A COMPARATIVE EVALUATION ON NETWORK REPRESENTATION LEARNING TECHNIQUES

Konstantinos Ameranis **Isidora Kiourti** **Konstantinos Sotiropoulos** **Erasmo Tani**

Isidora Tourni

ABSTRACT

The abstract paragraph should be indented 1/2 inch (3 picas) on both left and right-hand margins. Use 10 point type, with a vertical spacing of 11 points. The word ABSTRACT must be centered, in small caps, and in point size 12. Two line spaces precede the abstract. The abstract must be limited to one paragraph.

1 INTRODUCTION

2 NETWORK EMBEDDINGS

2.1 MATRIX FACTORIZATION

MULTIDIMENSIONAL SCALING (MDS)

Multidimensional Scaling (MDS) is a method for creating a Euclidean embedding of data for which one has distance / dissimilarity information. For instance, given an $N \times N$ matrix of distances between N points, one can embed the points into $\mathbb{R}^k$ so as to preserve distance information. In particular, one can use MDS as a way to create useful features for graphs by considering the shortest path distance between vertices. MDS is similar to PCA, except instead of using correlation information, we make use of pointwise distances.

Classical Multidimensional Scaling works as follows: let D be the dissimilarity matrix. Then:

1. Let $D^{(2)}$ be the point-wise square of the distance matrix,
2. Let $J = I - \frac{1}{n}\vec{1}\vec{1}^T$,
3. Let $B = -\frac{1}{2}JD^{(2)}J$,
4. Find the top m eigenvalues of B $\lambda_1, ...\lambda_m$, and the corresponding eigenvalues $e_1, ..., e_m$,
5. Let $X = E_m\Lambda^{1/2}$.

(note to self:doesn't work if distance not euclidean)

Classical Multidimensional Scaling minimizes a loss function called *strain*:

$$Strain_D(x_1, ..., x_N) = \left(\frac{(\sum_{i,j} b_{i,j} - \langle x_i, x_j\rangle)^2}{\sum_{i,j} b_{i,j}^2}\right).$$

SPECTRAL EMBEDDING

Another way to embed a graph in Euclidean space is given by the spectral embedding. This method computes the k eigenvectors of the normalized Laplacian matrix $\mathcal{L}$ corresponding to the k smallest eigenvalues, and uses each of them as an embedding of the vertices into $\mathbb{R}$, resulting in an embedding into $\mathbb{R}^k$. The normalized Laplacian matrix is given by:

$$\mathcal{L} = D^{-\frac{1}{2}}(D - A)D^{-\frac{1}{2}}$$

One can prove that the quadratic form of the Laplacian is a relaxation to the minimum conductance cut problem defined as follows:

$$\underset{S \subseteq V}{\text{minimize}} \frac{|E(S, \overline{S})|}{\min\{vol(S), vol(\overline{S})\}}.$$

The eigenvectors of the Laplacian therefore act as optimizers of the relaxation, and tend to align points in space so as to keep connected points close to each other.

ISOMAP

Isomap is a method for manifold learning / non-linear dimensionality reduction. In a general setting, given data points living in a (non-linear) manifold in $\mathbb{R}^n$ we would like to embed them into lower dimensional space while preserving geodesic distances.

2.2 RANDOM WALKS

3 GRAPH CONVOLUTIONAL NETWORKS

The recent popularity of convolutional neural networks **?**, and their success in classifying images and other tasks with a striking accuracy, has intrigued the scientific community with the question of whether -at least a variant of them- can be leveraged in learning tasks of graph networked data. The use of graph structures in computer vision**?**, as well in the representation of social networks **?**, has made this task look appealing and worth of investigation-research. However, using convolutional neural networks for networked data directly, is not straightforward and it poses significant challenges.

Challenges in Graph Convolutional Networks First and foremost, as CNNs were mainly used for image classification, they assume that data lie on a regular grid in a geometric space (most commonly Euclidean). On the contrary, graph data are a typical form of unordered data that lie in an irregular domain. Also, the heavy-tailed distribution of node degrees in real networks **?** makes the filtering-convolution part difficult, as it is not easy to define a constant-sized neighborhood and apply a localized filter, as in the traditional CNN case. Also, the pooling stage has to be defined as well.

Convolution in graphs First efforts to address the above mentioned issues by Bruna et al. **?** and introduce the architecture of CNNs to networked data, mainly draw from the field of Graph Signal Processing **?**. Graphs are generally considered undirected and their representation is given through their **Laplacian** $L = D - A$ or more commonly the normalized **Laplacian** $L = I_n - D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$. As this matrix is a real symmetric and positive semidefinite, it admits an eigenvector decomposition $L = U \Lambda U^T$, where U is a square orthonormal matrix with the eigenvectors as its columns, and Λ is the diagonal matrix of eigenvalues. Letting $x \in \mathbb{R}^n$ be a feature vector of the nodes of a graph, the *graph fourier transform* is then defined as $\hat{x} = U^T x \in \mathbb{R}^n$ and its inverse as $x = U\hat{x}$. The **convolution operator** on a graph $\mathcal{G}$ is defined on the Fourier domain, as:

$$x *_{\mathcal{G}} y = U((U^T x) \odot (U^T y))$$

where $\odot$ denotes the element-wise Hadamard product.
Therefore, a signal x is filtered by g_θ, as:

$$y = g_\theta(L)x = g_\theta(U \Lambda U^T)x = U g_\theta(\Lambda) U^T x \tag{1}$$

where $g_\theta(\Lambda) = diag(\theta)$ is a non-parametric filter.
This approach has, however, the following limitations:

1. Filters are not localized
2. Their learning complexity scales with the dimensionality of the data $O(n)$
3. The computational cost of filtering is high- $O(n^2)$, due to the multiplication with the Fourier basis U.