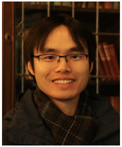
Dynamic Autoregressive Tensor Factorization for Pattern Discovery of Spatiotemporal Systems

Xinyu Chen

Postdoctoral Associate, MIT

June 20, 2025



Xinyu Chen	Dingyi Zhuang	HanQin Cai	Shenhao Wang	Jinhua Zhao
MIT	MIT	UCF	UF	MIT

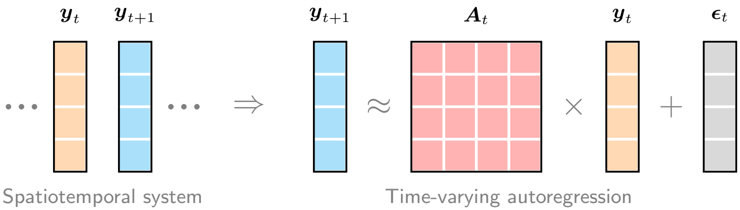
Mainly from:

- Xinyu Chen, Dingyi Zhuang, HanQin Cai, Shenhao Wang, Jinhua Zhao (2025). "Dynamic autoregressive tensor factorization for pattern discovery of spatiotemporal systems". *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 47 (10):8524-8537.

  https://doi.org/10.1109/TPAMI.2025.3576719

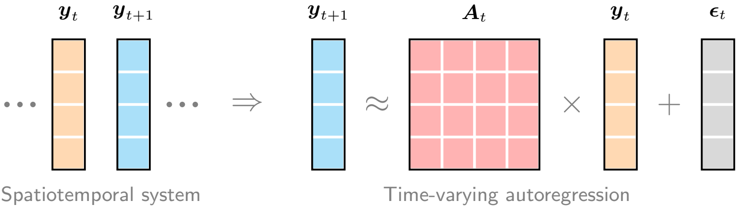
Autoregression

- How to characterize dynamical systems?



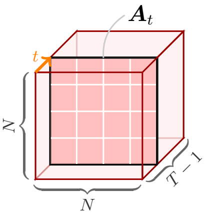
Autoregression

- How to characterize dynamical systems?

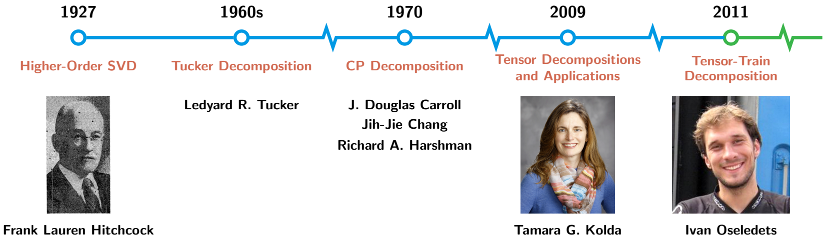


- On spatiotemporal systems $\boldsymbol{Y} \in \mathbb{R}^{N \times T}$:

$$\underbrace{\boldsymbol{y}_{t+1} = \boldsymbol{A}\boldsymbol{y}_t + \boldsymbol{\epsilon}_t}_{\text{time-invariant (e.g., DMD)}} \quad \text{v.s.} \quad \underbrace{\boldsymbol{y}_{t+1} = \boldsymbol{A}_t\boldsymbol{y}_t + \boldsymbol{\epsilon}_t}_{\text{time-varying}}$$

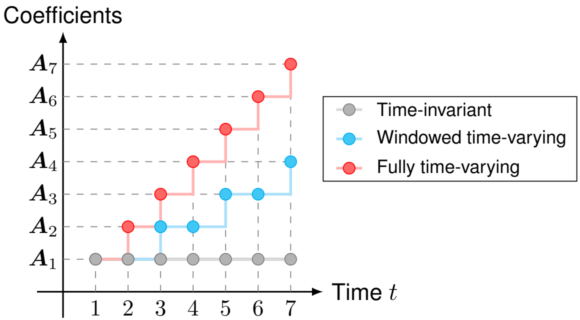


- How to discover spatial/temporal modes (patterns) from the tensor $\boldsymbol{\mathcal{A}} \triangleq \{\boldsymbol{A}_t\}_{t\in[T-1]}$?


1927
1960s
1970
2009
2011
Higher-Order SVD
Tucker Decomposition
CP Decomposition
Tensor Decompositions and Applications
Tensor-Train Decomposition
Frank Lauren Hitchcock
Ledyard R. Tucker
J. Douglas Carroll
Jih-Jie Chang
Richard A. Harshman
Tamara G. Kolda
Ivan Oseledets

- On the data $\boldsymbol{Y} \in \mathbb{R}^{N \times T}$:

$$\underbrace{\boldsymbol{y}_{t+1} = \boldsymbol{A}\boldsymbol{y}_t + \boldsymbol{\epsilon}_t}_{\text{time-invariant (e.g., DMD)}} \quad \text{v.s.} \quad \underbrace{\boldsymbol{y}_{t+1} = \boldsymbol{A}_t\boldsymbol{y}_t + \boldsymbol{\epsilon}_t}_{\text{fully time-varying (ours)}}$$



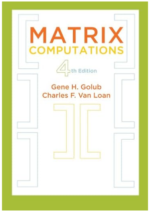
DATF

- Tensor factorization:

$$\boldsymbol{\mathcal{A}} = \underbrace{\boldsymbol{\mathcal{G}} \times_1 \boldsymbol{W} \times_2 \boldsymbol{V} \times_3 \boldsymbol{X}}_{\text{Tucker decomposition}}$$

$$\Updownarrow$$

$$\boldsymbol{A}_t = \boldsymbol{\mathcal{G}} \times_1 \underbrace{\boldsymbol{W}}_{\text{spatial modes}} \times_2 \boldsymbol{V} \times_3 \underbrace{\boldsymbol{x}_t^\top}_{\text{temporal modes}}$$



- **(Ours)** Dynamic autoregressive tensor factorization (DATF):

$$\min_{\boldsymbol{\mathcal{G}}, \boldsymbol{W}, \boldsymbol{V}, \boldsymbol{X}} \frac{1}{2} \sum_{t \in [T-1]} \left\| \boldsymbol{y}_{t+1} - (\boldsymbol{\mathcal{G}} \times_1 \boldsymbol{W} \times_2 \boldsymbol{V} \times_3 \boldsymbol{x}_t^\top) \boldsymbol{y}_t \right\|_2^2$$

$$\text{s.t.} \quad \underbrace{\boldsymbol{W}^\top \boldsymbol{W} = \boldsymbol{I}_R}_{\text{orthogonal spatial modes}}$$

- Solution: $\boldsymbol{\mathcal{G}}$ (LS) $\rightarrow$ $\boldsymbol{W}$ **(OPP)** $\rightarrow$ $\boldsymbol{V}$ (CG) $\rightarrow$ $\boldsymbol{x}_t$ (LS)

OPP

- **Orthogonal Procrustes problem (OPP)**: For any $\boldsymbol{Q} \in \mathbb{R}^{m \times r}$, $m \geq r$, the solution to

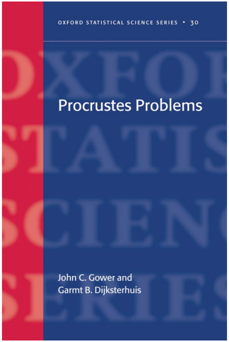
$$\min_{\boldsymbol{F}} \ \|\boldsymbol{F} - \boldsymbol{Q}\|_F^2$$
$$\text{s. t.} \ \underbrace{\boldsymbol{F}^\top \boldsymbol{F} = \boldsymbol{I}_r}_{\text{orthogonal}}$$

is

$$\boldsymbol{F} := \boldsymbol{U}\boldsymbol{V}^\top$$

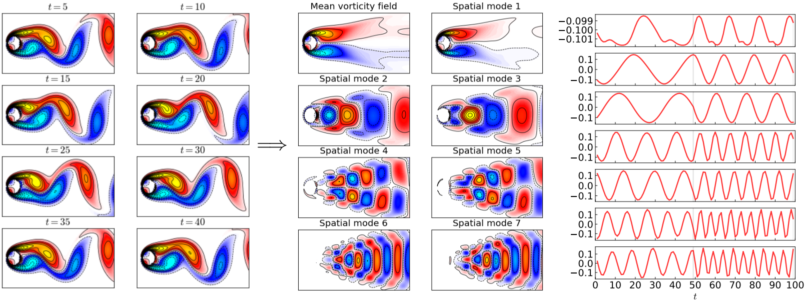
where

$$\underbrace{\boldsymbol{Q} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top}_{\text{singular value decomposition}}$$



Benchmark Evaluation

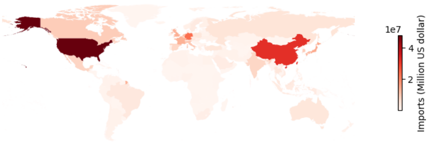
- **Multi-resolution fluid flow dataset** (the first 50 snapshots + 50 snapshots randomly selected from the last 100 snapshots)
  - Produce interpretable patterns: Low-frequency modes (dominant patterns) & high-frequency modes (e.g., secondary patterns, outliers)
  - Identify the system of different frequencies (i.e., at $t = 50$)



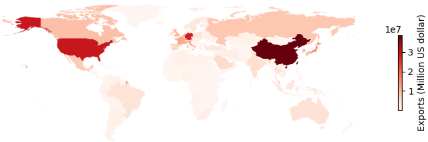
Vectorize snapshots as $\{\boldsymbol{y}_t\}_t$ | Spatial modes in $\boldsymbol{W}$ | Temporal modes in $\boldsymbol{X}$

International Trade

- **Import/Export merchandise trade values (annual)**[1] (215 countries/regions & period of 2000-2022)
  - Total merchandise trade values
  - Represent import/export trade data as a 215-by-23 matrix

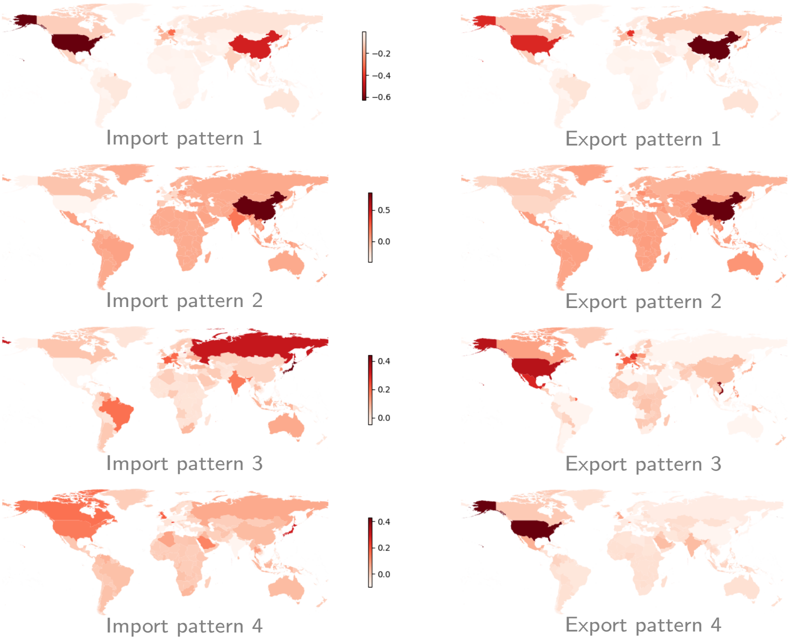


Imports from 2000 to 2022



Exports from 2000 to 2022

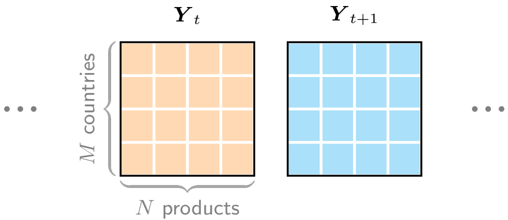
[1] The dataset is available at https://stats.wto.org.



−0.2
−0.4
−0.6
Import pattern 1
−0.2
−0.4
Export pattern 1
0.5
0.0
Import pattern 2
0.5
0.0
Export pattern 2
0.4
0.2
0.0
Import pattern 3
0.2
0.1
0.0
Export pattern 3
0.4
0.2
0.0
Import pattern 4
0.50
0.25
0.00
Export pattern 4

International Trade

- Three-dimensional trade (Economy, Product, Year)



- On spatiotemporal systems $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{M \times N \times T}$:

$$\underbrace{\boldsymbol{y}_{n,t+1} = \boldsymbol{A}_{n,t}\boldsymbol{y}_{n,t} + \boldsymbol{\epsilon}_{n,t}}_{\text{time-varying \& product-varying}}$$

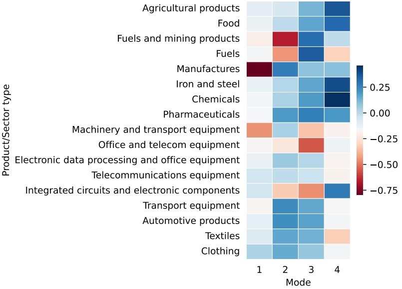
- Optimization problem of DATF:

$$\min_{\boldsymbol{\mathcal{G}},\boldsymbol{W},\boldsymbol{U},\boldsymbol{V},\boldsymbol{X}} \frac{1}{2}\sum_{n\in[N]}\sum_{t\in[T-1]} \left\|\boldsymbol{y}_{n,t+1} - (\boldsymbol{\mathcal{G}} \times_1 \boldsymbol{W} \times_2 \boldsymbol{U} \times_3 \boldsymbol{V} \times_4 \boldsymbol{x}_t^\top)\boldsymbol{y}_{n,t}\right\|_2^2$$

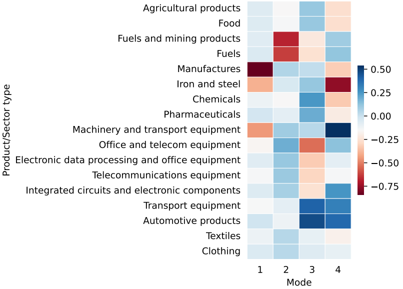
$$\text{s.t.} \quad \underbrace{\boldsymbol{W}^\top \boldsymbol{W} = \boldsymbol{I}_R}_{\text{orthogonal country patterns}}$$

Product Patterns

- On 17 merchandise types



Imports



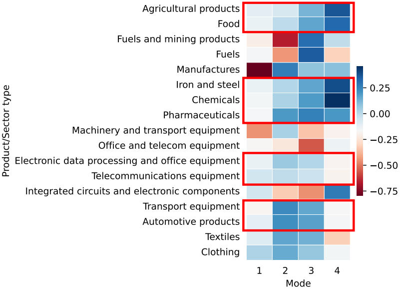
Exports

- Classify import/export merchandise according to product patterns
- Basic principle:

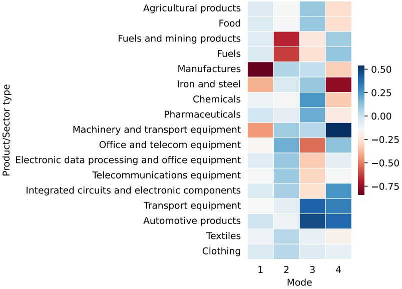
Import: What we buy? (demand) vs. Export: What we sell? (supply)

Product Patterns

- On 17 merchandise types



Imports



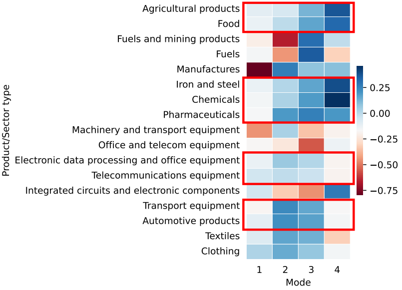
Exports

- Classify import/export merchandise according to product patterns
- Basic principle:

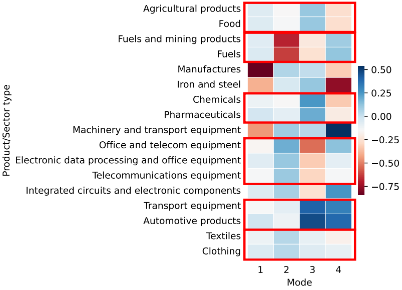
Import: What we buy? (demand) vs. Export: What we sell? (supply)

Product Patterns

- On 17 merchandise types



Imports



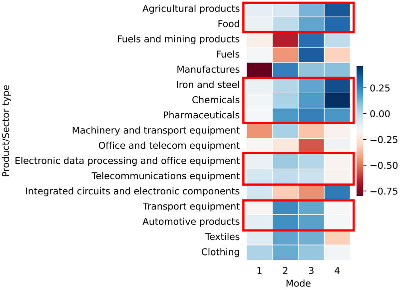
Exports

- Classify import/export merchandise according to product patterns
- Basic principle:

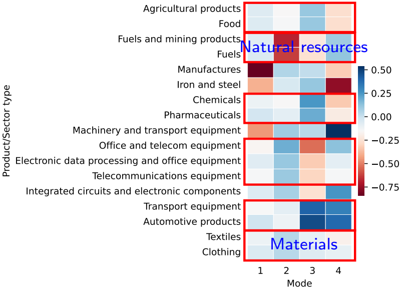
Import: What we buy? (demand) vs. Export: What we sell? (supply)

Product Patterns

- On 17 merchandise types



Imports



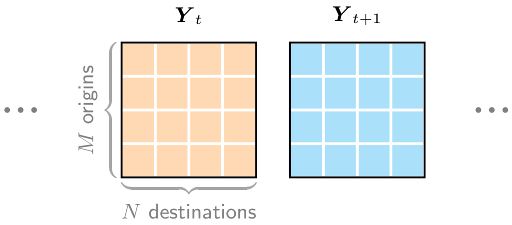
Exports

- Classify import/export merchandise according to product patterns
- Basic principle:

Import: What we buy? (demand) vs. Export: What we sell? (supply)

Human Mobility

- Origin-Destination (OD) matrices



- On spatiotemporal systems $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{M \times N \times T}$:

$$\underbrace{\boldsymbol{y}_{n,t+1} = \boldsymbol{A}_{n,t}\boldsymbol{y}_{n,t} + \boldsymbol{\epsilon}_{n,t}}_{\text{time-varying \& destination-varying}}$$

- Optimization problem of DATF:

$$\begin{aligned} \min_{\boldsymbol{\mathcal{G}},\boldsymbol{W},\boldsymbol{U},\boldsymbol{V},\boldsymbol{X}} \quad & \frac{1}{2}\sum_{n\in[N]}\sum_{t\in[T-1]}\left\|\boldsymbol{y}_{n,t+1} - (\boldsymbol{\mathcal{G}} \times_1 \boldsymbol{W} \times_2 \boldsymbol{U} \times_3 \boldsymbol{V} \times_4 \boldsymbol{x}_t^\top)\boldsymbol{y}_{n,t}\right\|_2^2 \\ \text{s.t.} \quad & \underbrace{\boldsymbol{W}^\top\boldsymbol{W} = \boldsymbol{I}_R}_{\text{orthogonal origin patterns}} \end{aligned}$$

Human Mobility

- Chicago taxi/ridesharing data

Matching Taxi Trips with Community Areas

There are three basic steps to follow for processing taxi trip data:

- Download taxi trips in 2022 in the `.csv` format, e.g., `Taxi_Trips_-_2022.csv`.
- Use the `pandas` package in Python to process the raw trip data.
- Match trip pickup/dropoff locations with boundaries of the community area.

```
import pandas as pd

data = pd.read_csv('Taxi_Trips_-_2022.csv')
data.head()
```

For each taxi trip, one can select some important information:

- `Trip Start Timestamp`: When the trip started, rounded to the nearest 15 minutes.
- `Trip Seconds`: Time of the trip in seconds.
- `Trip Miles`: Distance of the trip in miles.
- `Pickup Community Area`: The Community Area where the trip began. This column will be blank for locations ourside Chicago.
- `Dropoff Community Area`: The Community Area where the trip ended. This column will be blank for locations outside Chicago.

```
df = pd.DataFrame()
df['Trip Start Timestamp'] = data['Trip Start Timestamp']
df['Trip Seconds'] = data['Trip Seconds']
df['Trip Miles'] = data['Trip Miles']
df['Pickup Community Area'] = data['Pickup Community Area']
df['Dropoff Community Area'] = data['Dropoff Community Area']
del data
df
```

Figure 2 shows taxi pickup and dropoff trips (2022) on 77 community areas in the City of Chicago. Note that the average trip duration is **1207.75 seconds** and the average trip distance is **6.16 miles**.

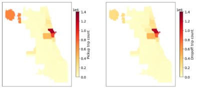


Figure 2. Taxi pickup and dropoff trips (2022) in the City of Chicago, USA. There are 4,763,961 remaining trips after the data processing.

For comparison, Figure 3 shows taxi pickup and dropoff trips (2019) on 77 community areas in the City of Chicago. Note that the average trip duration is **915.62 seconds** and the average trip distance is **3.93 miles**.

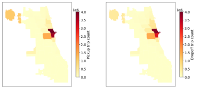


Figure 3. Taxi pickup and dropoff trips (2019) in the City of Chicago, USA. There are 12,484,572 remaining trips after the data processing. See the data processing codes.

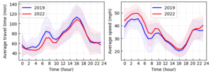


Figure 6. Average travel time and speed from area 32 (i.e., Downtown) to area 76 (i.e., Airport) in both 2019 and 2022.

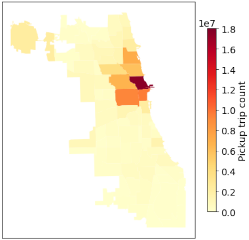
```
import numpy as np
import matplotlib.pyplot as plt

fig = plt.figure(figsize = (8, 2.5))
ax = fig.add_subplot(1, 2, 1)
# Average travel time in 2019
m1 = df1.groupby(['hour'])['Trip Seconds'].mean().values / 30
s1 = df1.groupby(['hour'])['Trip Seconds'].std().values / 30
plt.plot(m1, color = 'blue', linewidth = 1.8, label = '2019')
upper = m1 + s1
lower = m1 - s1
x_bound = np.append(np.append(np.append(np.array([0, 0]), np.arange(0, 24)),
                    np.array([24 - 1, 24 - 1])), np.arange(24 - 1, -1,
y_bound = np.append(np.append(np.append(np.array([upper[0], lower[0]]), lower),
                    np.array([lower[-1], upper[-1]])), np.flip(upper))
plt.fill(x_bound, y_bound, color = 'blue', alpha = 0.05)

# Average travel time in 2022
m1 = df2.groupby(['hour'])['Trip Seconds'].mean().values / 30
s1 = df2.groupby(['hour'])['Trip Seconds'].std().values / 30
plt.plot(m1, color = 'red', linewidth = 1.8, label = '2022')
upper = m1 + s1
lower = m1 - s1
```

Source: https://spatiotemporal-data.github.io/Chicago-mobility/taxi-data

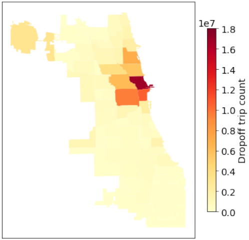
Human Mobility

- **Ridesharing:** 96,642,881 trips in 2019 vs. 57,290,954 trips in 2022

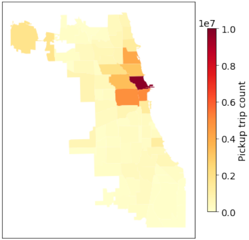
Pickup trips (2019)



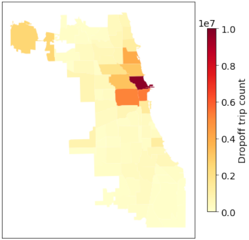
Dropoff trips (2019)



Pickup trips (2022)



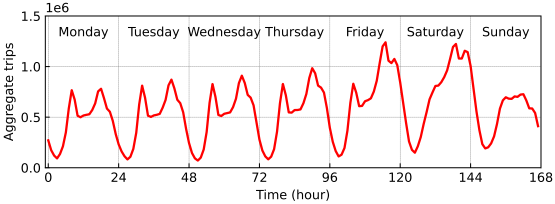
Dropoff trips (2022)



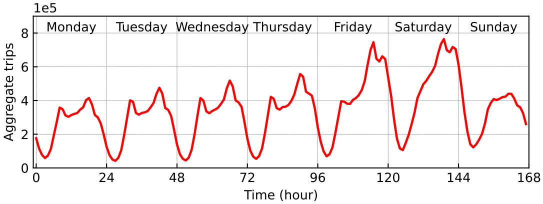
Human Mobility

- **Ridesharing:** 96,642,881 trips in 2019 vs. 57,290,954 trips in 2022

Pickup trips aggregated over 52 weeks in 2019

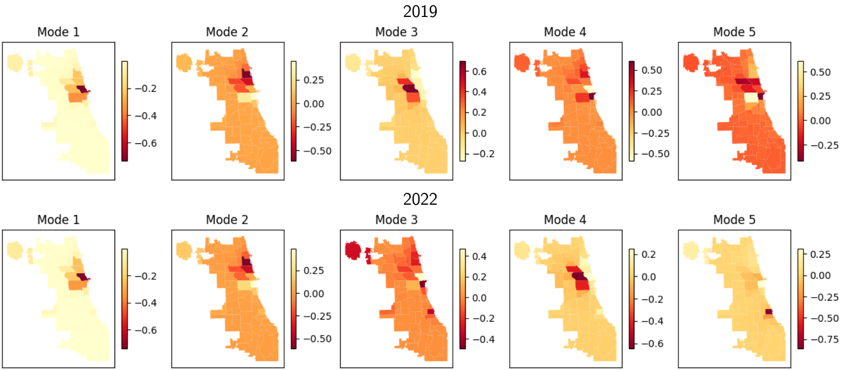


Pickup trips aggregated over 52 weeks in 2022



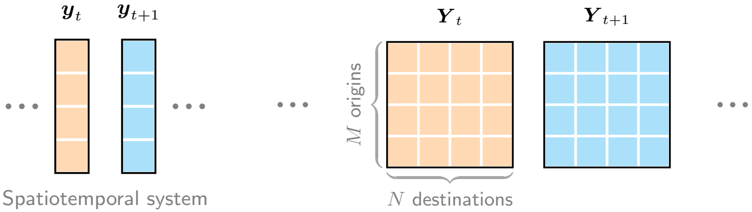
Human Mobility

- Ridesharing trip data: 77 origins $\times$ 77 destinations $\times$ 168 hours
- Our model Identifies the changes in pickup zones before and after COVID-19



Concluding Remark

- Discovering spatial/temporal patterns from 2D and 3D spatiotemporal systems with unsupervised learning:
  - Time-varying autoregression on the data
  - Tensor factorization on the coefficients



Thanks for your attention!

Any Questions?

Slides: https://xinychen.github.io/slides/dynamic_tensor.pdf

About me:

- Homepage: https://xinychen.github.io
- GitHub: https://github.com/xinychen
- How to reach me: chenxy346@gmail.com