Graph Neural Networks: The Convolutional Layer

Edoardo Grasso

A Comprehensive Survey on Graph Neural Networks

Zonghan Wu, Shirui Pan, *Member, IEEE*, Fengwen Chen, Guodong Long, Chengqi Zhang, *Senior Member, IEEE*, and Philip S. Yu, *Life Fellow, IEEE*

Machine Learning on Networks?

Network Embedding

- Represent nodes as low dimensional vector;
- Preserve network topology structure as well as node content information;
- Use off-the-shelf ML algorithms.

GNN

- Develop ad hoc algorithm to address graph-related tasks.
- RecGNN, ConvGNN, GAE, GGAN, ...

Convolutional Neural Networks

2	4	9	1	4
2	1	4	4	6
1	1	2	9	2
7	3	5	1	3
2	3	4	8	5

Image

X

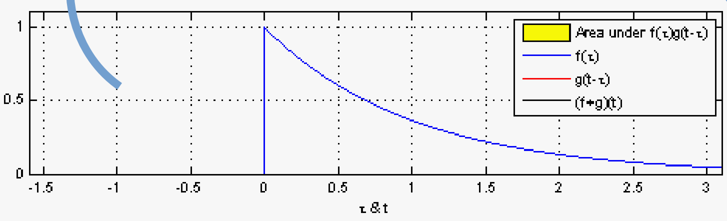
1	2	3
-4	7	4
2	-5	1

Filter / Kernel

=

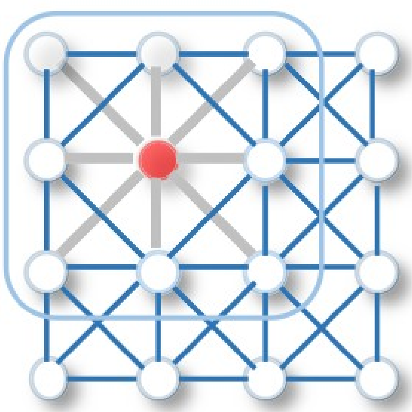
51		

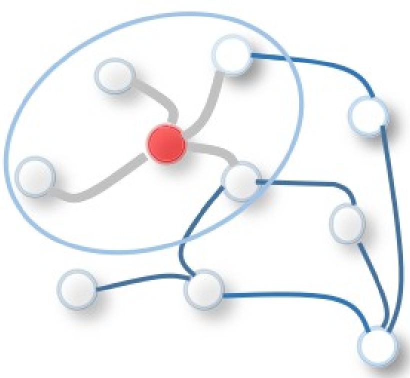
Feature



$$(f * g)(t) := \int_{-\infty}^{\infty} f(\tau) g(t-\tau) d\tau$$

Problem:





What about non-euclidean spaces?
- *Spectral-based ConvGNN*
- *Spatial-based ConvGNN*

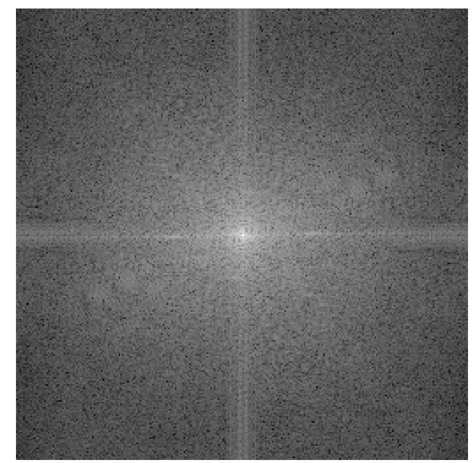
FCNN: Fourier Convolutional Neural Networks

Harry Pratt, Bryan Williams, Frans Coenen, and Yalin Zheng

Convolution Theorem:

$$\mathcal{F}(\kappa * u) = \mathcal{F}(\kappa) \odot \mathcal{F}(u)$$

$$F(k,l) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(i,j)\, e^{-\iota 2\pi \left(\frac{ki}{N} + \frac{lj}{N}\right)}$$



- Shift filter and multiply image by filter ~ N^2 times.

- Multiply image by filter one time;
- Calculate DFT for NxN image Cooley-Tukey → N*log(N);
- Very efficient pooling.

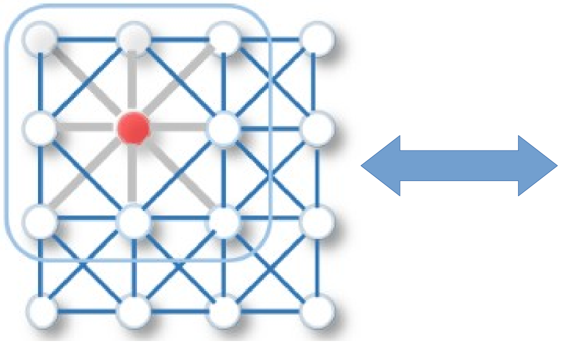
Same thing applied on Graphs

For a graph with n vertices:

$$\mathbf{L} = \mathbf{I_n} - \mathbf{D}^{-(1/2)}\mathbf{A}\mathbf{D}^{-(1/2)}$$ Symmetrically normalized laplacian

$$\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T \qquad \mathbf{U} = [\mathbf{u}_0, \mathbf{u}_1, \ldots, \mathbf{u}_{n-1}] \in \mathbf{R}^{n \times n}$$ Spectral decomposition

Let $\mathbf{x} \in \mathbf{R}^n$ be a graph signal, each entry x_i is the value of the signal for the i-th node.



Using an analogy, this can be viewed as the greyscale value of the pixel in a matrix-like graph.

Same thing applied on Graphs

The graph Fourier Transform of a signal is defined as: $\mathscr{F}(\mathbf{x}) = \mathbf{U}^T\mathbf{x}$

$$F(k,l) = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} f(i,j)\, e^{-\iota 2\pi(\frac{ki}{N}+\frac{lj}{N})}$$

And the inverse Fourier Transform: $\mathscr{F}^{-1}(\hat{\mathbf{x}}) = \mathbf{U}\hat{\mathbf{x}}$

The graph convolution of the input signal $\mathbf{x}$ with a filter $\mathbf{g} \in \mathbf{R}^n$ is:

$$\begin{aligned} \mathbf{x} *_G \mathbf{g} &= \mathscr{F}^{-1}(\mathscr{F}(\mathbf{x}) \odot \mathscr{F}(\mathbf{g})) \\ &= \mathbf{U}(\mathbf{U}^T\mathbf{x} \odot \mathbf{U}^T\mathbf{g}) \end{aligned}$$

$$\left\{ \mathbf{x}_1 \odot \mathbf{x}_2 = \mathrm{diag}(\mathbf{x}_2)\mathbf{x}_1 \right\}$$

Let $\mathbf{g}_\theta = \mathrm{diag}(\mathbf{U}^T\mathbf{g})$ $\rightarrow$ $\mathbf{U}(\mathbf{U}^T\mathbf{x} \odot \mathbf{U}^T\mathbf{g}) = \mathbf{U}\mathbf{g}_\theta\mathbf{U}^T\mathbf{x}$

Semi-Supervised Classification with Graph Convolutional Networks

Thomas N. Kipf **Max Welling**

$$g_\theta \star x = U g_\theta U^\top x$$

Where $g_\theta = \mathrm{diag}(\theta)$ is a filter parametrized by $\theta \in \mathbb{R}^N$ in the **Fourier domain**.

g_θ is a function of the eigenvalues of the laplacian: $g_\theta(\Lambda)$.

PROBLEM

- Multiplication with the eigenvector matrix U is O(N²);
- Computing the eigendecomposition of L might be prohibitively expensive for large graphs.

SOLUTION

$g_\theta(\Lambda)$ → Expansion in terms of the Chebychev polynomials, truncated to the first order. [Hammond et. al (2011)]

The Convolutional Layer

$$g_\theta \star x \approx \theta \underbrace{\left(I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \right)}_{\text{Has eigenvalues} \leq 2 \rightarrow \text{Exploding Gradients}} x$$

Renormalization: $I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \rightarrow \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}}$

$\tilde{A} = A + I_N$ and $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$

Generalize for signal $X \in \mathbb{R}^{N \times C}$: $Z = \tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} X \Theta$

where $\Theta \in \mathbb{R}^{C \times F}$ is a matrix of filter parameters → learnable weigths and $Z \in \mathbb{R}^{N \times F}$ is the convolved signal matrix.

Complexity: $\mathcal{O}(|\mathcal{E}|FC)$

The Convolutional Layer

FINAL STEP: Apply non-linearity

$$H^{(l+1)} = \sigma\left(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right)$$

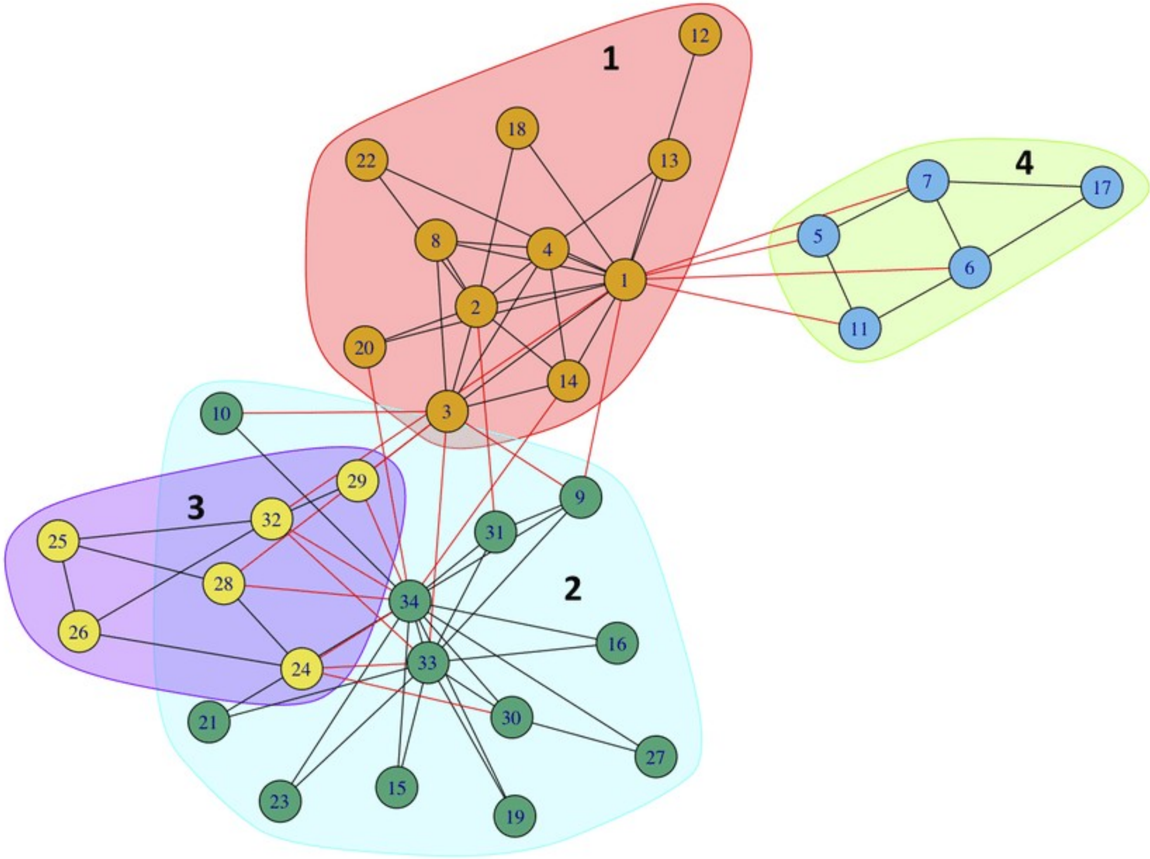
EXAMPLE: If we consider a two-layer GCN for semi-supervised node classification task we will have

$$Z = f(X, A) = \mathrm{softmax}\left(\hat{A}\ \mathrm{ReLU}\left(\hat{A}XW^{(0)}\right)W^{(1)}\right)$$

with $\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$.

Zachary's Karate Club?

$$Z = f(X, A) = \mathrm{softmax}\Big(\hat{A}\ \mathrm{ReLU}\Big(\hat{A}XW^{(0)}\Big)\, W^{(1)}\Big)$$



Questions?

Thanks for listening.