NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

Module 8: Eigenanalysis

Learning Objectives

- Learn the concepts of eigenvalues and eigenvectors
- Learn power iteration methods
- Learn (unshifted) QR algorithm
- Understand how PageRank works in search engines

Sources

- Textbook (Chapter 12: Eigenvalues and Singular Values)

- K.Bryan and T.Leise. “The $25,000,000,000 eigenvector: The linear algebra behind Google”. *SIAM Review* 48(3): 569-581, 2006.
- *https://www.rose-hulman.edu/~bryan/googleFinalVersionFixed.pdf*

Outline

Introduction

- Eigenvalues and eigenvectors are fundamental concepts of linear algebra and they play powerful roles in many applications, such as
  - Principal components analysis (PCA)
  - Correspondence analysis
  - Machine learning
  - Spectral analysis
  - Geometry problems
  - …
- There is need to compute eigenvalues and eigenvectors.

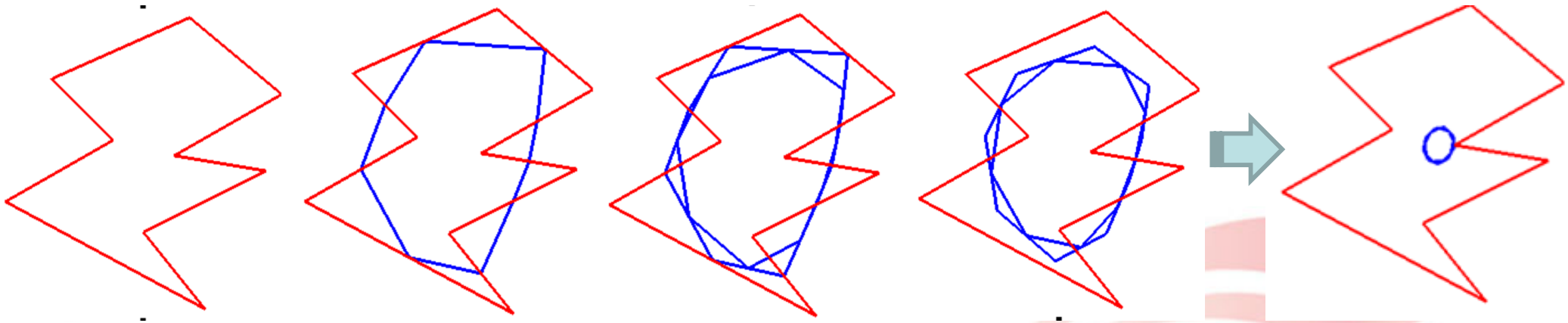
Application example: mid-point algorithm

- **The Midpoint algorithm:**
  - We are given a closed planar polygon with n vertices $p_1, \cdots, p_n \in \mathbf{R}^2$.
  - Let

$$q_i = \frac{p_i + p_{i+1}}{2}, \quad i = 1, \cdots, n$$

be the edge midpoint of the polygon. Here indices are used cyclically with respect to $1, \cdots, n$ so that $p_{n+1} = p_1$ and $p_0 = p_n$. This results in a new polygon with n vertices $q_1, \cdots, q_n$.

  - Iterating this process yields a sequence of polygons that "shrink" to a point, called the *barycentre* of the initial polygon: $\bar{p} = \sum_{i=1}^{n} p_i/n$.



Application example: mid-point algorithm

- **Analysis:**
  - Rewrite the Midpoint algorithm in matrix/vector notation

$$\underbrace{\begin{pmatrix} q_1 \\ q_2 \\ \vdots \\ q_{n-1} \\ q_n \end{pmatrix}}_{Q \in \mathbf{R}^{n\times 2}} = \underbrace{\frac{1}{2}\begin{pmatrix} 1 & 1 & 0 & \cdots & 0 \\ 0 & 1 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 1 \\ 1 & 0 & \cdots & 0 & 1 \end{pmatrix}}_{M_n \in \mathbf{R}^{n\times n}} \underbrace{\begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_{n-1} \\ p_n \end{pmatrix}}_{P \in \mathbf{R}^{n\times 2}}$$

  where M_n is the **averaging matrix**.

  - Iterating the averaging process can be understood as repeatedly multiplying P with M_n, so that the polygon after k iterations is

$$P^{(k)} = \underbrace{M_n M_n \cdots M_n}_{k \text{ times}} P = (M_n)^k P$$

Application example: mid-point algorithm

- The "limit" polygon is

$$P^{(\infty)} = \lim_{k\to\infty} P^{(k)} = \left(\lim_{k\to\infty} (M_n)^k\right) P = \frac{(1,\cdots,1)P}{n}$$

via eigenanalysis

Need of numerical methods

- It is possible to compute eigenvalues and eigenvectors analytically for very small matrices.
- For large matrices, eigenanalysis requires numerical methods.
- Moreover, we need good numerical methods.
  - A simple example is to find the eigenvalues of the matrix

$$A = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 2 & & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 20 \end{bmatrix}.$$

We will calculate the characteristic polynomial, which is a Wilkinson polynomial $P(\lambda) = (\lambda - 1)(\lambda - 2) \cdots (\lambda - 20)$, and use a rootfinder to find the roots. However, some of the roots of the machine version of $P(\lambda)$ are far from the roots of the true version of $P(\lambda)$.

Outline

Eigenvalues and eigenvectors

- **Definition:** Let A be an $n \times n$ matrix. If there exist a number λ and a non-zero n-dimensional vector x such that

$$Ax = \lambda x,$$

then

  - λ is called the eigenvalue of A;
  - x is the corresponding eigenvector.

- **Example:** Since

$$\begin{bmatrix} 1 & 3 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = 4 \begin{bmatrix} 1 \\ 1 \end{bmatrix},$$

matrix $\begin{bmatrix} 1 & 3 \\ 2 & 2 \end{bmatrix}$ has an eigenvector $\begin{bmatrix} 1 \\ 1 \end{bmatrix}$ with the corresponding eigenvalue 4.

Computation of eigenvalues & eigenvectors

- **Characteristic polynomial:**
  - Eigenvalues are roots λ of the polynomial $\det(\lambda I - A)$, which is called the characteristic polynomial.
  - For eigenvalue λ, $(\lambda I - A)x = 0$ has non-trivial solution(s), which are the eigenvectors corresponding to λ.
- **Example:** For $A = \begin{bmatrix} 1 & 3 \\ 2 & 2 \end{bmatrix}$, its characteristic polynomial is

$$\det(\lambda I - A) = \det \begin{vmatrix} \lambda - 1 & -3 \\ -2 & \lambda - 2 \end{vmatrix} = (\lambda - 1)(\lambda - 2) - 6 = (\lambda - 4)(\lambda + 1)$$

So eigenvalues are $\lambda = 4, -1$.

$$\lambda = 4: \quad \begin{bmatrix} 3 & -3 \\ -2 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0 \implies \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\lambda = -1: \quad \begin{bmatrix} -2 & -3 \\ -2 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0 \implies \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3 \\ -2 \end{bmatrix}$$

Diagonalization

- If a matrix A has eigenvectors that form a basis for $\mathbf{R}^n$, then A can be diagonalizable.

  - Assume $Ax_i = \lambda_i x_i$ for $i = 1, \cdots, n$.

  - $A[x_1, \cdots, x_n] = [x_1, \cdots, x_n] \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix}$

  - $A = [x_1, \cdots, x_n] \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_n \end{bmatrix} [x_1, \cdots, x_n]^{-1}$

- **Example:** For $A = \begin{bmatrix} 1 & 3 \\ 2 & 2 \end{bmatrix}$,

  we have $\begin{bmatrix} 1 & 3 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} 1 & 3 \\ 1 & -2 \end{bmatrix} = \begin{bmatrix} 1 & 3 \\ 1 & -2 \end{bmatrix} \begin{bmatrix} 4 & 0 \\ 0 & -1 \end{bmatrix}$

  and $\begin{bmatrix} 1 & 3 \\ 2 & 2 \end{bmatrix} = \begin{bmatrix} 1 & 3 \\ 1 & -2 \end{bmatrix} \begin{bmatrix} 4 & 0 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} 1 & 3 \\ 1 & -2 \end{bmatrix}^{-1}$.

Eigenanalysis for "midpoint algorithm"

Outline of the analysis:

- Averaging matrix M_n has n eigenvalues satisfying $\lambda_1 = 1 > |\lambda_2| \geq \cdots \geq |\lambda_n| > 0$.They correspond to eigenvectors $v_1, \cdots, v_n$ with $v_1 = (1, \cdots, 1)^T$.

- All these eigenvectors v_i form a basis. Thus any P can be expressed as

$$P = c_1 v_1 + c_2 v_2 + \cdots + c_n v_n$$

  with some coefficients $c_1, \cdots, c_n$.

- It can be proven that $c_1 = \sum_{i=1}^{n} p_i/n$.

- Multiplying P with M_n by k times:

$$P^{(k)} = (M_n)^k x = c_1 v_1 + \lambda_2^k c_2 v_2 + \cdots + \lambda_n^k c_n v_n.$$

- As $k \to \infty$, $P^{(\infty)} = c_1 v_1 = c_1 \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} p_i/n \\ \vdots \\ \sum_{i=1}^{n} p_i/n \end{bmatrix}$.

Outline

3.1. Power Iteration

- Definition: Let A be an $n \times n$ matrix.
  - A **dominant eigenvalue** of A is an eigenvalue whose magnitude is greater than all other eigenvalues of A.
  - Its corresponding eigenvector is called a **dominant eigenvector**.
- Example. Consider matrix $A = \begin{bmatrix} 1 & 3 \\ 2 & 2 \end{bmatrix}$. Let us start with a random vector [-5,5]T, and repeatedly multiply matrix A to it:

$$x_1 = Ax_0 = \begin{bmatrix} 1 & 3 \\ 2 & 2 \end{bmatrix}\begin{bmatrix} -5 \\ 5 \end{bmatrix} = \begin{bmatrix} 10 \\ 0 \end{bmatrix}$$

$$x_2 = A^2x_0 = \begin{bmatrix} 1 & 3 \\ 2 & 2 \end{bmatrix}\begin{bmatrix} 10 \\ 0 \end{bmatrix} = \begin{bmatrix} 10 \\ 20 \end{bmatrix}$$

$$x_3 = A^3x_0 = \begin{bmatrix} 1 & 3 \\ 2 & 2 \end{bmatrix}\begin{bmatrix} 10 \\ 20 \end{bmatrix} = \begin{bmatrix} 70 \\ 60 \end{bmatrix}$$

$$x_4 = A^4x_0 = \begin{bmatrix} 1 & 3 \\ 2 & 2 \end{bmatrix}\begin{bmatrix} 70 \\ 60 \end{bmatrix} = \begin{bmatrix} 250 \\ 260 \end{bmatrix} = 260\begin{bmatrix} \frac{25}{26} \\ 1 \end{bmatrix}$$

Close to the dominant eigenvector.

Power Iteration

- Next, how to find the dominant eigenvalue?
  - Consider the eigenvalue equation $x\lambda = Ax$, where x is an approximate eigenvector and λ is unknown.
  - We appeal to least squares. The normal equations give the least squares solution (called the Rayleigh quotient):

$$\lambda = \frac{x^T Ax}{x^T x}$$

- Power iteration algorithm

Power Iteration

Given initial vector x_0.

for $j = 1, 2, 3, \ldots$

Normalize vector → $u_{j-1} = x_{j-1}/||x_{j-1}||_2$

Power step → $x_j = Au_{j-1}$

Rayleigh quotient → $\lambda_j = u_{j-1}^T Au_{j-1}$

end

$u_j = x_j/||x_j||_2$

3.2. Inverse Power Iteration

- If “Power Iteration” is applied to the inverse of the matrix, the **smallest eigenvalue** can be found.

- **Lemma:** Denote the eigenvalues of the $n \times n$ matrix A by $\lambda_1, \lambda_2, \ldots, \lambda_n$.
  - The eigenvalues of the **inverse** matrix A^{-1} are $\lambda_1^{-1}, \lambda_2^{-1}, \ldots, \lambda_n^{-1}$, assuming that the inverse exists. The eigenvectors are the same as those of A.
  - The eigenvalues of the **shifted** matrix $A - s\,I$ are by $\lambda_1 - s, \lambda_2 - s, \ldots, \lambda_n - s$. The eigenvectors are the same as those of A.

Inverse Power Iteration

- To find the smallest eigenvalue of A, we apply Power Iteration to A^{-1}.
  - To avoid calculating the inverse, we replace the iteration $x_{k+1} = A^{-1}x_k$ by solving $Ax_{k+1} = x_k$ by Gaussian elimination.
- In general, to find an eigenvalue nearest to number s,
  - apply Power Iteration to $(A\text{-}sI)^{-1}$ to get the largest magnitude eigenvalue b of $(A\text{-}sI)^{-1}$
  - The power iteration is done by Gaussian elimination on $(A\text{-}sI)x_{k+1}=x_k$
  - Then $\lambda = b^{-1}+s$ is the eigenvalue of A nearest to s.

Inverse Power Iteration

Given initial vector x_0 and shift s

for $j = 1, 2, 3, \ldots$

$u_{j-1} = x_{j-1}/||x_{j-1}||_2$

Solve $(A - sI)x_j = u_{j-1}$

$\lambda_j = u_{j-1}^T x_j$

end

$u_j = x_j/||x_j||_2$

3.3. Rayleigh quotient iteration

- Observations
  - The Rayleigh quotient can be used together with the Inverse Power Iteration to find one eigenvalue at a time.
  - It converges to the eigenvector associated to the eigenvalue with the smallest distance to the shift *s*.
  - The convergence is fast if the distance is small.
- Basic ideas
  - If at any step an approximate eigenvalue were known, it could be used as the shift *s*, to speed convergence.
  - Using the Rayleigh quotient as the updated shift in Inverse Power Iteration leads to Rayleigh Quotient Iteration (RQI).

Rayleigh quotient iteration

- RQI

Rayleigh Quotient Iteration

Given initial vector x_0.

for $j = 1, 2, 3, \ldots$

$u_{j-1} = x_{j-1}/||x_{j-1}||$

$\lambda_{j-1} = u_{j-1}^T A u_{j-1}$

Solve $(A - \lambda_{j-1} I)x_j = u_{j-1}$

end

$u_j = x_j/||x_j||_2$

- Results:
  - While Inverse Power Iteration converges linearly, RQI is in general quadratically convergent.
  - RQI will converge cubically for symmetric matrices.
- Question for you:
  - Compare the computational complexity of QRI and Inverse Power Iteration.

Outline

Problem

- Goal: to find all eigenvalues in parallel.
- Special case: Here we consider the QR algorithm for symmetric matrices. It can be extended to handle general matrices.
- For a symmetric matrix with real entries,
  - its eigenvalues are real numbers;
  - the set of its unit eigenvectors is an orthonormal set that forms a basis.

Normalized simultaneous iteration (NSI)

Given n orthogonal initial vectors $v_1, \cdots, v_n$, we apply one step of the Power Iteration to each of these vectors, which gives $Av_1, \cdots, Av_n$. The problem with this approach is that $Av_1, \cdots, Av_n$ are no longer guaranteed to be orthogonal. Actually they tend to converge to the dominant eigenvector under further multiplications by A.
One way to overcome the problem is to re-orthogonalize the vectors by QR decomposition. That is,

- We start with identity matrix I whose columns serve as the initial orthogonal vectors.
- The first step of the Power Iteration by QR decomposition gives

$$AI = \bar{Q}_1 R_1.$$

- The columns of $\bar{Q}_1$ form the new orthogonal set of unit vectors for the next step of Power Iteration.
- Repeat the above process: $A\bar{Q}_i = \bar{Q}_{i+1} R_{i+1}$.

Normalized simultaneous iteration (NSI)

At the jth step, the columns of $\bar{Q}_j$ are approximations to the eigenvectors of A, and the diagonal elements of R_j are approximations to the eigenvalues.

Normalized Simultaneous Iteration

Set $\overline{Q}_0 = I$

for $j = 1, 2, 3, \ldots$

$A\overline{Q}_j = \overline{Q}_{j+1} R_{j+1}$

end

(Unshifted) QR algorithm

The unshifted QR algorithm makes a slight change by replacing A by the current R_k.

- A is not needed after the first step.
- R_k is an upper triangular matrix which will require fewer operations than multiplication with A.

NSI

$$
\begin{aligned}
AI &= \bar{Q}_1 R_1 \\
A\bar{Q}_1 &= \bar{Q}_2 R_2 \\
A\bar{Q}_2 &= \bar{Q}_3 R_3 \\
A\bar{Q}_3 &= \bar{Q}_4 R_4 \\
&\vdots
\end{aligned}
$$

Unshifted QR Algorithm

$$
\begin{aligned}
A_0 \equiv AI &= Q_1 R'_1 \\
A_1 \equiv R'_1 Q_1 &= Q_2 R'_2 \\
A_2 \equiv R'_2 Q_2 &= Q_3 R'_3 \\
A_3 \equiv R'_3 Q_3 &= Q_4 R'_4 \\
&\vdots
\end{aligned}
$$

(Unshifted) QR algorithm

- From the unshifted QR algorithm, we have

$$A = Q_1 R_1' = Q_1 R_1' Q_1 Q_1^T = Q_1 A_1 Q_1^T.$$

- Similarly,

$$A = Q_1 A_1 Q_1^T = Q_1 Q_2 A_2 Q_2^T Q_1^T = \cdots = (Q_1 Q_2 \cdots Q_k) A_k (Q_k^T \cdots Q_2^T Q_1^T).$$

  Therefore, all A_j are similar, and thus have the same eigenvalues.

- If A is symmetric with eigenvalues satisfying $|\lambda| > |\lambda_2| > \cdots > |\lambda_n|$, then the sequence of A_k converges linearly to the diagonal matrix with the eigenvalues on the diagonal.

Unshifted QR Algorithm

$$\begin{array}{lcll} A_0 & \equiv & AI & = Q_1 R_1' \\ A_1 & \equiv & R_1' Q_1 & = Q_2 R_2' \\ A_2 & \equiv & R_2' Q_2 & = Q_3 R_3' \\ A_3 & \equiv & R_3' Q_3 & = Q_4 R_4' \\ & & \vdots & \end{array}$$

Outline

PageRank

- **Background:** When doing a web search with a search engine like Google, the engine will first find all pages that match the query terms and then display them in some order, with the most important pages on top.
- **Google's idea:** For ordering the pages, Google computes an *importance score* $x_k, k = 1, \cdots, n$, for all the n pages of the web (and there are several billion) usng the **PageRank** algorithm.
  - L_k: the set of page k's backlinks, that is,

$$L_k = \{j : \text{ page } j \text{ links to page } k\},$$

  - N_k: the number of pages that page k links to, that is, the number of outgoing links for page k.
  - Then the main idea of the importance score computed by PageRank is that each score x_k should satisfy

$$x_k = \sum_{j \in L_k} \frac{x_j}{N_j}, \quad k = 1, \cdots, n.$$

PageRank

- This can be reformulated as an eigenvector problem, because $x = (x_1, \cdots, x_n)^T$ is an eigenvector with eigenvalue $\lambda = 1$ of the matrix $A \in \mathbf{R}^{n \times n}$ with entries

$$A_{i,j} = \begin{cases} \frac{1}{N_j}, & \text{if page j links to page i} \\ 0, & \textit{otherwise} \end{cases}$$

i.e.,

$$Ax = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

As long as the web has no *dangling nodes*, that is, nodes with no outgoing links, then the matrix A is a *column-stochastic*, that is, all its entries are non-negative and all column sums are 1.

For column-stochastic matrices, $\lambda = 1$ is an eigenvalue.

PageRank

- However, there might be problems with this approach. For example, the web separates into r sub-webs that do not link to each other. To solve this problem, we consider the matrix

$$M = (1-\mu)A + \mu E, \qquad \mu \in [0,1],$$

a comvex combination of A and the matrix E with all entries equal to $1/n$.

Both E and M are column-stochastic, and if $\mu > 0$, then M is positive.

For such a postive and column-stochastic matrix, one can show that $\lambda = 1$ is the dominant eigenvalue. Moreover, if x is a positive vector with unit 1-norm, then so is Mx.

Consequently, the eigenvector with unit 1-norm for eigenvalue $\lambda = 1$, which Google uses to rank the web pages, can be found by using the Power Iteration method by iteratively computing

$$x_{k+1} = Mx_k,$$

with initial guess $x_0 = (1/n, \cdots, 1/n)$.

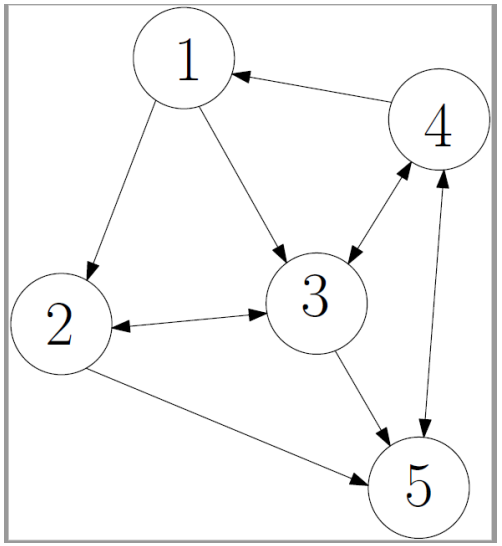
PageRank

- **Example.**

Look at the graph of web pages and links. The matrix is $A = \begin{bmatrix} 0 & 0 & 0 & 1/3 & 0 \\ 1/2 & 0 & 1/3 & 0 & 0 \\ 1/2 & 1/2 & 0 & 1/3 & 0 \\ 0 & 0 & 1/3 & 0 & 1 \\ 0 & 1/2 & 1/3 & 1/3 & 0 \end{bmatrix}$.

Solving

$$\begin{bmatrix} 0 & 0 & 0 & 1/3 & 0 \\ 1/2 & 0 & 1/3 & 0 & 0 \\ 1/2 & 1/2 & 0 & 1/3 & 0 \\ 0 & 0 & 1/3 & 0 & 1 \\ 0 & 1/2 & 1/3 & 1/3 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}$$



gives the eigenvector corresponding to $\lambda = 1$:

$$(x_1, x_2, x_3, x_4, x_5) = (0.1042, 0.125, 0.2187, 0.3125, 0.2396)$$

which are the page ranks (scores) of pages 1 to 5. Therefore, node 4 is the most influential Web page.

Outline

Homework

Let $x = (x_1, \cdots, x_n)$ and $y = (y_1, \cdots, y_n)$ be two vectors in $\mathbf{R}^n$ such that

$$\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} y_i = 0, \tag{1}$$

and let P be the closed polygon with n vertices $p_i = (x_i, y_i), i = 1, \cdots, n$. Clearly, the barycentre $\bar{p}$ of P is the origin, $\bar{p} = (0, 0)$.
Now consider the midpoint algorithm *with normalization*, which starts with the polygon $P^{(0)} = P$ and iteratively creates a sequence of polygons $P^{(k)}$ with n vertices $p_i^{(k)} = (x_i^{(k)}, y_i^{(k)}), i = 1, \cdots, n$ by computing

$$\begin{array}{rclcrcl} u^{(k+1)} & = & M_n x^{(k)}, & & x^{(k+1)} & = & u^{(k+1)} / \|u^{(k+1)}\|_2, \\ v^{(k+1)} & = & M_n y^{(k)}, & & y^{(k+1)} & = & v^{(k+1)} / \|v^{(k+1)}\|_2 \end{array}$$

for $k = 0, i, \cdots$, where M_n is the averaging matrix that we defined in the lecture. Implement this algorithm and run it for $n > 8$ and random initial points $p_i \in [-1, 1]^2$ satisfying the condition (1). Plot $P^{(k)}$ for $k = 10, 50, 100$ and 1000.

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Outline

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Summary

- Power iteration
  - computing the largest magnitude eigenvalue
  - computing the smallest magnitude eigenvalue
- QR algorithm
  - computing all eigenvalues

End