Looking at Big Data through Graph Signal Processing

Yashswi Jain, Krishna Subramani

Electrical Engineering
Indian Institute of Technology Bombay, India

EE771 - Recent Topics in Analytical Signal Processing

Motivation

- Analysis and processing of very large datasets, or 'Big Data' poses a significant challenge.
- If Knowledge is power and Data is knowledge then Data is power! Need an efficient way to channelise this power.
- Given the enormous volume of the data, analysis is often computationally demanding.
- This computational bottleneck can be efficiently dealt with by finding and exploiting some 'inherent structure' in the data.
- Graphs are an efficient way of representing 'Distributed' data. Hence, our motivation.

The Problem

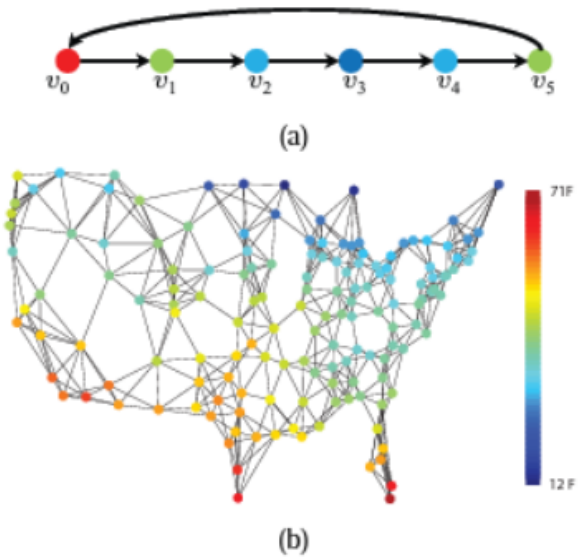
- Consider a weather dataset comprising of daily temperature readings for 400 days from 30 weather stations in Brazil. We intend to compress the given data.
- For GSP, the given data needs to modelled as a graph. A graph representing the above data will ideally comprise of 12000 nodes.
- This would mean a 12000×12000 adjacency matrix. Any matrix manipulation involving such a huge matrix is a computational nightmare.

- Spectral Decomposition is needed to obtain the Graph Fourier transform(GFT), from which compression can be achieved by retaining the dominant spectral components.
- This involves finding the eigen values and augmented eigen vector matrix of the Laplacian.
- Because of the huge size of the adjacency matrix, these are computationally expensive as we shall see later!
- Where is the light at the end of this tunnel?

- Spectral Decomposition is needed to obtain the Graph Fourier transform(GFT), from which compression can be achieved by retaining the dominant spectral components.
- This involves finding the eigen values and augmented eigen vector matrix of the Laplacian.
- Because of the huge size of the adjacency matrix, these are computationally expensive as we shall see later!
- Where is the light at the end of this tunnel?

The Proposed Solution

- In order to make the problem computationally efficient, we factor the graph into two smaller components,
  1. A graph depicting the sensor grid comprising of 30 nodes.
  2. A directed time series comprising of 400 nodes.
- The two graphs are shown below.



(a) Time series graph, (b) Sensor network graph, [1]

Challenges

The paper [1] is our base reference. It lays the foundation of data compression using discrete GSP.

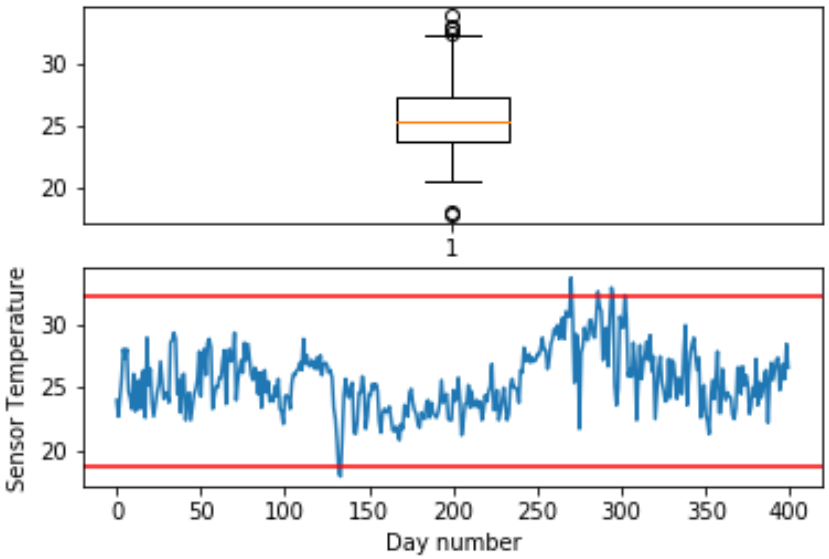
We believe that the paper has left the following crucial aspects unanswered:

- The most crucial step of any big data analysis is data pre-processing. The compression can be severely affected in presence of outliers and redundancies since all such points distort the spectrum.
- The paper says nothing about handling outliers, redundancies and missing data points.

- The paper does not talk about how the weight matrices of the factor graphs are generated. Moreover, the time series is directed as well, which adds to the complexity.
- For a directed graph, the Laplacian will not be symmetric. Therefore, Inverse GFT will now involve obtaining the inverse of the augmented eigen vector matrix instead of simple transposition.
- The results obtained in the literature could not be reproduced due to the lack of a definitive algorithm for compression and unavailability of the dataset that has been actually used in [1] as a benchmark to validate our performance.

Solution Sketch

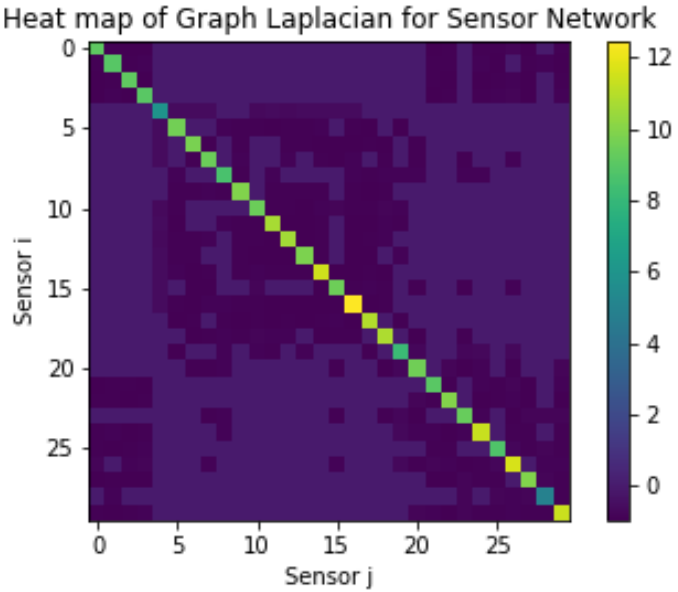
- *Obtaining the Dataset*: The weather dataset has been obtained from Kaggle[1]. The dataset obtained wasn't sanitised.
- *Outlier removal*: We have obtained the box plots to identify outliers and clip them.



Box plot and corresponding sensor data

[1] Dataset Link

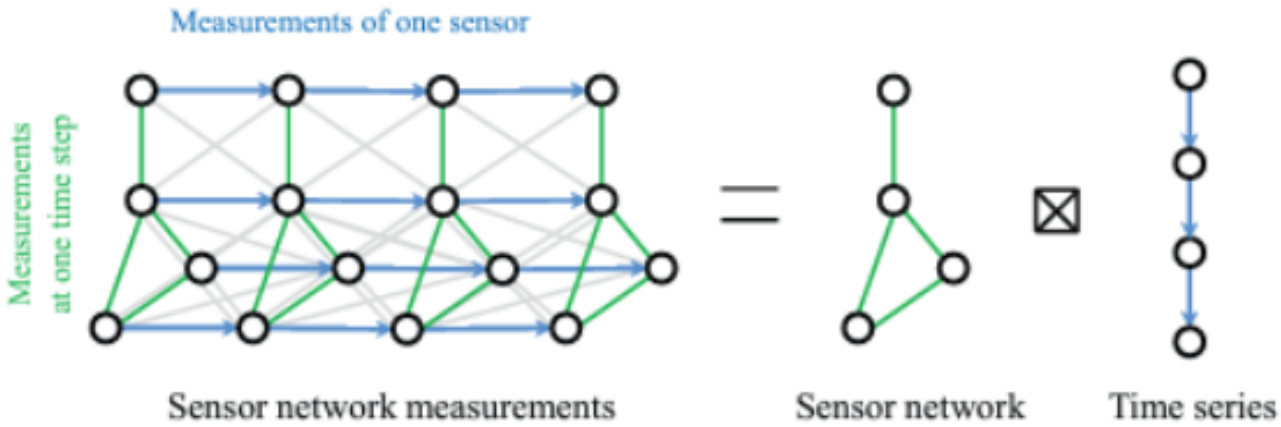
- *Data Imputation*: The missing values has been imputed with the neighbouring values in the time series for a given station.
- *Obtaining the weight matrices*: The weight for the directed time series has been taken to be 1 for the connected nodes, and for the sensor graph, the weights are given by the formula $W_{i,j} = Ke^{-\frac{|d_i - d_j|}{d_{max}}}$



Heat map of Laplacian Matrix

Product Graphs

- We can factorize our graph as a 'strong product' of two smaller graphs, a sensor network graph and a time series graph[2] as shown below,
- Let $L_1 \in R^{N_1 \times N_1}$ be the graph Laplacian for the sensor network where $N_1 = 30$, and let $L_2 \in R^{N_2 \times N_2}$ be the graph Laplacian for the time series network where $N_2 = 400$.



Strong Product of graphs, [1]

Formulating the product graph and its components

- The Laplacian of the product graph can be written as:

$$L_{\oplus} = L_1 \oplus L_2 = L_1 \otimes L_2 + L_1 \otimes I_{N_2} + L_2 \otimes I_{N_1}, \qquad (1)$$

where $\oplus$ is the Strong Kronecker product, and $\otimes$ is the Matrix Kronecker product.

- The spectral decomposition of the product graph can be obtained in the following way:

$$L_{\oplus} = V \times \left(\Lambda_1 \otimes \Lambda_2 + \Lambda_1 \otimes I_{N_2} + I_{N_1} \otimes \Lambda_2\right) \times V^{-1},$$

where Λ_1, Λ_2 are the individual graph laplacian eigenvalue matrices, and $V = V_1 \otimes V_2$ is the Kronecker product of the individual graph Laplacian augmented eigenvector matrices.

- Hence the inverse of the augmented eigen vector matrix for the product graph can be obtained as follow:

$$V^{-1} = (V_1 \otimes V_2)^{-1} = V_1^{-1} \otimes V_2^{-1},$$

Note: Instead of taking the inverse of a 12000×12000 matrix, the inverse operation is reduced to taking the inverse of a 400×400 matrix and a 30×30 matrix; this will significantly reduce the computation period as we shall see later.

- Obtaining the GFT $\hat{f}$ of the graph signal $f \in R^{12000}$ now reduces to:

$$\hat{f} = (V_1^{-1} \otimes V_2^{-1}).f,$$

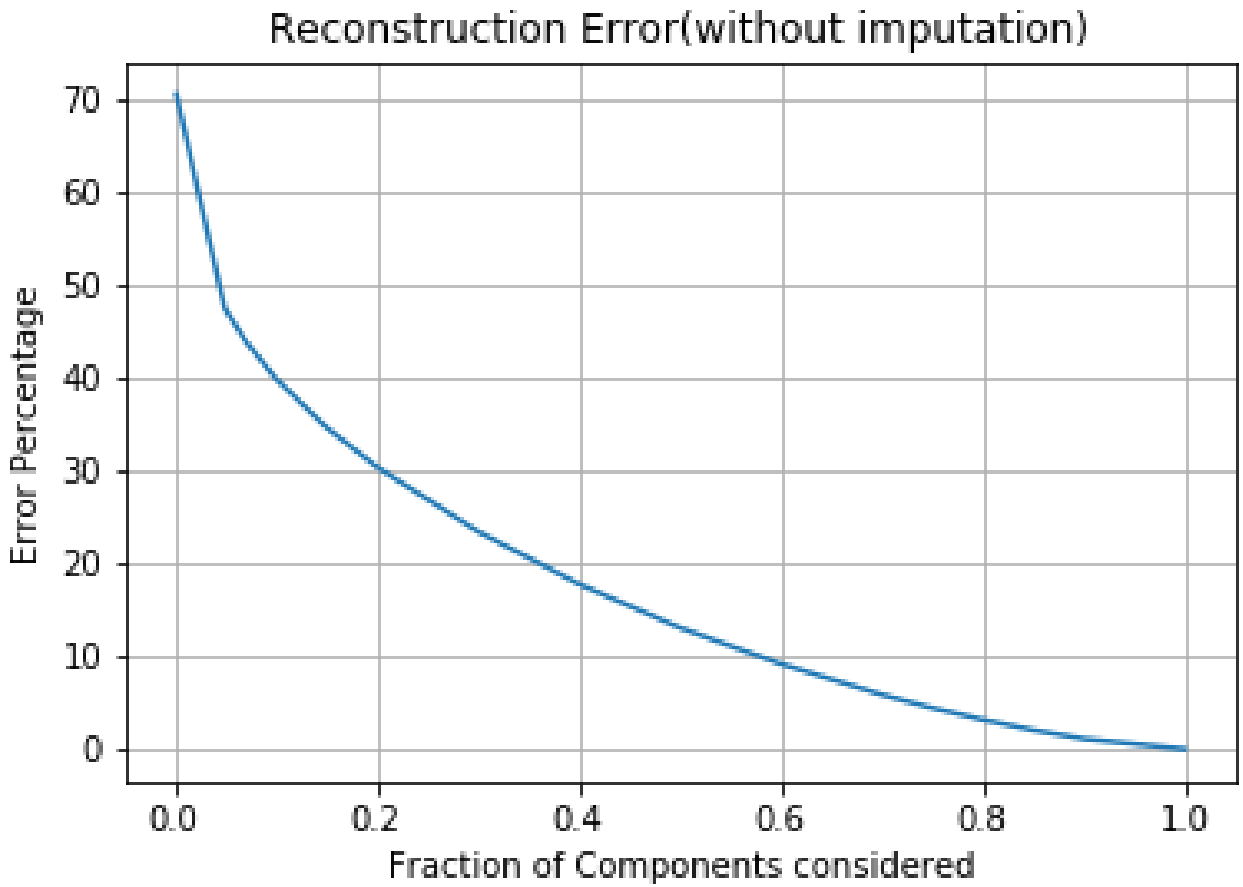
- The last step towards compression is throwing away the GFT components whose absolute values are lesser than a certain threshold and take the inverse GFT. The inverse can be taken as shown:

$$\tilde{f} = (V_1 \otimes V_2).\tilde{\hat{f}},$$

where $\tilde{\hat{f}}$ is the truncated GFT of the given graph signal and $\tilde{f}$ is the reconstructed graph signal.

Observations and Inferences

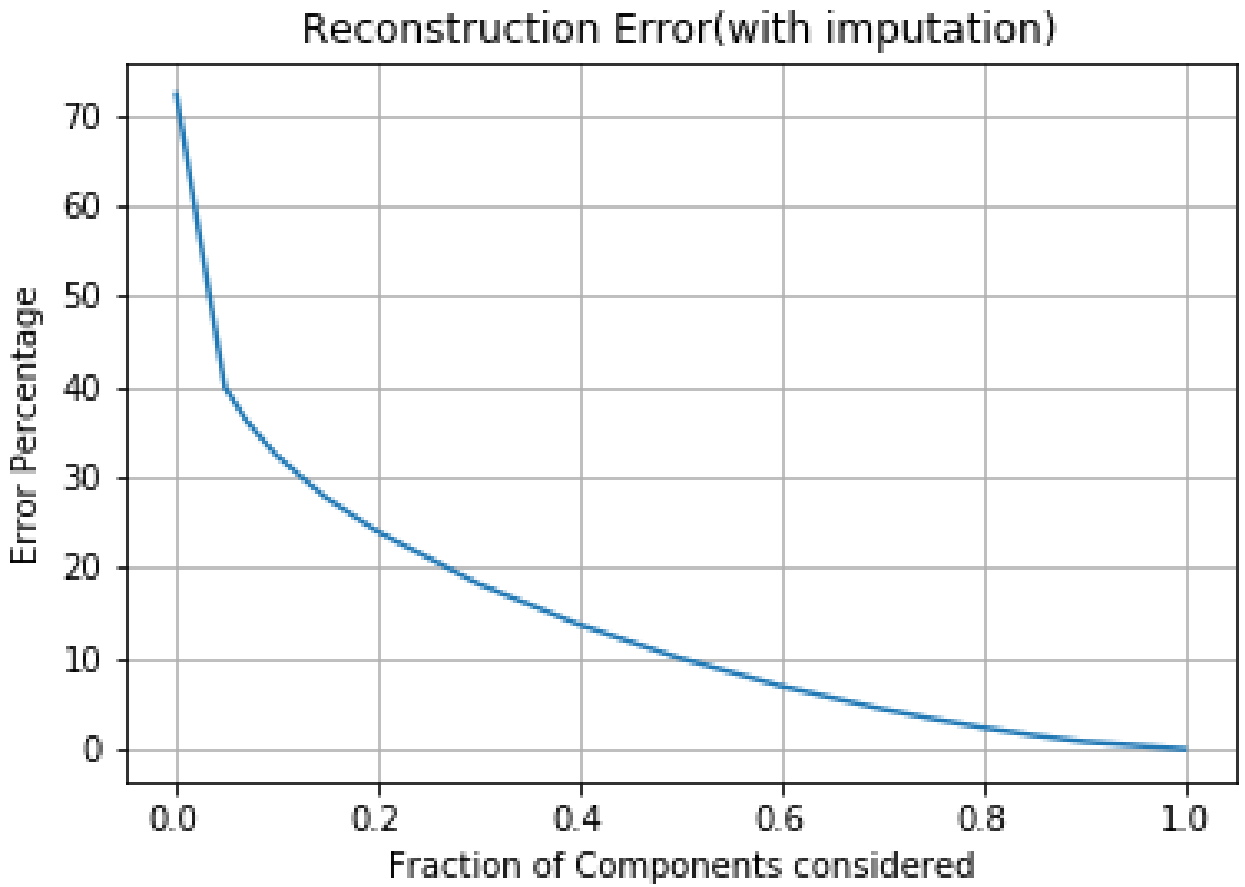
- The most interesting thing we observed was the improvement in the reconstruction error by performing data imputation.



Mean Squared Error without data imputation

Observations and Inferences

- The most interesting thing we observed was the improvement in the reconstruction error by performing data imputation.



Mean Squared Error with data imputation

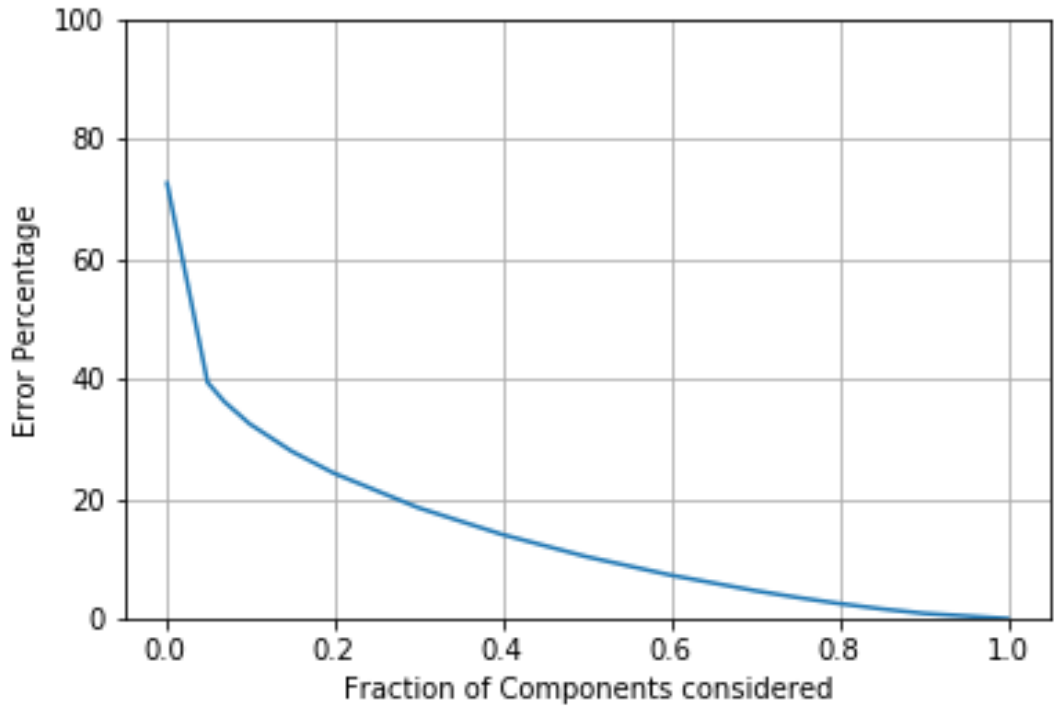
- The improvement of the MSE performance with data imputation is $\approx 5\%$ for 20% of data compression i.e. 80% is being used for data reconstruction.
- This improvement is indeed expected since missing values act as noise in the given data therefore leading to spectrum distortion.
- In the absence of suitable imputation technique, this missing data will be ignored leading to increased MSE.
- We are achieving an MSE of $\approx 10\%$ when 45% of the data is used for reconstruction i.e. achieving a compression of 55%.
- With 80% of the data being discarded the MSE is $\approx 25\%$.

- Another thing that we observed was the drastic reduction in time for the algorithm to run.
- Recall $V^{-1} = V_1^{-1} \otimes V_2^{-1}$,

Inverse of Factors	**Inverse of Product**
≈ 0.09 minutes	≈ 0.75 minutes

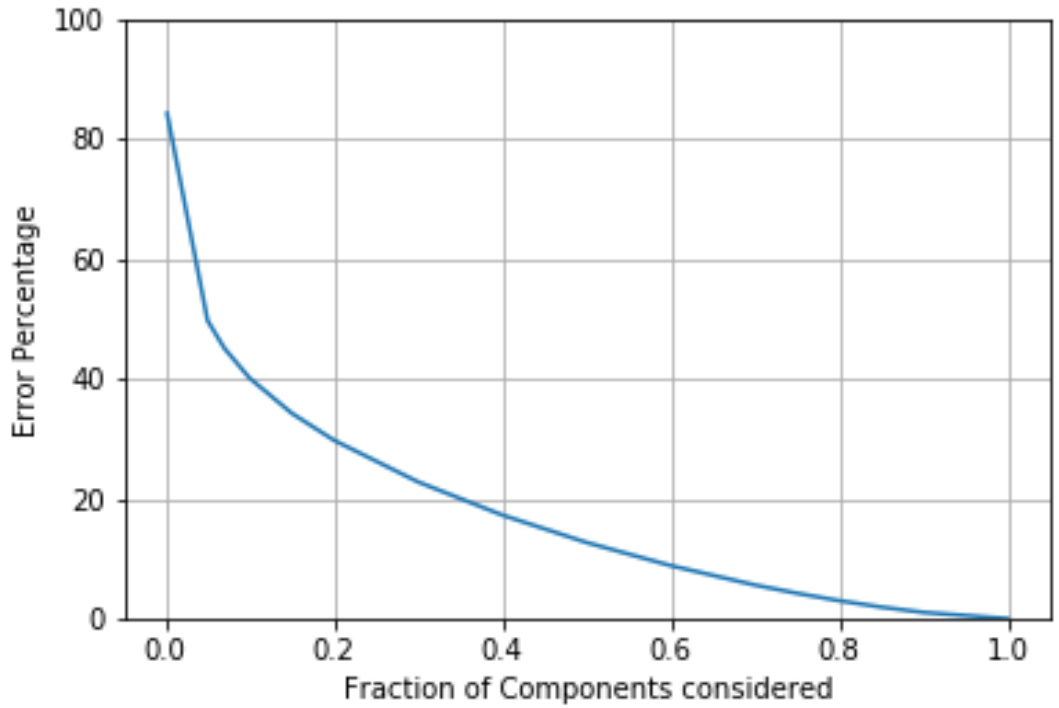
- This gain of ≈ 8 in runtime essentially proves the point that we are trying to make. This gain will be even more significant for larger sizes.

- Contrary to our expectations, the outlier removal technique does not improve the performance. The MSE increases overall. We are not able to explain why this happens.



Mean Squared Error without data clipping

- Contrary to our expectations, the outlier removal technique does not improve the performance. The MSE increases overall. We are not able to explain why this happens.



Mean Squared Error with data clipping

Obtaining the Product Graphs

- In the problem discussed so far, we assumed implicitly that the overall graph can be written as the strong product.
- Recall Eq.1,

$$L_{\oplus} = L_1 \oplus L_2 = L_1 \otimes L_2 + L_1 \otimes I_{N_2} + L_2 \otimes I_{N_1}.$$

- What do we do if we instead have access to the larger Laplacian $L_{\oplus}$, and we want to 'factorize' it to obtain the product graphs L_1,L_2?
- Using Linear Algebra and simple matrix manipulations, we show that the problem of estimating the products ultimately boils down to finding a lower rank representation of the larger matrix.

- We want to find the matrix which solves the following optimization problem,

$$\min_{B,C} \|A - B \oplus C\|_F \,, \tag{2}$$

where A $\sim L_{\oplus}$ i.e Laplacian of the product graph and B $\sim L_1$,C $\sim L_2$ are the factored laplacians, $\oplus$ is the Strong Kronecker product, and F is the Frobenius norm.

- Let us consider a simpler problem,

$$\min_{B,C} \|A - B \otimes C\|_F \,, \tag{3}$$

where $\otimes$ is the Matrix Kronecker product.

- We will show the solution to the above for small matrices A,B where $A \in R^{4\times4}$ and $B, C \in R^{2\times2}$

$$\|A - B \otimes C\|_F = \left\| \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} - \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} \otimes \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} \right\|_F$$

$$= \left\| \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} - \begin{bmatrix} b_{11}c_{11} & b_{11}c_{12} & b_{12}c_{11} & b_{12}c_{12} \\ b_{11}c_{21} & b_{11}c_{22} & b_{12}c_{21} & b_{12}c_{22} \\ b_{21}c_{11} & b_{21}c_{12} & b_{22}c_{11} & b_{22}c_{12} \\ b_{21}c_{21} & b_{21}c_{22} & b_{22}c_{21} & b_{22}c_{22} \end{bmatrix} \right\|_F$$

$$= \left\| \begin{bmatrix} a_{11} & a_{12} & a_{21} & a_{22} \\ a_{13} & a_{14} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{41} & a_{42} \\ a_{33} & a_{34} & a_{43} & a_{44} \end{bmatrix} - \begin{bmatrix} b_{11}c_{11} & b_{11}c_{12} & b_{11}c_{21} & b_{11}c_{22} \\ b_{12}c_{11} & b_{12}c_{12} & b_{12}c_{21} & b_{12}c_{22} \\ b_{21}c_{11} & b_{21}c_{12} & b_{21}c_{21} & b_{21}c_{22} \\ b_{22}c_{11} & b_{22}c_{12} & b_{22}c_{21} & b_{22}c_{22} \end{bmatrix} \right\|_F$$

$$= \left\| \begin{bmatrix} a_{11} & a_{12} & a_{21} & a_{22} \\ a_{13} & a_{14} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{41} & a_{42} \\ a_{33} & a_{34} & a_{43} & a_{44} \end{bmatrix} - \begin{bmatrix} b_{11} \\ b_{12} \\ b_{21} \\ b_{22} \end{bmatrix} \begin{bmatrix} c_{11} & c_{12} & c_{21} & c_{22} \end{bmatrix} \right\|_F$$

$$= \left\| \tilde{A} - \tilde{b}\tilde{c}^T \right\|_F$$

- We have thus reduced $\min_{B,C} \|A - B \oplus C\|_F$ to the problem,

$$\min_{\tilde{b},\tilde{c}} \left\| \tilde{A} - \tilde{b}\tilde{c}^T \right\|_F,$$

which is the optimal rank-1 estimate of the matrix $\tilde{A}$ which can be obtained using the SVD of $\tilde{A}$.

- The Kronecker strong product in Eq. 1 however involves three terms. Using the manipulations on the previous page, we can recast the optimization problem $\min_{B,C} \|A - B \oplus C\|_F$ as,

$$\min_{u_1,v_1,u_2,v_2,u_3,v_3} \left\| \tilde{A} - u_1 v_1^T - u_2 v_2^T - u_3 v_3^T \right\|_F, \qquad (4)$$

  which is like finding the best rank-3 estimate of $\tilde{A}$.
- A major computational issue with our analysis is that finding the factors again involves the computation of the SVD of the large matrix. However, the problem can be simplified if we assume a structural property like sparsity about the matrix.
- Our analysis here is inspired from [3]. A detailed analysis has been provided in [4].

References I

[1] A. Sandryhaila and J. M. Moura, "Big data analysis with signal processing on graphs," *IEEE Signal Processing Magazine*, vol. 31, no. 5, pp. 80–90, 2014.

[2] R. Hammack, W. Imrich, and S. Klavžar, *Handbook of product graphs*. CRC press, 2011.

[3] C. Van Loan, "The kronecker product svd," 2009.

[4] C. F. Van Loan and N. Pitsianis, "Approximation with kronecker products," in *Linear algebra for large scale and real-time applications*, pp. 293–314, Springer, 1993.