Comparative Performance Analysis on Different Approaches of Network Embedding

Kishalay Das Paarth Gupta Kapil Pathak Aalo Majumdar

Indian Institute Of Science

April 27, 2019

Overview

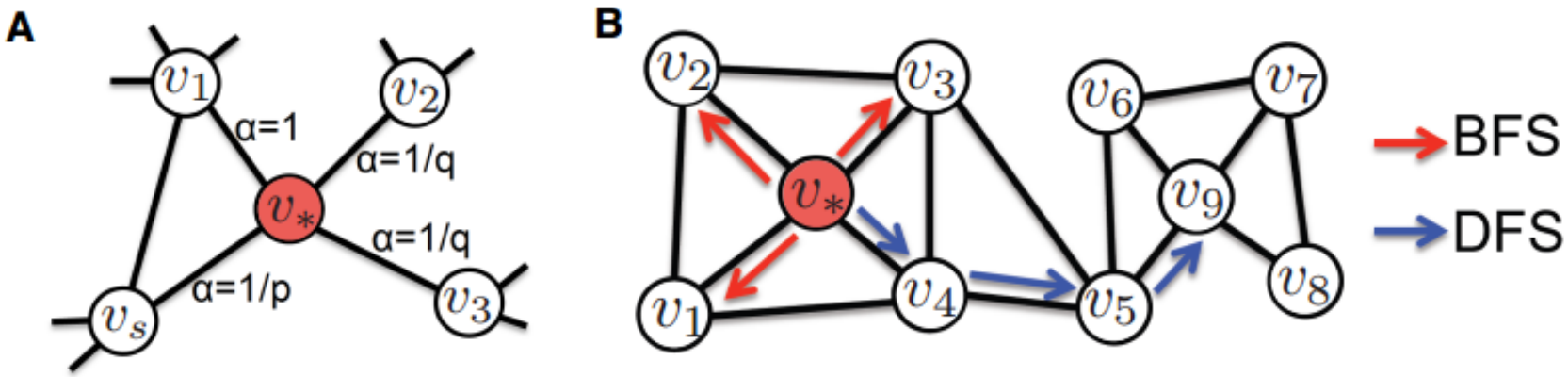
Introduction

- Machine learning on graphs is an crucial task with applications ranging from drug design to friendship recommendation in social networks.
- **Primary Challenge :** Finding a way to represent,or encode, graph structure in lower dimension.
- Recent Literature using techniques based on deep learning and nonlinear dimensionality reduction.
- We tried to make a survey of different approaches used to encode the graph structure into embedding and made a comparative study of their performances.

Random Walk Approaches - Deep Walk

- Based on word2vec
- Random walk generator
  - Sample a random vertex - random walk - sentences
- Objective function
  - minimise $-logPr[(v_{i-w}, .., v_{i-1}, v_{i+1}, .., v_{i+w})|\Phi(v_i)]$
  - Maximise log likelihood of context of vertex given its latent representation
- SkipGram model for gradient descent

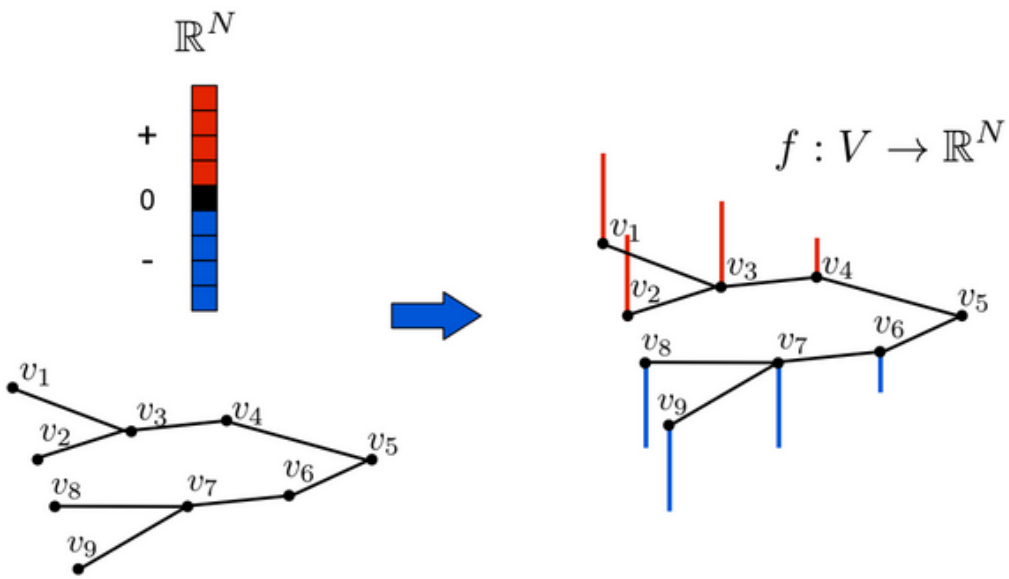
Random Walk approaches - node2vec



- Flexiblity in random walk
- 2 random walk hyperparams: p and q
- Return parameter p controls likelihood of revisiting a node on random walk
  - High value $\Rightarrow$ less likely
- In-out parameter q controls likelihood of revisiting a nodes one-hop neighborhood
  - $q > 1 \Rightarrow$ random walk is biased towards nodes closer to node
  - $q < 1 \Rightarrow$ biased towards nodes further away
- Smooth interpolation between BFS-like (community structures) and DFS-like walks (local structural roles)

Graph Fourier Transform

Unnormalised Laplacian	$L = D - W \in R^{n \times n}$
Normalised Laplacian	$L = I_n - D^{-1/2} W D^{-1/2}$
Eigendecomposition of L	$L = U \Lambda U^T$
Graph Fourier modes	$U = [u_0, ..., u_{n-1}] \in R^{n \times n}$
Frequencies of graph	$\Lambda = [\lambda_0, ..., \lambda_{n-1}] \in R^{n \times n}$
Fourier transform of signal	$\hat{x} = U^T x \in R^n$
Inverse transform	$x = U \hat{x}$



Spectral filtering in Fourier domain

- Convolution operator in Fourier domain: $x *_G g = F(x).F(g)$
- Spectral filtering:
  $x_{out} = \underbrace{U}_{\text{inverse transform}} \underbrace{g(\Lambda)}_{\text{spectral response}} \underbrace{U^T x_{in}}_{\text{GFT}} = g(L)x_{in}$
- Polynomial filters: $g_\theta(\lambda) = \sum_{k=0}^{K-1} \theta_k \Lambda^k$, where $\theta \in R^K$ is a vector of polynomial coeffecients
- Disadvantage: $O(n^2)$
- Solution: Use recursion of Chebyshev polynomials
- $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$ with $T_0 = 1$ and $T_1 = x$
- Compute scaled Chebyshev coeffecients i.e. $\Lambda = 2\Lambda/\lambda_{max} - I_n$, such that eigenvalues lie in [-1,1]
- $g_\theta(\lambda) = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{\Lambda})$
- K-localised filter with $O(KE)$ complexity

Graph Convolution Network

- Why Graphs :- Graphs are a general language for describing and modeling complex systems.
- How to represent graphs :- Embeddings
- Node embedding :- Map nodes to low-dimensional embeddings.
- Given a network/graph G=(V, E, W), where V is the set of nodes, E is the set of edges between the nodes, and W is the set of weights of the edges, the goal of node embedding is to represent each node i with a vector , which preserves the structure of networks.
- Belief :- Nodes have similar embeddings tend to co-occur on short random walks over graph

GCN:WHY are they Special

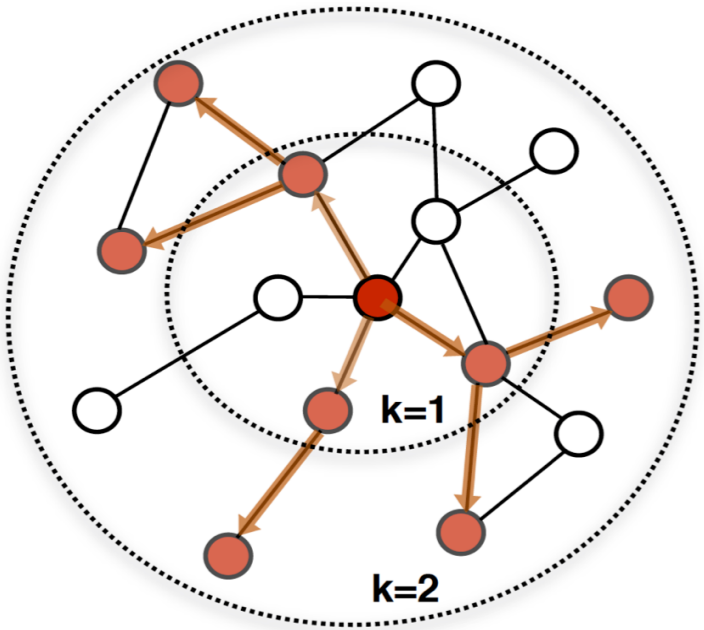
- Graph neural networks :- Deep learning architectures for graph structured data.
- Neighborhood Aggregation
- Intuition: Nodes aggregate information from their neighbors using neural networks. Network neighborhood defines a computation graph.
- GCNs are a slight variation on the neighborhood aggregation idea.
- Each Convolutional layer captures the next hop information in the network.
- Usually 2-3 layers deep.

GraphSAGE

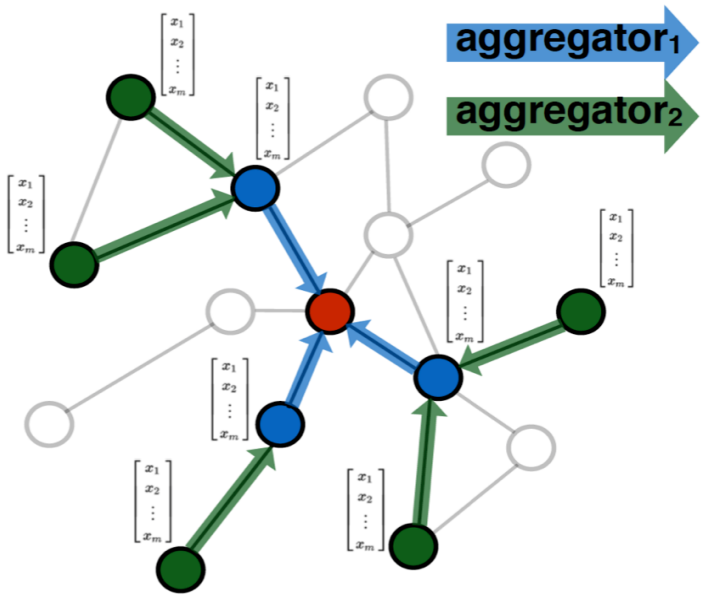
- Graph SAmpling and aggreGatE
- Adaptation of GCN Idea to Inductive Embedding
- Leverages node feature information to efficiently generate node embeddings functions that generalizes to unseen nodes.
- Learn a function that generates embeddings by sampling and aggregating features from a node's local neighborhood upto certain layers(search depth).
- During test, we use our trained system to generate embeddings for entirely unseen nodes by applying the learned aggregation functions.

GraphSAGE:Motivation

Basic Idea : Nodes neighborhood defines a computation graph.



1. Sample neighborhood



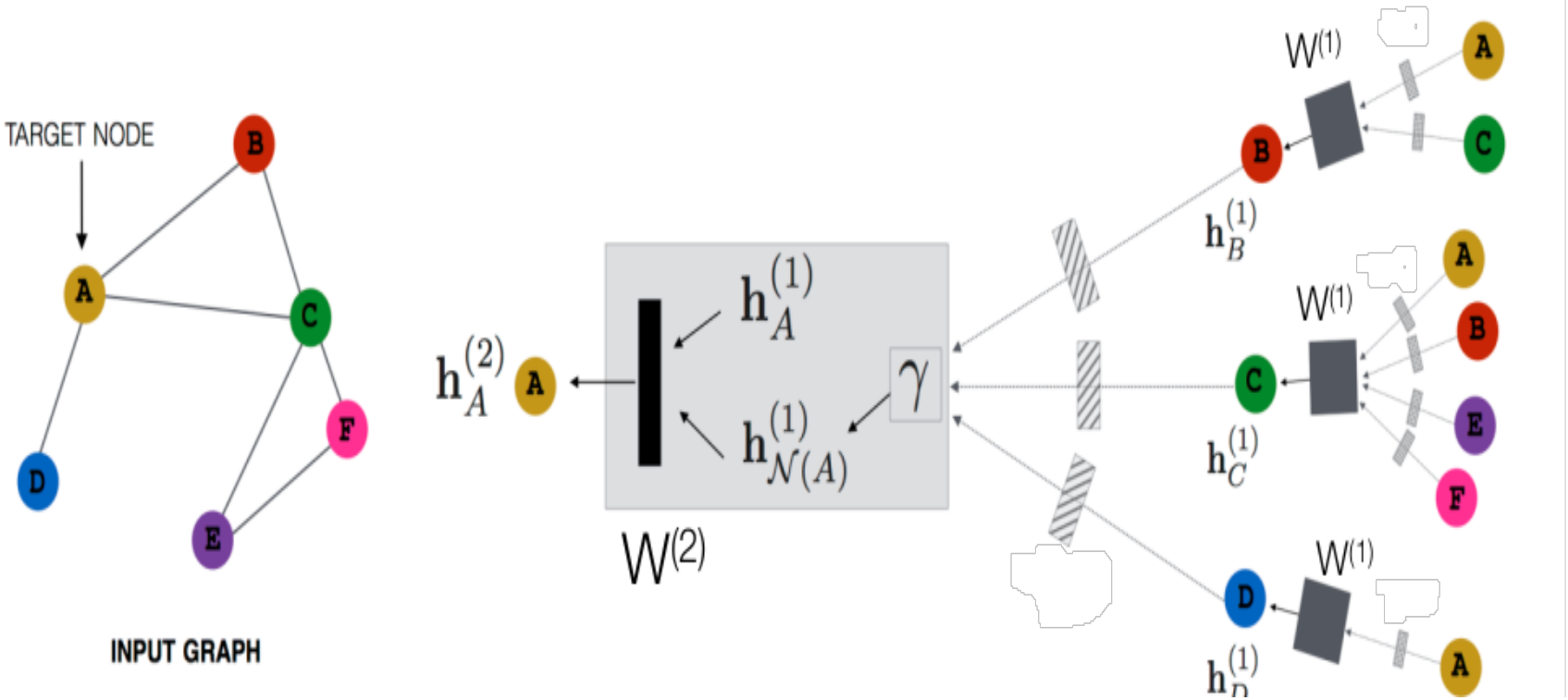
2. Aggregate feature information from neighbors

Learn how to propagate information across the graph to compute node features

GraphSAGE:Algorithm

$\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$;
for $k = 1...K$ **do**
  for $v \in \mathcal{V}$ **do**
    $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;
    $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$
  end
  $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$
end
$\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$

GraphSAGE:Example



Graph Attention Network

- Attention mechanism have become de *defacto* standard in many sequence -based tasks.
- One of benefits of attention mechanisms is that they allow for dealing with variable sized inputs, focusing on the most relevant parts of the input to make a decision.
- Introducing the same concept in graphs, this architecture computes the hidden representations of each node in the graph by attending over its neighbours.
- This architecture doesn't take the number of neighbours into consideration.

Graph Attention Layer

- Let input to the layer be $\mathbf{h} = \overrightarrow{h_1}, \overrightarrow{h_2}, .., \overrightarrow{h_N}, \overrightarrow{h_i} \in \mathbb{R}^F$ where N is the number of nodes and F is the number of features in the node
- The layer produces a new set of node features

$$\mathbf{h} = \overrightarrow{h'_1}, \overrightarrow{h'_2}, .., \overrightarrow{h'_N}, \overrightarrow{h'_i} \in \mathbb{R}^{F'}$$

- A shared attention mechanism $a : \mathbb{R}^{F'} \times \mathbb{R}^{F'} \rightarrow \mathbb{R}$ computes attention coefficients

$$\alpha_{ij} = \frac{exp(LeakyReLU(\overrightarrow{a^T}[W\overrightarrow{h_i}||W\overrightarrow{h_j}]))}{\sum_{k \in N_i} exp(LeakyReLU(\overrightarrow{a^T}[W\overrightarrow{h_i}||W\overrightarrow{h_k}]))}$$

- Finally the output of each layer $\overrightarrow{h'_i}$ can be calculated as

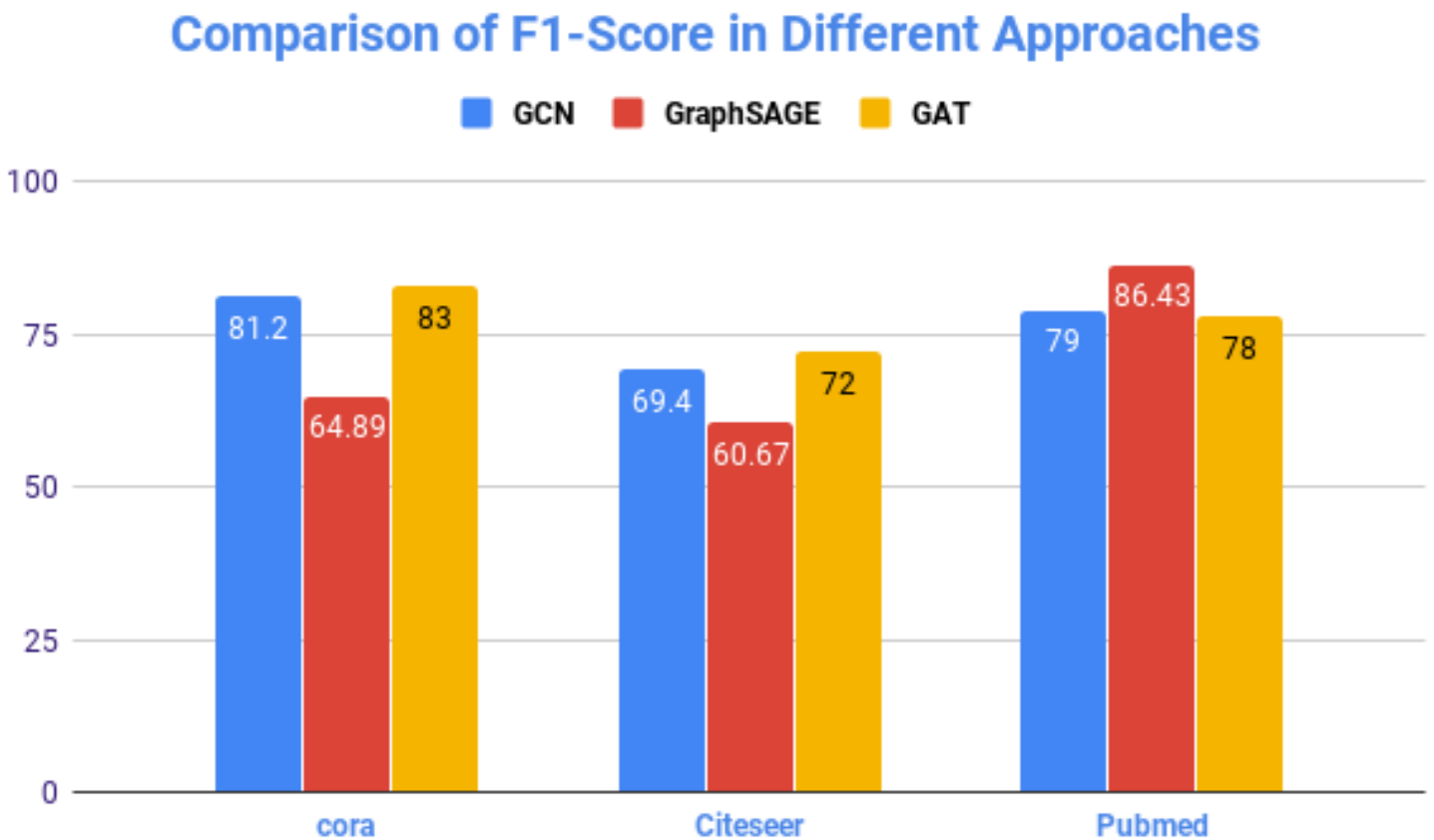
$$\overrightarrow{h'_i} = \sigma(\frac{1}{K}\sum_{k=1}^{K}\sum_{j \in N_i}\alpha_{ij}^k W^k \overrightarrow{h_j})$$

Advantages of Graph Attention Layer

- Computationally highly efficient: the operation of the self-attentional layer can be parallelized across all edges, and the computation of output features can be parallelized across all nodes.
- No eigendecompositions or similar costly matrix operations are required.
- As opposed to GCNs, our model allows for (implicitly) assigning different importances to nodes of a same neighborhood.
- Analyzing the learned attentional weights may lead to benefits in interpretability.

Experiment Results

Node Classification Task



Future Work

- Exploring more variants of GraphSAGE.
- Experimenting with Heterogenious and Dynamic Network
- Time comparisons between the different variants of GCN.
- Exploring other down-stream tasks like Anomaly Detection,Link Prediction,Community Detection.

Useful Resources

- Perozzi, B., Al-Rfou, R., and Skiena, S. Deepwalk: Onlinelearning of social representations.
- Grover, A. and Leskovec, J. Node2vec:Scalable featurelearning for networks.
- Kipf, T. N. and Welling, M.Semi-supervised classi-fication with graph convolutional networks.
- Hamilton, W., Ying, Z., and Leskovec, J. Inductive repre-sentation learning on large graphs.
- Velickovi c, P., Cucurull, G., Casanova, A., Romero, A.,Lio, P., and Bengio, Y. Graph attention networks.