**Tutorial 6 for Chapter 2**

Case study 8: Movie Recommendation by Singular Value Decomposition

*Reference: 数据挖掘原理与应用*

For the course AMA546 Statistical Data Mining
Lecturer: Dr. Catherine Liu
AMA, PolyU, HKSAR

In [1]:

```
import numpy as np
import pandas as pd
```

executed in 1.23s, finished 21:39:31 2023-03-05

**Content:**



We now take up a form of matrix analysis that leads to a **low-dimensional representation** of a high-dimensional matrix. This approach, called **singular value decomposition (SVD)**, not only allows an **exact representation** of any matrix, but also makes it easy to **eliminate the less important parts** of that representation to produce an approximate representation with any desired number of dimensions. The **fewer the dimensions** we choose, the **less complexity but less accurate** will be the approximation.

We begin with the necessary definitions. Then, we explore the idea that the SVD defines a small number of **features** that connect the rows (users) and columns (items) of the matrix. We show how eliminating the least important features gives us a smaller representation that closely approximates the original matrix. Final, we see how these features facilitate the movie recommendation.

# 1 Definition of SVD

Let $M$ be an $m \times n$ matrix, and let the rank of $M$ be $r$. Recall that the **rank of a matrix** is the largest number of rows (or columns) we can choose for which **NO nonzero linear combination of the rows (or columns) is the all-zero vector 0** (we say a set of such rows or columns is independent). Then we can find matrices $U$, $\Sigma$, and $V$ as shown in figure below with the following properties:

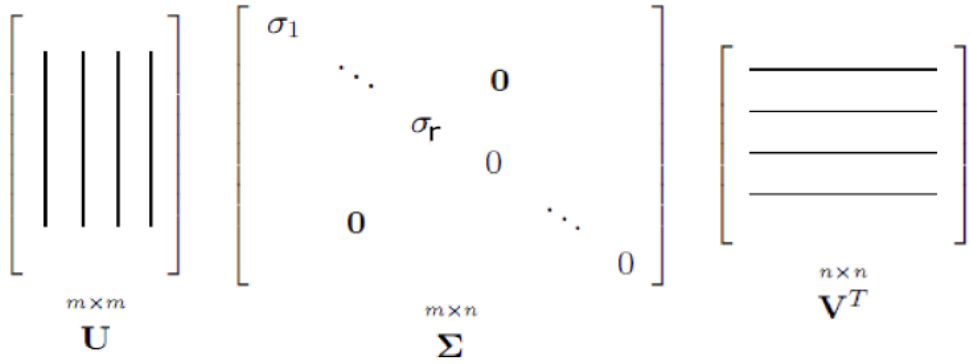



**Figure 1.** Structure of SVD matrices decomposition.

- $U$ is an $m \times m$ column-orthonormal matrix ; that is, each of its columns is a unit vector and the dot product of any two columns is 0.
- $V$ is an $n \times n$ column-orthonormal matrix. Note that we always use $V$ in its transposed form, so it is the rows of $V^T$ that are orthonormal.
- $\Sigma$ is an $m \times n$ rectangular diagonal matrix; that is, all elements not on the main diagonal are 0. The diagnal elements of $\Sigma$ are called the singular values of M.

Since the **first $r$ diagnal items of $\Sigma$ are non-zero**, only the **first $r$ columns in $U$** and **first $r$ rows in $V^T$** are in effect. Therefore, we can derive a **equivalent** definition of SVD:

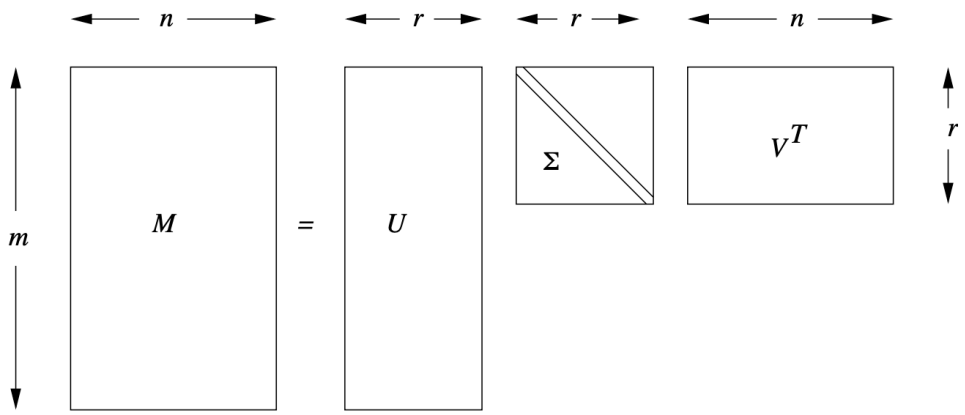



Figure 11.5: The form of a singular-value decomposition

- $U$ is an $m \times r$ column-orthonormal matrix ; that is, each of its columns is a unit vector and the dot product of any two columns is 0.
- $V$ is an $r \times n$ column-orthonormal matrix. Note that we always use $V$ in its transposed form, so it is the rows of $V^T$ that are orthonormal.
- $\Sigma$ is an $r \times r$ diagonal matrix. The diagnal elements of $\Sigma$ are called the singular values of M.

# 2 Computing the SVD of a Matrix

The SVD of a matrix $M$ is strongly connected to the eigenvalues of the symmetric matrices $M^{\mathrm{T}}M$ and $MM^{\mathrm{T}}$. This relationship allows us to obtain the SVD of $M$ from the eigenpairs of the latter two matrices. To begin the explanation, start with $M = U\Sigma V^{\mathrm{T}}$, the expression for the SVD of $M$. Then

$$M^{\mathrm{T}} = \left(U\Sigma V^{\mathrm{T}}\right)^{\mathrm{T}} = \left(V^{\mathrm{T}}\right)^{\mathrm{T}}\Sigma^{\mathrm{T}}U^{\mathrm{T}} = V\Sigma^{\mathrm{T}}U^{\mathrm{T}}$$

Since $\Sigma$ is a diagonal matrix, transposing it has no effect. Thus, $M^{\mathrm{T}} = V\Sigma U^{\mathrm{T}}$. Now, $M^{\mathrm{T}}M = V\Sigma U^{\mathrm{T}}U\Sigma V^{\mathrm{T}}$. Remember that $U$ is an orthonormal matrix, so $U^{\mathrm{T}}U$ is the identity matrix of the appropriate size. That is,

$$M^{\mathrm{T}}M = V\Sigma^2 V^{\mathrm{T}}$$

Multiply both sides of this equation on the right by $V$ to get

$$M^{\mathrm{T}}MV = V\Sigma^2 V^{\mathrm{T}}V$$

Since $V$ is also an orthonormal matrix, we know that $V^{\mathrm{T}}V$ is the identity. Thus

$$M^{\mathrm{T}}MV = V\Sigma^2$$

Since $\Sigma$ is a diagonal matrix, $\Sigma^2$ is also a diagonal matrix whose entry in the $i$ th row and column is the square of the entry in the same position of $\Sigma$. It says that $V$ **is the matrix of eigenvectors of** $M^{\mathrm{T}}M$ and $\Sigma^2$ **is the diagonal matrix whose entries are the corresponding eigenvalues**.

Thus, the same algorithm that computes the eigenpairs for $M^{\mathrm{T}}M$ gives us the matrix $V$ for the SVD of $M$ itself. It also gives us the singular values for this SVD; just take the square roots of the eigenvalues for $M^{\mathrm{T}}M$.

Only $U$ remains to be computed, but it can be found in the same way we found $V$. Start with

$$MM^{\mathrm{T}} = U\Sigma V^{\mathrm{T}}\left(U\Sigma V^{\mathrm{T}}\right)^{\mathrm{T}} = U\Sigma V^{\mathrm{T}}V\Sigma U^{\mathrm{T}} = U\Sigma^2 U^{\mathrm{T}}$$

Then by a series of manipulations analogous to the above, we learn that

$$MM^{\mathrm{T}}U = U\Sigma^2$$

That is, $U$ **is the matrix of eigenvectors of** $MM^{\mathrm{T}}$. A small detail needs to be explained concerning $U$ and $V$. Each of these matrices have $r$ columns, while $M^{\mathrm{T}}M$ is an $n \times n$ matrix and $MM^{\mathrm{T}}$ is an $m \times m$ matrix. Both $n$ and $m$ are at least as large as $r$. Thus, $M^{\mathrm{T}}M$ and $MM^{\mathrm{T}}$ should have an additional $n - r$ and $m - r$ eigenpairs, respectively, and these pairs do not show up in $U$, $V$, and $\Sigma$. Since the rank of $M$ is $r$, all other eigenvalues will be 0 , and these are not useful.

# 3 Example 1: Feature Extraction by SVD

Here gives a **rank-2 matrix** representing ratings of movies by users. In this contrived example there are **two "feature"** underlying the movies: **science-fiction and romance**. **All the boys rate only science-fiction, and all the girls rate only romance.** Our objective is to **recommend movies to customers based on their rateings** of movies via the Singular Value Decomposition.

| | Matrix | Alien | Star Wars | Casablanca | Titanic |
|---|---|---|---|---|---|
| Joe | 1 | 1 | 1 | 0 | 0 |
| Jim | 3 | 3 | 3 | 0 | 0 |
| John | 4 | 4 | 4 | 0 | 0 |
| Jack | 5 | 5 | 5 | 0 | 0 |
| Jill | 0 | 0 | 0 | 4 | 4 |
| Jenny | 0 | 0 | 0 | 5 | 5 |
| Jane | 0 | 0 | 0 | 2 | 2 |

It is this existence of **two concepts that gives the matrix a rank of 2**. That is, we may pick one of the first four rows and one of the last three rows and observe that there is no nonzero linear sum of these rows that is 0. But we cannot pick three independent rows. For example, if we pick rows 1, 2, and 7, then three times the first minus the second, plus zero times the seventh is 0. We can make a similar observation about the columns. We may pick one of the first three columns and one of the last two coluns, and they will be independent, but no set of three columns is independent.

The **matrix $M$ can be decomposed into $U$ , $\Sigma$, and $V$**,** with all elements correct to two significant digits, is shown in below. Since the rank of M is 2, we can use r = 2 in the decomposition. We have already see how to compute this decomposition in Section 2.

In [14]:

```
# load the data
x1 = np.matrix([[1,1,1,0,0],
                [3,3,3,0,0],
                [4,4,4,0,0],
                [5,5,5,0,0],
                [0,0,0,4,4],
                [0,0,0,5,5],
                [0,0,0,2,2]])
# singular value deconposition to the matrix
u1, sigma1, v1 = np.linalg.svd(x1)
```

executed in 14ms, finished 21:45:58 2023-03-05

In [15]:

```
# check the dimension and result of the decomposition result
for i in [-u1, sigma1, -v1]:
    print(i.shape, '\n',  i.round(2), '\n')
```

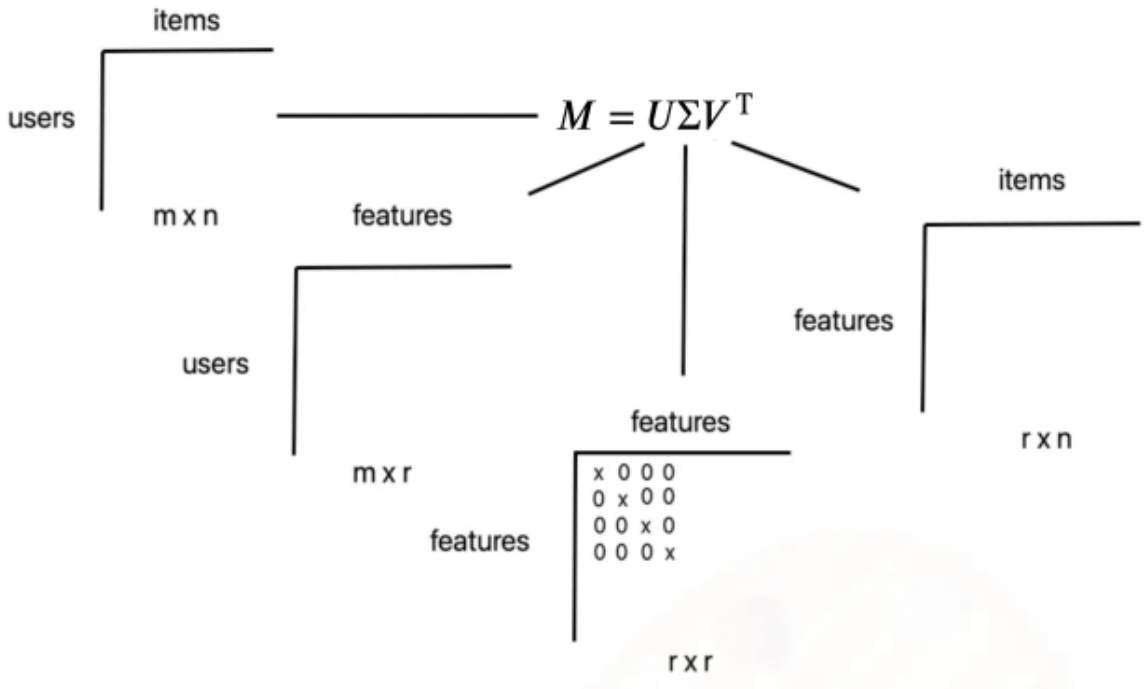
executed in 8ms, finished 21:45:58 2023-03-05

```
(7, 7)
 [[ 0.14 -0.   -0.42  0.56  0.42 -0.52 -0.21]
 [ 0.42 -0.   -0.15  0.21 -0.85 -0.19 -0.08]
 [ 0.56 -0.   -0.21 -0.72  0.21 -0.26 -0.1 ]
 [ 0.7  -0.    0.34  0.34  0.26  0.42  0.17]
 [-0.    0.6  -0.64 -0.   -0.    0.44  0.18]
 [-0.    0.75  0.44 -0.   -0.   -0.44  0.22]
 [-0.    0.3   0.18 -0.   -0.    0.22 -0.91]]

(5,)
 [12.37  9.49  0.    0.    0.  ]

(5, 5)
 [[ 0.58  0.58  0.58 -0.   -0.  ]
 [ 0.    0.    0.    0.71  0.71]
 [-0.   -0.   -0.    0.71 -0.71]
 [-0.    0.71 -0.71 -0.   -0.  ]
 [-0.82  0.41  0.41 -0.   -0.  ]]
```

***The decomposition result can be written into this form:***

$$\underbrace{\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 0 & 0 & 2 & 2 \end{bmatrix}}_{M} = \underbrace{\begin{bmatrix} 0.14 & 0 \\ 0.42 & 0 \\ 0.56 & 0 \\ 0.70 & 0 \\ 0 & 0.60 \\ 0 & 0.75 \\ 0 & 0.30 \end{bmatrix}}_{U} \underbrace{\begin{bmatrix} 12.4 & 0 \\ 0 & 9.5 \end{bmatrix}}_{\Sigma} \underbrace{\begin{bmatrix} 0.58 & 0.58 & 0.58 & 0 & 0 \\ 0 & 0 & 0 & 0.71 & 0.71 \end{bmatrix}}_{V^T} \quad (1)$$

***Interpretation of SVD:***




The key to understanding what SVD offers is in viewing **the $r$ columns of $U$, $\Sigma$, and $V$ as representing unobservable features** that are hidden in the original matrix $M$. In the example, these features are clear; one is "**science fiction**" and the other is "**romance**". Let us think of the rows of $M$ as people and the columns of $M$ as movies:

- The **matrix $U$ connects people to features**. For example, the person Joe, who corresponds to row 1 of $M$, likes only the feature science fiction. The value 0.14 in the first row and first column of $U$ is smaller than some of the other entries in that column, because while Joe watches only science fiction, he doesn't rate those movies highly. The second column of the first row of $U$ is 0 , because Joe doesn't rate romance movies at all.
- The **matrix $V$ relates movies to features**. The 0.58 in each of the first three columns of the first row of $V^{\mathrm{T}}$ indicates that the first three movies (The Matrix, Alien, and Star Wars) each are science-fiction, while the 0's in the last two columns of the first row say that these movies do not relate to the concept romance at all. Likewise, the second row of $V^{\mathrm{T}}$ tells us that the movies Casablanca and Titanic are exclusively romances.
- The **matrix $\Sigma$ gives the strength of each of the concepts in the dataset**. In our example, the strength of the science-fiction concept is 12.4 , while the strength of the romance concept is 9.5 . Intuitively, the science-fiction concept is stronger because the data provides more information about the movies of that people who like them.

***Gerneral case:***

In general, **the concepts will not be so clear**.There will be **fewer 0's in $U$ and $V$**, although $\Sigma$ is always a diagonal matrix and will always have 0's off the diagonal. The entities represented by the rows and columns of $M$ (analogous to people and movies in our example) will take up several different concepts. In practice, when the rank of $M$ is greater than the number of columns we want for the matrices $U, \Sigma$, and $V$, the decomposition is not exact. We need to **eliminate from the exact decomposition those columns of $U$ and $V$ that correspond to the smallest singular values**, in order to get the best approximation. The following example is a slight modification of the example that will illustrate the point.

# 4 Example 2: A More Complicated Feature Extraction

The dataset below is almost the same as the one above, **but Jill and Jane rated Alien**, although neither liked it very much. The rank of the matrix in here is 3; for example the first, sixth, and seventh rows are independent, but you can check that no four rows are independent.

| | Matrix | Alien | Star Wars | Casablanca | Titanic |
|---|---|---|---|---|---|
| Joe | 1 | 1 | 1 | 0 | 0 |
| Jim | 3 | 3 | 3 | 0 | 0 |
| John | 4 | 4 | 4 | 0 | 0 |
| Jack | 5 | 5 | 5 | 0 | 0 |
| Jill | 0 | **2** | 0 | 4 | 4 |
| Jenny | 0 | 0 | 0 | 5 | 5 |
| Jane | 0 | **1** | 0 | 2 | 2 |

Here we perform the SVD decomposition to the matrix:

In [16]:

```
# load the data
x2 = np.matrix([[1,1,1,0,0],
                [3,3,3,0,0],
                [4,4,4,0,0],
                [5,5,5,0,0],
                [0,2,0,4,4],
                [0,0,0,5,5],
                [0,1,0,2,2]])
# svd
u2, sigma2, v2 = np.linalg.svd(x2)
```

executed in 7ms, finished 21:46:08 2023-03-05

In [17]:

```
# check the dimension and result of the decomposition result
for i in [-u2, sigma2, -v2]:
    print(i.shape, '\n',  i.round(2), '\n')
```

executed in 10ms, finished 21:46:09 2023-03-05

```
(7, 7)
 [[ 0.14 -0.02  0.01  0.56 -0.38 -0.7  -0.19]
 [ 0.41 -0.07  0.03  0.21  0.76 -0.26  0.38]
 [ 0.55 -0.09  0.04 -0.72 -0.18 -0.34 -0.09]
 [ 0.69 -0.12  0.05  0.34 -0.23  0.57 -0.12]
 [ 0.15  0.59 -0.65  0.   -0.2  -0.    0.4 ]
 [ 0.07  0.73  0.68 -0.   -0.   -0.   -0.  ]
 [ 0.08  0.3  -0.33  0.    0.4  -0.   -0.8 ]]

(5,)
 [12.48  9.51  1.35  0.    0.  ]

(5, 5)
 [[ 0.56  0.59  0.56  0.09  0.09]
 [-0.13  0.03 -0.13  0.7   0.7 ]
 [ 0.41 -0.8   0.41  0.09  0.09]
 [-0.71  0.    0.71  0.   -0.  ]
 [-0.    0.   -0.    0.71 -0.71]]
```

***The decomposition result can be written into this form:***

$$\underbrace{\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix}}_{M'} = \underbrace{\begin{bmatrix} .13 & .02 & -.01 \\ .41 & .07 & -.03 \\ .55 & .09 & -.04 \\ .68 & .11 & -.05 \\ .15 & -.59 & .65 \\ .07 & -.73 & -.67 \\ .07 & -.29 & .32 \end{bmatrix}}_{U'} \underbrace{\begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 1.3 \end{bmatrix}}_{\Sigma'} \underbrace{\begin{bmatrix} .56 & .59 & .56 & .09 & .09 \\ .12 & -.02 & .12 & -.69 & -.69 \\ .40 & -.80 & .40 & .09 & .09 \end{bmatrix}}_{V'^T}$$

**Only three columns for $U'$, $\Sigma'$, and $V'$ need to be included, because the matrix is of rank three**. The columns of $U$ and $V$ still correspond to features. The first is still "science fiction" and the second is "romance." **It is harder to tell the third column's feature**, but it doesn't matter all that much, because **its weight**, as given by the **third nonzero entry in $\Sigma$**, **is very low** compared with the weights of the first two concepts.

# 5 Example 2 cont.: Dimensionality Reduction by SVD

In this section, we consider **eliminating some of the least important features**. For instance, we might want to **eliminate the third feature** in Example 2, since it tells little about the dataset, and the fact that its **associated singular value is so small** confirms its unimportance.

Dimensionality Reduction Using SVD Suppose we want to **represent a very large matrix $M$ by its SVD components $U$, $\Sigma$, and $V$**, but these matrices are also too large to store conveniently. The best way to **reduce the dimensionality** of the three matrices is to **set the smallest of the singular values to zero**. If we set the $s$

smallest singular values to 0 , then we can **also eliminate the corresponding $s$ columns of $U$ and $V$**.

The decomposition of **Example 2 has three singular values**. Suppose we want to reduce the number of dimensions to two. Then we **set the smallest of the singular values to zero.** The effect is that the **third column of $U'$** and the **third row of $V'^T$** are **multiplied only by 0's** when we perform the multiplication, so this row and this column may as well not be there. That is, the approximation to $M'$ obtained by using only the two largest singular values:

$$\underbrace{\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix}}_{M'} = \underbrace{\begin{bmatrix} .13 & .02 & -.01 \\ .41 & .07 & -.03 \\ .55 & .09 & -.04 \\ .68 & .11 & -.05 \\ .15 & -.59 & .65 \\ .07 & -.73 & -.67 \\ .07 & -.29 & .32 \end{bmatrix}}_{U'} \underbrace{\begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 0 \end{bmatrix}}_{\Sigma'} \underbrace{\begin{bmatrix} .56 & .59 & .56 & .09 & .09 \\ .12 & -.02 & .12 & -.69 & -.69 \\ .40 & -.80 & .40 & .09 & .09 \end{bmatrix}}_{V'^T}$$

$$= \begin{bmatrix} .13 & .02 \\ .41 & .07 \\ .55 & .09 \\ .68 & .11 \\ .15 & -.59 \\ .07 & -.73 \\ .07 & -.29 \end{bmatrix} \begin{bmatrix} 12.4 & 0 \\ 0 & 9.5 \end{bmatrix} \begin{bmatrix} .56 & .59 & .56 & .09 & .09 \\ .12 & -.02 & .12 & -.69 & -.69 \end{bmatrix}$$

$$= \begin{bmatrix} 0.93 & 0.95 & 0.93 & .014 & .014 \\ 2.93 & 2.99 & 2.93 & .000 & .000 \\ 3.92 & 4.01 & 3.92 & .026 & .026 \\ 4.84 & 4.96 & 4.84 & .040 & .040 \\ 0.37 & 1.21 & 0.37 & 4.04 & 4.04 \\ 0.35 & 0.65 & 0.35 & 4.87 & 4.87 \\ 0.16 & 0.57 & 0.16 & 1.98 & 1.98 \end{bmatrix}$$

The resulting matrix is quite close to the original matrix $M'$. Ideally, the entire difference is the result of making the last singular value be 0. However, in this simple example, **much of the difference is due to rounding error** caused by the fact that the decomposition of $M'$ was **only correct to two significant digits**.

# 6 Example 2 cont.: Movie Recommendation Using Features

In this section we shall look at how SVD can help us recommend the movies efficiently, with good accuracy. We make movie recommendation based on the following assumption: if a person **likes a certain movie**, then he **will also like the same type of movie**.

# 6.1 Recommend movie

## 6.1.1 Quincy

Let us make recommendation **based on the first two features in Example 3**. Quincy is not one of the people represented by the original matrix, but he wants to use the system to know what movies he would like. He has only seen one movie, The Matrix, and rated it 4 . Thus, we can represent Quincy by the vector $\mathbf{q} = [4, 0, 0, 0, 0]$, as if this were one of the rows of the original matrix.

We can **map Quincy into "feature space"** by multiplying him by the matrix $V$ of the decomposition.

In [6]:

```
# Quincy
M_Quincy = np.matrix([4,0,0,0,0]) # rateing matrix of Quincy
Quincy_feature = M_Quincy.dot(np.transpose(-v2[:2])) # use the first two columns
Quincy_feature
```

executed in 6ms, finished 21:39:31 2023-03-05

Out[6]:

```
matrix([[ 2.24903362, -0.50656553]])
```

We find $\mathbf{q}V = [2.25, -0.51]$. That is to say, **Quincy is high in science-fiction interest**, and not very interested in romance.

We now have a **representation of Quincy in feature space**, derived from, but different from his representation in the **original "movie space"**. One useful thing we can do is to **map his representation back into movie space** by multiplying $[2.25, -0.51]$ by $V^{\mathrm{T}}$.

In [7]:

```
Quincy_feature.dot(-v2[:2])
```

executed in 3ms, finished 21:39:31 2023-03-05

Out[7]:

```
matrix([[ 1.32869022,  1.31878766,  1.32869022, -0.14954027, -0.149540
27]])
```

This product is $[1.33, 1.32, 1.33, -0.15, -0.15]$. Thus, **the SVD decomposition suggests that Quincy would like Alien and Star Wars, but not Casablanca or Titanic**.

## 6.1.2 Leslie

Suppose Leslie assigns **rating 3 to Alien and rating 4 to Titanic**, giving us a representation of Leslie in "movie space" of $[0, 3, 0, 0, 4]$. Then the representation of Leslie in concept space is:

In [8]:

```
#Leslie
M_Leslie = np.matrix([0,3,0,0,4])
Leslie_feature = M_Leslie.dot(np.transpose(-v1[:2]))
Leslie_feature
```

executed in 3ms, finished 21:39:31 2023-03-05

Out[8]:

```
matrix([[1.73205081, 2.82842712]])
```

$\mathbf{q}V = [1.73, 2.83]$. **It seems Leslie prefer romance more than science-fiction.**

Then the predicted Leslie's rating of the movie is:

In [9]:

```
Leslie_feature.dot(-v2[:2])
```

executed in 4ms, finished 21:39:31 2023-03-05

Out[9]:

```
matrix([[0.61566421, 1.10823897, 0.61566421, 2.12293683, 2.12293683]])
```

For Leslie, the ratings on Casablanca or Titanic may higher than them on Matrix, Alien and Star Wars. Since Leslie has already watched the Titanic, it suggests that **Leslie will prefer to watch Casablanca**.

# 6.2 Recommend user

## 6.2.1 Quincy

Apart from the movie recommendation, we can also **find users similar to Quincy** in concept space. In other words, we can **recommend users with similar movie tastes to Quincy**. We can use $V$ to map all users in the dataset into concept space:

In [10]:

```
x2_feature = x2.dot(np.transpose(-v2[:2]))
x2_feature
```

executed in 3ms, finished 21:39:31 2023-03-05

Out[10]:

```
matrix([[ 1.71737671, -0.22451218],
        [ 5.15213013, -0.67353654],
        [ 6.86950685, -0.89804872],
        [ 8.58688356, -1.12256089],
        [ 1.9067881 ,  5.62055093],
        [ 0.90133537,  6.9537622 ],
        [ 0.95339405,  2.81027546]])
```

Joe (first row) maps to $[1.72, -0.22]$, and Jane (last row) maps to $[0.95, 2.81]$. Then, we can measure the similarity of users by their **cosine distance in concept space**.

The **cosine distance** and the code below has introduced in *Tutorial on Chapter 1 Data: type, quality, dis/similarity.*

In [11]:

```
def COSDistance(x,y):
    if len(x) != len(y):
        raise ValueError("Undefined for sequences of unequal length")
    x = np.array(x); y = np.array(y)
    return (x @ np.transpose(y)) / (np.linalg.norm(x)*np.linalg.norm(y))
```

executed in 2ms, finished 21:39:31 2023-03-05

In [12]:

```
# Calculate the cos distance between people in dataset and Quincy
for people in np.arange(0, len(x2_feature)):
    print(COSDistance(x2_feature[people], Quincy_feature)[0,0])
```

executed in 4ms, finished 21:39:31 2023-03-05

```
0.995812457586508
0.9958124575865079
0.995812457586508
0.995812457586508
0.10533291005166293
-0.09250782773698496
0.10533291005166293
```

For the case introduced above, note that the **concept vectors for Quincy and boys (first four rows)**, are **almost the same**. That is, their cosine distance is approximately 1. On the other hand, the **vectors for Quincy and girls (last three rows)**, have a **dot product of 0**, and therefore their angle is about 90 degrees. Therefore, we can **recommend those male users to Quincy** and they may become friends.

### 6.2.2 Leslie

Just replace the feature vector of Quincy by that of Leslie, then:

In [13]:

```
# Calculate the cos distance between people in dataset and Quincy
for people in np.arange(0, len(x2_feature)):
    print(COSDistance(x2_feature[people], Leslie_feature)[0,0])
```

executed in 3ms, finished 21:39:31 2023-03-05

```
0.40728076375769534
0.40728076375769523
0.40728076375769534
0.40728076375769534
0.9753711989030948
0.9128573731289783
0.9753711989030948
```

Therefore, **Leslie share the same movie tastes with the girls (last three rows)**, especilally the Jill (row 5) and Jane (row 7).

# 7 Appendix: Why does truncated SVD work?

In **truncated SVD**, the diagonal matrix $\Sigma$ is truncated by **removing the singular values below a certain threshold or beyond a certain rank**, which reduces the number of columns in $U$ and rows in $V^T$. The resulting truncated matrices are then used to **reconstruct an approximation (Low-rank approximation (https://en.wikipedia.org/w/index.php?search=Low-rank+approximation&title=Special%3ASearch&wprov=acrw1_-1)) of the original matrix**. The benefit of truncated SVD is that it can **reduce the computational complexity and storage requirements** of the full SVD algorithm, while still providing a good approximation of the original matrix.

Actually, by preserving the a few largest singular values and set the other to zero, the Frobenius norm (https://en.wikipedia.org/wiki/Matrix_norm#Frobenius_norm) $\left(\|A\|_{\mathrm{F}} = \sqrt{\sum_i^m \sum_j^n |a_{ij}|^2}\right)$ between the reconstructed matrix and original matrix is **minimized**. The result is referred to as the **matrix approximation lemma or Eckart–Young–Mirsky theorem**. Here we provide a brief proof:

**Proof:**

Let

$$M = U\Sigma V'$$

be the SVD of the $n \times r$ matrix $M$.

Since the **Frobenius norm** is invariant **under left- and right-multiplication by orthogonal matrices**, since **orthogonality by definition means preservation of the Euclidean norm** and the **Frobenius norm** (when squared) is both (a) **the sum of squared Euclidean norms of the rows** (and so is invariant under left multiplication, which preserves each row norm) and (b) **the sum of squared Euclidean norms of the columns** (and so is invariant under right multiplication, which preserves each column norm).

Therefore, for the **Frobenius norm**, whenever $\boldsymbol{P}$ is an $n \times n$ orthogonal matrix or $Q$ is an $r \times r$ orthogonal matrix, then

$$\left\|P'MQ\right\| = \|M\|$$

Let $A$ be the matrix reconstructed by the **some $k$ singular values**. Then, by the definition of the SVD and the norm, the orthogonality of $U$ and $V$ imply

$$\|M - A\|^2 = \left\|U'(M - A)V\right\|^2 = \left\|\Sigma - U'AV\right\|^2$$

Since $A$ is formulated to make $U'AV$ a diagonal matrix that agrees with the **some $k$ entries** of the diagonal matrix $\Sigma$, the right hand side is just the squared norm of $\Sigma$ after those $k$ diagonal entries have been zeroed out.

For the Frobenius norm (whose square is the sum of squared entries of its argument), the **squared norm** of this zeroed-out copy of $\Sigma$ is **the sum of squares of its remaining entries**, precisely

$$\left\|\Sigma - U'AV\right\|^2 = \sum_{j=k+1}^{r} \delta_j^2.$$

Thus, the **Frobenius norm of the reconstructing error is minimized** when the **singular values been zeroed-out are minimized**, that is, **removing the lowest singular values**.

A formal proof can be found in here (https://en.wikipedia.org/wiki/Low-rank_approximation#Proof_of_Eckart%E2%80%93Young%E2%80%93Mirsky_theorem_(for_Frobenius_norm)).

# 8 Summary

The **Singular Value Decomposition (SVD)** is a **matrix factorization method** that can be used to produce **low-dimensional approximations** of high-dimensional matrices.

The tutorial begins with the **definitions of SVD** and introduces the matrices $U$, $\Sigma$, and $V$ with their properties. The tutorial then explains how to compute the SVD of a matrix and how this is related to the eigenvalues of the symmetric matrices $M^T M$ and $M M^T$.

The tutorial also includes **examples of SVD applications**. The **first example is a rank-2 matrix** representing **movie ratings by users**, and it has two underlying features, **science-fiction and romance**. The matrix can be decomposed into $U$, $\Sigma$, and $V$, and since the rank of M is 2, the decomposition is straightforward.

The **second example has a rank of 3**. We can eliminate the third feature since it is insignificant to the dataset. In this section, we also learn how to **reduce dimensionality using SVD** by **setting the smallest of the singular values to zero**, which eliminates the corresponding columns of U and V.

Finally, the tutorial explains **how SVD can help in movie and user recommendations** by making recommendations based on the assumption that if a person likes a particular movie, they will likely enjoy the same type of movie. The tutorial illustrates **how to make recommendations based on the first two features** in Example 3.

In conclusion, SVD is a powerful tool that can be used to produce **low-dimensional approximations** of high-dimensional matrices. It facilitates the elimination of less important parts of the matrix representation, generating an approximate representation with **fewer dimensions**, and **simplifies the computations** required.