Homework 2: Data driven techniques for building a graph in IoT networks.

Jose M. Barcelo Ordinas, Jorge Garcia Vidal, Pau Ferrer Cid

Universidad Politècnica de Catalunya (UPC-BarcelonaTECH),
Computer Architecture Dept.
jose.maria.barcelo@upc.edu

April 8, 2024

1 Theory

1.1 Introduction

This motivation is taken from our paper [1]. The goal of this paper [1] is to present some well-known and novel graphical tools for finding a good representation for sensor networks using structured data for IoT networks. There are several ways of defining the graph that represents the network topology.

The classic method proposed in the literature for this kind of sensor networks is to create the edges that join the nodes of the graph following a function that depends on the distance between the nodes. This method has the disadvantage of not taking into account the similarity of the measurements taken by the sensors. Thus, one of the challenges and key points for the representation of a sensor network using graphs is to estimate the structure of the graph that underlies the measured data. There are several ways to build the graph from the data depending on whether you use statistical methods based on graphical probabilistic models such as Markov random fields (MRFs) such as Graphical Lasso (GLASSO), or methods based on graph signal processing (GSP) that obtain the Laplacian matrix to represent the graph topology.

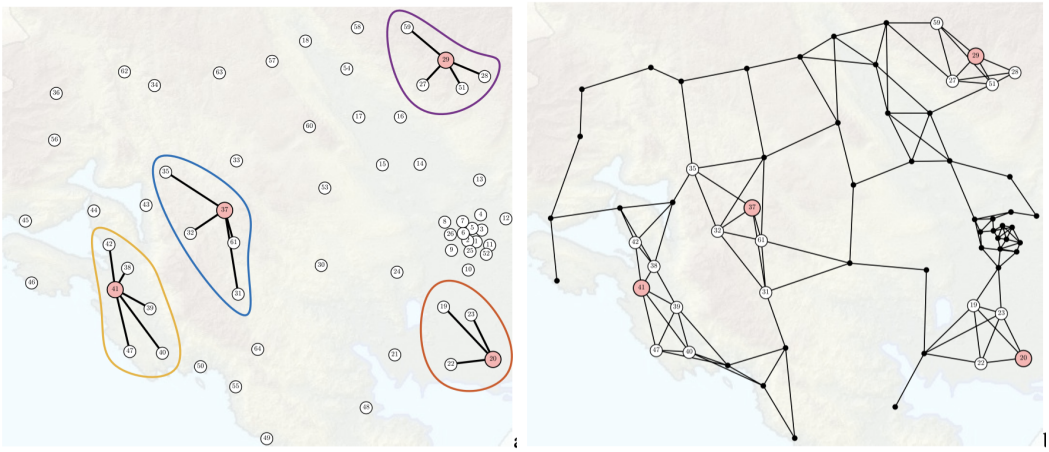
In this homework, we are going to explore 3 different ways of building the graph in a small sensor network of 8 nodes: i) a distance-based model, ii) a GLASSO method, and iii) a methods based on GSP in which we will obtain the Laplacian matrix.

1.2 Motivation

The framework of graph signal processing (GSP) was conceived in the last decade with the ambition of generalizing the tools from classical digital signal processing to the case in which the signal is defined over an irregular structure modeled by a graph.

Let us take the example (taken from Ljubisa Stankovic *et al.* survey paper, "Understanding the basis of graph signal processing via an intuitive example-driven approach", arXiv, May 2019) where a set of

temperature sensors are deployed over a large region. We are interested, Figure 1.a), of finding the local neighborhood of nodes, in such a way that we will find a graph that represents the network, Figure 1.b).



(a) Local neighborhood of single nodes.

(b) Local neighborhood for all sensors.

Figure 1: Multisensor IoT example, where nodes measure temperature (Figure taken from L. Stankovic et al "Understanding the Basis of Graph Signal Processing via an Intuitive Example-Driven Approach" paper). Graph representation.

We have been able to connect (we will see later how), for example, node 20 with nodes 19, 22, 23, and node 29 with nodes 27, 28, 51, 59, and so on.

If $x(n)$ is the temperature value of node n, then we can now draw the temperature using bars, Figure 2.a) or even with color in the vertices, Figure 2.b).

The objective is that now we can consider that the signal in a given node n is related to the node itself and its neighborhood:

$$y(n) = x(n) + \sum_{m \in N(n)} x(n) \tag{1}$$

with N(n) the neighborhood of node n. For example, for node n=20:

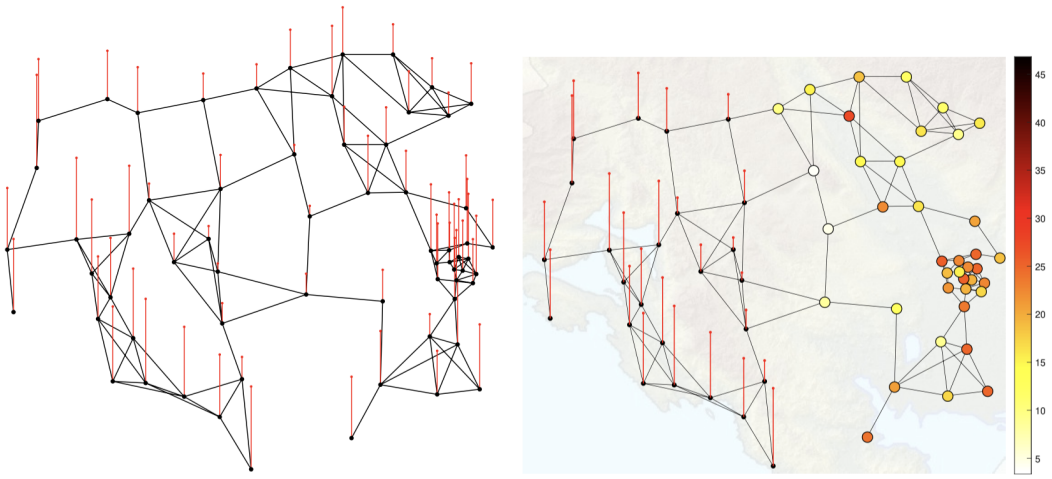
$$y(20) = x(20) + x(19) + x(22) + x(23) \tag{2}$$

For convenience, we will write this expression as:

$$\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{A}\boldsymbol{x} \tag{3}$$

where matrix $\boldsymbol{A}$ is the adjacency matrix. There are several ways of creating a graph. The key is in discovering which are the relationships among the nodes.

Example 1.1 (Subgraph of node 29)**.** *The subgraph represented by node* 29 *with neighbors 27, 28, 51,*



(a) Bars indicate temperature values.

(b) Temperature values as bars (left) and vertex color (right)

Figure 2: Multisensor IoT example, where nodes measure temperature (Figure taken from L. Stankovic et al "Understanding the Basis of Graph Signal Processing via an Intuitive Example-Driven Approach" paper). Colour representation of the temperature field.

59 in Figure 2.b) would be represented by matrix:

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{bmatrix} \tag{4}$$

The objective of GSP is to give a framework able to operate over such a graph making use of these relationships.

1.3 Adjacency matrix, the weighted matrix and the Laplacian matrix

The adjacency matrix is not the only way of representing the graph. Other ways is using a weighted matrix and the Laplacian matrix.

The weighted matrix $\boldsymbol{W}$ has coefficients $w_{ij}>0$ if node i is connected to node j, and zero otherwise. The idea behind the weighted matrix is that the cost of the edges between connected nodes is not equal to one, thus considering that are nodes better connected than others. In this case:

$$y(n) = x(n) + \sum_{m \in N(n)} w_{mn} x(n) \longrightarrow \boldsymbol{y} = \boldsymbol{x} + \boldsymbol{W}\boldsymbol{x} \tag{5}$$

We can observe that the operator $\boldsymbol{A}$ is a special case of the operator $\boldsymbol{W}$, in which all weights are considered of the same value.

Finally, we can use a third operator called the Laplacian matrix $\boldsymbol{L}$, which is obtained as:

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W} \tag{6}$$

where $\boldsymbol{D}$ is the degree matrix and it has coefficients in the diagonal $d_{ii} = \sum_{j \neq i} w_{ij}$ (sum of row except the value at the diagonal) and the rest are 0.

Example 1.2 (Laplacian matrix)**.** *We want to obtain the laplacian matrix $\boldsymbol{L}$, from the weight matrix $\boldsymbol{W}$:*

$$\boldsymbol{W} = \begin{bmatrix} 0 & .6 & .3 & 0 \\ .6 & 0 & .1 & .4 \\ .3 & .1 & 0 & .2 \\ 0 & .4 & .2 & 0 \end{bmatrix} \tag{7}$$

We first obtain the diagonal matrix $\boldsymbol{D}$ as sum of rows $d_{ii} = \sum_{j \neq i} w_{ij}$:

$$\boldsymbol{D} = \begin{bmatrix} .9 & 0 & 0 & 0 \\ 0 & 1.1 & 0 & 0 \\ 0 & 0 & .6 & 0 \\ 0 & 0 & 0 & .6 \end{bmatrix} \tag{8}$$

Then, the Laplacian will be:

$$\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{W} = \begin{bmatrix} .6 & -.6 & -.3 & 0 \\ -.6 & 1.1 & -.1 & -.4 \\ -.3 & -.1 & .6 & -.2 \\ 0 & -.4 & -.2 & 0.6 \end{bmatrix} \tag{9}$$

1.4 Creating the graph

There are several ways of creating a graph. The key is in discovering which are the relationships among the nodes:

- **Physically knowledge of the weights:** there is an intrinsic knowledge of what are the weights, for example, circuits in electronic systems, social networks, etc;
- **Geometry of the vertex:** use Euclidean distances. In this case it is built as a decreasing function of the distance:

$$w_{ij} = \begin{cases} e^{-d_{ij}^2/(2\theta^2)} & \text{if } d_{ij} < T, \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

  The choice of the parameters T (threshold) and θ (standard deviation of the distribution), and of how to use the metric (in this case, inside a Gaussian distribution), are dictated by the application and by the analyst experience.

- **Obtain the Weighted matrix W or the Laplacian matrix L from the data measured matrix $\boldsymbol{X}$.**

A first example is to obtain the weight matrix using the correlations between sensor data. For that purpose, we consider that the signals of each node $\boldsymbol{x_1}, \ldots, \boldsymbol{x_N}$ form a multivariate Gaussian distribution N(0, $\boldsymbol{\Sigma}$). Then, the weighted matrix is obtained from the precision matrix $\boldsymbol{\Theta}$= $\boldsymbol{\Sigma^{-1}}$. Since only the sample covariance is known, the graphical Lasso (GLASSO) problem is to estimate the precision matrix $\bar{\boldsymbol{\Theta}}$:

$$\begin{aligned} &\text{minimize} && \operatorname{tr}(S\Theta) - \log\det(\Theta) + \lambda \sum_{j \neq k} |\Theta_{jk}| \\ &\text{subject to} && \boldsymbol{\Theta} \geq 0 \\ &\text{variable} && \boldsymbol{\Theta} \end{aligned} \tag{11}$$

where $\boldsymbol{S}$ is the sample covariance of $\boldsymbol{X}$ and λ is the penalizing parameter. Now, we define the weight matrix as $\boldsymbol{W}$=$\hat{\boldsymbol{\Theta}}$. We can control the sparsity of the graph by fixing a threshold T: if $\hat{\boldsymbol{\Theta}}_{\boldsymbol{ij}} \leq T$ then $\hat{\boldsymbol{\Theta}}_{\boldsymbol{ij}}$=0. These links with values $\hat{\boldsymbol{\Theta}}_{\boldsymbol{ij}}$=0 are thus disconnected in the graph.

Another example is to use a non-linear optimization model to directly obtain the Laplacian matrix $\boldsymbol{L}$ from the data matrix $\boldsymbol{X}$ [3]:

$$\begin{aligned} &\text{minimize} && \underbrace{\|\mathbf{X} - \mathbf{Y}\|_F^2}_{data\ \ fidelity} + \underbrace{\underbrace{\alpha\ \operatorname{tr}(\mathbf{Y}^\top \mathbf{L} \mathbf{Y})}_{smoothness} + \beta \|\mathbf{L}\|_F^2}_{sparsity} \\ &\text{subject to} && \operatorname{tr}(\mathbf{L}) = n, \\ & && L_{ij} = L_{ji} \leq 0, \qquad i \neq j, \\ & && \mathbf{L} \cdot \mathbf{1} = \mathbf{0} \\ &\text{variable} && \mathbf{L}, \mathbf{Y} \end{aligned} \tag{12}$$

We can say that this optimization problem is not convex (it has not global minimum). But it can be solved using in an alternating procedure, in which we first fix $\boldsymbol{Y} = \boldsymbol{X}$, so we find $\boldsymbol{L}$ from our data measurements $\boldsymbol{X}$. We can play the sparsity (more L_{ij}=0, meaning more nodes disconnected) of our solution using the β parameter. Finally when $\boldsymbol{L}$ is found, we can solved again fixing the $\boldsymbol{L}$ found and finding $\boldsymbol{Y}$ that will be a filtered version of our data $\boldsymbol{X}$.

2 Description of the data

You have two CSV files. The first one called *Node-Location.csv* contains the latitude (Lat) and longitude (Lon) of each node (there are 8 nodes). The first column is the name of the node and the second column (EOI) is a ID. You have to convert geolocations to distances using the *haversine formula*. Look in google for python code to obtain these distances.

The second file called *data_matrix.csv* contains the data. The first column contains a timestamp with hourly measurements (2258 samples, from 1st of June 2017 to 06 of October 2017). Then, you have 8 columns (one per node) with O_3 measurements.

3 Project realization

The goal of the project is to obtain the graph using each of the three techniques: i) distance-based, ii) GLASSO, and iii) Laplacian.

You can use the library *sklearn* from python for GLASSO (It would also be nice if you try to obtain the optimization problem by yourself and compare with the sklearn version). For the Laplacian case, you must program the optimization problem as it is described using cvxpy. For plotting the graph you can use library *networkx*. Play with various parameters to obtain various sparsitites. Obtain results and plot graphs or draw tables showing what you deduce and learn. Explain how you solve the various steps and obtain the results. As always, you have to submit a report with your conclusions.

References

[1] Ferrer-Cid, Pau, Jose M. Barcelo-Ordinas, and Jorge Garcia-Vidal. *Graph learning techniques using structured data for IoT air pollution monitoring platforms.* IEEE Internet of Things Journal, 2021, vol. 8, no 17, pp. 13652-13663, `https://ieeexplore.ieee.org/document/9382408`.

[2] J. Friedman, T. Hastie, and R. Tibshirani, *Sparse inverse covariance estimation with the graphical lasso*, Biostatistics, vol. 9, no. 3, pp. 432- 441, 2008 `https://arxiv.org/abs/0708.3517`.

[3] X. Dong, D. Thanou, P. Frossard and P. Vandergheynst, *Laplacian matrix learning for smooth graph signal representation* 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), South Brisbane, QLD, Australia, 2015, pp. 3736-3740, `https://arxiv.org/pdf/1406.7842.pdf`.