Recap

- **Clustering**
- **Label Propagation**
- **Clusters and Communities in Graphs**
  - Evaluation Measures
- **Spectral Graph Clustering**
  - Spectra of affinity matrix
  - Spectra of graph Laplacian
  - Spectra of normalized Laplacian
  - Spectra of Modularity Matrix

Link Prediction

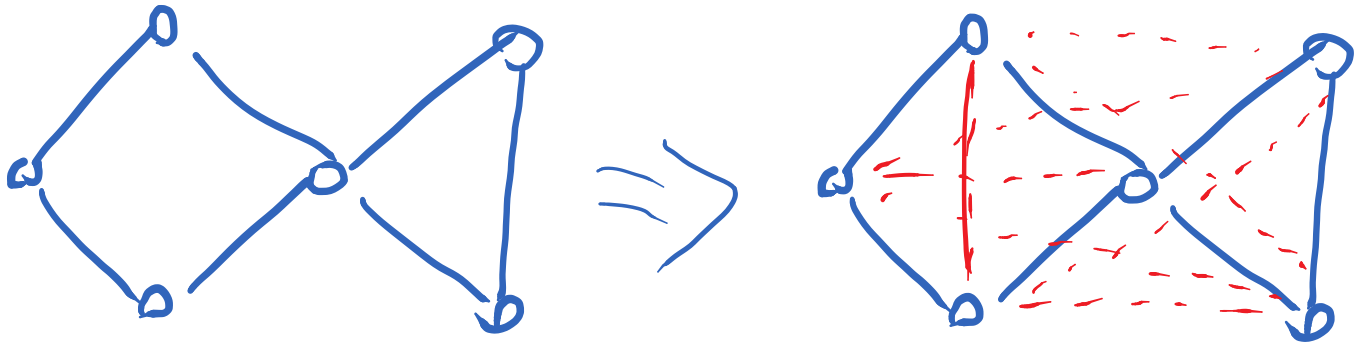
Sarunas Girdzijauskas

ID2211

What is Link Prediction?

- **Networks are (usually) not static**
  - Change over time
  - How do they change?
- **Link prediction**
  - Given a snapshot of a network at time *t,* predict edges added (removed) in the interval *(t,t')*
- **Link completion**
  - Network might have "missing links" (not well observed network, noise etc).
  - Finding unobserved edges
- **Link reliability**
  - Estimate the realiability of of given links in the graph
- *Type of Predictions:*
  - *Link existance (binary classification problem)*
  - *Link weight (regression problem). E.g., predicting movie rating for users (recommender systems – ID2222 course)*
  - *Link type (multi-class classification problem)*

Example



- Number of missing Edges?
  - Possible edges $O(N^2)$
  - Missing edges $O(N^2\text{-}E)$, in sparse graphs $O(N^2)$
- Random guess?
  - $O(1/N^2)$

How to Predict a link?

The main issue:
How to come up with a good similarity function

- Any ideas?

- Link prediction by **proximity scoring**
  - For each pair of nodes **compute proximity (similarity) score** *sim(v1,v2)*
  - Sort all pairs by the decreasing score
  - **Select top n pairs** (or above some threshold) as new links
  - Can be **surprisingly effective**!

$sim(v_i, v_j)$?

- **Vertex feature aggregatoins**
  - Preferrential attachment

$$k_i \cdot k_j = |\mathcal{N}(v_i)| \cdot |\mathcal{N}(v_j)|$$

$$k_i + k_j = |\mathcal{N}(v_i)| + |\mathcal{N}(v_j)|$$

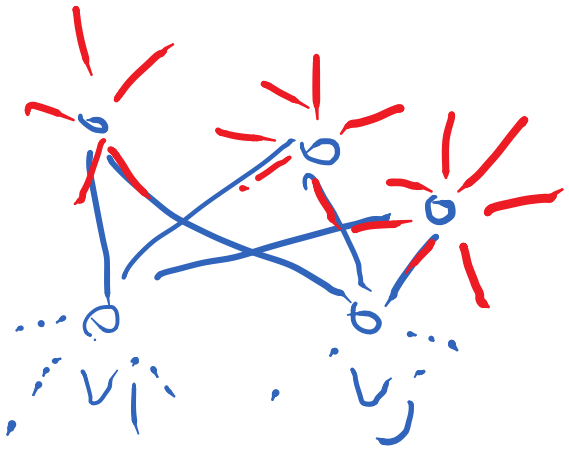
  - Clustering Coefficient

$$CC(v_i) \cdot CC(v_j)$$

$$CC(v_i) + CC(v_j)$$

Scoring/Similarity functions (cont.)

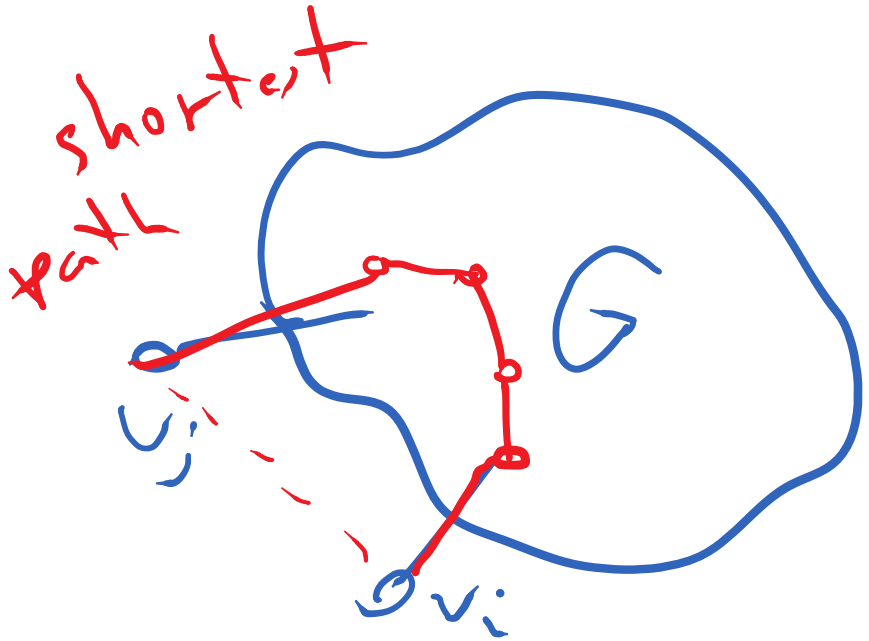
These are **Local-Neighborhood** scores!



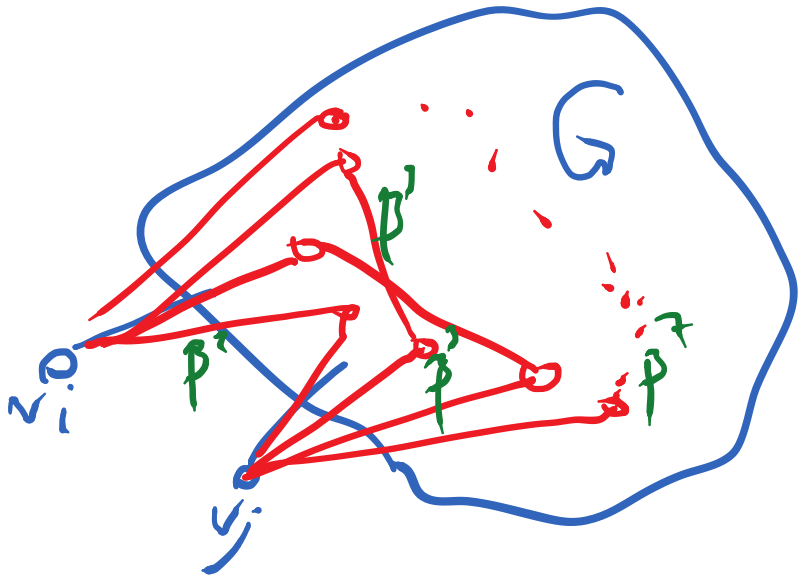
- $sim(v_i,v_j)$? Other suggestions?
  - Capturing **Node Homophily** property!
  - Most of the networks are born by this principle
- Number of **common neighbors** $|\mathcal{N}(v_i) \cap \mathcal{N}(v_j)|$
- **Jaccard's coefficient** $\frac{|\mathcal{N}(v_i) \cap \mathcal{N}(v_j)|}{|\mathcal{N}(v_i) \cup \mathcal{N}(v_j)|}$
- **Adamic/Adar Score** $\sum_{v \in \mathcal{N}(v_i) \cap \mathcal{N}(v_j)} \frac{1}{\log |\mathcal{N}(v)|}$

One of the strongest link predictors

Non-local neighborhood?

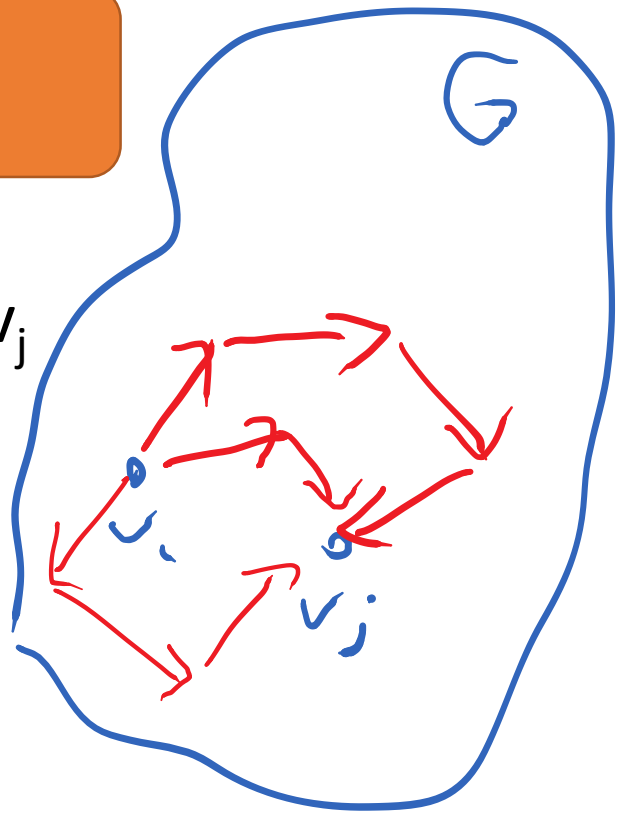


- **sim(v_i,v_j)** based on paths
  - Suggestions?
- **Shortest path** $-\min_s\{path_{ij}^s > 0\}$
- **Katz Score** $\sum_{l=1}^{\infty} \beta^l |paths^{(l)}(v_i, v_j)| = \sum_{l=1}^{\infty} (\beta A)_{ij}^l = (I - \beta A)^{-1} - I$
  - Propotional to the sum of the lengths of all the paths between the two nodes
  - Longer path lengths are penalized with β^{path_length}
  - Sum Converges if $\beta < 1/\lambda_1$ where, λ_1 is a dominant eigenvalue of A



More similarity scores

These are **Path-based** scores!

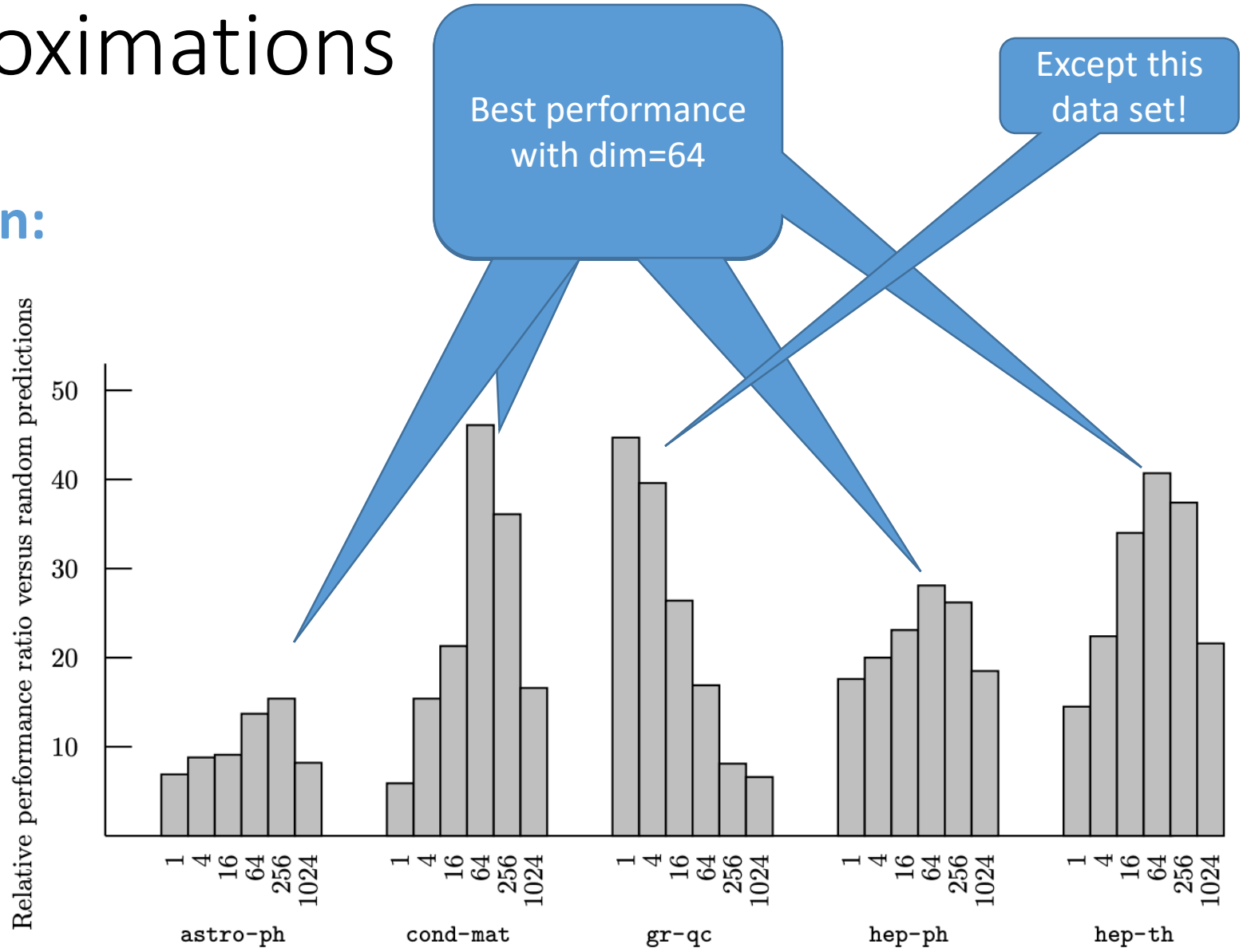


- **Expected number of random walk steps** from v_i to v_j
  - Similar to label propagation with absorbing states
  - Hitting time: - H_{ij}
  - Commute time: - (H_{ij}+H_{ji})
    - For directed graphs
  - Normalized Hitting/commute time: (H_{ij} π_j +H_{ji} π_i)
    - Normalized by stationary distribution of walks from i: π_l
- **PageRank based variants**
  - SimRank/Personalized Page Rank

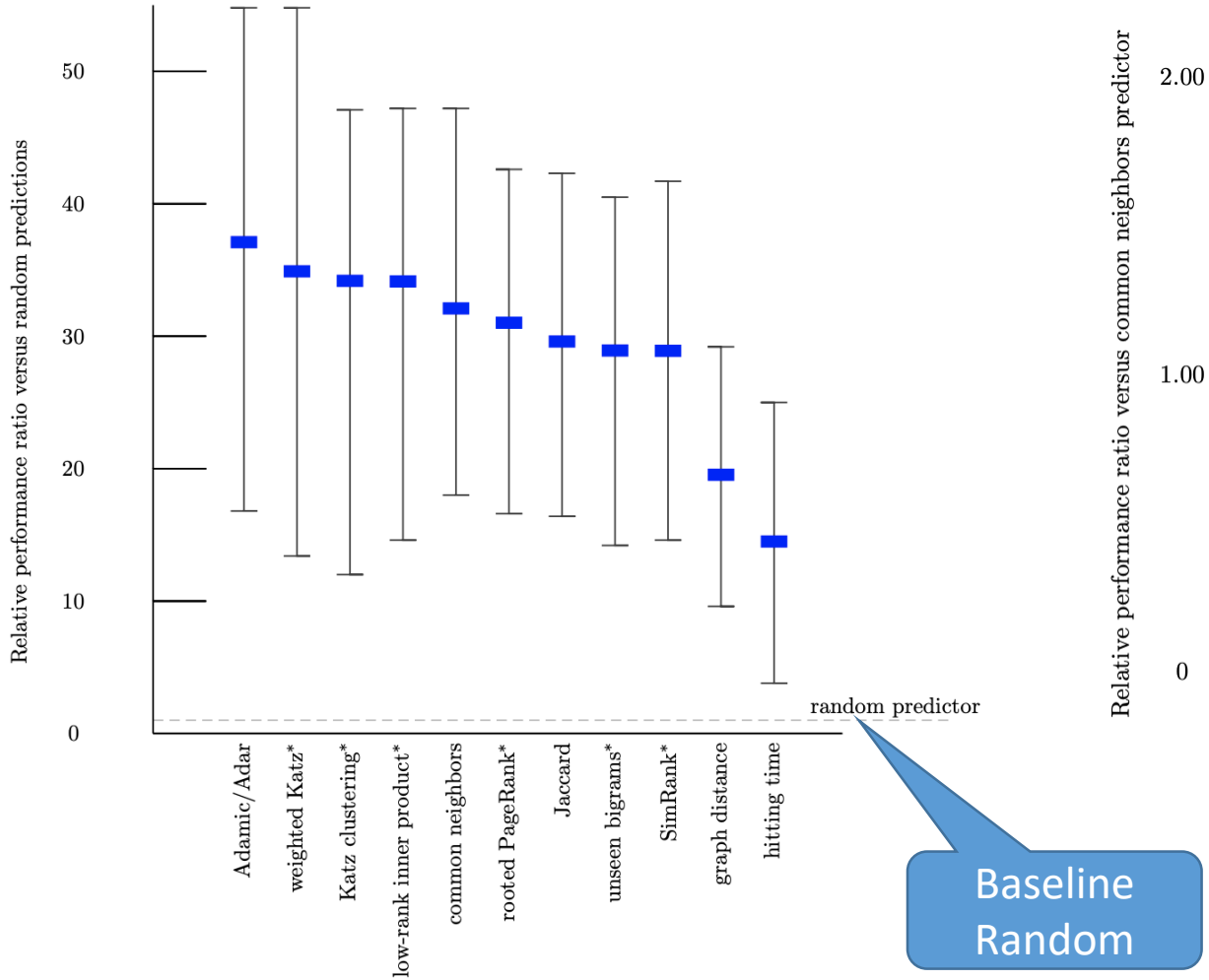
Low-Rank Approximations

- **Eigen-decomposition:**
  - $\boldsymbol{A = X \Lambda X^T}$
- **Truncated SVD:**

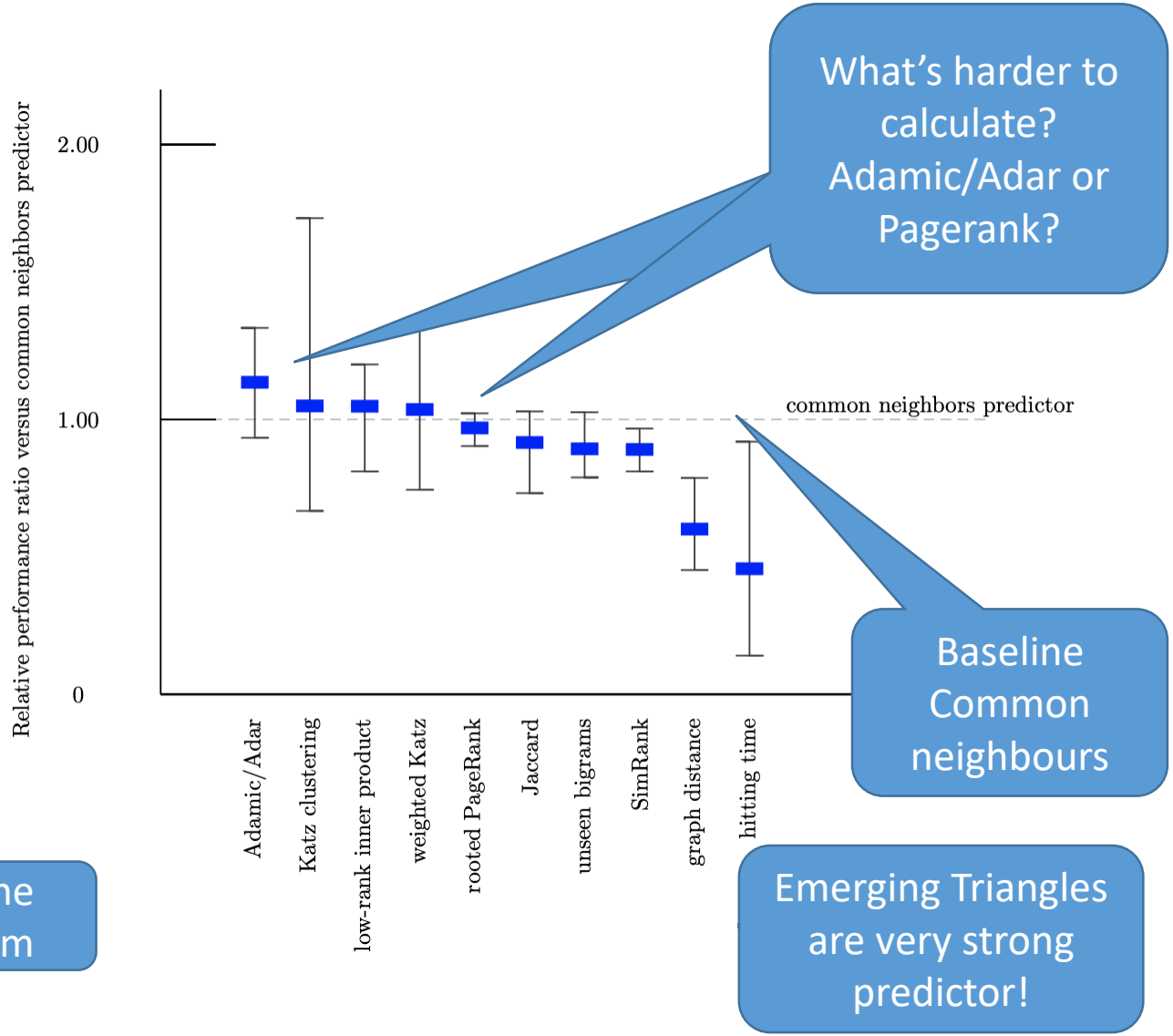
$$A \approx \sum_k U_k S_k V_k^T$$



D. Liben-Nowell and J. Kleinberg. *The link prediction problem for social networks*. 2007



Baseline Random



What's harder to calculate? Adamic/Adar or Pagerank?

Baseline Common neighbours

Emerging Triangles are very strong predictor!

Treating as ML Binary Classification problem

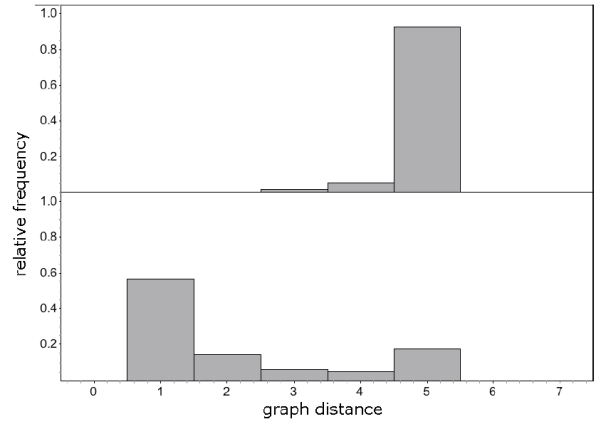
- **Supervised learning**
  - Features Generation
  - Model training (choose your “favourite” ML model, e.g., logistic regression, SVM, decision trees etc).
  - Testing (model application)
- **Features:**
  - Treat the similarity scores as features (structure of the graph)
    - Make sure not to take correlated features (e.g., linear regression would work bad)
      - E.g., random walk based scores
      - Local neighbourhood based scores
  - Content based node proximity features
    - E.g., on DBLP graph you might want to see what kind of papers are co-written, which domain etc (i.e., “paper domain” as one more feature for your classifier)

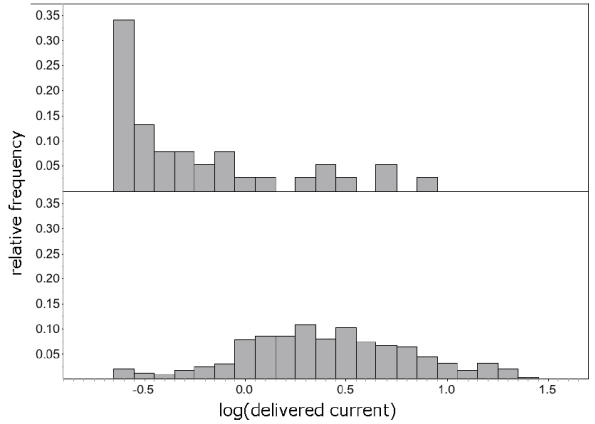
Challenging classification problem

- **Computational cost** of evaluating of very large number of possible edges (quadratic in number of nodes)
- **Highly imbalanced class distribution:** number of positive examples (existing edges) grows linearly and negative quadratically with number on nodes
  - ML classifier just might ”choose” to label everything as ”negative” since there are not many positive examples anyway.
  - Training with imbalanced classes is very difficult
    - Down-sampling (i.e. take very few negative examples) and up-sampling (oversampling positive examples)

Discriminative abilities of features

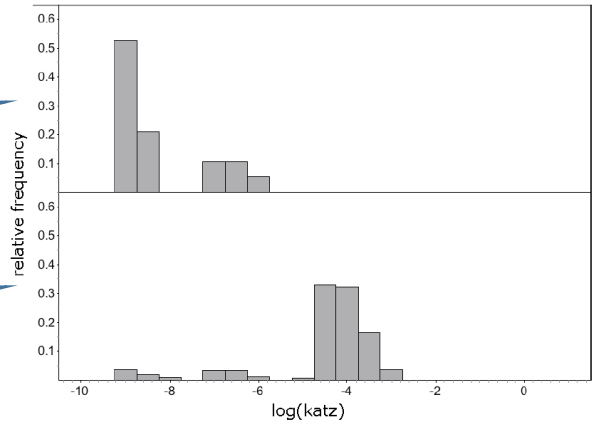
- The values of the features should be different for positive and negative examples!
  - Otherwise it is a bad feature!
- E.e., Katz score very strong predictor (different values for positive and negative examples
  - Rattigan et al. The Case For Anomalous Link Discovery, KDD 2005





Edge does not exist

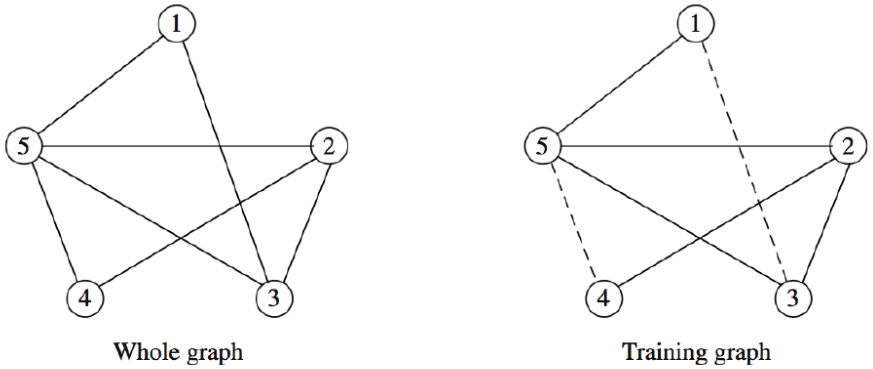
Edge exists



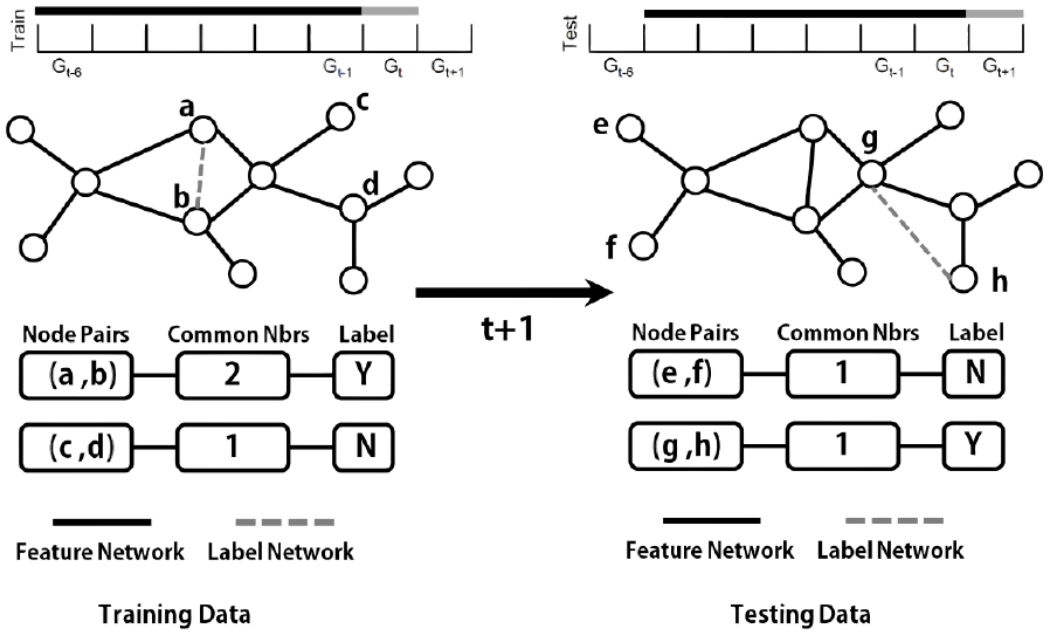
From M. Rattigan, D. Jensen. The case for anomalous link discovery. ACM SIGKDD Explorations Newsletter. v 7, n 2, pp 41-47, 2005

Evaluation

- No "**time evolution**"

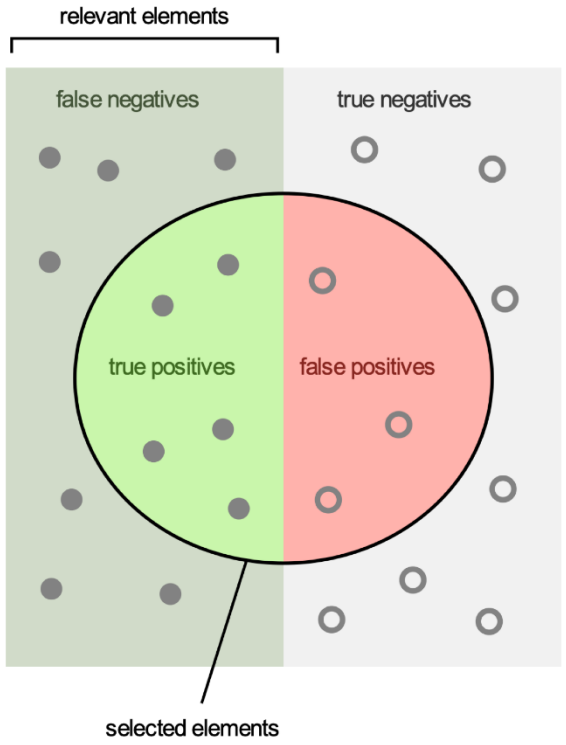


- Evaluation of **Evolving Networks**
  - Shifting training-testing data through time
    - "Evaluating Link Prediction Methods" Yang et a. 2015



Evaluation Measures

- Precision
  - How many ”selected items are relevant”?
- Recall (True positive rate)
  - How many “relevant items were selected”?
- False Alarm Ratio (False positive rate)
  - FPR=FP/(FP+TN)
- Accuracy
  - (TP+TN)/ALL
- F-measure =2*Precision*Recall/(Precision + Recall)
- Area under the ROC
  - TPR vs FPR while changing threshold



Real Life ☺

- Prediction results don't have to be very good.
- **Recall?**
  - Not important (as long as you get at least some good predictions).
- **Precision?**
  - Not important (as long as at least some of the top predictions you return are good)

Relevant Papers

- D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. Journal of the American Society for Information Science and Technology, 58(7):1019?1031, 2007
- R. Lichtenwalter, J.Lussier, and N. Chawla. New perspectives and methods in link prediction. KDD 10: Proceedings of the 16th ACM SIGKDD, 2010
- M. Al Hasan, V. Chaoji, S. Salem, M. Zaki, Link prediction using supervised learning. Proceedings of SDM workshop on link analysis, 2006
- M. Rattigan, D. Jensen. The case for anomalous link discovery. ACM SIGKDD Explorations Newsletter. v 7, n 2, pp 41-47, 2005
- M. Al. Hasan, M. Zaki. A survey of link prediction in social networks. In Social Networks Data Analytics, Eds C. Aggarwal, 2011.

Acknowledgement: Some slides were based on Leonid Zhukov's lectures

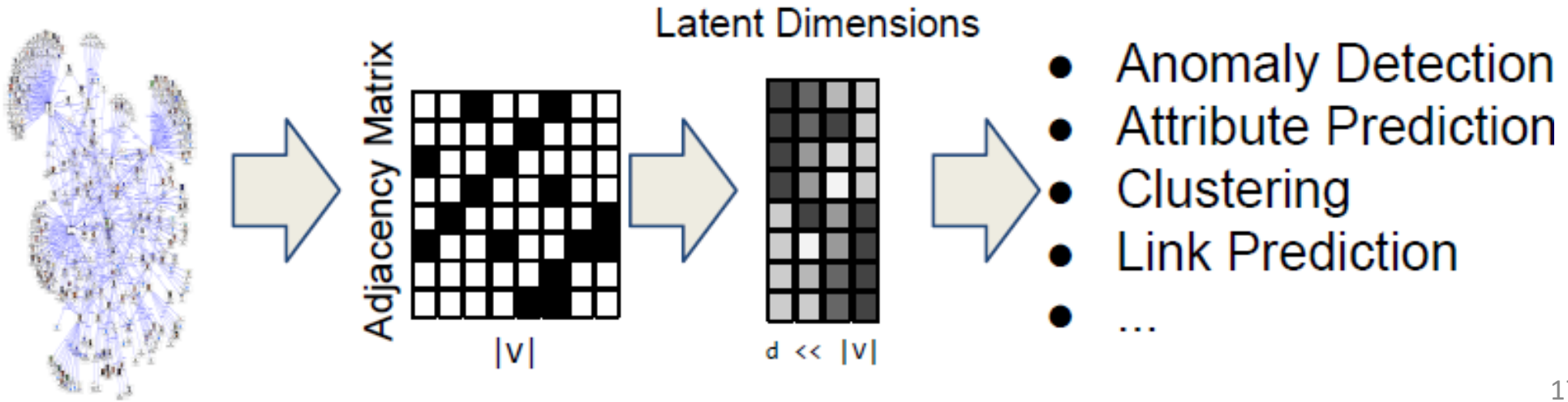
Kahoot! time

Graph Represntation Learning

Or... Yet another way to look at the same problem

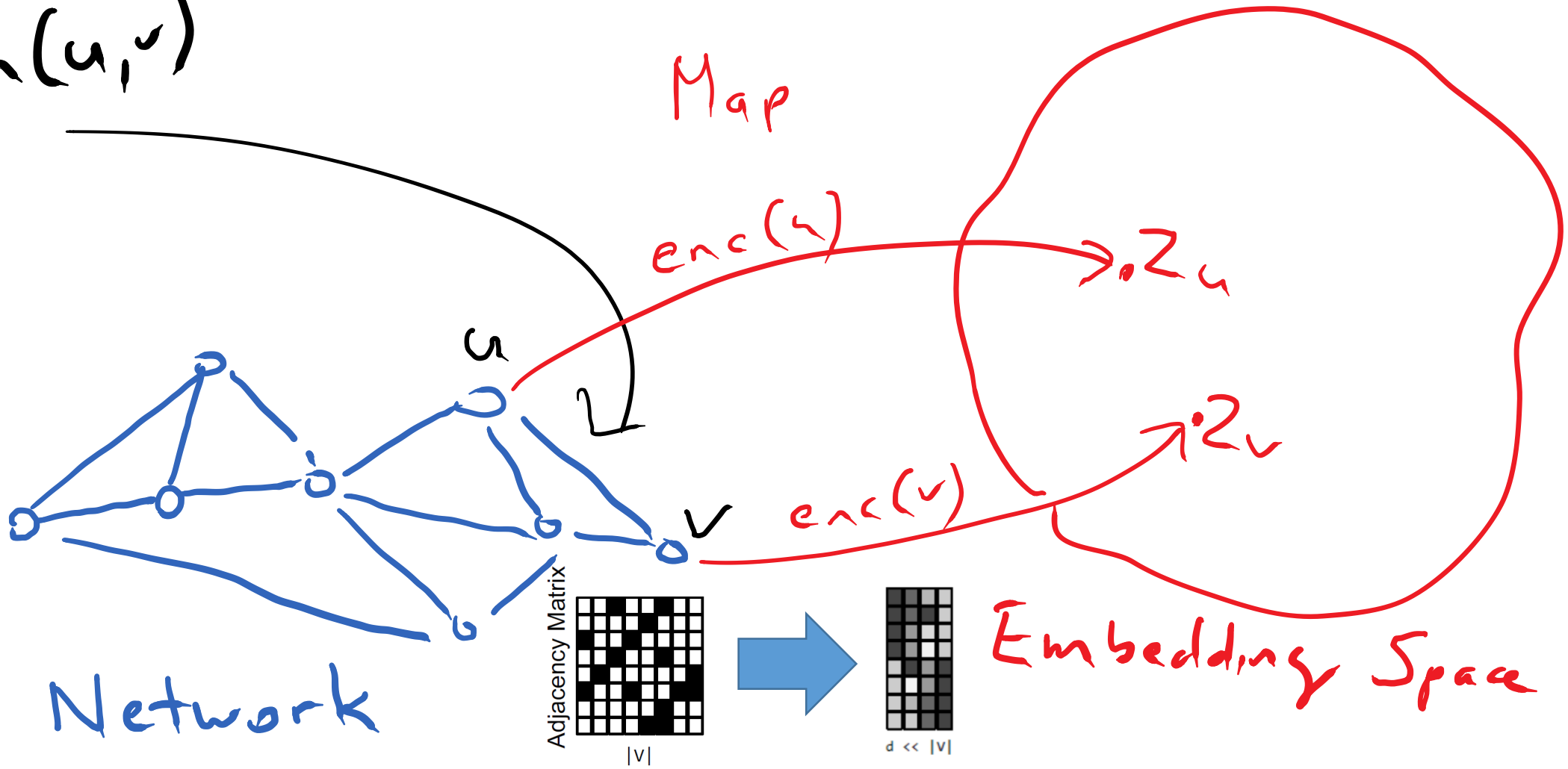
Graph Representation Learning

- **Graph Representation Learning** or **Network Embedding**
- We **map each node** in a network into **a low-dimensional space**
  - **Similarity of embedding** between nodes indicates their **Network similarity**
  - **Encode** network information and generate node representation

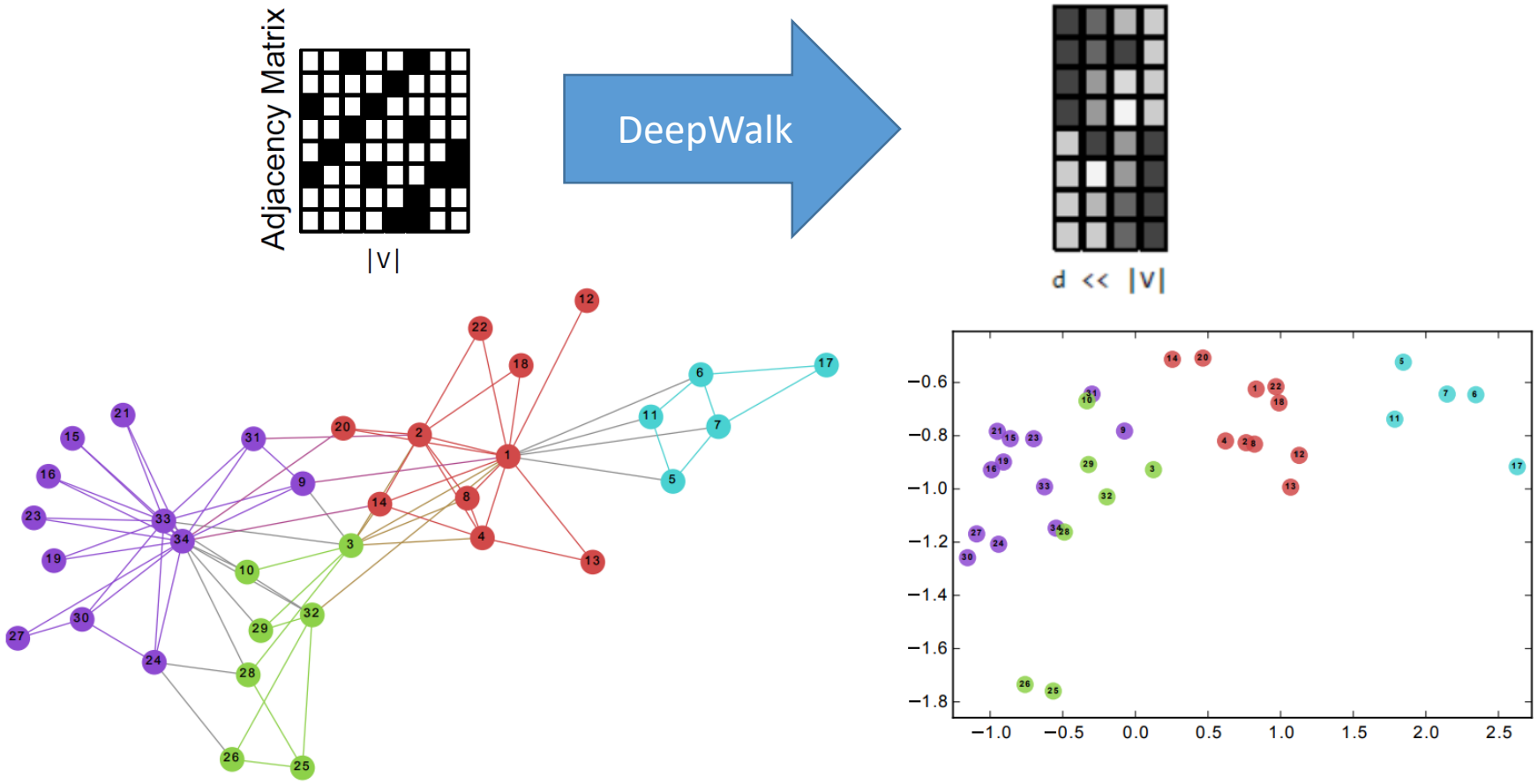


Define $sim(u,v)$

Such that $sim(u,v) \approx z_v^T z_u$



Example from Perozzi et al. KDD 2014



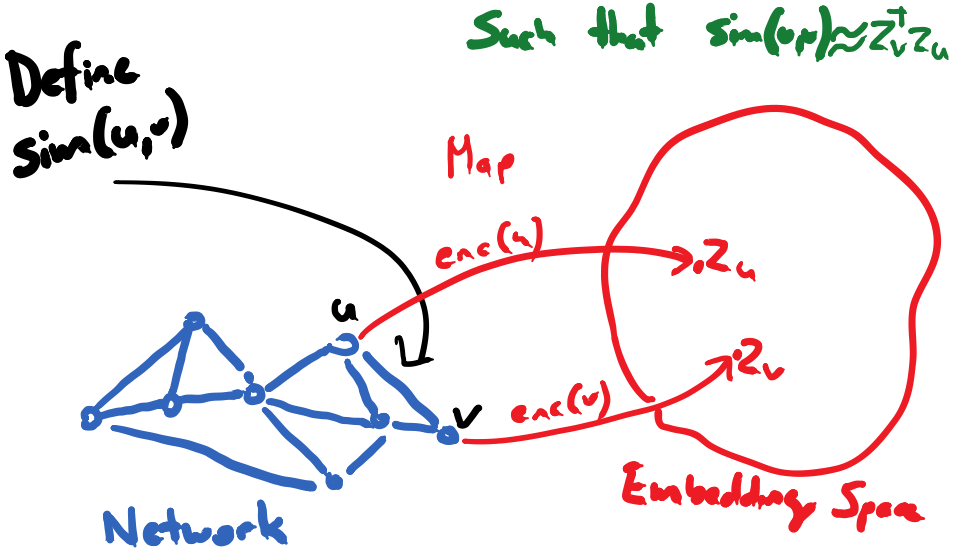
(a) Input: Karate Graph (b) Output: Representation

Perozzi et al. “DeepWalk: Online Learning of Social Representations” KDD 2014

Learning Node Embeddings

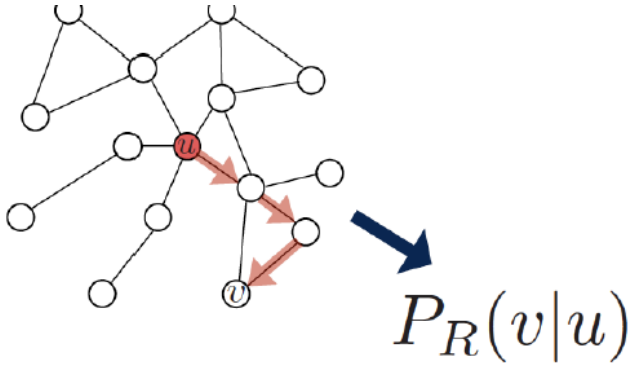
1. **Define an encoder** (i.e., a mapping from nodes to embeddings)
2. **Define a node similarity function** (i.e., a measure of similarity in the original network).
3. **Optimize the parameters of the encoder so that:**

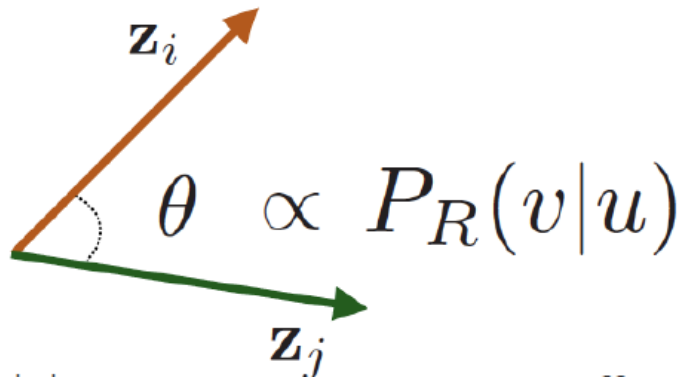
   $\text{sim}(u, v) \approx \mathbf{z}_v^T \mathbf{z}_u$



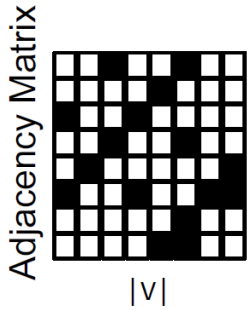
Graph Representation Learning

- Plan of Action:
  - Take a Network and **define your similarity between two nodes**
    - Sim functions can be used as from the beginning of the class.
    - However, "**Random Walk Approaches**" are mainly used in Graph Representation Learning
  - Simulate or Extract information activity/interactions behaviour on top of the network
    - E.g., Estimate probability $P_R(v|u)$ of visiting node *v* starting from node *u* using some **(random) walk strategy *R*** over the network.
      - *i.e.,* Simulate r random walks of length l starting from each node *u*
  - **Optimize embeddings** to encode these random walk statistics
    - e.g., by using Stochastic Gradient Descent
    - The most promintnet papers: DeepWalk, Node2vec

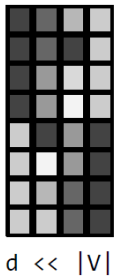




Graph Representation Learning

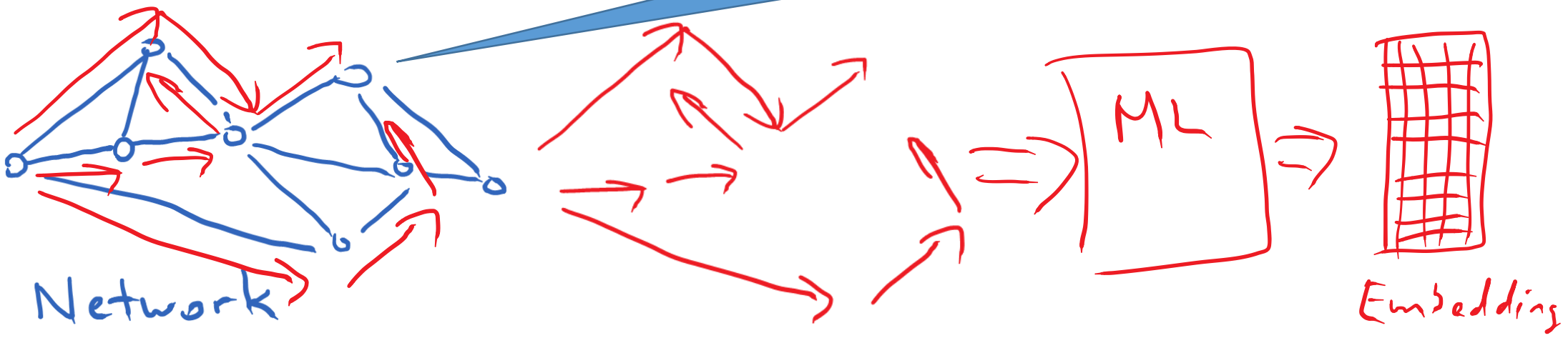


DeepWalk,
Node2Vec, LINE,
Cas2Vec
Etc...



- Anomaly Detection
- Attribute Prediction
- Clustering
- Link Prediction
- ...

Sometimes we do not even have the network (can't observe), but can only observe the activity (e.g., Twitter hashtag cascades, e.g., cas2vec)



Different Random Walk strategies to define Node Similarities

- **Non-Biased Random walk:**
  - **DeepWalk** (Perozzi et al 2013)
- **Different kinds of biased random walks:**
  - Parametrized Random Walk (tradeoff between BFS vs DFS) (Node2vec: Grover et al. 2016)
  - Based on node attributes (Dong et al., 2017).
  - Based on a learned weights (Abu-El-Haija et al., 2017)
- **Alternative optimization schemes:**
  - Directly optimize based on 1-hop and 2-hop random walk probabilities (as in LINE from Tang et al. 2015).
- **Network preprocessing techniques:**
  - Run random walks on modified versions of the original network (e.g., Ribeiro et al. 2017's struct2vec, Chen et al. 2016's HARP).
- **Network agnostic strategies**
  - Take the cascades observed on the networks instead. (Kefato et al. 2018, Cas2vec)
- **Important: you should choose node similarity definition/walk strategy that matches your application!!!**

The End…

Thanks for bearing with me!