

Network Science : Lecture X

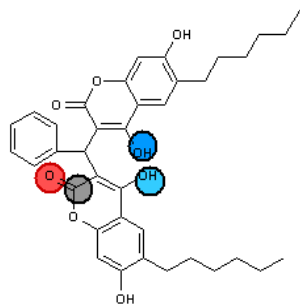
Graph Classification and Clustering

Computer Science Department
Data Mining Research

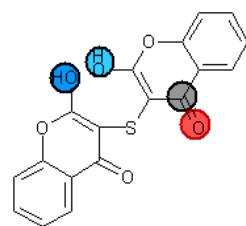
Nov 26, 2014

Graph Classification (I)

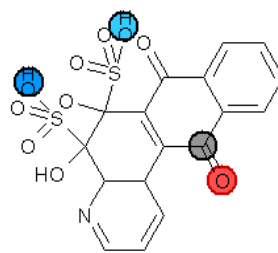
HIV-1 protease inhibitors [Wang et al., 1996]



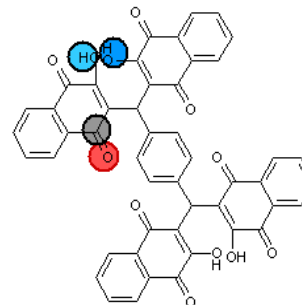
NSC32180



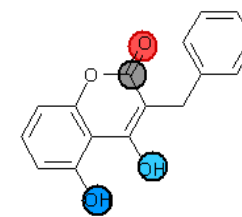
NSC41234



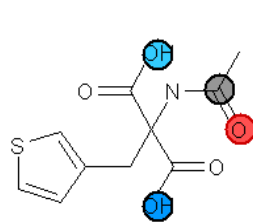
NSC7576



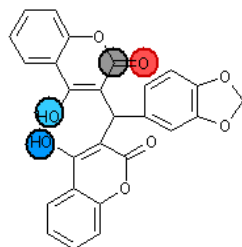
NSC117027



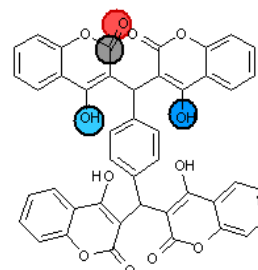
NSC251156



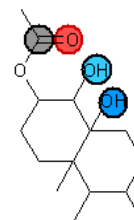
NSC109412



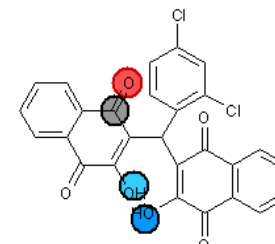
NSC373937



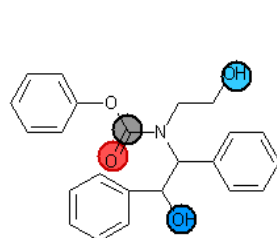
NSC158393



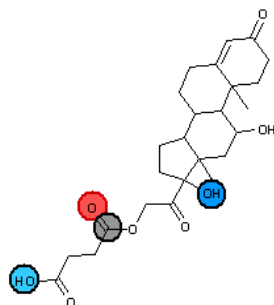
NSC122500



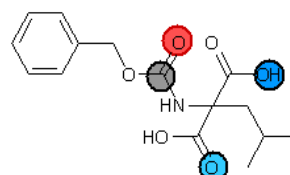
NSC117272



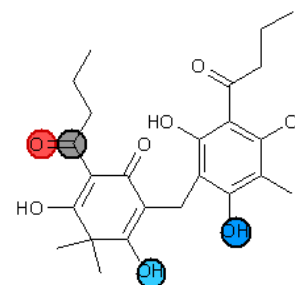
NSC40328



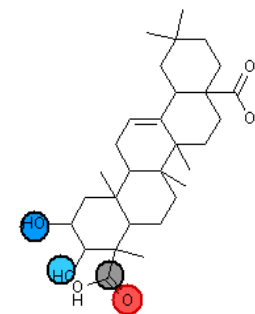
NSC9152



NSC20143



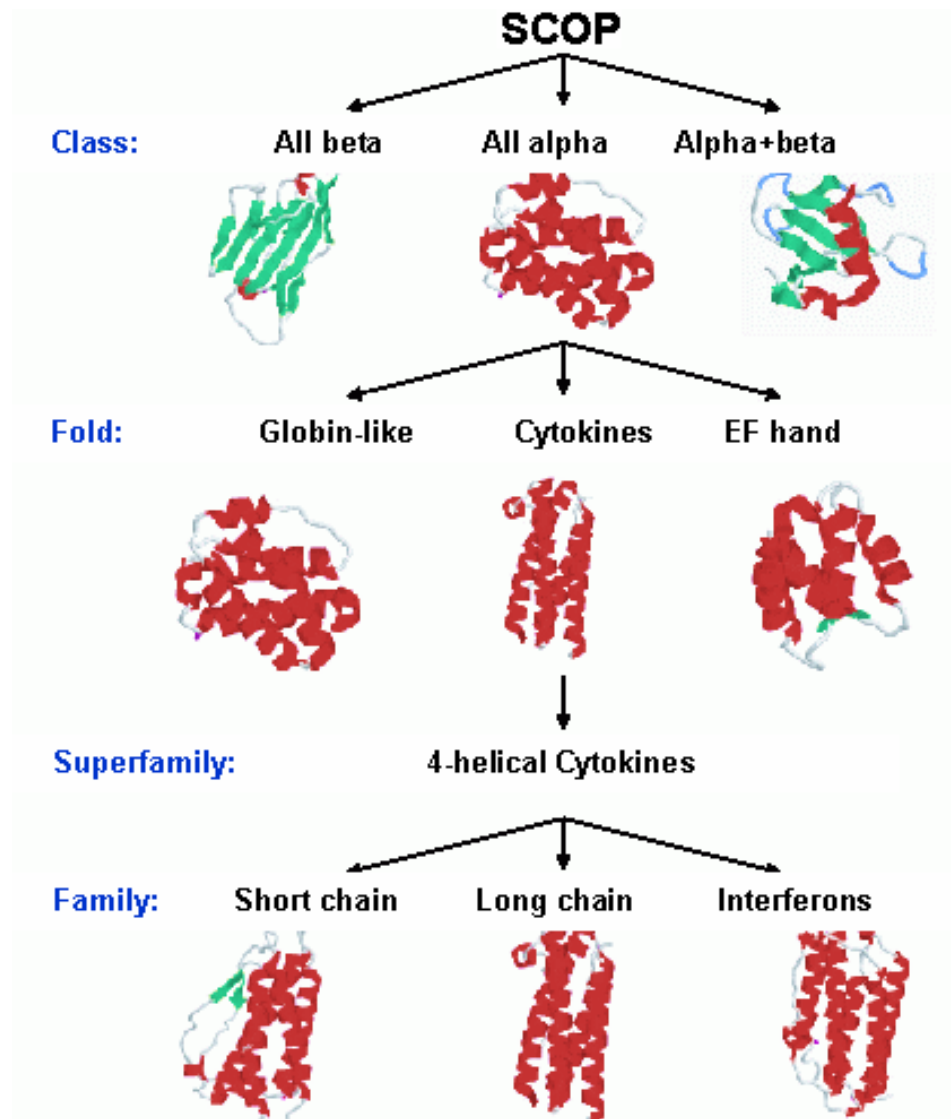
NSC115497



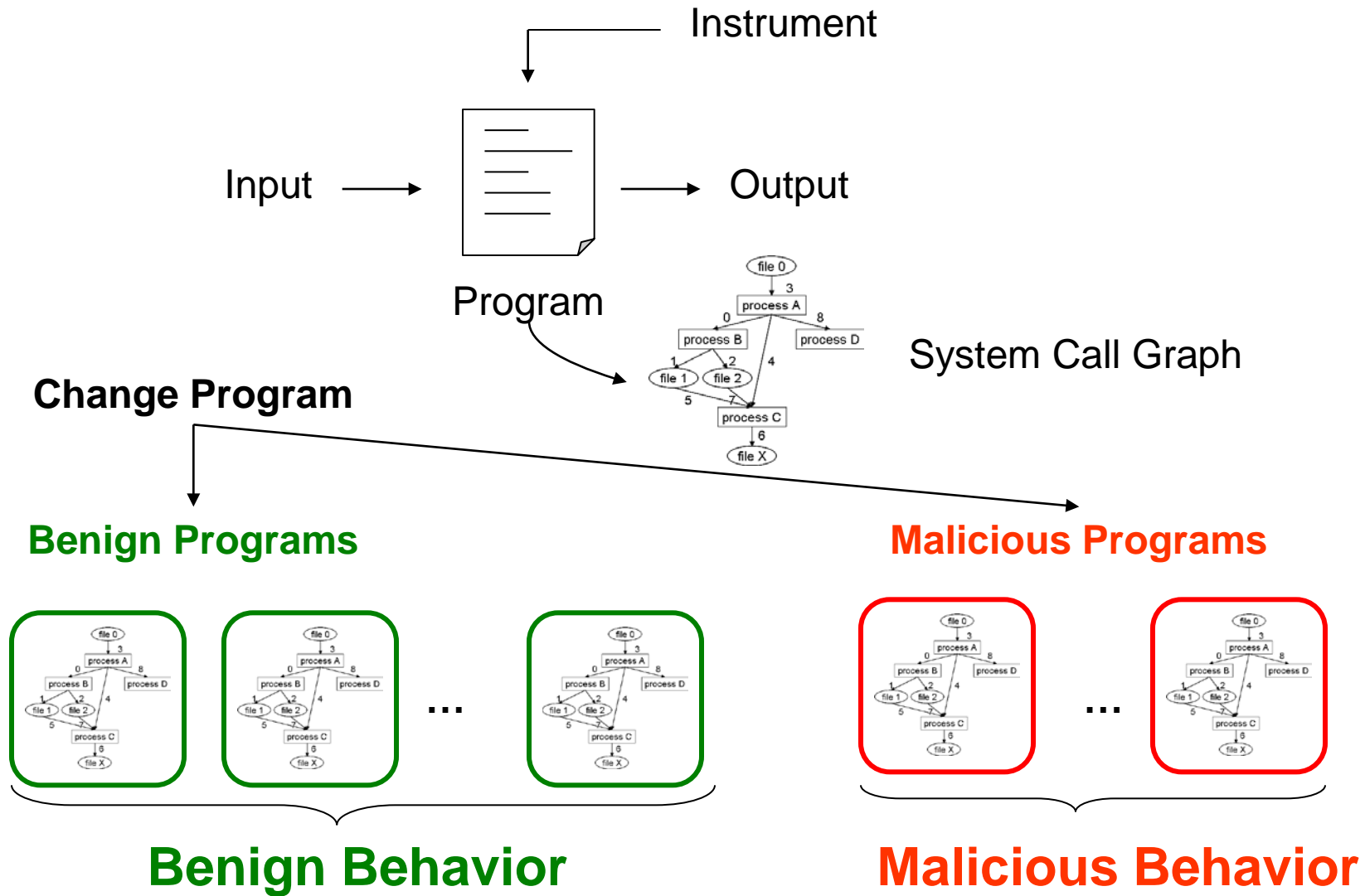
NSC382025

Wang S, Milne GW, Yan X, Posey IJ, Nicklaus MC, Graham L, Rice WG. J Med Chem. 1996 May 10;39(10):2047-54.

Protein Family Classification

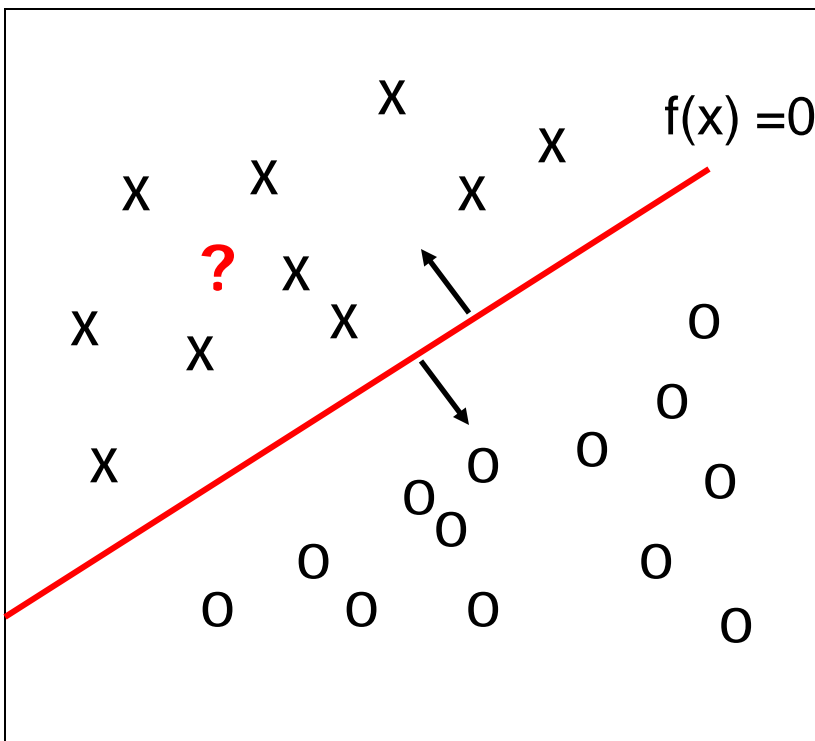


Malware Detection



Classification in Vector Space

- Binary Classification Problem
- Examples: Decision Tree, Naïve Bayes, SVM, ...



Input: $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots\}$

Output: classification function

e.g., $f(\mathbf{x}): \mathbf{w}^t \mathbf{x} + b = 0$

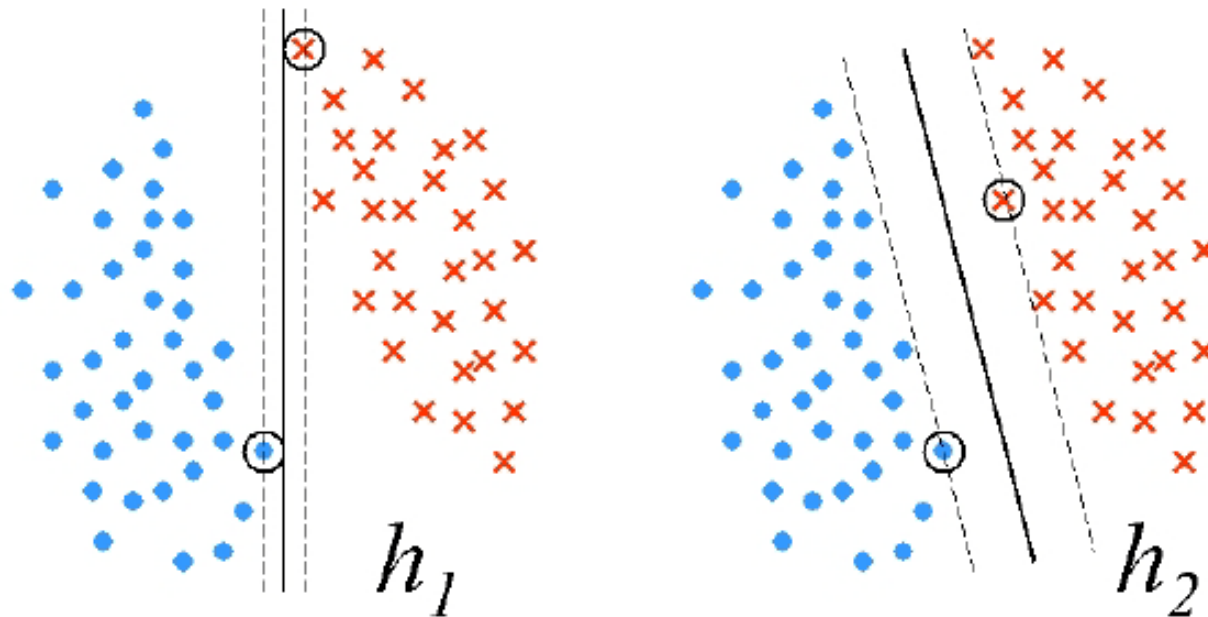
or $w_1 x_1 + w_2 x_2 + b = 0$

Given a new data point

$f(\mathbf{x}) > 0$ for $y = +1$

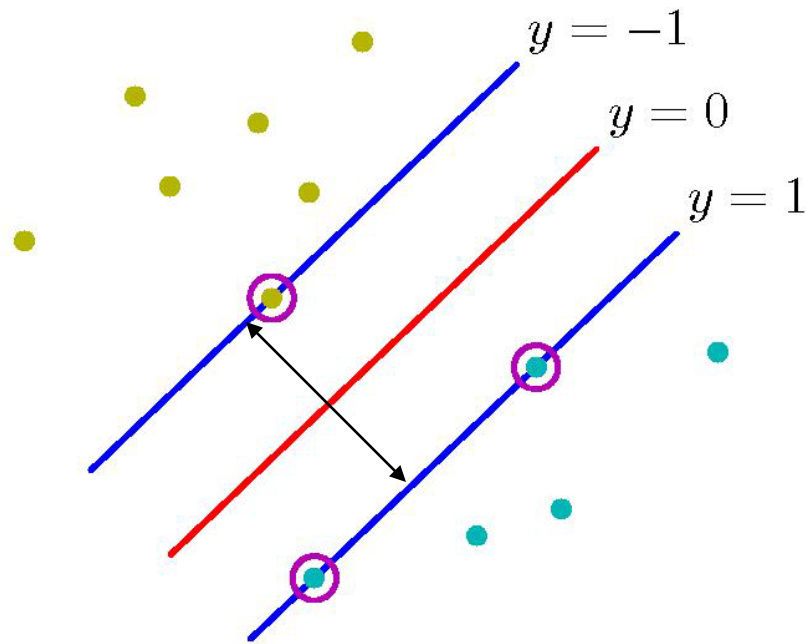
$f(\mathbf{x}) < 0$ for $y = -1$


Support Vector Machine



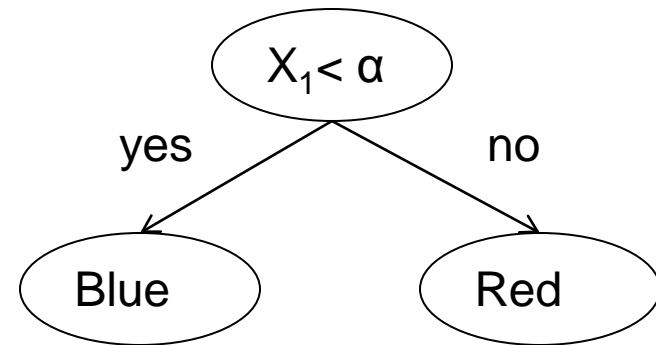
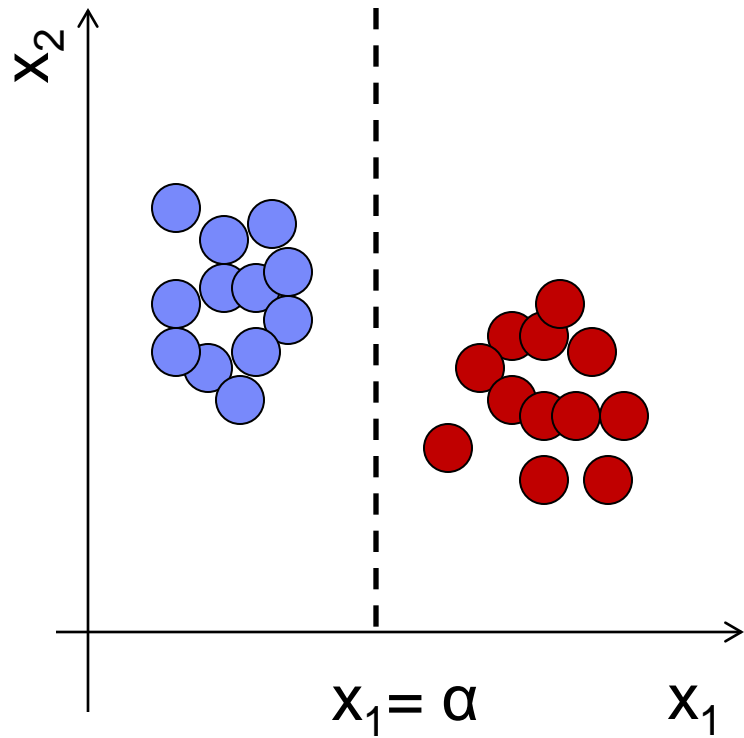
which one is better?

Maximum Margin



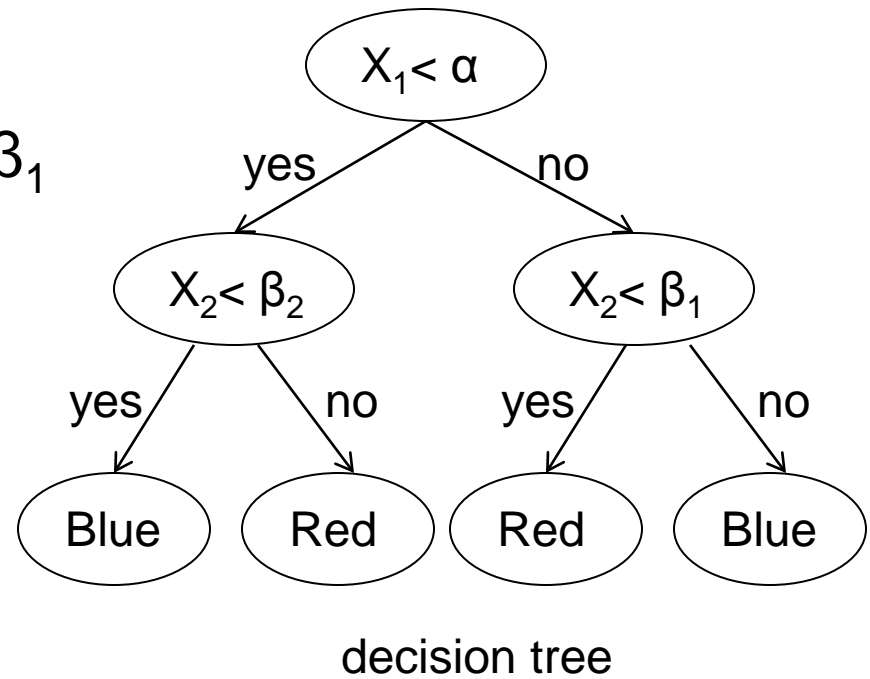
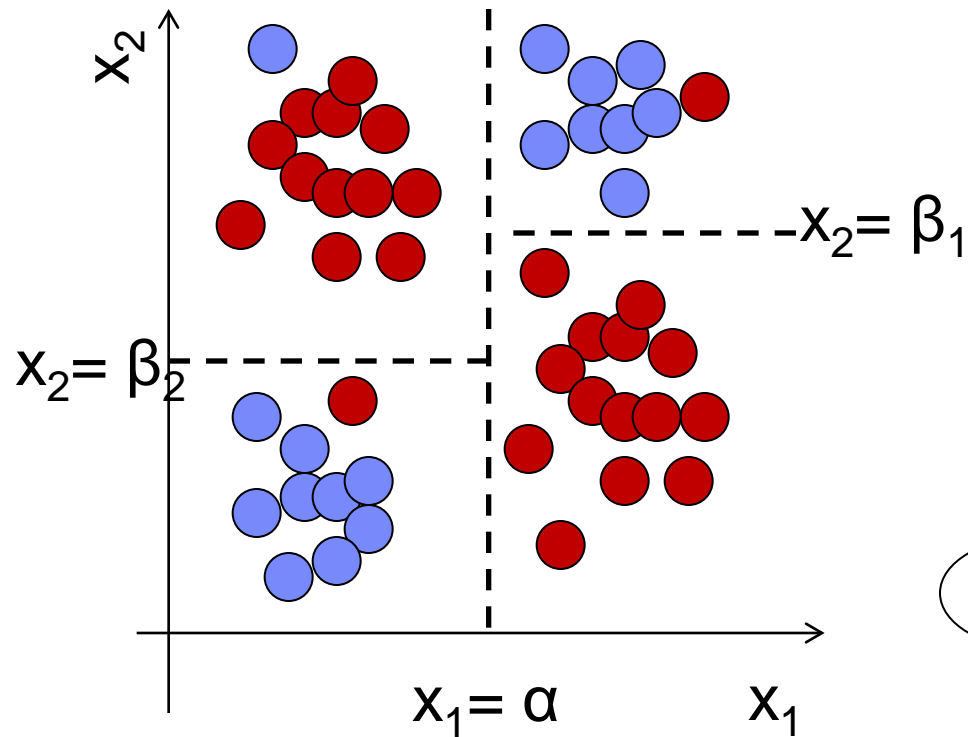
If we fix $f(x) = 1$ and -1 , then the margin is $\frac{2}{\|w\|}$  minimize

Decision Tree



decision tree

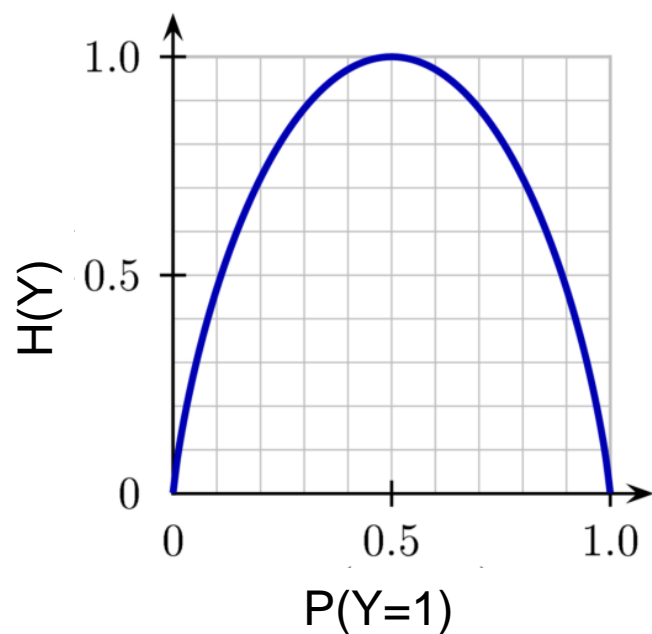
Decision Tree



Entropy

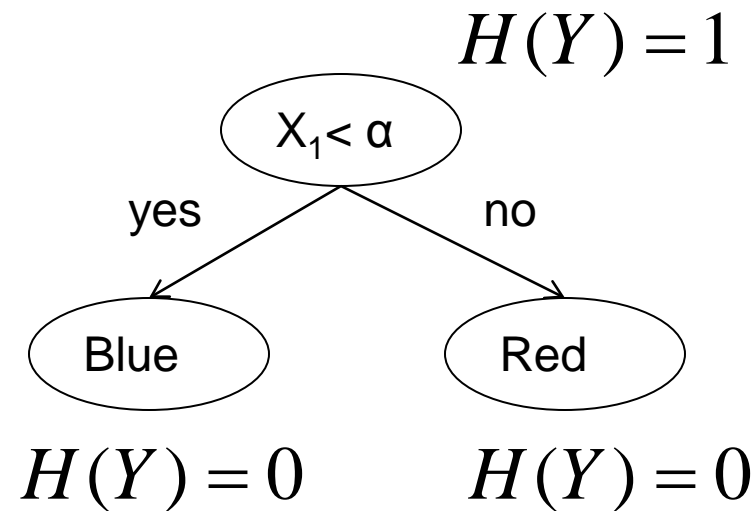
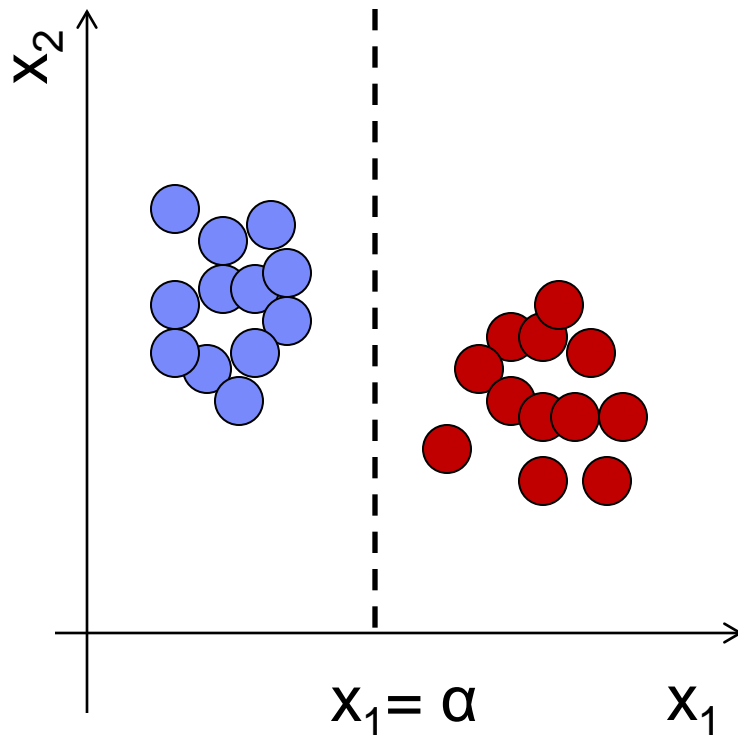
- Let p_i be the probability that an arbitrary tuple in D belongs to class C_i , estimated by $P(Y) = |C_i|/|D|$
- Expected information (entropy, $H(Y)$) needed to classify a tuple in D :

$$H(Y) = -\sum_{i=1}^m p_i \log_2(p_i)$$



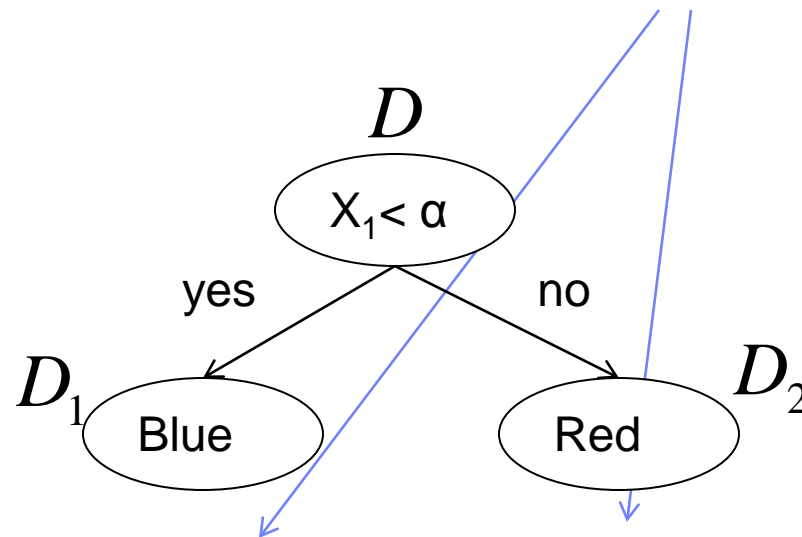
Information Gain

- Select the attribute with the highest information gain
 - Find an attribute that could reduce the entropy -> better predicate the class label



Attribute Selection Measure: Information Gain (ID3/C4.5)

$$H(Y | X) = \sum_i p(x) \times H(Y | D_i)$$



$$H(Y | D_1) = 0 \quad H(Y | D_2) = 0$$

$$Gain(A) = H(Y) - H(Y | X)$$

Graph Classification

Structure-based Approach

- Local structures in a graph, e.g., neighbors surrounding a vertex, paths with fixed length

Pattern-based Approach

- Subgraph patterns from domain knowledge
- Graph pattern mining
- Decision Tree (Fan et al. KDD'08)
- Boosting (Kudo et al. NIPS'04)
- LAR-LASSO (Tsuda, ICML'07)

Kernel-based Approach

- Random walk (Gärtner '02, Kashima et al. '02, ICML'03, Mahé et al. ICML'04)
- Optimal local assignment (Fröhlich et al. ICML'05)
- Many others

Structure/Pattern-based Classification

Basic Idea

- Transform each graph in the dataset into a feature vector,

$$G \rightarrow \mathbf{x} = \{x_1, x_2, \dots, x_n\}$$

where x_i is the frequency of the i -th structure/pattern in G . Each vector is associated with a class label. Classify these vectors in a vector space

Structure Features

- Local structures in a graph, e.g., neighbors surrounding a vertex, paths with fixed length

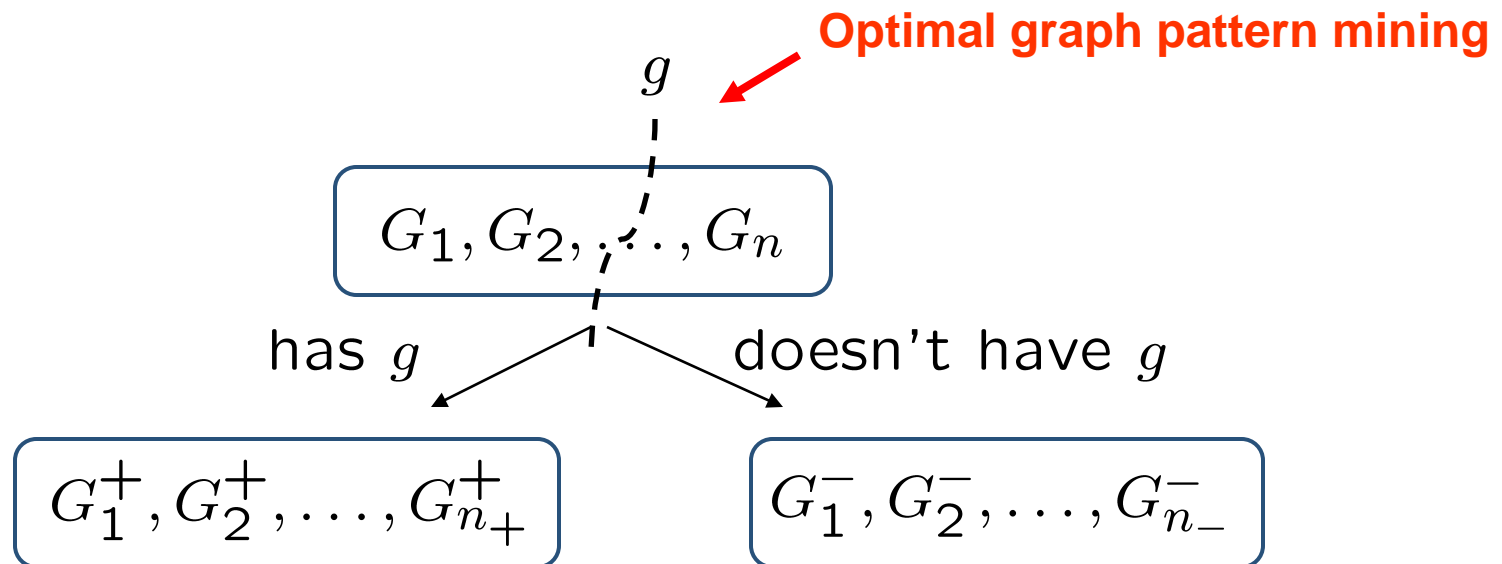
Enumerate all of the subgraphs and select the best features?

- Subgraph patterns from domain knowledge
 - Molecular descriptors
- Subgraph patterns from data mining

Decision-Tree

Basic Idea

- Partition the data in a top-down manner and construct the tree using the best feature at each step according to some criterion
- Partition the data set into two subsets, one containing this feature and the other does not



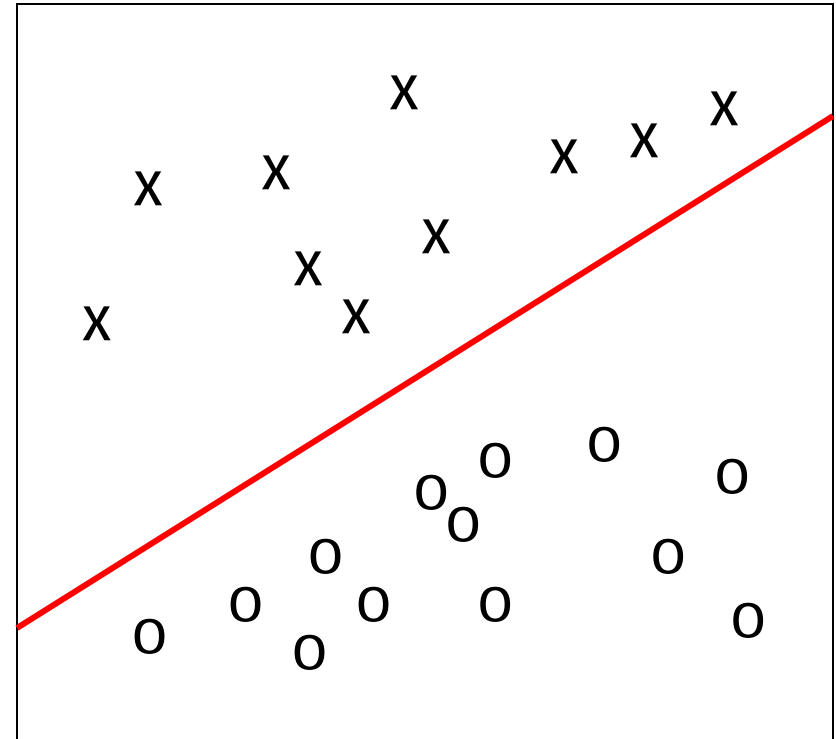
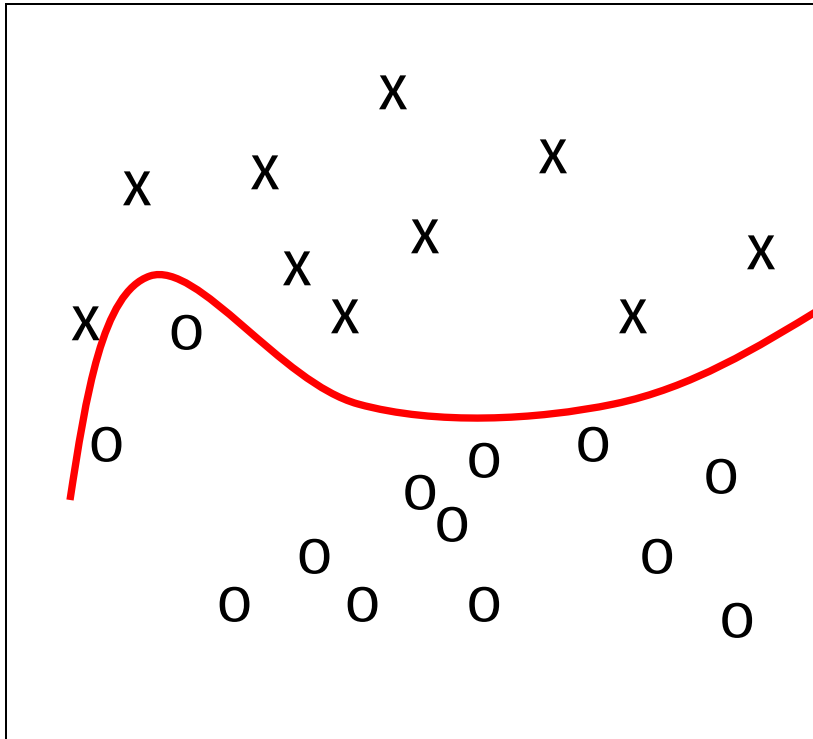
Kernel

- Map two objects x and x' via mapping Φ to feature space H .
- Measure their similarity in H as $\Phi(x), \Phi(x')$.
- Kernel Trick: Compute inner product in H as kernel in input space

$$K(x, x') = \langle \phi(x), \phi(x') \rangle$$

- Goal: reuse linear classifier, e.g., support vector machine, by replacing the kernel

Feature Space

 $\phi(x)$ 

$$f(x) = w^t \phi(x) + b = \sum_k \alpha_k y_k \phi(x_k) \cdot \phi(x) + b = \sum_k \alpha_k y_k \langle \phi(x_k), \phi(x) \rangle + b$$

The Mercer Condition

- Is there a mapping $\Phi(x)$ for any symmetric function $K(x, x')$?
No
- The SVM dual formulation requires calculation $K(x_i, x_j)$ for each pair of training instances. The array $G_{ij} = K(x_i, x_j)$ is called the Gram matrix
- There is a feature space $\Phi(x)$ when G is always semi-positive definite (Mercer condition)
- A matrix M is semi-positive definite if and only if $x^t M x \geq 0$, for all non-zero vector x .

Graph Kernel

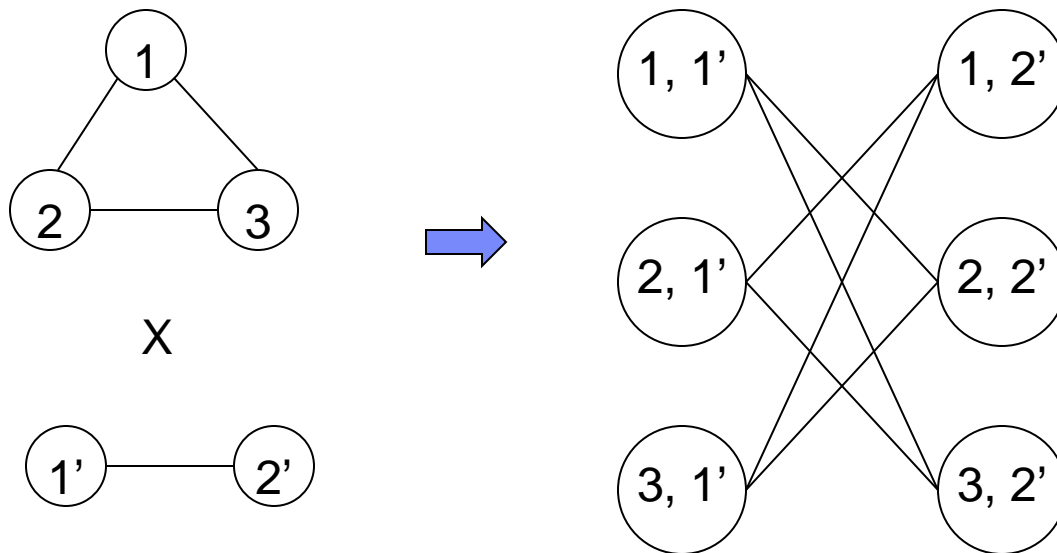
- Kernels on pairs of graphs
- A graph kernel makes the whole family of kernel methods applicable to graphs, e.g., support vector machine!
- More details: Graph Kernel Tutorial:
http://www.cs.ucsb.edu/~xyan/tutorial/kdd08_graph.htm,
Part II
- Video lecture is available:
http://videolectures.net/kdd08_borgwardt_gmgk/ (Part II)

Random Walk-based Graph Kernel

- Random walks are sequences of nodes that allow repetitions of nodes
- Count the number of matching walks in two graphs
- Discount contribution of longer walks
- Two graphs are similar if many walks are matching

Direct Product Graph

- Given two graphs, G and H , the vertex set of $G \times H$ is the Cartesian product $V(G) \times V(H)$
- Two vertices (u, u') and (v, v') are adjacent in $G \times H$ if and only if u is adjacent with v and u' is adjacent with v' .



Random Walk-based Graph Kernel

- Construct direct product graph, A_x of G and H
- Count walks in this product graph $A_x=(V_x,E_x)$
- Each walk in the product graph corresponds to one walk in G and H
- Walks of length k can be computed by looking at the k -th power of the adjacency matrix

$$K(G, H) = \sum_{i,j=1}^{|V_x|} \left[\sum_{k=1}^{\infty} \lambda^k A_x^k \right]_{ij}$$

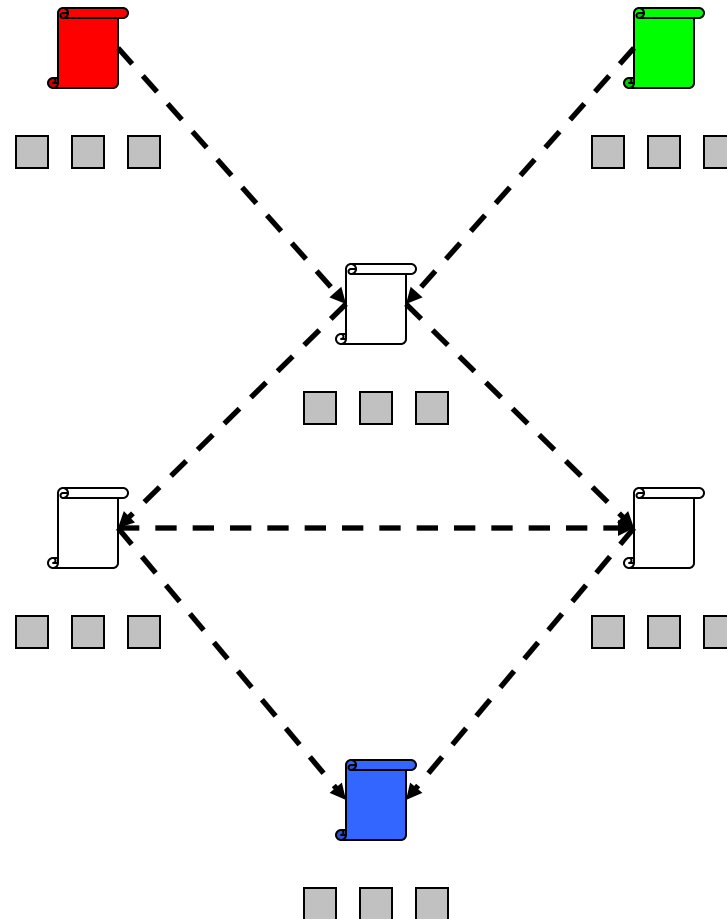
- λ is a decay factor for the sum to converge

Modular Product of graphs

- Given two graphs, G and H , the vertex set of $G \times H$ is the Cartesian product $V(G) \times V(H)$
- Two vertices (u, u') and (v, v') are adjacent in $G \times H$ if and only if u is adjacent with v and u' is adjacent with v' , or u is not adjacent with v and u' is not adjacent with v' .
- Cliques in the modular product graph correspond to isomorphism of induced subgraphs of G and H .
- Specifically, the largest graph that is an induced subgraph of both G and H corresponds to the maximum clique in their modular product.

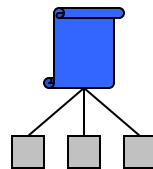
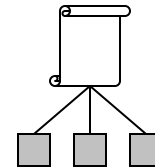
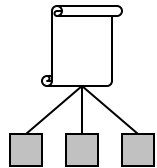
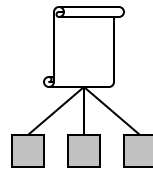
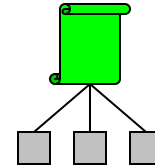
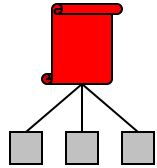
Graph Classification (II)

Task: Classify the Nodes

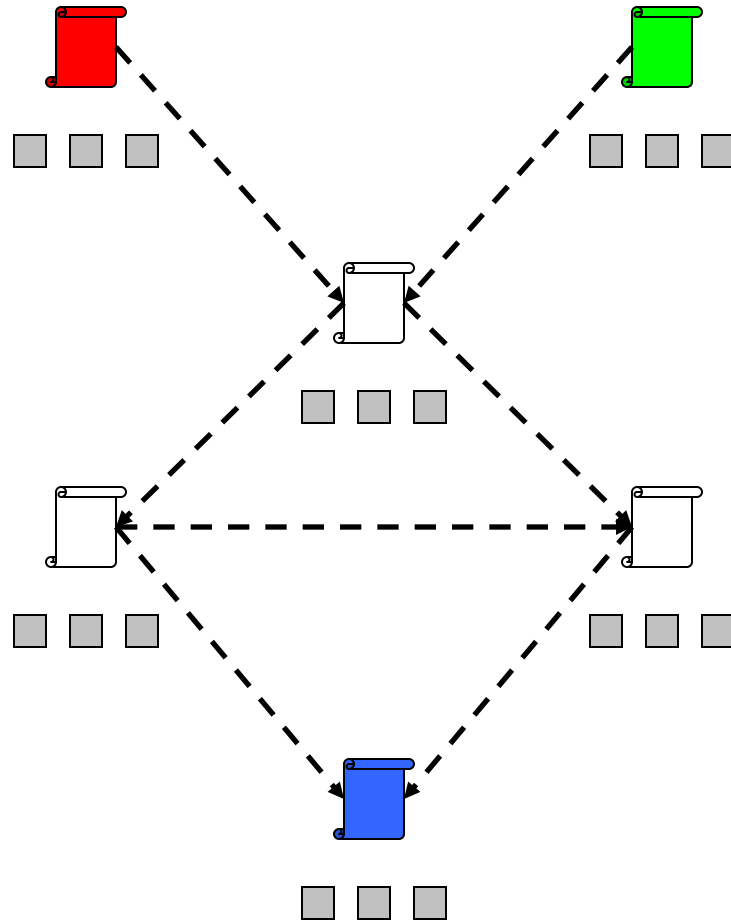


slides adapted from M. Bilgic

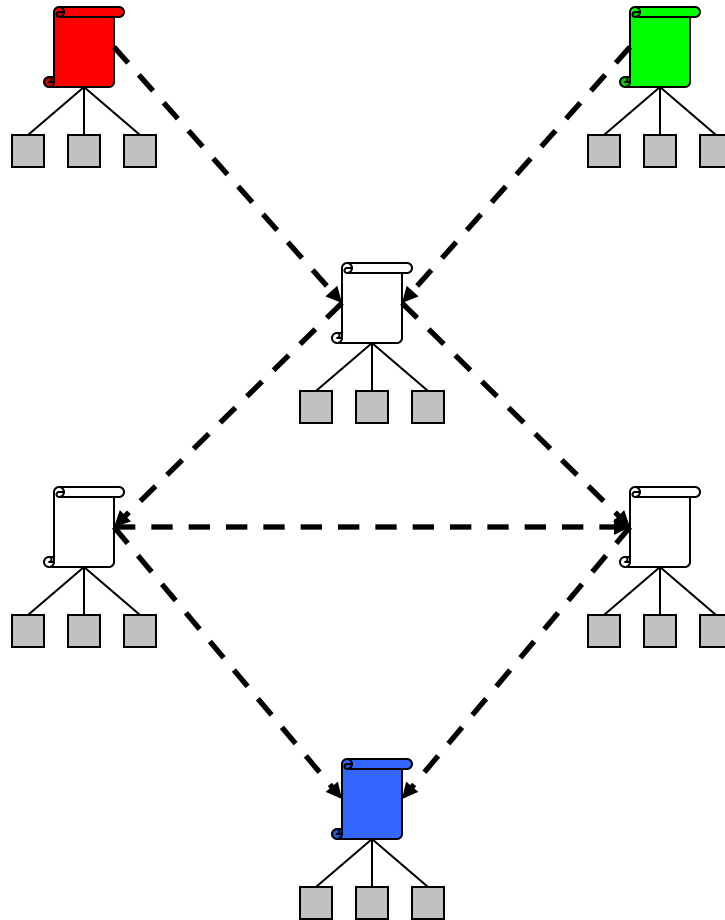
Content-only Classification



Relational Classification

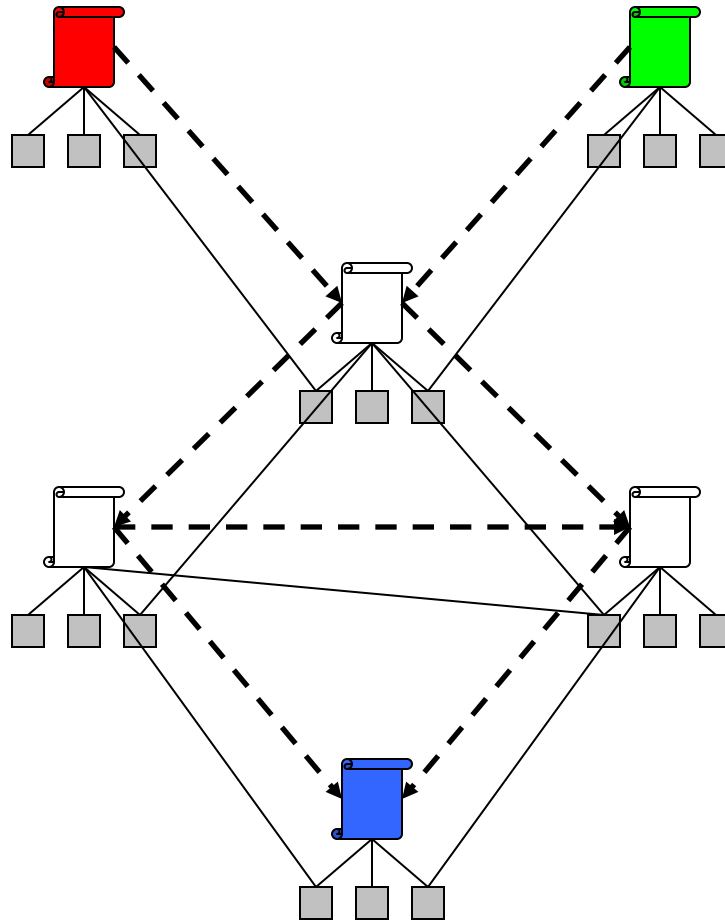


Relational Classification



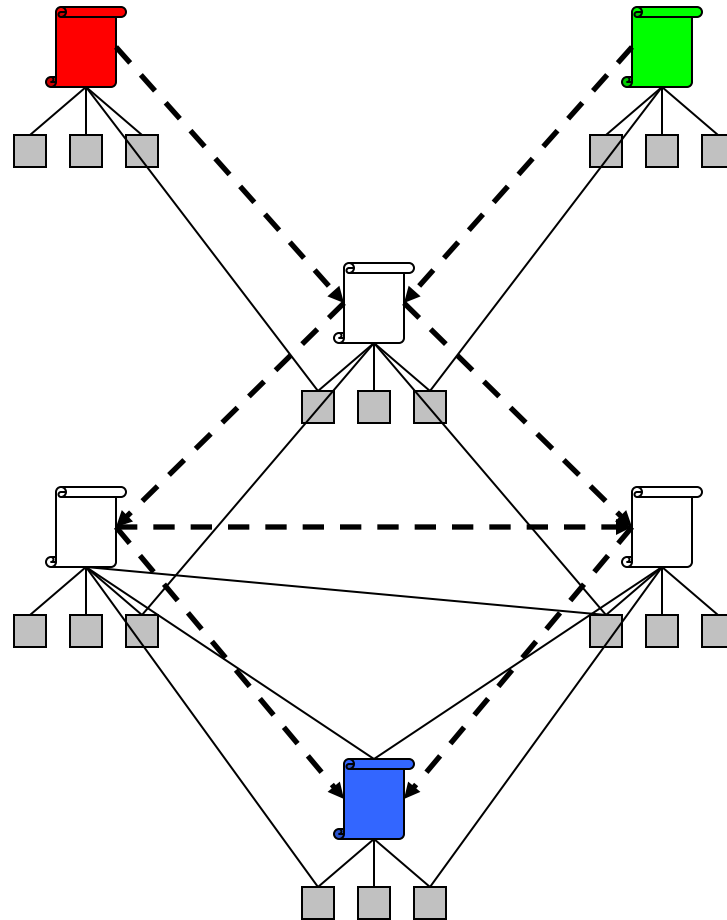
Use the attributes (content).

Relational Classification



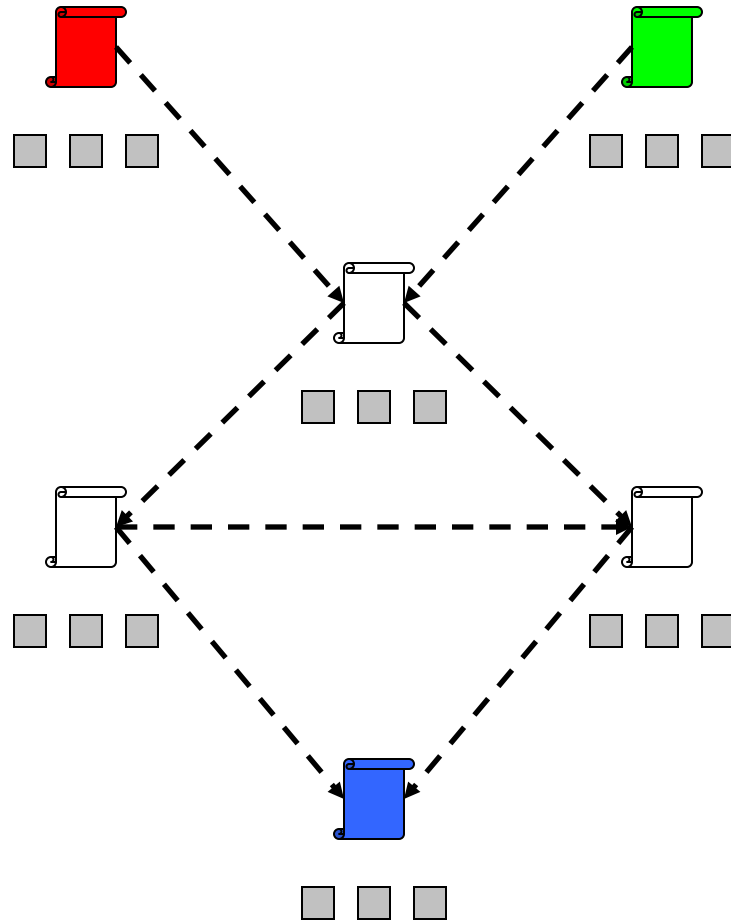
Use the attributes of the related objects.

Relational Classification

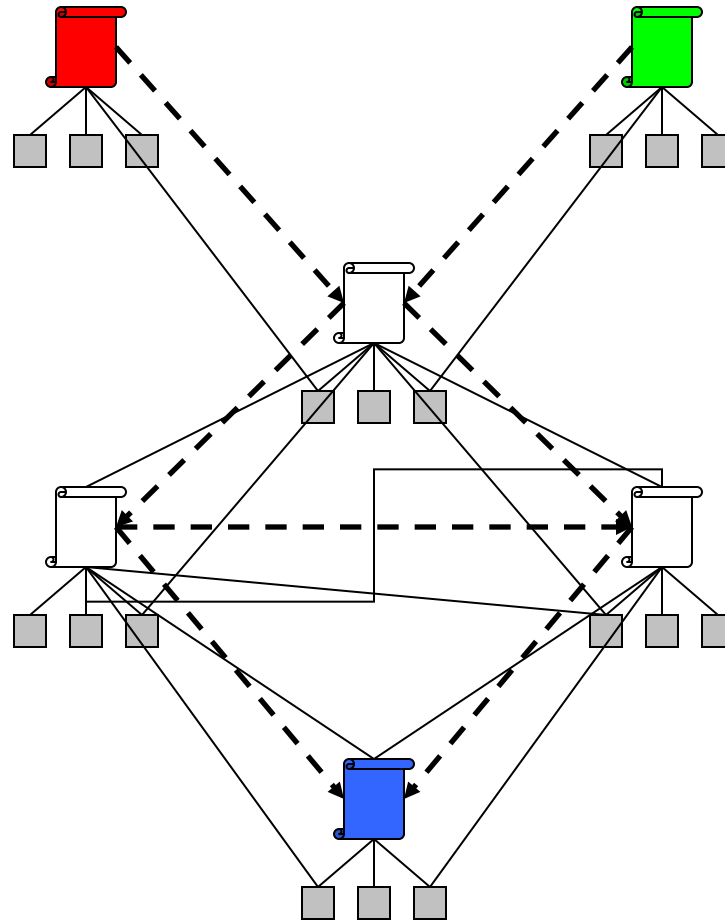


Use the known labels of the related objects.

Collective Classification

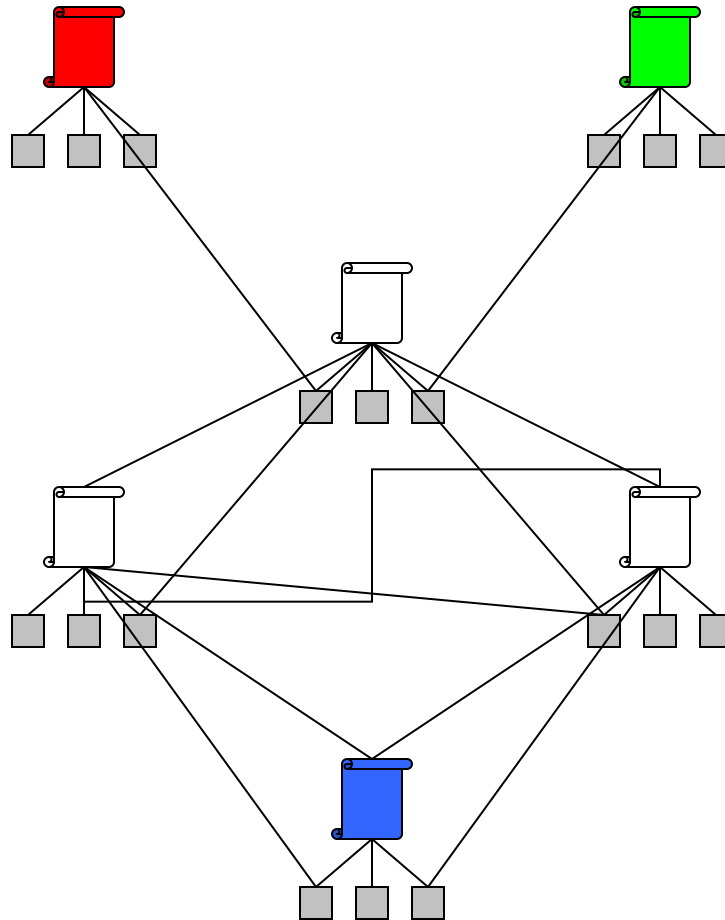


Collective Classification



Use the unknown labels of the related objects (during testing).

Collective Classification

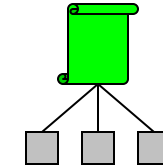
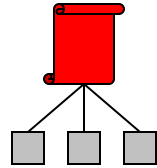


Summary – Information Used

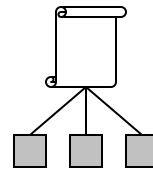
- Content-only classification
 - Each object's own attributes only
- Relational classification
 - Each object's own attributes
 - Attributes of the neighbors
 - Known labels of the neighbors
- Collective classification
 - Each object's own attributes
 - Attributes of the neighbors
 - Known labels of the neighbors
 - Unknown labels of the neighbors (during testing)

Content-only Classification

a1	a2	a3	L
0	1	0	R

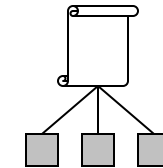
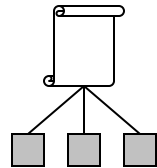


a1	a2	a3	L
1	1	0	G

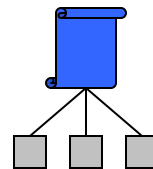


a1	a2	a3	L
1	1	0	?

a1	a2	a3	L
1	0	0	?

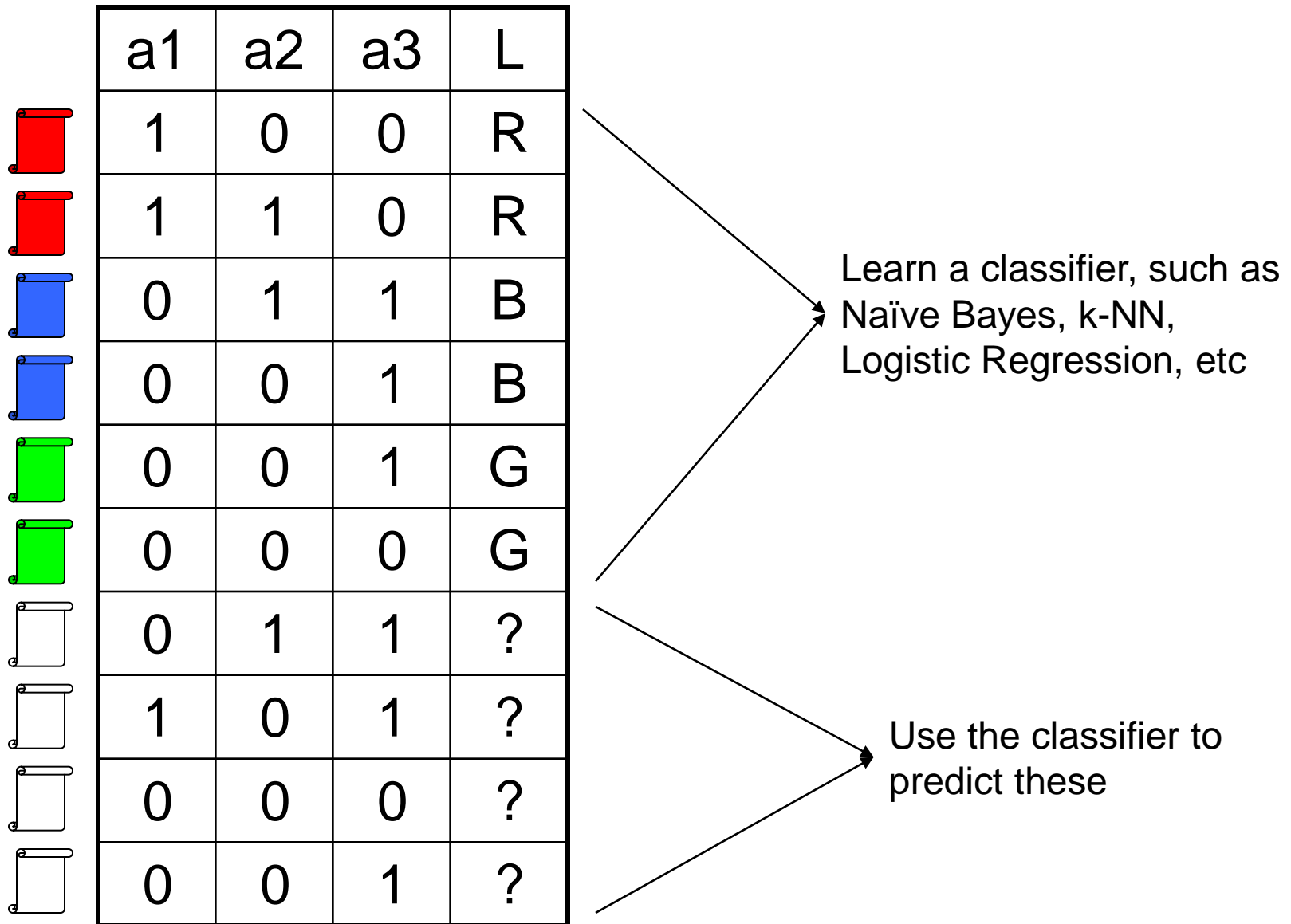


a1	a2	a3	L
1	0	1	?

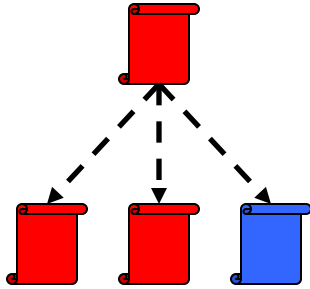


a1	a2	a3	L
1	1	1	B

Content-only Classification

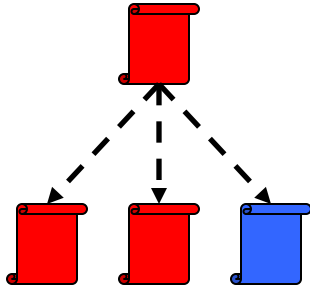


Problems



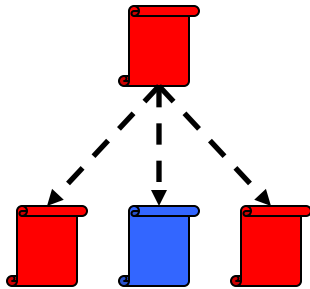
a1	a2	a3	N1	N2	N3	L
0	1	0	R	R	B	R

Problems



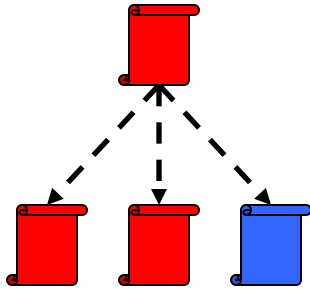
a1	a2	a3	N1	N2	N3	L
0	1	0	R	R	B	R

How do we order the neighbors?

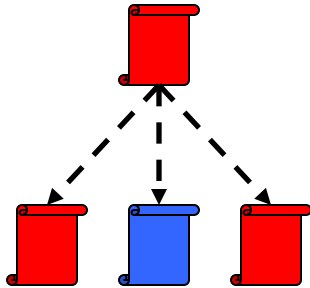


a1	a2	a3	N1	N2	N3	L
0	1	0	R	B	R	R

Problems

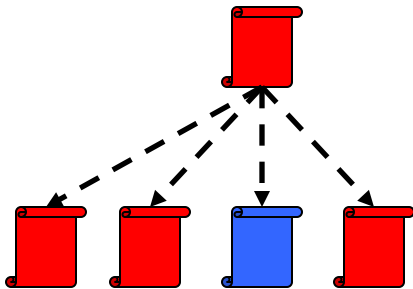


a1	a2	a3	N1	N2	N3	L
0	1	0	R	R	B	R



a1	a2	a3	N1	N2	N3	L
0	1	0	R	B	R	R

What if different nodes have different number of neighbors?



a1	a2	a3	N1	N2	N3	N4	L
0	1	0	R	R	B	R	R

Aggregation

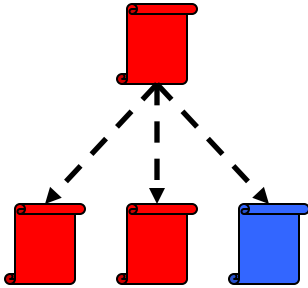
□ Main idea:

- Aggregate a set of attributes into a fixed length representation

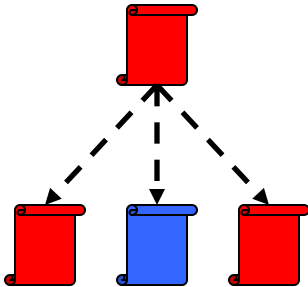
□ Examples

- Count
- Proportion
- Mod
- Exist
- Mean

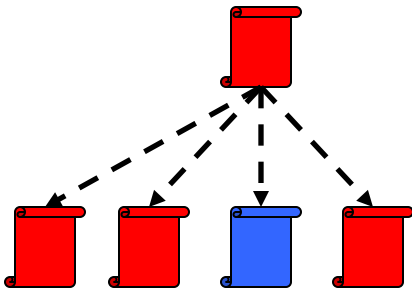
Count



a1	a2	a3	CR	CB	CG	L
0	1	0	2	1	0	R

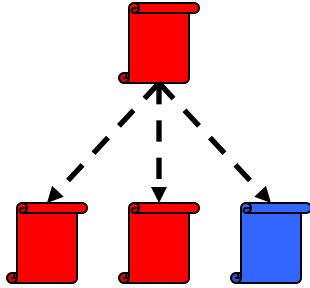


a1	a2	a3	CR	CB	CG	L
0	1	0	2	1	0	R

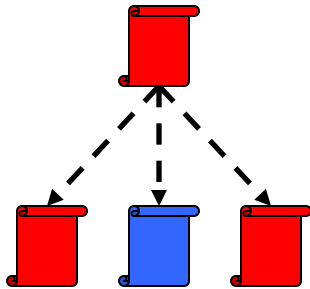


a1	a2	a3	CR	CB	CG	L
0	1	0	3	1	0	R

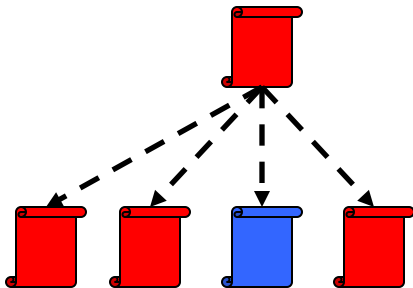
Proportion



a1	a2	a3	PR	PB	PG	L
0	1	0	0.67	0.33	0	R

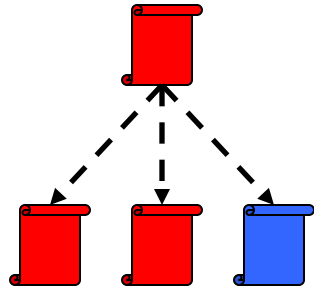


a1	a2	a3	PR	PB	PG	L
0	1	0	0.67	0.33	0	R

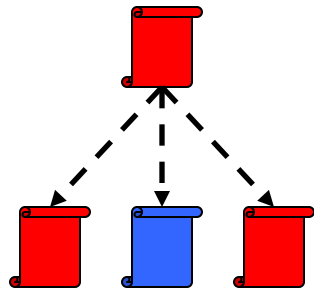


a1	a2	a3	PR	PB	PG	L
0	1	0	0.75	0.25	0	R

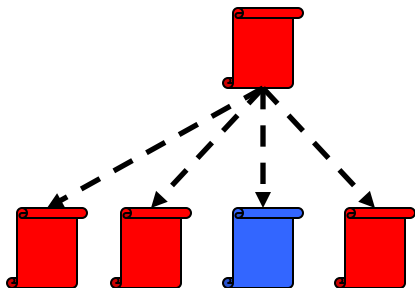
Exist



a1	a2	a3	ER	EB	EG	L
0	1	0	1	1	0	R



a1	a2	a3	ER	EB	EG	L
0	1	0	1	1	0	R



a1	a2	a3	ER	EB	EG	L
0	1	0	1	1	0	R

Feature Construction

- Aggregation is just the tip of the iceberg
- Which relationships to use?
 - In-links
 - Out-links
 - Both
 - Co-citation
- Which attributes to borrow from the neighbors?
 - All
 - Specific ones
 - Words from only the title
 - Age of my friends

Additional Reading

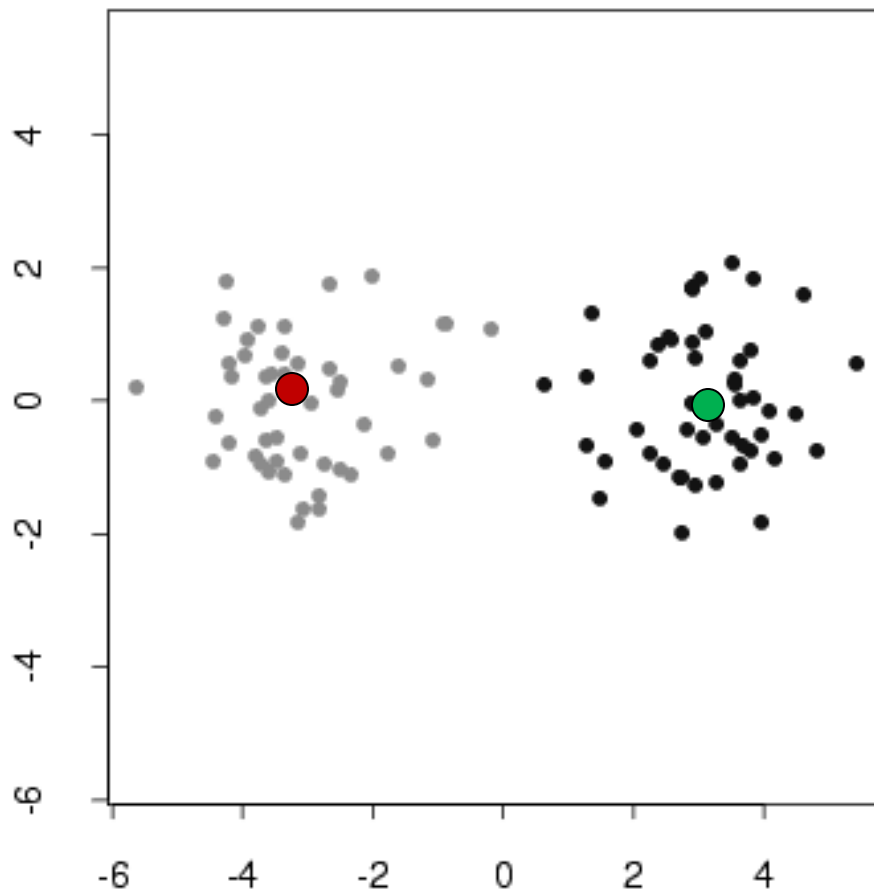
Lise Getoor's tutorial is available at the lecture repository

<http://www.cs.umd.edu/projects/linqs/Tutorials/SDM11/Home.html>

Graph Clustering

Clustering

Grouping a collection of objects into clusters, such that those within each cluster are **more closely related**




K-Means

- Squared Euclidean Distance

$$d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|^2$$

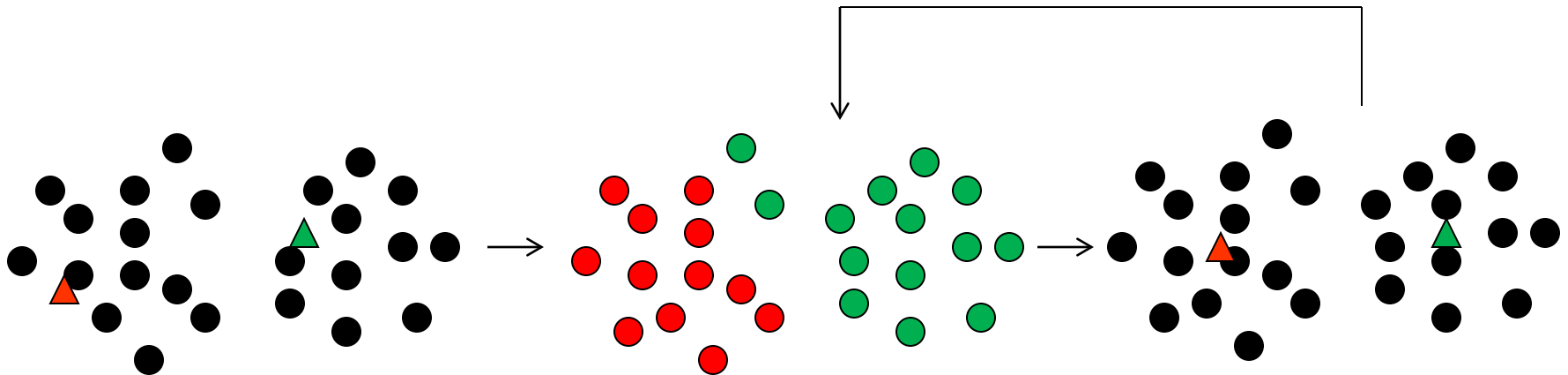
- Sum of Squared Error Distance

$$J = \sum_{k=1}^K \sum_{\mathbf{x}_i \in C_k} \|\mathbf{x}_i - \bar{\mathbf{x}}_k\|^2$$


$$\bar{\mathbf{x}}_k = \frac{1}{n_k} \sum_{\mathbf{x}_i \in C_k} \mathbf{x}_i$$

K-Means Iterative Optimization

- Initialize: Randomly partition the data into k initial clusters
- Step 1: Compute the mean of each cluster
- Step 2: Assign each point to the closest partition
- Step 3: If any point changed its cluster membership
Then repeat Step 1



Variants: K-medians and K-medoids

- Both minimize the sum of the distances from the centroids to the points
- K-medians: Instead of calculating the mean for each cluster to determine its centroid, K-median instead calculates the median.
- K-medoids: It requires that the center of each cluster be a sample point.
- Both problems can be solved using an iterative method like K-means.

Graph Clustering: Four Strategies

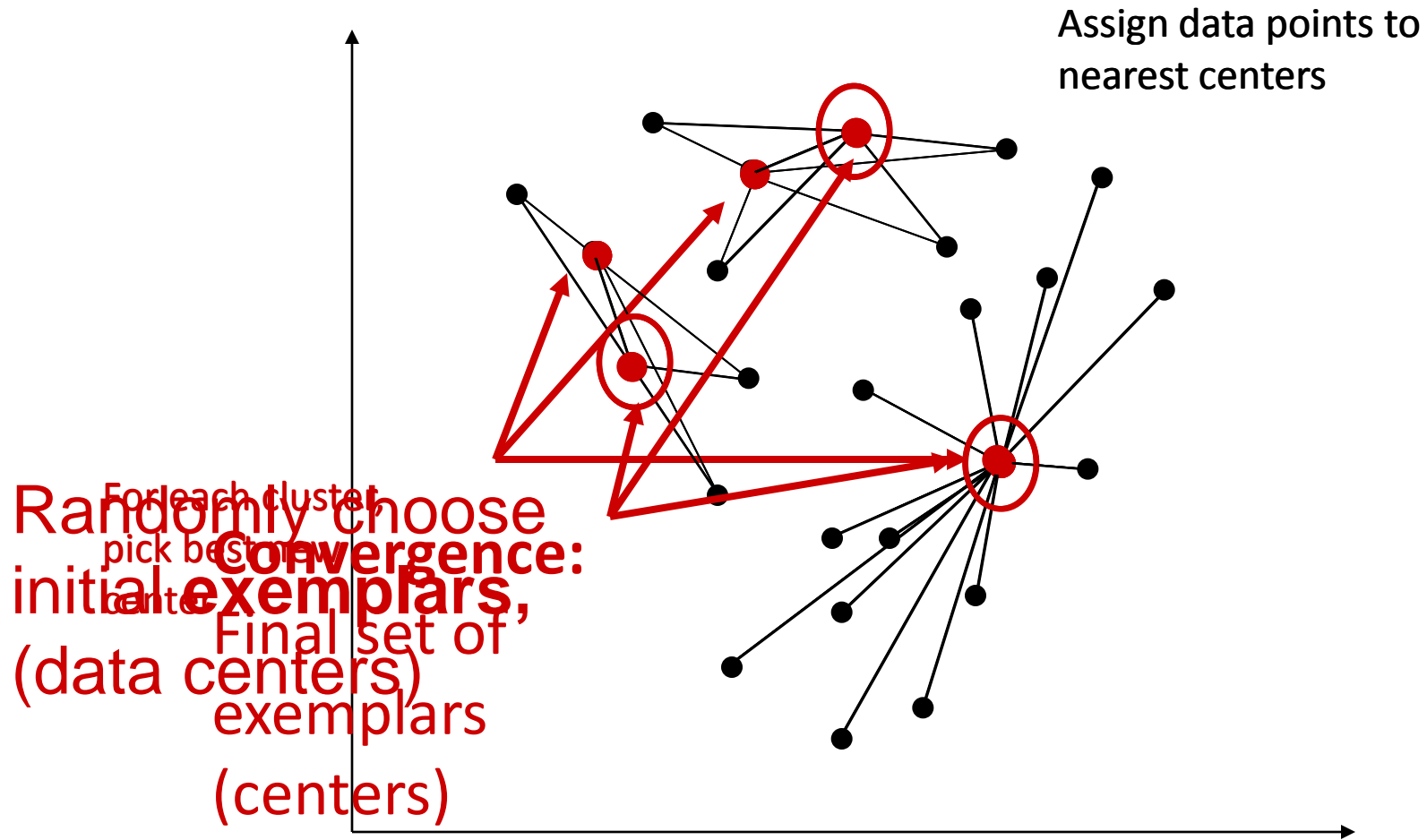
- Similar Behavior: u, v are in the same group if and only if u and v have similar connections w.r.t other nodes

- Graph Cuts
 - Remove some edges => disconnected graph
 - The groups are the connected components

- Embedding: Map *nodes to vectors in a Euclidean space*, then use standard clustering methods

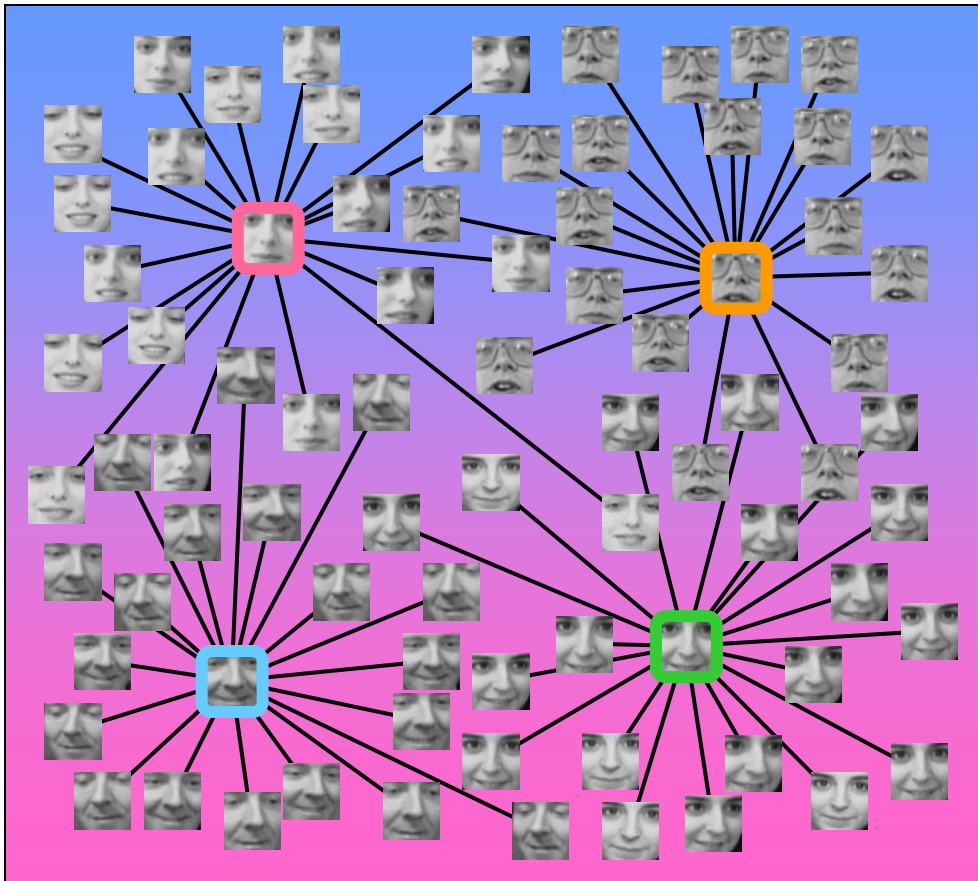
- Close Distance: u, v are in the same group if and only if u and v are close to each other

Greedy Method: k -medians clustering



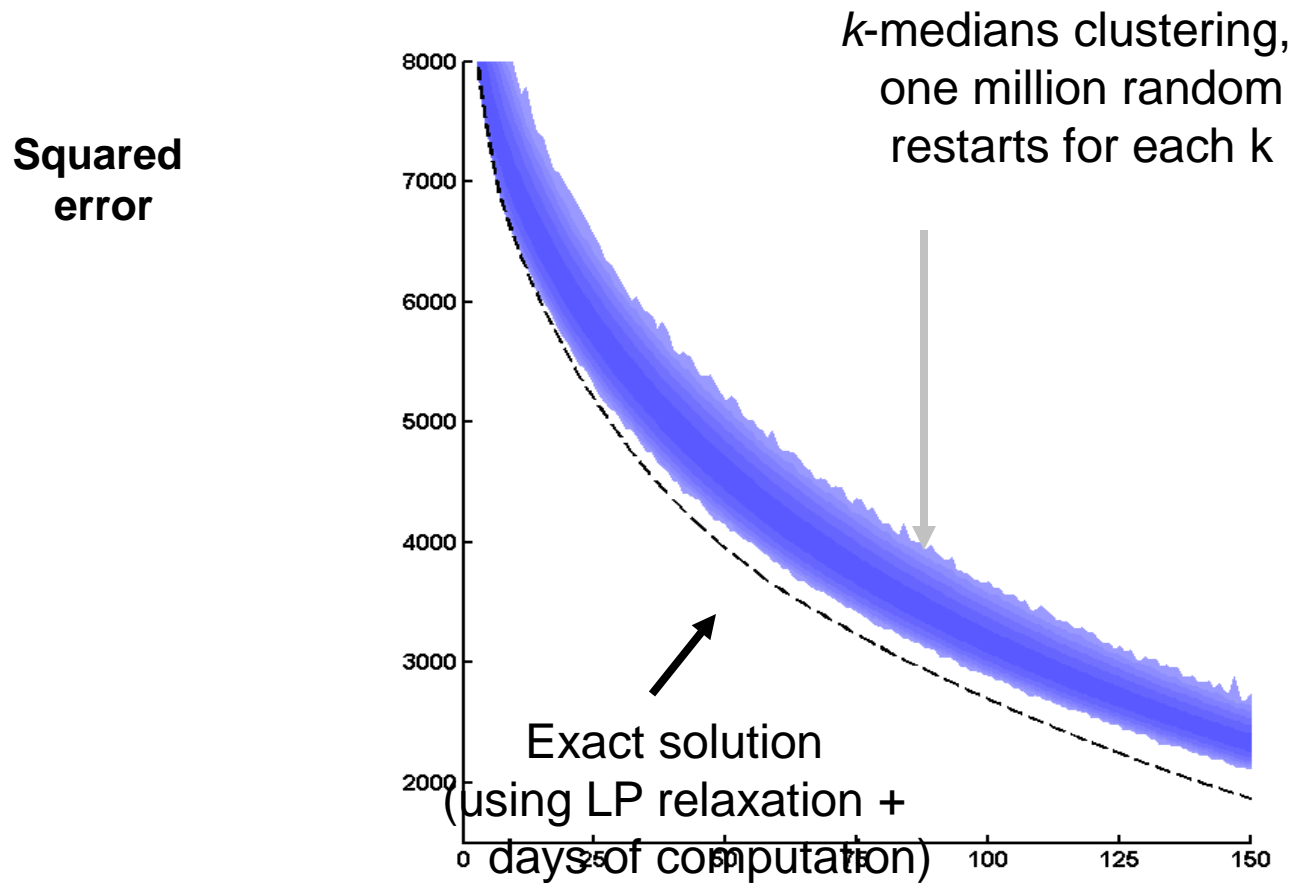
Slides from Delbert Dueck

Example: Olivetti face images



- Olivetti face database contains 400 greyscale 64×64 images from 40 people
 - Similarity is based on sum-of-squared distance using a central 50×50 pixel window
- Small enough problem to find exact solution

Olivetti faces: squared error achieved by ONE MILLION runs of k -medians clustering



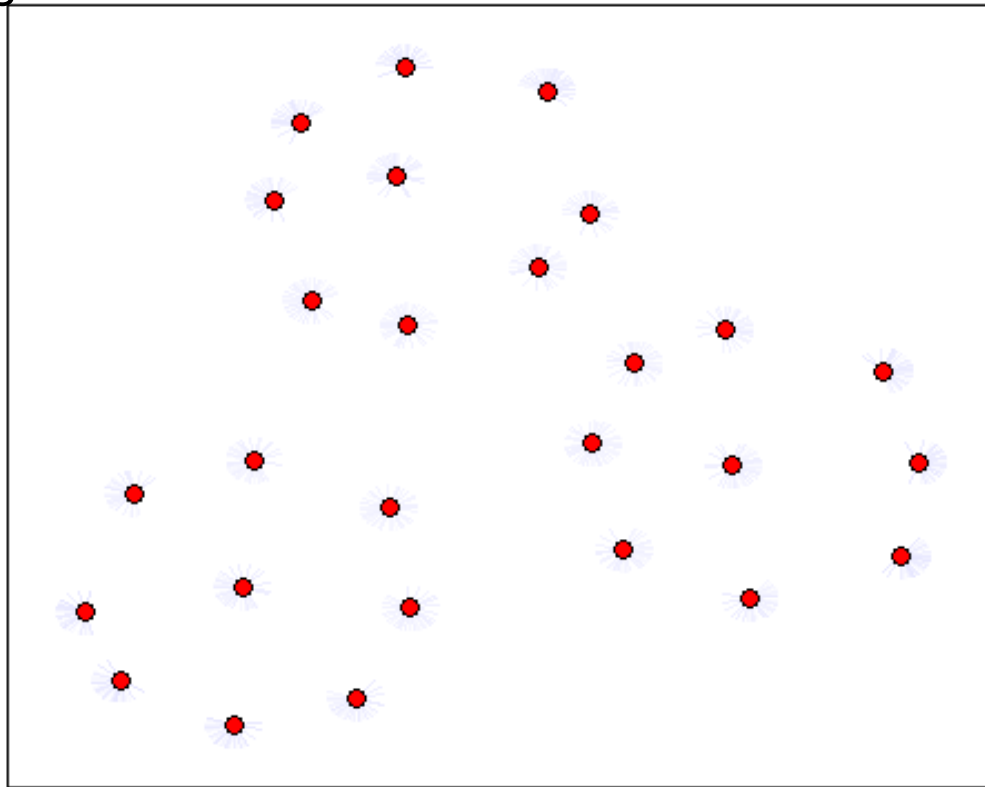
AFFINITY PROPAGATION

Science, 16 Feb. 2007
joint work with Brendan Frey

One-sentence summary:
All data points are simultaneously considered as exemplars, but exchange deterministic messages while a good set of exemplars gradually emerges.

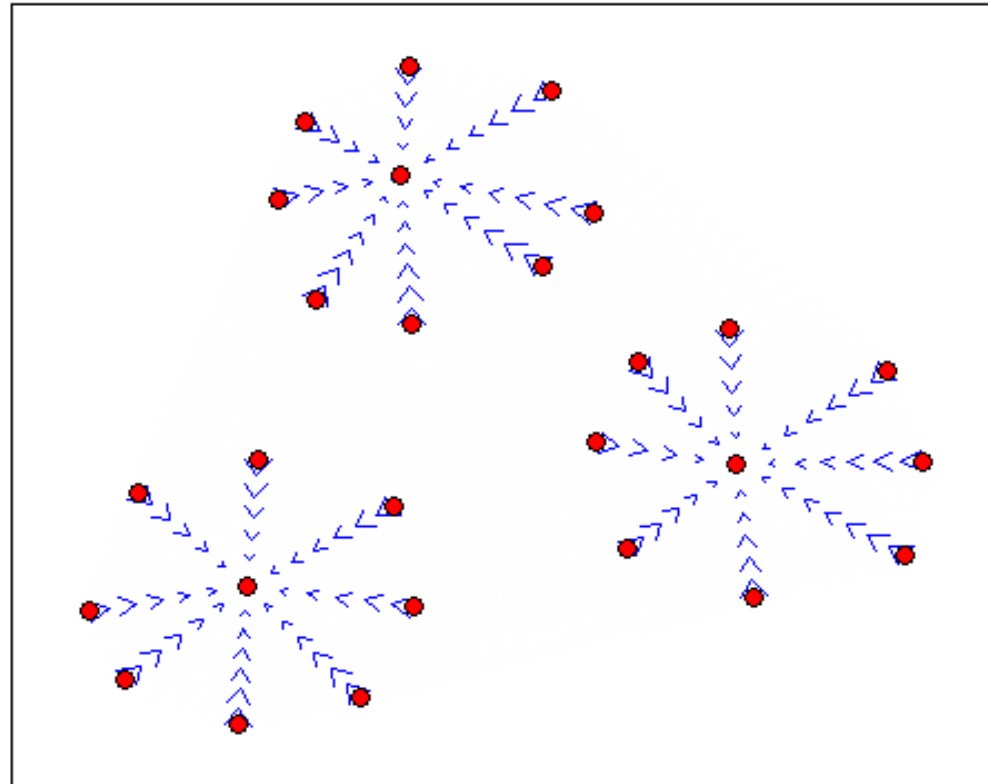
Affinity Propagation: visualization

All data points are simultaneously considered as exemplars, but exchange deterministic messages while a good set of exemplars gradually emerges.



ITERATION 1 of 72

Affinity Propagation: visualization



ITERATION 72 of 72

Affinity Propagation

□ TASK:

Identify a subset of data points as exemplars and assign every other data point to one of those exemplars

□ INPUTS:

- A set of pairwise **similarities**, $\{s(i,k)\}$, where $s(i,k)$ is a real number indicating how well-suited data point k is as an exemplar for data point i

$$\text{e.g. } s(i,k) = -\|x_i - x_k\|^2, i \neq k$$

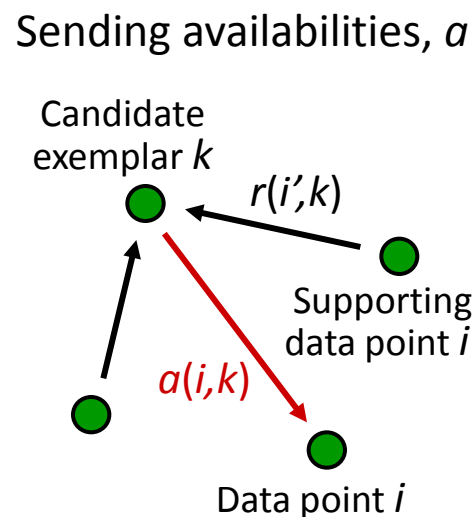
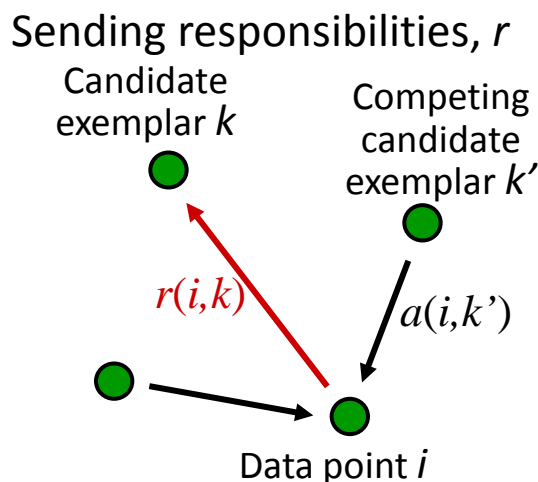
**Need not
be metric!**

- For each data point k , a real number, $s(k,k)$, indicating the *a priori preference* that it be chosen as an exemplar

$$\text{e.g. } s(k,k) = p \quad \forall k$$

Affinity Propagation: message-passing

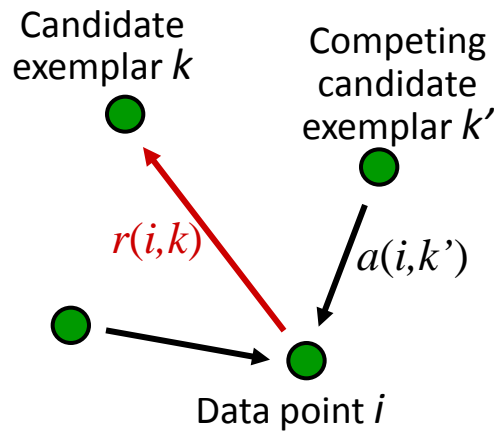
- Affinity propagation can be viewed as data points exchanging messages amongst themselves
 - It can be derived as belief propagation (max-product) on a completely-connected factor graph



Responsibilities are how much you think you're in someone else's cluster.
 Availabilities are how much I think someone is in my cluster.

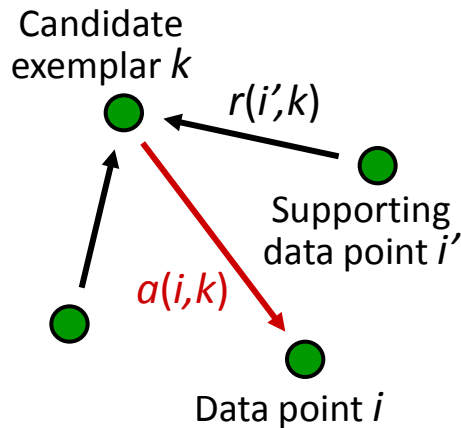
Affinity Propagation: update equations

Sending responsibilities



$$r(i,k) \leftarrow s(i,k) - \max_{k' \text{ s.t. } k' \neq k} \{ s(i,k') \}$$

Sending availabilities



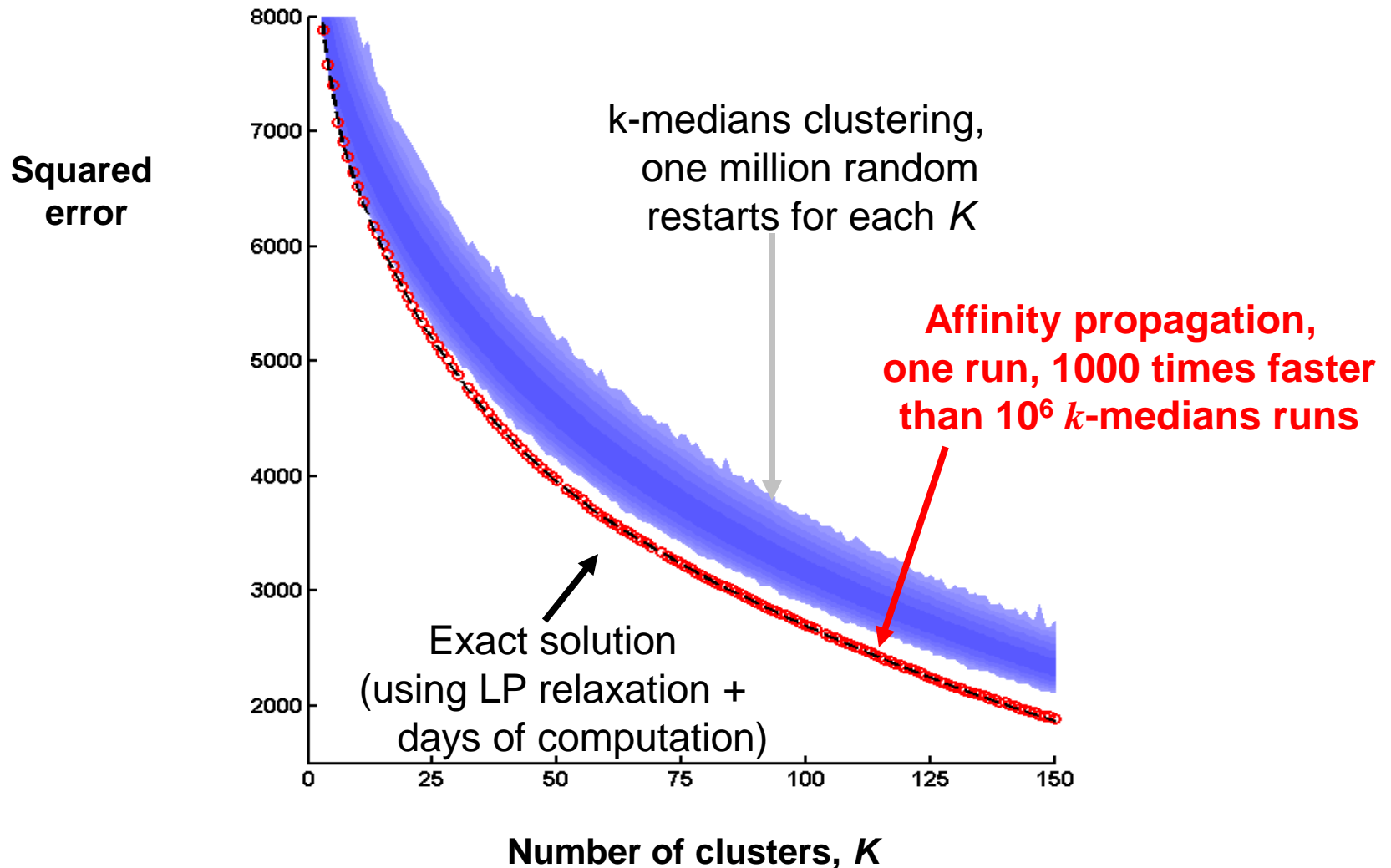
$$a(i,k) \leftarrow \sum_{i' \text{ s.t. } i' \neq i} r(i',k)$$

$$a(k,k) \leftarrow \sum_{i' \text{ s.t. } i' \neq k} \max\{0, r(i',k)\}$$

Making decisions:

$$\operatorname{argmax}_k \{a(i,k) + r(i,k)\}$$

Olivetti faces: squared error achieved by Affinity Propagation

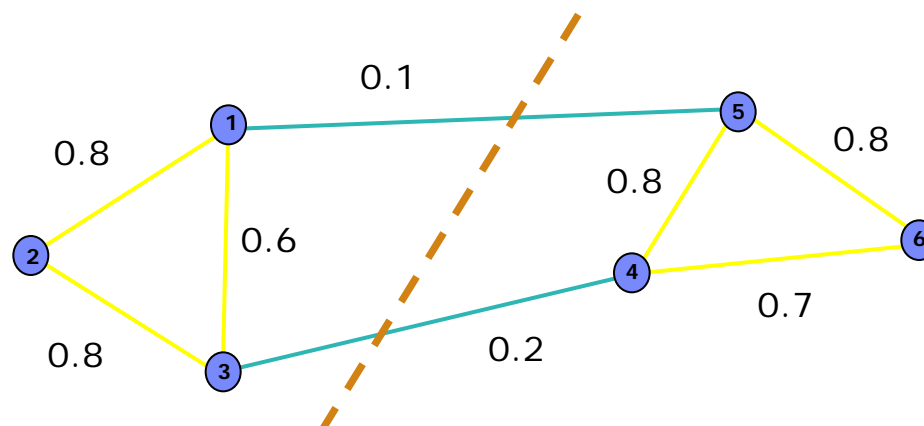


Clustering Objectives

□ Traditional definition of a “good” clustering:

1. Points assigned to the same cluster should be highly similar.
2. Points assigned to different clusters should be highly dissimilar.

Apply these objectives to our graph representation



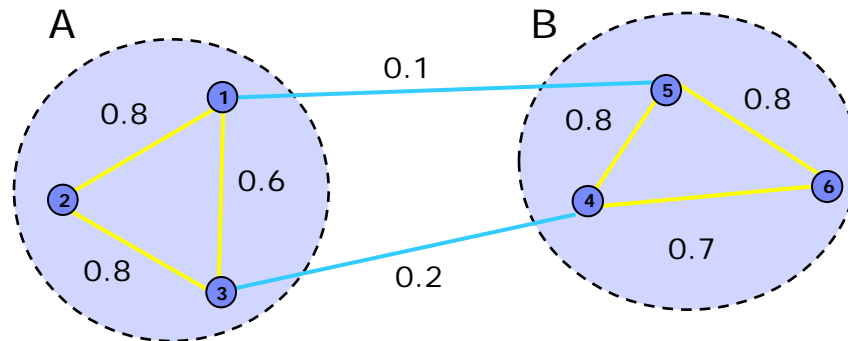
1. Maximize weight of **within-group** connections
2. Minimize weight of **between-group** connections

Slides Adapted from Royi Itzhak

Graph Cuts

- Express partitioning objectives as a function of the “edge cut” of the partition.
- *Cut*: Set of edges with only one vertex in a group.

$$cut(A, B) = \sum_{i \in A, j \in B} w_{ij}$$



$$\Rightarrow cut(A, B) = 0.3$$

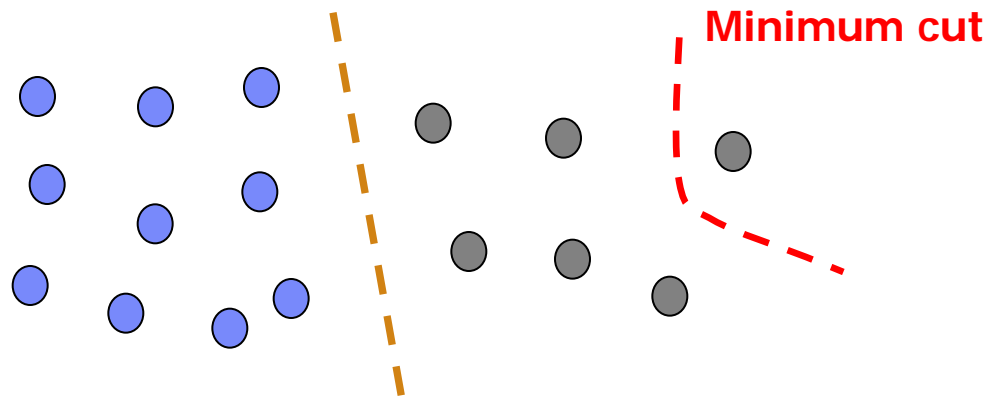
Graph Cut Criteria

□ Criterion: Minimum-cut

- Minimize weight of connections between groups

$$\min \text{cut}(A, B)$$

- Degenerate case: **Optimal cut**



- Problem:

- Only considers external cluster connections
- Does not consider internal cluster density

Graph Cut Criteria (continued)

□ **Criterion: Normalized-cut** (Shi & Malik,'97)

- Consider the connectivity between groups relative to the density of each group.

$$\min Ncut(A, B) = \frac{cut(A, B)}{vol(A)} + \frac{cut(A, B)}{vol(B)}$$

- Normalize the association between groups by *volume*.
 - $Vol(A)$: The total weight of the edges originating from group A .
- Why use this criterion?
 - Produces more balanced partitions.
- Computing an optimal cut is NP-hard

Spectral Graph Theory

□ Possible approach

- Represent a similarity graph as a matrix
- Apply knowledge from Linear Algebra...

- The *eigenvalues* and *eigenvectors* of a matrix provide global information about its structure.

$$\begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \lambda \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

■ *Spectral Graph Theory*

- Analyze the “spectrum” of matrix representing a graph.
- *Spectrum* : The eigenvectors of a graph, ordered by the magnitude of their corresponding eigenvalues.

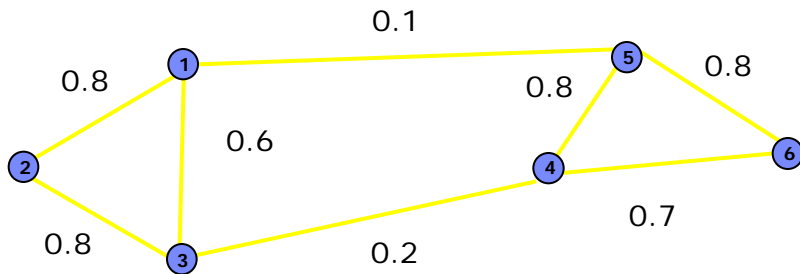
$$\Lambda = \{\lambda_1, \lambda_2, \dots, \lambda_n\}$$

Matrix Representations

□ Adjacency matrix (A)

■ $n \times n$ matrix

■ $A = [w_{ij}]$: edge weight between vertex x_i and x_j



	x_1	x_2	x_3	x_4	x_5	x_6
x_1	0	0.8	0.6	0	0.1	0
x_2	0.8	0	0.8	0	0	0
x_3	0.6	0.8	0	0.2	0	0
x_4	0.8	0	0.2	0	0.8	0.7
x_5	0.1	0	0	0.8	0	0.8
x_6	0	0	0	0.7	0.8	0

■ Important properties:

■ Symmetric matrix

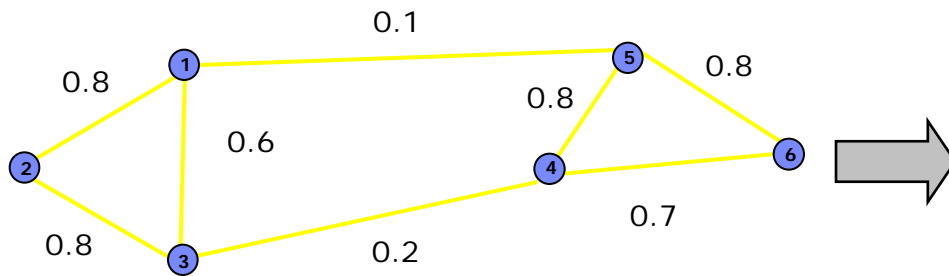
⇒ Eigenvectors are real and orthogonal

Matrix Representations (continued)

□ Degree matrix (**D**)

- $n \times n$ diagonal matrix

- $D(i,i) = \sum_j w_{ij}$: total weight of edges incident to vertex x_i



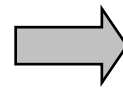
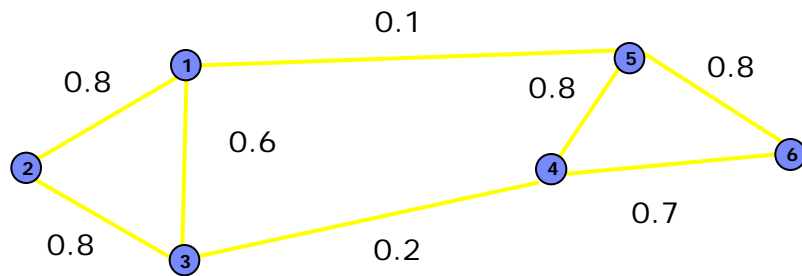
	x_1	x_2	x_3	x_4	x_5	x_6
x_1	1.5	0	0	0	0	0
x_2	0	1.6	0	0	0	0
x_3	0	0	1.6	0	0	0
x_4	0	0	0	2.5	0	0
x_5	0	0	0	0	1.7	0
x_6	0	0	0	0	0	1.5

- Important application:
 - Normalize adjacency matrix

Matrix Representations (continued)

□ **Laplacian matrix** $L = D - A$

■ $n \times n$ symmetric matrix



	x_1	x_2	x_3	x_4	x_5	x_6
x_1	1.5	-0.8	-0.6	0	-0.1	0
x_2	-0.8	1.6	-0.8	0	0	0
x_3	-0.6	-0.8	1.6	-0.2	0	0
x_4	-0.8	0	-0.2	2.5	-0.8	-0.7
x_5	-0.1	0	0	0.8	1.7	-0.8
x_6	0	0	0	-0.7	-0.8	1.5

Important properties:

- Eigenvalues are non-negative real numbers
- Eigenvectors are real and orthogonal
- Eigenvalues and eigenvectors provide an insight into the connectivity of the graph...

Find An Optimal Min-Cut (Hall'70, Fiedler'73)

- Express a bi-partition (A, B) as a vector $p_i = \begin{cases} +1 & \text{if } x_i \in A \\ -1 & \text{if } x_i \in B \end{cases}$
- We can minimize the cut of the partition by finding a non-trivial vector p that minimizes the function

$$f(p) = \sum_{i, j \in V} w_{ij} (p_i - p_j)^2 = p^T L p$$

Laplacian matrix

- The *Rayleigh Theorem* shows:
 - The minimum value for $f(p)$ is given by the 2nd smallest eigenvalue of the Laplacian L .
 - The optimal solution for p is given by the corresponding eigenvector \mathbf{x}_2 , referred as the *Fiedler Vector*.

So far...

- How can we define a “good” partition of a graph?
 - Minimize a given graph cut criterion.

- How can we efficiently identify such a partition?
 - Approximate using information provided by the eigenvalues and eigenvectors of a graph.

⇒ **Spectral Clustering** (Simon et. al, '90)

Spectral Clustering Algorithms

□ Three basic stages:

1. Pre-processing

- Construct a matrix representation of the dataset.

2. Decomposition

- Compute eigenvalues and eigenvectors of the matrix.
- Map each point to a lower-dimensional representation based on one or more eigenvectors.

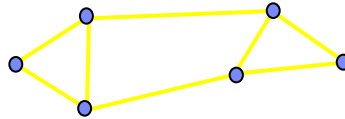
3. Grouping

- Assign points to two or more clusters, based on the new representation.

Spectral Bi-partitioning Algorithm (Simon,'90)

1. Pre-processing

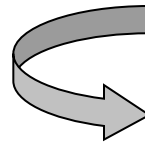
- Build Laplacian matrix L of the graph



x_1	1.5	-0.8	-0.6	0	-0.1	0
x_2	-0.8	1.6	-0.8	0	0	0
x_3	-0.6	-0.8	1.6	-0.2	0	0
x_4	-0.8	0	-0.2	2.5	-0.8	-0.7
x_5	-0.1	0	0	0.8	1.7	-0.8
x_6	0	0	0	-0.7	-0.8	1.5

2. Decomposition

- Find eigenvalues λ and eigenvectors X of the matrix L
- Map vertices to corresponding components of λ_2



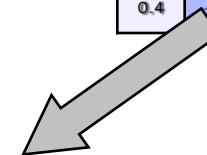
$A =$

0.0
0.4
2.2
2.3
2.5
3.0

$X =$

0.4	0.2	0.1	0.4	-0.2	-0.9
0.4	0.2	0.1	-0.1	0.4	0.3
0.4	0.2	-0.2	0.0	-0.2	0.6
0.4	-0.4	0.9	0.2	-0.4	-0.6
0.4	-0.7	-0.4	-0.8	-0.6	-0.2
0.4	-0.7	-0.2	0.5	0.8	0.9

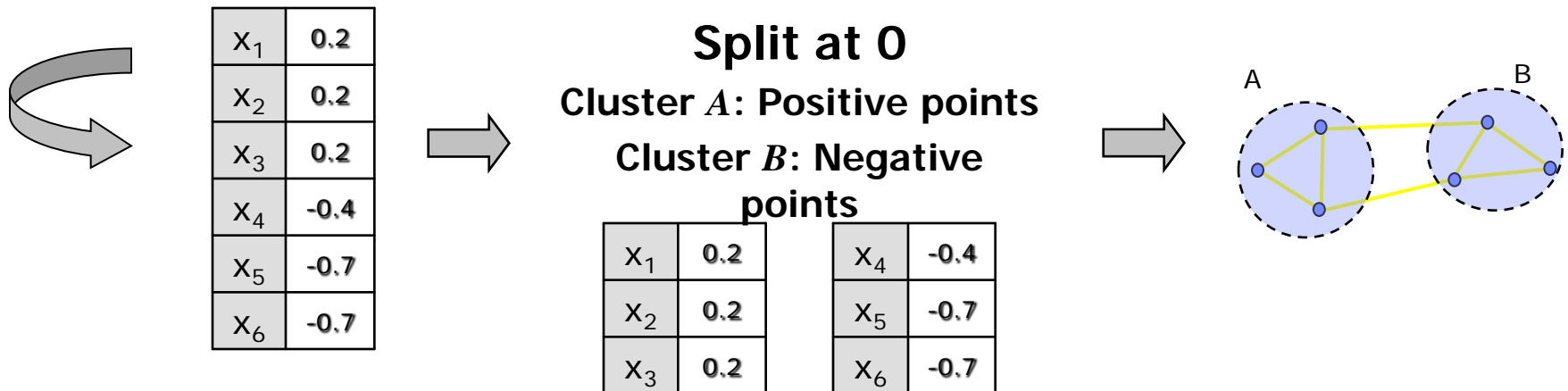
x_1	0.2
x_2	0.2
x_3	0.2
x_4	-0.4
x_5	-0.7
x_6	-0.7



How do we find the clusters?

Spectral Bi-partitioning (continued)

- Grouping
 - Sort components of reduced 1-dimensional vector.
 - Identify clusters by splitting the sorted vector in two.
- How to choose a splitting point?
 - Naïve approaches:
 - Split at 0, mean or median value
 - More expensive approaches
 - Attempt to minimize normalised cut criterion in 1-dimension



K -Eigenvector Clustering

□ **K-eigenvector Algorithm** (Ng et al., '01)

1. Pre-processing

- Construct the scaled adjacency matrix

$$A' = D^{-1/2} A D^{-1/2}$$

2. Decomposition

- Find the eigenvalues and eigenvectors of A' .
- Build embedded space from the eigenvectors corresponding to the k largest eigenvalues.

3. Grouping

- Apply k -means to reduced $n \times k$ space to produce k clusters.

Aside: How to select k ?

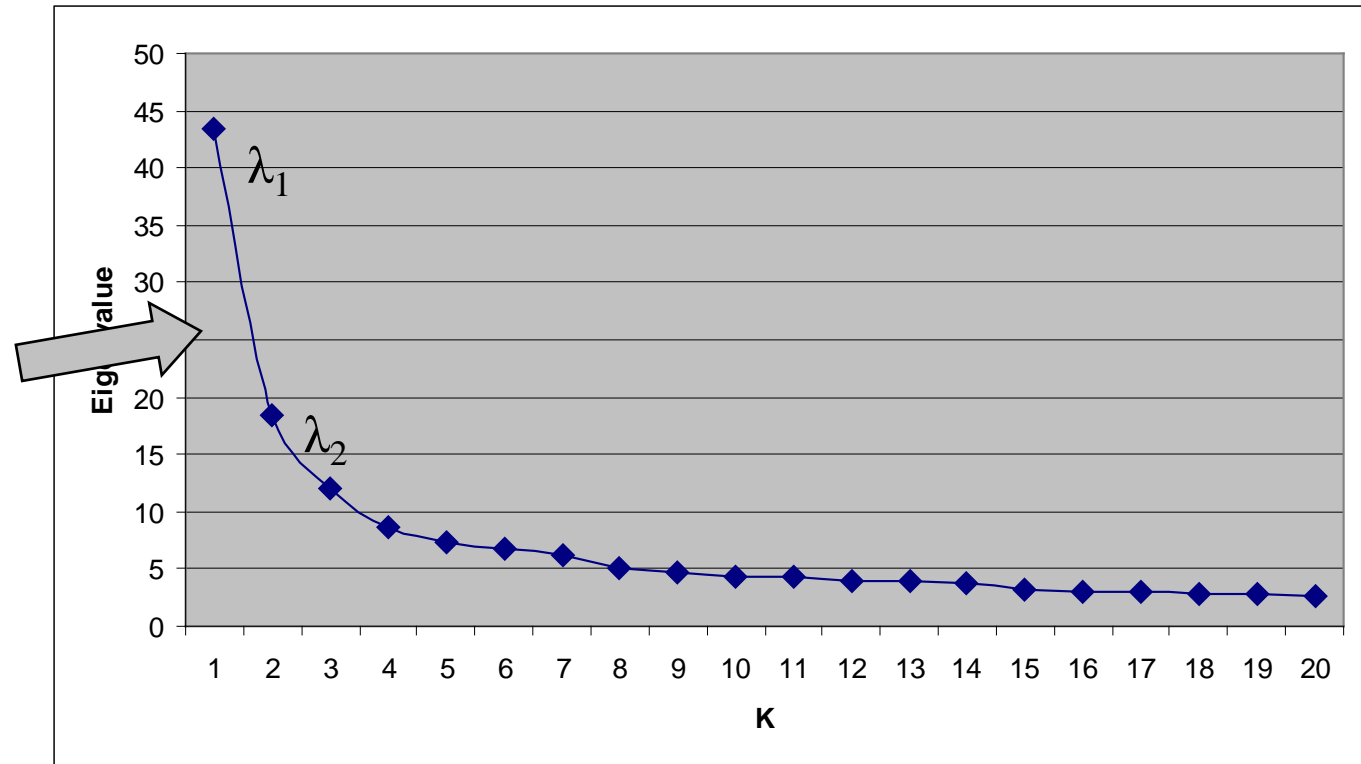
- *Eigengap*: the difference between two consecutive eigenvalues.
- Most stable clustering is generally given by the value k that maximizes the expression

$$\Delta_k = |\lambda_k - \lambda_{k-1}|$$

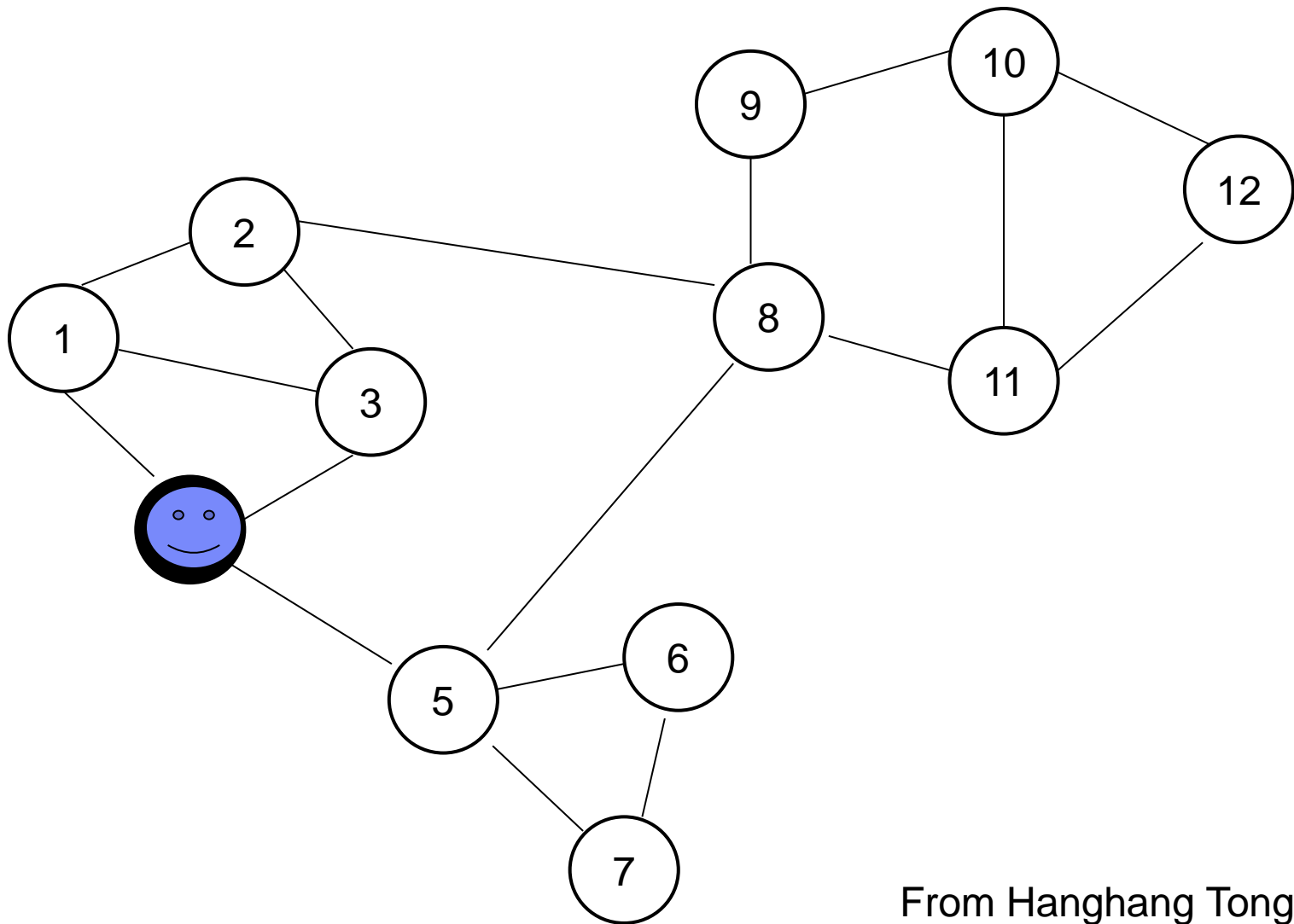
Largest eigenvalues
of Cisi/Medline data

$$\max \Delta_k = |\lambda_2 - \lambda_1|$$

⇒ Choose $k=2$

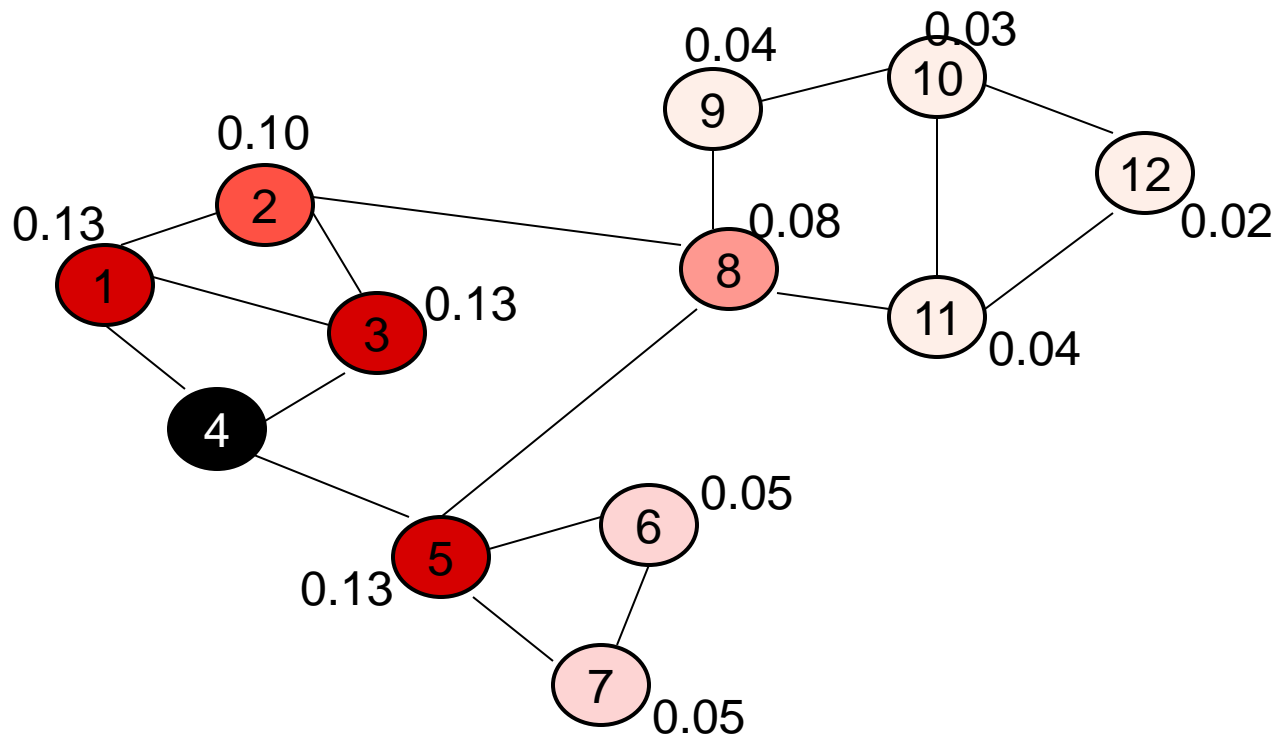


Random Walk with Restart



From Hanghang Tong

Random Walk with Restart



	Node 4
Node 1	0.13
Node 2	0.10
Node 3	0.13
Node 4	0.22
Node 5	0.13
Node 6	0.05
Node 7	0.05
Node 8	0.08
Node 9	0.04
Node 10	0.03
Node 11	0.04
Node 12	0.02

Nearby nodes, higher scores

More red, more relevant

ranking vector

Random Walk with Restart

- The walk distribution satisfies a simple equation:

$$\boldsymbol{\pi} = (1 - c)\mathbf{P}\boldsymbol{\pi} + c\mathbf{e}$$

- \mathbf{P} : Transition matrix
- c : Restart probability
- \mathbf{e} : Start node
- $\boldsymbol{\pi}$: Ranking vector
- Solution: $\boldsymbol{\pi} = c(\mathbf{I} - (1 - c)\mathbf{P})^{-1}\mathbf{e}$

Example of RWR

Iterative update until convergence

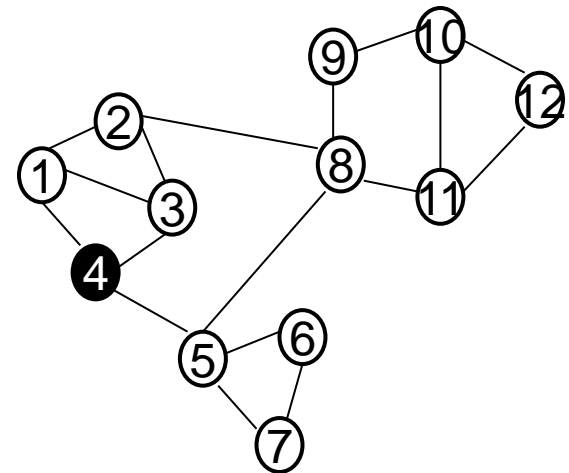
$$\boldsymbol{\pi}^t = (1 - c)\mathbf{P}\boldsymbol{\pi}^{t-1} + c\mathbf{e}$$

$$\begin{pmatrix} 0.13 \\ 0.10 \\ 0.13 \\ 0.22 \\ 0.13 \\ 0.05 \\ 0.05 \\ 0.08 \\ 0.04 \\ 0.03 \\ 0.04 \\ 0.02 \end{pmatrix} = 0.9 \times \begin{pmatrix} 0 & 1/3 & 1/3 & 1/3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/3 & 0 & 1/3 & 0 & 0 & 0 & 0 & 1/4 & 0 & 0 & 0 & 0 \\ 1/3 & 1/3 & 0 & 1/3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/3 & 0 & 1/3 & 0 & 1/4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/3 & 0 & 1/2 & 1/2 & 1/4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/4 & 0 & 1/2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/4 & 1/2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/3 & 0 & 0 & 1/4 & 0 & 0 & 0 & 1/2 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/4 & 0 & 1/3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/2 & 0 & 1/3 & 1/2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/4 & 0 & 1/3 & 0 & 1/2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/3 & 1/3 & 0 \end{pmatrix} \begin{pmatrix} 0.13 \\ 0.10 \\ 0.13 \\ 0.22 \\ 0.13 \\ 0.05 \\ 0.05 \\ 0.08 \\ 0.04 \\ 0.03 \\ 0.04 \\ 0.02 \end{pmatrix} + 0.1 \times \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

$n \times 1$

$n \times n$

$n \times 1$



References

- S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. In *SIGMOD*, 1998.
- B. Gallagher and T. Eliassi-Rad. An evaluation of experimental methodology for classifiers of relational data. In *Workshop MGCS in ICDM*, 2007.
- B. Gallagher and T. Eliassi-Rad. Leveraging network structure to infer missing values in relational data. *Technical Report UCRL-TR-231993, Lawrence Livermore National Laboratory*, 2007.
- L. Getoor. Link-based classification. Chapter in *Advanced Methods for Knowledge Discovery from Complex Data*, 2005.
- D. Jensen and J. Neville. Linkage and autocorrelation cause feature selection bias in relational learning. In *ICML*, 2002.
- D. Jensen, J. Neville, and B. Gallagher. Why collective inference improves relational classification. In *KDD*, 2004.
- A. Knobbe, M. deHaas, and A. Siebes. Propositionalization and aggregates. In *PKDD*, 2001.
- Z. Kou and W. W. Cohen. Stacked graphical models for efficient inference in markov random fields. In *SDM*, 2007.
- S. Kramer, N. Lavrac, and P. Flach. Propositionalization approaches to relational data mining. Chapter in *Relational Data Mining*, 2001.
- Q. Lu and L. Getoor. Link based classification. In *ICML*, 2003.

References

- S. Macskassy and F. Provost. Classification in networked data: A toolkit and a univariate case study. *JMLR*, 2007.
- S. Macskassy. Improving learning in networked data by combining explicit and mined links. In *AAAI*, 2007.
- L. McDowell, K. M. Gupta, and D. W. Aha. Cautious inference in collective classification. In *AAAI*, 2007.
- J. Neville and D. Jensen. Iterative classification in relational data. In *Workshop on SRL, AAAI*, 2000.
- J. Neville and D. Jensen. Bias/variance analysis for relational domains. In *ILP*, 2007.
- C. Perlich and F. Provost. Distribution based aggregation for relational learning with identifier attributes. In *Machine Learning Journal*, 2006.
- A. Popescul and L. Ungar. Cluster-based concept invention for statistical relational learning. In *KDD*, 2004.
- P. Sen, G. M. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad. Collective classification in network data. *Technical Report CS-TR-4905, University of Maryland, College Park*, 2008.
- B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In *UAI*, 2002.
- J. Yedidia, W. T. Freeman, and Y. Weiss. Generalized belief propagation. In *NIPS*, 2002.