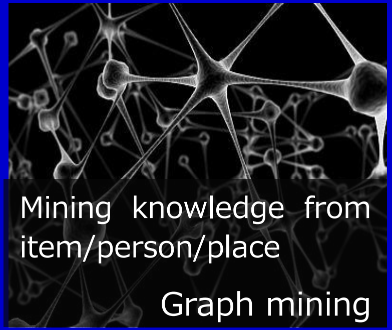
Techniques

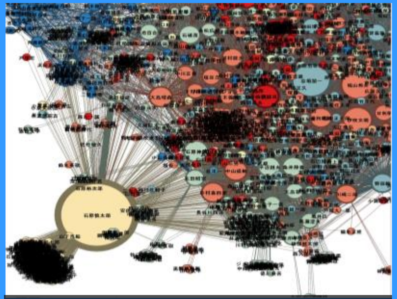
Real time big data processing

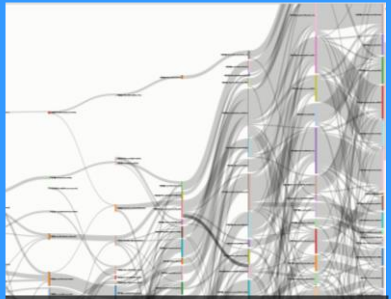
Parallel/distributed

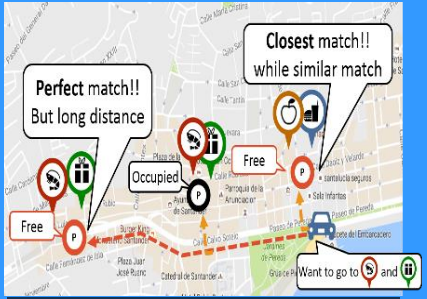

Mining knowledge from item/person/place

Graph mining

Applications


Social graph analysis


Trend analysis



Route recommendation

Analytical query optimization: distributed processing, graph mining, and applications

Osaka University

Prof. Makoto Onizuka (oni@acm.org)

Onizuka-Lab.

BTW

- At SIGMOD2022
  - Edgar F. Codd Innovations Award　Dan Suciu
- I was supervised by Dan during 2000-2001. What I learned:
  - Theory to practice: automaton to XML streaming processing
  - Practice to theory: engineering techniques are required to make the theory work

My main research projects

1. Query optimization for analytical queries
   - Application: Intrinsic variable discovery
   - Techniques: join query optimization, materialized view selection, outlier detection
   - Partner: National Astronomical Observatory of Japan, Toshiba, Treasure data
2. Graph database（graph mining and graph query）
   - Tasks: clustering, classification, link prediction, subgraph matching
   - Techniques: Graph neural networks（GCN, ANEPN）
   - Partner: AI samurai（patent search, patentability evaluation）
3. Data integration
   - Techniques: bidirectional transformation, distributed transactions
   - Applications: Ride-sharing alliance, clinical data integration

1. Query optimization for analytical queries

OPTIMIZATION FOR ITERATIVE QUERIES ON MAPREDUCE

Makoto Onizuka, Hiroyuki Kato, Soichiro Hidaka, Keisuke Nakano, Zhenjiang Hu

VLDB 2014

Overview

- OptIQ is an optimization framework for iterative queries
  - Declarative high level language: extended SQL with iterations for optimization
- Two techniques for removing inefficiency
  - view materialization for invariant views
  - incrementalization for variant views
- We implement on MapReduce and Spark

Running example: PageRank

This program is not efficient. Which parts?

```
class Mapper
  method Map(nid n; node N)
    p ← N.PageRank/|N.AdjacencyList|
    Emit(nid n, N) // Pass along graph structure
    for all nodeid m ∈ N.AdjacencyList do
      Emit(nid m, p) // Pass PageRank mass to neighbors

class Reducer
  method Reduce(nid m, [p1,p2,...])
    M ← $\phi$
    for all p ∈ counts [p1,p2,...
      if IsNode(p) then
        M ← p // Recover graph str
      else
        s ← s + p // Sum incoming PageRank contributions
    M.PageRank ← s
    Emit(nid m, node M)
```

map function shuffles whole graph structure in every iteration

scores are computed even if the nodes are converged

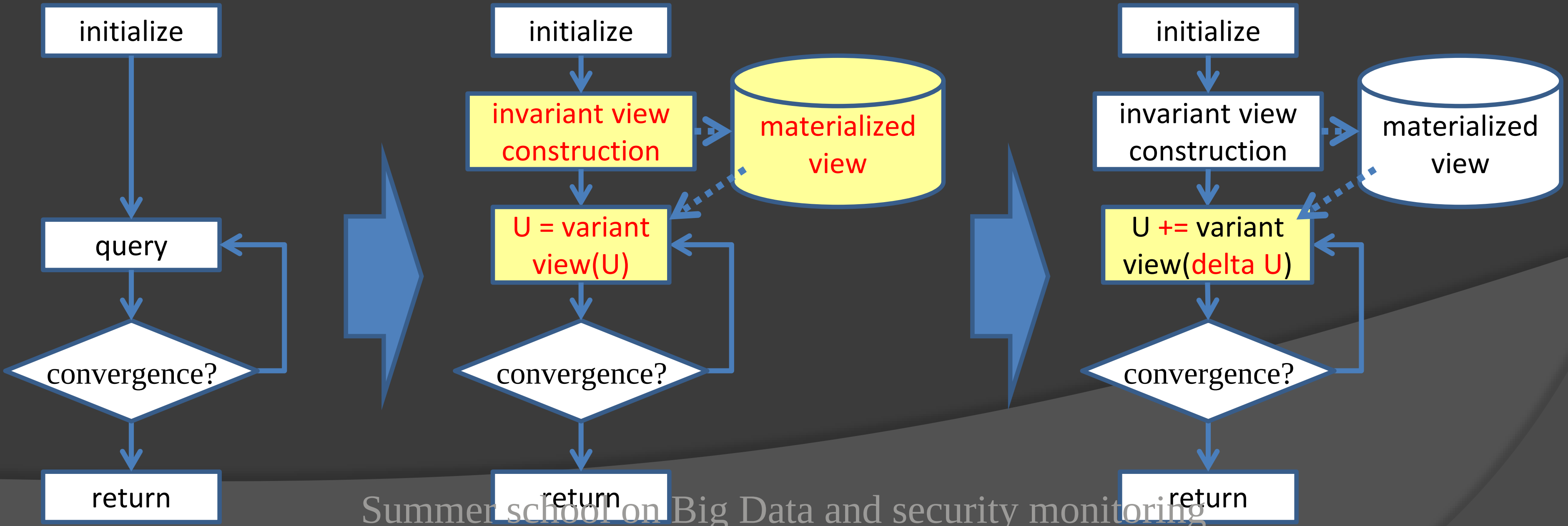
Issues for iterative analysis

- Can we optimize the program?
- Possible but difficult to manually remove the above redundant computations
  - Actually, Spark, HaLoop, REX force programmers to manually remove them

- Our goal: Automatically remove redundant computations for iterative queries

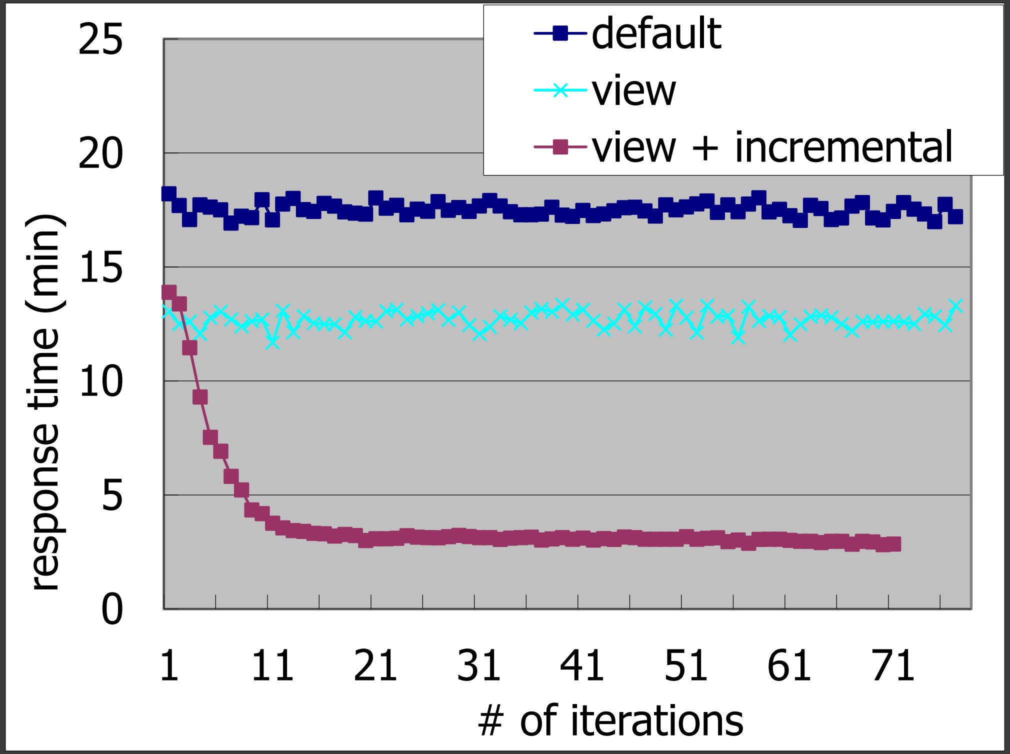
Ideas for removing inefficiency

- View materialization
  1. Decompose tables/views into variant/invariant tables/views
  2. Materialize invariant views
- Incremental evaluation
  1. Evaluate incrementally variant views

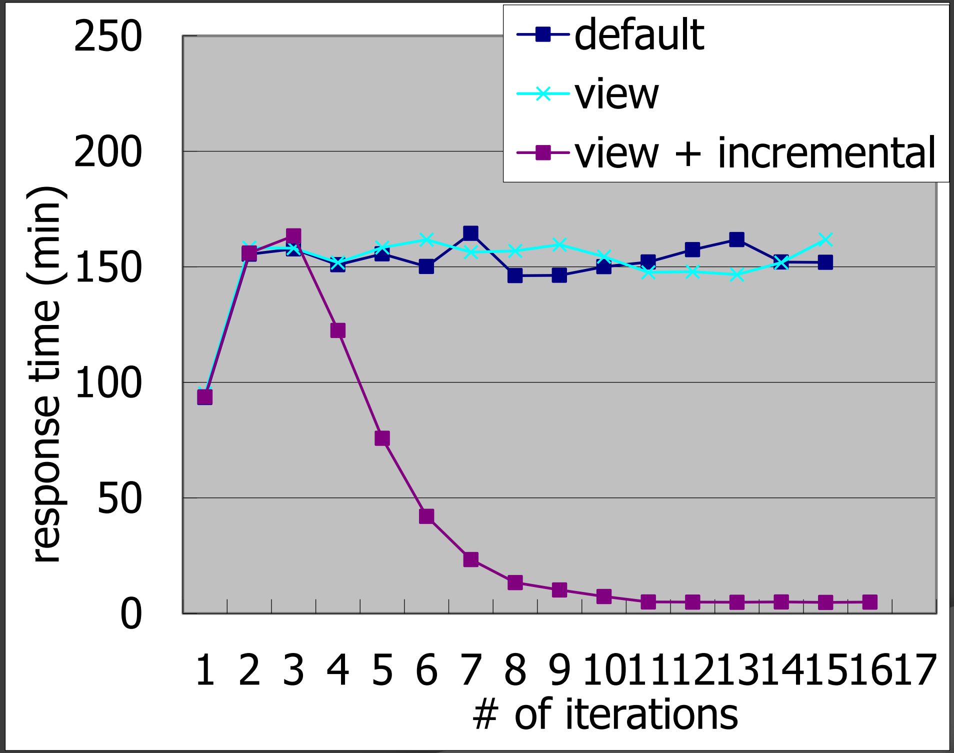


Experiments

- **Up to five times faster for PageRank/k-means both in MapReduce/Spark**



PageRank Computation/webbase-2001



K-means clustering/mnist8m

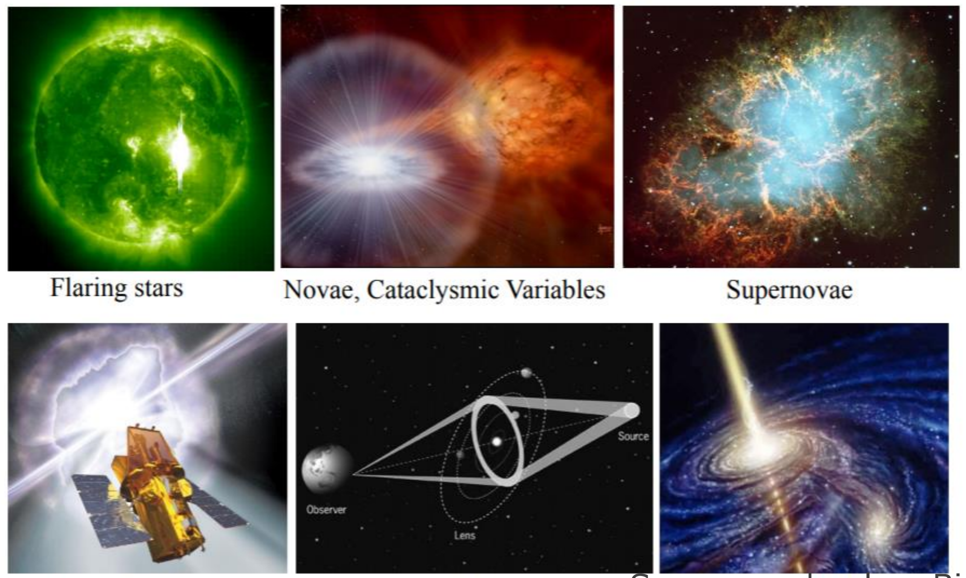
Exploratory Data Analysis

- Technique for discovering interesting data that largely differ from ordinary/average data
  - Join work with National Astronomical Observatory of Japan (NAOJ)

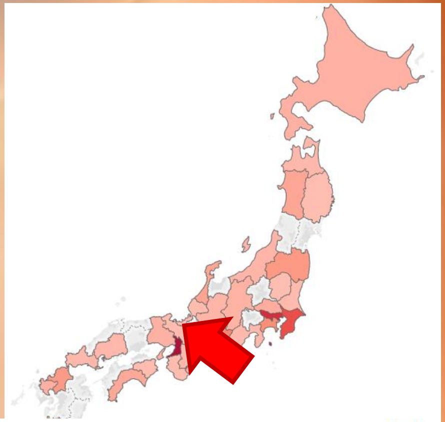
Intrinsic variable discovery

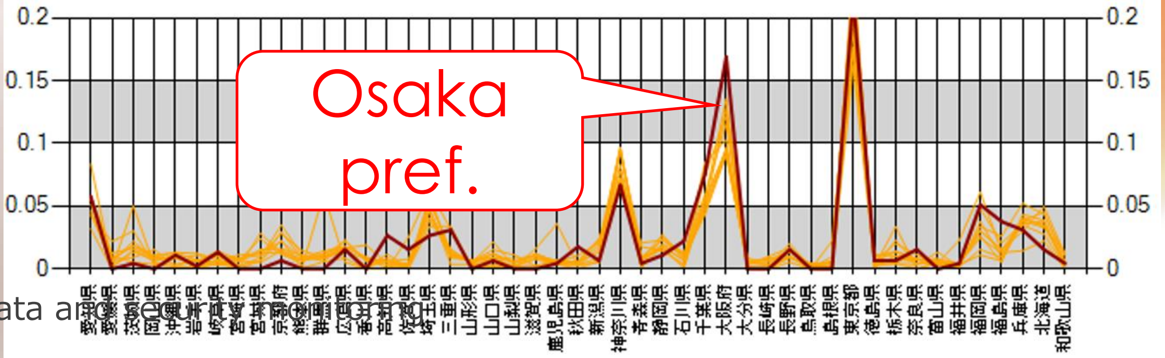
Outlier detection

Time Domain: A Broad Variety of Phenomena



Osaka was ranked at top in the emission of garbage per person in 2014

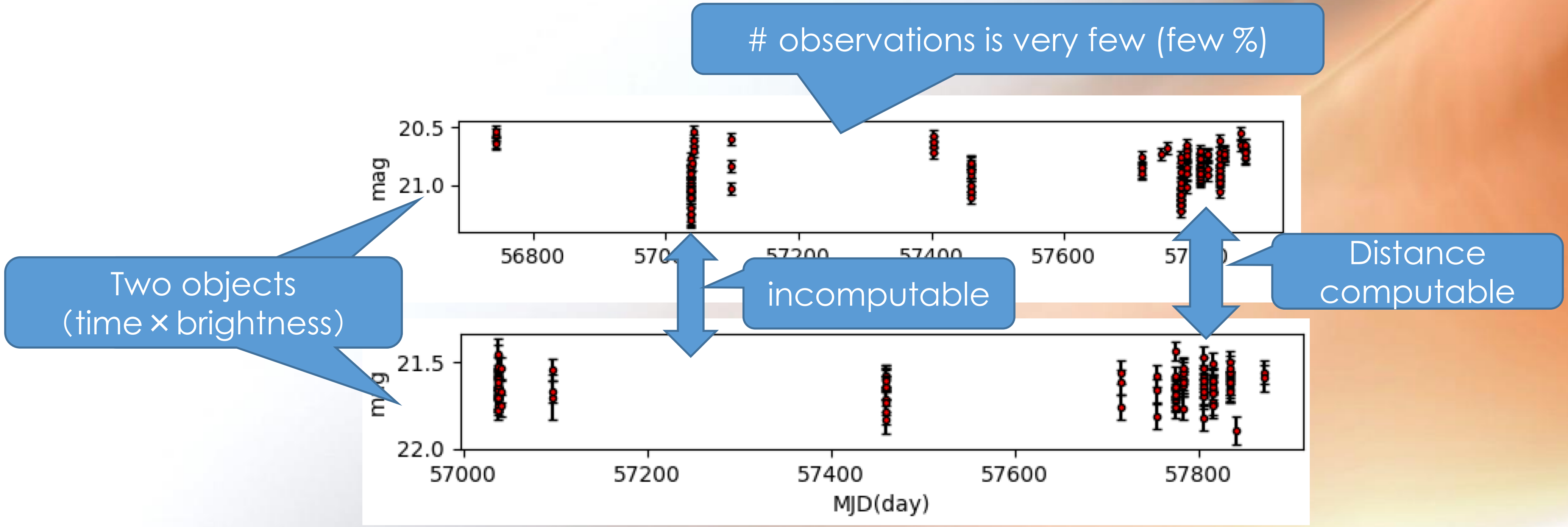




https://www.astro.caltech.edu/~george/ay111/Djorgovski_Ay111_Jan12.pdf

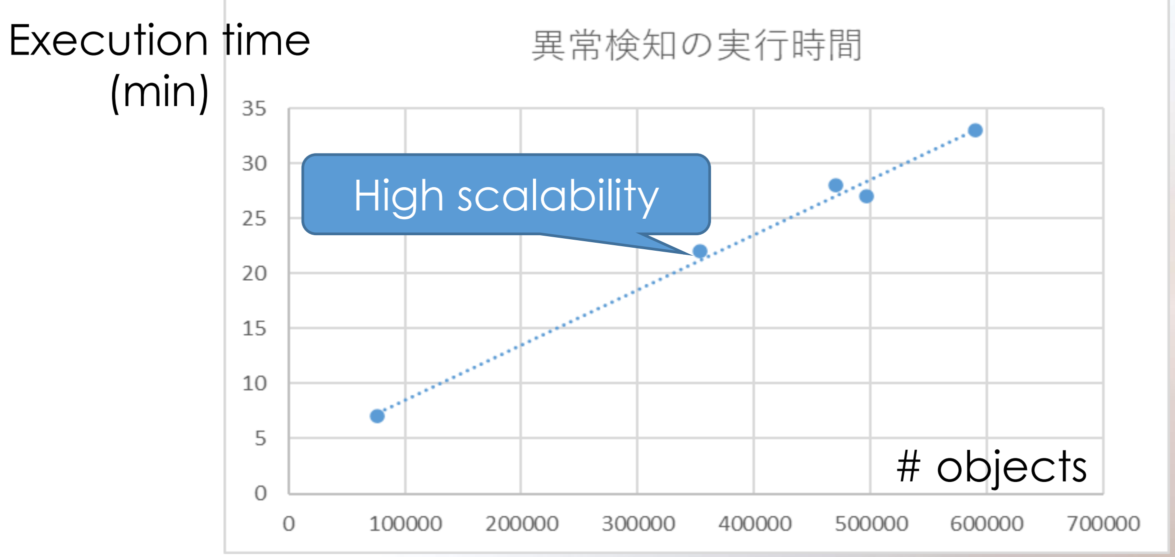
Application: Intrinsic variable discovery

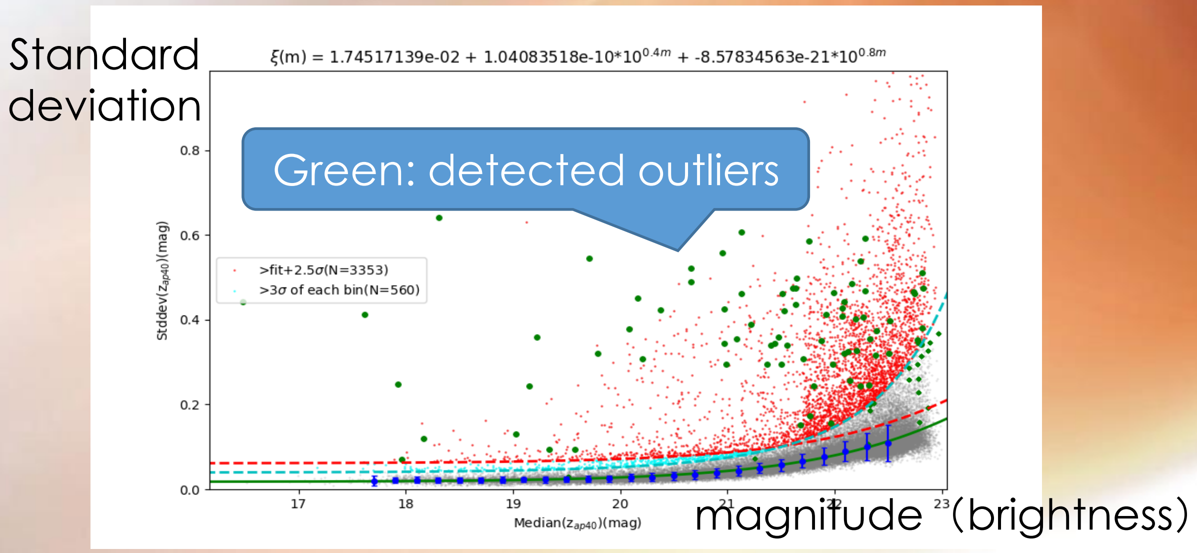
- **Problem:** outlier detection from very sparse time-series data
- **Approach:** cluster data and make imputation for each cluster



Application: Intrinsic variable discovery

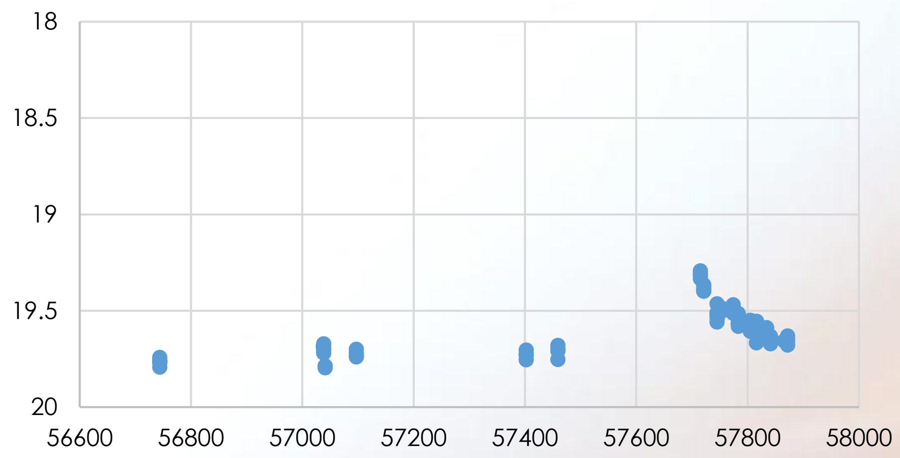
- **Implementation: Spark + PySpark**
- **Achievement**
  - **Response time: high scalability（40mins for 240M records)**
  - **Analysis quality:（under evaluation)**

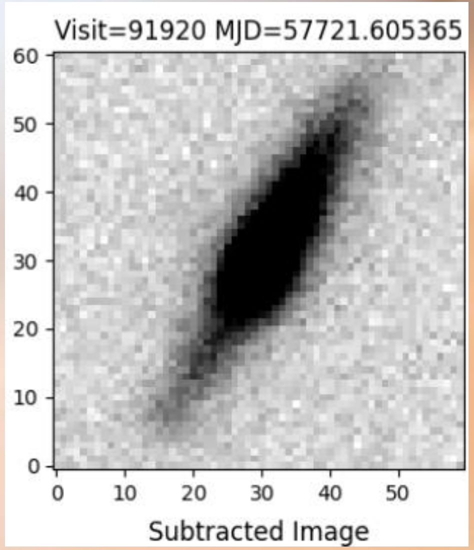




Application: Intrinsic variable discovery

- **Identified examples: supernova**

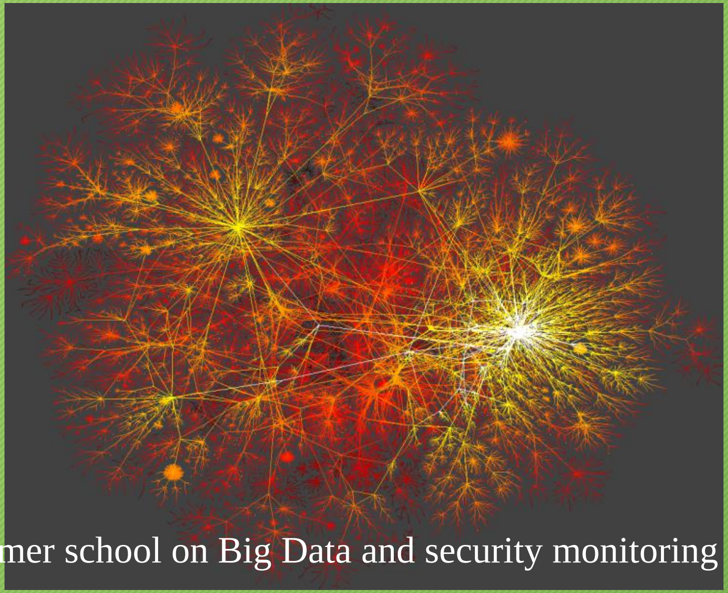




2. Graph mining

Big graphs everywhere

- Web graph: 10B pages in the world
- Social graph: 3B users in Facebook
- User-item graph: 0.1B in amazon.com



The architecture of complexity, ASIS Keynote 2006

https://www.facebook.com/notes/facebook-engineering/visualizing-friendships/469716398919

Our target: graph mining

- Large-scale graphs have emerged
  - Web graph: 10B pages in the world
  - Social graph: 3B users in Facebook
  - User-item graph: 0.1B in amazon.com
- Expensive cost of graph mining
  - Clustering: $O(N^2)$, N is node size
  - Random walk: $O(mt)$, m is edge size, t is iterations

Effective techniques are demanded

Our contributions

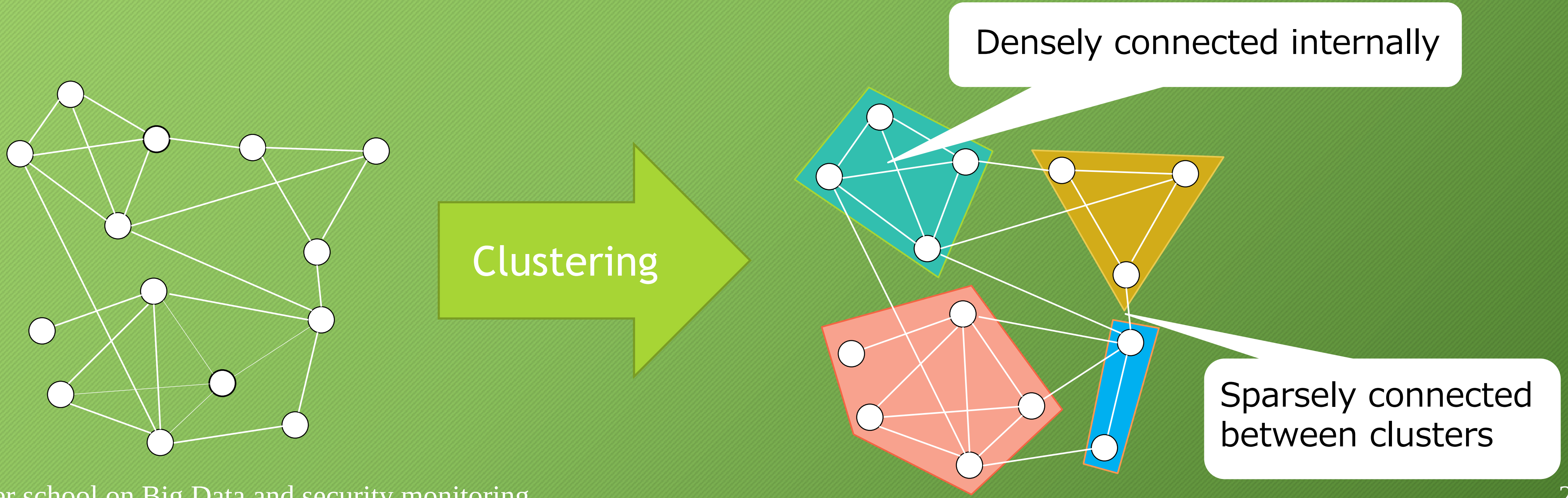
- Graph clustering/Graph classification/Graph query
  - Modularity [AAAI13], SCAN [VLDB15], PPNMF [GEM19]
  - ANEPN [ECML21], LC transformation [ECML22]
  - Subgraph matching [ICDE22]
- Distributed/parallel query processing
  - Distributed query optimization [VLDB14]
  - Graph ordering [IPDPS16], Graph partition [DSE17]

What are typical tasks for graph mining?

- Typical graph mining tasks
  - Clustering
  - Classification
  - Link prediction

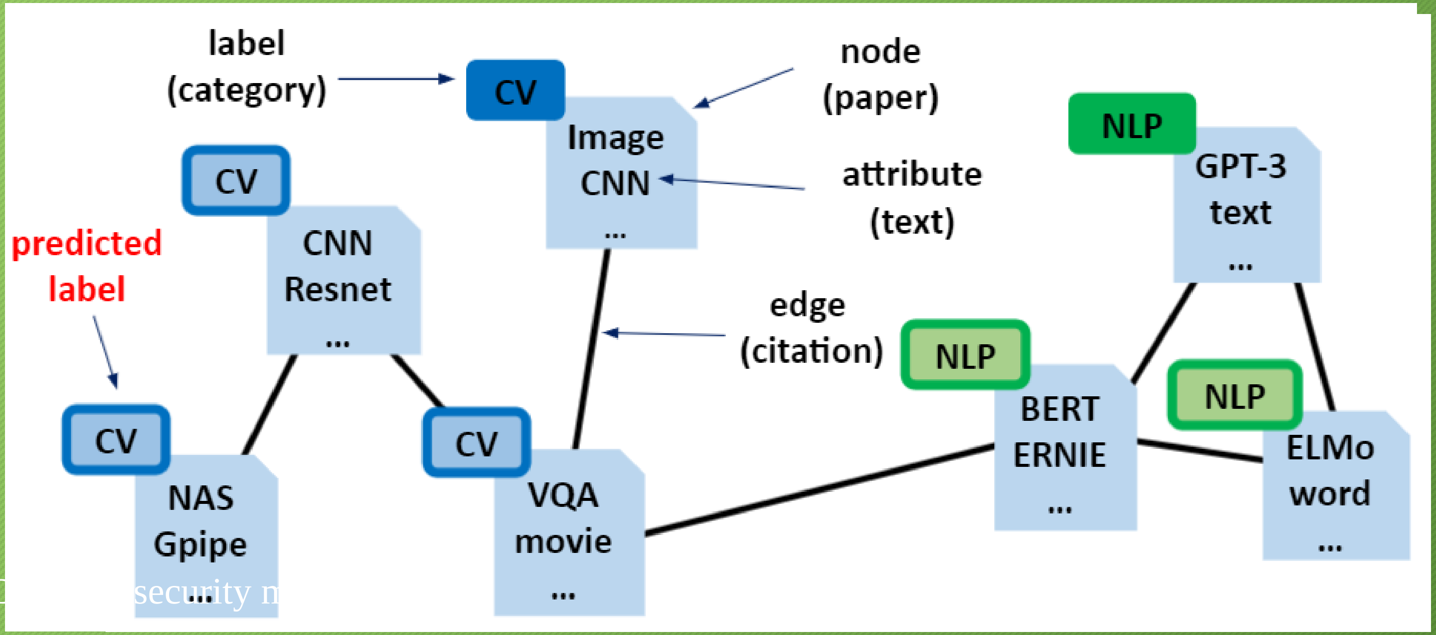
Graph clustering

- Identify communities based on graph structure and attributes
- Idea: Many edges in same clusters/sparse between different clusters


Clustering
Densely connected internally
Sparsely connected between clusters

Node classification

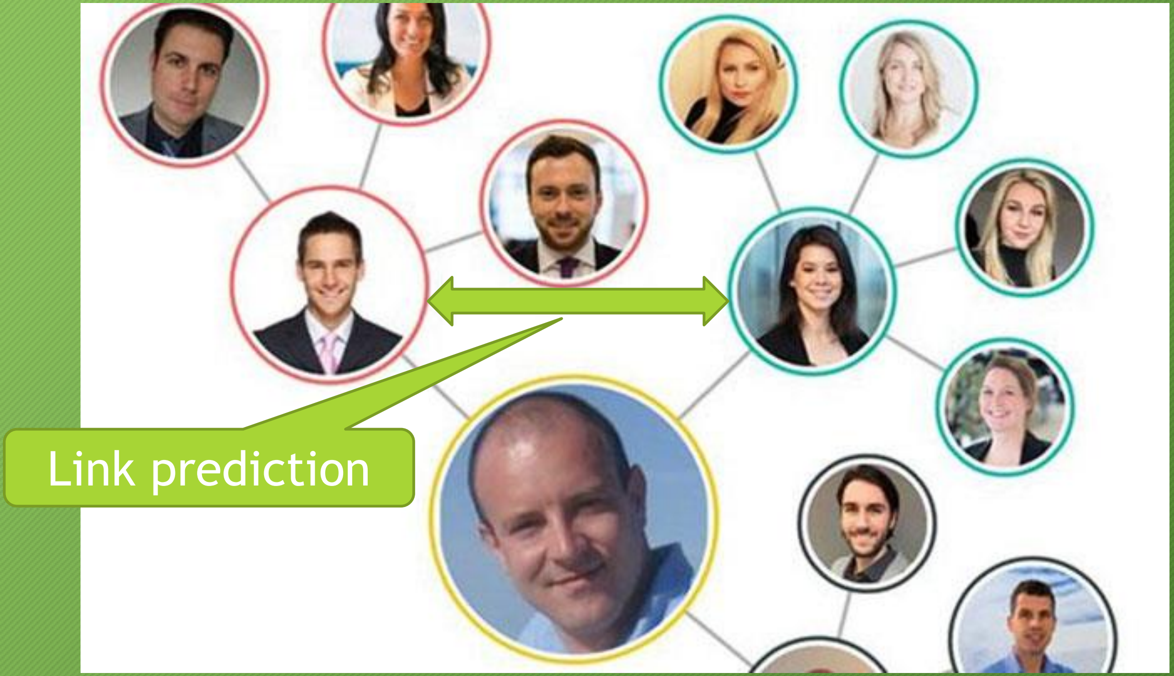
- Predict label of nodes based on given labels of other nodes
- Idea: Not only using node attributes, we leverage structure: node feature is affected by its neighbor nodes.



Link prediction

https://lab.pasona.co.jp/data-operation/skill/788/

- Predict future link between nodes
- Applications
  - Friend recommendation in SNS
  - Protein-protein interaction
  - Item recommendation

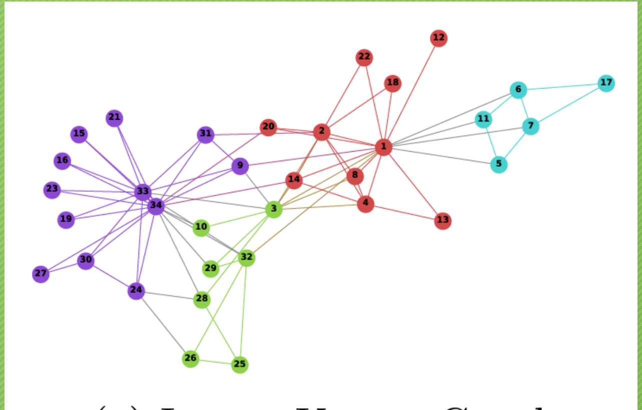


Techniques for graph mining

- Representation learning/node embedding
- Graph neural networks (GNN)

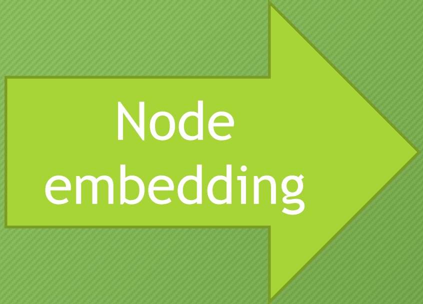
Representation learning/node embedding

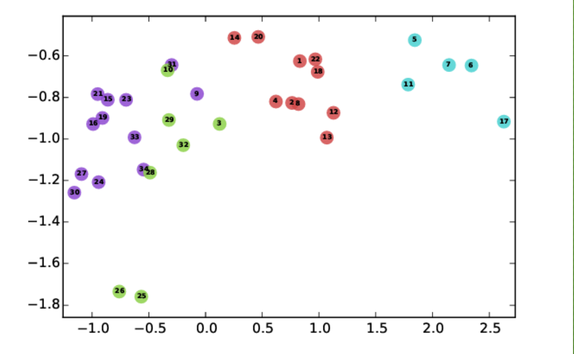
- Node embedding from graph space to multi-dimension space
  - Obtain node feature using structure and/or attributes
  - Benefit: we can utilize standard ML techniques
  - Note: adjacent nodes should be embedded into close in feature space (DeepWalk example)



(a) Input: Karate Graph

Graph space



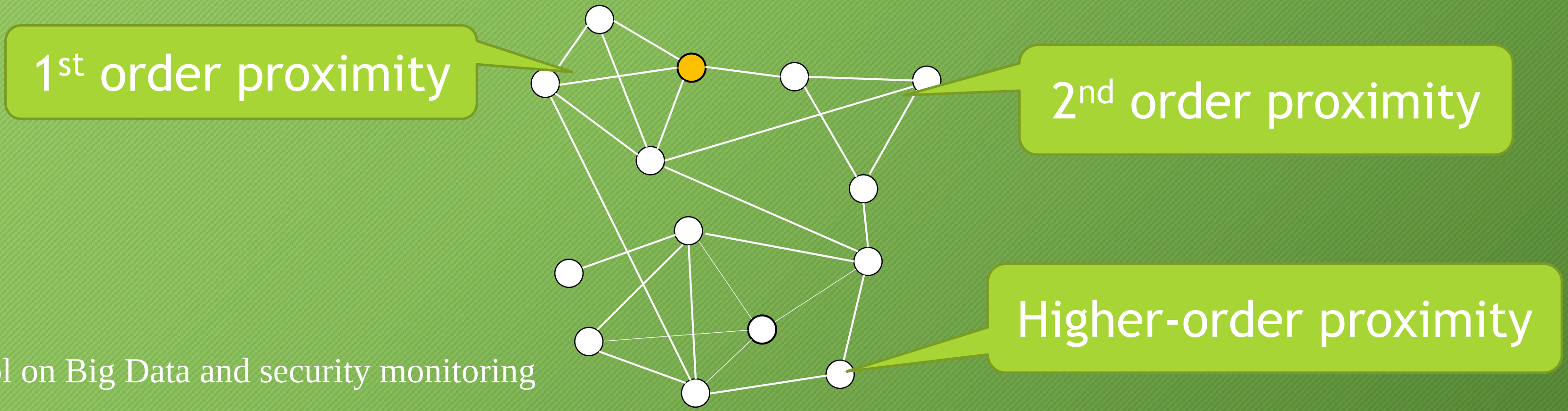


(b) Output: Representation

multi-dimension space

Microscopic and macroscopic aspects

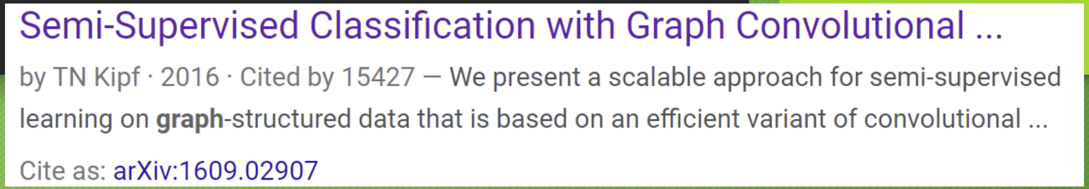
- We should take microscopic and macroscopic aspects in node embedding
  - Microscopic (local): 1st order/2nd order proximity (friends and friends of friends) are useful for effective embedding
  - Macroscopic (global): higher-order proximity is also useful, in particular when labeled nodes are few



Microscopic and macroscopic aspects

- We should take microscopic and macroscopic aspects in node embedding
  - Microscopic (local): 1st order/2nd order proximity (friends and friends of friends) are useful for effective embedding
  - Macroscopic (global): higher-order proximity is also useful, in particular when labelled nodes are few.
- Technical trends
  - 1st order proximity: Spectral clustering (NIPS2001)
  - 2nd order proximity: SCAN clustering (KDD2007)
  - 1st +2nd order proximities: SDNE (KDD2016), GCN (ICLR2017), SEAL
  - microscopic + mesoscopic: M-NMF (regularized with modularity)
  - microscopic + macroscopic: node2vec, ALaGCN, ANEPN

GCN (Graph Convolutional Networks), ICLR2017



Semi-Supervised Classification with Graph Convolutional ...

by TN Kipf · 2016 · Cited by 15427 — We present a scalable approach for semi-supervised learning on **graph**-structured data that is based on an efficient variant of convolutional ...

Cite as: arXiv:1609.02907

- GCN is designed for graph classification
  - Loss: classification loss + 1st order proximity loss （f is a neural projection)

Adjacency matrix

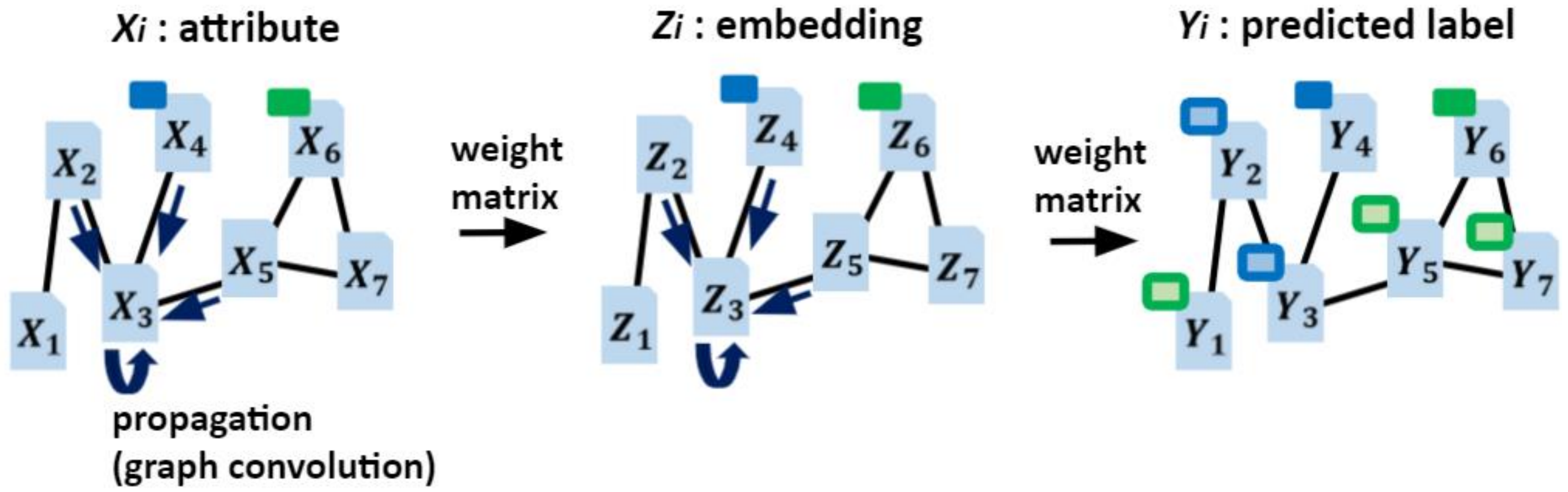
1st order proximity loss on node i and j

$$\mathcal{L} = \mathcal{L}_0 + \lambda\mathcal{L}_{\text{reg}}\,, \quad \text{with} \quad \mathcal{L}_{\text{reg}} = \sum_{i,j} A_{ij}\|f(X_i) - f(X_j)\|^2 = f(X)^\top \Delta f(X)\,.$$

  - Design: learn graph neural network (f) to minimize classification loss $\mathcal{L}_0$
    - $\mathcal{L}_{\text{reg}}$ is implemented as graph convolution operation f, which update node feature by aggregating features its neighbors （repeating k-layer）.
    - 2-layer GCN performs best in general

GCN: Graph Convolutional Networks [1]

- GCNs typically are used as two-layer neural networks.
- They utilize graph structure within two-hops by propagating node attributes and embeddings.



X_i : attribute
Z_i : embedding
Y_i : predicted label
X_1 X_2 X_3 X_4 X_5 X_6 X_7
weight matrix
Z_1 Z_2 Z_3 Z_4 Z_5 Z_6 Z_7
weight matrix
Y_1 Y_2 Y_3 Y_4 Y_5 Y_6 Y_7
propagation (graph convolution)

[1] Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: ICLR (2017)

Adaptive Node Embedding Propagation for Semi-Supervised Classification (ECML/PKDD 2021)

Yuya Ogawa*, Seiji Maekawa*, Yuya Sasaki*,
Yasuhiro Fujiwara**, Makoto Onizuka*

* Osaka University
**NTT Communication Science Laboratories

ANEPN, ECML/PKDD2021

- GCN does not work well for semi-supervised learning setting
  - 2-layer GCN does not propagate information enough to all nodes.
  - Many-layer GCN suffers from overfitting and over-smoothing
- Key observation:
  - Layer size is tightly coupled with # convolutions and model's expressive power
- Our Idea: separate #convolutions from layer size
  - We recover 1st order proximity to the loss function and repeat propagation many times using 2-layer GCN
  - We introduce anti-proximity loss to keep distant nodes to have different embedding features
  - We choose an appropriate number of propagations based on cluster coefficient

Architecture and loss of ANEPN

ANEPN uses two-layer neural network

$$Z = \bar{X}W^{(0)} + B^{(0)}$$

$$Y = softmax(ZW^{(1)} + B^{(1)})$$

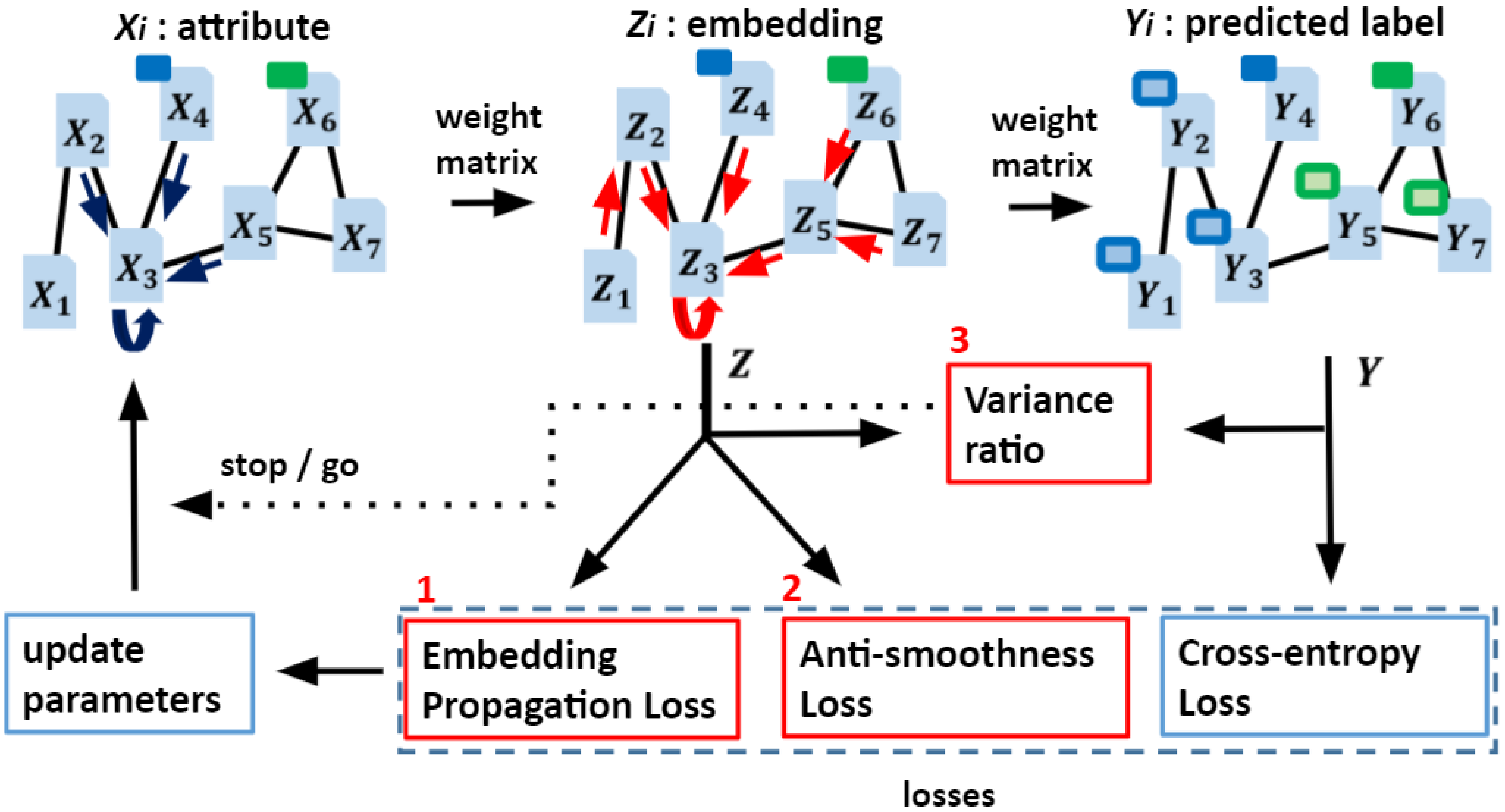
Z : node embedding
X : preprocessed attributes
W : weight matrix
B : bias matrix
Y : predicted labels

Its loss consists of three losses (Embedding Propagation Loss L_{ep} , Anti-Smoothness Loss L_{asm}, Cross Entropy Loss L_{ce})

$$L = \alpha \overbrace{L_{ep}(Z)}^{\text{1st order proximity loss}} + \alpha \overbrace{L_{asm}(Z)}^{\text{Anti-proximity loss (distant nodes)}} + \overbrace{L_{ce}(Y)}^{\text{Classification loss}}$$

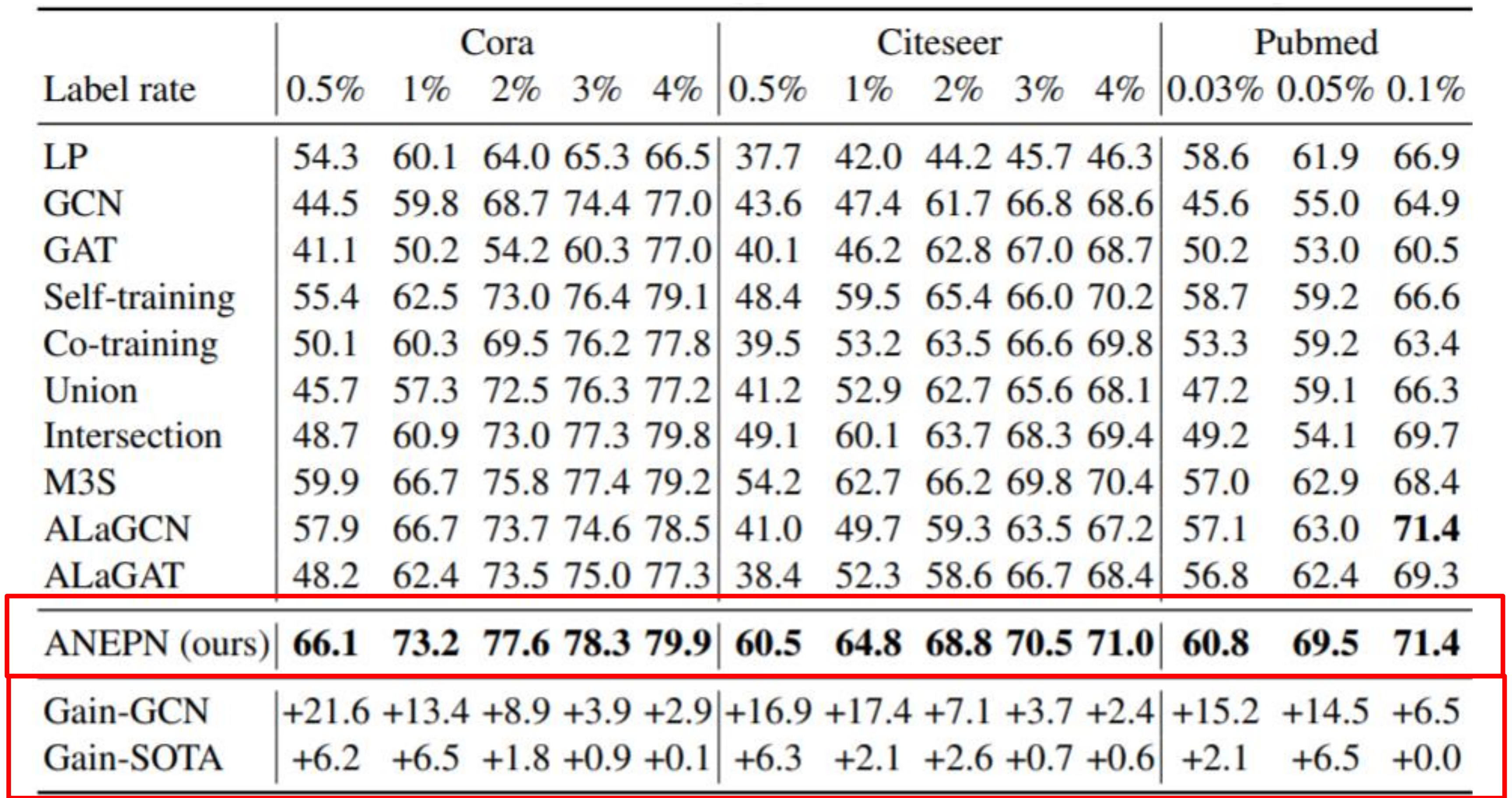
***α* : coefficient**

Training of ANEPN



Results of classification accuracy

- ANEPN outperforms existing approaches.
- ANEPN achieves larger performance gains under low label rate.



	Cora					Citeseer					Pubmed		
Label rate	0.5%	1%	2%	3%	4%	0.5%	1%	2%	3%	4%	0.03%	0.05%	0.1%
LP	54.3	60.1	64.0	65.3	66.5	37.7	42.0	44.2	45.7	46.3	58.6	61.9	66.9
GCN	44.5	59.8	68.7	74.4	77.0	43.6	47.4	61.7	66.8	68.6	45.6	55.0	64.9
GAT	41.1	50.2	54.2	60.3	77.0	40.1	46.2	62.8	67.0	68.7	50.2	53.0	60.5
Self-training	55.4	62.5	73.0	76.4	79.1	48.4	59.5	65.4	66.0	70.2	58.7	59.2	66.6
Co-training	50.1	60.3	69.5	76.2	77.8	39.5	53.2	63.5	66.6	69.8	53.3	59.2	63.4
Union	45.7	57.3	72.5	76.3	77.2	41.2	52.9	62.7	65.6	68.1	47.2	59.1	66.3
Intersection	48.7	60.9	73.0	77.3	79.8	49.1	60.1	63.7	68.3	69.4	49.2	54.1	69.7
M3S	59.9	66.7	75.8	77.4	79.2	54.2	62.7	66.2	69.8	70.4	57.0	62.9	68.4
ALaGCN	57.9	66.7	73.7	74.6	78.5	41.0	49.7	59.3	63.5	67.2	57.1	63.0	**71.4**
ALaGAT	48.2	62.4	73.5	75.0	77.3	38.4	52.3	58.6	66.7	68.4	56.8	62.4	69.3
ANEPN (ours)	**66.1**	**73.2**	**77.6**	**78.3**	**79.9**	**60.5**	**64.8**	**68.8**	**70.5**	**71.0**	**60.8**	**69.5**	**71.4**
Gain-GCN	+21.6	+13.4	+8.9	+3.9	+2.9	+16.9	+17.4	+7.1	+3.7	+2.4	+15.2	+14.5	+6.5
Gain-SOTA	+6.2	+6.5	+1.8	+0.9	+0.1	+6.3	+2.1	+2.6	+0.7	+0.6	+2.1	+6.5	+0.0

Summary

- Query optimization for analytical queries
  - Iterative query optimization on MapReduce/Spark
  - Isolation forest on Spark for Intrinsic variable discovery
- Graph mining
  - Tasks: clustering, classification, link prediction, subgraph matching
  - Techniques: Graph neural networks（GCN, ANEPN)

References

- Ryuichi Ito, Chuan Xiao, Makoto Onizuka, Robust Cardinality Estimator by Non-Autoregressive Model. SFDI 2021
- Yuya Ogawa, Seiji Maekawa, Yuya Sasaki, Yasuhiro Fujiwara, Makoto Onizuka: Adaptive Node Embedding Propagation for Semi-supervised Classification. ECML/PKDD 2021
- Daichi Amagata, Makoto Onizuka, Takahiro Hara: Fast and Exact Outlier Detection in Metric Spaces: A Proximity Graph-based Approach. SIGMOD Conference 2021: 36-48
- Seiji Maekawa, Yuya Sasaki, George Fletcher, Makoto Onizuka: GenCAT: Generating Attributed Graphs with Controlled Relationships between Classes, Attributes, and Topology. CoRR abs/2109.04639 (2021)
- Seiji Maekawa, Koh Takeuchi, Makoto Onizuka: New Attributed Graph Clustering by Bridging Attribute and Topology Spaces. J. Inf. Process. 28: 427-435 (2020)
- Yuya Ogawa, Koh Takeuchi, Yuya Sasaki, Makoto Onizuka: Proximity Preserving Nonnegative Matrix Factorization. J. Inf. Process. 28: 445-452 (2020)
- Faith W. Mutinda, Atsuhiro Nakashima, Koh Takeuchi, Yuya Sasaki, Makoto Onizuka: Time Series Link Prediction Using NMF. J. Inf. Process. 27: 752-761 (2019)