16-811

Erica Weng

September 2020

1

To run the implementation, run `python3 code/q1.py`. It will test 5 several test cases. They should each print out the $PA = LDU$ decomposition.

To test your own test case, call the function `lu(A)` on your numpy array of choice `A` from the `code/q1.py` file. The function returns `P, L, D, U` in a tuple in that order.

2

LDU

A_1:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 10 & -10 & 0 \\ 0 & -4 & 2 \\ 2 & 0 & -5 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{1}{5} & 0 & 1 \end{bmatrix} \begin{bmatrix} 10 & -10 & 0 \\ 0 & -4 & 2 \\ 0 & 2 & -5 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{1}{5} & -\frac{1}{2} & 1 \end{bmatrix} \begin{bmatrix} 10 & -10 & 0 \\ 0 & -4 & 2 \\ 0 & 0 & -4 \end{bmatrix}$$

$$LDU = \boxed{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{1}{5} & -\frac{1}{2} & 1 \end{bmatrix} \begin{bmatrix} 10 & 0 & 0 \\ 0 & -4 & 0 \\ 0 & 0 & -4 \end{bmatrix} \begin{bmatrix} 1 & -1 & 0 \\ 0 & 1 & -\frac{1}{2} \\ 0 & 0 & 1 \end{bmatrix}}$$

A_2:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 & -5 & 0 & 0 \\ 5 & 5 & 5 & 0 \\ 0 & -1 & 4 & 1 \\ 0 & 4 & -1 & 2 \\ 0 & 0 & 2 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & & & & \\ 1 & 1 & & 0 & \\ 0 & 0 & 1 & & \\ 0 & 0 & 0 & 1 & \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 & -5 & 0 & 0 \\ 0 & 10 & 5 & 0 \\ 0 & -1 & 4 & 1 \\ 0 & 4 & -1 & 2 \\ 0 & 0 & 2 & 1 \end{bmatrix}$$

eliminate 1st col under diagonal

$$\begin{bmatrix} 1 & & & & \\ 1 & 1 & & 0 & \\ 0 & -1/10 & 1 & & \\ 0 & 2/5 & 0 & 1 & \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 & -5 & 0 & 0 \\ 0 & 10 & 5 & 0 \\ 0 & 0 & 9/2 & 1 \\ 0 & 0 & -3 & 2 \\ 0 & 0 & 2 & 1 \end{bmatrix}$$

2nd col

$4 - 10 \cdot \frac{4}{10}$

$$\begin{bmatrix} 1 & & & & \\ 0 & 1 & & 0 & \\ 0 & 1/10 & 1 & & \\ 0 & 2/5 & -2/3 & 1 & \\ 0 & 0 & 4/9 & 0 & 1 \end{bmatrix} \begin{bmatrix} 5 & -5 & 0 & 0 \\ 0 & 10 & 5 & 0 \\ 0 & 0 & 9/2 & 1 \\ 0 & 0 & 0 & 8/3 \\ 0 & 0 & 0 & 5/9 \end{bmatrix}$$

3rd col

$-3 - (-3) \cdot \frac{9}{2} \cdot \left(\frac{2}{9}\right)$

$\frac{6}{3} + \frac{2}{3} \quad \frac{-6}{9}$

$2 - (2) \cdot \left(\frac{2}{9}\right) \cdot \left(\frac{9}{2}\right)$

$\frac{9}{9} - \frac{4}{9}$

L $\quad$ DU

$$\begin{bmatrix} 1 & & & & \\ 0 & 1 & & 0 & \\ 0 & 1/10 & 1 & & \\ 0 & 2/5 & -2/3 & 1 & \\ 0 & 0 & 4/9 & 5/24 & 1 \end{bmatrix} \begin{bmatrix} 5 & -5 & 0 & 0 \\ 0 & 10 & 5 & 0 \\ 0 & 0 & 9/2 & 1 \\ 0 & 0 & 0 & 8/3 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

last col

$5/9 - \frac{5}{9} \cdot \frac{8}{3} \cdot \frac{3}{8}$

$\frac{15}{72} = \frac{5}{24}$

$$D = \begin{bmatrix} 5 & & & & \\ & 10 & & 0 & \\ & & 9/2 & & \\ & 0 & & 8/3 & \\ & & & & 0 \end{bmatrix} \qquad U = \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & 1/2 & 0 \\ 0 & 0 & 1 & 2/9 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

A_3**:**

$$\begin{bmatrix} 1 & & 0 \\ & 1 & \\ 0 & & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 10 & 2 & 9 \\ 8 & 0 & 7 \end{bmatrix}$$

$$\begin{bmatrix} 1 & & \\ 10 & 1 & \\ 8 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 0 & -8 & -1 \\ 0 & -8 & -1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & & 0 \\ 10 & 1 & \\ 8 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 0 & -8 & -1 \\ 0 & 0 & 0 \end{bmatrix}$$

$$LDU = \begin{bmatrix} 1 & & 0 \\ 10 & 1 & \\ 8 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -8 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1/8 \\ 0 & 0 & 0 \end{bmatrix}$$

SVD

Code for all three matrices:

```python
def svd(arr):
  U, s, Vh = numpy.linalg.svd(arr)
  M, N = U.shape[0], Vh.shape[0]
  S = scipy.linalg.diagsvd(s, M, N}
  return U, S, Vh
```

The code is in `code/q2.py`

A_1**:**

$$U = \begin{bmatrix} -0.98 & 0.02 & -0.22 \\ -0.20 & -0.51 & 0.84 \\ -0.10 & 0.86 & 0.50 \end{bmatrix} S = \begin{bmatrix} 14.50 & 0.00 & 0.00 \\ 0.00 & 5.95 & 0.00 \\ 0.00 & 0.00 & 1.86 \end{bmatrix} V^T = \begin{bmatrix} -0.69 & 0.73 & 0.01 \\ 0.32 & 0.31 & -0.89 \\ -0.65 & -0.61 & -0.45 \end{bmatrix}$$

A_2**:**

$$U = \begin{bmatrix} 0.11 & 0.87 & 0.37 & -0.31 & -0.02 \\ -0.93 & 0.15 & 0.16 & 0.28 & 0.02 \\ -0.20 & 0.23 & -0.75 & -0.32 & -0.50 \\ -0.24 & -0.41 & 0.38 & -0.77 & -0.18 \\ -0.14 & 0.06 & -0.35 & -0.36 & 0.85 \end{bmatrix} S = \begin{bmatrix} 9.14 & 0.00 & 0.00 & 0.00 \\ 0.00 & 7.80 & 0.00 & 0.00 \\ 0.00 & 0.00 & 4.42 & 0.00 \\ 0.00 & 0.00 & 0.00 & 2.24 \\ 0.00 & 0.00 & 0.00 & 0.00 \end{bmatrix} V^T =$$

$$\begin{bmatrix} -0.45 & -0.65 & -0.60 & -0.09 \\ 0.65 & -0.70 & 0.28 & -0.07 \\ 0.61 & 0.28 & -0.74 & -0.08 \\ -0.05 & 0.09 & 0.09 & -0.99 \end{bmatrix}$$

A_3**:**

$$U = \begin{bmatrix} -0.09 & -0.57 & -0.82 \\ -0.79 & -0.46 & 0.41 \\ -0.61 & 0.68 & -0.41 \end{bmatrix} S = \begin{bmatrix} 17.28 & 0.00 & 0.00 \\ 0.00 & 1.51 & 0.00 \\ 0.00 & 0.00 & 0.00 \end{bmatrix} V^T = \begin{bmatrix} -0.74 & -0.10 & -0.66 \\ 0.13 & -0.99 & -0.01 \\ 0.66 & 0.09 & -0.75 \end{bmatrix}$$

3

a

$$V : \begin{pmatrix} -0.74 & 0.13 & 0.66 \\ -0.10 & -0.99 & 0.09 \\ -0.66 & -0.01 & -0.75 \end{pmatrix} \frac{1}{\Sigma} : \begin{pmatrix} 0.06 & 0.00 & 0.00 \\ 0.00 & 0.66 & 0.00 \\ 0.00 & 0.00 & 0.00 \end{pmatrix} U^T : \begin{pmatrix} -0.09 & -0.79 & -0.61 \\ -0.57 & -0.46 & 0.68 \\ -0.82 & 0.41 & -0.41 \end{pmatrix}$$

$$x = V\frac{1}{\Sigma}U^T b = \boxed{\begin{pmatrix} 0.02 \\ 0.86 \\ 0.12 \end{pmatrix}}$$

verify:

$$Ax - b = \begin{pmatrix} 0.00 \\ -0.00 \\ 0.00 \end{pmatrix}$$

b

SVD same as in (a)

$$x = V\frac{1}{\Sigma}U^T b = \boxed{\begin{pmatrix} 0.02 \\ 0.86 \\ 0.12 \end{pmatrix}}$$

verify:

$$Ax - b = \begin{pmatrix} -2.00 \\ 1.00 \\ -1.00 \end{pmatrix}$$

which is orthogonal to A's column space (spanned by $\begin{pmatrix} 1.00 \\ 2.00 \\ 0.00 \end{pmatrix}$ and $\begin{pmatrix} 1.00 \\ 9.00 \\ 7.00 \end{pmatrix}$).

for parts (a) and (b) is that the SVD decomposition is the same. However, part (a)'s b is in A's column space, so $Ax - b = 0$, whereas in part (b) b is not in A's column space, so $Ax - b$ is a vector perpendicular to A's column space.

c

The code used to generate these solutions is in `code/q3.py`

$$V : \begin{pmatrix} -0.69 & 0.32 & -0.65 \\ 0.73 & 0.31 & -0.61 \\ 0.01 & -0.89 & -0.45 \end{pmatrix} \frac{1}{\Sigma} : \begin{pmatrix} 0.07 & 0.00 & 0.00 \\ 0.00 & 0.17 & 0.00 \\ 0.00 & 0.00 & 0.54 \end{pmatrix} U^T : \begin{pmatrix} -0.98 & -0.20 & -0.10 \\ 0.02 & -0.51 & 0.86 \\ -0.22 & 0.84 & 0.50 \end{pmatrix}$$

$$x = V\frac{1}{\Sigma}U^T b = \boxed{\begin{pmatrix} -1.00 \\ -2.00 \\ -3.00 \end{pmatrix}}$$

4

a

Given u, v both orthogonal vectors: $Av = (I - uu^T)v = v - uu^T v = v - u(u \cdot v)$. In other words, the matrix A subtracts from v the component in the direction of u ($u(u \cdot v)$). Thus, the matrix A transforms the vector v such that it becomes orthogonal to u.

b

The eigenvalues of A:

$$(I - uu^T - I\lambda)x = 0$$

$$det((1 - \lambda)I - uu^T) = 0$$

$$det((1 - \lambda)I - uu^T) = \left(1 - u^T((1 - \lambda)I)^{-1}u\right) \cdot det(1 - \lambda)I$$

by Matrix determinant lemma (`https://en.wikipedia.org/wiki/Matrix_determinant_lemma`)

$$= \left(1 - u^T \frac{1}{1 - \lambda} Iu\right) \cdot det((1 - \lambda)I)$$

$$= \left(1 - \frac{1}{1-\lambda}\right) \cdot det((1-\lambda)I)$$

because u is unit-length

$$= det((1-\lambda)I) - \frac{1}{1-\lambda} det((1-\lambda)I)$$

$$= (1-\lambda)^n - \frac{1}{1-\lambda}(1-\lambda)^n$$

$$= (1-\lambda)^n - (1-\lambda)^{n-1} = (-\lambda)(1-\lambda)^{n-1}$$

Thus $\boxed{\lambda = 0, 1}$

c

According to part a, we know that for any vector v we have $Av = v - u(u \cdot v)$. Thus if $v = u$, then $Av = u - u(u \cdot u) = u - u = 0$. So the null space of A is spanned by u.

d

$A^2 = (I - uu^T)^2 = I^2 - 2uu^T + uu^T uu^T = I - uu^T = A$

5

The problem calls us to find the best rotation matrix R and translation t for the two sets of points $p_1, \ldots, p_n$ and $q_1, \ldots, q_n$, where $q_i = Rp_i + t$. That means solving the least-squares solution to the equation

$$L = \arg\min_{R,t} \sum_{i=1}^{n} ||Rp_i + t - q_i||^2$$

.

Solving for t is simple:

$$\frac{\partial L}{\partial t} = 2\left(R\sum_{i=1}^{n} p_i + nt - \sum_{i=1}^{n} q_i\right) \quad = 0$$

$$t = \bar{q} - R\bar{p}$$

Solving for R now:

$$L = \arg\min_{R} \sum_{i=1}^{n} ||Rp_i + (\bar{q} - R\bar{p}) - q_i||^2$$

$$L = \arg\min_{R} \sum_{i=1}^{n} ||R(p_i - \bar{p}) - (q_i - \bar{q})||^2$$

change variables for ease of calculation using $p' = p_i - \bar{p}$ and $q' = q_i - \bar{q}$

$$
\begin{aligned}
L &= \arg\min_R \sum_{i=1}^{n} ||Rp'_i - q'_i||^2 \\
&= \arg\min_R \sum_{i=1}^{n} (Rp'_i - q'_i)^T (Rp'_i - q'_i) \\
&= \arg\min_R \sum_{i=1}^{n} (Rp'_i - q'_i)^T (Rp'_i - q'_i) \\
&= \arg\min_R \sum_{i=1}^{n} p'^T_i R^T R p'_i - q'^T_i R p'_i - p'^T_i R^T q'_i + q'^T_i q'_i \\
&= \arg\min_R \sum_{i=1}^{n} p'^T_i p'_i - 2p'^T_i R^T q'_i + q'^T_i q'_i \\
&= \arg\min_R \sum_{i=1}^{n} -Tr(Rp'_i q'^T_i) \\
&= \arg\max_R Tr(RP'Q'^T)
\end{aligned}
$$

the third-to-last line gotten by $R^T R = I$ because rotation matrices are orthogonal, and because $q'^T_i Rp_i$ is a scalar and $q'^T_i Rp_i = (p'_i R^T q'_i)^T$, thus $q'^T_i Rp_i = p'_i R^T q'_i$. Then the second-to-last line gotten by taking out constants and factors that don't depend on R, and using Hint #2. The last line gotten by putting all column-vector points p_i into a matrix $P = [p_1; \ldots; p_n]$ and similarly for all q_i into matrix $Q = [q_1, \ldots; q_n]$. Now, computing the SVD of $P'Q'^T$:

$$
\begin{aligned}
&= \arg\max_R Tr(RU\Sigma V^T) \\
&= \arg\max_R Tr(\Sigma V^T RU)
\end{aligned}
$$

because $Tr(AB) = Tr(BA)$. because V, U, and R are all orthogonal, and any multiplication of them will give another orthogonal matrix, then to maximize $Tr(\Sigma V^T RU)$ means to have $V^T RU = I$ where $Tr(\Sigma V^T RU) = Tr(\Sigma)$. So $R = VU^T$.

So now here is the algorithm in the file `code/q5.py`:

1. For sets of points $P = [p_i; \ldots; p_n]$ and $Q = [q_i; \ldots; q_n]$, find the centroid (average value for each of the 3 dimensions) of each set of points, $\bar{p} = \frac{1}{n}\sum_{i=1}^{n} p_i$ and $\bar{q} = \frac{1}{n}\sum_{i=1}^{n} q_i$, respectively
2. Find new sets of points $P' = P - \bar{p}$ and $Q' = Q - \bar{q}$ to get the translation from each set of points to the origin
3. Use the SVD to solve for the $P'Q'^T = U\Sigma V^T$, and get rotation matrix $R = VU^T$

4. Calculate translation $t = \bar{q} - R\bar{p}$
5. return R and t.

To run the implementation, run `python3 code/q5.py`. It will run several several test cases.