GMNN: Graph Markov Neural Network

Cheng Peng
Chenlei Song
Qiyue Yao
Shijia Hu
Motivation

- Relational Data Modeling
- Semi-Supervised Object Classification
- Link Problem
Problem Definition

Consider the problem of semi-supervised object classification:

\[ G = (V, E, x_V) \]

The goal is to predict the labels \( y_U \) for unlabeled objects, given some of the labeled objects \( L \subset V \).

More generally, one wants to find \( p_\phi(y_U | y_L, x_V) \).
Traditional Methods

- Statistical Relational Learning (SRL)
- Graph Neural Networks (GNN)
Statistical Relational Learning

• Recap: Conditional Random Field -- Models label dependency w/ conditional distribution

\[ p(y_V | x_V) = \frac{1}{Z(x_V)} \prod_{(i,j) \in E} \psi_{i,j}(y_i, y_j, x_V). \]

• Example: Part-of-Speech tagging with Linear Chain CRF using feature functions

• Problem:
  • Potential functions relies on handcrafted feature functions
  • Intractable posterior inference
Graph Neural Networks

- Convolutional Neural Networks (CNN) operates on regular Euclidean 2D grid and 1D sequence.
- Non-Euclidean data:
  - Consumer-product interaction in e-commerce system.
  - Molecule bioactivity
  - Citation network
  - ...

Fig. 1. Left: image in Euclidean space. Right: graph in non-Euclidean space
GNN explained

- Learn a state embedding for each node on the graph.
- State embeddings are used to produce an output, such as a node label in our scenario.
- A parametric function (local transition function) updates the node state according to the input neighborhood.
- A local output function will describe how the output is produced.

Whiteboard time!
\[ x_1 = f_w(l_1, l_{(1,2)}, l_{(3,1)}, l_{(1,4)}, l_{(6,1)}, x_2, x_3, x_4, x_6, l_2, l_3, l_4, l_6) \]

\[ l_{co[1]} \]

\[ x_{ne[1]} \]

\[ l_{ne[n]} \]
Graph Markov Neural Network
Graph Markov Neural Network
\[ p(y_V | x_V) = \prod_{n \in V} p(y_n | x_V) \]

\[ h = g(x_V, E) \quad p(y_n | x_V) = \text{Cat}(y_n | \text{softmax}(W h_n)) \]
GMNN: Graph Markov Neural Networks

Model: joint distribution of object labels conditioned on object attributes

\[ p_\phi (y_V \mid x_V) \]

Learn: model parameters \( \Phi \) by maximizing the log-likelihood function (of observed object labels)

\[ \log p_\phi (y_L \mid x_V) \]
Recall: Evidence Lower Bound (ELBO)

Jensen’s Inequality: if $f$ is concave, $f(E[X]) \geq E[f(X)]$

$$\log p(x) = \log \int_z p(x, z)$$
$$= \log \int_z p(x, z) \frac{q(z)}{q(z)}$$
$$= \log \left( E_q \left[ \frac{p(x, Z)}{q(z)} \right] \right)$$
$$\geq E_q[\log p(x, Z)] - E_q[\log q(Z)].$$

Optimize ELBO of

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

**E-step (inference):** fix $p_{\phi}$ update variational distribution $q_{\theta}(y_U|x_V)$

**M-step (learning):** fix $q_{\theta}$ update $p_{\phi}$ to maximize the likelihood function

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

Pseudolikelihood

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

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

E-Step: a.k.a inference procedure — compute posterior distribution

$$p_\phi(y_U | y_L, x_V)$$

instead, we use $q_\theta(Y_U | X_V)$ to approximate the real distribution

— fix $\phi$ and optimize $q_\theta$
Inference (E-step)

mean-field method:

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

— all object labels are assumed to be independent
Inference (E-step)

we use a GNN to parameterize $q_\theta$

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

- $q_\theta$ is formulated as a categorical distribution
- $h_{\theta,n}$ is learned by a GNN with $X_V$ as features and $\theta$ as parameters
Inference (E-step)

under mean-field method, the optimal distribution is:

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

— proof ...
Inference (E-step)

\[
\log q_\theta(y_n|x_V) = \\
\mathbb{E}_{q_\theta(y_{NB(n)} \cap U|x_V)} \left[ \log p_\phi(y_n|y_{NB(n)}, x_V) \right] + \text{const.}
\]

— we have \(q_\theta\) in right side. still intractable...

take a sample to estimate the expectation
Inference (E-step)

\[
\mathbb{E}_{q_\theta(y_{NB(n)} \cap U | x_V)} \left[ \log p_\phi(y_n | y_{NB(n)}, x_V) \right] \\
\approx \log p_\phi(y_n | \hat{y}_{NB(n)}, x_V).
\]

for unlabeled neighbor:  \( \hat{y}_k \sim q_\theta(y_k | x_V) \)

for labeled neighbor:  \( \hat{y}_k \) set as the ground-truth label

\[
q_\theta(y_n | x_V) \approx p_\phi(y_n | \hat{y}_{NB(n)}, x_V).
\]
Inference (E-step)

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

now, we only need to optimize \( q_\theta \) to approximate the right side

— minimize the reverse KL divergence

\[ -KL(P_\phi(y_n | \hat{y}_{NB(n)}, X_V) \mid \mid q_\theta(y_n | X_V)) \]
Inference (E-step)

we get the objective function:

\[ O_{\theta,U} = \sum_{n \in U} \mathbb{E}_{p_{\phi}(y_{n}|\hat{y}_{\text{NB}(n)},x_V)} [\log q_{\theta}(y_{n}|x_V)] \]

We can also use labeled objects to train \( q_{\theta} \)

\[ O_{\theta,L} = \sum_{n \in L} \log q_{\theta}(y_{n}|x_V) \]

— So, the overall objective function for optimizing

\[ O_{\theta} = O_{\theta,U} + O_{\theta,L} \]
Learning (M-step)

Log pseudolikelihood:

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

Parameterize with another non-linear GNN:

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

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

$$\hat{y}_n \sim q_{\theta}(y_n | x_V)$$

Ground-Truth label

Object function:

$$O_\phi = \sum_{n \in V} \log p_{\phi}(\hat{y}_n | \hat{y}_{NB(n)}, x_V)$$
Algorithm 1 Optimization Algorithm

**Input:** A graph \( G \), some labeled objects \((L, y_L)\).

**Output:** Object labels \( y_U \) for unlabeled objects \( U \).

Pre-train \( q_\theta \) with \( y_L \) according to Eq. (12).

**while** not converge **do**

- **M-Step: Learning Procedure**
  - Annotate unlabeled objects with \( q_\theta \).
  - Denote the sampled labels as \( \hat{y}_U \).
  - Set \( \hat{y}_V = (y_L, \hat{y}_U) \) and update \( p_\phi \) with Eq. (15).

- **E-Step: Inference Procedure**
  - Annotate unlabeled objects with \( p_\phi \) and \( \hat{y}_V \).
  - Denote the predicted label distribution as \( p_\phi(y_U) \).
  - Update \( q_\theta \) with Eq. (11), (12) based on \( p_\phi(y_U), y_L \).

**end while**

Classify each unlabeled object \( n \) based on \( q_\theta(y_n | x_V) \).
Applications

Unsupervised Node Representation Learning

- No labeled nodes
- Instead, predict the neighbors for each node
- 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
Applications

Link classification

- Use the original graph $G$ to construct a dual graph $\tilde{G}$
- The object set $\tilde{V}$ in dual graph corresponds to the link set $\tilde{E}$ in original graph
- Two objects are linked in $\tilde{G}$ if their corresponding links in $G$ share a node
Compared Algorithms

GNN Methods
- Graph Convolutional Network
- Graph Attention Network

SRL Methods
- Probabilistic Relational Model
- Relational Markov Network
- Markov Logic Network

SSL Methods
- Label Propagation
## Results

<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>83.7</td>
<td>72.9</td>
<td>81.8</td>
</tr>
</tbody>
</table>

<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><strong>65.86</strong></td>
<td><strong>66.83</strong></td>
</tr>
</tbody>
</table>
Questions

1. Briefly describe the motivation for combining SRL & GNN respectively

2. In GMNN, we used GNN twice. What’s the main difference between them?
Reference

http://www.ipam.ucla.edu/abstract/?tid=16001&pcode=GLWS4
https://github.com/DeepGraphLearning/GMNN