15-826: Multimedia Databases and Data Mining

Lecture #29: Graph mining - Generators & tools

Christos Faloutsos

Must-read material (1 of 2)

Fully Automatic Cross-Associations, by D. Chakrabarti, S. Papadimitriou, D. Modha and C. Faloutsos, in KDD 2004 (pages 79-88), Washington, USA

Must-read material (2 of 2)


Main outline

- Introduction
- Indexing
- Mining
  - Graphs – patterns
  - Graphs – generators and tools
  - Association rules
  - …
Detailed outline

• Graphs – generators
  – Erdos-Renyi
  – Other generators
  – Kronecker
• Graphs - tools

Generators

• How to generate random, realistic graphs?
  – Erdos-Renyi model: beautiful, but unrealistic
  – degree-based generators
  – process-based generators
  – recursive/self-similar generators

Erdos-Renyi

• random graph – 100 nodes, avg degree = 2
• Fascinating properties (phase transition)
• But: unrealistic (Poisson degree distribution != power law)

E-R model & Phase transition

• vary avg degree D
• watch $P_c =$ Prob( there is a giant connected component)
• How do you expect it to be?
E-R model & Phase transition

- vary avg degree D
- watch $P_c = \text{Prob( there is a giant connected component)}$
- How do you expect it to be?

 Degree-based

- Figure out the degree distribution (eg., ‘Zipf’)
- Assign degrees to nodes
- Put edges, so that they match the original degree distribution

Process-based

- Barabasi; Barabasi-Albert: Preferential attachment -> power-law tails!
  - ‘rich get richer’
- [Kumar+]: preferential attachment + mimick
  - Create ‘communities’

Process-based (cont’d)

- [Fabrikant+, ‘02]: H.O.T.: connect to closest, high connectivity neighbor
- [Pennock+, ‘02]: Winner does NOT take all
Detailed outline

- Graphs – generators
  - Erdos-Renyi
  - Other generators
  - Kronecker
- Graphs - tools

Recursive generators

- (RMAT [Chakrabarti+,’04])
- Kronecker product

Wish list for a generator:

- Power-law-tail in- and out-degrees
- Power-law-tail scree plots
- shrinking/constant diameter
- Densification Power Law
- communities-within-communities

Q: how to achieve all of them?
A: Kronecker matrix product [Leskovec+05b]

Graph gen.: Problem dfn

- Given a growing graph with count of nodes \( N_1, N_2, \ldots \)
- Generate a realistic sequence of graphs that will obey all the patterns
  - Static Patterns
    - S1 Power Law Degree Distribution
    - S2 Power Law eigenvalue and eigenvector distribution
    - Small Diameter
  - Dynamic Patterns
    - T2 Growth Power Law (2x nodes; 3x edges)
    - T1 Shrinking/Stabilizing Diameters
Graph Patterns

- Power Laws
- Count vs Indegree
- Count vs Outdegree
- Eigenvalue vs Rank

How to match all these properties (+ small diameters, etc)?

**Hint: self-similarity**

- A: RMAT/Kronecker generators
  - With self-similarity, we get all power-laws, automatically,
  - And small/shrinking diameter
  - And 'no good cuts'

**R-MAT: A Recursive Model for Graph Mining**, by D. Chakrabarti, Y. Zhan and C. Faloutsos, SDM 2004, Orlando, Florida, USA


Kronecker Graphs

- Adjacency matrix

**Adjacency matrix**

Intermediate stage
Kronecker Graphs

- Continuing multiplying with \( G_1 \) we obtain \( G_4 \) and so on …
Cartesian product (Kronecker graphs)

- Continuing multiplying with $G_j$ we obtain $G_4$ and so on …

Properties:
- We can PROVE that
  - Degree distribution is multinomial – power law
  - Diameter: constant
  - Eigenvalue distribution: multinomial
  - First eigenvector: multinomial

Problem Definition
- Given a growing graph with nodes $N_1, N_2, \ldots$
- Generate a realistic sequence of graphs that will obey all the patterns
  - Static Patterns
    - Power Law Degree Distribution
    - Power Law eigenvalue and eigenvector distribution
    - Small Diameter
  - Dynamic Patterns
    - Growth Power Law
  - Shrinking/Stabilizing Diameters
- First generator for which we can prove all these properties
Impact: Graph500

- Based on RMAT (= 2x2 Kronecker)
- Standard for graph benchmarks
- [http://www.graph500.org/](http://www.graph500.org/)
- Competitions 2x year, with all major entities: LLNL, Argonne, ITC-U. Tokyo, Riken, ORNL, Sandia, PSC, …

To iterate is human, to recurse is divine

Conclusions - Generators

- Erdos-Renyi: phase transition
- Preferential attachment (Barabasi)
  - Power-law-tail in degree distribution
- Variations
- Recursion – Kronecker graphs
  - Numerous power-laws, + small diameters

Resources

Generators:
- Kronecker (christos@cs.cmu.edu)
- INET: [http://topology.eecs.umin.edu/inet](http://topology.eecs.umin.edu/inet)

Other resources

Visualization - graph algo’s:

References


References, cont’d

- [Broder+, '00] Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet Wiener. *Graph structure in the web*, WWW, 2000
- [Chakrabarti+, '04] D. Chakrabarti, Y. Zhan, C. Faloutsos, SIAM-DM 2004

References, cont’d

References, cont’d


References, cont’d


References, cont’d


References, cont’d

References, cont’d


Graph mining: tools

Main outline

- Introduction
- Indexing
- Mining
  - Graphs – patterns
  - Graphs – generators and tools
  - Association rules
  - ...

Detailed outline

- Graphs – generators
- Graphs – tools
  - Community detection / graph partitioning
  - Algo’s
  - Observation: ‘no good cuts’
  - Node proximity – personalized RWR
  - Influence/virus propagation & immunization
  - ‘Belief Propagation’ & fraud detection
  - Anomaly detection
Problem

- Given a graph, and $k$
- Break it into $k$ (disjoint) communities

Solution #1: METIS

- Arguably, the best algorithm
- Open source, at
  - [http://www.cs.umn.edu/~metis](http://www.cs.umn.edu/~metis)
- and *many* related papers, at same url
- Main idea:
  - coarsen the graph;
  - partition;
  - un-coarsen

Solution #1: METIS

- <and many extensions>
Solution #2

(problem: hard clustering, $k$ pieces)

Spectral partitioning:
- Consider the 2\textsuperscript{nd} smallest eigenvector of the (normalized) Laplacian

Solutions #3, …

Many more ideas:
- Clustering on the $A^2$ (square of adjacency matrix) [Zhou, Woodruff, PODS’04]
- Minimum cut / maximum flow [Flake+, KDD’00]
- …

Detailed outline

• Motivation
• Hard clustering – $k$ pieces
• Hard co-clustering – ($k$, $l$) pieces
• Hard clustering – optimal # pieces
• Soft clustering – matrix decompositions
• Observations

Problem definition

• Given a bi-partite graph, and $k$, $l$
• Divide it into $k$ row groups and $l$ row groups
• (Also applicable to uni-partite graph)
Co-clustering

- Given data matrix and the number of row and column groups \( k \) and \( l \)
- Simultaneously
  - Cluster rows into \( k \) disjoint groups
  - Cluster columns into \( l \) disjoint groups

Simultaneously Given data matrix and the number of row –
Cluster columns into \( l \) disjoint groups

Key Obstacles in Clustering Contingency Tables
- High Dimensionality, Sparsity, Noise
- Need for robust and scalable algorithms

Reference:
1. Dhillon et al. Information-Theoretic Co-clustering, KDD’03
Co-clustering

Observations

- uses KL divergence, instead of L2
- the middle matrix is not diagonal
  - Like in the Tucker tensor decomposition
- s/w at:
  www.cs.utexas.edu/users/dml/Software/cocluster.html

Detailed outline

- Motivation
- Hard clustering – k pieces
- Hard co-clustering – (k,l) pieces
- Hard clustering – optimal # pieces
- Soft clustering – matrix decompositions
- Observations

Problem with Information Theoretic Co-clustering

- Number of row and column groups must be specified

Desiderata:

- ✔ Simultaneously discover row and column groups
- ✗ Fully Automatic: No “magic numbers”
- ✔ Scalable to large graphs

Graph partitioning

- Documents x terms
- Customers x products
- Users x web-sites
Graph partitioning

- Documents x terms
- Customers x products
- Users x web-sites

Q: HOW MANY PIECES?

Graph partitioning

- Documents x terms
- Customers x products
- Users x web-sites

Q: HOW MANY PIECES?

A: MDL/ compression

Cross-association

Desiderata:
- Simultaneously discover row and column groups
- Fully Automatic: No “magic numbers”
- Scalable to large matrices

Reference:
1. Chakrabarti et al. Fully Automatic Cross-Associations, KDD’04

What makes a cross-association “good”?

Why is this better?
What makes a cross-association “good”?

- Column groups versus Row groups

Why is this better?

simpler; easier to describe
easier to compress!

Problem definition: given an encoding scheme
- decide on the # of col. and row groups $k$ and $l$
- and reorder rows and columns,
- to achieve best compression

Main Idea

Good Compression

Better Clustering

Total Encoding Cost = \[ \sum_i \text{size}_i \times H(x_i) + \text{Cost of describing cross-associations} \]

Code Cost

Description Cost

Minimize the total cost (# bits)
for lossless compression

Algorithm

1=5 col groups

$k=1$,
$l=2$

$k=2$,
$l=3$

$k=3$,
$l=4$

$k=4$,
$l=5$
Experiments

“CLASSIC”
- 3,893 documents
- 4,303 words
- 176,347 “dots”

Combination of 3 sources:
- MEDLINE (medical)
- CISI (info. retrieval)
- CRANFIELD (aerodynamics)

“CLASSIC” graph of documents & words:
- \( k=15, l=19 \)

Experiments

MEDLINE (medical)

insipidus, alveolar, aortic, death, prognosis, intravenous

blood, disease, clinical, cell, tissue, patient

“CLASSIC” graph of documents & words:
- \( k=15, l=19 \)

Experiments

CISI (Information Retrieval)

providing, studying, records, development, students, rules

abstract, notation, works, construct, bibliographies

“CLASSIC” graph of documents & words:
- \( k=15, l=19 \)
**Experiments**

"CLASSIC" graph of documents & words:

- **k=15, l=19**

**Algorithm**

Code for cross-associations (matlab):


Variations and extensions:
- ‘Autopart’ [Chakrabarti, PKDD’04]
- [www.cs.cmu.edu/~deepay](http://www.cs.cmu.edu/~deepay)

**Algorithm**

- Hadoop implementation [ICDM’08]
Detailed outline

• Motivation
• Hard clustering – $k$ pieces
• Hard co-clustering – $(k,l)$ pieces
• Hard clustering – optimal # pieces
• (Soft clustering – matrix decompositions
  – PCA, ICA, non-negative matrix factorization,
  …)
• Observations

Observation #1

• Skewed degree distributions – there are
  nodes with huge degree (>10^4), in
  facebook/linkIn popularity contests!
• TRAP: ‘find all pairs of nodes, within 2
  steps from each other’

Observation #2

• TRAP: shortest-path between two nodes
• (cheat: look for 2, at most 3-step paths)
• Why:
  – If they are close (within 2-3 steps): solved
  – If not, after ~6 steps, you’ll have – the whole
    graph, and the path won’t be very meaningful,
    anyway.
Observation #3

• Maybe there are no good cuts: ``jellyfish” shape [Tauro+'01], [Siganos+,'06], strange behavior of cuts [Chakrabarti+,'04], [Leskovec+,'08]

Jellyfish model [Tauro+]

Strange behavior of min cuts

• ‘negative dimensionality’ (!)

---


“Min-cut” plot

- Do min-cuts recursively.

\[ \log(\text{mincut-size} / \#\text{edges}) \]

\[ \log(\# \text{edges}) \]

N nodes

For a d-dimensional grid, the slope is \(-1/d\)

\( \text{Mincut size} = \sqrt{N} \)
“Min-cut” plot

• What does it look like for a real-world graph?

\[
\log(\text{mincut-size} / \#\text{edges})
\]

\[
\log(\#\text{edges})
\]

Experiments

• Datasets:
  – Google Web Graph: 916,428 nodes and 5,105,039 edges
  – Lucent Router Graph: Undirected graph of network routers from [link]
    www.isi.edu/scan/mercator/maps.html; 112,969 nodes and 181,639 edges
  – User Website Clickstream Graph: 222,704 nodes and 952,580 edges

Experiments

• Used the METIS algorithm [Karypis, Kumar, 1995]

  • Google Web graph
  • Values along the y-axis are averaged
  • We observe a “lip” for large edges
  • Slope of -0.4, corresponds to a 2.5-dimensional grid!

  Slope ~ -0.4

Experiments

• Used the METIS algorithm [Karypis, Kumar, 1995]

  • Similarly, for
    • Lucent routers
    • clickstream

  -0.57; -0.45

Experiments

Conclusions – Practitioner’s guide

- Hard clustering – $k$ pieces
- Hard co-clustering – $(k,l)$ pieces
- Hard clustering – optimal # pieces
- Observations
  
  ‘jellyfish’:
  Maybe, there are no good cuts