Co-clustering documents and words using Bipartite Spectral Graph Partitioning

Inderjit S. Dhillon
Presenter: Lei Tang

16th April 2006

The past work focus on clustering on one axis(either document or word)

- Document Clustering: Agglomerative clustering, k-means, LSA, self-organizing maps, multidimensional scaling etc.
- Word Clustering: distributional clustering, information bottleneck etc.

Co-clustering

simultaneous cluster words and documents!

The past work focus on clustering on one axis(either document or word)

- Document Clustering: Agglomerative clustering, k-means, LSA, self-organizing maps, multidimensional scaling etc.
- Word Clustering: distributional clustering, information bottleneck etc.

Co-clustering

simultaneous cluster words and documents!

Adjacency Matrix $M_{ij} = \begin{cases} E_{ij}, & if\ there\ is\ an\ edge\{i,j\} \\ 0, & otherwise \end{cases}$

$$Cut(V_1, V_2) = \sum_{i \in V_1, j \in V_2} M_{ij}$$

- $G = (D, W, E)$ where D: docs; W: words; E: edges representing a word occurring in a doc.
- The adjacency matrix:

$$M = \begin{bmatrix} 0 & A_{|D| \times |W|} \\ A^T & 0 \end{bmatrix}$$

- No links between documents; No links between words

Adjacency Matrix $M_{ij} = \begin{cases} E_{ij}, & if\ there\ is\ an\ edge\{i,j\} \\ 0, & otherwise \end{cases}$

$$Cut(V_1, V_2) = \sum_{i \in V_1, j \in V_2} M_{ij}$$

- $G = (D, W, E)$ where D: docs; W: words; E: edges representing a word occurring in a doc.
- The adjacency matrix:

$$M = \begin{bmatrix} 0 & A_{|D| \times |W|} \\ A^T & 0 \end{bmatrix}$$

- No links between documents; No links between words

- Disjoint document clusters: $D_1, D_2, \cdots, D_k$
- Disjoint word clusters: $W_1, W_2, \cdots, W_k$
- **Idea:** Document clusters determine word clusters; word clusters in turn determine (better) document clusters. (seems familiar? recall HITS: Authorities/ Hub Computation)
- The "best" partition is the k-way cut of the bipartite graph.

$$cut(W_1 \cup D_1, \cdots, W_k \cup D_k) = \min_{V_1, \cdots, V_k} cut(V1, \cdots, V_k)$$

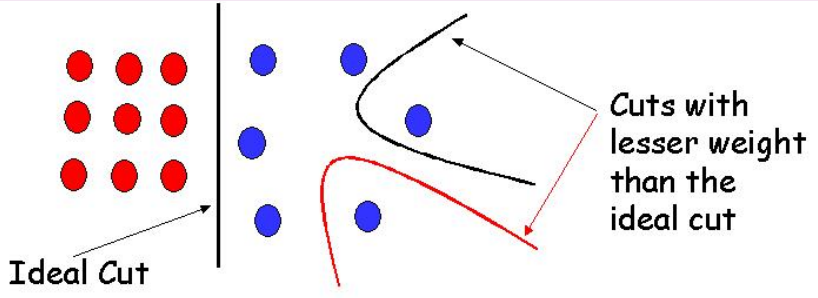
- Solution: **Spectral Graph Partition**

- Disjoint document clusters: $D_1, D_2, \cdots, D_k$
- Disjoint word clusters: $W_1, W_2, \cdots, W_k$
- **Idea:** Document clusters determine word clusters; word clusters in turn determine (better) document clusters. (seems familiar? recall HITS: Authorities/ Hub Computation)
- The "best" partition is the k-way cut of the bipartite graph.

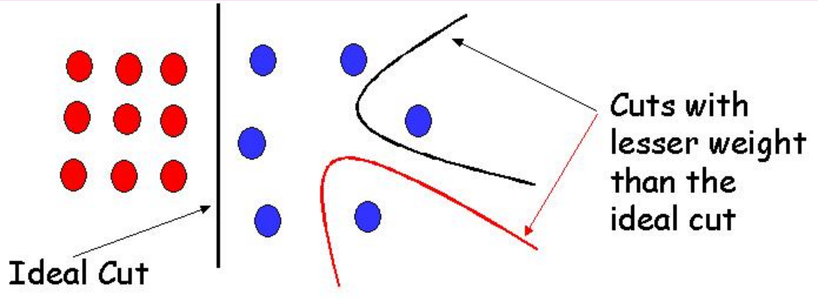
$$cut(W_1 \cup D_1, \cdots, W_k \cup D_k) = \min_{V_1, \cdots, V_k} cut(V1, \cdots, V_k)$$

- Solution: **Spectral Graph Partition**

- 2-partition problem: Partition a graph (not necessarily bipartite) into two parts with minimum between-cluster weights.
- The above problem actually tries to find a minimum cut to partition the graph into two parts.
- Drawbacks: Always find unbalanced cut. *Weight of cut is directly proportional to the number of edges in the cut.*



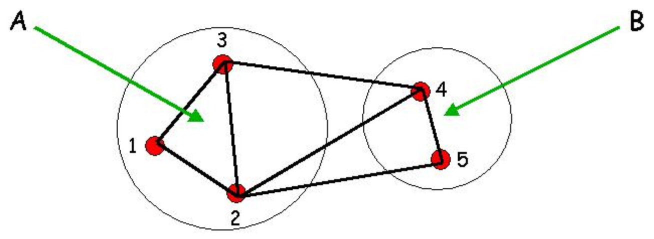
- 2-partition problem: Partition a graph (not necessarily bipartite) into two parts with minimum between-cluster weights.
- The above problem actually tries to find a minimum cut to partition the graph into two parts.
- Drawbacks: Always find unbalanced cut. *Weight of cut is directly proportional to the number of edges in the cut.*



An effective heuristic:

$$WeightedCut(A,B) = \frac{cut(A,B)}{weight(A)} + \frac{cut(A,B)}{weight(B)}$$

If $weight(A) = |A|$, then ***Ratio-cut***;
If $weight(A) = cut(A,B) + within(A)$, then ***Normalized-cut***.

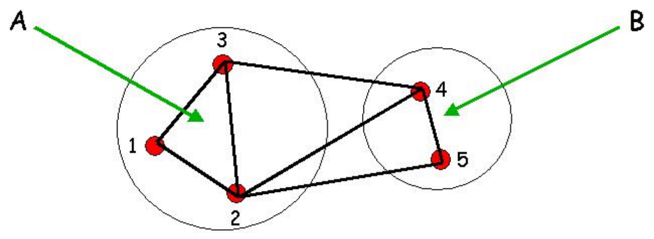


$$\begin{aligned} cut(A,B) &= w(3,4)+w(2,4)+w(2,5) \\ weight(A) &= w(1,3)+w(1,2)+w(2,3)+w(3,4)+w(2,4)+w(2,5) \\ weight(B) &= w(4,5)+w(3,4)+w(2,4)+w(2,5) \end{aligned}$$

An effective heuristic:

$$WeightedCut(A,B) = \frac{cut(A,B)}{weight(A)} + \frac{cut(A,B)}{weight(B)}$$

If $weight(A) = |A|$, then ***Ratio-cut***;
If $weight(A) = cut(A,B) + within(A)$, then ***Normalized-cut***.



$$\begin{aligned} cut(A,B) &= w(3,4) + w(2,4) + w(2,5) \\ weight(A) &= w(1,3) + w(1,2) + w(2,3) + w(3,4) + w(2,4) + w(2,5) \\ weight(B) &= w(4,5) + w(3,4) + w(2,4) + w(2,5) \end{aligned}$$

Solution

Finding the weighted cut boils down to solve a generalized eigenvalue problem:

$$Lz = \lambda W z$$

where L is Laplacian matrix and W is a diagonal weight matrix and z denotes the cut.

Laplacian Matrix for $G(V, E)$:

$$L_{ij} = \begin{cases} \sum_k E_i k, & i = j \\ -E_{ij}, & i \neq j and\, there \quad is\, an\, edge\{i, j\} \\ 0 & otherwise \end{cases}$$

Properties

- $L = D - M$. M is the adjacency matrix, D is the diagonal "degree" matrix with $D_{ii} = \sum_k E_{ik}$
- $L = I_G I_G^T$ where I_G is the $|V| \times |E|$ incidence matrix. For edge (i,j), I_G is 0 except for the i-th and j-th entry which are $\sqrt{E_{ij}}$ and $-\sqrt{E_{ij}}$ respectively.
- $L\hat{\mathbf{1}} = 0$
- $x^T L x = \sum_{i,j \in E} E_{ij}(x_i - x_j)$
- $(\alpha x + \beta \hat{\mathbf{1}})^T L (\alpha x + \beta \hat{\mathbf{1}}) = \alpha^2 x^T L x.$

Let p be a vector to denote a cut:

$$So \qquad p_i = \begin{cases} +1, & i \in A \\ -1, & i \in B \end{cases}$$

$$p^T L p = \sum_{i,j \in E} E_{ij}(p_i - p_j)^2 = 4cut(A,B)$$

Introduce another vector q s.t.

$$q_i = \begin{cases} +\sqrt{\frac{weight(B)}{weight(A)}}, & i \in A \\ -\sqrt{\frac{weight(A)}{weight(B)}}, & i \in B \end{cases}$$

$$\begin{aligned} Then \quad q &= \frac{w_A + w_B}{2\sqrt{w_A w_B}} p + \frac{w_B - w_A}{2\sqrt{w_A w_B}} \hat{\mathbf{1}} \\ q^T L q &= \frac{(w_A + w_B)^2}{4 w_A w_B} p^T L p \qquad (as \quad L\hat{\mathbf{1}} = 0) \\ &= \frac{(w_A + w_B)^2}{w_A w_B} \cdot cut(A,B) \end{aligned}$$

Let p be a vector to denote a cut:

$$So \qquad p_i = \begin{cases} +1, & i \in A \\ -1, & i \in B \end{cases}$$

$$p^T L p = \sum_{i,j \in E} E_{ij}(p_i - p_j)^2 = 4cut(A, B)$$

Introduce another vector q s.t.

$$q_i = \begin{cases} +\sqrt{\frac{weight(B)}{weight(A)}}, & i \in A \\ -\sqrt{\frac{weight(A)}{weight(B)}}, & i \in B \end{cases}$$

$$\begin{aligned} Then \quad q &= \frac{w_A + w_B}{2\sqrt{w_A w_B}} p + \frac{w_B - w_A}{2\sqrt{w_A w_B}} \hat{\mathbf{1}} \\ q^T L q &= \frac{(w_A + w_B)^2}{4 w_A w_B} p^T L p \qquad (as \quad L\hat{\mathbf{1}} = 0) \\ &= \frac{(w_A + w_B)^2}{w_A w_B} \cdot cut(A, B) \end{aligned}$$

Property of q

$$
\begin{aligned}
q^T W e &= 0 \\
q^T W q &= weight(V) = w_A + w_B
\end{aligned}
$$

Then

$$
\begin{aligned}
\frac{q^T L q}{q^T W q} &= \frac{\frac{(w_A+w_B)^2}{w_A w_B} \cdot cut(A,B)}{w_A + w_B} \\
&= \frac{w_A + w_B}{w_A w_B} \cdot cut(A,B) \\
&= \frac{cut(A,B)}{weight(A)} + \frac{cut(A,B)}{weight(B)} \\
&= WeightedCut(A,B)
\end{aligned}
$$

Property of q

$$\begin{aligned} q^T W e &= 0 \\ q^T W q &= weight(V) = w_A + w_B \end{aligned}$$

Then

$$\begin{aligned} \frac{q^T L q}{q^T W q} &= \frac{\frac{(w_A+w_B)^2}{w_A w_B} \cdot cut(A,B)}{w_A + w_B} \\ &= \frac{w_A + w_B}{w_A w_B} \cdot cut(A,B) \\ &= \frac{cut(A,B)}{weight(A)} + \frac{cut(A,B)}{weight(B)} \\ &= WeightedCut(A,B) \end{aligned}$$

So, we need to find a vector q s.t.

$$\min_{q \neq 0} \frac{q^T L q}{q^T W q}, \quad s.t. \quad q^T W e = 0.$$

This is solved when q is the eigenvector corresponds to the 2nd smallest eigenvalue λ_2 of the generalized eigenvalue problem:

$$Lz = \lambda W z$$

In nature, a relaxation to the discrete optimization problem of finding minimum normalized cut.

So, we need to find a vector q s.t.

$$\min_{q \neq 0} \frac{q^T L q}{q^T W q}, \quad s.t. \quad q^T W e = 0.$$

This is solved when q is the eigenvector corresponds to the 2nd smallest eigenvalue λ_2 of the generalized eigenvalue problem:

$$Lz = \lambda W z$$

In nature, a relaxation to the discrete optimization problem of finding minimum normalized cut.

So, we need to find a vector q s.t.

$$\min_{q \neq 0} \frac{q^T L q}{q^T W q}, \quad s.t. \quad q^T W e = 0.$$

This is solved when q is the eigenvector corresponds to the 2nd smallest eigenvalue λ_2 of the generalized eigenvalue problem:

$$Lz = \lambda W z$$

In nature, a relaxation to the discrete optimization problem of finding minimum normalized cut.

$$L = \begin{bmatrix} D_1 & -A \\ -A^T & D_2 \end{bmatrix}; W = \begin{bmatrix} D_1 & 0 \\ 0 & D_2 \end{bmatrix}$$

where $D_1(i,i) = \sum_j A(i,j)$ and $D_2(j,j) = \sum_i A(i,j)$.

Can we make the computation of $Lz = \lambda W z$ more efficiently by taking the advantage of bipartite?

$$L = \begin{bmatrix} D_1 & -A \\ -A^T & D_2 \end{bmatrix}; W = \begin{bmatrix} D_1 & 0 \\ 0 & D_2 \end{bmatrix}$$

where $D_1(i,i) = \sum_j A(i,j)$ and $D_2(j,j) = \sum_i A(i,j)$.

Can we make the computation of $Lz = \lambda W z$ more efficiently by taking the advantage of bipartite?

$$\begin{bmatrix} D_1 & -A \\ -A^T & D_2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \lambda \begin{bmatrix} D_1 & 0 \\ 0 & D_2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Reformulation

$$\begin{aligned} D_1^{1/2}x - D_1^{-1/2}Ay &= \lambda D_1^{1/2}x \\ -D_2^{-1/2}A^Tx + D_2^{1/2}y &= \lambda D_2^{1/2}y \end{aligned}$$

Let $u = D_1^{1/2}x$ and $v = D_2^{1/2}y$,

$$\begin{aligned} D_1^{-1/2}AD_2^{-1/2}v &= (1-\lambda)u \\ D_2^{-1/2}AD_1^{-1/2}u &= (1-\lambda)v \end{aligned}$$

Instead of computing the 2nd **smallest** eigenvector, we can compute the left and right singular vectors corresponding to the 2nd **largest** singular value of A_n:

$$A_n v_2 = \sigma_2 u_2; \quad A_n^T u_2 = \sigma_2 v_2 \quad where \quad \sigma_2 = 1 - \lambda_2$$

$$Then \quad z_2 = \begin{bmatrix} D_1^{-1/2} u_2 \\ D_2^{-1/2} v_2 \end{bmatrix}$$

Bipartition Algorithm:

1. Given A, form $A_n = D_1^{1/2} A D_2 - 1/2$. (note that D_1 and D_2 are both diagonal, easy to compute)
2. Compute z_2 by SVD
3. Run k-means with $k = 2$ on the 1-dimentional z_2 to obtain the desired partitioning.

Multipartition Algorithm:

For k clusters, compute $l = \lceil log_2 k \rceil$ singular vectors of A_n and form l eigenvectors Z.
Then apply k-means to find k-way partitioning.

Experiment Result

- Both Bipartition and multipartition algorithm works fine in text domain even without removing the stop words
- Comment: No comparison is performed. I think this work's major contribution is to introduce spectral clustering into text domain and present a neat formulation for co-clustering.

Multipartition Algorithm:

For k clusters, compute $l = \lceil log_2 k \rceil$ singular vectors of A_n and form l eigenvectors Z.
Then apply k-means to find k-way partitioning.

Experiment Result

- Both Bipartition and multipartition algorithm works fine in text domain even without removing the stop words
- Comment: No comparison is performed. I think this work's major contribution is to introduce spectral clustering into text domain and present a neat formulation for co-clustering.

Contributions

1. Model document collection as a bipartite graph (Extendable to almost all the data sets. Two components: data points, Feature set)
2. Use spectral graph partitioning for Co-clustering
3. Reslove the problem using SVD
4. Beautiful Theory

Questions

1. Connection to HITS? Docs as hubs, Words as authorities. Can we get the same result as bipartitioning? In HITS, $a_i = A^T A a_{i-1}$ and $h_i = AA^T h_{i-1}$ corresponding to the largest eigenvector of AA^T and $A^T A$, respectively.
2. Extendable to Semi-supervised Learning? How to solve the problem is some documents and words are already labeled? (This is done?) Can we get good result by applying DengYong Zhou's semi-supervised method?

Any other question?

Thank you!