# transdim: Machine Learning for Transportation Data Imputation and Prediction

Reproducible Research Workshop

TRB 103rd Annual Meeting · Washington, D.C., USA

**Xinyu Chen**

January 11, 2024

Open-source & reproducible research:

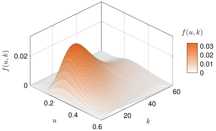
1. GitHub: https://github.com/xinychen
2. Slides: https://xinychen.github.io/slides/transdim.pdf

ML algorithms

transdim

**(1.1k stars)**

Visualization tools

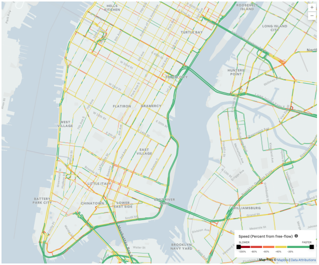



awesome-latex-drawing

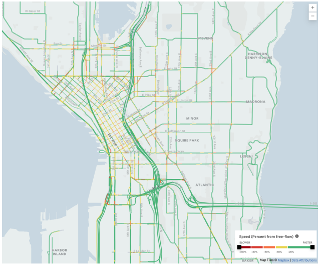
**(1.2k stars)**

# Storytelling with Data

- Uber (hourly) movement speed data



NYC movement



Seattle movement

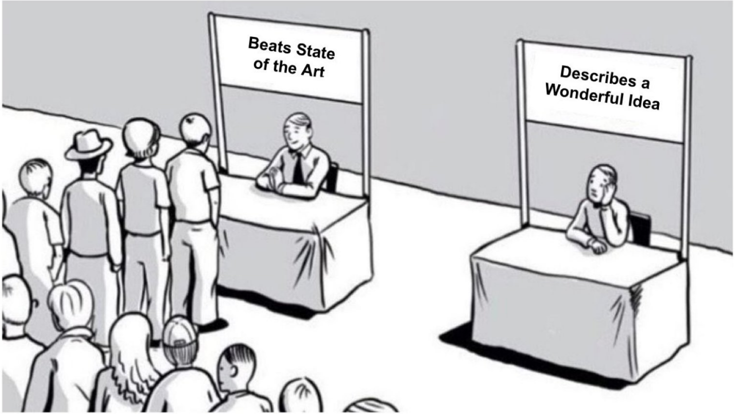
- {road segment, time step (hour), average speed}
- $\boldsymbol{Y} \in \mathbb{R}^{N \times T}$ with $N$ spatial locations $\times$ $T$ time steps
- Computing hourly speed: Road segments have 5+ unique trips.

**Issue**: Insufficient sampling of ridesharing vehicles on the road network!

# Storytelling with Data

- Data
- Quality
- Sparsity
- Estimation
- Imputation
- Interpolation
- Forecasting

# Storytelling with Data




Source: Twitter

# Traffic Data Processing

Computing with `numpy` (numerical computing in Python)

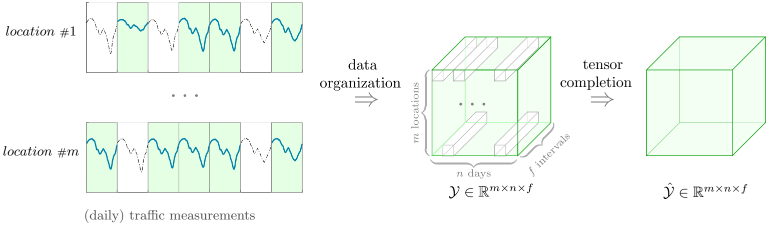
- Data format: `.npz` (compressed format of `.npy`)
- Example
- Easy to connect with `numpy` (in CPU environment) & `cupy` (in GPU environment)

# Reformulate Traffic Data Imputation

- Represent traffic data as tensors

$$\text{Tensorization: } \boldsymbol{Y} \in \mathbb{R}^{m \times t} \rightarrow \boldsymbol{\mathcal{Y}} \in \mathbb{R}^{m \times n \times f}$$

w/ $m$ locations, $n$ days, and $f$ time intervals per day.




- Tensor completion (Observed index set $\Omega$)

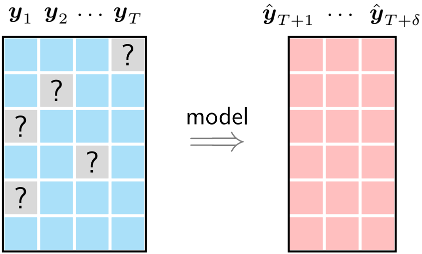
$$\underbrace{\mathcal{P}_{\Omega}(\boldsymbol{\mathcal{Y}})}_{\text{Partially observed}} \xrightarrow{\text{Estimate}} \underbrace{\mathcal{P}_{\Omega}^{\perp}(\boldsymbol{\mathcal{Y}})}_{\text{Unobserved}}$$

# Reformulate Traffic Forecasting

Forecasting urban traffic states with sparse data

- Problem definition ($\delta$-step ahead forecasting)

$$\underbrace{\{\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_T\}}_{\text{Current traffic states}} \xrightarrow{\text{Estimate}} \underbrace{\{\hat{\boldsymbol{y}}_{T+1}, \hat{\boldsymbol{y}}_{T+2}, \ldots, \hat{\boldsymbol{y}}_{T+\delta}\}}_{\text{Future traffic states}}$$



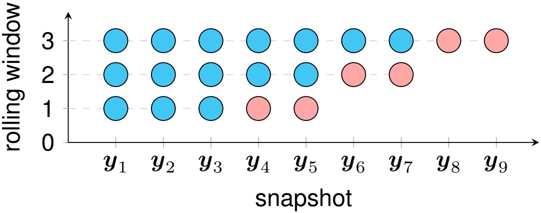
# Reformulate Traffic Forecasting

(Rolling) Forecasting urban traffic states with sparse data

1st rolling step: $\{\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_3\} \rightarrow \{\boldsymbol{y}_4, \boldsymbol{y}_5\}$

2nd rolling step: $\{\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_3, \boldsymbol{y}_4, \boldsymbol{y}_5\} \rightarrow \{\boldsymbol{y}_6, \boldsymbol{y}_7\}$

3rd rolling step: $\{\boldsymbol{y}_1, \boldsymbol{y}_2, \boldsymbol{y}_3, \boldsymbol{y}_4, \boldsymbol{y}_5, \boldsymbol{y}_6, \boldsymbol{y}_7\} \rightarrow \{\boldsymbol{y}_8, \boldsymbol{y}_9\}$



# TMF

TMF[1,2]
Jupyter Notebook

[1] tracebase:
[2] tpami

# Switch from CPU to GPU

Python implementation of algorithms with the `numpy` package (Using less packages can improve the reproducibility)
Easy to convert the codes from CPU to GPU

`import numpy as np` $\Rightarrow$ `import cupy as np`

# Post Something That Matters

Post well-documented data processing files (e.g., processing Chicago taxi data)

- Beginners to build coding skills
- Researchers to build research ideas

### 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**.

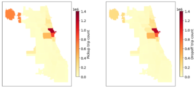



**Figure 2.** Taxi pickup and dropoff trips (2022) in the City of Chicago, USA. There are 4,763,961 remaining trips after the data processing.

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**.

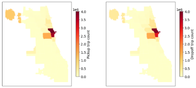



**Figure 3.** Taxi pickup and dropoff trips (2019) in the City of Chicago, USA. There are 12,484,572 remaining trips after the data processing. See the data processing codes.

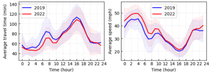



**Figure 6.** Average travel time and speed from area 32 (i.e., Downtown) to area 76 (i.e., Airport) in both 2019 and 2022.

```
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

# Post Something That Matters

Post scientific data modeling problems

## Optimizing Interpretable Time-Varying Autoregression with Orthogonal Constraints

Generally speaking, any spatiotemporal data in the form of a matrix can be written as $\boldsymbol{Y}\in\mathbb{R}^{N\times T}$ with $N$ spatial areas/locations and $T$ time steps. To discover interpretable spatial/temporal patterns, one can build a time-varying autoregression on the time snapshots $\boldsymbol{y}_1,\boldsymbol{y}_2,\ldots,\boldsymbol{y}_T\in\mathbb{R}^N$ (Chen et al., 2023). The time-varying coefficients in the autoregression allow one to characterize the time-varying system behavior, but the challenges still remain.

To capture interpretable modes/patterns, one can use tensor factorization formulas to parameterize the coefficients and the optimization problem can be easily built. However, a great challenge would be how to make the modes "more interpretable", specifically, e.g., how to learn orthogonal modes in the modeling process. In this post, we present an optimization problem of the time-varying autoregression with orthogonal constraints as follows,

$$\begin{aligned}\min_{\boldsymbol{W},\boldsymbol{G},\boldsymbol{V},\boldsymbol{X}}\;&\frac{1}{2}\sum_{t=2}^{T}\left\|\boldsymbol{y}_t-\boldsymbol{W}\boldsymbol{G}(\boldsymbol{x}_t^\top\otimes\boldsymbol{V})^\top\boldsymbol{y}_{t-1}\right\|_2^2\\ \text{s.t.}\;&\begin{cases}\boldsymbol{W}^\top\boldsymbol{W}=\boldsymbol{I}_R\\ \boldsymbol{V}^\top\boldsymbol{V}=\boldsymbol{I}_R\\ \boldsymbol{X}^\top\boldsymbol{X}=\boldsymbol{I}_R\end{cases}\end{aligned}$$

where $\boldsymbol{W}\in\mathbb{R}^{N\times R}$ and $\boldsymbol{X}\in\mathbb{R}^{(T-1)\times R}$ refer to as the spatial modes and the temporal modes, respectively. This model can discover urban mobility transition patterns.

Source: https://spatiotemporal-data.github.io/probs/orth-var

# Why?

Academic:

- Sustainable research environment (w.r.t. us & followers)
- Interact with researchers from different fields
- Provide platform and benchmark for comparison
- Stimulate new algorithmic ideas

Industry:

- Solution to ...

Next-step plan:

- Spatiotemporal data modeling:
  https://spatiotemporal-data.github.io

POLYTECHNIQUE MONTRÉAL

UNIVERSITÉ D'INGÉNIERIE

IVADO

# Thanks for your attention!

## Any Questions?

**About me**:

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