西南大学

含弘光大 · 继往开来

Inductive Representation Learning on Large Graphs

William L. Hamilton* wleif@stanford.edu

Rex Ying* rexying@stanford.edu

Jure Leskovec jure@cs.stanford.edu

Department of Computer Science
Stanford University
Stanford, CA, 94305

NIPS 2017

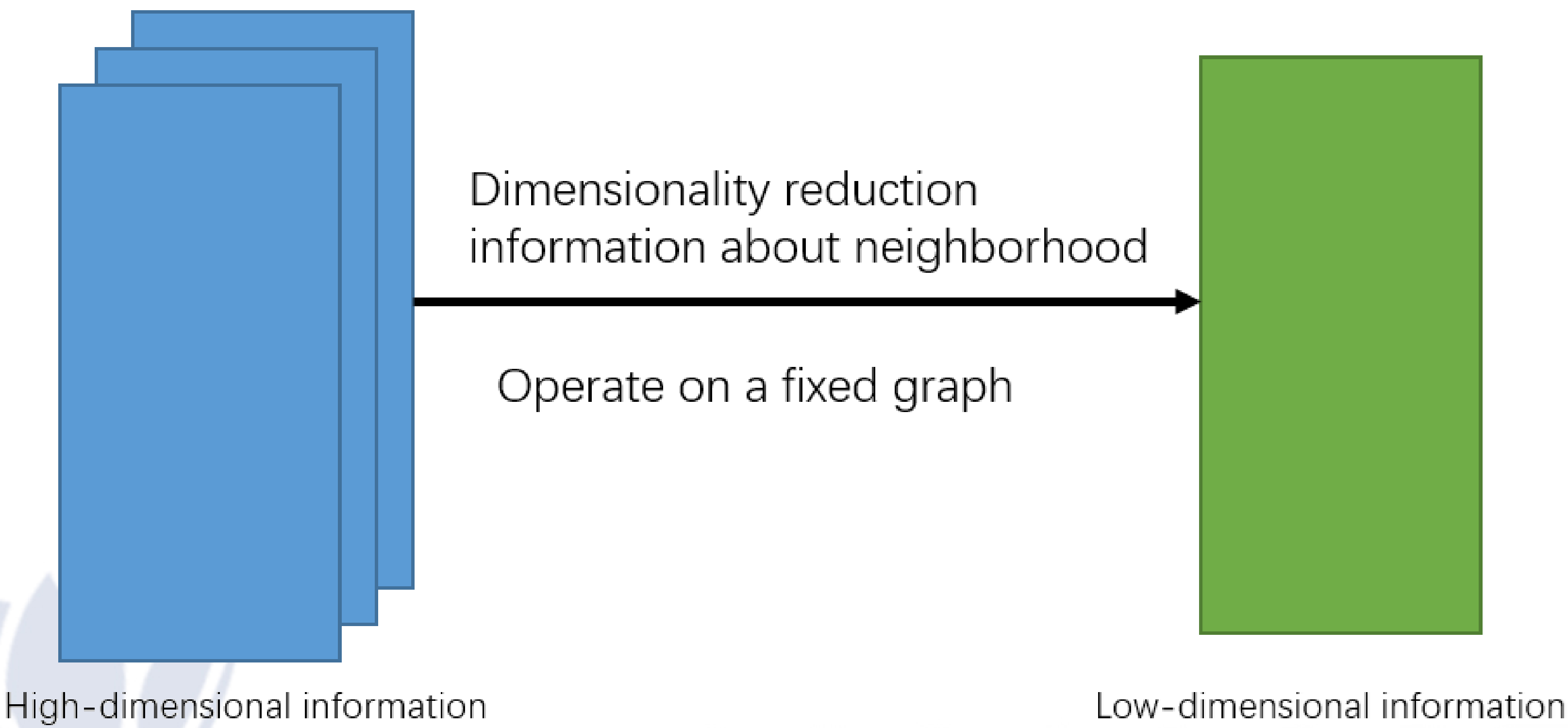
http://snap.stanford.edu/graphsage/

Made by Godder
2019/10/14

SOUTHWEST UNIVERSITY

introduction

- Previous methods:



introduction

- Read-world application problem
  - A fixed graph can not simulate practical model.

e.g. recommender system

introduction

- Read-world application problem
  - A fixed graph can not simulate practical model.
  - Multi-graph and unseen nodes in existing graph.

introduction

- Deficiency of Previous methods:
  - Can not naturally generalize to unseen data.
  - Adding operation in an inductive setting will tend to be computational expensive.

introduction

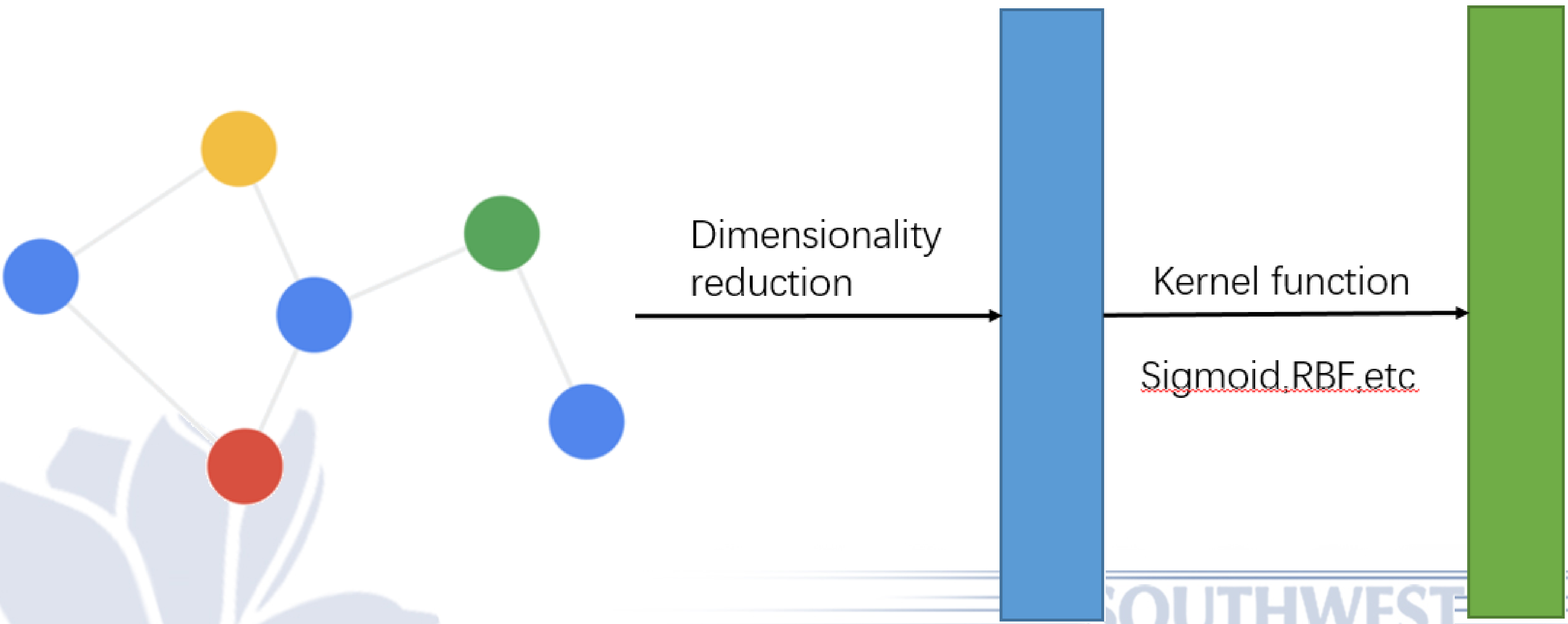
- Work of paper
  - Extend GCN task in unsupervised learning
  - Propose a framework to generalize GCN with using trainable aggregation function.

Related work

- Factorization-based embedding
  - Random walking statistics
  - Matrix factorization-based learning objective

Related work

- Supervised learning: Kernel-based method
  - Graph embedding



Additional content

- Radial basis function(RBF)

  The real-valued function which value only depends on the distance to the origin or other center point c.

$$\phi(\mathbf{x}) = \phi(\|\mathbf{x}\|) \qquad \phi(\mathbf{x}, \mathbf{c}) = \phi(\|\mathbf{x} - \mathbf{c}\|)$$

- Radial basis function(RBF) kernel

  For input x and x', the output can be regard as feature vector of input space

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|_2^2}{2\sigma^2}\right)$$

Related work

- Supervised learning: Kernel-based method
  - Graph embedding
  - Graph kernel

Input: $G1(V_1,E_1)$, $G2(V_2,E_2)$, graph analytic function F

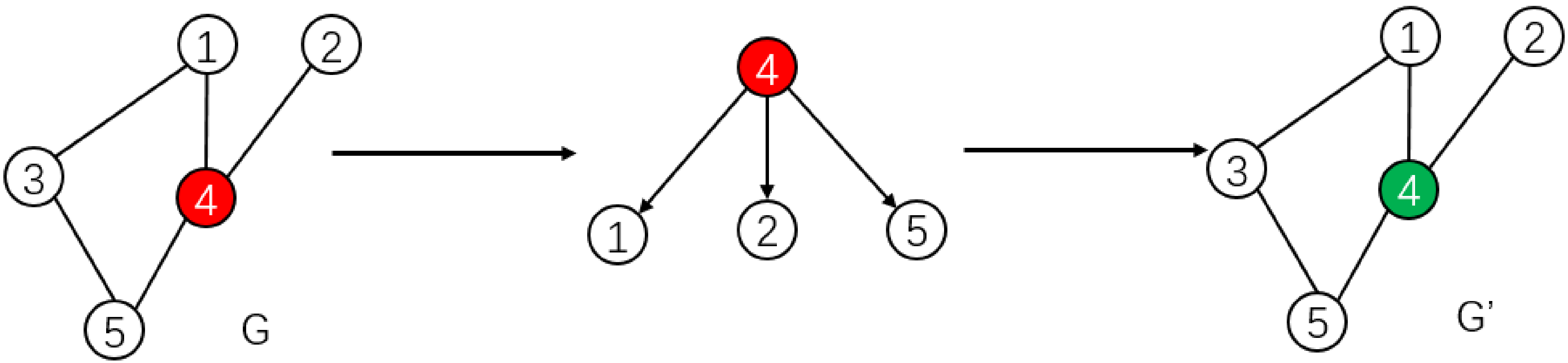
Output: $K_R(G_1,G_2)=\sum_{n_1=1}^{N_1}\sum_{n_2=1}^{N_2}\delta(S_{1,n_1},S_{2,n_2})$

$\mathcal{F}(G_1)=\{S_{1,1},S_{1,2},\ldots,S_{1,N_1}\}$ $\mathcal{F}(G_2)=\{S_{2,1},S_{2,2},\ldots,S_{2,N_2}\}$

Where $\delta(S_{1,n_1},S_{2,n_2})$ is 1 when $S_{1,n1}$ and $S_{2,n2}$ are isomorphic

Additional content

- WL(Weisfeiler-Lehman) kernel



Additional content

- WL(Weisfeiler-Lehman) kernel

With graph analytic function F

$$\mathcal{F}(G) = \{S_1, S_2, \ldots, S_n\}$$

Update code in each node

$$\mathcal{L}^{new} = \{hash(S_1), hash(S_2), \ldots, hash(S_n)\}$$

Decide nodes isomorphism

$$hash(S_i) = hash(S_j) \Rightarrow S_i \simeq S_j$$

Decide graphs isomorphism

$$k(G_1, G_2) = \frac{|\mathcal{L}_1^{new} \cap \mathcal{L}_2^{new}|}{|\mathcal{L}_1^{new} \cup \mathcal{L}_2^{new}|} = \frac{|\mathcal{L}_1^{new} \cap \mathcal{L}_2^{new}|}{|\mathcal{L}_1^{new}| + |\mathcal{L}_2^{new}| - |\mathcal{L}_1^{new} \cap \mathcal{L}_2^{new}|}$$

Related work

- Graph Convolutional Network

  Algorithm requires that the full graph Laplacian is known during training.

Additional content

- GCN中卷积的计算方式
  - 欧拉公式 对于$\theta \in \mathbb{R}$，有$e^{i\theta} = cos\theta + isin\theta$。
  - 傅里叶级数 $f(x) = C + \sum_{n=1}^{\infty}\left(a_n cos(\frac{2\pi n}{T}x) + b_n sin(\frac{2\pi n}{T}x)\right), C \in \mathbb{R}$
  - 傅里叶变换 $\mathcal{F}\{f\}(v) = \int_{\mathbb{R}} f(x)e^{-2\pi ix\cdot v}dx$
  - 傅里叶逆变换 $\mathcal{F}^{-1}\{f\}(x) = \int_{\mathbb{R}} f(v)e^{2\pi ix\cdot v}dv$
  - 卷积公式 $(f * g)(t) = \int_{\mathbb{R}} f(x)g(t-x)dx$
  - 变换后的卷积 $f * g = \mathcal{F}^{-1}\{\mathcal{F}\{f\} \cdot \mathcal{F}\{g\}\}$

Additional content

- GCN中卷积的计算方式
  - Laplacian算子 $\Delta f(x)=\lim_{h\to 0}\frac{f(x+h)-2f(x)+f(x-h)}{h^2}$
  - 图中Laplacian算子 $L=D-A$
  - 标准化 $L=I_N-D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$
  - 图中Laplacian算子分解 $L=U\Lambda U^T$ 其中 Λ 是特征值组成的对角矩阵 $U=[u_1\ldots u_n]$
  - 图中傅里叶变换 $\mathcal{GF}\{f\}(\lambda_l)=\sum_{i=1}^{n}f(i)u_l^*(i)$ $\mathcal{GF}\{x\}=U^Tx$ $x=(f(1)\ldots f(n))\in\mathbb{R}^n$
  - 图中傅里叶逆变换 $\mathcal{IGF}\{\hat{f}\}(i)=\sum_{l=0}^{n-1}\hat{f}(\lambda_l)u_l(i)$ $\mathcal{IGF}\{x\}=Ux$

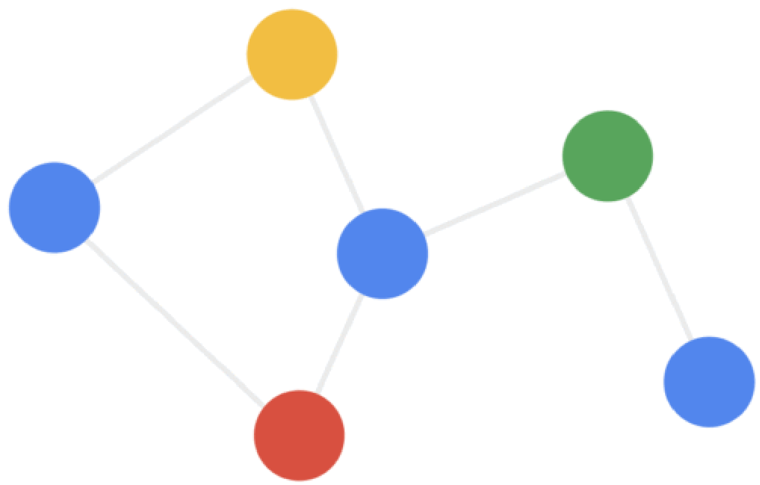
Additional content

- GCN中卷积的计算方式
  - 图卷积公式 $g * x = U(U^T g \cdot U^T x)$
  - 利用Laplacian矩阵实现类似CNN的局部性
    定义g为Laplacian矩阵的函数g(L)
  - 改写后 $g_\theta * x = U g_\theta U^T x = U g_{\theta'}(\Lambda) U^T x$
  - 化简
$$g_{\theta'} * x \approx \theta(I_N + L)x$$
$$= \theta(I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}})x$$
  - 加上激活函数 $H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}} \tilde{A} \tilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)})$

Additional content

- GCN如何传导

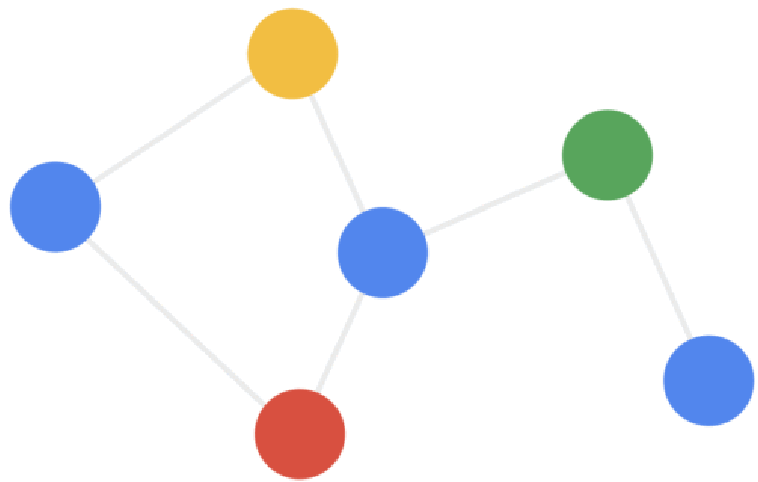
Step1 send



Additional content

- GCN如何传导

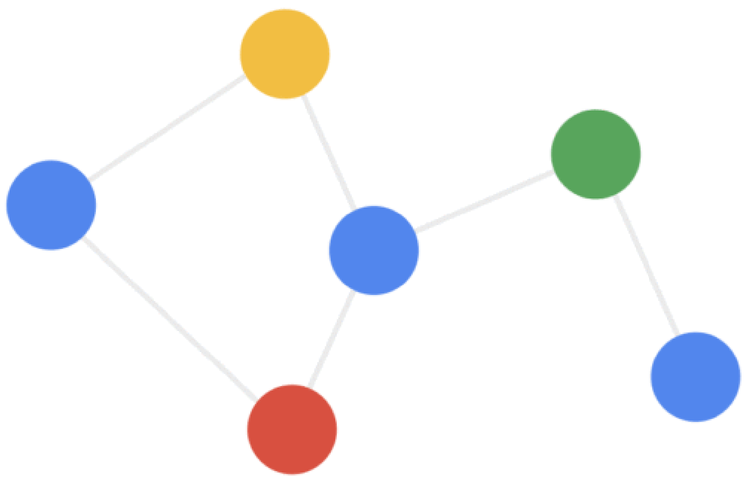
Step2 receive



Additional content

- GCN如何传导

Step3 transform



Additional content

欧拉公式推导

n阶泰勒公式

$$f(x)=\frac{f(x_0)}{0!}+\frac{f'(x_0)}{1!}(x-x_0)+\frac{f''(x_0)}{2!}(x-x_0)^2+\ldots+\frac{f^{(n)}(x_0)}{n!}(x-x_0)^n+R_n(x)$$

$$e^x=1+x+\frac{1}{2!}x^2+\frac{1}{3!}x^3+\cdots$$

$$sin(x)=x-\frac{1}{3!}x^3+\frac{1}{5!}x^5+\cdots$$

$$cos(x)=1-\frac{1}{2!}x^2+\frac{1}{4!}x^4+\cdots$$ 将$x=i\theta$代入e

$$\begin{aligned}e^{i\theta}&=1+i\theta+\frac{(i\theta)^2}{2!}+\frac{(i\theta)^3}{3!}+\frac{(i\theta)^4}{4!}+\frac{(i\theta)^5}{5!}+\frac{(i\theta)^6}{6!}+\frac{(i\theta)^7}{7!}+\frac{(i\theta)^8}{8!}+\cdots\\&=1+i\theta-\frac{\theta^2}{2!}-\frac{i\theta^3}{3!}+\frac{\theta^4}{4!}+\frac{i\theta^5}{5!}-\frac{\theta^6}{6!}-\frac{i\theta^7}{7!}+\frac{\theta^8}{8!}+\cdots\\&=\left(1-\frac{\theta^2}{2!}+\frac{\theta^4}{4!}-\frac{\theta^6}{6!}+\frac{\theta^8}{8!}-\cdots\right)+i\left(\theta-\frac{\theta^3}{3!}+\frac{\theta^5}{5!}-\frac{\theta^7}{7!}+\cdots\right)\\&=\cos\theta+i\sin\theta\end{aligned}$$

Additional content

卷积傅里叶变换推导

$$h(z) = \int_{\mathbb{R}} f(x)g(z-x)dx$$

$$\begin{aligned}\mathcal{F}\{f*g\}(v) &= \mathcal{F}\{h\}(v) \\ &= \int_{\mathbb{R}} h(z)e^{-2\pi i z\cdot v}dz \\ &= \int_{\mathbb{R}}\int_{\mathbb{R}} f(x)g(z-x)e^{-2\pi i z\cdot v}dxdz \\ &= \int_{\mathbb{R}} f(x)(\int_{\mathbb{R}} g(z-x)e^{-2\pi i z\cdot v}dz)dx\end{aligned}$$

带入 $y = z - x$；$dy = dz$

$$\begin{aligned}\mathcal{F}\{f*g\}(v) &= \int_{\mathbb{R}} f(x)(\int_{\mathbb{R}} g(y)e^{-2\pi i(y+x)\cdot v}dy)dx \\ &= \int_{\mathbb{R}} f(x)e^{-2\pi i x\cdot v}(\int_{\mathbb{R}} g(y)e^{-2\pi i y\cdot v}dy)dx \\ &= \int_{\mathbb{R}} f(x)e^{-2\pi i x\cdot v}dx\int_{\mathbb{R}} g(y)e^{-2\pi i y\cdot v}dy \\ &= \mathcal{F}\{f\}(v)\cdot\mathcal{F}\{g\}(v)\end{aligned}$$

Additional content

图卷积公式化简

化简前图卷积公式

$$g_\theta * x = U g_\theta U^T x = U g_{\theta'}(\Lambda) U^T x$$

$$g_{\theta'}(\Lambda) \approx \sum_{k=0}^{K} \theta'_k T_k(\tilde{\Lambda})$$

$$\tilde{\Lambda} = \frac{2}{\lambda_{\max}} \Lambda - I_N$$

$\lambda_{\max}$ denotes the largest eigenvalue of L

其中 T_k 是Chebyshev多项式。这里可以把简单 $g_\theta(\Lambda)$ 简单看成是 Λ 的多项式。

因为 $U \Lambda^k U^T = (U \Lambda U^T)^k = L^k$

$$g_{\theta'}(\Lambda) \approx \sum_{k=0}^{K} \theta'_k T_k(\tilde{L})$$

$$\tilde{L} = \frac{2}{\lambda_{\max}} L - I_N$$

设定 $K = 1$ 那卷积公式可以简化为

$$\begin{aligned} g_{\theta'} * x &\approx \theta(I_N + L)x \\ &= \theta(I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}})x \end{aligned}$$

GraphSAGE

- Focus on feature-rich graph
- Learning the topological structure of each node's neighborhood.
- Learning the distribution of node features

GraphSAGE

- Embedding generation algorithm

Algorithm 1: GraphSAGE embedding generation (i.e., forward propagation) algorithm

---

Input : Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth K; weight matrices $\mathbf{W}^k, \forall k \in \{1, ..., K\}$; non-linearity σ; differentiable aggregator functions $\text{AGGREGATE}_k, \forall k \in \{1, ..., K\}$; neighborhood function $\mathcal{N} : v \to 2^{\mathcal{V}}$

Output : Vector representations $\mathbf{z}_v$ for all $v \in \mathcal{V}$

$\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$;
for $k = 1...K$ **do**
  for $v \in \mathcal{V}$ **do**
    $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;
    $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$
  end
  $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$
end
$\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$

GraphSAGE

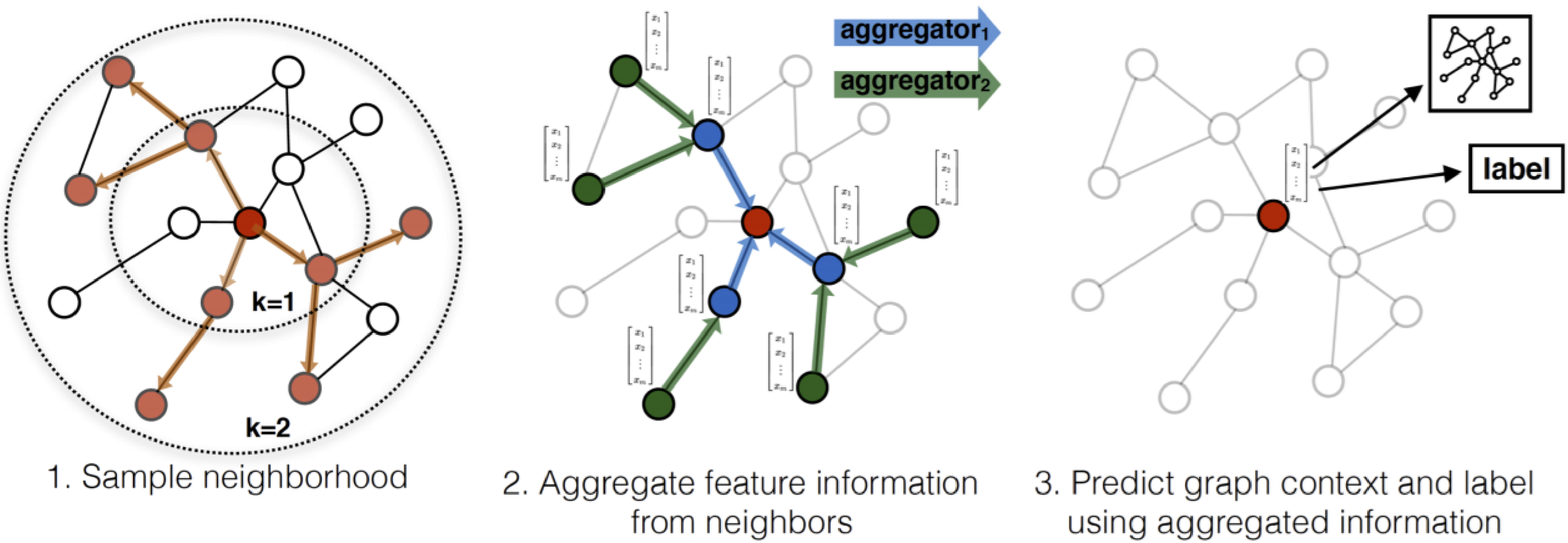


Figure 1: Visual illustration of the GraphSAGE sample and aggregate approach.

GraphSAGE

- Relation to the WL Isomorphism Test
  - Algorithm 1 is an instance of the WL isomorphism test

  Set K = |V|

  Set weight matrices as the identity

  Use appropriate hash function as aggregator function

  - Algorithm 1 is a continuous approximation to WL test

  Replace hash function with trainable neural network

GraphSAGE

- Learning the parameters of GraphSAGE
  - Unsupervised loss

$$J_{\mathcal{G}}(\mathbf{z}_u) = -\log\left(\sigma(\mathbf{z}_u^\top \mathbf{z}_v)\right) - Q \cdot \mathbb{E}_{v_n \sim P_n(v)} \log\left(\sigma(-\mathbf{z}_u^\top \mathbf{z}_{v_n})\right)$$

v is node that co-occurs near u on fixed-length random walk. σ is sigmoid function. Q is the number of negative smaple(non-adjacent node in fixed-length random walk)

  - Supervised loss

Cross-entropy loss etc

GraphSAGE

- Aggregator Architectures
  - Mean aggregator

$$h_{N(v)}^{k} = mean(\{h_u^{k-1}, u \in N(v)\})$$

  - Mean aggregator base GCN

$$\mathbf{h}_v^k \leftarrow \sigma(\mathbf{W} \cdot \text{MEAN}(\{\mathbf{h}_v^{k-1}\} \cup \{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$$

GraphSAGE

- Aggregator Architectures
  - LSTM aggregator

    Adapt LSTM to operate on an unordered set.(random permutation of the node's neighbors)
  - Pooling aggregator

    Each neighbor's vector is independently fed through a FC

$$\text{AGGREGATE}_k^{\text{pool}} = \max(\{\sigma\left(\mathbf{W}_{\text{pool}}\mathbf{h}_{u_i}^k + \mathbf{b}\right), \forall u_i \in \mathcal{N}(v)\})$$

Experiments

- Setting
  - Hyper-parameter

  K = 2(aggregate 2 times). S_1=25(sample number in first iteration). S_2=10(sampl number in second iteration)

  - Optimizer

  Adam

  - Supervised loss function

  Cross-entropy

  - Batch size

  512

Experiments

- Mini-batch load
  - Uniformly sample a fixed-size set of neighbors.
  - Draw different uniform samples at each iteration

Experiments

- Dataset & Task
  - Citation data (node classification)

    Contain 302,424 nodes with an average degree of 9.15.
    300-dimensional word vectors as node feature
  - Reddit data (node classification)

    Contain 232,965 nodes with an average degree 492.

    300-dimensional word vectors as node feature
  - Protein-protein interactions (graph classification)

    Include 121 labels and average graph contains 2373 nodes, with an average degree of 28.8. train with 20 graphs.

Experiments

- Evaluation protocol
  - F1 score

$$F1 = \frac{2TP}{2TP + FN + FP} = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

  - Time cost

Experiments

- Performance

	Citation		Reddit		PPI	
Name	Unsup. F1	Sup. F1	Unsup. F1	Sup. F1	Unsup. F1	Sup. F1
Random	0.206	0.206	0.043	0.042	0.396	0.396
Raw features	0.575	0.575	0.585	0.585	0.422	0.422
DeepWalk	0.565	0.565	0.324	0.324	—	—
DeepWalk + features	0.701	0.701	0.691	0.691	—	—
GraphSAGE-GCN	0.742	0.772	**0.908**	0.930	0.465	0.500
GraphSAGE-mean	0.778	0.820	0.897	0.950	0.486	0.598
GraphSAGE-LSTM	0.788	0.832	**0.907**	**0.954**	0.482	**0.612**
GraphSAGE-pool	**0.798**	**0.839**	0.892	0.948	**0.502**	0.600
% gain over feat.	39%	46%	55%	63%	19%	45%

Experiments

- Performance

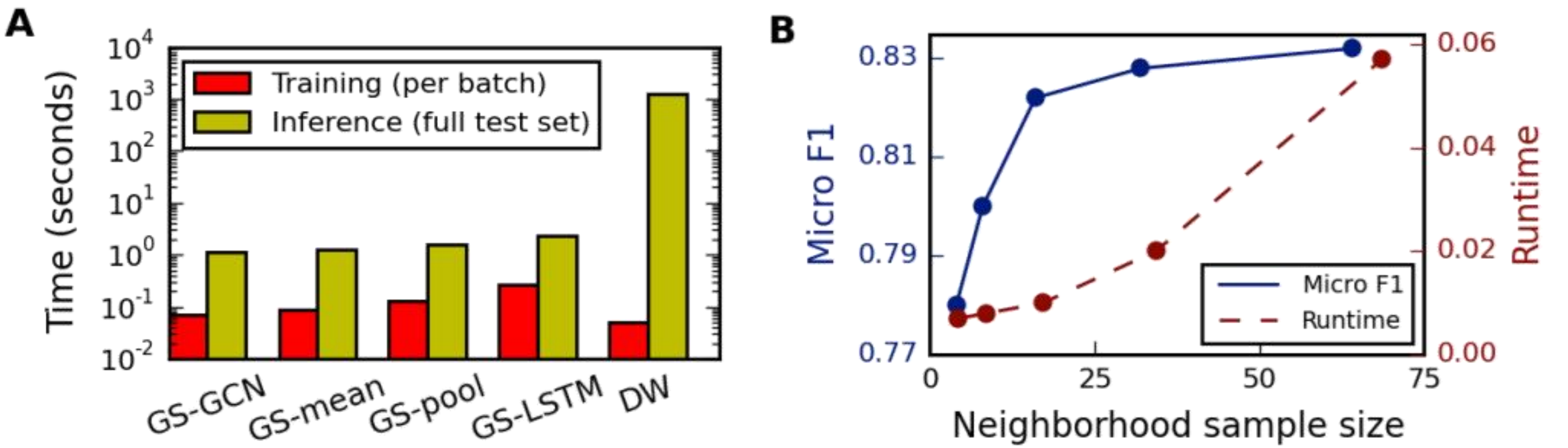


Figure 2: **A**: Timing experiments on Reddit data, with training batches of size 512 and inference on the full test set (79,534 nodes). **B**: Model performance with respect to the size of the sampled neighborhood, where the "neighborhood sample size" refers to the number of neighbors sampled at each depth for $K = 2$ with $S_1 = S_2$ (on the citation data using GraphSAGE-mean).

Reference

- Haussler, D. "Convolution kernels on discrete structures." *Tech Rep 7(1999)*:95-114
- N. Shervashidze, P. Schweitzer, E. J. v. Leeuwen, K. Mehlhorn, and K. M. Borgwardt. Weisfeilerlehman graph kernels. Journal of Machine Learning Research, 12:2539–2561, 2011.