GMNN: Graph Markov Neural Networks

Jian Tang
HEC Montreal
Mila-Quebec AI Institute
CIFAR AI Research Chair

Graphs are Ubiquitous

- A general and flexible data structure to encode the relations between objects

- Cover a variety of domains and applications
  - node classification
  - Link prediction
  - Information diffusion
  - ...
Semi-supervised Object Classification

• Given $G = (V, E, \mathbf{x}_V)$
  - $V = V_L \cup V_U$: objects/nodes
  - $E$: edges
  - $\mathbf{x}_V$: object features

• Give some labeled objects $V_L$, we want to infer the labels of the rest of objects $V_U$

• Many other tasks on graphs can be formulated as object classification
  - E.g., link classification
Related Work: Statistical Relational Learning

• Models the joint distribution of the object labels given the object features, i.e., $p(y_V|x_V)$ with conditional random fields

$$p(y_V|x_V) = \frac{1}{Z(x_V)} \sum_{(i,j) \in E} \psi_{i,j}(y_i, y_j, x_V)$$

$$\psi_{i,j}(y_i, y_j, x_V) = \exp \left( \sum_{k=1}^{K} \lambda_k f_k(y_i, y_j, x_i, x_j) + \mu_k g_k(y_i, x_i) \right)$$
Optimization of Statistical Relational Learning

- Learning by maximizing the likelihood of the observed labels
  \[ \log p(y_L|x_V) \]
- Inferring the posterior distributions \( p(y_U|y_L, x_V) \) with approximate methods such as loop belief propagation is used
Pros and Cons of Statistical Relation Learning

• Pros
  • Capable of modeling the dependency between the object labels

• Cons
  • Some manually defined potential functions
  • Limited model capacity
  • Inference is difficult due to the complicated graph structures
Related Work: Graph Neural Networks

• Learning effective node representations and then predicting the node labels independently
  • Graph convolutional Networks (Kipf et al. 2016)
  • Graph attention networks (Veličković et al. 2017)
  • Neural message passing (Gilmer et al. 2017)

• Predicting the node labels independently with the node representations
Graph Convolutional Networks
(Kipf et al. 2016)

• Iteratively update the node representations by aggregating the node representations of neighbors and its own node representations
  • Starting from the initial node features $H_0 = x_V$

\[
H_{(l+1)} = f(H_l, A) \\
\hat{A} = A + I \\
f(H_l, A) = \sigma(D^{-\frac{1}{2}} \hat{A}D^{-\frac{1}{2}}H_l W_l)
\]
Pros and Cons of Graph Neural Networks

• Pros
  • Learns effective node representations by feature propagations
  • High model compacity by using multiple non-linear graph convolutional layers

• Cons
  • Ignore the dependency between node labels
Can we combine the advantages of both worlds?
GMNN: Graph Markov Neural Networks (Qu, Bengio, and Tang, ICML’19)

- Towards combining statistical relational learning and graph neural networks
- Learning effective node representations for predicting the node labels
- Modeling the label dependencies of nodes
- State-of-the-art performance
  - semi-supervised node classification
  - unsupervised node representation
  - link classification
GMNN: Graph Markov Neural Networks

• Model the joint distribution of object labels $y_V$ conditioned on object attributes $x_V$, i.e., $p_\phi(y_V|x_V)$

• Learning the model parameters $\phi$ by maximizing the lower-bound of log-likelihood of the observed data, $\log p_\phi(y_L|x_V)$

$$\log p_\phi(y_L|x_V) \geq \mathbb{E}_{q_\theta(y_U|x_V)}[\log p_\phi(y_L, y_U|x_V) - \log q_\theta(y_U|x_V)]$$
Optimization with Pseudolikelihood
Variational-EM

• E-step: fix $p_\phi$ and update the variational distribution $q_\theta(y_U|x_V)$ to approximate the true posterior distribution $p_\phi(y_U|y_L,x_V)$.

• M-step: fix $q_\theta$ and update $p_\phi$ to maximize the lower bound

$$\ell(\phi) = \mathbb{E}_{q_\theta(y_U|x_V)}[\log p_\phi(y_L, y_U|x_V)]$$

• Directly optimize the joint likelihood is difficult due to the partition function in $p_\phi$, instead we optimize the pseudolikelihood function

$$\ell_{PL}(\phi) \triangleq \mathbb{E}_{q_\theta(y_U|x_V)}[\sum_{n \in V} \log p_\phi(y_n|y_{V\setminus n}, x_V)]$$

$$= \mathbb{E}_{q_\theta(y_U|x_V)}[\sum_{n \in V} \log p_\phi(y_n|y_{NB(n)}, x_V)]$$
Inference/E-step: approximate $p_\phi(y_U | y_L, x_V)$

• Approximate it with variational distribution $q_\theta(y_U | x_V)$. Specifically we use mean-field method:

$$q_\theta(y_U | x_V) = \prod_{n \in U} q_\theta(y_n | x_V)$$

• We parametrize each variational distribution with a Graph Neural Network

$$q_\theta(y_n | x_V) = \text{Cat}(y_n | \text{softmax}(W_\theta h_{\theta,n}))$$

Object representations learned by GNN
Inference/E-step: approximate $p_{\phi}(y_U | y_L, x_V)$

- The optimal variational distribution satisfies:

$$\log q_{\theta}(y_n | x_V) = E_{q_{\theta}(y_{NB(n)} \cap U | x_V)} [\log p_{\phi}(y_n | y_{NB(n)}, x_V)] + \text{const.}$$

- Estimate the right term by sampling from $q_{\theta}(y_{NB(n)} \cap U | x_V)$, and then we have

$$q_{\theta}(y_n | x_V) \approx p_{\phi}(y_n | \hat{y}_{NB(n)}, x_V)$$

Label distribution of object n by the learning module
Inference/E-step: approximate $p_\phi(y_U|y_L,x_V)$

- Minimize the KL-divergence between the two distributions
  - The supervision from the learning module is used as pseudo label to train the variational distribution
    \[
    O_{\theta,U} = \sum_{n \in U} \mathbb{E}_{p_\phi(y_n|\hat{y}_{NB(n)},x_V)} \left[ \log q_\theta(y_n|x_V) \right]
    \]
- The variational distribution can also by trained on the labeled data
  \[
  O_{\theta,L} = \sum_{n \in L} \log q_\theta(y_n|x_V).
  \]
- Final objective:
  \[
  O_\theta = O_{\theta,U} + O_{\theta,L}
  \]
Learning/M-step:

• The log-pseudo likelihood:

\[
\ell_{PL}(\phi) \triangleq \mathbb{E}_{q_\theta(y_U|x_V)} \left[ \sum_{n \in V} \log p_\phi(y_n|y_{V\setminus n}, x_V) \right] \\
= \mathbb{E}_{q_\theta(y_U|x_V)} \left[ \sum_{n \in V} \log p_\phi(y_n|y_{\text{NB}(n)}, x_V) \right]
\]

• According to the inference, only the \( p_\phi(y_n|y_{\text{NB}(n)}, x_V) \) is required

• Parametrize \( p_\phi(y_n|y_{\text{NB}(n)}, x_V) \) with another GCN

\[
p_\phi(y_n|y_{\text{NB}(n)}, x_V) = \text{Cat}(y_n|\text{softmax}(W_\phi h_\phi, n))
\]
Learning/M-step:

• Estimate the expectation by drawing a sample from $q_\theta(y_U|x_V)$

• Final objective:

$$O_\phi = \sum_{n \in V} \log p_\phi(\hat{y}_n|\hat{y}_{NB(n)}, x_V)$$

$\hat{y}_n \sim q_\theta(y_U|x_V)$, if $n$ is unlabeled

$\hat{y}_n$ is the ground truth label, if $n$ is labeled
Overall Optimization Procedure

• Two Graph Neural Networks Collaborate with each other
  • $p_\phi$: learning network, modeling the label dependency
  • $q_\theta$: inference network, learning the object representations

• $q_\theta$ infer the labels of unlabeled objects trained with supervision from $p_\phi$ and labeled objects

• $p_\phi$ is trained with a fully labeled graph, where the unlabeled objects are labeled by $q_\theta$
Applications: Object/Node Classification

• Train, validation, and test are standard split
• State-of-the-art performance

<table>
<thead>
<tr>
<th>Category</th>
<th>Algorithm</th>
<th>Cora</th>
<th>Citeseer</th>
<th>Pubmed</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSL</td>
<td>LP</td>
<td>74.2</td>
<td>56.3</td>
<td>71.6</td>
</tr>
<tr>
<td>SRL</td>
<td>PRM</td>
<td>77.0</td>
<td>63.4</td>
<td>68.3</td>
</tr>
<tr>
<td></td>
<td>RMN</td>
<td>71.3</td>
<td>68.0</td>
<td>70.7</td>
</tr>
<tr>
<td></td>
<td>MLN</td>
<td>74.6</td>
<td>68.0</td>
<td>75.3</td>
</tr>
<tr>
<td>GNN</td>
<td>Planetoid *</td>
<td>75.7</td>
<td>64.7</td>
<td>77.2</td>
</tr>
<tr>
<td></td>
<td>GCN *</td>
<td>81.5</td>
<td>70.3</td>
<td>79.0</td>
</tr>
<tr>
<td></td>
<td>GAT *</td>
<td>83.0</td>
<td>72.5</td>
<td>79.0</td>
</tr>
<tr>
<td>GMNN</td>
<td>W/o Attr. in $p_{\phi}$</td>
<td>83.4</td>
<td>73.1</td>
<td>81.4</td>
</tr>
<tr>
<td></td>
<td>With Attr. in $p_{\phi}$</td>
<td><strong>83.7</strong></td>
<td>72.9</td>
<td><strong>81.8</strong></td>
</tr>
</tbody>
</table>
### Results of Node Classification

- Random data splits
- State-of-the-art performance

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Cora</th>
<th>Citeseer</th>
<th>Pubmed</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCN</td>
<td>81.5</td>
<td>71.3</td>
<td>80.3</td>
</tr>
<tr>
<td>GAT</td>
<td>82.1</td>
<td>71.5</td>
<td>80.1</td>
</tr>
<tr>
<td>GMNN</td>
<td><strong>83.1</strong></td>
<td><strong>73.0</strong></td>
<td><strong>81.9</strong></td>
</tr>
</tbody>
</table>
Few-shot Learning Settings

- 5 labeled objects for each class
- The performance gain are even larger

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Cora</th>
<th>Citeseer</th>
<th>Pubmed</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCN</td>
<td>74.9</td>
<td>69.0</td>
<td>76.9</td>
</tr>
<tr>
<td>GAT</td>
<td>77.0</td>
<td>68.9</td>
<td>75.4</td>
</tr>
<tr>
<td>GMNN</td>
<td>78.6</td>
<td>72.7</td>
<td>79.1</td>
</tr>
</tbody>
</table>
Ablation Study on the Learning Network

\[ p_\phi(y_n | y_{NB(n)}) \]

- 1 mean pooling layer: just take the average distribution of labels of neighbors,
  - label propagation!
  - Model the label dependency in a linear way

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Cora</th>
<th>Citeseer</th>
<th>Pubmed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Mean Pooling Layer</td>
<td>82.4</td>
<td>71.9</td>
<td>80.7</td>
</tr>
<tr>
<td>1 GC Layer</td>
<td>83.1</td>
<td>73.1</td>
<td>80.9</td>
</tr>
<tr>
<td>2 GC Layers</td>
<td>83.4</td>
<td>73.1</td>
<td>81.4</td>
</tr>
<tr>
<td>3 GC Layers</td>
<td>83.6</td>
<td>73.0</td>
<td>81.5</td>
</tr>
</tbody>
</table>
Ablation Study on the Inference Network $q_\theta(y_n|x_V)$

- Non-amortized: treat $q_\theta(y_n|x_V)$ as parameter, independent of $x_V$
- 1 Linear Layer: only use the features $x_n$

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Cora</th>
<th>Citeseer</th>
<th>Pubmed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-amortized</td>
<td>45.3</td>
<td>28.1</td>
<td>42.2</td>
</tr>
<tr>
<td>1 Linear Layer</td>
<td>55.8</td>
<td>57.5</td>
<td>69.8</td>
</tr>
<tr>
<td>1 GC Layer</td>
<td>72.9</td>
<td>67.6</td>
<td>71.8</td>
</tr>
<tr>
<td>2 GC Layers</td>
<td>83.4</td>
<td>73.1</td>
<td>81.4</td>
</tr>
<tr>
<td>3 GC Layers</td>
<td>82.0</td>
<td>70.6</td>
<td>80.7</td>
</tr>
</tbody>
</table>
Convergence Analysis of Optimization

(a) Cora

(b) Citeseer
Applications: Unsupervised Node Representation Learning

• There are no labeled nodes!!
• Instead, we introduce a pseudo task. For each node n, we aims to predict the neighbors, i.e., $p(y | n, x_V)$
• E-step: infer the neighbor distribution for each node with $q_\theta$
• M-step: update the $p_\phi$ to model the local dependency of the inferred neighbor distributions
Node/Object classification

- DGI: Deep Graph Infomax, Veličković et al. 2019

<table>
<thead>
<tr>
<th>Category</th>
<th>Algorithm</th>
<th>Cora</th>
<th>Citeseer</th>
<th>Pubmed</th>
</tr>
</thead>
<tbody>
<tr>
<td>GNN</td>
<td>DeepWalk *</td>
<td>67.2</td>
<td>43.2</td>
<td>65.3</td>
</tr>
<tr>
<td></td>
<td>DGI *</td>
<td>82.3</td>
<td>71.8</td>
<td>76.8</td>
</tr>
<tr>
<td>GMNN</td>
<td>With only $q_\theta$</td>
<td>78.1</td>
<td>68.0</td>
<td>79.3</td>
</tr>
<tr>
<td></td>
<td>With $q_\theta$ and $p_\phi$</td>
<td><strong>82.8</strong></td>
<td>71.5</td>
<td><strong>81.6</strong></td>
</tr>
</tbody>
</table>
Applications: Link Classification

• Construct a dual graph \( \tilde{G} \) from the original graph \( G \)
  • Each edge in \( G \) -> a node in \( \tilde{G} \)
  • Two nodes in \( \tilde{G} \) are connected if the corresponding edges in \( G \) share a node

<table>
<thead>
<tr>
<th>Category</th>
<th>Algorithm</th>
<th>Bitcoin Alpha</th>
<th>Bitcoin OTC</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSL</td>
<td>LP</td>
<td>59.68</td>
<td>65.58</td>
</tr>
<tr>
<td>SRL</td>
<td>PRM</td>
<td>58.59</td>
<td>64.37</td>
</tr>
<tr>
<td></td>
<td>RMN</td>
<td>59.56</td>
<td>65.59</td>
</tr>
<tr>
<td></td>
<td>MLN</td>
<td>60.87</td>
<td>65.62</td>
</tr>
<tr>
<td>GNN</td>
<td>DeepWalk</td>
<td>62.71</td>
<td>63.20</td>
</tr>
<tr>
<td></td>
<td>GCN</td>
<td>64.00</td>
<td>65.69</td>
</tr>
<tr>
<td>GMNN</td>
<td>W/o Attr. in ( p_\phi )</td>
<td>65.59</td>
<td>66.62</td>
</tr>
<tr>
<td></td>
<td>With Attr. in ( p_\phi )</td>
<td>65.86</td>
<td>66.83</td>
</tr>
</tbody>
</table>
Summary

• A fundamental problem on graphs: semi-supervised node classification

• GMNN: towards combining statistical relational learning and graph neural networks
  • Model the label dependency with one graph neural network
  • Learn effective node representations with another graph neural network

• State-of-the-art results on semi-supervised node classification, unsupervised node representation, and link classification

• Code available at: https://github.com/DeepGraphLearning/GMNN
Questions?