Chapter 8: Eigenvalues and Singular Values

In this chapter we consider algorithms for solving two problems.

Chapter 8: Eigenvalues and Singular Values

In this chapter we consider algorithms for solving two problems.

- The eigenvalue problem: $A\vec{x} = \lambda\vec{x}$.

Chapter 8: Eigenvalues and Singular Values

In this chapter we consider algorithms for solving two problems.

- The eigenvalue problem: $A\vec{x} = \lambda\vec{x}$.
- The singular value decomposition: $A = U\Sigma V^T$.

Chapter 8: Eigenvalues and Singular Values

Example (eigenvalue problem): Find the eigenvalues, eigenvectors and diagonalizing matrix S, for $A \begin{bmatrix} 7 & 2 \\ -15 & -4 \end{bmatrix}$.

Chapter 8: Eigenvalues and Singular Values

Example (eigenvalue problem for symmetric, positive definite matrices): Find the eigenvalues, eigenvectors and diagonalizing matrix Q, for $A \begin{bmatrix} 3 & 1 \\ 1 & 3 \end{bmatrix}$.

Chapter 8: Eigenvalues and Singular Values

Example (eigenvalue problem for symmetric, positive definite matrices): Find the eigenvalues, eigenvectors and diagonalizing matrix Q, for $A \begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix}$.

Chapter 8: Eigenvalues and Singular Values

An important matrix factorization is the famous SVD, $A = U\Sigma V^T$. It joins our other important factorizations: $PA = LU$ (Guassian Elimination), $A = QR$ (Gram-Schmidt), $A = S\Lambda S^{-1}$, and, when A is symmetric $A = Q\Lambda Q^T$.

Chapter 8: Eigenvalues and Singular Values

Chapter 8: Eigenvalues and Singular Values

- $A = U\Sigma V^T =$ (orthogonal)(diagonal)(orthogonal) can be written as $AV = U\Sigma$.

Chapter 8: Eigenvalues and Singular Values

- $A = U\Sigma V^T =$ (orthogonal)(diagonal)(orthogonal) can be written as $AV = U\Sigma$.
- The r singular values on the diagonal

$$\Sigma = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_r \end{bmatrix}$$

when $\sigma_1 \geq \sigma_2 \geq \ldots \sigma_r \geq 0$ are the square roots of the eigenvalues of both A^TA and AA^T.

Chapter 8: Eigenvalues and Singular Values

- $A = U\Sigma V^T =$ (orthogonal)(diagonal)(orthogonal) can be written as $AV = U\Sigma$.
- The r singular values on the diagonal

$$\Sigma = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_r \end{bmatrix}$$

when $\sigma_1 \geq \sigma_2 \geq \ldots \sigma_r \geq 0$ are the square roots of the eigenvalues of both $A^T A$ and AA^T.

- The columns of the $m \times m$ matrix U are eigenvectors of AA^T.

Chapter 8: Eigenvalues and Singular Values

- $A = U\Sigma V^T =$ (orthogonal)(diagonal)(orthogonal) can be written as $AV = U\Sigma$.
- The r singular values on the diagonal

$$\Sigma = \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_r \end{bmatrix}$$

  when $\sigma_1 \geq \sigma_2 \geq \ldots \sigma_r \geq 0$ are the square roots of the eigenvalues of both $A^T A$ and AA^T.
- The columns of the $m \times m$ matrix U are eigenvectors of AA^T.
- The columns of the $n \times n$ matrix V are eigenvectors of $A^T A$.

The SVD chooses the bases U and V in a special way. They are more than just orthonormal. When A multiplies a column v_i of V, it produces σ_j times a column of U. That comes directly from $AV = U\Sigma$, looked at a column at a time.

Chapter 8: Eigenvalues and Singular Values

Eigenvalues and singular values generally cannot be computed precisely in a finite number of steps, even in the absence of floating point error. All algorithms for computing eigenvalues and singular values are therefore necessarily iterative, unlike the algorithms introduced in chapters 5 and 6.

Chapter 8: Eigenvalues and Singular Values

Methods for finding eigenvalues can be split into two categories.

Chapter 8: Eigenvalues and Singular Values

Methods for finding eigenvalues can be split into two categories.

- Algorithms using decompositions involving similarity transformations for finding several or all eigenvalues.

Chapter 8: Eigenvalues and Singular Values

Methods for finding eigenvalues can be split into two categories.

- Algorithms using decompositions involving similarity transformations for finding several or all eigenvalues.
- Algorithms based on matrix-vector products to find just a few of the eigenvalues.

Chapter 8: Eigenvalues and Singular Values

Methods for finding eigenvalues can be split into two categories.

- Algorithms using decompositions involving similarity transformations for finding several or all eigenvalues.
- Algorithms based on matrix-vector products to find just a few of the eigenvalues.
- You will only be tested on second category of methods, the methods related to the power method described below.

8.1 The power method and variants

The power method is based on repeated multiplication of the $n \times n$ square matrix A on a random vector $\vec{x}_0$ (almost any initial vector $\vec{x}_0$ will do).

$$\vec{x}_0 = c_1 \vec{v}_1 + c_1 \vec{v}_1 + \ldots + c_1 \vec{v}_1$$

when $(v_1, v_2, \ldots, v_n)$ is a basis of $\mathbb{R}^n$ of eigenvectors of A.

8.1 The power method and variants

The power method is based on repeated multiplication of the $n \times n$ square matrix A on a random vector $\vec{x}_0$ (almost any initial vector $\vec{x}_0$ will do). The resulting FPI

$$\vec{x}_{k+1} = g(\vec{x}_k) = A\vec{x}_k$$

gravitates towards the direction of the dominant eigenvector.

8.1 Example of the dominant eigenvector: Google's PageRank

$$x_i = \sum_{j \in B_i} x_j$$

for $i = 1, \ldots, n$.

8.1 Example of the dominant eigenvector: Google's PageRank

$$x_i = \sum_{j \in B_i} x_j$$

for $i = 1, \ldots, n$.

- Given a network linkage graph with n nodes (page content is overlooked in this view).

8.1 Example of the dominant eigenvector: Google's PageRank

$$x_i = \sum_{j \in B_i} x_j$$

for $i = 1, \ldots, n$.

- Given a network linkage graph with n nodes (page content is overlooked in this view).
- Importance or *rank* of ith page is x_i.

8.1 Example of the dominant eigenvector: Google's PageRank

$$x_i = \sum_{j \in B_i} x_j$$

for $i = 1, \ldots, n$.

- Given a network linkage graph with n nodes (page content is overlooked in this view).
- Importance or *rank* of ith page is x_i.
- N_j is the number of pages pointing to page j with rank x_j.

8.1 Example of the dominant eigenvector: Google's PageRank

$$x_i = \sum_{j \in B_i} x_j$$

for $i = 1, \ldots, n$.

8.1 Example of the dominant eigenvector: Google's PageRank

$$x_i = \sum_{j \in B_i} x_j$$

for $i = 1, \ldots, n$.

- Looking carefully at this sum gives $x = Ax$, an eigenvalue $\lambda = 1$ problem.

8.1 Example of the dominant eigenvector: Google's PageRank

$$x_i = \sum_{j \in B_i} x_j$$

for $i = 1, \dots, n$.

- Looking carefully at this sum gives $x = Ax$, an eigenvalue $\lambda = 1$ problem.
- The entries a_{ij} of A are the elements $\frac{1}{N_j}$ associated with page i.

8.1 Example of the dominant eigenvector: Google's PageRank

$$x_i = \sum_{j \in B_i} x_j$$

for $i = 1, \ldots, n$.

- Looking carefully at this sum gives $x = Ax$, an eigenvalue $\lambda = 1$ problem.
- The entries a_{ij} of A are the elements $\frac{1}{N_j}$ associated with page i.
- Since the number of links in and out of a given page is tiny compared with the total number of webpages, A is extremely sparse.

8.1 Example of the dominant eigenvector: Google's PageRank

8.1 Example of the dominant eigenvector: Google's PageRank

- Example: mini-web with three sites.

8.1 Example of the dominant eigenvector: Google's PageRank

- Example: mini-web with three sites.
- Transition matrix $A = \begin{bmatrix} 0.7 & 0.1 & 0.2 \\ 0.2 & 0.4 & 0.2 \\ 0.1 & 0.5 & 0.6 \end{bmatrix}$.

8.1 Example of the dominant eigenvector: Google's PageRank

- Example: mini-web with three sites.
- Transition matrix $A = \begin{bmatrix} 0.7 & 0.1 & 0.2 \\ 0.2 & 0.4 & 0.2 \\ 0.1 & 0.5 & 0.6 \end{bmatrix}$.
- Eigenvalues: $\lambda_1 = 1, \lambda_2 = 0.5, \lambda_3 = 0.2$.

8.1 Example of the dominant eigenvector: Google's PageRank

- Example: mini-web with three sites.
- Transition matrix $A = \begin{bmatrix} 0.7 & 0.1 & 0.2 \\ 0.2 & 0.4 & 0.2 \\ 0.1 & 0.5 & 0.6 \end{bmatrix}$.
- Eigenvalues: $\lambda_1 = 1, \lambda_2 = 0.5, \lambda_3 = 0.2$.
- Corresponding eigenvectors:
  $v_1 = \begin{bmatrix} 7 \\ 5 \\ 8 \end{bmatrix}, v_2 = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}, v_3 = \begin{bmatrix} -1 \\ -3 \\ 4 \end{bmatrix}$

8.1 Example of the dominant eigenvector: Google's PageRank

8.1 Example of the dominant eigenvector: Google's PageRank

- Write the intial distribution vector $x_0 \begin{bmatrix} \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \end{bmatrix}$ in terms of the eigenbasis.

8.1 Example of the dominant eigenvector: Google's PageRank

- Write the intial distribution vector $x_0 \begin{bmatrix} \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \end{bmatrix}$ in terms of the eigenbasis.
- $x_0 = c_1 v_1 + c_2 v_2 + c_3 v_3$ to find $c_1 = \frac{1}{20}, c_2 = \frac{-2}{45}, c_3 = \frac{-1}{36}$.

8.1 Example of the dominant eigenvector: Google's PageRank

- Write the intial distribution vector $x_0 \begin{bmatrix} \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \end{bmatrix}$ in terms of the eigenbasis.
- $x_0 = c_1 v_1 + c_2 v_2 + c_3 v_3$ to find $c_1 = \frac{1}{20}, c_2 = \frac{-2}{45}, c_3 = \frac{-1}{36}$.
- Now iterate (FPI): $x_k = A^k x_0 = \sum_1^3 c_i \lambda_i^k v_i$.

8.1 Example of the dominant eigenvector: Google's PageRank

- Write the intial distribution vector x_0 $\begin{bmatrix} \frac{1}{3} \\ \frac{1}{3} \\ \frac{1}{3} \end{bmatrix}$ in terms of the eigenbasis.
- $x_0 = c_1 v_1 + c_2 v_2 + c_3 v_3$ to find $c_1 = \frac{1}{20}, c_2 = \frac{-2}{45}, c_3 = \frac{-1}{36}$.
- Now iterate (FPI): $x_k = A^k x_0 = \sum_1^3 c_i \lambda_i^k v_i$.
- $\lim_{k\to\infty} x_k = c_1 v_1 = \frac{1}{20} \begin{bmatrix} 7 \\ 5 \\ 8 \end{bmatrix} = \begin{bmatrix} 35\% \\ 25\% \\ 40\% \end{bmatrix}$.

8.1 The power method and variants

The power method is based on repeated multiplication of the $n \times n$ square matrix A on a random vector $\vec{x}_0$ (almost any initial vector $\vec{x}_0$ will do). The resulting FPI

$$\vec{x}_{k+1} = g(\vec{x}_k) = A\vec{x}_k$$

gravitates towards the direction of the dominant eigenvector.

8.1 The power method and variants

The power method:

```
#ALGORITHM: Power Method p. 222.
import numpy as np

# matrix A.  Looking for dominant eigenvector v so that Av
A = np.array([[7,4],
              [3,6]])

v = np.array([1,1])  # initial guess for dominant eigenvect

for k in range(20):
    v = A@v
    v = v / np.linalg.norm(v)
    lam = np.dot(v,A@v)
    print(k+1, v, lam)
```

8.1 The power method and variants

In order to understand the power method focus on the code portion when $v_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$:

```
for k in range(20):
    v = A@v
```

8.1 The power method and variants

In order to understand the power method focus on the code portion when $v_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$:

```
for k in range(20):
    v = A@v
```

- You should compute by hand $v_1 = \begin{bmatrix} 11 \\ 9 \end{bmatrix}$ and $v_2 = \begin{bmatrix} 113 \\ 87 \end{bmatrix}$.

8.1 The power method and variants

In order to understand the power method focus on the code portion when $v_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$:

```
for k in range(20):
    v = A@v
```

- You should compute by hand $v_1 = \begin{bmatrix} 11 \\ 9 \end{bmatrix}$ and $v_2 = \begin{bmatrix} 113 \\ 87 \end{bmatrix}$.
- Then use a computer to generate more iterations noting that $\frac{87}{113} = .77 \approx .75$.

8.1 The power method and variants

Assume that the $n \times n$ matrix A has n linearly independent eigenvectors then the power method works since for any initial guess v_0

8.1 The power method and variants

Assume that the $n \times n$ matrix A has n linearly independent eigenvectors then the power method works since for any initial guess v_0

- we can write v_0 as a linear combination of the eigenvectors $v_0 = \sum_{j=1}^{n} \beta_j x_j$.

8.1 The power method and variants

Assume that the $n \times n$ matrix A has n linearly independent eigenvectors then the power method works since for any initial guess v_0

- we can write v_0 as a linear combination of the eigenvectors $v_0 = \sum_{j=1}^{n} \beta_j x_j$.
- Then repeated multiplication by A is easily computed as $A^k v_0 = \sum_{j=1}^{n} \beta_j \lambda_j^k x_j$ and the dominant eigenvalue will dominate the sum as $k \to \infty$.

8.1 The power method and variants

Assume that the $n \times n$ matrix A has n linearly independent eigenvectors then the power method works since for any initial guess v_0

- we can write v_0 as a linear combination of the eigenvectors $v_0 = \sum_{j=1}^{n} \beta_j x_j$.
- Then repeated multiplication by A is easily computed as $A^k v_0 = \sum_{j=1}^{n} \beta_j \lambda_j^k x_j$ and the dominant eigenvalue will dominate the sum as $k \to \infty$.
- The code

  ```
  v = v / np.linalg.norm(v)
  lam = np.dot(v,A@v)
  ```

  is introduced so that we do have our computers store massive numbers and risk roundoff errors.

8.1 The power method and variants

The inverse power method is a clever twist on the power method:

```
#ALGORITHM: Inverse Iteration p. 228 when alpha = 0 to fin
import numpy as np

# matrix A.  Looking for minimum eigenvector v so that Av =
A = np.array([[7,4],
              [3,6]])

v = np.array([1,1])  # initial guess for dominant eigenvec

for k in range(20):
    v = np.linalg.solve(A,v)
    v = v / np.linalg.norm(v)
    lam = np.dot(v,A@v)
    print(k+1, v, lam)
```

8.1 The power method and variants

For an eigenvalue that is not greatest or lowest but you have a good approximation.

```
#ALGORITHM: Inverse Iteration p. 228 for approximate alpha
import numpy as np

# matrix A.  Looking for dominant eigenvector v so that Av
A = np.array([[0.7, 0.1, 0.2],
              [0.2, 0.4, 0.2],
              [0.1, 0.5, 0.6]])

v = np.array([1,0,0])  # initial guess for dominant eigenve
alpha = 0.44  # eigenvalue approximate
for k in range(20):
    v = np.linalg.solve(A - alpha*eye(3),v)
    v = v / np.linalg.norm(v)
    lam = np.dot(v,A@v)
    print(k+1, v, lam)
```

8.1 The power method and variants

Here is the way to use professional code to find the eigenvectors and eigenvalues in numpy of a square matrix:

8.1 The power method and variants

Here is the way to use professional code to find the eigenvectors and eigenvalues in numpy of a square matrix:

- `>>> a = np.array([[4, 2, 0], [9, 3, 7], [1, 2, 1]], flo`

8.1 The power method and variants

Here is the way to use professional code to find the eigenvectors and eigenvalues in numpy of a square matrix:

- `>>> a = np.array([[4, 2, 0], [9, 3, 7], [1, 2, 1]], flo`
- `vals, vecs = np.linalg.eig(a)`

8.1 The power method and variants

You may want to watch Gilbert Strang's linear algebra videos 21 and 22 for a good review of eigenvectors, eigenvalues and the decomposition $A = S\Lambda S^{-1}$. There are links to these videos on our course page.

8.2 SVD

The QR factorization shows that we can always decompose a matrix A into a matrix Q whose columns are orthogonal and an upper triangular matrix R. The SVD factorization is similar. We factorize A into two orthogonal matrices U and V^T and a diagonal matrix Σ. In total

$$A = U\Sigma V^T.$$

8.2 SVD

Follow these steps for the SVD factorization.

8.2 SVD

Follow these steps for the SVD factorization.

- Find the eigenvalues λ_i and unit eigenvectors v_i of $A^T A$.

8.2 SVD

Follow these steps for the SVD factorization.

- Find the eigenvalues λ_i and unit eigenvectors v_i of $A^T A$.
- Let the columns of the matrix V be these unit eigenvectors v_i.

8.2 SVD

Follow these steps for the SVD factorization.

- Find the eigenvalues λ_i and unit eigenvectors v_i of $A^T A$.
- Let the columns of the matrix V be these unit eigenvectors v_i.
- Let $u_1 = \frac{Av_1}{\|Av_1\|}, u_2 = \frac{Av_2}{\|Av_2\|}, \dots u_n = \frac{Av_n}{\|Av_n\|}$ be the columns of the matrix U.

8.2 SVD

Follow these steps for the SVD factorization.

- Find the eigenvalues λ_i and unit eigenvectors v_i of $A^T A$.
- Let the columns of the matrix V be these unit eigenvectors v_i.
- Let $u_1 = \frac{Av_1}{\|Av_1\|}, u_2 = \frac{Av_2}{\|Av_2\|}, \dots u_n = \frac{Av_n}{\|Av_n\|}$ be the columns of the matrix U.
- Lastly let the diagonal matrix have diagonal elements the singular values $\sigma_1 = \sqrt{\lambda_1}, \sigma_2 = \sqrt{\lambda_2}, \dots, \sigma_m = \sqrt{\lambda_m}$.

8.2 SVD EXAMPLE

8.2 SVD EXAMPLE

- $A = \begin{bmatrix} 6 & 2 \\ -7 & 6 \end{bmatrix}$.

8.2 SVD EXAMPLE

- $A = \begin{bmatrix} 6 & 2 \\ -7 & 6 \end{bmatrix}$.
- $A^T A = \begin{bmatrix} 85 & -30 \\ -30 & 40 \end{bmatrix}$.

8.2 SVD EXAMPLE

- $A = \begin{bmatrix} 6 & 2 \\ -7 & 6 \end{bmatrix}.$
- $A^T A = \begin{bmatrix} 85 & -30 \\ -30 & 40 \end{bmatrix}.$
- characteristic polynomial of $A^T A$ is
  $\lambda^2 - 125\lambda + 2500 = (\lambda - 100)(\lambda - 25)$

8.2 SVD EXAMPLE

- $A = \begin{bmatrix} 6 & 2 \\ -7 & 6 \end{bmatrix}.$
- $A^T A = \begin{bmatrix} 85 & -30 \\ -30 & 40 \end{bmatrix}.$
- characteristic polynomial of $A^T A$ is $\lambda^2 - 125\lambda + 2500 = (\lambda - 100)(\lambda - 25)$
- Corresponding eigenvectors: $v_1 = \frac{1}{\sqrt{5}} \begin{bmatrix} 2 \\ -1 \end{bmatrix}, v_2 = \frac{1}{\sqrt{5}} \begin{bmatrix} 1 \\ 2 \end{bmatrix}.$

8.2 SVD

8.2 SVD

- $\sigma_1 = 10, \sigma_2 = 5$ so $\Sigma = \begin{bmatrix} 10 & 0 \\ 0 & 5 \end{bmatrix}$.

8.2 SVD

- $\sigma_1 = 10, \sigma_2 = 5$ so $\Sigma = \begin{bmatrix} 10 & 0 \\ 0 & 5 \end{bmatrix}$.
- $Av_1 = \sigma_1 u_1 = 10\frac{(1,-2)}{\sqrt{5}}$ and $Av_2 = \sigma_2 u_2 = 5\frac{(2,1)}{\sqrt{5}}$. So $U = \frac{1}{\sqrt{5}}\begin{bmatrix} 1 & 2 \\ -2 & 1 \end{bmatrix}$.

8.2 SVD

- $\sigma_1 = 10, \sigma_2 = 5$ so $\Sigma = \begin{bmatrix} 10 & 0 \\ 0 & 5 \end{bmatrix}$.
- $Av_1 = \sigma_1 u_1 = 10\frac{(1,-2)}{\sqrt{5}}$ and $Av_2 = \sigma_2 u_2 = 5\frac{(2,1)}{\sqrt{5}}$. So $U = \frac{1}{\sqrt{5}} \begin{bmatrix} 1 & 2 \\ -2 & 1 \end{bmatrix}$.
- $\|A\|_2 = \sigma_1$.

8.2 SVD

- $\sigma_1 = 10, \sigma_2 = 5$ so $\Sigma = \begin{bmatrix} 10 & 0 \\ 0 & 5 \end{bmatrix}$.
- $Av_1 = \sigma_1 u_1 = 10\frac{(1,-2)}{\sqrt{5}}$ and $Av_2 = \sigma_2 u_2 = 5\frac{(2,1)}{\sqrt{5}}$. So $U = \frac{1}{\sqrt{5}} \begin{bmatrix} 1 & 2 \\ -2 & 1 \end{bmatrix}$.
- $\|A\|_2 = \sigma_1$.
- $\kappa(A) = \frac{\sigma_1}{\sigma_r}$.

8.2 SVD

8.2 SVD

- Suppose a satellite transmits a picture containing 1000×1000 pixels. If the color of each pixel is digitized this information can be represented in a 1000×1000 matrix A. How can one transmit the important information contained in the picture without sending all 1000000 numbers?

8.2 SVD

- Suppose a satellite transmits a picture containing 1000×1000 pixels. If the color of each pixel is digitized this information can be represented in a 1000×1000 matrix A. How can one transmit the important information contained in the picture without sending all 1000000 numbers?
- Use the SVD:

$$A = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \ldots + \sigma_r u_r v_r^T.$$

8.2 SVD

- Suppose a satellite transmits a picture containing 1000×1000 pixels. If the color of each pixel is digitized this information can be represented in a 1000×1000 matrix A. How can one transmit the important information contained in the picture without sending all 1000000 numbers?
- Use the SVD:

$$A = \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \ldots + \sigma_r u_r v_r^T.$$

- Watch Gilbert Strang's video 29 from MIT's linear algebra course to learn more about the SVD. There is a link to Strang's videos on our coursepage.

8.3 Methods to Compute Eigenvalues and Singular Values

We will not cover this section. These algorithms are more advanced and important. Needless to say, Python has some of these algorithms in its numpy library.

8.3 Methods to Compute Eigenvalues and Singular Values

We will not cover this section. These algorithms are more advanced and important. Needless to say, Python has some of these algorithms in its numpy library.

- D,V = np.linalg.eig(A)
- U, S, V = np.linalg.svd(A)

8.3 Methods to Compute Eigenvalues and Singular Values

```
import numpy as np

# matrix A.  Use numpy svd code to find the svd of matrix A
A = np.array([[0,1],
              [1,1],
              [1,0]])

U,S,V = np.linalg.svd(A)

print(U)
print(S)
print(V)
```