Looking at Big Data through Graph Signal Processing

Krishna Subramani, Yashswi Jain
Department of Electrical Engineering
Indian Institute of Technology Bombay, India
Email: {krishna.subramani,150110039}@iitb.ac.in

***Abstract*—Analysis and processing of very large datasets, or 'Big Data' poses a significant challenge. 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 to cast the problem of Big Data to the framework of Graph Signal Processing which can allow us to tackle the problem more efficiently.**

I. Introduction

Consider the problem of data compression. If the dimensionality of the data-set is small enough, techniques like Principal Component Analysis(PCA) can be used. However, in certain cases like the one of time series data which captures daily temperature readings through a network of weather stations distributed across a country, the dimensionalilty blows up and hence it can not be handled efficiently by a technique like PCA.
In a nutshell, consider the problem of compressing temperature data from weather stations. For huge datasets like this compression through graph signal procesing is more efficient. But, finding the Graph Fourier Transform(GFT) involves finding the eigen-decomposition of huge matrices which is a computational nightmare. To address this issue, we will first introduce what are called product graphs. Then, we shall discuss in detail the stepis involved in compressing a dataset using this idea.
Most of our work is inspired from Sandryhaila et al. [1]. However, 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 as a benchmark to validate our performance.

II. Product Graphs

We can factorize our graph as a 'strong product' of two smaller graphs, a sensor network graph and a time series graph (refer [2] for the technical details) as shown in Fig. 1.

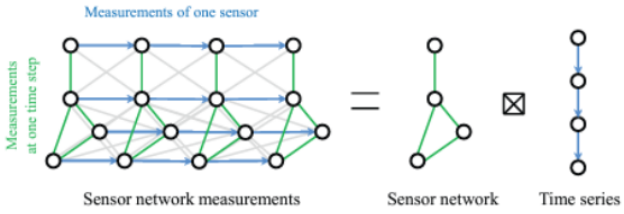

Fig. 1. Strong Product of graphs, [1]

Let $L_1 \in R^{N_1 \times N_1}$ be the graph Laplacian for the sensor network and let $L_2 \in R^{N_2 \times N_2}$ be the graph Laplacian for the time series network.The Laplacian of the product graph can be written as:

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

where $\oplus$ is the Strong Kronecker product, and $\otimes$ is the Matrix Kronecker product. Recall that the Kronecker product of matrices $A = [a_{mn}] \in R^{M \times N}$ and $B = [b_{mn}] \in R^{K \times L}$ is,

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \dots & a_{1N}B \\ a_{21}B & a_{22}B & \dots & a_{2N}B \\ \vdots & \vdots & \vdots & \vdots \\ a_{M1}B & \dots & \dots & a_{MN}B \end{bmatrix}$$

and belongs to $R^{MK \times NL}$.
The spectral decomposition of the product graph can be obtained in the following way,

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

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. Thus, we can write the inverse of the augmented eigen vector matrix for the product graph can be obtained as follows,

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

The important thing to observe is instead of taking the inverse of a $N_1N_2 \times N_1N_2$ matrix, the inverse operation is reduced to taking the inverse of a $N_1 \times N_1$ matrix and a $N_2 \times N_2$ matrix; this will significantly reduce the computation time.

III. Proposed System

Due to the lack of a definitive algorithm and dataset, we make a few assumptions along the way and use a publicly available dataset as well. We shall highlight all of these along the way.

A. Dataset

The weather dataset has been obtained from Kaggle [1]. It consists of daily temperature reading from $N_1 = 30$ station distributed throughout Brazil for $N_2 = 400$ days. Since there were missing values in the data-set, we perform data imputation by assigning the neighbouring values in the time series for a given station. We also observed a few outliers in the data, hence we obtain box plots to identify outliers and clip them as shown in Fig. 2.

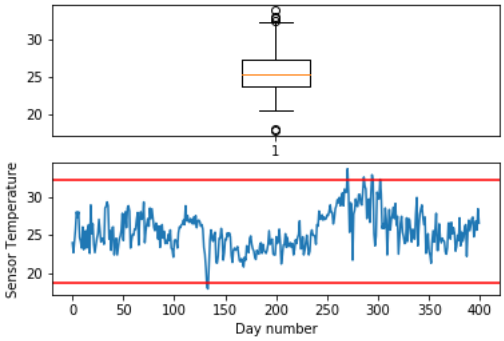

Fig. 2. Box plot and corresponding sensor data

B. Setting up the Graph

We factor the bigger graph which represents the time series points for all the time instants across graph into two smaller components,

1) A graph depicting the sensor grid comprising of $N_1 = 30$ nodes.
2) A directed time series comprising of $N_2 = 400$ nodes.

The two graphs are shown in Fig. 3
To obtain the weights of the graphs, we assume the time series is directed as given in Fig. 3 (a), and we use four different methods to find the weights for the sensor matrix,

1) Unnormalized Distance Matrix calculated from the latitude and longitude of each station $W = D$.
2) Normalized Distance matrix $W = D_{normalized} = \frac{D}{D_{maxval}}$.

[1] https://www.kaggle.com/guispadaccia/brazil-weather-data-from-2010-to-2017

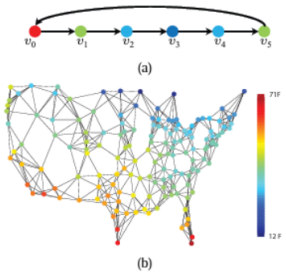

Fig. 3. (a) Time series graph, (b) Sensor network graph, [1]

3) Exponential function on the distances $W_{i,j} = Ke^{-\frac{|d_i - d_j|}{d_{max}}}$.
4) Distance matrix using the altitudes instead of the latitude and longitude.
5) Nearest Neighbour weight matrix i.e. keeping the nearest N neighbours for each node. We use $N = 10$.

C. Performing the Compression

We first define the $N_1 \times N_2$ graph signal as f, the time series signal at all the N_1 stations for all the N_2 time instants. We unroll this into a single vector to obtain the 'effective' graph signal for the overall graph. We then obtain its GFT the following way,

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

Where $(V_1^{-1} \otimes V_2^{-1}) = V^{-1}$ is the transformation matrix, and f is the graph signal. We then throw away the GFT components whose absolute values are lesser than a certain threshold and take the inverse GFT. The inverse can be found as,

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

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

IV. Observations

The quality of reconstruction is analyzed using the normalized mean squared error defined as,

$$e_p = \frac{\left\| f - \tilde{f} \right\|}{\|f\|}$$

We shall refer to e_p as the reconstruction error henceforth.
We obtained the plot of reconstruction error vs fractions retained for different cases i.e.

- Fully-connected graph with weights as exponential distance without and with data imputation as shown in Fig. 4 and Fig. 5 respectively.
  Observation: Performing data imputation gave us the maximum improvement in the reconstruction error among all the permutations that we tried.
- Nearest Neighbor graph with N=10 and the weight matrix being constructed from the difference in the

altitude instead of exponential distance as considered in the earlier case.
Observation: Marginal improvement in the reconstruction error as compared to the case when the weight matrix is constructed for a fully connected graph from the exponential distance.

- Reconstruction error with and without clipping the outliers as shown in Fig. 7 and Fig. 6 respectively. *Observation: Contrary to our expectations however, the outlier removal technique does not improve the performance. The MSE increases overall. We are not able to explain why this happens.*

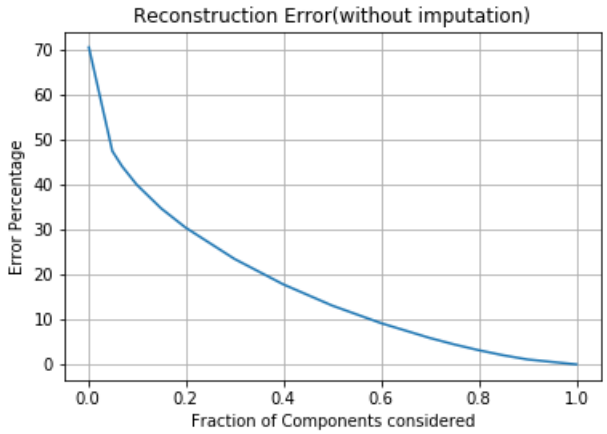

Fig. 4. Mean Squared Error without data imputation

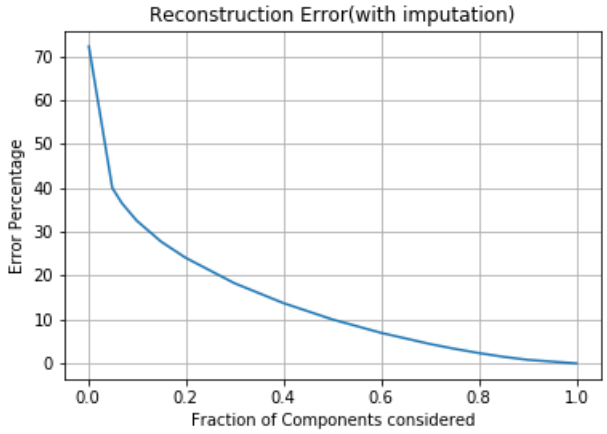

Fig. 5. Mean Squared Error with data imputation

Another thing that improves is the runtime(as expected). The improvement is ≈ 8 times better.

Inverse of Factors	Inverse of Product
$\approx$ 0.09 minutes	$\approx$ 0.75 minutes

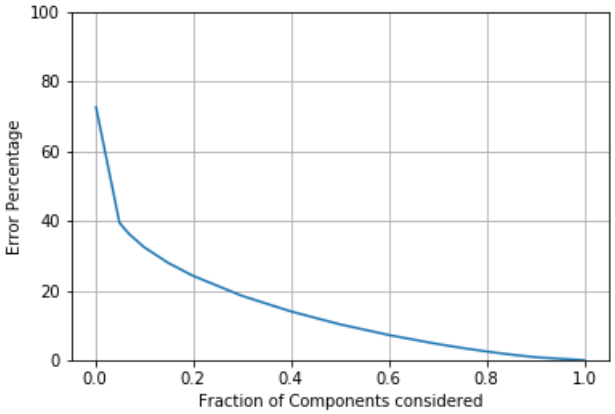

Fig. 6. Mean Squared Error without data clipping

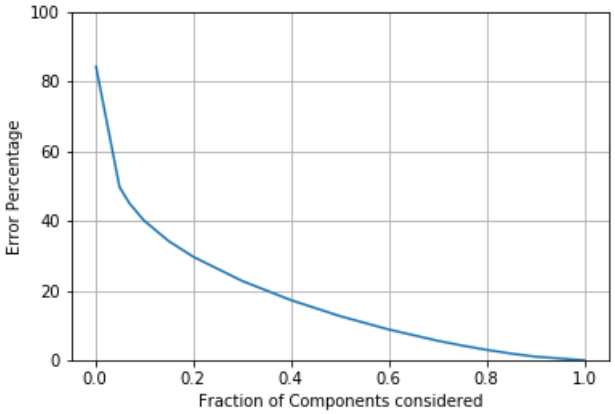

Fig. 7. Mean Squared Error with data clipping

V. Obtaining the Product Graphs for General Laplacians

In the problem discussed so far, we assumed implicitly that the overall graph can be factorized as the strong product. However, consider Eq. 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 Eq. 3 for small matrices A,B where $A \in R^{4\times4}$ and $B, C \in R^{2\times2}$.

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

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

VI. Conclusion and Future Work

Through this discussion, we have demonstrated that for certain graphs, we can perform compression much more efficiently and faster by decomposing it into its product graphs. However, an interesting future application would be to learn the Laplacian from the data itself(as it is done in [5]), and then perform the factorization of this Laplacian. Further, performance of the compression can be improved by using a regularization technique like Tikhonov regularization to remove the outliers from the data and instead of just imputing the missing values with the nearest neighbors something like moving averages or a regression model can be used to predict the missing values and filling them in the missing places.

REFERENCES

[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.

[5] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst, "Learning laplacian matrix in smooth graph signal representations," *IEEE Transactions on Signal Processing*, vol. 64, no. 23, pp. 6160–6173, 2016.