Clustering of Unsigned and Signed Graphs Using Normalized Graph Cuts

Jean Gallier

CIS Department
University of Pennsylvania

jean@cis.upenn.edu

November 4, 2016

Special thanks to Jocelyn Quaintance, Joao Sedoc, Jianbo Shi, and Stella Yu.

Special thanks to Jocelyn Quaintance, Joao Sedoc, Jianbo Shi, and Stella Yu.

Complete details are in

Spectral Theory of Unsigned and Signed Graphs Applications to Graph Clustering: a Survey.

http://www.cis.upenn.edu/∼jean/spectral-graph-notes.pdf

Special thanks to Jocelyn Quaintance, Joao Sedoc, Jianbo Shi, and Stella Yu.

Complete details are in

Spectral Theory of Unsigned and Signed Graphs Applications to Graph Clustering: a Survey.

http://www.cis.upenn.edu/∼jean/spectral-graph-notes.pdf

Detailed slides in Web page for CIS515:

http://www.cis.upenn.edu/∼cis515/cis515-notes-15.html

Figure 1: Dog Logic

1. Graph Clustering

Given a set of data, the goal of clustering is to *partition* the data into different groups according to their *similarities*.

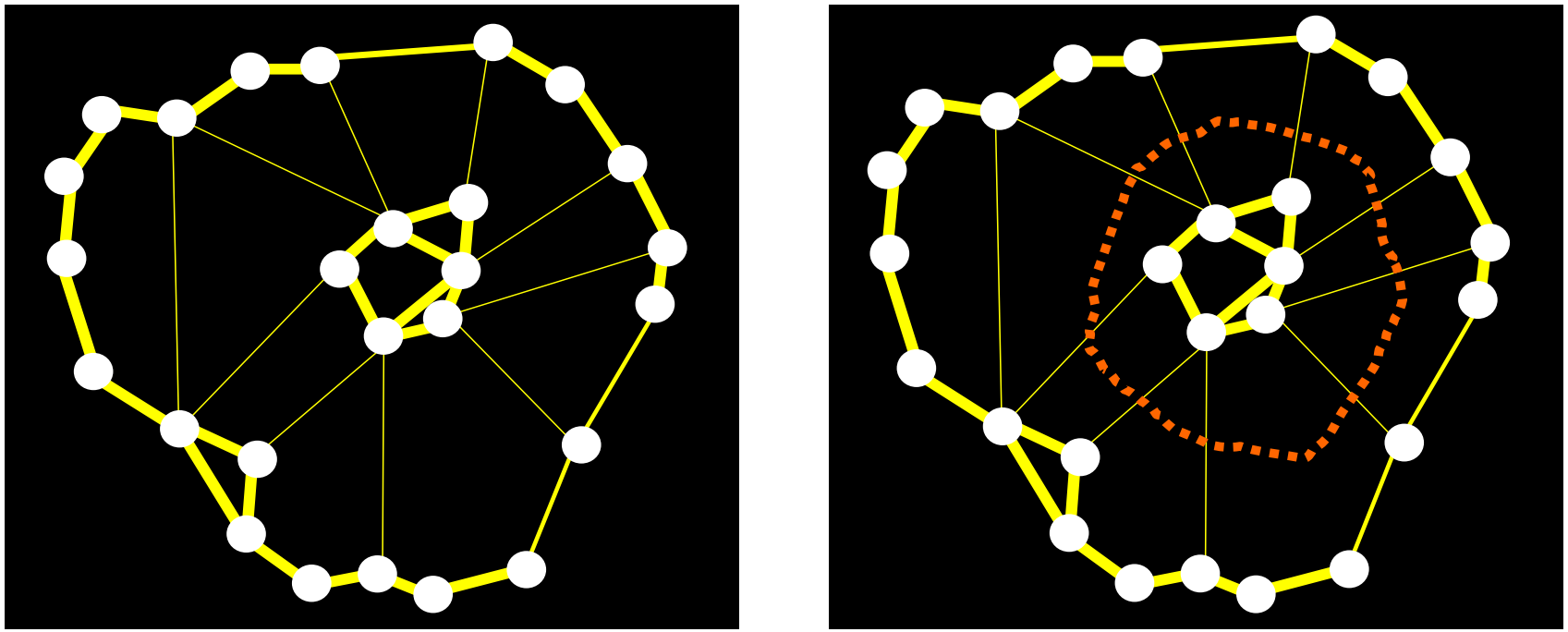


Figure 2: A weighted graph and its partition into two clusters.

When the data is given in terms of a *similarity graph* G, where the *weight* w_{ij} between two nodes v_i and v_j is a measure of similarity of v_i and v_j, the problem can be stated as follows:

When the data is given in terms of a *similarity graph* G, where the *weight* w_{ij} between two nodes v_i and v_j is a measure of similarity of v_i and v_j, the problem can be stated as follows:

Find a partition $(A_1, \ldots, A_K)$ of the set of nodes V into different groups such that the *edges between different groups have very low weight* (which indicates that the points in different clusters are dissimilar), and the *edges within a group have high weight* (which indicates that points within the same cluster are similar).

When the data is given in terms of a *similarity graph* G, where the *weight* w_{ij} between two nodes v_i and v_j is a measure of similarity of v_i and v_j, the problem can be stated as follows:

Find a partition $(A_1, \ldots, A_K)$ of the set of nodes V into different groups such that the *edges between different groups have very low weight* (which indicates that the points in different clusters are dissimilar), and the *edges within a group have high weight* (which indicates that points within the same cluster are similar).

The above graph clustering problem can be formalized as an optimization problem, using the notion of *cut*.

Example in computer vision: finding contours in an image.

Figure 3: Images with top 20 contours extracted

Typically, the following steps are followed:

Typically, the following steps are followed:

1. Formulate the *discrete optimization problem* in matrix form.

Typically, the following steps are followed:

1. Formulate the *discrete optimization problem* in matrix form.
2. The discrete problem is often very hard (NP-hard, ...). *Relax* the problem (drop some constraints and look for *continuous solutions*).

Typically, the following steps are followed:

1. Formulate the *discrete optimization problem* in matrix form.
2. The discrete problem is often very hard (NP-hard, ...). *Relax* the problem (drop some constraints and look for *continuous solutions*).
3. Find a *discrete solution* as close as possible to a continuous solution.

Typically, the following steps are followed:

1. Formulate the *discrete optimization problem* in matrix form.
2. The discrete problem is often very hard (NP-hard, ...). *Relax* the problem (drop some constraints and look for *continuous solutions*).
3. Find a *discrete solution* as close as possible to a continuous solution.

Step (2) often reduces to some kind of eigenvalue problem.

Typically, the following steps are followed:

1. Formulate the *discrete optimization problem* in matrix form.
2. The discrete problem is often very hard (NP-hard, ...). *Relax* the problem (drop some constraints and look for *continuous solutions*).
3. Find a *discrete solution* as close as possible to a continuous solution.

Step (2) often reduces to some kind of eigenvalue problem.

Step (3) is usually the hardest step.

2. Weigted Graphs, Cuts, Laplacians

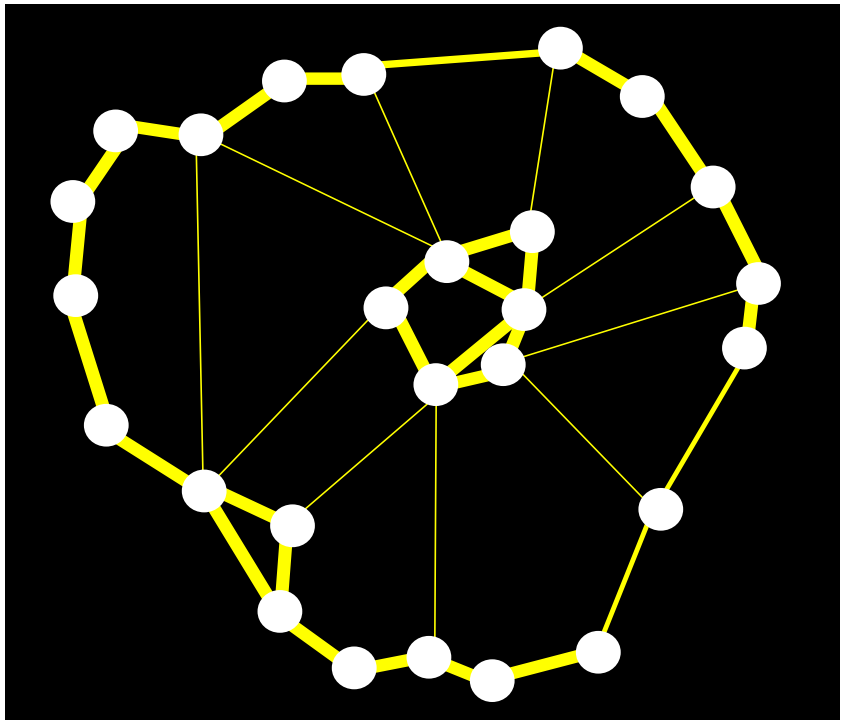


Figure 4: A weighted graph.

The thickness of an edge corresponds to the magnitude of its weight.

Given the weight matrix

$$W = \begin{pmatrix} 0 & 3 & 6 & 3 \\ 3 & 0 & 0 & 3 \\ 6 & 0 & 0 & 3 \\ 3 & 3 & 3 & 0 \end{pmatrix},$$

Given the weight matrix

$$W = \begin{pmatrix} 0 & 3 & 6 & 3 \\ 3 & 0 & 0 & 3 \\ 6 & 0 & 0 & 3 \\ 3 & 3 & 3 & 0 \end{pmatrix},$$

with node set $V = \{v_1, v_2, v_3, v_4\}$, the corresponding graph G is:

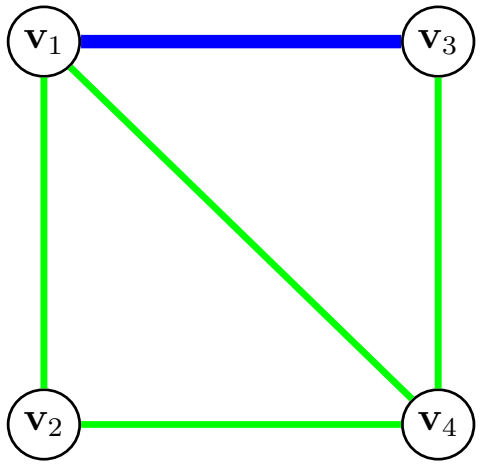


Figure 5: The weighted graph corresponding to W.

For every node $v_i \in V$, the *degree* $d(v_i)$ of v_i is the sum of the weights of the edges adjacent to v_i:

$$d(v_i) = \sum_{j=1}^{m} w_{ij}.$$

For every node $v_i \in V$, the *degree* $d(v_i)$ of v_i is the sum of the weights of the edges adjacent to v_i:

$$d(v_i) = \sum_{j=1}^{m} w_{ij}.$$

Note that in the above sum, only nodes v_j such that there is an edge $\{v_i, v_j\}$ have a nonzero contribution. Such nodes are said to be *adjacent* to v_i.

For every node $v_i \in V$, the *degree* $d(v_i)$ of v_i is the sum of the weights of the edges adjacent to v_i:

$$d(v_i) = \sum_{j=1}^{m} w_{ij}.$$

Note that in the above sum, only nodes v_j such that there is an edge $\{v_i, v_j\}$ have a nonzero contribution. Such nodes are said to be *adjacent* to v_i.

The *degree matrix* D is defined by $D = \mathrm{diag}(d(v_1), \ldots, d(v_m))$.

Given any subset of nodes $A \subseteq V$, we define the *volume* $\mathrm{vol}(A)$ of A as the sum of the weights of all edges adjacent to nodes in A:

$$\mathrm{vol}(A) = \sum_{v_i \in A} d(v_i) = \sum_{v_i \in A} \sum_{j=1}^{m} w_{ij}.$$

Given any subset of nodes $A \subseteq V$, we define the *volume* $\mathrm{vol}(A)$ of A as the sum of the weights of all edges adjacent to nodes in A:

$$\mathrm{vol}(A) = \sum_{v_i \in A} d(v_i) = \sum_{v_i \in A} \sum_{j=1}^{m} w_{ij}.$$

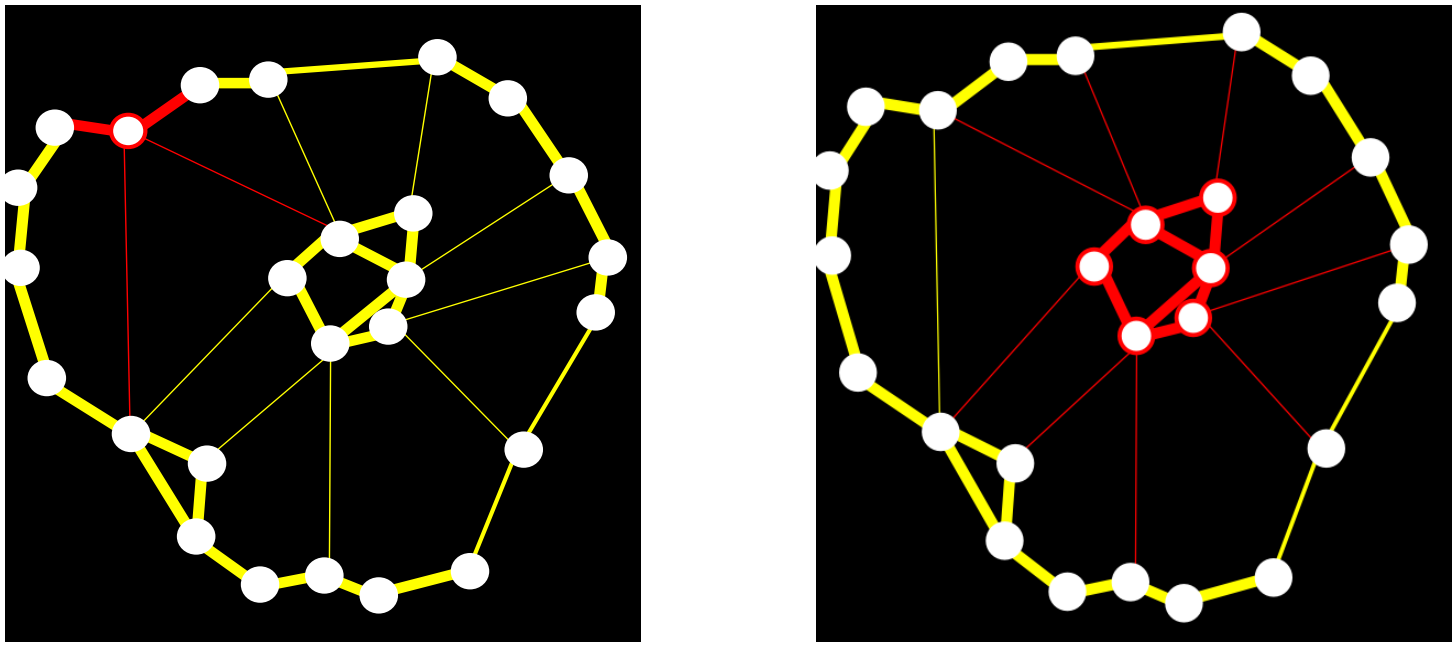


Figure 6: Degree and volume.

Observe that $\mathrm{vol}(A) = 0$ if A consists of isolated vertices ($w_{ij} = 0$ for all $v_i \in A$). Thus, it is best to assume that G does not have isolated vertices.

Observe that $\mathrm{vol}(A) = 0$ if A consists of isolated vertices ($w_{ij} = 0$ for all $v_i \in A$). Thus, it is best to assume that G does not have isolated vertices.

Given any two subset $A, B \subseteq V$ (not necessarily distinct), we define $\mathrm{links}(A, B)$ by

$$\mathrm{links}(A, B) = \sum_{v_i \in A, v_j \in B} w_{ij}.$$

Observe that $\mathrm{vol}(A) = 0$ if A consists of isolated vertices ($w_{ij} = 0$ for all $v_i \in A$). Thus, it is best to assume that G does not have isolated vertices.

Given any two subset $A, B \subseteq V$ (not necessarily distinct), we define $\mathrm{links}(A, B)$ by

$$\mathrm{links}(A, B) = \sum_{v_i \in A, v_j \in B} w_{ij}.$$

Since the matrix W is symmetric, we have

$$\mathrm{links}(A, B) = \mathrm{links}(B, A).$$

The quantity $\mathrm{links}(A, \overline{A}) = \mathrm{links}(\overline{A}, A)$, where $\overline{A} = V - A$ denotes the complement of A in V, *measures how many links escape from A* (and $\overline{A}$), and the quantity $\mathrm{links}(A, A)$ *measures how many links stay within A itself*.

The quantity $\mathrm{links}(A, \overline{A}) = \mathrm{links}(\overline{A}, A)$, where $\overline{A} = V - A$ denotes the complement of A in V, *measures how many links escape from A* (and $\overline{A}$), and the quantity $\mathrm{links}(A, A)$ *measures how many links stay within A itself*.

The quantity

$$\mathrm{cut}(A) = \mathrm{links}(A, \overline{A})$$

is often called the *cut* of A.

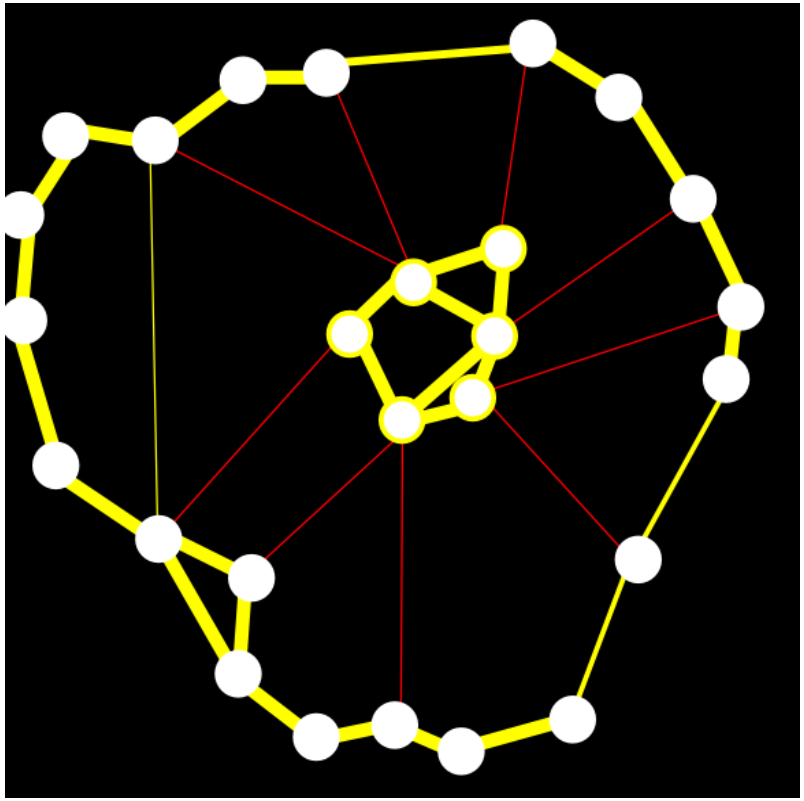


Figure 7: A Cut involving the set of nodes in the center and the nodes on the perimeter.

We now define the most important concept of this talk: The *Laplacian matrix* of a graph. Actually, as we will see, it comes in several flavors.

We now define the most important concept of this talk: The *Laplacian matrix* of a graph. Actually, as we will see, it comes in several flavors.

Definition 1

Given any weighted graph $G = (V, W)$ with $V = \{v_1, \ldots, v_m\}$, the *(unnormalized) graph Laplacian $L(G)$ of G* is defined by

$$L(G) = D(G) - W,$$

where $D(G) = \mathrm{diag}(d_1, \ldots, d_m)$ is the degree matrix of G (a diagonal matrix), with

$$d_i = \sum_{j=1}^{m} w_{ij}.$$

Consider the weight matrix

$$W = \begin{pmatrix} 0 & 3 & 6 & 3 \\ 3 & 0 & 0 & 3 \\ 6 & 0 & 0 & 3 \\ 3 & 3 & 3 & 0 \end{pmatrix},$$

Consider the weight matrix

$$W = \begin{pmatrix} 0 & 3 & 6 & 3 \\ 3 & 0 & 0 & 3 \\ 6 & 0 & 0 & 3 \\ 3 & 3 & 3 & 0 \end{pmatrix},$$

and the corresponding graph G

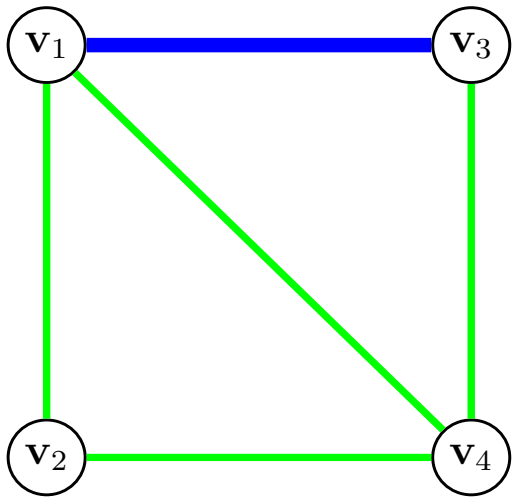


Figure 8: The weighted graph corresponding to W.

The degree matrix (`diag(sum(W))`) is

$$D(G) = \begin{pmatrix} 12 & 0 & 0 & 0 \\ 0 & 6 & 0 & 0 \\ 0 & 0 & 9 & 0 \\ 0 & 0 & 0 & 9 \end{pmatrix},$$

The degree matrix (`diag(sum(W))`) is

$$D(G) = \begin{pmatrix} 12 & 0 & 0 & 0 \\ 0 & 6 & 0 & 0 \\ 0 & 0 & 9 & 0 \\ 0 & 0 & 0 & 9 \end{pmatrix},$$

and the Laplacian is

$$L = D(G) - W = \begin{pmatrix} 12 & -3 & -6 & -3 \\ -3 & 6 & 0 & -3 \\ -6 & 0 & 9 & -3 \\ -3 & -3 & -3 & 9 \end{pmatrix}.$$

The degree matrix (`diag(sum(W))`) is

$$D(G) = \begin{pmatrix} 12 & 0 & 0 & 0 \\ 0 & 6 & 0 & 0 \\ 0 & 0 & 9 & 0 \\ 0 & 0 & 0 & 9 \end{pmatrix},$$

and the Laplacian is

$$L = D(G) - W = \begin{pmatrix} 12 & -3 & -6 & -3 \\ -3 & 6 & 0 & -3 \\ -6 & 0 & 9 & -3 \\ -3 & -3 & -3 & 9 \end{pmatrix}.$$

The eigenvalues of L are: $0, 6.8038, 12, 17.1962$.

The vector $\mathbf{1}$ is the nullspace of L, but it is less obvious that L is positive semidefinite.

The vector $\mathbf{1}$ is the nullspace of L, but it is less obvious that L is positive semidefinite.

Proposition 1

For any $m \times m$ symmetric matrix W, if we let $L = D - W$ where D is the degree matrix of $W = (w_{ij})$, then we have

$$x^\top L x = \frac{1}{2} \sum_{i,j=1}^{m} w_{ij}(x_i - x_j)^2 \quad \text{for all } x \in \mathbb{R}^m.$$

Consequently, $x^\top L x$ does not depend on the diagonal entries in W, and if $w_{ij} \geq 0$ for all $i, j \in \{1, \ldots, m\}$, then L is positive semidefinite.

Proposition 1 immediately implies the following facts: For any weighted graph $G = (V, W)$,

Proposition 1 immediately implies the following facts: For any weighted graph $G = (V, W)$,

1. The eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_m$ of L are real and nonnegative, and there is an orthonormal basis of eigenvectors of L.
2. The smallest eigenvalue λ_1 of L is equal to 0, and $\mathbf{1}$ is a corresponding eigenvector.

Normalized variants of the graph Laplacian are needed, especially in applications to graph clustering.

These variants make sense only if G has no isolated vertices. In this case, the degree matrix D contains positive entries, so it is invertible and $D^{-1/2}$ makes sense; namely

$$D^{-1/2} = \mathrm{diag}(d_1^{-1/2}, \ldots, d_m^{-1/2}).$$

These variants make sense only if G has no isolated vertices. In this case, the degree matrix D contains positive entries, so it is invertible and $D^{-1/2}$ makes sense; namely

$$D^{-1/2} = \mathrm{diag}(d_1^{-1/2}, \ldots, d_m^{-1/2}).$$

Definition 2

Given any weighted directed graph $G = (V, W)$ with no isolated vertex and with $V = \{v_1, \ldots, v_m\}$, the *(normalized) graph Laplacian* L_{sym} is defined by

$$L_{\mathrm{sym}} = D^{-1/2} L D^{-1/2} = I - D^{-1/2} W D^{-1/2}$$

Proposition 2

Let $G = (V, W)$ be a weighted graph without isolated vertices. The graph Laplacians L and L_{sym} satisfy the following properties:

(1) *The normalized graph Laplacian L_{sym} has a spectrum $(0 = \nu_1 \leq \nu_2 \leq \ldots \leq \nu_m \leq 2)$.*

(2) *The graph Laplacians L and L_{sym} are symmetric, positive, semidefinite.*

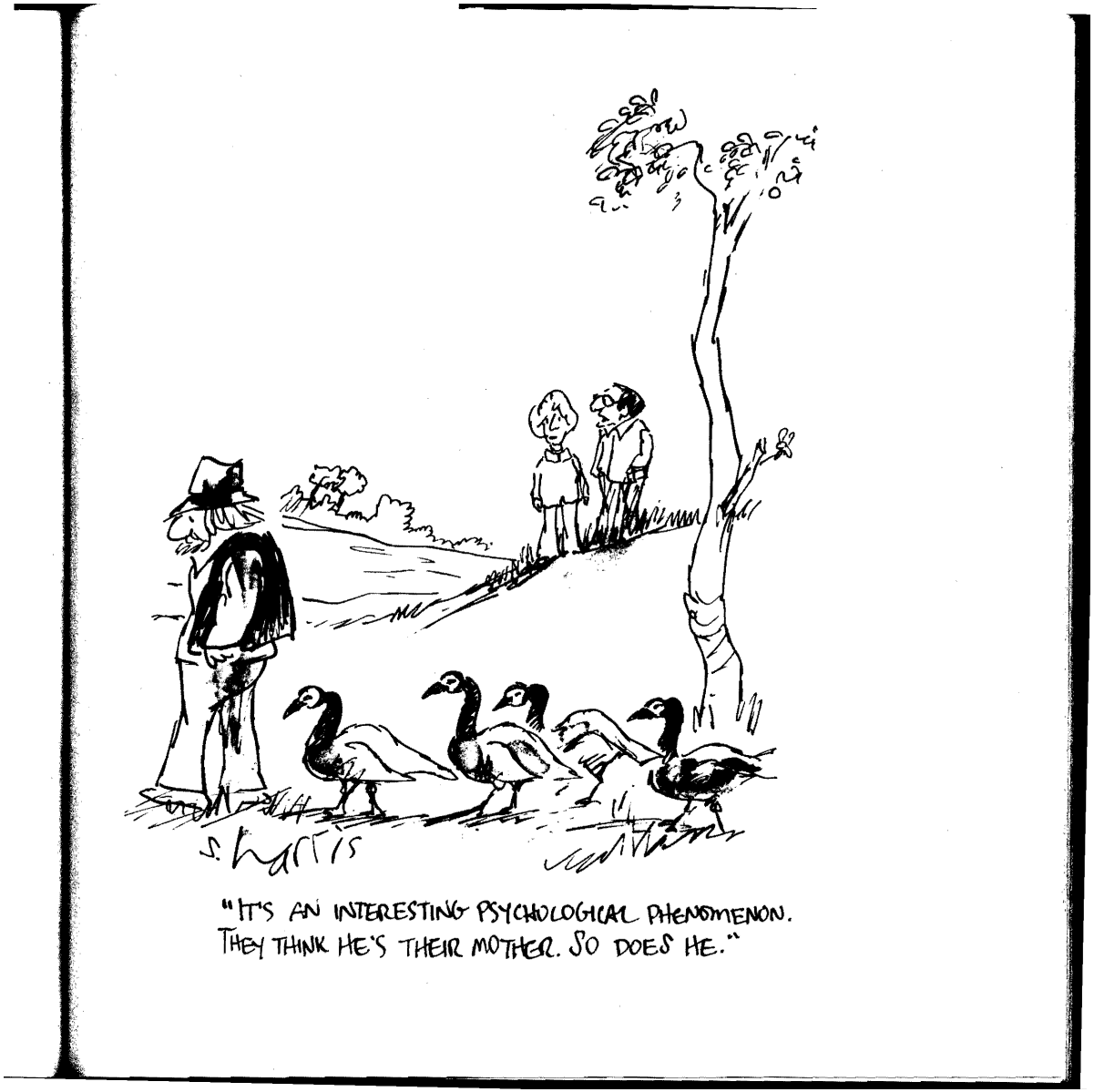


Figure 9: Are you my mother?

3. Back to Graph Clustering

If we want to partition V into K clusters, we can do so by finding a partition $(A_1, \ldots, A_K)$ that minimizes the quantity

$$\mathrm{cut}(A_1, \ldots, A_K) = \frac{1}{2} \sum_{1=1}^{K} \mathrm{cut}(A_i).$$

3. Back to Graph Clustering

If we want to partition V into K clusters, we can do so by finding a partition $(A_1, \ldots, A_K)$ that minimizes the quantity

$$\mathrm{cut}(A_1, \ldots, A_K) = \frac{1}{2} \sum_{1=1}^{K} \mathrm{cut}(A_i).$$

For $K = 2$, the mincut problem is a classical problem that can be solved efficiently, but in practice, it does not yield satisfactory partitions.

3. Back to Graph Clustering

If we want to partition V into K clusters, we can do so by finding a partition $(A_1, \ldots, A_K)$ that minimizes the quantity

$$\mathrm{cut}(A_1, \ldots, A_K) = \frac{1}{2} \sum_{1=1}^{K} \mathrm{cut}(A_i).$$

For $K = 2$, the mincut problem is a classical problem that can be solved efficiently, but in practice, it does not yield satisfactory partitions.

Indeed, in many cases, the mincut solution separates one vertex from the rest of the graph. What we need is to design our cost function in such a way that it keeps the subsets A_i "reasonably large" (reasonably balanced).

A way to get around this problem is to normalize the cuts by *dividing by some measure of each subset A_i*.

A way to get around this problem is to normalize the cuts by *dividing by some measure of each subset A_i*.

One possibility is to use the size (the number of elements) of A_i.

A way to get around this problem is to normalize the cuts by *dividing by some measure of each subset A_i*.

One possibility is to use the size (the number of elements) of A_i.

Another is to use the *volume* $\mathrm{vol}(A_i)$ of A_i. A solution using the second measure (the volume) (for $K = 2$) was proposed and investigated in a seminal paper of Shi and Malik.

A way to get around this problem is to normalize the cuts by *dividing by some measure of each subset A_i*.

One possibility is to use the size (the number of elements) of A_i.

Another is to use the *volume* $\mathrm{vol}(A_i)$ of A_i. A solution using the second measure (the volume) (for $K = 2$) was proposed and investigated in a seminal paper of Shi and Malik.

Subsequently, Stella Yu (in her dissertation) and Yu and Shi extended the method to $K > 2$ clusters.

The idea is to minimize the cost function

$$\mathrm{Ncut}(A_1, \ldots, A_K) = \sum_{i=1}^{K} \frac{\mathrm{links}(A_i, \overline{A_i})}{\mathrm{vol}(A_i)} = \sum_{i=1}^{K} \frac{\mathrm{cut}(A_i, \overline{A_i})}{\mathrm{vol}(A_i)}.$$

The idea is to minimize the cost function

$$\mathrm{Ncut}(A_1, \ldots, A_K) = \sum_{i=1}^{K} \frac{\mathrm{links}(A_i, \overline{A_i})}{\mathrm{vol}(A_i)} = \sum_{i=1}^{K} \frac{\mathrm{cut}(A_i, \overline{A_i})}{\mathrm{vol}(A_i)}.$$

We proceed directly to the case $K > 2$ which is the most interesting case, and is harder to handle.

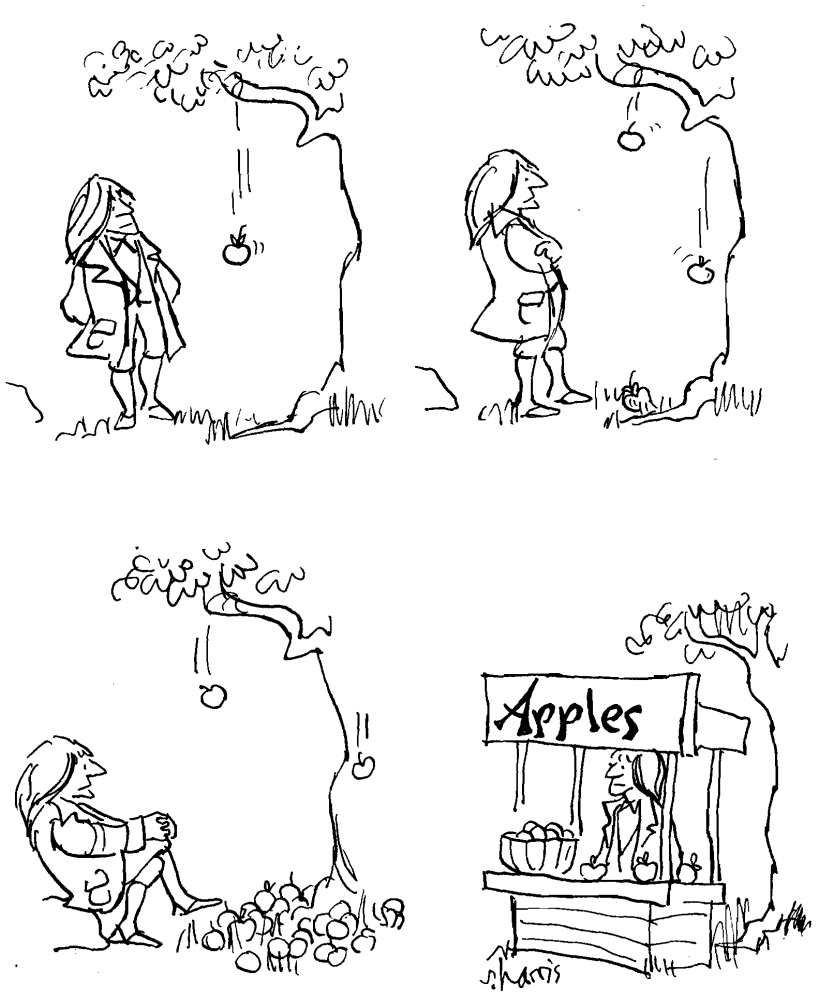


Figure 10: Newton goes to Wharton

4. K-Way Clustering Using Normalized Cuts

Two crucial issues need to be addressed:

4. K-Way Clustering Using Normalized Cuts

Two crucial issues need to be addressed:

1. The choice of a matrix representation for partitions on the set of vertices.

4. K-Way Clustering Using Normalized Cuts

Two crucial issues need to be addressed:

1. The choice of a matrix representation for partitions on the set of vertices.

   It is important that such a representation be *scale-invariant*.

4. K-Way Clustering Using Normalized Cuts

Two crucial issues need to be addressed:

1. The choice of a matrix representation for partitions on the set of vertices.

   It is important that such a representation be *scale-invariant*.

   It is also necessary to state necessary and sufficient conditions for such matrices to represent a partition.

4. K-Way Clustering Using Normalized Cuts

Two crucial issues need to be addressed:

1. The choice of a matrix representation for partitions on the set of vertices.

   It is important that such a representation be *scale-invariant*.

   It is also necessary to state necessary and sufficient conditions for such matrices to represent a partition.

2. The choice of a *metric* to compare solutions.

We describe a partition $(A_1, \ldots, A_K)$ of the set of nodes V by an $N \times K$ matrix $X = [X^1 \cdots X^K]$ whose columns $X^1, \ldots, X^K$ are indicator vectors of the partition $(A_1, \ldots, A_K)$.

We describe a partition $(A_1, \ldots, A_K)$ of the set of nodes V by an $N \times K$ matrix $X = [X^1 \cdots X^K]$ whose columns $X^1, \ldots, X^K$ are indicator vectors of the partition $(A_1, \ldots, A_K)$.

We assume that the vector X^j is of the form

$$X^j = (x_1^j, \ldots, x_N^j),$$

where $x_i^j \in \{a_j, 0\}$ for $j = 1, \ldots, K$ and $i = 1, \ldots, N$, and with $a_j \neq 0$.

We describe a partition $(A_1, \ldots, A_K)$ of the set of nodes V by an $N \times K$ matrix $X = [X^1 \cdots X^K]$ whose columns $X^1, \ldots, X^K$ are indicator vectors of the partition $(A_1, \ldots, A_K)$.

We assume that the vector X^j is of the form

$$X^j = (x_1^j, \ldots, x_N^j),$$

where $x_i^j \in \{a_j, 0\}$ for $j = 1, \ldots, K$ and $i = 1, \ldots, N$, and with $a_j \neq 0$.

$x_i^j = a_j$ means that node v_i belongs to cluster K_j.

When $N = 10$ and $K = 4$, an example of a matrix X representing the partition of $V = \{v_1, v_2, \ldots, v_{10}\}$ into the four blocks

$$\{A_1, A_2, A_3, A_4\} = \{\{v_2, v_4, v_6\}, \{v_1, v_5\}, \{v_3, v_8, v_{10}\}, \{v_7, v_9\}\},$$

is shown below:

When $N = 10$ and $K = 4$, an example of a matrix X representing the partition of $V = \{v_1, v_2, \ldots, v_{10}\}$ into the four blocks

$$\{A_1, A_2, A_3, A_4\} = \{\{v_2, v_4, v_6\}, \{v_1, v_5\}, \{v_3, v_8, v_{10}\}, \{v_7, v_9\}\},$$

is shown below:

$$X = \begin{pmatrix} 0 & a_2 & 0 & 0 \\ a_1 & 0 & 0 & 0 \\ 0 & 0 & a_3 & 0 \\ a_1 & 0 & 0 & 0 \\ 0 & a_2 & 0 & 0 \\ a_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & a_4 \\ 0 & 0 & a_3 & 0 \\ 0 & 0 & 0 & a_4 \\ 0 & 0 & a_3 & 0 \end{pmatrix}.$$

Let $d = \mathbf{1}^\top D\mathbf{1}$ and $\alpha_j = \mathrm{vol}(A_j)$, so that $\alpha_1 + \cdots + \alpha_K = d$.

Let $d = \mathbf{1}^\top D\mathbf{1}$ and $\alpha_j = \mathrm{vol}(A_j)$, so that $\alpha_1 + \cdots + \alpha_K = d$.

Then, $\mathrm{vol}(\overline{A_j}) = d - \alpha_j$, and we find that

Let $d = \mathbf{1}^\top D\mathbf{1}$ and $\alpha_j = \mathrm{vol}(A_j)$, so that $\alpha_1 + \cdots + \alpha_K = d$.

Then, $\mathrm{vol}(\overline{A_j}) = d - \alpha_j$, and we find that

$$\frac{\mathrm{cut}(A_j, \overline{A_j})}{\mathrm{vol}(A_j)} = \frac{(X^j)^\top L X^j}{(X^j)^\top D X^j} \quad j = 1, \ldots, K.$$

Let $d = \mathbf{1}^\top D\mathbf{1}$ and $\alpha_j = \mathrm{vol}(A_j)$, so that $\alpha_1 + \cdots + \alpha_K = d$.

Then, $\mathrm{vol}(\overline{A_j}) = d - \alpha_j$, and we find that

$$\frac{\mathrm{cut}(A_j, \overline{A_j})}{\mathrm{vol}(A_j)} = \frac{(X^j)^\top L X^j}{(X^j)^\top D X^j} \quad j = 1, \ldots, K.$$

This gives us a matrix expression for the cost

$$\mathrm{Ncut}(A_1, \ldots, A_K) = \sum_{i=1}^{K} \frac{\mathrm{cut}(A_i, \overline{A_i})}{\mathrm{vol}(A_i)}.$$

We also denote $\mathrm{Ncut}(A_1, \ldots, A_K)$ as $\mu(X^1, \ldots, X^K)$.

If we let

$$\mathcal{X} = \left\{ [X^1 \ \dots \ X^K] \mid X^j = a_j(x_1^j, \dots, x_N^j),\ x_i^j \in \{1, 0\}, a_j \in \mathbb{R},\ X^j \neq 0 \right\}$$

then our optimization problem is:

If we let

$$\mathcal{X} = \left\{ [X^1 \ \ldots \ X^K] \mid X^j = a_j(x_1^j, \ldots, x_N^j),\ x_i^j \in \{1, 0\}, a_j \in \mathbb{R},\ X^j \neq 0 \right\}$$

then our optimization problem is:

K-way Clustering of a graph using Normalized Cut, Version 1: Problem PNC1

$$\begin{aligned} \text{minimize} \quad & \mu(X^1, \ldots, X^K) = \sum_{j=1}^{K} \frac{(X^j)^\top L X^j}{(X^j)^\top D X^j} \\ \text{subject to} \quad & (X^i)^\top D X^j = 0, \quad 1 \leq i, j \leq K,\ i \neq j, \\ & X(X^\top X)^{-1} X^\top \mathbf{1} = \mathbf{1}, \qquad\qquad X \in \mathcal{X}. \end{aligned}$$

Let's ignore the second technical contraint for now.

The solutions that we are seeking are K-tuples $(X^1 : \cdots : X^K)$ of points in projective space $\mathbb{RP}^{N-1}$ determined by their homogeneous coordinates $X^1, \ldots, X^K$.

The solutions that we are seeking are K-tuples $(X^1 : \cdots : X^K)$ of points in projective space $\mathbb{RP}^{N-1}$ determined by their homogeneous coordinates $X^1, \ldots, X^K$.

Our original formulation (PNC1) can be converted to a more convenient form, by chasing the denominators in the Rayleigh ratios, and by expressing the objective function in terms of the *trace* of a certain matrix.

K-way Clustering of a graph using Normalized Cut, Version 1: Problem PNC1

$$\begin{aligned} \text{minimize} \quad & \mu(X^1, \ldots, X^K) = \sum_{j=1}^{K} \frac{(X^j)^\top L X^j}{(X^j)^\top D X^j} \\ \text{subject to} \quad & (X^i)^\top D X^j = 0, \quad 1 \leq i, j \leq K,\ i \neq j, \\ & X \in \mathcal{X}. \end{aligned}$$

K-way Clustering of a graph using Normalized Cut, Version 1: Problem PNC1

$$\begin{aligned} \text{minimize} \quad & \mu(X^1, \ldots, X^K) = \sum_{j=1}^{K} \frac{(X^j)^\top L X^j}{(X^j)^\top D X^j} \\ \text{subject to} \quad & (X^i)^\top D X^j = 0, \quad 1 \leq i, j \leq K,\ i \neq j, \\ & X \in \mathcal{X}. \end{aligned}$$

K-way Clustering of a graph using Normalized Cut, Version 2: Problem PNC2

$$\begin{aligned} \text{minimize} \quad & \operatorname{tr}(X^\top L X) \\ \text{subject to} \quad & X^\top D X = I, \\ & X \in \mathcal{X}. \end{aligned}$$

Problem PNC2 is equivalent to problem PNC1 if we view the solutions as homogeneous coordinates (up to a nonzero scalar).

Problem PNC2 is equivalent to problem PNC1 if we view the solutions as homogeneous coordinates (up to a nonzero scalar).

The main problem in finding a good relaxation of problem PNC2 is that it is very difficult to enforce the condition $X \in \mathcal{X}$.

Problem PNC2 is equivalent to problem PNC1 if we view the solutions as homogeneous coordinates (up to a nonzero scalar).

The main problem in finding a good relaxation of problem PNC2 is that it is very difficult to enforce the condition $X \in \mathcal{X}$.

The first natural relaxation of problem PNC2 is to drop the condition that $X \in \mathcal{X}$, and we obtain

Problem $(*_2)$

$$\begin{aligned} &\text{minimize} && \operatorname{tr}(X^\top L X) \\ &\text{subject to} && X^\top D X = I. \end{aligned}$$

Problem $(*_2)$

$$\begin{array}{ll} \text{minimize} & \operatorname{tr}(X^\top L X) \\ \text{subject to} & X^\top D X = I. \end{array}$$

Actually, since the discrete solutions $X \in \mathcal{X}$ that we are ultimately seeking are solutions of problem PNC1, the preferred relaxation is the one obtained from problem PNC1 by dropping the condition $X \in \mathcal{X}$, and simply requiring that $X^j \neq 0$, for $j = 1, \ldots, K$:

Problem $(*_1)$

$$\begin{aligned} \text{minimize} \quad & \sum_{j=1}^{K} \frac{(X^j)^\top L X^j}{(X^j)^\top D X^j} \\ \text{subject to} \quad & (X^i)^\top D X^j = 0, X^j \neq 0 \qquad 1 \leq i, j \leq K,\ i \neq j. \end{aligned}$$

Problem $(*_1)$

$$\begin{aligned} \text{minimize} \quad & \sum_{j=1}^{K} \frac{(X^j)^\top L X^j}{(X^j)^\top D X^j} \\ \text{subject to} \quad & (X^i)^\top D X^j = 0, X^j \neq 0 \qquad 1 \leq i, j \leq K,\ i \neq j. \end{aligned}$$

Problem $(*_2)$

$$\begin{aligned} \text{minimize} \quad & \operatorname{tr}(X^\top L X) \\ \text{subject to} \quad & X^\top D X = I, \qquad X^j \neq 0 \qquad 1 \leq j \leq K. \end{aligned}$$

Proposition 3

For any orthogonal $K \times K$ matrix R, any symmetric $N \times N$ matrix A, and any $N \times K$ matrix $X = [X^1 \cdots X^K]$, the following properties hold:

(1) $\mu(X) = \mathrm{tr}(\Lambda^{-1} X^\top L X)$, *where*

$$\Lambda = \mathrm{diag}((X^1)^\top D X^1, \ldots, (X^K)^\top D X^K).$$

(2) *If* $(X^1)^\top D X^1 = \cdots = (X^K)^\top D X^K = \alpha^2$, *then*

$$\mu(X) = \mu(XR) = \frac{1}{\alpha^2}\mathrm{tr}(X^\top L X).$$

(3) *The condition* $X^\top A X = \alpha^2 I$ *is preserved if* X *is replaced by* XR.

(4) *The condition* $X(X^\top X)^{-1} X^\top \mathbf{1} = \mathbf{1}$ *is preserved if* X *is replaced by* XR.

Every solution Z of problem $(*_2)$ yields a *family of solutions* of problem $(*_1)$; namely, all matrices of the form $ZR\Lambda$, where $R \in \mathbf{O}(K)$ and Λ is a diagonal invertible matrix.

Every solution Z of problem $(*_2)$ yields a *family of solutions* of problem $(*_1)$; namely, all matrices of the form $ZR\Lambda$, where $R \in \mathbf{O}(K)$ and Λ is a diagonal invertible matrix.

We will take advantage of this fact in looking for a discrete solution X "close" to a solution Z of the relaxed problem $(*_2)$.

Every solution Z of problem $(*_2)$ yields a *family of solutions* of problem $(*_1)$; namely, all matrices of the form $ZR\Lambda$, where $R \in \mathbf{O}(K)$ and Λ is a diagonal invertible matrix.

We will take advantage of this fact in looking for a discrete solution X "close" to a solution Z of the relaxed problem $(*_2)$.

Observe that a matrix is of the form $R\Lambda$ with $R \in \mathbf{O}(K)$ and Λ a diagonal invertible matrix iff its columns are nonzero and pairwise orthogonal.

If we make the change of variable $Y = D^{1/2}X$ or equivalently $X = D^{-1/2}Y$, we get

If we make the change of variable $Y = D^{1/2}X$ or equivalently $X = D^{-1/2}Y$, we get

Problem $(**_2)$

$$\begin{aligned} &\text{minimize} && \operatorname{tr}(Y^\top D^{-1/2} L D^{-1/2} Y) \\ &\text{subject to} && Y^\top Y = I. \end{aligned}$$

We recognize $L_{\text{sym}} = D^{-1/2} L D^{-1/2}$.

If we make the change of variable $Y = D^{1/2}X$ or equivalently $X = D^{-1/2}Y$, we get

Problem $(**_2)$

$$\begin{aligned} &\text{minimize} && \operatorname{tr}(Y^\top D^{-1/2} L D^{-1/2} Y) \\ &\text{subject to} && Y^\top Y = I. \end{aligned}$$

We recognize $L_{\text{sym}} = D^{-1/2} L D^{-1/2}$.

We pass from a solution Y of problem $(**_2)$ to a solution Z of problem $(*_2)$ by $Z = D^{-1/2}Y$.

It is not a priori obvious that the minimum of $\mathrm{tr}(Y^\top L_{\mathrm{sym}} Y)$ over all $N \times K$ matrices Y satisfying $Y^\top Y = I$ is equal to the sum $\nu_1 + \cdots + \nu_K$ of the first K eigenvalues of $L_{\mathrm{sym}} = D^{-1/2} L D^{-1/2}$.

It is not a priori obvious that the minimum of $\mathrm{tr}(Y^\top L_{\mathrm{sym}} Y)$ over all $N \times K$ matrices Y satisfying $Y^\top Y = I$ is equal to the sum $\nu_1 + \cdots + \nu_K$ of the first K eigenvalues of $L_{\mathrm{sym}} = D^{-1/2} L D^{-1/2}$.

Fortunately, the Poincaré separation theorem guarantees that the sum of the K smallest eigenvalues of L_{sym} is a lower bound for $\mathrm{tr}(Y^\top L_{\mathrm{sym}} Y)$.

It is not a priori obvious that the minimum of $\mathrm{tr}(Y^\top L_{\mathrm{sym}} Y)$ over all $N \times K$ matrices Y satisfying $Y^\top Y = I$ is equal to the sum $\nu_1 + \cdots + \nu_K$ of the first K eigenvalues of $L_{\mathrm{sym}} = D^{-1/2} L D^{-1/2}$.

Fortunately, the Poincaré separation theorem guarantees that the sum of the K smallest eigenvalues of L_{sym} is a lower bound for $\mathrm{tr}(Y^\top L_{\mathrm{sym}} Y)$.

Furthermore, the minimum of problem $(**_2)$ is achieved by any K unit eigenvectors $(u_1, \ldots, u_K)$ associated with the smallest eigenvalues

$$0 = \nu_1 \leq \nu_2 \leq \ldots \leq \nu_K$$

of L_{sym}.

We may assume that $\nu_2 > 0$, namely that the underlying graph is connected (otherwise, we work with each connected component), in which case $Y^1 = D^{1/2}\mathbf{1} / \left\| D^{1/2}\mathbf{1} \right\|_2$, because $\mathbf{1}$ is in the nullspace of L.

We may assume that $\nu_2 > 0$, namely that the underlying graph is connected (otherwise, we work with each connected component), in which case $Y^1 = D^{1/2}\mathbf{1} / \left\| D^{1/2}\mathbf{1} \right\|_2$, because $\mathbf{1}$ is in the nullspace of L.

Theorem 3

The matrix $Z = D^{-1/2}Y$ where $Y = [u_1 \ \dots \ u_K]$ is the matrix of unit eigenvectors associated with the K smallest eigenvalues of L_{sym} yields a minimum of our relaxed problem $(*_1)$

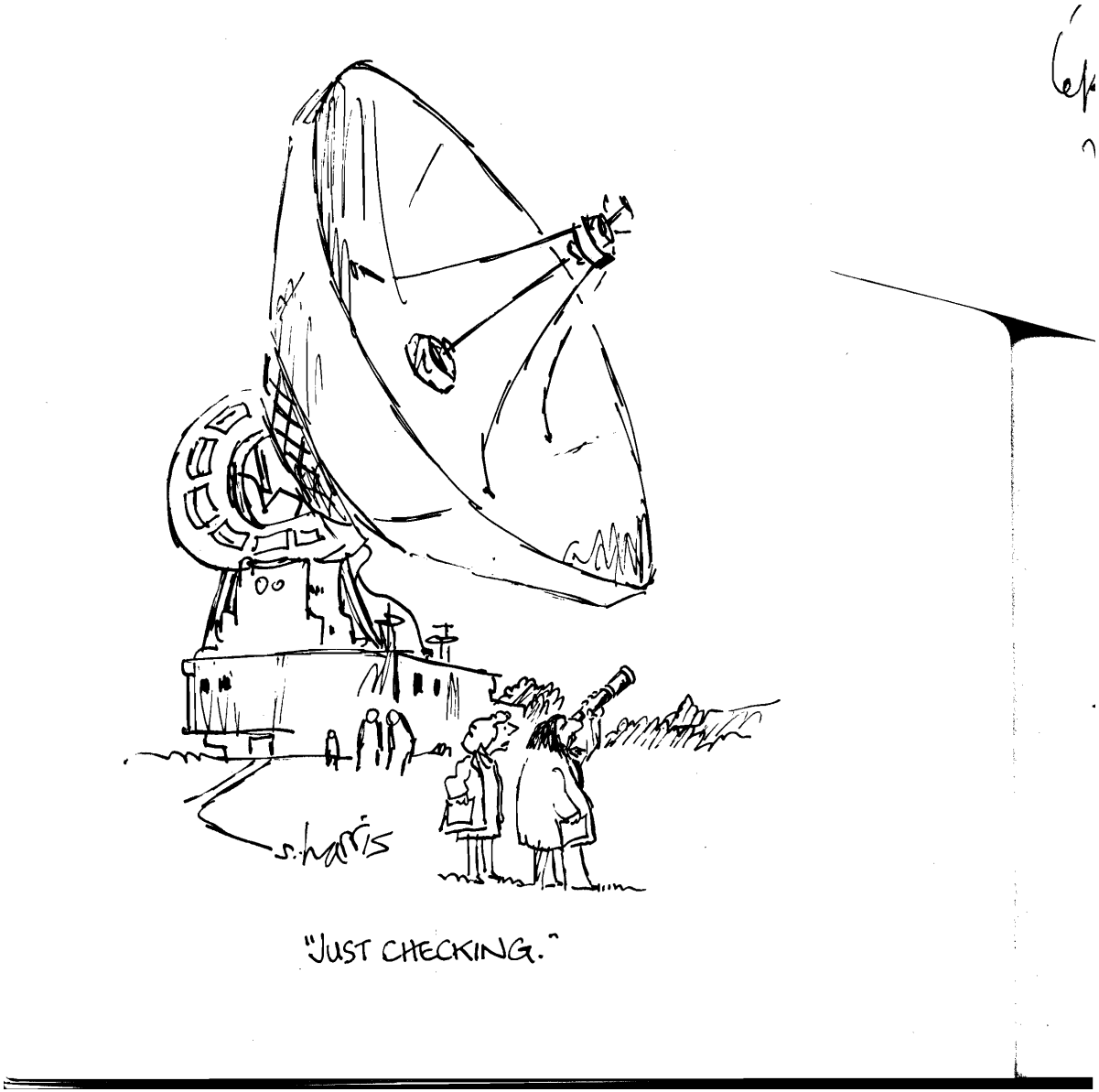


Figure 11: Just Checking!

In practice the matrix W can be too large to use SVD.

In practice the matrix W can be too large to use SVD.

We can use randomized algorithms to compute a partial matrix decomposition. There are fast approximate SVD algorithms due to Halko, Martinsson and Tropp.

In practice the matrix W can be too large to use SVD.

We can use randomized algorithms to compute a partial matrix decomposition. There are fast approximate SVD algorithms due to Halko, Martinsson and Tropp.

Finding a discrete solution close to a continuous approximation is nontrivial. Yu and Shi proposed a method that we have generalized.

5. Signed Graphs

Intuitively, in a weighted graph, an edge with a positive weight denotes similarity or proximity of its endpoints.

5. Signed Graphs

Intuitively, in a weighted graph, an edge with a positive weight denotes similarity or proximity of its endpoints.

For many reasons, it is desirable to allow edges labeled with *negative weights*, the intuition being that *a negative weight indicates dissimilarity or distance*.

5. Signed Graphs

Intuitively, in a weighted graph, an edge with a positive weight denotes similarity or proximity of its endpoints.

For many reasons, it is desirable to allow edges labeled with *negative weights*, the intuition being that *a negative weight indicates dissimilarity or distance*.

Weighted graphs for which the weight matrix is a symmetric matrix in which negative and positive entries are allowed are called *signed graphs*.

Such graphs (with weights $(-1, 0, +1)$) were introduced as early as 1953 by Harary, to model social relations involving disliking, indifference, and liking.

Such graphs (with weights $(-1, 0, +1)$) were introduced as early as 1953 by Harary, to model social relations involving disliking, indifference, and liking.

The problem of clustering the nodes of a signed graph arises naturally as a generalization of the clustering problem for weighted graphs.

Given the signed matrix

$$W = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & -1 & 1 & 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 & -1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & -1 & 0 & 1 & 0 & -1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & -1 & -1 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & -1 & 0 \end{pmatrix}$$

the corresponding signed graph is

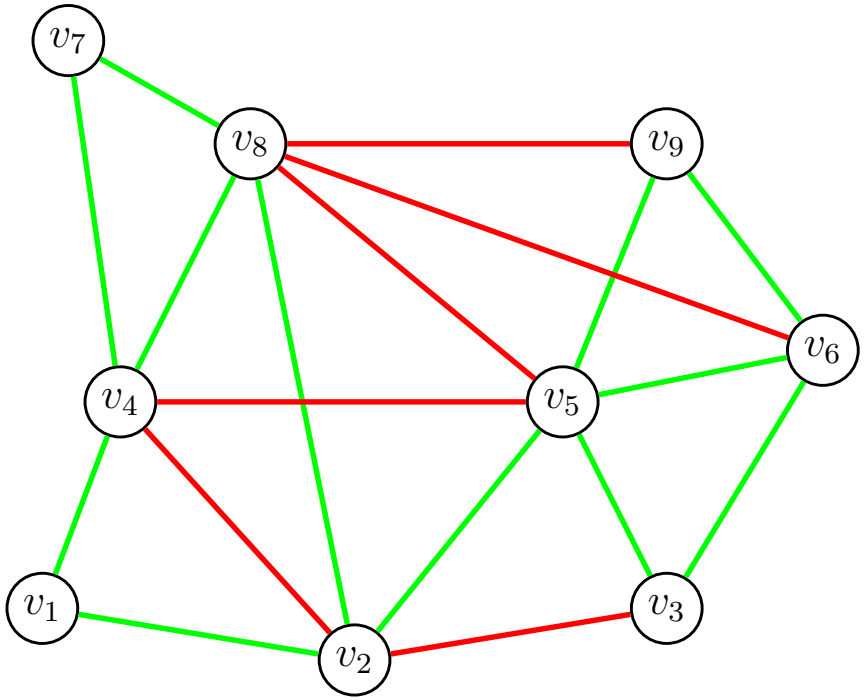


Figure 12: A signed graph G.

The first obstacle is that the degree matrix may now contain *zero or negative entries*.

The first obstacle is that the degree matrix may now contain *zero or negative entries*.

As a consequence, the Laplacian L may no longer be positive semidefinite, and worse, $D^{-1/2}$ may not exist.

The first obstacle is that the degree matrix may now contain *zero or negative entries*.

As a consequence, the Laplacian L may no longer be positive semidefinite, and worse, $D^{-1/2}$ may not exist.

A simple remedy is to use the *absolute values of the weights in the degree matrix!*

The first obstacle is that the degree matrix may now contain *zero or negative entries*.

As a consequence, the Laplacian L may no longer be positive semidefinite, and worse, $D^{-1/2}$ may not exist.

A simple remedy is to use the *absolute values of the weights in the degree matrix!*

This idea applied to signed graph with weights $(-1, 0, 1)$ occurs in Hou. Kolluri, Shewchuk and O'Brien take the natural step of using absolute values of weights in the degree matrix in their original work on surface reconstruction from noisy point clouds.

Kunegis et al. appear to be the first to make a systematic study of spectral methods applied to signed graphs.

Kunegis et al. appear to be the first to make a systematic study of spectral methods applied to signed graphs.

However, it should be noted that only 2-clustering is considered in the above papers.

Kunegis et al. appear to be the first to make a systematic study of spectral methods applied to signed graphs.

However, it should be noted that only 2-clustering is considered in the above papers.

The trick of using absolute values of weights in the degree matrix allows the whole machinery that we have presented to be used to attack the problem of clustering signed graphs using normalized cuts.

Kunegis et al. appear to be the first to make a systematic study of spectral methods applied to signed graphs.

However, it should be noted that only 2-clustering is considered in the above papers.

The trick of using absolute values of weights in the degree matrix allows the whole machinery that we have presented to be used to attack the problem of clustering signed graphs using normalized cuts.

This requires a modification of the notion of normalized cut. Degrees and volumes use absolute values of weights, but not the cuts.

If (V, W) is a signed graph, where W is an $m \times m$ symmetric matrix with zero diagonal entries and with the other entries $w_{ij} \in \mathbb{R}$ arbitrary, for any node $v_i \in V$, the *signed degree* of v_i is defined as

$$\overline{d}_i = \overline{d}(v_i) = \sum_{j=1}^{m} |w_{ij}|,$$

and the *signed degree matrix* $\overline{D}$ as

$$\overline{D} = \mathrm{diag}(\overline{d}(v_1), \ldots, \overline{d}(v_m)).$$

If (V, W) is a signed graph, where W is an $m \times m$ symmetric matrix with zero diagonal entries and with the other entries $w_{ij} \in \mathbb{R}$ arbitrary, for any node $v_i \in V$, the *signed degree* of v_i is defined as

$$\overline{d}_i = \overline{d}(v_i) = \sum_{j=1}^{m} |w_{ij}|,$$

and the *signed degree matrix* $\overline{D}$ as

$$\overline{D} = \mathrm{diag}(\overline{d}(v_1), \ldots, \overline{d}(v_m)).$$

For any subset A of the set of nodes V, let

$$\mathrm{vol}(A) = \sum_{v_i \in A} \overline{d}_i.$$

For any two subsets A and B of V, define $\mathrm{links}^+(A,B)$, $\mathrm{links}^-(A,B)$, and $\mathrm{cut}(A,\overline{A})$ by

$$\mathrm{links}^+(A,B) = \sum_{\substack{v_i \in A, v_j \in B \\ w_{ij} > 0}} w_{ij}$$

$$\mathrm{links}^-(A,B) = \sum_{\substack{v_i \in A, v_j \in B \\ w_{ij} < 0}} -w_{ij}$$

$$\mathrm{cut}(A,\overline{A}) = \sum_{\substack{v_i \in A, v_j \in \overline{A} \\ w_{ij} \neq 0}} |w_{ij}|.$$

For any two subsets A and B of V, define $\text{links}^+(A, B)$, $\text{links}^-(A, B)$, and $\text{cut}(A, \overline{A})$ by

$$\text{links}^+(A, B) = \sum_{\substack{v_i \in A, v_j \in B \\ w_{ij} > 0}} w_{ij}$$

$$\text{links}^-(A, B) = \sum_{\substack{v_i \in A, v_j \in B \\ w_{ij} < 0}} -w_{ij}$$

$$\text{cut}(A, \overline{A}) = \sum_{\substack{v_i \in A, v_j \in \overline{A} \\ w_{ij} \neq 0}} |w_{ij}|.$$

Note that

$$\text{cut}(A, \overline{A}) = \text{links}^+(A, \overline{A}) + \text{links}^-(A, \overline{A}).$$

Then, the *signed Laplacian* $\overline{L}$ is defined by

$$\overline{L} = \overline{D} - W,$$

and its normalized version $\overline{L}_{\mathrm{sym}}$ by

$$\overline{L}_{\mathrm{sym}} = \overline{D}^{-1/2}\,\overline{L}\,\overline{D}^{-1/2} = I - \overline{D}^{-1/2} W \overline{D}^{-1/2}.$$

Then, the *signed Laplacian* $\overline{L}$ is defined by

$$\overline{L} = \overline{D} - W,$$

and its normalized version $\overline{L}_{\mathrm{sym}}$ by

$$\overline{L}_{\mathrm{sym}} = \overline{D}^{-1/2}\,\overline{L}\,\overline{D}^{-1/2} = I - \overline{D}^{-1/2} W \overline{D}^{-1/2}.$$

For a graph without isolated vertices, we have $\overline{d}(v_i) > 0$ for $i = 1, \ldots, m$, so $\overline{D}^{-1/2}$ is well defined.

Then, the *signed Laplacian* $\overline{L}$ is defined by

$$\overline{L} = \overline{D} - W,$$

and its normalized version $\overline{L}_{\mathrm{sym}}$ by

$$\overline{L}_{\mathrm{sym}} = \overline{D}^{-1/2}\,\overline{L}\,\overline{D}^{-1/2} = I - \overline{D}^{-1/2} W \overline{D}^{-1/2}.$$

For a graph without isolated vertices, we have $\overline{d}(v_i) > 0$ for $i = 1, \ldots, m$, so $\overline{D}^{-1/2}$ is well defined.

The signed Laplacian is symmetric positive semidefinite.

The signed Laplacian of the matrix W given earlier is

$$\overline{L} = \begin{pmatrix} 2 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 5 & 1 & 1 & -1 & 0 & 0 & -1 & 0 \\ 0 & 1 & 3 & 0 & -1 & -1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 5 & 1 & 0 & -1 & -1 & 0 \\ 0 & -1 & -1 & 1 & 6 & -1 & 0 & 1 & -1 \\ 0 & 0 & -1 & 0 & -1 & 4 & 0 & 1 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 2 & -1 & 0 \\ 0 & -1 & 0 & -1 & 1 & 1 & -1 & 6 & 1 \\ 0 & 0 & 0 & 0 & -1 & -1 & 0 & 1 & 3 \end{pmatrix}.$$

The signed Laplacian of the matrix W given earlier is

$$\overline{L} = \begin{pmatrix} 2 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 5 & 1 & 1 & -1 & 0 & 0 & -1 & 0 \\ 0 & 1 & 3 & 0 & -1 & -1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 5 & 1 & 0 & -1 & -1 & 0 \\ 0 & -1 & -1 & 1 & 6 & -1 & 0 & 1 & -1 \\ 0 & 0 & -1 & 0 & -1 & 4 & 0 & 1 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 2 & -1 & 0 \\ 0 & -1 & 0 & -1 & 1 & 1 & -1 & 6 & 1 \\ 0 & 0 & 0 & 0 & -1 & -1 & 0 & 1 & 3 \end{pmatrix}.$$

The eigenvalues of $\overline{L}$ are

0.5175, 1.5016, 1.7029, 2.7058, 3.7284, 4.9604, 5.6026, 7.0888, 8.1921.

The matrix $\overline{L}$ is actually positive definite!

For any real $\lambda \in \mathbb{R}$, define $\mathrm{sgn}(\lambda)$ by

$$\mathrm{sgn}(\lambda) = \begin{cases} +1 & \text{if } \lambda > 0 \\ -1 & \text{if } \lambda < 0 \\ 0 & \text{if } \lambda = 0. \end{cases}$$

For any real $\lambda \in \mathbb{R}$, define $\mathrm{sgn}(\lambda)$ by

$$\mathrm{sgn}(\lambda) = \begin{cases} +1 & \text{if } \lambda > 0 \\ -1 & \text{if } \lambda < 0 \\ 0 & \text{if } \lambda = 0. \end{cases}$$

Proposition 4

For any $m \times m$ symmetric matrix $W = (w_{ij})$, if we let $\overline{L} = \overline{D} - W$ where $\overline{D}$ is the signed degree matrix associated with W, then we have

$$x^\top \overline{L} x = \frac{1}{2} \sum_{i,j=1}^{m} |w_{ij}|(x_i - \mathrm{sgn}(w_{ij})x_j)^2 \quad \text{for all } x \in \mathbb{R}^m.$$

Consequently, $\overline{L}$ is positive semidefinite.

6. Signed Normalized Cuts

As before, given a partition of V into K clusters $(A_1, \ldots, A_K)$, if we represent the jth block of this partition by a vector X^j such that

$$X_i^j = \begin{cases} a_j & \text{if } v_i \in A_j \\ 0 & \text{if } v_i \notin A_j, \end{cases}$$

for some $a_j \neq 0$, then we have the following result.

6. Signed Normalized Cuts

As before, given a partition of V into K clusters $(A_1, \ldots, A_K)$, if we represent the jth block of this partition by a vector X^j such that

$$X_i^j = \begin{cases} a_j & \text{if } v_i \in A_j \\ 0 & \text{if } v_i \notin A_j, \end{cases}$$

for some $a_j \neq 0$, then we have the following result.

Proposition 5

For any vector X^j representing the jth block of a partition $(A_1, \ldots, A_K)$ of V, we have

$$(X^j)^\top \overline{L} X^j = a_j^2(\mathrm{cut}(A_j, \overline{A_j}) + 2\mathrm{links}^-(A_j, A_j)).$$

With the revised definition of $\mathrm{vol}(A_j)$, we deduce that

$$\frac{(X^j)^\top \overline{L} X^j}{(X^j)^\top \overline{D} X^j} = \frac{\mathrm{cut}(A_j, \overline{A_j}) + 2\mathrm{links}^-(A_j, A_j)}{\mathrm{vol}(A_j)}.$$

With the revised definition of $\mathrm{vol}(A_j)$, we deduce that

$$\frac{(X^j)^\top \overline{L} X^j}{(X^j)^\top \overline{D} X^j} = \frac{\mathrm{cut}(A_j, \overline{A_j}) + 2\mathrm{links}^-(A_j, A_j)}{\mathrm{vol}(A_j)}.$$

The calculations of the previous paragraph suggest the following definition.

Definition 4

The *signed normalized cut* $\mathrm{sNcut}(A_1, \ldots, A_K)$ of the partition $(A_1, \ldots, A_K)$ is defined as

$$\mathrm{sNcut}(A_1, \ldots, A_K) = \sum_{j=1}^{K} \frac{\mathrm{cut}(A_j, \overline{A_j})}{\mathrm{vol}(A_j)} + 2 \sum_{j=1}^{K} \frac{\mathrm{links}^{-}(A_j, A_j)}{\mathrm{vol}(A_j)}.$$

Based on previous computations, we have

$$\mathrm{sNcut}(A_1, \ldots, A_K) = \sum_{j=1}^{K} \frac{(X^j)^\top \overline{L} X^j}{(X^j)^\top \overline{D} X^j},$$

where X is the $N \times K$ matrix whose jth column is X^j.

Based on previous computations, we have

$$\mathrm{sNcut}(A_1, \ldots, A_K) = \sum_{j=1}^{K} \frac{(X^j)^\top \overline{L} X^j}{(X^j)^\top \overline{D} X^j},$$

where X is the $N \times K$ matrix whose jth column is X^j.

Therefore, *this is the same problem as in the unsigned case, with L replaced by $\overline{L}$ and D replaced by $\overline{D}$*.

Observe that minimizing $\mathrm{sNcut}(A_1, \ldots, A_K)$ amounts to

1. minimizing the number of positive and negative edges between clusters, and also
2. minimizing the number of negative edges within clusters.

Observe that minimizing $\mathrm{sNcut}(A_1, \ldots, A_K)$ amounts to

1. minimizing the number of positive and negative edges between clusters, and also
2. minimizing the number of negative edges within clusters.

This second minimization captures the intuition that *nodes connected by a negative edge should not be together* (they do not "like" each other; they should be far from each other).

Observe that minimizing $\mathrm{sNcut}(A_1, \ldots, A_K)$ amounts to

1. minimizing the number of positive and negative edges between clusters, and also
2. minimizing the number of negative edges within clusters.

This second minimization captures the intuition that *nodes connected by a negative edge should not be together* (they do not "like" each other; they should be far from each other).

The method has been applied by Joao Sedoc to semantic world clustering, where there are synonyms and antonyms

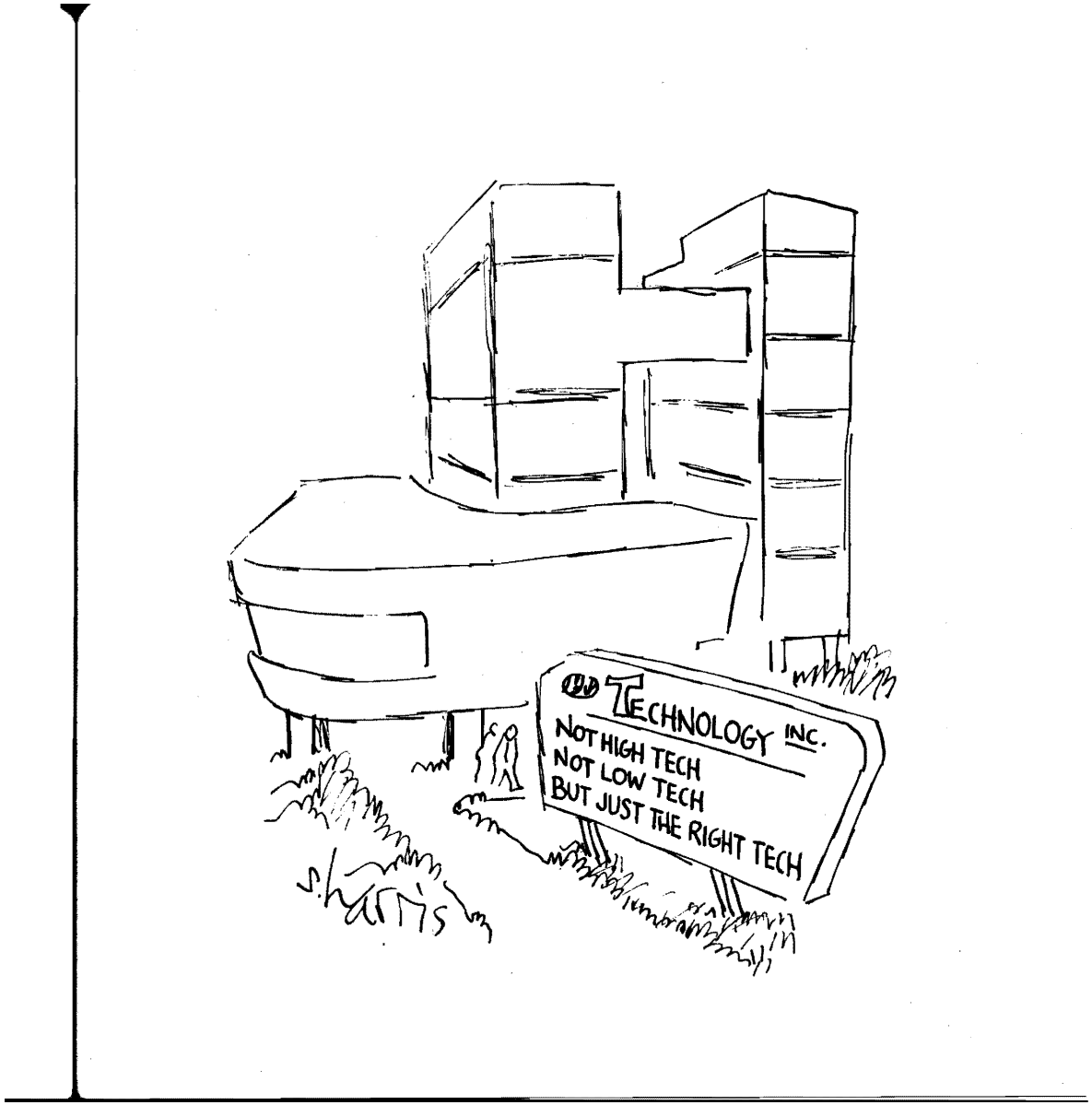


Figure 13: Just the right tech

7. What if the weight matrix is very large?

Given a $n \times n$ matrix W for $n \approx 10^5$, we want to find a rank-k approximation, with $k \ll n$ (where $k \sim$ the number of clusters),

$$\underset{n \times n}{W} \approx \underset{n \times k}{E} \underset{k \times n}{F^\top}.$$

7. What if the weight matrix is very large?

Given a $n \times n$ matrix W for $n \approx 10^5$, we want to find a rank-k approximation, with $k \ll n$ (where $k \sim$ the number of clusters),

$$\underset{n \times n}{W} \approx \underset{n \times k}{E} \underset{k \times n}{F^\top}.$$

The columns of E and F are required to be orthogonal.

7. What if the weight matrix is very large?

Given a $n \times n$ matrix W for $n \approx 10^5$, we want to find a rank-k approximation, with $k \ll n$ (where $k \sim$ the number of clusters),

$$\underset{n \times n}{W} \approx \underset{n \times k}{E} \underset{k \times n}{F^\top}.$$

The columns of E and F are required to be orthogonal.

This problem requires algorithms for computing the *Singular Value Decomposition (SVD)*.

What if the weight matrix is very large?

We use *randomized algorithms* that compute partial matrix decompositions (Halko, Martinsson, Tropp).

What if the weight matrix is very large?

We use *randomized algorithms* that compute partial matrix decompositions (Halko, Martinsson, Tropp).

- For a dense input matrix, randomized algorithms require $O(mn\log(k))$ floating-point operations in contrast with $O(mnk)$ for classical algorithms.
- randomized techniques require only a constant number of passes over the data.
- probabilistic bound on accuracy.

What if the weight matrix is very large?

We use *randomized algorithms* that compute partial matrix decompositions (Halko, Martinsson, Tropp).

- For a dense input matrix, randomized algorithms require $O(mn\log(k))$ floating-point operations in contrast with $O(mnk)$ for classical algorithms.
- randomized techniques require only a constant number of passes over the data.
- probabilistic bound on accuracy.

We want to find $A \approx U\Sigma_k V^\top$.

Fast SVD

Given an $m \times n$ matrix A and integers ℓ and q, this algorithm computes an $m \times \ell$ orthonormal matrix Q whose range approximates the range of A.

Fast SVD

Given an $m \times n$ matrix A and integers ℓ and q, this algorithm computes an $m \times \ell$ orthonormal matrix Q whose range approximates the range of A.

Algorithm 2 Computing SVD using Randomized Algorithm

1: Draw an $n \times \ell$ Gaussian random matrix Ω.
2: Form the $m \times \ell$ matrix $Y = (AA^\top)^q A\Omega$ via alternating application of A and $A^\top$.
3: Construct an $m \times \ell$ matrix Q whose columns form an orthonormal basis for the range of Y , e.g., via the QR factorization $Y = QR$.
4: Form $B = Q^\top A$.
5: Compute an SVD of the small matrix: $B = \tilde{U}\Sigma V^\top$.
6: Set $U = Q\tilde{U}$.

Probabilistic Bounds

What is the error $e_k = \left\| A - U\Sigma_k V^\top \right\|$?

Probabilistic Bounds

What is the error $e_k = \left\| A - U\Sigma_k V^\top \right\|$?

Theorem 5

Eckart-Young Theorem: *e_k is bounded from below by the $(k+1)$th singular value σ_{k+1} of A.*

Probabilistic Bounds

What is the error $e_k = \left\| A - U\Sigma_k V^\top \right\|$?

Theorem 5

Eckart-Young Theorem: *e_k is bounded from below by the $(k+1)$th singular value σ_{k+1} of A.*

We want e_k to be close to σ_{k+1}, but this is not true.
The expectation of $\frac{e_k}{\sigma_{k+1}}$ is large with high variance.

Probabilistic Bounds

What is the error $e_k = \left\| A - U\Sigma_k V^\top \right\|$?

Theorem 5

Eckart-Young Theorem: *e_k is bounded from below by the $(k+1)$th singular value σ_{k+1} of A.*

We want e_k to be close to σ_{k+1}, but this is not true.
The expectation of $\frac{e_k}{\sigma_{k+1}}$ is large with high variance.

However, using oversampling where we compute $k + p$ where $p = k$ solves this issue.

Probabilistic Bounds

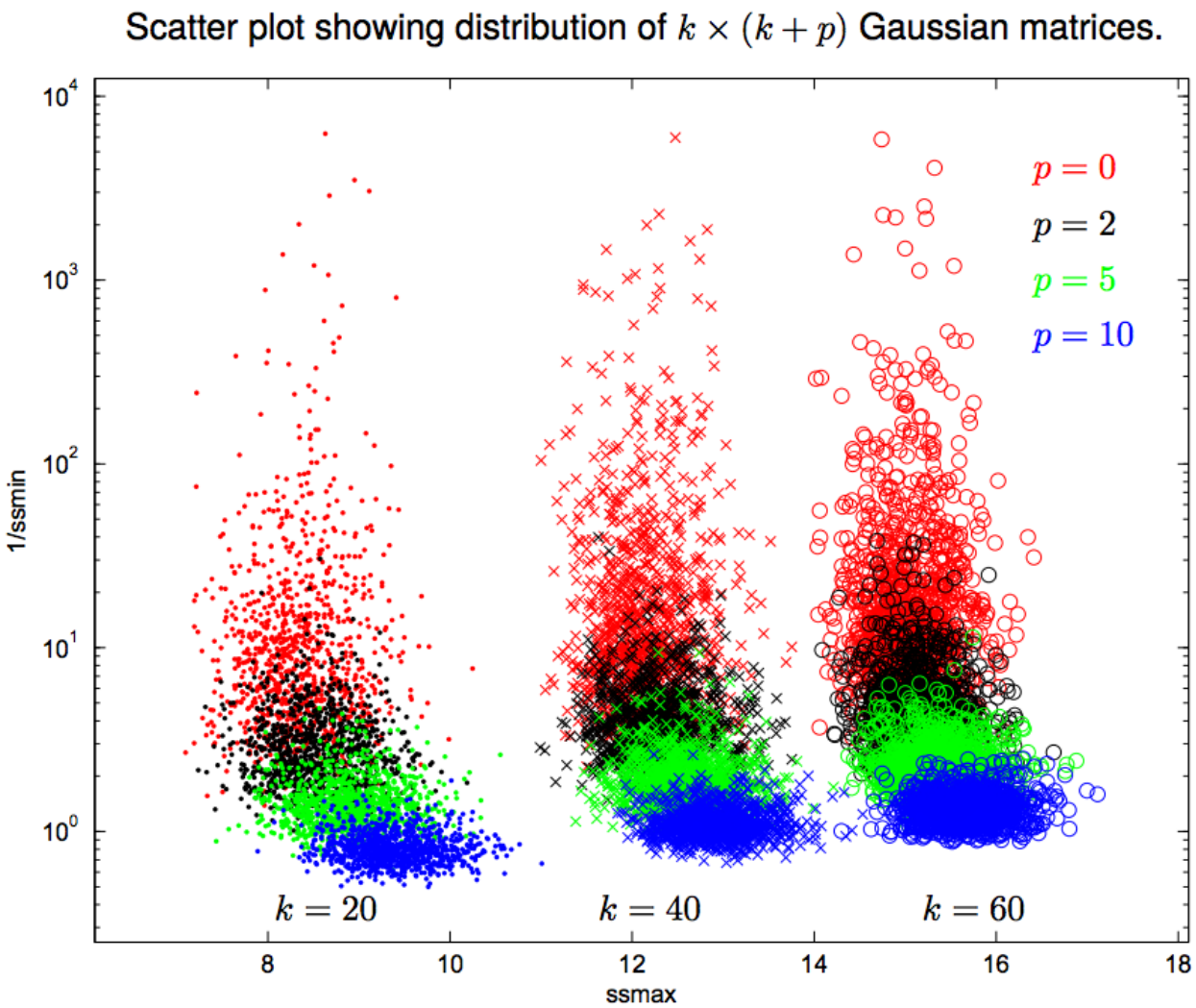
Theorem 6

Suppose that A is a real $m \times n$ matrix. Select an exponent q and a target number k of singular vectors, where $2 \leq k \leq 0.5 \min\{m, n\}$. Randomized SVD algorithm to obtain a rank-$2k$ factorization $U\Sigma V^\top$. Then

$$\mathbb{E}\left[\|A - U\Sigma_k V^\top\|\right] \leq \left[1 + 4\sqrt{\frac{2\min\{m,n\}}{k-1}}\right]^{1/(2q+1)} \sigma_{k+1},$$

where $\mathbb{E}$ denotes expectation with respect to the random test matrix and σ_{k+1} is the $(k+1)$th singular value of A. (Halko, Martinsson, Tropp 2011)

Bound Example



$1/\sigma_{\min}$ *is plotted against* $\sigma_{\max}$.

Figure 14: Beethoven and Twitter

8. Finding a Discrete Solution Close to a Continuous Approximation

The next step is to find an exact solution $(\mathbb{P}(X^1), \ldots, \mathbb{P}(X^K)) \in \mathbb{P}(\mathcal{K})$ which is the closest (in a suitable sense) to our approximate solution $(Z^1, \ldots, Z^K)$.

8. Finding a Discrete Solution Close to a Continuous Approximation

The next step is to find an exact solution $(\mathbb{P}(X^1), \ldots, \mathbb{P}(X^K)) \in \mathbb{P}(\mathcal{K})$ which is the closest (in a suitable sense) to our approximate solution $(Z^1, \ldots, Z^K)$.

Since the solutions ZQ of $(*_1)$ are all equivalent (they yield the same minimum for the normalized cut), it makes sense to look for a discrete solution X closest to one of these ZQ.

If we use the Riemannian metric on $\mathbb{RP}^{N-1}$ induced by the Euclidean metric on $\mathbb{R}^N$ and the product distance on $(\mathbb{RP}^{N-1})^K$ given by

$$d\big((\mathbb{P}(X^1),\dots,\mathbb{P}(X^K)),(\mathbb{P}(Z^1),\dots,\mathbb{P}(Z^K))\big) = \sum_{j=1}^{K} d(\mathbb{P}(X^j),\mathbb{P}(Z^j)),$$

If we use the Riemannian metric on $\mathbb{RP}^{N-1}$ induced by the Euclidean metric on $\mathbb{R}^N$ and the product distance on $(\mathbb{RP}^{N-1})^K$ given by

$$d\big((\mathbb{P}(X^1),\dots,\mathbb{P}(X^K)),(\mathbb{P}(Z^1),\dots,\mathbb{P}(Z^K))\big) = \sum_{j=1}^{K} d(\mathbb{P}(X^j),\mathbb{P}(Z^j)),$$

it can be shown that minimizing the distance
$d\big((\mathbb{P}(X^1),\dots,\mathbb{P}(X^K)),(\mathbb{P}(Z^1),\dots,\mathbb{P}(Z^K))\big)$ in $(\mathbb{RP}^{N-1})^K$
is equivalent to minimizing

$$\sum_{j=1}^{K} \left\|X^j - Z^j\right\|_2, \quad \text{subject to} \quad \left\|X^j\right\|_2 = \left\|Z^j\right\|_2 \ (j = 1,\dots,K).$$

We are not aware of any optimization method to solve the above problem, which seems difficult to tackle due to constraints $\left\|X^j\right\|_2 = \left\|Z^j\right\|_2$ $(j = 1, \ldots, K)$.

We are not aware of any optimization method to solve the above problem, which seems difficult to tackle due to constraints $\left\|X^j\right\|_2 = \left\|Z^j\right\|_2$ $(j = 1, \ldots, K)$.

Therefore, we drop these constraints and attempt to minimize

$$\|X - Z\|_F^2 = \sum_{j=1}^{K} \left\|X^j - Z^j\right\|_2^2,$$

the Frobenius norm of $X - Z$. This is implicitly the choice made by Yu.

Inspired by Yu and the previous discussion, given a solution Z of problem $(*_2)$, we look for pairs (X, Q) with $X \in \mathcal{X}$ and where Q is a $K \times K$ matrix with nonzero and pairwise orthogonal columns, with $\|X\|_F = \|Z\|_F$, that minimize

$$\varphi(X, Q) = \|X - ZQ\|_F .$$

Inspired by Yu and the previous discussion, given a solution Z of problem $(*_2)$, we look for pairs (X, Q) with $X \in \mathcal{X}$ and where Q is a $K \times K$ matrix with nonzero and pairwise orthogonal columns, with $\|X\|_F = \|Z\|_F$, that minimize

$$\varphi(X, Q) = \|X - ZQ\|_F .$$

Yu and Shi consider the special case where $Q \in \mathbf{O}(K)$.

Inspired by Yu and the previous discussion, given a solution Z of problem $(*_2)$, we look for pairs (X, Q) with $X \in \mathcal{X}$ and where Q is a $K \times K$ matrix with nonzero and pairwise orthogonal columns, with $\|X\|_F = \|Z\|_F$, that minimize

$$\varphi(X, Q) = \|X - ZQ\|_F .$$

Yu and Shi consider the special case where $Q \in \mathbf{O}(K)$.

We consider the more general case where $Q = R\Lambda$, with $R \in \mathbf{O}(K)$ and Λ is a diagonal invertible matrix.

The key to minimizing $\|X - ZQ\|_F$ rests on the following result:

$$\|X - ZQ\|_F^2 = \|X\|_F^2 - 2\mathrm{tr}(Q^\top Z^\top X) + \mathrm{tr}(Z^\top ZQQ^\top).$$

The key to minimizing $\|X - ZQ\|_F$ rests on the following result:

$$\|X - ZQ\|_F^2 = \|X\|_F^2 - 2\mathrm{tr}(Q^\top Z^\top X) + \mathrm{tr}(Z^\top ZQQ^\top).$$

Therefore, since $\|X\|_F = \|Z\|_F$ is fixed, minimizing $\|X - ZQ\|_F^2$ is equivalent to

minimizing $-2\mathrm{tr}(Q^\top Z^\top X) + \mathrm{tr}(Z^\top ZQQ^\top)$.

The key to minimizing $\|X - ZQ\|_F$ rests on the following result:

$$\|X - ZQ\|_F^2 = \|X\|_F^2 - 2\mathrm{tr}(Q^\top Z^\top X) + \mathrm{tr}(Z^\top ZQQ^\top).$$

Therefore, since $\|X\|_F = \|Z\|_F$ is fixed, minimizing $\|X - ZQ\|_F^2$ is equivalent to

minimizing $-2\mathrm{tr}(Q^\top Z^\top X) + \mathrm{tr}(Z^\top ZQQ^\top)$.

This is a hard problem because it is a nonlinear optimization problem involving two matrix unknowns X and Q.

To simplify the problem, we proceed by *alternating steps* during which

1. we *minimize* $\varphi(X, Q) = \|X - ZQ\|_F$ *with respect to* X *holding* Q *fixed*, and
2. steps during which we *minimize* $\varphi(X, Q) = \|X - ZQ\|_F$ *with respect to* Q *holding* X *fixed*.

To simplify the problem, we proceed by *alternating steps* during which

1. we *minimize* $\varphi(X, Q) = \|X - ZQ\|_F$ *with respect to* X *holding* Q *fixed*, and
2. steps during which we *minimize* $\varphi(X, Q) = \|X - ZQ\|_F$ *with respect to* Q *holding* X *fixed*.

This *second step in which* X *is held fixed* has been studied, but it is still a hard problem for which no closed–form solution is known. Consequently, we further simplify the problem.

Since Q is of the form $Q = R\Lambda$ where $R \in \mathbf{O}(K)$ and Λ is a diagonal invertible matrix, we minimize $\|X - ZR\Lambda\|_F$ in two stages.

To simplify the problem, we proceed by *alternating steps* during which

1. we *minimize* $\varphi(X, Q) = \|X - ZQ\|_F$ *with respect to* X *holding* Q *fixed*, and
2. steps during which we *minimize* $\varphi(X, Q) = \|X - ZQ\|_F$ *with respect to* Q *holding* X *fixed*.

This *second step in which* X *is held fixed* has been studied, but it is still a hard problem for which no closed–form solution is known. Consequently, we further simplify the problem.

Since Q is of the form $Q = R\Lambda$ where $R \in \mathbf{O}(K)$ and Λ is a diagonal invertible matrix, we minimize $\|X - ZR\Lambda\|_F$ in two stages.

1. We set $\Lambda = I$ and find $R \in \mathbf{O}(K)$ that minimizes $\|X - ZR\|_F$.
2. Given X, Z, and R, find a diagonal invertible matrix Λ that minimizes $\|X - ZR\Lambda\|_F$.

In *stage 1*, the matrix $Q = R$ is orthogonal, so $QQ^\top = I$, and since Z and X are given, the problem reduces to minimizing $-2\mathrm{tr}(Q^\top Z^\top X)$; that is, *maximizing* $\mathrm{tr}(Q^\top Z^\top X)$.

In *stage 1*, the matrix $Q = R$ is orthogonal, so $QQ^\top = I$, and since Z and X are given, the problem reduces to minimizing $-2\mathrm{tr}(Q^\top Z^\top X)$; that is, *maximizing* $\mathrm{tr}(Q^\top Z^\top X)$.

This is a standard result:

Proposition 6

For any two fixed $N \times K$ matrices X and Z, the minimum of the set

$$\{\|X - ZR\|_F \mid R \in \mathbf{O}(K)\}$$

is achieved by $R = UV^\top$, for any SVD decomposition $U\Sigma V^\top = Z^\top X$ of $Z^\top X$.

The following proposition takes care of *stage 2*.

Proposition 7

For any two fixed $N \times K$ matrices X and Z, where Z has no zero column, there is a unique diagonal matrix $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_K)$ minimizing $\|X - Z\Lambda\|_F$ given by

$$\lambda_j = \frac{(Z^\top X)_{jj}}{\|Z^j\|_2^2} \quad j = 1, \ldots, K.$$

The following proposition takes care of *stage 2*.

Proposition 7

For any two fixed $N \times K$ matrices X and Z, where Z has no zero column, there is a unique diagonal matrix $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_K)$ minimizing $\|X - Z\Lambda\|_F$ given by

$$\lambda_j = \frac{(Z^\top X)_{jj}}{\|Z^j\|_2^2} \quad j = 1, \ldots, K.$$

It should be noted that Proposition 7 does not guarantee that Λ is invertible.

We now deal with *step 1, where $Q = R\Lambda$ is held fixed*.

We now deal with *step 1, where $Q = R\Lambda$ is held fixed*.

For fixed Z and Q, we would like to find some $X \in \mathcal{K}$ with $\|X\|_F = \|Z\|_F$ so that $\|X - ZQ\|_F$ is minimal.

We now deal with *step 1, where $Q = R\Lambda$ is held fixed*.

For fixed Z and Q, we would like to find some $X \in \mathcal{K}$ with $\|X\|_F = \|Z\|_F$ so that $\|X - ZQ\|_F$ is minimal.

Without loss of generality, we may assume that the entries $a_1, \ldots, a_K$ occurring in the matrix X are positive and all equal to some common value $a \neq 0$.

We now deal with *step 1, where $Q = R\Lambda$ is held fixed*.

For fixed Z and Q, we would like to find some $X \in \mathcal{K}$ with $\|X\|_F = \|Z\|_F$ so that $\|X - ZQ\|_F$ is minimal.

Without loss of generality, we may assume that the entries $a_1, \ldots, a_K$ occurring in the matrix X are positive and all equal to some common value $a \neq 0$.

Recall that a matrix $X \in \mathcal{X}$ has the property that every row contains exactly one nonzero entry, and that every column is nonzero.

We now deal with *step 1, where $Q = R\Lambda$ is held fixed*.

For fixed Z and Q, we would like to find some $X \in \mathcal{K}$ with $\|X\|_F = \|Z\|_F$ so that $\|X - ZQ\|_F$ is minimal.

Without loss of generality, we may assume that the entries $a_1, \ldots, a_K$ occurring in the matrix X are positive and all equal to some common value $a \neq 0$.

Recall that a matrix $X \in \mathcal{X}$ has the property that every row contains exactly one nonzero entry, and that every column is nonzero.

The problem is to decide for each row, which column contains the nonzero entry.

After having found X, we rescale its columns so that $\|X\|_F = \|Z\|_F$.

For example, consider the following continuous solution and the discrete solution X:

$$\begin{pmatrix} 0.00 & -10.31 & 30.40 & 6.36 \\ 0.00 & -1.37 & 22.27 & -6.15 \\ -32.73 & -32.60 & -1.29 & 2.58 \\ 0.00 & -1.37 & 22.27 & -6.15 \\ 0.00 & 8.95 & 8.03 & -23.86 \\ -23.14 & -20.55 & -5.00 & -9.39 \\ 32.73 & -32.60 & -1.29 & 2.58 \\ 23.14 & -20.55 & -5.00 & -9.39 \\ -0.00 & -1.75 & -7.20 & -25.67 \end{pmatrix} \quad X = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}.$$

After having found X, we rescale its columns so that $\|X\|_F = \|Z\|_F$.

For example, consider the following continuous solution and the discrete solution X:

$$\begin{pmatrix} 0.00 & -10.31 & 30.40 & 6.36 \\ 0.00 & -1.37 & 22.27 & -6.15 \\ -32.73 & -32.60 & -1.29 & 2.58 \\ 0.00 & -1.37 & 22.27 & -6.15 \\ 0.00 & 8.95 & 8.03 & -23.86 \\ -23.14 & -20.55 & -5.00 & -9.39 \\ 32.73 & -32.60 & -1.29 & 2.58 \\ 23.14 & -20.55 & -5.00 & -9.39 \\ -0.00 & -1.75 & -7.20 & -25.67 \end{pmatrix} \quad X = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}.$$

We keep the leftmost largest entry on every row and set the others entries to 0.

Unfortunately, the matrix X may not be a correct solution, because the above prescription does not guarantee that every column of X is nonzero.

Unfortunately, the matrix X may not be a correct solution, because the above prescription does not guarantee that every column of X is nonzero.

When this happens, we *reassign certain nonzero entries in columns having "many" nonzero entries to zero columns, so that we get a matrix in* $\mathcal{K}$.

If we apply the method to the graph associated with the the matrix W_1 shown in Figure 15 for $K = 4$ clusters, the algorithm converges in 3 steps and we find the clusters shown in Figure 16.

If we apply the method to the graph associated with the the matrix W_1 shown in Figure 15 for $K = 4$ clusters, the algorithm converges in 3 steps and we find the clusters shown in Figure 16.

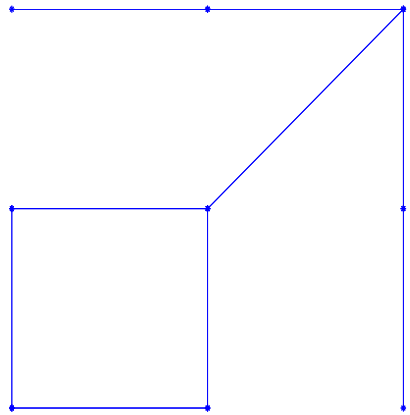


Figure 15: Underlying graph of the matrix W_1.

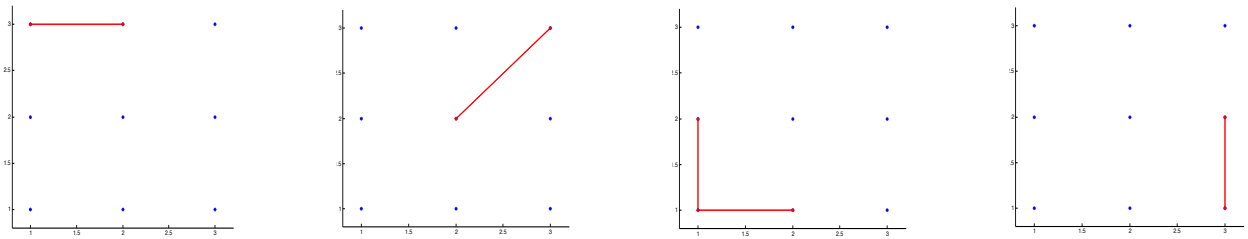


Figure 16: Four blocks of a normalized cut for the graph associated with W_1 .

The solution Z of the relaxed problem is

$$Z = \begin{pmatrix} -21.3146 & -0.0000 & 19.4684 & -15.4303 \\ -4.1289 & 0.0000 & 16.7503 & -15.4303 \\ -21.3146 & 32.7327 & -19.4684 & -15.4303 \\ -4.1289 & -0.0000 & 16.7503 & -15.4303 \\ 19.7150 & 0.0000 & 9.3547 & -15.4303 \\ -4.1289 & 23.1455 & -16.7503 & -15.4303 \\ -21.3146 & -32.7327 & -19.4684 & -15.4303 \\ -4.1289 & -23.1455 & -16.7503 & -15.4303 \\ 19.7150 & -0.0000 & -9.3547 & -15.4303 \end{pmatrix}.$$

We find the following sequence for $Q, Z * Q, X$:

$$Q = \begin{pmatrix} 0 & 0.6109 & -0.3446 & -0.7128 \\ -1.0000 & 0.0000 & 0.0000 & -0.0000 \\ 0.0000 & 0.5724 & 0.8142 & 0.0969 \\ -0.0000 & 0.5470 & -0.4672 & 0.6947 \end{pmatrix},$$

which is the initial Q obtained by method 1;

$$Z * Q = \begin{pmatrix} 0.0000 & -10.3162 & 30.4065 & 6.3600 \\ 0.0000 & -1.3742 & 22.2703 & -6.1531 \\ -32.7327 & -32.6044 & -1.2967 & 2.5884 \\ 0.0000 & -1.3742 & 22.2703 & -6.1531 \\ 0.0000 & 8.9576 & 8.0309 & -23.8653 \\ -23.1455 & -20.5505 & -5.0065 & -9.3982 \\ 32.7327 & -32.6044 & -1.2967 & 2.5884 \\ 23.1455 & -20.5505 & -5.0065 & -9.3982 \\ -0.0000 & -1.7520 & -7.2027 & -25.6776 \end{pmatrix}$$

$$X = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix};$$

$$Q = \begin{pmatrix} -0.0803 & 0.8633 & -0.4518 & -0.2102 \\ -0.6485 & 0.1929 & 0.1482 & 0.7213 \\ -0.5424 & 0.0876 & 0.5546 & -0.6250 \\ -0.5281 & -0.4581 & -0.6829 & -0.2119 \end{pmatrix}$$

$$Z * Q = \begin{pmatrix} -0.6994 & -9.6267 & 30.9638 & -4.4169 \\ -0.6051 & 4.9713 & 21.6922 & -6.3311 \\ -0.8081 & -6.7218 & 14.2223 & 43.5287 \\ -0.6051 & 4.9713 & 21.6922 & -6.3311 \\ 1.4913 & 24.9075 & 6.8186 & -6.7218 \\ 2.5548 & 6.5028 & 6.5445 & 31.3015 \\ 41.6456 & -19.3507 & 4.5190 & -3.6915 \\ 32.5742 & -2.4272 & -0.3168 & -2.0882 \\ 11.6387 & 23.2692 & -3.5570 & 4.9716 \end{pmatrix}$$

$$X = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix};$$

$$Q = \begin{pmatrix} -0.3201 & 0.7992 & -0.3953 & -0.3201 \\ -0.7071 & -0.0000 & 0.0000 & 0.7071 \\ -0.4914 & -0.0385 & 0.7181 & -0.4914 \\ -0.3951 & -0.5998 & -0.5728 & -0.3951 \end{pmatrix}$$

$$Z * Q = \begin{pmatrix} 3.3532 & -8.5296 & 31.2440 & 3.3532 \\ -0.8129 & 5.3103 & 22.4987 & -0.8129 \\ -0.6599 & -7.0310 & 3.2844 & 45.6311 \\ -0.8129 & 5.3103 & 22.4987 & -0.8129 \\ -4.8123 & 24.6517 & 7.7629 & -4.8123 \\ -0.7181 & 6.5997 & -1.5571 & 32.0146 \\ 45.6311 & -7.0310 & 3.2844 & -0.6599 \\ 32.0146 & 6.5997 & -1.5571 & -0.7181 \\ 4.3810 & 25.3718 & -5.6719 & 4.3810 \end{pmatrix}$$

$$X = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}.$$

During the next round, the exact same matrices are obtained and the algorithm stops.

Any matrix obtained by flipping the signs of some of the columns of a solution ZR of problem $(*_2)$ is still a solution.

Any matrix obtained by flipping the signs of some of the columns of a solution ZR of problem $(*_2)$ is still a solution.

Moreover, all entries in X are nonnegative. It follows that a "good" solution ZQ_p (that is, close to a discrete solution) should have the property that the average of each of its column is nonnegative.

Any matrix obtained by flipping the signs of some of the columns of a solution ZR of problem $(*_2)$ is still a solution.

Moreover, all entries in X are nonnegative. It follows that a "good" solution ZQ_p (that is, close to a discrete solution) should have the property that the average of each of its column is nonnegative.

We found that the following heuristic is quite helpful in finding a better discrete solution X:

Any matrix obtained by flipping the signs of some of the columns of a solution ZR of problem $(*_2)$ is still a solution.

Moreover, all entries in X are nonnegative. It follows that a "good" solution ZQ_p (that is, close to a discrete solution) should have the property that the average of each of its column is nonnegative.

We found that the following heuristic is quite helpful in finding a better discrete solution X:

Given a solution ZR of problem $(*_2)$, we compute ZQ_p, defined such that if the average of column $(ZR)^j$ is negative, then $(ZQ_p)^j = -(ZR)^j$, else $(ZQ_p)^j = (ZR)^j$.

Figure 17 shows a graph (on the left) and the graph drawings X and $Z * R$ obtained by applying our method for three clusters.

Figure 17 shows a graph (on the left) and the graph drawings X and $Z * R$ obtained by applying our method for three clusters.

The rows of X are represented by the red points along the axes, and the rows of $Z * R$ by the green points (on the right).

Figure 17 shows a graph (on the left) and the graph drawings X and $Z * R$ obtained by applying our method for three clusters.

The rows of X are represented by the red points along the axes, and the rows of $Z * R$ by the green points (on the right).

The original vertices corresponding to the rows of Z are represented in blue.

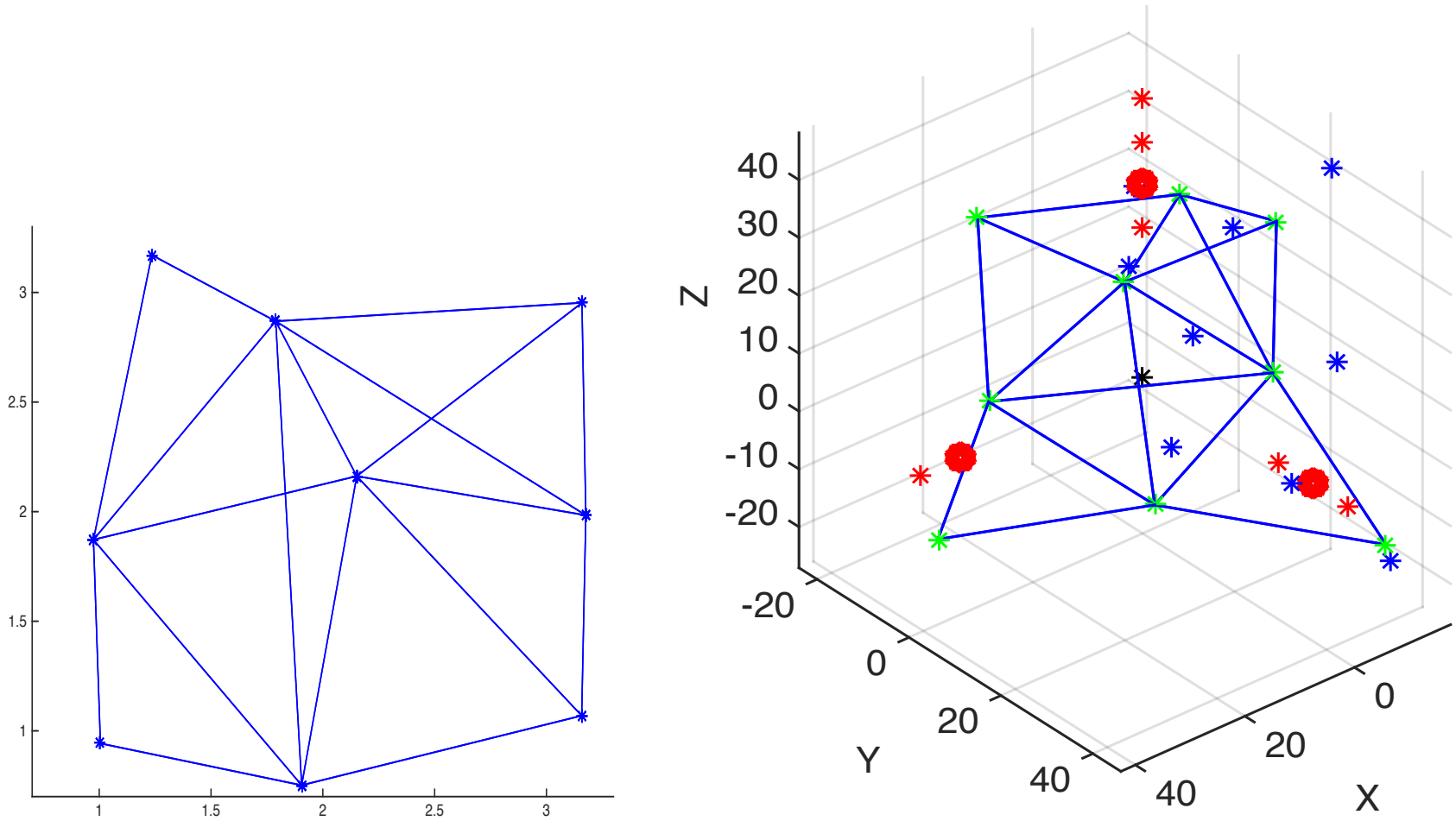


Figure 17: A graph and its drawing to find 3 clusters.

We can see how the two red points correspond to an edge, the three red points correspond to a triangle, and the four red points to a quadrangle.

We can see how the two red points correspond to an edge, the three red points correspond to a triangle, and the four red points to a quadrangle.

These constitute the clusters.

It remains to initialize Q^* to start the process, and then steps (1) (holding Q fixed) and (2) (holding X fixed) are iterated, starting with step (1).

It remains to initialize Q^* to start the process, and then steps (1) (holding Q fixed) and (2) (holding X fixed) are iterated, starting with step (1).

Actually, what we really need is a "good" initial X^*, but to find it, we need an initial R^*.

It remains to initialize Q^* to start the process, and then steps (1) (holding Q fixed) and (2) (holding X fixed) are iterated, starting with step (1).

Actually, what we really need is a "good" initial X^*, but to find it, we need an initial R^*.

Method 1. One method is to use an orthogonal matrix denoted R_1, such that distinct columns of ZR_1 are *simultaneously orthogonal and D-orthogonal*.

It remains to initialize Q^* to start the process, and then steps (1) (holding Q fixed) and (2) (holding X fixed) are iterated, starting with step (1).

Actually, what we really need is a "good" initial X^*, but to find it, we need an initial R^*.

Method 1. One method is to use an orthogonal matrix denoted R_1, such that distinct columns of ZR_1 are *simultaneously orthogonal and D-orthogonal*.

The matrix R_1 can be found by diagonalizing $Z^\top Z$ as $Z^\top Z = R_1 \Sigma R_1^\top$, as we explained earlier. We write $Z_2 = ZR_1$.

Method 2. The method advocated by Yu is to *pick K rows of Z that are as orthogonal to each other as possible* and to make a matrix R whose columns consist of these rows normalized to have unit length.

Method 2. The method advocated by Yu is to *pick K rows of Z that are as orthogonal to each other as possible* and to make a matrix R whose columns consist of these rows normalized to have unit length.

The intuition behind this method is that if a continuous solution Z can be sent close to a discrete solution X by a rigid motion, then many rows of Z viewed as vectors in $\mathbb{R}^K$ should be nearly orthogonal.

Method 2. The method advocated by Yu is to *pick K rows of Z that are as orthogonal to each other as possible* and to make a matrix R whose columns consist of these rows normalized to have unit length.

The intuition behind this method is that if a continuous solution Z can be sent close to a discrete solution X by a rigid motion, then many rows of Z viewed as vectors in $\mathbb{R}^K$ should be nearly orthogonal.

This way, ZR should contain at least K rows well aligned with the canonical basis vectors, and these rows are good candidates for some of the rows of the discrete solution X.

Method 2. The method advocated by Yu is to *pick K rows of Z that are as orthogonal to each other as possible* and to make a matrix R whose columns consist of these rows normalized to have unit length.

The intuition behind this method is that if a continuous solution Z can be sent close to a discrete solution X by a rigid motion, then many rows of Z viewed as vectors in $\mathbb{R}^K$ should be nearly orthogonal.

This way, ZR should contain at least K rows well aligned with the canonical basis vectors, and these rows are good candidates for some of the rows of the discrete solution X.

We also have implemented various methods for improving the initial X.

9. Semantic Word Clusters

Finding sets of similar words is important for various Natural Language Processing (NLP) tasks. The desired similarity can be part-of-speech, tense, etc. and in our case we want closest semantic equivalence. For tasks such as machine translation, considering antonyms as equivalent is extremely problematic.

This is akin to thesaurus sets, but given the fact that language changes rapidly be changes in meaning, new words or spellings (especially for twitter), as well as multitudes of languages, we aim to have data driven clusters.

Semantic Word Cluster Problem

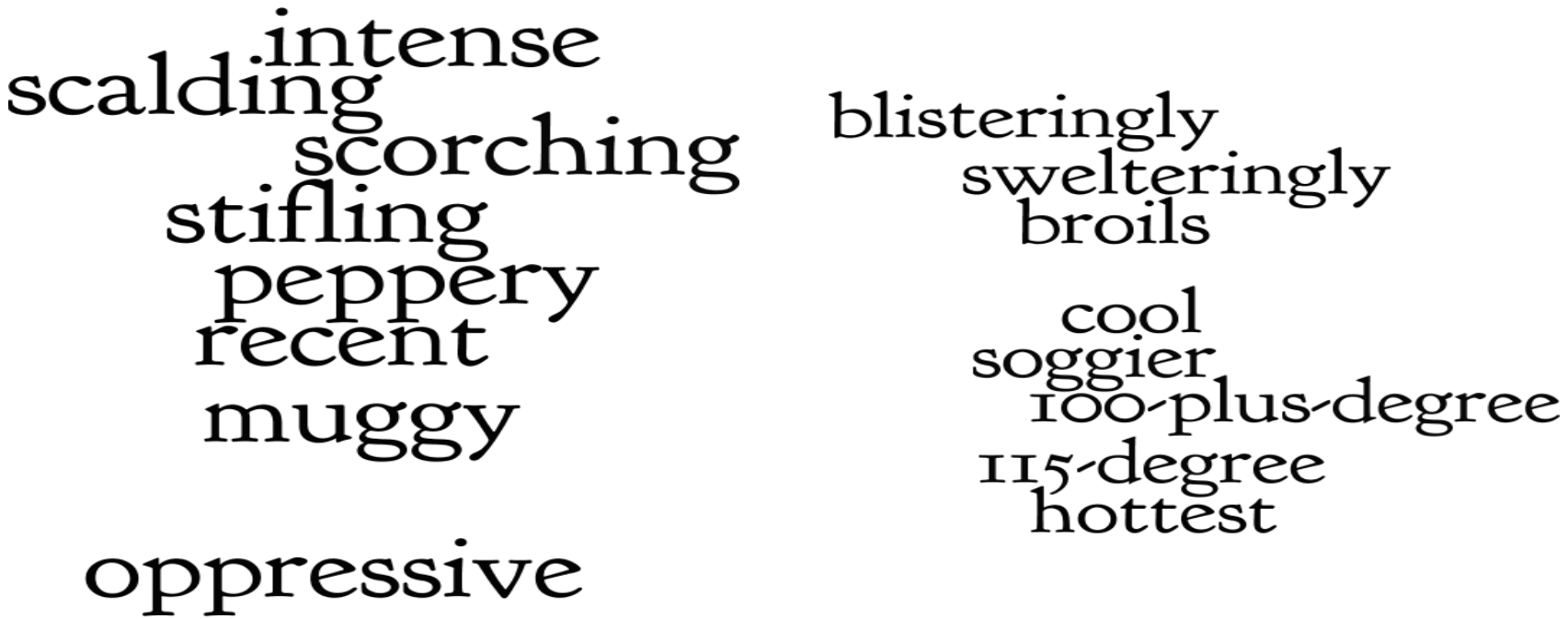


Figure 18: Thesaurus based (left) versus data-driven (right) clusters for "hot". It is important to note **cool** on the right.

Representing Words as Vectors

For word representations the initial obvious vector representation is a "one-hot" representations where word i is represented by a vector having all zeros for the size of our vocabulary aside from position i which is 1.

$$\text{hot} = \begin{pmatrix} 0 & 0 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 & 0 \end{pmatrix}^\top$$

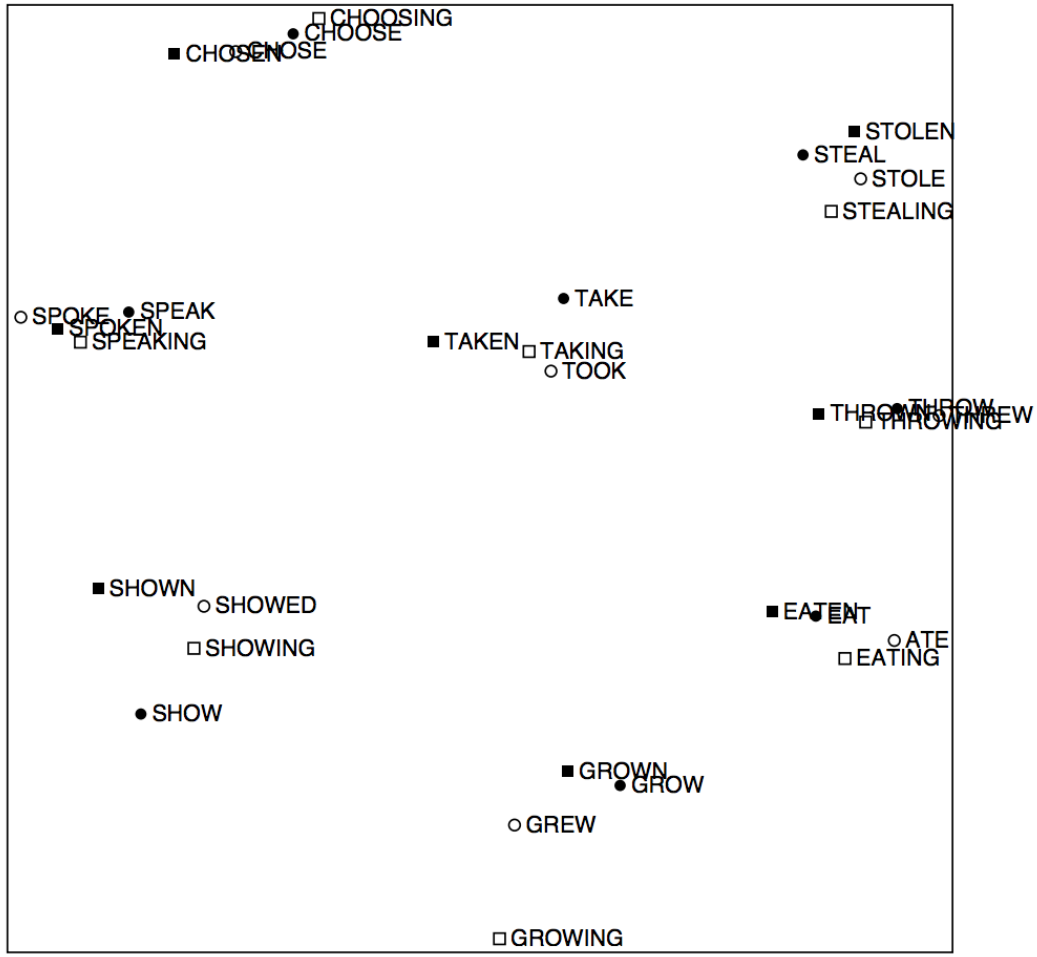
$$\text{scorching} = \begin{pmatrix} 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 1 & 0 \end{pmatrix}^\top$$

However, in this representation all words are orthogonal, which is highly undesirable.

Instead, we use so called word embeddings, which are dense vector representations in $\mathbb{R}^D$ where D is much smaller than the vocabulary.

Representing Words as Vectors

$$shown = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}$$



[Rohde et al. 2005. An Improved Model of Semantic Similarity Based on Lexical Co-Occurrence]

Representing Words as Vectors

The distributional hypothesis is that similar words are used in similar context (Harris 1954). Many vector popular representations (Eigenwords, GloVe, and word2vec) use adjacent words. However often antonyms such as "hot" and "cold" occur in similar contexts and thus have similar representations.

I meant... do I have time to fix you a **hot** lunch?
Tossing the **hot** pan holder on the counter, she untied the apron
He sipped the **hot** liquid and grimaced.
Her face felt **hot** again.
"Aren't you **cold**?" he asked
But, unfortunately, I struck my foot on a rock and fell forward into the **cold** water.

Table 1: "hot" and "cold" in sentence contexts [1] .

[1] from http://sentence.yourdictionary.com/

Embedding into Graph

We define the distance between two words $word_i$ and $word_j$ as $dist(word_i, word_j) = \|word_i - word_j\|$.
For the edge weight between two words

$$W_{ij} = \begin{cases} 0 & \text{if } e^{-\frac{dist(word_i, word_j)^2}{\sigma}} < thresh \\ e^{-\frac{dist(word_i, word_j)^2}{\sigma}} & \text{otherwise} \end{cases}.$$

We can represent the thesaurus as a matrix where

$$T_{ij} = \begin{cases} 1 & \text{if words } i \text{ and } j \text{ are synonyms} \\ -1 & \text{if words } i \text{ and } j \text{ are antonyms} \\ 0 & \text{otherwise} \end{cases}.$$

We can write the weight matrix of the signed graph as $\hat{W}_{ij} = T_{ij} W_{ij}$ or in matrix form $\hat{W} = T \odot W$ where $\odot$ denotes element-wise multiplication.

Embedding into Graph