IE 531 HW#3

TIANQI WU

Ex 3.7. Since the rows of A is orthonormal,

Denote r_i as the ith row of A.

Then, $r_i \cdot r_j = 1$ if $i=j$ and

$r_i \cdot r_j = 0$ if $i \neq j$

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

$\Downarrow$

Hence, $AA^T = A^TA = I$ Since A is square $n \times n$ matrix.

Then, denote c_i as the ith column of A

Then, $c_i \cdot c_j = 1$ if $i=j$ and

$c_i \cdot c_j = 0$ if $i \neq j$

Hence the columns of A are orthonormal.

Ex 3.13

(a). $A_k = \sum_{i=1}^{k} \sigma_i u_i v_i^T$

Since $\|A\|_F^2 = \sum_{i=1}^{r} \sigma_i^2(A)$

$\|A_k\| = \sum_{i=1}^{k} \sigma_i^2$

(b). Since $\|A\|_2 = \sigma_1(A)$

$\|A_k\|_2^2 = \max(\sigma_i^2(A_k)) = \sigma_1^2$

(c). $A - A_k = \sum_{i=1}^{r} \sigma_i u_i v_i^T - \sum_{i=1}^{k} \sigma_i u_i v_i^T$

$= \sum_{k+1}^{r} \sigma_i u_i v_i^T$

$\|A - A_k\|_F^2 = \sum_{k+1}^{r} \sigma_i^2$

(d). $\|A - A_k\|_2^2 = \max(\sigma_i^2(A - A_k)) = \sigma_{k+1}^2$

Ex 3.14. let $A = UDV^T$

Then $A^T = (UDV^T)^T$

$= VD^TU^T$

Since A is symmetric, u_i are pairwise orthogonal

$A = A^T$

$\therefore UDV^T = VD^TU^T$

$\because D = D^T$

$U = V$, $V^T = U^T$

Hence, the left and right singular values are the same.

And $A = VDV^T$

Or, we can prove it as following:

let B be a matrix with SVD $= \sum_i \sigma_i u_i v_i^T$

Then, B^TB is a symmetric matrix because

$\because (B^TB)^T = B^TB$,

let $A = B^TB = (\sum_i \sigma_i v_i u_i^T)(\sum_j \sigma_j u_j v_j^T)$

$= \sum_{ij} \sigma_i \sigma_j v_i (u_i^T u_j) v_j^T$

Since U is orthonormal, $= \sum_i \sigma_i^2 v_i v_i^T$

Hence, the left and right singular values are the same

Ex3.16 is solved by python (jupyter notebook) and the file is attached.

```
print('U: \n', U, '\nsigma: \n', sigma, '\nV: \n', V_t)
```

```
U:
 [[ 0.40861658  0.9268237 ]
 [-0.98949853  0.14454294]]
sigma:
 [ 5.39444409  0.37980901]
V:
 [[ 0.59118004  0.80653962]
 [ 0.80653962 -0.59118004]]
```

```
In [1]: import numpy as np
```

```
In [11]: A = np.array([[1, 2], [3, 4]])
         n = 2
         k = 14

         U = np.zeros((n, n))
         sigma = np.zeros(n)
         V_t = np.zeros((n, n))
```

```
In [12]: def get_svd(A):
             B = np.transpose(A).dot(A)
             B_power = B.copy()

             for j in range(k):
                 B_power = np.matmul(B_power, B)

             v1 = B_power[:, 0] / np.linalg.norm(B_power[:, 0])
             s1 = np.sqrt((np.dot(B, v1) / v1)[0])
             u1 = np.dot(A, v1) / s1

             return v1, s1, u1
```

```
In [13]: v1, s1, u1 = get_svd(A)
         V_t[0, :] = v1
         U[0, :] = u1
         sigma[0] = s1

         large_contrib = s1 * np.outer(u1, v1)
         A2 = A - large_contrib

         v2, s2, u2 = get_svd(A2)
         V_t[1, :] = v2
         U[1, :] = u2
         sigma[1] = s2
```