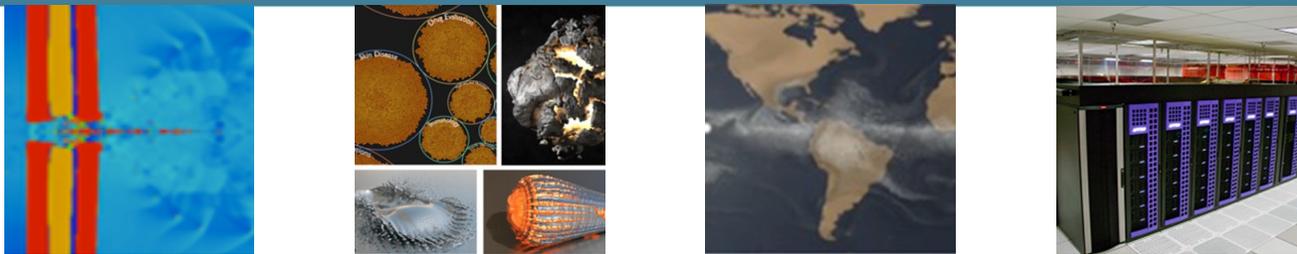




Streaming Generalized Canonical Polyadic Tensor Decompositions

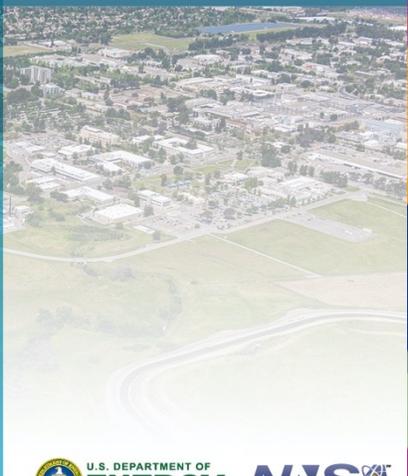


Eric Phipps (etphipp@sandia.gov): Sandia National Laboratories

Nick Johnson: Cerebras Systems, Inc

Tammy Kolda: MathSci.ai

PASC 23, June 26-28, 2023



Multiway “Tensor” Data is Ubiquitous



Neuron activity:
Neuron x Time x Trial



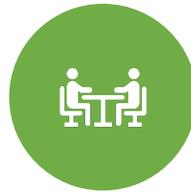
Travel data: Start Location
x Finish Location
x Departure Hour
x Departure Date



Crime data: Crime x Location
x Hour x Date



Signal processing:
Sensor x Frequency x Time



Social interaction data:
Person A x Person B
x Venue x Time



Cyber data: Src IP x Dst IP
x Dst Port x Time



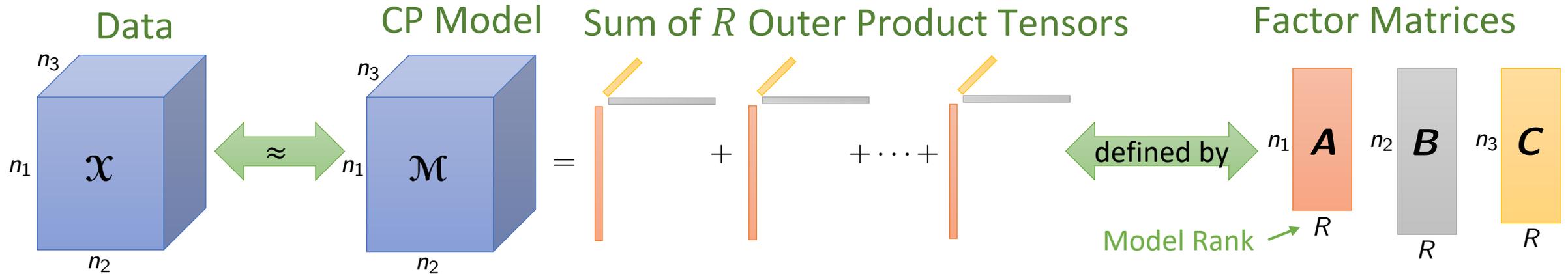
Twitter co-occurrence:
Term A x Term B x Time



Host data: Host
x Action/Library x Time

Tensor Decomposition Finds
Patterns in Multiway Data

Canonical Polyadic (CP) Tensor Decomposition



$$x(i, j, k) \approx m(i, j, k) = \sum_{l=1}^R a(i, l)b(j, l)c(k, l) = \sum_{l=1}^R \lambda_l \hat{a}(i, l)\hat{b}(j, l)\hat{c}(k, l)$$

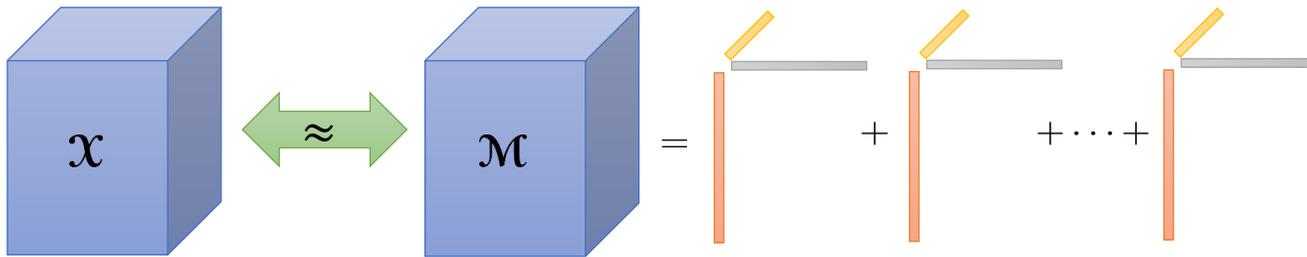
Order sum with decreasing λ

Vectors comprising each factor describe patterns in the data

$$\mathcal{X} \approx \mathcal{M} \equiv [\mathbf{A}, \mathbf{B}, \mathbf{C}], \quad \min_{[\mathbf{A}, \mathbf{B}, \mathbf{C}]} \|\mathcal{X} - \mathcal{M}\|_F^2 \Leftrightarrow \min_{[\mathbf{A}, \mathbf{B}, \mathbf{C}]} \sum_{i, j, k} (x(i, j, k) - m(i, j, k))^2$$

Traditional CP model computed by solving a nonlinear least-squares problem

Generalized CP (GCP) Tensor Decomposition Allows Flexible Loss Function



Generalized CP (GCP)

$$\min_{\mathcal{M}} F(\mathcal{X}, \mathcal{M}) = \sum_i f(x_i, m_i)$$

$$\text{s.t. } \mathcal{M} = \llbracket \mathbf{A}, \mathbf{B}, \mathbf{C} \rrbracket$$

Example Loss Functions

Normal ($x, m \in \mathbb{R}$)

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

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

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

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

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

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

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



$$\min_{\mathcal{M}} F(\mathcal{X}, \mathcal{M}) = \sum_i f(x_i, m_i) \quad \text{s.t.} \quad \mathcal{M} = \llbracket \mathbf{A}, \mathbf{B}, \mathbf{C} \rrbracket$$

Lose the least-squares structure underlying traditional CP solution algorithms

- Instead pursue gradient-based optimization approach

Cost of gradient computation prohibitive for large, sparse tensors

- Cost grows with total tensor size $O(n_1 n_2 n_3)$ (i.e., exponential in the number of dimensions), regardless of sparsity

Instead, employ Stochastic Gradient Descent (SGD) where gradient is randomly sampled

- Specialized sampling algorithms designed for large sparse tensors (sample contributions from zeros and nonzeros separately)
- ADAM SGD optimization method for robustness

¹Kolda and Hong [Stochastic Gradients for Large-Scale Tensor Decomposition](#), SIMODS, 2020.

ArXiv Abstract Data



Kaggle provides metadata for ArXiv abstracts that is regularly updated

- <https://www.kaggle.com/datasets/Cornell-University/arxiv>
- Publication date, going back to first paper backdated to 1986
- Paper category (e.g., cs.LG:Machine Learning)
- Abstracts text

Process abstracts into unique words using Spacy's English core parser

- <https://spacy.io/>
- Resulted in $\sim 25\text{K}$ unique words

For each month, construct 2-way tensor (matrix):

- Mode 1: Paper category
- Mode 2: Word
- Value: Number of abstracts of the given category containing the given word

Collection of matrices for each month since start-date forms a sparse 3-way tensor

- Start date is 10 years after first paper (1986) to allow for sufficient volume
- $172 \times 24558 \times 300$, $\sim 2.4\%$ sparsity

CORNELL UNIVERSITY AND 4 COLLABORATORS · UPDATED 2 DAYS AGO

1055 New Notebook Download (1 GB)

arXiv Dataset

arXiv dataset and metadata of 1.7M+ scholarly papers across STEM

arXiv

Data Card Code (84) Discussion (45)

About Dataset

Usability 8.75

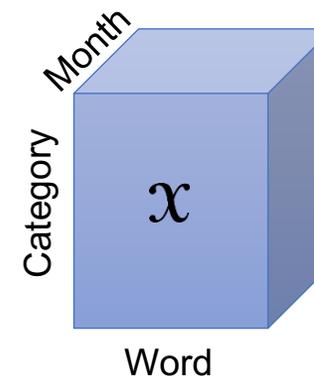
About ArXiv

For nearly 30 years, ArXiv has served the public and research communities by providing open access to scholarly articles, from the vast branches of physics to the many subdisciplines of computer science to everything in between, including math, statistics, electrical engineering, quantitative biology, and economics. This rich corpus of information offers significant, but sometimes overwhelming depth.

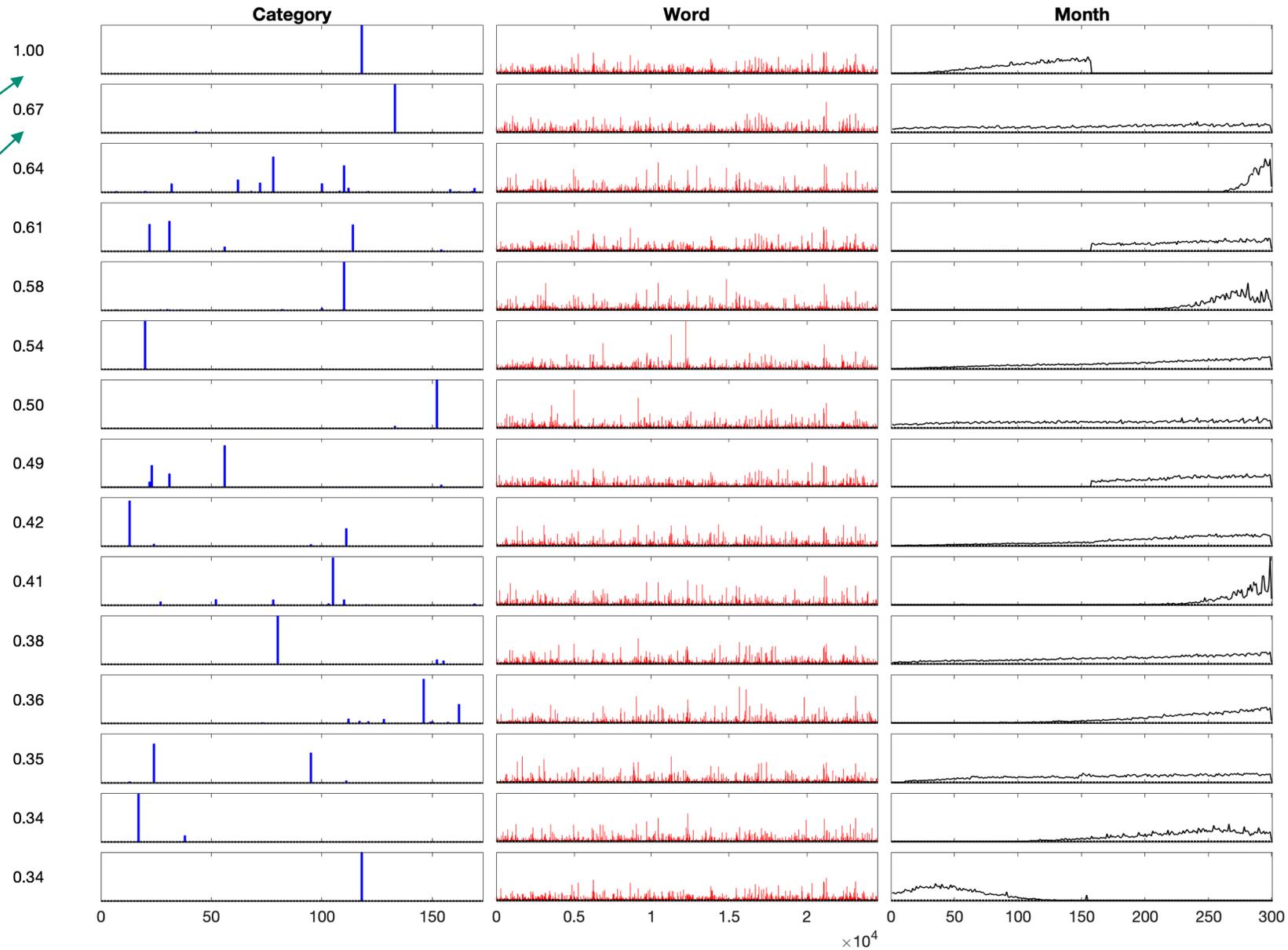
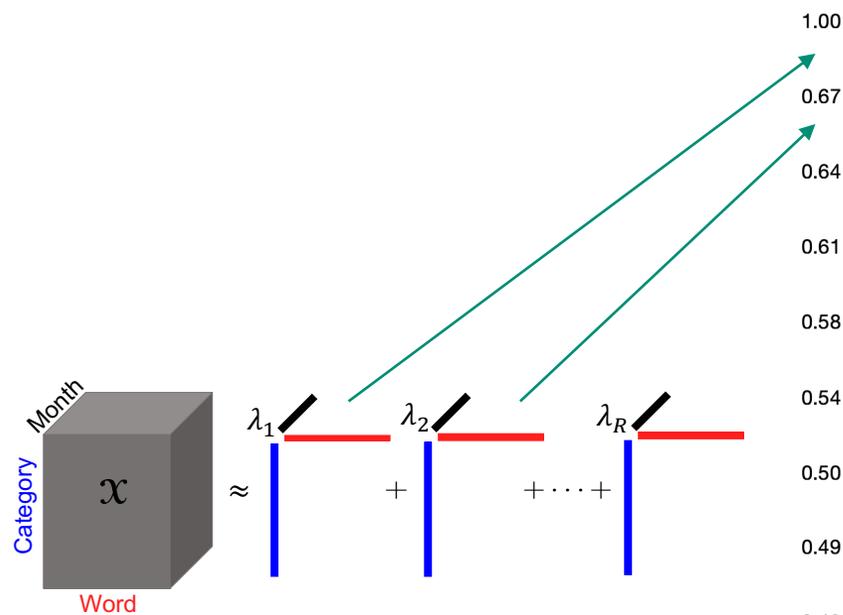
License CC0: Public Domain

Expected update frequency Monthly

In these times of unique global challenges, efficient extraction of insights from data is essential. To help make the arXiv more accessible, we present a free, open pipeline on Kaggle to the machine-readable arXiv dataset: a repository of 1.7 million articles, with relevant features such as article titles, authors, categories, abstracts, full text PDFs, and more.



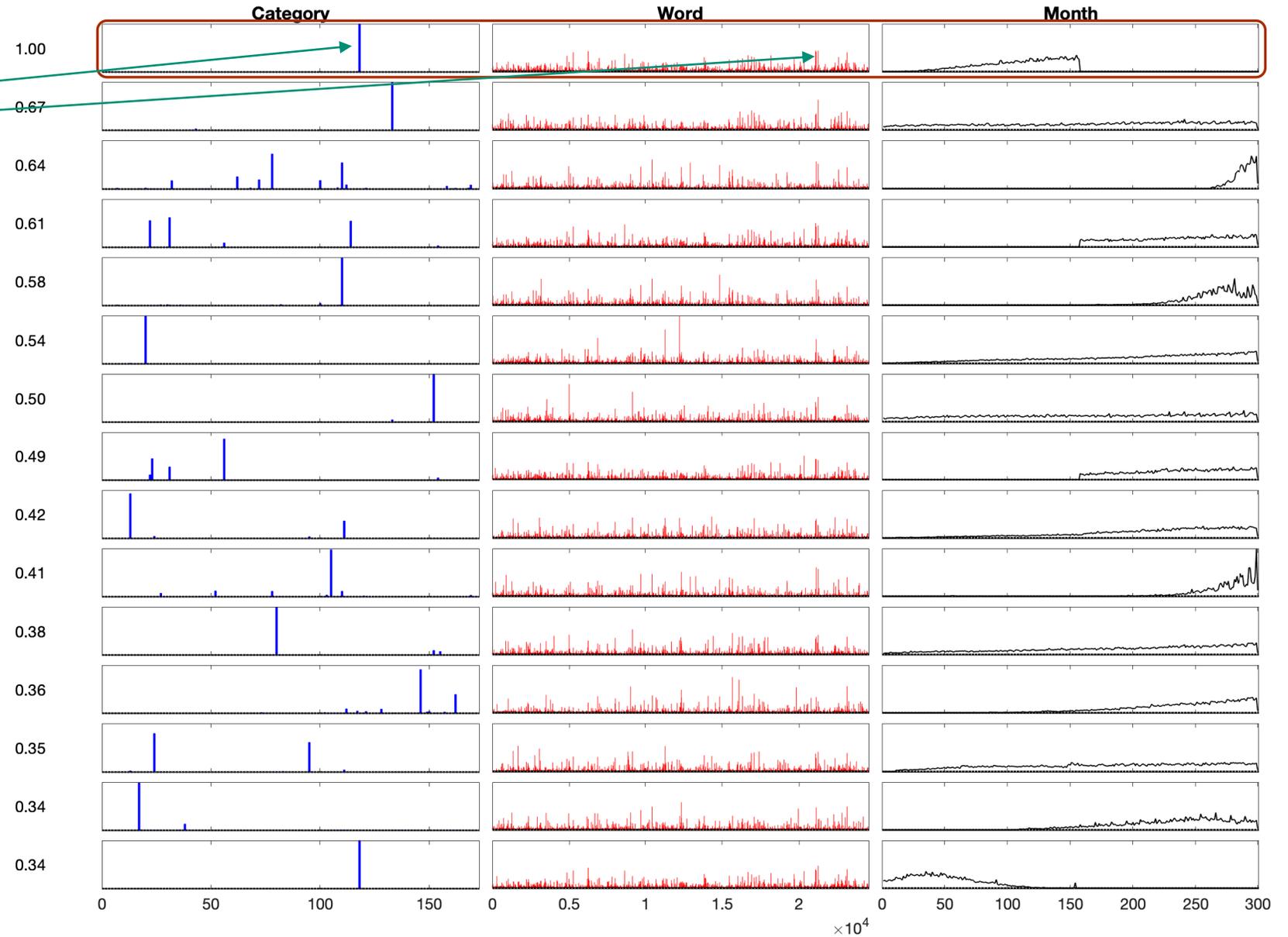
Top 15 (of 50) ArXiv CP Factors (GCP, Poisson Loss)



Component #1 (of 50): Astrophysics



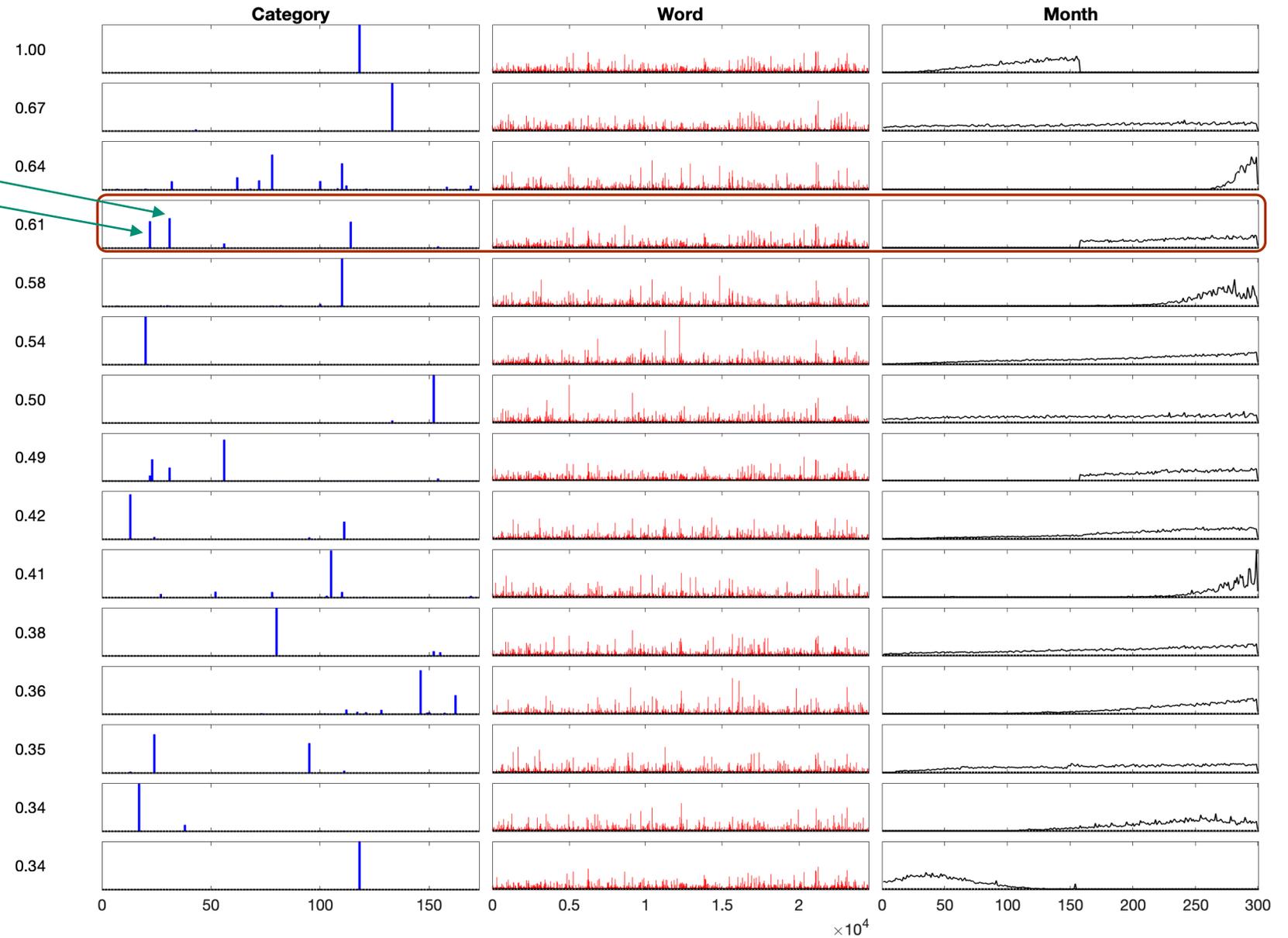
Top Categories	Category Weight	Top Words	Word Weight
astro-ph	1.00	find	1.00
		model	1.00
		present	0.97
		use	0.94
		star	0.92
		high	0.92
		result	0.92
		galaxy	0.85
		observation	0.77
		large	0.77
		observe	0.75
		mass	0.72
		datum	0.69
		low	0.68
		study	0.67
		ray	0.64
		emission	0.64
		time	0.62
		spectrum	0.62
		field	0.60



Component #4 (of 50): Astrophysics Reorganized



Top Categories	Category Weight	Top Words	Word Weight
astro-ph.GA	1.00	find	1.00
astro-ph.HE	0.90	galaxy	0.94
astro-ph.CO	0.88	model	0.90
astro-ph.SR	0.15	high	0.85
		use	0.85
		star	0.75
		large	0.72
		mass	0.72
		result	0.71
		present	0.70
		observation	0.69
		observe	0.64
		study	0.63
		ray	0.62
		datum	0.62
		emission	0.61
		low	0.60
		energy	0.56
		spectrum	0.55
		source	0.54



Streaming Tensor Decompositions



Desire algorithms that can compute GCP decompositions as data streams in

Many formulations of this problem and assumptions on the type of data and processes generating it

Assumptions in this work:

- Updates are processed in discrete batches indexed by time $t = 1, 2, \dots$
- A complete d -way tensor is observed at each time step
- The dimensions of each update are fixed throughout time
- The CP rank is known and fixed
- The number of time steps is unbounded (infinite streaming)

Requires algorithm that can incrementally update the decomposition without revisiting old data

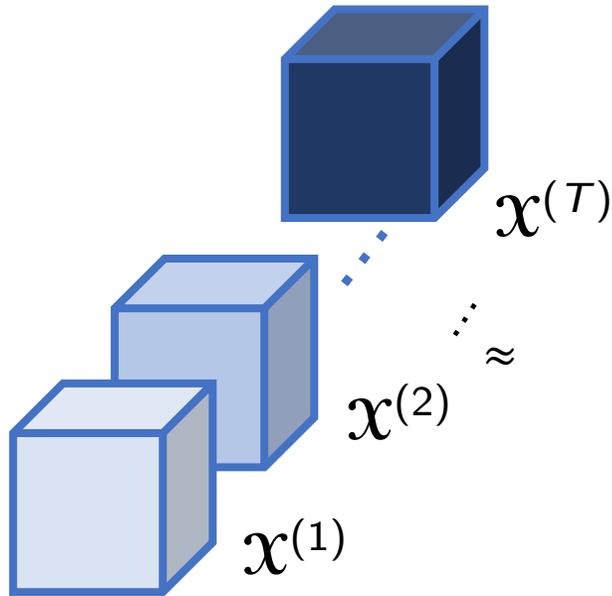
Development of streaming variant of GCP inspired by three prior works (for Gaussian loss):

- Online SGD [[Mardani et al, 2015](#)] – gradient descent updates for factor matrices
- CP-Stream [[Smith et al, 2018](#)] – approximating old data by factor matrices from prior step

Honorable mention:

- OnlineCP [[Zhou et al](#)] – very fast method for incorporating prior time steps (but specific to Gaussian)

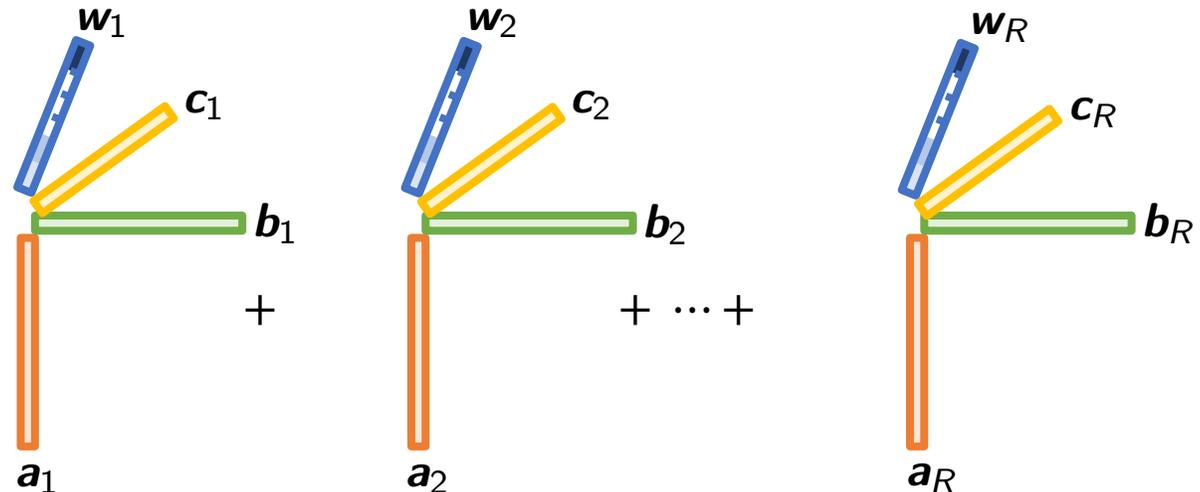
At each time step t , new
3-way hyperslice added



$$\mathcal{X} \in \mathbb{R}^{N_1 \times N_2 \times N_3 \times T}$$

$$\mathcal{X}^{(t)} \in \mathbb{R}^{N_1 \times N_2 \times N_3}$$

If we assume factor matrices \mathbf{A} , \mathbf{B} , and \mathbf{C} fixed
through time, then the ideal factorization looks like...



$$\mathcal{X} \approx \sum_{j=1}^R \mathbf{a}_j \circ \mathbf{b}_j \circ \mathbf{c}_j \circ \mathbf{w}_j = [\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{W}]$$

$$\mathbf{A} = [\mathbf{a}_1 \ \mathbf{a}_2 \ \dots \ \mathbf{a}_R] \in \mathbb{R}^{N_1 \times R}$$

$$\mathbf{B} = [\mathbf{b}_1 \ \mathbf{b}_2 \ \dots \ \mathbf{b}_R] \in \mathbb{R}^{N_2 \times R}$$

$$\mathbf{C} = [\mathbf{c}_1 \ \mathbf{c}_2 \ \dots \ \mathbf{c}_R] \in \mathbb{R}^{N_3 \times R}$$

$$\mathbf{W} = [\mathbf{w}_1 \ \mathbf{w}_2 \ \dots \ \mathbf{w}_R] \in \mathbb{R}^{T \times R}$$

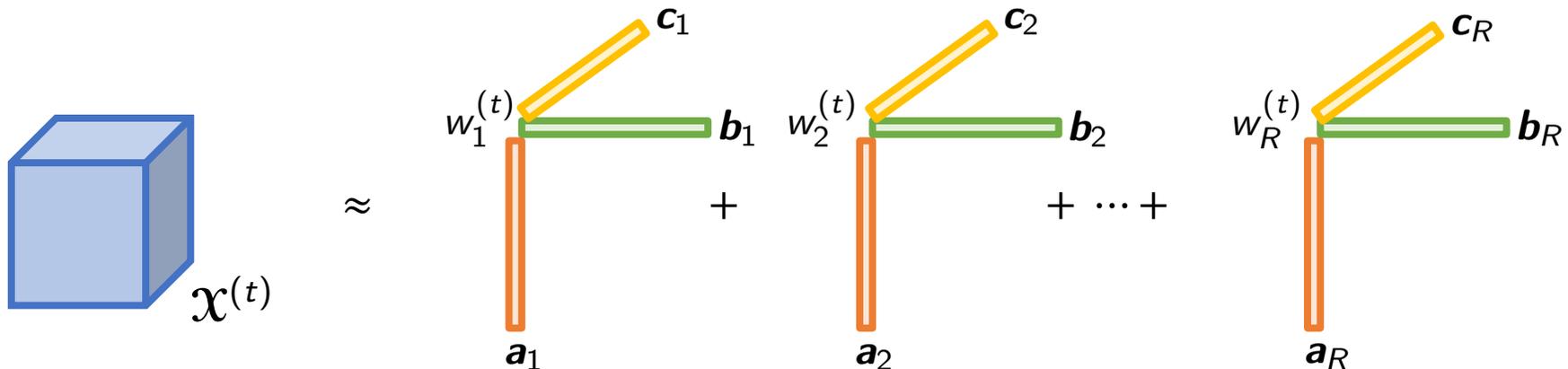
Generic d -way setup is similar.

Streaming Tensors – Two Points-of-View



At each time step t , new
3-way hyperslice added

If we assume factor matrices \mathbf{A} , \mathbf{B} , and \mathbf{C} fixed
through time, then the ideal factorization looks like...



$$\mathbf{x}^{(t)} \approx \sum_{j=1}^R w_j^{(t)} \mathbf{a}_j \circ \mathbf{b}_j \circ \mathbf{c}_j = \llbracket \mathbf{w}^{(t)}; \mathbf{A}, \mathbf{B}, \mathbf{C} \rrbracket$$

$$\mathbf{A} = [\mathbf{a}_1 \ \mathbf{a}_2 \ \dots \ \mathbf{a}_R] \in \mathbb{R}^{N_1 \times R}$$

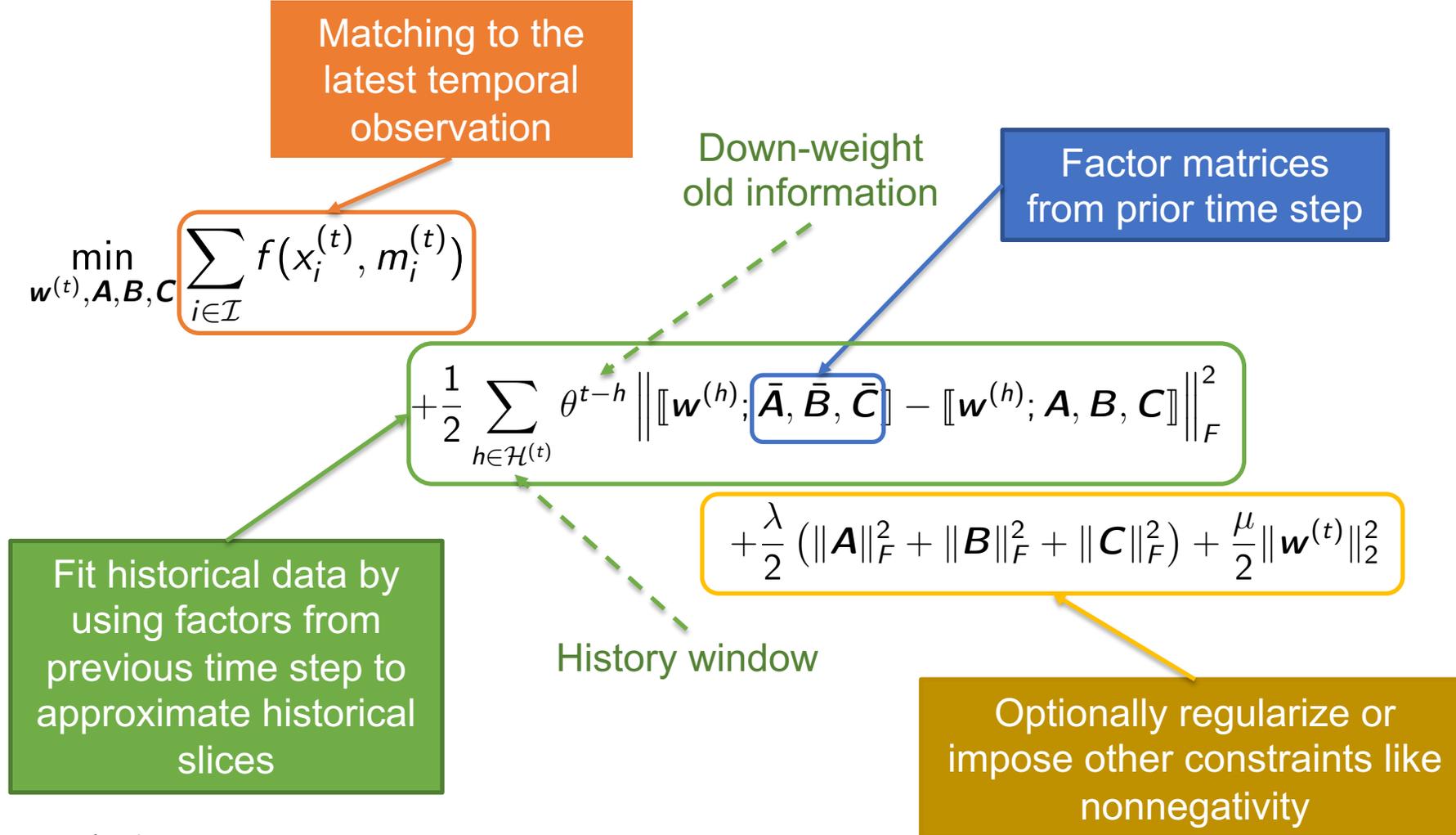
$$\mathbf{B} = [\mathbf{b}_1 \ \mathbf{b}_2 \ \dots \ \mathbf{b}_R] \in \mathbb{R}^{N_2 \times R}$$

$$\mathbf{C} = [\mathbf{c}_1 \ \mathbf{c}_2 \ \dots \ \mathbf{c}_R] \in \mathbb{R}^{N_3 \times R}$$

$$\mathbf{x}^{(t)} \in \mathbb{R}^{N_1 \times N_2 \times N_3}$$

Generic d-way setup is similar.

Streaming GCP “OnlineGCP” Formulation



Two step solution process:

- Solve for temporal weights $\mathbf{w}^{(t)}$ with factor matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ held fixed using GCP-SGD
- Solve for updated factor matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ using GCP-SGD

GenTen : Software for Generalized Canonical Polyadic Tensor Decompositions



GenTen : Software for Generalized Canonical Polyadic Tensor Decompositions¹

- HPC-oriented toolbox for CP decompositions
- CP-ALS, CP-OPT, GCP-OPT, GCP-SGD, OnlineGCP
- Supports both sparse and dense tensors

Targeted at extreme-scale architectures

- Distributed memory parallelism via MPI
- Shared memory parallelism via Kokkos
 - Multicore CPUs (OpenMP, pThreads)
 - NVIDIA (CUDA), AMD (HIP) and Intel (SYCL) GPUs

Callable from Matlab and Python

- Integrates with Matlab² and Python³ Tensor Toolboxes
- Manipulate/analyze tensor data but call GenTen HPC decomposition algorithms



¹<https://gitlab.com/tensors/genten>

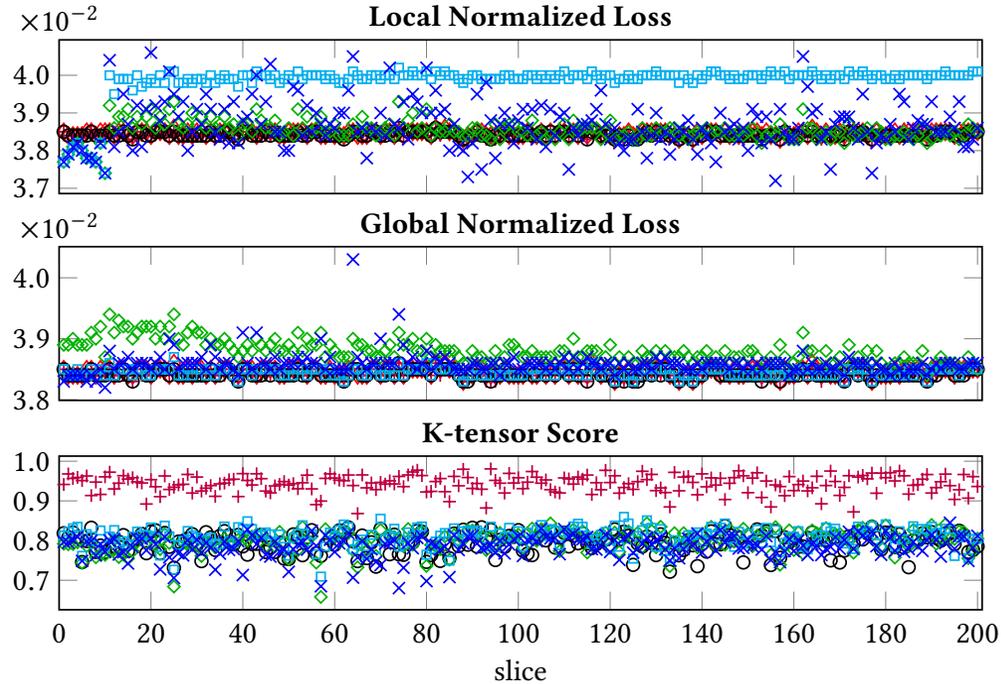
²<https://www.tensortoolbox.org/>

³<https://github.com/sandia-labs/pyttb>

Synthetic Data Experiments

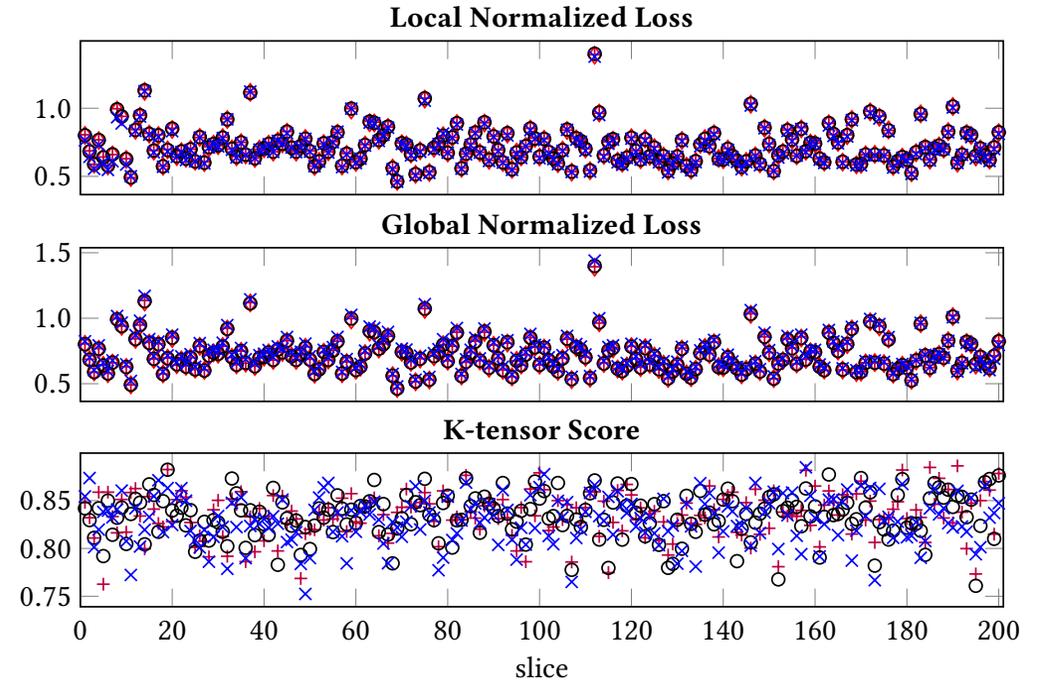


Gaussian 300x300x200, dense, R=20



× OnlineGCP □ OnlineCP ◇ Online SGD ○ GCP (static) + CP-ALS (static) ◇ True

Poisson 300x300x200, 3.2% sparsity, R=20



× OnlineGCP ○ GCP (static) + CP-APR (static) ◇ True

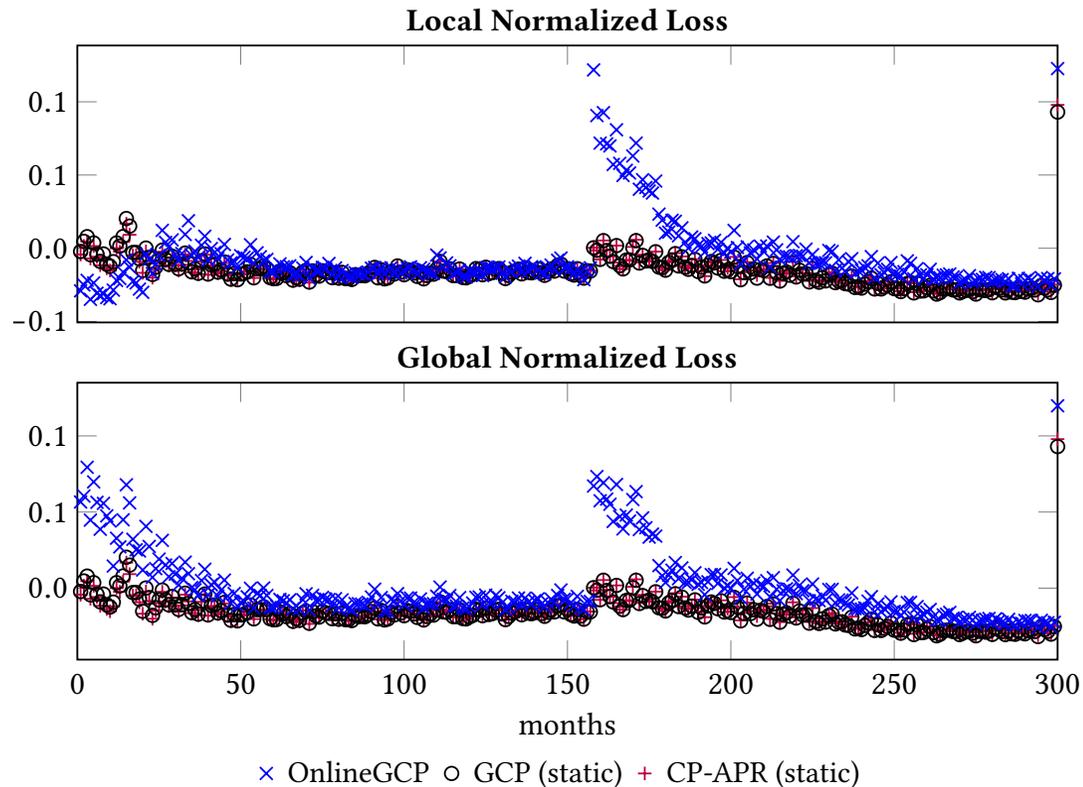
$$\text{Local Normalized Loss: } F_{local}(\mathcal{X}^{(t)}, \mathcal{M}^{(t)}) = \frac{1}{\|\mathcal{X}^{(t)}\|_F^2} \sum_{i \in \mathcal{I}} f(x_i^{(t)}, m_i^{(t)}), ; \mathcal{M}^{(t)} = [\mathbf{w}^{(t)}; \mathbf{A}_1^{(t)}, \dots, \mathbf{A}_d^{(t)}]$$

$$\text{Global Normalized Loss: } F_{global}(\mathcal{X}^{(t)}, \tilde{\mathcal{M}}^{(t)}) = \frac{1}{\|\mathcal{X}^{(t)}\|_F^2} \sum_{i \in \mathcal{I}} f(x_i^{(t)}, \tilde{m}_i^{(t)}), ; \tilde{\mathcal{M}}^{(t)} = [\mathbf{w}^{(t)}; \mathbf{A}_1^{(T)}, \dots, \mathbf{A}_d^{(T)}]$$

Real Data Experiments



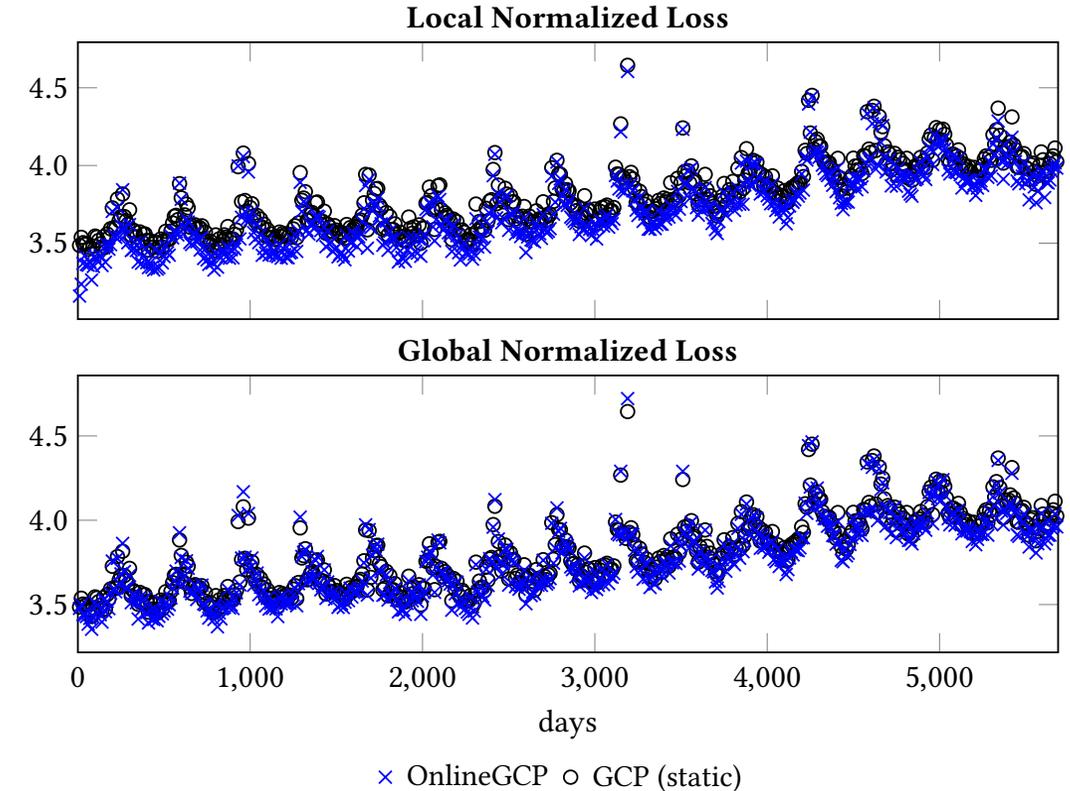
ArXiv Abstracts (Poisson)



Three-way tensor of counts of words in abstracts published on arxiv.org

- Category (172)
- Word (24558)
- Month since 10 years after first abstract (300)
- 2.4% sparsity

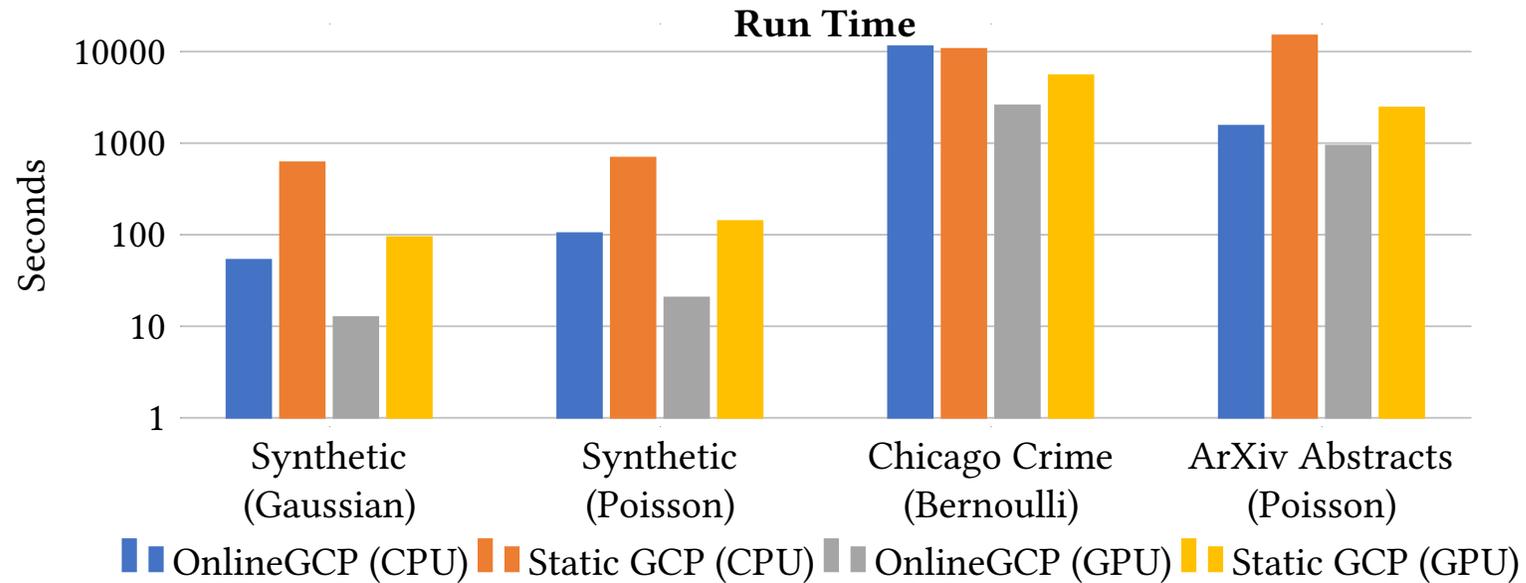
Chicago Crime (Bernoulli)



Four-way tensor of counts of crimes in the city of Chicago

- Hour of the day (24)
- Crime type (77)
- Neighborhood (32)
- Day since start date (5687)
- 1.6% sparsity
- Most counts are just 1, so use Bernoulli instead of Poisson

Computational Performance



CPU:

- Dual-socket Intel Xeon Platinum 8260, 24 cores/socket
- OpenMP Kokkos backend

GPU:

- NVIDIA Volta V100
- CUDA Kokkos backend

Same total number of samples between Static GCP and OnlineGCP

Summary and Conclusions



Summary

- Streaming tensor decomposition method for general statistical data types (continuous, count, binary, ...)
- Provides accuracy comparable to static/batch methods
- HPC software implementation in GenTen (with Matlab usability front-end)

Challenges

- Online (and static) GCP require tuning of many hyper-parameters (number of samples, learning rates, ...) particularly when balancing cost versus accuracy
- SGD solver often converges slowly increasing computational cost

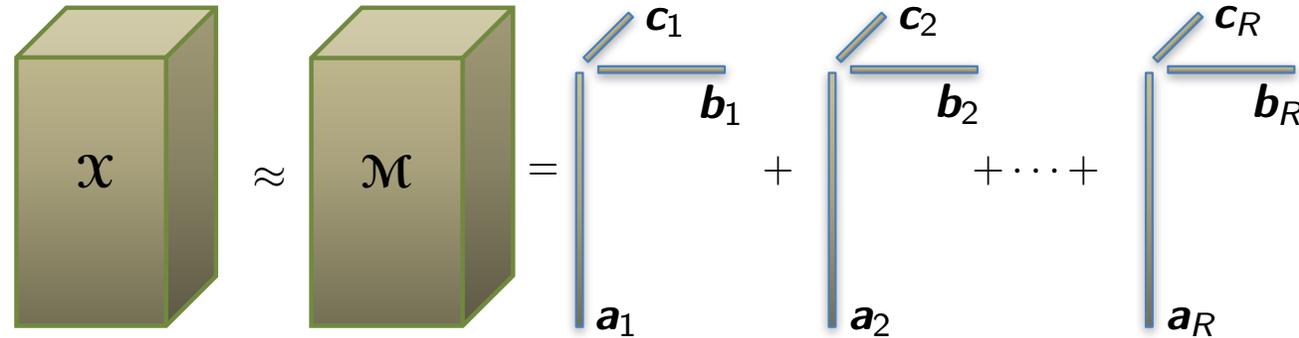
Moving forward

- Address online and static GCP solver challenges
- Incorporating distributed (synchronous and asynchronous parallelism)
- Python front-end for OnlineGCP



Backup Slides

Standard CP and ALS Solution Method



$$\min_{\mathcal{M}} \|\mathcal{X} - \mathcal{M}\|_F^2 \quad \text{s.t.} \quad \mathcal{M} = \mathbf{a}_1 \circ \mathbf{b}_1 \circ \mathbf{c}_1 + \dots + \mathbf{a}_R \circ \mathbf{b}_R \circ \mathbf{c}_R = [\mathbf{A}, \mathbf{B}, \mathbf{C}]$$

$$\mathbf{A} = [\mathbf{a}_1 \ \dots \ \mathbf{a}_R]$$

$$\mathbf{B} = [\mathbf{b}_1 \ \dots \ \mathbf{b}_R]$$

$$\mathbf{C} = [\mathbf{c}_1 \ \dots \ \mathbf{c}_R]$$

Using...

$$\mathcal{M}_{(1)} = \mathbf{A}(\mathbf{C} \odot \mathbf{B})^T$$

$$\mathcal{M}_{(2)} = \mathbf{B}(\mathbf{C} \odot \mathbf{A})^T$$

$$\mathcal{M}_{(3)} = \mathbf{C}(\mathbf{B} \odot \mathbf{A})^T$$

Repeat until convergence...

$$\min_{\mathbf{A}} \|\mathcal{X}_{(1)} - \mathbf{A}(\mathbf{C} \odot \mathbf{B})^T\|_F^2$$

$$\min_{\mathbf{B}} \|\mathcal{X}_{(2)} - \mathbf{B}(\mathbf{C} \odot \mathbf{A})^T\|_F^2$$

$$\min_{\mathbf{C}} \|\mathcal{X}_{(3)} - \mathbf{C}(\mathbf{B} \odot \mathbf{A})^T\|_F^2$$

Repeat until convergence...

$$\mathbf{A} = \mathcal{X}_{(1)}(\mathbf{C} \odot \mathbf{B})(\mathbf{C}^T \mathbf{C} * \mathbf{B}^T \mathbf{B})^\dagger$$

$$\mathbf{B} = \mathcal{X}_{(2)}(\mathbf{C} \odot \mathbf{A})(\mathbf{C}^T \mathbf{C} * \mathbf{A}^T \mathbf{A})^\dagger$$

$$\mathbf{C} = \underbrace{\mathcal{X}_{(3)}(\mathbf{B} \odot \mathbf{A})}_{\text{MTTKRP}}(\mathbf{B}^T \mathbf{B} * \mathbf{A}^T \mathbf{A})^\dagger$$

MTTKRP

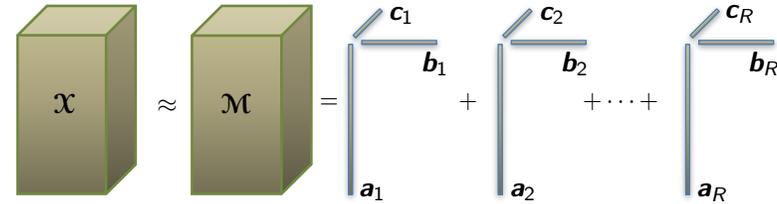
Review of Generalized CP (GCP)*



Standard CP optimization problem:

$$\min_{\mathcal{M}} F(\mathcal{X}, \mathcal{M}) = \|\mathcal{X} - \mathcal{M}\|_F^2 = \sum_i (x_i - m_i)^2$$

$$\text{s.t. } \mathcal{M} = \llbracket \mathbf{A}, \mathbf{B}, \mathbf{C} \rrbracket = \mathbf{a}_1 \circ \mathbf{b}_1 \circ \mathbf{c}_1 + \cdots + \mathbf{a}_R \circ \mathbf{b}_R \circ \mathbf{c}_R$$



Objective function derived through maximum likelihood estimation assuming the data tensor (\mathcal{X}) entries are i.i.d. Gaussian:

$$x_i \sim N(x_i | m_i, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x_i - m_i)^2 / (2\sigma^2)} \implies$$

$$L(\mathcal{M} | \mathcal{X}) = \prod_i N(x_i | m_i, \sigma) \implies$$

$$-\log L(\mathcal{M} | \mathcal{X}) = \sum_i \left(\frac{(x_i - m_i)^2}{2\sigma^2} + \frac{1}{2} \log(2\pi\sigma^2) \right) \sim \sum_i (x_i - m_i)^2$$

*Hong, Kolda, Duersch. Generalized Canonical Polyadic Tensor Decomposition. SIAM Review, 2019.

Review of Generalized CP (GCP)*



Generalize to other types of data through arbitrary loss function:

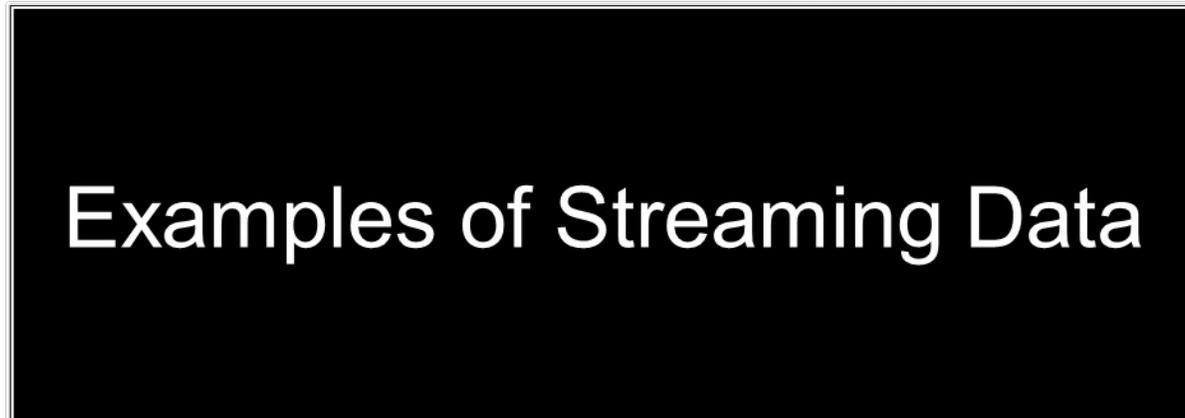
$$\begin{array}{c}
 x_i \sim p(x_i|\theta_i) \\
 f(x_i, m_i) \equiv -\log p(x_i|\ell^{-1}(m_i)) \implies \min_{\mathcal{M}} F(\mathcal{X}, \mathcal{M}) = \sum_i f(x_i, m_i) \\
 \text{s.t. } \mathcal{M} = \llbracket \mathbf{A}, \mathbf{B}, \mathbf{C} \rrbracket
 \end{array}$$

loss function

link function

Distribution	Link function	Loss function	Constraints
$\mathcal{N}(\mu, \sigma)$	$m = \mu$	$(x-m)^2$	$x, m \in \mathbb{R}$
Gamma(k, σ)	$m = k\sigma$	$x/(m+\epsilon) + \log(m+\epsilon)$	$x > 0, m \geq 0$
Poisson(λ)	$m = \lambda$	$m - x \log(m+\epsilon)$	$x \in \mathbb{N}, m \geq 0$
	$m = \log \lambda$	$e^m - xm$	$x \in \mathbb{N}, m \in \mathbb{R}$
Bernoulli(ρ)	$m = \rho / (1-\rho)$	$\log(m+1) - x \log(m+\epsilon)$	$x \in \{0, 1\}, m \geq 0$
	$m = \log(\rho / (1-\rho))$	$\log(1+e^m) - xm$	$x \in \{0, 1\}, m \in \mathbb{R}$

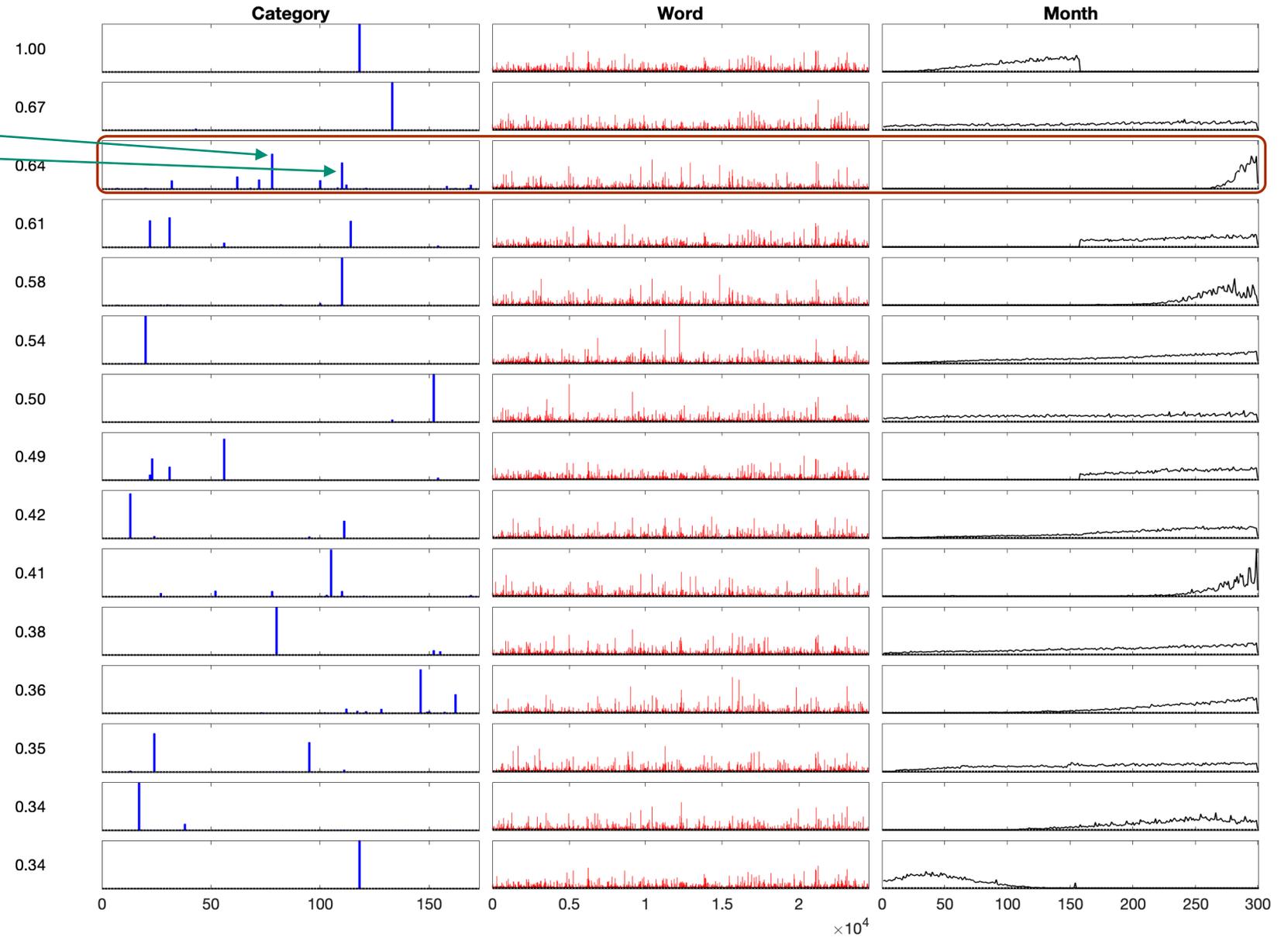
*Hong, Kolda, Duersch. Generalized Canonical Polyadic Tensor Decomposition. SIAM Review, 2019.



Component #3 (of 50): Computer Vision/Machine Learning



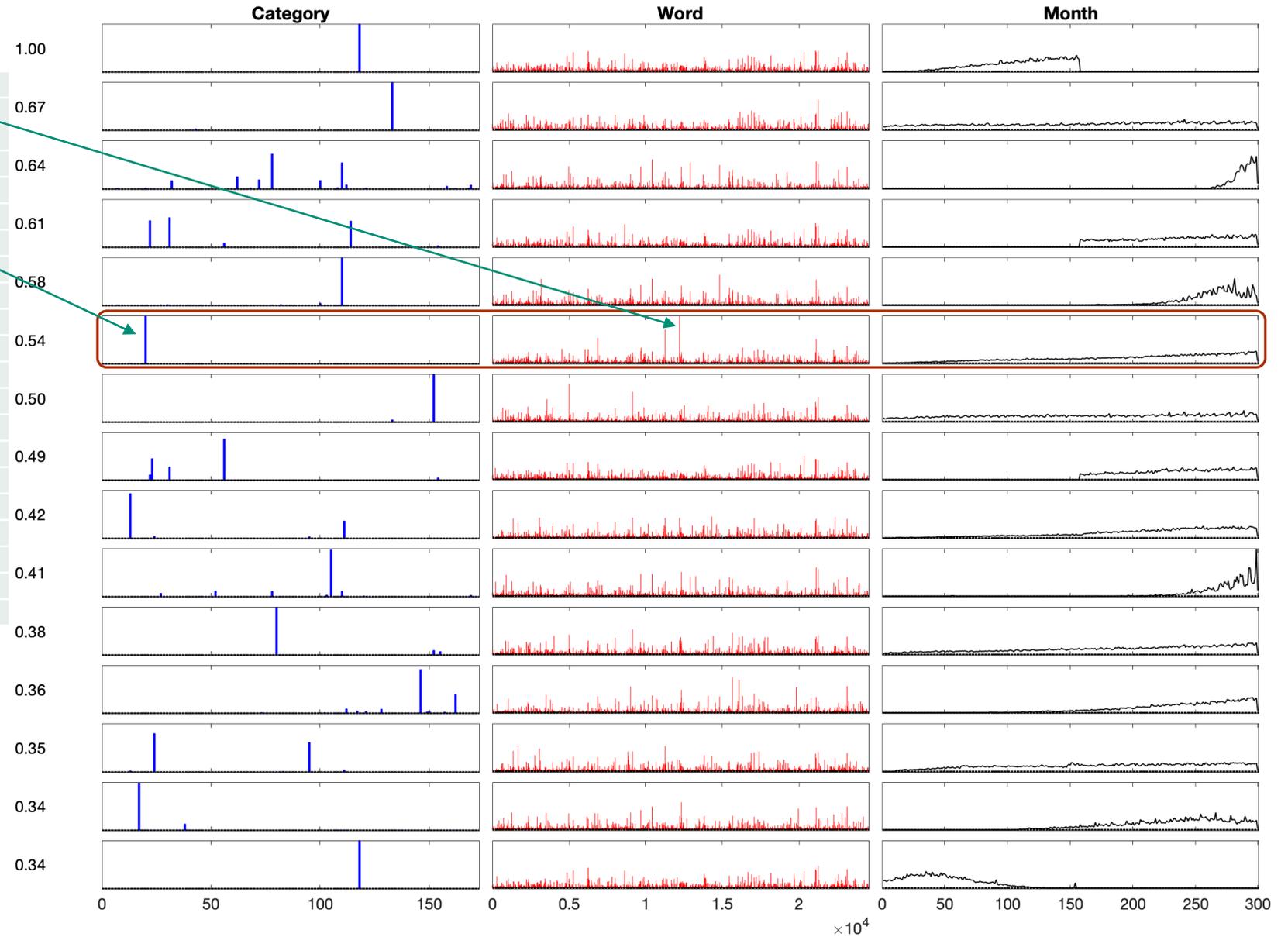
Top Categories	Category Weight	Top Words	Word Weight
cs.LG	1.00	propose	1.00
cs.CV	0.75	use	0.93
eess.IV	0.35	learn	0.89
eess.SP	0.27	model	0.85
cs.RO	0.25	method	0.81
eess.SY	0.24	network	0.77
math.OC	0.12	base	0.75
eess.AS	0.12	paper	0.74
		result	0.73
		train	0.65
		datum	0.65
		performance	0.60
		approach	0.59
		state	0.56
		dataset	0.56
		neural	0.52
		deep	0.51
		problem	0.49
		achieve	0.49
		demonstrate	0.48



Component #6 (of 50): Quantum Physics



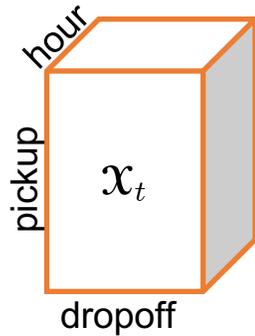
Top Categories	Category Weight	Top Words	Word Weight
quant-ph	1.00	quantum	1.00
		state	0.71
		system	0.53
		use	0.51
		result	0.36
		classical	0.33
		time	0.32
		measurement	0.30
		entanglement	0.30
		qubit	0.30
		present	0.29
		information	0.29
		non	0.27
		study	0.26
		single	0.26
		base	0.25
		photon	0.25
		propose	0.24
		provide	0.23
		case	0.23



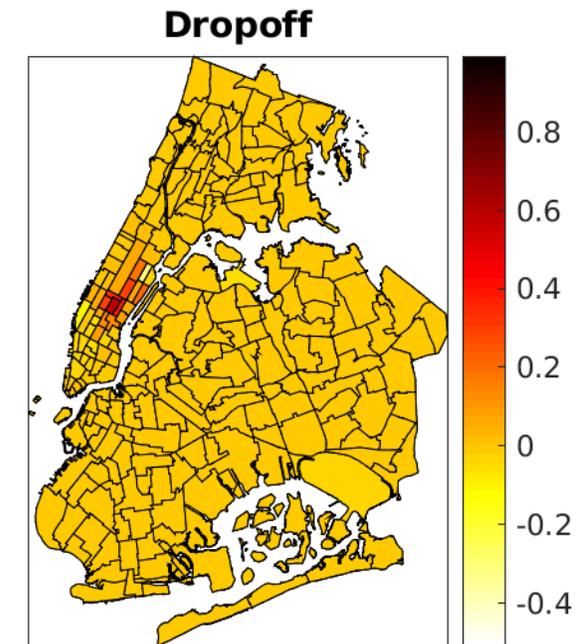
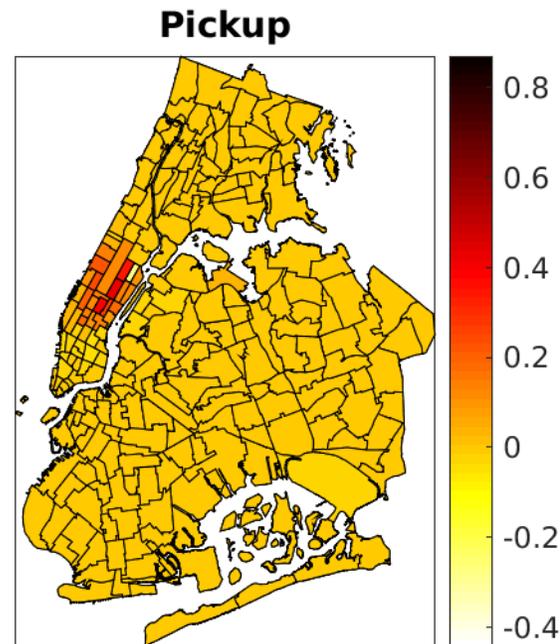
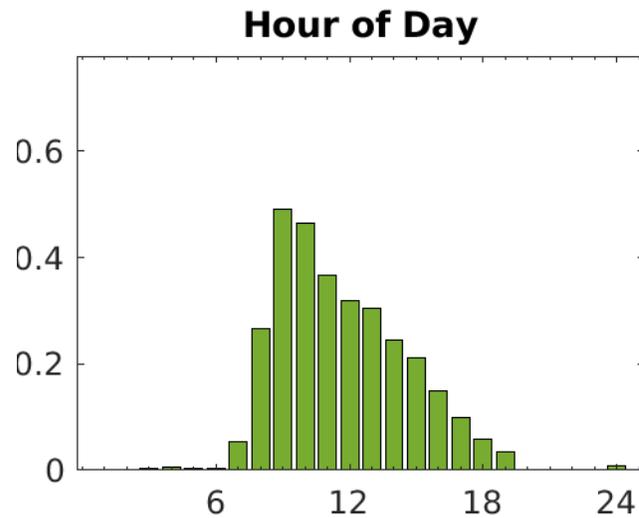
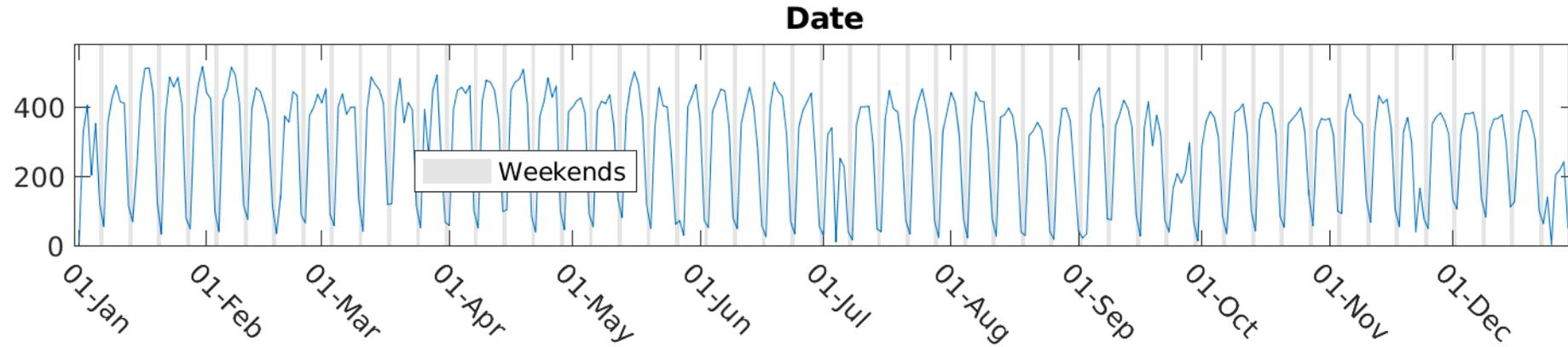
NYC Taxi Dataset – 4-way Tensor



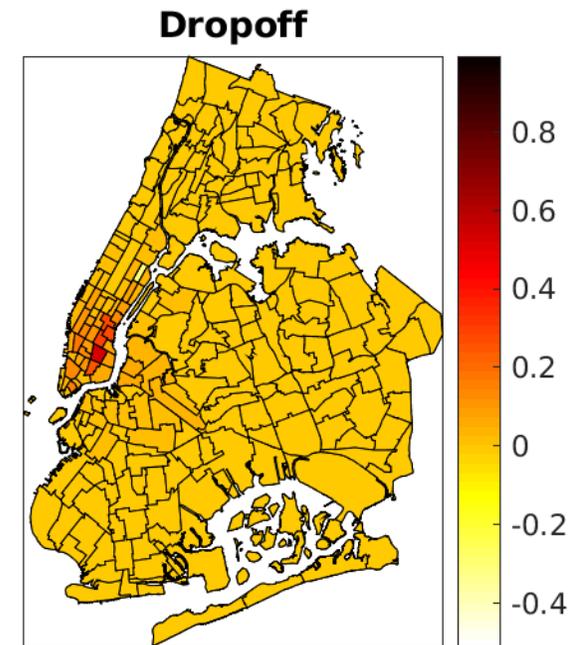
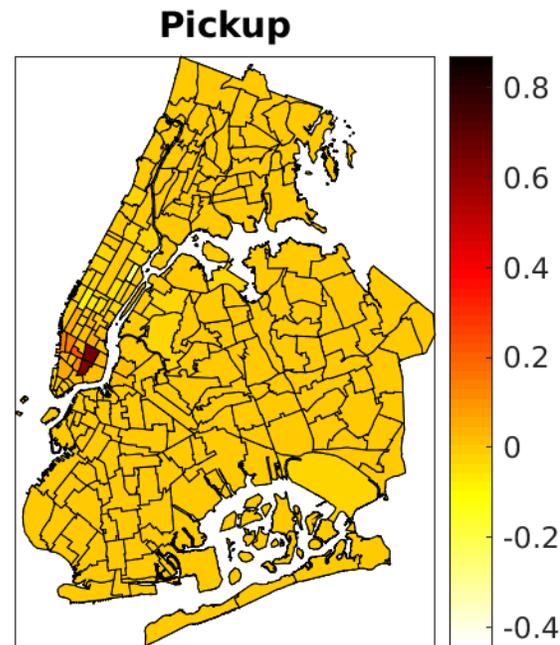
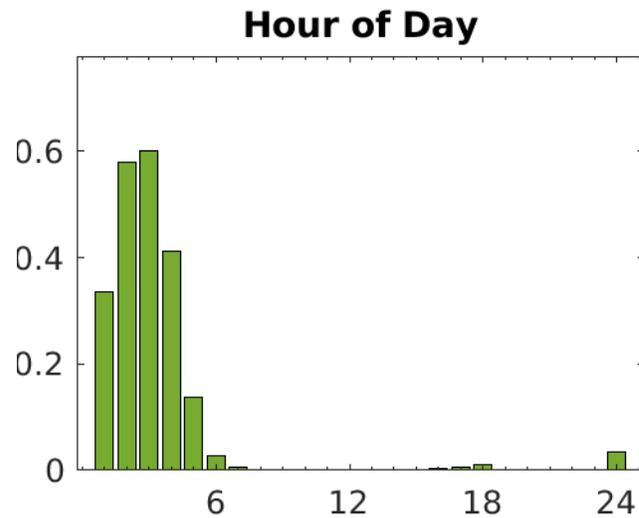
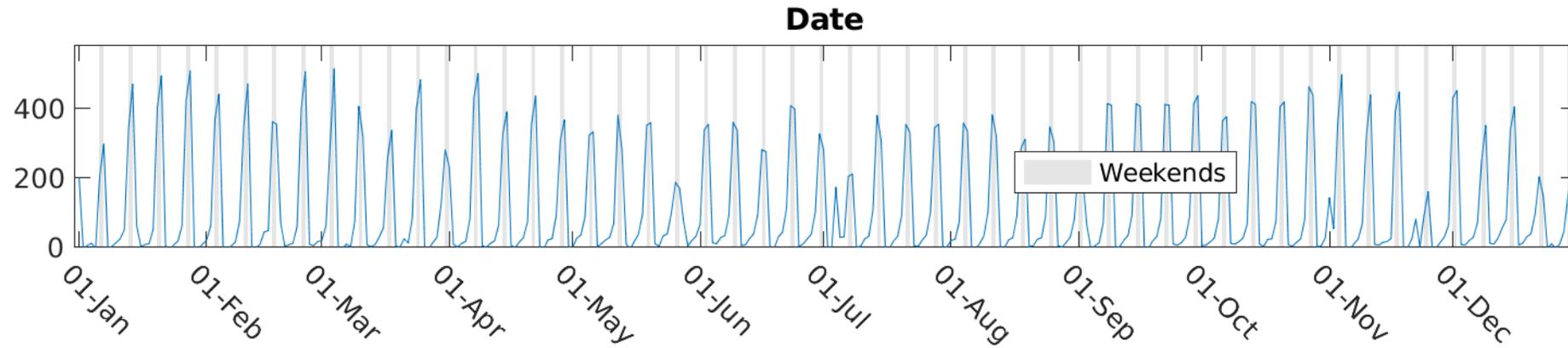
- Data from NYC public records
<https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page>
- 10+ Years of Data
- 4-way Tensor, **Updated Daily**
 - Pickup Zone
 - Dropoff Zone
 - Pickup Hour
 - *New 3-way sparse tensor each day*
- 265 Taxi Zones
<https://catalog.data.gov/dataset/nyc-taxi-zones>



Component #1 (of 50) – Standard Mid-morning Weekday Traffic



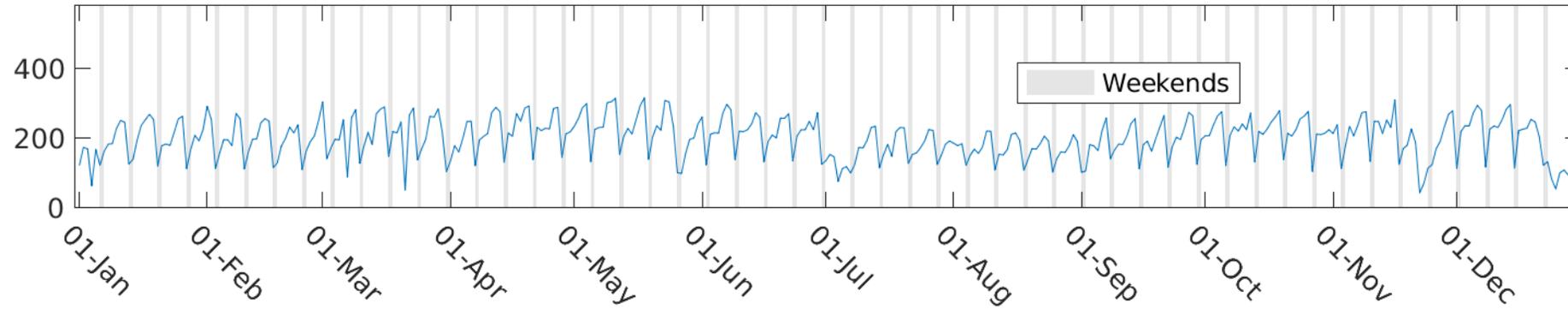
Component #21 (of 50): Weekend Nightlife



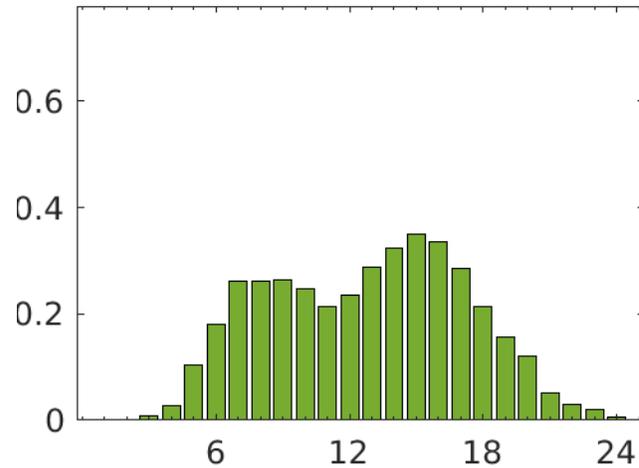
Component #17 (of 50): Travel to JFK and La Guardia Airports



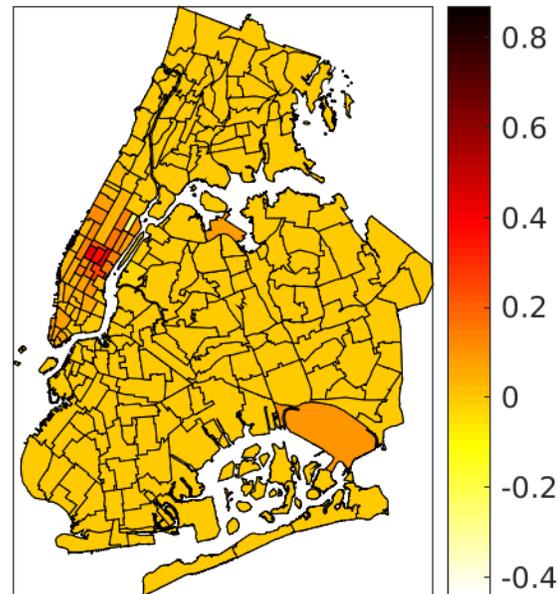
Date



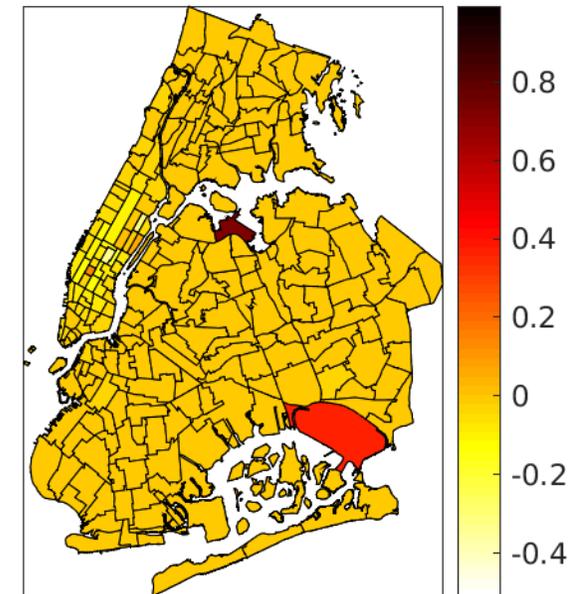
Hour of Day



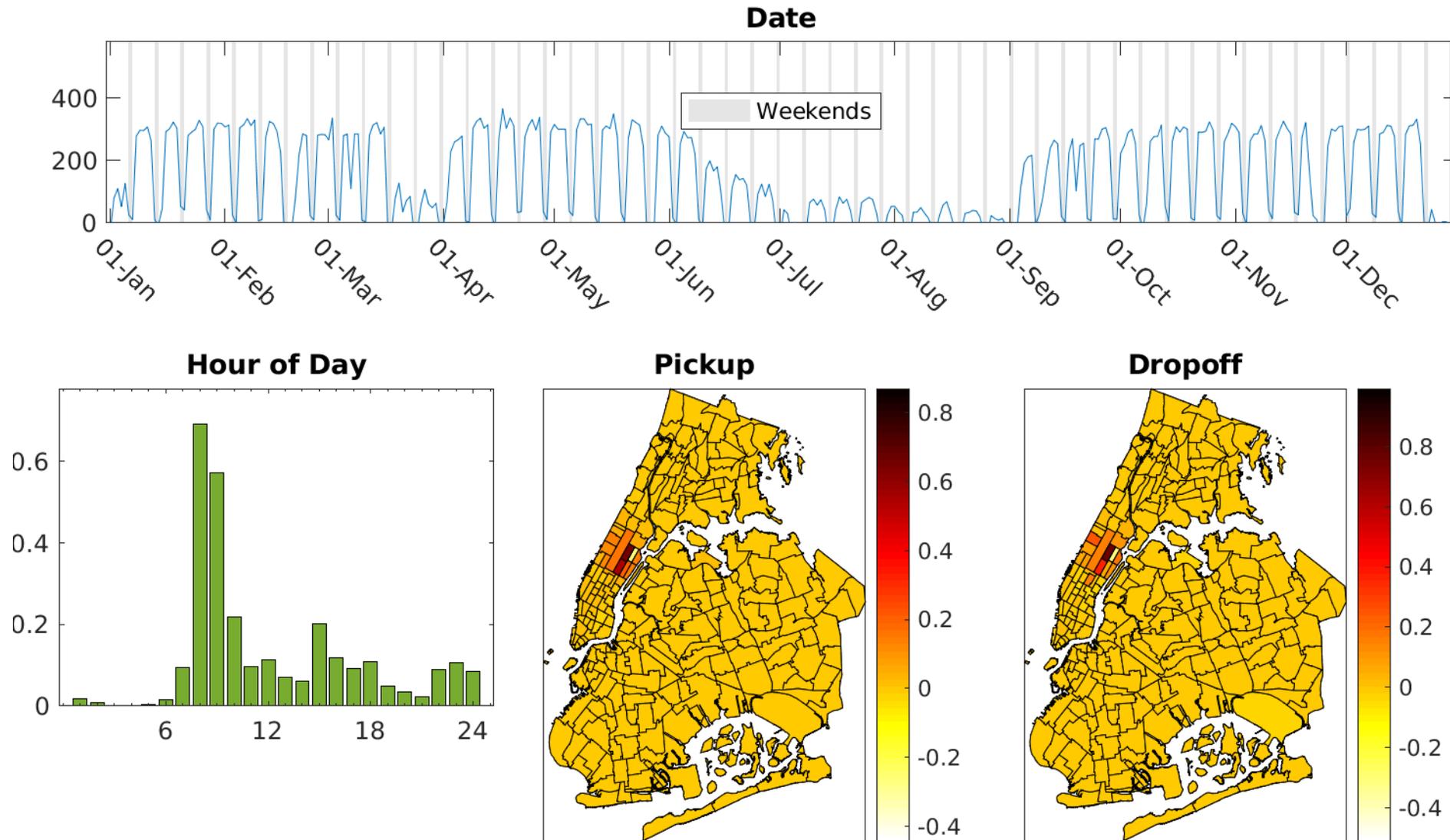
Pickup



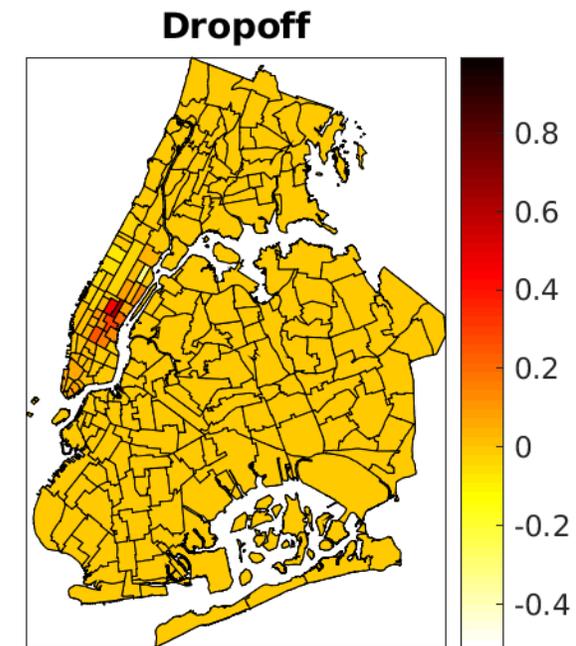
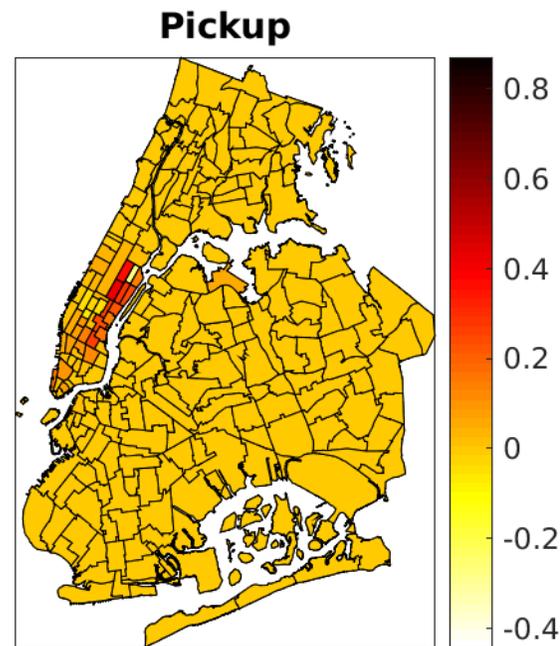
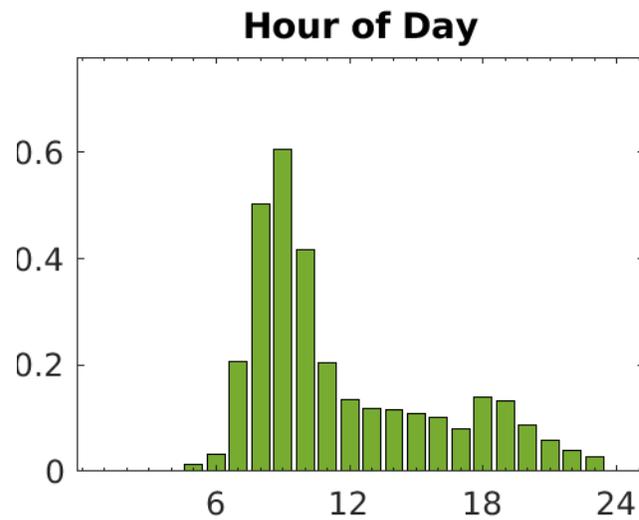
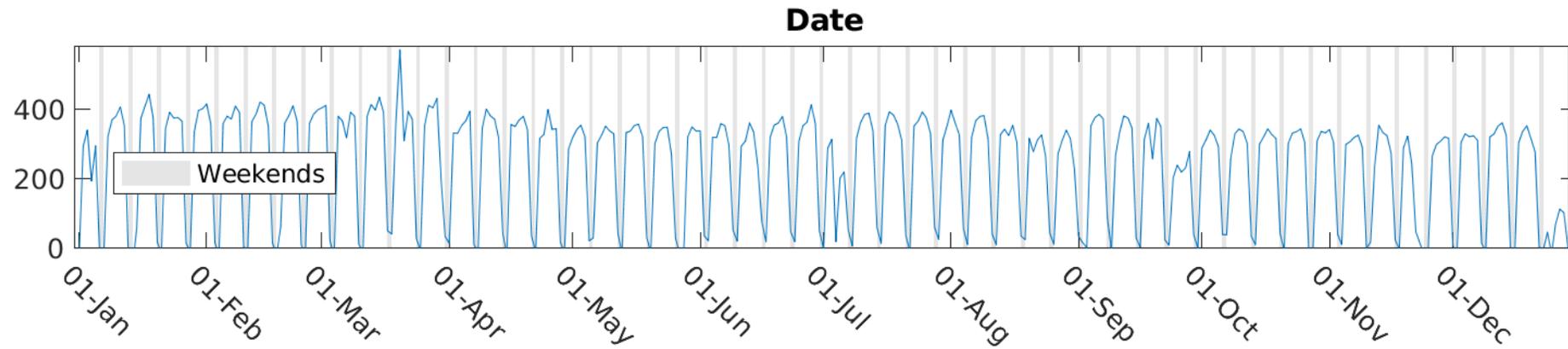
Dropoff



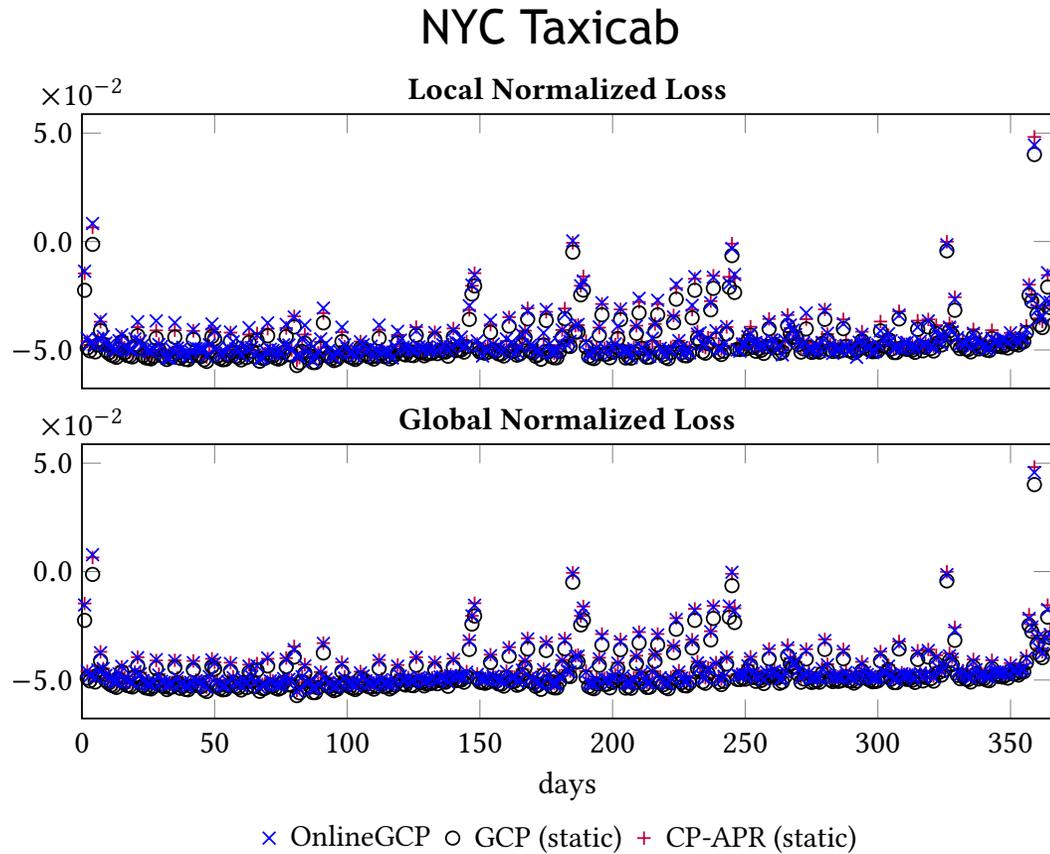
Component #20 (of 50): School Morning Dropoff



Component #4 (of 50): Morning Commute to Rockefeller Center

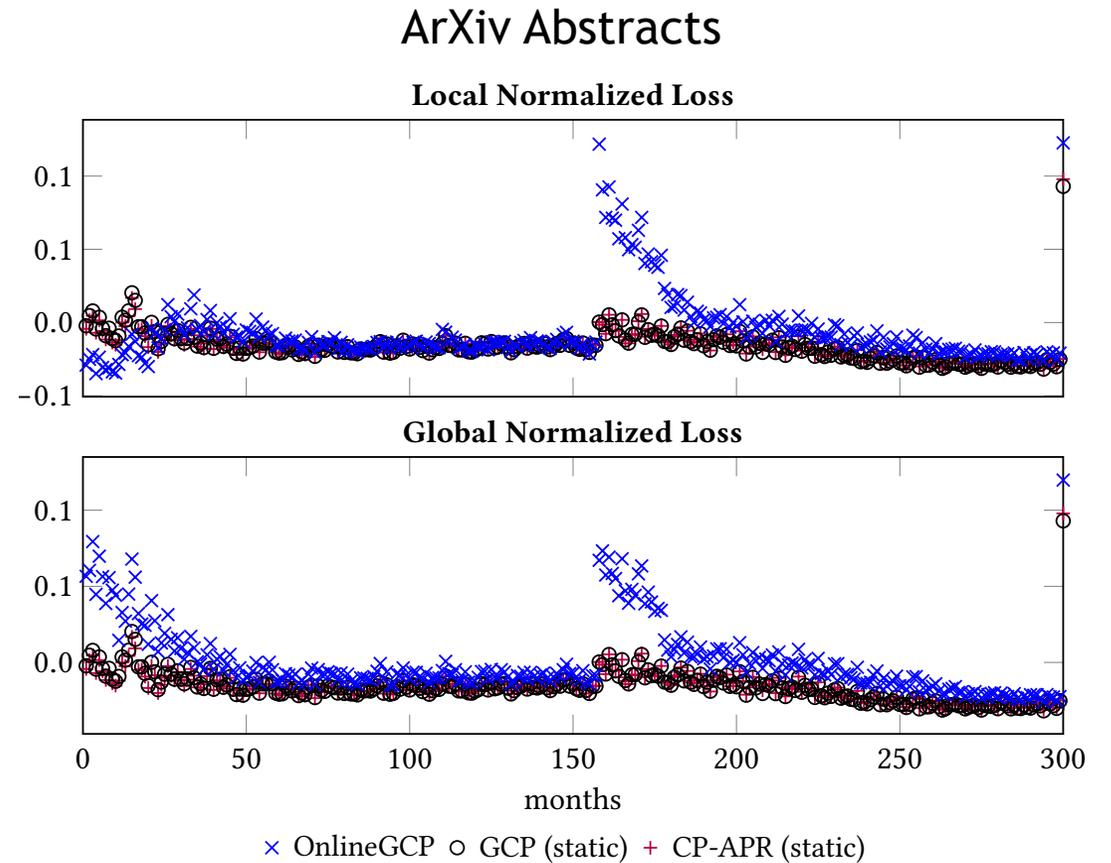


Real Data Experiments (Poisson)



Four-way tensor of counts of taxi rides in NYC in 2018

- Pickup zone (263)
- Dropoff zone (263)
- Pickup hour (24)
- Day (265)
- 3.8% sparsity



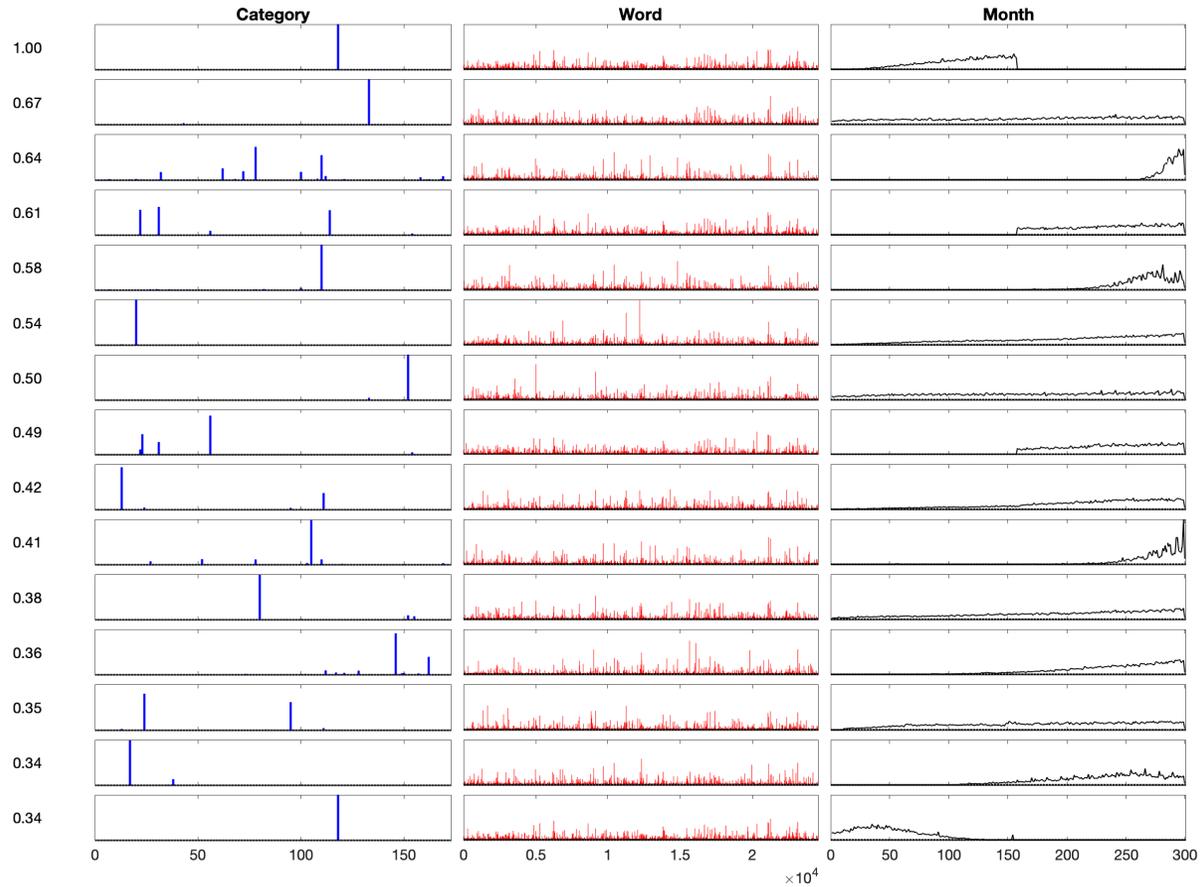
Three-way tensor of counts of words in abstracts published on arxiv.org

- Category (172)
- Word (24558)
- Month since 10 years after first abstract (300)
- 2.4% sparsity

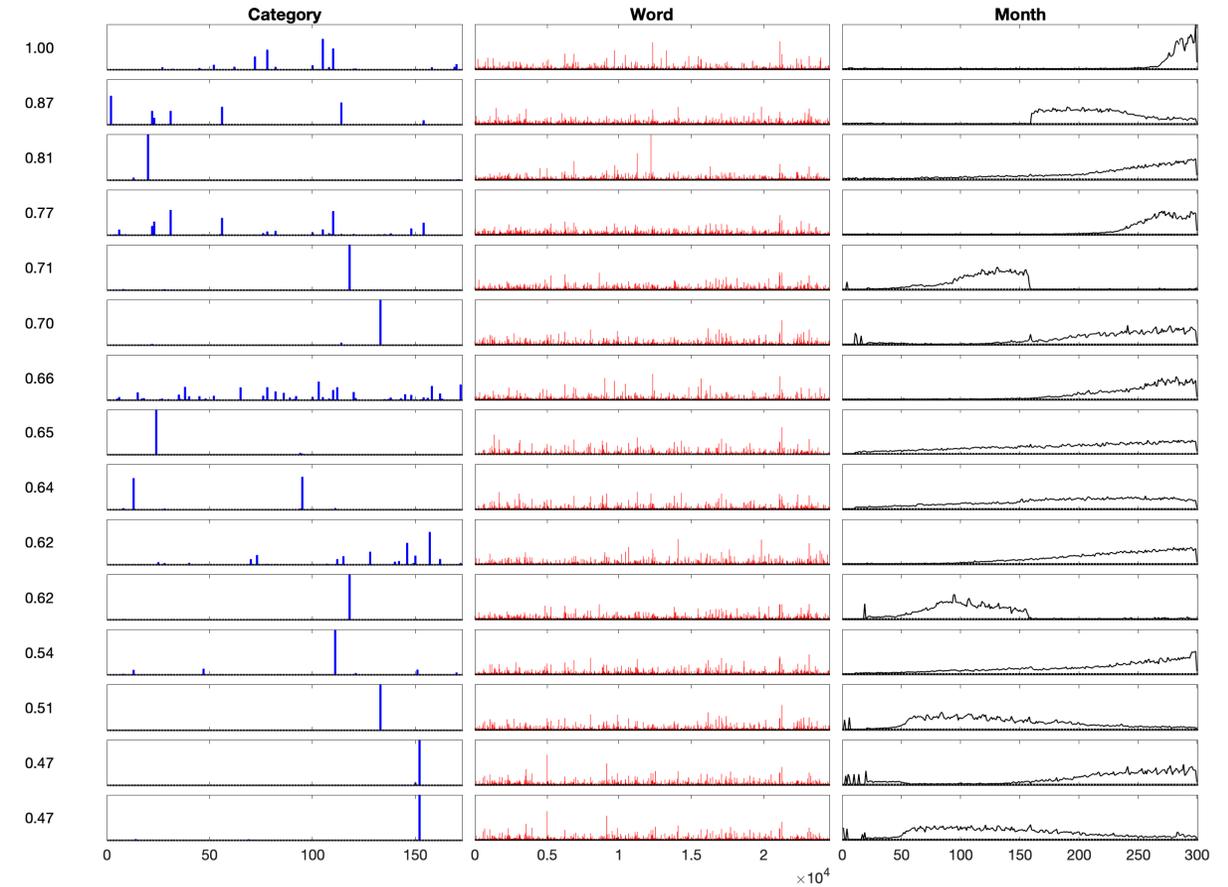
Comparing ArXiv Factors



Static GCP



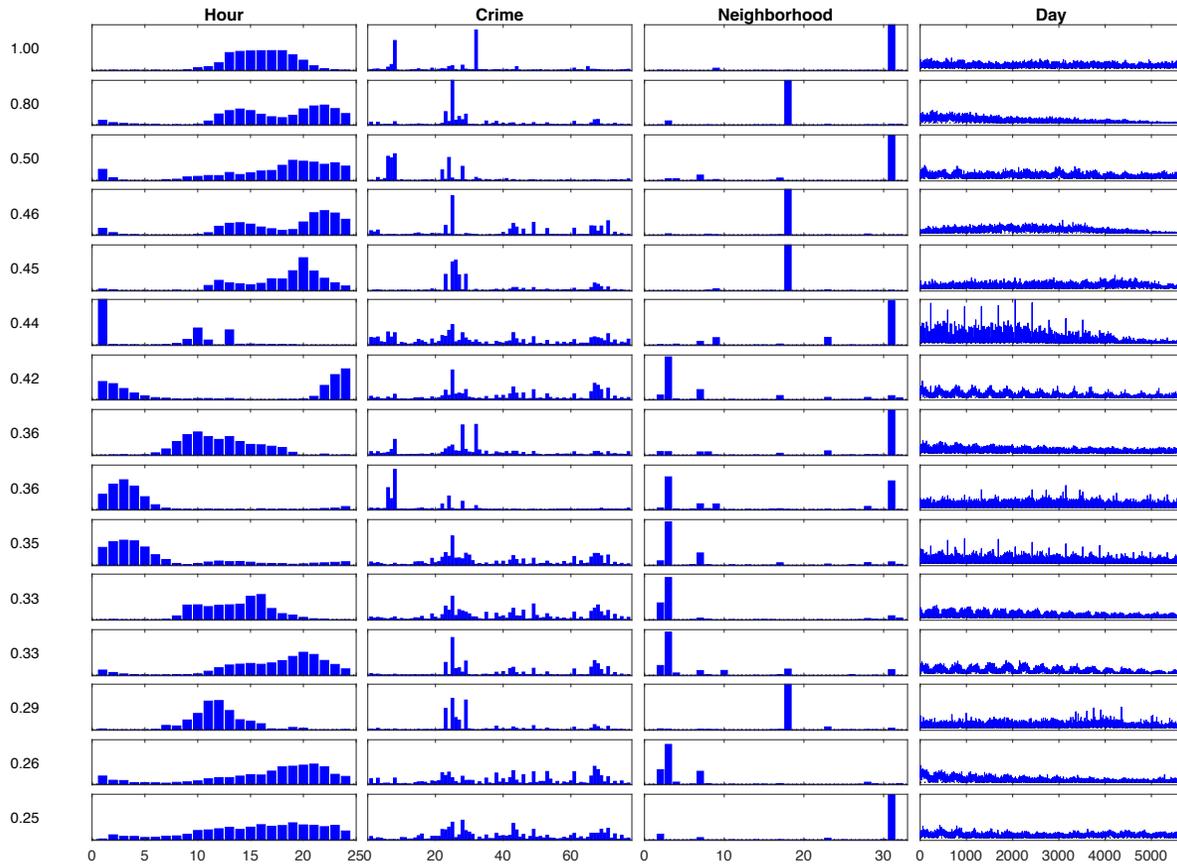
Online GCP



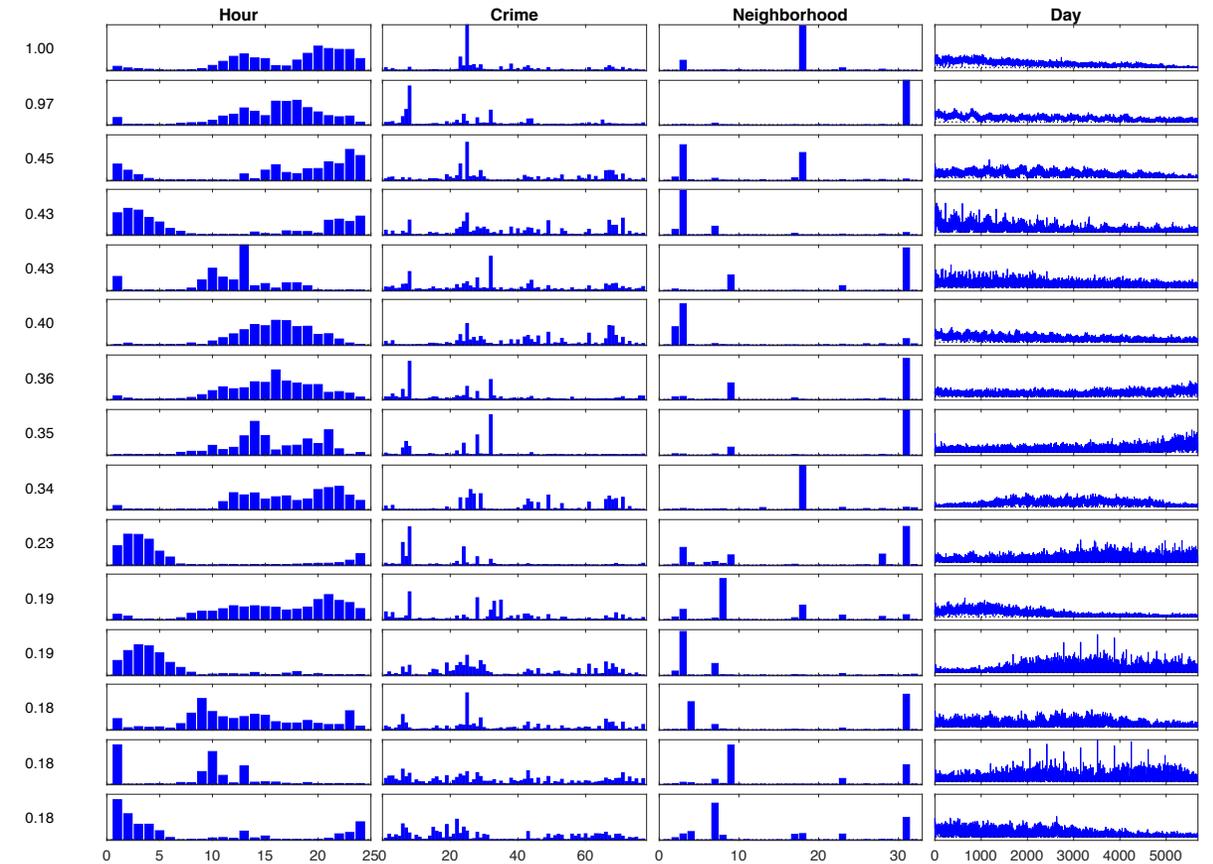
Comparing Chicago Crime Factors



Static GCP



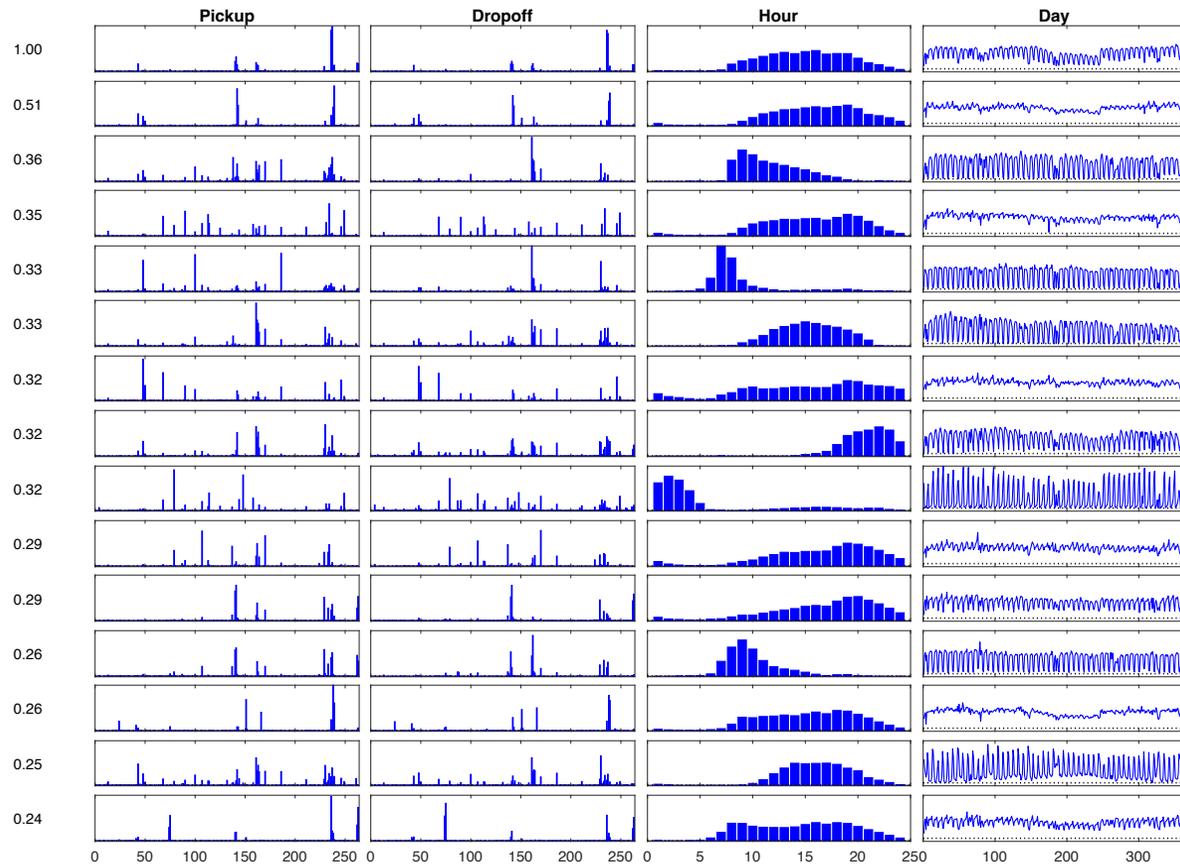
Online GCP



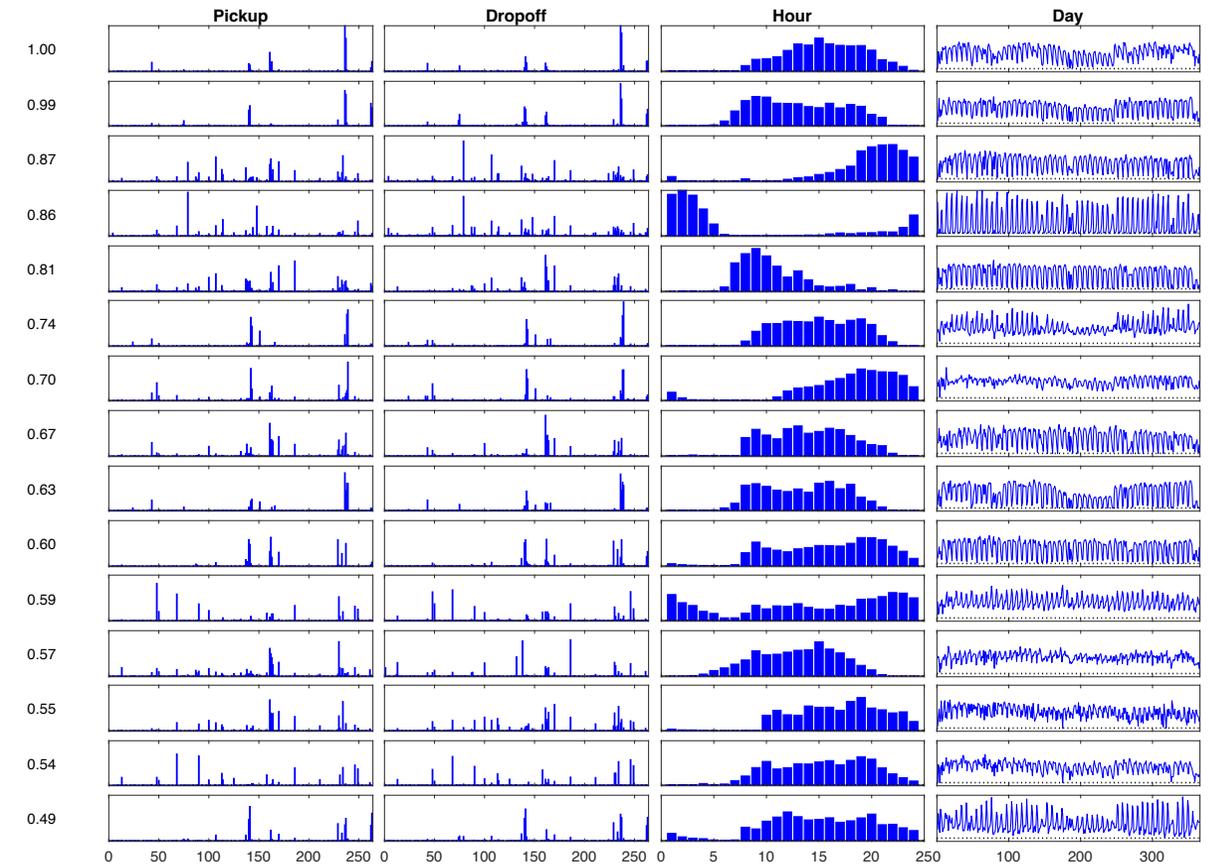
Comparing NYC Taxicab Factors



Static GCP



Online GCP





$$\min_{\mathcal{M}} F(\mathcal{X}, \mathcal{M}) = \sum_i f(x_i, m_i) \quad \text{s.t.} \quad \mathcal{M} = [\mathbf{A}_1, \dots, \mathbf{A}_d]$$

Lose the least-squares structure underlying traditional algorithms. Instead pursue gradient-based optimization approach.

Define tensor \mathbf{Y} such that

$$y(i_1, \dots, i_d) = y_i = \frac{\partial f}{\partial m}(x_i, m_i)$$

Then gradient of objective function given by

$$\mathbf{G}_k = \frac{\partial F}{\partial \mathbf{A}_k} = \mathbf{y}_{(k)}(\mathbf{A}_d \odot \dots \odot \mathbf{A}_{k+1} \odot \mathbf{A}_{k-1} \odot \dots \odot \mathbf{A}_1) \quad \leftarrow \text{MTTKRP!}$$

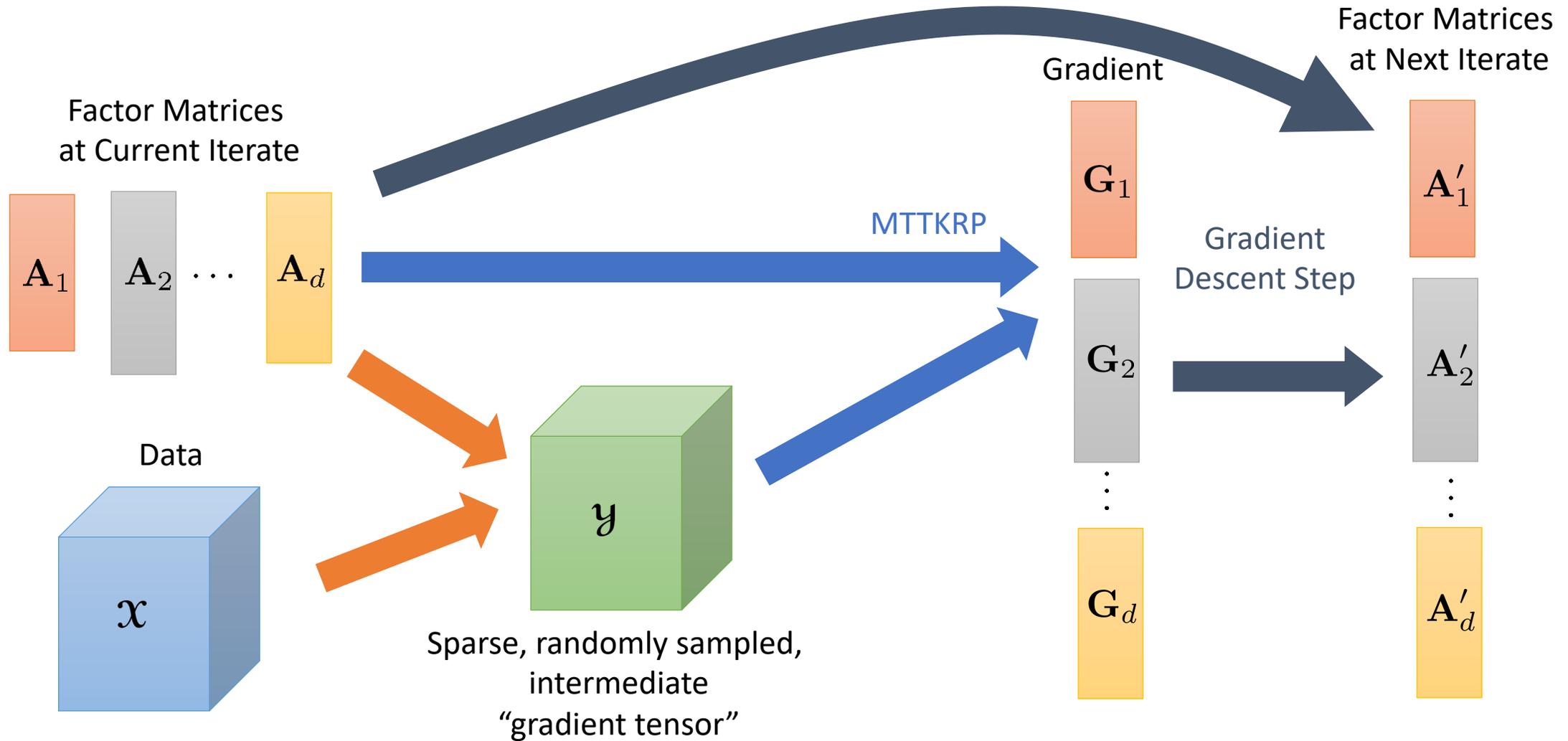
Unfortunately, \mathbf{Y} is in general dense, even when \mathbf{X} is sparse, making standard optimization infeasible.

Instead, employ Stochastic Gradient Descent (SGD) where \mathbf{Y} is only randomly sampled

- Stratified: sample zeros and nonzeros separately (requires tensor search)
- Semi-stratified: skip tensor search and adjust for “zeros” that are really nonzeros

¹Kolda and Hong [Stochastic Gradients for Large-Scale Tensor Decomposition](#), SIMODS, 2020.

High-level View of GCP Optimization & Dependencies



A Few (of Many) Related Methods



Definitions

- CP model at current time step: $\mathcal{M}^{(t)} = \llbracket \mathbf{w}^{(t)}; \mathbf{A}_1, \dots, \mathbf{A}_d \rrbracket$
- CP model with factor matrices from prior time step: $\bar{\mathcal{M}}^{(t)} = \llbracket \mathbf{w}^{(t)}; \bar{\mathbf{A}}_1, \dots, \bar{\mathbf{A}}_d \rrbracket$

Online SGD¹

- Formulation: $\min_{\mathbf{w}^{(t)}, \mathbf{A}_1, \dots, \mathbf{A}_d} \|\mathbf{u}^{(t)} * (\mathcal{X}^{(t)} - \mathcal{M}^{(t)})\|^2 + \sum_{h=1}^{t-1} \theta^{t-h} \|\mathbf{u}^{(h)} * (\mathcal{X}^{(h)} - \mathcal{M}^{(h)})\|^2 + \bar{\lambda}_t \sum_{j=1}^d \|\mathbf{A}_j\|_F^2 + \lambda \|\mathbf{w}^{(t)}\|_2^2$
- Two-step solution process at each time step:
 - Solve for temporal weights $\mathbf{w}^{(t)}$ with factor matrices \mathbf{A}_j held fixed (linear least squares)
 - Apply 1 step of gradient descent to update factor matrices \mathbf{A}_j (history is not included in these updates)

CP-Stream²

- Formulation: $\min_{\mathbf{w}^{(t)}, \mathbf{A}_1, \dots, \mathbf{A}_d} \|\mathcal{X}^{(t)} - \mathcal{M}^{(t)}\|^2 + \sum_{h=1}^{t-1} \theta^{t-h} \|\bar{\mathcal{M}}^{(h)} - \mathcal{M}^{(h)}\|^2 + \lambda \|\mathbf{w}^{(t)}\|_2^2$
- Two-step solution process at each time step:
 - Solve for temporal weights $\mathbf{w}^{(t)}$ with factor matrices \mathbf{A}_j held fixed (linear least squares)
 - Apply ADMM algorithm for updating factor matrices \mathbf{A}_j

OnlineCP³

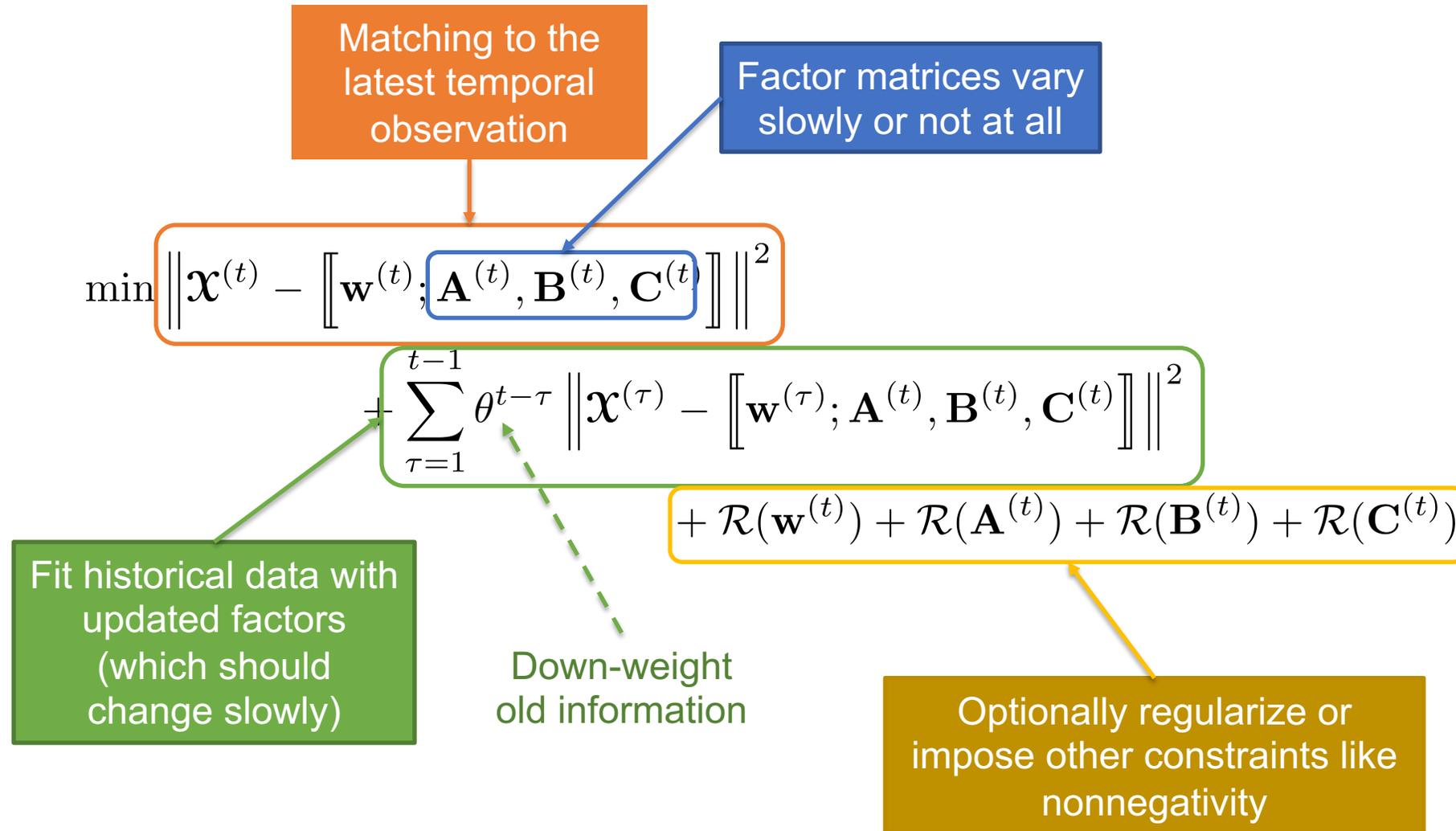
- Formulation: $\min_{\mathbf{w}^{(t)}, \mathbf{A}_1, \dots, \mathbf{A}_d} \sum_{h=1}^t \|\mathcal{X}^{(h)} - \mathcal{M}^{(h)}\|^2$
- Applies one step of ALS each time step for temporal weights $\mathbf{w}^{(t)}$ and factor matrices \mathbf{A}_j
- Leverages least-squares structure to incorporate prior tensor data into MTTKRP and Gram computations without explicitly storing prior slices

¹Mardani et al, [Subspace Learning and Imputation for Streaming Big Data Matrices and Tensors](#), 2015

²Smith et al, [Streaming Tensor Factorization for Infinite Data Sources](#), 2018

³Zhou et al, [Accelerating Online CP Decompositions for Higher Order Tensors](#), 2016

Generic Streaming Formulation (For Gaussian Data)



Streaming GCP (“OnlineGCP”)



$$\min_{\mathbf{w}^{(t)}, \mathbf{A}_1, \dots, \mathbf{A}_d} \sum_{i \in \mathcal{I}} f(x_i^{(t)}, m_i^{(t)}) + \frac{1}{2} \sum_{h \in \mathcal{H}^{(t)}} s^{(h)} \|\bar{\mathcal{M}}^{(h)} - \mathcal{M}^{(h)}\|^2 + \frac{\lambda}{2} \sum_{k=1}^d \|\mathbf{A}_k\|_F^2 + \frac{\mu}{2} \|\mathbf{w}^{(t)}\|_2^2$$

History regularization penalizing large changes in CP model

GCP Loss for current time step

History window

Factor matrix regularization to encourage low-rank

Two step solution process

- Solve for temporal weights $\mathbf{w}^{(t)}$ with factor matrices \mathbf{A}_j held fixed using GCP-SGD
- Solve for updated factor matrices \mathbf{A}_j using GCP-SGD

Leverage ADAM SGD solvers

- Start new ADAM solver each time step for temporal weights
- Keep momentum terms from ADAM across factor matrix solves
- Sampled gradient approximations each ADAM epoch (stratified sampling)

History window

- Many possibilities depending on the problem
- We used *reservoir sampling*, which provides a uniform sample of $[1, t]$, by randomly evicting previous entries with diminishing probability

What is Kokkos?

