Graph Neural Networks: A Review of Methods and Applications

Jie Zhou*, Ganqu Cui*, Zhengyan Zhang*, Cheng Yang, Zhiyuan Liu, Maosong Sun
Tsinghua University

arxiv 2019

`https://qdata.github.io/deep2Read`

Presenter : Jack Lanchantin

Outline

1. Introduction
2. Graph Neural Networks
3. Graph Variants
4. Objective Variants
5. Propagation Variants
    - Convolution
        - Spectral
        - Spatial
    - Gating
    - Attention
    - Skip Connections
6. Training Variants
7. Applications and Datasets

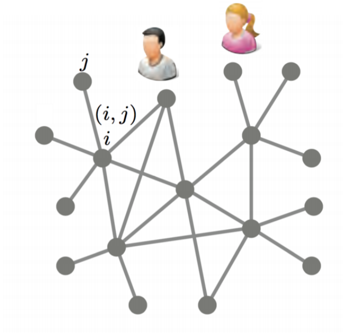
Graphs

Type of data structure which models a set of objects (vertices) and their relationships (edges).

Graphs

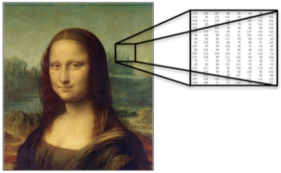
Type of data structure which models a set of objects (vertices) and their relationships (edges).

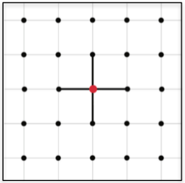
- **Graph** $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
- **Vertices** $\mathcal{V} = \{1, \dots, n\}$
- **Edges** $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$
- **Vertex weights** $b_i > 0$ for $i \in \mathcal{V}$
- **Edge weights** $a_{ij} \geq 0$ for $(i, j) \in \mathcal{E}$



Euclidean Data: RNNs/CNNs

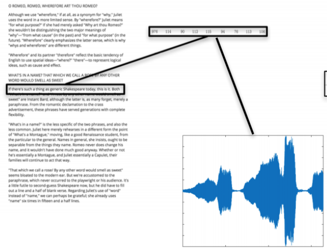
- Image, volume, video: 2D, 3D, 2D+1 Euclidean domains





2D grids

- Sentence, word, sound: 1D Euclidean domain



O ROMEO, ROMEO, WHEREFORE ART THOU ROMEO?

Although we use "wherefore," if at all, as a synonym for "why," Juliet uses the word in a more limited sense. By "wherefore?" Juliet means "for what purpose?" If she had merely asked "Why art thou Romeo?" she wouldn't be distinguishing the two major meanings of "why"—"from what cause" (in the past) and "for what purpose" (in the future). "Wherefore" clearly emphasizes the latter sense, which is why "whys and wherefores" are different things.

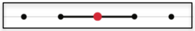
"Wherefore" and its partner "therefore" reflect the basic tendency of English to use spatial ideas—"where?" "there"—to represent logical ideas, such as cause and effect.

WHAT'S IN A NAME? THAT WHICH WE CALL A ROSE BY ANY OTHER WORD WOULD SMELL AS SWEET

If there's such a thing as generic Shakespeare today, this is it. Both [illegible] sweet" are instant Bard, although the latter is, as many forget, merely a paraphrase. From the romantic declamation to the crass advertisement, these phrases have served generations with complete flexibility.

"What's in a name?" is the less specific of the two phrases, and also the less common. Juliet here merely rehearses in a different form the point of "What's a Montague," moving, like a good Renaissance student, from the particular to the general. Names in general, she insists, ought to be separable from the things they name. Romeo never does change his name, and it wouldn't have done much good anyway. Whether or not he's essentially a Montague, and Juliet essentially a Capulet, their families will continue to act that way.

"That which we call a rose/ By any other word would smell as sweet" seems bloated to the modern ear. But we're accustomed to the paraphrase, which never occurred to the playwright or his audience. It's a little futile to second-guess Shakespeare now, but he did have to fill out a line and a half of blank verse. Regarding Juliet's use of "word" instead of "name," we can perhaps be grateful; she already uses "name" six times in fifteen and a half lines.



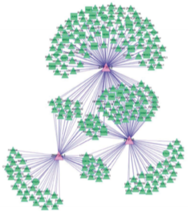
1D grid

- These domains have nice regular spatial structures.

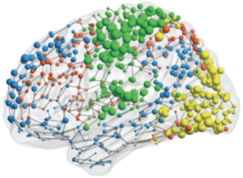
  $\Rightarrow$ All ConvNet operations are math well defined and fast (convolution, pooling).

Non-Euclidean Data

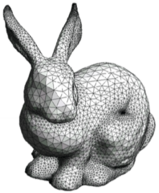
Social networks



Regulatory networks

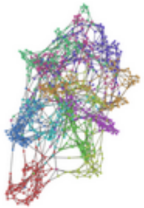


Functional networks



3D shapes

=



Graphs/ Networks

- Also chemistry, NLP, physics, social science, communication networks, etc.

How to extend CNNs to graph-structured data?

- Assumption: Non-Euclidean data are locally stationary and manifest hierarchical structures.
- But, how to define compositionality on graphs? (convolution and pooling on graphs)

Outline

1 Introduction

2 Graph Neural Networks

3 Graph Variants

4 Objective Variants

5 Propagation Variants
- Convolution
  - Spectral
  - Spatial
- Gating
- Attention
- Skip Connections

6 Training Variants

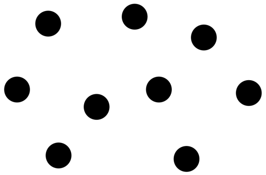
7 Applications and Datasets

Learning on Graphs

Naive approach: a per-node classifier

- Represent each node v as vector $\mathbf{h}_v \in \mathbb{R}^s$
- Completely drop the graph structure, and classify each node individually, with a shared deep neural network classifier:

$$\mathbf{o}_v = f(\mathbf{h}_v; \mathbf{W}) \tag{1}$$

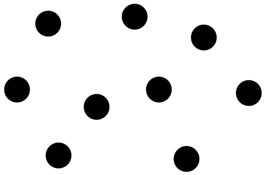


Learning on Graphs

Naive approach: a per-node classifier

- Represent each node v as vector $\mathbf{h}_v \in \mathbb{R}^s$
- Completely drop the graph structure, and classify each node individually, with a shared deep neural network classifier:

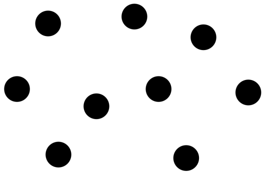
$$\mathbf{o}_v = f(\mathbf{h}_v; \mathbf{W}) \tag{1}$$



- Methods like DeepWalk also classify each node independently, but inject graph structure indirectly using learned embeddings

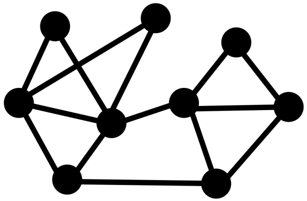
Learning on Graphs
(from Petar Velickovic)

- Most deep learning is done this way, even if there are relationships between training examples



Graph Neural Networks

- Originally introduced by Scarselli et. al. (2009)
- The goal of GNNs is to learn a state embedding $\mathbf{h}_v \in \mathbb{R}^s$ which contains information of the neighborhood for each node
- $\mathbf{h}_v$ can be used to produce an output $\mathbf{o}_v$ such as the node label



Graph Neural Networks

- Local transition function f: updates each node state according to its neighborhood (shared among all nodes)
- Output function g: describes how the output is produced

Graph Neural Networks

- Local transition function f: updates each node state according to its neighborhood (shared among all nodes)
- Output function g: describes how the output is produced
- Then, $\mathbf{h}_v$ and $\mathbf{o}_v$ are defined as follows:

$$\mathbf{h}_v = f(\mathbf{x}_v, \mathbf{x}_{e(v)}, \mathbf{h}_{n(v)}, \mathbf{x}_{n(v)}) \tag{2}$$

$$\mathbf{o}_v = g(\mathbf{h}_v, \mathbf{x}_v) \tag{3}$$

where $\mathbf{x}_v$ are the features of v, $\mathbf{x}_{e(v)}$ the features of its edges, $\mathbf{h}_{n(v)}$ the states of the nodes in the neighborhood of v, and $\mathbf{x}_{n(v)}$ the features of the nodes in the neighborhood of v

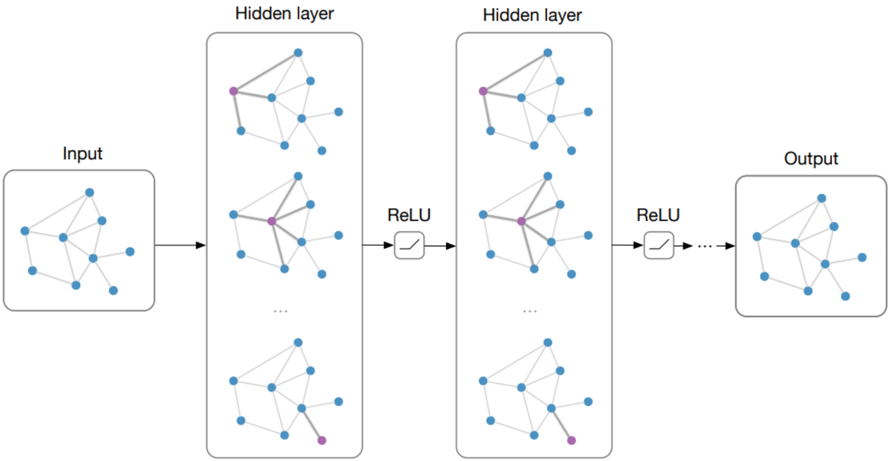
Graph Neural Networks Update Step

GNNs use the following iterative update to compute the state:

$$\mathbf{H}^{t+1} = f(\mathbf{H}^{t}, \mathbf{X}) \tag{4}$$

where $\mathbf{H}^t$ denotes the t-th iteration of $\mathbf{H}$.

Graph Neural Networks



Graph Neural Network Training

- Given a GNN framework, learn the parameters of f and g
- With the target information ($\mathbf{t}_v$ for a specific node) for the supervision, the loss can be written as:

$$loss = \sum_{i=1}^{p} (\mathbf{t}_i - \mathbf{o}_i) \tag{5}$$

  where p is the number of supervised nodes.
- Weights $\mathbf{W}$ of f and g are updated via gradient descent

Graph Neural Networks Training

- Original GNN constrains f to be a contractive map
  - Contractive map, on a metric space (M, d) is a function f from M to itself, with the property that there is some real number $0 \leq k < 1$ such that for all x and y in M, $d(f(x), f(y)) \leq k\, d(x, y)$

Graph Neural Networks Training

- Original GNN constrains f to be a contractive map
  - Contractive map, on a metric space (M, d) is a function f from M to itself, with the property that there is some real number $0 \leq k < 1$ such that for all x and y in M, $d(f(x), f(y)) \leq k\, d(x, y)$
- This implies that the h_v vectors will always converge to a unique fixed point (very restrictive)
- Impossible to inject problem-specific information into h_v^0 (will always converge to same value regardless of initialization)

Graph Neural Networks (compact representation)

Let $\mathbf{H}$, $\mathbf{O}$, $\mathbf{X}$, and $\mathbf{X}_N$ be the vectors constructed by stacking all the states, all the outputs, all the features, and all the node features, respectively. Then we have a compact form as:

$$\mathbf{H} = f(\mathbf{H}, \mathbf{X}) \tag{6}$$

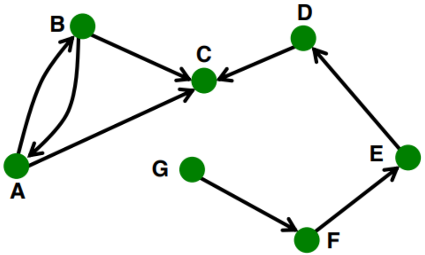
$$\mathbf{O} = g(\mathbf{H}, \mathbf{X}_N) \tag{7}$$

Outline

1. Introduction
2. Graph Neural Networks
3. Graph Variants
4. Objective Variants
5. Propagation Variants
   - Convolution
     - Spectral
     - Spatial
   - Gating
   - Attention
   - Skip Connections
6. Training Variants
7. Applications and Datasets

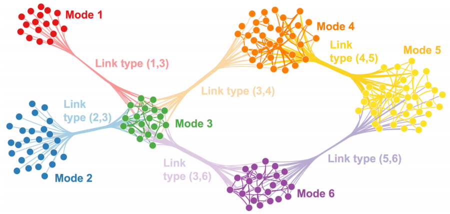
Directed Graphs

- e.g. Knowledge graphs where there is a parent and child.
- Can use different weights for each edge type



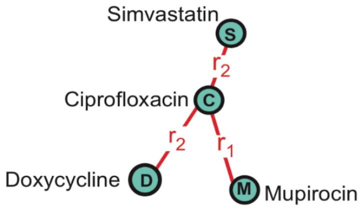
Heterogeneous Graphs

- e.g. Multi-model biological network data
- Can group neighbors based on types or distance

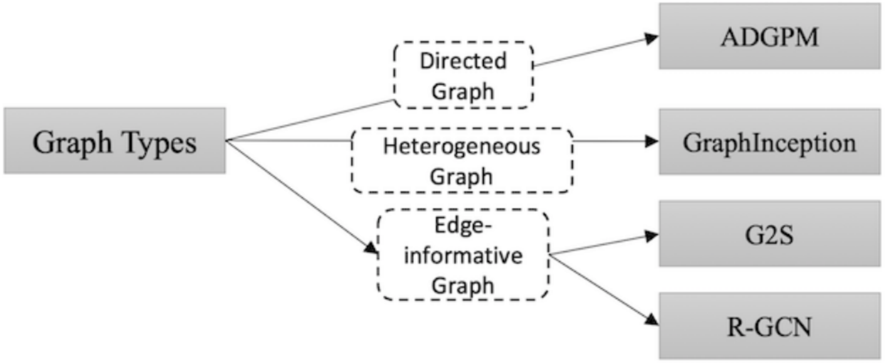


Graphs with Edge Information

- e.g. Drug interaction data with different types of interactions (edges). Each edge also has its information like the weight or the type of the edge.
- Can use different types of weight matrices



Graph Types



Graph Types
Directed Graph
Heterogeneous Graph
Edge-informative Graph
ADGPM
GraphInception
G2S
R-GCN

Outline

1. Introduction
2. Graph Neural Networks
3. Graph Variants
4. Objective Variants
5. Propagation Variants
   - Convolution
     - Spectral
     - Spatial
   - Gating
   - Attention
   - Skip Connections
6. Training Variants
7. Applications and Datasets

Objectives

- **Node-level outputs**: prediction for each node in the graph (e.g. classify each person in a social network)
- **Edge-level outputs**: prediction for each edge in the graph (e.g. classify each friendship in a social network)
- **Graph-level outputs**; prediction on the graph as a whole (e.g. classify an entire group in a social network)

Objective Frameworks

- **Supervised learning for node, edge, or graph level classification**: Given full labeled networks, predict target objective
- **Semi-supervised learning for node or edge level classification**: Given a single network with partial nodes being labeled and others remaining unlabeled
- **Unsupervised learning for graph embedding**: When no class labels are available in graphs, we can learn the graph embedding in a purely unsupervised way

Outline

1. Introduction
2. Graph Neural Networks
3. Graph Variants
4. Objective Variants
5. Propagation Variants
   - Convolution
     - Spectral
     - Spatial
   - Gating
   - Attention
   - Skip Connections
6. Training Variants
7. Applications and Datasets

Propagation Variants

- The propagation (update) step and output step are the crucial components to obtain the hidden states of nodes (or edges).

$$\mathbf{H} = f(\mathbf{H}, \mathbf{X})$$

- Variants utilize different aggregators to gather information from each node's neighbors and updaters to update nodes' hidden states

Outline

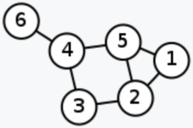
1. Introduction
2. Graph Neural Networks
3. Graph Variants
4. Objective Variants
5. Propagation Variants
   - Convolution
     - Spectral
     - Spatial
   - Gating
   - Attention
   - Skip Connections
6. Training Variants
7. Applications and Datasets

Convolution

Advances in this direction are often categorized into either spectral or spatial approaches

Spectral Convolution Methods

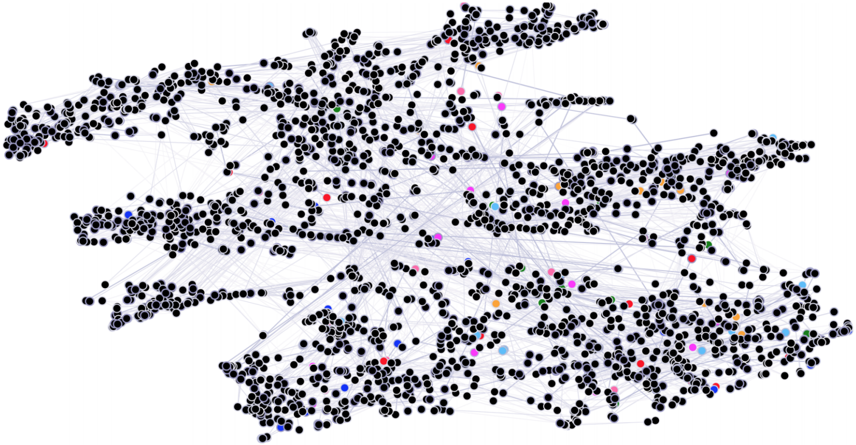
- Define a convolutional operation by operating on the graph in the spectral domain, leveraging the convolution theorem
- These approaches utilise the graph Laplacian matrix, L, defined as $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D}$ is the degree matrix (diagonal matrix with $\mathbf{D}_{ii} = \deg(i)$) and $\mathbf{A}$ is the adjacency matrix.

Labeled graph	Degree matrix	Adjacency matrix	Laplacian matrix
	$\begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 2 & -1 & 0 & 0 & -1 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 3 & -1 & -1 \\ -1 & -1 & 0 & -1 & 3 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 \end{pmatrix}$

Limitations of Spectral Methods

- Poor generalization to new/different graphs
- Graphs with variable size: spectral techniques work with fixed size graphs.
- Directed graphs: definition of directed graph Laplacian is unclear.

Transductive Learning
(from Petar Velickovic)

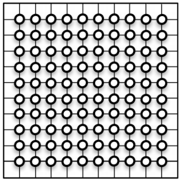


Training algorithm sees all features (including test nodes)

Inductive Learning
(from Petar Velickovic)

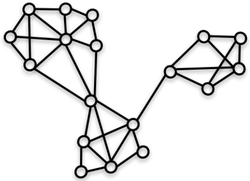
- Algorithm does not have access to all nodes upfront. Two variations:
  1. Test nodes are (incrementally) inserted into training graphs
  2. Test graphs are disjoint and completely unseen
- Requires generalizing across arbitrary graph structures, thus many transductive methods are useless

Spectral vs Spatial



Euclidean
space/grid

Standard
ConvNets



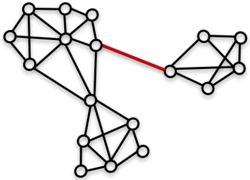
Non-Euclidean
space/graph

Fixed domain

Spectral graph
ConvNets

Spectral NNs offer rich
families of spectral filters

$\Rightarrow$



Change one single
edge

Variable domain

Can we still use
spectral graph
ConvNets?

$\Rightarrow$

Are spectral filters
transferable?

Spatial Convolution Methods

- Spatial approaches define convolutions directly on the graph, operating on spatially close neighbors
- Major challenge: defining convolution with different sized neighborhoods and maintaining the local invariance of CNNs

Spatial Convolution Methods

Duvenaud GCN - Duvenaud et. al., 2015

- Different weight matrices are used for nodes with different degrees,

$$\begin{aligned} \mathbf{x} &= \mathbf{h}_v + \sum_{i=1}^{\mathcal{N}_v} \mathbf{h}_i \\ \mathbf{h}'_v &= \sigma\left(\mathbf{x}\mathbf{W}_L^{\mathcal{N}_v}\right) \end{aligned} \tag{8}$$

  where $\mathbf{W}_{\mathbf{L}}^{\mathcal{N}_{\mathbf{v}}}$ is the weight matrix for nodes with degree $\mathcal{N}_v$ at layer L
- **Problem**: doesn't scale to graphs with very wide degree distributions

Spatial Convolution Methods

Diffusion-convolutional neural networks (DCNNs) - Atwood et. al.,2016

- Transition matrices are used to define the neighborhood for nodes in DCNN. For node classification, it has

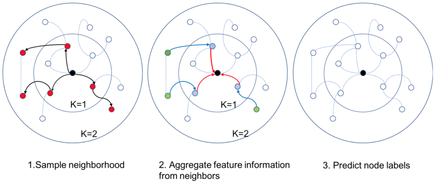
$$\mathbf{H} = f\left(\mathbf{W}^c \odot \mathbf{P}^*\mathbf{X}\right) \tag{9}$$

- $\mathbf{X}$ is an $N \times F$ matrix of input features (N is the number of nodes and F is the number of features)
- $\mathbf{P}^*$ is an $N \times K \times N$ tensor which contains the power series $\{\mathbf{P}, \mathbf{P}^2$, ..., $\mathbf{P}^K\}$ of matrix $\mathbf{P}$ (where $\mathbf{P}$ is the degree- normalized transition matrix from the adjacency matrix $\mathbf{A}$)
  - $\mathbf{P}^*$ transforms each node into a $K \times F$ diffusion convolutional representation

Spatial Convolution Methods
GraphSAGE - Hamilton et. al.,2017

- General inductive framework
- Restricts every degree to be the same (by sampling a fixed-size set a node's local neighborhood, during both training and inference)

$$\begin{aligned} \mathbf{h}_{\mathcal{N}_v}^t &= \mathrm{AGGREGATE}_t\left(\{\mathbf{h}_u^{t-1}, \forall u \in \mathcal{N}_v\}\right) \\ \mathbf{h}_v^t &= \sigma\left(\mathbf{W}^t \cdot [\mathbf{h}_v^{t-1} \| \mathbf{h}_{\mathcal{N}_v}^t]\right) \end{aligned} \tag{10}$$

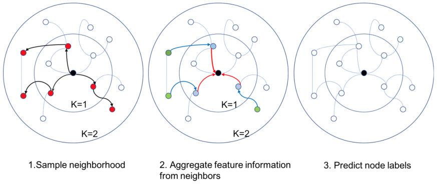


Spatial Convolution Methods

GraphSAGE - Hamilton et. al.,2017

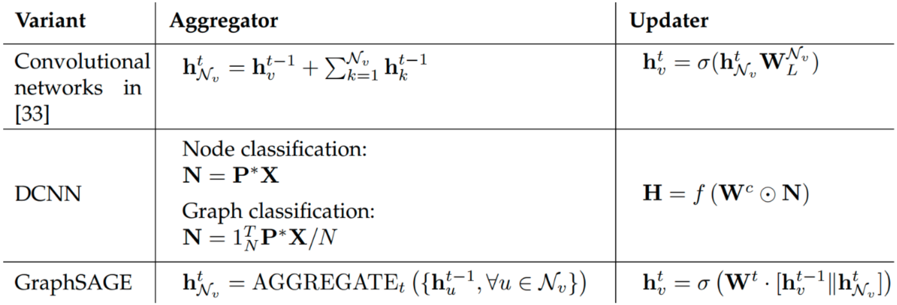
- General inductive framework
- Restricts every degree to be the same (by sampling a fixed-size set a node's local neighborhood, during both training and inference)

$$\begin{aligned} \mathbf{h}^t_{\mathcal{N}_v} &= \mathrm{AGGREGATE}_t\left(\{\mathbf{h}^{t-1}_u, \forall u \in \mathcal{N}_v\}\right) \\ \mathbf{h}^t_v &= \sigma\left(\mathbf{W}^t \cdot [\mathbf{h}^{t-1}_v \| \mathbf{h}^t_{\mathcal{N}_v}]\right) \end{aligned} \tag{10}$$



- Key is that $\mathbf{W}^t$ is not shared across time steps T

Spatial Methods: Summary



Variant	Aggregator	Updater
Convolutional networks in [33]	$\mathbf{h}_{\mathcal{N}_v}^t = \mathbf{h}_v^{t-1} + \sum_{k=1}^{\mathcal{N}_v} \mathbf{h}_k^{t-1}$	$\mathbf{h}_v^t = \sigma(\mathbf{h}_{\mathcal{N}_v}^t \mathbf{W}_L^{\mathcal{N}_v})$
DCNN	Node classification: $\mathbf{N} = \mathbf{P}^* \mathbf{X}$ Graph classification: $\mathbf{N} = 1_N^T \mathbf{P}^* \mathbf{X}/N$	$\mathbf{H} = f\left(\mathbf{W}^c \odot \mathbf{N}\right)$
GraphSAGE	$\mathbf{h}_{\mathcal{N}_v}^t = \text{AGGREGATE}_t\left(\{\mathbf{h}_u^{t-1}, \forall u \in \mathcal{N}_v\}\right)$	$\mathbf{h}_v^t = \sigma\left(\mathbf{W}^t \cdot [\mathbf{h}_v^{t-1} \Vert \mathbf{h}_{\mathcal{N}_v}^t]\right)$

Outline

1. Introduction
2. Graph Neural Networks
3. Graph Variants
4. Objective Variants
5. Propagation Variants
   - Convolution
     - Spectral
     - Spatial
   - Gating
   - Attention
   - Skip Connections
6. Training Variants
7. Applications and Datasets

Gating

Gated Graph Neural Networks (GGNNs) - Li et al. 2016

- Extension of original GNN (Scarselli et. al. 2009)
- Propagate for T steps, but do not restrict the propagation model to be contractive
    - Use gating in the propagation step to alleviate gradient issues

Gating

Initial idea: update $\mathbf{h}_v^t$ using neighborhood summation and tanh

$$\mathbf{a}_v^t = \mathbf{b} + \sum_{j \in \mathcal{N}_v} \mathbf{h}_j^{t-1} \tag{11}$$

$$\mathbf{h}_v^t = \tanh(\mathbf{W}\mathbf{a}_v^t) \tag{12}$$

Now, extend this to incorporate gating mechanisms, to prevent full overwrite of $\mathbf{h}_v^{t-1}$ by $\mathbf{h}_v^t$

Gating

Gated Graph Neural Networks (GGNNs) - Li et al. 2016

Gating mechanism defined similar to LSTM:

$$\mathbf{a}_v^t = \mathbf{b} + \sum_{j \in \mathcal{N}_v} \mathbf{h}_j^{t-1} \tag{13}$$

$$\mathbf{z}_v^t = \sigma\left(\mathbf{W}^z \mathbf{a}_v^t + \mathbf{U}^z \mathbf{h}_v^{t-1}\right) \tag{14}$$

$$\mathbf{r}_v^t = \sigma\left(\mathbf{W}^r \mathbf{a}_v^t + \mathbf{U}^r \mathbf{h}_v^{t-1}\right) \tag{15}$$

$$\widetilde{\mathbf{h}_v^t} = \tanh\left(\mathbf{W}\mathbf{a}_v^t + \mathbf{U}\left(\mathbf{r}_v^t \odot \mathbf{h}_v^{t-1}\right)\right) \tag{16}$$

$$\mathbf{h}_v^t = \left(1 - \mathbf{z}_v^t\right) \odot \mathbf{h}_v^{t-1} + \mathbf{z}_v^t \odot \widetilde{\mathbf{h}_v^t} \tag{17}$$

Where $\odot$ is element-wise multiplication, and $\mathbf{z}_v$ and $\mathbf{r}_v$ are the update and reset vectors, respectively

Outline

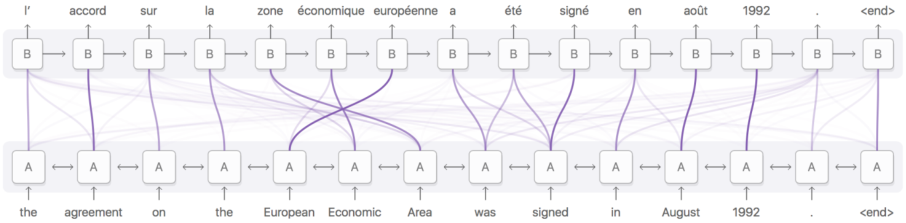
1. Introduction
2. Graph Neural Networks
3. Graph Variants
4. Objective Variants
5. Propagation Variants
   - Convolution
     - Spectral
     - Spatial
   - Gating
   - Attention
   - Skip Connections
6. Training Variants
7. Applications and Datasets

Attention

Graph attention network (GAT) - Velickovic et. al., 2017

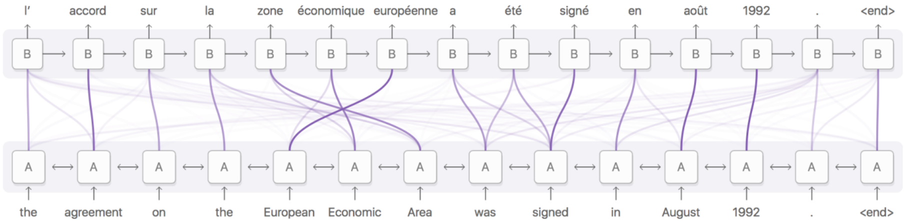
- Gating mechanisms are designed for data that changes sequentially; however, our graphs have static features
- GAT incorporates attention mechanisms into the propagation step

Attention



l'
accord
sur
la
zone
économique
européenne
a
été
signé
en
août
1992
.
<end>
B
A
the
agreement
on
the
European
Economic
Area
was
signed
in
August
1992
.
<end>

Attention



GAT uses a *self-attention* mechanism to compute the hidden states of each node by attending over itself and its neighbors

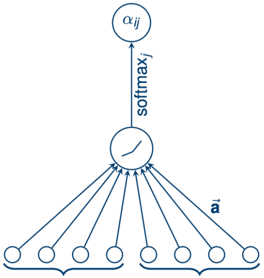
Attention

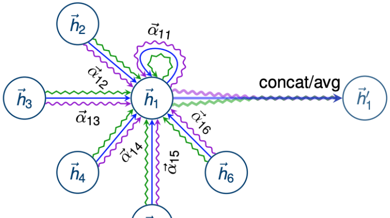
Graph attention network (GAT) - Velickovic et. al., 2017

$$e_{vj} = a(\mathbf{W}^a \mathbf{h}_v, \mathbf{W}^b \mathbf{h}_j) \tag{18}$$

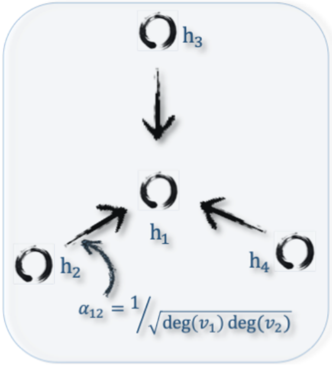
$$\alpha_{vj} = \frac{\exp(e_{vj})}{\sum_{k \in \mathcal{N}_v} \exp(e_{ik})} \tag{19}$$

$$\mathbf{h}_v = \sigma\left(\sum_{j \in \mathcal{N}_v} \alpha_{vj} \mathbf{W}^z \mathbf{h}_j \right) \tag{20}$$

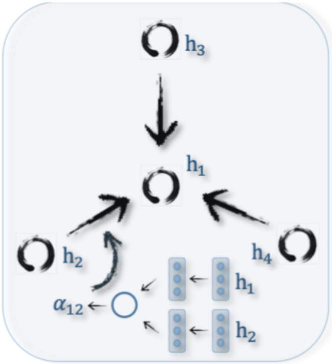




Attention

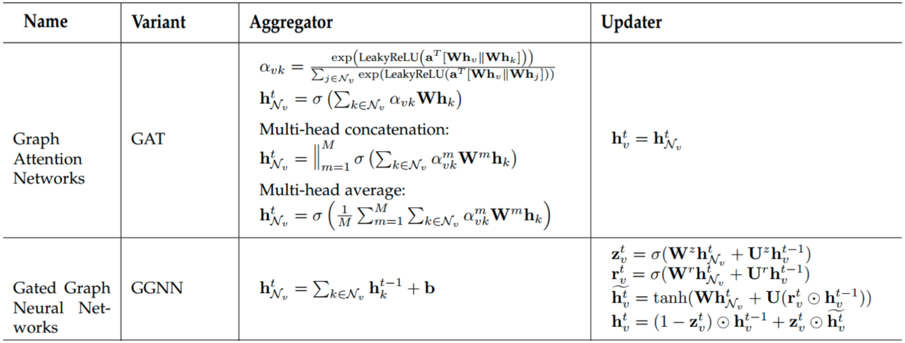


(a) Graph Convolution Networks [14] explicitly assign a non-parametric weight $a_{ij} = \frac{1}{\sqrt{deg(v_i)deg(v_j)}}$ to the neighbor v_j of v_i during the aggregation process.



(b) Graph Attention Networks [15] implicitly capture the weight a_{ij} via an end to end neural network architecture, so that more important nodes receive larger weights.

Gating and Attention



Name	Variant	Aggregator	Updater
Graph Attention Networks	GAT	$\alpha_{vk} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^T[\mathbf{W}\mathbf{h}_v \Vert \mathbf{W}\mathbf{h}_k]\right)\right)}{\sum_{j\in\mathcal{N}_v}\exp(\text{LeakyReLU}(\mathbf{a}^T[\mathbf{W}\mathbf{h}_v \Vert \mathbf{W}\mathbf{h}_j]))}$ $\mathbf{h}^t_{\mathcal{N}_v} = \sigma\left(\sum_{k\in\mathcal{N}_v}\alpha_{vk}\mathbf{W}\mathbf{h}_k\right)$ Multi-head concatenation: $\mathbf{h}^t_{\mathcal{N}_v} = \Vert_{m=1}^{M}\sigma\left(\sum_{k\in\mathcal{N}_v}\alpha^m_{vk}\mathbf{W}^m\mathbf{h}_k\right)$ Multi-head average: $\mathbf{h}^t_{\mathcal{N}_v} = \sigma\left(\frac{1}{M}\sum_{m=1}^{M}\sum_{k\in\mathcal{N}_v}\alpha^m_{vk}\mathbf{W}^m\mathbf{h}_k\right)$	$\mathbf{h}^t_v = \mathbf{h}^t_{\mathcal{N}_v}$
Gated Graph Neural Networks	GGNN	$\mathbf{h}^t_{\mathcal{N}_v} = \sum_{k\in\mathcal{N}_v}\mathbf{h}^{t-1}_k + \mathbf{b}$	$\mathbf{z}^t_v = \sigma(\mathbf{W}^z\mathbf{h}^t_{\mathcal{N}_v} + \mathbf{U}^z\mathbf{h}^{t-1}_v)$ $\mathbf{r}^t_v = \sigma(\mathbf{W}^r\mathbf{h}^t_{\mathcal{N}_v} + \mathbf{U}^r\mathbf{h}^{t-1}_v)$ $\widetilde{\mathbf{h}}^t_v = \tanh(\mathbf{W}\mathbf{h}^t_{\mathcal{N}_v} + \mathbf{U}(\mathbf{r}^t_v \odot \mathbf{h}^{t-1}_v))$ $\mathbf{h}^t_v = (1-\mathbf{z}^t_v)\odot\mathbf{h}^{t-1}_v + \mathbf{z}^t_v \odot \widetilde{\mathbf{h}}^t_v$

Outline

1 Introduction

2 Graph Neural Networks

3 Graph Variants

4 Objective Variants

5 Propagation Variants
- Convolution
  - Spectral
  - Spatial
- Gating
- Attention
- Skip Connections

6 Training Variants

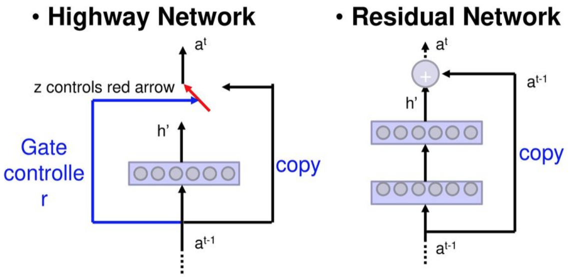
7 Applications and Datasets

Skip Connections

Highway GCN - Rahimi et. al.,2018

Uses uses layer-wise gates. The output of a layer is summed with its input with gating weights (inspired by Highway nets)

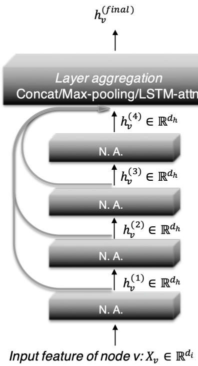
$$\begin{aligned} \mathbf{T}(\mathbf{h}^t) &= \sigma\left(\mathbf{W}^t\mathbf{h}^t + \mathbf{b}^t\right) \\ \mathbf{h}^{t+1} &= \mathbf{h}^{t+1} \odot \mathbf{T}(\mathbf{h}^t) + \mathbf{h}^t \odot (1 - \mathbf{T}(\mathbf{h}^t)) \end{aligned} \tag{21}$$



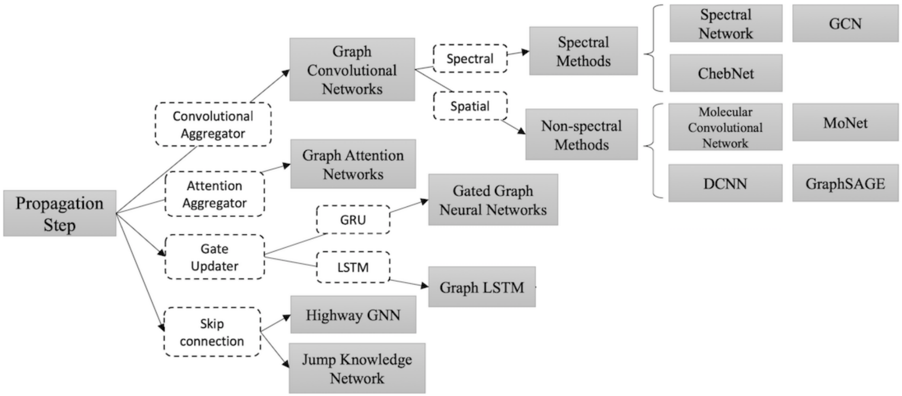
Skip Connections

Jump Knowledge Network (JKN) - Xu et. al., 2018

- With neighborhood aggregation, the receptive field of each node grows exponentially w.r.t. the number of layers (steps) T
- The Jump Knowledge Network selects from all of the intermediate representations for each node at the last layer
  - Allows the model adapt the effective neighborhood size for each node as needed


$h_v^{(final)}$
Layer aggregation
Concat/Max-pooling/LSTM-attn
$h_v^{(4)} \in \mathbb{R}^{d_h}$
N. A.
$h_v^{(3)} \in \mathbb{R}^{d_h}$
N. A.
$h_v^{(2)} \in \mathbb{R}^{d_h}$
N. A.
$h_v^{(1)} \in \mathbb{R}^{d_h}$
N. A.
Input feature of node v: $X_v \in \mathbb{R}^{d_i}$

Propagation Methods: Summary



Outline

1 Introduction

2 Graph Neural Networks

3 Graph Variants

4 Objective Variants

5 Propagation Variants
- Convolution
  - Spectral
  - Spatial
- Gating
- Attention
- Skip Connections

6 Training Variants

7 Applications and Datasets

Training Methods

- GraphSAGE solved the problems of the original GCN by replacing full graph Laplacian with learnable aggregation functions, which are key to generalize to unseen nodes.
- In addition, GraphSAGE uses neighbor sampling to alleviate receptive field expansion

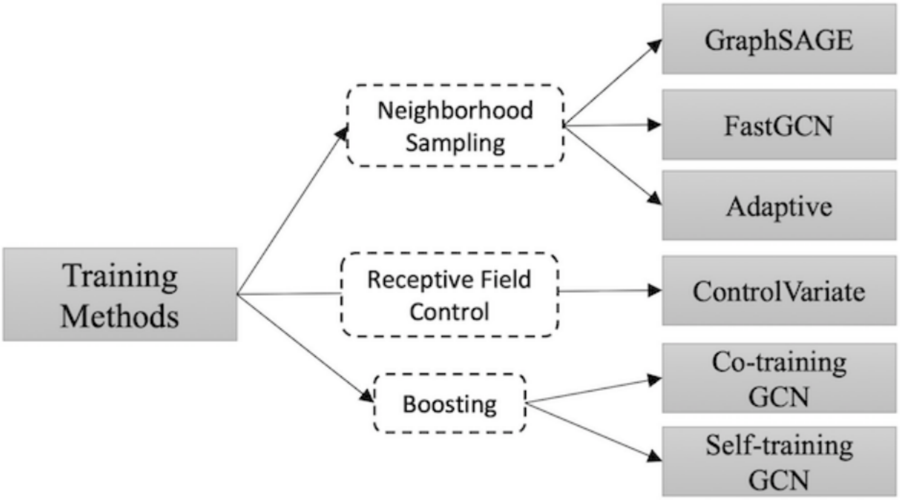
Training Methods

- Chen et. al (2018) proposed a control-variate based stochastic approximation algorithms by utilizing the historical activations of nodes as a control variate.
- This limits the receptive field in the 1-hop neighborhood, but is efficient

Training Methods

- Li et. al. (2018) note that GNNs requires many additional labeled data for validation and also suffers from the localized nature of the convolutional filter
- To solve the limitations, the authors propose a method to find the nearest neighbors of training data and a boosting-like method

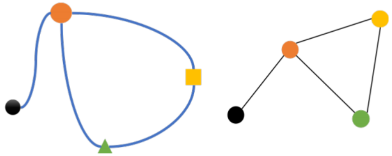
Training Methods



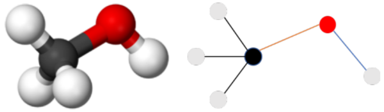
Outline

1. Introduction
2. Graph Neural Networks
3. Graph Variants
4. Objective Variants
5. Propagation Variants
   - Convolution
     - Spectral
     - Spatial
   - Gating
   - Attention
   - Skip Connections
6. Training Variants
7. Applications and Datasets

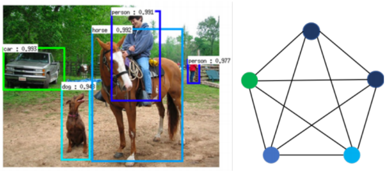
Application Areas



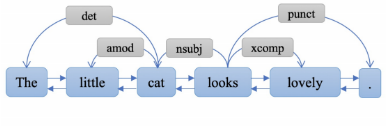
(a) physics



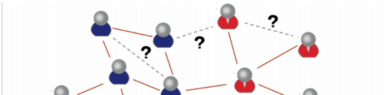
(b) molecule

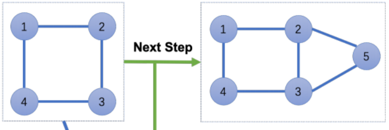


(c) image

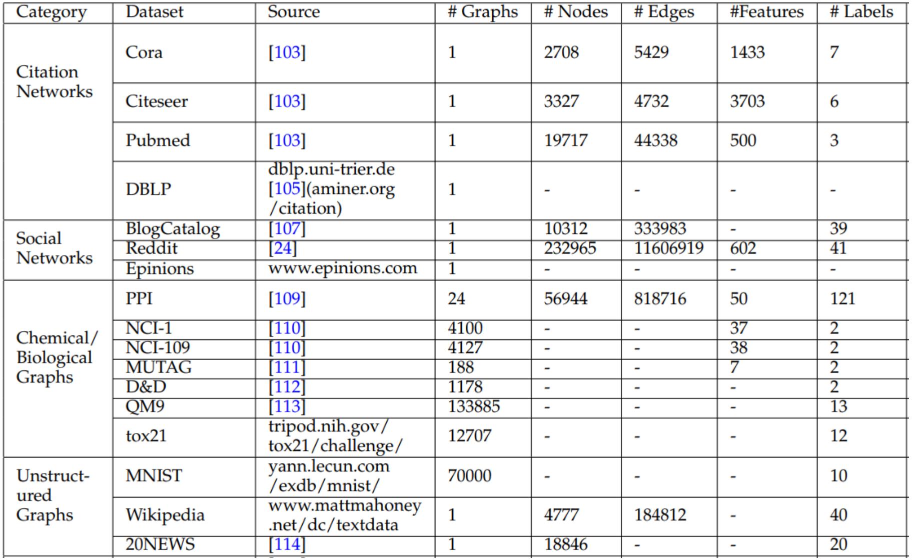


(d) text





Commonly Used Datasets



Category	Dataset	Source	# Graphs	# Nodes	# Edges	#Features	# Labels
Citation Networks	Cora	[103]	1	2708	5429	1433	7
	Citeseer	[103]	1	3327	4732	3703	6
	Pubmed	[103]	1	19717	44338	500	3
	DBLP	dblp.uni-trier.de [105](aminer.org /citation)	1	-	-	-	-
Social Networks	BlogCatalog	[107]	1	10312	333983	-	39
	Reddit	[24]	1	232965	11606919	602	41
	Epinions	www.epinions.com	1	-	-	-	-
Chemical/ Biological Graphs	PPI	[109]	24	56944	818716	50	121
	NCI-1	[110]	4100	-	-	37	2
	NCI-109	[110]	4127	-	-	38	2
	MUTAG	[111]	188	-	-	7	2
	D&D	[112]	1178	-	-	-	2
	QM9	[113]	133885	-	-	-	13
	tox21	tripod.nih.gov/ tox21/challenge/	12707	-	-	-	12
Unstruct-ured Graphs	MNIST	yann.lecun.com /exdb/mnist/	70000	-	-	-	10
	Wikipedia	www.mattmahoney .net/dc/textdata	1	4777	184812	-	40
	20NEWS	[114]	1	18846	-	-	20