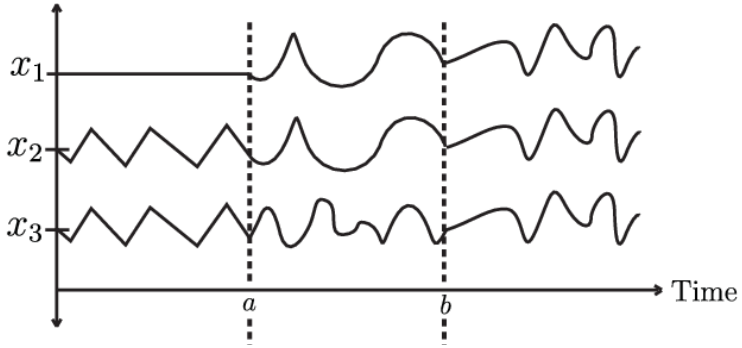
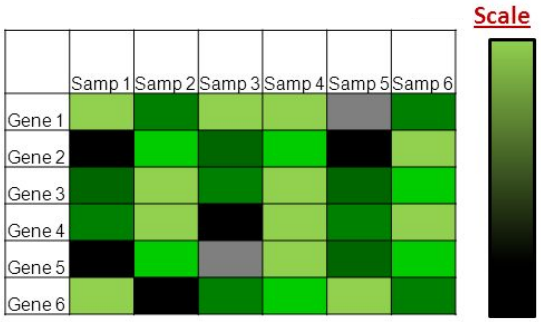
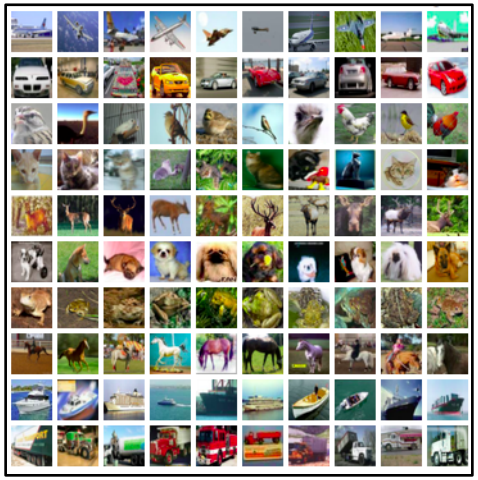
Network Construction, Inference, and Deconvolution

CS224W: Analysis of Networks
Jure Leskovec, Stanford University
http://cs224w.stanford.edu

Raw Data are often not Networks





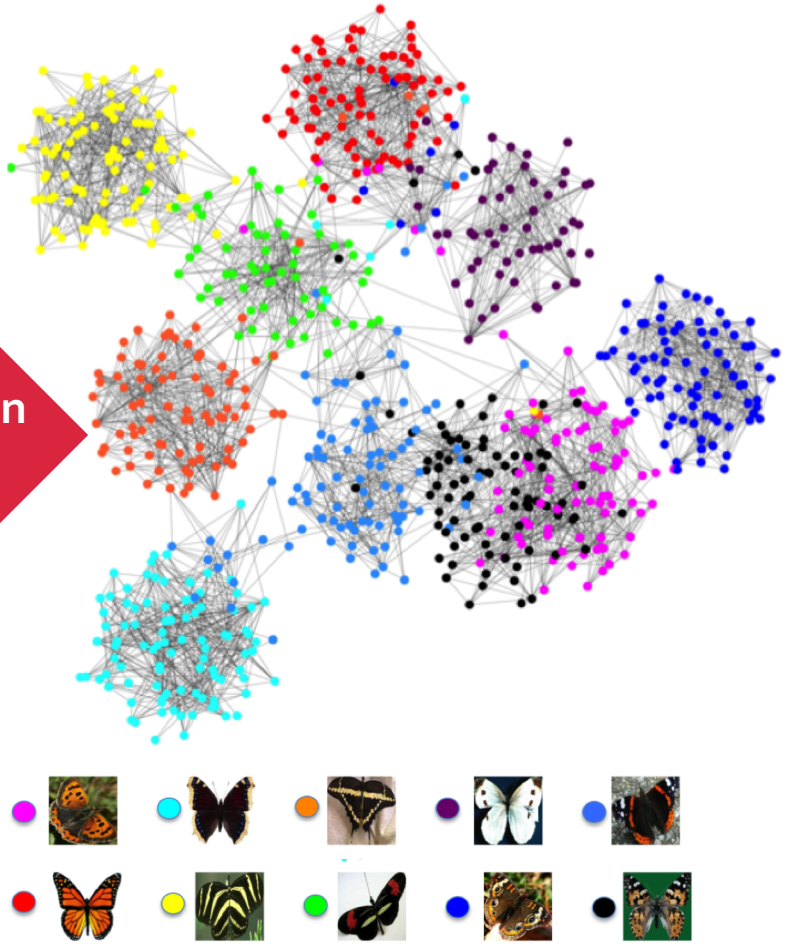


	1	2	3	4	5	6	7	8	9
man	1	0	0	0	0	0	0	0	0
woman	0	1	0	0	0	0	0	0	0
boy	0	0	1	0	0	0	0	0	0
girl	0	0	0	1	0	0	0	0	0
prince	0	0	0	0	1	0	0	0	0
princess	0	0	0	0	0	1	0	0	0
queen	0	0	0	0	0	0	1	0	0
king	0	0	0	0	0	0	0	1	0
monarch	0	0	0	0	0	0	0	0	1

Feature matrices, relationship tables, time series, document corpora, image datasets, etc.

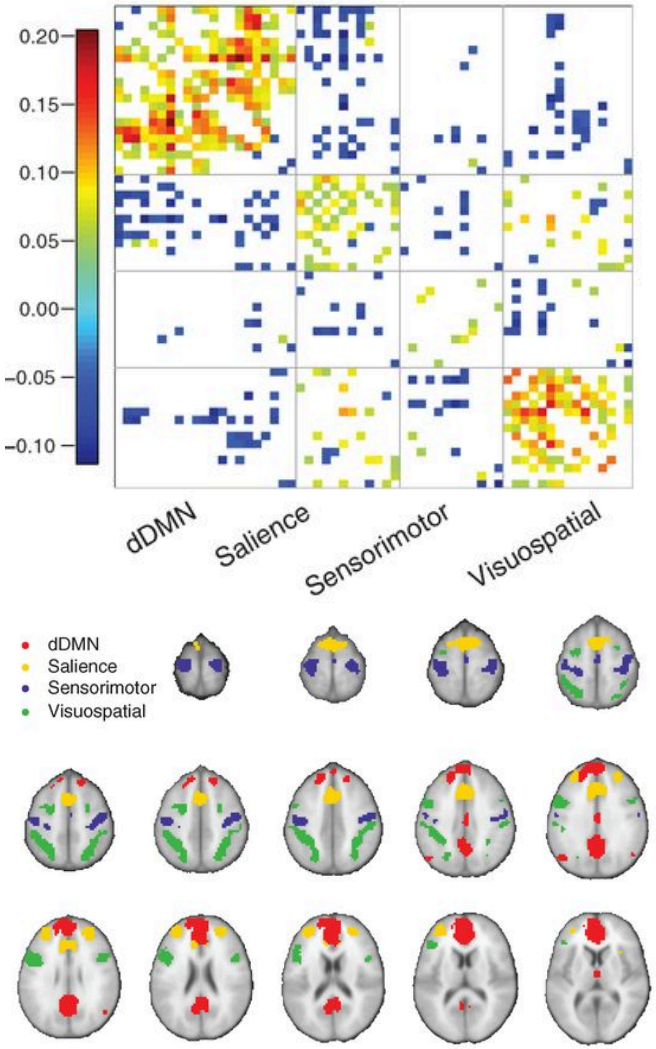
How to Construct Networks?

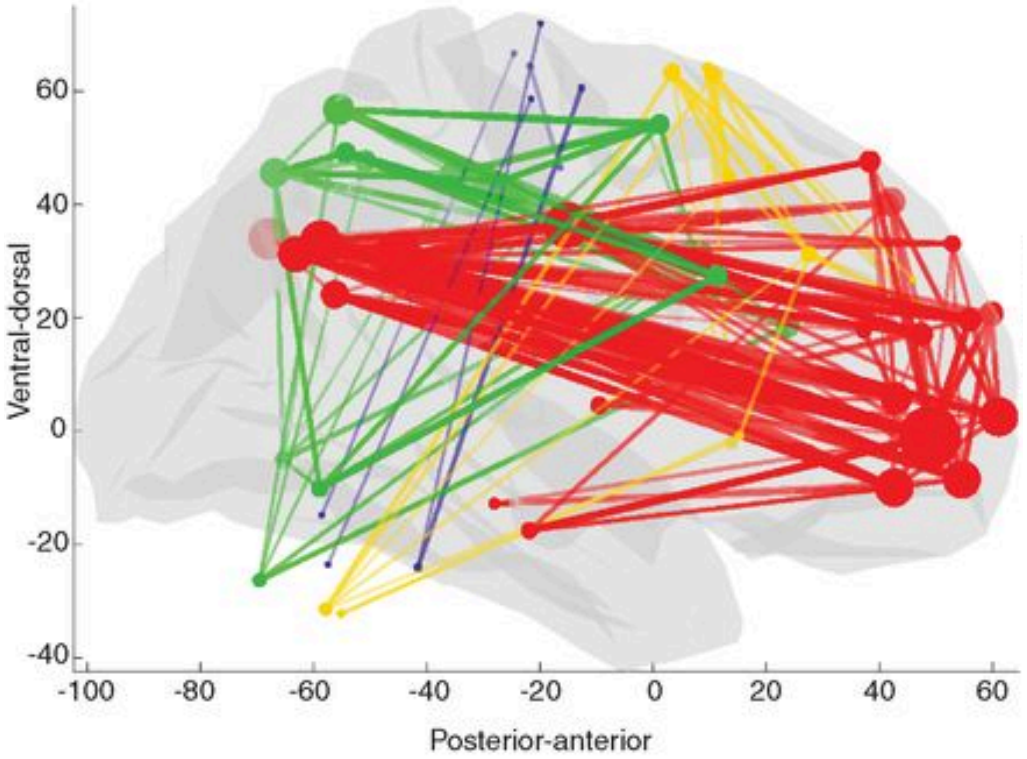
Network construction and inference



Today: How to construct and infer networks from raw data?

Why? -- Networks are Useful





Jonas Richiardi *et al.*, Correlated gene expression supports synchronous activity in brain networks. *Science* 348:6240, 2015.

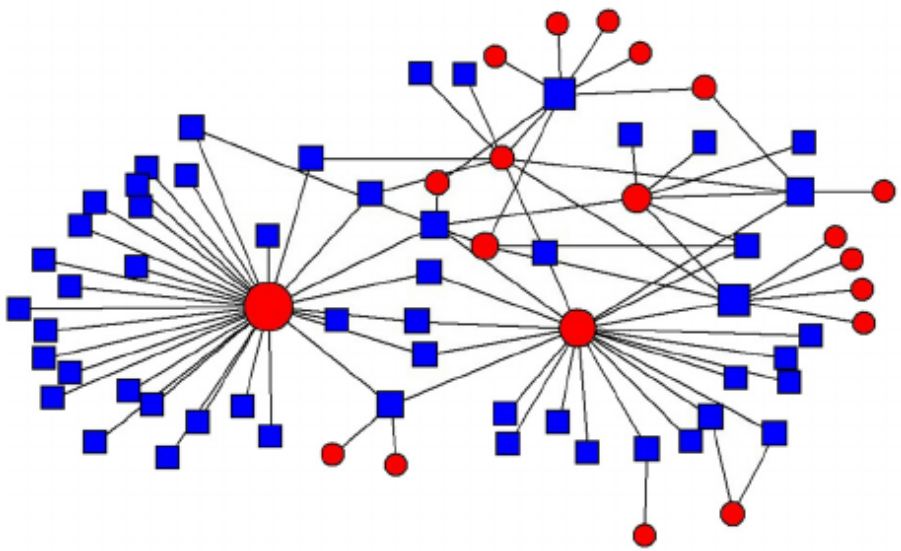
Plan for Today

1) **Multimode Network Transformations:**
   - K-partite and bipartite graphs
   - One-mode network projections/folding
   - Graph contractions
2) **K-Nearest Neighbor Graph Construction**
3) **Network Deconvolution:**
   - Direct and and indirect effects in a network
   - Inferring networks by network deconvolution

Multimode Network Transformations

Bipartite and K-partite Networks

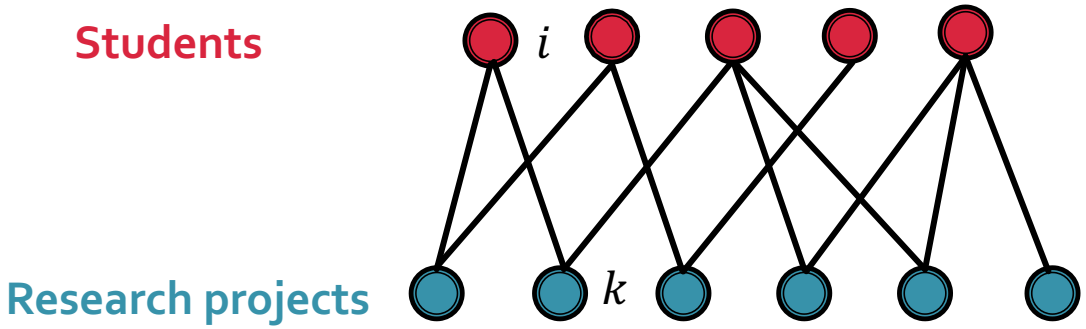
- Most of the time, when we create a network, all nodes represent **objects of the same type:**
  - People in social nets, bus stops in route nets, genes in gene nets
- **Multi-partite networks** have **multiple types of nodes**, where edges exclusively go from one type to the other:
  - **2-partite student net:** Students <-> Research projects
  - **3-partite movie net:** Actors <-> Movies <-> Movie Companies



Network on the left is a social bipartite network. **Blue squares** stand for people and **red circles** represent organizations

One-mode Projections: Example

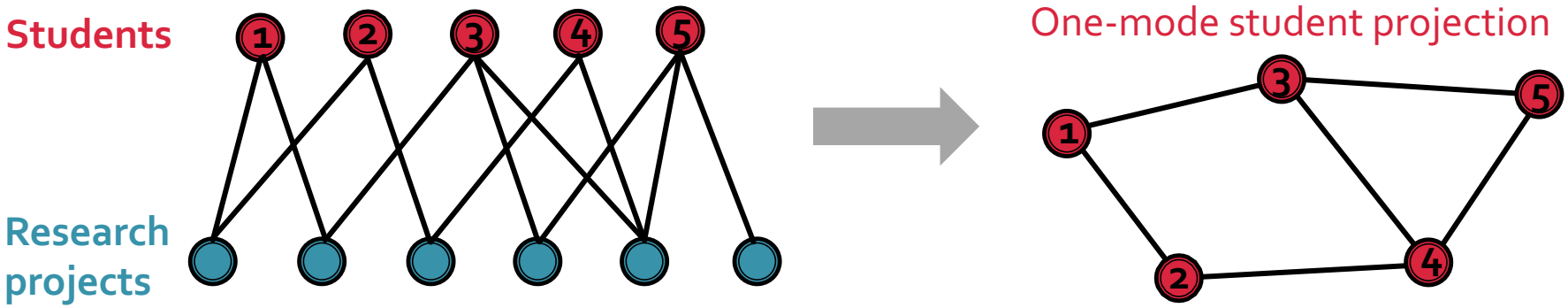
- **Example:** Bipartite student-project network:
  - **Edge:** Student i works on research project k



- **Two network projections of student-project network:**
  - **Student network:** Students are linked if they work together in **one or more projects**
  - **Project network:** Research projects are linked if **one or more students** work on both projects
- **In general:** K-partite network has K one-mode network projections

One-mode Projections: Example

- **Example:** Projection of **bipartite student-project network** onto the **student mode**:



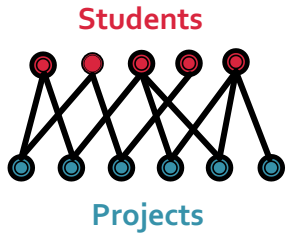
- **Consider students 3, 4, and 5 connected in a triangle:**
  - Triangle can be a result of:
    - **Scenario #1:** Each pair of students work on a different project
    - **Scenario #2:** Three students work on the same project
  - One-mode network projections **discard some information:**
    - Cannot distinguish between #1 and #2 just by looking at the projection

(1) Constructing One-mode Projections

- **One-mode projection** onto **student mode**:
  - #(projects) that students i and j work together on is equivalent to the number of **paths of length 2** connecting i and j in the bipartite network

- Let C be **incidence matrix** of student-project net:

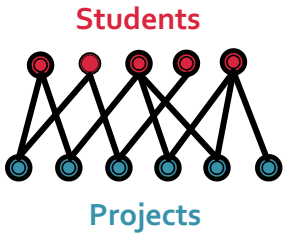
$$C_{ik} = \begin{cases} 1 \text{ if } i \text{ works on project } k \\ 0 \text{ otherwise} \end{cases}$$



- C is an $n \times m$ **binary non-symmetric matrix**:
  - n is #(students), m is #(projects)

(2) Constructing One-mode Projections

- **Idea:** Use C to construct various one-mode network projections
- **Weighted student network:**



$$B_{ij} = \begin{cases} w_{ij}, \#(\text{projects}) \text{ that } i \text{ and } j \text{ collaborate on} \\ 0 \quad \text{otherwise} \end{cases}$$

  - $B_{ij} = \sum_{k=1}^{m} C_{ik} C_{jk}$, *i.e.,* the number of **paths of length 2** connecting students i and j in the bipartite network
  - $\boldsymbol{B} = \boldsymbol{C}\boldsymbol{C}^{\boldsymbol{T}}$ and B_{ii} represents #(projects) that student i works on

- **Similarly, weighted project network:**

$$D_{kl} = \begin{cases} w_{kl}, \#(\text{students}) \text{ that work on } k \text{ and } l \\ 0 \quad \text{otherwise} \end{cases}$$

  - $D_{kl} = \sum_{i=1}^{n} C_{ik} C_{il}$, *i.e.,* the number of **paths of length 2** connecting projects k and l in the bipartite network
  - $\boldsymbol{D} = \boldsymbol{C}^{\boldsymbol{T}}\boldsymbol{C}$ and D_{kk} represents #(students) that work on project k

- **Next:** Use $\boldsymbol{B}$ and $\boldsymbol{D}$ to obtain different network projections

(3) Construct One-mode Projections

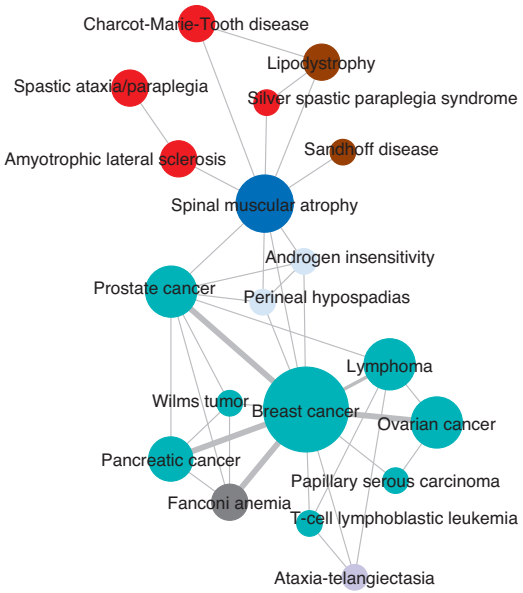
- Construct network projections by applying **a node similarity measure** to $\boldsymbol{B}$ and $\boldsymbol{D}$

- **Two node similarity measures:**
  - **Common neighbors:** #(shared neighbors of nodes)
    - **Student network:** i and j are linked if they work together in r **or more projects, *i.e.*,** if $B_{ij} \geq r$
    - **Project network:** k and l are linked if r **or more students** work on both projects, *i.e.,* if $D_{kl} \geq r$
  - **Jaccard index:**
    - Common neighbors with a penalization for each non-shared neighbor:
      - Ratio of shared neighbors in the complete set of neighbors for 2 nodes
    - **Student network:** i and j are linked if they work together in **at least p fraction of their projects, *i.e.*,** if $B_{ij}/(B_{ii} + B_{jj} - B_{ij}) \geq p$
    - **Project network:** k and l are linked if **at least p fraction of their students** work on both projects, *i.e.,* if $D_{kl}/(D_{kk} + D_{ll} - D_{kl}) \geq p$

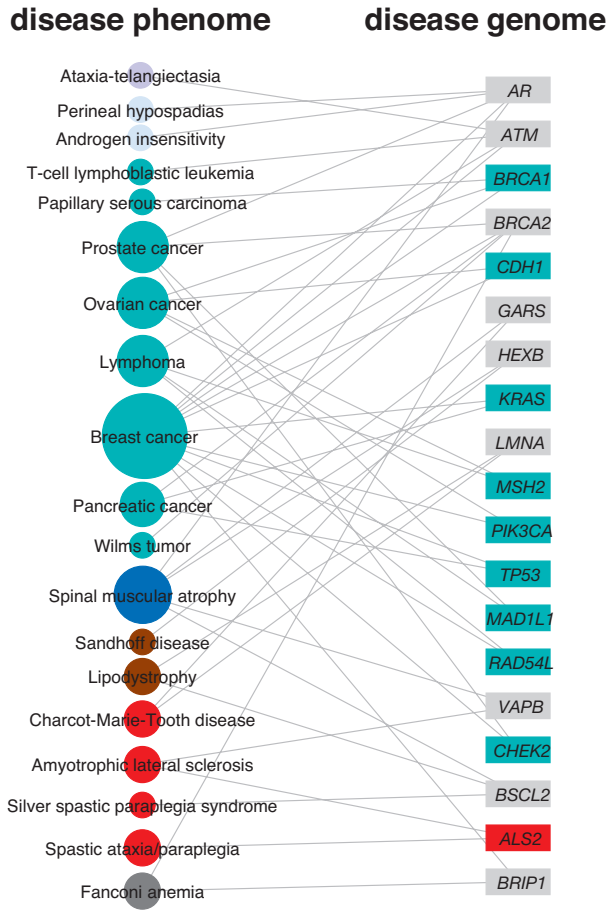
Example: The Human Disease Net

Homework 1

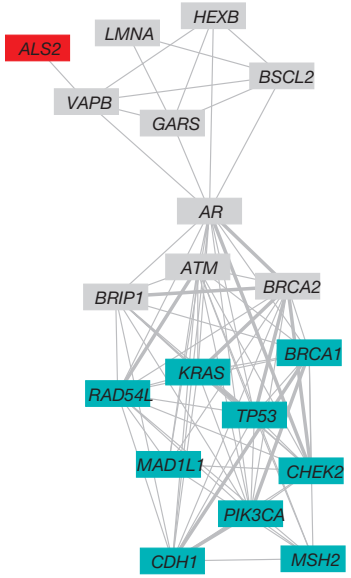
Human Disease Network (HDN)



DISEASOME



Disease Gene Network (DGN)

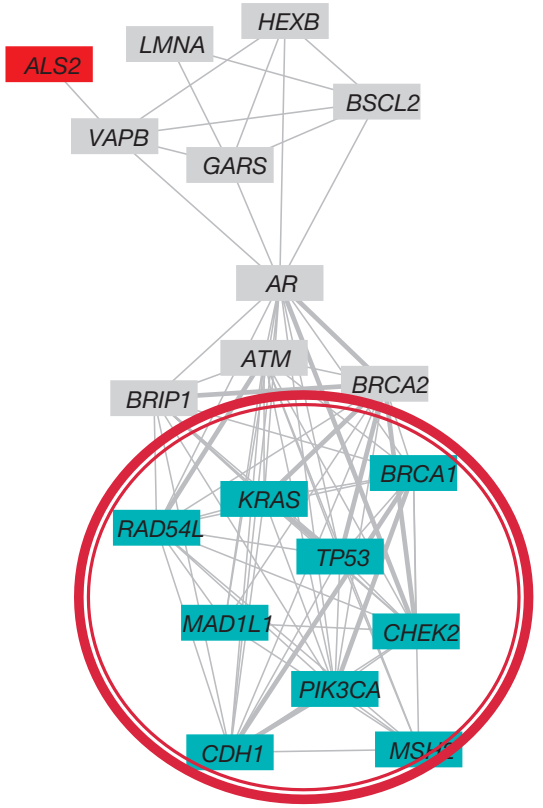


Kwang-Il Goh *et al.*, The human disease network. *PNAS,* 104:21, 2007.

Example: The Human Disease Net

- **Issue:** Folded gene network contains many **cliques:**
  - Why do cliques arise in the folded gene network?
    - **Homework 1**
- **Cliques make the network difficult to analyze:**
  - Computational complexity of many algorithms depends on the **size** and **number of large cliques**
- **Solution:** Use **graph contraction** to eliminate cliques

Disease Gene Network (DGN)



A clique of 9 gene nodes

Graph Contraction

- **Graph contraction:** Technique for computing properties of networks in parallel:
  - **Divide-and-conquer principle**
- **Idea:**
  - **Contract the graph** into a smaller graph, ideally a constant fraction smaller
  - Recurse on the smaller graph
  - Use the result from the recursion along with the initial graph to calculate the desired result

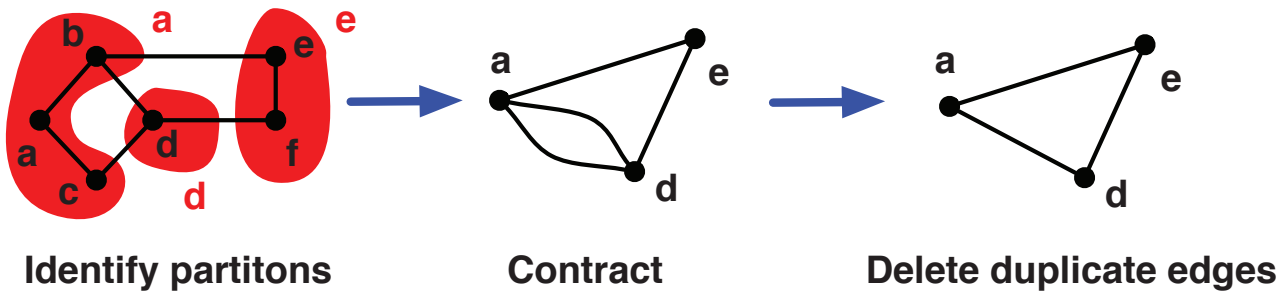
- **Next:** How to contract ("shrink") a graph?

Graph Contraction: Algorithm

- Start with the input graph G:
  1. Select a **node-partitioning** of G to guide the contraction:
     - Partitions are disjoint and they include all nodes in G
  2. Contract each partition into a single node, a **supernode**
  3. Drop edges internal to a partition
  4. Reroute cross edges to corresponding **supernodes**
  5. Set G to be the smaller graph; Repeat

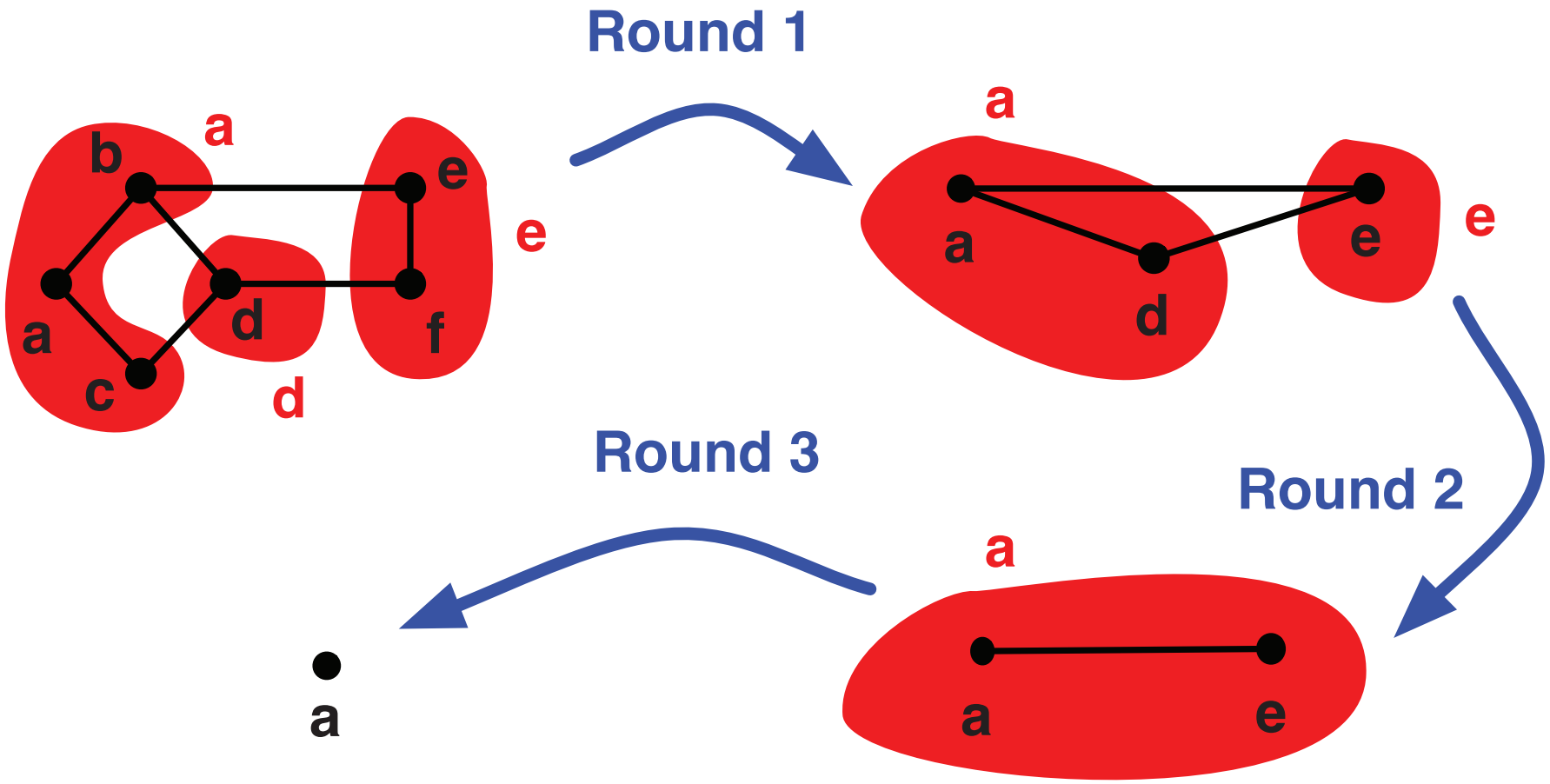
- Example: one round of graph contraction:

3 partitions: a, d, e



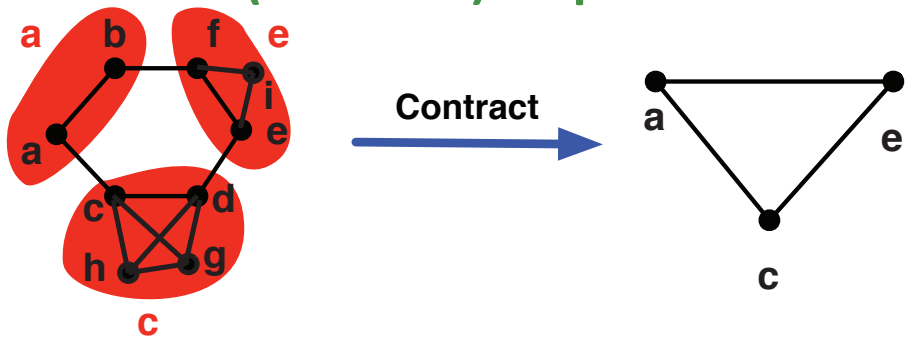
Graph Contraction: Example

Contracting a graph down to a single node in three rounds:



Different Types of Node-partitioning

- Partitions should be **disjoint** and **include all nodes** in G
- **Three types of node-partitioning:**
  - Each partition is a **(maximal) clique of nodes:**



  - Each partition is a **single node** or **two connected nodes:**

  - Each partition is a **star of nodes, etc.**

Plan for Today

1) Multimode Network Transformations:

- K-partite and bipartite graphs
- One-mode network projections/folding
- Graph contractions

2) K-Nearest Neighbor Graph Construction

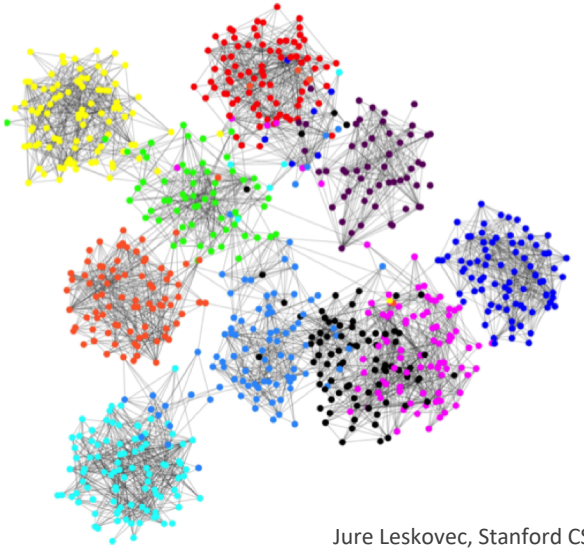
3) Network Deconvolution:

- Direct and and indirect effects in a network
- Inferring networks by network deconvolution

Efficient Construction of K-Nearest Neighbor Graph

K-Nearest Neighbor Graph

- **K-nearest neighbor graph (K-NNG)** for a set of objects V is a directed graph with vertex set V:
  - **Edges** from each $v \in V$ to its K most similar objects in V under a given similarity measure:
    - *e.g.,* Cosine similarity for text
    - *e.g.,* l_2 distance of CNN-derived features for images

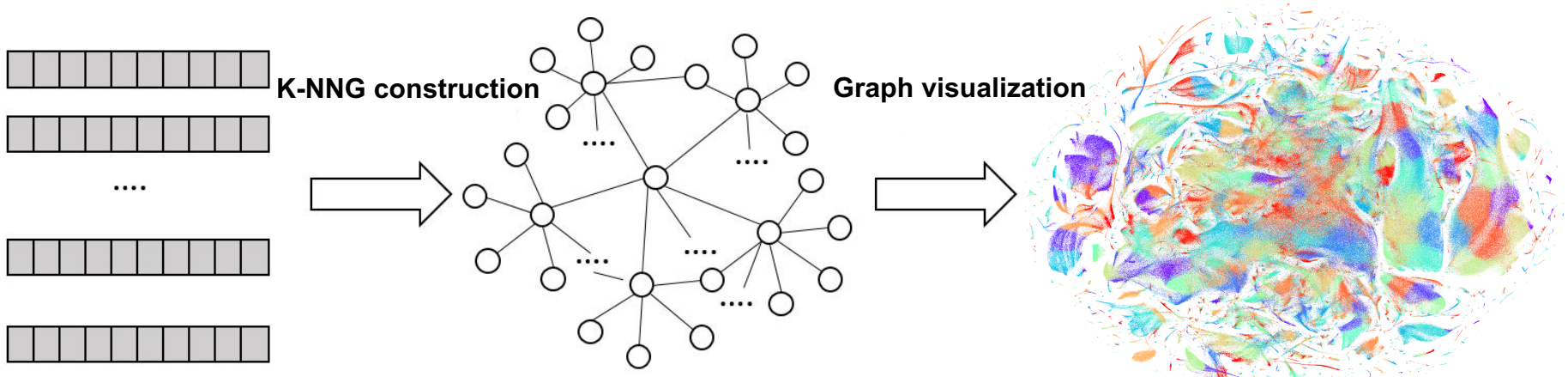


Why Constructing K-NNGs?

- K-NNG construction is an **important operation:**
  - **Recommender systems:** connect users with similar product rating patterns, then make recommendations based on the user's graph neighbors
  - **Document retrieval systems:** connect documents with similar content, quickly answer input queries
  - Other problems in **clustering**, **visualization**, **information retrieval**, **data mining**, **manifold learning**
- K-NNGs allow us to use network methods on datasets with no **explicit graph structure**

Example: K-NNG in Visualization

- **Problem:** Visualize large high-dim data in 2D space
- **Traditional approach:**
  - Compute similarities between objects
  - Project objects into a 2D space by preserving the similarities
  - **Does not scale** to millions of objects and hundreds of dimensions
- K-NNG can substantially reduce computational costs



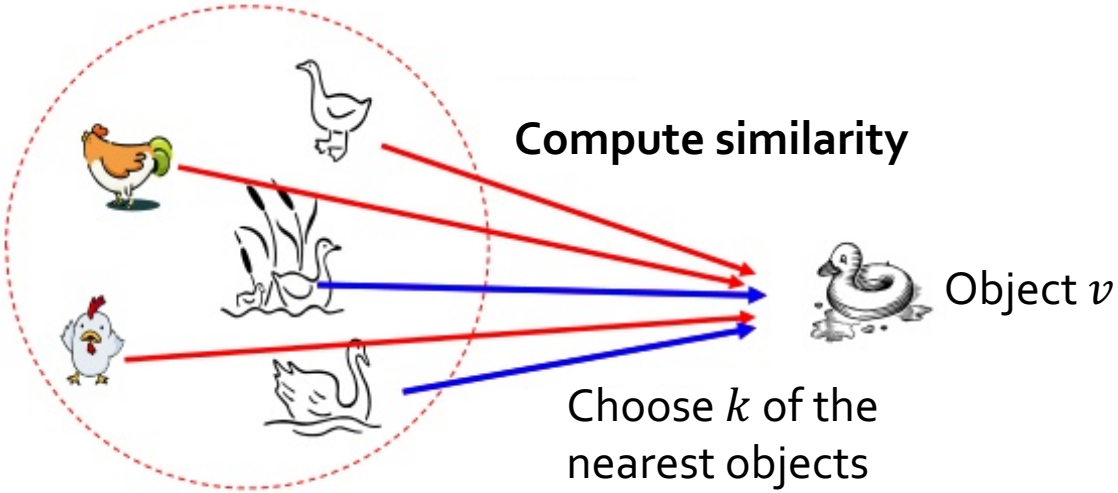
(a) High-dimensional feature vectors

(b) K-nearest neighbor graph (K-NNG)

(c) 2-dimensional layout
WikiDoc data (t-SNE)

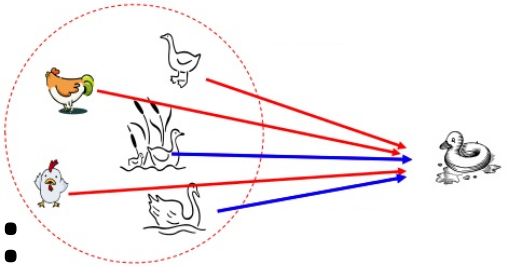
K-NNG: A Brute-force Approach

- Let's construct a K-NNG by **brute-force:**
  - Given n objects V and a distance metric $\sigma: V \times V \rightarrow [0, \infty)$
  - For each possible pair of (u, v), compute $\sigma(u, v)$
  - For each v, let $\boldsymbol{B_K(v)}$ be v's K-NN, *i.e.*, the K objects in V (other than v) most similar to v



K-NNG: A Brute-force Approach

- Computational cost of brute-force: $O(n^2)$



- **Issues with brute-force approach:**
  - **Not scalable:** Practical for only small datasets
  - **Not general:** Many custom heuristics designed to speed up computations:
    - Many heuristics are specific to a similarity measure
  - **Not efficient:** Compute all neighbors for every v
    - We only need k nearest neighbors for every v

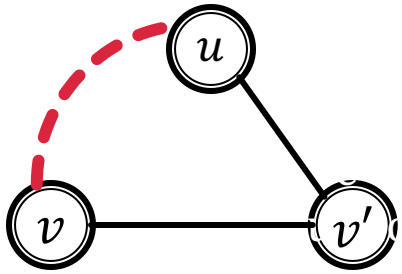
Today: NN-Descent Approach

- **Can we do better than brute-force?**
- **Yes, and we will learn about it today!**

- **NN-Descent** [Dong et al., WWW 2011]:
  - Efficient algorithm to **approximate K-NNG construction** with **arbitrary similarity measure**
- Other published methods (not covered today):
  - **Locality Sensitive Hashing (LSH):** A new hash function needs to be designed for a new similarity measure
  - **Recursive Lanczos bisection:** Recursively divide the dataset, so objects in different partitions are not compared
  - **K-NN search problem:** If K-NN problem is solved, K-NNG can be constructed by running a K-NN query for each $v \in V$

NN-Descent: Key Principle

- **Key principle:** A neighbor of a neighbor is also likely to be a neighbor



- **Use this principle in a NN-Descent method:**
  - Start with an approximation of the K-NNG, B
  - Improve B by **exploring each point's neighbors' neighbors** as defined by the current approximation
  - Stop when no improvement can be made

NN-Descent: Notation

- **Let:**
  - V be a metric space with distance metric $d: V \times V \rightarrow [0, \infty)$, $\sigma = -d$ is the similarity measure
  - $B_K(v)$ be **v's K-NN**
  - $R_K(v) = \{u \in V; v \in B_K(u)\}$ be **v's reverse K-NN**
  - $B[v]$ be **current approximation** of $B_K(v)$
  - $B'[v] = \cup_{v' \in B[v]} B[v']$ be **neighbors of v's neighbors**
  - For any $r > 0$, let **r-ball around v** be:

$$B_r(v) = \{u \in V; d(u, v) \leq r\}$$

(1) NN-Descent: Overview

Details

- **Def:** Metric space V is **growth-restricted** if there exists a constant c, such that:

$$|B_{2r}(v)| \leq c|B_r(v)|, \quad \forall v \in V$$

- The smallest such c is **growing constant** of V

- **Approach:**
  - Start with an approximation of the K-NNG, B
  - Use the **growing constant of V** to show that B can be improved **by comparing each object v against its current neighbors' neighbors $B'[v]$**

- **Next:** Use the **growing-constant argument** on B

(2) NN-Descent: Proof Outline

Details

- **Two assumptions:**
  - Let c be the growing constant of V and let $K = c^3$
  - Have an approximate K-NNG B **that is reasonably good:**
    - For a fixed radius r, for all v, $B[v]$ contains K neighbors that are uniformly distributed in $B_r(v)$
- **Lemma:** $B'[v]$ is likely to contain K nearest neighbors in $B_{r/2}(v)$
- **Corollary:** We expect to **halve the maximal distance** to the set of approximate K nearest neighbors by exploring $B'[v]$ for every v
- **Next:** Let's prove the lemma

(3) NN-Descent: Proof

Details

- **Lemma:** $B'[v]$ is likely to contain K nearest neighbors in $B_{r/2}(v)$

Proof:

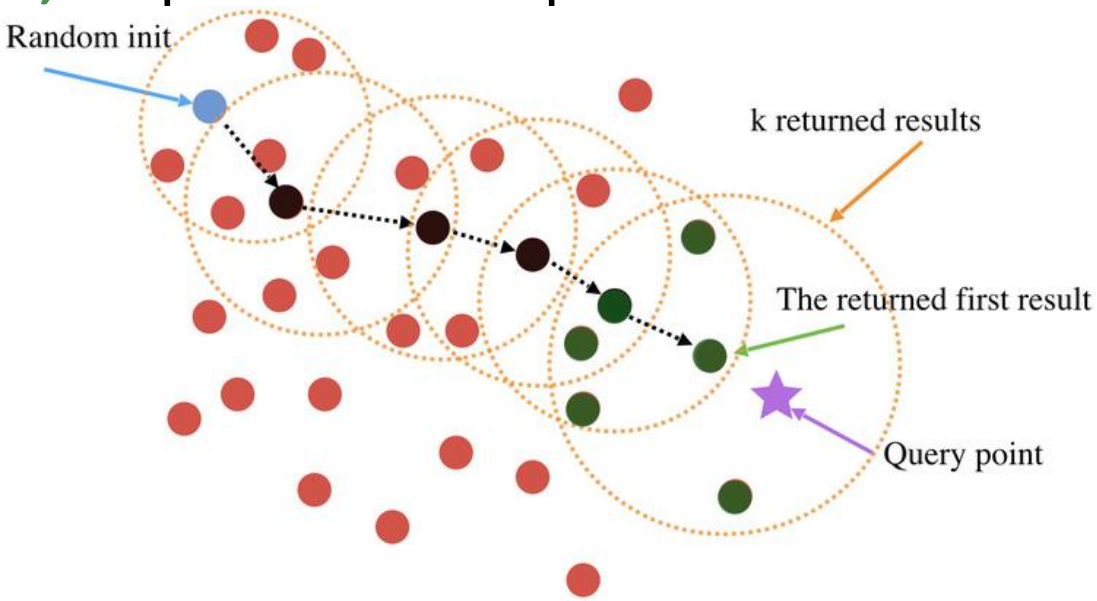
- For any $u \in B_{r/2}(v)$ to be found in $B'[v]$, we need to have at least one v' such that:
$$v' \in B[v] \wedge u \in B[v']$$
- Any $v' \in B_{r/2}(v)$ is likely to satisfy this requirement, as we have:
  1. v' is also in $B_r(v)$, so $\Pr\{v' \in B[v]\} \geq K/|B_r(v)|$
  2. $d(u,v') \leq d(u,v) + d(v,v') \leq r$, so $\Pr\{u \in B[v']\} \geq K/|B_r(v')|$
  3. $|B_r(v)| \leq c|B_{r/2}(v)|$, and $|B_r(v')| \leq c|B_{r/2}(v')| \leq c|B_r(v)| \leq c^2|B_{r/2}(v)|$
- Combining 1-3 and assuming independence, we get:
$$\Pr\{v' \in B[v] \wedge u \in B[v']\} \geq K/\left|B_{r/2}(v)\right|^2$$
- In total, we have $|B_{r/2}(v)|$ candidates for such v', so that: $\Pr\{u \in B'[v]\} \geq 1 - (1 - K/\left|B_{r/2}(v)\right|^2)^{|B_{r/2}(v)|} \approx K/|B_{r/2}(v)|$

QED

NN-Descent: Recap

Details

- **Lemma** suggests the following algorithm:
  - Pick a large enough K (depending on **growing constant c**)
  - Start from a random K-NNG approximation
  - For each v, find K nearest objects by exploring v's neighbors' neighbors, B'
  - **Repeat;** stop when no improvement can be made



NN-Descent: Algorithm

Details

Algorithm 1: NNDescent

Data: dataset V, similarity oracle σ, K
Result: K-NN list B
begin
  $B[v] \longleftarrow \text{Sample}(V, K) \times \{\infty\}, \quad \forall v \in V$
  loop
    $R \longleftarrow \text{Reverse}(B)$
    $\overline{B}[v] \longleftarrow B[v] \cup R[v], \quad \forall v \in V;$
    $c \longleftarrow 0 \quad$ //update counter
    for $v \in V$ **do**
      for $u_1 \in \overline{B}[v], u_2 \in \overline{B}[u_1]$ **do**
        $l \longleftarrow \sigma(v, u_2)$
        $c \longleftarrow c + \text{UpdateNN}(B[v], \langle u_2, l \rangle)$
    return B **if** $c = 0$

function Sample(S, n)
return Sample n items from set S

function Reverse(B)
begin
  $R[v] \longleftarrow \{u \mid \langle v, \cdots \rangle \in B[u]\} \quad \forall v \in V$
  return R

function UpdateNN(H, $\langle u, l, \ldots \rangle$)
Update K-NN heap H; return 1 if changed, or 0 if not.

A. Start by picking a random approximation of K-NN for each object

B. Improve the approximation by comparing each object against its current neighbors' neighbors, including K-NN and reverse K-NN

C. Stop when no improvement can be made

Experimental Setup: Data

- Datasets:
  - **Corel:** Each **image** is segmented into 14 regions, a feature is extracted from each region
  - **Audio:** Each **sentence** is described by 192 features
  - **Shape:** Each **shape** is described by 544-dim feature vector
  - **DBLP:** Each **record** includes authors' names and pub. title
  - **Flickr:** Each **image** is segmented into regions, a pixel-based feature is extracted from each region
- Similarity measures: L1, L2, Cosine, Jaccard, EMD

Dataset	# Objects	Dimension	Similarity Measures
Corel	662,317	14	l_1, l_2
Audio	54,387	192	l_1, l_2
Shape	28,775	544	l_1, l_2
DBLP	857,820	N/A	Cosine, Jaccard
Flickr	100,000	N/A	EMD

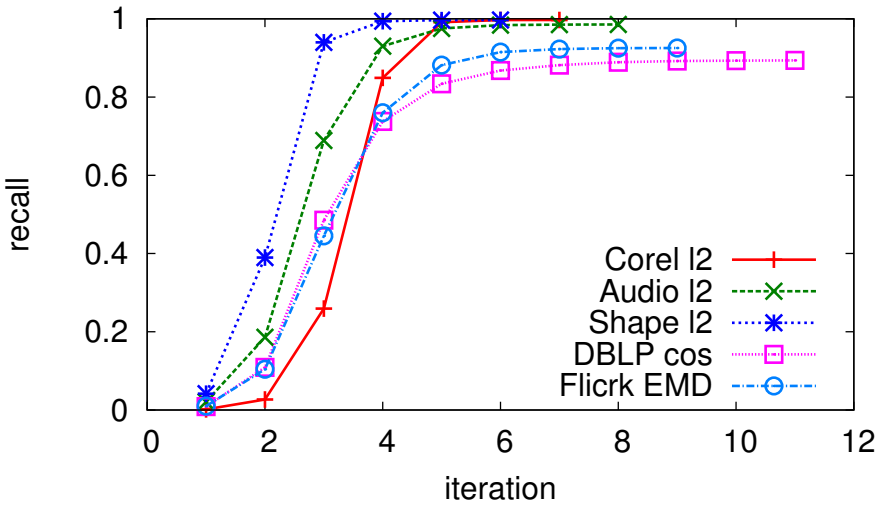
(EMD: earth mover's distance)

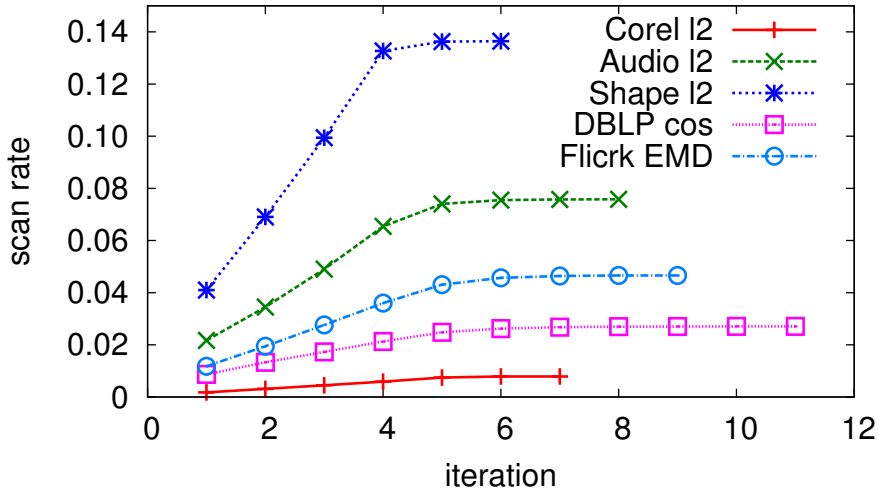
Experimental Setup: Measures

- Use recall as an **accuracy measure:**
  - **Ground-truth:** true K-NNs obtained by scanning the datasets in brute force
  - **Recall of one object** is the number of its true K-NN members found divided by K
  - **Recall of an approximate K-NNG** is the average recall of all objects
- Use #(sim. evaluations) as a **measure of computational cost:**

$$\text{scan rate} = \frac{\#(\text{similarity evaluations})}{n(n-1)/2}$$

(1) Exp.: Overall Performance



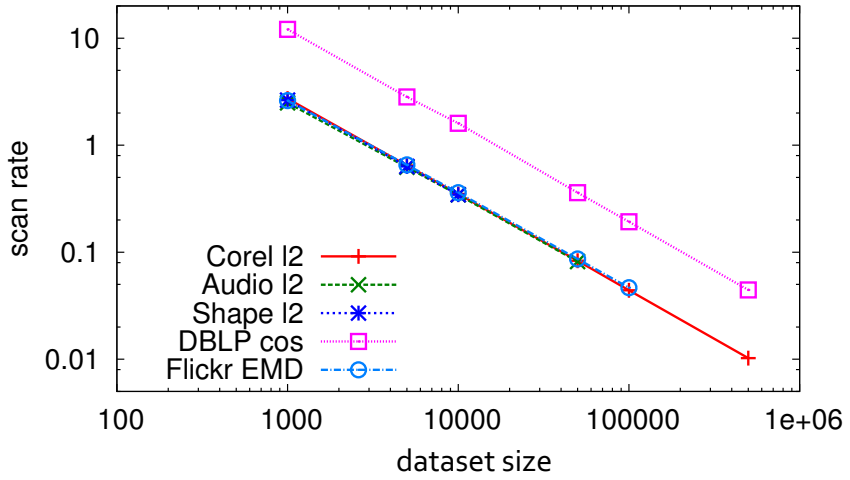


- Similar performance trends on different datasets
- Fast convergence across all datasets:
  - Curves are close to their final recall after 5 iterations
  - All curves converge within 12 iterations

(2) Exp.: Performance as Data Scales

Size	Corel l_2	Audio l_2	Shape l_2	DBLP cos	Flickr EMD
1K	1.000	0.999	1.000	0.959	0.999
5K	1.000	0.996	0.992	0.970	0.991
10K	1.000	0.993	0.998	0.970	0.983
50K	0.999	0.988	-	0.951	0.953
100K	0.999	-	-	0.940	0.925
500K	0.997	-	-	0.907	-

(recall values)



- Run experiments on samples of the full datasets and observe changes in recall and scan rate as sample size grows
- **Results:**
  - As dataset grows, there is only a minor decline in recall
  - All curves form parallel straight lines in the scan rate vs. dataset size:
    - NN-descent has a polynomial time complexity
    - Fit the scan rate curves to obtain empirical complexity of NN-Descent:
      - $O(n^{1.14}) \ll O(n^2)$ (=brute-force)

Plan for Today

1) Multimode Network Transformations:

- K-partite and bipartite graphs
- One-mode network projections/folding
- Graph contractions

2) K-Nearest Neighbor Graph Construction

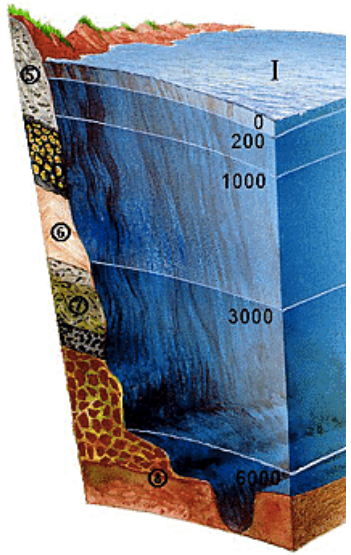
3) Network Deconvolution:

- Direct and and indirect effects in a network
- Inferring networks by network deconvolution

Network Deconvolution and Inference

Motivation

- Networks represent **dependencies among objects:**
  - Co-authorships between scientists
  - Friendships between people
  - Who-eats-whom in food webs
  - Bonds between molecular residues
  - Regulatory relationships between genes



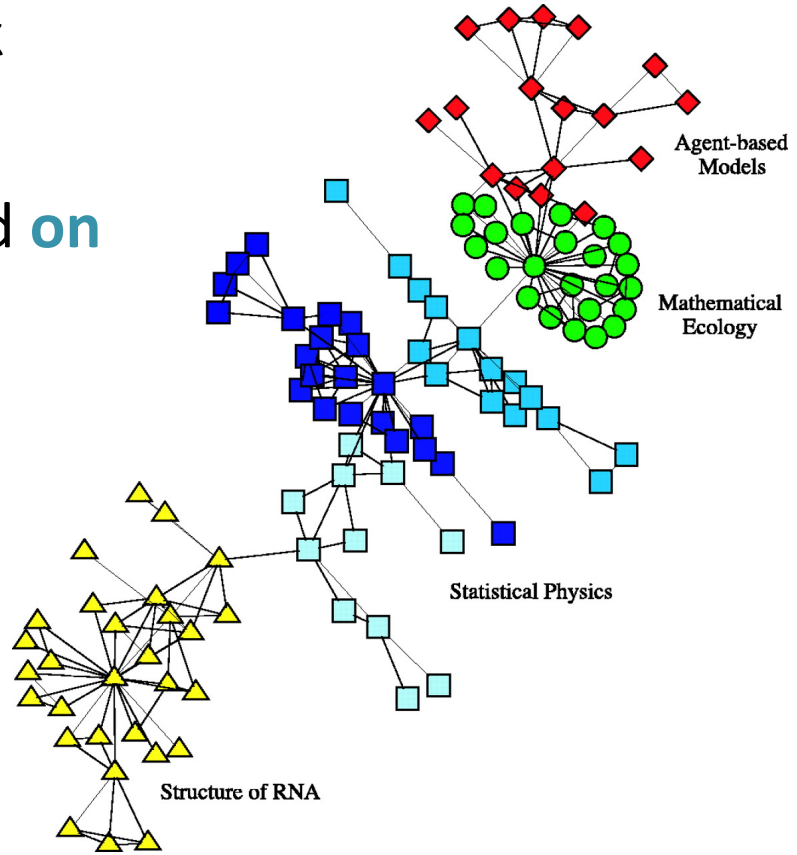
- **Indirect dependencies** occur because of **transitive effects of correlation**

- **Problem: How to separate direct dependencies from indirect ones?**

Application: Co-authorship Net

- **Goal:** Distinguish **strong** and **weak** collaborations between scientists

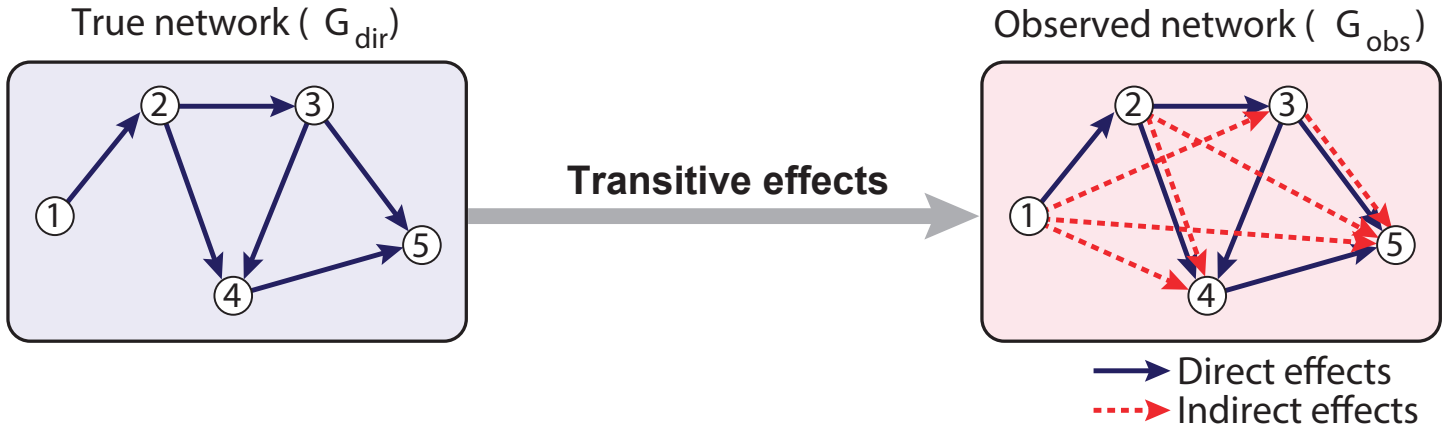
- Collaboration tie strengths depend **on publication details**, such as:
  - #(papers) each pair of scientists has collaborated on
  - #(co-authors) on each of the papers



- Strength of ties are important for:
  - **Recommending friends** and colleagues
  - Recognizing **conflicts of interest**
  - Evaluating authors' **contribution to teams**

Observed Network

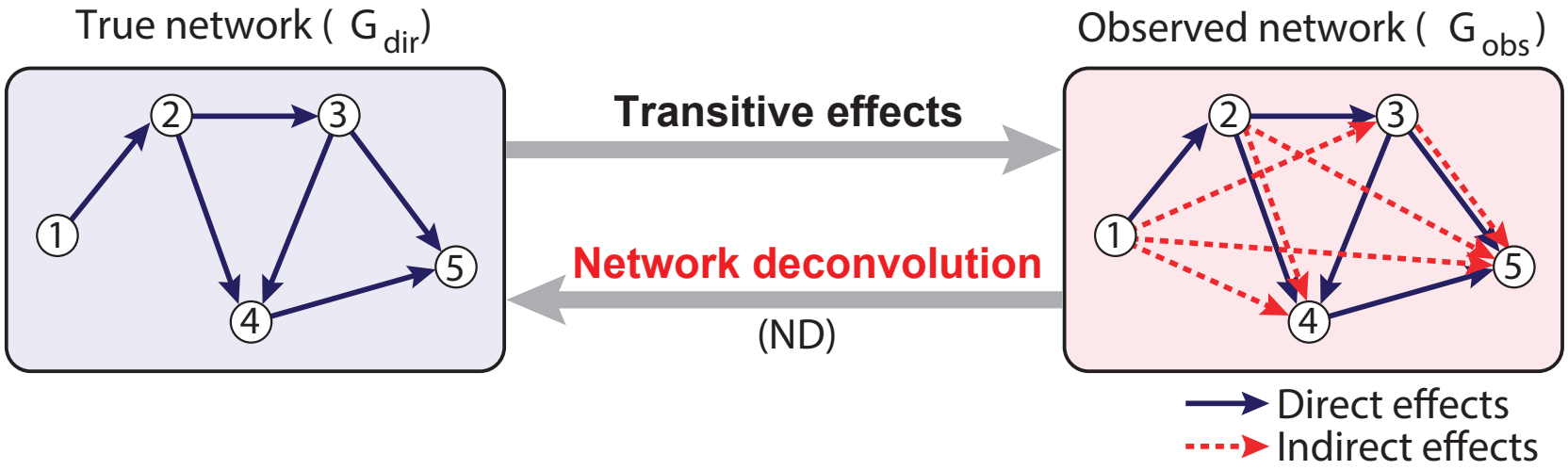
- **Observed network:** Combined direct and indirect effects:



- **Indirect edges** might be due to higher-order interactions (*e.g.*, 1→4)
- Each edge might contain **both direct and indirect components** (*e.g.*, 2→4)

Network Deconvolution

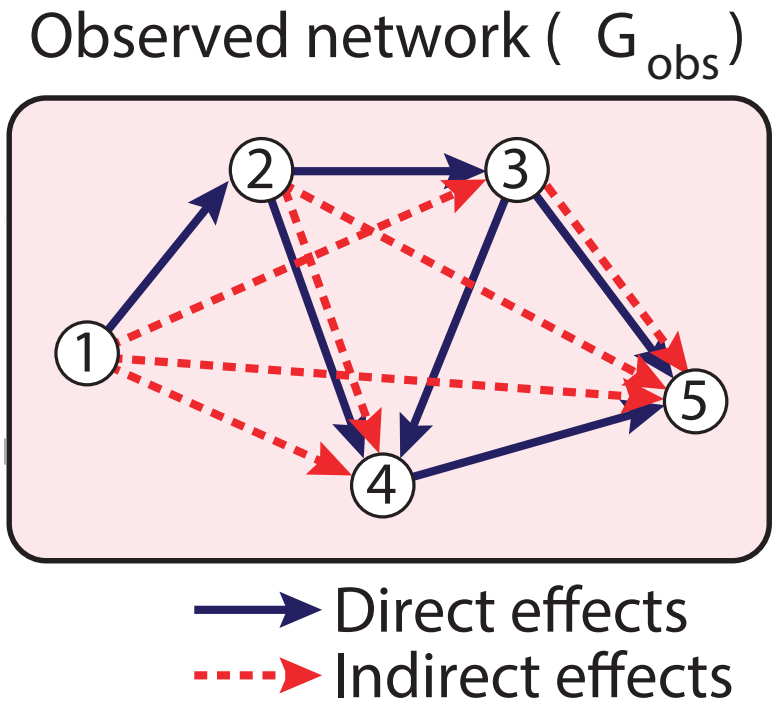
- **Goal:** Reverse the effect of transitive information flow across all indirect paths:
  - Recover **true direct network (blue edges, $G_{\mathbf{dir}}$)** based on **observed network (combined blue and red edges, $G_{\mathbf{obs}}$)**



Feizi *et al.*, *Nature Biotechnology*, 31:8, 2013.

Network Deconvolution: Challenge

- Direct edges in a network can lead to indirect relationships:
  - **Transitive information flow**
- **Indirect effects** can be of **length**:
  - **2** (*e.g.*, 1→2→3)
  - **3** (*e.g.*, 1→2→3→5)
  - **higher-order**
- Indirect effects can combine:
  - **Both direct and indirect** effects (*e.g.*, 2→4)
  - Multiple indirect effects along **varying paths** (*e.g.*, 2→3→5, 2→4→5)



Network Deconvolution: Formally

- **Transitive effects in $G_{\mathbf{obs}}$** can be expressed as an infinite sum of G_{dir} and all indirect effects:

$$G_{\mathrm{obs}} = \mathrm{G}_{\mathrm{dir}} + \boldsymbol{G_{\mathbf{indir}}}$$

- Indirect effects can be of **increasing lengths:**

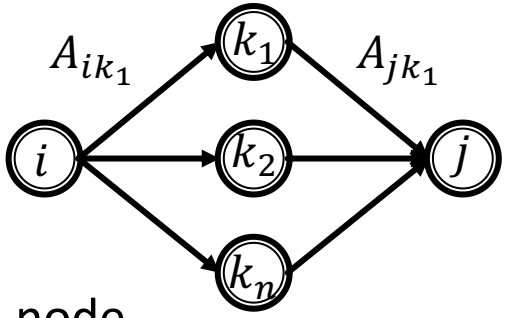
$$G_{\mathrm{indir}} = \mathrm{G}^2_{\mathrm{dir}} + \mathrm{G}^3_{\mathrm{dir}} + G^4_{\mathrm{dir}} + \cdots$$

- **2nd order effects:** $G^2_{\mathrm{dir}} = \mathrm{A}^2_{\mathrm{dir}}$
  - **The number of edges in $G_{\mathbf{obs}}$ of indirect paths of length 2**
- **3rd order effects:** $G^3_{\mathrm{dir}} = \mathrm{A}^3_{\mathrm{dir}}$
  - **The number of edges in $G_{\mathbf{obs}}$ of indirect paths of length 3**

Powers of Adjacency Matrices

- Let's raise adjacency matrix A_{dir} **to the second power:**
  - The (i, j)-th entry of A^2_{dir} is:

$$A^2_{\text{dir}}(i,j) = \sum_{k=1}^{n} A_{\text{dir}}(i,k)\, A_{\text{dir}}(k,j)$$



    - This sum is only greater than zero if there exists a node k for which $A_{\text{dir}}(i,k)$ and $A_{\text{dir}}(k,j)$ are both nonzero:
      - There exists a node ***k* that is connected to both nodes *i* and *j***
      - The sum counts the number of neighbors that nodes i and j share
      - The sum counts **the paths of length 2 between nodes *i* and *j***

- This reasoning is valid for higher powers of A_{dir}:
  - $A^3_{\text{dir}}(i,j)$ counts the **paths of length 3** between i and j
  - $A^4_{\text{dir}}(i,j)$ counts the **paths of length 4** between i and j

Network Deconvolution: Formally

- **Idea:** Model indirect flow as **power series** of direct flow:

$$G_{\text{obs}} = \mathrm{G}_{\text{dir}} + \mathrm{G}_{\text{dir}}^2 + \mathrm{G}_{\text{dir}}^3 + G_{\text{dir}}^4 + \cdots$$

Converges with correct scaling

Indirect effects

Transitive closure of G_{dir}

- **Note:** Linear scaling of G_{obs} so that max absolute eigenvalue of $\mathrm{G}_{\text{dir}} < 1$:
  - Indirect effects decay exponentially with path length
  - Infinite series converges

Network Deconvolution: Formally

- **Transitive closure of G_{dir}** can be expressed as an infinite sum of:
  - True direct network, G_{dir}
  - All indirect effects along paths of increasing lengths, G_{dir}^2, G_{dir}^3, G_{dir}^4 , …

- **Idea:** Can be written in a closed form as an infinite-series sum using **Taylor series expansions**:

$$G_{\text{obs}} = \mathrm{G}_{\text{dir}} + \mathrm{G}_{\text{dir}}^2 + \mathrm{G}_{\text{dir}}^3 + G_{\text{dir}}^4 + \cdots =$$
$$G_{\text{dir}}\left(\mathrm{I} + \mathrm{G}_{\text{dir}} + G_{\text{dir}}^2 + \mathrm{G}_{\text{dir}}^3 + \cdots\right) = G_{\text{dir}}(\mathrm{I} - \mathrm{G}_{\text{dir}})^{-1}$$

Note: Let X be any square matrix with max absolute eigenvalue < 1. Then the following series converges: $\mathrm{I} + X + X^2 + \mathrm{X}^3 + \cdots$
The series converges to: $\sum_{k=0}^{\infty} X^k = (1 - X)^{-1}$

Network Deconvolution: Solution

- Using **Taylor series expansions** we get a **closed-form expression for $G_{\mathbf{obs}}$**:

$$G_{\text{obs}} = G_{\text{dir}}(\text{I} - \text{G}_{\text{dir}})^{-1}$$

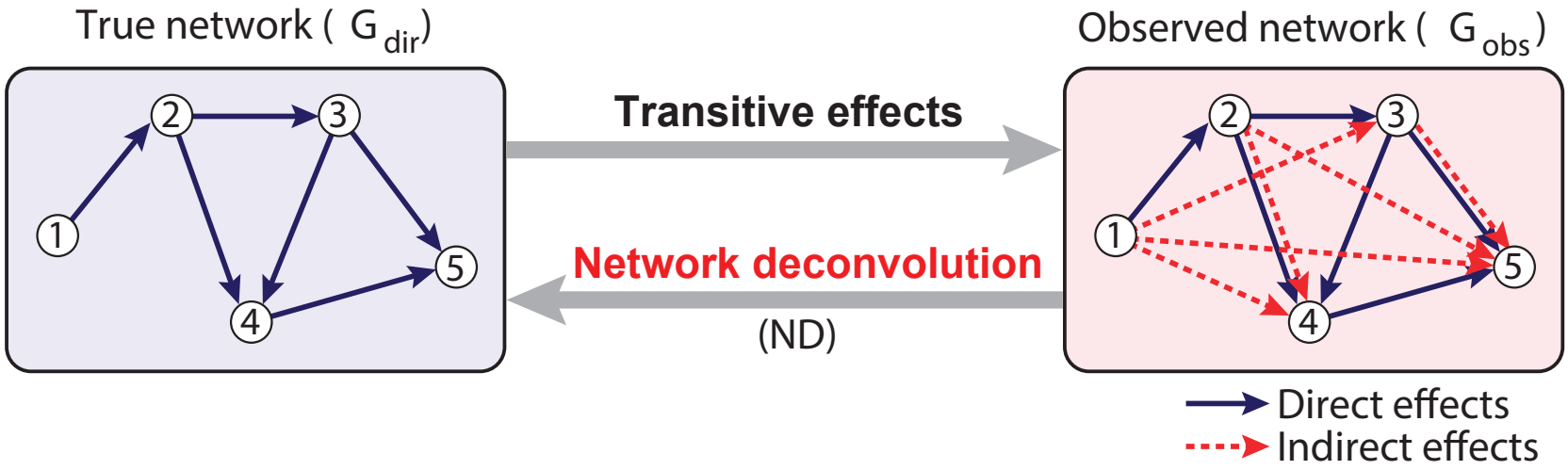
- **In network deconvolution:**
  - **Observed network $G_{\mathbf{obs}}$** is known
  - **True direct network $G_{\mathbf{dir}}$** needs to be **recovered**

- Finally, we get **a closed-form solution for $G_{\mathbf{dir}}$**:

$$G_{\text{dir}} = G_{\text{obs}}(\text{I} + \text{G}_{\text{obs}})^{-1}$$

Network Deconvolution: Recap

- Use **closed-form expression for G_{obs}** to **recover** true direct network G_{dir}



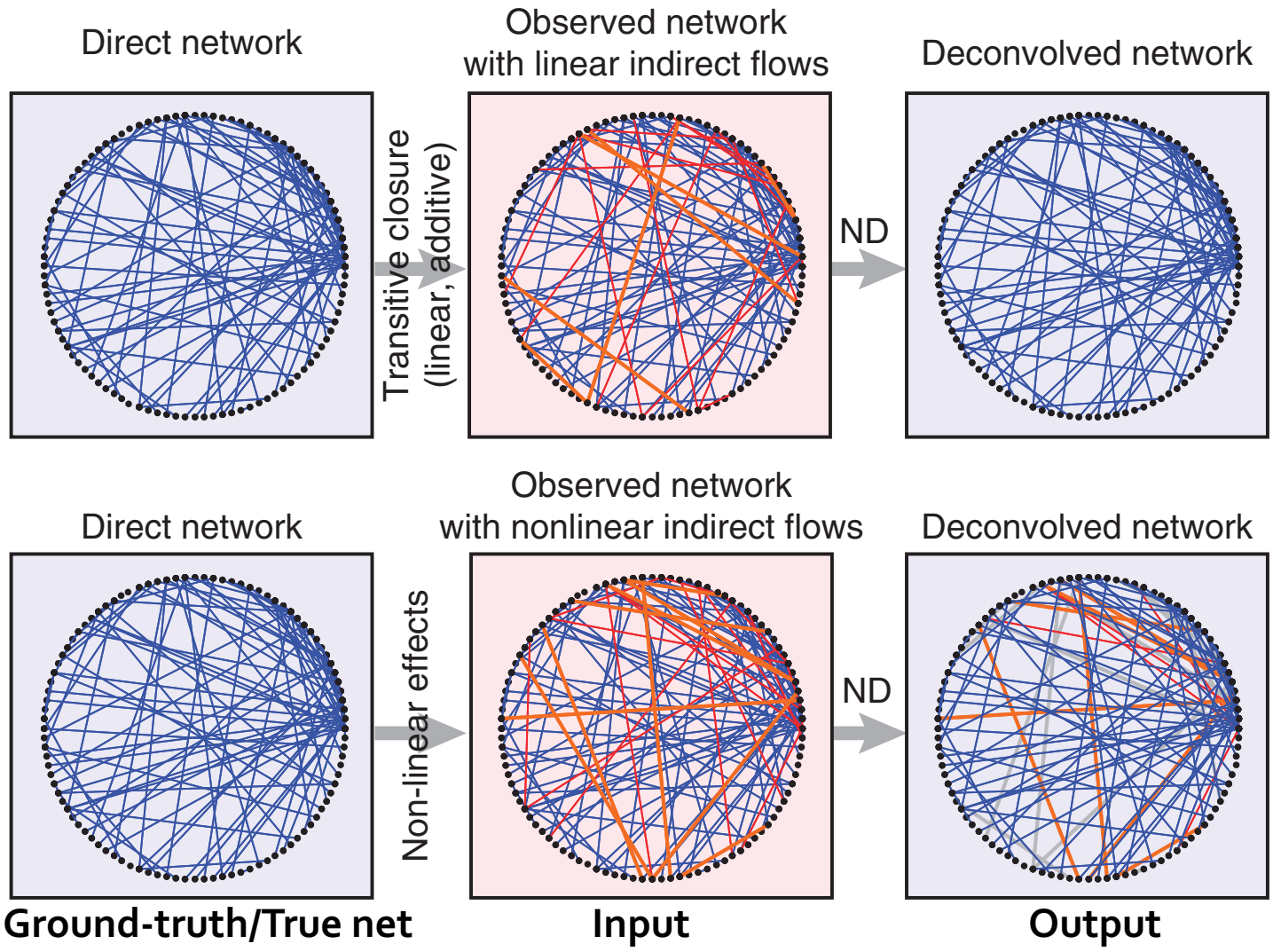
Transitive closure: $G_{obs} = G_{dir} + \overbrace{G_{dir}^2 + G_{dir}^3 + \ldots}^{\text{Indirect effects}} = \overbrace{G_{dir}(I - G_{dir})^{-1}}^{\text{Series closed form}}$

Network deconvolution: $G_{dir} = G_{obs}(I + G_{obs})^{-1}$

How to compute $G_{\text{obs}}(\text{I} + \text{G}_{\text{obs}})^{-1}$

- The solution for G_{dir} is: $G_{\text{dir}} = G_{\text{obs}}(I + G_{\text{obs}})^{-1}$
- How to **efficiently** calculate G_{dir}:
  - Without calculating matrix inverse $(I + G_{\text{obs}})^{-1}$
- **Approach:**
  - Use the **eigen-decomposition principle:**
    1. Express G_{obs} by decomposition into eigenvectors U and eigenvalues Σ_{obs}: $G_{\text{obs}} = U\Sigma_{\text{obs}}\text{U}^{-1}$
    2. Express **each eigenvalue $\boldsymbol{\lambda_i^{\text{dir}}}$** as a nonlinear function of a single corresponding eigenvalue λ_i^{obs}:
    $$\lambda_i^{\text{dir}} = \lambda_i^{\text{obs}}\left(1 + \lambda_i^{\text{obs}}\right)^{-1}$$
    3. Form a diagonal matrix Σ_{dir} such that $\Sigma_{\text{dir}}(i,i) = \lambda_i^{\text{dir}}$
    4. **Recover true direct network as: $\boldsymbol{G_{\text{dir}} = \text{U}\Sigma_{\text{dir}}\text{U}^{-1}}$**

Network Deconvolution: Overview



Length $n > 2$ indirect interactions (false positives)

True interactions removed by ND (false negatives)

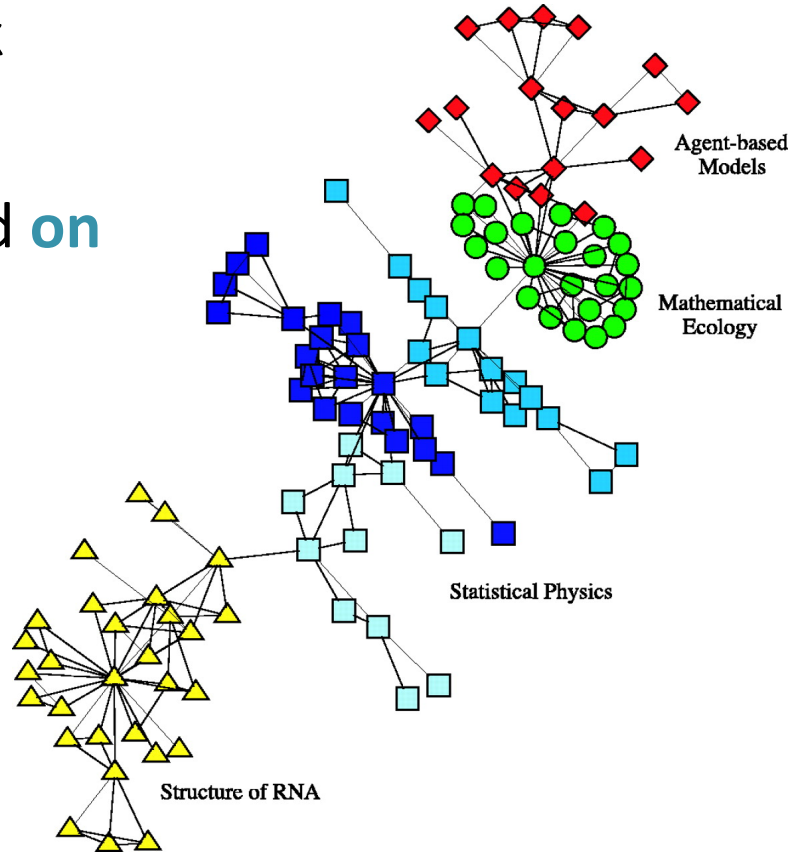
Direct interactions, correctly recovered (true positives)

Length-2 indirect interactions (false positives)

Application: Co-authorship Net

- **Goal:** Distinguish **strong** and **weak** collaborations between scientists

- Collaboration tie strengths depend **on publication details**, such as:
  - #(papers) each pair of scientists has collaborated on
  - #(co-authors) on each of the papers

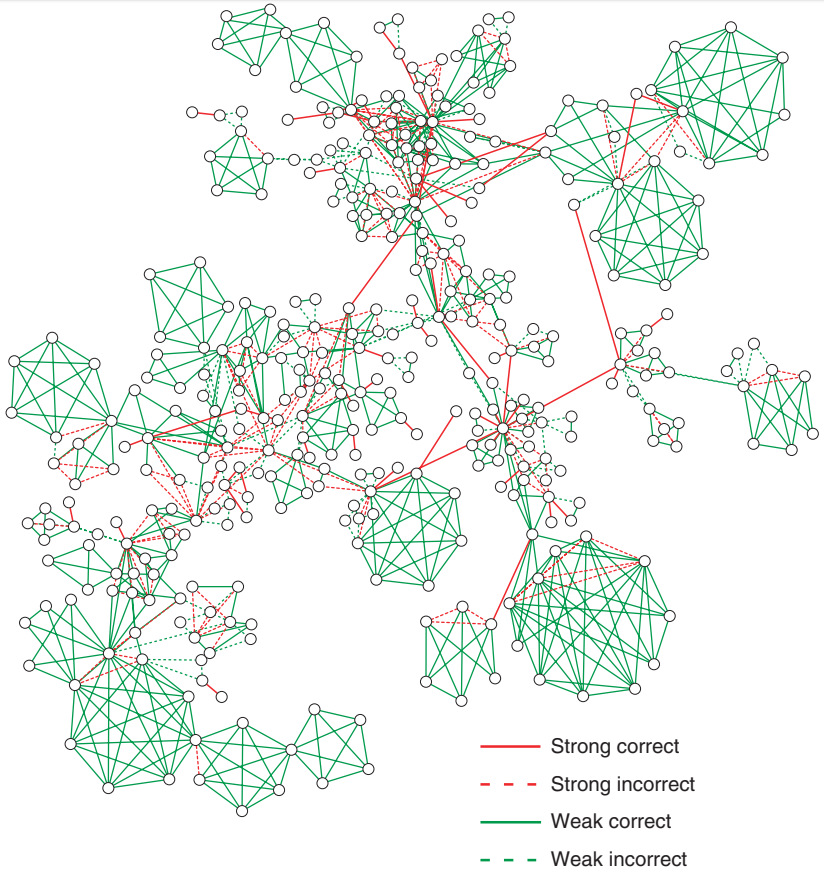


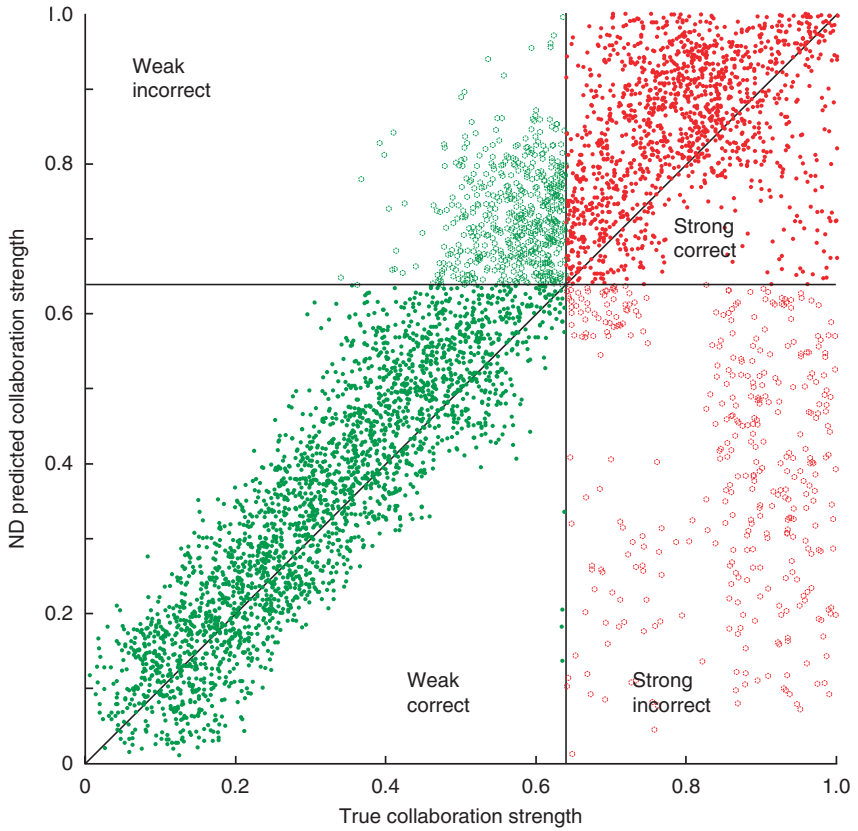
- Strength of ties are important for:
  - **Recommending friends** and colleagues
  - Recognizing **conflicts of interest**
  - Evaluating authors' **contribution to teams**

Application: Co-authorship Net

- **Data:** Unweighted network of **scientists** working in the field of network science:
  - Two authors are linked if they co-authored at least one paper
- **Setup:** Apply ND on the co-authorship network:
  - ND returns a weighted network whose:
    - Transitive closure most closely captures the input network
    - Weights represent the inferred strength of direct interactions
  - **Output:** Rank co-authorship edges by **the ND-assigned weights**
- **Ground-truth data:**
  - **True collaboration strengths** are computed by summing the number of co-authored papers and down-weighting each paper by the number of additional co-authors
  - Compute **correlation** between **ND-assigned weights** and **true collaboration strengths**

Co-authorship Network: Results



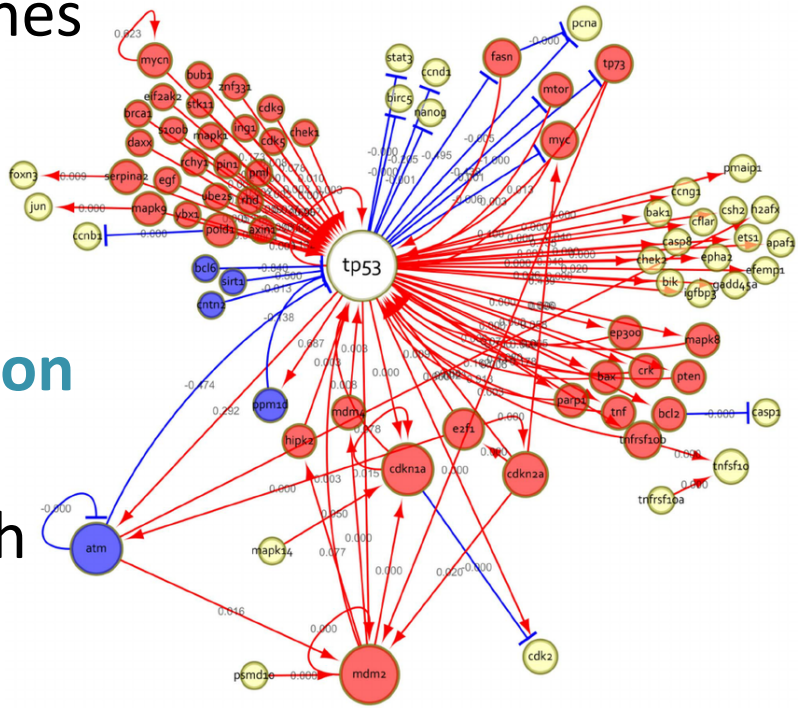


- Agreement between the rank obtained by the true collaboration strength and the rank provided by the ND weight, $R^2 = 0.76$
- **Conclusion:** ND predict collaboration tie strengths **solely by using network topology** (*i.e.*, not using other publication details)

Application: Gene Network Inference

- **Goal:** Infer a **gene regulatory network** from gene feature vectors describing gene activity:
  - **Nodes** represent genes
  - **Edges** represent regulatory relationships between regulators and their target genes

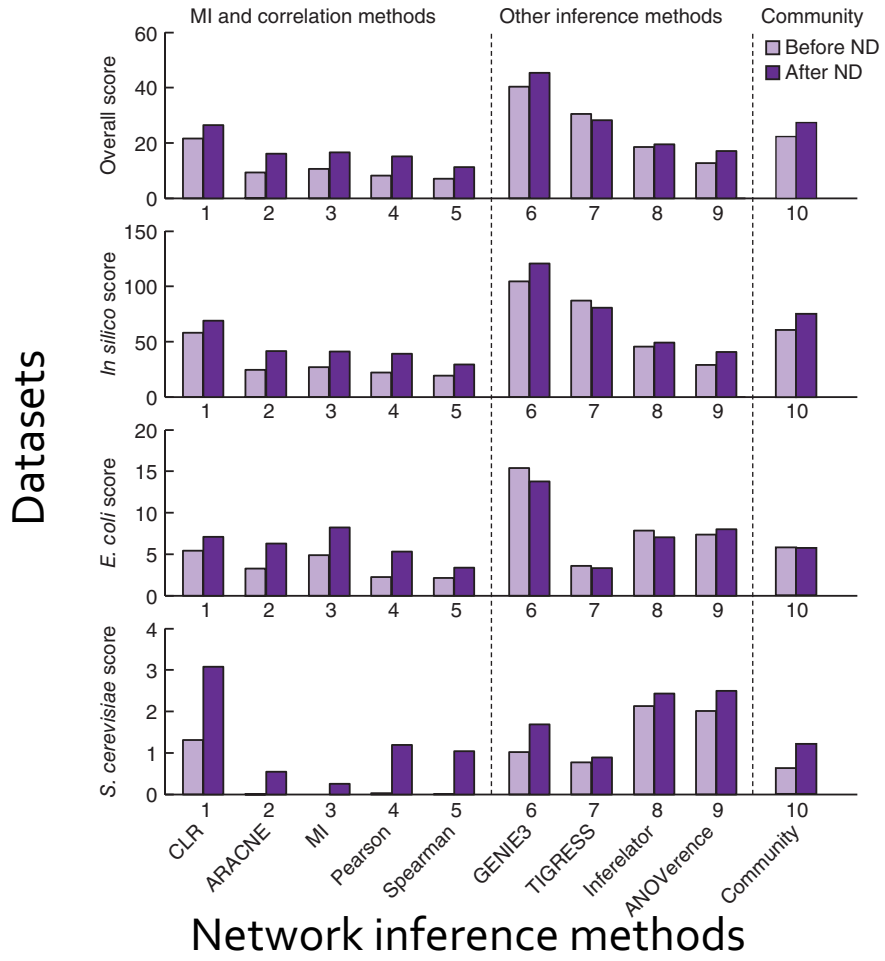
- Well-studied **problem in bioinformatics:**
  - A dataset is a **gene-by-condition expression matrix**
  - Expression matrix is **noisy** with many **indirect** measurements

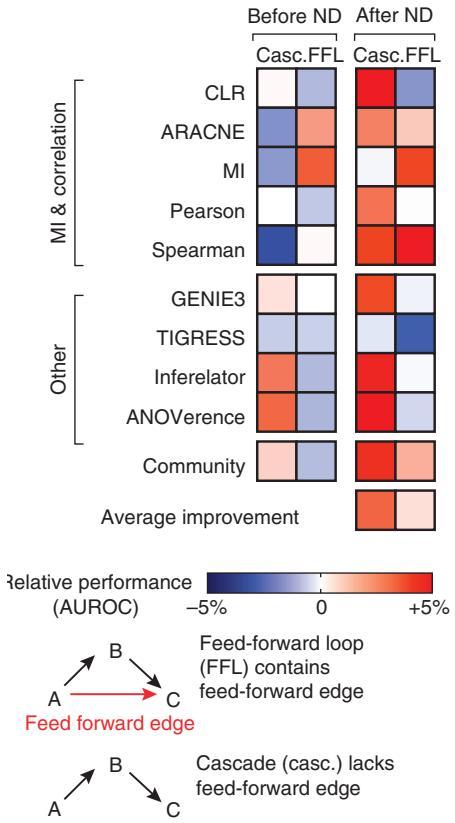


Application: Gene Network Inference

- **3 datasets:** Gene expression datasets from: bacterium *E. coli,* yeast *S. cerevisiae,* and a simulated env (*in silico)*
- **Setup:** Use ND to improve network inference methods by eliminating indirect edges in the inferred networks:
  1. Infer a gene regulatory network using a particular network inference method
  2. Apply ND to the inferred network to deconvolve the network
  3. Evaluate deconvolved network against ground-truth data
- **Ground-truth data:**
  - True positive regulatory relationships (*i.e.,* edges) are defined as a set of interactions experimentally validation in a laboratory

Gene Network Inference: Results





Relative performance of inference methods for cascades **(casc.)** and feed-forward loops **(FFL)** before and after network deconvolution

ND improves the performance of top-performing network inference methods

Network Deconvolution: Recap

- General approach to **identify direct dependencies between objects in a network:**
  - Remove **spurious edges** that are due to indirect effects
  - Decrease **over-estimated edge weights**
  - **Rescale edge weights** so that they correspond to direct dependencies between objects
- Other published methods (not covered today):
  - Partial correlations and random matrix theory
  - Graphical models, *e.g.*, Graphical lasso, Bayesian nets, Markov random fields
  - Causal inference models

Plan for Today

1) Multimode Network Transformations:

- K-partite and bipartite graphs
- One-mode network projections/folding
- Graph contractions

2) K-Nearest Neighbor Graph Construction

3) Network Deconvolution:

- Direct and and indirect effects in a network
- Inferring networks by network deconvolution