Chapter 12: Eigenvalues and Singular Values

Liangda Fang

Dept. of Computer Science
Jinan University

Outline

Problem

Given a square matrix, how to compute its eigenvalues and eigenvectors?

Power Iteration

Definition

Let A be a square matrix.

- The **dominant eigenvalue** $\lambda_{\max}$ of A: an eigenvalue λ_i s.t. $|\lambda_i| > |\lambda_j|$ for $i \neq j$.
- If it exists, an eigenvector $v_{\max}$ associated to $\lambda_{\max}$ is called a **dominant eigenvector**.

Main idea: multiplication by a matrix tends to move vectors toward the dominant eigenvector direction.

Power Iteration

Example

- Let A be a matrix $\begin{bmatrix} 1 & 3 \\ 2 & 2 \end{bmatrix}$;
- Two eigenvalues: -1 and 4;
- An eigenvector associated to -1: $\begin{bmatrix} -3 & 2 \end{bmatrix}^\top$;
- An eigenvector associated to 4: $\begin{bmatrix} 1 & 1 \end{bmatrix}^\top$;
- 4 is the dominant eigenvalue and $\begin{bmatrix} 1 & 1 \end{bmatrix}^\top$ is a dominant eigenvector.

Power Iteration

Example

- Let us multiply the matrix A by a random vector $\begin{bmatrix} -5 & 5 \end{bmatrix}^\top$;

Power Iteration

Example

- Let us multiply the matrix A by a random vector $\begin{bmatrix} -5 & 5 \end{bmatrix}^\top$;
- $x_1 = Ax_0 = \begin{bmatrix} 1 & 3 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} -5 \\ 5 \end{bmatrix} = \begin{bmatrix} 10 \\ 0 \end{bmatrix}$;

Power Iteration

Example

- Let us multiply the matrix A by a random vector $\begin{bmatrix}-5 & 5\end{bmatrix}^\top$;
- $x_1 = Ax_0 = \begin{bmatrix}1 & 3\\ 2 & 2\end{bmatrix}\begin{bmatrix}-5\\ 5\end{bmatrix} = \begin{bmatrix}10\\ 0\end{bmatrix}$;
- $x_2 = A^2x_0 = Ax_1 = \begin{bmatrix}1 & 3\\ 2 & 2\end{bmatrix}\begin{bmatrix}10\\ 0\end{bmatrix} = \begin{bmatrix}10\\ 20\end{bmatrix}$;

Power Iteration

Example

- Let us multiply the matrix A by a random vector $\begin{bmatrix} -5 & 5 \end{bmatrix}^\top$;
- $x_1 = Ax_0 = \begin{bmatrix} 1 & 3 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} -5 \\ 5 \end{bmatrix} = \begin{bmatrix} 10 \\ 0 \end{bmatrix}$;
- $x_2 = A^2 x_0 = Ax_1 = \begin{bmatrix} 1 & 3 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} 10 \\ 0 \end{bmatrix} = \begin{bmatrix} 10 \\ 20 \end{bmatrix}$;
- $x_3 = A^3 x_0 = Ax_2 = \begin{bmatrix} 1 & 3 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} 10 \\ 20 \end{bmatrix} = \begin{bmatrix} 70 \\ 60 \end{bmatrix}$;

Power Iteration

Example

- Let us multiply the matrix A by a random vector $\begin{bmatrix}-5 & 5\end{bmatrix}^\top$;
- $x_1 = Ax_0 = \begin{bmatrix}1 & 3\\2 & 2\end{bmatrix}\begin{bmatrix}-5\\5\end{bmatrix} = \begin{bmatrix}10\\0\end{bmatrix}$;
- $x_2 = A^2x_0 = Ax_1 = \begin{bmatrix}1 & 3\\2 & 2\end{bmatrix}\begin{bmatrix}10\\0\end{bmatrix} = \begin{bmatrix}10\\20\end{bmatrix}$;
- $x_3 = A^3x_0 = Ax_2 = \begin{bmatrix}1 & 3\\2 & 2\end{bmatrix}\begin{bmatrix}10\\20\end{bmatrix} = \begin{bmatrix}70\\60\end{bmatrix}$;
- $x_4 = A^4x_0 = Ax_3 = \begin{bmatrix}1 & 3\\2 & 2\end{bmatrix}\begin{bmatrix}70\\60\end{bmatrix} = \begin{bmatrix}250\\260\end{bmatrix} = 260\begin{bmatrix}\frac{25}{26}\\1\end{bmatrix}$;

Power Iteration

Example

- Let us multiply the matrix A by a random vector $\begin{bmatrix}-5 & 5\end{bmatrix}^\top$;
- $x_1 = Ax_0 = \begin{bmatrix}1 & 3\\ 2 & 2\end{bmatrix}\begin{bmatrix}-5\\ 5\end{bmatrix} = \begin{bmatrix}10\\ 0\end{bmatrix}$;
- $x_2 = A^2x_0 = Ax_1 = \begin{bmatrix}1 & 3\\ 2 & 2\end{bmatrix}\begin{bmatrix}10\\ 0\end{bmatrix} = \begin{bmatrix}10\\ 20\end{bmatrix}$;
- $x_3 = A^3x_0 = Ax_2 = \begin{bmatrix}1 & 3\\ 2 & 2\end{bmatrix}\begin{bmatrix}10\\ 20\end{bmatrix} = \begin{bmatrix}70\\ 60\end{bmatrix}$;
- $x_4 = A^4x_0 = Ax_3 = \begin{bmatrix}1 & 3\\ 2 & 2\end{bmatrix}\begin{bmatrix}70\\ 60\end{bmatrix} = \begin{bmatrix}250\\ 260\end{bmatrix} = 260\begin{bmatrix}\frac{25}{26}\\ 1\end{bmatrix}$;
- Multiplying a random vector repeatedly results in moving vector close to the dominant eigenvector of A.

Power Iteration

Example

- $x_0 = 1\begin{bmatrix}1\\1\end{bmatrix} + 2\begin{bmatrix}-3\\2\end{bmatrix};$

Power Iteration

Example

- $x_0 = 1 \begin{bmatrix} 1 \\ 1 \end{bmatrix} + 2 \begin{bmatrix} -3 \\ 2 \end{bmatrix};$
- $x_1 = Ax_0 = 4 \begin{bmatrix} 1 \\ 1 \end{bmatrix} + (-1) \cdot 2 \begin{bmatrix} -3 \\ 2 \end{bmatrix};$

Power Iteration

Example

- $x_0 = 1 \begin{bmatrix} 1 \\ 1 \end{bmatrix} + 2 \begin{bmatrix} -3 \\ 2 \end{bmatrix};$
- $x_1 = Ax_0 = 4 \begin{bmatrix} 1 \\ 1 \end{bmatrix} + (-1) \cdot 2 \begin{bmatrix} -3 \\ 2 \end{bmatrix};$
- $x_2 = A^2 x_0 = 4^2 \begin{bmatrix} 1 \\ 1 \end{bmatrix} + (-1)^2 \cdot 2 \begin{bmatrix} -3 \\ 2 \end{bmatrix};$

Power Iteration

Example

- $x_0 = 1 \begin{bmatrix} 1 \\ 1 \end{bmatrix} + 2 \begin{bmatrix} -3 \\ 2 \end{bmatrix};$
- $x_1 = Ax_0 = 4 \begin{bmatrix} 1 \\ 1 \end{bmatrix} + (-1) \cdot 2 \begin{bmatrix} -3 \\ 2 \end{bmatrix};$
- $x_2 = A^2 x_0 = 4^2 \begin{bmatrix} 1 \\ 1 \end{bmatrix} + (-1)^2 \cdot 2 \begin{bmatrix} -3 \\ 2 \end{bmatrix};$
- $x_3 = A^3 x_0 = 4^3 \begin{bmatrix} 1 \\ 1 \end{bmatrix} + (-1)^3 \cdot 2 \begin{bmatrix} -3 \\ 2 \end{bmatrix};$

Power Iteration

Example

- $x_0 = 1 \begin{bmatrix} 1 \\ 1 \end{bmatrix} + 2 \begin{bmatrix} -3 \\ 2 \end{bmatrix};$
- $x_1 = Ax_0 = 4 \begin{bmatrix} 1 \\ 1 \end{bmatrix} + (-1) \cdot 2 \begin{bmatrix} -3 \\ 2 \end{bmatrix};$
- $x_2 = A^2 x_0 = 4^2 \begin{bmatrix} 1 \\ 1 \end{bmatrix} + (-1)^2 \cdot 2 \begin{bmatrix} -3 \\ 2 \end{bmatrix};$
- $x_3 = A^3 x_0 = 4^3 \begin{bmatrix} 1 \\ 1 \end{bmatrix} + (-1)^3 \cdot 2 \begin{bmatrix} -3 \\ 2 \end{bmatrix};$
- $x_4 = A^4 x_0 = 4^4 \begin{bmatrix} 1 \\ 1 \end{bmatrix} + (-1)^4 \cdot 2 \begin{bmatrix} -3 \\ 2 \end{bmatrix} = 256 \begin{bmatrix} 1 \\ 1 \end{bmatrix} + 2 \begin{bmatrix} -3 \\ 2 \end{bmatrix};$

Power Iteration

Example

- $x_0 = 1 \begin{bmatrix} 1 \\ 1 \end{bmatrix} + 2 \begin{bmatrix} -3 \\ 2 \end{bmatrix};$
- $x_1 = Ax_0 = 4 \begin{bmatrix} 1 \\ 1 \end{bmatrix} + (-1) \cdot 2 \begin{bmatrix} -3 \\ 2 \end{bmatrix};$
- $x_2 = A^2 x_0 = 4^2 \begin{bmatrix} 1 \\ 1 \end{bmatrix} + (-1)^2 \cdot 2 \begin{bmatrix} -3 \\ 2 \end{bmatrix};$
- $x_3 = A^3 x_0 = 4^3 \begin{bmatrix} 1 \\ 1 \end{bmatrix} + (-1)^3 \cdot 2 \begin{bmatrix} -3 \\ 2 \end{bmatrix};$
- $x_4 = A^4 x_0 = 4^4 \begin{bmatrix} 1 \\ 1 \end{bmatrix} + (-1)^4 \cdot 2 \begin{bmatrix} -3 \\ 2 \end{bmatrix} = 256 \begin{bmatrix} 1 \\ 1 \end{bmatrix} + 2 \begin{bmatrix} -3 \\ 2 \end{bmatrix};$
- To keep the numbers from getting out of hand, it is necessary to normalize x_i's.

Power Iteration

Given a dominant eigenvector $v_{\max}$, how to compute a dominant eigenvalue $\lambda_{\max}$?

Power Iteration

Given a dominant eigenvector $v_{\max}$, how to compute a dominant eigenvalue $\lambda_{\max}$?

Definition

Let A be an $n \times n$ matrix and v an n-dimensional vector. Then $\lambda = \frac{v^\top A v}{v^\top v}$ is called **Rayleigh quotient**.

Power Iteration

Given a dominant eigenvector $v_{\max}$, how to compute a dominant eigenvalue $\lambda_{\max}$?

Definition

Let A be an $n \times n$ matrix and v an n-dimensional vector. Then $\lambda = \frac{v^\top A v}{v^\top v}$ is called **Rayleigh quotient**.

$Av = \lambda v$

Power Iteration

Given a dominant eigenvector $v_{\max}$, how to compute a dominant eigenvalue $\lambda_{\max}$?

Definition

Let A be an $n \times n$ matrix and v an n-dimensional vector. Then $\lambda = \frac{v^\top Av}{v^\top v}$ is called **Rayleigh quotient**.

$$Av = \lambda v \Rightarrow v^\top Av = v^\top \lambda v \Rightarrow$$

Power Iteration

Given a dominant eigenvector $v_{\max}$, how to compute a dominant eigenvalue $\lambda_{\max}$?

Definition

Let A be an $n \times n$ matrix and v an n-dimensional vector. Then $\lambda = \frac{v^\top Av}{v^\top v}$ is called **Rayleigh quotient**.

$$Av = \lambda v \Rightarrow v^\top Av = v^\top \lambda v \Rightarrow v^\top Av = \lambda v^\top v \Rightarrow$$

Power Iteration

Given a dominant eigenvector $v_{\max}$, how to compute a dominant eigenvalue $\lambda_{\max}$?

Definition

Let A be an $n \times n$ matrix and v an n-dimensional vector. Then $\lambda = \frac{v^\top Av}{v^\top v}$ is called **Rayleigh quotient**.

$$Av = \lambda v \Rightarrow v^\top Av = v^\top \lambda v \Rightarrow v^\top Av = \lambda v^\top v \Rightarrow \lambda = \frac{v^\top Av}{v^\top v}.$$

Power Iteration

Algorithm 1: Power Iteration

x_0 = initial vector

for $j = 1, 2, \ldots$ **do**

$\quad v_{j-1} = \frac{x_{j-1}}{\|x_{j-1}\|_2}$

$\quad x_j = A v_{j-1}$

$v_{\max} = \frac{x_j}{\|x_j\|_2}$

$\lambda_{\max} = {v_{\max}}^\top A v_{\max}$

Power Iteration

Theorem

Let A be an $n \times n$ matrix with real eigenvalues $\lambda_1, \ldots, \lambda_n$ satisfying $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|$. Assume that the eigenvalues of A span R^n. For almost every initial vector, Power Iteration converges to an eigenvector associated to λ_1.

Power Iteration

Proof.

- Let $v_1, \cdots, v_n$ be the eigenvectors w.r.t. $\lambda_1, \ldots, \lambda_n$ respectively.

Power Iteration

Proof.

- Let $v_1, \cdots, v_n$ be the eigenvectors w.r.t. $\lambda_1, \ldots, \lambda_n$ respectively.
- The initial vector x_0 can be expressed as $c_1 v_1 + \cdots + c_n v_n$ where $c_1 \neq 0$.

Power Iteration

Proof.

- Let $v_1, \cdots, v_n$ be the eigenvectors w.r.t. $\lambda_1, \ldots, \lambda_n$ respectively.
- The initial vector x_0 can be expressed as $c_1 v_1 + \cdots + c_n v_n$ where $c_1 \neq 0$.
- Applying Power Iteration yields

$$Ax_0 = c_1 \lambda_1 v_1 + \cdots + c_n \lambda_n v_n$$

Power Iteration

Proof.

- Let $v_1, \cdots, v_n$ be the eigenvectors w.r.t. $\lambda_1, \ldots, \lambda_n$ respectively.
- The initial vector x_0 can be expressed as $c_1 v_1 + \cdots + c_n v_n$ where $c_1 \neq 0$.
- Applying Power Iteration yields

$$\begin{aligned} Ax_0 &= c_1\lambda_1 v_1 + \cdots + c_n\lambda_n v_n \\ A^2 x_0 &= c_1\lambda_1^2 v_1 + \cdots + c_n\lambda_n^2 v_n \end{aligned}$$

Power Iteration

Proof.

- Let $v_1, \cdots, v_n$ be the eigenvectors w.r.t. $\lambda_1, \ldots, \lambda_n$ respectively.
- The initial vector x_0 can be expressed as $c_1 v_1 + \cdots + c_n v_n$ where $c_1 \neq 0$.
- Applying Power Iteration yields

$$\begin{aligned} Ax_0 &= c_1\lambda_1 v_1 + \cdots + c_n\lambda_n v_n \\ A^2x_0 &= c_1\lambda_1^2 v_1 + \cdots + c_n\lambda_n^2 v_n \\ &\vdots \\ A^kx_0 &= c_1\lambda_1^k v_1 + \cdots + c_n\lambda_n^k v_n \end{aligned}$$

Power Iteration

Proof.

- Let $v_1, \cdots, v_n$ be the eigenvectors w.r.t. $\lambda_1, \ldots, \lambda_n$ respectively.
- The initial vector x_0 can be expressed as $c_1 v_1 + \cdots + c_n v_n$ where $c_1 \neq 0$.
- Applying Power Iteration yields

$$\begin{aligned} Ax_0 &= c_1\lambda_1 v_1 + \cdots + c_n\lambda_n v_n \\ A^2x_0 &= c_1\lambda_1^2 v_1 + \cdots + c_n\lambda_n^2 v_n \\ &\vdots \\ A^kx_0 &= c_1\lambda_1^k v_1 + \cdots + c_n\lambda_n^k v_n \end{aligned}$$

- $\lim_{k\to\infty} \frac{A^k x_0}{\lambda_1^k} = \lim_{k\to\infty} [c_1 v_1 + \cdots + c_n(\frac{\lambda_n}{\lambda_1})^k v_n] = c_1 v_1$.

Outline

Inverse Power Iteration

Sometimes, we want to compute the smallest eigenvalue λ_i, *i.e.*, $|\lambda_i| < |\lambda_j|$ for $i \neq j$.

Inverse Power Iteration

Sometimes, we want to compute the smallest eigenvalue λ_i, *i.e.*, $|\lambda_i| < |\lambda_j|$ for $i \neq j$.

Lemma

Let the eigenvalues of the $n \times n$ matrix A be denoted by $\lambda_1, \lambda_2, \ldots, \lambda_n$, and the eigenvector associated to λ_i is v_i. If A^{-1} exists, then

1. *The eigenvalues of A^{-1} are $\frac{1}{\lambda_1}, \frac{1}{\lambda_2}, \ldots, \frac{1}{\lambda_n}$;*
2. *The eigenvector associated to $\frac{1}{\lambda_i}$ is also v_i.*

Inverse Power Iteration

Sometimes, we want to compute the smallest eigenvalue λ_i, *i.e.*, $|\lambda_i| < |\lambda_j|$ for $i \neq j$.

Lemma

Let the eigenvalues of the $n \times n$ matrix A be denoted by $\lambda_1, \lambda_2, \ldots, \lambda_n$, and the eigenvector associated to λ_i is v_i. If A^{-1} exists, then

1. *The eigenvalues of A^{-1} are $\frac{1}{\lambda_1}, \frac{1}{\lambda_2}, \ldots, \frac{1}{\lambda_n}$;*
2. *The eigenvector associated to $\frac{1}{\lambda_i}$ is also v_i.*

Proof.

$Av = \lambda v$

Inverse Power Iteration

Sometimes, we want to compute the smallest eigenvalue λ_i, *i.e.*, $|\lambda_i| < |\lambda_j|$ for $i \neq j$.

Lemma

Let the eigenvalues of the $n \times n$ matrix A be denoted by $\lambda_1, \lambda_2, \ldots, \lambda_n$, and the eigenvector associated to λ_i is v_i. If A^{-1} exists, then

1. *The eigenvalues of A^{-1} are $\frac{1}{\lambda_1}, \frac{1}{\lambda_2}, \ldots, \frac{1}{\lambda_n}$;*
2. *The eigenvector associated to $\frac{1}{\lambda_i}$ is also v_i.*

Proof.

$$Av = \lambda v \Rightarrow A^{-1}Av = A^{-1}\lambda v$$

Inverse Power Iteration

Sometimes, we want to compute the smallest eigenvalue λ_i, *i.e.*, $|\lambda_i| < |\lambda_j|$ for $i \neq j$.

Lemma

Let the eigenvalues of the $n \times n$ matrix A be denoted by $\lambda_1, \lambda_2, \ldots, \lambda_n$, and the eigenvector associated to λ_i is v_i. If A^{-1} exists, then

1. *The eigenvalues of A^{-1} are $\frac{1}{\lambda_1}, \frac{1}{\lambda_2}, \ldots, \frac{1}{\lambda_n}$;*
2. *The eigenvector associated to $\frac{1}{\lambda_i}$ is also v_i.*

Proof.

$$Av = \lambda v \Rightarrow A^{-1}Av = A^{-1}\lambda v \Rightarrow v = \lambda A^{-1} v$$

Inverse Power Iteration

Sometimes, we want to compute the smallest eigenvalue λ_i, *i.e.*, $|\lambda_i| < |\lambda_j|$ for $i \neq j$.

Lemma

Let the eigenvalues of the $n \times n$ matrix A be denoted by $\lambda_1, \lambda_2, \ldots, \lambda_n$, and the eigenvector associated to λ_i is v_i. If A^{-1} exists, then

1. *The eigenvalues of A^{-1} are $\frac{1}{\lambda_1}, \frac{1}{\lambda_2}, \ldots, \frac{1}{\lambda_n}$;*
2. *The eigenvector associated to $\frac{1}{\lambda_i}$ is also v_i.*

Proof.

$$Av = \lambda v \Rightarrow A^{-1}Av = A^{-1}\lambda v \Rightarrow v = \lambda A^{-1} v \Rightarrow \frac{1}{\lambda} v = A^{-1} v.$$

Inverse Power Iteration

Sometimes, we want to compute the smallest eigenvalue λ_i, *i.e.*, $|\lambda_i| < |\lambda_j|$ for $i \neq j$.

Lemma

Let the eigenvalues of the $n \times n$ matrix A be denoted by $\lambda_1, \lambda_2, \ldots, \lambda_n$, and the eigenvector associated to λ_i is v_i. If A^{-1} exists, then

1. *The eigenvalues of A^{-1} are $\frac{1}{\lambda_1}, \frac{1}{\lambda_2}, \ldots, \frac{1}{\lambda_n}$;*
2. *The eigenvector associated to $\frac{1}{\lambda_i}$ is also v_i.*

Proof.

$$Av = \lambda v \Rightarrow A^{-1}Av = A^{-1}\lambda v \Rightarrow v = \lambda A^{-1}v \Rightarrow \frac{1}{\lambda}v = A^{-1}v.$$

The smallest eigenvalue of A = the dominant eigenvalue of A^{-1}.

Inverse Power Iteration

Algorithm 2: Inverse Power Iteration 1

1 x_0 = initial vector

2 **for** $j = 1, 2, \ldots$ **do**

3 $\quad v_{j-1} = \frac{x_{j-1}}{\|x_{j-1}\|_2}$

4 $\quad$ Solve $Ax_j = v_{j-1}$ $(x_j = A^{-1}v_{j-1})$

5 $v_{\min} = \frac{x_j}{\|x_j\|_2}$

6 $\lambda_{\min} = \frac{1}{v_{j-1}^\top x_j}$

Inverse Power Iteration

If we know an eigenvalue λ is close to s, how to compute it?

Inverse Power Iteration

If we know an eigenvalue λ is close to s, how to compute it?

Lemma

Let the eigenvalues of the $n \times n$ matrix A be denoted by $\lambda_1, \lambda_2, \ldots, \lambda_n$, the eigenvector associated to λ_i is v_i, and a shift s. Then

1. *The eigenvalues of $A - sI$ are $\lambda_1 - s, \lambda_2 - s, \ldots, \lambda_n - s$;*
2. *The eigenvector associated to $\lambda_i - s$ is also v_i.*

Inverse Power Iteration

If we know an eigenvalue λ is close to s, how to compute it?

Lemma

Let the eigenvalues of the $n \times n$ matrix A be denoted by $\lambda_1, \lambda_2, \ldots, \lambda_n$, the eigenvector associated to λ_i is v_i, and a shift s. Then

1. *The eigenvalues of $A - sI$ are $\lambda_1 - s, \lambda_2 - s, \ldots, \lambda_n - s$;*
2. *The eigenvector associated to $\lambda_i - s$ is also v_i.*

Proof.

$(A - sI)v = Av - sv = \lambda v - sv = (\lambda - s)v.$

Inverse Power Iteration

If we know an eigenvalue λ is close to s, how to compute it?

Lemma

Let the eigenvalues of the $n \times n$ matrix A be denoted by $\lambda_1, \lambda_2, \ldots, \lambda_n$, the eigenvector associated to λ_i is v_i, and a shift s. Then

1. *The eigenvalues of $A - sI$ are $\lambda_1 - s, \lambda_2 - s, \ldots, \lambda_n - s$;*
2. *The eigenvector associated to $\lambda_i - s$ is also v_i.*

Proof.

$(A - sI)v = Av - sv = \lambda v - sv = (\lambda - s)v.$

The eigenvalue of A close to s is the dominant eigenvalue of $(A - sI)^{-1}$.

Inverse Power Iteration

Algorithm 3: Inverse Power Iteration 2

x_0 = initial vector
for $j = 1, 2, \ldots$ **do**
  $v_{j-1} = \frac{x_{j-1}}{\|x_{j-1}\|_2}$
  Solve $(A - sI)x_j = v_{j-1}$ $(x_j = (A - sI)^{-1} v_{j-1})$
$v = \frac{x_j}{\|x_j\|_2}$
$\lambda = \frac{1}{v_{j-1}^{\top} x_j} + s$

Outline

Rayleigh Quotient Iteration

- The Rayleigh quotient can be used in conjunction with Inverse Power Iteration.
- It converges to the eigenvector associated to the eigenvalue with the smallest distance to the shift s.
- At each step, let an approximate eigenvalue to be the shift so as to speed convergence.

Rayleigh Quotient Iteration

Algorithm 4: Rayleigh Quotient Iteration

$x_0 =$ initial vector

for $j = 1, 2, \ldots$ **do**

  $v_{j-1} = \frac{x_{j-1}}{\|x_{j-1}\|_2}$

  $\lambda_{j-1} = v_{j-1}^\top A v_{j-1}$

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

$v_{\max} = \frac{x_j}{\|x_j\|_2}$

$\lambda_{\max} = v_{\max}^\top A v_{\max}$

Outline

Webpages

- Nowadays, the Internet is indispensable for us.
- Many person browser many webpages when surfing the Internet.

Webpages


报考学生 在校生 教职员 家长|访客 校友
110周年校庆 门户 邮件 通知 会议 | 图书馆 | English
暨南大學
JINAN UNIVERSITY
1906
忠信篤敬
学校概况 组织机构 招生就业 人才培养 人才招聘 科学研究 合作交流 综合服务
赢在创新
暨南梦工厂
赢在创新，暨南学子的“造梦工厂”

Search Engine

- If we search some materials, we will resort to some search engine.
- When we enter some keywords into the search engine, and the latter will recommend us the addresses of some related webpages.

Search Engine

The principle of the search engine

A search engine runs as follows:

1. Use web crawler to collect most web pages in the Internet;

The principle of the search engine

A search engine runs as follows:

1. Use web crawler to collect most web pages in the Internet;
2. Store the contents of the web pages into a distributed storage center, namely Bigtable;

The principle of the search engine

A search engine runs as follows:

1. Use web crawler to collect most web pages in the Internet;
2. Store the contents of the web pages into a distributed storage center, namely Bigtable;
3. Measure the relative importance of each web page, namely PageRank;

The principle of the search engine

A search engine runs as follows:

1. Use web crawler to collect most web pages in the Internet;
2. Store the contents of the web pages into a distributed storage center, namely Bigtable;
3. Measure the relative importance of each web page, namely PageRank;
4. Select the top n web pages, which contains the keywords, from a distributed storage center, namely MapReduce.

The topic of this talk: PageRank

In this talk, we focus on the third step of the search engine, PageRank.

The link structure of webpages

A webpage contains many links of other webpages.

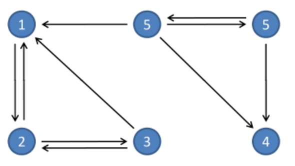
The link structure of webpages

A webpage contains many links of other webpages.

The link structure of webpages as a adjacency matrix

Example

Suppose that we have 6 webpages. The link structure of them is as follows:

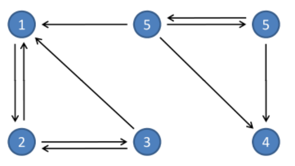


$$L = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

The link structure of webpages as a adjacency matrix

Example

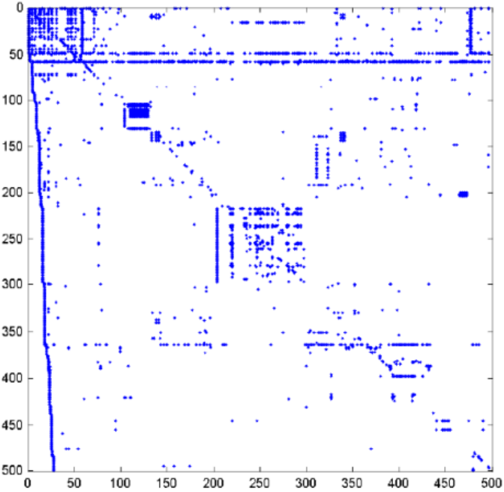
Suppose that we have 6 webpages. The link structure of them is as follows:



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

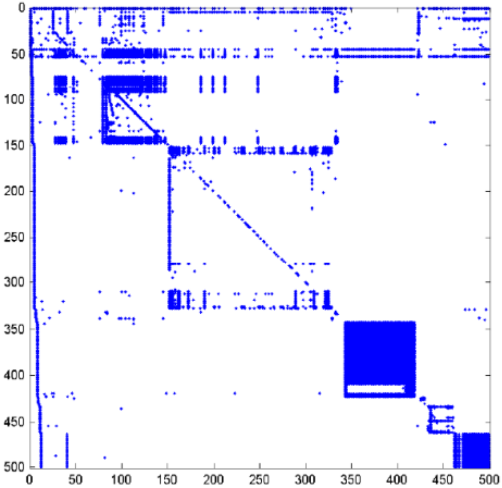
- The adjacency matrix of webpages is very spare.
- Sparcity $=$ the number of edges / (the number of nodes)^2.

The property of the adjacency matrix: sparse



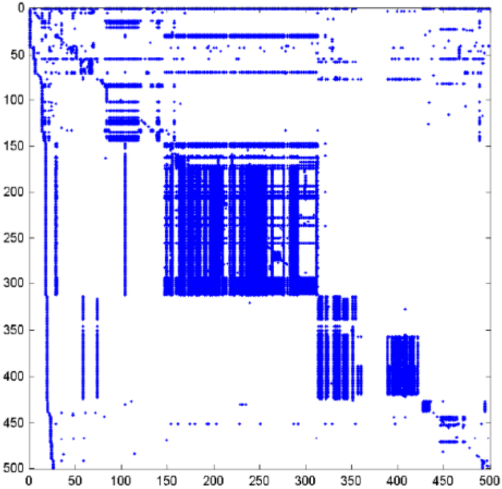
Sparsity = 1.24%

The property of the adjacency matrix: sparse



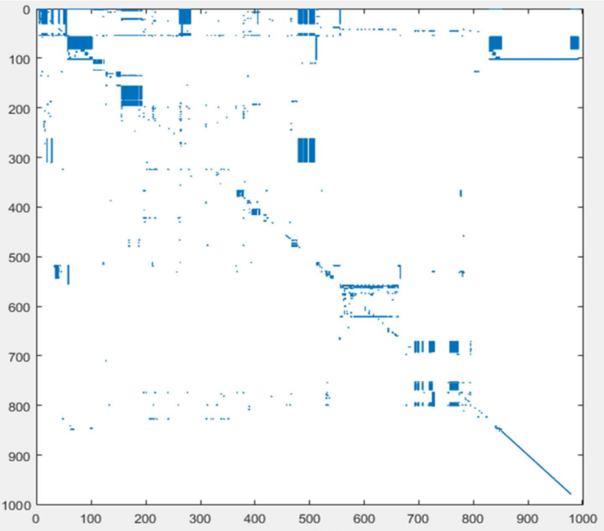
Sparsity = 4.7%

The property of the adjacency matrix: sparse



Sparsity = 8.4%

The property of the adjacency matrix: sparse



The score of the webpages

- Each webpage i has a score $r_i > 0$ representing the importance of the webpage;
- If $r_i > r_j$, then we consider webpage i more important than webpage j;
- Normalize all scores of webpages, *i.e.*, $\|r\|_1 = \sum_{i=1}^{k} r_i = 1$ for $r = [r_1, r_2, \ldots, r_k]^\top$.

The insights behind PageRank

- A webpage with good score have inlinks from those with good scores;
- A webpage with good score have outlinks to those with good scores;
- Inlinks from good webpages should carry more weight than inlinks from marginal webpages.

The insights behind PageRank

1. Webpages vote for the importance of other webpages by linking to them;
   - The more inlinks a page has, the more important it is.

The insights behind PageRank

1. Webpages vote for the importance of other webpages by linking to them;
   - The more inlinks a page has, the more important it is.
2. One webpage has only one vote;
   - If a webpage has more than one outlinks, its vote must be split.

The insights behind PageRank

1. Webpages vote for the importance of other webpages by linking to them;
    - The more inlinks a page has, the more important it is.
2. One webpage has only one vote;
    - If a webpage has more than one outlinks, its vote must be split.
3. A link to webpage i from an important page increases webpage i's importance more than a link from an unimportant one.
    - It matters who your supporters are.

Weighing the votes

- $A_{ij} = \frac{L_{ij}}{\sum\limits_{i=1}^{n} L_{ij}}$: the value of A_{ij} is dividing L_{ij} by the sum of the row j of L.

Example

$$L = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \to A = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & \frac{1}{3} \\ 1 & 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{3} \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{3} \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 0 \end{bmatrix}$$

First attempt at determining the rank of each page

- The rank r_i of page i: the weighted sum of the ranks of all webpages pointing to it.

First attempt at determining the rank of each page

- The rank r_i of page i: the weighted sum of the ranks of all webpages pointing to it.

Example

$$A = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & \frac{1}{3} \\ 1 & 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{3} \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{3} \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 0 \end{bmatrix} \rightarrow \begin{cases} r_1 = \frac{1}{2} \cdot r_2 + \frac{1}{2} \cdot r_3 + \frac{1}{3} \cdot r_6 \\ r_2 = 1 \cdot r_1 + \frac{1}{2} \cdot r_3 \\ r_3 = \frac{1}{2} \cdot r_2 \\ r_4 = \frac{1}{2} \cdot r_5 + \frac{1}{3} \cdot r_6 \\ r_5 = \frac{1}{3} \cdot r_6 \\ r_6 = \frac{1}{2} \cdot r_5 \end{cases}$$

First attempt at determining the rank of each page

- The rank r_i of page i: the weighted sum of the ranks of all webpages pointing to it.

Example

$$A = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & \frac{1}{3} \\ 1 & 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{3} \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{3} \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 0 \end{bmatrix} \to \begin{cases} r_1 = \frac{1}{2} \cdot r_2 + \frac{1}{2} \cdot r_3 + \frac{1}{3} \cdot r_6 \\ r_2 = 1 \cdot r_1 + \frac{1}{2} \cdot r_3 \\ r_3 = \frac{1}{2} \cdot r_2 \\ r_4 = \frac{1}{2} \cdot r_5 + \frac{1}{3} \cdot r_6 \\ r_5 = \frac{1}{3} \cdot r_6 \\ r_6 = \frac{1}{2} \cdot r_5 \end{cases}$$

- $r = Ar$, *i.e.*, r is an eigenvector of A if an eigenvalue of A is equal to 1;
- It is possible to reduce ranking webpages to computing the eigenvalue of the adjacency matrix from the link structure.

Some difficulties on reducing

1. Unfortunately, many adjacency matrices does not have eigenvalue 1.

2. Even if the matrix has an eigenvalue 1, it is computationally expensive to get the eigenvector wrt 1.

The first difficulty on reducing

1. Unfortunately, many adjacency matrices does not have eigenvalue 1.

2. Even if the matrix has an eigenvalue 1, it is computationally expensive to get the eigenvector wrt 1.

Column stochastic matrix

- Make sure that adjacency matrix has an eigenvalue 1.

Definition

We say that a matrix A is **column stochastic**, if

1. all entries of A are non-negative;
2. each column of A sums up to 1.

Proposition

Let A be a column stochastic matrix.

1. 1 *is always an eigenvalue of* A.
2. $\|A\|_1 = 1$ *where* $\|A\|_1$ *is the largest column sum.*

The second difficulty on reducing

1. Unfortunately, many adjacency matrices do not have eigenvalue 1.

2. Even if the matrix has an eigenvalue 1, it is computationally expensive to get the eigenvector wrt 1.

Power Iteration

- If we restrict the matrix s.t. it is spare and its dominant eigenvalue is 1, then power iteration is an efficient solution to ranking pages.
- How to ensure the dominant eigenvalue of a given matrix to be 1?

Theorem (Perron-Frobenius theorem)

If every entries of the matrix A are positive (in short, we say A is ***positive****), then there exists a positive real number λ such that λ is the dominant eigenvalue of A.*

Together with column stochastic property

Proposition

Let A be a column stochastic matrix.

1. 1 *is always an eigenvalue of* A*.*
2. $\|A\|_1 = 1$ *where* $\|A\|_1$ *is the largest column sum.*

Together with column stochastic property

Proposition

Let A be a column stochastic matrix.

1. 1 *is always an eigenvalue of* A.
2. $\|A\|_1 = 1$ *where* $\|A\|_1$ *is the largest column sum.*

Proposition

Let A be a matrix. Then, every eigenvalue of A is less than or equal to $\|A\|_1$.

Together with column stochastic property

Proposition

Let A be a column stochastic matrix.

1. 1 *is always an eigenvalue of* A.
2. $\|A\|_1 = 1$ *where* $\|A\|_1$ *is the largest column sum.*

Proposition

Let A be a matrix. Then, every eigenvalue of A is less than or equal to $\|A\|_1$.

- If A is positive and column stochastic, then the dominant eigenvalue of A is 1.

Adjusting the adjacency matrix

- The original adjacency matrix is not positive or column stochastic.
- We need to adjust it.

Making it to be column stochastic

Example

$$A = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & \frac{1}{3} \\ 1 & 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{3} \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{3} \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 0 \end{bmatrix}$$ is not column stochastic.

Dealing with the dangling node

- At a "dangling node", the web surfer can choose to stay put or jump to any other node with equal probability.
- $B = A + \frac{1}{n} e d^\top$ where
  1. $e = \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}^\top$;
  2. $d^\top = \begin{bmatrix} d_1 & d_2 & \cdots & d_n \end{bmatrix}$ with $d_i = \begin{cases} 1, & \text{if node } i \text{ is dangling;} \\ 0, & \text{otherwise.} \end{cases}$
- B is column stochastic.

Dealing with the dangling node

Example

$$A = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & \frac{1}{3} \\ 1 & 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & \frac{1}{3} \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{3} \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 0 \end{bmatrix} \to B = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} & \frac{1}{6} & 0 & \frac{1}{3} \\ 1 & 0 & \frac{1}{2} & \frac{1}{6} & 0 & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{6} & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{6} & \frac{1}{2} & \frac{1}{3} \\ 0 & 0 & 0 & \frac{1}{6} & 0 & \frac{1}{3} \\ 0 & 0 & 0 & \frac{1}{6} & \frac{1}{2} & 0 \end{bmatrix}.$$

- But B is not positive.

Enforcing it positive

- If a surfer at a "dangling node" can jump to any other node, surfers at nodes with outlinks should be able to do the same.
- $G = \alpha B + \frac{1}{n}(1-\alpha)ee^\top$ where $\alpha = 0.85$ and $ee^\top$ is an $n \times n$ matrix with the all entries equal to 1.
- G is positive and column stochastic.

Enforcing it positive

Example

$$B = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} & \frac{1}{6} & 0 & \frac{1}{3} \\ 1 & 0 & \frac{1}{2} & \frac{1}{6} & 0 & 0 \\ 0 & \frac{1}{2} & 0 & \frac{1}{6} & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{6} & \frac{1}{2} & \frac{1}{3} \\ 0 & 0 & 0 & \frac{1}{6} & 0 & \frac{1}{3} \\ 0 & 0 & 0 & \frac{1}{6} & \frac{1}{2} & 0 \end{bmatrix} \to G =$$

$$\begin{bmatrix} \frac{1-\alpha}{6} & \frac{1}{2}\alpha + \frac{1-\alpha}{6} & \frac{1}{2}\alpha + \frac{1-\alpha}{6} & \frac{1}{6} & \frac{1-\alpha}{6} & \frac{1}{3}\alpha + \frac{1-\alpha}{6} \\ 1\cdot\alpha + \frac{1-\alpha}{6} & \frac{1-\alpha}{6} & \frac{1}{2}\alpha + \frac{1-\alpha}{6} & \frac{1}{6} & \frac{1-\alpha}{6} & \frac{1-\alpha}{6} \\ \frac{1-\alpha}{6} & \frac{1}{2}\alpha + \frac{1-\alpha}{6} & \frac{1-\alpha}{6} & \frac{1}{6} & \frac{1-\alpha}{6} & \frac{1-\alpha}{6} \\ \frac{1-\alpha}{6} & \frac{1-\alpha}{6} & \frac{1-\alpha}{6} & \frac{1}{6} & \frac{1}{2}\alpha + \frac{1-\alpha}{6} & \frac{1}{3}\alpha + \frac{1-\alpha}{6} \\ \frac{1-\alpha}{6} & \frac{1-\alpha}{6} & \frac{1-\alpha}{6} & \frac{1}{6} & \frac{1-\alpha}{6} & \frac{1}{3}\alpha + \frac{1-\alpha}{6} \\ \frac{1-\alpha}{6} & \frac{1-\alpha}{6} & \frac{1-\alpha}{6} & \frac{1}{6} & \frac{1}{2}\alpha + \frac{1-\alpha}{6} & \frac{1-\alpha}{6} \end{bmatrix}.$$

Enforcing it positive

- G is positive and column stochastic, but not sparse.
- Computing the dominant eigenvector of G via directly using the power method: inefficient.
- We need to modify the power method for G.

Power Iteration for G

Proposition

If A is column stochastic and $\|x\|_1 = 1$, then $\|Ax\|_1 = 1$.

- The normalization step $v_{j-1} = \frac{x_{j-1}}{\|x_{j-1}\|_2}$ can be removed.

Power Iteration for G

- Simplify the step: $x_j = Gx_{j-1}$.

$$x_j = Gx_{j-1}$$

Power Iteration for G

- Simplify the step: $x_j = Gx_{j-1}$.

$$
\begin{aligned}
x_j &= Gx_{j-1} \\
&= [\alpha B + (1-\alpha)\frac{1}{n}ee^\top]x_{j-1}
\end{aligned}
$$

Power Iteration for G

- Simplify the step: $x_j = Gx_{j-1}$.

$$
\begin{aligned}
x_j &= Gx_{j-1} \\
&= [\alpha B + (1-\alpha)\frac{1}{n}ee^\top]x_{j-1} \\
&= \alpha[A + \frac{1}{n}ed^\top]x_{j-1} + (1-\alpha)\frac{1}{n}ee^\top x_{j-1}
\end{aligned}
$$

Power Iteration for G

- Simplify the step: $x_j = Gx_{j-1}$.

$$\begin{aligned}
x_j &= Gx_{j-1} \\
&= [\alpha B + (1-\alpha)\frac{1}{n}ee^\top]x_{j-1} \\
&= \alpha[A + \frac{1}{n}ed^\top]x_{j-1} + (1-\alpha)\frac{1}{n}ee^\top x_{j-1} \\
&= \alpha A x_{j-1} + \frac{1}{n}[\alpha d^\top x_{j-1} + (1-\alpha)e^\top x_{j-1}]e
\end{aligned}$$

Power Iteration for G

- Simplify the step: $x_j = Gx_{j-1}$.

$$
\begin{aligned}
x_j &= Gx_{j-1} \\
&= [\alpha B + (1-\alpha)\frac{1}{n}ee^\top]x_{j-1} \\
&= \alpha[A + \frac{1}{n}ed^\top]x_{j-1} + (1-\alpha)\frac{1}{n}ee^\top x_{j-1} \\
&= \alpha A x_{j-1} + \frac{1}{n}[\alpha d^\top x_{j-1} + (1-\alpha)e^\top x_{j-1}]e
\end{aligned}
$$

Power Iteration for G

Let $\beta = \alpha d^\top x_{j-1} + (1-\alpha) e^\top x_{j-1}$.

$$e^\top x_j \quad = \quad e^\top [\alpha A x_{j-1} + \frac{1}{n} \beta e]$$

Power Iteration for G

Let $\beta = \alpha d^\top x_{j-1} + (1-\alpha) e^\top x_{j-1}$.

$$
\begin{aligned}
e^\top x_j &= e^\top [\alpha A x_{j-1} + \frac{1}{n}\beta e] \\
&= \alpha e^\top A x_{j-1} + \beta \frac{1}{n} e^\top e
\end{aligned}
$$

Power Iteration for G

Let $\beta = \alpha d^\top x_{j-1} + (1-\alpha) e^\top x_{j-1}$.

$$
\begin{aligned}
e^\top x_j &= e^\top [\alpha A x_{j-1} + \frac{1}{n}\beta e] \\
&= \alpha e^\top A x_{j-1} + \beta \frac{1}{n} e^\top e \\
&= \alpha \|A x_{j-1}\|_1 + \beta
\end{aligned}
$$

Power Iteration for G

Let $\beta = \alpha d^\top x_{j-1} + (1-\alpha) e^\top x_{j-1}$.

$$
\begin{aligned}
e^\top x_j &= e^\top [\alpha A x_{j-1} + \frac{1}{n} \beta e] \\
&= \alpha e^\top A x_{j-1} + \beta \frac{1}{n} e^\top e \\
&= \alpha \|A x_{j-1}\|_1 + \beta \\
&= 1
\end{aligned}
$$

Power Iteration for G

Let $\beta = \alpha d^\top x_{j-1} + (1-\alpha) e^\top x_{j-1}$.

$$\begin{aligned} e^\top x_j &= e^\top [\alpha A x_{j-1} + \frac{1}{n}\beta e] \\ &= \alpha e^\top A x_{j-1} + \beta \frac{1}{n} e^\top e \\ &= \alpha \|A x_{j-1}\|_1 + \beta \\ &= 1 \end{aligned}$$

- So $\beta = 1 - \alpha \|A x_{j-1}\|_1$.
- $x_j = G x_{j-1} \Rightarrow$
  1. $y = A x_{j-1}$;
  2. $\beta = 1 - \alpha \|y\|_1$;
  3. $x_j = \alpha y + \frac{\beta}{n} e$.

Power Iteration for G

Algorithm 5: Power Iteration for G

```
x_0 = initial vector with ||x_0||_1 = 1
for j = 1, 2, ... do
    y = A x_{j-1}
    β = 1 − α||y||_1
    x_j = α y + (β/n) e
v_max = x_j
```

$x_0 =$ initial vector with $\|x_0\|_1 = 1$

for $j = 1, 2, \ldots$ **do**

- $y = Ax_{j-1}$
- $\beta = 1 - \alpha\|y\|_1$
- $x_j = \alpha y + \frac{\beta}{n} e$

$v_{max} = x_j$

Outline

Simultaneous iteration

- Now, we consider how to compute all eigenvalues of a matrix.
- First, we focus on symmetric matrices since their eigenvectors are pairwise orthogonal.

Normalized Simultaneous iteration

1. Assume we have n pairwise orthogonal initial vectors $q_1^0, \ldots, q_n^0$;

Normalized Simultaneous iteration

1. Assume we have n pairwise orthogonal initial vectors $q_1^0, \ldots, q_n^0$;
2. Multiplications by A leads to $Aq_1^0, \ldots, Aq_n^0$;

Normalized Simultaneous iteration

1. Assume we have n pairwise orthogonal initial vectors $q_1^0, \ldots, q_n^0$;
2. Multiplications by A leads to $Aq_1^0, \ldots, Aq_n^0$;
3. In general, they are not pairwise orthogonal, so we re-orthogonalize them, *i.e.*, $[Aq_1^0| \ldots |Aq_n^0] = \overline{Q}_1 R_1 = [q_1^1| \ldots |q_n^1] R_1$;

Normalized Simultaneous iteration

1. Assume we have n pairwise orthogonal initial vectors $q_1^0, \dots, q_n^0$;
2. Multiplications by A leads to $Aq_1^0, \dots, Aq_n^0$;
3. In general, they are not pairwise orthogonal, so we re-orthogonalize them, *i.e.*, $[Aq_1^0| \dots |Aq_n^0] = \overline{Q}_1 R_1 = [q_1^1| \dots |q_n^1]R_1$;
4. We use the column vectors of $\overline{Q}_1$ as new pairwise orthogonal vectors.

Normalized Simultaneous iteration

1. Assume we have n pairwise orthogonal initial vectors $q_1^0, \ldots, q_n^0$;
2. Multiplications by A leads to $Aq_1^0, \ldots, Aq_n^0$;
3. In general, they are not pairwise orthogonal, so we re-orthogonalize them, *i.e.*, $[Aq_1^0|\ldots|Aq_n^0] = \overline{Q}_1 R_1 = [q_1^1|\ldots|q_n^1]R_1$;
4. We use the column vectors of $\overline{Q}_1$ as new pairwise orthogonal vectors.
5. Multiplications by A leads to $Aq_1^1, \ldots, Aq_n^1$;

Normalized Simultaneous iteration

1. Assume we have n pairwise orthogonal initial vectors $q_1^0, \ldots, q_n^0$;
2. Multiplications by A leads to $Aq_1^0, \ldots, Aq_n^0$;
3. In general, they are not pairwise orthogonal, so we re-orthogonalize them, *i.e.*, $[Aq_1^0|\ldots|Aq_n^0] = \overline{Q}_1 R_1 = [q_1^1|\ldots|q_n^1]R_1$;
4. We use the column vectors of $\overline{Q}_1$ as new pairwise orthogonal vectors.
5. Multiplications by A leads to $Aq_1^1, \ldots, Aq_n^1$;
6. Re-orthogonalization: $[Aq_1^1|\ldots|Aq_n^1] = \overline{Q}_2 R_2 = [q_1^2|\ldots|q_n^2]R_2$.

Normalized Simultaneous iteration

1. Assume we have n pairwise orthogonal initial vectors $q_1^0, \ldots, q_n^0$;
2. Multiplications by A leads to $Aq_1^0, \ldots, Aq_n^0$;
3. In general, they are not pairwise orthogonal, so we re-orthogonalize them, *i.e.*, $[Aq_1^0|\ldots|Aq_n^0] = \overline{Q}_1 R_1 = [q_1^1|\ldots|q_n^1]R_1$;
4. We use the column vectors of $\overline{Q}_1$ as new pairwise orthogonal vectors.
5. Multiplications by A leads to $Aq_1^1, \ldots, Aq_n^1$;
6. Re-orthogonalization: $[Aq_1^1|\ldots|Aq_n^1] = \overline{Q}_2 R_2 = [q_1^2|\ldots|q_n^2]R_2$.
7. Repeat Step 4 - 6 until $\overline{Q}_i$ is close to all eigenvectors of A.

Normalized Simultaneous Iteration

Algorithm 6: Normalized Simultaneous Iteration

$\overline{Q}_0 = I$

for $i = 0, 1, \ldots$ **do**

  $X_i = A\overline{Q}_i$

  $\overline{Q}_{i+1} R_{i+1} = X_i$

$\lambda = \text{diag}(\overline{Q}_{i+1} A \overline{Q}_{i+1}^\top)$

- The columns of $\overline{Q}_i$ are approximations to the eigenvectors of A.
- The diagonal elements of R_i $(r_{11}^i, \ldots, r_{nn}^i)$ are approximations to the eigenvalues.

Unshifted QR algorithm

Normalized simultaneous iteration (NSI):

1. $A\overline{Q}_0 = \overline{Q}_1 R_1$
2. $A\overline{Q}_1 = \overline{Q}_2 R_2$
3. $A\overline{Q}_2 = \overline{Q}_3 R_3$
4. $\vdots$

Unshifted QR algorithm

Normalized simultaneous iteration (NSI):

1. $A\overline{Q}_0 = \overline{Q}_1 R_1$
2. $A\overline{Q}_1 = \overline{Q}_2 R_2$
3. $A\overline{Q}_2 = \overline{Q}_3 R_3$
4. $\vdots$

Consider a similar iteration: $Q_0 = I$

1. $A_0 = A\overline{Q}_0 = Q_1 R_1'$
2. $A_1 = R_1' Q_1 = Q_2 R_2'$
3. $A_2 = R_2' Q_2 = Q_3 R_3'$
4. $\vdots$

The latter is called **unshifed QR algorithm**.

Unshifted QR algorithm

Algorithm 7: Unshifted QR algorithm

$Q_0 = I$

$A_0 = AQ_0$

for $i = 0, 1, \ldots$ **do**

  $Q_{i+1}R'_{i+1} = A_i$

  $A_{i+1} = R'_{i+1}Q_{i+1}$

$\lambda = \text{diag}(A_{i+1})$

Relation between Unshifted QR algorithm and NSI

- Let $Q_1 = \overline{Q}_1$ and $R_1 = R_1'$;

Relation between Unshifted QR algorithm and NSI

- Let $Q_1 = \overline{Q}_1$ and $R_1 = R_1'$;
- $\overline{Q}_2 R_2 = A\overline{Q}_1 = Q_1 R_1' Q_1 = Q_1 Q_2 R_2'$;

Relation between Unshifted QR algorithm and NSI

- Let $Q_1 = \overline{Q}_1$ and $R_1 = R'_1$;
- $\overline{Q}_2 R_2 = A\overline{Q}_1 = Q_1 R'_1 Q_1 = Q_1 Q_2 R'_2$;
- We chose $\overline{Q}_2 = Q_1 Q_2$ and $R_2 = R'_2$;

Relation between Unshifted QR algorithm and NSI

- Let $Q_1 = \overline{Q}_1$ and $R_1 = R'_1$;
- $\overline{Q}_2 R_2 = A\overline{Q}_1 = Q_1 R'_1 Q_1 = Q_1 Q_2 R'_2$;
- We chose $\overline{Q}_2 = Q_1 Q_2$ and $R_2 = R'_2$;
- Suppose that we define $\overline{Q}_k = Q_1 Q_2 \cdots Q_k$ and $R_k = R'_k$ for all $k < i$;
- We have $R_{k-1} Q_{k-1} = Q_k R_k$.

Relation between Unshifted QR algorithm and NSI

$$
\begin{aligned}
\overline{Q}_i R_i &= A\overline{Q}_{i-1} \\
&= A\,Q_1\,Q_2\,Q_3\,Q_4 \cdots Q_{i-1} \\
&= \overline{Q}_2\,R_2\,Q_2\,Q_3\,Q_4 \cdots Q_{i-1} \\
&= Q_1\,Q_2\,Q_3\,R_3\,Q_3\,Q_4 \cdots Q_{i-1} \\
&= Q_1\,Q_2\,Q_3\,Q_4\,R_4\,Q_4 \cdots Q_{i-1} \\
&= \vdots \\
&= Q_1 \cdots Q_i R_i
\end{aligned}
$$

The intuitive meaning of unshifted QR algorithm

Theorem

Let A and B be two similar matrices. Then they have the same set of eigenvalues.

The intuitive meaning of unshifted QR algorithm

Theorem

Let A and B be two similar matrices. Then they have the same set of eigenvalues.

- $A_{i-1} = Q_i R_i = Q_i R_i Q_i Q_i^\top = Q_i A_i Q_i^\top$;
- All A_i's are similar and have the same set of eigenvalues;
- As $i \to \infty$, A_i converges to a diagonal matrix;
- The eigenvalues of A are on the main diagonal of A_i;
- The column vectors of $Q_1 \cdots Q_i$ are the eigenvectors.

Convergence of unshifted QR algorithm

Theorem

Assume that A is a symmetric $n \times n$ matrix with eigenvalues λ_i s.t. $|\lambda_1| > |\lambda_2| > \cdots > |\lambda_n|$. The unshifted QR algorithm converges linearly to the eigenvectors and eigenvalues of A. As $j \to \infty$,

1. *A_j converges to a diagonal matrix containing the eigenvalues on the main diagonal.*
2. *$\overline{Q_j} = Q_1 \cdots Q_j$ converges to an orthogonal matrix whose columns are the eigenvectors.*

Outline

Shifted QR algorithm

- Unshifted QR algorithm:
  1. $A_0 = Q_1 R_1$;
  2. $A_1 = R_1 Q_1$;

Shifted QR algorithm

- Unshifted QR algorithm:
  1. $A_0 = Q_1 R_1$;
  2. $A_1 = R_1 Q_1$;
- Shifted QR algorithm:
  1. $A_0 - sI = Q_1 R_1$;
  2. $A_1 = R_1 Q_1 + sI$;

Shifted QR algorithm

- Unshifted QR algorithm:
  1. $A_0 = Q_1 R_1$;
  2. $A_1 = R_1 Q_1$;
- Shifted QR algorithm:
  1. $A_0 - sI = Q_1 R_1$;
  2. $A_1 = R_1 Q_1 + sI$;
  3. $A_1 - sI$

Shifted QR algorithm

- Unshifted QR algorithm:
  1. $A_0 = Q_1 R_1$;
  2. $A_1 = R_1 Q_1$;

- Shifted QR algorithm:
  1. $A_0 - sI = Q_1 R_1$;
  2. $A_1 = R_1 Q_1 + sI$;
  3. $A_1 - sI = R_1 Q_1$

Shifted QR algorithm

- Unshifted QR algorithm:
  1. $A_0 = Q_1 R_1$;
  2. $A_1 = R_1 Q_1$;
- Shifted QR algorithm:
  1. $A_0 - sI = Q_1 R_1$;
  2. $A_1 = R_1 Q_1 + sI$;
  3. $A_1 - sI = R_1 Q_1 = Q_1^\top (A_0 - sI) Q_1$

Shifted QR algorithm

- Unshifted QR algorithm:
  1. $A_0 = Q_1 R_1$;
  2. $A_1 = R_1 Q_1$;
- Shifted QR algorithm:
  1. $A_0 - sI = Q_1 R_1$;
  2. $A_1 = R_1 Q_1 + sI$;
  3. $A_1 - sI = R_1 Q_1 = Q_1^\top (A_0 - sI) Q_1 = Q_1^\top A_0 Q_1 - sI.$
- Repeating this step generates a sequence of A_k's which are similar to A_0.

Shifted QR algorithm

- Question: How to select a good shift s_k for each step?

Shifted QR algorithm

- Question: How to select a good shift s_k for each step?

- Answer: The bottom right entry of the matrix A_k.

Shifted QR algorithm

- Question: How to select a good shift s_k for each step?
- Answer: The bottom right entry of the matrix A_k.
- The reason:
  1. The iteration with this choice move the bottom row to a row of zeros, except for the bottom right entry.
  2. Obviously, the bottom right entry is one of the eigenvalue of A.

Shifted QR algorithm

- Question: How to select a good shift s_k for each step?
- Answer: The bottom right entry of the matrix A_k.
- The reason:
  1. The iteration with this choice move the bottom row to a row of zeros, except for the bottom right entry.
  2. Obviously, the bottom right entry is one of the eigenvalue of A.
- The procedure:
  1. After acquiring the eigenvalue, we deflate the matrix by eliminating the last row and column.
  2. We proceed to find another eigenvalue.

Shifted QR algorithm

Algorithm 8: Shifted QR algorithm

$Q^n = I$

$A^n = AQ^n - sI$

for $j = n, \ldots, 2$ **do**

  while $\Sigma_{k=1}^{j-1}|A_{jk}^j|$ *is not sufficiently small* **do**

    $s = A_{jj}^j$

    $Q^j R^j = A^j - sI$

    $A^j = R^j Q^j + sI$

  $\lambda_j = A_{jj}^j$

  let A^{j-1} be the matrix which results by eliminating the last column and row from A^j

$\lambda_1 = A_{11}^1$

Outline

Upper Hessenberg form

Definition

The $m \times n$ matrix A is **upper Hessenberg** if $a_{ij} = 0$ for $i > j + 1$.

Example

- $\begin{bmatrix} 1 & 3 & 4 & 2 \\ 2 & 2 & 5 & 2 \\ 0 & 3 & 2 & 4 \\ 0 & 0 & 4 & 5 \end{bmatrix}$ is upper Hessenberg.
- But $\begin{bmatrix} 1 & 3 & 4 & 2 \\ 2 & 2 & 5 & 2 \\ 3 & 3 & 2 & 4 \\ 0 & 0 & 4 & 5 \end{bmatrix}$ is not.

Upper Hessenberg form

- Before we apply the shifted QR algorithm, it is better to transform A to a similar matrix which is in upper Hessenberg form.
- The preprocess will increase the efficiency of the QR algorithm.

Upper Hessenberg form

- Before we apply the shifted QR algorithm, it is better to transform A to a similar matrix which is in upper Hessenberg form.
- The preprocess will increase the efficiency of the QR algorithm.

Theorem

Let A be a square matrix. There exist an orthogonal matrix Q and an upper Hessenberg matrix B s.t. $A = QBQ^\top$.

Upper Hessenberg form

- Before we apply the shifted QR algorithm, it is better to transform A to a similar matrix which is in upper Hessenberg form.
- The preprocess will increase the efficiency of the QR algorithm.

Theorem

Let A be a square matrix. There exist an orthogonal matrix Q and an upper Hessenberg matrix B s.t. $A = QBQ^\top$.

Via Householder reflection.

Upper Hessenberg form

Proof.

Given an $n \times n$ matrix $A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$,

1. Let $x_1 = \begin{bmatrix} a_{21} & a_{31} & \cdots & a_{n1} \end{bmatrix}^\top$;

Upper Hessenberg form

Proof.

Given an $n \times n$ matrix $A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$,

1. Let $x_1 = \begin{bmatrix} a_{21} & a_{31} & \cdots & a_{n1} \end{bmatrix}^\top$;
2. Let $w_1 = \begin{bmatrix} \text{sgn}(x_{11})\|x_1\|_2 & 0 & \cdots & 0 \end{bmatrix}^\top$;

Upper Hessenberg form

Proof.

Given an $n \times n$ matrix $A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$,

1. Let $x_1 = \begin{bmatrix} a_{21} & a_{31} & \cdots & a_{n1} \end{bmatrix}^\top$;
2. Let $w_1 = \begin{bmatrix} \text{sgn}(x_{11})\|x_1\|_2 & 0 & \cdots & 0 \end{bmatrix}^\top$;
3. So $u_1 = w_1 - x_1$, $v_1 = \frac{u_1}{\|u_1\|_2}$ and $\hat{H}_1 = I - 2v_1 v_1^\top$;

Upper Hessenberg form

Proof.

Given an $n \times n$ matrix $A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$,

1. Let $x_1 = \begin{bmatrix} a_{21} & a_{31} & \cdots & a_{n1} \end{bmatrix}^\top$;
2. Let $w_1 = \begin{bmatrix} \text{sgn}(x_{11})\|x_1\|_2 & 0 & \cdots & 0 \end{bmatrix}^\top$;
3. So $u_1 = w_1 - x_1$, $v_1 = \frac{u_1}{\|u_1\|_2}$ and $\hat{H}_1 = I - 2v_1 v_1^\top$;
4. $H_1 = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & & & \\ \vdots & & \hat{H}_1 & \\ 0 & & & \end{bmatrix}$

Upper Hessenberg form

Proof.

1. $$C = H_1 A = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ 0 & c_{32} & \cdots & c_{3n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & c_{n2} & \cdots & c_{nn} \end{bmatrix}$$

2. $$D = CH_1 = C \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & & & \\ \vdots & & \hat{H}_1 & \\ 0 & & & \end{bmatrix} = \begin{bmatrix} d_{11} & d_{12} & \cdots & d_{1n} \\ d_{21} & d_{22} & \cdots & d_{2n} \\ 0 & d_{32} & \cdots & d_{3n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & d_{n2} & \cdots & d_{nn} \end{bmatrix}.$$

Upper Hessenberg form

Proof.

1. Let $x_2 = \begin{bmatrix} d_{32} & d_{33} & \cdots & d_{n3} \end{bmatrix}^\top$;
2. Let $w_2 = \begin{bmatrix} \mathrm{sgn}(x_{21})\|x_2\|_2 & 0 & \cdots & 0 \end{bmatrix}^\top$;
3. So $u_2 = w_2 - x_2$, $v_2 = \frac{u_2}{\|u_2\|_2}$ and $\hat{H}_2 = I - 2v_2v_2^\top$.
4. $H_2 = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & & & \\ \vdots & \vdots & & \hat{H}_2 & \\ 0 & 0 & & & \end{bmatrix}$.

Upper Hessenberg form

Proof.

1. $$E = H_2 D = \begin{bmatrix} e_{11} & e_{12} & \cdots & e_{1n} \\ e_{21} & e_{22} & \cdots & e_{2n} \\ 0 & e_{32} & \cdots & e_{3n} \\ 0 & 0 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & e_{nn} \end{bmatrix}$$

2. $$F = EH_2 = E \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & & & \\ \vdots & \vdots & & \hat{H}_2 & \\ 0 & 0 & & & \end{bmatrix} = \begin{bmatrix} f_{11} & f_{12} & \cdots & f_{1n} \\ f_{21} & f_{22} & \cdots & f_{2n} \\ 0 & f_{32} & \cdots & f_{3n} \\ 0 & 0 & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & f_{nn} \end{bmatrix}.$$

Upper Hessenberg form

- Following the above process, we can construct $n-1$ Householder reflector $H_1, \ldots, H_{n-1}$ for A.
- Let $B = H_{n-1} \cdots H_1 A H_1 \cdots H_{n-1}$.
- Obviously, B is upper Hessenberg.
- Since Householder reflector is orthogonal and symmetric, so $B = H_{n-1} \cdots H_1 A (H_{n-1} \cdots H_1)^\top$.
- Hence, $A = QBQ^\top$ where $Q = H_{n-1} \cdots H_1$.

Outline

Singular values and vectors

Definition

Let A be an $m \times n$ matrix, $U : \{u_1, \cdots, u_m\}$ and $V : \{v_1, \cdots, v_n\}$ two orthonormal sets, and $s_1 \cdots, s_n$ s.t. $Av_i = s_i u_i$ for $1 \leq i \leq n$. Then,

- v_i is called the **right singular vector** of A;
- v_i is called the **left singular vector** of A;
- s_i is called the **singular value** of A.

$USV^\top$ is the **singular value decomposition** (SVD) of A where S is the diagonal $m \times n$ matrix whose diagonal entries are s_i's.

Singular values and vectors

Example

- $A = \begin{bmatrix} 0 & 1 \\ 0 & -1 \end{bmatrix}$;
- $U = \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \end{bmatrix}$ where $u_1 = \begin{bmatrix} \frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} \end{bmatrix}$ and $u_2 = \begin{bmatrix} \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \end{bmatrix}$;
- $S = \begin{bmatrix} \sqrt{2} & 0 \\ 0 & 0 \end{bmatrix}$ where $s_1 = \sqrt{2}$ and $s_2 = 0$;
- $V = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ where $v_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ and $v_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$;
- $A = USV^\top$.

Finding the SVD

Suppose that $m \leq n$. Given that

- $\lambda_1, \lambda_2, \cdots, \lambda_n$: the set of eigenvalues of $A^\top A$ where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$;
- $v_1, v_2, \cdots, v_n$: the set of unit vectors where v_i is the eigenvector of $A^\top A$ w.r.t. λ_i.

Finding the SVD

Suppose that $m \leq n$. Given that

- $\lambda_1, \lambda_2, \cdots, \lambda_n$: the set of eigenvalues of $A^\top A$ where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$;
- $v_1, v_2, \cdots, v_n$: the set of unit vectors where v_i is the eigenvector of $A^\top A$ w.r.t. λ_i.

s_i's and u_i's (for $1 \leq i \leq m$) are computed as follows:

- $s_i = \sqrt{\lambda_i}$;
- $u_i = \begin{cases} \frac{Av_i}{s_i} & \text{if } \lambda_i \neq 0; \\ \text{an unit vector orthogonal to } u_1, u_2, \cdots, u_{i-1} & \text{otherwise.} \end{cases}$

Finding the SVD

Lemma

Let A be an $m \times n$ matrix. The eigenvalues of $A^\top A$ and $AA^\top$ are nonnegative.

Finding the SVD

Lemma

Let A be an $m \times n$ matrix. The eigenvalues of $A^\top A$ and $AA^\top$ are nonnegative.

Proof.

Let v be a unit eigenvector of $A^\top A$ and $A^\top Av = \lambda v$.
Then, $0 \leq \|Av\|^2 = v^\top A^\top Av = \lambda v^\top v = \lambda$.

Finding the SVD

Proposition

$AA^\top$ and $A^\top A$ are symmetric.

Finding the SVD

Proposition

$AA^\top$ *and* $A^\top A$ *are symmetric.*

Proof.

$(AA^\top)^\top = (A^\top)^\top A^\top = AA^\top$.

Similarly, $(A^\top A)^\top = A^\top A$.

Finding the SVD

Proposition

$u_1, u_2, \cdots, u_m$ *forms an orthonormal set of eigenvectors of* $AA^\top$.

Finding the SVD

Proposition

$u_1, u_2, \cdots, u_m$ *forms an orthonormal set of eigenvectors of* $AA^\top$.

Proof.

Unitary: $u_i^\top u_i = \frac{Av_i}{s_i}^\top \frac{Av_i}{s_i} = \frac{v_i^\top A^\top A v_i}{s_i^2} = \frac{\lambda_i v_i^\top v_i}{\lambda_i} = 1.$

Finding the SVD

Proposition

$u_1, u_2, \cdots, u_m$ *forms an orthonormal set of eigenvectors of* $AA^\top$.

Proof.

Unitary: $u_i^\top u_i = \frac{Av_i}{s_i}^\top \frac{Av_i}{s_i} = \frac{v_i^\top A^\top A v_i}{s_i^2} = \frac{\lambda_i v_i^\top v_i}{\lambda_i} = 1.$

Orthogonality: $u_i^\top u_j = \frac{Av_i}{s_i}^\top \frac{Av_j}{s_j} = \frac{v_i^\top A^\top A v_j}{s_i s_j} = \frac{\lambda v_i^\top v_j}{s_i s_j} = 0$ for $i \neq j$.

Finding the SVD

Proposition

$u_1, u_2, \cdots, u_m$ *forms an orthonormal set of eigenvectors of* $AA^\top$.

Proof.

Unitary: $u_i^\top u_i = \frac{Av_i}{s_i}^\top \frac{Av_i}{s_i} = \frac{v_i^\top A^\top A v_i}{s_i^2} = \frac{\lambda_i v_i^\top v_i}{\lambda_i} = 1.$

Orthogonality: $u_i^\top u_j = \frac{Av_i}{s_i}^\top \frac{Av_j}{s_j} = \frac{v_i^\top A^\top A v_j}{s_i s_j} = \frac{\lambda v_i^\top v_j}{s_i s_j} = 0$ for $i \neq j$.

Eigenvector: $AA^\top u_i = \frac{AA^\top A v_i}{s_i} = s_i^2 \frac{Av_i}{s_i} = \lambda_i u_i.$

Finding the SVD

Theorem

Let A be an $m \times n$ matrix. There are two orthonormal bases $\{v_1, \cdots, v_n\}$ of R^n and $\{u_1, \cdots, u_m\}$ of R^m, and real numbers $s_1 \geq s_2 \geq \cdots \geq 0$ s.t. $Av_i = s_i u_i$ for $1 \leq i \leq \min\{m, n\}$.

Finding the SVD: an alternative way

- $B = \begin{bmatrix} 0 & A^\top \\ A & 0 \end{bmatrix}$: an $(m+n)\times(m+n)$ matrix;
- λ: an eigenvalue of B;
- $\begin{bmatrix} v \\ w \end{bmatrix}$: an eigenvector of B w.r.t. λ;
- $\begin{bmatrix} A^\top w \\ Av \end{bmatrix} = \begin{bmatrix} 0 & A^\top \\ A & 0 \end{bmatrix} \begin{bmatrix} v \\ w \end{bmatrix} = \lambda \begin{bmatrix} v \\ w \end{bmatrix}$
- $Av = \lambda w \Rightarrow A^\top A v = \lambda A^\top w = \lambda^2 v$.
- v is an eigenvector of $A^\top A$ w.r.t. λ^2.
- SVD of A is a matter of finding eigenvalues and eigenvectos of $\begin{bmatrix} 0 & A^\top \\ A & 0 \end{bmatrix}$.

Special case: symmetric matrices

- The eigenvectors of a symmetric matrix is a orthonormal set;
- Finding the SVD of a symmetric matrix is a matter of finding the eigenvalue and vectors.

Special case: symmetric matrices

- The eigenvectors of a symmetric matrix is a orthonormal set;
- Finding the SVD of a symmetric matrix is a matter of finding the eigenvalue and vectors.
- λ_i: the eigenvalues satisfying $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_m|$;
- v_i: the corresponding eigenvector w.r.t. λ_i;

Special case: symmetric matrices

- The eigenvectors of a symmetric matrix is a orthonormal set;
- Finding the SVD of a symmetric matrix is a matter of finding the eigenvalue and vectors.
- λ_i: the eigenvalues satisfying $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_m|$;
- v_i: the corresponding eigenvector w.r.t. λ_i;
- $s_i = |\lambda_i|$;
- $u_i = \begin{cases} v_i, & \text{if } \lambda_i \geq 0; \\ -v_i, & \text{otherwise.} \end{cases}$

Properties of the SVD

Definition

The **rank** of an $m \times n$ matrix A is the number of linearly independent rows (columns).

Proposition

The rank of the matrix $A = USV^{\top}$ *is the number of nonzero entries in* S*.*

Properties of the SVD

Definition

The **rank** of an $m \times n$ matrix A is the number of linearly independent rows (columns).

Proposition

The rank of the matrix $A = USV^\top$ is the number of nonzero entries in S.

Proof.

U and V are orthogonal, and hence invertible.
$\texttt{rank}(A) = \texttt{rank}(S)$, which is the number of nonzero entries in S.

Properties of the SVD

Proposition

If A is an $n \times n$ matrix, $|\det(A)| = s_1 \cdots s_n$.

Properties of the SVD

Proposition

If A is an $n \times n$ matrix, $|\det(A)| = s_1 \cdots s_n$.

Proof.

Since $U^\top U = I$, $\det(U) = 1$ or $\det(U) = -1$.

Properties of the SVD

Proposition

If A is an $n \times n$ matrix, $|\det(A)| = s_1 \cdots s_n$.

Proof.

Since $U^\top U = I$, $\det(U) = 1$ or $\det(U) = -1$.
Similarly, $\det(V^\top) = \det(V) = 1$ or $\det(V^\top) = \det(V) = -1$.

Properties of the SVD

Proposition

If A is an $n \times n$ matrix, $|\det(A)| = s_1 \cdots s_n$.

Proof.

Since $U^\top U = I$, $\det(U) = 1$ or $\det(U) = -1$.
Similarly, $\det(V^\top) = \det(V) = 1$ or $\det(V^\top) = \det(V) = -1$.
$|\det(A)| = |\det(USV^\top)| = |\det(U)| \cdot |\det(S)| \cdot |\det(V^\top)| = s_1 \cdots s_n$.

Properties of the SVD

Proposition

If A is an invertible $n \times n$ matrix, then $A^{-1} = VS^{-1}U^{\top}$.

Properties of the SVD

Proposition

If A is an invertible $n \times n$ matrix, then $A^{-1} = VS^{-1}U^{\top}$.

Proof.

We can get that S is also invertible.

Properties of the SVD

Proposition

If A is an invertible $n \times n$ matrix, then $A^{-1} = VS^{-1}U^{\top}$.

Proof.

We can get that S is also invertible.
By definition of the SVD, U and $V^{\top}$ is also invertible.

Properties of the SVD

Proposition

If A is an invertible $n \times n$ matrix, then $A^{-1} = VS^{-1}U^{\top}$.

Proof.

We can get that S is also invertible.
By definition of the SVD, U and $V^{\top}$ is also invertible.
Further, $U^{-1} = U^{\top}$ and $\left(V^{\top}\right)^{-1} = V$.

Properties of the SVD

Proposition

If A is an invertible $n \times n$ matrix, then $A^{-1} = VS^{-1}U^{\top}$.

Proof.

We can get that S is also invertible.
By definition of the SVD, U and $V^{\top}$ is also invertible.
Further, $U^{-1} = U^{\top}$ and $\left(V^{\top}\right)^{-1} = V$.
Finally, $A^{-1} = \left(USV^{\top}\right)^{-1} = \left(V^{\top}\right)^{-1} S^{-1} U^{-1} = VS^{-1}U^{\top}$.

Properties of the SVD

Proposition

The $m \times n$ matrix A can be written as the sum of rank-1 matrices, i.e.,

$$A = \sum_{i=1}^{r} s_i u_i v_i^\top,$$

where r is the rank of A, and u_i and v_i are the i-th columns of U and V, respectively.

Properties of the SVD

Proof.

$$
\begin{aligned}
A &= USV^\top = U \begin{bmatrix} s_1 & & \\ & \ddots & \\ & & s_r \\ & & \end{bmatrix} V^\top \\
&= U\left(\begin{bmatrix} s_1 & & \\ & & \\ & & \end{bmatrix} + \begin{bmatrix} & & \\ & s_2 & \\ & & \end{bmatrix} + \cdots + \begin{bmatrix} & & \\ & & \\ & & s_r \end{bmatrix} \right) V^\top \\
&= s_1 u_1 v_1^\top + s_r u_r v_r^\top + \cdots + s_r u_r v_r^\top
\end{aligned}
$$

Dimension reduction

- $A = \sum_{i=1}^{r} s_i u_i v_i^\top$ by Property 4.
- The matrix A can be represented by three components: s_i's, u_i's and v_i's.

Dimension reduction

- $A = \sum_{i=1}^{r} s_i u_i v_i^\top$ by Property 4.
- The matrix A can be represented by three components: s_i's, u_i's and v_i's.
- The rank-p approximation: $A = \sum_{i=1}^{p} s_i u_i v_i^\top$ where $p < r$.

Dimension reduction

Example

- $A = \begin{bmatrix} 3 & 2 & -2 & -3 \\ 2 & 4 & -1 & -5 \end{bmatrix}$
- $U = \begin{bmatrix} 0.5886 & -0.8084 \\ 0.8084 & 0.5886 \end{bmatrix}$
- $S = \begin{bmatrix} 8.2809 & 0 & 0 & 0 \\ 0 & 1.8512 & 0 & 0 \end{bmatrix}$
- $V^{\top} = \begin{bmatrix} 0.4085 & 0.5327 & -0.2398 & -0.7014 \\ -0.6741 & 0.3985 & 0.5554 & -0.2798 \\ 0.5743 & -0.1892 & 0.7924 & -0.0801 \\ 0.2212 & 0.7223 & 0.0780 & 0.6507 \end{bmatrix}$

The rank-1 approximation

Example

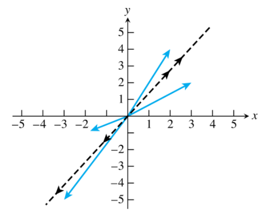


Figure: Dimension reduction by SVD

- $u_1 = \begin{bmatrix} 0.5886 \\ 0.8084 \end{bmatrix}$
- $s_1 = 8.2809;$
- ${v_1}^\top = \begin{bmatrix} 0.4085 & 0.5327 & -0.2398 & -0.7014 \end{bmatrix}$
- $A \approx u_1 s_1 {v_1}^\top = \begin{bmatrix} 1.9912 & 2.5964 & -1.1689 & -3.4188 \\ 2.7346 & 3.5657 & -1.6052 & -4.6951 \end{bmatrix}.$

Compression

- An $n \times n$ matrix needs $n \times n$ numbers.
- By property 4, the rank-p approximation needs p n-dimensional u_i's, n-dimensional v_i's and s_i's (total $2pn + p$ numbers).
- When $p < n$, some storage space can be saved.

Compression

- An $n \times n$ matrix needs $n \times n$ numbers.
- By property 4, the rank-p approximation needs p n-dimensional u_i's, n-dimensional v_i's and s_i's (total $2pn + p$ numbers).
- When $p < n$, some storage space can be saved.

Example

Let $n = 256$.

Approximation	Numbers of values	Compression rate
Original	$65,536$	1
Rank-8	$4,104$	$\approx \frac{1}{16}$
Rank-16	$8,208$	$\approx \frac{1}{8}$
Rank-32	$16,416$	$\approx \frac{1}{4}$

Compression

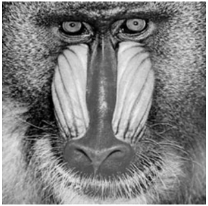


Figure: The original photo

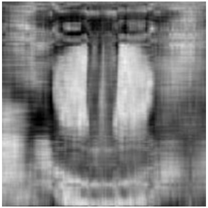


Figure: Rank-8 approximation

Compression

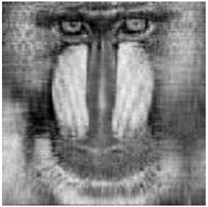


Figure: Rank-16 approximation

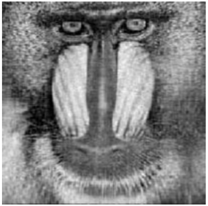


Figure: Rank-32 approximation

Conclusions

- Finding the dominant eigenvalue and eigenvector:
  1. Power Iteration
  2. Rayleigh Quotient Iteration
  3. Applying Power Iteration to PageRank
- Finding some eigenvalue and eigenvector given a shift s
  1. Inverse Power Iteration

Conclusions

- Finding all eigenvalues and eigenvectors
  1. Normalized Simultaneous Iteration
  2. Unshifted QR Algorithm
- Two improvements to Unshifted QR Algorithm
  1. Shifted QR Algorithm
  2. First put the matrix into upper Hessenberg form
- Singular Value Decomposition
  1. Finding the SVD
  2. Dimension Reduction and Compression via SVD

Thank you!