Graph Convolutional Networks
A Tutorial

Naganand Y

Overview

- T. N. Kipf and M. Welling, Semi-supervised classification with graph convolutional networks

Overview

- T. N. Kipf and M. Welling, Semi-supervised classification with graph convolutional networks
- Important real-world datasets are in the form of graphs/networks

Overview

- T. N. Kipf and M. Welling, Semi-supervised classification with graph convolutional networks
- Important real-world datasets are in the form of graphs/networks
  - Knowledge graphs, Social Networks, World Wide Web, etc.

Overview

- T. N. Kipf and M. Welling, Semi-supervised classification with graph convolutional networks
- Important real-world datasets are in the form of graphs/networks
  - Knowledge graphs, Social Networks, World Wide Web, etc.
- A challenge is to generalize well-established neural networks to such structured datasets

Problem Statement

Graph $G = (V, E)$, $N = |V|$

Problem Statement

Graph $G = (V, E)$, $N = |V|$

- Inputs:
  - Adjacency matrix $A_{N \times N}$
  - A feature matrix $X_{N \times D}$ (A feature description x_i for every node i)

Problem Statement

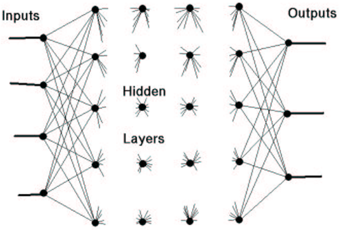
Graph $G = (V, E)$, $N = |V|$

- Inputs:
  - Adjacency matrix $A_{N \times N}$
  - A feature matrix $X_{N \times D}$ (A feature description x_i for every node i)
- Output:
  - A feature matrix $Z_{N \times F}$ (at the node-level)

Problem Statement

Graph $G = (V, E)$, $N = |V|$

- Inputs:
  - Adjacency matrix $A_{N \times N}$
  - A feature matrix $X_{N \times D}$ (A feature description x_i for every node i)
- Output:
  - A feature matrix $Z_{N \times F}$ (at the node-level)
- Neural Network Layers:
  - $H^{(0)} = X$
  - $H^{(l+1)} = f(H^{(l)}, A)$
  - $H^{(L)} = Z$



A Simple Neural Network

Propagation Rule: An Example

$$f(H^{(l)}, A) = \sigma\left(AH^{(l)}W^{(l)}\right)$$

Propagation Rule: An Example

weight matrix for the l-th layer

$$f(H^{(l)}, A) = \sigma\left(AH^{(l)}W^{(l)}\right)$$

Propagation Rule: An Example

non-linear activation function

weight matrix for the l-th layer

$$f(H^{(l)}, A) = \sigma\left(AH^{(l)}W^{(l)}\right)$$

Limitations of the Rule

$$f(H^{(l)}, A) = \sigma\left(AH^{(l)}W^{(l)}\right)$$

Limitations of the Rule

$$f(H^{(l)}, A) = \sigma\left(AH^{(l)}W^{(l)}\right)$$

- For every node i, we do not take into account the feature vector of i

Limitations of the Rule

$$f(H^{(l)}, A) = \sigma\left(AH^{(l)}W^{(l)}\right)$$

- For every node i, we do not take into account the feature vector of i
  - Enforce self-loops i.e. use $\hat{A} = A + \mathcal{I}$

Limitations of the Rule

$$f(H^{(l)}, A) = \sigma\left(AH^{(l)}W^{(l)}\right)$$

- For every node i, we do not take into account the feature vector of i
  - Enforce self-loops i.e. use $\hat{A} = A + \mathcal{I}$
- $AH^{(l)}$ will completely change the scale of the feature vectors

Limitations of the Rule

$$f(H^{(l)}, A) = \sigma\left(AH^{(l)}W^{(l)}\right)$$

- For every node i, we do not take into account the feature vector of i
  - Enforce self-loops i.e. use $\hat{A} = A + \mathcal{I}$
- $AH^{(l)}$ will completely change the scale of the feature vectors
  - Normalize A i.e. use $D^{-1}A$

Limitations of the Rule

$$f(H^{(l)}, A) = \sigma\left(AH^{(l)}W^{(l)}\right)$$

- For every node i, we do not take into account the feature vector of i
  - Enforce self-loops i.e. use $\hat{A} = A + \mathcal{I}$
- $AH^{(l)}$ will completely change the scale of the feature vectors
  - Normalize A i.e. use $D^{-1}A$
  - $D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ is more interesting

Limitations of the Rule

$$f(H^{(l)}, A) = \sigma\left(AH^{(l)}W^{(l)}\right)$$

- For every node i, we do not take into account the feature vector of i
  - Enforce self-loops i.e. use $\hat{A} = A + \mathcal{I}$
- $AH^{(l)}$ will completely change the scale of the feature vectors
  - Normalize A i.e. use $D^{-1}A$
  - $D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ is more interesting

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

Embedding the Karate Club Network

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

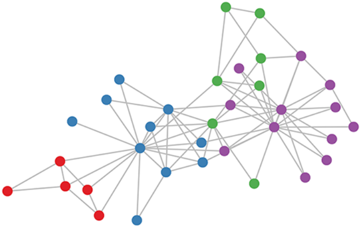


Figure : Colors denote communities of modularity-based clustering

Embedding the Karate Club Network

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

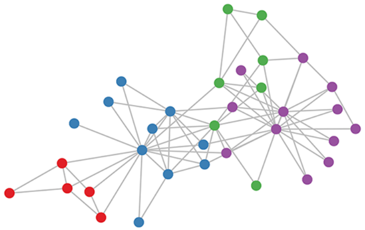


Figure : Colors denote communities of modularity-based clustering

- $L = 3$
- $X = \mathcal{I}$
- $W^{(l)}$ is random

Embedding the Karate Club Network

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

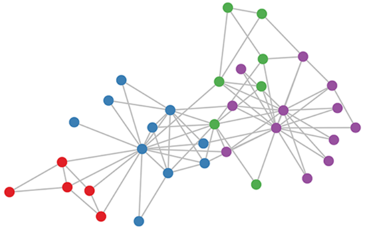


Figure : Colors denote communities of modularity-based clustering

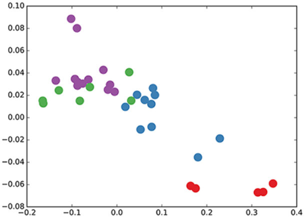


Figure : GCN embedding (with random weights)

Embedding the Karate Club Network

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

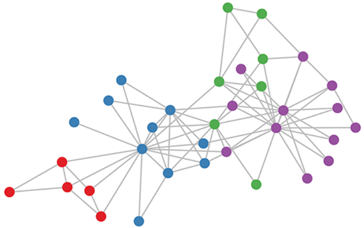


Figure : Colors denote communities of modularity-based clustering

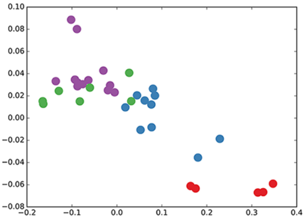


Figure : GCN embedding (with random weights)

- Embedding closely resembles the community structure
- No feature vectors!
- Weights are random and no training updates!

Semi-Supervised Node Classification

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

- Note everything in the model is differentiable and parameterized
- Add some labels (e.g. one label per community)
- Train the model to learn weights

Semi-Supervised Learning: A 2-layer Example

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

- $W^{(0)} \in \mathbb{R}^{D \times H}$ and $W^{(1)} \in \mathbb{R}^{H \times F}$

Semi-Supervised Learning: A 2-layer Example

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

- $W^{(0)} \in \mathbb{R}^{D \times H}$ and $W^{(1)} \in \mathbb{R}^{H \times F}$
- Minimize

$$\mathcal{L} = -\sum_{l \in \mathcal{Y}_L} \sum_{f=1}^{F} Y_{lf} \ln Z_{lf}$$

Semi-Supervised Learning: A 2-layer Example

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

- $W^{(0)} \in \mathbb{R}^{D \times H}$ and $W^{(1)} \in \mathbb{R}^{H \times F}$
- Minimize

$$\mathcal{L} = -\sum_{l \in \mathcal{Y}_L} \sum_{f=1}^{F} Y_{lf} \ln Z_{lf}$$

- Train $W^{(0)}$ and $W^{(1)}$ using gradient descent

Semi-Supervised Learning for the Karate Club Network

(Semi-Supervised Learning)

Experiments

- Citation networks

Experiments

- Citation networks
  - Citeseer, Cora and Pubmed

Experiments

- Citation networks
  - Citeseer, Cora and Pubmed
  - They contain sparse bag-of-words feature vectors for each document

Experiments

- Citation networks
  - Citeseer, Cora and Pubmed
  - They contain sparse bag-of-words feature vectors for each document
  - Citation links are treated as (undirected) edges to construct A

Experiments

- Citation networks
  - Citeseer, Cora and Pubmed
  - They contain sparse bag-of-words feature vectors for each document
  - Citation links are treated as (undirected) edges to construct A
- Knowledge Graph (NELL)

Experiments

- Citation networks
  - Citeseer, Cora and Pubmed
  - They contain sparse bag-of-words feature vectors for each document
  - Citation links are treated as (undirected) edges to construct A
- Knowledge Graph (NELL)
  - Each (e_1, r, e_2) is assigned separate (e_1, r_1) and (e_2, r_2)

Experiments

- Citation networks
  - Citeseer, Cora and Pubmed
  - They contain sparse bag-of-words feature vectors for each document
  - Citation links are treated as (undirected) edges to construct A
- Knowledge Graph (NELL)
  - Each (e_1, r, e_2) is assigned separate (e_1, r_1) and (e_2, r_2)
  - Entity nodes are described by sparse feature vectors

Experiments

- Citation networks
  - Citeseer, Cora and Pubmed
  - They contain sparse bag-of-words feature vectors for each document
  - Citation links are treated as (undirected) edges to construct A
- Knowledge Graph (NELL)
  - Each (e_1, r, e_2) is assigned separate (e_1, r_1) and (e_2, r_2)
  - Entity nodes are described by sparse feature vectors
  - Each relation node is assigned a unique one-hot feature vector

Experiments

- Citation networks
  - Citeseer, Cora and Pubmed
  - They contain sparse bag-of-words feature vectors for each document
  - Citation links are treated as (undirected) edges to construct A
- Knowledge Graph (NELL)
  - Each (e_1, r, e_2) is assigned separate (e_1, r_1) and (e_2, r_2)
  - Entity nodes are described by sparse feature vectors
  - Each relation node is assigned a unique one-hot feature vector
- The method outperforms related methods (upto 6% improvement)