MTH27101: Methods of Applied Mathematics

Homework 2

20121229 JunPyo Park

1. Let A be a 3×3 matrix with eigenvectors $\mathbf{v_1}^T = [1 \quad 0 \quad 0], \mathbf{v_2}^T = [1 \quad 1 \quad 0], \mathbf{v_3}^T = [1 \quad 1 \quad 1]$ corresponding to eigenvalues $\lambda_1 = -1/3, \lambda_2 = 1/3$ and $\lambda_3 = 1$, respectively, and let $\mathbf{x}^T = [2 \quad 1 \quad 2]$ Discuss the following problems

Matrix A can be calculated by following code.

Problem 1

In [81]:
```
# P : composed with eigenvectors
P = np.matrix([[1,0,0],[1,1,0],[1,1,1]]).transpose()
P
```
```
matrix([[1, 1, 1],
        [0, 1, 1],
        [0, 0, 1]])
```

In [82]:
```
P_inv = np.linalg.inv(P)
P_inv
```
```
matrix([[ 1., -1.,  0.],
        [ 0.,  1., -1.],
        [ 0.,  0.,  1.]])
```

In [83]:
```
# D : Eigenvalued Diagonal Matrix
D = np.diag([-1/3, 1/3, 1])
D
```
```
array([[-0.33333333,  0.        ,  0.        ],
       [ 0.        ,  0.33333333,  0.        ],
       [ 0.        ,  0.        ,  1.        ]])
```

In [84]:
```
A = P * D * P_inv
A
```
```
matrix([[-0.33333333,  0.66666667,  0.66666667],
        [ 0.        ,  0.33333333,  0.66666667],
        [ 0.        ,  0.        ,  1.        ]])
```

(1) (A^20)x could be calculated as following code.

```
In [85]:
x = np.matrix([2,1,2]).transpose()
x

matrix([[2],
        [1],
        [2]])
```

```
In [55]:
# calculate directly
A**20 * x

matrix([[ 2.],
        [ 2.],
        [ 2.]])
```

```
In [49]:
# caculation by decomposition A^20 = P(D^20)P_inv
P * D**20 * P_inv * x

matrix([[ 2.],
        [ 2.],
        [ 2.]])
```

(2)

$$A^k = P D^k P^{-1}$$

$$= \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} (-1/3)^k & 0 & 0 \\ 0 & (1/3)^k & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \left(-\frac{1}{3}\right)^k & \left(\frac{1}{3}\right)^k & 1 \\ 0 & \left(\frac{1}{3}\right)^k & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$A^k x = \begin{bmatrix} \left(-\frac{1}{3}\right)^k & -\left(-\frac{1}{3}\right)^k + \left(\frac{1}{3}\right)^k & -\left(\frac{1}{3}\right)^k + 1 \\ 0 & \left(\frac{1}{3}\right)^k & -\left(\frac{1}{3}\right)^k + 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ 1 \\ 2 \end{bmatrix}$$

$$= \begin{bmatrix} 2\left(-\frac{1}{3}\right)^k - \left(-\frac{1}{3}\right)^k + \left(\frac{1}{3}\right)^k - 2\left(\frac{1}{3}\right)^k + 2 \\ \left(\frac{1}{3}\right)^k - 2\left(\frac{1}{3}\right)^k + 2 \\ 2 \end{bmatrix}$$

$$\therefore \lim_{k \to \infty} A^k x = \begin{bmatrix} 2 \\ 2 \\ 2 \end{bmatrix}$$

2. Discuss the following problems

- If A and B are invertible matrices, show that AB and BA are similar.

Solution

* A and B are invertible

show that AB and BA are similar

$$AB = ABAA^{-1} = AB$$

By definition, $AB \sim BA$

- Let A and B be $n \times n$ matrices, each with n distinct eigenvalues. Prove that A and B have same eigenvalues if and only if $AB = BA$.

I think this is wrong, instead I prove for same eigenvector case.

* Proof

A and B have same eigenvectors $\Rightarrow$ $AB = BA$

$$\begin{cases} A = P D_A P^{-1} \\ B = P D_B P^{-1} \end{cases}$$

P : eigenvectored matrix

D : eigenvalue-diagonal matrix

$$AB = P D_A P^{-1} P D_B P^{-1} = P D_A D_B P^{-1}$$

$$= P D_B D_A P^{-1} = P D_B P^{-1} P D_A P^{-1} = BA$$

Proof.

$AB = BA \Rightarrow A, B$ have same eigenvector

Let v be an eigenvector of A corresponding to eigenvalue λ

then $Av = \lambda v$

commute

$$BAv = \lambda Bv$$

$$A(Bv) = \lambda (Bv)$$

$\Rightarrow$ $Bv = 0$ or $Bv = \mu v$

Since A has n distinct eigenvalues,

eigenspace of λ is one-dimension.

Then $Bv = \mu v$ for some scalar $\mu \in \mathbb{R}$

$\therefore$ v is also eigenvector for B

Since v is chosen to be arbitrary, A and B have same eigenvectors.

- Prove that if A is a diagonalizable matrix such that every eigenvalues of A is either 0 or a, then $A^2 = A$

* A : Diagonalizable

$\Rightarrow \exists P$: invertible and D : diagonal such that

$$A = PDP^{-1}$$

$$A^2 = PD^2P$$

If $A^2 = A \Rightarrow \cancel{P}D^2\cancel{P^{-1}} = \cancel{P}D\cancel{P^{-1}}$

$$\Rightarrow D^2 = D$$

Suppose $D = diag(\lambda_1 \cdots \lambda_n)$

then $D^2 = diag(\lambda_1^2 \cdots \lambda_n^2)$

Then $D^2 = D$ means, for any i between 1 to n,

$$\lambda_i^2 = \lambda_i \Rightarrow \lambda_i = 0 \text{ or } 1$$

3. Discuss the following problems

- Find an SVD of the following matrix A

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ -2 & 2 \end{pmatrix}$$

We can calculate SVD of the A by following codes.

Problem 3

In [57]:
```
a = np.matrix([[1,0],[0,1],[-2,2]])
a
```
```
matrix([[ 1,  0],
        [ 0,  1],
        [-2,  2]])
```

using numpy.linalg.svd method to get U,s,V

In [58]:
```
U, s, V = np.linalg.svd(a, full_matrices=True)
```

Show the result

In [59]:
```
U
```
```
matrix([[ -2.35702260e-01,  -7.07106781e-01,   6.66666667e-01],
        [  2.35702260e-01,  -7.07106781e-01,  -6.66666667e-01],
        [  9.42809042e-01,  -1.11022302e-16,   3.33333333e-01]])
```

In [60]:
```
s
```
```
array([ 3.,  1.])
```

In [61]:
```
V
```
```
matrix([[-0.70710678,  0.70710678],
        [-0.70710678, -0.70710678]])
```

Check it is right or not

diag_s : change s(vectors with singular value) to appropriate shape matrix

In [66]:
```
diag_s = np.diag(s)
diag_s = np.vstack([diag_s, [0,0]])
diag_s
```
```
array([[ 3.,  0.],
       [ 0.,  1.],
       [ 0.,  0.]])
```

In [63]:
```
U * diag_s * V
```
```
matrix([[  1.00000000e+00,   6.10622664e-16],
        [ -1.11022302e-16,   1.00000000e+00],
        [ -2.00000000e+00,   2.00000000e+00]])
```

There exists a little numerical computation error, but almost fit to a(original matrix)

- For the square matrix A which is invertible, there is unique solution for $A\mathbf{x} = \mathbf{b}$. For non-square matrix A, $A\mathbf{x} = \mathbf{b}$ has no unique solution, but best approximation is given by the unique lease square solution $\bar{\mathbf{x}} = (A^T A)^{-1} A^T \mathbf{b}$(because $A^T A\mathbf{x} = A^T\mathbf{b}$) . Here $A^+ \equiv (A^T A)^{-1} A^T$ is called the pseudoinverse of A.

  1. Suppose A be a matrix with linearly independent columns. Show that (1) $AA^+A = A$, (2) AA^+ is symmetric.

Proofs are following.

(1) $AA^+A = A(A^TA)^{-1}A^TA$, A : $m \times n$ matrix

$= AI_n = A$

(2) $AA^+ = A(A^TA)^{-1}A^T$

$(AA^+)^T = A((A^TA)^{-1})^TA^T$

$= A((A^TA)^T)^{-1}A^T$

$= A(A^TA)^{-1}A^T = AA^+$

$\therefore$ AA^+ is symmetric.

2. Let $A = U\Sigma V^T$. Sketch $A^+ = V\Sigma U^T$(Just show your idea, do not prove it clearly), where
$$\Sigma = \begin{pmatrix} D & 0 \\ 0 & 0 \end{pmatrix} \quad \Sigma^+ = \begin{pmatrix} D^{-1} & 0 \\ 0 & 0 \end{pmatrix}$$

For $A \in n\times n$ & invertible matrix

$A = U\Sigma V^T$ <SVD>

Here all U, Σ, V is $n\times n$ matrix

and all invertible.

Then $A^{-1} = (V^T)^{-1}\Sigma^{-1}U^{-1}$

$= V\Sigma^{-1}U^T$ ($\because$ orthogonality of U, V)

For $A \in m\times n$ ($m>n$) which is non-square case

$A = \underbrace{U}_{m\times m}\underbrace{\Sigma}_{m\times n}\underbrace{V^T}_{n\times n}$

In here $U^{-1} = U^T$ and $V^{-1} = V^T$ ($\because$ orthogonality)

But we cannot define Σ^{-1}, since Σ is not square

I think that's why Σ^+ appears here.

$$\Sigma = \left[\begin{array}{c|c} \begin{matrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_a \end{matrix} & 0 \\ \hline 0 & 0 \end{array}\right] \in \mathbb{R}^{m\times n}$$

Define them to $D \in \mathbb{R}^{a\times a}$

$$\Rightarrow \Sigma = \left[\begin{array}{c|c} D & 0 \\ \hline 0 & 0 \end{array}\right]$$

Since Σ^{-1} doesn't exist here, we introduce pseudo inverse of Σ denoted as Σ^{+} as follow

$$\Sigma^{+} = \left[\begin{array}{c|c} D^{-1} & 0 \\ \hline 0 & 0 \end{array}\right] \in \mathbb{R}^{n \times m}$$

Then pseudo inverse of A (A^{+}) is defined by follow,

$$A^{+} = \underbrace{V}_{n \times n} \underbrace{\Sigma^{+}}_{n \times m} \underbrace{U^{T}}_{m \times m} \in n \times m$$

3. Find SVD of A^{+} if A is defined by

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Calculation by Python, codes are following.

Problem 3-3

In [69]:
```
A = np.matrix([[1,1],[1,0],[0,1]])
A
```
```
matrix([[1, 1],
        [1, 0],
        [0, 1]])
```

First find SVD of A

In [70]:
```
U,s,V = np.linalg.svd(A, full_matrices=True)
```

In [76]:
```
pseudo_inv_A = np.linalg.inv(A.transpose() * A) * A.transpose()
pseudo_inv_A
```
```
matrix([[ 0.33333333,  0.66666667, -0.33333333],
        [ 0.33333333, -0.33333333,  0.66666667]])
```

Using s(singular value list), compose the pseudo inverse matrix of sigma, sigma+

In [92]:
```
pseudo_inv_s = np.linalg.inv(np.diag(s))
# Since it's shape should be 2 * 3, we add the empty column
pseudo_inv_s = np.column_stack((pseudo_inv_s,[0,0]))
pseudo_inv_s
```
```
array([[ 0.57735027,  0.        ,  0.        ],
       [ 0.        ,  1.        ,  0.        ]])
```

Check it is correct or not

In [97]:
```
V
```
```
matrix([[-0.70710678, -0.70710678],
        [-0.70710678,  0.70710678]])
```

In [98]:
```
U.transpose()
```
```
matrix([[ -8.16496581e-01,  -4.08248290e-01,  -4.08248290e-01],
        [  1.66533454e-16,  -7.07106781e-01,   7.07106781e-01],
        [ -5.77350269e-01,   5.77350269e-01,   5.77350269e-01]])
```

In [99]:
```
V * pseudo_inv_s * U.transpose()
```
```
matrix([[ 0.33333333,  0.66666667, -0.33333333],
        [ 0.33333333, -0.33333333,  0.66666667]])
```

In [100]:
```
np.linalg.inv(A.transpose()*A) * A.transpose()
```
```
matrix([[ 0.33333333,  0.66666667, -0.33333333],
        [ 0.33333333, -0.33333333,  0.66666667]])
```

We get the same result for A+