최신컴퓨터특강1
May 8, 2019

빅데이터 분석을 위한 차원 축소

임성수 교수

Data Intelligence Lab.

목차

- **차원 축소** ⇐
- **그래프 임베딩**
- **최신 연구 결과 소개**
  1. BlackHole embedding (ICDE 2016)
  2. Motif-based embedding (JSTAT 2016)
  3. LinkBlackHole* using link embedding (ICDE 2014, TKDE 2019+)

차원 축소

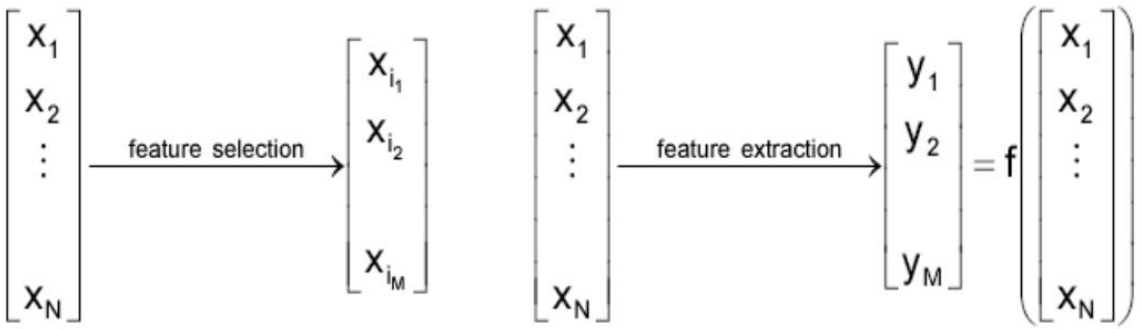
- **복잡한 세상, 복잡한 데이터**
  - 데이터는 다양한 속성으로 이루어진 수많은 레코드의 모임
  - 속성(attribute)은 각 레코드의 성질 및 **특징(feature)**을 표현
  - 속성의 개수를 **차원(dimension)**이라고 부름 → **고차원 데이터 등장**

	A	B	C	D	E	F
1	Country	Salesperson	Order Date	OrderID	Units	Order Amount
2	USA	Fuller	1/01/2011	10392	13	1,440.00
3	UK	Gloucester	2/01/2011	10397	17	716.72
4	UK	Bromley	2/01/2011	10771	18	344.00
5	USA	Finchley	3/01/2011	10393	16	2,556.95
6	USA	Finchley	3/01/2011	10394	10	442.00
7	UK	Gillingham	3/01/2011	10395	9	2,122.92
8	USA	Finchley	6/01/2011	10396	7	1,903.80
9	USA	Callahan	8/01/2011	10399	17	1,765.60
10	USA	Fuller	8/01/2011	10404	7	1,591.25
11	USA	Fuller	9/01/2011	10398	11	2,505.60
12	USA	Coghill	9/01/2011	10403	18	855.01
13	USA	Finchley	10/01/2011	10401	7	3,868.60
14	USA	Callahan	10/01/2011	10402	11	2,713.50
15	UK	Rayleigh	13/01/2011	10406	15	1,830.78
16	USA	Callahan	14/01/2011	10408	10	1,622.40
17	USA	Farnham	14/01/2011	10409	19	319.20
18	USA	Farnham	15/01/2011	10410	16	802.00

차원 축소

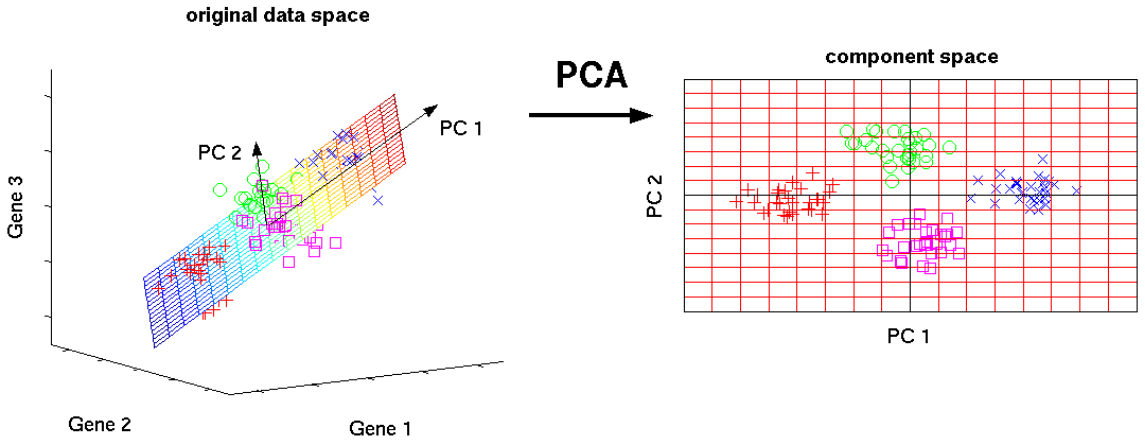
- **차원의 저주(Curse of dimensionality)**
  - 데이터의 차원이 높아질수록 문제가 발생함
  - 차원이 증가하면 공간 대비 밀도가 기하급수적으로 감소
  - 중요도가 떨어지는 특징 사용 시 **과적합(overfitting)**에 빠질 수 있음

- **데이터의 차원 낮추기**
  - 특징 선택(feature selection) : 전체 특징 중 중요한 일부를 선택
  - **특징 추출(feature extraction)** : 전체 특징에 대한 함수로 중요 특징 추출

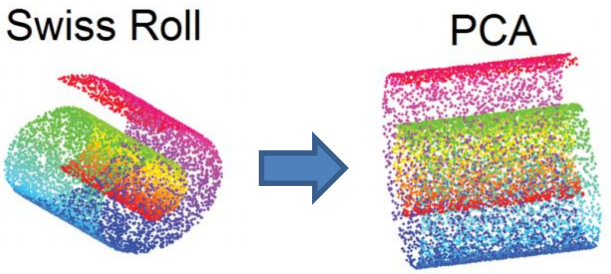


차원 축소

- **차원 축소 방법**
  - **선형 방법 (Linear methods)** : 특징 추출 함수가 선형인 경우로 projection에 해당
    예) PCA, SVD, LDA, Classical MDS, NMF, etc.



Principal component analysis



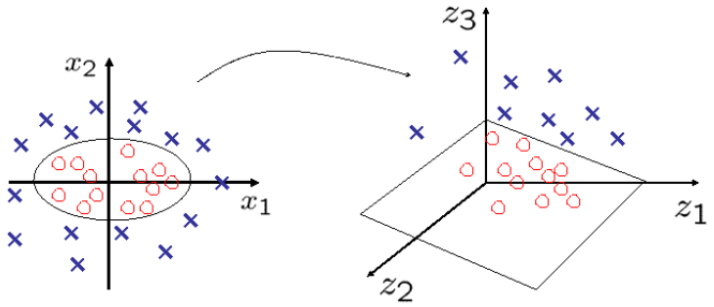
PCA: bad example

* 참고자료: Linear dimensionality reduction: a survey, JMLR 2015 [LINK]
Decomposing signals in components, scikit-learn [LINK]

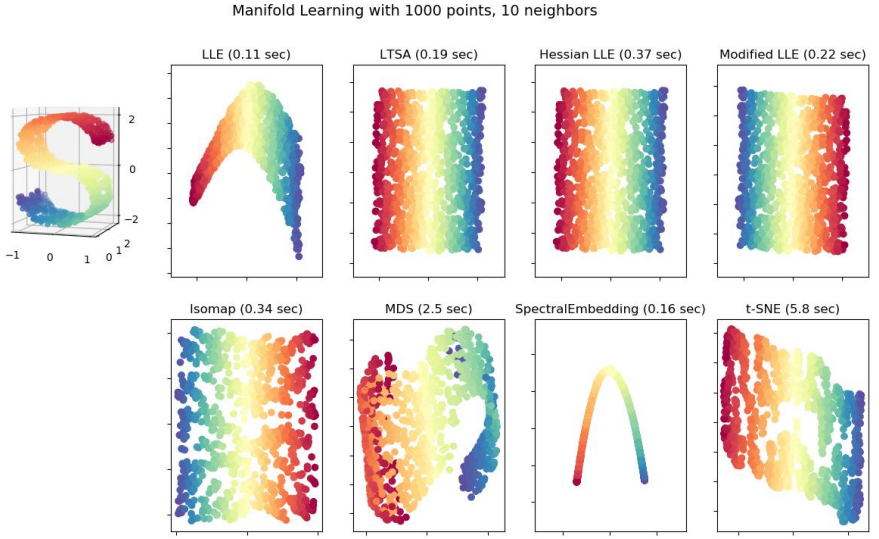
차원 축소

- **차원 축소 방법**
  - **비선형 방법 (nonlinear methods)**: 선형 방법의 한계를 해결하기 위한 비선형 방법

    예) Kernel PCA, Manifold learning (Isomap, LLE, LE, **t-SNE**), etc.



Kernel trick + PCA



Manifold learning

* 참고자료: Manifold learning, scikit-learn [LINK]

차원 축소

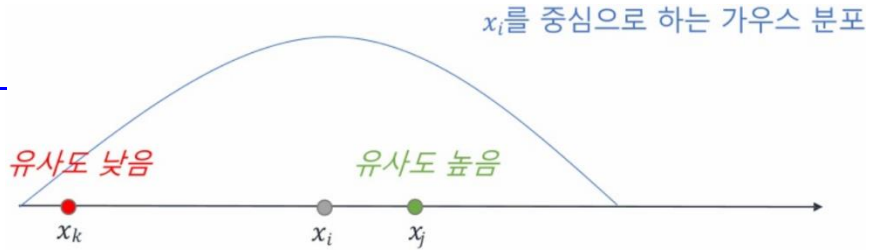
- **차원 축소 방법**
  - SNE & t-SNE: 데이터 간 유사도를 고려한 embedding 방법
  - **SNE**: 1. 고차원 데이터 값 x_i에 대한 데이터 값 x_j의 조건부 확률 정의

$$p_{j|i} = \frac{\exp\left(-\|x_i - x_j\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\|x_i - x_j\|^2 / 2\sigma_i^2\right)}$$: x_i 기준으로 x_j가 가까우면 더 높은 확률

2. 저차원 벡터 y_i, y_j에 대해 정의되는 $q_{j|i}$들에 대해 $p_{j|i}$들과의 거리

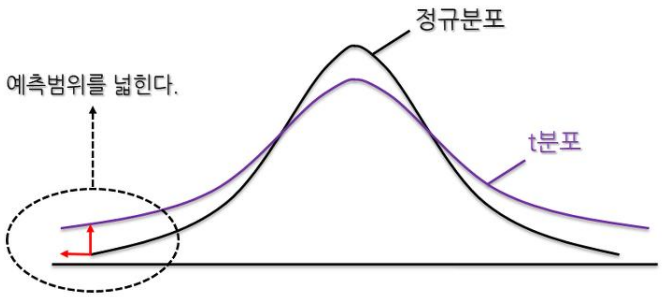
$\sum_i KL(P_i|Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$가 최소화되도록 저차원 좌표 탐색

* 참고자료: 엔트로피 및 Kullback-Leibler 거리 설명 [LINK]
t-SNE를 이용한 시각화 [LINK]

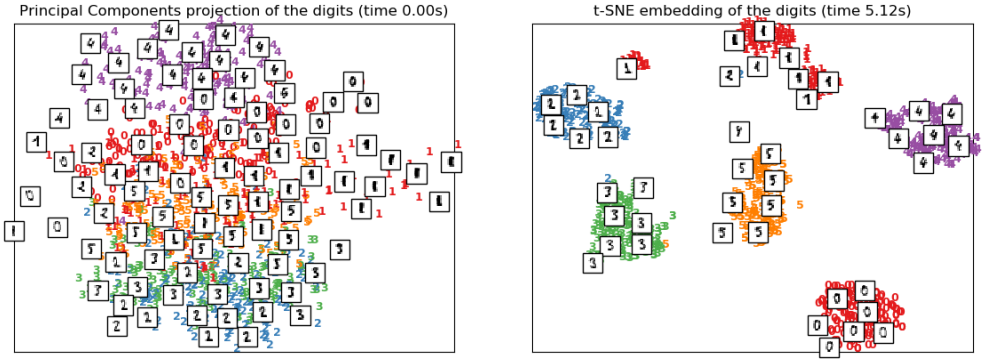


차원 축소

- **차원 축소 방법**
  - t-SNE: 조건부 확률분포가 아닌 결합 확률분포를 사용, JMLR 2008, [LINK]
    → 가까운 경우 더 가깝게, 먼 경우 더 멀게
    → 대칭적이기 때문에 최적화 식이 비교적 간편
    → 근사적으로 빠르게 계산 가능 (tree approx.), JMLR 2014, [Link]



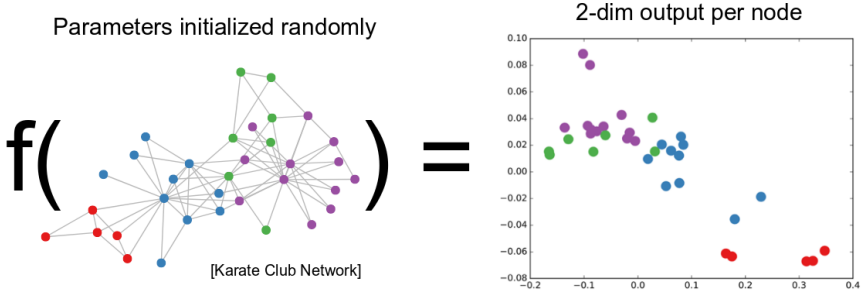
정규분포와 t-분포 [LINK]



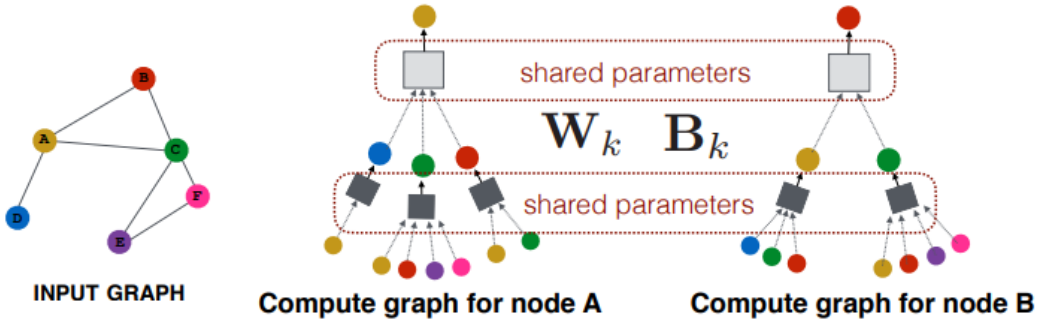
손글씨 인식하기: PCA vs t-SNE [LINK]

차원 축소

- **차원 축소 방법**
  - 그래프 표현 학습(Graph representation learning): 그래프 데이터의 차원 축소 방법
    예) Graph embedding(node2vec, DeepWalk), Graph neural networks, etc.



Graph embedding



Graph convolutional networks

* 참고자료: Representation learning on graphs, IEEE Data Eng. Bulletin 2017 [LINK]
A comprehensive survey of graph embedding, IEEE TKDE 2018 [LINK]
A survey on network embedding, IEEE TKDE 2018 [LINK]

목차

- 차원 축소
- 그래프 임베딩 ⟸
- 최신 연구 결과 소개
  1. BlackHole embedding (ICDE 2016)
  2. Motif-based embedding (JSTAT 2016)
  3. LinkBlackHole* using link embedding (ICDE 2014, TKDE 2019+)

그래프 임베딩

- **그래프 데이터 분석**
  - 복잡하게 연결된 사회 속에서 다양한 종류의 요소가 다양한 형태의 관계를 가짐
  - 연결(link)을 분석하기 위해 **네트워크 과학(network science)**이 쓰임

  - **소셜 네트워크**

    Facebook(~2billion active users)

    Wechat(~1billion active users)

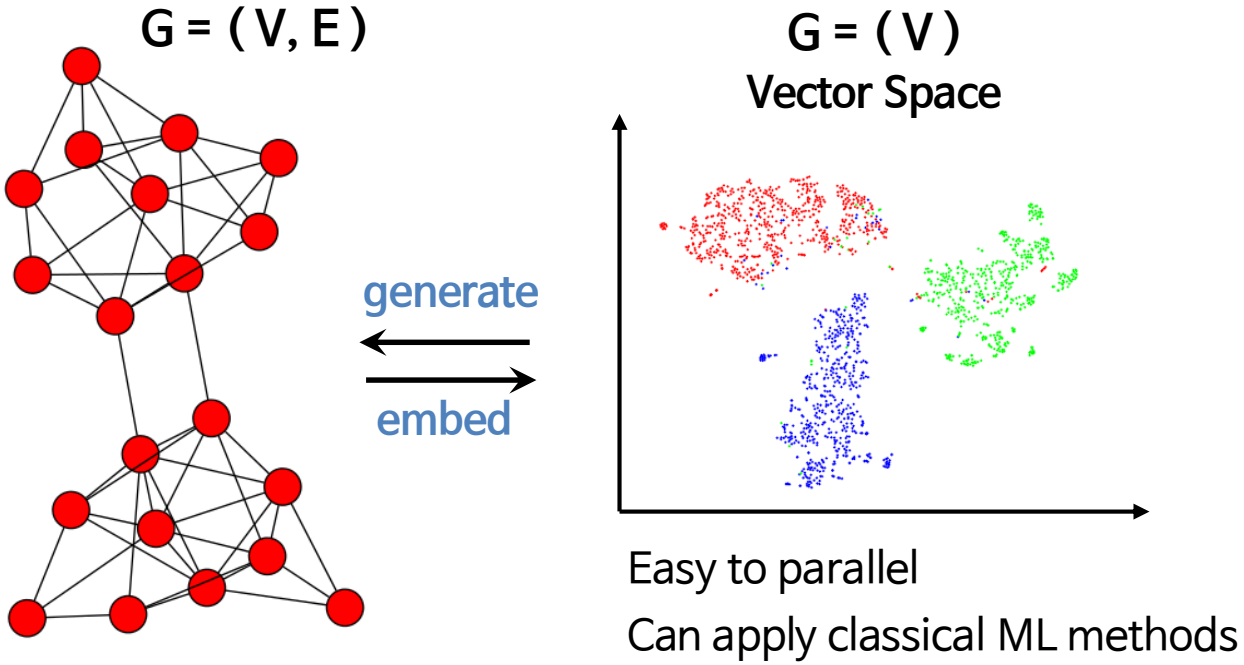
  - **전자 상거래 네트워크**

    Amazon(400M customers, 400M products)

    JD.com(300M customers, 800M products)

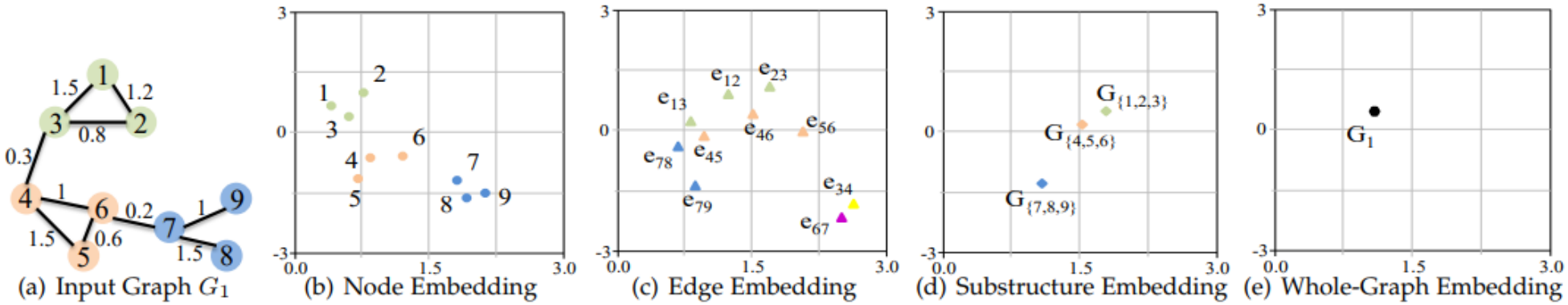
그래프 임베딩

- **그래프 임베딩**
  - 그래프를 벡터 공간으로 임베딩하여 그래프 **데이터 분석이 더 쉽도록** 변환
  - 다양한 **네트워크 추론(network inference)** 문제를 벡터 공간에서 해결



그래프 임베딩

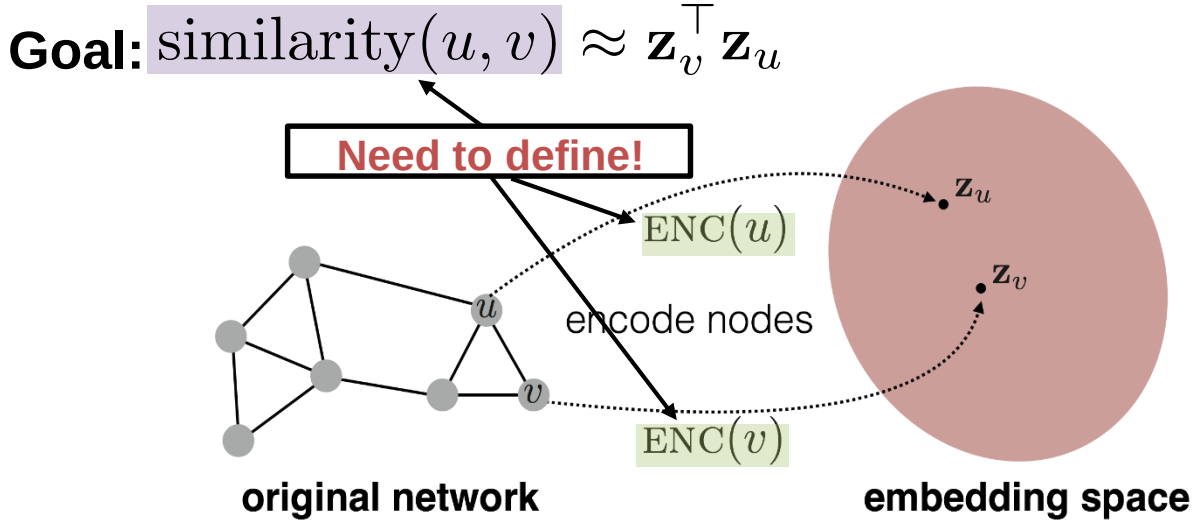
- **좋은 그래프 임베딩**
  - 대상: 임베딩의 대상은 노드, 연결, 부분 구조 등 다양하게 정의 가능
  - 목표: 그래프의 **구조(structure) 및 성질(property)이 유지**되는 변환
  - 활용: 그래프 데이터 분석을 보다 효과적으로 수행



Graph embedding: a toy example from TKDE 2018 [LINK]

그래프 임베딩

- **좋은 노드 임베딩**
  - 대상: 임베딩의 대상은 노드
  - 목표: 그래프의 **노드 간 유사도(similarity)가 유지**되는 변환
  - 활용: 그래프 데이터 분석을 보다 효과적으로 수행



Vertex embedding: a toy example from WWW tutorial 2018 [LINK]

그래프 임베딩

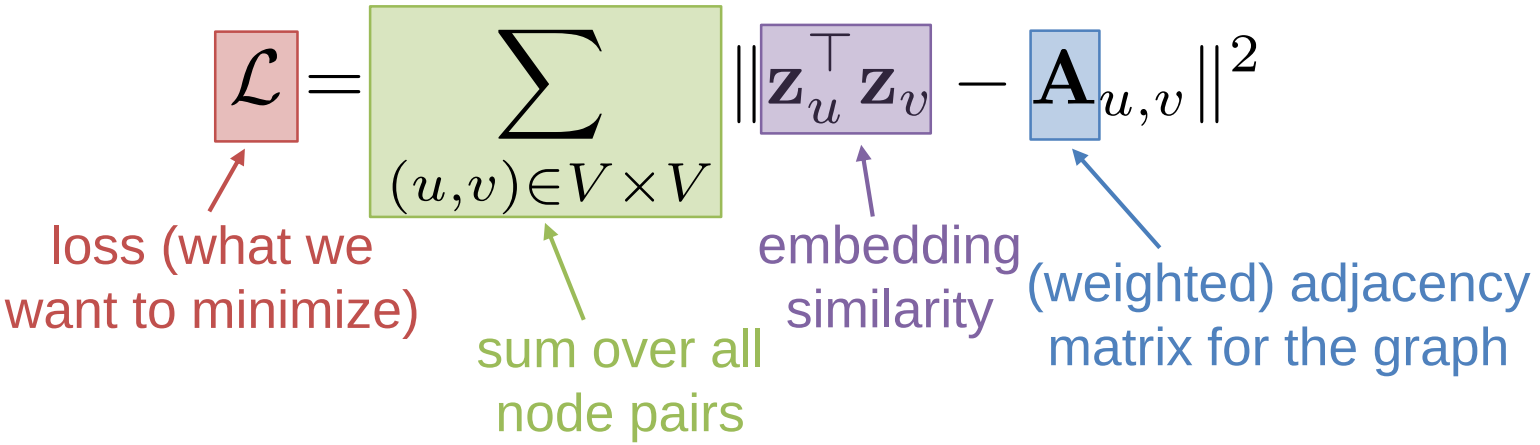
- **좋은 노드 임베딩을 위하여**
  - 해야 할 일
    1. Encoder 정의: 임베딩하는 방식을 정하기
    2. 노드 간의 **유사도 정의**: 임베딩을 평가하는 방식 정하기
    3. 노드 간의 유사도와 임베딩 후 벡터 간의 **유사도의 차이를 최소화**

  - 유사도의 예
    1. Adjacency-based similarity
    2. Multi-hop similarity
    3. Random walk approaches

* 참고자료: Representation learning on graphs, IEEE Data Eng. Bulletin 2017 [LINK]
WWW 2018 tutorial: representation learning on graphs [LINK]

그래프 임베딩

- **좋은 노드 임베딩의 예**
  - 최신 기법: Adjacency-based similarity 활용, WWW 2013 [LINK]
    - → 노드 간의 유사도: 연결 가중치(edge weight)
    - → 노드 간의 연결이 있을 때 임베딩 후 벡터 간의 유사도와 차이 최소화



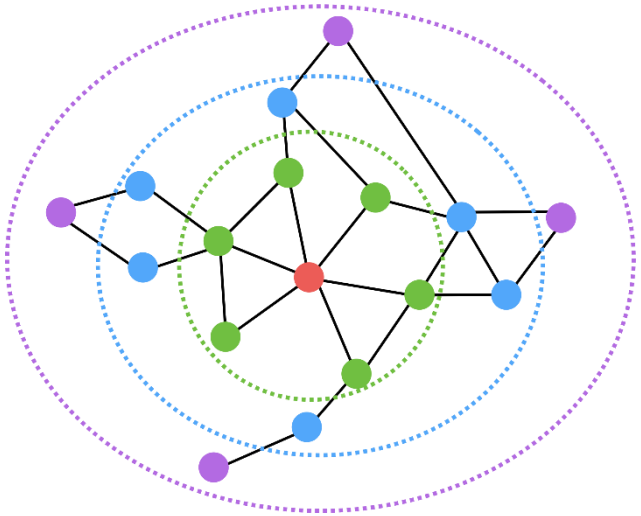
그래프 임베딩

- **좋은 노드 임베딩의 예**
  - 최신 기법: Multi-hop similarity 활용,

    GraRap, CIKM 2015 [LINK], HOPE, KDD 2016 [LINK]

    → 노드 간의 유사도: 연결 가중치 대신 아래와 같이 k-hop을 고려

    → 노드 간의 연결이 있을 때 임베딩 후 벡터 간의 유사도와 차이 최소화



$$\mathcal{L} = \sum_{(u,v)\in V\times V} \left\| \mathbf{z}_u^\top \mathbf{z}_v - \mathbf{A}_{u,v}^k \right\|^2$$

$$\tilde{\mathbf{A}}_{i,j}^k = \max\left(\log\left(\frac{(\mathbf{A}_{i,j}/d_i)}{\sum_{l\in V}(\mathbf{A}_{l,j}/d_l)^k} \right)^k - \alpha, 0 \right)$$

node degree

constant shift

그래프 임베딩

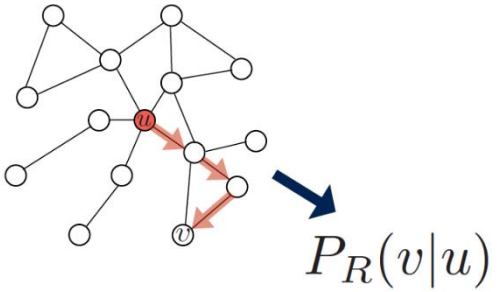
- **좋은 노드 임베딩의 예**
  - 최신 기법: Random walk approaches 활용,

    DeepWalk, KDD 2014 [LINK], node2vec, KDD 2016 [LINK]

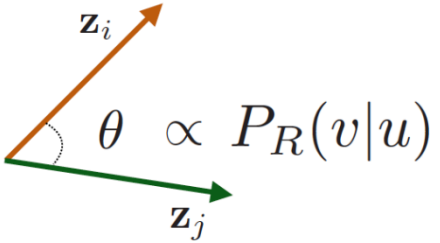
    → 벡터 간의 유사도 $\boldsymbol{z}_u^T \boldsymbol{z}_v$와 u & v가 random walk에서 함께 나올 확률 비교

    → 다양한 방식으로 random walk 전략을 세워서 해결

1. u에서 시작한 random walk R이 v를 방문할 확률 추정



2. 임베딩 후 벡터 간의 유사도와 차이 최소화



목차

- **차원 축소**
- **그래프 임베딩**
- **최신 연구 결과 소개** ⟸

1. BlackHole embedding (ICDE 2016)
2. Motif-based embedding (JSTAT 2016)
3. LinkBlackHole* using link embedding (ICDE 2014, TKDE 2019+)

Part I
BlackHole embedding

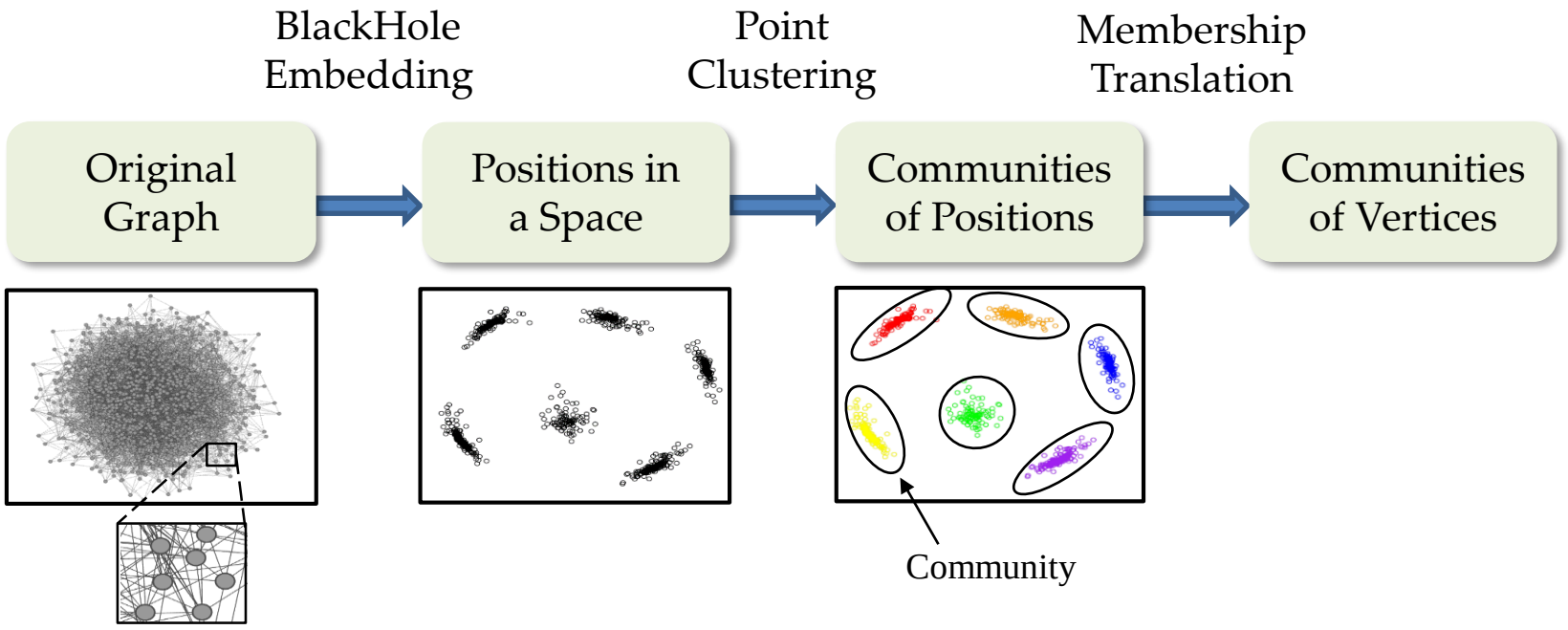
BlackHole: Robust Community Detection Inspired by Graph Drawing
IEEE ICDE 2016

Joint work with J. Kim (NTU, Singapore) and J.-G. Lee (KAIST)

[+] Follow-up research is ongoing

Proposed Algorithm: BlackHole

- Proposing the ***BlackHole embedding*** that transforms a given graph into the points in a low-dimensional space
- Developing an algorithm that performs clustering on the embedded space, which enables us to discover **highly mixed communities**

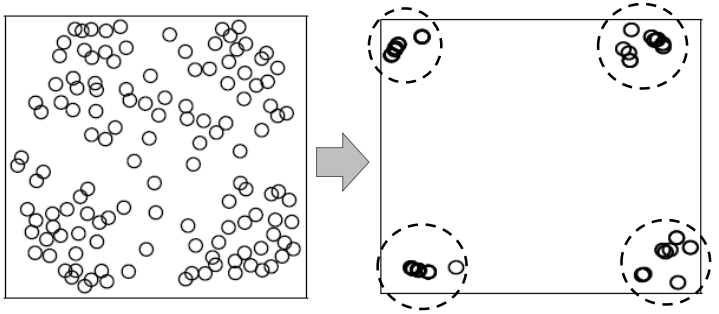


Phase I: BlackHole Embedding

- Solving an energy minimization problem through multiple iterations to determine vertex positions

$$\min_{p:V\to S} \sum_{\{i,j\}\in V^{(2)}} \left(\underbrace{\frac{w_{ij}}{a+1}\|p(i)-p(j)\|^{a+1}}_{\text{Attraction}} - \underbrace{\frac{w_i w_j}{r+1}\|p(i)-p(j)\|^{r+1}}_{\text{Repulsion}} \right)$$

  - LinLog: $a = 0, r = -1$
    - Conventional design
  - ***BlackHole***: $a = -0.95, r = -1$
    - Relatively strong repulsion
    - Exponential growth in attraction



LinLog **BlackHole**

Design of Our Repulsive Forces

- Increasing the order of repulsive forces ($r = -1$)
  - cf., Fruchterman-Reingold ($r = -1$)
    Davidson-Harel ($r = -3$)
    LinLog ($r = -1$)

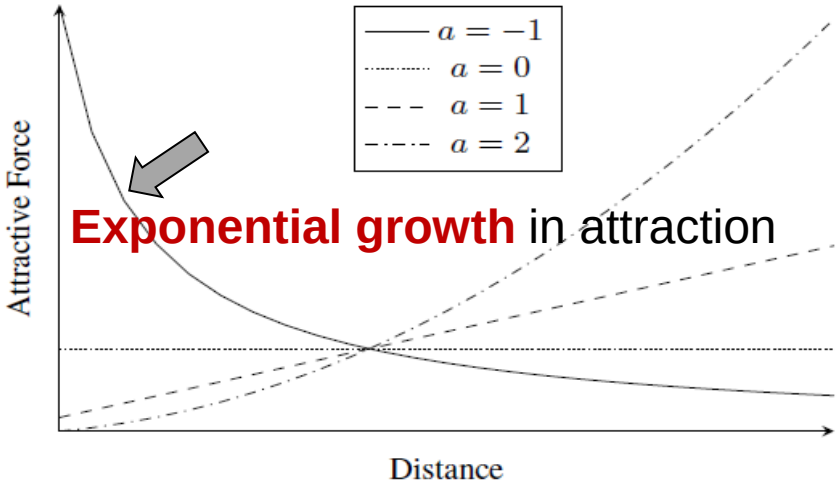
  $\Rightarrow$ Making positions more separable in early stages

How to **break** balls?
Increase the **force**!

Design of Our Attractive Forces

- Increasing the attractive force **much stronger** as the connected vertices **get closer to each other** ($a \approx -1$)
  - cf., Fruchterman-Reingold ($a = 2$)
    Davidson-Harel ($a = 1$)
    LinLog ($a = 0$)

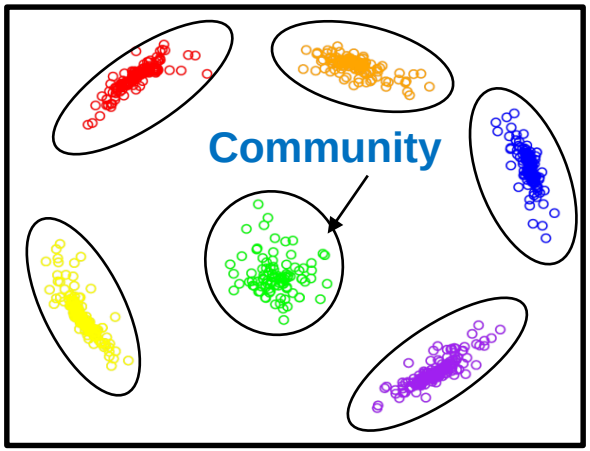
$\Rightarrow$ Attracting the nearby vertices into a ***black hole***

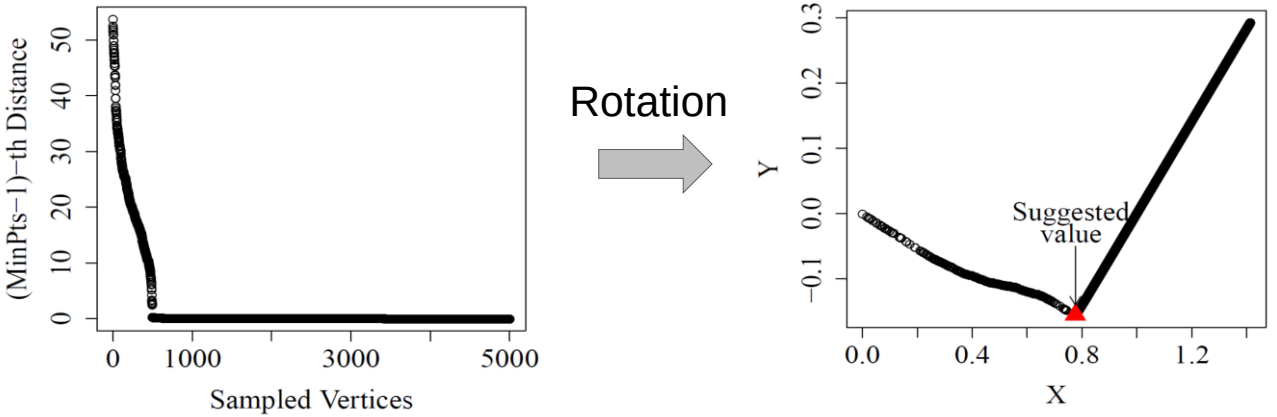


Phase II: Clustering

- Applying conventional clustering algorithms to the vertex positions obtained in Phase I

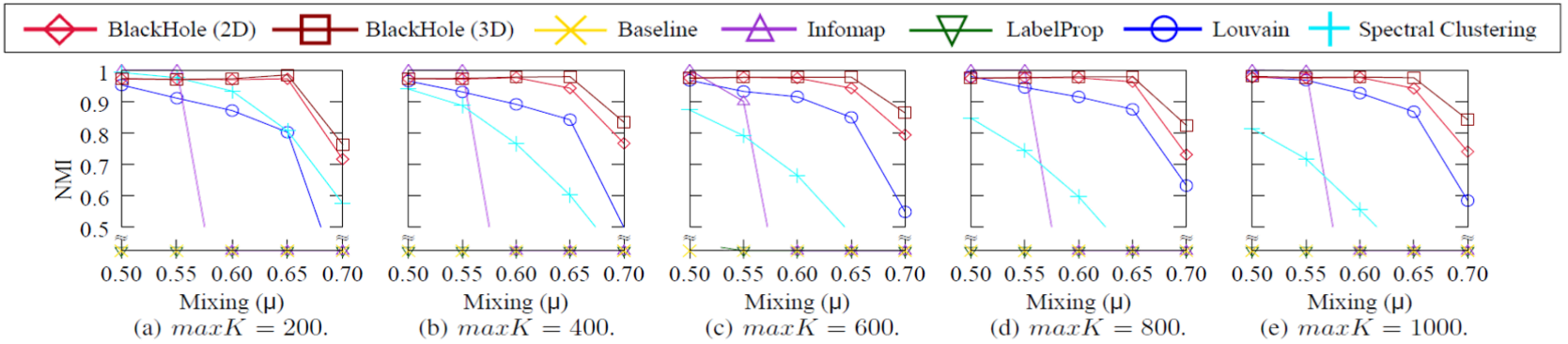
- Adopting **DBSCAN**
  - The two parameters ε and *MinPts* are determined by the heuristic



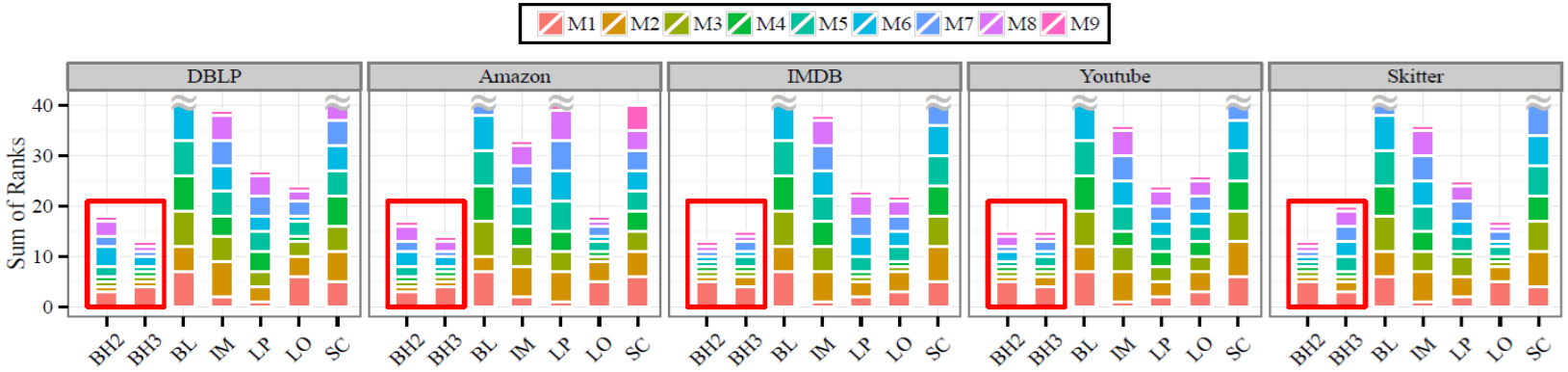


Performance Evaluation

- Providing higher quality for both synthetic and real-world networks



Effect of mixing parameter (fraction of inter-community cuts)



Cumulative ranks of M1~M9 for each algorithm

Part II
Motif-based embedding

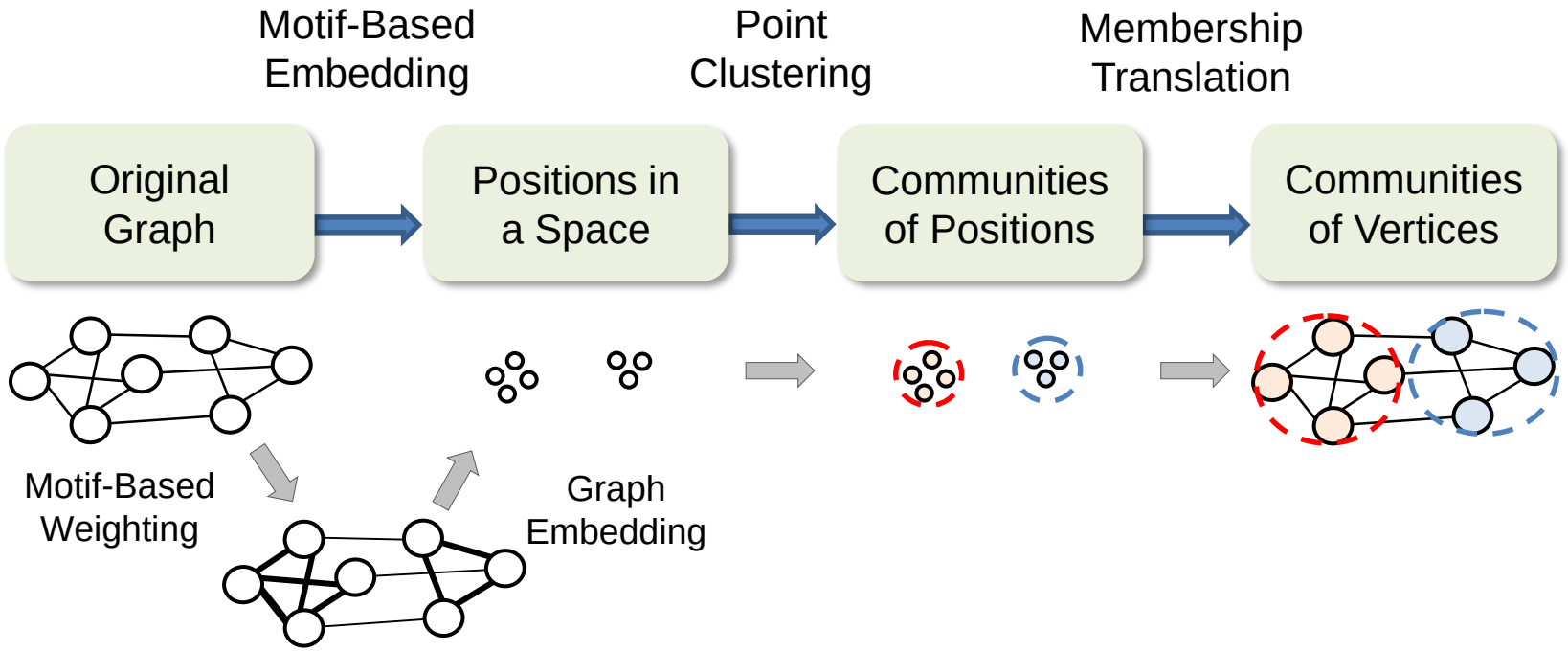
Motif-Based Embedding for Graph Clustering
Journal of Statistical Mechanics: Theory and Experiments, 2016

Joint work with J.-G. Lee (KAIST)

[+] Follow-up research is ongoing

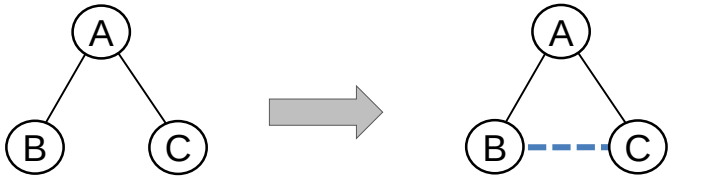
Proposed Algorithm: Motif-Based Embedding

- Proposing the ***motif-based embedding*** that transforms a given graph into points in a low-dimensional space
- Developing an algorithm that performs clustering on the embedded space, which enables us to discover **higher-order graph substructure**

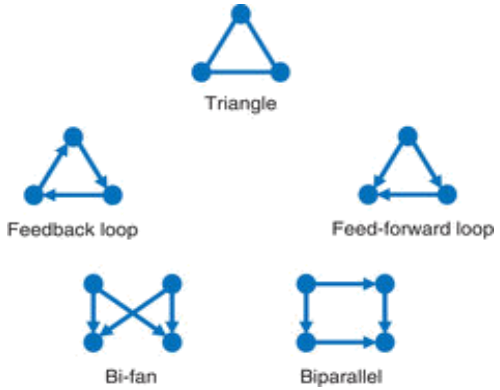


Network Motifs

- Higher-order graph substructures
  - The relationships involve **multiple vertices** within clusters, *e.g.*, triangles in social networks
  - By incorporating the motifs, the ***motif-based weighting*** method reflects motif substructures in a given graph



Triadic closure
(a concept from sociology)



Motifs in biological networks
[Tran et al. 2013]

Motif-Based Embedding

- Solving an energy minimization problem through multiple iterations to determine vertex positions

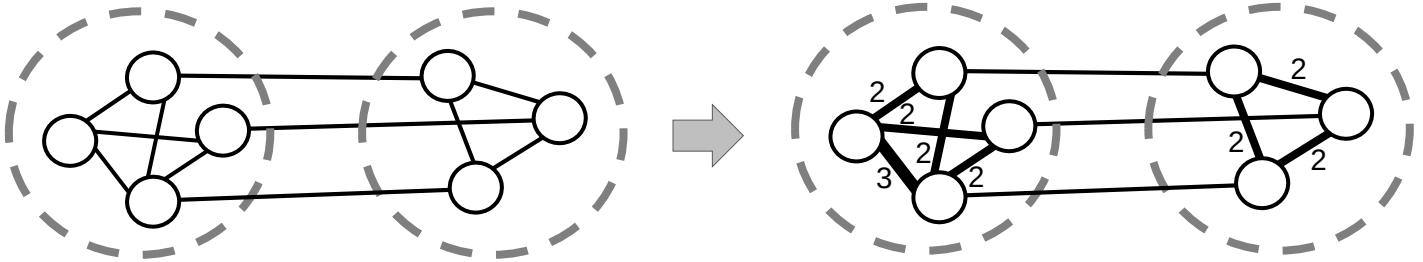
$$\min_{p:V\to S} \sum_{\{i,j\}\in V^{(2)}} \underbrace{\frac{\boldsymbol{f(i,j)}}{a+1}\|p(i)-p(j)\|^{a+1}}_{\textbf{Attraction}} \underbrace{-\frac{w_i w_j}{r+1}\|p(i)-p(j)\|^{r+1}}_{\textbf{Repulsion}}$$

  - Force-directed embedding: $f(i,j) = w_{ij}$
    - Conventional design
  - ***Motif-Based Embedding***: $f(i,j) = w_{ij} + m_{ij}$
    - Weights calculated from edges and motifs
    - Strong attraction within motifs

Motif-Based Weighting

- Weighting every pair of vertices by $f: V \times V \to \mathbb{R}$, with **strong attraction** within the motifs of interest
  - **Example:**
    $f: \{i, j\} \mapsto w_{ij} + m_{ij}$
    w_{ij}: the existence of an edge $\{i, j\}$
    m_{ij}: # of motifs containing i & j together

Many motifs (triangles) within a cluster but not across clusters



Mixing: 0.43 Mixing: 0.26

- The lower the mixing (fraction of sum of inter-community weights) is, the more detectable the community structure is

Performance Evaluation: Synthetic Networks

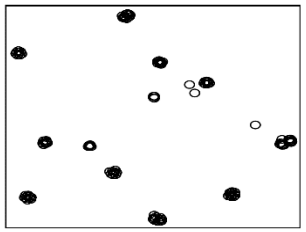
- **Embedding methods**
  - (i) Motif-based embedding (2-dim. space) ⇐ ***Proposed method***
  - (ii) Force-directed embedding (2-dim. space)
  - (iii) Spectral embedding (k-dim. space)
  - (iv) Spectral embedding with motif-based weights (k-dim. space)
- **Used motifs**
  - Triangles for all synthetic networks
- **NMI results** (clustering: embedding + k-means)

	Proposed	Method (ii)	Method (iii)	Method (iv)
Mixing = 0.4	**0.99**	0.89	0.70	0.73
Mixing = 0.5	**0.84**	0.79	0.37	0.41
Mixing = 0.6	**0.58**	0.47	0.13	0.26

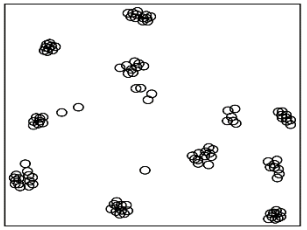
Performance Evaluation: Real Networks

- **Used motifs**
  - Social graphs (Football network): triangles
  - Bipartite graphs (Malaria network): wedges
- **Accuracy**

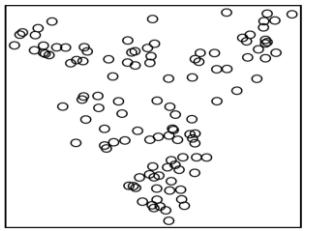
	Proposed	Method (ii)	Method (iii)	Method (iv)
Social graphs	**0.93**	0.91	0.75	0.89
Bipartite graphs	**1**	0.52	0.01	**1**



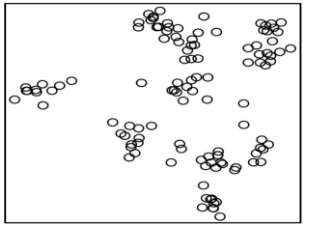
(i) Proposed embedding



(ii) Edge-based embedding



(iii) Spectral embedding



(iv) Spectral embedding with motif-based weights

Embedding results of each algorithm for the football network

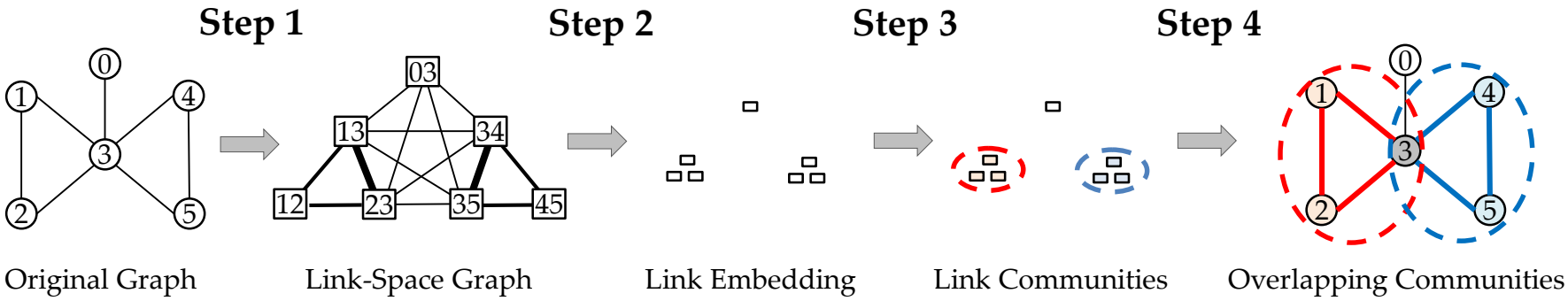
Part III
Link embedding

LinkBlackHole: Robust Overlapping Community Detection Using Link Embedding*
IEEE TKDE 2019 (Prior work: IEEE ICDE 2014 & 2016)

Joint work with J. Kim (ETRI), B.S. Lee (U. Vermont), and J.-G. Lee (KAIST)
(Prior work: collaborated with S. Ryu (ADD), S. Kwon (Samsung), and K. Jung (SNU))

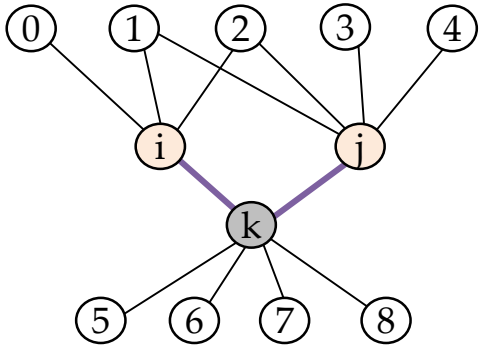
Proposed Algorithm: LinkBlackHole*

- Step 1: **Link-space transformation** on the original graph
- Step 2: **BlackHole transformation** on the link-space graph
- Step 3: Clustering for the vertex positions of the link-space graph
- Step 4: Membership translation procedure
  - 4-1: Disjoint communities of the vertex positions of the link-space graph
  - 4-2: Disjoint communities of the links of the original graph
  - 4-3: Overlapping communities of the vertices of the original graph

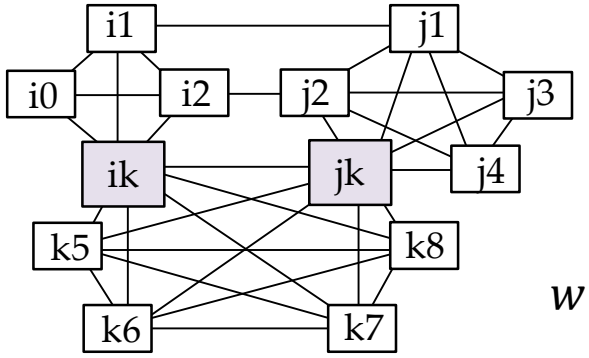


Phase I: Link-Space Transformation

- **Topological structure**
  - Link (original graph) → node (link-space graph)
  - Two incident links (original graph) → link (link-space graph)
- **Weights**
  - Similarity between links (original graph) → weight of a link (link-space graph)



Original graph

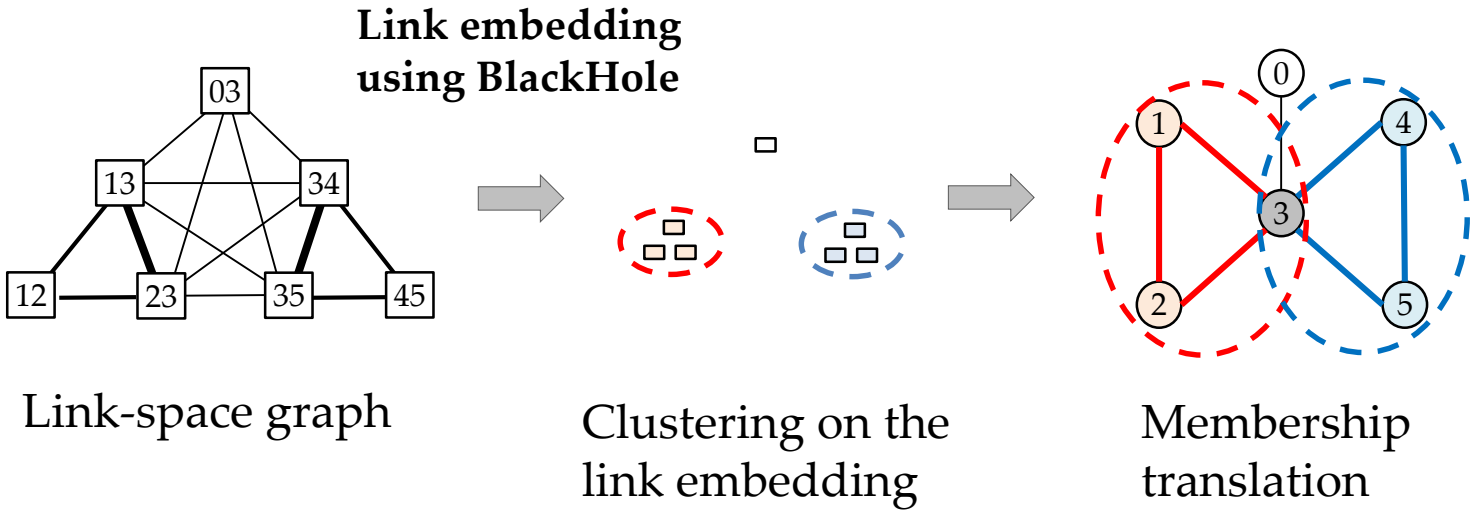


Link-space graph

$$w(v_{ik}, v_{jk}) = \sigma(e_{ik}, e_{jk})$$

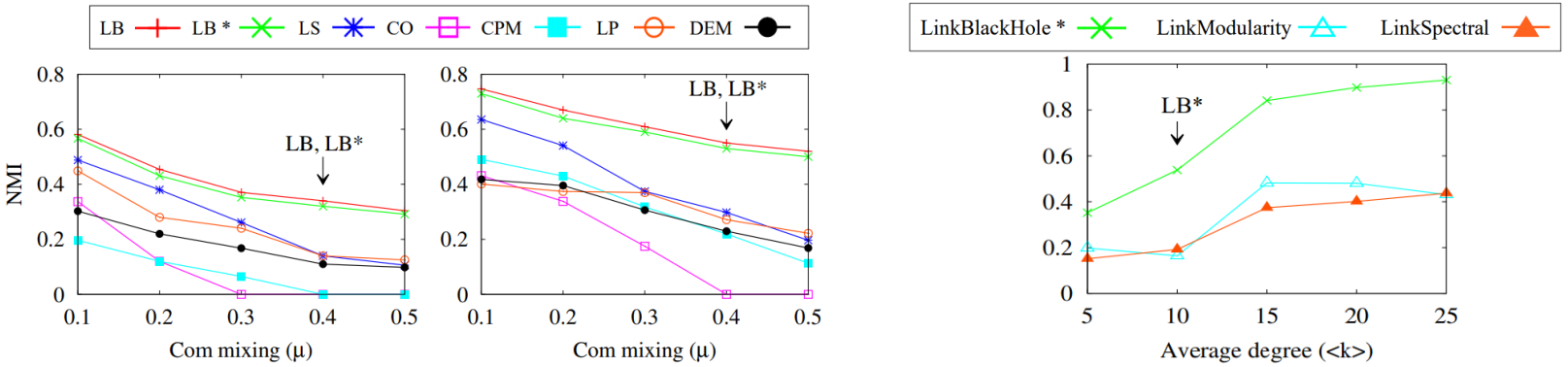
Phase II: BlackHole Embedding + Clustering

- **Link Embedding**: Applying a BlackHole embedding to the link-space graph
- Using structural clustering (SCAN) that can assign a node into hubs or outliers (neutral membership)

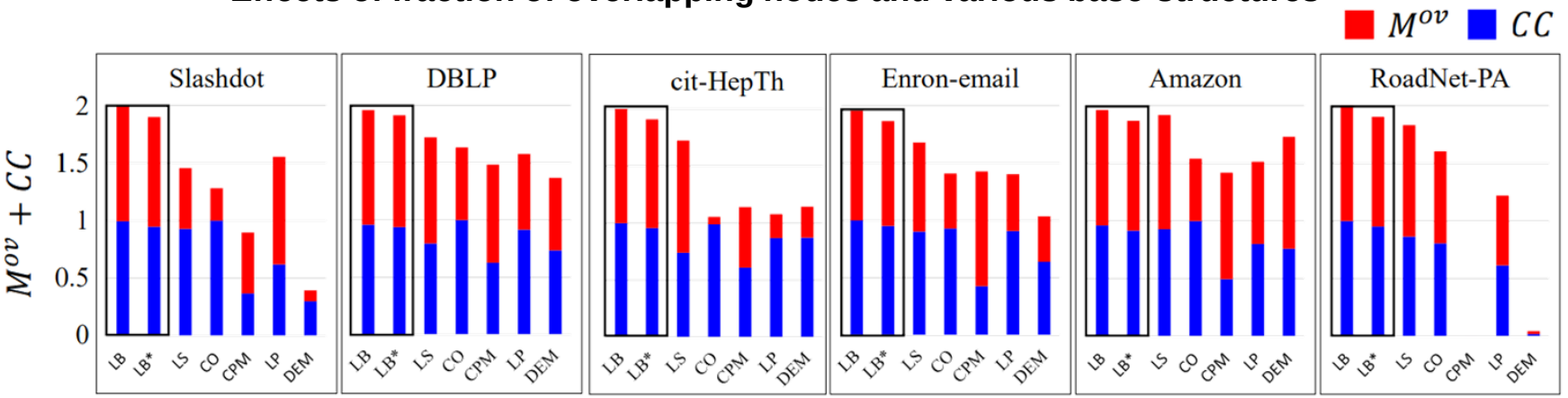


Performance Evaluation

- Providing higher quality for both synthetic and real-world networks



Effects of fraction of overlapping nodes and various base-structures



Normalized measure of (Quality + Coverage) for each algorithm

맺음말

- **Research Goal**
  - ***Understanding complex systems with Big Data***
  - Developing algorithms for solving intelligence and real-world data problems

- **Research Areas and Tools**
  - **Big Data Analysis**
  - **Network Science**
  - Artificial Intelligence & Machine Learning
  - Areas that deal with math & comp. sciences

감사합니다

Thank You Very Much!

Any Questions?