Understanding Negative Sampling in Graph Representation Learning

Zhen Yang*,†, Ming Ding*,†, Chang Zhou‡, Hongxia Yang‡, Jingren Zhou‡, Jie Tang†

†Department of Computer Science and Technology, Tsinghua University
‡DAMO Academy, Alibaba Group

*These authors contributed equally to this work.
Graph Representation Learning

- Node Classification
- Link Prediction
- Recommendation
- ...
Sampled Noise Contrastive Estimation Framework

SampledNCE Framework

Positive Sampler  Trainable Encoder $E_\theta$  Negative Sampler

Positive node pairs  Negative node pairs

Sample positive nodes  Generate node embeddings  Sample negative nodes

Positive Sampling  $p_d(\cdot|v)$  Used for training  Negative Sampling

Cross-entropy loss

$p_n(\cdot|v)$
**Problems & Challenges**

- Unexplored
- Lacking systematically analyzed

**Related Work:**

1. **Degree-based Negative Sampling**
   - **Advantage:** simple and fast
   - **Disadvantage:** static, inconsiderate to the personalization of nodes.

2. **Hard-samples Negative Sampling**
   - **Advantage:** mine hard negative samples
   - **Disadvantage:** sampling with rejection may cost so many time to try.

3. **GAN-based Negative Sampling**
   - **Advantage:** adversially generate “difficult” samples
   - **Disadvantage:** Training difficulties; Long training time
Negative Sampling

• Definition:

Given a graph \( G(\mathcal{V}, \mathcal{E}) \), where \( \mathcal{V} = \{v_1, v_2, v_3, \ldots, v_m\} \) is the node set,

\( \mathcal{E} = \{e_1, e_2, e_3, \ldots, e_n\} \) is the edge set.

For a node pair \((v_1, v_2)\), maximize the log-likelihood of this pair and minimize the log-likelihood of all unconnected node pairs:

\[
J = \log(\sigma(\vec{v}_1 \cdot \vec{v}_2)) - \log \left( \sum_{u \in \mathcal{V}} \sigma(\vec{v}_1 \cdot \vec{u}) \right)
\]

- **Negative Sampling**: sample \( k \) negative nodes to replace all nodes.

\[
J = \log(\sigma(\vec{v}_1 \cdot \vec{v}_2)) - k \cdot \mathbb{E}_{u \sim p_n(u)} \log (\sigma(\vec{v}_1 \cdot \vec{u}))
\]

• Purpose:

1) Accelerate the training process.  2) Reduce computational complexity
How does negative sampling influence the learning?

**Q1:** Does $p_n$ affect the embedding learning?

- ○ Yes
- ◯ No

**Q2:** What is the relationship between $p_n$ and $p_d$?

- ○ $p_n \propto p_d$
- ○ $p_n \cdot p_d$
- ○ $p_n \propto \frac{1}{p_d}$
- ◯ $\frac{p_n}{p_d}$

**Notations:**
- Positive Sampling Distribution $p_d$
- Negative Sampling Distribution $p_n$
How does negative sampling influence the learning?

Objective Function:

\[ J = \mathbb{E}_{(u,v) \sim p_d} \log \sigma (\tilde{u}^T \tilde{v}) + \mathbb{E}_{v \sim p_d(v)} [k \mathbb{E}_{u' \sim p_n(u'|v)} \log \sigma (-\tilde{u}'^T \tilde{v})] \]

Simplify As:

\[ J = - \sum_u (p_d(u|v) + kp_n(u|v)) H(P_{u,v}, Q_{u,v}) \text{ where } H(p, q) \text{ is the cross entropy} \]

Optimal Embedding:

\[ \tilde{u}^T \tilde{v} = - \log \frac{k \cdot p_n(u|v)}{p_d(u|v)} \]
What extent does negative sampling influence the learning?

**Empirical Risk:**

\[
J_T(u) = \frac{1}{T} \sum_{i=1}^{T} \log \sigma(u_i^T \tilde{v}) + \frac{1}{T} \sum_{i=1}^{kT} \log \sigma(-u_i' \tilde{v}),
\]

where \{u_1, ..., u_T\} are sampled from \(p_d(u|v)\) and \{u_1', ..., u'_kT\} are sampled from \(p_n(u|v)\).

**Theorem:**

The random variable \(\sqrt{T}(\theta_T - \theta^*)\) asymptotically converges to a distribution with zero mean vector and covariance matrix \(\text{Cov}\).

**Proof by Taylor Expansion**

\[
\text{Cov}(\sqrt{T}(\theta_T - \theta^*)) = \text{diag}(m)^{-1} - (1 + 1/k)1^T1
\]

where \(m = \left[\frac{k p_d(u_0|v)p_n(u_0|v)}{p_d(u_0|v)+k p_n(u_0|v)}, ..., \frac{k p_d(u_{N-1}|v)p_n(u_{N-1}|v)}{p_d(u_{N-1}|v)+k p_n(u_{N-1}|v)}\right]^T\) and \(1 = [1, ..., 1]^T\).

**Mean Squared Error:**

\[
\mathbb{E}[\| (\theta_T - \theta^*)_u \|^2] = \frac{1}{T} \left( \frac{1}{p_d(u|v)} - 1 + \frac{1}{k p_n(u|v)} - \frac{1}{k} \right)
\]
The Principle of Negative Sampling

A simple solution is to sample negative nodes positively but sub-linearly correlated to their positive sampling distribution.

\[ p_n(u|v) \propto p_d(u|v)^\alpha, \quad 0 < \alpha < 1 \]

- Monotonicity:

\[ p_d(u_i|v) > p_d(u_j|v) \]

\[ \overrightarrow{u}_i^T \overrightarrow{v} = -\log \frac{k \cdot p_n(u|v)}{p_d(u|v)} \]

\[ \overrightarrow{u}_i^T \overrightarrow{v} = \log p_d(u_i|v) - \alpha \log p_d(u_i|v) + c > (1 - \alpha) \log p_d(u_j|v) + c = \overrightarrow{u}_j^T \overrightarrow{v} \]
Our Solution: MCNS Model

Markov chain Monte Carlo Negative Sampling (MCNS):

• an effective and scalable negative sampling strategy.
• applies our theory with an approximated positive distribution based on current embeddings.
• leverages a special Metropolis-Hastings algorithm for sampling.

An Approximated Positive Distribution:

• Self-contrast approximation:

\[ p_d(u|v) \approx \frac{E_\theta(u) \cdot E_\theta(v)}{\sum_{u' \in U} E_\theta(u') \cdot E_\theta(v)} \]
Our Solution: MCNS Model

Negative Distribution:

\[ p_n(u|v) \propto p_d(u|v)^\alpha \approx \frac{(E_\theta(u) \cdot E_\theta(v))^\alpha}{\sum_{u' \in U} (E_\theta(u') \cdot E_\theta(v))^\alpha} \]

- Very time-consuming
- Each sampling requires \( O(n) \) time, making it impossible for middle- or large-scale graphs.
- Accelerating by Metropolis-Hastings algorithm.
Our Solution: MCNS Model

DFS sequence: 0 \rightarrow 1 \rightarrow 2 \rightarrow 4 \rightarrow 5 \rightarrow 6 \rightarrow 4 \rightarrow 7

Markov chain: \ldots \rightarrow 3 \rightarrow 1 (reject) \rightarrow 1 \rightarrow 2 \rightarrow 5 \rightarrow 0

positive context node u = 8

drafted from \hat{p}_d(\cdot | 7)

current central node v = 7

\mathcal{L} = \max(0, E_\theta(v) \cdot E_\theta(x) - E_\theta(u) \cdot E_\theta(v) + \gamma)
Our Solution: MCNS Model

Proposal Distribution $q(y|\mathbf{x})$:

- mixing uniform sampling
- and sampling from the nearest $k$ nodes with probability $\frac{1}{2}$ each.
Experimental Settings

3 representative tasks. 3 graph representation learning algorithms.
5 datasets. 19 experimental settings.

<table>
<thead>
<tr>
<th>Task</th>
<th>Dataset</th>
<th>Nodes</th>
<th>Edges</th>
<th>Classes</th>
<th>Evaluation Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recommendation</td>
<td>MovieLens</td>
<td>2,625</td>
<td>100,000</td>
<td>/</td>
<td>MRR/Hits@k</td>
</tr>
<tr>
<td></td>
<td>Amazon</td>
<td>255,404</td>
<td>1,689,188</td>
<td>/</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Alibaba</td>
<td>159,633</td>
<td>907,470</td>
<td>/</td>
<td></td>
</tr>
<tr>
<td>Link Prediction</td>
<td>Arxiv</td>
<td>5,242</td>
<td>28,980</td>
<td>/</td>
<td>AUC</td>
</tr>
<tr>
<td>Node Classification</td>
<td>BlogCatalog</td>
<td>10,312</td>
<td>333,983</td>
<td>39</td>
<td>Micro-F1</td>
</tr>
</tbody>
</table>
### Recommendation Results

<table>
<thead>
<tr>
<th></th>
<th>MovieLens</th>
<th>Amazon</th>
<th>Alibaba</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DeepWalk</td>
<td>GCN</td>
<td>GraphSAGE</td>
</tr>
<tr>
<td><strong>MRR</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deg$^0.75$</td>
<td>0.025±.001</td>
<td>0.062±.001</td>
<td>0.063±.001</td>
</tr>
<tr>
<td>WRMF</td>
<td>0.022±.001</td>
<td>0.038±.001</td>
<td>0.040±.001</td>
</tr>
<tr>
<td>RNS</td>
<td>0.031±.001</td>
<td>0.082±.002</td>
<td>0.079±.001</td>
</tr>
<tr>
<td>PinSAGE</td>
<td>0.036±.001</td>
<td>0.091±.002</td>
<td>0.090±.002</td>
</tr>
<tr>
<td>WARP</td>
<td>0.041±.003</td>
<td>0.114±.003</td>
<td>0.111±.003</td>
</tr>
<tr>
<td>DNS</td>
<td>0.040±.003</td>
<td>0.113±.003</td>
<td>0.115±.003</td>
</tr>
<tr>
<td>IRGAN</td>
<td>0.047±.002</td>
<td>0.111±.002</td>
<td>0.101±.002</td>
</tr>
<tr>
<td>KBGAN</td>
<td>0.049±.001</td>
<td>0.114±.003</td>
<td>0.100±.001</td>
</tr>
<tr>
<td><strong>MCNS</strong></td>
<td><strong>0.053±.001</strong></td>
<td><strong>0.122±.004</strong></td>
<td><strong>0.114±.001</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>MovieLens</th>
<th>Amazon</th>
<th>Alibaba</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>DeepWalk</td>
<td>GCN</td>
<td>GraphSAGE</td>
</tr>
<tr>
<td><strong>Hits@30</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deg$^0.75$</td>
<td>0.115±.002</td>
<td>0.270±.002</td>
<td>0.270±.001</td>
</tr>
<tr>
<td>WRMF</td>
<td>0.110±.003</td>
<td>0.187±.002</td>
<td>0.181±.002</td>
</tr>
<tr>
<td>RNS</td>
<td>0.143±.004</td>
<td>0.362±.004</td>
<td>0.356±.001</td>
</tr>
<tr>
<td>PinSAGE</td>
<td>0.158±.003</td>
<td>0.379±.005</td>
<td>0.383±.005</td>
</tr>
<tr>
<td>WARP</td>
<td>0.164±.005</td>
<td>0.406±.002</td>
<td>0.404±.005</td>
</tr>
<tr>
<td>DNS</td>
<td>0.166±.005</td>
<td>0.404±.006</td>
<td>0.410±.006</td>
</tr>
<tr>
<td>IRGAN</td>
<td>0.207±.002</td>
<td>0.415±.004</td>
<td>0.408±.004</td>
</tr>
<tr>
<td>KBGAN</td>
<td>0.198±.003</td>
<td>0.420±.003</td>
<td>0.401±.005</td>
</tr>
<tr>
<td><strong>MCNS</strong></td>
<td><strong>0.230±.003</strong></td>
<td><strong>0.426±.005</strong></td>
<td><strong>0.413±.003</strong></td>
</tr>
</tbody>
</table>

MCNS achieves significant gains of 2%~13% over the best baselines.
## Link Prediction Results

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>DeepWalk</th>
<th>GCN</th>
<th>GraphSAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Deg^{0.75}$</td>
<td>64.6±0.1</td>
<td>79.6±0.4</td>
<td>78.9±0.4</td>
</tr>
<tr>
<td>WRMF</td>
<td>65.3±0.1</td>
<td>80.3±0.4</td>
<td>79.1±0.2</td>
</tr>
<tr>
<td>RNS</td>
<td>62.2±0.2</td>
<td>74.3±0.5</td>
<td>74.7±0.5</td>
</tr>
<tr>
<td>PinSAGE</td>
<td>67.2±0.4</td>
<td>80.4±0.3</td>
<td>80.1±0.4</td>
</tr>
<tr>
<td>WARP</td>
<td>70.5±0.3</td>
<td>81.6±0.3</td>
<td>82.7±0.4</td>
</tr>
<tr>
<td>DNS</td>
<td>70.4±0.3</td>
<td>81.5±0.3</td>
<td>82.6±0.4</td>
</tr>
<tr>
<td>IRGAN</td>
<td>71.1±0.2</td>
<td>82.0±0.4</td>
<td>82.2±0.3</td>
</tr>
<tr>
<td>KBGAN</td>
<td>71.6±0.3</td>
<td>81.7±0.3</td>
<td>82.1±0.3</td>
</tr>
<tr>
<td><strong>MCNS</strong></td>
<td><strong>73.1±0.4</strong></td>
<td><strong>82.6±0.4</strong></td>
<td><strong>83.5±0.5</strong></td>
</tr>
</tbody>
</table>

MCNS outperforms all baselines with various graph representation learning methods.
Node Classification Results

<table>
<thead>
<tr>
<th>Micro-F1</th>
<th>Algorithm</th>
<th>DeepWalk</th>
<th>GCN</th>
<th>GraphSAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$T_R(%)$</td>
<td>10</td>
<td>50</td>
<td>90</td>
</tr>
<tr>
<td>Deg$^{0.75}$</td>
<td></td>
<td>31.6</td>
<td>36.6</td>
<td>39.1</td>
</tr>
<tr>
<td>WRMF</td>
<td></td>
<td>30.9</td>
<td>35.8</td>
<td>37.5</td>
</tr>
<tr>
<td>RNS</td>
<td></td>
<td>29.8</td>
<td>34.1</td>
<td>36.0</td>
</tr>
<tr>
<td>PinSAGE</td>
<td></td>
<td>32.0</td>
<td>37.4</td>
<td>40.1</td>
</tr>
<tr>
<td>WARP</td>
<td></td>
<td>35.1</td>
<td>40.3</td>
<td>42.1</td>
</tr>
<tr>
<td>DNS</td>
<td></td>
<td>35.2</td>
<td>40.4</td>
<td>42.5</td>
</tr>
<tr>
<td>IRGAN</td>
<td></td>
<td>34.3</td>
<td>39.6</td>
<td>41.8</td>
</tr>
<tr>
<td>KBGAN</td>
<td></td>
<td>34.6</td>
<td>40.0</td>
<td>42.3</td>
</tr>
<tr>
<td><strong>MCNS</strong></td>
<td></td>
<td><strong>36.1</strong></td>
<td><strong>41.2</strong></td>
<td><strong>43.3</strong></td>
</tr>
</tbody>
</table>

MCNS stably outperforms all baselines regardless of the training set ratio $T_R$. 
Efficiency Comparison

Runtime Comparisons:

The runtime of MCNS and hard-samples or GAN-based strategies with GraphSAGE encoder in recommendation task.
Futher Analysis: Comparison with Power of Degree

Degree-based NS:

\( p_n(v) \propto \text{deg}(v)^\beta \)

- **Abscissa**: \( \beta \) varies from -1 to 1.
- **Results**:
  1) Best \( \beta \) varies on datasets.
  2) MCNS naturally adapts to different datasets.
Futher Analysis: Parameter Analysis

- Margin $\gamma$:
  - the hinge loss begins to take effect when $\gamma \geq 0$
  - reaches its optimum at $\gamma \approx 0.1$

- Embedding Dimension:
  - set as 512
  - achieve the trade-off between performance and time consumption.
Further Understanding

• Whether sampling more negative samples is always helpful?

• Improve at first: decrease the risk

• Decrease after the optimum: extra bias is added

![Graph showing MRR and Hits@30 metrics against the number of negative sampling k. The optimum is at k=20.]
Further Understanding

• Why our conclusion contradicts with the intuition “positively sampling nearby nodes and negatively sampling far away nodes”?
  – InverseDNS: selecting the one scored lowest in the candidate items.
  – Performance go down as $M$ increases.
Summary

• We systematically analyze the role of negative sampling from the perspectives of both objective and risk; and quantify that the negative sampling distribution should be positively but sub-linearly correlated to their positive sampling distribution.

• We propose MCNS, approximating the positive distribution with self-contrast approximation and accelerating negative sampling by Metropolis-Hastings.

• We achieve state-of-the-art performance in recommendation, link prediction and node classification, on a total of 19 experimental settings.
Thank you~

Code & Data: https://github.com/THUDM/MCNS