COMP4222 Machine Learning with Structured Data

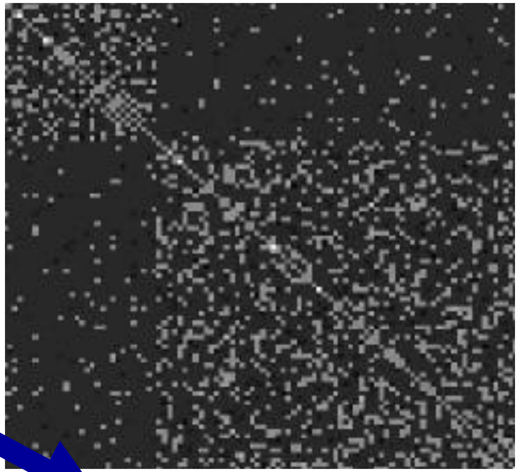
Graph Laplacian

Yangqiu Song

Slides credits: Jianbo Shi and Alireza Tavakkoli

Graph-based Image Segmentation

Image (I)

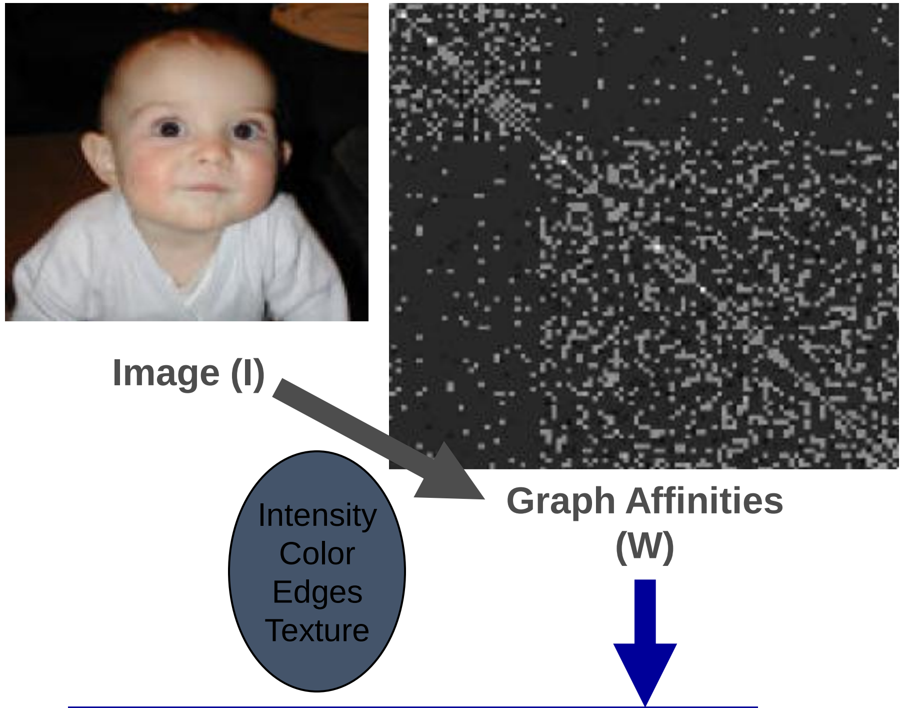


Graph Affinities (W)

Intensity
Color
Edges
Texture

Slide from Timothee Cour (http://www.seas.upenn.edu/~timothee)

Graph-based Image Segmentation

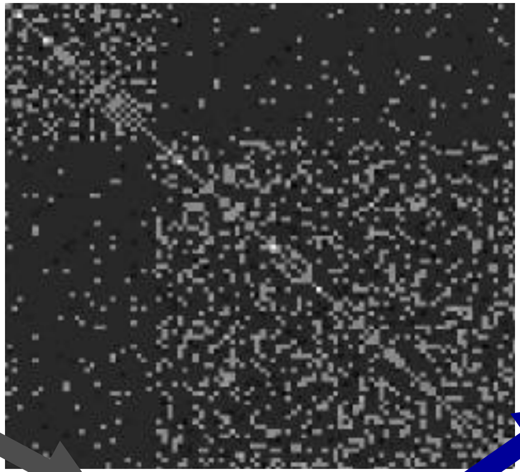


$$
Ncut(A,B) = cut(A,B)\left(\frac{1}{vol(A)} + \frac{1}{vol(B)}\right)
$$

Slide from Timothee Cour (http://www.seas.upenn.edu/~timothee)

Graph-based Image Segmentation

Image (I)



Graph Affinities (W)

Eigenvector X(W)

Intensity
Color
Edges
Texture

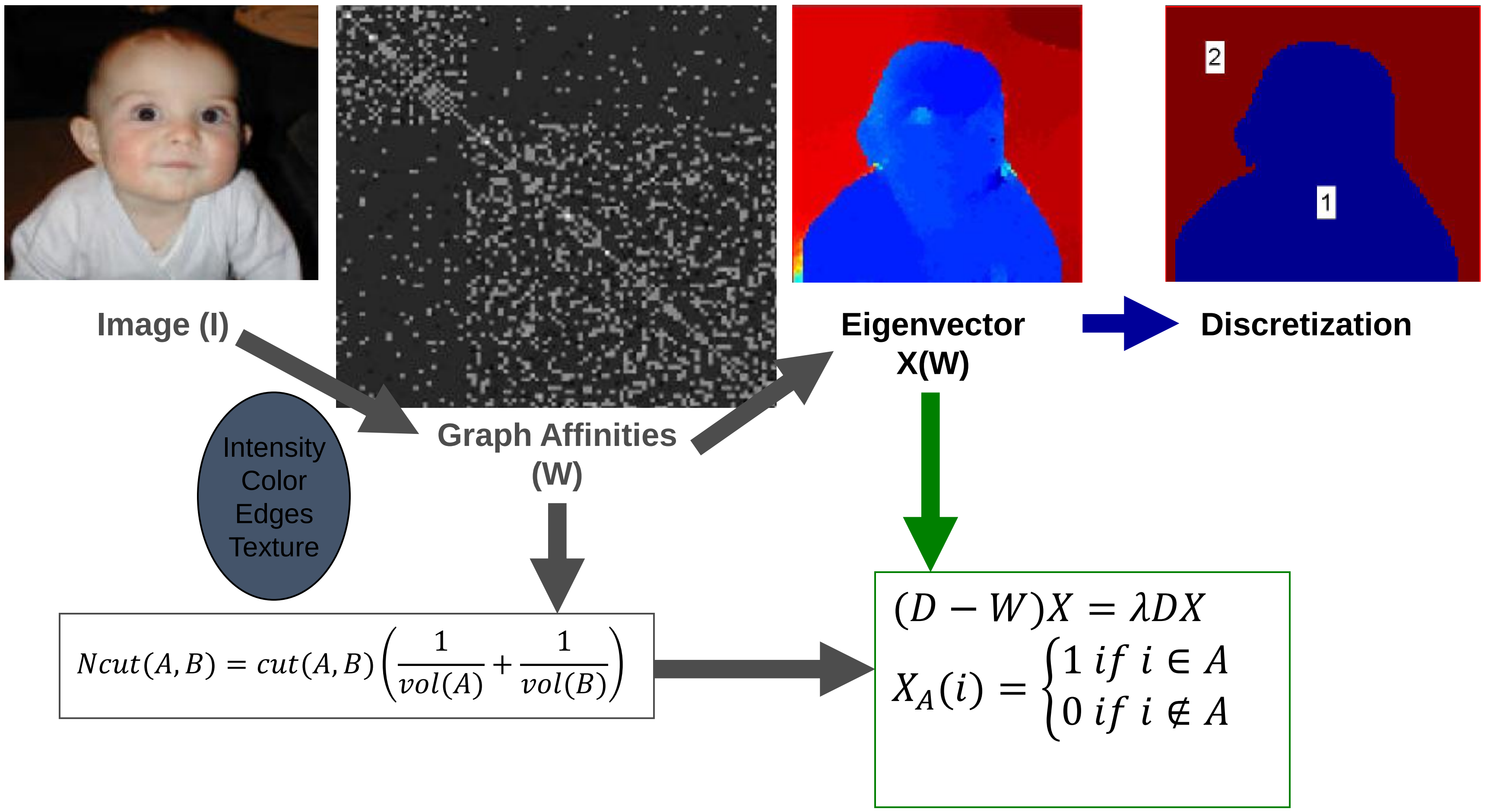
$$Ncut(A,B) = cut(A,B)\left(\frac{1}{vol(A)} + \frac{1}{vol(B)}\right)$$

$$(D-W)X = \lambda DX$$

$$X_A(i) = \begin{cases} 1 \; if \; i \in A \\ 0 \; if \; i \notin A \end{cases}$$

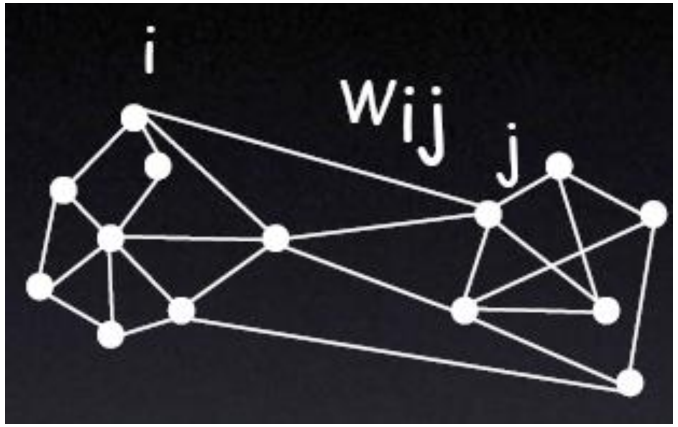
Slide from Timothee Cour (http://www.seas.upenn.edu/~timothee)

Graph-based Image Segmentation

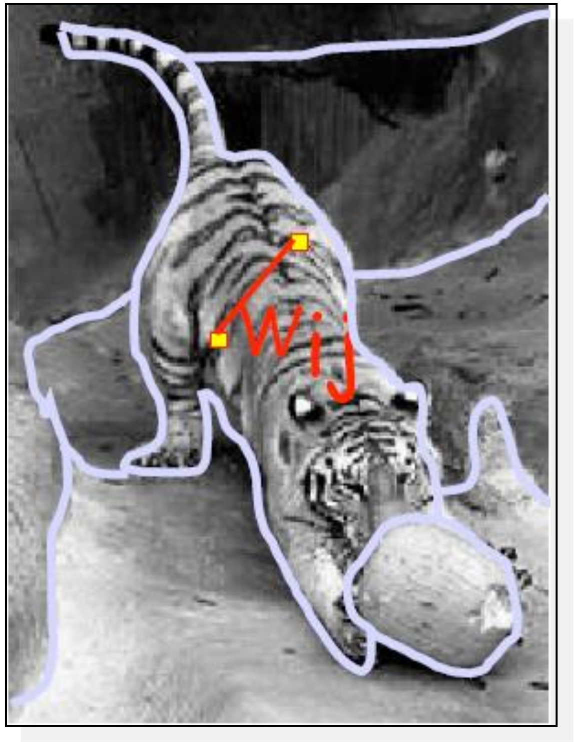


Slide from Timothee Cour (http://www.seas.upenn.edu/~timothee)

Graph-based Image Segmentation



G = {V,E}



V: graph nodes
E: edges connection nodes

→

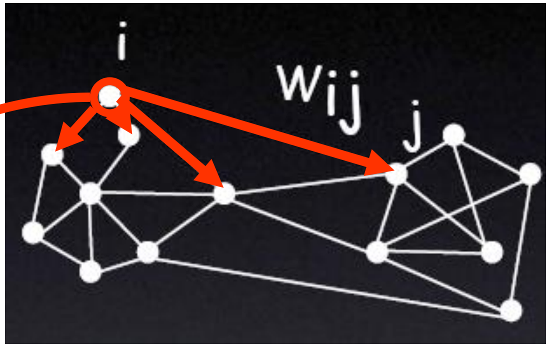
Pixels
Pixel similarity

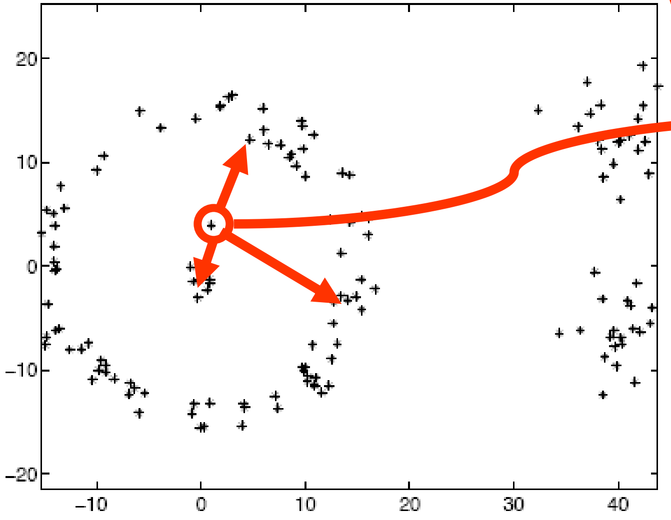
Slides from Jianbo Shi

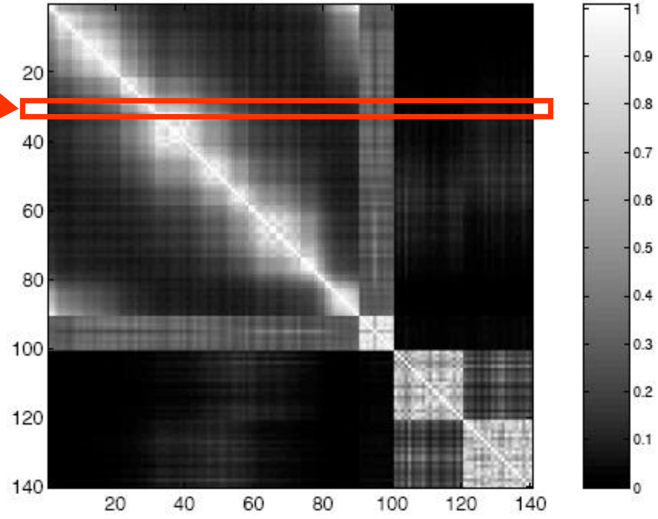
Graph Terminology

- **Similarity matrix:** $W = [w_{i,j}]$

$$w_{i,j} = e^{\frac{-\left\|X_{(i)} - X_{(j)}\right\|_2^2}{\sigma_X^2}}$$

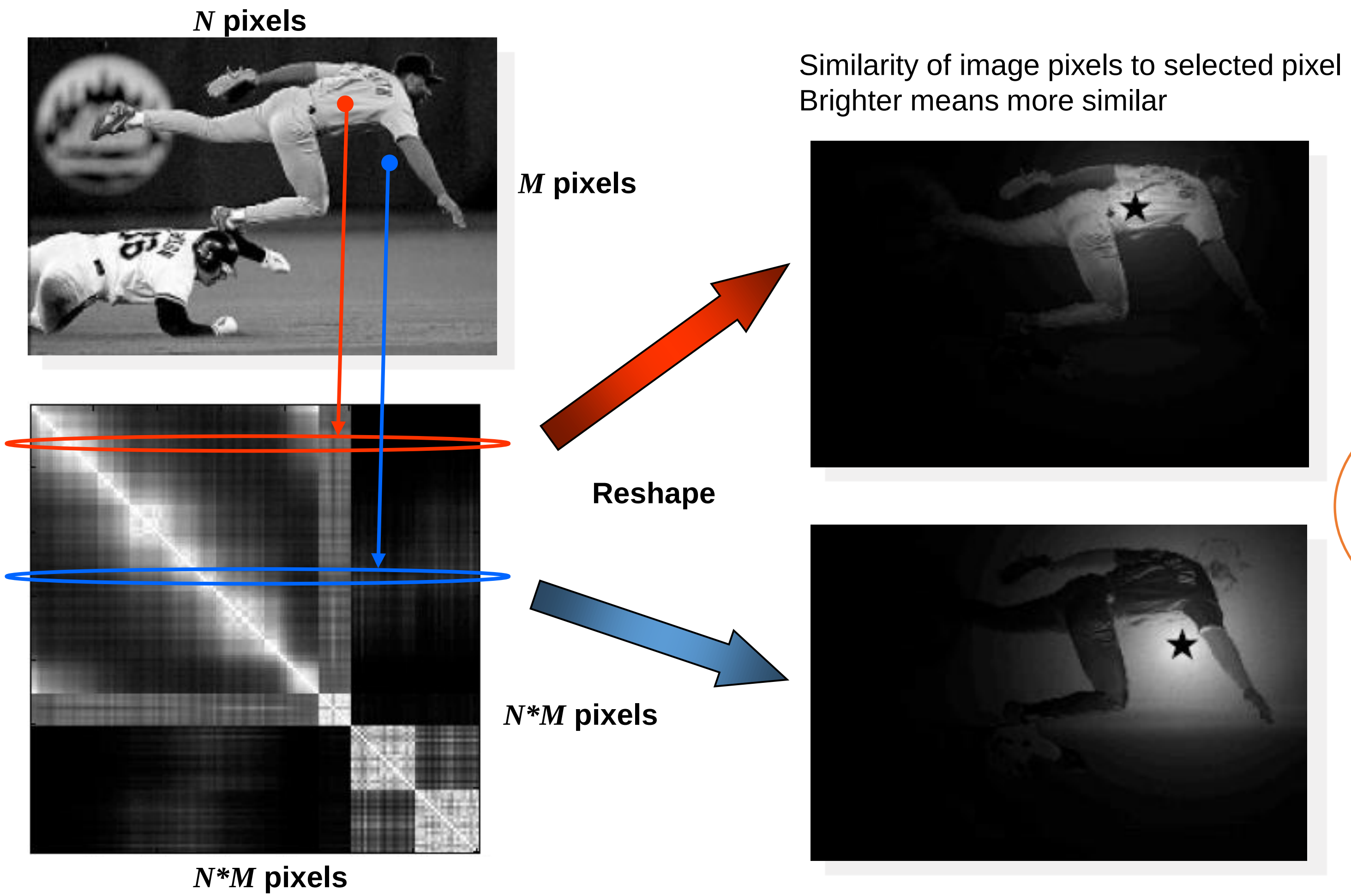






Slides from Jianbo Shi

Affinity Matrix

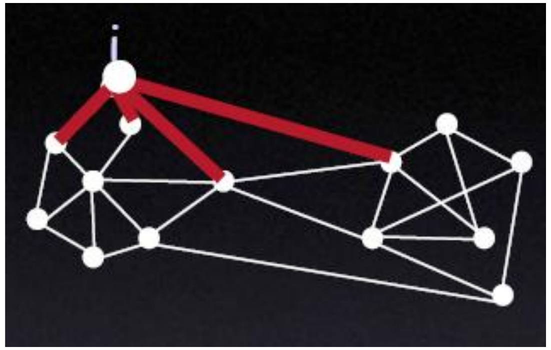


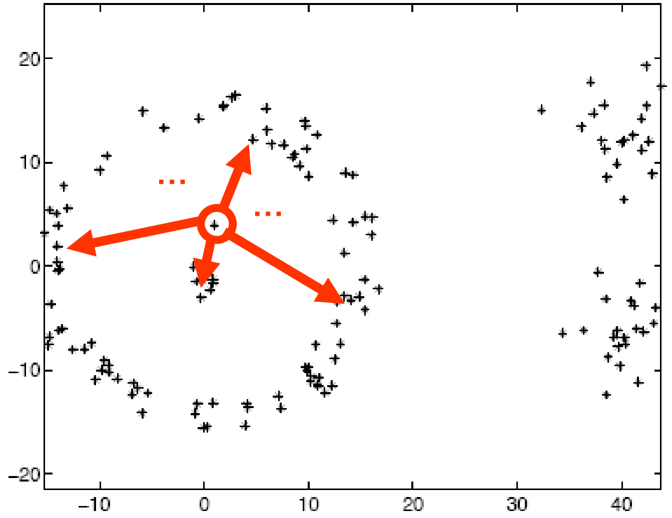
Warning
the size of W is quadratic
with the number
of parameters!

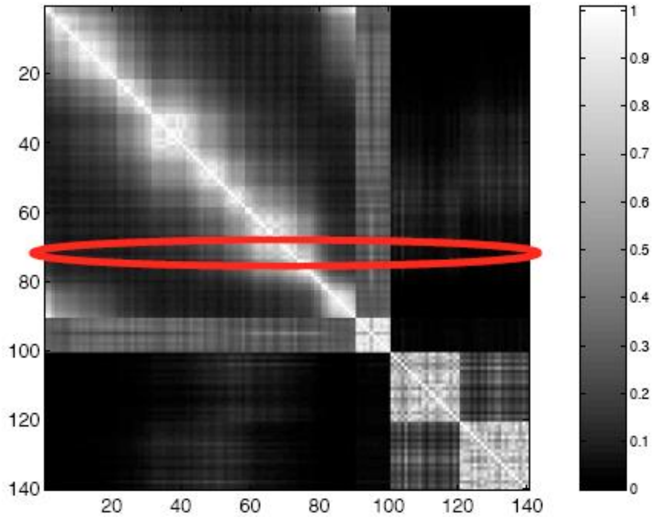
Graph Terminology

- Degree of node:

$$d_i = \sum_j w_{i,j}$$





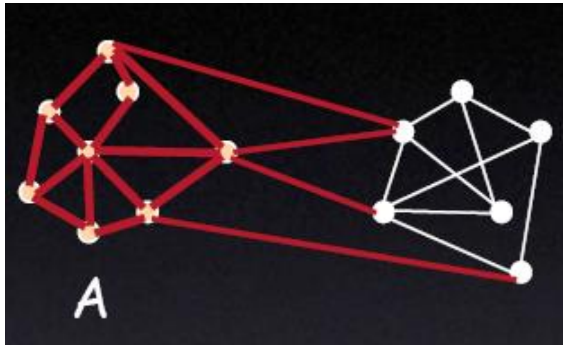


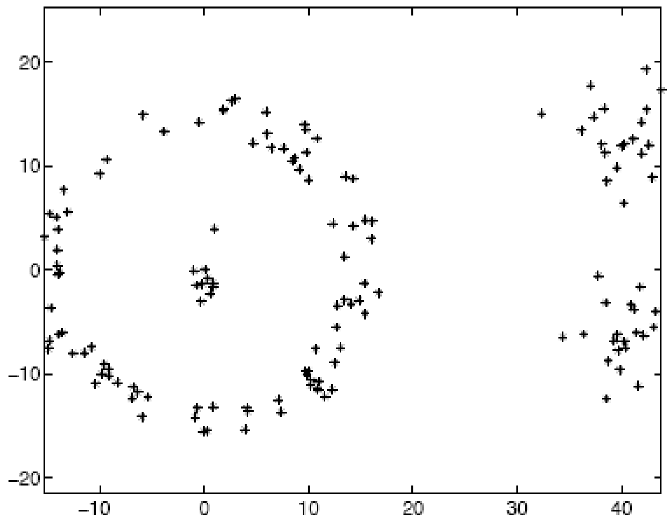
Slides from Jianbo Shi

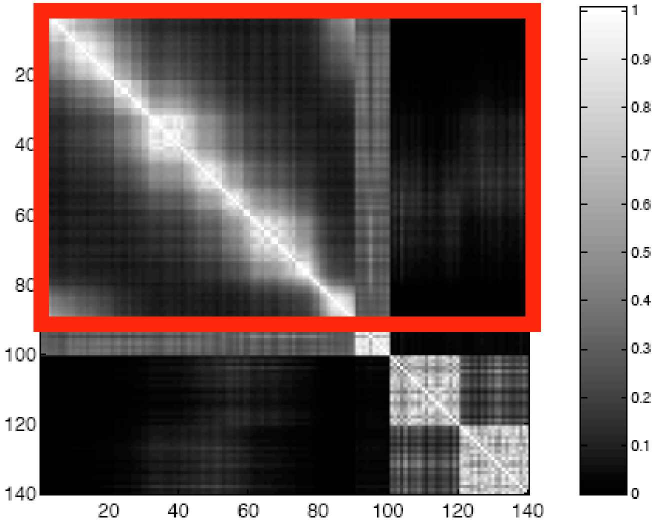
Graph Terminology

- Volume of set:

$$vol(A) = \sum_{i \in A} d_i, A \subseteq V$$





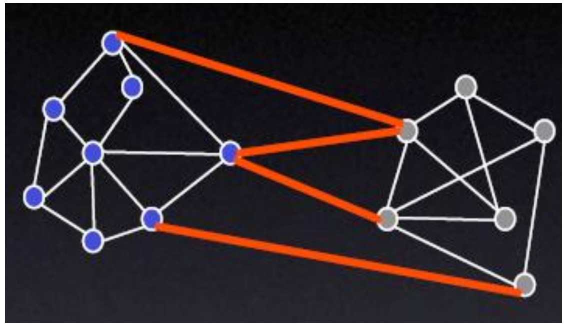


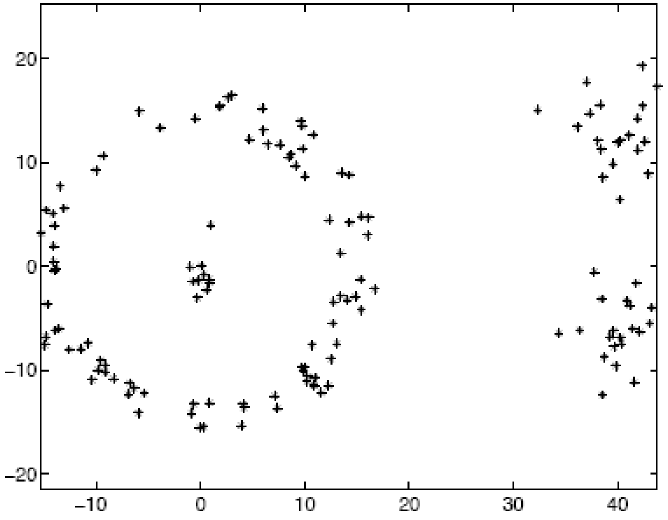
Slides from Jianbo Shi

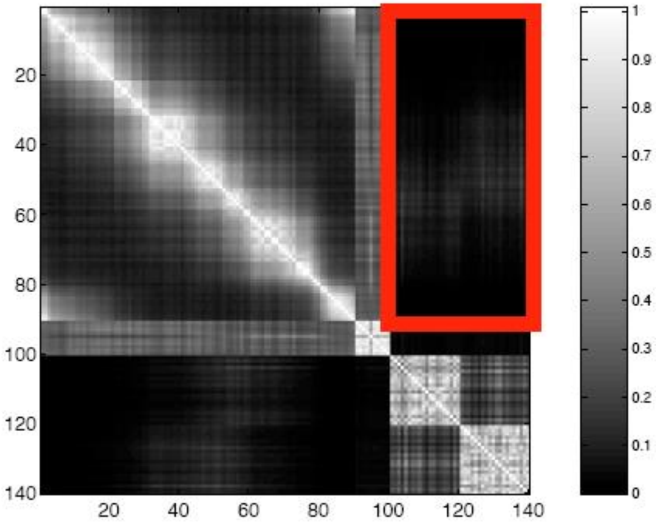
Graph Terminology

- Cuts in a graph:

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







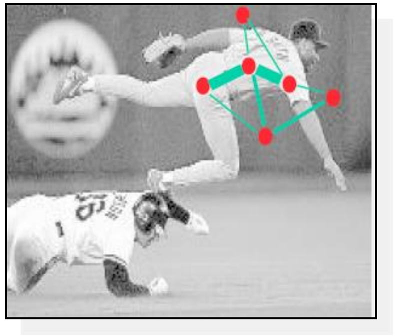
Slides from Jianbo Shi

Representation

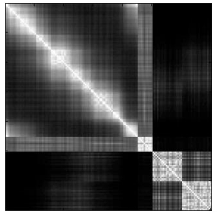
Partition matrix X:

$$X = [X_1, \dots, X_K]$$

$$X = \begin{array}{cc} \text{segments} & \\ \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} & \text{pixels} \end{array}$$



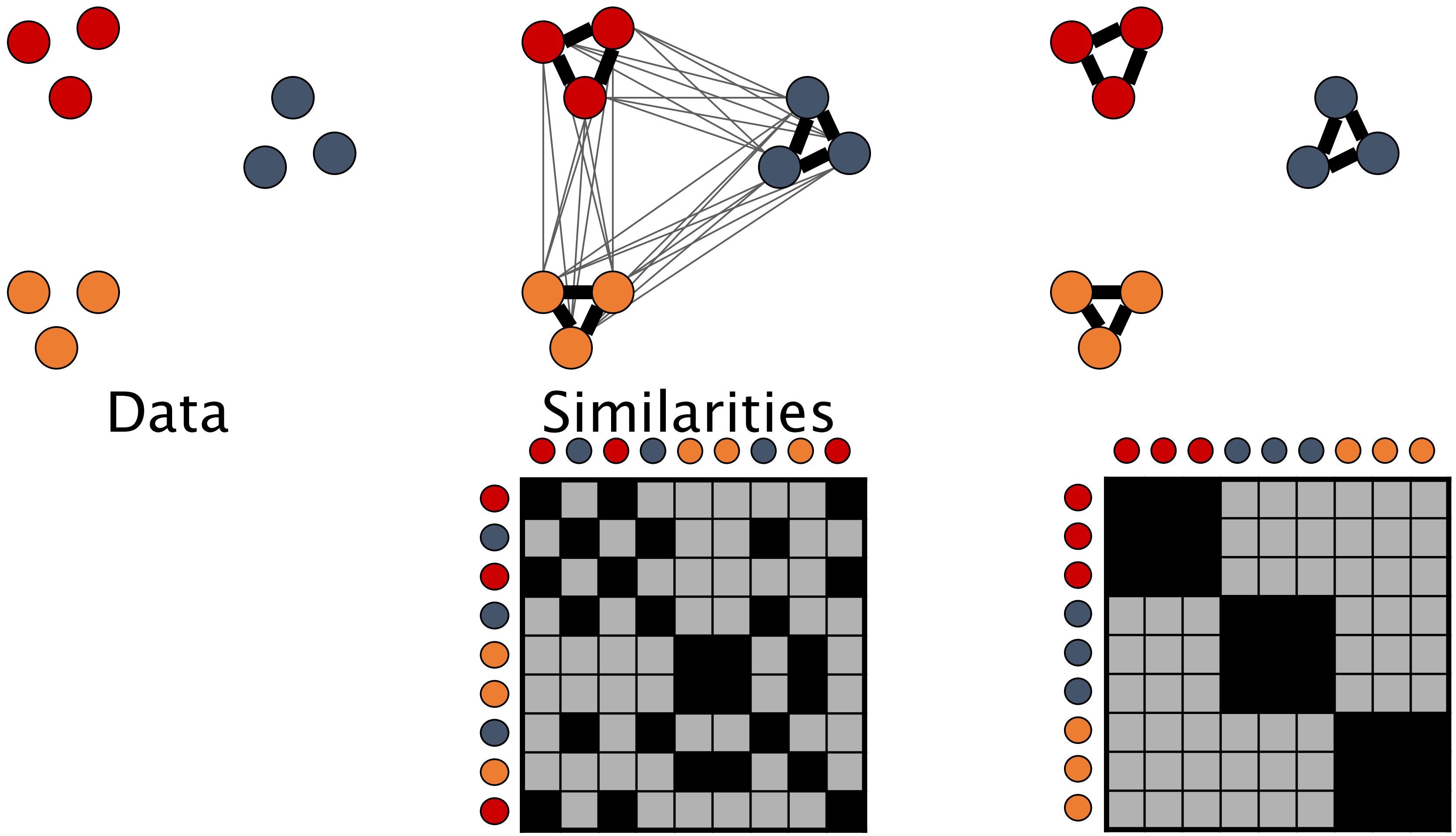
Pair-wise similarity matrix W: $W(i,j) = Sim(i,j)$



Degree matrix D: $D(i,i) = \sum_j w_{i,j}$

Laplacian matrix L: $L = D - W$

Spectral Clustering



* Slides from Dan Klein, Sep Kamvar, Chris Manning, Natural Language Group Stanford University

Eigenvectors and Blocks

- Block matrices have block eigenvectors:

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \xrightarrow{\text{eigensolver}} \underset{\lambda_1 = 2}{\begin{bmatrix} .71 \\ .71 \\ 0 \\ 0 \end{bmatrix}} \quad \underset{\lambda_2 = 2}{\begin{bmatrix} 0 \\ 0 \\ .71 \\ .71 \end{bmatrix}} \qquad \lambda_3 = 0 \quad \lambda_4 = 0$$

- Near-block matrices have near-block eigenvectors:

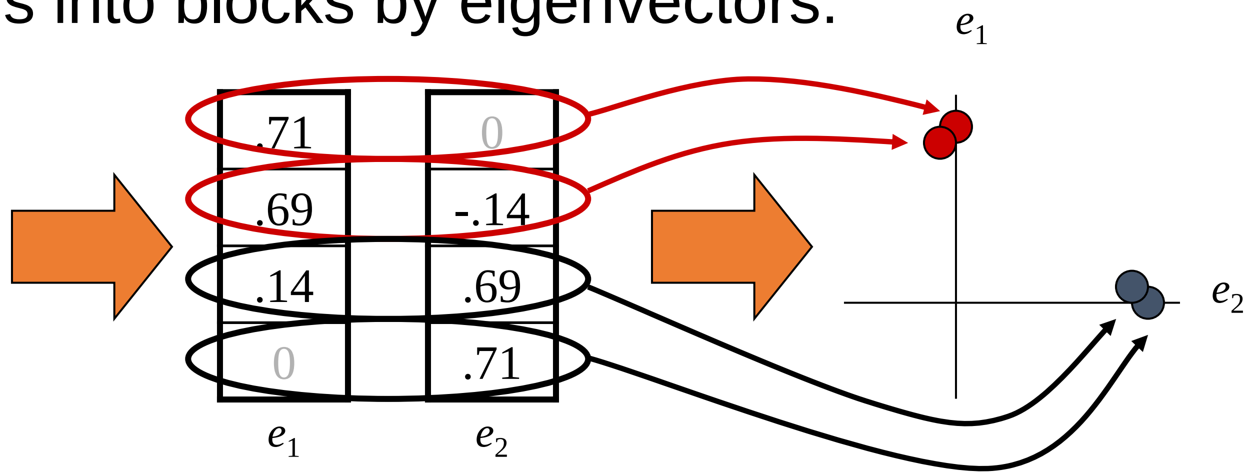
$$\begin{bmatrix} 1 & 1 & .2 & 0 \\ 1 & 1 & 0 & -.2 \\ .2 & 0 & 1 & 1 \\ 0 & -.2 & 1 & 1 \end{bmatrix} \xrightarrow{\text{eigensolver}} \underset{\lambda_1 = 2.02}{\begin{bmatrix} .71 \\ .69 \\ .14 \\ 0 \end{bmatrix}} \quad \underset{\lambda_2 = 2.02}{\begin{bmatrix} 0 \\ -.14 \\ .69 \\ .71 \end{bmatrix}} \qquad \lambda_3 = -0.02 \quad \lambda_4 = -0.02$$

* Slides from Dan Klein, Sep Kamvar, Chris Manning, Natural Language Group Stanford University

Spectral Space

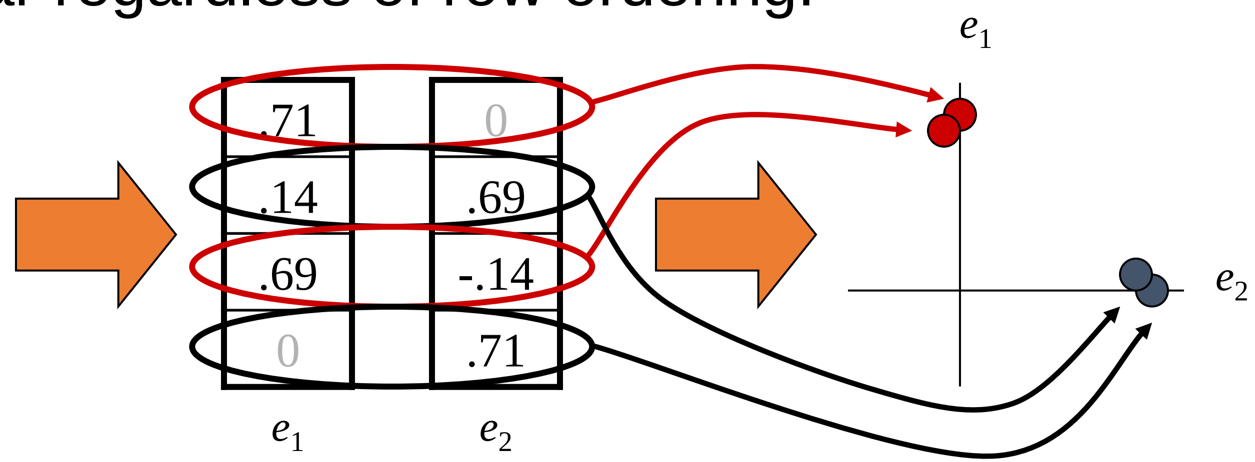
- Can put items into blocks by eigenvectors:

1	1	.2	0
1	1	0	-.2
.2	0	1	1
0	-.2	1	1



- Clusters clear regardless of row ordering:

1	.2	1	0
.2	1	0	1
1	0	1	-.2
0	1	-.2	1

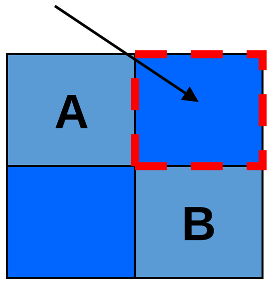


Min Cut vs Normalized Cut

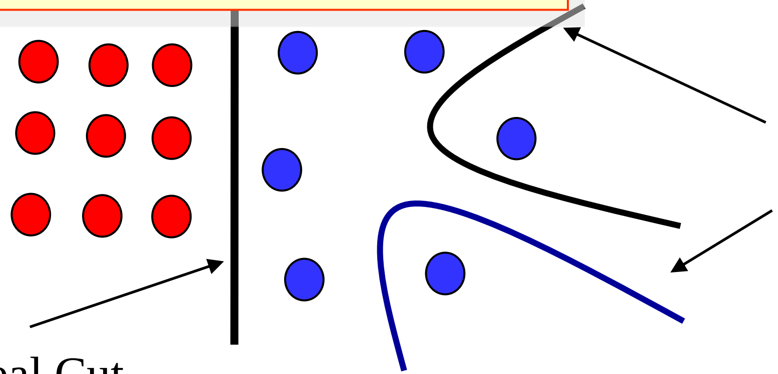
Minimum Cut

- Criterion for partition:

$$\min cut(A,B) = \min_{A,B} \sum_{u\in A, v\in B} w(u,v)$$



Problem!
Weight of cut is directly proportional to the number of edges in the cut.

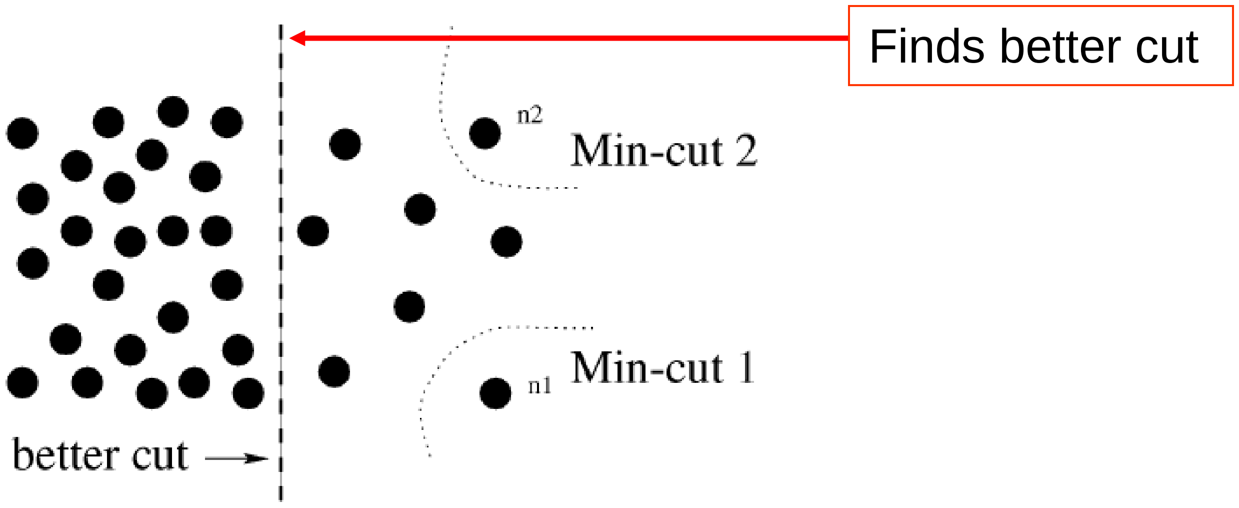


First proposed by Wu and Leahy

Normalized Cut

Normalized cut or balanced cut:

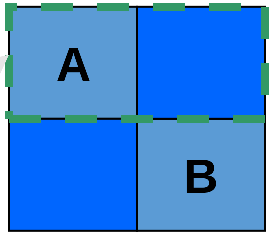
$$Ncut(A,B) = cut(A,B)\left(\frac{1}{vol(A)} + \frac{1}{vol(B)}\right)$$



Normalized Cut

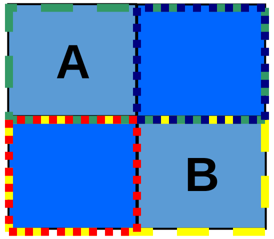
- Volume of set (or association):

$$vol(A) = assoc(A,V) = \sum_{u \in A, t \in V} w(u,t)$$



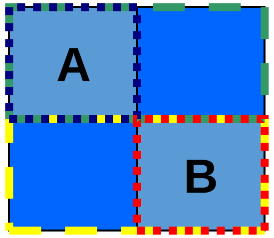
- Define normalized cut: "a fraction of the total edge connections to all the nodes in the graph":

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)}$$



- Define normalized association: "how tightly on average nodes within the cluster are connected to each other"

$$Nassoc(A,B) = \frac{assoc(A,A)}{assoc(A,V)} + \frac{assoc(B,B)}{assoc(B,V)}$$



Observations(I)

- Maximizing $Nassoc$ is the same as minimizing $Ncut$, since they are related:

$$Ncut(A,B) = 2 - Nassoc(A,B)$$

- How to minimize $Ncut$?
  - Transform $Ncut$ equation to a matricial form.
  - After simplifying:

$$D(i,i) = \sum_j W(i,j)$$

$$\min_x N\,cut(x) = \min_y \frac{y^T(D - W)y}{y^T D y}$$

Rayleigh quotient

NP-Hard!

y's values are quantized

Subject to: $y^T D 1 = 0$

Observations(II)

- Instead, relax into the continuous domain by solving generalized eigenvalue system:

$$\min_y \left(y^T (D - W) y\right) \text{ subject to } \left(y^T D y = 1\right)$$

- Which gives: $(D - W)y = \lambda D y$
- Note that $(D - W)1 = 0$ so, the first eigenvector is y_0=1 with eigenvalue 0.
- The second smallest eigenvector is the real valued solution to this problem!!

Algorithm

1. Define a similarity function between 2 nodes. i.e.:

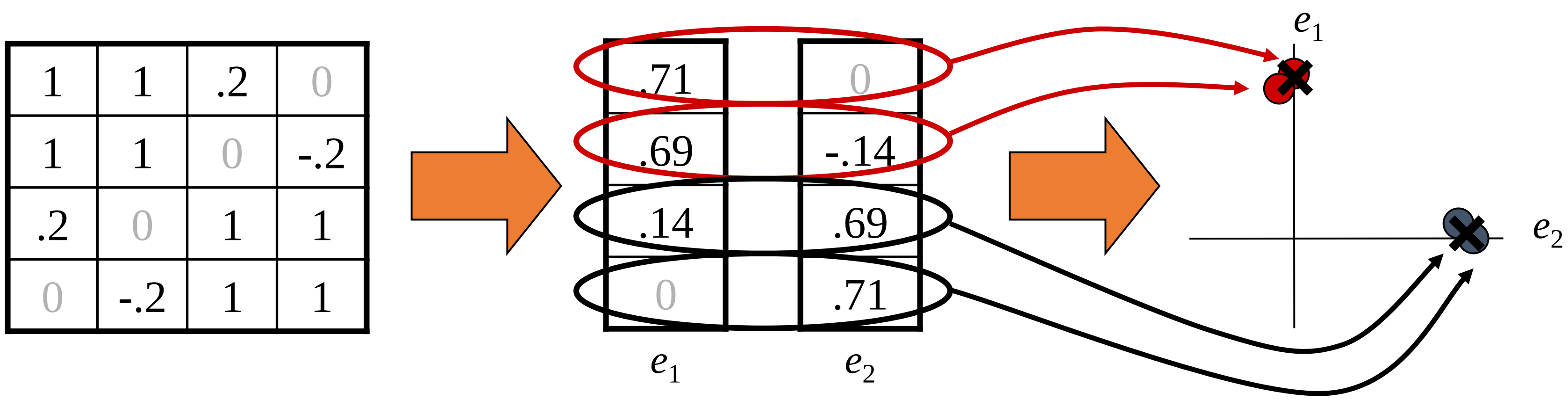
$$w_{i,j} = e^{\frac{-\left\|X_{(i)}-X_{(j)}\right\|_2^2}{\sigma_X^2}}$$

2. Compute affinity matrix (W) and degree matrix (D).
3. Solve $(D - W)y = \lambda D y$
4. Use the eigenvector with the second smallest eigenvalue to bipartition the graph.
5. Decide if re-partition current partitions.

Note: since precision requirements are low, $\boldsymbol{W}$ is very sparse and only few eigenvectors are required, the eigenvectors can be extracted very fast using Lanczos algorithm.

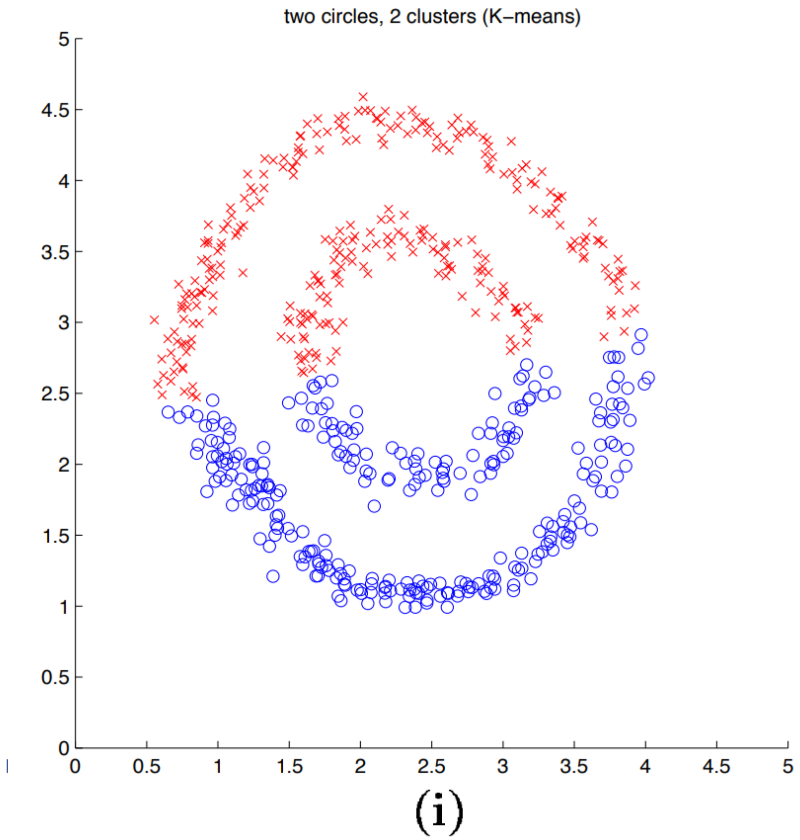
Use ***k***-eigenvectors

- We can use more eigenvectors to re-partition the graph
- Procedure: compute *k-means* with a high *k*.

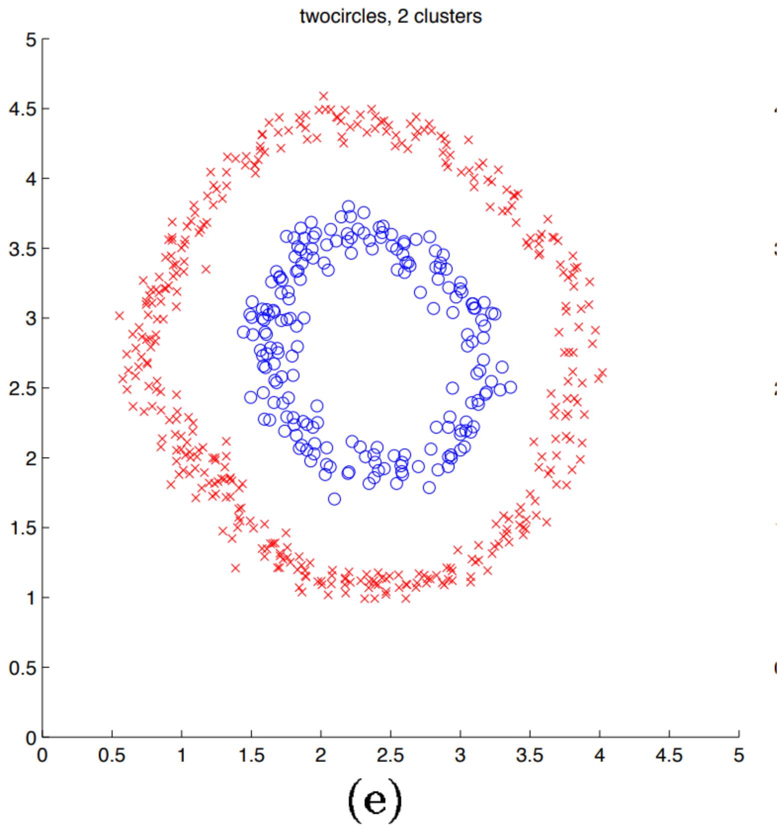


Results

Original K-means

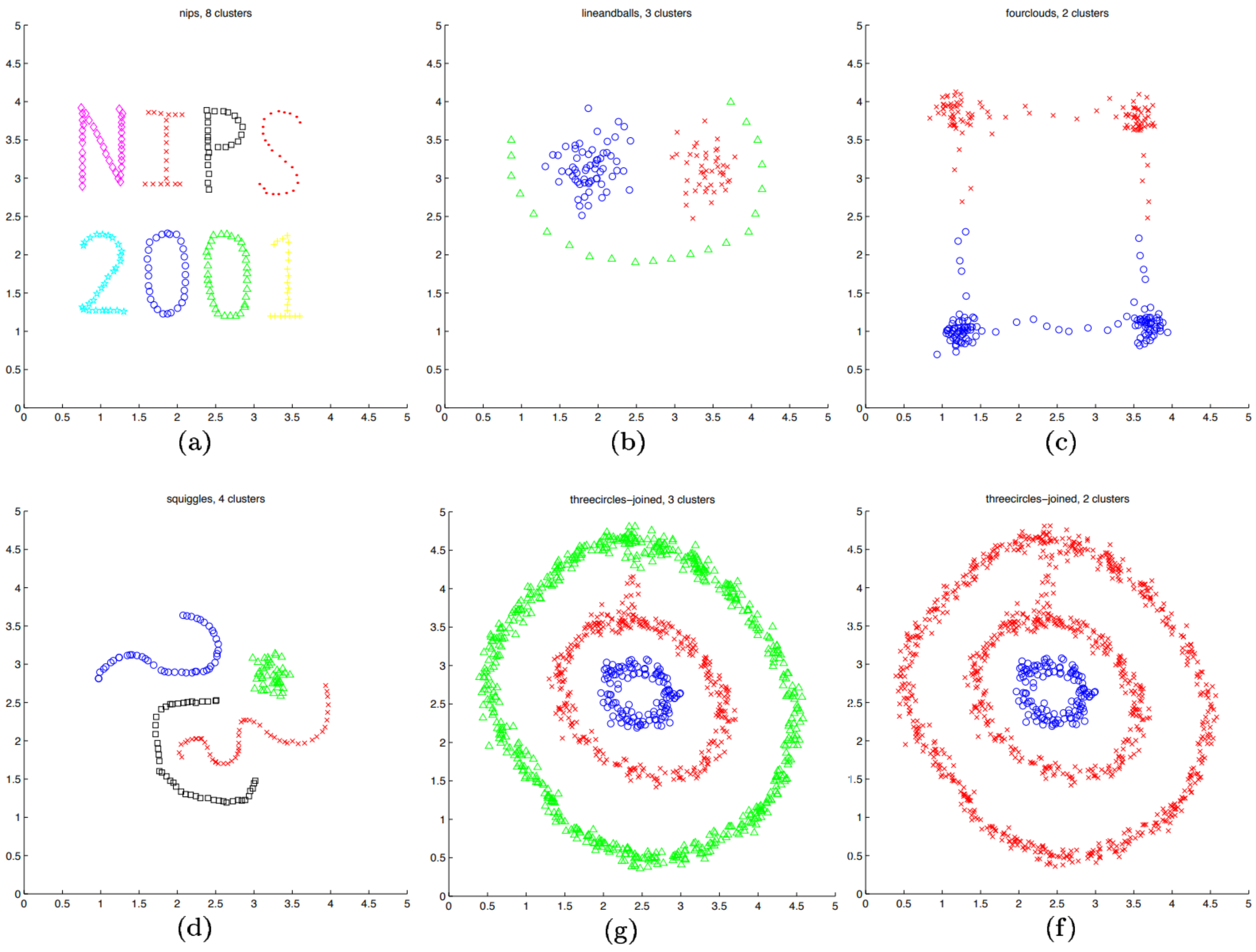


Spectral clustering + K-means



[Shi & Malik '00; Ng, Jordan, Weiss NIPS '01]

Results

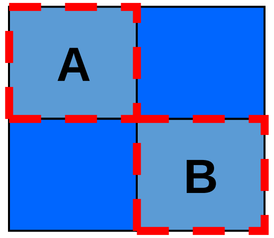


[Shi & Malik ‘00; Ng, Jordan, Weiss NIPS ‘01]

Other Methods

- Average association
  - Use the eigenvector of W associated to the biggest eigenvalue for partitioning.
  - Tries to maximize:

$$\frac{assoc(A,A)}{|A|} + \frac{assoc(B,B)}{|B|}$$



  - |A|: number of nodes in A
  - Has a bias to find tight clusters. Useful for Gaussian distributions.

Other Methods

- Average cut
  - Tries to minimize: $$\frac{cut(A,B)}{|A|}+\frac{cut(A,B)}{|B|}$$
  - Very similar to normalized cuts.
  - We cannot ensure that partitions will have a tight within-group similarity since this equation does not have the nice properties of the equation of normalized cuts.

Other Methods

Finding clumps $\longleftrightarrow$ Finding splits

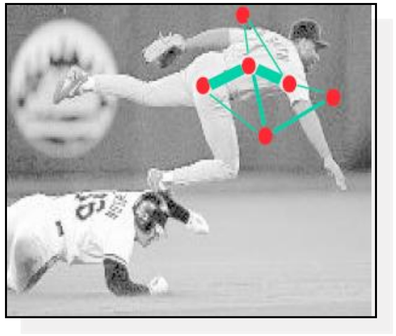
	Average association	Normalized Cut	Average cut
Discrete formulation	$\frac{asso(A,A)}{\lvert A\rvert} + \frac{asso(B,B)}{\lvert B\rvert}$	$\frac{cut(A,B)}{asso(A,V)} + \frac{cut(A,B)}{asso(B,V)}$ or $2 - \left(\frac{asso(A,A)}{asso(A,V)} + \frac{asso(B,B)}{asso(B,V)}\right)$	$\frac{cut(A,B)}{\lvert A\rvert} + \frac{cut(A,B)}{\lvert B\rvert}$
Continuous solution	$Wx = \bar{\lambda}\, x$	$(D-W)\, x = \bar{\lambda}\, D\, x$ or $W\, x = (1-\bar{\lambda})D\, x$	$(D-W)\, x = \bar{\lambda}\, x$

Summary: Representation

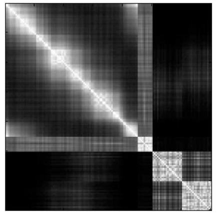
Partition matrix X:

$$X = [X_1, \ldots, X_K]$$

$$X = \begin{array}{cc} \text{segments} & \\ \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} & \text{pixels} \end{array}$$



Pair-wise similarity matrix W: $W(i,j) = aff(i,j)$



Degree matrix D: $D(i,i) = \sum_j w_{i,j}$

Laplacian matrix L: $L = D - W$

Laplacian Matrices of Graphs

- (Un-normalized) Laplacian matrix $L = D - W$
- The spectrum (eigenvalues) of L contains a lot of information about the combinatorial structure of the graph G. Leverage of this information is the object of spectral graph theory.
- For clustering, normalized graph Laplacian are usually need
  - Symmetric: $L_{sym} = D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}} = I - D^{-\frac{1}{2}}WD^{-\frac{1}{2}}$
  - Random walk: $L_{rw} = D^{-1}(D - W) = I - D^{-1}W$
  - $L_{sym} = D^{-\frac{1}{2}} L_{rw} D^{\frac{1}{2}}$

The Computation of $y^T(D-W)y$

- Assume we have an undirected graph
- Let y be a R^N dim vector, where *N* is the number of nodes
- For binary class case: $y_i = \begin{cases} 1\, if\ i \in A \\ -1\, if\ i \notin A \end{cases}$
- $2y^T(D-W)y$
  - $= 2\sum_i D_{ii} y_i^2 - 2\sum_{ij} y_i y_j w_{ij}$
  - $= 2\sum_i \left(\sum_j w_{ij}\right) y_i^2 - 2\sum_{ij} y_i y_j w_{ij}$
  - $= 2\sum_{ij} y_i^2 w_{ij} - 2\sum_{ij} y_i y_j w_{ij}$
  - $= \sum_{ij} y_i^2 w_{ij} - 2\sum_{ij} y_i y_j w_{ij} + \sum_{ij} y_j^2 w_{ij}$
  - $= \sum_{ij} w_{ij} (y_i^2 - 2y_i y_j + y_j^2)$
  - $= \sum_{ij} w_{ij} \left(y_i - y_j\right)^2$
- $\min 2y^T(D-W)y$ defines the smoothness of labels on a graph

Normalized Version

- Similarly, $y^T D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}y = \frac{1}{2}\sum_{ij} w_{ij}\left(\frac{y_i}{\sqrt{D_{ii}}} - \frac{y_j}{\sqrt{D_{jj}}}\right)^2$
- The smoothness is weighted by nodes' degrees

Laplacian as an Operator

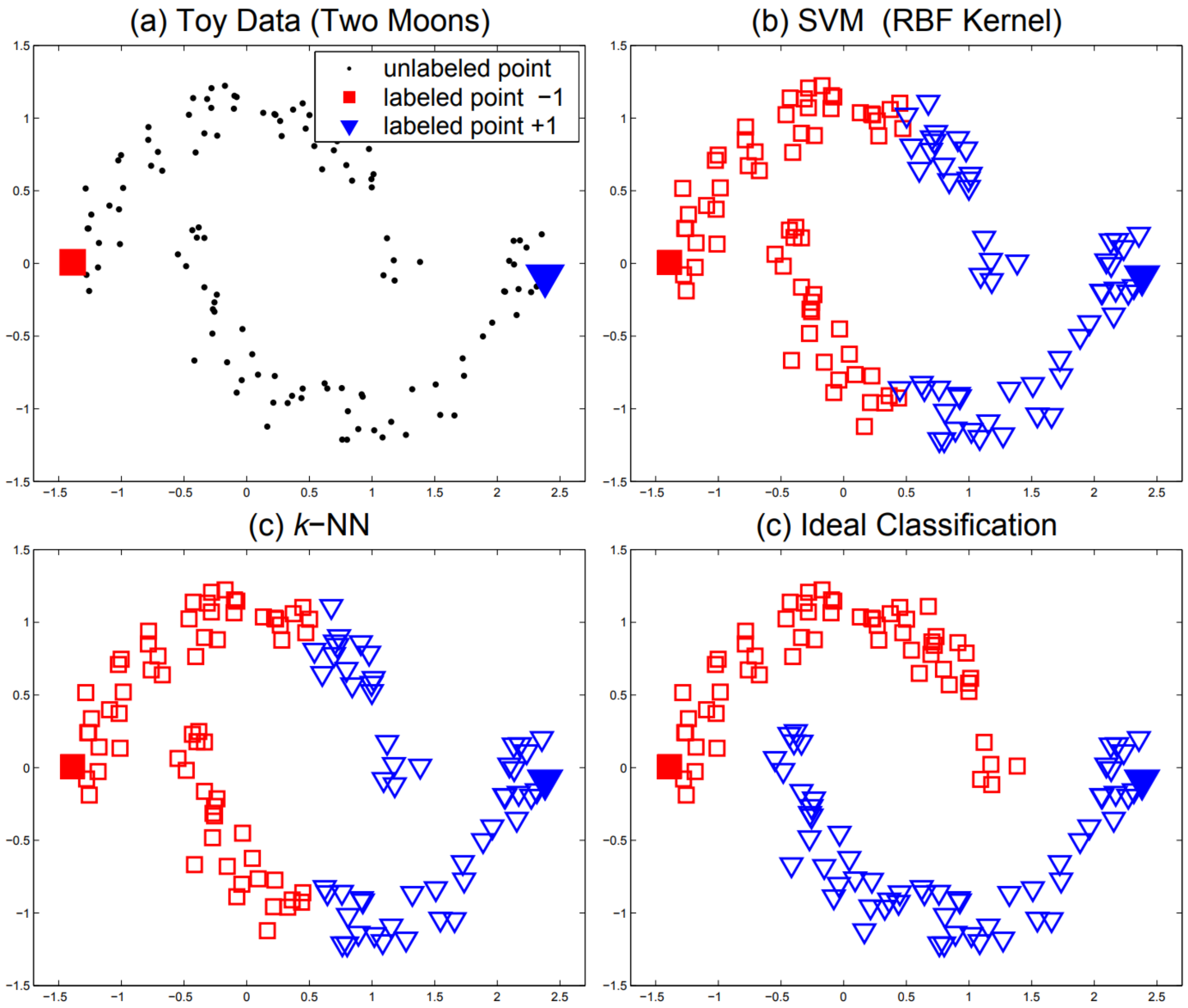
- The weight matrix can be viewed as a linear map from R^N to itself

$$(Wy)_i = \sum_{j \in N(i)} w_{ij} y_j$$

- Similarly, $L = D - W$ can also be viewed as a linear map from R^N to itself

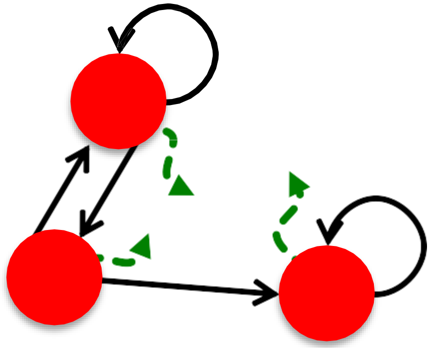
$$(Ly)_i = \sum_{j \in N(i)} w_{ij} (y_i - y_j)$$

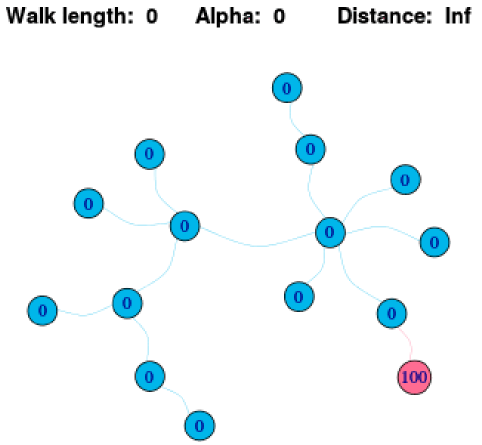
Semi-supervised Learning on Graphs



Personalized PageRank

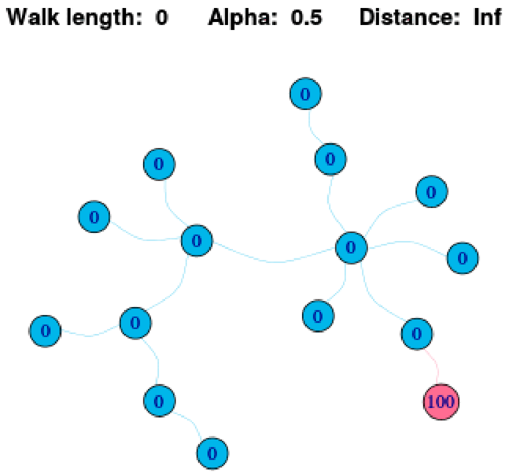
- PageRank: Random Walk over Graph [Page et al., '98]
  - $\mathrm{p}^{t+1} = ((1-\beta)\mathrm{E} + \beta \mathrm{W})\mathrm{p}^{t}$
  - With a probability to randomly/lazily jump





Random Walk

- Semi-supervised learning (and Personalized PageRank)
  - [Haveliwala et al., TKDE'03, Jeh and Widom, WWW'03]
  - [Zhu et al., ICML'03, Zhou et al., NIPS'03]
  - $\mathrm{p}^{t+1} = (1-\beta)\ \mathrm{q} + \beta \mathrm{W}\mathrm{p}^{t}$
  - With a probability to restart with a label: prior



PPR (beta = 0.5)

https://www.r-bloggers.com/from-random-walks-to-personalized-pagerank/

Semi-supervised Learning

- $\mathrm{p}^{t+1} = (1-\beta)\,\mathrm{q} + \beta \mathrm{W} \mathrm{p}^{t}$

- We rewrite the equation in the normalized form as

$$f^{t+1} = (1-\alpha)y + \alpha S f^{t}$$

where

- $f \in R^N$ are the predicted labels on graphs
- $y \in R^N$ is the prior labels on graphs $y_i = \begin{cases} 1 \; if \; i \in A \\ 0 \;\; unknown \end{cases}$
  - For multiple labels on graph, we can have multiple y vectors for each class.
- $S = D^{-1/2} W D^{-1/2}$, here we use the normalized W for the random walk

Label Propagation

- By iterating the equation $f^{t+1} = (1-\alpha)y + \alpha S f^t$, and suppose $f^0 = y$, we have
$$f^t = (1-\alpha)\sum_{k=0}^{t-1}(\alpha S)^k y + (\alpha S)^{t-1} f^t$$
- As $0 < \alpha < 1$, and eigenvalues of S are in [-1, 1]
$$\lim_{t\to\infty}(\alpha S)^{t-1} = 0 \text{ and } \lim_{t\to\infty} \sum_{k=0}^{t-1}(\alpha S)^k = (I - \alpha S)^{-1}$$
- Hence, we have
$$f^* = \lim_{t\to\infty} f^t = (1-\alpha)(I-\alpha S)^{-1} y$$

Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, Bernhard Schölkopf: Learning with Local and Global Consistency. NIPS 2003: 321-328

Graph Regularization Framework

- In fact, we have an objective function for the semi-supervised learning

$$L(f) = \frac{1}{2}\left(\sum_{ij} W_{ij} \left\| \frac{f_i}{\sqrt{D_{ii}}} - \frac{f_j}{\sqrt{D_{jj}}} \right\|^2 + \mu \sum_i \|f_i - y_i\|^2\right)$$

$$= \frac{1}{2}\left(f^T D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}} f + \mu (f - y)^T (f - y)\right)$$

$$= \frac{1}{2}(f^T (I - S) f + \mu (f - y)^T (f - y))$$

- By setting $\frac{\partial L(f)}{\partial f} = f - Sf + \mu(f - y) = 0$, and $\alpha = \frac{1}{1+\mu}$ and $\beta = \frac{\mu}{1+\mu}$ we have $f^* = \beta(I - \alpha S)^{-1} y$

L^{-1} as an Operator

- L^{-1}can act as an operator too
  - $(L^{-1}y)_i$ can be viewed as a linear map from R^N to itself
- Let $G = L^{-1}$ and let $L = I - D^{-1}W$ (convenient to interprete for the probability of random walk)
- Note that $(I - D^{-1}W)^{-1} = \lim_{t\to\infty} \sum_{k=0}^{t-1}(D^{-1}W)^k$
  - $\left((D^{-1}W)^k\right)_{ij}$ shows the *k*-th hop probability from node *i* to node *j*
  - $(I - D^{-1}W)^{-1}$ shows an aggregation of probabilities of all paths from node *i* to node *j*
  - Larger $\left((I - D^{-1}W)^{-1}\right)_{ij}$ indicates similar nodes *i* and *j*
- The classification given by $f^* = L^{-1}y$ is given by

$$p_+(i) = \sum_{y_j=1} G_{ij}$$

Dengyong Zhou, Bernhard Schölkopf: Learning from Labeled and Unlabeled Data Using Random Walks. DAGM-Symposium 2004: 237-244

Summary

- Adjacency matrix of a graph shows clustering property of nodes
- Graph Laplacian indicate informative graph structures locally and globally
- Laplacian as an operator will be useful for graph convolutional network design