Graph Convolutional Network (GCN)

Xi Chen

xchen595@163.com

why GCN?

1 CNN cannot hold the characteristics of translation invariance on non-euclidean structure data

2 but we want to implement machine learning by fetching spatial features from this kind of topology data

3 Topology connects is actually one of generalized data structures (i.e spectral clustering)

How to extract spatial feature from non-euclidean structure data?

From the perspective of spectral domain, researches went through Fourier Transformation, and convolution on graph, to Graph Convolutional Network which inspired by deep learning

Note: Spectral graph theory: making use of the eigenvalues and eigenvectors of the Laplace matrix of the graph to explore what the graph brings us

why Laplace matrix plays critical role on GCN?

Since GCN cannot pass Laplace matrix on Graph Fourier Transformation and the definition of Graph Conovolution, we would like to see what is Laplace matrix.

There are mainly 3 versions:

(1)
$$L = D - A$$
: Combinatorial Laplacian

(2)
$$L^{sys} = D^{\frac{-1}{2}} L D^{\frac{1}{2}}$$
: Symmetric normalized Laplacian

(3)
$$L^{rw} = D^{-1} A$$
: Random walk normalized Laplacian

Great nature of Laplacian matrix:

(1) The Laplace matrix is a symmetric matrix, which can be decomposed by eigenvalues
(2) The Laplace matrix has only non-zero elements on 1-hop neighbor, and everything else is zero
(3) Laplacian operator and Laplacian matrix are anology to each other

Decomposition of Laplace matrix

The decomposition:
$$L = U \begin{bmatrix} \lambda_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \lambda_n \end{bmatrix} U^{-1} = U \begin{bmatrix} \lambda_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \lambda_n \end{bmatrix} U^T$$

Note we have the orthogonal matrix, i.e $UU^T = E$

Fourier Transformation and convolution on GRAPH

(1) Fourier Transformation on Graph

- (a) Fourier Transformation

Classical Fourier Transformation:

$$F(\omega) = F[f(t)] = \int f(t)e^{-i\omega t}\,dt$$

Here, $e^{-i\omega t}$ is the eigenvalue function of Laplace operator, which leads to the relationship between ω and eigenvalue

Generalized eigenvalue equation:

$$AV = \lambda V$$

Here are transformation, eigenvalue vector, and eigenvalue for $AV and \lambda$ respectively.

We have $e^{-i\omega t}$ satisfied:

$$\Delta e^{-i\omega t} = \frac{\partial^2}{\partial t^2} e^{-i\omega t} = -\omega^2 e^{-i\omega t}$$

$e^{-i\omega t}$ is the eigenvalue function of Tranformation Δ, and ω has strong bond with eigenvalue.

Now, when we turn to handle the Graph issue, Laplace matrix naturally brings us to find the eigenvector.

So:

$$LV = \lambda V$$

$$F(\lambda_l) = \hat{f}(\lambda_l) = \sum_{i=1}^{N} f(i)u_l^*(i)$$

f is N dimension vector on Graph, $f(i)$ connected to vetexes correspondently, $u_l(i)$ is the ith component of lth eigenvector. Under λ_l, the Fourier transformation of f in Graph is the inner product of λ_l and its eigenvector u_l

The matrix form of Fourier Transformation on Graph

$$\begin{bmatrix} \hat{f}(\lambda_1) \\ \hat{f}(\lambda_2) \\ \vdots \\ \hat{f}(\lambda_N) \end{bmatrix} = \begin{bmatrix} u_1(1) & u_1(2) & \dots & u_1(N) \\ u_2(1) & u_2(2) & \dots & u_2(N) \\ \vdots & \vdots & \ddots & \vdots \\ u_N(1) & u_N(2) & \dots & u_N(N) \end{bmatrix} \begin{bmatrix} f(1) \\ f(2) \\ \vdots \\ f(N) \end{bmatrix}$$

- (b) Inverse Fourier Transformation

Similarily, we transfer the integration of frequence ω, i.e.

$$F^{-1}[F(\omega)] = \frac{1}{2II} \int F(\omega)e^{-i\omega t}\ d\omega$$

to the sum of eigenvalue λ_l, $f(i) = \sum_{i=1}^{N} \hat{f}(\lambda_l)u_l(i)$.

And the matrix form,

$$\begin{bmatrix} f(1) \\ f(2) \\ \vdots \\ f(N) \end{bmatrix} = \begin{bmatrix} u_1(1) & u_1(2) & \dots & u_1(N) \\ u_2(1) & u_2(2) & \dots & u_2(N) \\ \vdots & \vdots & \ddots & \vdots \\ u_N(1) & u_N(2) & \dots & u_N(N) \end{bmatrix} \begin{bmatrix} \hat{f}(\lambda_1) \\ \hat{f}(\lambda_2) \\ \vdots \\ \hat{f}(\lambda_N) \end{bmatrix}$$

so $f = U\hat{f}$.

(2) Convolution on Graph

Convolution theorem: the Fourier transform of the function convolution is the product of the Fourier transform of the function, that is, the convolution of $f(t)$ and $h(t)$ is the inverse transformation of the product of the Fourier transform of the function

$$f * h = F^{-1}[\hat{f}9\omega)\hat{h}(\omega)] = \frac{1}{2II} \int \hat{f}(\omega)\hat{h}(\omega)e^{i\omega t} d\omega$$

So,

$$(f * h)_G = U \begin{bmatrix} \hat{h}(\lambda_1) & & \\ & \ddots & \\ & & \hat{h}(\lambda_n) \end{bmatrix} U^T f$$

Note: we also have $(f * h)_G = U((U^T h) hadamardproduct(U^T f))$

Why is the eigenvector of the Laplace matrix the basis for the Fourier transform? eigenvvalue stands for frequency.

- (a) basis The Fourier Transform is actually, **representing any function as a linear combination of orthogonal functions, sine and cosine.**

then, $f = \hat{f}(1)u_1 + \hat{f}(2)u_2 + ... + \hat{f}(n)u_n$

Since we have linearly independent vectors in an n-dimensional space can form a basis for a space, and the eigenvectors of the Laplace matrix are also an orthonormal basis, and $(u_1, u_2, ..., u_n$ is n-dimensianl linearly independent vectors on graph, any vector f can be expressed like this combination.

- (b) frequency

The n non-negative real eigenvalues of Laplace matrix L are arranged from small to large as $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$, and the minimum eigenvalue, $\lambda_1 = 0$

And the n-dimensional space defined by graph, the smaller eigenvalue λ_l, the less information which means the corresponding basis u_l weights can be ignored.

y is $O(n)$ - (3) better spatial localization, receptice field (weighted sum over K-hop neighbor, weights are α_k)

CNN convolution essentially is the use of a Shared parameters of filter, by calculating the center pixel and adjacent pixels of the weighted space and to form feature map for feature extraction, the weighted coefficient is the convolution kernel weight coefficient

Graph convolution's parameters is $diag(\hat{h}(\lambda_l))$

- 1th generation of GCN:

$$y_{output} = \sigma \left\{ U \left\{ \begin{matrix} \theta_1 & & \\ & \ddots & \\ & & \theta_n \end{matrix} \right\} U^T x \right\}$$

- 2nd generation of GCN:

$$y_{output} = \sigma \left\{ U \begin{Bmatrix} \sum_{j=1}^{K} \alpha_j \lambda_l^j & & \\ & \ddots & \\ & & \sum_{j=1}^{K} \alpha_j \lambda_n^j \end{Bmatrix} U^T x \right\}$$

  Here we have,

$$\begin{Bmatrix} \sum_{j=1}^{K} \alpha_j \lambda_l^j & & \\ & \ddots & \\ & & \sum_{j=1}^{K} \alpha_j \lambda_n^j \end{Bmatrix} = \sum_{j=1}^{K} \alpha_j \Lambda^j$$

and $L^2 = U\Lambda U^T U \Lambda U^T = U\Lambda^2 U^T, U^T U = E$

then,

$$U \sum_{j=1}^{K} \alpha_j \lambda_n^j U^T = \sum_{j=1}^{K} \alpha_j U \lambda_n^j U^T = \sum_{j=1}^{K} \alpha_j L^j$$

so, $y_{output} = \sigma(\sum_{j=1}^{K} \alpha_j L^j x)$

This adventages can be summarized in:

- (1) there are only K parameters in convolution kernel
- (2) Don't need matrix decomposition anymore, we switch into Lapacian matrix L transformation, time complexit is $O(n)$
- (3) spatial localization, receptive filed (weighted sum over K-hop neighbor feature, weights α_k)