Many complex matrix operations cannot be solved efficiently or with stability using the limited precision of computers.

MATRIX FACTORIZATION/DECOMPOSITION

Many complex matrix operations cannot be solved efficiently or with stability using the limited precision of computers.

Matrix decompositions are methods that reduce a matrix into constituent parts that make it easier to calculate more complex matrix operations.

Matrix decomposition methods, also called matrix factorization methods, are a foundation of linear algebra in computers, even for basic operations such as solving systems of linear equations, calculating the inverse, and calculating the determinant of a matrix.

A matrix decomposition is a way of reducing a matrix into its constituent parts.

It is an approach that can simplify more complex matrix operations that can be performed on the decomposed matrix rather than on the original matrix itself.

Like factoring real values, there are many ways to decompose a matrix, hence there are a range of different matrix decomposition techniques.

Two simple and widely used matrix decomposition methods are the LU matrix decomposition and the QR matrix decomposition.

LU MATRIX DECOMPOSITION

The LU decomposition is for **square matrices** and decomposes a matrix into L and U components.

A = L . U

Where **A is the square matrix** that we wish to decompose, **L is the lower triangle** matrix and **U is the upper triangle matrix**.

The LU decomposition is found using an iterative numerical process and can fail for those matrices that cannot be decomposed or decomposed easily.

A variation of this decomposition that is numerically more stable to solve in practice is called the LUP decomposition, or the LU decomposition with partial pivoting.

A=P.L.U

The rows of the parent matrix are re-ordered to simplify the decomposition process and the additional P matrix specifies a way to permute the result or return the result to the original order. There are also other variations of the LU

The LU decomposition is often used to simplify the solving of systems of linear equations, such as finding the coefficients in a linear regression, as well as in calculating the determinant and inverse of a matrix.

The LU decomposition can be implemented in Python with the lu() function. More specifically, this function calculates an LPU decomposition.

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]

[[ 0.  1.  0.]
 [ 0.  0.  1.]     L
 [ 1.  0.  0.]]

[[ 1.          0.          0.        ]
 [ 0.14285714  1.          0.        ]  P
 [ 0.57142857  0.5         1.        ]]

[[  7.00000000e+00   8.00000000e+00   9.00000000e+00]
 [  0.00000000e+00   8.57142857e-01   1.71428571e+00]   Q
 [  0.00000000e+00   0.00000000e+00  -1.58603289e-16]]

[[ 1.  2.  3.]
 [ 4.  5.  6.]
 [ 7.  8.  9.]]
```

QR MATRIX DECOMPOSITION

The QR decomposition is for m x n matrices (not limited to square matrices) and decomposes a matrix into Q and R components.

A=Q.R

Where A is the matrix that we wish to decompose, Q a matrix with the size m x m, and R is an upper triangle matrix with the size m x n.

The QR decomposition is found using an iterative numerical method that can fail for those matrices that cannot be decomposed, or decomposed easily.

Like the LU decomposition, the QR decomposition is often used to solve systems of linear equations, although is not limited to square matrices.

The QR decomposition can be implemented in NumPy using the qr() function.

```
[[1 2]
 [3 4]
 [5 6]]

[[-0.16903085  0.89708523  0.40824829]
 [-0.50709255  0.27602622 -0.81649658]
 [-0.84515425 -0.34503278  0.40824829]]

[[-5.91607978 -7.43735744]
 [ 0.          0.82807867]
 [ 0.          0.        ]]

[[ 1.  2.]
 [ 3.  4.]
 [ 5.  6.]]
```

MATRIX COMPLETION

- **Matrix Completion** is a method for recovering lost information. It originates from machine learning and usually deals with highly **sparse** matrices. Missing or unknown data is estimated using the low-rank matrix of the known data.
- the task of filling in the missing entries of a partially observed matrix.
- A wide range of datasets are naturally organized in matrix form.

- One example is the movie-ratings matrix
- Given a ratings matrix in which each entry (i,j) represents the rating of movie j by customer i, if customer i has watched movie j and is otherwise missing, we would like to predict the remaining entries in order to make good recommendations to customers on what to watch next.
- Another example is the term-document matrix: the frequencies of words used in a collection of documents can be represented as a matrix, where each entry corresponds to the number of times the associated term appears in the indicated document.

- matrix completion often seeks to find the lowest rank matrix or, if the rank of the completed matrix is known, a matrix of rank r that matches the known entries.
- Other applications include computer vision, where missing pixels in images need to be reconstructed, detecting the global positioning of sensors in a network from partial distance information, and multiclass learning.
- The matrix completion problem is in general NP-hard, but under additional assumptions there are efficient algorithms that achieve exact reconstruction with high probability.

- a popular algorithm for matrix completion is Singular Value Threshold (SVT)

MIXTURE MODEL

- In statistics, a **mixture model** is a probabilistic model for representing the presence of subpopulations within an overall population, without requiring that an observed data set should identify the sub-population to which an individual observation belongs.
- Formally a mixture model corresponds to the mixture distribution that represents the probability distribution of observations in the overall population.
- However, while problems associated with "mixture distributions" relate to deriving the properties of the overall population from those of the sub-populations, "mixture models" are used to make statistical inferences about the properties of the sub-populations given only observations on the pooled population, without sub-population identity information.

General mixture model

A typical finite-dimensional mixture model is a hierarchical model consisting of the following components:

•*N* random variables that are observed, each distributed according to a mixture of *K* components, with the components belonging to the same parametric family of distributions (e.g., all normal, all Zipfian, etc.) but with different parameters

•*N* random latent variables specifying the identity of the mixture component of each observation, each distributed according to a *K*-dimensional categorical distribution

•A set of *K* mixture weights, which are probabilities that sum to 1.

•A set of *K* parameters, each specifying the parameter of the corresponding mixture component. In many cases, each "parameter" is actually a set of parameters. For example, if the mixture components are Gaussian distributions, there will be a mean and variance for each component. If the mixture components are categorical distributions (e.g., when each observation is a token from a finite alphabet of size *V*), there will be a vector of *V* probabilities summing to 1.

EXAMPLE:

- GAUSSIAN MIXTURE MODEL

- CATEGORICAL MIXTURE MODEL