Spectral Clustering

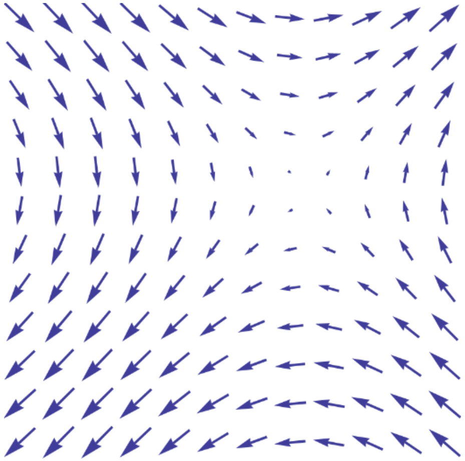
Part 1: The Graph Laplacian

Ng Yen Kaow

Laplacian of a function

- □ Given a multivariate function $f: \mathbb{R}^n \rightarrow \mathbb{R}$

- □ $\nabla f(\boldsymbol{x})$, the gradient at $f(\boldsymbol{x})$, is a vector pointing at the steepest ascent of $f(\boldsymbol{x})$



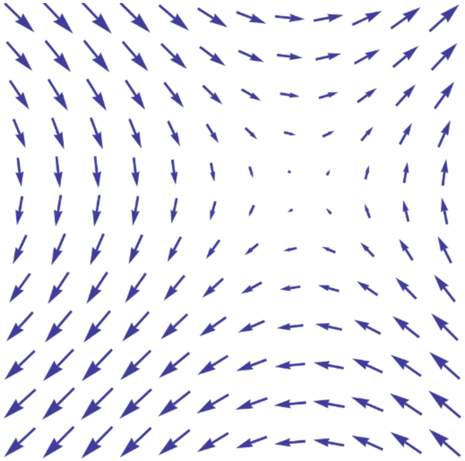
Vector field ∇f

- □ Δf, the Laplacian of f, is the divergence of ∇f, that is, $\Delta f(\boldsymbol{x}) = \nabla \cdot \nabla f(\boldsymbol{x})$
  - A scalar measurement of the smoothness in $\nabla f(\boldsymbol{x})$ about point $\boldsymbol{x}$

Laplacian of a function

- Given a multivariate function $f: \mathbb{R}^n \to \mathbb{R}$

- $\nabla f(\boldsymbol{x})$, the gradient at $f(\boldsymbol{x})$, is a vector pointing at the steepest ascent of $f(\boldsymbol{x})$



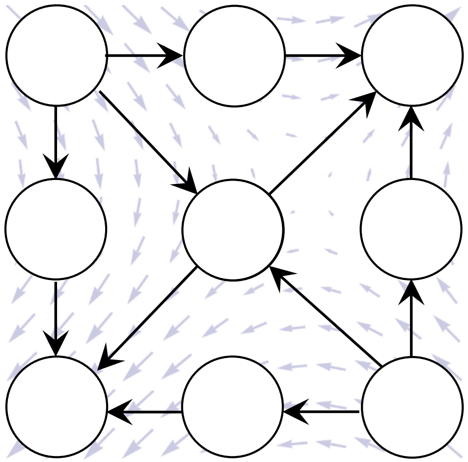
Vector field ∇f

Extend the concept
(from a **continuous space**)
to **graphs**

- e divergence of ∇f, that is, $\Delta f(\boldsymbol{x}) = \nabla \cdot \nabla f(\boldsymbol{x})$
  - A scalar measurement of the smoothness in $\nabla f(\boldsymbol{x})$ about point $\boldsymbol{x}$

Laplacian of a function

- Given a multivariate function $f: \mathbb{R}^n \rightarrow \mathbb{R}$
- $\nabla f(\boldsymbol{x})$, the gradient at $f(\boldsymbol{x})$, is a vector pointing at the steepest ascent of $f(\boldsymbol{x})$

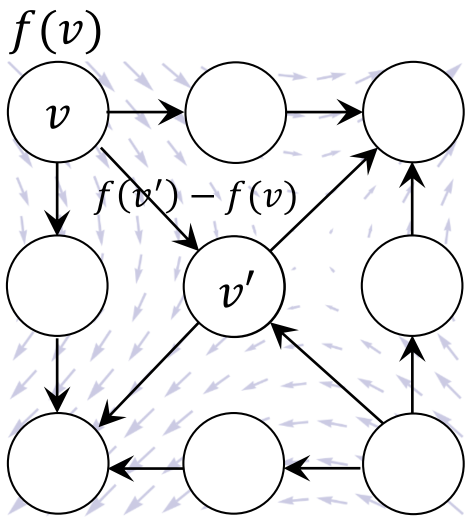


Extend the concept
(from a **continuous space**)
to **graphs**

- Consider each vertex as a point on a grid

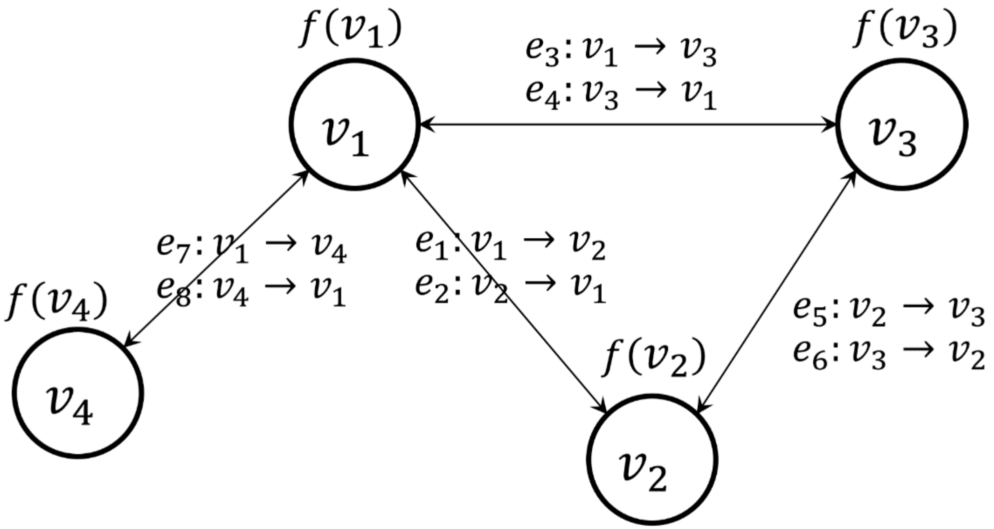
Laplacian of a function

- Given a multivariate function $f: \mathbb{R}^n \to \mathbb{R}$
- The domain of f are vertices
- f operates on each vertex v
  - Write $f(v)$ instead of $f(x)$
- The gradient from vertex v to v' is $f(v') - f(v)$ and is assigned to the edge $e: v \to v'$



- We want a matrix that encodes all the gradients ⇒ **The Graph Laplacian matrix**
  - We first construct an incidence matrix

Incidence matrix



Let vector $f = \begin{bmatrix} f(v_1) \\ f(v_2) \\ f(v_3) \\ f(v_4) \end{bmatrix}$

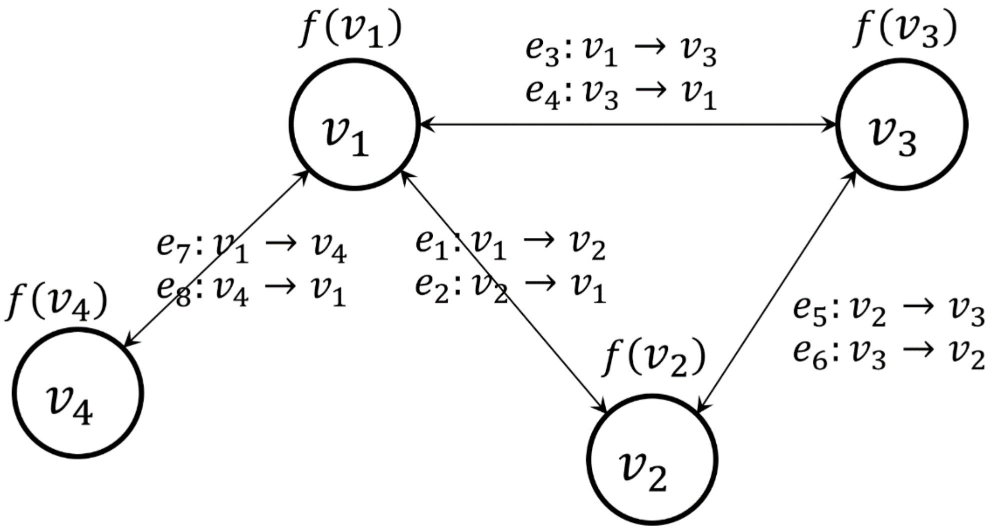
- □ Incidence matrix M

$$M = \begin{array}{c|cccccccc} & e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 \\ \hline v_1 & 1 & -1 & 1 & -1 & 0 & 0 & 1 & -1 \\ v_2 & -1 & 1 & 0 & 0 & 1 & -1 & 0 & 0 \\ v_3 & 0 & 0 & -1 & 1 & -1 & 1 & 0 & 0 \\ v_4 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{array}$$

- □ Every column of M represents an edge

$$(M^\top)_{\underbrace{1}_{\text{column 1 of } M}} f = \begin{bmatrix} 1 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} f(v_1) \\ f(v_2) \\ f(v_3) \\ f(v_4) \end{bmatrix} = f(v_1) - f(v_2) \overset{\text{def}}{=} w(e_1)$$

Incidence matrix



Let vector $f = \begin{bmatrix} f(v_1) \\ f(v_2) \\ f(v_3) \\ f(v_4) \end{bmatrix}$

- Incidence matrix M

$$M = \begin{array}{c|cccccccc} & e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 \\ \hline v_1 & 1 & -1 & 1 & -1 & 0 & 0 & 1 & -1 \\ v_2 & -1 & 1 & 0 & 0 & 1 & -1 & 0 & 0 \\ v_3 & 0 & 0 & -1 & 1 & -1 & 1 & 0 & 0 \\ v_4 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{array}$$

- Every column of M represents an edge

$$M^\top f = \begin{bmatrix} w(e_1) \\ w(e_2) \\ \vdots \\ w(e_8) \end{bmatrix}$$

- $M^\top f$ encodes all the edges

The graph Laplacian L

- The graph Laplacian L is obtained by

$$\Delta f = \nabla \cdot \nabla f = MM^\top f$$

  - $MM^\top f$ is a vector of length $|V|$ where each element is the divergence of a vertex

$$MM^\top \begin{bmatrix} f(v_1) \\ f(v_2) \\ \vdots \end{bmatrix} = \begin{bmatrix} \Delta f(v_1) \\ \Delta f(v_2) \\ \vdots \end{bmatrix}$$

    - e.g.

$$(MM^\top f)_1 = \begin{bmatrix} 1 & -1 & 1 & -1 & 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} w(e_1) \\ w(e_2) \\ w(e_3) \\ w(e_4) \\ w(e_5) \\ w(e_6) \\ w(e_7) \\ w(e_8) \end{bmatrix} = \underbrace{w(e_1) - w(e_2) + w(e_3) - w(e_4) + w(e_7) - w(e_8)}_{\text{divergence of vertex } v_1}$$

  - $MM^\top$ is a $|V| \times |V|$ matrix

The graph Laplacian L

```
import numpy as np

M = np.array([[ 1,-1, 1,-1, 0, 0, 1,-1],
              [-1, 1, 0, 0, 1,-1, 0, 0],
              [ 0, 0,-1, 1,-1, 1, 0, 0],
              [ 0, 0, 0, 0, 0, 0,-1, 1]])

# Compute MM^T
M @ M.transpose()
```

- Output

```
array([[ 6, -2, -2, -2],
       [-2,  4, -2,  0],
       [-2, -2,  4,  0],
       [-2,  0,  0,  2]])
```

There will be a lot of hands-on so please try this on your own computer **now**

- $MM^{\top}$ is a $|V| \times |V|$ matrix

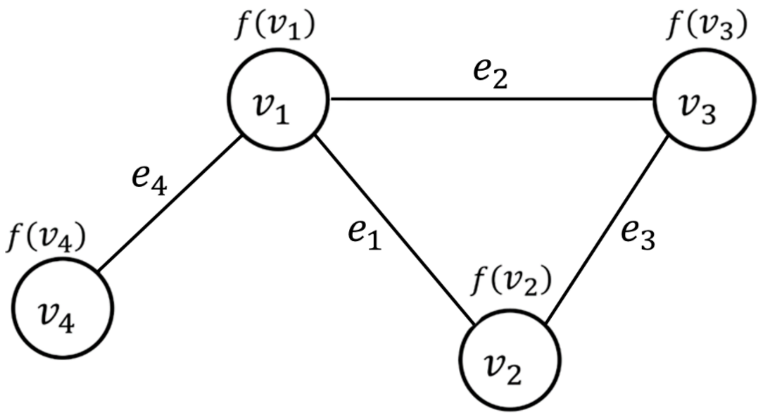
Properties of L

- □ The graph Laplacian L is obtained as $L = MM^\top$

1. For an undirected graph, L **can be computed as** $L = D - A$ from the degree matrix D and the adjacency matrix A
   - That is, $MM^\top = D - A$
2. For an undirected graph, L is **symmetric** (and in fact, positive semidefinite)
   - This allows us to obtain a **real orthogonal eigenbasis with real eigenvalues**
   - *The eigenbasis has topological significance but we will save this discussion for Part 3*
3. L has a **mathematical interpretation** which will allow us to make use of the eigenbasis

Property 1: $L = D - A$

- The undirected incidence matrix M of earlier graph



$$M = \begin{array}{c|cccc} & e_1 & e_2 & e_3 & e_4 \\ v_1 & 1 & 1 & 0 & 1 \\ v_2 & -1 & 0 & 1 & 0 \\ v_3 & 0 & -1 & -1 & 0 \\ v_4 & 0 & 0 & 0 & -1 \end{array}$$

  - Observe that for the undirected case, we let the second non-zero value that appear in every column be -1

- Adjacency matrix of the graph, $A = \begin{array}{c|cccc} & v_1 & v_2 & v_3 & v_4 \\ v_1 & 0 & 1 & 1 & 1 \\ v_2 & 1 & 0 & 1 & 0 \\ v_3 & 1 & 1 & 0 & 0 \\ v_4 & 1 & 0 & 0 & 0 \end{array}$

  - A is easier to construct than M (no need to name the edges and no messy -1 values)

Property 1: $L = D - A$

- Run the following to verify that $L = MM^\top = D - A$

$MM^\top$

```
import numpy as np

M = np.array([[ 1, 1, 0, 1],
              [-1, 0, 1, 0],
              [ 0,-1,-1, 0],
              [ 0, 0, 0,-1]])

# Compute MM^T
L = M @ M.transpose()

L
```

$D - A$

```
import numpy as np

A = np.array([[0, 1, 1, 1],
              [1, 0, 1, 0],
              [1, 1, 0, 0],
              [1, 0, 0, 0]])

D = np.diag(A.sum(axis=0))

L=D-A
L
```

Property 2: Eigenbasis

- A eigenvector for a square matrix L is a vector u where

$$Lu = \lambda u$$

  - u is invariant under transformation L
  - The scaling factor λ is a eigenvalue
    - Each L has a unique set of eigenvalues
- For real symmetric L
  - The eigenvalues are real
  - A set of **real** and **orthogonal** eigenvectors that correspond to distinct eigenvalues can be computed

Property 2: Eigenbasis

- Let $\lambda_1, \ldots, \lambda_n$ where $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$ be the eigenvalues of L and define the Rayleigh quotient $\frac{x^\top L x}{x^\top x}$ for arbitrary vector x
- **Min-max Theorem**
  - Maximum of the Rayleigh quotient,

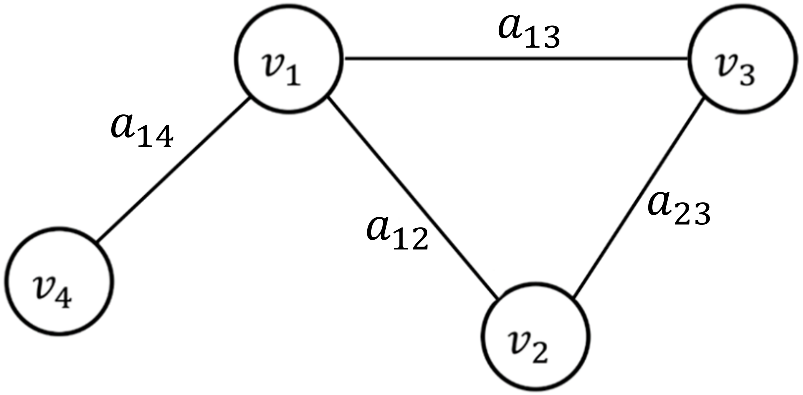
$$\max_{\|x\|=1} \frac{x^\top L x}{x^\top x} = \lambda_1$$

  - Minimum of the Rayleigh quotient,

$$\min_{\|x\|=1} \frac{x^\top L x}{x^\top x} = \lambda_n$$

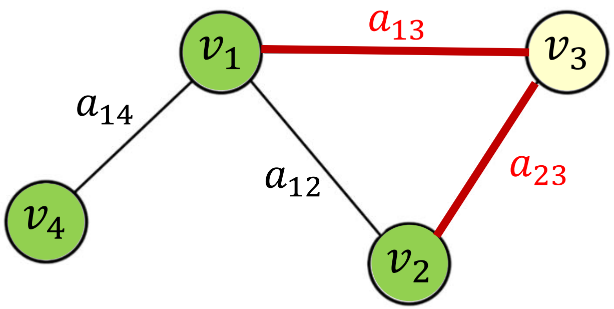
Property 3: Mathematical property

- A precise mathematical property of L relates it to "sparsest cut" problems
- Let the adjacency matrix $A = (a_{ij})$



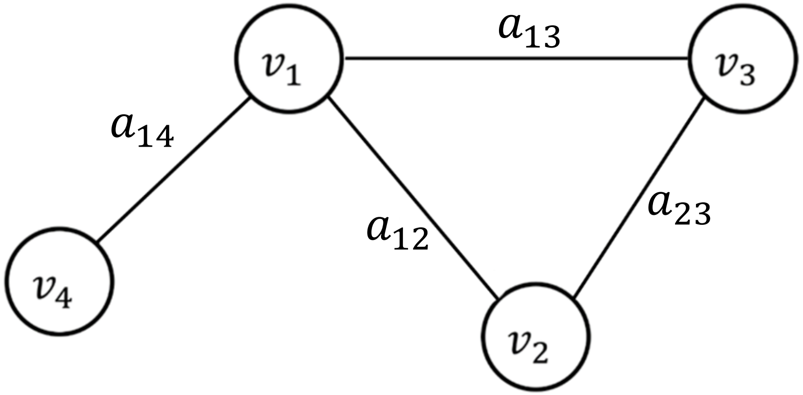
$$A = \begin{bmatrix} 0 & a_{12} & a_{13} & a_{14} \\ a_{12} & 0 & a_{23} & 0 \\ a_{13} & a_{23} & 0 & 0 \\ a_{14} & 0 & 0 & 0 \end{bmatrix}$$

- Consider partitioning graph into 2 parts, S and $\bar{S}$
  - If $S = \{v_1, v_2, v_4\}$, then we need to remove the two edges which sum to $a_{13} + a_{23}$



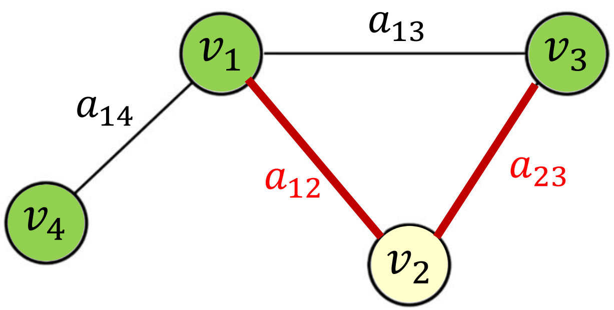
Property 3: Mathematical property

- A precise mathematical property of L relates it to “sparsest cut” problems
- Let the adjacency matrix $A = (a_{ij})$



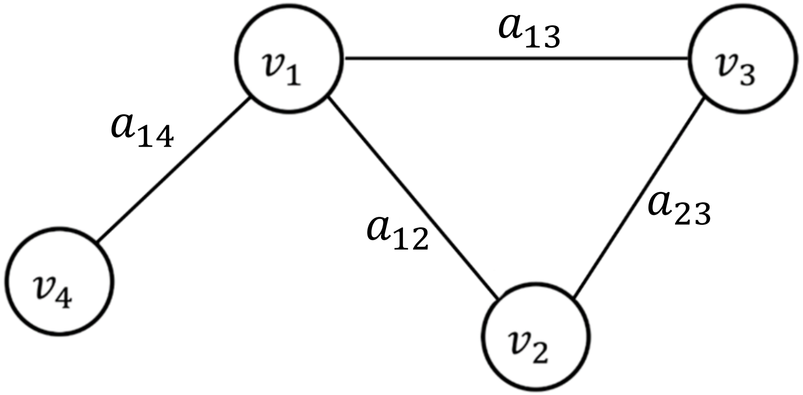
$$A = \begin{bmatrix} 0 & a_{12} & a_{13} & a_{14} \\ a_{12} & 0 & a_{23} & 0 \\ a_{13} & a_{23} & 0 & 0 \\ a_{14} & 0 & 0 & 0 \end{bmatrix}$$

- Consider partitioning graph into 2 parts, S and $\bar{S}$
  - If $S = \{v_1, v_3, v_4\}$, then the sum of the edges to be removed is $a_{12} + a_{23}$



Property 3: Mathematical property

- A precise mathematical property of L relates it to "sparsest cut" problems
- Let the adjacency matrix $A = (a_{ij})$



$$A = \begin{bmatrix} 0 & a_{12} & a_{13} & a_{14} \\ a_{12} & 0 & a_{23} & 0 \\ a_{13} & a_{23} & 0 & 0 \\ a_{14} & 0 & 0 & 0 \end{bmatrix}$$

- Consider partitioning graph into 2 parts, S and $\bar{S}$
  - The Laplacian L can be related to the sum of the edges to remove

Property 3: Mathematical property

- A precise mathematical property of L relates it to "sparsest cut" problems
- We first note that

$$x^\top Lx = \frac{1}{2}\sum_{i,j=1}^{m} a_{ij}\left(x_i - x_j\right)^2$$

$$x^\top Lx = x^\top Dx - x^\top Ax = \sum_{i=1}^{m} d_i x_i^2 - \sum_{i,j=1}^{m} a_{ij} x_i x_j$$

$$= \frac{1}{2}\left(\sum_{i=1}^{m} d_i x_i^2 - 2\sum_{i,j=1}^{m} a_{ij} x_i x_j + \sum_{i=1}^{m} d_i x_i^2\right)$$

$$= \frac{1}{2}\sum_{i,j=1}^{m} a_{ij}\left(x_i - x_j\right)^2$$

Hall. An *r*-dimensional quadratic placement algorithm, 1970

Property 3: Mathematical property

- □ A precise mathematical property of L relates it to “sparsest cut” problems
- □ Then

$$x^{\top}Lx = \frac{1}{2}\sum_{i,j=1}^{m} a_{ij}\left(x_i - x_j\right)^2$$

$$= \frac{1}{2}\sum_{i,j=1,i\neq j}^{m} a_{ij}\left(x_i - x_j\right)^2$$

$$= \sum_{i,j=1,i<j}^{m} a_{ij}\left(x_i - x_j\right)^2$$

Property 3: Mathematical property

- □ A precise mathematical property of L relates it to "sparsest cut" problems
- □ Furthermore

$$x^\top Lx = \sum_{i,j=1,i<j}^{m} a_{ij}\left(x_i - x_j\right)^2$$

If $x_i = x_j$

$$\left(x_i - x_j\right)^2 = 0$$

If $x_i \neq x_j$

$$\left(x_i - x_j\right)^2 = 4$$

i.e. $(-1-1)^2$ or $(1-(-1))^2 = 4$

- □ Suppose x is a vector of only the values +1 and -1, indicating the membership of the vertices in a set S

$$x_i = \begin{cases} 1 & \text{if } v_i \in S \\ -1 & \text{if } v_i \notin S \end{cases}$$

This way x can indicate the result of a 2-partition, S and $\bar{S}$

Property 3: Mathematical property

- A precise mathematical property of L relates it to “sparsest cut” problems
- Finally

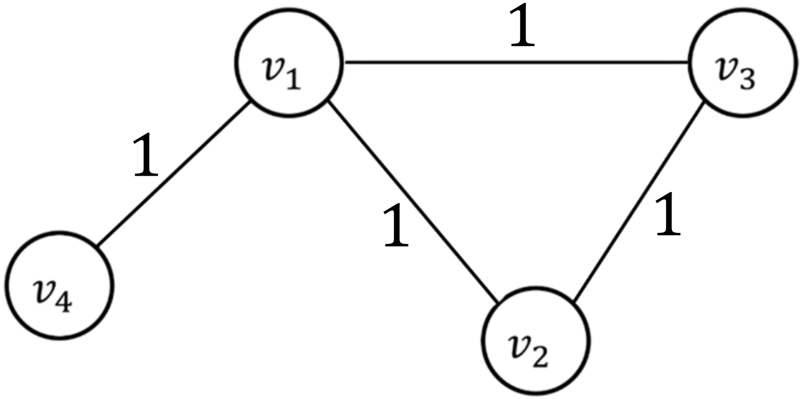
$$x^\top Lx = \sum_{i,j=1,i<j}^{m} a_{ij}\left(x_i - x_j\right)^2$$

$$= 4\sum_{1\le i<j\le m, x_i \neq x_j} a_{ij}$$

- Hence $x^\top Lx$ is 4 times the number of edges between the adjacent vertices from S and $\bar{S}$

Finding x that minimizes $x^\top Lx$

- Compute $x^\top Lx$
  - e.g.



$$A = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$L = \begin{bmatrix} 3 & -1 & -1 & -1 \\ -1 & 2 & -1 & 0 \\ -1 & -1 & 2 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix}$$

  - when $x = [1\ -1\ -1\ -1]$, $x^\top Lx = 12$

$$x^\top Lx = \begin{bmatrix} 1 & -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} 3 & -1 & -1 & -1 \\ -1 & 2 & -1 & 0 \\ -1 & -1 & 2 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ -1 \\ -1 \end{bmatrix} = 12$$

```
x = np.array([1, -1, -1, -1])
x @ L @ x
```

Finding x that minimizes $x^{\top}Lx$

- **Exercise:** Compute $x^{\top}Lx$ for all x

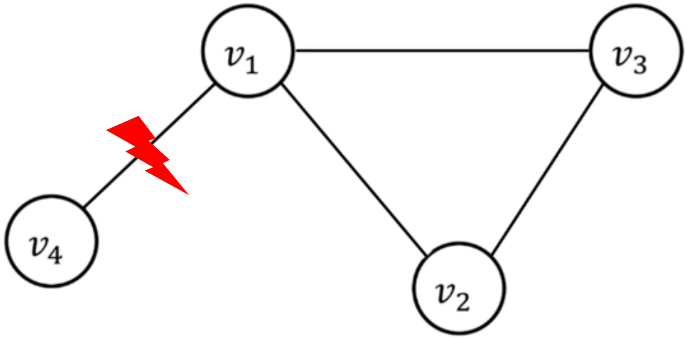
[redacted]

- Sample output

```
['v1' 'v2' 'v3' 'v4'] [] 0
['v1' 'v2' 'v3'] ['v4'] 4
['v1' 'v2' 'v4'] ['v3'] 8
['v1' 'v2'] ['v3' 'v4'] 12
['v1' 'v3' 'v4'] ['v2'] 8
['v1' 'v3'] ['v2' 'v4'] 12
['v1' 'v4'] ['v2' 'v3'] 8
['v1'] ['v2' 'v3' 'v4'] 12
```

Finding x that minimizes $x^\top Lx$

- $x^\top Lx = 0$ when $x = \mathbf{1} = [1\ 1\ 1\ 1]$ (or $x = -\mathbf{1} = [-1\ -1\ -1\ -1]$)
  - We do not want this solution
  - Use $x^\top Lx = 4$ instead



Group 1	Group 2	$x^\top Lx$
v_1	$v_2\ v_3\ v_4$	12
v_2	$v_1\ v_3\ v_4$	8
v_3	$v_1\ v_2\ v_4$	8
v_4	$v_1\ v_2\ v_3$	4
$v_1\ v_2$	$v_3\ v_4$	12
$v_1\ v_3$	$v_2\ v_4$	12
$v_1\ v_4$	$v_2\ v_3$	8
$v_1\ v_2\ v_3\ v_4$	$\emptyset$	0

- Next we compute the $\frac{x^\top Lx}{x^\top x}$ values from these

Finding x that minimizes $x^\top Lx/x^\top x$

- □ Complete list of $\frac{x^\top Lx}{x^\top x}$ values

  (x is of only +1 and -1 ⇒ $x^\top x = |x| = 4$)

Group 1	Group 2	$x^\top Lx$	$\frac{x^\top Lx}{x^\top x}$
v_1	$v_2\ v_3\ v_4$	12	3
v_2	$v_1\ v_3\ v_4$	8	2
v_3	$v_1\ v_2\ v_4$	8	2
v_4	$v_1\ v_2\ v_3$	**4**	**1**
$v_1\ v_2$	$v_3\ v_4$	12	3
$v_1\ v_3$	$v_2\ v_4$	12	3
$v_1\ v_4$	$v_2\ v_3$	8	2

- □ Optimal $\frac{x^\top Lx}{x^\top x} = 1$, when $x = [1 \quad 1 \quad 1 \quad -1]$ or $[-1 \quad -1 \quad -1 \quad 1]$
- □ **This optimal x can be approximately obtained…**

Finding x that minimizes $x^\top Lx/x^\top x$

- Let $\lambda_1,\ldots,\lambda_k$ where $\lambda_1 \geq \ldots \geq \lambda_k$ be the eigenvalues of L, and $\mu_1,\ldots,\mu_k$ the respective eigenvectors
  - By the min-max theorem of Rayleigh quotient,

$$\min_x \frac{x^\top Lx}{x^\top x} = \lambda_k$$

```
# Find the eigenvalues and eigenvectors
eigenvalues, eigenvectors = np.linalg.eig(L)

# Sort eigenvalues in decreasing order
idx = eigenvalues.argsort()[::-1]
eigenvalues = eigenvalues[idx]
eigenvectors = eigenvectors[:,idx]
eigenvalues
```

```
array([4.000000e+00, 3.000000e+00, 1.000000e+00, 1.110223e-16])
```

Eigendecomposition example

- Eigenvalues

λ_1	λ_2	λ_3	λ_4
4.0000	3.0000	1.0000	0.0000

Trivial solution (no partition)

- Eigenvectors

More precisely, -9.51E-17

μ_1	μ_2	μ_3	μ_4
0.8660	0.0000	0.0000	-0.5000
-0.2887	0.7071	-0.4082	-0.5000
-0.2887	-0.7071	-0.4082	-0.5000
-0.2887	0.0000	0.8165	-0.5000

- $\lambda_3 = 1 =$ optimal value for $\frac{1}{2}\sum_{1\le i,j\le m} a_{ij}\left(x_i - x_j\right)^2$
- If group by the ($\pm$) sign, μ_3 correctly places v_1, v_2, v_3 in one group ($-$) and v_4 in another ($+$)

Compromise in +1/-1 restriction

- By relaxing the restriction of +1 and -1 in x to allow any real number, an $x^\top Lx$ smaller than the optimal under the restriction is often achieved
  - The improvement can be guaranteed if x is orthogonal to $\mathbf{1}$ (or $-\mathbf{1}$) since by the min-max theorem, $\frac{{\mu_{k-1}}^\top L\mu_{k-1}}{{\mu_{k-1}}^\top \mu_{k-1}}$ is minimal among all $\frac{x^\top Lx}{x^\top x}$ that are orthogonal to μ_k
    - However, in the present case, $x = [1\ 1\ 1\ {-1}]$ and not orthogonal to $\mu_4 = [1\ 1\ 1\ 1]$
    - Still, $\frac{{\mu_3}^\top L\mu_3}{{\mu_3}^\top \mu_3} = \lambda_3 = 1 = \min_{\boldsymbol{x}\in\{1,-1\}^4} \frac{x^\top Lx}{x^\top x}$
    - Though no guarantee, improvements are usual

Historical use of μ_{k-1}

- Historically μ_{k-1} received more attention than the other eigenvectors
  - (Shi and Malik, 2000) started using multiple eigenvectors for clustering (see Part 3)
- μ_{k-1} is called the Fiedler vector
- λ_{k-1} is called the Fiedler value
  - The multiplicity of λ_{k-1} is always 1
  - Also called the algebraic connectivity
    - The further λ_{k-1} is from 0, the more highly connected is the graph (hard to separate)

Recap

- An intuition from the Laplacian function (in continuous space) gave us the **graph Laplacian matrix** (in graph space)

- Subsequently people found out that the graph Laplacian possesses several properties that lend it to solve graph cutting problems