# Numerical Linear Algebra Workshop on Eigenvalues

Zhouyang Lou

Industrial Engineering, Purdue University

April 4, 2016

# Overview

## Applications

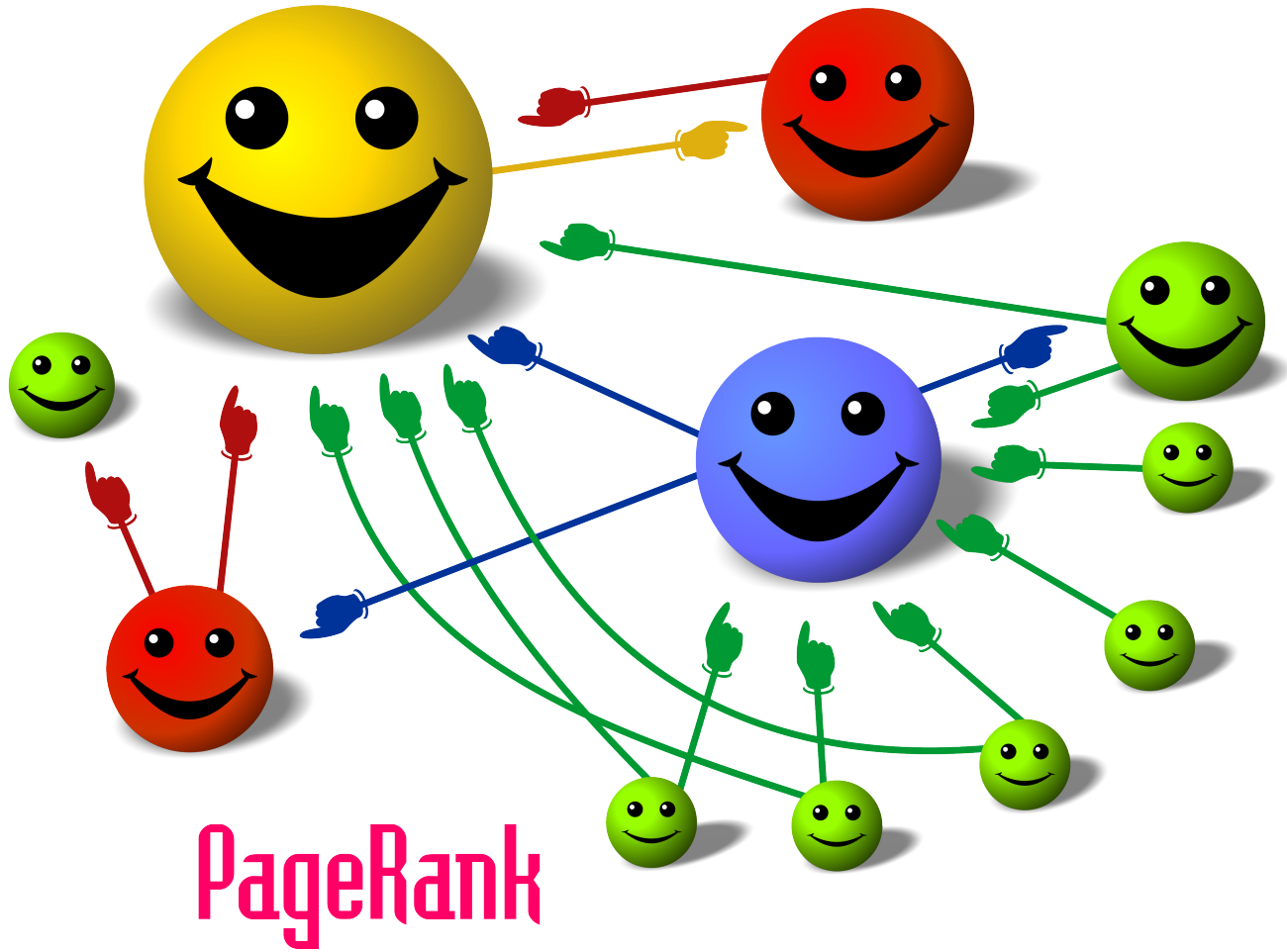
- PageRank problem
- Graph Clustering Problem

## Numerical Methods for Solving Eigenvalues

- Eigenvalue Problem Statement
- Normalized Power Method
- Simultaneous Iteration
- Arnoldi Method
- Lanczos Method
- QR Algorithm
- K-smallest Eigenvalue

## Conditioning of Eigenvalues

# PageRank



# PageRank

- Web search: e.g. Google
- PageRank is an objective measure of the citation importance of a web page. [Brin & Page 1998]
- What does importance mean?

## Definition

A page is important if other important pages link to it.

## Advantage

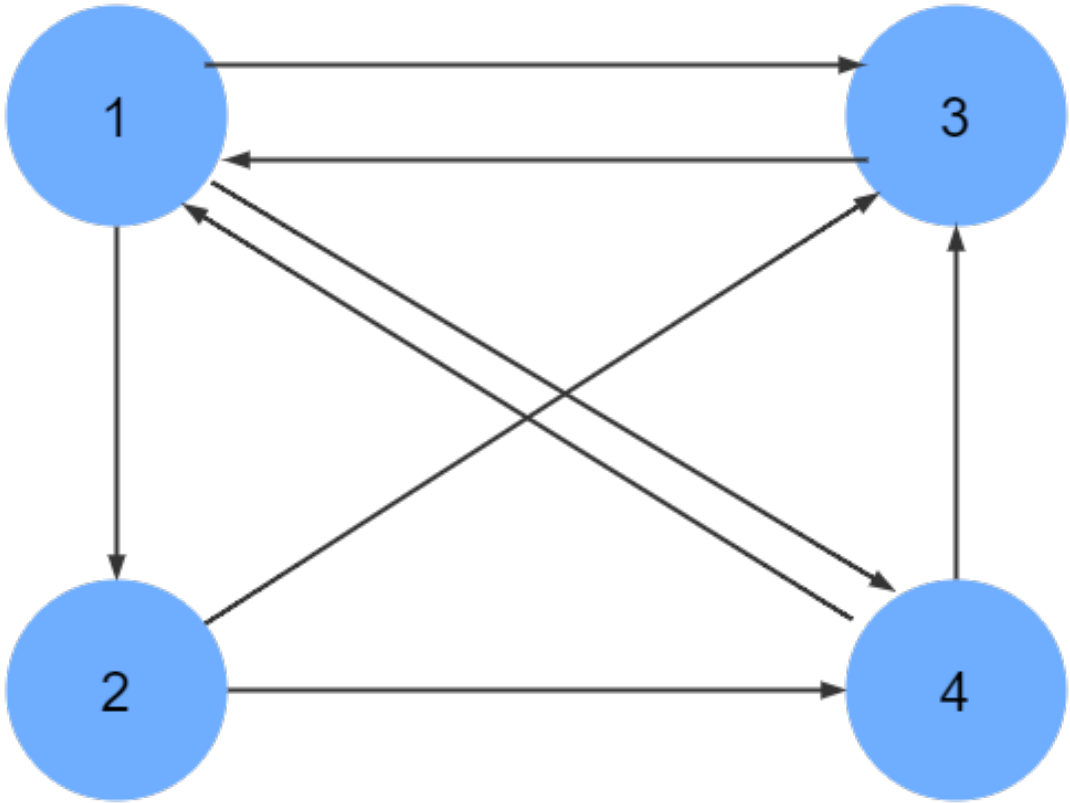
The importance of a web page does not depend on the contents, but what pages link to it.

# Procedure

1. Represent the PageRank problem as *graph*
2. Model the graph as *stochastic matrix*
3. Modify *stochastic matrix* to *Google matrix*
4. Compute the *PageRank*, which is the dominant eigenvector of *Google matrix*

# Make a Graph

## Example




Web Graph:
Web pages = nodes
Links = edges

# PageRank

- The PageRank model:

## Definition

Given a Webpage data set, the PageRank of the $i$th webpage is the $i$th element $\pi_i$ of $\pi$, which satisfies

$$\pi^T P = \pi^T$$

where $P$ is a matrix of weights of webpages that indicate their importance

- The stochastic matrix: $P$, is non-negative, defined by the hyperlink structure
- The PageRank vector: $\pi$, is probability vector satisfied $\pi^T \mathbf{e} = 1$ where $\mathbf{e}$ is a vector of all-ones.

# The Stochastic Matrix

## Definition

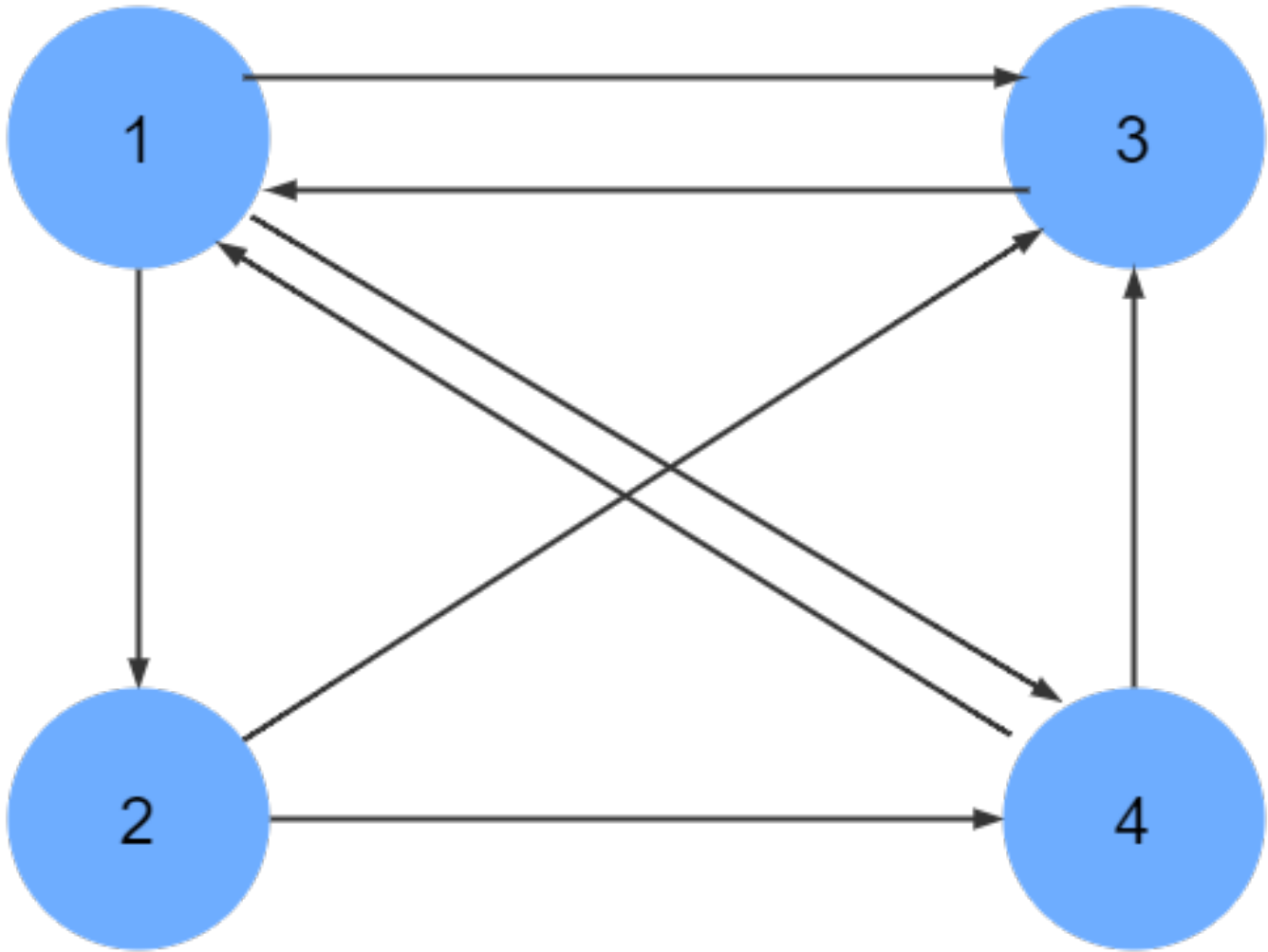
Let $d_i \geq 1$ be the number of pages pointing to it. The Google matrix $P = \{p_{ij}\}$ is defined as

$$p_{ij} = \begin{cases} 0, & \text{if there is no outlink from page } i \text{ to page } j; \\ 0, & \text{if } i = j; \\ 1/d_i, & \text{Otherwise} \end{cases}$$

# PageRank

## Example

4 web pages with directed links shown below



# The Stochastic Matrix

According to the construction of our model, the matrix could be

$$\begin{bmatrix} 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 1 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 \end{bmatrix}$$

# Modification

## Possible Difficulty 1

The existence of an all-zero row in the matrix, which could have very important pages but without any outlinks.

## Example

$$\begin{bmatrix} 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

# Modification

## Remdy: Example

$$\begin{bmatrix} 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 1 & 0 & 0 & 0 \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{bmatrix}$$

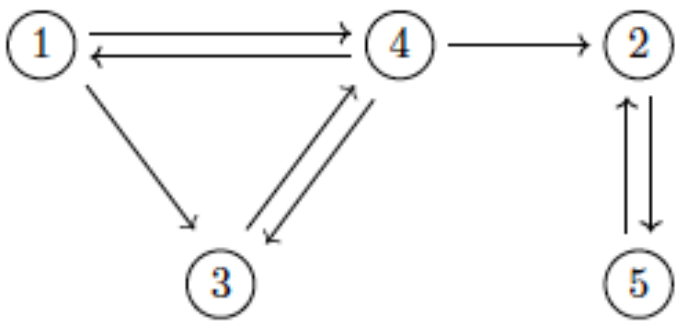
# Modification

## Possible Difficulty 2

Periodicity: a cyclic path in the Webgraph. (e.g. one points only to his wife's webpage and his wife points only to his.)

## Example: Periodicity

$$B = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{pmatrix} 0 & 0 & \star & \star & 0 \\ 0 & 0 & 0 & 0 & \star \\ 0 & 0 & 0 & \star & 0 \\ \star & \star & \star & 0 & 0 \\ 0 & \star & 0 & 0 & 0 \end{pmatrix} \end{array} \xrightarrow{\text{permutation matrix P}} B = \begin{array}{c} \\ 1 \\ 3 \\ 4 \\ 2 \\ 5 \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 3 & 4 & 2 & 5 \end{array} \\ \begin{pmatrix} 0 & \star & \star & 0 & 0 \\ 0 & 0 & \star & 0 & 0 \\ \star & \star & 0 & \star & 0 \\ 0 & 0 & 0 & 0 & \star \\ 0 & 0 & 0 & \star & 0 \end{pmatrix} \end{array}$$



# Modification

## Remedy: The Google Matrix

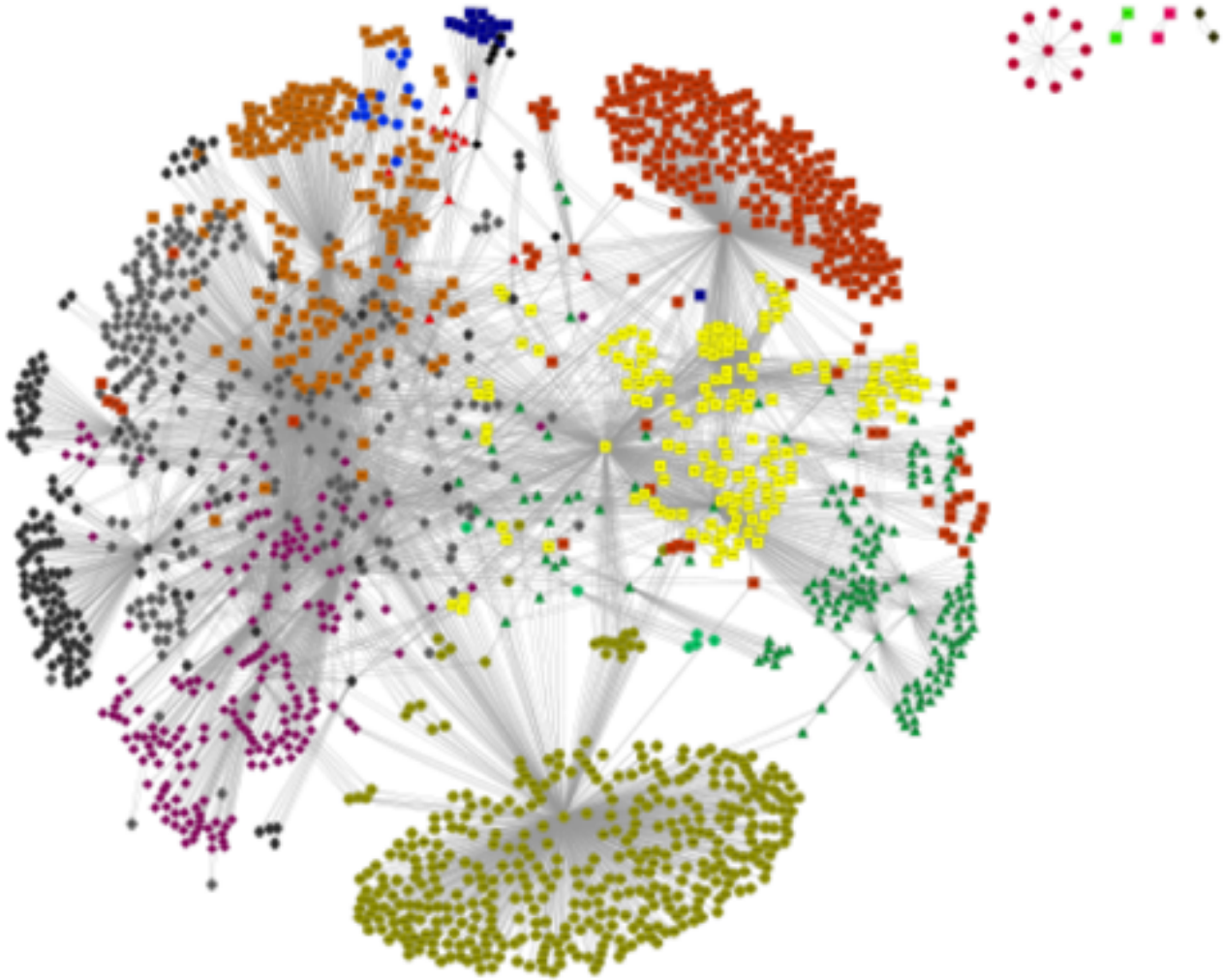
$P$ can be replaced by

$$P_\alpha = \alpha P + (1 - \alpha)\mathbf{e}\mathbf{v}^T$$

where $\alpha \in [0, 1)$ and $\mathbf{v}$ a personalized probability vector satisfied $\mathbf{v} \geq 0$ and $\mathbf{e}\mathbf{v}^T = 1$

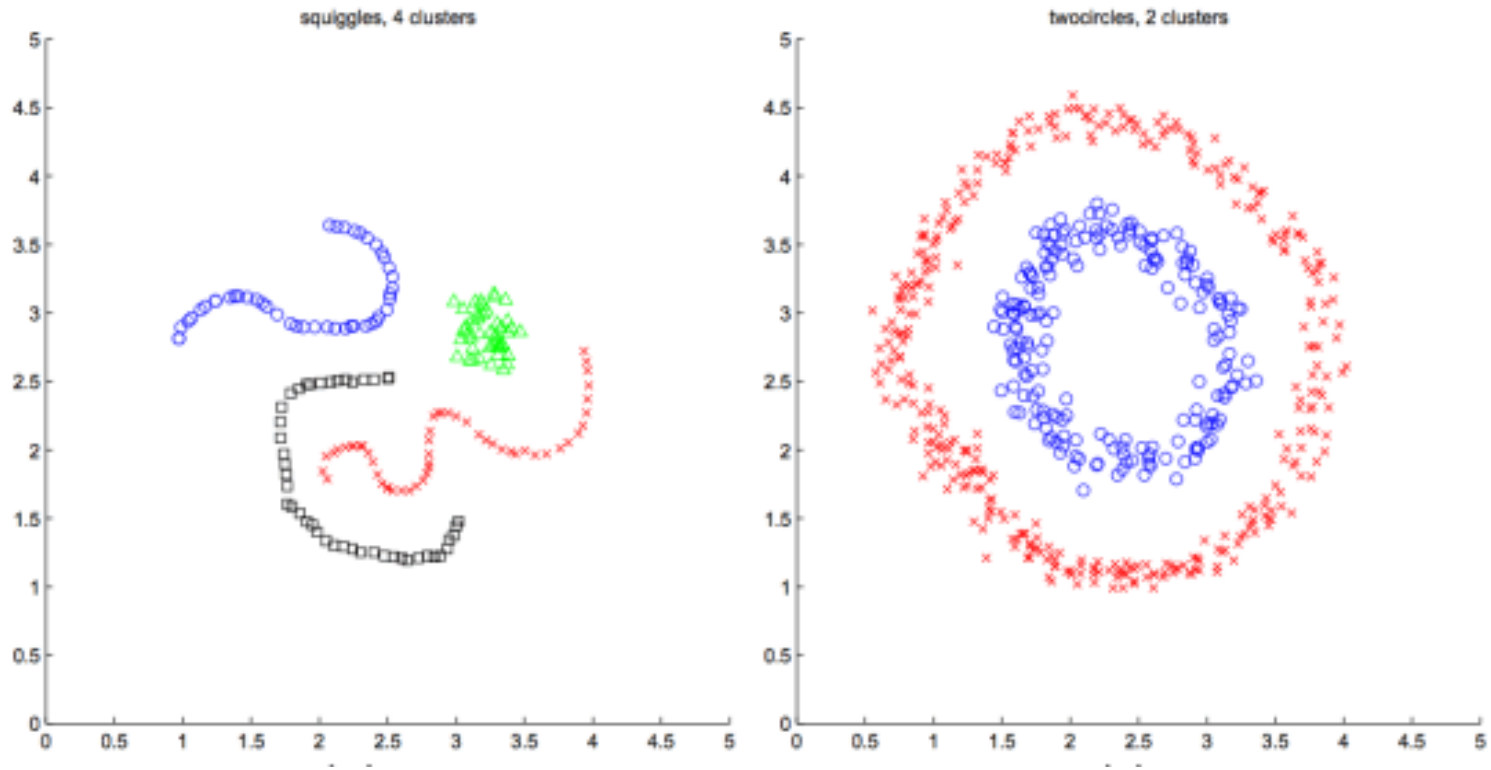
# Graph Clustering Problem

## Example: Community Detection



# Graph Clustering Problem

## Example: Spectral Clustering



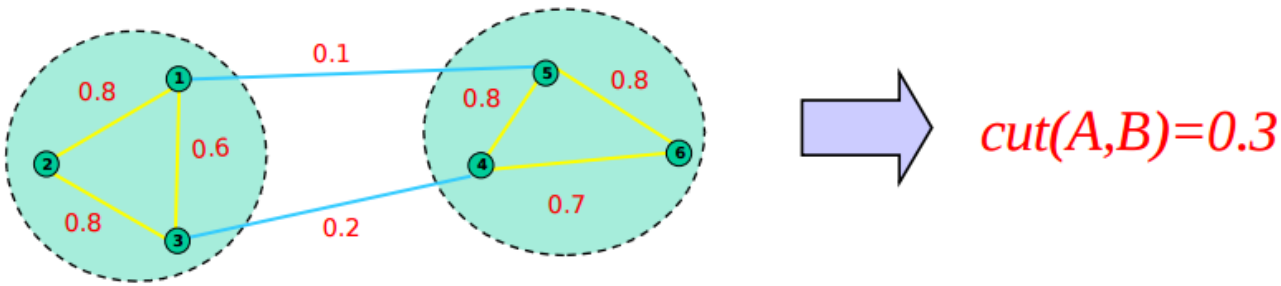
# Clustering Idea

- Objective:
  1. Data points assigned to the same cluster should be highly similar.
  2. Data points assigned to the different cluster should be highly dissimilar.
- Apply the objectives to our graph representation:
  1. Maximize the weight of within-group connections.
  2. Minimize the weight of between-group connections.

# Graph Partition

- **Min-cut**: Partition graph into two sets A and B such that weight of edges connecting vertices in A to vertices in B is minimum. And can be expressed as

$$\min Cut(A, B) = \sum_{i \in A, j \in B} W_{ij}$$




NP-hard!

# Procedure

For undirected graph

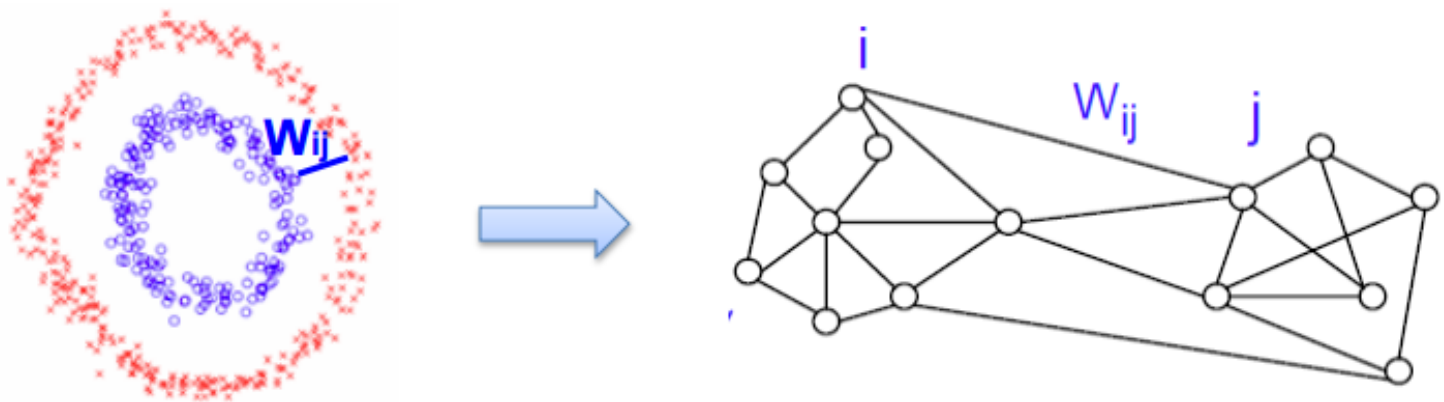
1. Construct pairwise adjacency matrix.
2. Construct degree matrix
3. Compute Laplacian
4. Compute the first $k$ eigen-vectors of the Laplacian
5. Construct a new matrix $U$ with the eigenvectors as columns
6. Cluster the points corresponding to the $i$th row of $U$ with K-means

# Construct Adjacency Matrix

## E.g.Gaussian Kernel similarity function:

$$W_{ij} = e^{\frac{||x_i - x_j||^2}{2\sigma^2}}$$

## Example



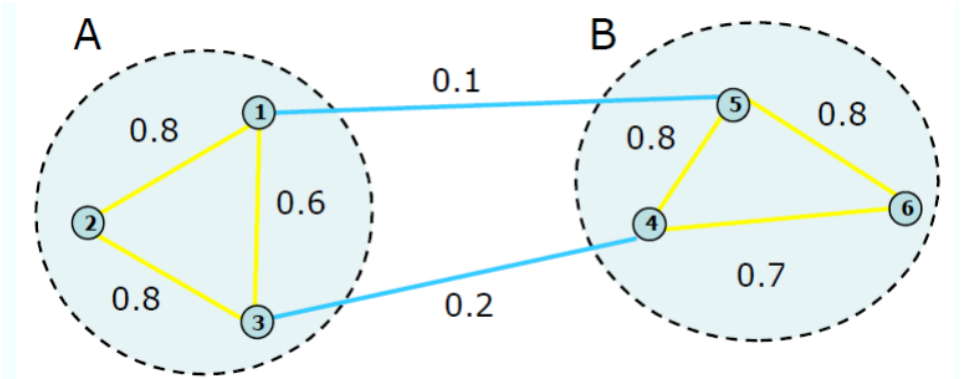
# Construct Adjacency Matrix

## E.g. Social Networks:

For a graph with $n$ vertices, the entries of the $n \times n$ adjacency matrix are defined by

$$A := \begin{cases} A_{ij} = 1 & \text{if there is an edge connecting two vertices;} \\ A_{ij} = 0 & \text{if there is no edge;} \\ A_{ii} = 0 \end{cases}$$

## Example



# Construct Adjacency Matrix

Example:

| | $X_1$ | $X_2$ | $X_3$ | $X_4$ | $X_5$ | $X_6$ |
|---|---|---|---|---|---|---|
| $X_1$ | 0 | 0.8 | 0.6 | 0 | 0.1 | 0 |
| $X_2$ | 0.8 | 0 | 0.8 | 0 | 0 | 0 |
| $X_3$ | 0.6 | 0.8 | 0 | 0.2 | 0 | 0 |
| $X_4$ | 0 | 0 | 0.2 | 0 | 0.8 | 0.7 |
| $X_5$ | 0.1 | 0 | 0 | 0.8 | 0 | 0.8 |
| $X_6$ | 0 | 0 | 0 | 0.7 | 0.8 | 0 |

# Construct Laplacian Matrix

## Degree Matrix $D$:

A degree matrix $D$ is an $n \times n$ diagonal matrix that contains information about the degree of each vertex $v_i$, which is the number of edges incident to the vertex.

## Laplacian Matrix $L$:

$L = D - A$

Some Properties:

- $L$ is symmetric and positive semi-definite
- The smallest eigenvalue is 0, the corresponding eigenvector is the constant 1.

# Construct Laplacian Matrix

## Example:

| | $X_1$ | $X_2$ | $X_3$ | $X_4$ | $X_5$ | $X_6$ |
|---|---|---|---|---|---|---|
| $X_1$ | 1.5 | -0.8 | -0.6 | 0 | -0.1 | 0 |
| $X_2$ | -0.8 | 1.6 | -0.8 | 0 | 0 | 0 |
| $X_3$ | -0.6 | -0.8 | 1.6 | -0.2 | 0 | 0 |
| $X_4$ | 0 | 0 | -0.2 | 1.7 | -0.8 | -0.7 |
| $X_5$ | -0.1 | 0 | 0 | -0.8 | 1.7 | -0.8 |
| $X_6$ | 0 | 0 | 0 | -0.7 | -0.8 | 1.5 |

# Compute Eigenvector

## Example:

```
lambda =

    0.0000         0         0         0         0         0
         0    0.1882         0         0         0         0
         0         0    2.0840         0         0         0
         0         0         0    2.2853         0         0
         0         0         0         0    2.4690         0
         0         0         0         0         0    2.5735

x =

    0.4082   -0.4084
    0.4082   -0.4418
    0.4082   -0.3713
    0.4082    0.3713
    0.4082    0.4050
    0.4082    0.4452
```

# Eigenvalue Problem

Given $n \times n$ matrix $A$, find Find scalar $\lambda$ and nonzero $x$ such that

$$A\mathbf{x} = \lambda\mathbf{x}$$

where $\lambda$ is eigenvalue, and $x$ is corresponding eigenvector.

# Normalized Power Method

Good for approximating the dominant eigenvalue.

Procedure:

1. Pick an initial guess $\mathbf{x}_0$
2. Repeat

$$\mathbf{y}_k = A\mathbf{x}_{k-1}$$

$$\mathbf{x}_k = \frac{\mathbf{y}_k}{||\mathbf{y}_k||_\infty}$$

   until the chosen stopping criteria.
3. Normalization applied since the geometric growth of components at each iteration may eventually cause overflow/underflow.
4. Note: The maximum eigenvalue could be obtained by the Rayleigh Quotient(find $r(\mathbf{x}) = \alpha$ such that $||A\mathbf{x} - \alpha\mathbf{x}||_2$)

$$r(\mathbf{x}_k) = \frac{\mathbf{x}_k^T A\mathbf{x}_k}{\mathbf{x}_k^T \mathbf{x}_k}$$

# Power Method Convergence

1. Why it works?
2. Express the starting vector $\mathbf{x}_0$ as linear combination

$$\mathbf{x}_0 = \sum_{i=1}^{n} \alpha_i \mathbf{v}_i$$

   where $\mathbf{v}_i$'s are eigenvectors of $A$, and $|\lambda_1| > |\lambda_2| > \ldots > |\lambda_n|$
3. Then

$$x_k = Ax_{k-1} = A^2 x_{k-2} = \ldots = A^k x_0 =$$

$$\sum_{i=1}^{n} \lambda_i^k \alpha_i \mathbf{v}_i = \lambda_1^k (\alpha_1 \mathbf{v}_1 + \sum_{i=2}^{n} (\frac{\lambda_i}{\lambda_1})^k \alpha_i \mathbf{v}_i)$$

4. Since $|\frac{\lambda_i}{\lambda_1}| < 1$ for $i > 1$, and higher powers go to zero, leaving $\lim_{k\to\infty} x_k = \lambda_1^k \alpha_1 \mathbf{v}_1$

# Limitations of Power Method

1. There may be more than one eigenvalue dominants others.(e.g. $|\lambda_1| = |\lambda_2|$)
2. For real matrix and starting vector, iteration can never converge to complex vector

## Good conditions:

When matrix is symmetric.

# Simultaneous Iteration

- What if we want different eigenvalues of a matrix?
- We use *Simultaneous iteration*, also called *Subspace Method*, which repeatedly multiplies matrix times matrix of initial starting vectors.
- Starting from $n \times p$ matrix $\mathbf{X}_0$ of rank $p$, iteration scheme is

$$\mathbf{X}_k = A\mathbf{X}_{k-1}$$

- $span(\mathbf{X}_k)$ converges to invariant subspace determined by $p$ largest eigenvalues of $A$, provided $|\lambda_p| > |\lambda_{p+1}|$

# Simultaneous Iteration

- But Each column of $\mathbf{X}_k$ converges to dominant eigenvector, so columns of $\mathbf{X}_k$ become increasingly ill-conditioned basis for $span(\mathbf{X}_k)$
- Remedy is to obtain an orthonormal set of eigenvector estimates during each iteration, forcing the eigenvector approximations to be orthogonal at all times. **Orthogonalization is the basis of Arnoldi and Lanczos method.**
- This is realized by using *QR factorization*, or *modified Gram-Schmidt process*.

# Arnoldi Method

- Given $A \in \mathcal{R}^{n \times n}$ and $q_0 \in \mathcal{R}^n$, compute a sequence of orthonormal basis

$$Q_k = [q_0, q_1, \ldots, q_{k-1}]$$

or Krylov subspace

$$K_k = span\{q_0, Aq_0, \ldots, A^{k-1}q_0\}$$

so that

$$AQ_k = Q_{k+1}H_{k+1,k}, Q_k^T AQ_k = H_{k,k}$$

where $H_{k+1,k} \in \mathcal{R}^{(k+1) \times k}$ is upper Hessenberg

Recall: A subspace should satisfy following three conditions: contains zero vector, addition and scaling.

# Arnoldi Method

- Let $\lambda$ and $\mathbf{x}$ be the eigenvalue and eigenvector of $A$,

$$Q_n^T A Q_n (Q_n^T \mathbf{x}) = \lambda (Q_n^T \mathbf{x})$$

- Let$H_{k,k} = Q_k^T A Q_k$ also upper Hessenberg. The eigenvalue of $H_{k,k}$ called Ritz values of $A$, are approximate eigenvalues of $A$, and the eigenvectors of $H_{k,k}$ are corresponding approximate eigenvectors of $A$.

# Arnoldi Method

- Given an arbitrary starting vector $\mathbf{q}$, $q, Aq, \ldots, A^k q$ usually ill-conditioned
- therefore: replace these vectors by orthogonal vectors $q_1, \ldots, q_{k+1}$ that span the same subspace
- Gram-Schmidt process with slight modification
- theory: orthogonalize $A^k q$ against $q_1, \ldots, q_k$
- practice: orthogonalize $Aq_k$ against $q_1, \ldots, q_k$
- produces exactly the same sequence of vectors as the Gram-Schmidt process applied to $q, Aq, \ldots, A_q^k$

# Arnoldi Method

- Normalization

$$q_1 = \frac{q}{||q||_2}$$

On subsequent steps $k = 1, 2, \ldots$ take

$$\tilde{q}_{k+1} = A_{q_k} - \sum_{j=1}^{k} q_j h_{jk}$$

where $h_{jk}$ is the Gram-Schmidt coefficient $h_{jk} =< A_{q_k}, q_j >$.

- Normalization

$$\tilde{q}_{k+1} = \frac{\tilde{q}_{k+1}}{h_{k+1,k}}$$

where $h_{k+1,k} = ||\tilde{q}_{k+1}||_2$

# Limitation and Modification

- Arnoldi iteration fairly expensive in work and storage because each new vector $q_k$ must be orthogonalized against all previous columns of $Q_k$, and all must be stored for that purpose.

# Algorithm

1: $q_1 = \dfrac{q}{||q||_2}$
2: **for** $k = 1$ to $m - 1$ **do**
3: $q_{k+1} = Aq_k$
4: **for** $j = 1$ to $k$ **do**
5: $h_{jk} = q_j q_{k+1}$
$q_{k+1} = q_{k+1} - q_j h_{jk}$
6: **end for**
7: %reorthogonalize
$h_{k+1,k} = ||q_{k+1}||_2$
**if** $h_{k+1,k} = 0$ **then**
| stop
**end**
8: $q_{k+1} = \dfrac{q_{k+1}}{h_{k+1,k}}$
Compute eigenvalues and eigenvectors by using QR decomposition
9: **end for**

# Lanczos Method

- Lanczos method is actually a special case of Arnoldi method, and in this case, $T_{k,k} = Q_k^T A Q_k$, where $A$ is symmetric.
- The process can be written in the form

$$AQ_k = Q_k T_k + q_{k+1}\beta_k e_k^T$$

, where $T$ is

$$T_k = \begin{bmatrix} \alpha_1 & \beta_1 & & \\ \beta_1 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_{k-1} \\ & & \beta_{k-1} & \alpha_k \end{bmatrix}$$

# Algorithm

1: $q_1 = \frac{q}{||q||_2}$
2: **for** $k = 1$ to $m - 1$ **do**
3: $q_{k+1} = Aq_k$
$\alpha_k = q_k q_{k+1}$
$q_{k+1} = q_{k+1} - \alpha_k q_k$
**if** $k > 1$ **then**
$q_{k+1} = q_{k+1} - q_{k-1}\beta_{k-1}$
**end**
4: $\beta_k = ||q_{k+1}||_2$
**if** $\beta_k = 0$ **then**
stop
**end**
5: $q_{k+1} = \frac{q_{k+1}}{\beta_k}$
Compute eigenvalues and eigenvectors by using QR decomposition
6: **end for**

# QR Algorithm

1. The QR decomposition (also called the QR factorization) of a matrix is a decomposition of the matrix into an orthogonal matrix and a triangular matrix.
2. Given

$$A = \begin{bmatrix} a_1 | & a_2 | & \dots | & a_n \end{bmatrix}$$

, then

$$\begin{aligned}
\mathbf{u}_1 &= \mathbf{a}_1, & \mathbf{e}_1 &= \frac{\mathbf{u}_1}{\|\mathbf{u}_1\|}, \\
\mathbf{u}_2 &= \mathbf{a}_2 - (\mathbf{a}_2 \cdot \mathbf{e}_1)\mathbf{e}_1, & \mathbf{e}_2 &= \frac{\mathbf{u}_2}{\|\mathbf{u}_2\|}. \\
\mathbf{u}_{k+1} &= \mathbf{a}_{k+1} - (\mathbf{a}_{k+1} \cdot \mathbf{e}_1)\mathbf{e}_1 - \cdots - (\mathbf{a}_{k+1} \cdot \mathbf{e}_k)\mathbf{e}_k, & \mathbf{e}_{k+1} &= \frac{\mathbf{u}_{k+1}}{\|\mathbf{u}_{k+1}\|}.
\end{aligned}$$

$$A = \begin{bmatrix} \mathbf{a}_1 \,|\, \mathbf{a}_2 \,|\, \cdots \,|\, \mathbf{a}_n \end{bmatrix} = \begin{bmatrix} \mathbf{e}_1 \,|\, \mathbf{e}_2 \,|\, \cdots \,|\, \mathbf{e}_n \end{bmatrix} \begin{bmatrix} \mathbf{a}_1 \cdot \mathbf{e}_1 & \mathbf{a}_2 \cdot \mathbf{e}_1 & \cdots & \mathbf{a}_n \cdot \mathbf{e}_1 \\ 0 & \mathbf{a}_2 \cdot \mathbf{e}_2 & \cdots & \mathbf{a}_n \cdot \mathbf{e}_2 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{a}_n \cdot \mathbf{e}_n \end{bmatrix} = QR.$$

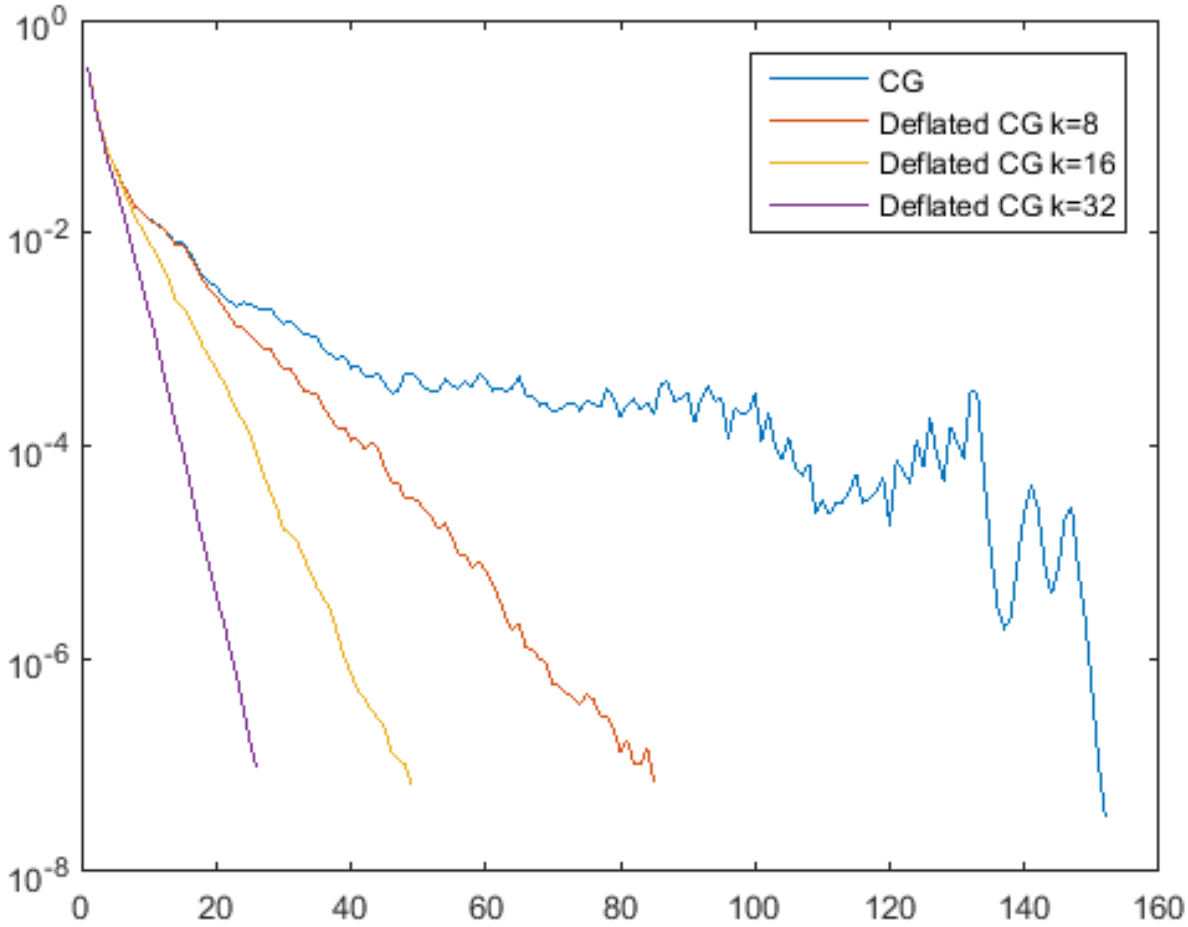
# Condion of use

| Method | Applies to | Produces |
|---|---|---|
| Power iteration | General | Eigenpair with largest value |
| Arnoldi iteration | General | Hessenberg |
| Lanczos iteration | Hermitian(symmetric) | Tridiagonal |
| QR Decompostion | Hessenberg(tridiagonal) | all eigenpairs |

| Method | Convergence rate |
|---|---|
| Power iteration | linear |
| QR Decomposition | Cubic |

# K-smallest Eigenvalue

Converge faster after removing more k smallest eigenvalues.




By Zhengyi

# Conditioning of Eigenvalues

- Condition of eigenvalue problem is sensitivity of eigenvalues and eigenvectors to changes in matrix
- Conditioning of eigenvalue problem is not same as conditioning of solution to linear system for same matrix
- Different eigenvalues and eigenvectors are not necessarily equally sensitive to perturbations in matrix

## Condition number of a given eigenvalue

$$\kappa(\lambda, A) = \sup_{\delta A} \frac{|\delta\lambda|}{||\delta A||} = \frac{1}{\cos\theta_{xy}}$$

# Useful Tips from Condition Number

- Matlab code: condeig()
- Unitarily diagonalizable matrices $\mathbf{y} = \mathbf{x}$, and $[\kappa(\lambda, A) = 1$, are perfectly conditioned.
- Non-diagonalizable matrices' eigenvalues are very badly conditioned

Thank you!