Understanding Negative Sampling in Graph Representation Learning

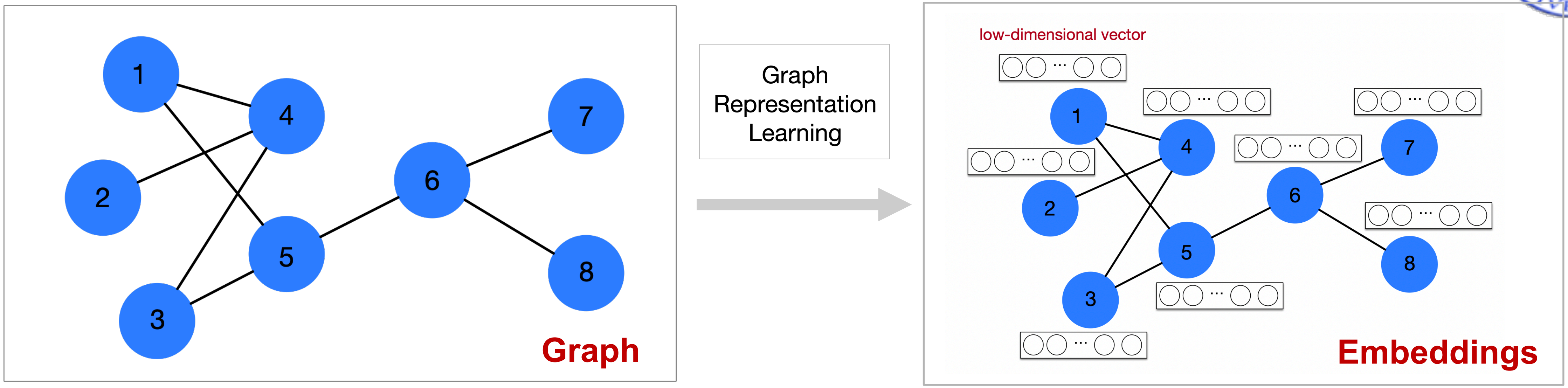
Zhen Yang*†, Ming Ding*†, Chang Zhou‡, Hongxia Yang‡, Jingren Zhou‡, Jie Tang†

†Department of Computer Science and Technology, Tsinghua University

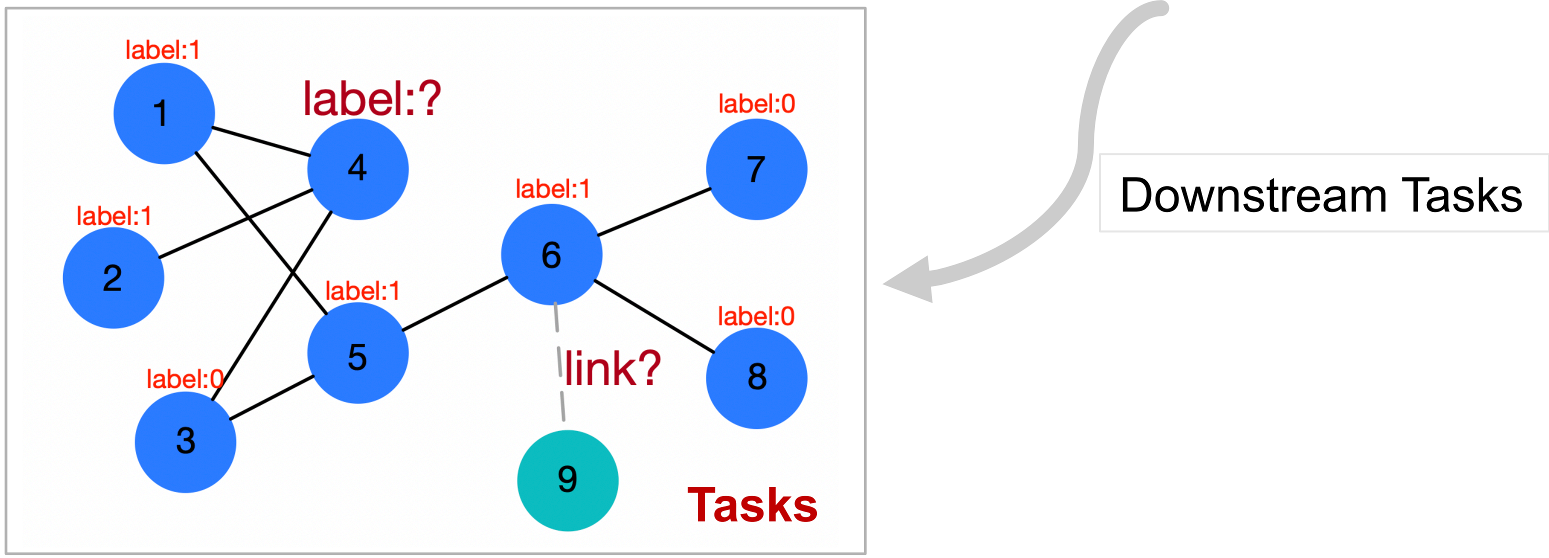
‡DAMO Academy, Alibaba Group

*These authors contributed equally to this work.

Graph Representation Learning

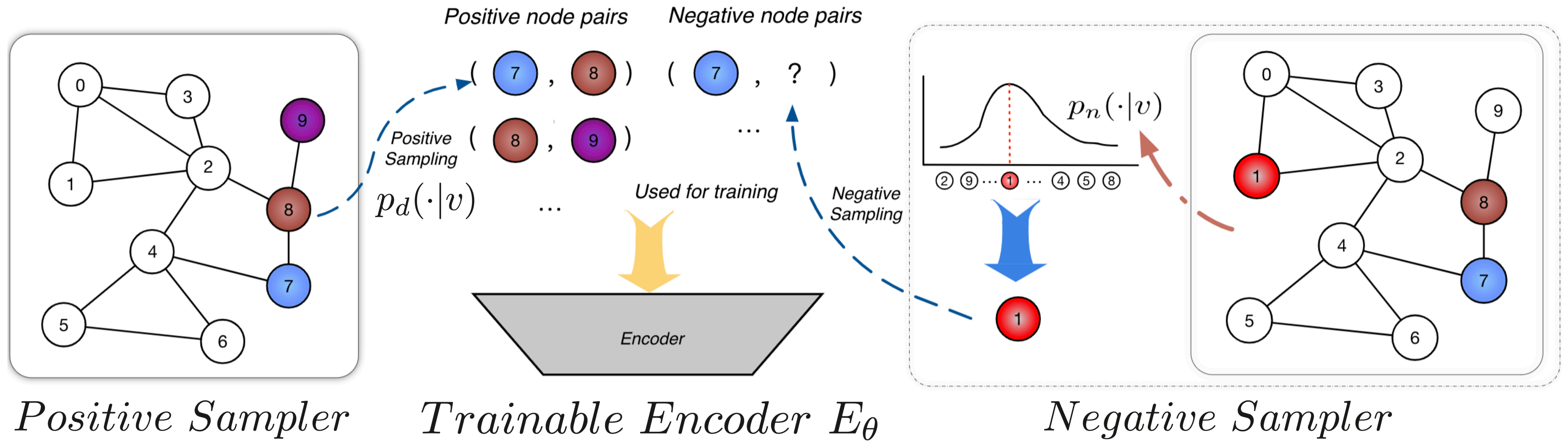


- Node Classification
- Link Prediction
- Recommendation
- ...



Sampled Noise Contrastive Estimation Framework

SampledNCE Framework



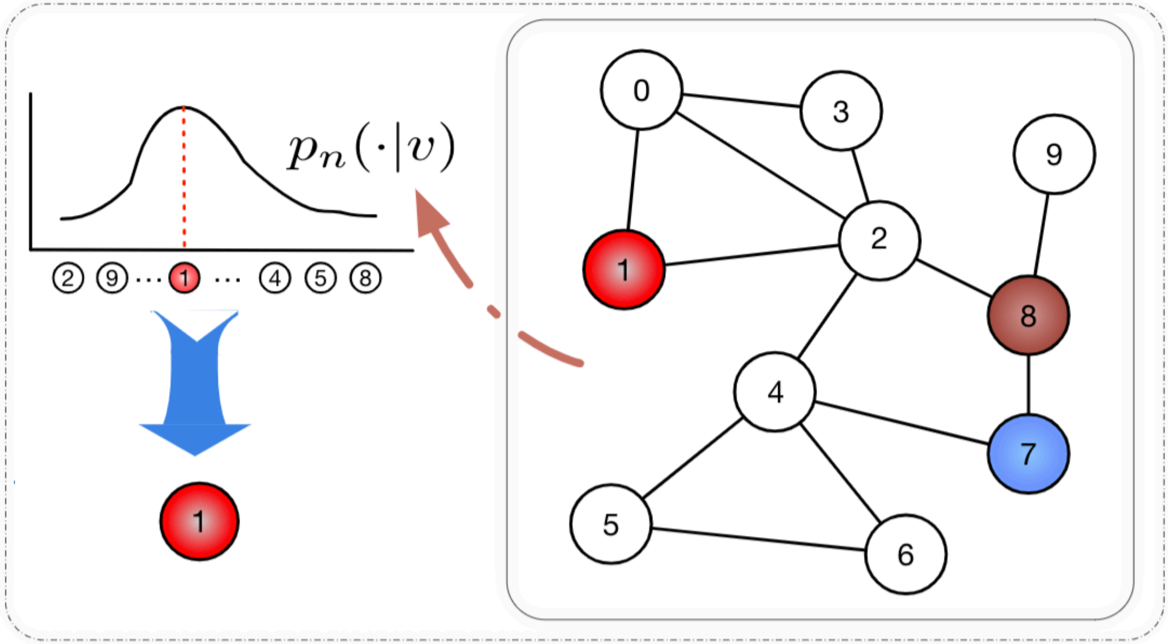
Cross-entropy loss

Sample **positive** nodes

Generate node **embeddings**

Sample **negative** nodes

Problems & Challenges



Negative Sampler

Unexplored

Lacking systematically analyzed

Related Work:

01 Degree-based Negative Sampling

Advantage: simple and fast

Disadvantage: static, inconsiderate to the personalization of nodes.

02 Hard-samples Negative Sampling

Advantage: mine hard negative samples

Disadvantage: sampling with rejection may cost so many time to try.

03 GAN-based Negative Sampling

Advantage: adversially generate “difficult” samples

Disadvantage: Training difficulties; Long training time

Negative Sampling

- **Definition**:

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

$\mathcal{E} = \{e_1, e_2, e_3, \cdots, e_n\}$ is the edge set.

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

$$J = \log(\sigma(\vec{v}_1 \cdot \vec{v}_2)) - \log \boxed{\left(\sum_{u \in \mathcal{V}} \sigma(\vec{v_1} \cdot \vec{u})\right)}$$

calculate all nodes

Negative Sampling: sample k negative nodes to replace all nodes.

$$J = \log(\sigma(\vec{v}_1 \cdot \vec{v}_2)) - \boxed{k \cdot \mathbb{E}_{u \sim p_n(u)} \log\left(\sigma(\vec{v_1} \cdot \vec{u})\right)}$$

only calculate k nodes

- **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:

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

Bernoulli distributions:

$$P_{u,v}(x=1) = \frac{p_d(u|v)}{p_d(u|v) + kp_n(u|v)} \qquad Q_{u,v}(x=1) = \sigma(\vec{\boldsymbol{u}}^T \vec{\boldsymbol{v}})$$

Simplify As:

$$J = -\sum_u (p_d(u|v) + kp_n(u|v)) H(P_{u,v}, Q_{u,v}) \quad \textit{where } H(p,q) \textit{ is the cross entropy.}$$

Gibbs Inequality:

$$P = Q$$

Optimal Embedding:

$$\vec{\boldsymbol{u}}^T \vec{\boldsymbol{v}} = -\log \frac{k \cdot p_n(u|v)}{p_d(u|v)}$$

What extent does negative sampling influence the learning?

Empirical Risk：

$$J_T^{(v)} = \frac{1}{T}\sum_{i=1}^{T} \log\sigma(\vec{\boldsymbol{u}}_i^T \vec{\boldsymbol{v}}) + \frac{1}{T}\sum_{i=1}^{kT} \log\sigma(-\vec{\boldsymbol{u'}}_i^T \vec{\boldsymbol{v}}),$$

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

Proof by

Taylor Expansion

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

$$\mathrm{Cov}\left(\sqrt{T}(\theta_T - \theta^*)\right) = \mathbf{diag}(m)^{-1} - (1 + 1/k)1^\top 1$$

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

Mean Squared Error:

$$\mathbb{E}\left[\|(\theta_T - \theta^*)_u\|^2\right] = \frac{1}{T}\left(\frac{1}{p_d(u|v)} - 1 + \frac{1}{kp_n(u|v)} - \frac{1}{k}\right)$$

The Principle of Negative Sampling

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

$$p_n(u|v) \propto p_d(u|v)^\alpha, 0 < \alpha < 1$$

- Monotonicity:

$$p_d(u_i|v) > p_d(u_j|v)$$

$$\vec{\boldsymbol{u}}^T \vec{\boldsymbol{v}} = -\log \frac{k \cdot p_n(u|v)}{p_d(u|v)}$$

Optimal Embedding

$$\begin{aligned} \vec{\boldsymbol{u}}_i^T \vec{\boldsymbol{v}} &= \log p_d(u_i|v) - \alpha \log p_d(u_i|v) + c \\ &> (1-\alpha) \log p_d(u_j|v) + c = \vec{\boldsymbol{u}}_j^T \vec{\boldsymbol{v}} \end{aligned}$$

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:
  - replacing p_d by inner products based on the current encoder

$$p_d(u|v) \approx \frac{E_\theta(u) \cdot E_\theta(v)}{\sum_{u' \in U} E_\theta(u') \cdot E_\theta(v)}$$

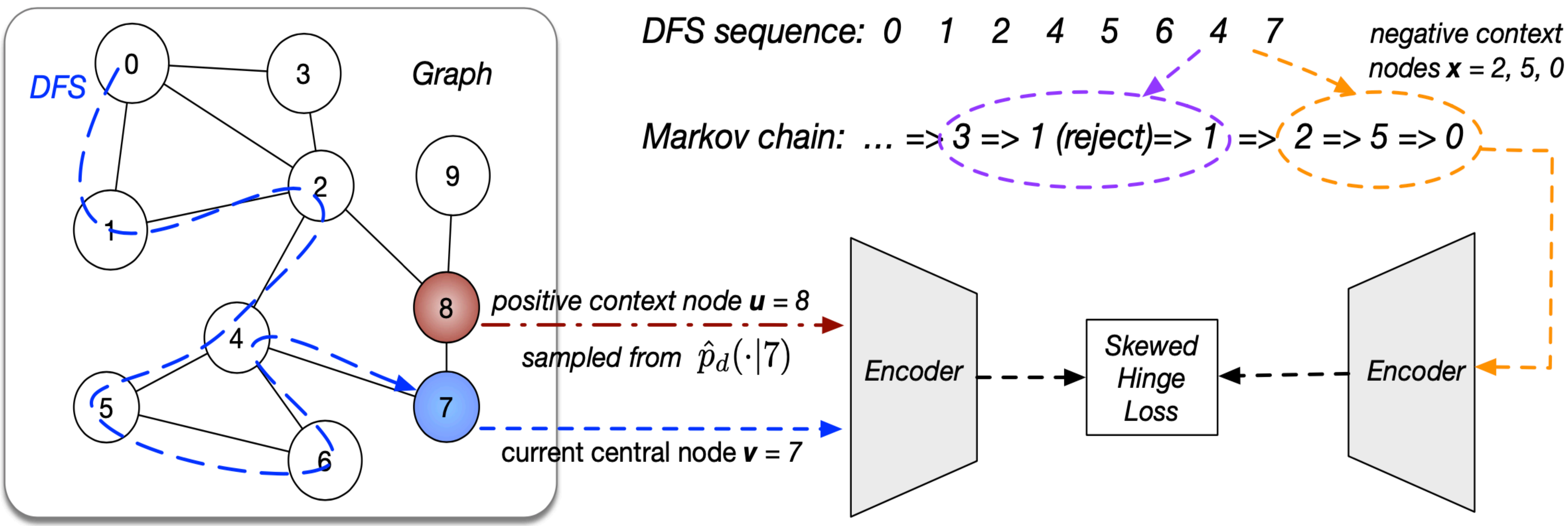
Our Solution: MCNS Model

Negative Distribution:

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

- 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



$$\mathcal{L} = \max\left(0, E_\theta(v) \cdot E_\theta(x) - E_\theta(u) \cdot E_\theta(v) + \gamma\right)$$

Our Solution: MCNS Model

Proposal Distribution $q(y|x)$ **:**

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

Algorithm 2: The training process with MCNS

Input: DFS sequence $T = [v_1, ..., v_{|T|}]$, **Step-1**
Encoder E_θ, Positive Sampling Distribution $\hat{p_d}$,
Proposal Distribution q, Number of Negative Samples k.

repeat

Initialize current negative node x at random **Step-2**

for *each node v in DFS sequence T* **do**

Sample 1 positive sample u from $\hat{p_d}(u|v)$

Initialize loss $\mathcal{L} = 0$

//Negative Sampling via Metropolis-Hastings

for $i = 1$ *to* k **do**

Sample a node y from $q(y|x)$

Generate a uniform random number $r \in [0, 1]$

if $r < min\left(1, \frac{(E_\theta(v) \cdot E_\theta(y))^\alpha}{(E_\theta(v) \cdot E_\theta(x))^\alpha} \frac{q(x|y)}{q(y|x)}\right)$ **then**

$x = y$

end

Step-3

$\mathcal{L} = \mathcal{L} + \max(0, E_\theta(v) \cdot E_\theta(x) - E_\theta(u) \cdot E_\theta(v) + \gamma)$ **Step-4**

Update θ by descending the gradients $\nabla_\theta \mathcal{L}$ **Step-5**

end

end

until *Early Stopping or Convergence*;

Experimental Settings

3 representative tasks.

3 graph representation learning algorithms.

5 datasets.

19 experimental settings.

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

Recommendation Results

		MovieLens			Amazon		Alibaba	
		DeepWalk	GCN	GraphSAGE	DeepWalk	GraphSAGE	DeepWalk	GraphSAGE
MRR	$Deg^{0.75}$	0.025±.001	0.062±.001	0.063±.001	0.041±.001	0.057±.001	0.037±.001	0.064±.001
	WRMF	0.022±.001	0.038±.001	0.040±.001	0.034±.001	0.043±.001	0.036±.001	0.057±.002
	RNS	0.031±.001	0.082±.002	0.079±.001	0.046±.003	0.079±.003	0.035±.001	0.078±.003
	PinSAGE	0.036±.001	0.091±.002	0.090±.002	0.057±.004	0.080±.001	0.054±.001	0.081±.001
	WARP	0.041±.003	0.114±.003	0.111±.003	0.061±.001	0.098±.002	0.067±.001	0.106±.001
	DNS	0.040±.003	0.113±.003	0.115±.003	0.063±.001	0.101±.003	0.067±.001	0.090±.002
	IRGAN	0.047±.002	0.111±.002	0.101±.002	0.059±.001	0.091±.001	0.061±.001	0.083±.001
	KBGAN	0.049±.001	0.114±.003	0.100 ±.001	0.060±.001	0.089±.001	0.065±.001	0.087±.002
	MCNS	**0.053±.001**	**0.122±.004**	**0.114±.001**	**0.065±.001**	**0.108±.001**	**0.070±.001**	**0.116±.001**
Hits@30	$Deg^{0.75}$	0.115±.002	0.270±.002	0.270±.001	0.161±.003	0.238±.002	0.138±.003	0.249±.004
	WRMF	0.110±.003	0.187±.002	0.181±.002	0.139±.002	0.188±.001	0.121±.003	0.227±.004
	RNS	0.143±.004	0.362±.004	0.356±.001	0.171±.004	0.317±.004	0.132±.004	0.302±.005
	PinSAGE	0.158±.003	0.379±.005	0.383±.005	0.176±.004	0.333±.005	0.146±.003	0.312±.005
	WARP	0.164±.005	0.406±.002	0.404±.005	0.181±.004	0.340±.004	0.178±.004	0.342±.004
	DNS	0.166±.005	0.404±.006	0.410±.006	0.182±.003	0.358±.004	0.186±.005	0.336±.004
	IRGAN	0.207±.002	0.415±.004	0.408±.004	0.183±.004	0.342±.003	0.175±.003	0.320±.002
	KBGAN	0.198±.003	0.420±.003	0.401±.005	0.181±.003	0.347±.003	0.181±.003	0.331±.004
	MCNS	**0.230±.003**	**0.426 ±.005**	**0.413±.003**	**0.207±.003**	**0.386±.004**	**0.201±.003**	**0.387±.002**

MCNS achieves significant gains of 2%~ 13% over the best baselines.

Link Prediction Results

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

MCNS outperforms all baselines with various graph representation learning methods

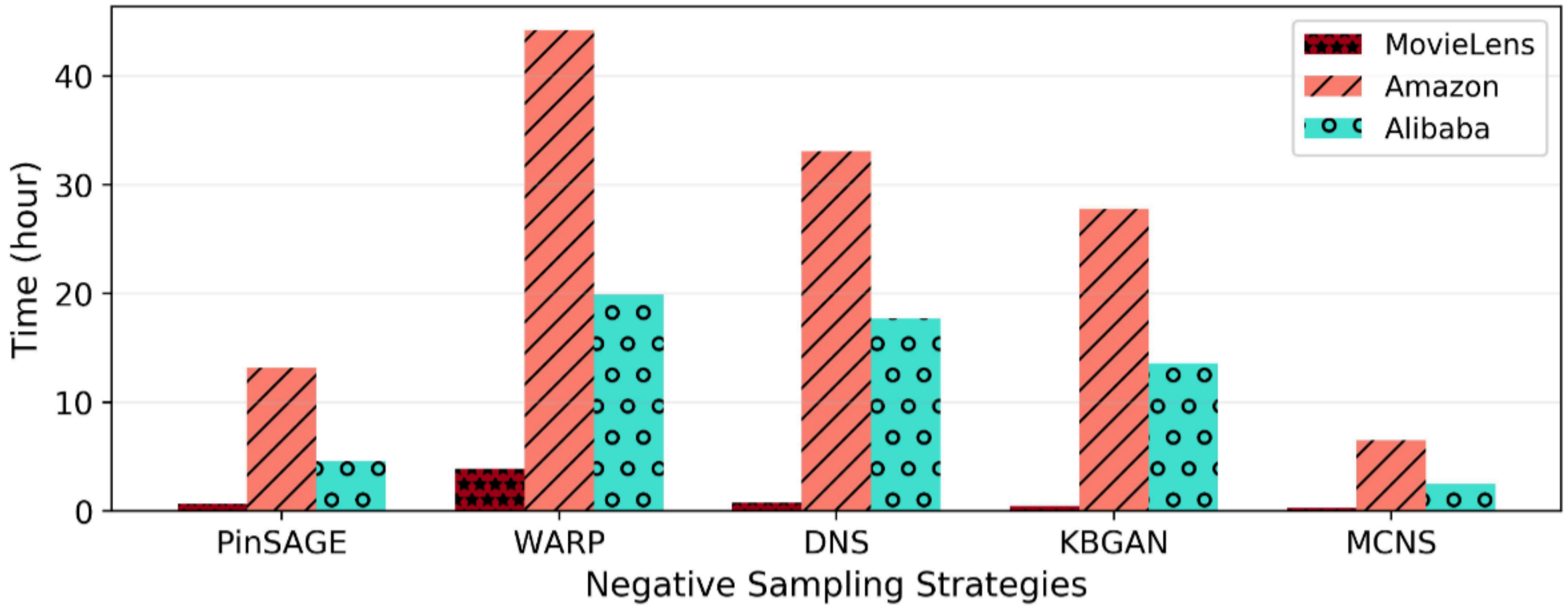
Node Classification Results

	Algorithm	DeepWalk			GCN			GraphSAGE		
	T_R (%)	10	50	90	10	50	90	10	50	90
Micro -F1	$Deg^{0.75}$	31.6	36.6	39.1	36.1	41.8	44.6	35.9	42.1	44.0
	WRMF	30.9	35.8	37.5	34.2	41.4	43.3	34.4	41.0	43.1
	RNS	29.8	34.1	36.0	33.4	40.5	42.3	33.5	39.6	41.6
	PinSAGE	32.0	37.4	40.1	37.2	43.2	45.7	36.9	43.2	45.1
	WARP	35.1	40.3	42.1	39.9	45.8	47.7	40.1	45.5	47.5
	DNS	35.2	40.4	42.5	40.4	46.0	48.6	40.5	46.3	48.5
	IRGAN	34.3	39.6	41.8	39.1	45.2	47.9	38.9	45.0	47.6
	KBGAN	34.6	40.0	42.3	39.5	45.5	48.3	39.6	45.3	48.5
	MCNS	**36.1**	**41.2**	**43.3**	**41.7**	**47.3**	**49.9**	**41.6**	**47.5**	**50.1**

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

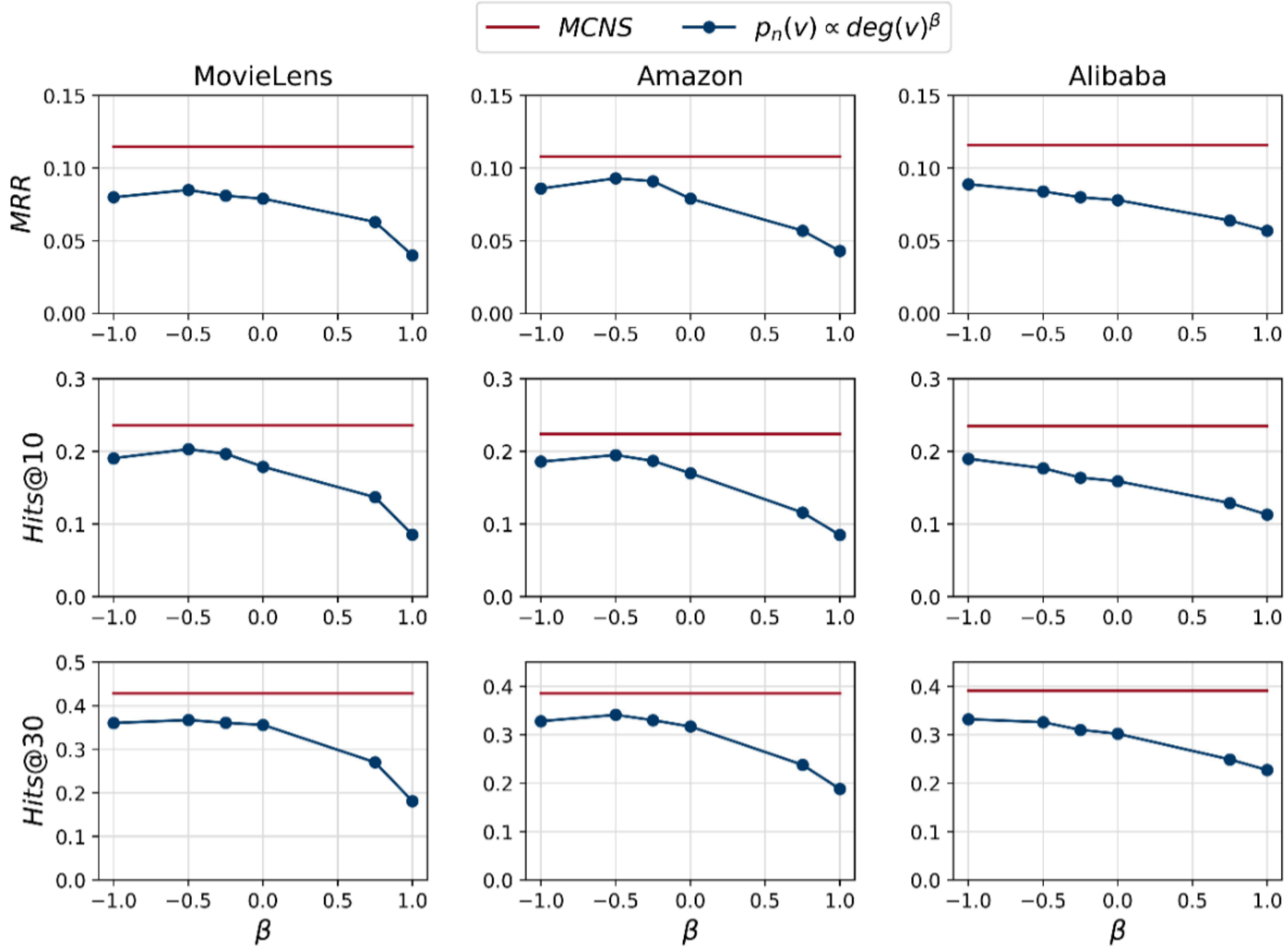
Degree-based NS:

$$p_n(v) \propto deg(v)^\beta$$

- Abscissa: β varies from -1 to 1.
- Results:

  1) Best β varies on datasets.

  2) MCNS naturally adapts to different datasets.

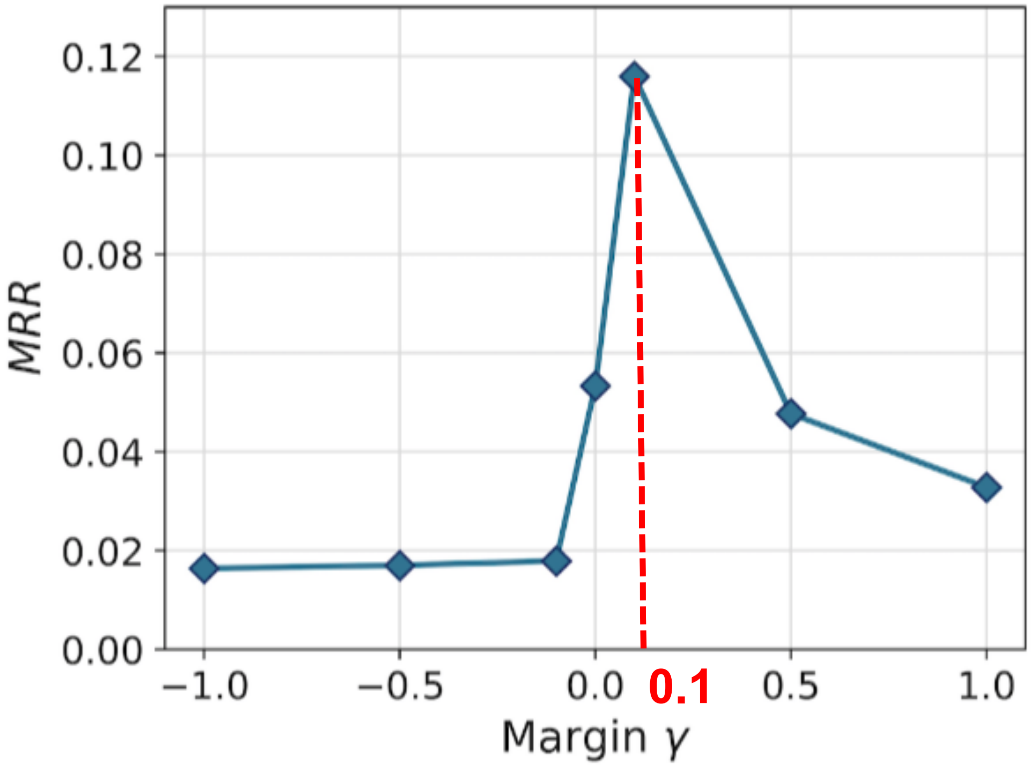


Futher Analysis: Parameter Analysis

- Margin γ:

- the hinge loss begins to take effect when $\gamma \geq 0$

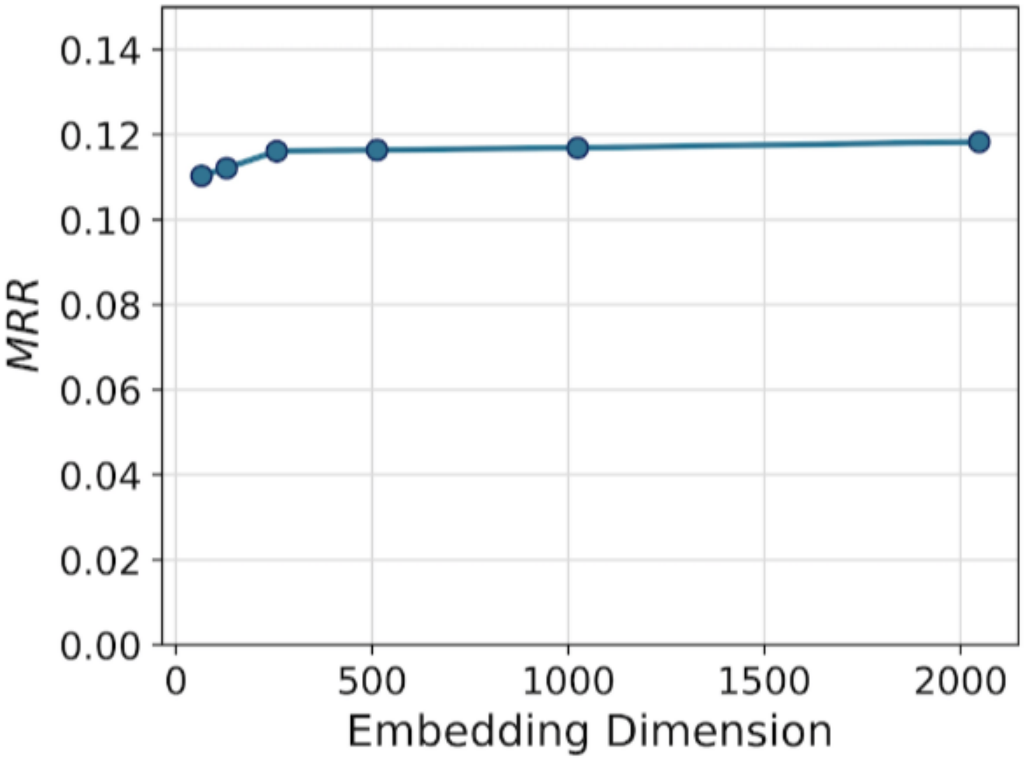
- reaches its optimum at $\gamma \approx 0.1$



- Embedding Dimension:

- set as 512

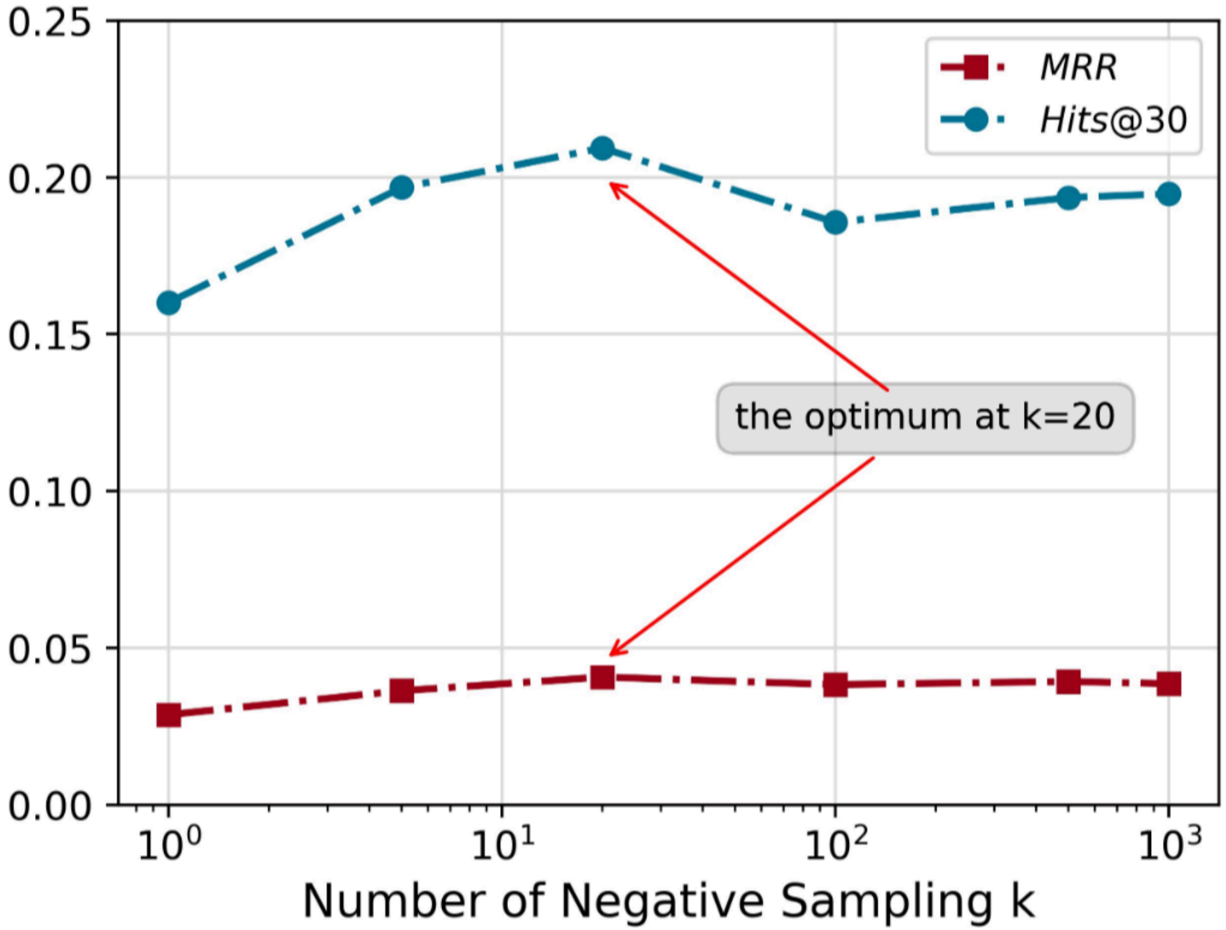
- achieve the trade-off between performtance and time consumption.



Further Understanding

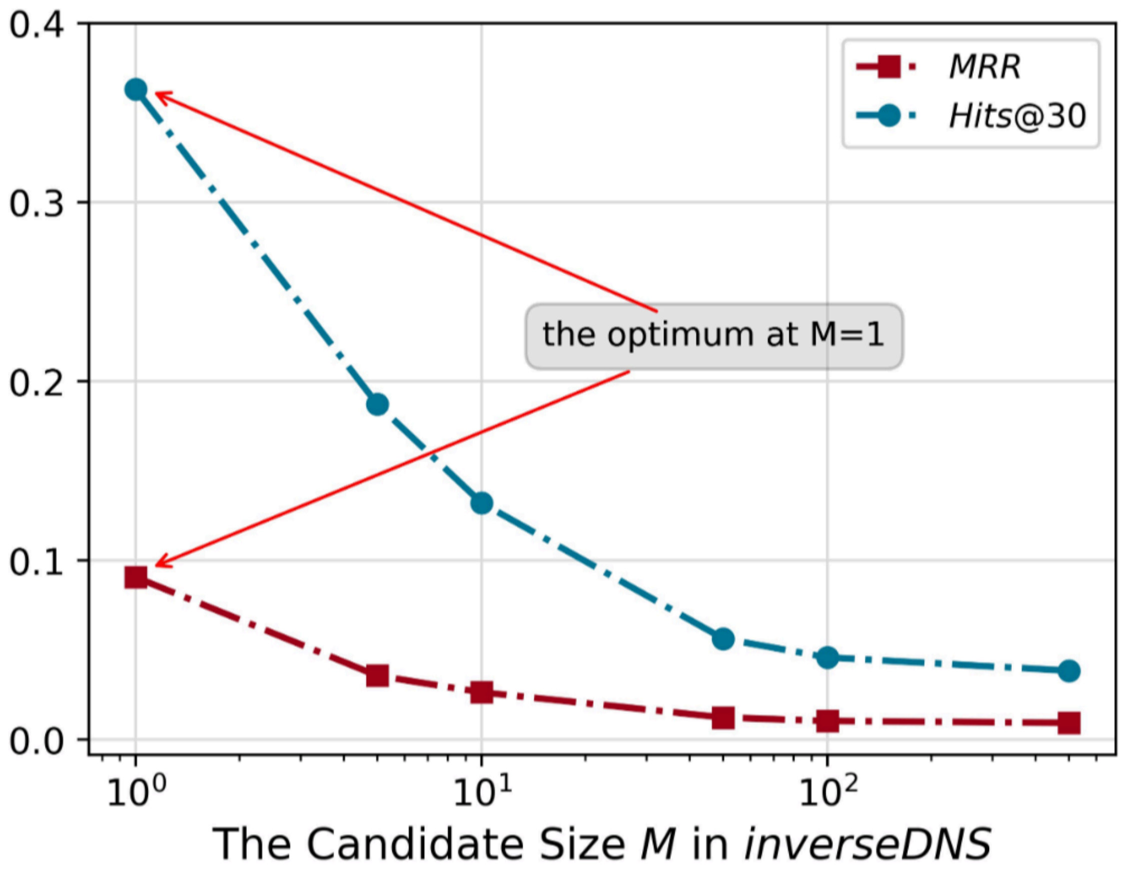
- Whether sampling more negative samples is always helpful ?

- Improve at first: decrease the risk
- Decrease after the optimum: extra bias is added



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