Robust Factorization of Real-world Tensor Streams with Patterns, Missing Values, and Outliers

Dongjin Lee

Kijung Shin
Road Map

- **Introduction**
- Problem Definition
- Proposed Method: **SOFIA**
  - Tensor Factorization Model
  - Optimization Algorithm
- Experiment Results
- Conclusions
Tensors are Everywhere

Tensors

• Multi-dimensional arrays
• Tensors are effective tools to represent multi-aspect data.

Ex) Taxi origin-destination data
Ex) Indoor sensor data
Tensor Streams

- In many applications, tensor data are usually collected incrementally over time in the form of a tensor stream.

Ex) Taxi origin-destination data
Ex) Indoor sensor data
Missing Values and Outliers in Real-world Tensor Data

- Real-world tensor streams often include **missing entries** (e.g., due to network disconnection) and unexpected **outliers** (e.g., due to system errors).

Ex) Taxi ride origin-destination tensor

Ex) Indoor sensor record tensor
Questions

Given that real-world tensor streams with missing values and outliers

?” How can we estimate the missing entries?

?” Can we also predict future entries?

?” Can both imputation and prediction be performed accurately in an online manner?
Traditional Methods and Their Limitations

• **CANDECOMP/PARAFAC (CP) Factorization** of Incomplete Tensors

**Consider:** $\mathbf{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, $\Omega \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, rank $R$

**To Find:** factor matrices $\mathbf{U}^{(1)}, \ldots, \mathbf{U}^{(N)}$

Minimize: $\frac{1}{2} \left\| \Omega \odot (\mathbf{X} - [U^{(1)}, \ldots, U^{(N)}]) \right\|_F^2$

where $[U^{(1)}, \ldots, U^{(N)}] = \sum_{r=1}^R U^{(1)}(:, r) \circ \cdots \circ U^{(N)}(:, r)$.

Ex) 3-order tensor case

$\omega_{i_1 \cdots i_N} = \begin{cases} 1 & \text{if } x_{i_1 \cdots i_N} \text{ is known,} \\ 0 & \text{if } x_{i_1 \cdots i_N} \text{ is missing.} \end{cases}$

※ Limitations of standard CP factorization
• Does not handle tensor streams
• Vulnerable to outliers
Traditional Methods and Their Limitations (cont.)

Holt-Winters Forecasting Algorithm (Exponential Smoothing)

- Decomposes time series into **level**, **trend (slope)**, **seasonality** components
- Using three components, it can forecast future evolution of the time series.

Three smoothing equations:

Forecasting equation:

\[
\hat{y}_{t+h|t} = l_t + hb_t + s_{t+h-m(\lfloor \frac{h-1}{m} \rfloor + 1)}
\]

From Hyndman, Rob J., and George Athanasopoulos. *Forecasting*. OTexts, 2018.

※ Limitations of standard Holt-Winters Method

- Does not handle missing values
- Vulnerable to outliers
In this work

We propose **SOFIA**, a CP factorization-based streaming algorithm for real-world tensors with missing entries and outliers.

**Questions**

- How can we *estimate the missing entries*?
- Can we also *predict future entries*?
- Can both imputation and prediction *be performed accurately in an online manner*?
Road Map

- Introduction
- **Problem Definition**
- Proposed Method: **SOFIA**
  - Tensor Factorization Model
  - Optimization Algorithm
- Experiment Results
- Conclusions
Problem Definition: Imputation & Forecasting

• **Given:**
  • A sequence of incomplete and noisy subtensors

• **Estimate:**
  • Missing entries in the subtensors (**imputation**)
  • Future values of the tensor stream (**forecasting**)

• **To minimize:**
  • Imputation and forecasting error

Estimate the value of missing entries

Forecasting future tensor streams
Road Map

• Introduction
• Problem Definition
• Proposed Method: SOFIA
  • Tensor Factorization Model
  • Optimization Algorithm
• Experiment Results
• Conclusions
Proposed Tensor Factorization Model

For static tensors

• **Input**
  - Corrupted tensor: \( \mathbf{Y} \in \mathbb{R}^{I_1 \times \cdots \times I_N} (\mathbf{Y} = \mathbf{X} + \mathbf{O}) \)

• **Optimization Problem**

\[
\min_{\{\mathbf{U}^{(n)}\}_{n=1}^N, \mathbf{O}} \| \mathbf{O} \odot (\mathbf{Y} - \mathbf{O} - \mathbf{X}) \|_F^2 + \lambda_1 \| \mathbf{L}_1 \mathbf{U}^{(N)} \|_F^2 + \lambda_2 \| \mathbf{L}_m \mathbf{U}^{(N)} \|_F^2 + \lambda_3 \| \mathbf{O} \|_1
\]

- Encourage temporal smoothness
- Encourage seasonal smoothness
- Encourage sparsity of outliers

• **Output**
  - Non-temporal factor matrices: \( \{\mathbf{U}^{(n)}\}_{n=1}^{N-1} \)
  - Temporal factor matrix: \( \mathbf{U}^{(N)} \) (Temporally and seasonally smooth)
  - Outlier tensor: \( \mathbf{O} \)

Temporal properties in real-world tensor streams.

1. **Temporal smoothness**
2. **Seasonal smoothness**

\[
\begin{align*}
\mathbf{L}_1 & = \begin{bmatrix}
1 & -1 \\
1 & 1 \\
& \ddots
\end{bmatrix} \\
\mathbf{L}_m & = \begin{bmatrix}
1 & -1 \\
1 & 1 \\
& \ddots
\end{bmatrix}
\end{align*}
\]

\( \star m: \) seasonal period
For dynamic tensors

**Input**
- Sequence of corrupted subtensor: $\mathbf{y}_t \in \mathbb{R}^{I_1 \times \cdots \times I_{N-1}}$ ($\mathbf{y}_t = \mathbf{x}_t + \mathbf{o}_t$), where $t = 1, 2, \cdots$

**Optimization Problem**
- At each time $t$,
  $$
  \min_{\{u^{(n)}_{(n)}\}_{n=1}^{N-1}, \{u^{(N)}_{(n)}\}, \{o_t\}} \sum_{\tau=1}^{t} \left[ \|\Omega_{\tau} \otimes (\mathbf{y}_\tau - \mathbf{o}_\tau - \mathbf{x}_\tau)\|_F^2 + \lambda_1 \left\| u^{(N)}_{\tau-1} - u^{(N)}_{\tau} \right\|_F^2 + \lambda_2 \left\| u^{(N)}_{\tau-m} - u^{(N)}_{\tau} \right\|_F^2 + \lambda_3 \left\| \mathbf{o}_\tau \right\|_1 \right]
  $$

**Output**
- Non-temporal factor matrices: $\{U^{(n)}\}_{n=1}^{N-1}$
- Temporal factor vectors: $u^{(N)}_{t}$, where $t = 1, 2, \cdots$ (Temporally and seasonally smooth)
- Outlier tensor: $\mathbf{o}_t$, where $t = 1, 2, \cdots$

**Q. How to solve the optimization problem incrementally in an online manner?**
Proposed Optimization Algorithm

• SOFIA’s optimization algorithm consists of three steps.

  1. Initialization
  2. Fitting the Holt-Winters model
  3. Dynamic Update
**SOFIA – Step1. Initialization**

- **Initialize all the factor matrices** $\{U^{(n)}\}_{n=1}^N$ by solving the batch optimization problem using a subset of the corrupted tensor data over a short period of time (e.g., 3 seasons).

**Ex) 3-order tensor case**

**Step1. Initialization**

- **Input corrupted and incomplete tensor**
- **$Y_{init}$**
- **$U^{(1)}$** (1st Mode)
- **$U^{(2)}$** (2nd Mode)
- **$U^{(3)}$** (3rd Mode (Time Mode))
- **temporally and seasonally smooth**
- **Cleaned low-rank tensor**
- **Sparse outlier tensor**
**SOFIA – Step1. Initialization (cont.)**

**Procedure:**

**SOFIA Initialization**

**Input:** $Y_{\text{init}}$

**Initialization:** $\{U^{(n)}\}_{n=1}^N, O_{\text{init}}$

While $\{U^{(n)}\}_{n=1}^N$ does not converge:

- Reject Outliers, $Y_{\text{init}} - O_{\text{init}}$
- Update $\{U^{(n)}\}_{n=1}^N$ by SOFIA$_{\text{ALS}}$
- Update $O_{\text{init}}$

**Output:** $\{U^{(n)}\}_{n=1}^N, O_{\text{init}}$

※ **SOFIA$_{\text{ALS}}$:** the alternating least squares (ALS) method to minimize the objective function in the batch scenario.

Ex) 3-order tensor case
SOFIA – Step 1. Initialization (cont.)

**Procedure:**

**SOFIA Initialization**

**Input:** $\mathbf{Y}_{\text{init}}$

**Initialization:** $\{\mathbf{U}^{(n)}\}_{n=1}^{N}, \mathbf{O}_{\text{init}}$

While $\{\mathbf{U}^{(n)}\}_{n=1}^{N}$ does not converge:

- Reject Outliers, $\mathbf{Y}_{\text{init}} - \mathbf{O}_{\text{init}}$
- Update $\{\mathbf{U}^{(n)}\}_{n=1}^{N}$ by SOFIA$_{\text{ALS}}$
- Update $\mathbf{O}_{\text{init}}$

**Output:** $\{\mathbf{U}^{(n)}\}_{n=1}^{N}, \mathbf{O}_{\text{init}}$

※ **SOFIA$_{\text{ALS}}$:** the alternating least squares (ALS) method to minimize the objective function in the batch scenario.

### Example: 3-order tensor case

![3-order tensor case](image)
SOFIA – Step1. Initialization (cont.)

Procedure:
SOFIA Initialization

Input: $\mathcal{Y}_{init}$
Initialization: $\{U^{(n)}\}_{n=1}^N, \mathcal{O}_{init}$

While $\{U^{(n)}\}_{n=1}^N$ does not converge:

- Reject Outliers, $\mathcal{Y}_{init} - \mathcal{O}_{init}$
- Update $\{U^{(n)}\}_{n=1}^N$ by SOFIA$_{ALS}$
- Update $\mathcal{O}_{init}$

Output: $\{U^{(n)}\}_{n=1}^N, \mathcal{O}_{init}$

※ SOFIA$_{ALS}$: the alternating least squares (ALS) method to minimize the objective function in the batch scenario.

$$\min_{\{U^{(n)}\}_{n=1}^N, \mathcal{O}} \|\mathcal{O} \otimes (\mathcal{Y} - \mathcal{O} - \mathcal{X})\|_F^2 + \lambda_1 \|L_1 U^{(N)}\|_F^2 + \lambda_2 \|L_m U^{(N)}\|_F^2 + \lambda_3 \|\mathcal{O}\|_1$$

Ex) 3-order tensor case

$\mathcal{Y}_{init}$ $\approx$ $\times$ $U^{(3)}$ (Temporal factor matrix)

$\times$ $U^{(1)}$ $\times$ $U^{(2)}$
Procedure: SOFIA Initialization

Input: $\mathbf{y}_{\text{init}}$
Initialization: $\{\mathbf{u}^{(n)}\}_{n=1}^{N}, \phi_{\text{init}}$

While $\{\mathbf{u}^{(n)}\}_{n=1}^{N}$ does not converge:

- Reject Outliers, $\mathbf{y}_{\text{init}} - \phi_{\text{init}}$
- Update $\{\mathbf{u}^{(n)}\}_{n=1}^{N}$ by SOFIA$_{\text{ALS}}$
- Update $\phi_{\text{init}}$

Output: $\{\mathbf{u}^{(n)}\}_{n=1}^{N}, \phi_{\text{init}}$

※ SOFIA$_{\text{ALS}}$: the alternating least squares (ALS) method to minimize the objective function in the batch scenario.

$$\min_{\{\mathbf{u}^{(n)}\}_{n=1}^{N}, \phi} \| \mathbf{\Omega} \otimes (\mathbf{y} - \phi - \mathbf{x}) \|_F^2 + \lambda_1 \| \mathbf{L}_1 \mathbf{u}^{(N)} \|_F^2 + \lambda_2 \| \mathbf{L}_m \mathbf{u}^{(N)} \|_F^2 + \lambda_3 \| \phi \|_1$$

Ex) 3-order tensor case

$$= \mathbf{y}_{\text{init}} - \left[ \{\mathbf{u}^{(n)}\}_{n=1}^{N} \right] = \mathbf{r}$$

$$= \text{sign} (\mathbf{r}) \cdot \max (|\mathbf{r}| - \lambda_3, 0)$$
Procedure: 
SOFIA Initialization

Input: $\mathbf{Y}_{\text{init}}$

Initialization: $\{\mathbf{U}^{(n)}\}_{n=1}^{N}, \mathbf{O}_{\text{init}}$

While $\{\mathbf{U}^{(n)}\}_{n=1}^{N}$ does not converge:

- Reject Outliers, $\mathbf{Y}_{\text{init}} - \mathbf{O}_{\text{init}}$
- Update $\{\mathbf{U}^{(n)}\}_{n=1}^{N}$ by SOFIA\textsubscript{ALS}
- Update $\mathbf{O}_{\text{init}}$

Output: $\{\mathbf{U}^{(n)}\}_{n=1}^{N}, \mathbf{O}_{\text{init}}$

※ SOFIA\textsubscript{ALS}: the alternating least squares (ALS) method to minimize the objective function in the batch scenario.

\[
\min_{\{\mathbf{U}^{(n)}\}_{n=1}^{N}, \mathbf{O}} \|\mathbf{Y} \otimes (\mathbf{U} - \mathbf{O} - \mathbf{X})\|_F^2 + \lambda_1 \|\mathbf{L}_1 \mathbf{U}^{(N)}\|_F^2 + \lambda_2 \|\mathbf{L}_m \mathbf{U}^{(N)}\|_F^2 + \lambda_3 \|\mathbf{O}\|_1
\]
SOFIA – Step1. Initialization (cont.)

**Procedure:**

**SOFIA Initialization**

Input: \( \mathbf{Y}_{\text{init}} \)

Initialization: \( \{ \mathbf{U}^{(n)} \}_{n=1}^{N}, \mathbf{O}_{\text{init}} \)

While \( \{ \mathbf{U}^{(n)} \}_{n=1}^{N} \) does not converge:

- Reject Outliers, \( \mathbf{Y}_{\text{init}} - \mathbf{O}_{\text{init}} \)
- Update \( \{ \mathbf{U}^{(n)} \}_{n=1}^{N} \) by SOFIA\(_{ALS} \)
- Update \( \mathbf{O}_{\text{init}} \)

Output: \( \{ \mathbf{U}^{(n)} \}_{n=1}^{N}, \mathbf{O}_{\text{init}} \)

※ **SOFIA\(_{ALS} \):** the alternating least squares (ALS) method to minimize the objective function in the batch scenario.

\[
\min_{\{ \mathbf{U}^{(n)} \}_{n=1}^{N}, \mathbf{O}} \| \mathbf{Y} \otimes (\mathbf{Y} - \mathbf{O} - \mathbf{X}) \|_F^2 + \lambda_1 \| \mathbf{L}_1 \mathbf{U}^{(N)} \|_F^2 + \lambda_2 \| \mathbf{L}_m \mathbf{U}^{(N)} \|_F^2 + \lambda_3 \| \mathbf{O} \|_1
\]
SOFIA – Step 1. Initialization (cont.)

**Procedure:**

**SOFIA Initialization**

**Input:** $\mathbf{Y}_{\text{init}}$

**Initialization:** \{\mathbf{U}^{(n)}\}_{n=1}^N, \mathbf{O}_{\text{init}}$

While \{\mathbf{U}^{(n)}\}_{n=1}^N does not converge:

- Reject Outliers, $\mathbf{Y}_{\text{init}} - \mathbf{O}_{\text{init}}$
- Update \{\mathbf{U}^{(n)}\}_{n=1}^N by SOFIA\text{ALS}
- Update $\mathbf{O}_{\text{init}}$

**Output:** \{\mathbf{U}^{(n)}\}_{n=1}^N, $\mathbf{O}_{\text{init}}$

※ SOFIA\text{ALS}: the alternating least squares (ALS) method to minimize the objective function in the batch scenario.

**Objective Function:**

$$\min_{\{\mathbf{U}^{(n)}\}_{n=1}^N, \mathbf{O}} \| \mathbf{\Omega} \odot (\mathbf{Y} - \mathbf{O} - \mathbf{X}) \|_F^2 + \lambda_1 \| \mathbf{L}_1 \mathbf{U}^{(N)} \|_F^2 + \lambda_2 \| \mathbf{L}_m \mathbf{U}^{(N)} \|_F^2 + \lambda_3 \| \mathbf{O} \|_1$$

Ex) 3-order tensor case

$$= \mathbf{Y}_{\text{init}} - [\{\mathbf{U}^{(n)}\}_{n=1}^N] = \mathbf{R}$$

$$= \text{sign}(\mathbf{R}) \cdot \max(|\mathbf{R}| - \lambda_3, 0)$$
Step 1. Initialization (cont.)

Input: $\mathbf{Y}_\text{init}$

**Initialization:**

$\mathbf{U}(n)_{n=1}^{N}$, $\mathbf{O}_\text{init}$

While $\mathbf{U}(n)_{n=1}^{N}$ does not converge:

- Reject Outliers, $\mathbf{Y}_\text{init} - \mathbf{O}_\text{init}$
- Update $\{\mathbf{U}(n)_{n=1}^{N}\}$ by SOFIA ALS
- Update $\mathbf{O}_\text{init}$

Output: $\mathbf{U}(n)_{n=1}^{N}$, $\mathbf{O}_\text{init}$

Procedure: SOFIA Initialization

※ SOFIA ALS: the alternating least squares (ALS) method to minimize the objective function in the batch scenario.

$\min \{\mathbf{U}(n)_{n=1}^{N}, \mathbf{O}_\text{init}\} \| \mathbf{Y} - \mathbf{O}_\text{init} - \mathbf{X} \|_F^2 + \lambda_1 \| \mathbf{L}_1 \mathbf{U}(n)_{n=1}^{N} \|_F^2 + \lambda_2 \| \mathbf{L}_m \mathbf{U}(n)_{n=1}^{N} \|_F^2 + \lambda_3 \| \mathbf{O}_\text{init} \|_1$

- 3-order tensor case
- Temporally and seasonally smooth
- Sparse Outliers

EX) 3-order tensor case
SOFIA – Step 2. Fitting the Holt-Winters model

- Each column vectors of $U^{(N)}$
  
  = Seasonal time series of length $3m$ with period $m$
- Using the Holt-Winters model, we can forecast the next temporal factor vector.

Ex) 3-order tensor case

For each column, we obtain
- Level ($l$)
- Trend ($b$)
- Seasonality ($s$) components.
SOFIA – Step3. Dynamic Update

**Procedure:**

**SOFIA Dynamic Update**

**Input:** $y_t$

**Given:**

\[
\{u_{t-1}^{(n)}\}_{n=1}^{N-1}, \{u_{t}^{(N)}\}_{\tau=t-m}^{t-1}
\]

For $t = 3m + 1, 3m + 2, \ldots$ do

- Estimate $\mathcal{O}_t$
- Update $\{u_{t}^{(n)}\}_{n=1}^{N-1}$
- Update $u_{t}^{(N)}$
- Update Holt-Winters model
- Estimate missing values

**Ex) 3-order tensor case**

**Given:**

\[
\{u_{t-1}^{(n)}\}_{n=1}^{N-1}, \{u_{t}^{(N)}\}_{\tau=t-m}^{t-1}
\]

**Input:** $y_t$
**Procedure:**
SOFIA Dynamic Update

**Input:** $Y_t$

**Given:**
\[ \{u_{t-1}^{(n)}\}_{n=1}^{N-1}, \{u_{t}^{(N)}\}_{t=t-m}^{t-1} \]

For $t = 3m + 1, 3m + 2, \ldots$ do
- Estimate $O_t$
- Update $\{u_{t}^{(n)}\}_{n=1}^{N-1}$
- Update $u_{t}^{(N)}$
- Update Holt-Winters model
- Estimate missing values

**Ex) 3-order tensor case**

**Estimates:**
- $\hat{Y}_{t|t-1}$
- $\hat{u}_{t|t-1}$

**Filtered by 2-sigma rule**

$\sigma_t$
SOFIA - Step 3. Dynamic Update (cont.)

**Procedure:**
SOFIA Dynamic Update

**Input:** \( y_t \)

**Given:** \( \{ u_t^{(n)} \}_{n=1}^{N-1}, \{ u_{t}^{(N)} \} \)

For \( t = 3m + 1, 3m + 2, \ldots \) do
- Estimate \( O_t \)
- Update \( \{ u_t^{(n)} \}_{n=1}^{N-1} \)
- Update \( u_t^{(N)} \)
- Update Holt-Winters model
- Estimate missing values

**Ex) 3-order tensor case**

\[
\min_{\{ u_t^{(n)} \}_{n=1}^{N-1}} \min_{u_t^{(N)}}, \sigma_t \left\| \Omega_t \odot (y_t - o_t - x_t) \right\|_F^2 + \lambda_1 \left\| u_{t-1}^{(n)} - u_t^{(n)} \right\|_F^2 + \lambda_2 \left\| u_{t-m}^{(n)} - u_t^{(n)} \right\|_F^2 + \lambda_3 \left\| \sigma_t \right\|_1
\]
**Procedure: SOFIA Dynamic Update**

Input: $\mathbf{y}_t$

Given: $\left\{\mathbf{u}^{(n)}_{t-1}\right\}_{n=1}^{N-1}, \left\{\mathbf{u}^{(N)}_t\right\}_{t=t-m}^{t-1}$

For $t = 3m + 1, 3m + 2, \ldots$ do

- Estimate $\mathbf{O}_t$
- Update $\left\{\mathbf{u}^{(n)}_t\right\}_{n=1}^{N-1}$
- Update $\mathbf{u}^{(N)}_t$
- Update Holt-Winters model
- Estimate missing values

Gradient Descent

\[
\begin{aligned}
\min_{\left\{\mathbf{u}^{(n)}_t\right\}_{n=1}^{N-1}, \mathbf{O}_t} \|\mathbf{O}_t \odot (\mathbf{y}_t - \mathbf{O}_t - \mathbf{X}_t)\|_F^2 + \lambda_1 \left\|\mathbf{u}^{(N)}_{t-1} - \mathbf{u}^{(N)}_t\right\|_F^2 + \lambda_2 \left\|\mathbf{u}^{(N)}_{t-m} - \mathbf{u}^{(N)}_t\right\|_F^2 + \lambda_3 \|\mathbf{O}_t\|_1
\end{aligned}
\]
SOFIA – Step 3. Dynamic Update (cont.)

**Procedure:**

**SOFIA Dynamic Update**

**Input:** $y_t$

**Given:**

\[ \{u_{t-1}^{(n)}\}_{n=1}^{N-1}, \{u_t^{(N)}\}_{t=t-m}^{t-1} \]

For $t = 3m + 1, 3m + 2, \cdots$ do

- Estimate $\Omega_t$
- Update $\{u_t^{(n)}\}_{n=1}^{N-1}$
- Update $u_t^{(N)}$
- Update Holt-Winters model
- Estimate missing values

**Ex) 3-order tensor case**

Update the HW model with $u_t^{(N)}$

We obtain slightly updated

- Level ($l$)
- Trend ($b$)
- Seasonality ($s$)

components.
**Procedure:**

**SOFIA Dynamic Update**

**Input:** \( y_t \)

**Given:** \( \{ u_{t-1}^{(n)} \}_{n=1}^{N-1}, \{ u_\tau^{(N)} \}_{\tau=t-m}^{t-1} \)

For \( t = 3m + 1, 3m + 2, \cdots \) do
- Estimate \( \mathcal{O}_t \)
- Update \( \{ u_t^{(n)} \}_{n=1}^{N-1} \)
- Update \( u_t^{(N)} \)
- Update Holt-Winters model
- Estimate missing values

Ex) 3-order tensor case
Forecasting Future Evolution of Tensor Streams

• Given any $t = t_{end} + h$, we can forecast future subtensors.

$$\hat{Y}_{t|t_{end}} = \left[ \left\{ U_{t_{end}}^{(n)} \right\}_{n=1}^{N-1}, \hat{u}_{t|t_{end}} \right]$$

Ex) 3-order tensor case

By the Holt-Winters method
Road Map

• Introduction
• Problem Definition
• Proposed Method: SOFIA
  • Tensor Factorization Model
  • Optimization Algorithm
• Experimental Results
• Conclusions
Experimental Settings

• 4 real-world time series datasets modeled by a tensor stream

- Intel Lab Sensor
- Network Traffic
- Chicago Taxi Traffic
- NYC Taxi Traffic

• A tuple \((X, Y, Z)\) denotes the experimental setting.

\[
(X, Y, Z) \quad \begin{align*}
\text{missing ratio (\%)} & \quad \text{outlier ratio (\%)} & \quad \text{outlier magnitude}
\end{align*}
\]

\[
\text{ex) (20, 10, 2) \quad \begin{align*}
20\% \text{ of entries are missing} & \\
10\% \text{ of entries are outliers} & \\
\text{outlier magnitude is } \pm 2 \cdot \max(X) & \text{ with equal probability}
\end{align*}
\]
Exp1. Initialization Accuracy

- **SOFIA\textsubscript{ALS}** accurately captured **temporal patterns** from an incomplete and noisy tensor in the initialization step.

---

**Ground Truth**

- Use vanilla ALS
- Use SOFIA\textsubscript{ALS}

**Corruption**

- Missing %: 90%
- Outlier %: 20%
- Outlier Magnitude: $\pm 7 \cdot \max(\mathcal{X})$

**Evolution of the temporal factor matrix as the initialization step proceeded.**

- **1 Iteration**
- **200 Iterations**
- **500 Iterations**
- **1000 Iterations**

**Normalized Residual Error**

- Vanilla ALS
- SOFIA\textsubscript{ALS} (Proposed)

**Number of Iterations**

$0 \rightarrow 10^0 \rightarrow 10^1 \rightarrow 10^2 \rightarrow 10^3 \rightarrow 10^4 \rightarrow 10^5$
Exp2. Imputation Accuracy

• The normalized residual error (NRE) under 4 experimental settings from the mildest (leftmost) to the harshest (rightmost).

• **SOFIA was the most accurate** in all the tensor streams.

![Graphs showing normalized residual error for different experimental settings.](image-url)
Exp2. Imputation Accuracy (cont.)

- SOFIA gave up to 76% smaller running average error (RAE) than the second-most accurate approach.

RAE: $$\frac{1}{T} \sum_{t=1}^{T} \frac{\| \hat{x}_t - x_t \|_F}{\| x_t \|_F}$$
Exp3. Speed

- SOFIA was up to $935 \times$ faster than the second-most accurate algorithm.
Exp4. Forecasting Accuracy

- **Forecasting with SOFIA was the most accurate**, despite the presence of missing values.

SOFIA can operate on various levels of missing ratios.

Assume that all entries are observed.

\[
\text{AFE: } \frac{1}{t_f} \sum_{h=1}^{t_f} \frac{\| \tilde{X}_{t+h|t} - X_{t+h} \|_F}{\|X_{t+h}\|_F}
\]

---

**Intel Lab Sensor**

**Network Traffic**

**Chicago Taxi**

**NYC Taxi**
Exp5. Scalability

- Elapsed time taken by SOFIA per time step was **almost constant** regardless of the number of subtensors processed so far.
Road Map

- Introduction
- Problem Definition
- Proposed Method: SOFIA
  - Tensor Factorization Model
  - Optimization Algorithm
- Experiment Results
- Conclusions
Conclusions

• We propose **SOFIA**, a streaming factorization algorithm for real-world tensors with missing entries and outliers.

**✓ Robust and Accurate**

**✓ Fast**

**✓ Scalable**
Robust Factorization of Real-world Tensor Streams with Patterns, Missing Values, and Outliers

Dongjin Lee
Kijung Shin