1 Laplacian Eigenmaps

Frequently we are interested in understanding underlying large-scale structure in networks, often taking the form of a latent-space structure. In this latent space, nodes occupy positions along some axis or axes and the presence or absence of links between nodes depends on their positions. Typically, links are more likely to exist between nodes that are closely situated in this latent space. However, we usually lack direct knowledge of how this latent space looks like, as we can only observe the network itself. To extract this latent space, we apply optimization algorithms to estimate the positions of nodes within the latent space solely based on the network's structure. Therefore, the goal of an embedding algorithm is to learn the nodes' positions.

Suppose that we have a network represented by its adjacency matrix A. Following the intuition that links are more likely between nodes that are close in the latent space, formulate an objective function for a minimization problem to learn the nodes' position x_i (for simplicity, embed the nodes in a one-dimensional space). Solve the minimization problem. Interpret and discuss your solution including any (technical) problems that you encountered. Give directions on how to solve these problems. How would you extend your formulation to a multi-dimensional space?

Hint: It may be a better choice to work with graph matrices that you can derive directly from the adjacency matrix A. For example, we learned about the degree matrix, incidence matrix, and graph Laplacian.

1.1 Objective function

Consider an undirected unweighted graph G(V, E) with n nodes and adjacency matrix $A \in R^{n \times n}$. Let's assume that for each node i we can find $x_i \in R$ and for $(i, j) \in E$ x_i and x_j are very close. For the objective function, we will use the the sum of squares divided by 2, because each link was taken twice:

$$
\begin{aligned}
J &= \frac{1}{2}\sum_{i,j}^{N}(x_i - x_j)^2 A_{ij} \\
&= \frac{1}{2}\sum_{i,j}^{N} x_i^2 A_{ij} + \frac{1}{2}\sum_{i,j}^{N} x_j^2 A_{ij} - \sum_{i,j}^{N} x_i x_j A_{ij} \\
&= \frac{1}{2} 2\sum_{i,j}^{N} x_i^2 A_{ij} - \sum_{i,j}^{N} x_i x_j A_{ij} \\
&= \sum_{i}^{N} x_i^2 d_i - \sum_{i,j}^{N} x_i x_j A_{ij} \\
&= x^T D x - x^T A x \\
&= x^T L x
\end{aligned}
$$

So our objective is to find $\hat{x} = \underset{x \in R^n}{argmin}\, x^T L x$. This problem is unconstrained. Let‘s make an objective function scaling invariant in x:

$$
\begin{aligned}
\sum_{i}^{N} x_i^2 &= 1 \\
x^T x &= 1
\end{aligned}
$$

So the problem now is constrained and we will resolve it using Lagrange multipliers.

$$
\begin{aligned}
&argmin\, x^T L x + \lambda(1 - x^T x) \\
&x \in R^n \\
&x^T x = 1
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial}{\partial x} x^T L x + \lambda(1 - x^T x) &= 0 \\
2Lx - 2\lambda x &= 0 \\
Lx &= \lambda x
\end{aligned}
$$

The last equation is the eigenvalue problem. Let‘s sort eigenvalues of a Laplacian matrix in an ascending way:

$$0 = \lambda_0 \leq \ldots \leq \lambda_j \leq \ldots \leq \lambda_n$$

At least one eigenvalue in a Laplacian matrix will be equal to 0 and have a $\vec{1}$ eigenvector. This is a trivial solution that maps all points to the same point 1. To avoid it we will apply the constraint of orthogonality to the $\vec{1}$:

$$\begin{aligned}
&argmin\, x^T Lx \\
&x \in R^n \\
&x^T x = 1 \\
&x^T \vec{1} = 0
\end{aligned}$$

It will mean that x will be the eigenvector of the first lowest non-zero eigenvalue.

1.2 Problems

So we got the primitive result which also has a problem: the higher the degree of the node the higher the eigenvecor element that corresponds to that node. So we will need to consider node degrees in the constraints of the optimization problem:

$$\begin{aligned}
&argmin\;\; x^T Lx \\
&x \in R^n \\
&x^T Dx = 1 \\
&x^T D\vec{1} = 0
\end{aligned}$$

The solution of this problem would be also an eigenvector problem:

$$Lx = \lambda Dx$$

The solution for this problem would be the eigenvector of the lowest non-zero eigenvalue of the normalized Laplacian matrix.

1.3 Multidimensional case

Let's consider $X_i \in R^n$ as a n-dimensional encoding vector of node i. So an objective function will be:

$$\begin{aligned}
J &= \frac{1}{2}\sum_{i,j}^{N} ||X_i - X_j||^2 A_{ij} \\
&= \frac{1}{2}\sum_{i,j}^{N}\sum_{k=1}^{d}(x_i^k - X_j^k)^2 A_{ij} \\
&= \frac{1}{2}\sum_{k=1}^{d}\sum_{i,j}^{N}(x_i^k - X_j^k)^2 A_{ij}
\end{aligned}$$

So we have splited the multidimensional problem into the sum of 1-dimensional problems. Each of that 1-dimensional problems is resolved by smallest non-zero eigenvector of Laplacian matrix. For avoiding trivial solution if we use d dimensions in encosings we take d first non-zero eigenvalues and corresponding eigenvectors. The encoding for the node i will be:

$$X_i = (x_i^1, \dots, x_i^d),$$

where x_i^k - i-th element of the eigenvector which correspond to k-th non-zero eigenvalue of the Laplacian matrix.