Eigenvalues, Eigenvectors and Applications

Intensive Computation

Annalisa Massini
2018/2019

INTRODUCTION

Eigenvalue Problems

- Solving linear systems $\boldsymbol{Ax} = \boldsymbol{b}$ is one part of numerical linear algebra, and involves manipulating the rows of a matrix
- The second main part of numerical linear algebra is about find **eigenvalues** and **eigenvectors**
- This is done by *transforming a matrix to leave its eigenvalues unchanged*

Eigenvalue Problems

- The standard algebraic **eigenvalue problem** is:

Given an nxn matrix A, find a scalar λ and a nonzero vector x such that $\mathbf{Ax} = \lambda\mathbf{x}$

where: λ is an ***eigenvalue*** of A

$\boldsymbol{x}$ (non-zero) is the corresponding ***eigenvector***

Example

$$\mathrm{Av} = \begin{pmatrix} 1 & 2 \\ 8 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 2 \end{pmatrix} = 5 \begin{pmatrix} 1 \\ 2 \end{pmatrix} = \lambda v$$

v=(1,2) is an eigenvector

$\lambda = 5$ an eigenvalue

Eigenvalue Problems

Eigenvalue problems occur in many areas of science and engineering:

- The *natural modes and frequencies of vibration* of a structure are determined by the eigenvectors and eigenvalues of an appropriate matrix
- The *stability of the structure* is determined by the locations of the eigenvalues
- Eigenvalues are useful in *analyzing numerical methods* (the convergence analysis of iterative methods for solving systems of algebraic equations, and the stability analysis of methods for solving systems of differential equations)
- *Graph theory*

Eigenvalue Problems

- An **eigenvector** of a matrix ***determines a direction*** in which the *effect of the matrix is particularly simple*:
  - The ***matrix expands or shrinks*** any vector lying in that direction by a scalar multiple, and
  - the *expansion or contraction factor* is given by the ***corresponding eigenvalue***

- Thus, eigenvalues and eigenvectors provide a means of **understanding the complicated behavior** of a general linear transformation by decomposing it into simpler actions

Eigenvalue Problems

- Although many examples involve only **real matrices**, both the theory and computational procedures are generally applicable to **complex matrices**

- The notation difference for complex matrices is that the conjugate transpose, A^H, is used instead of the transpose, A^T

- The set of all the eigenvalues of a matrix A is called the ***spectrum*** of A and is denoted by $\lambda(A)$

- The maximum modulus of the eigenvalues is called the **spectral radius** of A: $\rho(A) = \max\{|\lambda| : \lambda \text{ in } \lambda(A)\}$

Eigenvalue Problems

- The equation $Ax = \lambda x$ is equivalent to

$$(A - \lambda I)\, x = 0$$

- This homogeneous equation has a **nonzero solution x** if and only if its matrix is **singular**, that is the eigenvalues of A are the values λ such that

$$\mathbf{\det(A - \lambda I) = 0}$$

- $\det(A - \lambda I)$ is a polynomial of degree n in λ:
  - It is the **characteristic polynomial** of A
  - *Its roots are the **eigenvalues*** of A

Eigenvalue Problems

- An nxn matrix A always has n eigenvalues (*Fundamental Theorem of Algebra*)
- Eigenvalues need be neither *distinct* nor *real*

- The product of the eigenvalues is $\det A = \prod_{i=1}^{n} \lambda_i$

- The sum of the eigenvalues is $\sum_{i=1}^{n} a_{ii} = \sum_{i=1}^{n} \lambda_i$ called ***trace***

POWER METHOD

Computing Eigenvalues and Eigenvectors

- Many numerical methods for **computing eigenvalues and eigenvectors** are based on **reducing the original matrix** to a simpler form, whose eigenvalues and eigenvectors are then **easily determined**

- **Finding the eigenvalues and eigenvectors** is equivalent to **transforming** the underlying system of equations into a special set of coordinate axes in which the **matrix is diagonal**

- The **eigenvalues** are the entries of the **diagonal** matrix

- The eigenvectors are the new set of coordinate axes

Similarity Transformations

- We need to identify:
  - what types of transformations preserve eigenvalues
  - for what types of matrices the eigenvalues are easily determined

- A matrix B is ***similar*** to a matrix A if there is a nonsingular matrix T such that

$$B = T^{-1}AT$$

- Then $By = \lambda y \rightarrow T^{-1}AT\,y = \lambda y \rightarrow A(Ty) = \lambda(Ty)$

so that A **and** B **have the same eigenvalues**, and if y is an eigenvector of B, then $x = Ty$ is an eigenvector of A

Similarity Transformations

- Similarity transformations:
  - Preserve eigenvalues
  - Do not preserve eigenvectors
  - But the eigenvectors are still easily recovered

- Note that the converse is not true
  - two matrices that are **similar** must have the **same eigenvalues**
  - ***but*** two matrices that have the same eigenvalues are ***not necessarily similar***

Similarity Transformations

- The **eigenvalues of a diagonal matrix are its diagonal entries**, and the eigenvectors are the corresponding columns of the identity matrix I
- Note that:
  - Diagonal form simplifies eigenvalue problems for general matrices by similarity transformations
  - But **some matrices cannot be transformed into diagonal form** by a similarity transformation
- Fortunately:
  - **every matrix** can be transformed into **triangular** by a similarity transformation
  - The **eigenvalues of a triangular matrix are also the diagonal entries**

Computing Eigenvalues and Eigenvectors

- There are *several methods* designed to compute all of the *eigenvalues* of a matrix and require a great deal of work

- In practice, one may need ***only one or a few eigenvalues*** and corresponding eigenvectors

- The simplest method for computing a single eigenvalue and eigenvector of a matrix is the **power method**, which takes ***successively*** *higher powers of the matrix* times *an initial starting vector*

Power Method

- Assume that the matrix has a **unique eigenvalue λ_1 of maximum modulus**, with corresponding eigenvector u_1
- Let us consider the following ***iteration scheme***, starting from a given nonzero vector x_0

$$x_k = Ax_{k\text{-}1}$$

- The iteration scheme **converges** to a multiple of u_1, the eigenvector corresponding to the dominant eigenvalue λ_1

Power Method

- In fact, if we express the starting vector x_0 as a linear combination, $x_0 = \sum_{i=1}^{n} \alpha_i u_i$ where u_i are eigenvectors of A

Power Method

- In fact, if we express the starting vector x_0 as a linear combination, $x_0 = \sum_{i=1}^{n} \alpha_i u_i$ where u_i are eigenvectors of A, then

$$x_k = Ax_{k-1} = A^2 x_{k-2} = \cdots = A^k x_0 = A^k \sum_{i=1}^{n} \alpha_i u_i =$$

$$= \sum_{i=1}^{n} \alpha_i A^k u_i = \sum_{i=1}^{n} \lambda_i^k \alpha_i u_i = \lambda_1^k (\alpha_1 u_1 + \sum_{i=2}^{n} (\lambda_i / \lambda_1)^k \alpha_i u_i)$$

Power Method

- In fact, if we express the starting vector x_0 as a linear combination, $x_0 = \sum_{i=1}^{n} \alpha_i u_i$ where u_i are eigenvectors of A, then

$$x_k = Ax_{k-1} = A^2 x_{k-2} = \cdots = A^k x_0 = A^k \sum_{i=1}^{n} \alpha_i u_i =$$

$$= \sum_{i=1}^{n} \alpha_i A^k u_i = \sum_{i=1}^{n} \lambda_i^k \alpha_i u_i = \lambda_1^k (\alpha_1 u_1 + \sum_{i=2}^{n} (\lambda_i / \lambda_1)^k \alpha_i u_i)$$

- Since $|\lambda_i / \lambda_1| < 1$ for $i > 1$, successively higher powers go to zero, leaving only the component corresponding to u_1

Power Method

- Then, having $$x_k = \sum_{i=1}^{n} \lambda_i^k \alpha_i u_i = \lambda_1^k (\alpha_1 u_1 + \sum_{i=2}^{n} (\lambda_i / \lambda_1)^k \alpha_i u_i)$$

- We can obtain

$$\frac{\|x_{k+1}\|}{\|x_k\|} = \frac{\|Ax_k\|}{\|x_k\|} \rightarrow \frac{\|\alpha_1 \lambda_1^{k+1} u_1\|}{\|\alpha_1 \lambda_1^k u_1\|} = \frac{\alpha_1 \lambda_1^{k+1} \|u_1\|}{\alpha_1 \lambda_1^k \|u_1\|} = |\lambda_1| \text{ as } n \rightarrow \infty$$

Power Method

- To avoid eventual overflow (or underflow if the dominant eigenvalue is less than 1 in magnitude), it is better to **normalize** the approximate eigenvector at each iteration
- We can require its largest component to have modulus 1

- This step gives the **iteration scheme**

$$y_k = Ax_{k-1}$$

$$x_k = y_k / \|y_k\|_\infty = Ax_{k-1} / \|Ax_{k-1}\|_\infty$$

- With this normalization $\|y_k\|_\infty \to |\lambda_1|$ and $x_k \to u_1 / \|u_1\|_\infty$

Deflation Methods

- Suppose that an eigenvalue λ_1 and corresponding eigenvector x_1 for a matrix A have been computed
- We can **compute additional eigenvalues** $\lambda_2, \ldots, \lambda_n$ of *A*, by a process called **deflation**, which removes the known eigenvalue
- Construct a new matrix *B* with eigenvalues $\lambda_2, \ldots, \lambda_n$ that is *deflate* the matrix A, removing λ_1
- Then λ_2 can be obtained by the power method

Deflation Methods

- Let H be any nonsingular matrix such that $Hx_1 = \alpha e_1$, a scalar multiple of the first column of the identity matrix I

- Then the **similarity transformation** determined by H transforms A into the form

$$HAH^{-1} = \begin{bmatrix} \lambda_1 & b^T \\ 0 & B \end{bmatrix}$$

where B is a matrix of order n - 1 having eigenvalues λ_2, …, λ_n

- For example, good choice for H can be an appropriate Householder transformation

Deflation Methods

- We use B to compute next eigenvalue λ_2 and eigenvector y_2
- Given y_2 eigenvector of B, we want to compute the second eigenvector x_2 of matrix A

Deflation Methods

- We use B to compute next eigenvalue λ_2 and eigenvector y_2
- Given y_2 eigenvector of B, we want to compute the second eigenvector x_2 of matrix A
- We need to add an element to vector y_2 (that consist of n-1 elements), that is $x_2 = (s_2\ y_2)$
- s_2 can be the element α such that

$$x_2 = H^{-1}\begin{bmatrix}\alpha \\ y_2\end{bmatrix} \quad \text{where} \quad \alpha = \frac{b^T y_2}{\lambda_2 - \lambda_1}$$

- Hence, x_2 is an eigenvector corresponding to λ_2 for the original matrix A, provided $\lambda_2 \neq \lambda_1$
- Process can be repeated to find additional eigenvalues and eigenvectors

Deflation Methods

Hotelling deflation

- A given, λ_1 and u_1 known (e.g., by power method)
- Consider $B = A - \lambda_1 u_1 u_1^T$
- We can verify that B has the same eigenvectors as **A**, and the same eigenvalues as **A** except that the largest one has been replaced by 0
- In fact

$$(A - \lambda_1 u_1 u_1^T) u_j = A u_j - \lambda_1 u_1 u_1^T u_j = \lambda_j u_j - \lambda_1 u_1 u_1^T u_j$$

$$\text{If } j = 1 \quad \text{then} \quad (A - \lambda_1 u_1 u_1^T) u_1 = \lambda_1 u_1 - \lambda_1 u_1 (u_1^T u_1) = 0 u_1$$

$$\text{If } j \neq 1 \quad \text{then} \quad (A - \lambda_1 u_1 u_1^T) u_j = \lambda_j u_j - \lambda_1 u_1 (0) = \lambda_j u_j$$

- Then we compute the value of λ_2 by the power method

Deflation Methods

An **alternative approach** is the following:

- A given, λ_1 and x_1 known (e.g., by power method)
- Denote by a^T the first row of A (or the p-th row of A), i.e., $a^T=(a_{11}, a_{12}, \ldots, a_{1n})$
- Consider $B = A - x_1^* a^T$
- Where x_1* is the vector x_1 normalized by dividing by its first element (and x_1* a^T is an nxn matrix)
- We can verify that $B\, x_1=0$ that is 0 is eigenvalue of B (instead of λ_1) and $\lambda_2, \ldots, \lambda_n$ are still eigenvalues (of A)
- Then we compute the value of λ_2 by the power method

Deflation Methods

A variant of the previous **alternative approach** is:

- A given, λ_1 and x_1 known (e.g., by power method)
- Denote by v_1 any vector such that $v_1^T x_1 = \lambda_1$
- Then the matrix $A - x_1 v_1^T$ has eigenvalues of $0, \lambda_2, \ldots, \lambda_n$
- Then we compute the value of λ_2 by the power method
- There are several possible choices for v_1

Smallest eigenvalue

- For some applications, the **smallest eigenvalue** of a matrix is required rather than the largest
- We use the fact that the *eigenvalues of A^{-1} are the reciprocals of those of A*
- Hence the smallest eigenvalue of A is the reciprocal of the largest eigenvalue of A^{-1}
- In fact

$$\min_{i=1,\ldots,n} \left|\lambda_i(A)\right| = \min_{i=1,\ldots,n} \left|\frac{1}{\lambda_i(A^{-1})}\right| = \frac{1}{\max_{i=1,\ldots,n} \left|\lambda_i(A^{-1})\right|}$$

- We therefore use the inverse iteration scheme

$$Ay_k = x_{k-1} \Rightarrow y_k = A^{-1}x_{k-1}$$

Convergence

- The **convergence rate** of the power method depends on the ratio $|\lambda_2|/|\lambda_1|$, where λ_2 is the eigenvalue having second-largest modulus

- **The smaller $|\lambda_2|/|\lambda_1|$, the faster the convergence**

- Hence the power method will converge
  - Quickly if $|\lambda_2|/|\lambda_1|$ is small
  - Slowly if $|\lambda_2|/|\lambda_1|$ is close to 1

EXAMPLES

Gould Index of Accessibilty

- https://www.math.washington.edu/~morrow/336_11/papers/leo.pdf
- http://matrixapps.blogspot.it/2010/07/gould-index-matrix-application-to.html

- The method proposed by Peter Gould (1967), also known as eigenvector centrality, is one method of computing the *centrality*, or *approximate importance*, of each node in a graph

- The assumption is that each node's centrality is the sum of the centrality values of the nodes that it is connected to

Gould Index of Accessibilty

- Let us look at the method

- We begin with the adjacency matrix A of the graph
- It is usual to define the entries a_{ii}, the diagonal, as 0
- We replace the diagonal zeros with ones

- The index that Gould uses the (*normalized*) **eigenvector from the principle eigenvalue** of the **modified adjacency matrix $B=A+I$**

- The i-th entry corresponds to the i-th vertex and this is its **accessibility rank**

Gould Index of Accessibilty

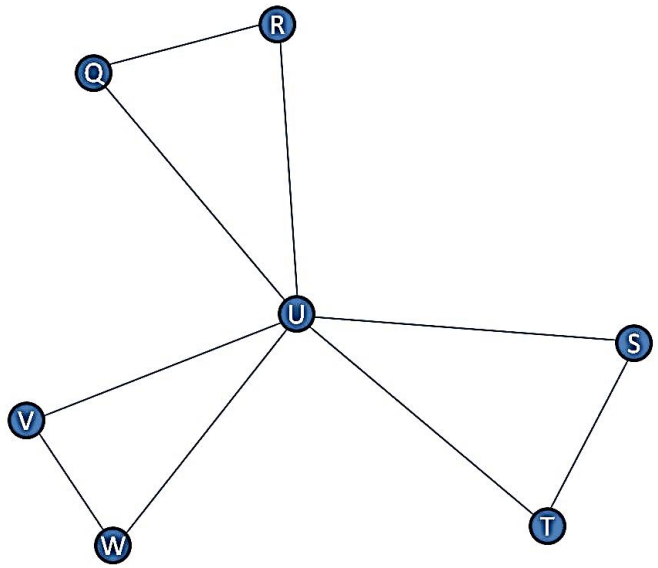
- Consider a graph that represents a set of towns (the vertices) and the travel routes between those towns (the edges)
- Historical geographers were interested in which town would become the trade center for this region

Gould Index of Accessibilty

- Consider a graph that represents a set of towns (the vertices) and the travel routes between those towns (the edges)
- Historical geographers were interested in which town would become the trade center for this region
- Make the adjacency matrix for the graph and place a 1 in each diagonal position

Gould Index of Accessibilty

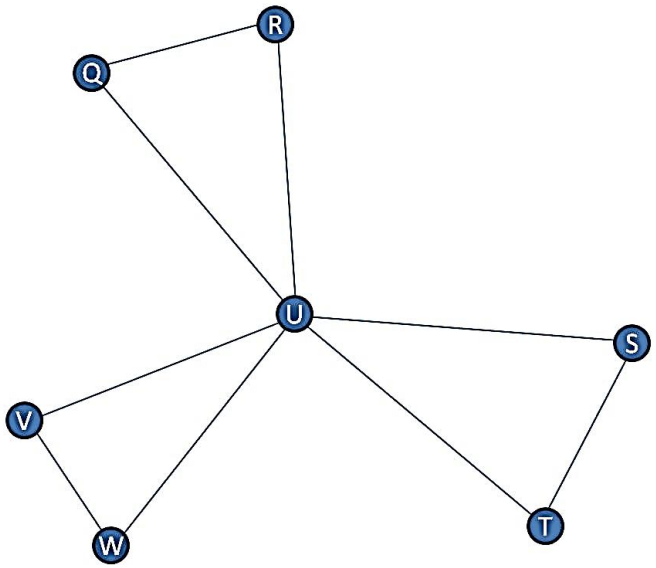
- Consider a graph that represents a set of towns (the vertices) and the travel routes between those towns (the edges)
- Historical geographers were interested in which town would become the trade center for this region
- Make the adjacency matrix for the graph and place a 1 in each diagonal position



	Q	R	S	T	U	V	W	
	1	1	0	0	1	0	0	Q
	1	1	0	0	1	0	0	R
	0	0	1	1	1	0	0	S
	0	0	1	1	1	0	0	T
	1	1	1	1	1	1	1	U
	0	0	0	0	1	1	1	V
	0	0	0	0	1	1	1	W

Gould Index of Accessibilty

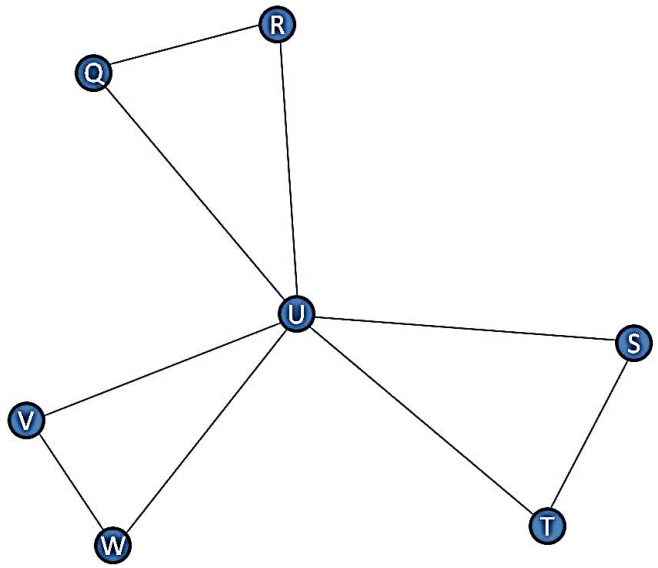
- Find the **largest eigenvalue** of the matrix
- The eigenvalues are: (2, 0, 4, -1, 0, 2, 0),
- The third eigenvalues has the largest absolute value, **4**



	Q	R	S	T	U	V	W
Q	1	1	0	0	1	0	0
R	1	1	0	0	1	0	0
S	0	0	1	1	1	0	0
T	0	0	1	1	1	0	0
U	1	1	1	1	1	1	1
V	0	0	0	0	1	1	1
W	0	0	0	0	1	1	1

Gould Index of Accessibilty

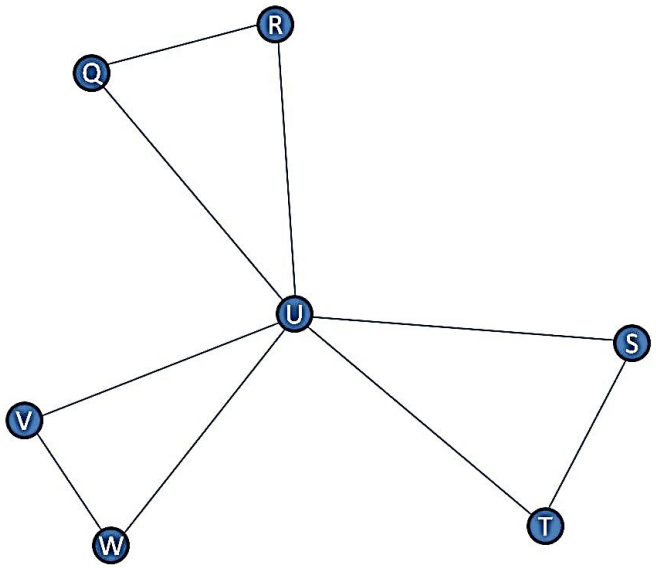
- Find the **largest eigenvalue** of the matrix
- The eigenvalues are: (2, 0, 4, -1, 0, 2, 0),
- The third eigenvalues has the largest absolute value, **4**
- Find the **eigenvector associated** with the eigenvalue of 4: (0.3162, 0.3162, 0.3162, 0.3162, 0.6325, 0.3162, 0.3162)



	Q	R	S	T	U	V	W	
	1	1	0	0	1	0	0	Q
	1	1	0	0	1	0	0	R
	0	0	1	1	1	0	0	S
	0	0	1	1	1	0	0	T
	1	1	1	1	1	1	1	U
	0	0	0	0	1	1	1	V
	0	0	0	0	1	1	1	W

Gould Index of Accessibilty

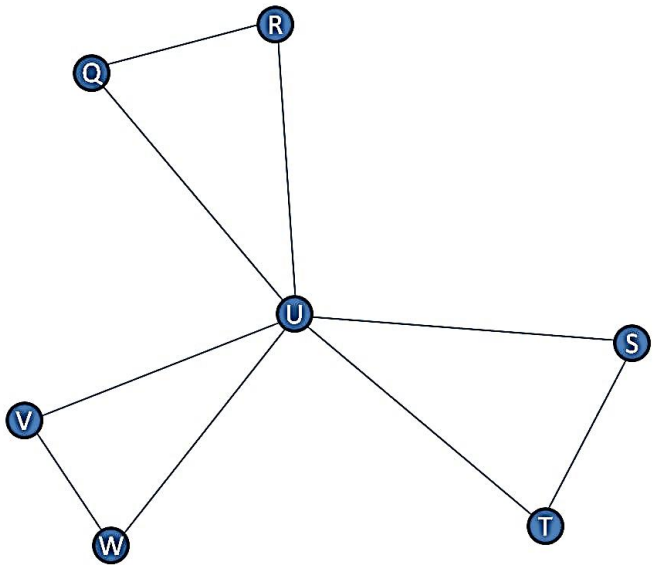
- You can normalize this vector by dividing by the sum of the entries, 2.5297. You get:
  (Q, R, S, T, U, V, W) = (0.125, 0.125, 0.125, 0.125, 0.25, 1.25, 1.25)



	Q	R	S	T	U	V	W	
	1	1	0	0	1	0	0	Q
	1	1	0	0	1	0	0	R
	0	0	1	1	1	0	0	S
	0	0	1	1	1	0	0	T
	1	1	1	1	1	1	1	U
	0	0	0	0	1	1	1	V
	0	0	0	0	1	1	1	W

Gould Index of Accessibilty

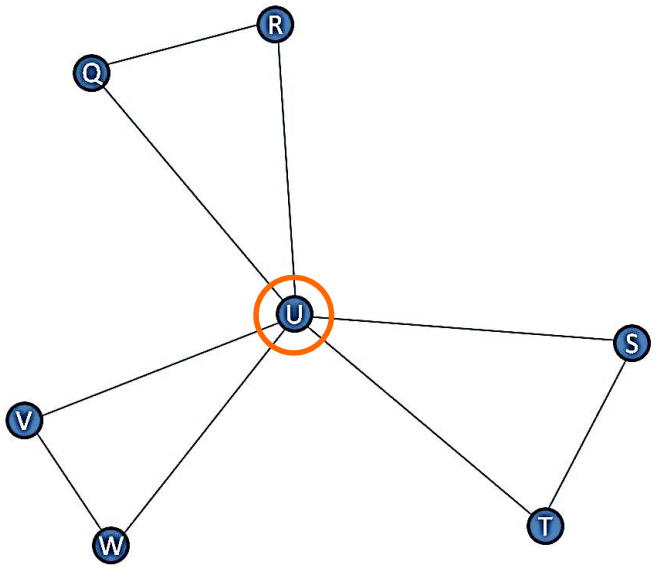
- You can normalize this vector by dividing by the sum of the entries, 2.5297. You get:
  (Q, R, S, T, U, V, W) = (0.125, 0.125, 0.125, 0.125, 0.25, 1.25, 1.25)
- These are the **Gould indices of each of the vertices**
- They describe **how strongly each vertex is connected to the other vertices**



	Q	R	S	T	U	V	W	
	1	1	0	0	1	0	0	Q
	1	1	0	0	1	0	0	R
	0	0	1	1	1	0	0	S
	0	0	1	1	1	0	0	T
	1	1	1	1	1	1	1	U
	0	0	0	0	1	1	1	V
	0	0	0	0	1	1	1	W

Gould Index of Accessibilty

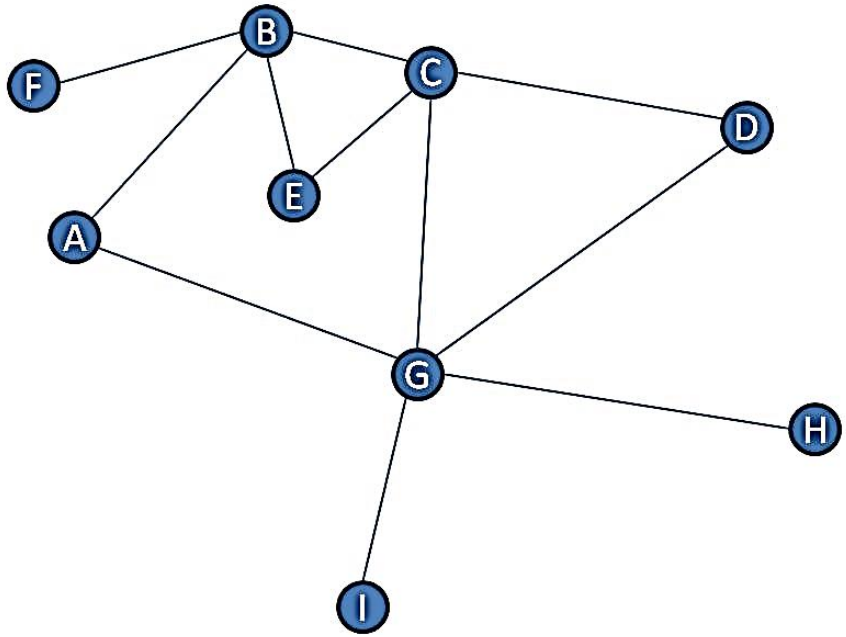
- You can normalize this vector by dividing by the sum of the entries, 2.5297. You get:
  (Q, R, S, T, U, V, W) = (0.125, 0.125, 0.125, 0.125, 0.25, 1.25, 1.25)
- These are the **Gould indices of each of the vertices**
- They describe **how strongly each vertex is connected to the other vertices**



	Q	R	S	T	U	V	W	
	1	1	0	0	1	0	0	Q
	1	1	0	0	1	0	0	R
	0	0	1	1	1	0	0	S
	0	0	1	1	1	0	0	T
	1	1	1	1	1	1	1	U
	0	0	0	0	1	1	1	V
	0	0	0	0	1	1	1	W

Gould Index of Accessibilty

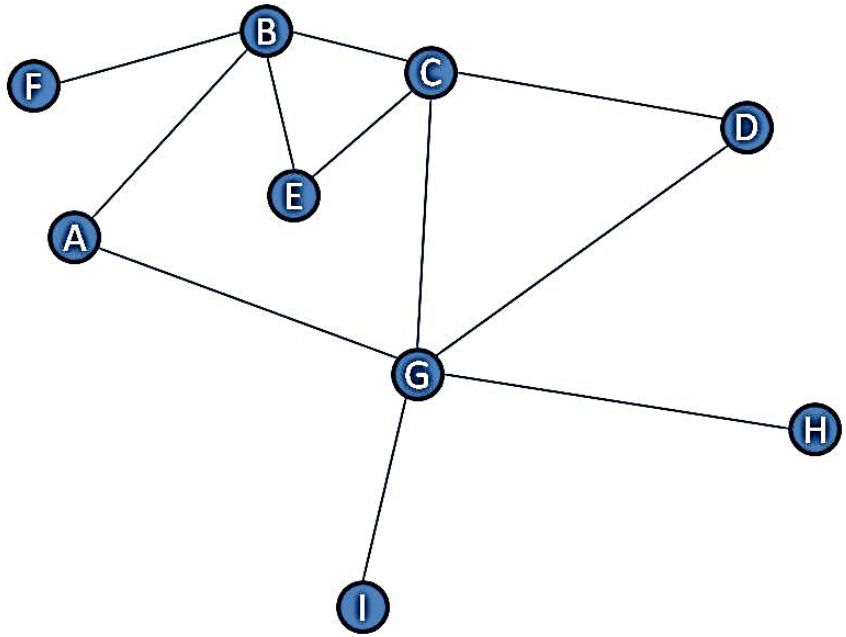
- Now consider a more realistic graph
- In this case, would **G** be the trade center, or would it be **C**?



Gould Index of Accessibilty

- Now consider a more realistic graph
- In this case, would G be the trade center, or would it be C?

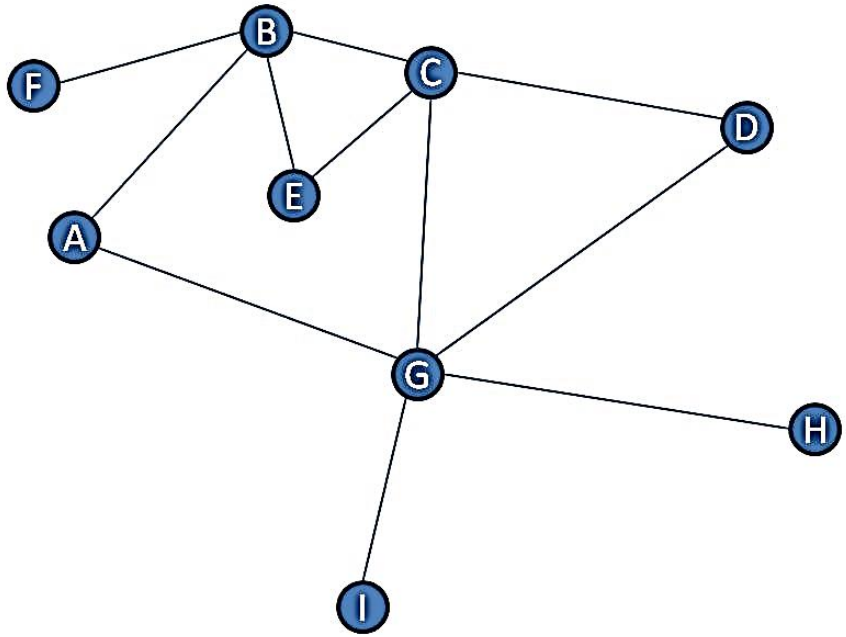
- The eigenvalues are:
  (4.01, -1.37, 1.71, 1, -0.37, -0.56, 2.58, 1, 1)



Gould Index of Accessibilty

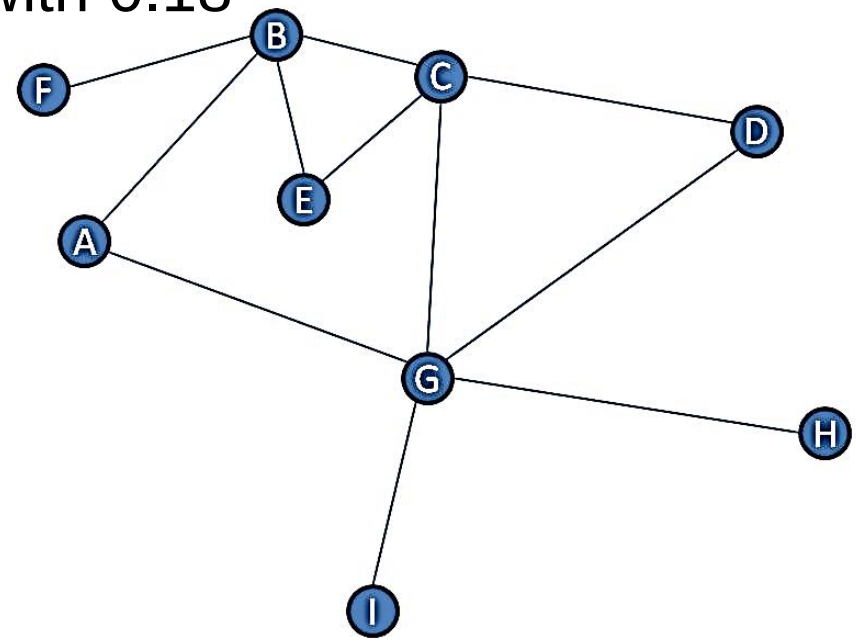
- The eigenvector associated with the largest eigenvalue is (0.2941, 0.4097, 0.5023, 0.3249, 0.3026, 0.1359, 0.4770, 0.1583, 0.1583)

- Normalizing we have (A, B, C, D, E, F, G, H, I) = (0.1064, 0.1482, 0.1818, 0.1176, 0.1095, 0.0492, 0.1726, 0.0573, 0.0573)



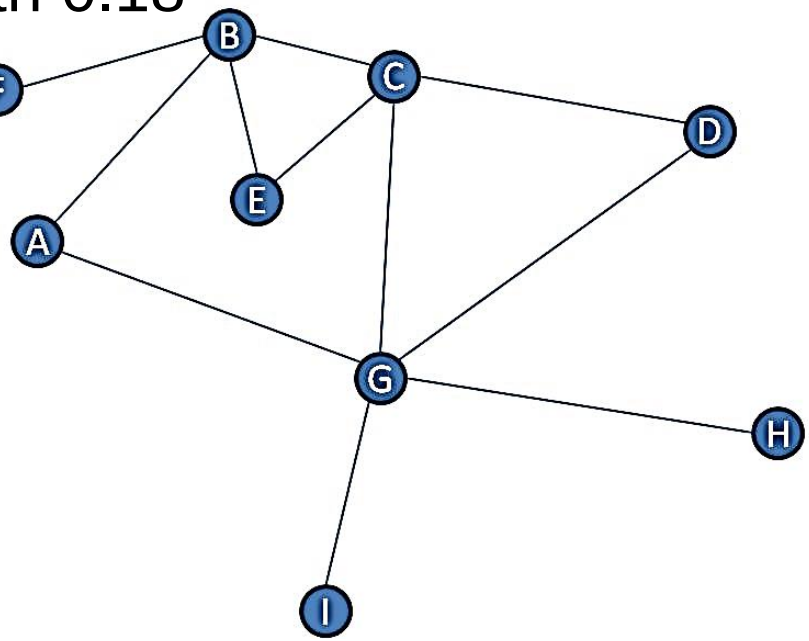
Gould Index of Accessibilty

- The eigenvector associated with the largest eigenvalue is (0.2941, 0.4097, 0.5023, 0.3249, 0.3026, 0.1359, 0.4770, 0.1583, 0.1583)
- Normalizing we have (A, B, C, D, E, F, G, H, I) = (0.1064, 0.1482, 0.1818, 0.1176, 0.1095, 0.0492, 0.1726, 0.0573, 0.0573)
- **C** has the largest Gould Index with 0.18
- **G** comes second with 0.17



Gould Index of Accessibilty

- The eigenvector associated with the largest eigenvalue is (0.2941, 0.4097, 0.5023, 0.3249, 0.3026, 0.1359, 0.4770, 0.1583, 0.1583)
- Normalizing we have (A, B, C, D, E, F, G, H, I) = (0.1064, 0.1482, 0.1818, 0.1176, 0.1095, 0.0492, 0.1726, 0.0573, 0.0573)
- **C** has the largest Gould Index with 0.18
- **G** comes second with 0.17
- Note that **I** and **H** have the same smallest index, as expected



Algebraic Connectivity of graphs

- The importance of the **algebraic connectivity** of a graph is due to the fact that it is a good parameter to measure, to a certain extent, how well a graph is connected

- The algebraic connectivity is for
  - application on trees,
  - application on hard problems in graph theory (the expanding properties of graphs, weighted graphs, absolute algebraic connectivity, isoperimetric number, genus and other invariants of a graph)
  - the study of the asymptotic behavior for random graphs;
  - applications on combinatorial optimization problems (the maximum cut problem and the traveling salesman problem)
  - the theory of elasticity
  - the correspondence between continuous and discrete mathematics

Algebraic Connectivity of graphs

- Based on theory of **Fiedler** (1970s)

- We define the Laplacian matrix **L(G)** of the graph G(N,E) as:
  - L(G) (i,i) = degree of node I (number of incident edges)
  - L(G) (i,j) = -1 if i != j and there is an edge (i,j)
  - L(G) (i,j) = 0 otherwise

- We have **L(G)=D-A** where *D is the diagonal matrix* of node degrees and *A is the adjacency matrix* of graph G(N,E)

Algebraic Connectivity of Graphs

- L(G) is **symmetric:**
  - the eigenvalues of L(G) are **real**
  - the eigenvectors are **real** and **orthogonal**

- Further, the **eigenvalues** of L(G) are **nonnegative**:
  - $0 = \lambda_1 <= \lambda_2 <= \ldots <= \lambda_n$

- The number of connected components of G is equal to the **number of λ_i equal to 0**

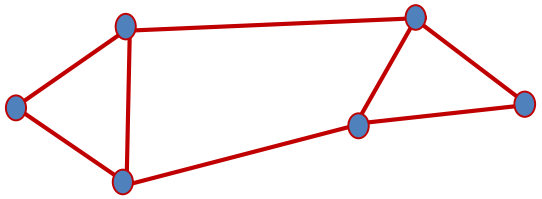
- G is connected **if and only if λ_2 != 0**

Algebraic Connectivity of Graphs

- The **second smallest eigenvalue** λ_2 of L(G) is called **algebraic connectivity** of G
- The eigenvector associated with the algebraic connectivity has been named the ***Fiedler vector***
- The Fiedler vector can be used to partition a graph

Graph Partitioning

Spectral Bi-partitioning Algorithm



- Build the Laplacian matrix L of the graph
- Find the second smallest eigenvalue and the corresponding eigenvector
- Map vertices to corresponding components of the Fiedler vector
- Grouping
  - Sort components of reduced 1-dimensional vector
  - Identify clusters by splitting the sorted vector in two

Graph Partitioning

How to choose a splitting point?

- Split at 0, that is positive and negative values
- Mean value
- Median value

- Partitioning a graph into *k* clusters can be done by **Recursive bi-partitioning** (Hagen et al.,'91)

- Recursively apply bi-partitioning algorithm in a hierarchical divisive manner
- Example: Image Segmentation
  - Uses 2nd (smallest) eigenvector to define optimal cut
  - Recursively generates two clusters with each cut

Markov chain

- http://webspace.ship.edu/deensley/m318/ppt/Section_49.pdf
- A vector with nonnegative entries that add up to 1 is called a **probability vector**
- A **stochastic matrix** is a square matrix whose columns are probability vectors
- A **Markov chain** is a sequence of probability vectors x_0, x_1, … together with a stochastic matrix P, such that

$$x_1 = Px_0 \qquad x_2 = Px_1 \qquad x_3 = Px_2 \qquad \ldots$$

Markov chain

Example

- Consider the following model of population movement between a city and the suburbs:
  - each year 5% of city dwellers move the suburbs and
  - 3% of suburbanites move to the city

- If in 2001 58.2% of the population lived in the city and 41.8% lived in the suburbs, what is the ***population distribution 20 years later***?

Markov chain

Example

- Consider matrix $M = \begin{pmatrix} 0.95 & 0.03 \\ 0.05 & 0.97 \end{pmatrix}$ and vector $x_0 = \begin{pmatrix} 0.582 \\ 0.418 \end{pmatrix}$

  to represent the population distribution in 2001

- In this case $x_1 = Mx_0 = \begin{pmatrix} 0.565 \\ 0.435 \end{pmatrix}$ gives the population distribution in 2002
- In general $x_n = M^n x_0$ **gives the population distribution in *n*** years after 2001

Markov chain

- The difference between x_i and x_{i+1} gets smaller every step

- Let P be stochastic matrix
- A **steady-state vector** (also called **equilibrium vector**) is a probability vector x such that $\boldsymbol{Px=x}$
- If state x is achieved, the system stays there

- **A nonzero steady-state vector** is in fact an **eigenvector of eigenvalue 1 of P** (since it satisfies $Px=x$)

Markov chain

- In our example, the **eigenvalues** are $\lambda_1 = 1$ and $\lambda_2 = 0.92$, and the corresponding **eigenvectors** are

$$v_1 = \begin{pmatrix} -0.514496 \\ -0.857493 \end{pmatrix} \qquad v_2 = \begin{pmatrix} -0.707107 \\ 0.707107 \end{pmatrix}$$

- So we form matrix P using eigenvectors and matrix D using eigenvalues

$$P = \begin{pmatrix} 0.514496 & -0.707107 \\ -0.85749 & 0.707107 \end{pmatrix} \qquad D = \begin{pmatrix} 1 & 0 \\ 0 & 0.92 \end{pmatrix}$$

Markov chain

- We can apply the diagonalization and obtain $M = PDP^{-1}$
- Now $M \cdot M = PDP^{-1}PDP^{-1} = PD^2P^{-1}$
- And, in general $M^k = PD^kP^{-1}$
- **Having** $M^k = P\begin{pmatrix} 1 & 0 \\ 0 & (0.92)^k \end{pmatrix}P^{-1}$
- **It follows** $\lim_{k\to\infty} M^k = P\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}P^{-1} = \begin{pmatrix} 0.375 & 0.375 \\ 0.625 & 0.625 \end{pmatrix}$

Markov chain

- For any initial stochastic vector x we will have

$$\lim_{k\to\infty} M^k x = \begin{pmatrix} 0.375 & 0.375 \\ 0.625 & 0.625 \end{pmatrix} x =$$

$$\begin{pmatrix} 0.375x_1 + 0.375x_2 \\ 0.625x_1 + 0.625x_2 \end{pmatrix} = \begin{pmatrix} 0.375 \\ 0.625 \end{pmatrix}$$