ICSI 499 Team 4: Milestone II

Michael Paglia, Proshanto Dabnath, Joseph Regan, Michael Alexander Smith

February 2024

1 Introduction

1.1 Project Statement

The goal of the project is to develop software for PySpady. This free-to-use Python library enables users to leverage novel and classical sparse encoding algorithms and methodologies to analyze spatial-temporal data. The library will contain features including missing value imputation, future value prediction, and more. The design and implementation of this software will contain the ability to determine the best combination of dictionaries/models/coding optimizers for a dataset based on some learning score.

1.2 Who will benefit

Potential models include 2D-OMP, Low-Rank Dictionary Selection, and various regularizers. Potential optimizers include ADMM, gradient-based, OMP, and more. On the front end, the input data will be preprocessed based on a .csv file. Methods found in PySpady will be applied on real-world datasets, which the team will prepare, to derive meaningful insights. This library is especially important to those who may not be involved in a computer science-related discipline and allow them to interpret their data using the algorithms above and methodologies easily.

1.3 Novelty

Our goal in this work is to obtain a low-dimensional embedding of a TGS which jointly exploits such graph and temporal properties. It is important to note that the TGS setting is fundamentally different from dynamic graph mining where the structure (sets of nodes and edges) evolves as opposed to signals over the nodes. The benefit of employing knowledge of the underlying graph structure has been demonstrated in graph signal processing, where node values are treated as a signal over the graph and the spectrum of the graph Laplacian is employed as basis for reconstructing the signal. Temporal extensions have also been recently proposed. Graph structure modeling has found valuable applications in compression/summarization of node attributes, in-network regression and optimization, missing value imputation, community detection and anomaly detection.

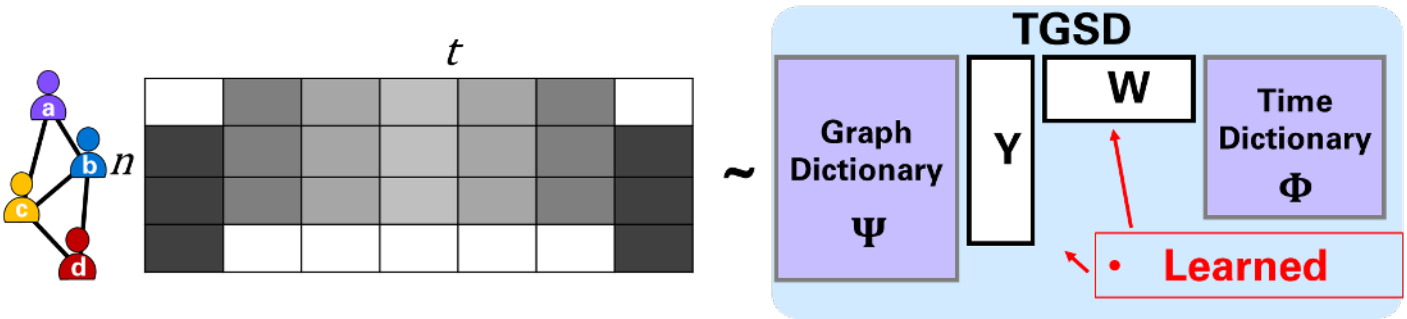

Figure 1: TGSD Diagram [5]

2 Description

2.1 Why is this an important problem?

Temporal graph signals are multivariate time series with individual components associated with nodes of a fixed graph structure. Data of this kind arises in many domains including activity of social network users, sensor network readings over time, and time course gene expression within the interaction network of a model organism. Traditional matrix decomposition methods applied to such data fall short of exploiting structural regularities encoded in the underlying graph and also in the temporal patterns of the signal.

2.2 How is this a challenging Problem?

The nature of the problem TGSD is trying to solve is non-trivial, from the size of the datasets to the complexity of the computations and algorithms. However, as we are not directly responsible for the research - it is beyond our domain of knowledge to more directly answer the question.

2.3 What are existing solutions?

Solution	**Main Utilities**	**Main Strength**	**Not Used For**	**Weakness**
MCG [3]	Missing value imputation (low-rank, regularizers)	Good missing value imputation when used with graph priors (graph)	Sparse modeling	Requires more coefficients on datasets and ignores periodicity
LRDS [7]	Decomposition; Missing value imputation (low-rank, regularizers); interpolation	Good missing value imputation when used with graph and temporal priors (graph and temporal smoothing)	Sparse modeling	Requires more coefficients on datasets
Gems-HD+ [9]	Decomposition; interpolation	Community preservation with larger models in the dataset	Imputation with many values missing, finer temporal patterns	Less sparse encoding, uses a single dictionary
BRITS [1]	Missing value imputation	Good reconstruction between low/high amounts of missing values	Finer temporal patterns	Too many parameters could cause overfitting of observed values
2D-OMP [8]	Decomposition; imputation	Good sparse reconstruction of a 2D signal as a sum of 2D atoms	Large dictionaries with a large number of coefficients	Outputs only one encoding dictionary, extremely slow with large dictionaries
CCTN [4]	Clustering	Minimize reconstruction error	Interpreting temporal patterns	High number of embedding dimensions
PCA [2]	Clustering	Speed	Rich graph and temporal structures	Too general

Table 1: Existing Solutions

Most existing methodology focuses on modeling either the graph structure or the structure of the temporal signal . The interplay between temporal and structural graph properties gives rise to important behaviors, however, no frameworks currently exist to facilitate a joint representation. For example, traffic levels in a transportation network are shaped by both network locality and the time of the day.

3 User Classes and Functional Requirements

3.1 User Classes

Our users will include anyone in need of data analysis, particularly those requiring matrix decomposition, imputation of missing values, and related techniques. This could encompass scientists without a strong background in computer science, computer scientists, and computer hobbyists.

3.2 Functional Requirements

3.2.1 Data Management System

The software will implement a data management system that allows the user to input data to run TGSD (Temporal Graph Signal Decomposition). Specifically, the end-user will be able to input their datasets as well as save any outputs.

3.2.2 Graphical User Interface (GUI)

The software will implement a GUI accessible either through a web interface or a local interface, providing users with intuitive interaction and visualization capabilities.

3.2.3 TGSD

The software will implement TGSD; a highly scalable decomposition algorithm for complete and incomplete data. The algorithm has advantages in matrix decomposition, imputation of missing values, temporal interpolation, clustering, period estimation, and rank estimation [6].

3.2.4 Auto-configuration

The software will implement an auto-configuration tool that will have default parameters, an estimated time to complete, and return explanatory figures to the user. Dictionaries include Graph/Discrete Fourier Transform, Ramanujan, etc.

3.2.5 Outlier Detection

The software will use TGSD to implement outlier detection. This functionality allows users to detect and identify anomalies within their datasets. Outlier detection plays a crucial role in various domains, including anomaly detection, fraud detection, and quality control, making it a valuable addition to the software's feature set.

4 Other Requirements

4.1 Performance

The library's performance should match current data analysis models, though it's heavily influenced by input data. PySpady's TGSD for time series graph decomposition primarily learns parameters 'Y' and 'W' through linear algebraic computations, with matrix multiplication being the most resource-intensive operation. As matrix sizes grow, computation time increases.

4.2 Scalability

PySpady will be designed to handle large datasets efficiently, allowing for seamless integration of more efficient algorithms in the future. While the computations for TGSD require 2D graphs, error checking will ensure data dimensionality. Users are responsible for verifying graph dimensions. The library should handle graphs of any size computationally, but data complexity may impact performance. Enhanced hardware may be necessary for processing larger datasets.

4.3 Usability

The goal is to create a user-friendly end product for those without Python coding skills who need to utilize TGSD for data analysis. To achieve this, we'll offer a GUI allowing users to input data and parameters easily. Advanced users can utilize a comprehensive config file for more flexibility. For those experienced in coding, the library itself will be available, offering the highest level of interaction.

4.4 Availability

At a minimum, PySpady will require a dataset to perform the computations for TGSD. Other parameters can be defined for computations more custom to the user. The system performing the computations should have hardware with enough computational power to perform the analysis for the data provided.

4.5 Reliability

PySpady needs to reliably compute the computations of TGSD on the data. There will be error handling involved to ensure critical errors are not made, however, this library is designed for a user that knows the data analysis they are trying to perform so some amount of responsibility will be placed on the user to ensure errors do not result in severe consequences.

4.6 Maintainability

The library will be well-documented should advanced users wish to analyze the code behind the functions. PySpady relies on other Python libraries to perform the computations so as these dependencies are updated, PySpady will be updated accordingly.

4.7 Portability

The only requirement to run this library will be Python, so the library will be system agnostic. The library is being developed in Python 3 and users will be recommended to run on Python 3 to ensure no issues with the library.

4.8 Security

Security should not be an issue as the data being provided, the computations on the data, and the results of the computation will all be done and stored on the system of the user. The library should not introduce any vulnerabilities into the user's system.

4.9 Operating Requirements

The operating requirements of this library will scale by the nature of the data being analyzed. At minimum, only a computer that can run Python will be required. As the size of the data increases, the time to perform the decomposition will also increase, requiring better hardware to perform the computations promptly.

4.10 Design Constraints

The PySpady library will be written purely in Python. Any functions and features of the library will be designed to fit within the Python ecosystem. Extraneous files that will be needed for the library will also need to be processed to ensure the various Python libraries being used can comprehend them.

4.11 Implementation Constraints

The entire PySpady library will be developed in Python. Python has numerous libraries available for data conversion, computations, and data visuals. Additionally, Python has available GUI libraries. Any features we wish to implement as part of the PySpady library will be limited by what is available within the Python ecosystem.

4.12 Skillset Constraints

The team will require knowledge of Python to develop the library. Team members involved in developing a GUI will need knowledge of developing GUIs within Python. Team members involved in additional features related to the data analysis, will require knowledge of linear algebra and data mining principles.

References

[1] W. Cao et al. "Brits: Bidirectional recurrent imputation for time series". In: *Advances in Neural Information Processing Systems 31*. 2018, pp. 6775–6785.

[2] I. T. Jolliffe and J. Cadima. "Principal component analysis: a review and recent developments". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 374.2065 (2016), p. 20150202.

[3] V. Kalofolias et al. "Matrix completion on graphs". In: *arXiv preprint arXiv:1408.1717* (2014).

[4] Y. Liu et al. "Coupled clustering of time series and networks". In: *2019 SIAM ICDM*. SIAM. 2019, pp. 531–539.

[5] Maxwell McNeil. *Sparse Representation learning for temporal networks*. Accessed: 2024-02-11. 2023.

[6] Maxwell McNeil, Lin Zhang, and Petko Bogdanov. "Temporal Graph Signal Decomposition". In: *ACM International Conference on Knowledge Discovery and Data Mining*. Online, 2021.

[7] K. Qiu et al. "Time-varying graph signal reconstruction". In: *IEEE J. of Selected Topics in Signal Processing* 11.6 (2017), pp. 870–883.

[8] Fang Y., B. Huang, and J. Wu. "2D Sparse Signal Recovery via 2D Orthogonal Matching Pursuit". In: ().

[9] Y. Yankelevsky and M. Elad. "Finding gems: Multi-scale dictionaries for high-dimensional graph signals". In: *IEEE Transactions on Signal Processing* 67.7 (2019), pp. 1889–1901.