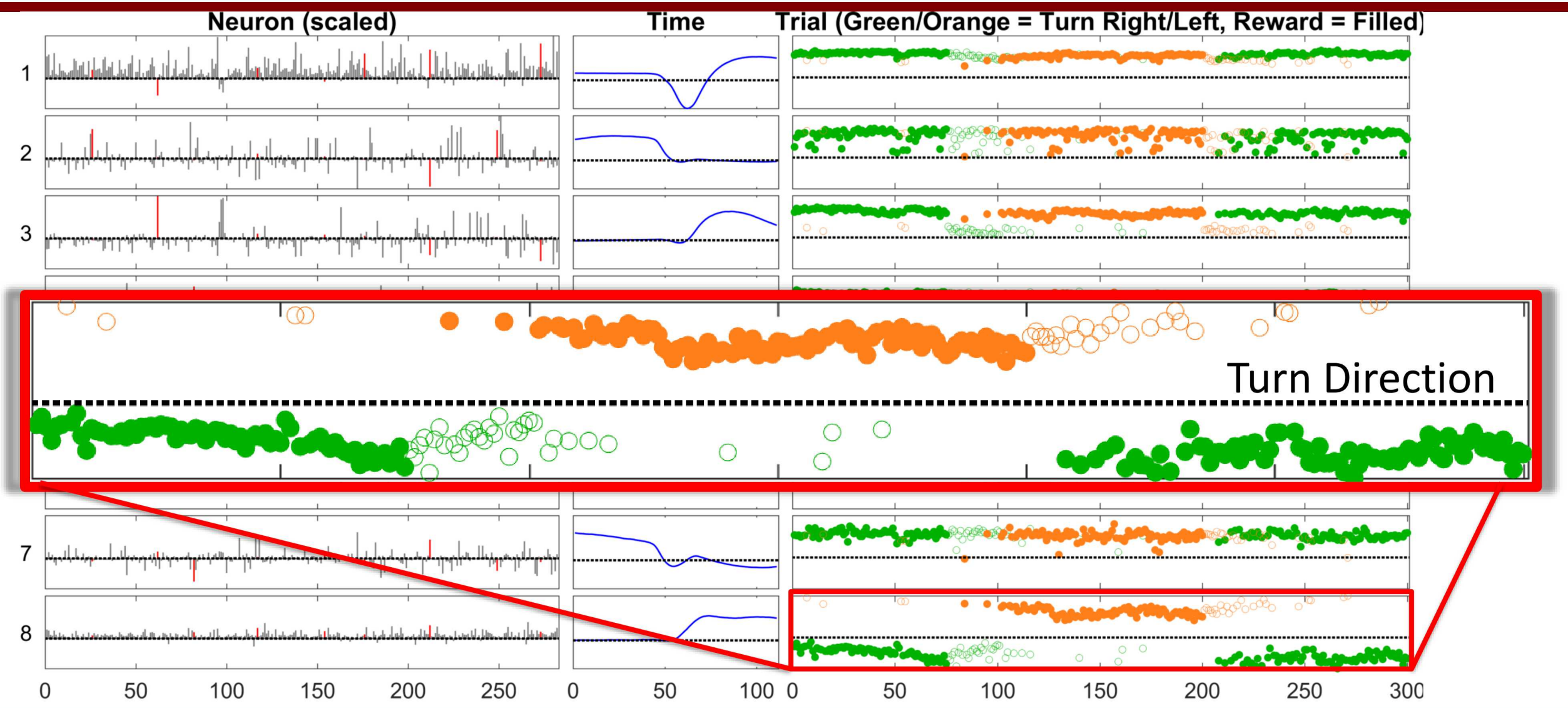
“Standard” CP Decomposition of Mouse Data, aka Gaussian



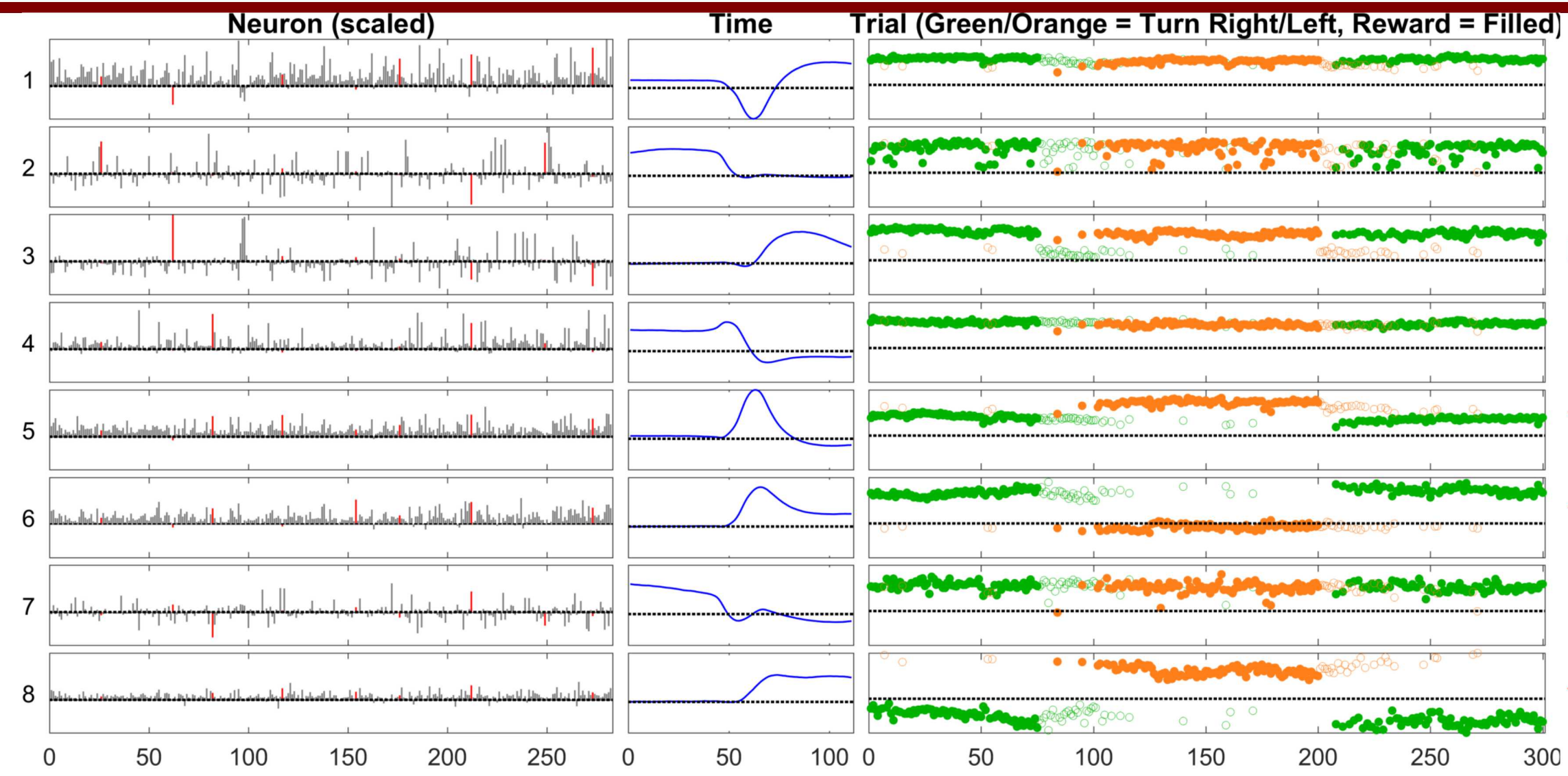
CP Tensor Decomposition "Sees" Reward



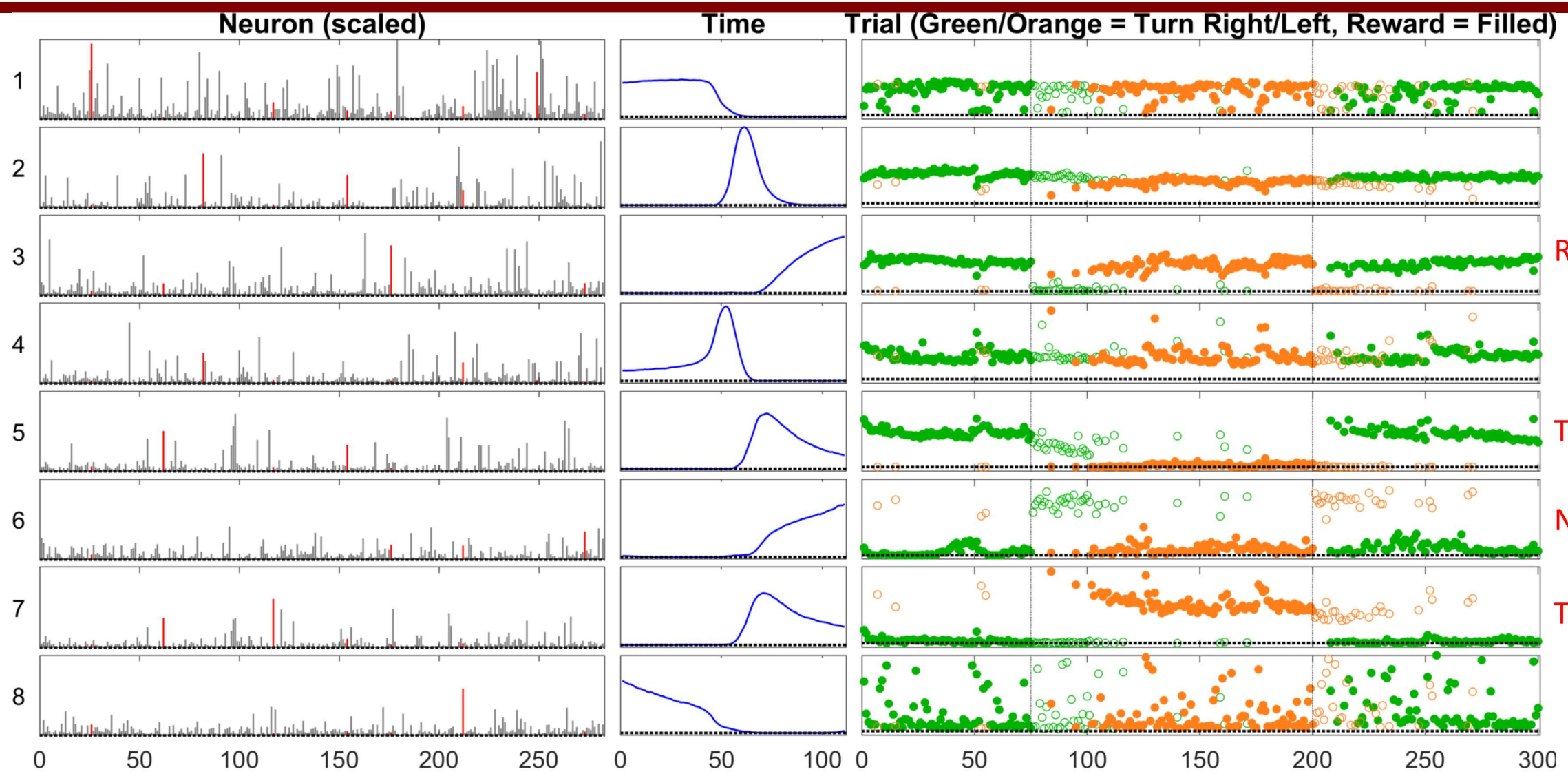
CP Tensor Decomposition "Sees" Turn Direction



CP Tensor Decomposition Can be Tough to Interpret due to Negative Entries

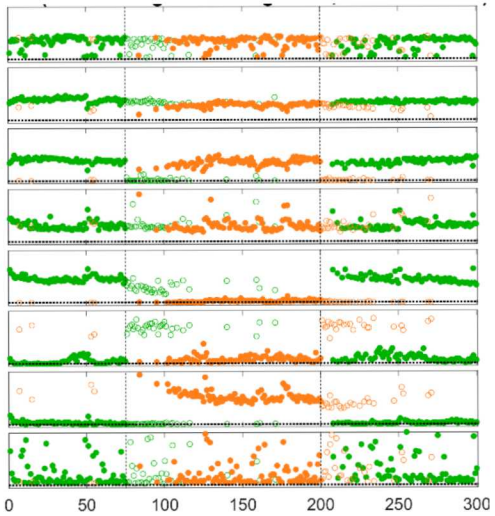


GCP Decomposition with Beta Divergence ($\beta = 0.5$)



Regression Using GCP Factors on Trial Mode

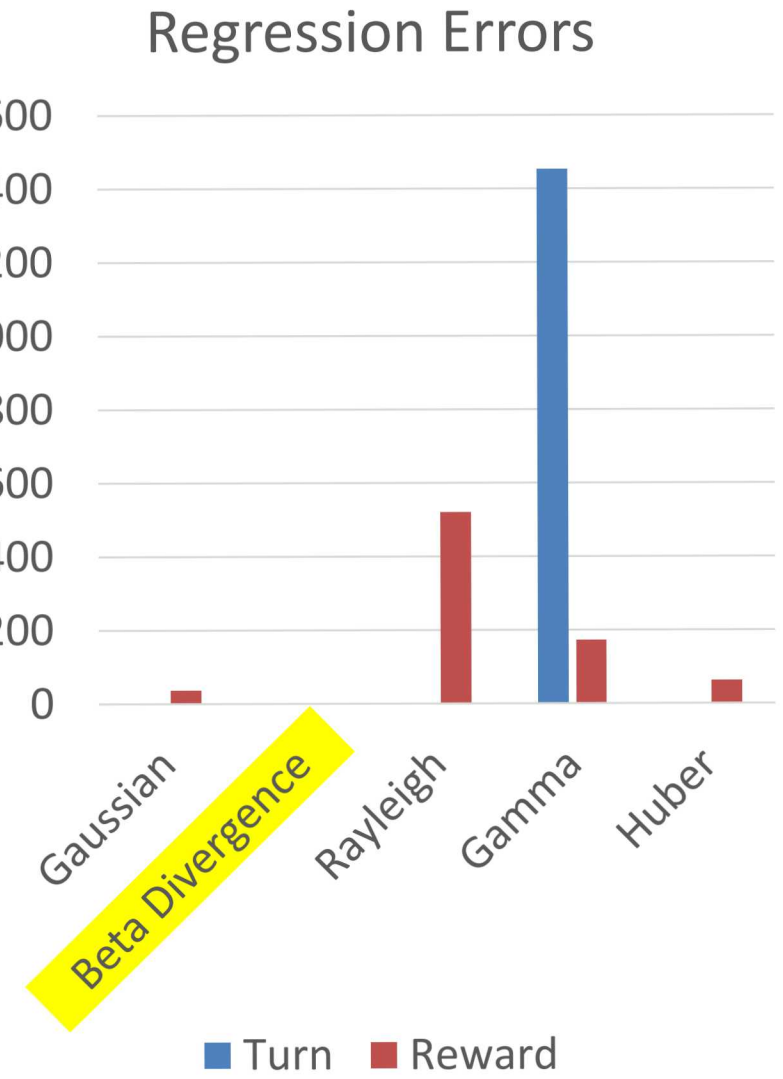
Trial Factor Matrix is 300×8



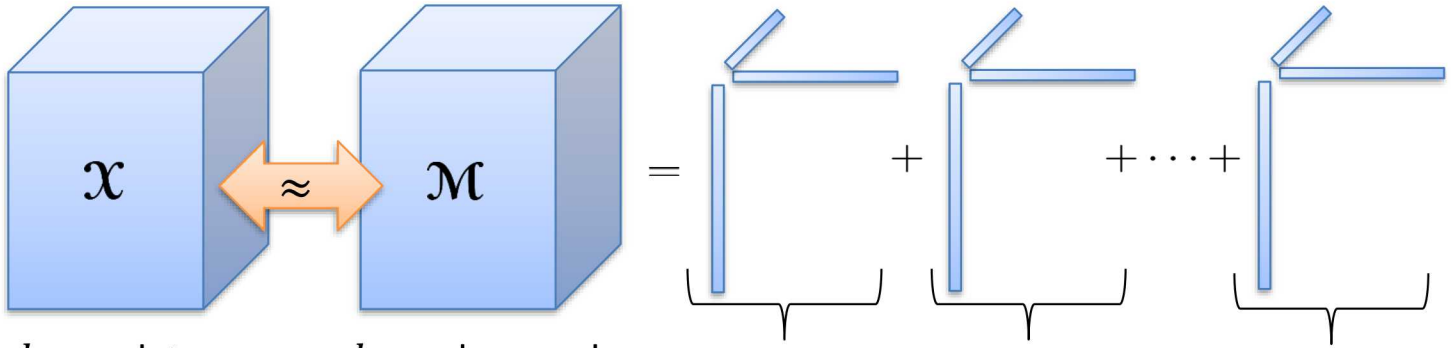
$$\min_{\boldsymbol{\beta}} \|\mathbf{A}_3^{\text{train}}\boldsymbol{\beta} - \mathbf{y}^{\text{train}}\|$$

$$\hat{\mathbf{y}}^{\text{test}} = [\mathbf{A}_3^{\text{test}}\boldsymbol{\beta} \geq 0.5]$$

Look at predicting turn and reward.
Split into two groups of 150 trials.
Train regression model with 1st group.
Test with 2nd group.
Repeat 100 times.



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 $k \in \{1, \ldots, d\}$

WLOG, $n = n_1 = \cdots = n_d$

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 (Indentity Link) [Welling & Webber, 2001; Chi & Kolda, 2009]
  $$f(x,m) = m - x \log m$$
- Logistic CP, etc. [Hong, Kolda, Duersch, 2018]
  $$f(x,m) = \log(m+1) - x \log(m)$$

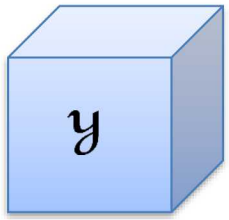
Gradient-based Optimization for Fitting the GCP Model

Sandia National Laboratories

GCP

$$\min \; F(\mathcal{X}, \mathcal{M}) \equiv \sum_{i \in \Omega} f(x_i, m_i)$$
$$\text{s.t. } \operatorname{rank}(\mathcal{M}) \leq r$$

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



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

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

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)

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

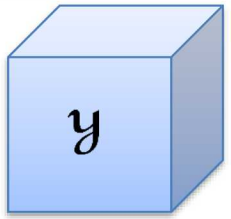
$$\mathbb{E}[\tilde{g}^{(t)}] = g^{(t)} \equiv \nabla f(x^{(t)})$$

Adam (Kingma & Ba, 2015)

Adaptive momentum SGD

Standard gradient

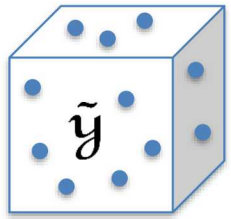
$$\mathbf{G}_k = \mathbf{Y}_{(k)}\mathbf{Z}_k \qquad \text{Cost: } O(rn^d) \text{ flops}$$



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

Stochastic gradient

$$\tilde{\mathbf{G}}_k = \tilde{\mathbf{Y}}_{(k)}\mathbf{Z}_k \qquad \text{Cost: } O(rs) \text{ flops}$$



Choose stochastic *sparse* Y-tensor

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

such that

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

By linearity of expectation: $\mathbb{E}[\tilde{\mathbf{G}}_k] = \mathbf{G}_k$

Uniform Sampling

Goal: Random ***sparse*** tensor of size n^d that equals the "Y-tensor" in expectation

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

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

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

$$\sum_{i \in \Omega} \tilde{s}_i = s$$

$$\sum_{i \in \Omega} \delta(\tilde{s}_i) \leq s$$

Choosing s, the number of sampled elements…

- Choose $s = O(rn)$
- Gradient = $O(rs) = O(r^2 n)$ versus $O(rn^d)$

Downside…

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

Theory

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

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

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

Intuition: Stratified 0/1 Sampling Decreases Variance

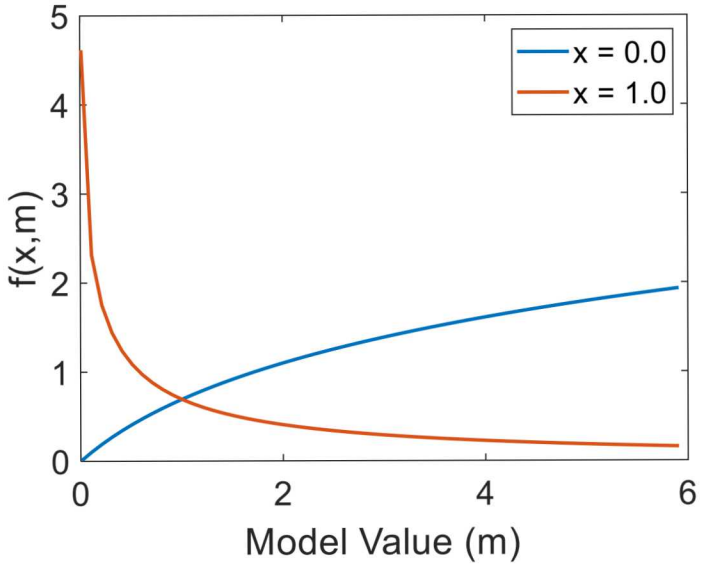
Needell, Srebro, and Ward (2013) justify **biased sampling toward functionals with higher Lipschitz smoothness** constants to reduce the variance in the stochastic gradient.

In our case, the functionals correspond to $f_i = f(x_i, m_i)$, and we contend that in many cases the **functionals with $x_i = 0$ have lower Lipschitz smoothness constants** and therefore needn't be sampled as often as the nonzeros.

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

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

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



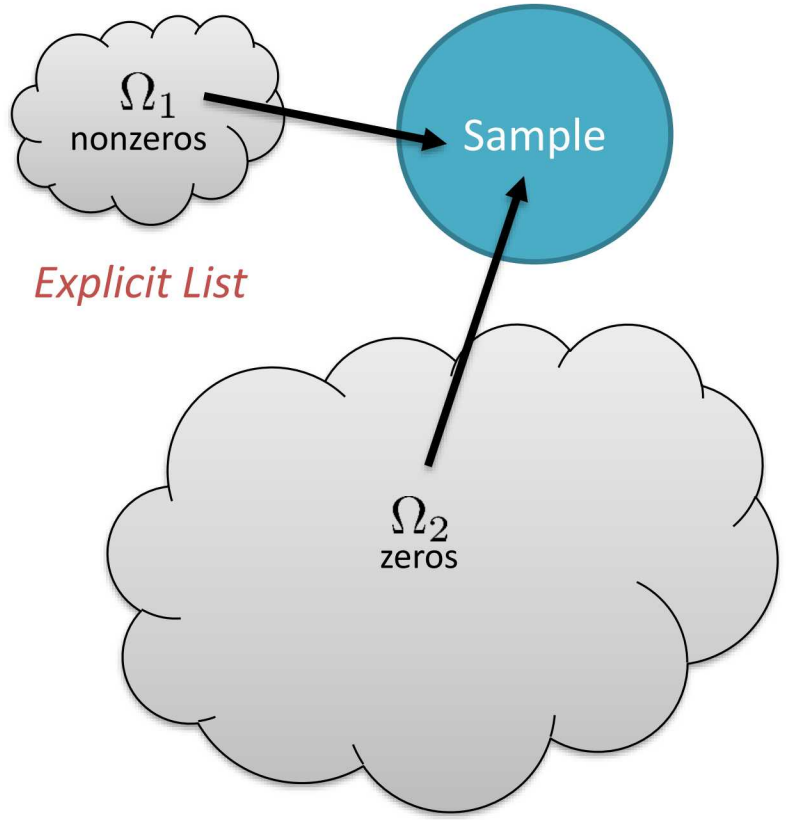
Stratified 0/1 Sampling

Goal: Random ***sparse*** tensor of size n^d that equals the "Y-tensor" in expectation

For each partition ℓ, sample $s_\ell \ll |\Omega_\ell|$ random tensor entries from Ω_ℓ (with replacement)

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

$$\tilde{y}_i = \tilde{s}_i \cdot \frac{|\Omega_\ell|}{s_\ell} \cdot y_i \quad \text{where} \quad i \in \Omega_\ell$$



Theory

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

Proof: $\mathbb{E}[\tilde{s}_i] = \frac{s_\ell}{|\Omega_\ell|} \quad \text{where} \quad i \in \Omega_\ell$

$$\mathbb{E}[\tilde{y}_i] = \mathbb{E}[\tilde{s}_i] \cdot \frac{|\Omega_\ell|}{s_\ell} \cdot y_i = y_i$$

Semi-Stratified 0/1 Sampling

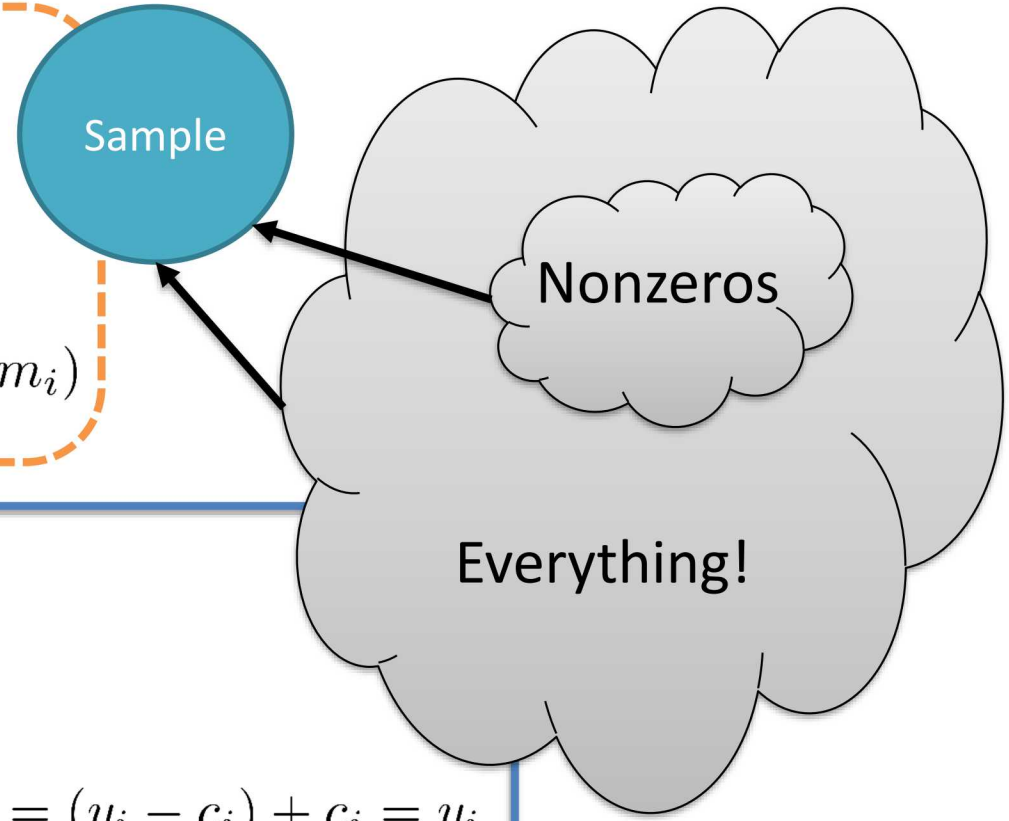
Goal: Random ***sparse*** tensor of size n^d that equals the "Y-tensor" in expectation

Sample s random *nonzero* tensor entries (with replacement) and q random entries (with replacement) and *assume zero*

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

$$\tilde{q}_i = \#\text{ times } i \text{ sampled as "zero"}$$

$$\tilde{y}_i = \tilde{s}_i \cdot \frac{\text{nnz}(\mathcal{X})}{s} \cdot (y_i - c_i) + \tilde{q}_i \cdot \frac{n^d}{q} c_i \quad \text{where} \quad c_i = f'(0, m_i)$$



Theory

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

Proof: if $x_i = 0$, $\mathbb{E}[\tilde{y}_i] = \mathbb{E}[\tilde{q}_i] \cdot \frac{n^d}{q} c_i = \frac{q}{n^d} \cdot \frac{n^d}{q} y_i = y_i$

if $x_i \neq 0$, $\mathbb{E}[\tilde{y}_i] = \mathbb{E}[\tilde{s}_i] \cdot \frac{\text{nnz}(\mathcal{X})}{s} \cdot (y_i - c_i) + \mathbb{E}[\tilde{q}_i] \cdot \frac{n^d}{q} c_i = (y_i - c_i) + c_i = y_i$

GCP with Stochastic Optimization

- Adam (Kingma & Ba, 2015) with default parameters and a few tweaks
  - Use stochastic gradient with a small number of *new* samples at each iteration
  - Group iterations into epochs of 1000 iterations
  - Estimate function values using a large and *fixed* set of sampled indices after each epoch
    - If function value ceases to improve, reduce learning rate ($\alpha = 0.001$) by a factor of 10
    - Once function value ceases to improve again, quit

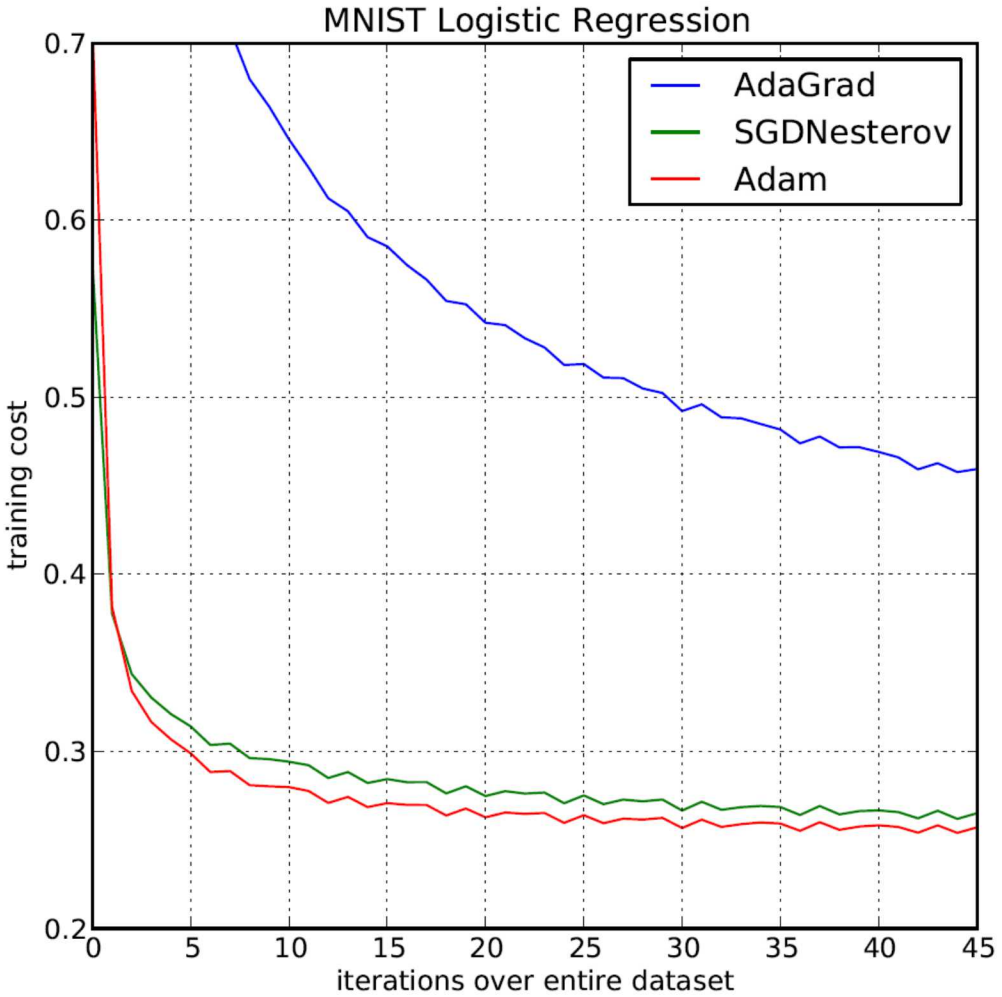
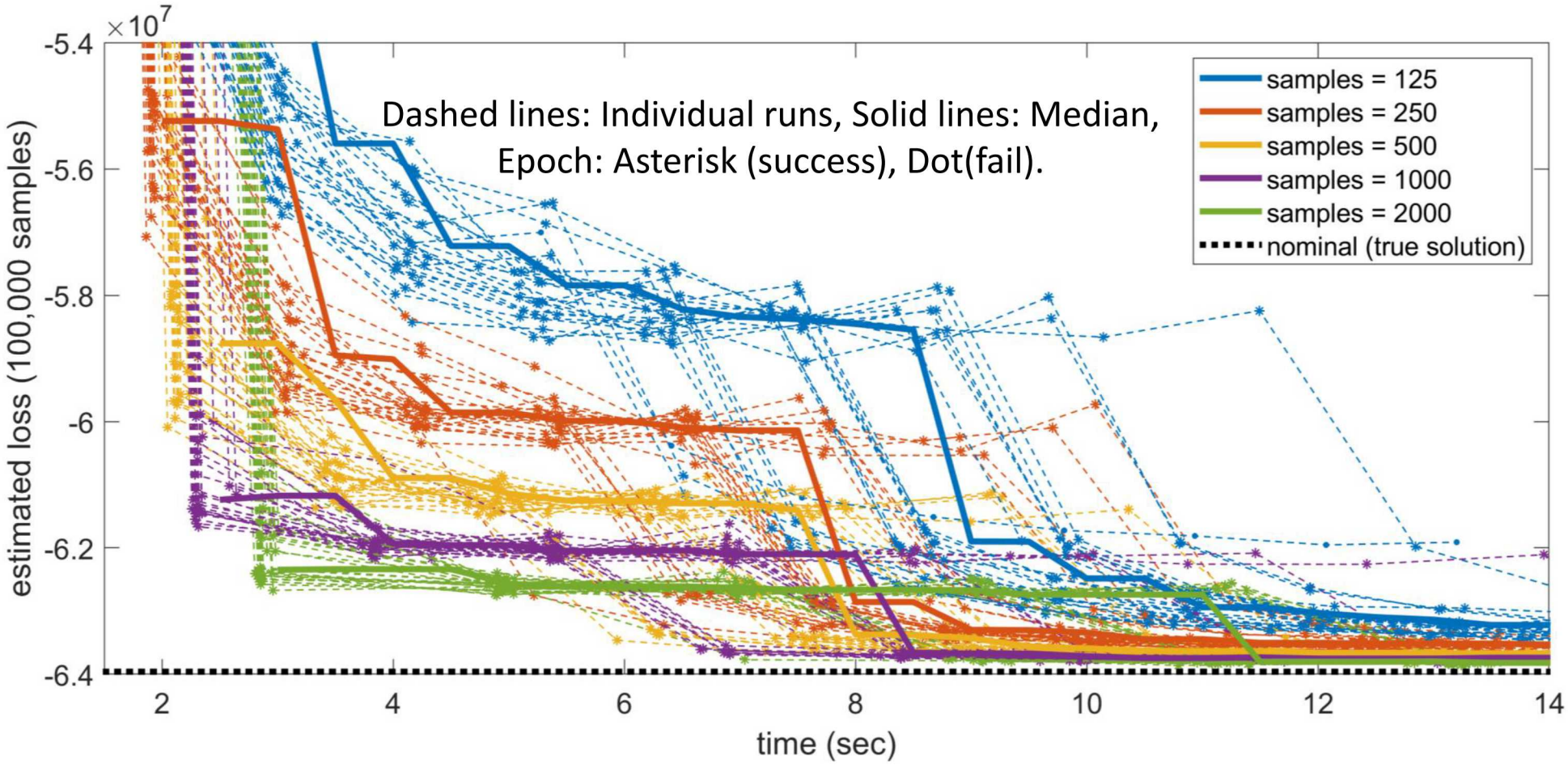


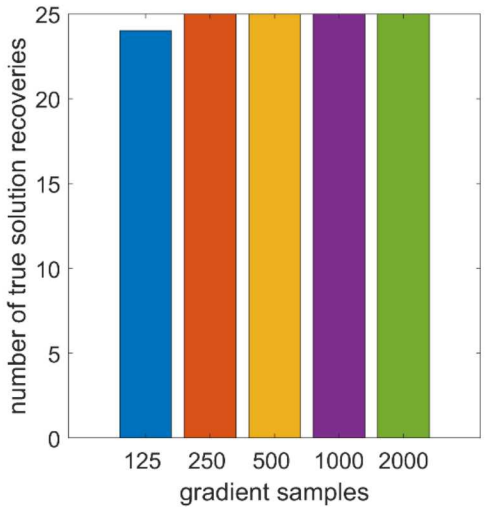
Image from Kingma & Ba, arXiv:1412.6980v9

Example on Gamma-Distributed Data

$200 \times 150 \times 100 \times 50$ Tensor with low-rank ($r = 5$) structure based on Gamma distribution ($k = 1$, θ from model).
Gamma loss: $f(x, m) = \frac{x}{m} + \log m$. Running stochastic GCP with 25 random starts and varying numbers of samples.



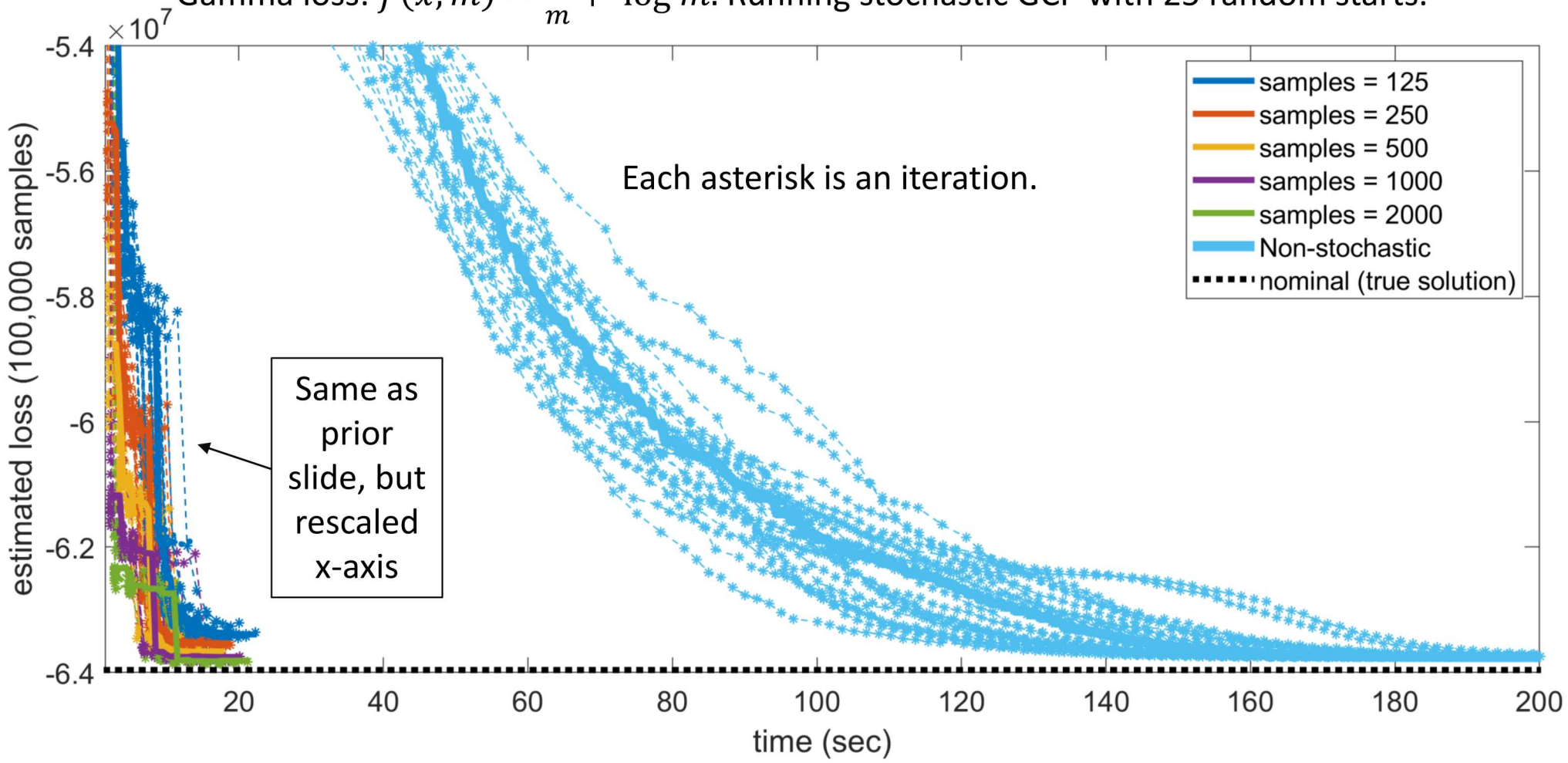
Success at Recovering Underlying Generative Factors



Stochastic vs. Non-Stochastic

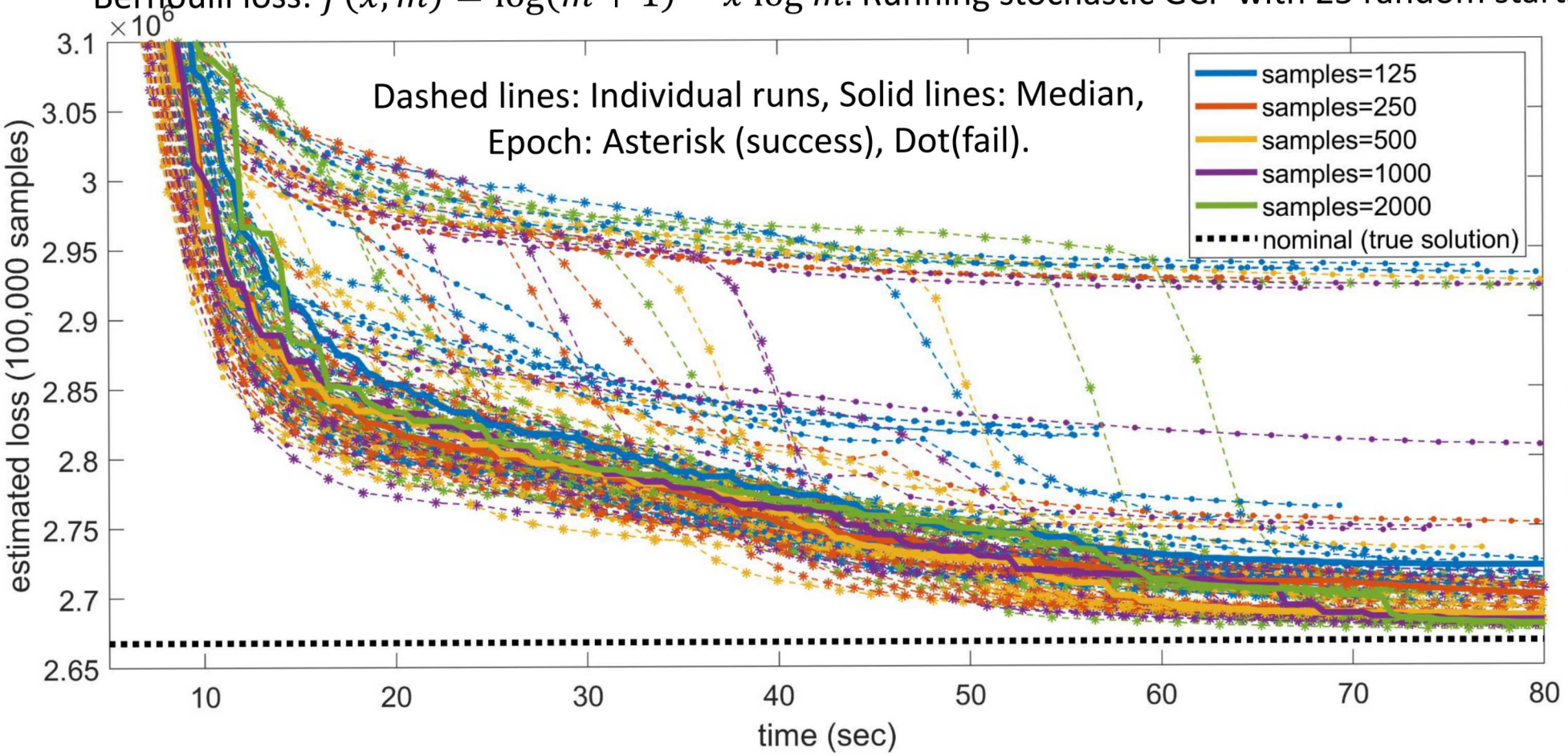
$200 \times 150 \times 100 \times 50$ Tensor with low-rank ($r = 5$) structure based on Gamma distribution ($k = 1$, θ from model).

Gamma loss: $f(x, m) = \frac{x}{m} + \log m$. Running stochastic GCP with 25 random starts.

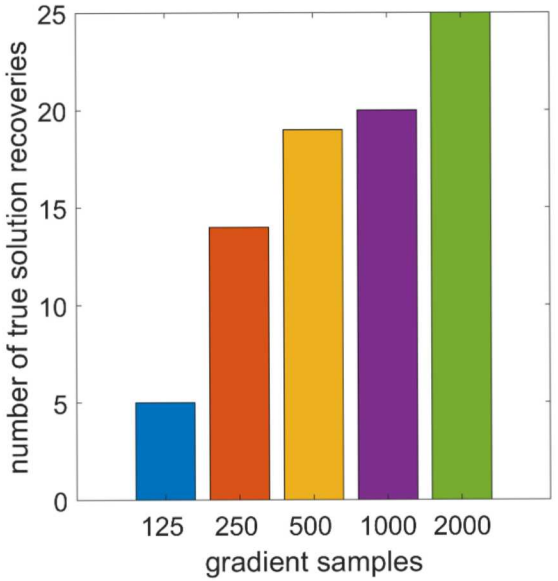


Example on Bernoulli-Distributed Data

$200 \times 150 \times 100 \times 50$ Tensor with low-rank ($r = 5$) structure based on Bernoulli distribution (odds from model).
Sparse tensor, less than 0.35% dense (~500K nonzeros).
Bernoulli loss: $f(x, m) = \log(m + 1) - x \log m$. Running stochastic GCP with 25 random starts, varying # of samples.

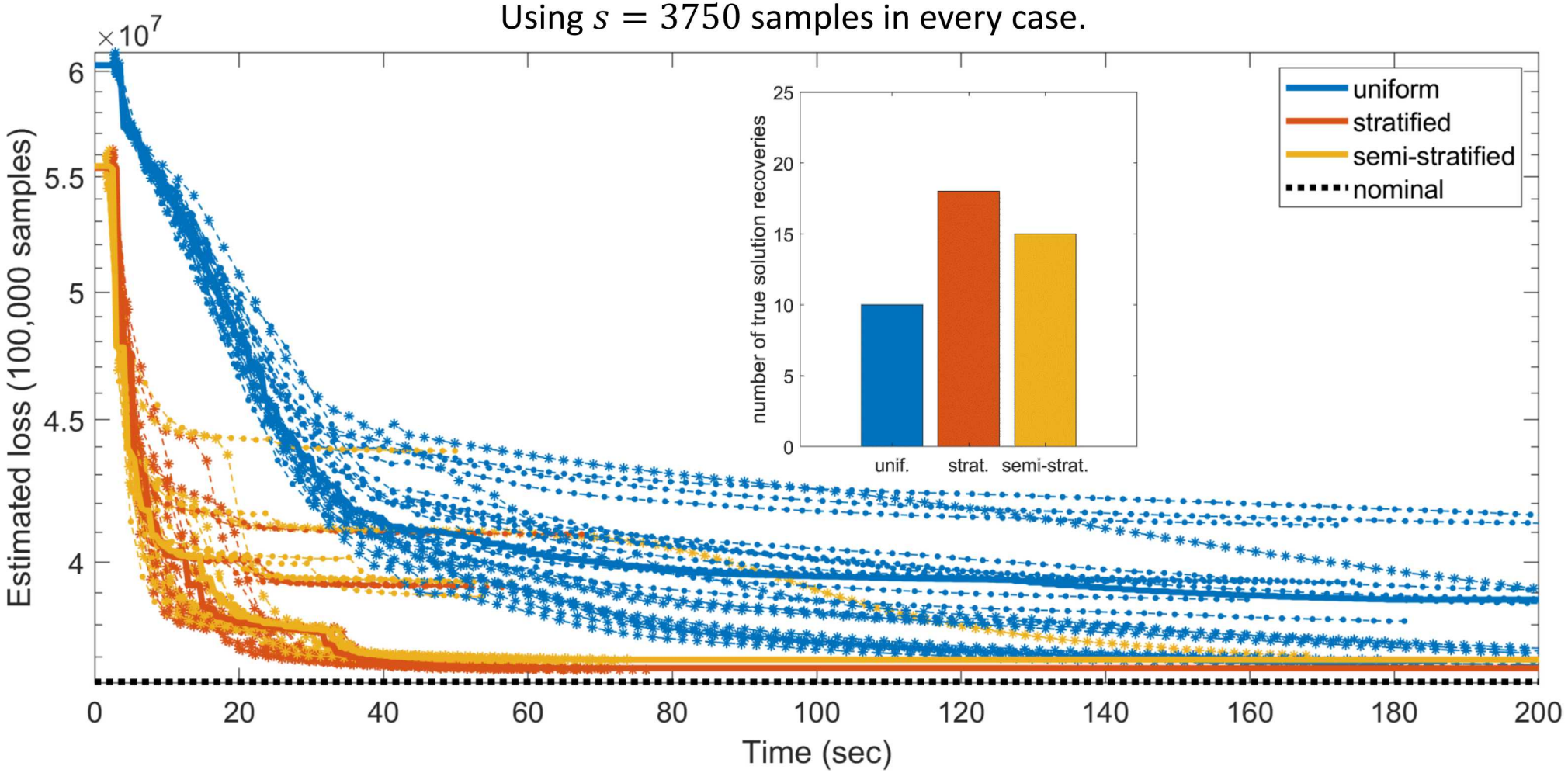


Success at Recovering Underlying Generative Factors



Uniform Sampling is Worse than Stratified for Sparse Tensors

Same set-up as binary experiments, but bigger tensor: $300 \times 225 \times 150 \times 75$, 0.38% dense (800M nonzeroes).
Using $s = 3750$ samples in every case.

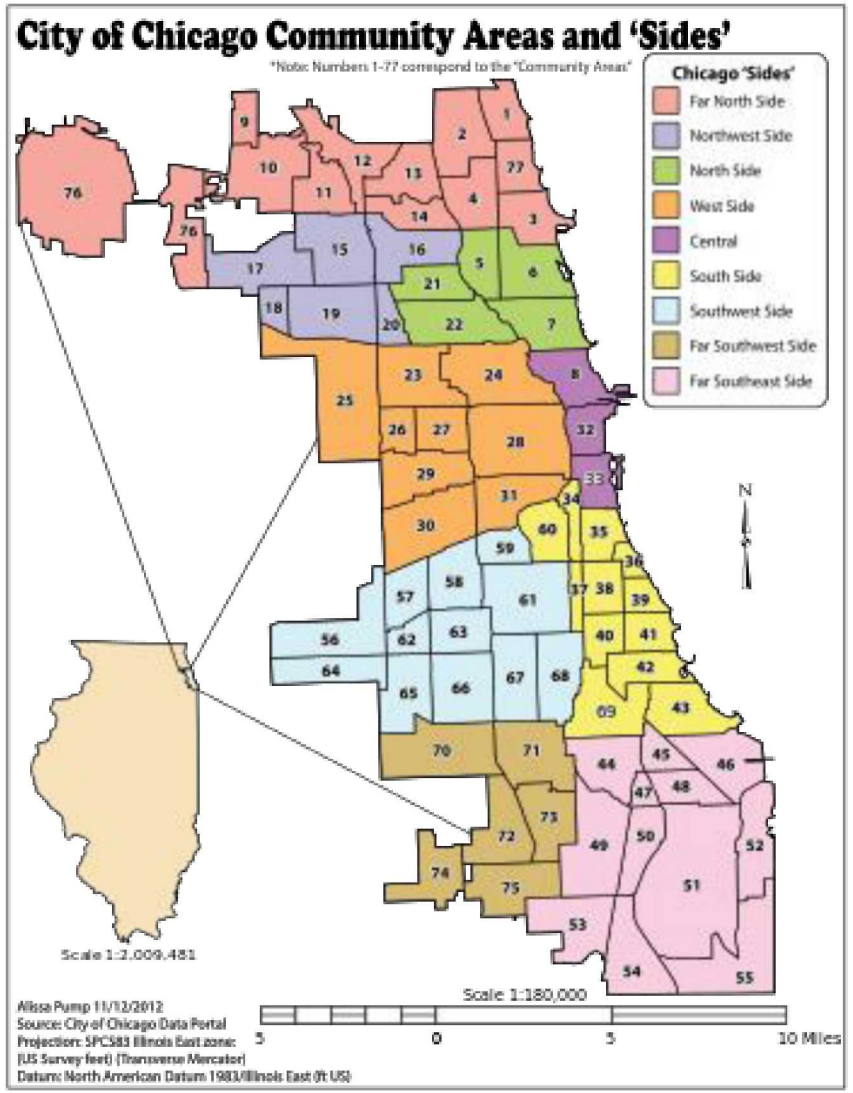


Chicago Crime Data

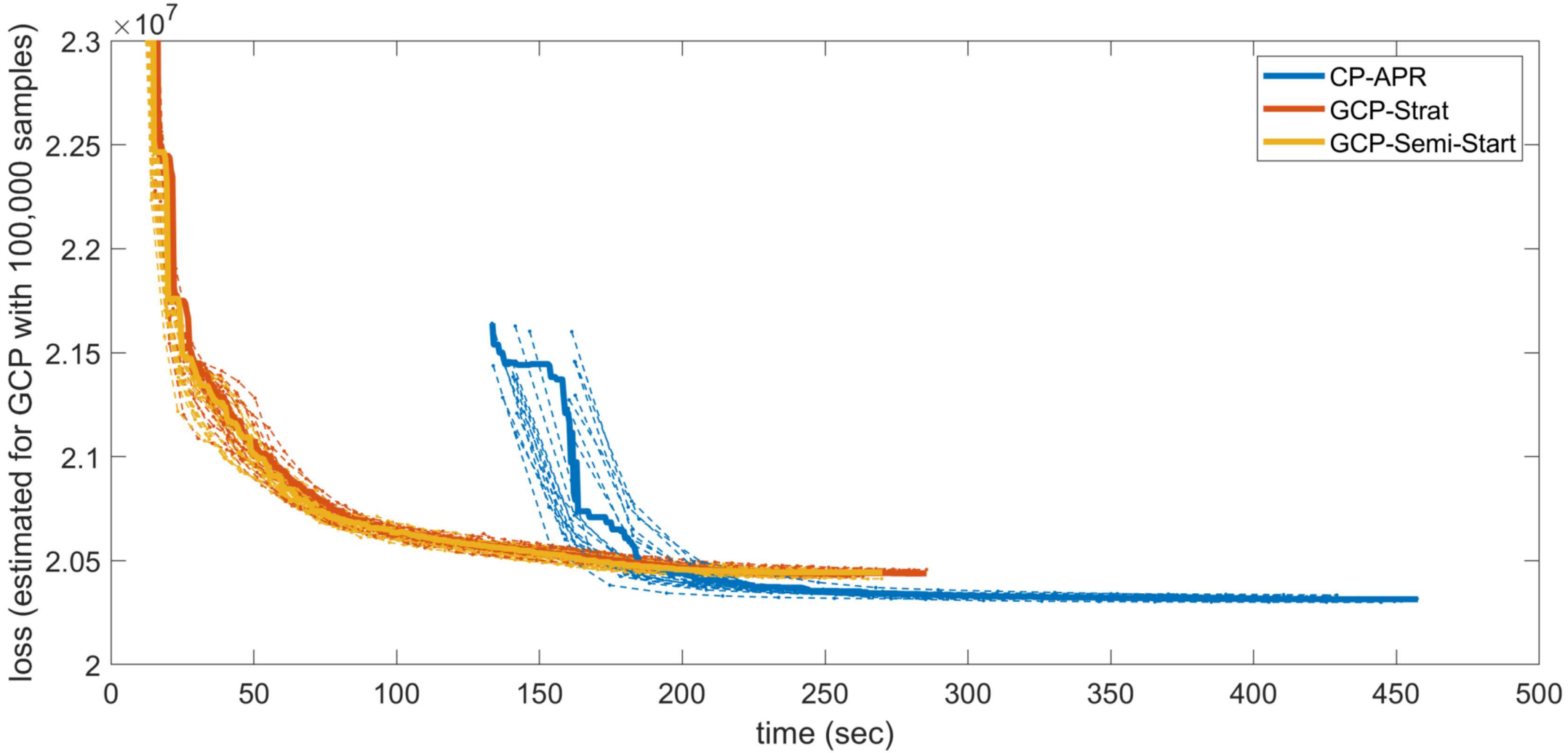
- 4-way count tensor
  - 6,186 Days
  - 24 Hours of the Day
  - 77 Community Areas
  - 32 Crime Types
- Non-zeros: 5,330,673
  - Storage: 0.21GB for sparse tensor
- Distribution of entries
  - 0: 98.54%
  - 1: 1.33%
  - ≥ 2: 0.12%
- Using binary version (every nonzero changed to 1)
- Obtained from FROSTT (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-Count
Rank = 10
$s = 6{,}319$

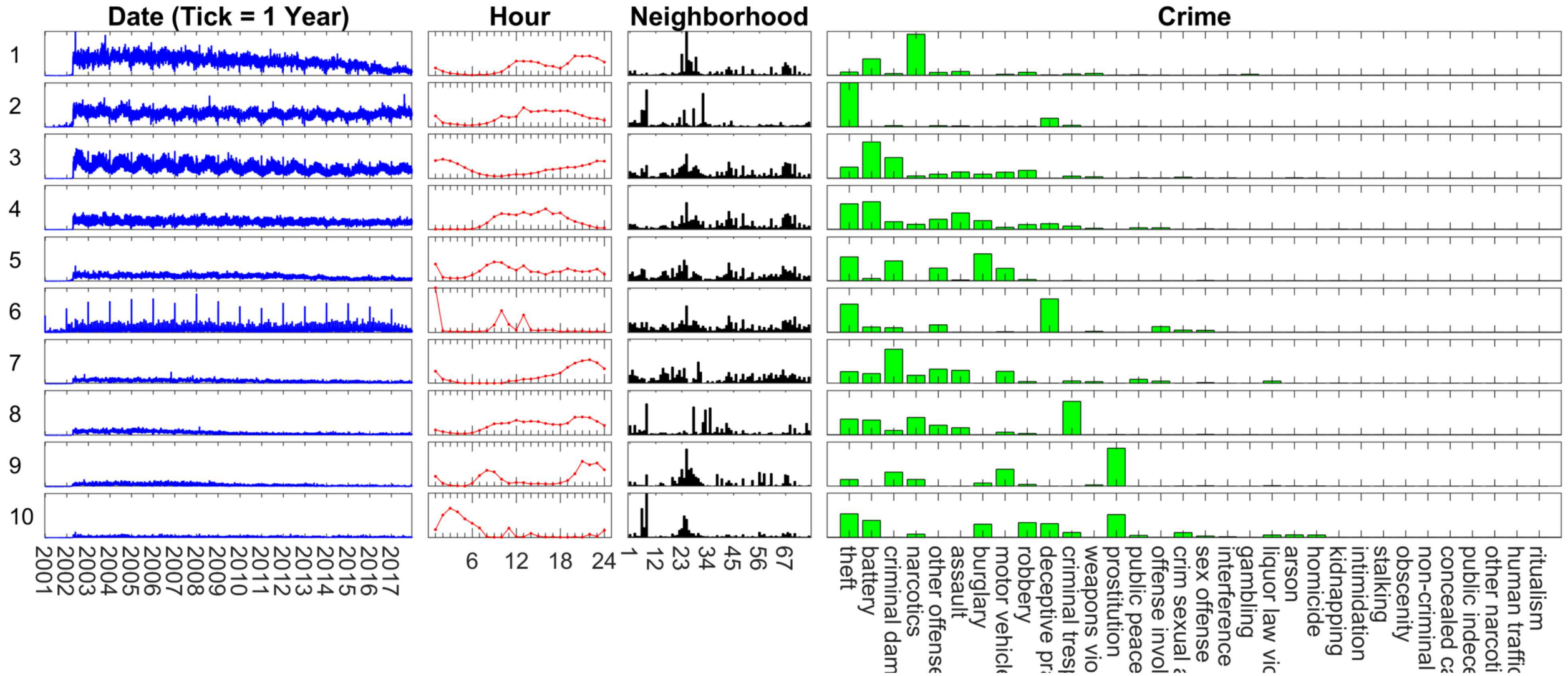
$$f(x, m) = m - x \log m$$



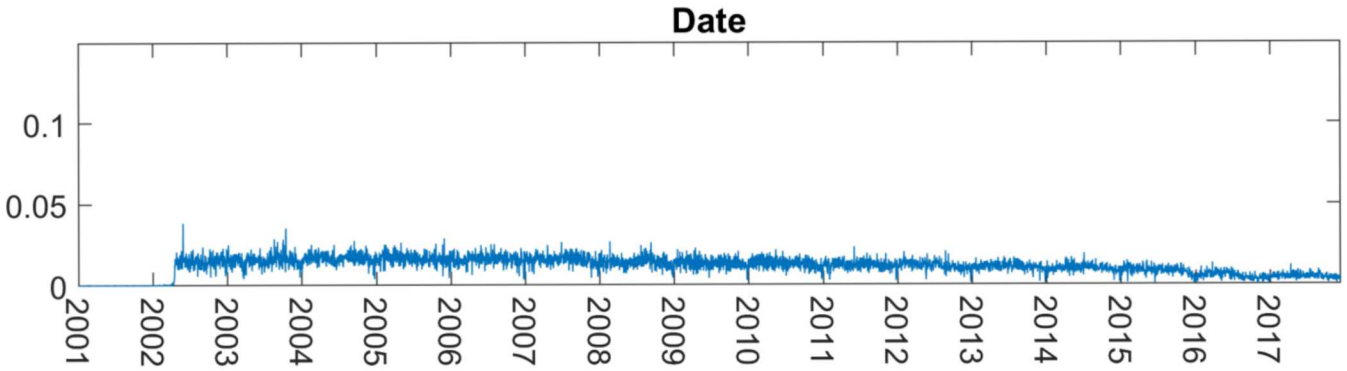
GCP-Adam versus CP-APR

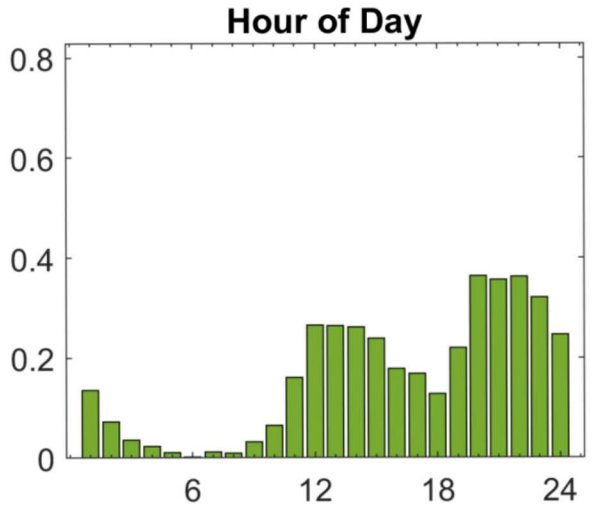


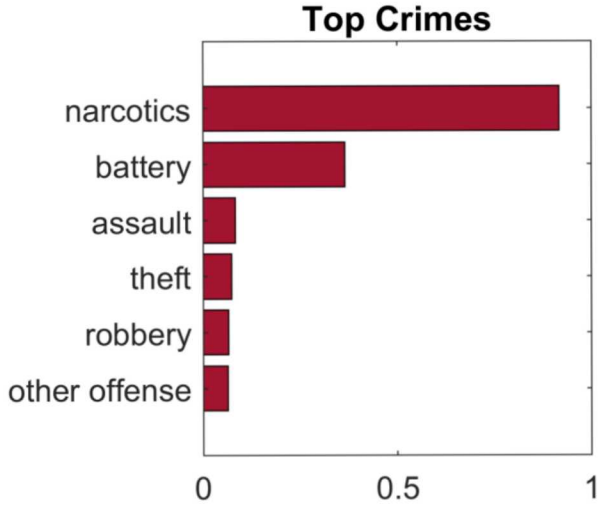
Application to Sparse Crime Binary Tensor (Semi-stratified results)

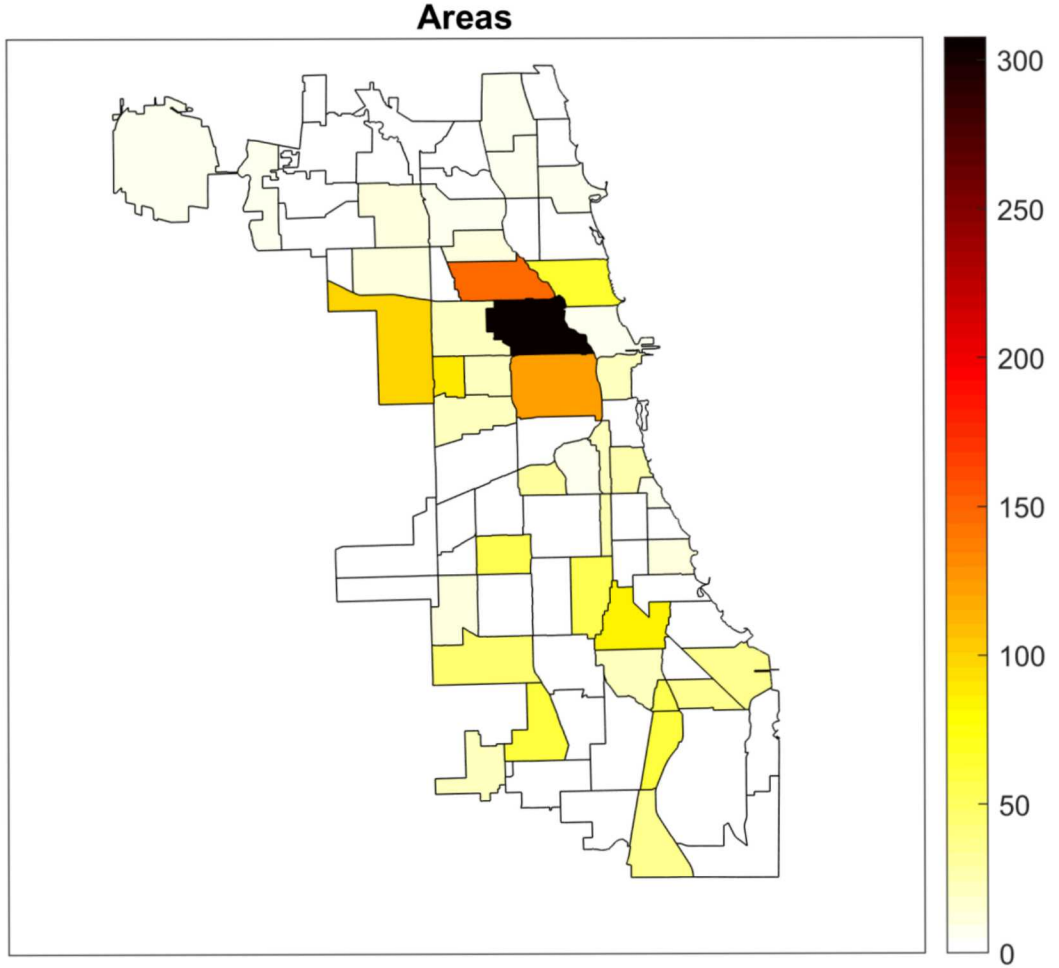


Component #1

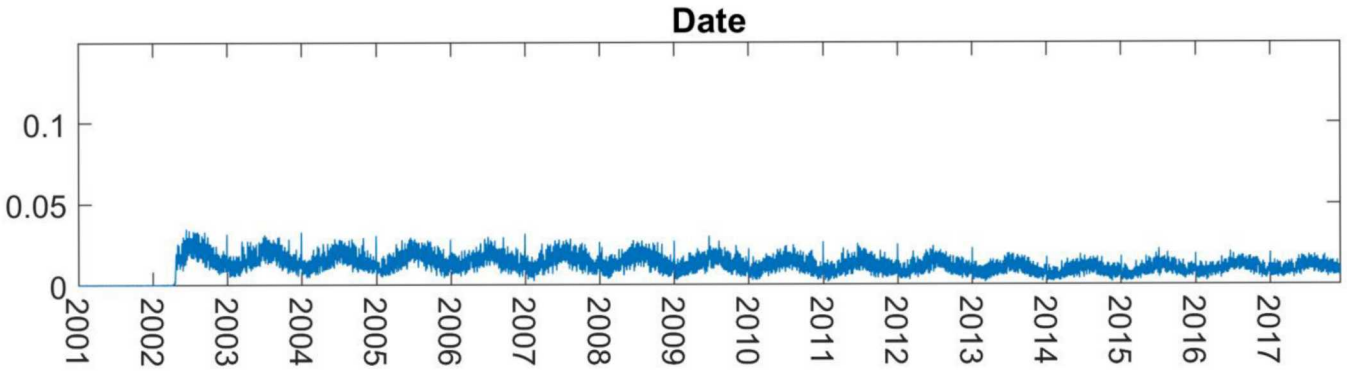

Date
0.1
0.05
0
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017

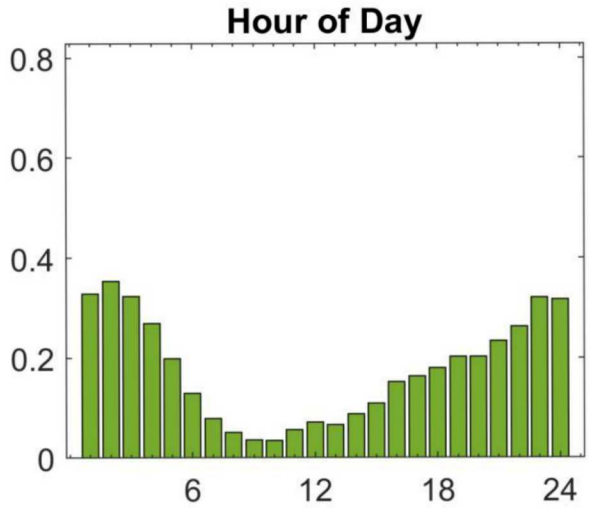

Hour of Day
0.8
0.6
0.4
0.2
0
6
12
18
24

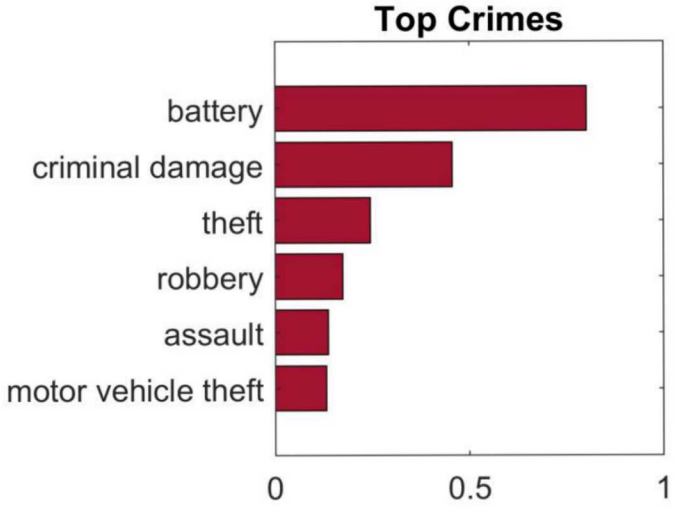

Top Crimes
narcotics
battery
assault
theft
robbery
other offense
0
0.5
1

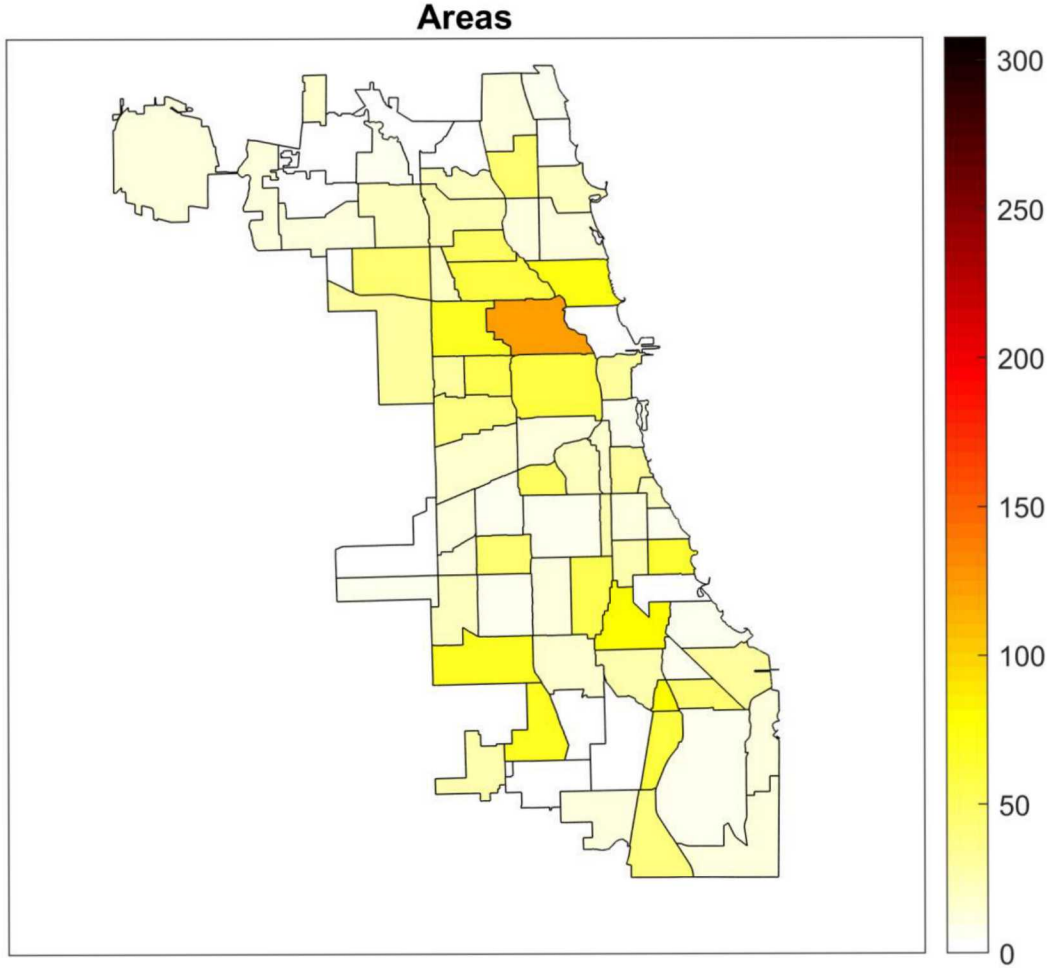

Areas
300
250
200
150
100
50
0

Component #3

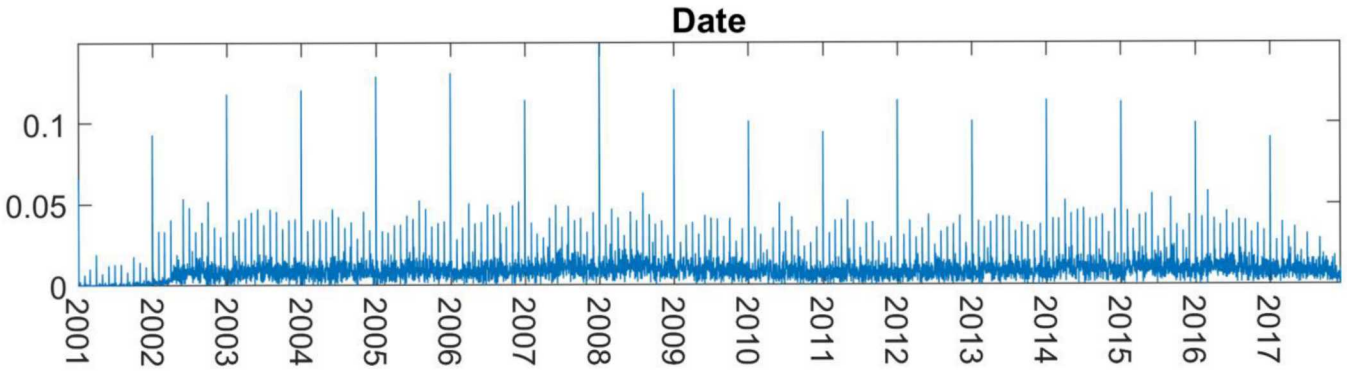

Date
0.1
0.05
0
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017

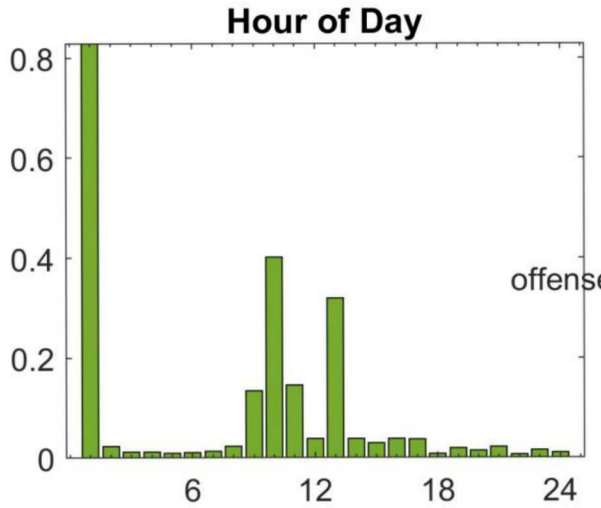

Hour of Day
0.8
0.6
0.4
0.2
0
6
12
18
24

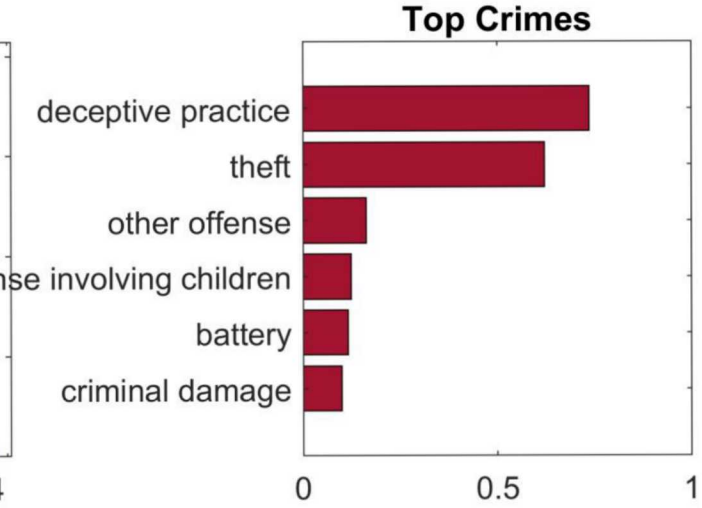

Top Crimes
battery
criminal damage
theft
robbery
assault
motor vehicle theft
0
0.5
1

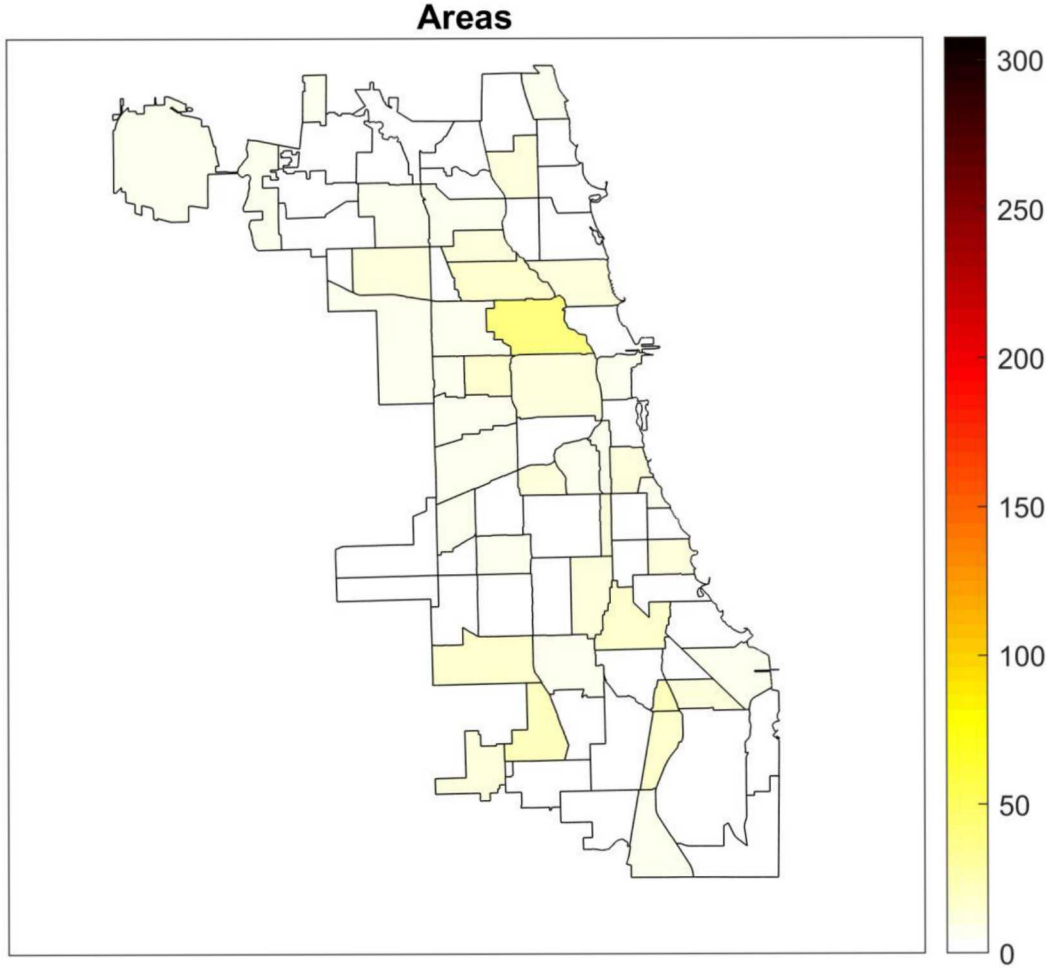

Areas
300
250
200
150
100
50
0

Component #6


Date
0.1
0.05
0
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017

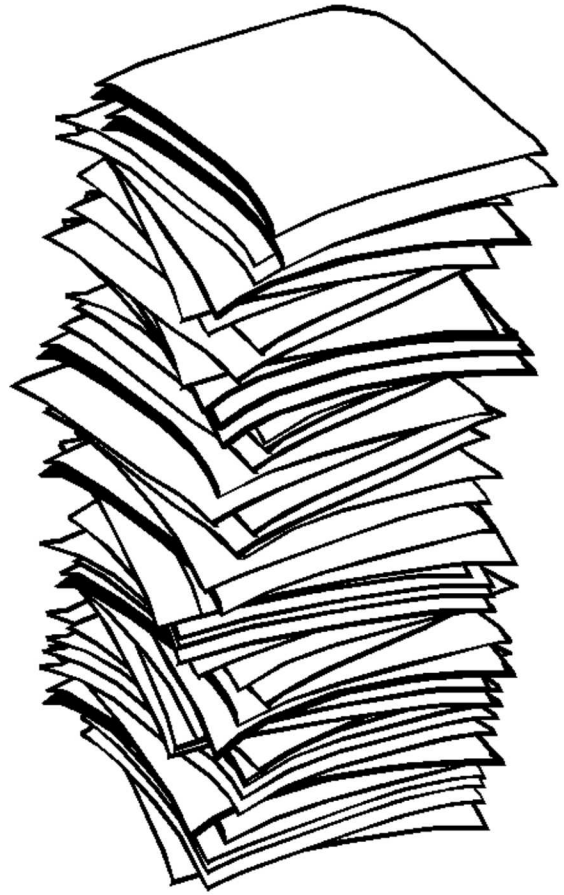

Hour of Day
0.8
0.6
0.4
0.2
0
6
12
18
24


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


Areas
300
250
200
150
100
50
0

Related Work

- SGD for Matrix Decomposition
  - **Gemulla, Nijkamp, Hass, Sismanis, 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, Chin, Juan, and Lin, RecSys'13** – Fast Parallel SGD (FPSGD) method: Matrix factorization in shared memory environment. No theoretical analysis. ***Only uses nonzero entries.***
- SGD for Tensor Decomposition
  - **Mardani, Mateos, Giannakis, IEEE TSP 2015** – OnlineCP uses SGD for tensors that are streaming, one slice at a time
  - **Maehara, Hayashi, Kawarabayashi, AAAI-16** – Tensor can be written as sum or average of a number of tensors. Proposes SGD plus several variations
  - **Ge, Huang, Jin, and Yuan, CoLT 2015** consider SGD for symmetric tensor decomposition
- Tensor Sketching
  - **Acar, Dunlavy, Kolda, Morup (CILS 2011)** – For dense tensors, it is heuristically possible to recover a full tensor decomposition with only a sketch of the data
  - **Jain and Oh (NIPS 2014)** and **Bhojanapalli and Sanghavi (arXiv 2015)** more formally prove under what conditions sketching works, albeit with a focus on orthogonal symmetric tensor decomposition



Conclusions, Future Work, References

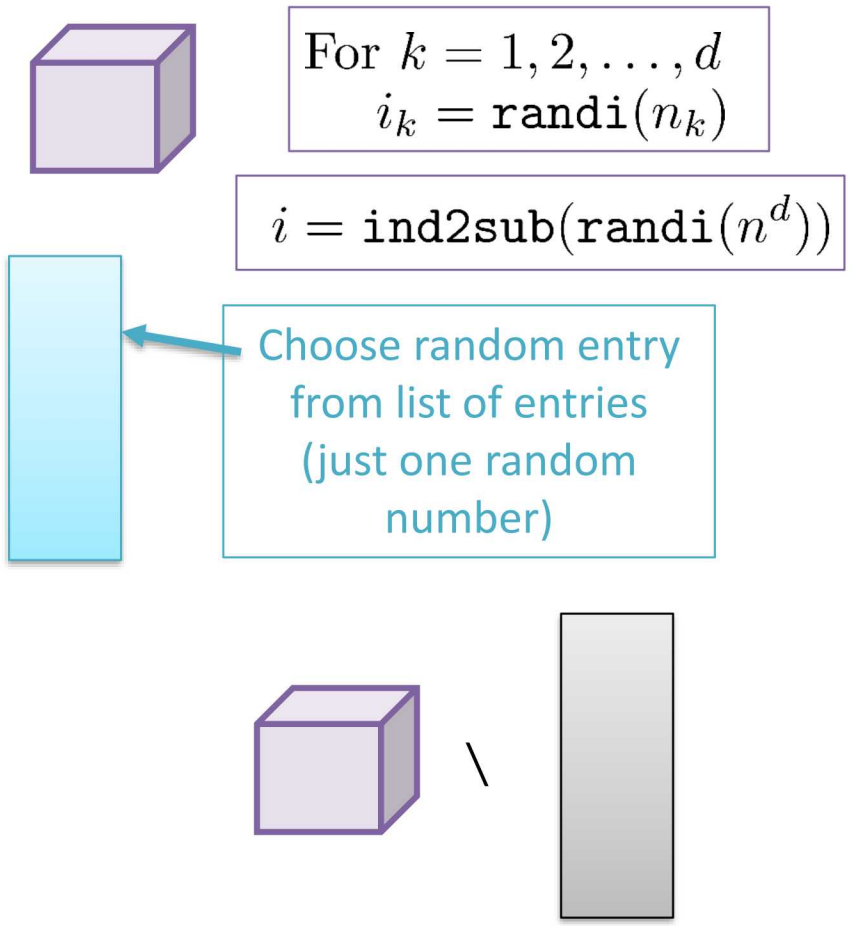
- GCP enables alternative loss functions, but…
  - Not amenable to scaling because gradient "dense"
  - Developed GCP stochastic gradient
  - With variations of stratified sampling for sparse tensors
- Future work
  - Release for MATLAB Tensor Toolbox
  - Parallel implementation (with Eric Phipps – GenTen)
  - Distributed implementation (with Karen Devine)
- References
  - D. Hong, T. G. Kolda, J. A. Duersch. **Generalized Canonical Polyadic Tensor Decomposition**. arXiv:1808.07452, 2018 (to appear in *SIAM Review*)
  - T. G. Kolda, D. Hong, J. A. Duersch. **Stochastic Optimization for Large-Scale Tensor Decomposition**, in preparation

Efficient Sampling

- Implicit sample, no exclusion list
  - Ex: Randomly sampling from a large tensor, no stratification
  - Requires d random numbers per sample
  - But can do with just 1 random number if $n^d < 2^{64}$

- Explicit list of indices
  - Ex: Randomly sampling from explicit list of known entries (scarce tensor)
  - Ex: Randomly sampling nonzeros
  - Requires just one random number per sample

- Implicit indices with explicit exclusion list
  - Ex: Randomly sampling zeros, and the nonzeros are excluded
  - Ex: Randomly sampling from a large tensor, with list of missing entries
  - Can estimate how many samples will be rejected and generate enough extra samples so that we have enough samples with 99% probability



Generate implicit sample and *reject* if in exclusion list