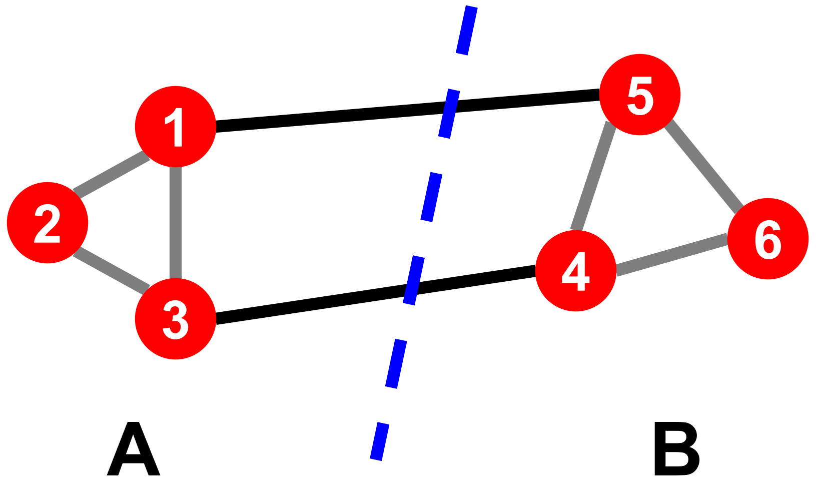
Spectral Clustering

Graph Partitioning

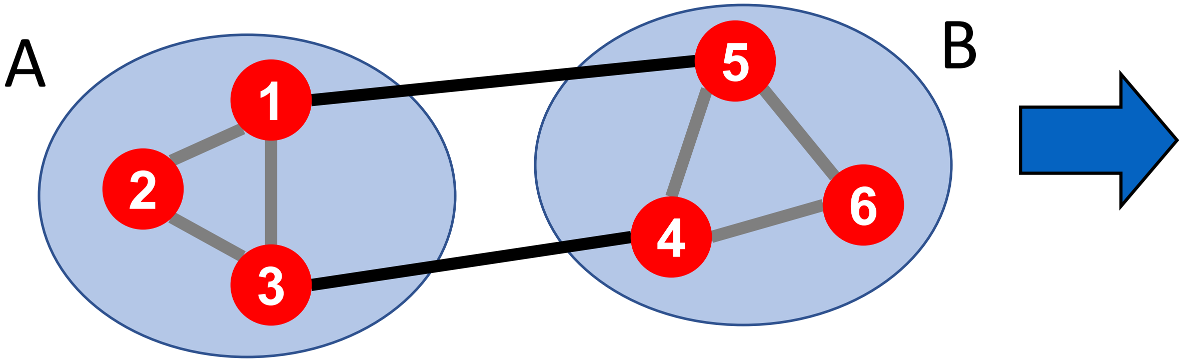
- **What makes a good partition?**
  - Maximize the number of within-group connections
  - Minimize the number of between-group connections



Graph Cuts

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

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



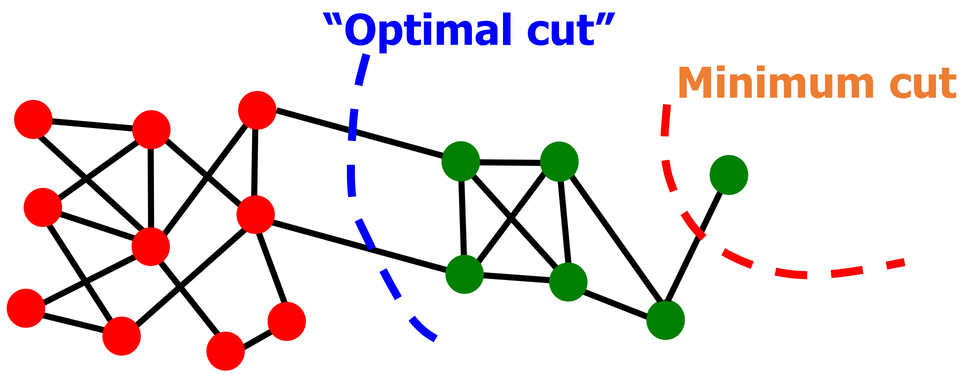
$cut(A,B) = 2$

Graph Cut Criterion

- **Criterion: Minimum-cut**
  - Minimize weight of connections between groups

$$\textbf{arg min}_{\mathbf{A,B}}\ \boldsymbol{cut(A,B)}$$

- **Degenerate case:**



- **Problem:**
  - Only considers external cluster connections
  - Does not consider internal cluster connectivity

Graph Cut Criteria

- **Criterion:** **Normalized-cut** Connectivity between groups relative to the density of each group

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

$\boldsymbol{vol(A)}$: total weight of the edges with at least one endpoint in $\boldsymbol{A}$

Produces more balanced partitions

- **How do we efficiently find a good partition?**
  - **Problem:** Computing optimal cut is NP-hard

Spectral Graph Partitioning

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

- **Spectral Graph Theory:**
  - Analyze the "spectrum" of matrix representing $\boldsymbol{G}$
  - **Spectrum:** Eigenvectors $\boldsymbol{x_i}$ of a graph Laplacian, ordered by the magnitude (strength) of their corresponding eigenvalues $\boldsymbol{\lambda_i}$:

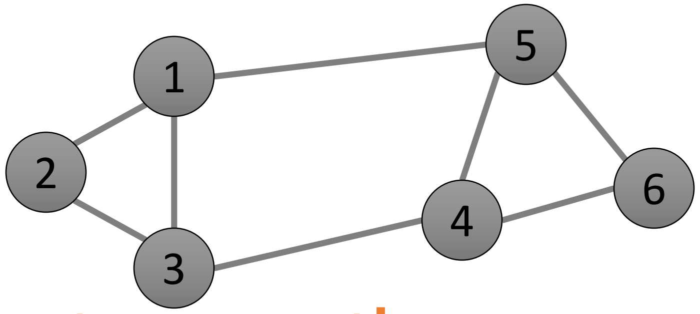
$$\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$$

$$\lambda_1 \le \lambda_2 \le \ldots \le \lambda_n$$

  - λ_2 and the corresponding eigenvector give us a partitioning.

Matrix Representations

- **Adjacency matrix (A):**
  - $n \times n$ matrix
  - $A=[a_{ij}]$, $a_{ij}=1$ if edge between node i and j

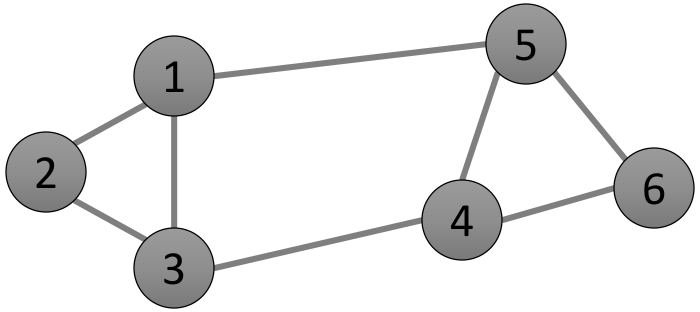


	1	2	3	4	5	6
1	0	1	1	0	1	0
2	1	0	1	0	0	0
3	1	1	0	1	0	0
4	0	0	1	0	1	1
5	1	0	0	1	0	1
6	0	0	0	1	1	0

- **Important properties:**
  - Symmetric matrix
  - Eigenvectors are real and orthogonal

Matrix Representations

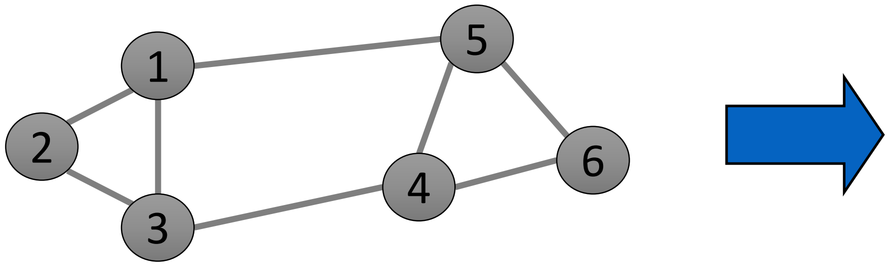
- **Degree matrix (D):**
  - $n \times n$ diagonal matrix
  - $D=[\mathbf{d}_{ii}]$, d_{ii} = degree of node i



	1	2	3	4	5	6
1	3	0	0	0	0	0
2	0	2	0	0	0	0
3	0	0	3	0	0	0
4	0	0	0	3	0	0
5	0	0	0	0	3	0
6	0	0	0	0	0	2

Matrix Representations

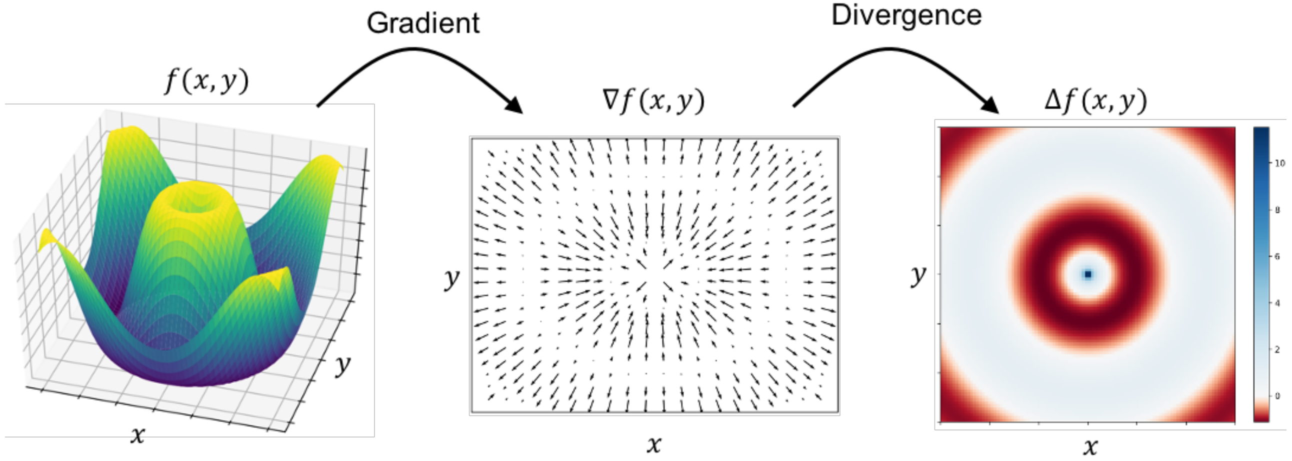
- **Laplacian matrix (L):**
  - $\boldsymbol{n \times n}$ symmetric matrix



$$\boldsymbol{L = D - A}$$

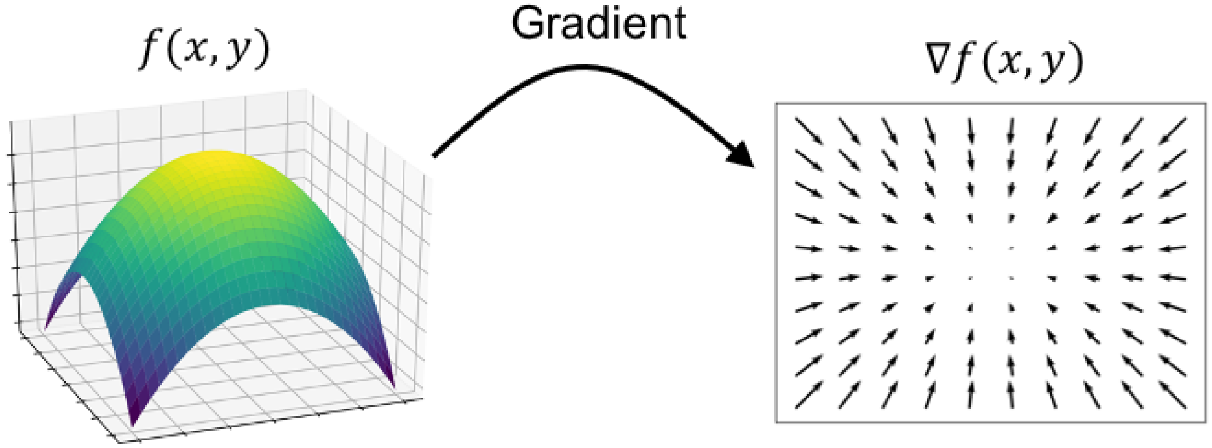
	1	2	3	4	5	6
1	3	-1	-1	0	-1	0
2	-1	2	-1	0	0	0
3	-1	-1	3	-1	0	0
4	0	0	-1	3	-1	-1
5	-1	0	0	-1	3	-1
6	0	0	0	-1	-1	2

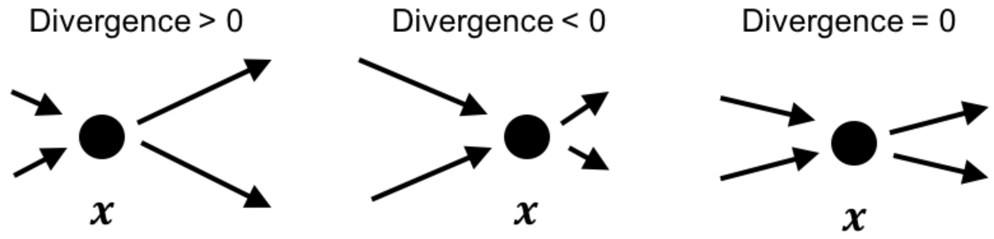
Graph Laplacian



- Why we call it graph Laplacian?
  - Analogue to the Laplacian operator on multivariate continuous functions
  - Given a multivariate function $f: R^d \to R$, the Laplacian of f is the **divergence** of f's **gradient**

$$\Delta f(\boldsymbol{x}) = \nabla \cdot \nabla f(\boldsymbol{x})$$





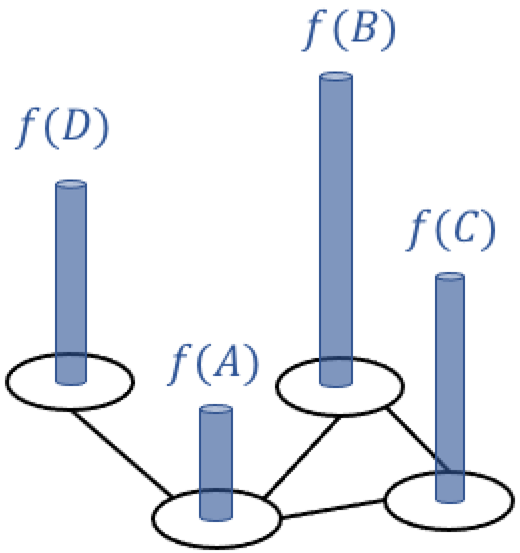
Graph Laplacian

- Why we call it graph Laplacian?
  - Constructing Laplacian for graphs

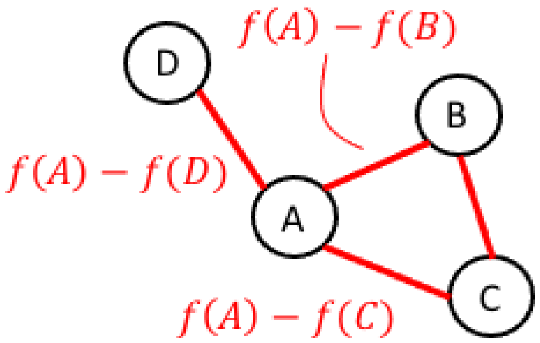
	1	2	3	4	5	6
1	3	-1	-1	0	-1	0
2	-1	2	-1	0	0	0
3	-1	-1	3	-1	0	0
4	0	0	-1	3	-1	-1
5	-1	0	0	-1	3	-1
6	0	0	0	-1	-1	2

$$\boldsymbol{L = D - A}$$

Function: $f: V \to R$



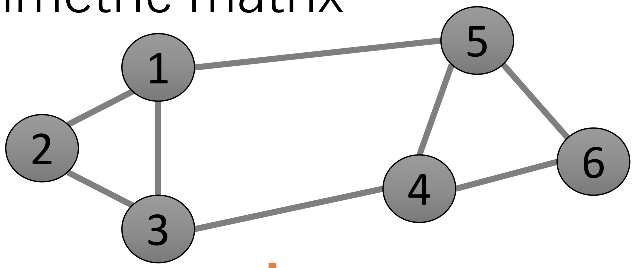
Gradient: $g(e_k) = f(v_i) - f(v_j)$



divergence(grad(A))=$\sum g(e_k)$
$= 3f(A) - f(B) - f(C) - f(D)$

Graph Laplacian

- **Laplacian matrix (L):**
  - $\boldsymbol{n \times n}$ symmetric matrix



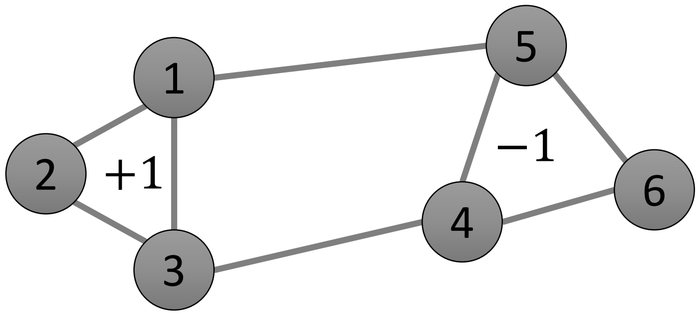
	1	2	3	4	5	6
1	3	-1	-1	0	-1	0
2	-1	2	-1	0	0	0
3	-1	-1	3	-1	0	0
4	0	0	-1	3	-1	-1
5	-1	0	0	-1	3	-1
6	0	0	0	-1	-1	2

- **Important properties:**
  - $x^T L x = \sum_{ij} L_{ij} x_i x_j = \sum_{(i,j)\in E} (x_i - x_j)^2 \geq 0$ for every x $\quad \boldsymbol{L = D - A}$
  - **Eigenvalues** are non-negative real numbers, $\boldsymbol{x = (1, \dots, 1)}$ then $\boldsymbol{L \cdot x = 0}$ and so $\boldsymbol{\lambda = \lambda_1 = 0}$
  - **Eigenvectors** are real and orthogonal

Graph Partitioning

- We desire 2 reasonably **large groups** of vertices with very **few edges** between them
  - Let's assign $+1$ and -1 to each vertex to represent two different groups, say vertex v_i is assigned value x_i.
  - If v_i and v_j are in different partitions, $\left(x_i - x_j\right)^2 = 4$, else 0.
- The number of edges between two partitions is

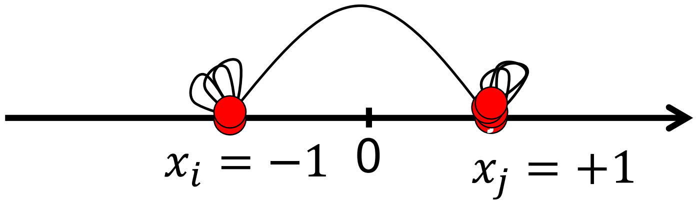
$$\sum_{\{(v_i, v_j) \in E\}} \frac{\left(x_i - x_j\right)^2}{4} = \frac{x^T L x}{4}$$



Graph Partitioning

- Assume we have a perfect partitioning. Exactly $|V|/2$ points are assigned $+1$ and the other half assigned -1.
- Therefore, we have

$$\sum_i x_i = 0,$$
$$\sum_i x_i^2 = |V|.$$



Graph Partitioning

- Our problem:

$$\min_{\mathbf{x}} \sum \frac{\left(x_i - x_j\right)^2}{4} = \frac{\boldsymbol{x}^T L \boldsymbol{x}}{4},$$

Subject to $\sum_i x_i = 0,$

$\sum_i x_i^2 = |V|.$

- The Lagrangian is $\frac{\boldsymbol{x}^T L \boldsymbol{x}}{4} + \eta_1(V - \boldsymbol{x}^T \boldsymbol{x}) + \eta_2(-\boldsymbol{x}^T \mathbf{1})$
- Take derivative $\nabla \frac{\boldsymbol{x}^T L \boldsymbol{x}}{4} + \nabla \eta_1(V - \boldsymbol{x}^T \boldsymbol{x}) + \nabla \eta_2(-\boldsymbol{x}^T \mathbf{1}) = 0$,

That is, $L\boldsymbol{x}$- $4\eta_1 \boldsymbol{x} - 2\eta_2 \mathbf{1} = \mathbf{0}$

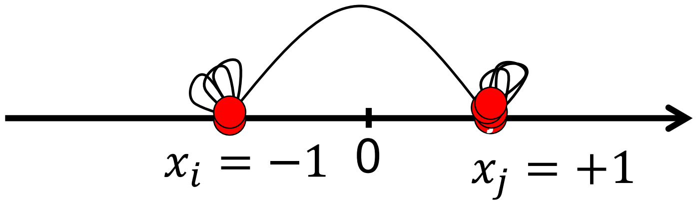
Graph Partitioning

- Multiply by $\mathbf{1}^T$, we have

$$\mathbf{1}^T L\boldsymbol{x} - 4\eta_1 \mathbf{1}^T \boldsymbol{x} - 2\eta_2 \mathbf{1}^T \mathbf{1} = 0,$$

$$\underbrace{\mathbf{1}^T L\boldsymbol{x}}_{=0} \quad \underbrace{\mathbf{1}^T \boldsymbol{x}}_{=0}$$

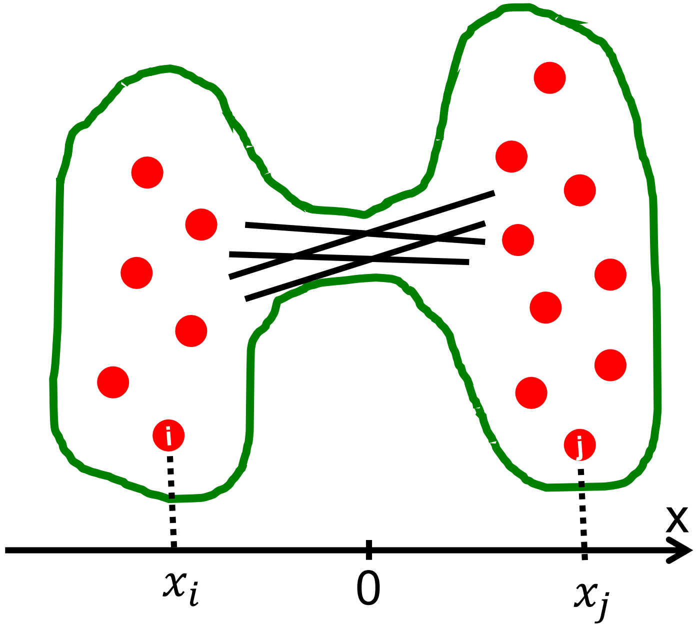
$$\eta_2 = 0.$$



- Finally, $L\boldsymbol{x} = 4\eta_1\boldsymbol{x}$. $4\eta_1$ is an eigenvalue, and corresponding eigenvector minimize $\boldsymbol{x}^T L\boldsymbol{x}/4$.
- Which eigenvalue? $\boldsymbol{x}^T L\boldsymbol{x}/4 \sim \eta_1$, make the eigenvalue as small as possible.
  - But the smallest eigenvalue is 0, and the corresponding eigenvector is all 1.
- Then the second smallest eigenvalue λ_2 minimizes $\boldsymbol{x}^T L\boldsymbol{x}/4$. The corresponding eigenvector v_2 gives us the partition.

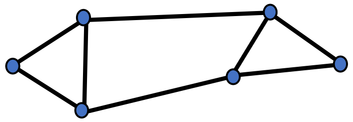
Graph Partitioning

- The eigenvector contains real values, not necessarily +1 and -1.
  - Assign positive entries of the eigenvector v_2 +1, and negative entries -1.
  - Sort the entries of v_2, and assign the smallest half entries +1, the other half -1.



Spectral Partitioning Algorithm

- 1) **Pre-processing:**
  - Build Laplacian matrix $\boldsymbol{L}$ of the graph



	1	2	3	4	5	6
1	3	-1	-1	0	-1	0
2	-1	2	-1	0	0	0
3	-1	-1	3	-1	0	0
4	0	0	-1	3	-1	-1
5	-1	0	0	-1	3	-1
6	0	0	0	-1	-1	2

- 2) **Decomposition:**
  - Find eigenvalues λ and eigenvectors $\boldsymbol{x}$ of the matrix $\boldsymbol{L}$

λ=

0.0
1.0
3.0
3.0
4.0
5.0

X =

0.4	0.3	-0.5	-0.2	-0.4	-0.5
0.4	0.6	0.4	-0.4	0.4	0.0
0.4	0.3	0.1	0.6	-0.4	0.5
0.4	-0.3	0.1	0.6	0.4	-0.5
0.4	-0.3	-0.5	-0.2	0.4	0.5
0.4	-0.6	0.4	-0.4	-0.4	0.0

  - Map vertices to corresponding components of λ_2

1	0.3
2	0.6
3	0.3
4	-0.3
5	-0.3
6	-0.6

How do we now find the clusters?

Spectral Partitioning

- **3) Grouping:**
  - Sort components of reduced 1-dimensional vector
  - Identify clusters by splitting the sorted vector in two

1	0.3
2	0.6
3	0.3
4	-0.3
5	-0.3
6	-0.6

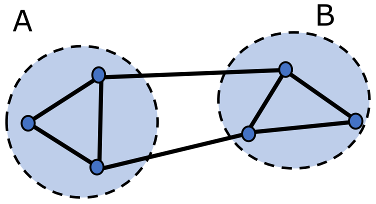
Split at 0:

Cluster A: Positive points

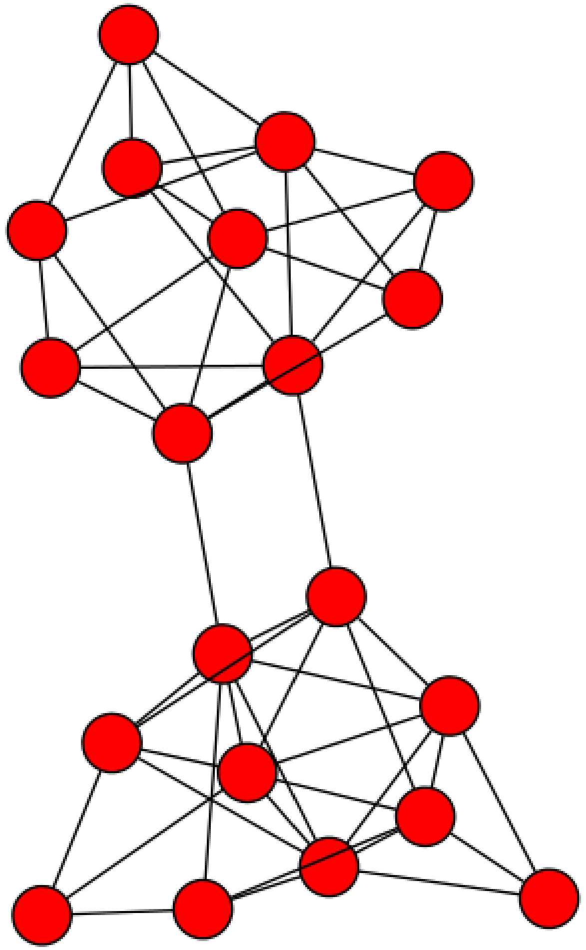
Cluster B: Negative points

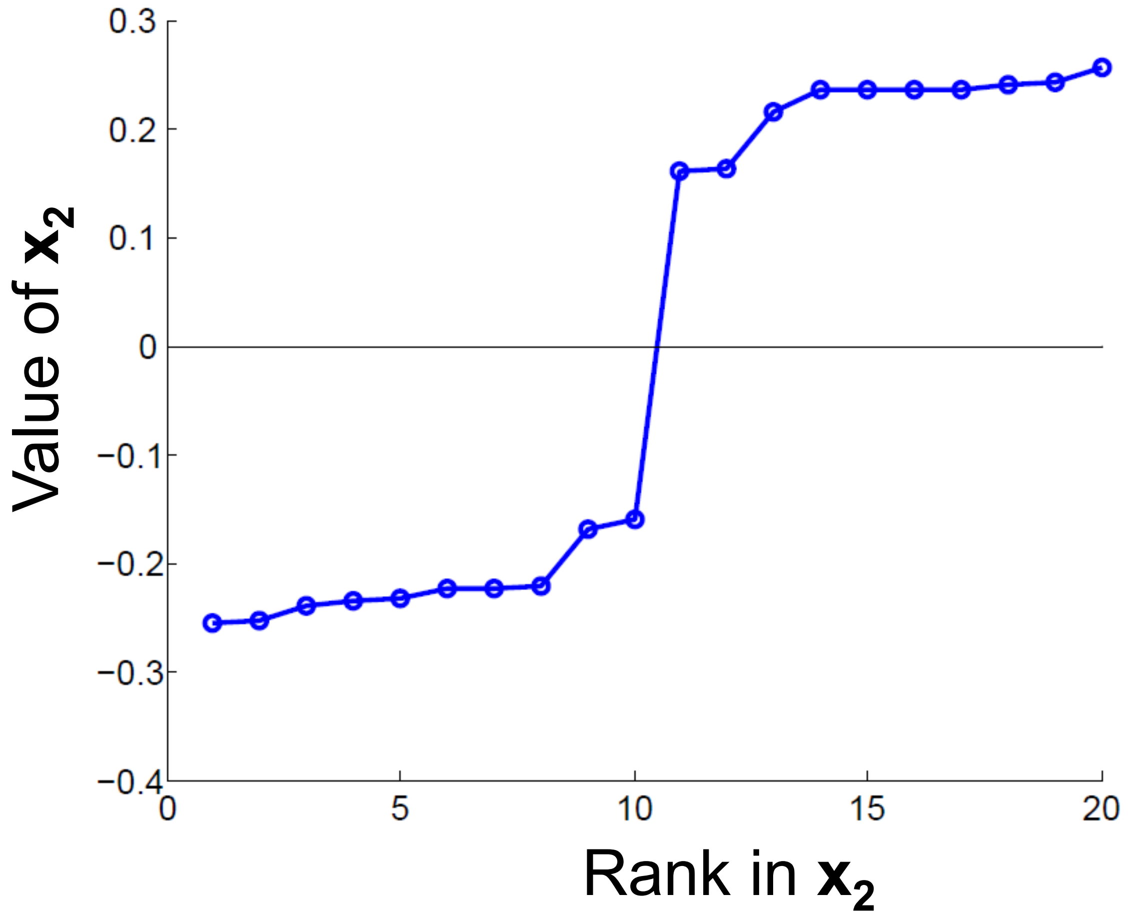
1	0.3
2	0.6
3	0.3

4	-0.3
5	-0.3
6	-0.6

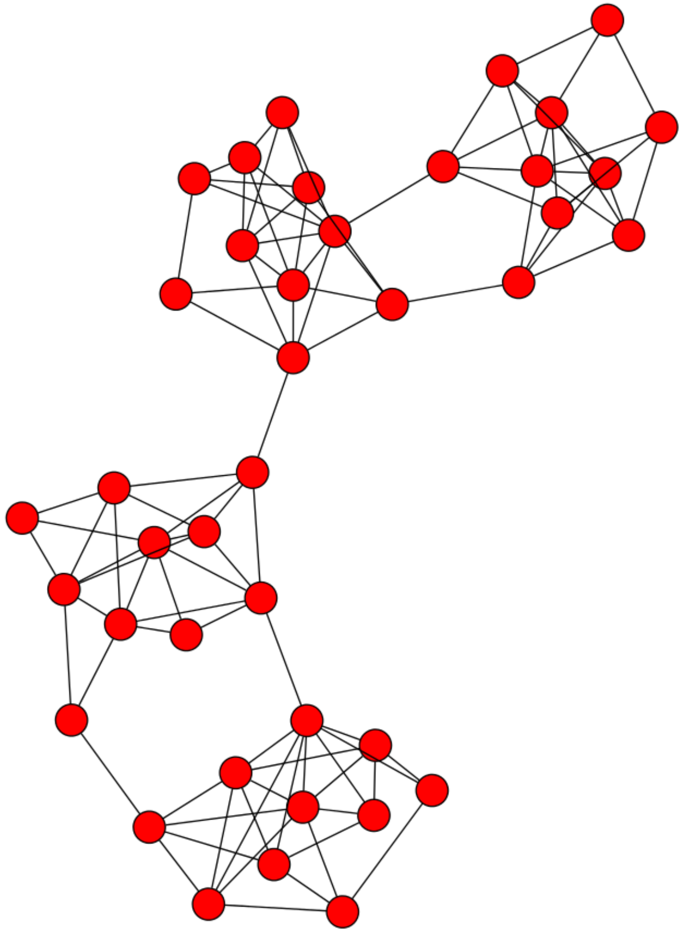


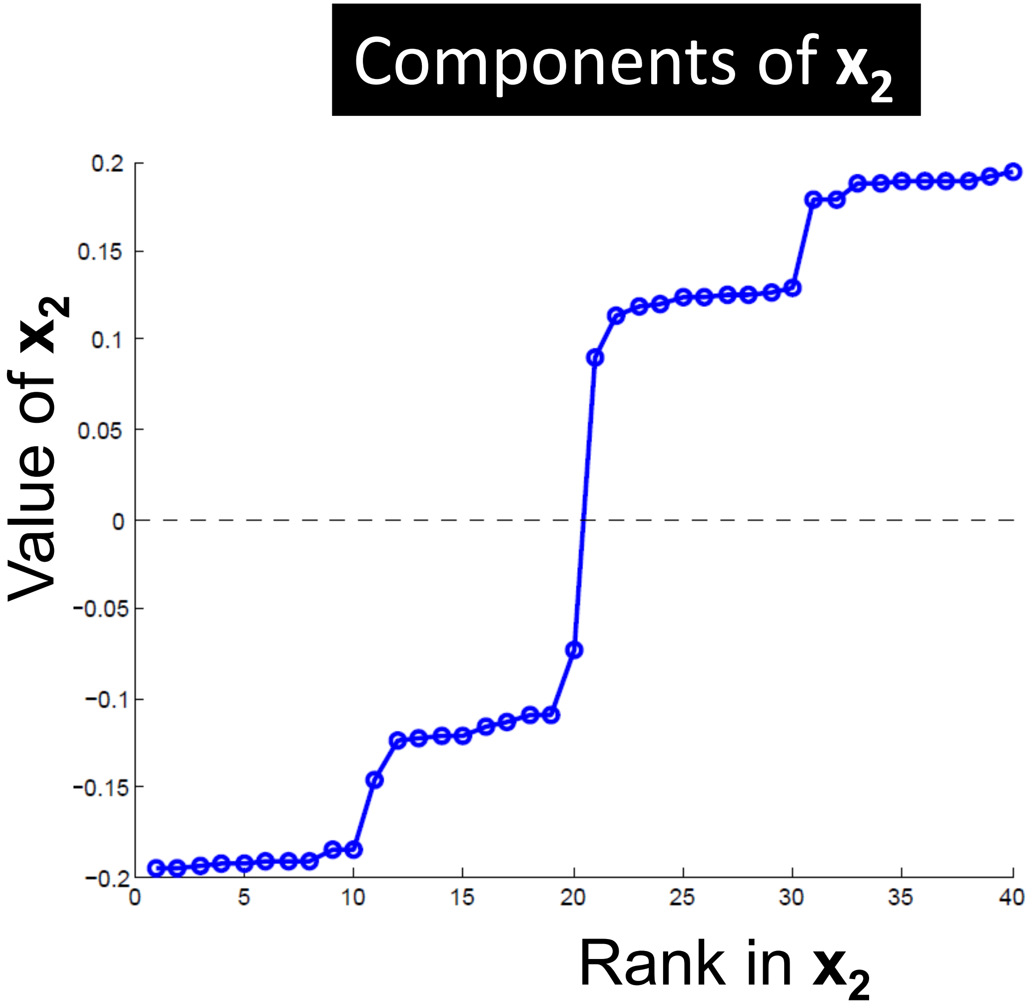
Example: Spectral Partitioning



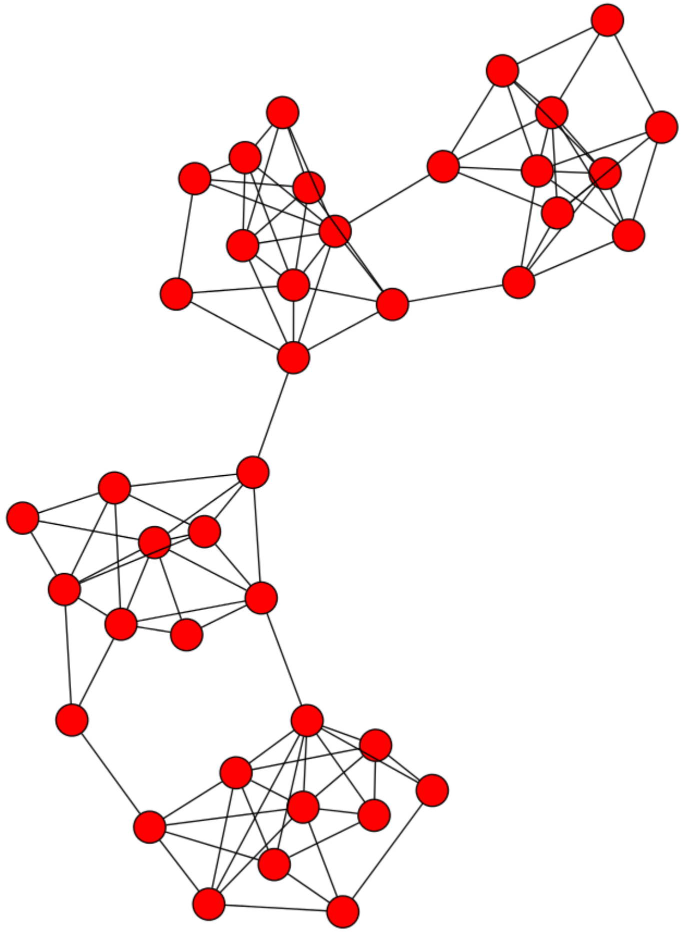


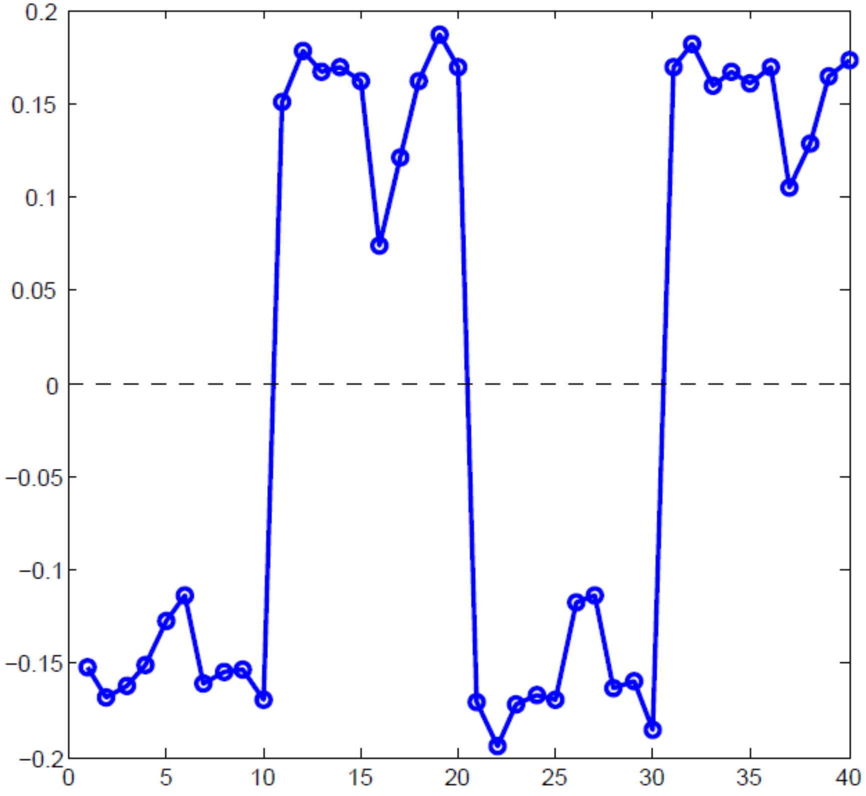
Example: Spectral Partitioning





Example: Spectral partitioning



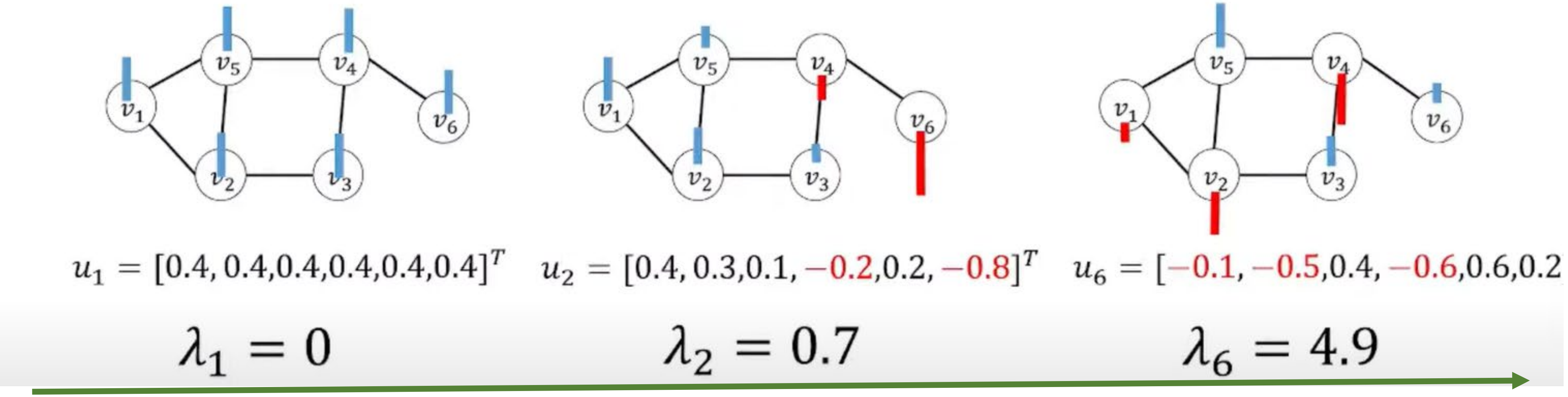


Components of $\mathbf{x}_3$

Spectral Domain

Orthonormal basis corresponding to graph frequency

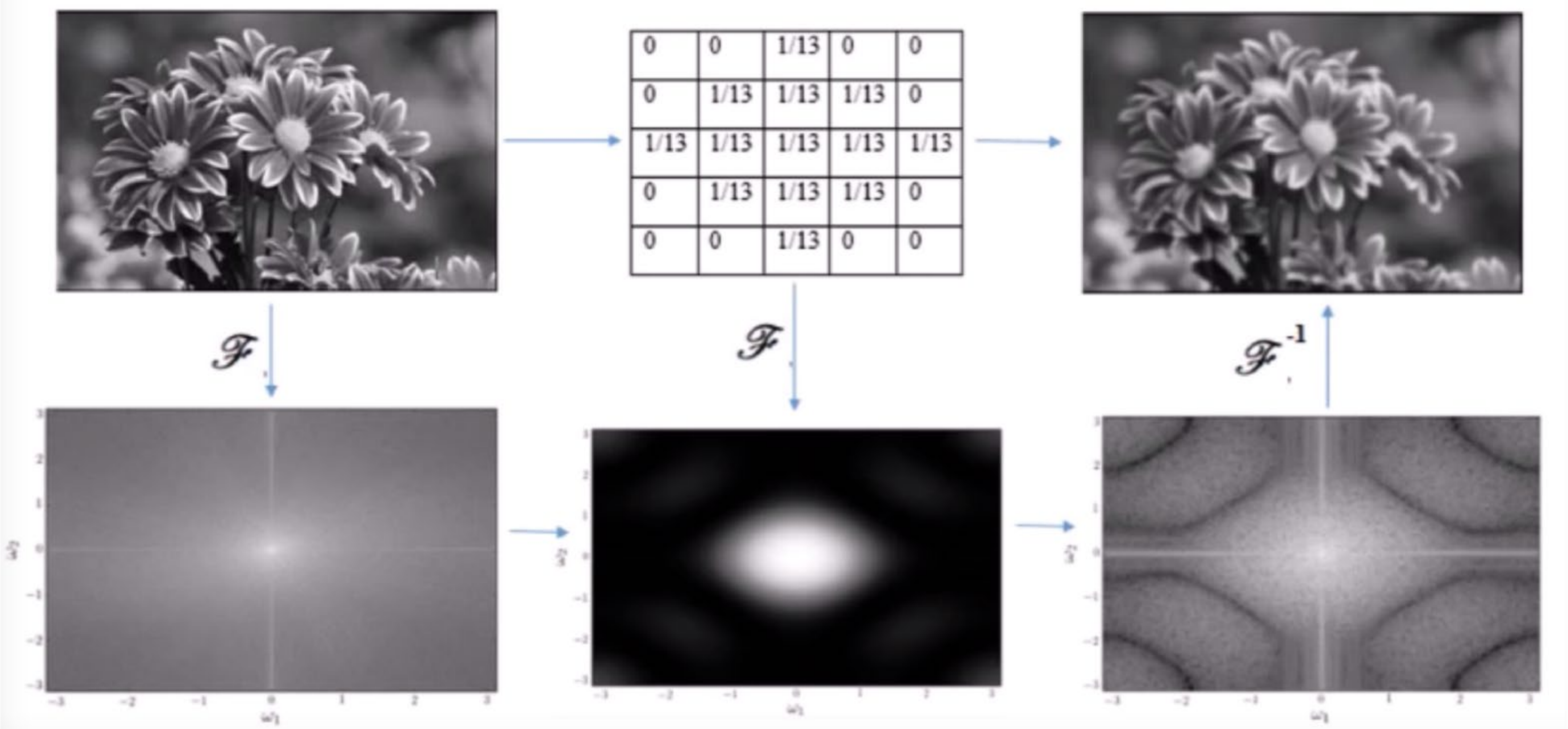
- $L = U\Lambda U^T = [u_1, u_2, \cdots u_n] \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix} [u_1, u_2, \cdots u_n]^T$



Low Frequency **High Frequency**

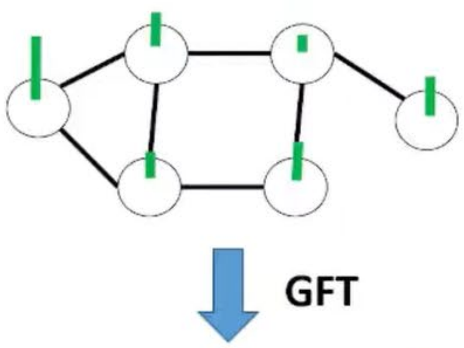
Spatial vs Spectral

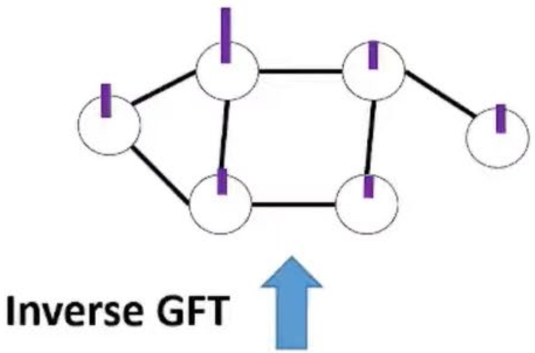
- Spatial vs spectral domain in image
  - Fourier transformation



Graph Convolution in Spectral Domain

- Graph Fourier transformation





$$F(x) = U^T x = \begin{bmatrix} \hat{x}_1 \\ \vdots \\ \hat{x}_6 \end{bmatrix}$$

$$F(g) = U^T g = \begin{bmatrix} \hat{g}_1 \\ \vdots \\ \hat{g}_6 \end{bmatrix}$$

$$g \star x = F^{-1}(\,F(g) \odot F(x)\,) = U(U^T g \odot U^T x)$$

$$= U(\begin{bmatrix} \hat{g}_1 \\ \vdots \\ \hat{g}_6 \end{bmatrix} \odot \begin{bmatrix} \hat{x}_1 \\ \vdots \\ \hat{x}_6 \end{bmatrix}) = U(\begin{bmatrix} \hat{g}_1 & & 0 \\ & \ddots & \\ 0 & & \hat{g}_6 \end{bmatrix} \begin{bmatrix} \hat{x}_1 \\ \vdots \\ \hat{x}_6 \end{bmatrix})$$

$$= \boldsymbol{U g_\theta U^T x} \quad , g_\theta = diag(U^T g)$$

The key difference in graph spectral convolution is the choice of $\boldsymbol{g_\theta}$

Further Readings on Graph Spectral Learning

- Bruna, Joan, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. "Spectral networks and locally connected networks on graphs." *arXiv preprint arXiv:1312.6203* (2013).

- Defferrard, Michaël, Xavier Bresson, and Pierre Vandergheynst. "Convolutional neural networks on graphs with fast localized spectral filtering." *Advances in neural information processing systems* 29 (2016).