Dimensionality Reduction

- Then number of points and features in the real datasets is usually large. The speed of calculation and storage is very slow if implemented in these datasets, Hence, Dimensionality Reduction is important step in Machine Learning
- Dimensionality reduction will implement the function *f* such that $f:\mathbb{R}^D \to \mathbb{R}^K$ with $K < D$. It can be applied in compression problems in which $\mathbf{x} \in \mathbb{R}^D$ can be compressed and deducted from $\mathbf{z} \in \mathbb{R}^K$, feature selection or feature extraction so that classification in a problem is improved
- **Principle Component Analysis** and **Linear Discriminant Analysis** will be mentioned in the following chapters. First of all, there is very important method to analyze the matrix – singular value decomposition

Singular Value Decomposition

Introduction

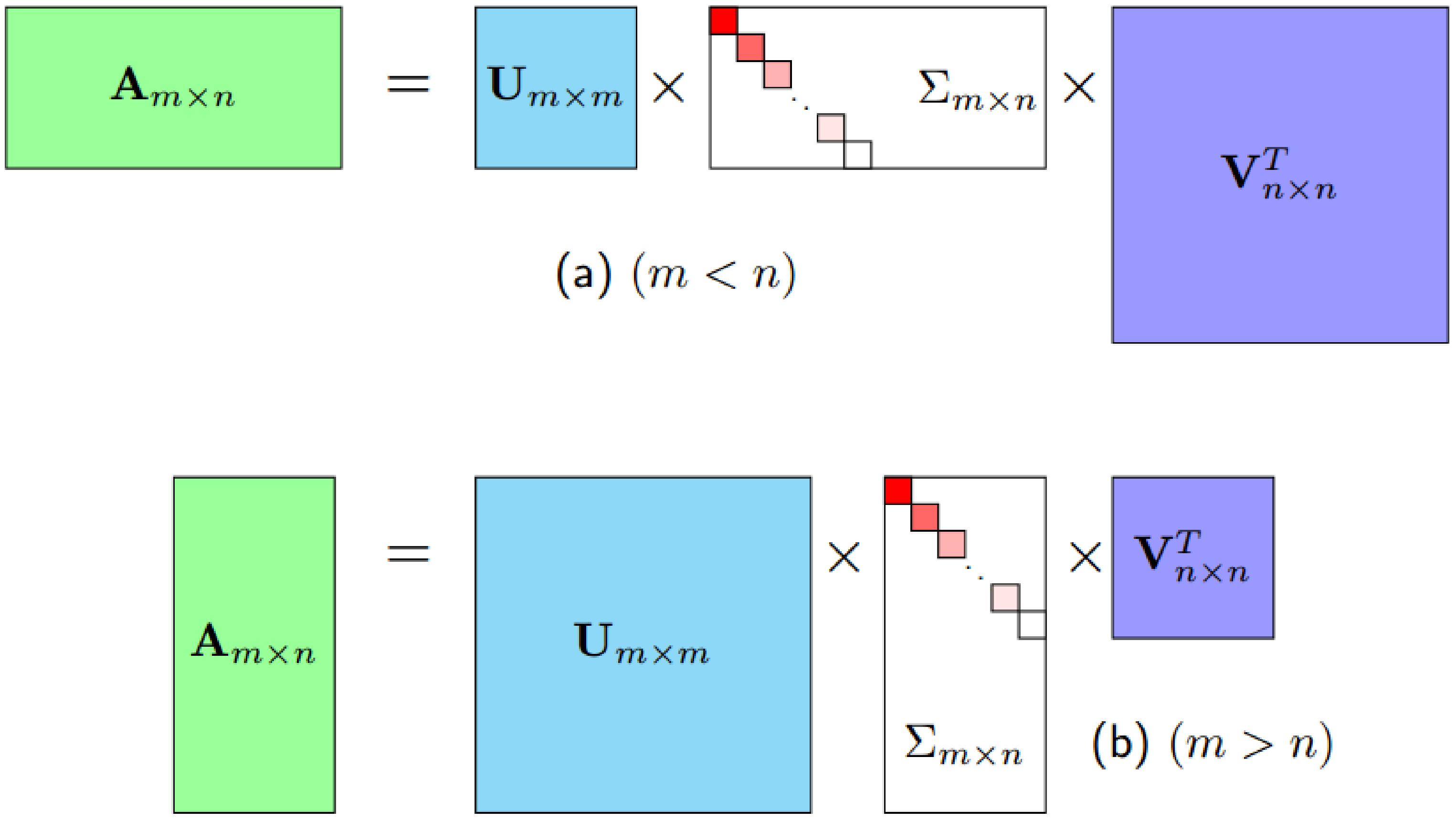
- The square matrix $A \in \mathbb{R}^{n\times n}$ is diagonalizable if diagonal matrix $D \in \mathbb{R}^{r\times r}$ and invertible matrix $P \in \mathbb{R}^{n\times r}$ exist such that

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

- It can be rewritten: $AP = PD$. Let $\mathbf{p}_i, \mathbf{d}_i$ be the column *i*th of matrix P and D and d_{ii} be the value *i*th of $\mathbf{d}_i$. Columns in left clause need to be equal columns in right clause, so we have $A\mathbf{p}_i = P\mathbf{d}_i = d_{ii}\mathbf{p}_i$. Second "=" occurs because matrix D is diagonal matrix
- **Theorem 1**: Rank of matrix A is the number of value different 0 in the diagonal matrix D. Proof: Let A be a matrix and for each $i = 1,\dots,n$, let d_{ii} be an eigenvector of A associated to an eigenvalue $\mathbf{p}_i$. Prove that if d_{ii} are all distinct, then $\{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_n\}$ are linearly independent
  - We do this by induction on n. For base case, $n = 2$. Let $x_1\mathbf{p}_1 + x_2\mathbf{p}_2 = \mathbf{0}$ *(1)* be the linear combination of $\mathbf{p}_i$. If A is multiplied to *(1)*, $A(x_1\mathbf{p}_1 + x_2\mathbf{p}_2) = d_{11}x_1\mathbf{p}_1 + d_{22}x_2\mathbf{p}_2 = \mathbf{0}$ *(2)*. On the other hand, if d_{11} is multiplied to *(1)*, $d_{11}(x_1\mathbf{p}_1 + x_2\mathbf{p}_2) = d_{11}x_1\mathbf{p}_1 + d_{11}x_2\mathbf{p}_2 = \mathbf{0}$ *(3)*. Take *(2) – (3)*, we have $d_{22}x_2\mathbf{p}_2 - d_{11}x_2\mathbf{p}_2 = (d_{22} - d_{11})x_2\mathbf{p}_2 = \mathbf{0}$
  - Because d_{ii} are all distinct, $(d_{22} - d_{11}) \neq 0$. It follows that $x_2 = 0$. Based on *(1)*, $x_1 = 0$, so $\{\mathbf{p}_1, \mathbf{p}_2\}$ are linearly independent
  - Suppose the statement is valid on m with $2 \le m \le n$ .We will show that it holds for $m+1$
  - Let $\sum_{i=1}^{m+1} x_i\mathbf{p}_i = \mathbf{0}$ *(4)* be the linear combination of $\mathbf{p}_i$. Because we suppose linear independence is valid on m, so $\{x_1, x_2, \dots, x_m\} = 0$. Based on *(4)*, $x_{m+1}\mathbf{p}_{m+1} = \mathbf{0}$, so $x_{m+1} = 0$. Hence, $\{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_{m+1}\}$ is linearly independent
  - Theorem 1 is proved because rank is the number of linearly independent vectors in a matrix
- The analysis method is called Eigen Decomposition, but it requires matrix A must be square matrix and have *r* eigenvalues
- Matrix Factorization or Matrix Decomposition is applied in numerous problems such as Recommender System, Dimensionality reduction, Compression, Clustering, Features Analysis
- The chapter will introduce one of the popular method of Matrix Factorization that is Singular Value Decomposition which does not require the square matrix constraint

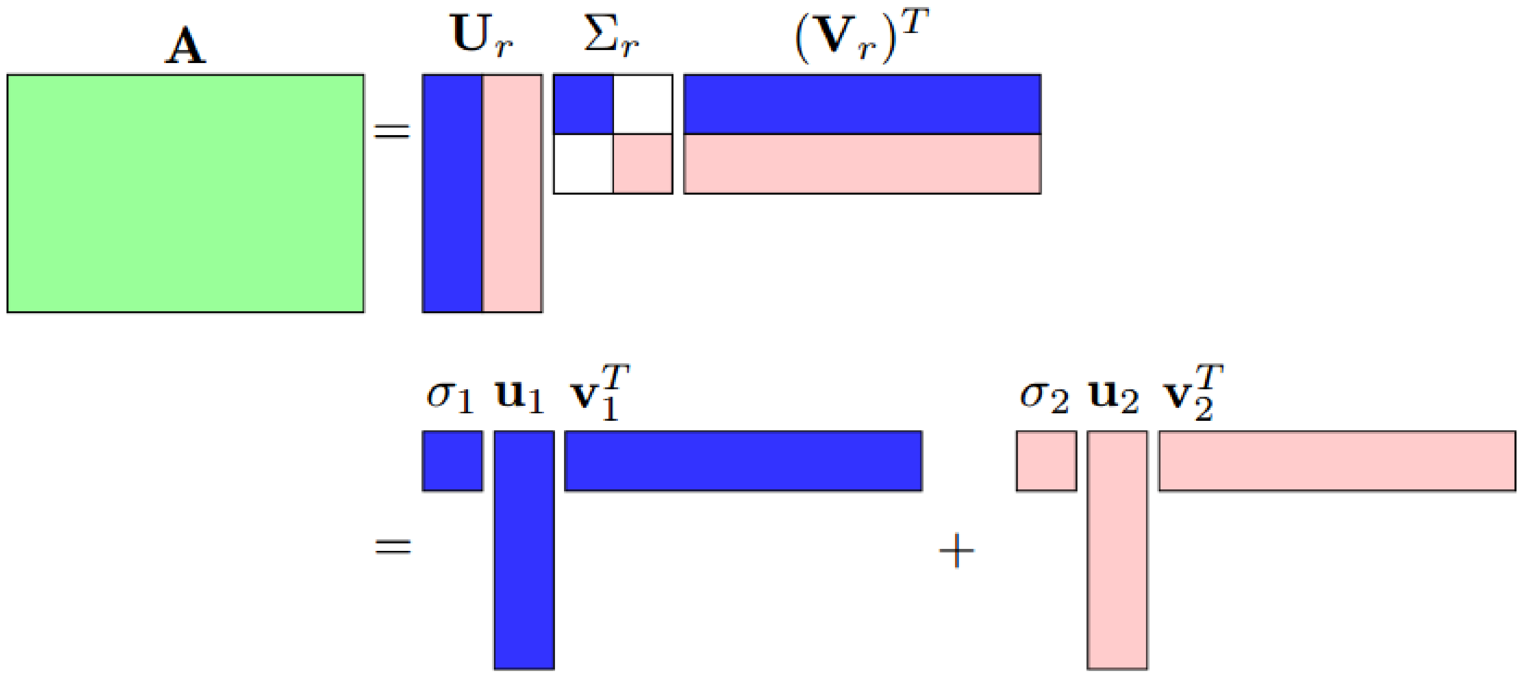
Definition

- $A_{m\times n}$ is the notation for $A \in \mathbb{R}^{m\times n}$
- Any $A_{m\times n}$ can be decomposed: $A_{m\times n} = U_{m\times m}\sum_{m\times n}(V_{n\times n})^T$ in which U, V is orthogonal matrix, $\sum$ is diagonal matrix same dimensions with $A_{m\times n}$. The values in the diagonal of matrix $\sum$ is ≥ 0 and in descending order $\sigma_1 \ge \sigma_2 \ge \dots \ge \sigma_r \ge 0 = 0 = \dots = 0$. Rank of matrix A is the number of value different 0 in the diagonal matrix $\sum$



Hình 20.1: SVD cho ma trận $\mathbf{A}$ khi: $m < n$ (hình trên), và $m > n$ (hình dưới). $\mathbf{\Sigma}$ là một ma trận đường chéo với các phần tử trên đó giảm dần và không âm. Màu đỏ càng đậm thể hiện giá trị càng cao. Các ô màu trắng trên ma trận này thể hiện giá trị 0.

- Based on $A_{m\times n} = U_{m\times m}\sum_{m\times n}(V_{n\times n})^T$, we have $AA^T = U\sum V^T\left(U\sum V^T\right)^T = U\sum V^TV\sum^TU^T = U\sum \sum^TU^{-1}$
- $\sum \sum^T$ is a diagonal matrix with diagonal values of $\sigma_1^2, \sigma_2^2, \ldots, \sigma_m^2$. Hence, $U\sum \sum^TU^{-1}$ is the Eigen decomposition of AA^T and $\sigma_1^2, \sigma_2^2, \ldots, \sigma_m^2$ is the Eigen values of AA^T. AA^T is PSD. σ_i is square root of Eigen values of AA^T, or also called **singular value** of matrix A. The columns of U is the Eigen vectors of AA^T, or called **left-singular** vectors of A. Likewise, $A^TA = V\sum \sum^TV^{-1}$, The columns of V is the Eigen vectors of A^TA, or called **right-singular** vectors of A
- Proof: The root of Eigen values of AA^T is singular value of matrix A
  - We have $A\mathbf{p}_i = d_{ii}\mathbf{p}_i \rightarrow A^TA\mathbf{p}_i = d_{ii}A\mathbf{p}_i = d_{ii}d_{ii}\mathbf{p}_i = d_{ii}^2\mathbf{p}_i$
  - Therefore, d_{ii}^2 is Eigen value of A^TA, so singular value of A is $\sqrt{d_{ii}^2} = d_{ii}$
- Compact SVD
  - $A_{m\times n} = U_{m\times m}\sum_{m\times n}(V_{n\times n})^T$ can be rewritten $A = \sigma_1\mathbf{u}_1\mathbf{v}_1^T + \sigma_2\mathbf{u}_2\mathbf{v}_2^T + \ldots + \sigma_r\mathbf{u}_r\mathbf{v}_r^T$ in which $\mathbf{u}_\mathbf{x}, \mathbf{v}_\mathbf{x}$ is column x of matrix U, V and the rank of matrix $\mathbf{u}_\mathbf{x}\mathbf{v}_\mathbf{x}^T$ is 1
  - Based on $A = \sigma_1\mathbf{u}_1\mathbf{v}_1^T + \sigma_2\mathbf{u}_2\mathbf{v}_2^T + \ldots + \sigma_r\mathbf{u}_r\mathbf{v}_r^T$, A is just dependent on first *r* columns of U, V and *r* diagonal values of $\sum$. A can be rewritten in short and also called **Compact SVD**: $A_r = U_r\sum_r(V_r)^T$ in which U_r, V_r is first *r* columns of matrix U, V, $\sum_r$ is first *r* columns and *r* rows of matrix $\sum$. If rank of matrix A *r* is much smaller than the number of columns and rows $r \ll m, n$, it's advantage of storage. The example: $m = 4, n = 6, r = 2$


$\mathbf{A}$
$\mathbf{U}_r$
Σ_r
$(\mathbf{V}_r)^T$
$=$
σ_1 $\mathbf{u}_1$ $\mathbf{v}_1^T$
σ_2 $\mathbf{u}_2$ $\mathbf{v}_2^T$
$+$

Hình 20.2: Biểu diễn SVD dạng thu gọn và biểu diễn ma trận dưới dạng tổng các ma trận có rank bằng 1. Các khối ma trận đặt cạnh nhau thể hiện phép nhân ma trận.

- Truncated SVD
  - Diagonal matrix $\sum$ has diagonal values of $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r \geq 0 = 0 = \ldots = 0$, but small portion of σ is much larger than 0, others are almost equal or equal 0. So, we can approximate A by sum of $k < r$
    $A \approx A_k = U_k \sum_k (V_k)^T = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T + \ldots + \sigma_k \mathbf{u}_k \mathbf{v}_k^T$
  - Theorem 2: Rank-*r* matrix A is approximated by truncated SVD of $k < r$, deviation of the approximation:
    $\|A - A_k\|_F^2 = \sum_{i=k+1}^{r} \sigma_i^2$. The following is proof

    Properties of trace: $\|X\|_F^2 = trace\left(XX^T\right)$ and $trace(XY) = trace(YX)$

    $$\|A - A_k\|_F^2 = \left\|\sum_{i=k+1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T\right\|_F^2 = trace\left[\left(\sum_{i=k+1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T\right)\left(\sum_{j=k+1}^{r} \sigma_j \mathbf{u}_j \mathbf{v}_j^T\right)^T\right] = trace\left(\sum_{i=k+1}^{r}\sum_{j=k+1}^{r} \sigma_i \sigma_j \mathbf{u}_i \mathbf{v}_i^T \mathbf{v}_j \mathbf{u}_j^T\right)$$

    Because $trace\left(\mathbf{v}_i^T \mathbf{v}_i\right) \approx 1$, $trace\left(\mathbf{v}_i^T \mathbf{v}_j\right) \approx 0$ and $trace\left(\mathbf{u}_i \mathbf{u}_j^T\right) \approx 0$

```
np.matrix.trace(V[:, 1].reshape(1, -1) * V[:, 1].reshape(-1, 1))
```

```
1.0000000000000004
```

```
np.matrix.trace(V[:, 0].reshape(1, -1) * V[:, 0].reshape(-1, 1))
```

```
1.0000000000000002
```

```
np.matrix.trace(V[:, 0].reshape(1, -1) * V[:, 1].reshape(-1, 1))
```

```
9.71445146547012e-17
```

```
np.matrix.trace(V[:, 1].reshape(1, -1) * V[:, 2].reshape(-1, 1))
```

```
9.71445146547012e-17
```

$$trace\left(\sum_{i=k+1}^{r}\sum_{j=k+1}^{r} \sigma_i \sigma_j \mathbf{u}_i \mathbf{v}_i^T \mathbf{v}_j \mathbf{u}_j^T\right) = trace\left(\sum_{i=k+1}^{r} \sigma_i^2 \mathbf{u}_i \mathbf{u}_i^T\right)$$

Because $trace\left(\mathbf{u}_i \mathbf{u}_i^T\right) \approx 1$

```
np.matrix.trace(U[:, 1].reshape(1, -1) * U[:, 1].reshape(-1, 1))
```

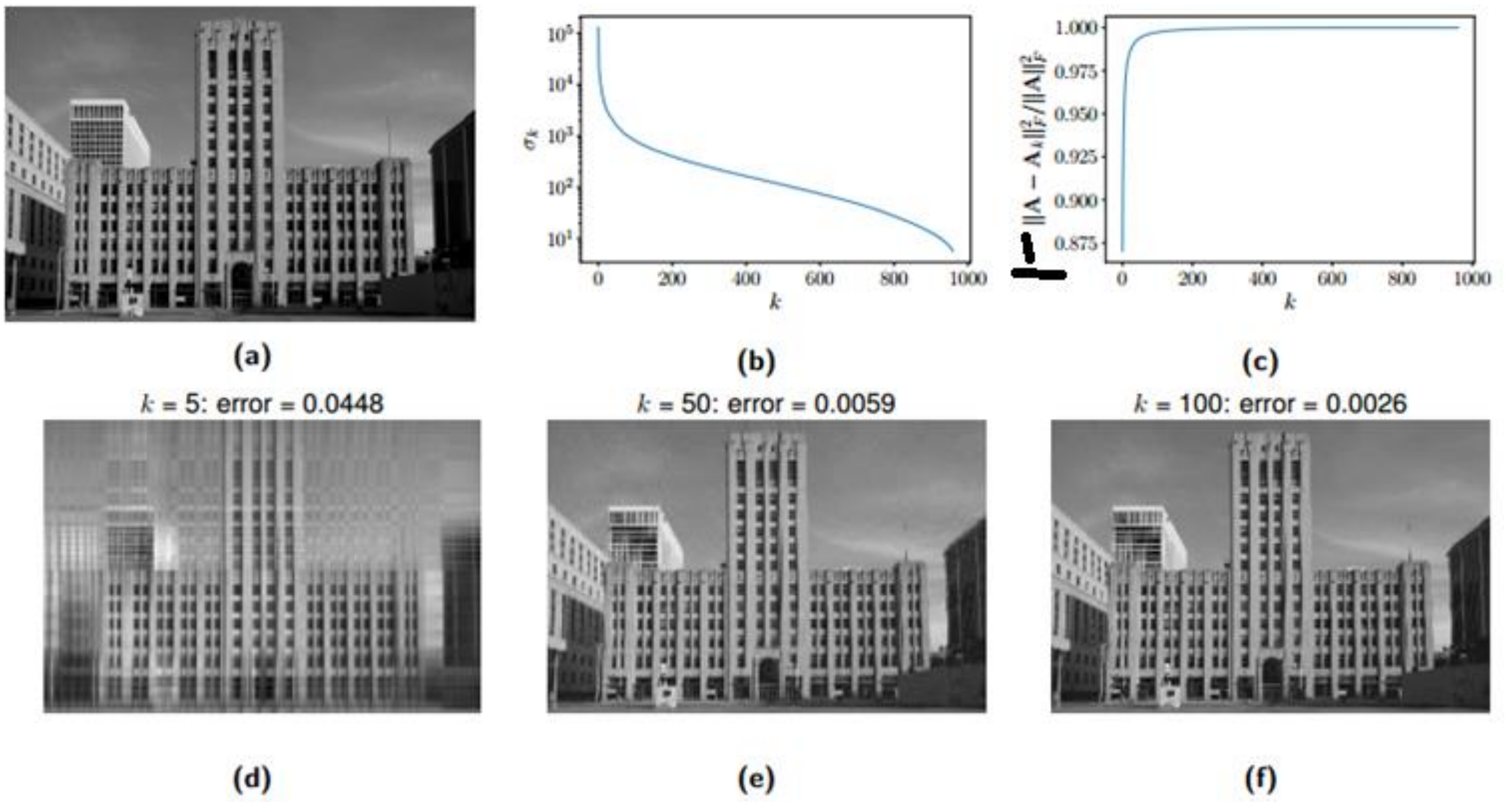
```
0.9999999999999991
```

```
np.matrix.trace(U[:, 0].reshape(1, -1) * U[:, 0].reshape(-1, 1))
```

```
0.9999999999999991
```

$$trace\left(\sum_{i=k+1}^{r}\sigma_i^2\mathbf{u}_i\mathbf{u}_i^T\right)=trace\left(\sum_{i=k+1}^{r}\sigma_i^2\right)=\sum_{i=k+1}^{r}\sigma_i^2$$

- If $k=0$, $\|A\|_F^2=\sum_{i=1}^{r}\sigma_i^2$, so $\frac{\|A-A_k\|_F^2}{\|A\|_F^2}=\frac{\sum_{i=k+1}^{r}\sigma_i^2}{\sum_{i=1}^{r}\sigma_i^2}$. Hence, the deviation of the approximation will be smaller if singular values, which are truncated, are small compared to value of other singular values. For example, if we want to keep more than 90% of matrix A, we choose k min such that $\frac{\sum_{i=1}^{k}\sigma_i^2}{\sum_{i=1}^{r}\sigma_i^2}\geq 0.9$
- When k is small, rank of A is k which is small, so Truncated SVD is also called **low-rank approximation**



Hình 20.3: Ví dụ về SVD cho ảnh. (a) Bức ảnh gốc là một ảnh xám, là một ma trận cỡ 960×1440. (b) Giá trị của các singular values của ma trận ảnh theo logscale. Có thể thấy rằng các singular value giảm nhanh ở khoảng $k=200$. (c) Biểu diễn lượng thông tin được giữ lại khi chọn các k khác nhau. Có thể nhận thấy từ khoảng $k=200$, lượng thông tin giữ lại là gần bằng 1. Vậy ta có thể xấp xỉ ma trận ảnh này bằng một ma trận có rank nhỏ hơn. (d), (e), (f) Các ảnh xấp xỉ với k lần lượt là 5, 50, 100.

- SVD for Image Compression: To save the truncated SVD image, we just save $U_k\in\mathbb{R}^{m\times k}, V_k\in\mathbb{R}^{n\times k}, \sum_k\in\mathbb{R}^k$ instead of $\mathbb{R}^{k\times k}$ because $\sum_k$ is diagonal matrix. Number of values which needs to be saved is $k(m+n+1)<mn$ which is number of

values of original matrix $A \in \mathbb{R}^{m \times n}$, so Image Compression Ratio is $\frac{4k\left(m+n+1\right)}{mn}$ if truncated SVD image save the real value of 4 bytes and original image of numeric of 1 byte

- Moreover, SVD is applied in Pseudo Inverse (Refer)
- It takes time to calculate truncated SVD (figure out all Eigen values, vectors and choose first *k* out of Eigen values, vectors), Power method for approximating eigenvalues (Refer) can calculating fast the first *k* out of Eigen values and vectors of AA^T