Principal Component Analysis and Autoencoders

Shuiwang Ji
Department of Computer Science & Engineering
Texas A&M University

Orthogonal Matrices

1. An orthogonal matrix is a square matrix whose columns and rows are orthogonal unit vectors, *i.e.*, orthonormal vectors. That is, if a matrix $\boldsymbol{Q}$ is an orthogonal matrix, we have

$$\boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{Q}\boldsymbol{Q}^T = \boldsymbol{I}.$$

2. It leads to $\boldsymbol{Q}^{-1} = \boldsymbol{Q}^T$, which is a very useful property as it provides an easy way to compute the inverse.
3. For an orthogonal $n \times n$ matrix $\boldsymbol{Q} = [\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_n]$, where $\boldsymbol{q}_i \in \mathbb{R}^n$, $i = 1, 2, \ldots, n$, it is easy to see that $\boldsymbol{q}_i^T\boldsymbol{q}_j = 0$ when $i \neq j$ and $\boldsymbol{q}_i^T\boldsymbol{q}_i = 1$.
4. Furthermore, suppose $\boldsymbol{Q}_1 = [\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_i]$ and $\boldsymbol{Q}_2 = [\boldsymbol{q}_{i+1}, \boldsymbol{q}_{i+2}, \ldots, \boldsymbol{q}_n]$, we have $\boldsymbol{Q}_1^T\boldsymbol{Q}_1 = \boldsymbol{I}$, $\boldsymbol{Q}_2^T\boldsymbol{Q}_2 = \boldsymbol{I}$, but $\boldsymbol{Q}_1\boldsymbol{Q}_1^T \neq \boldsymbol{I}$, $\boldsymbol{Q}_2\boldsymbol{Q}_2^T \neq \boldsymbol{I}$.

Eigen-Decomposition

1. A square $n \times n$ matrix $\boldsymbol{S}$ with n linearly independent eigenvectors can be factorized as

$$\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{-1},$$

   where $\boldsymbol{Q}$ is the square $n \times n$ matrix whose columns are eigenvectors of $\boldsymbol{S}$, and $\boldsymbol{\Lambda}$ is the diagonal matrix whose diagonal elements are the corresponding eigenvalues.
2. Note that only diagonalizable matrices can be factorized in this way.
3. If $\boldsymbol{S}$ is a symmetric matrix, its eigenvectors are orthogonal. Thus $\boldsymbol{Q}$ is an orthogonal matrix and we have

$$\boldsymbol{S} = \boldsymbol{Q}\boldsymbol{\Lambda}\boldsymbol{Q}^{T}.$$

Singular Value Decomposition

The singular value decomposition (SVD) of an $m \times n$ **real** matrix (without loss of generality, we assume $m \geq n$) can be written as

$$\boldsymbol{R} = \boldsymbol{U}\tilde{\boldsymbol{\Sigma}}\boldsymbol{V}^T,$$

where $\boldsymbol{U}$ is an orthogonal $m \times m$ matrix, $\boldsymbol{V}$ is an orthogonal $n \times n$ matrix, and $\tilde{\boldsymbol{\Sigma}}$ is a diagonal $m \times n$ matrix with non-negative real values on diagonal. That is,

$$\boldsymbol{U}^T\boldsymbol{U} = \boldsymbol{U}\boldsymbol{U}^T = \boldsymbol{I}_{m\times m},$$
$$\boldsymbol{V}^T\boldsymbol{V} = \boldsymbol{V}\boldsymbol{V}^T = \boldsymbol{I}_{n\times n},$$

$$\tilde{\boldsymbol{\Sigma}} = \begin{bmatrix} \boldsymbol{\Sigma}_{n\times n} \\ 0 \end{bmatrix}_{m\times n}, \boldsymbol{\Sigma}_{n\times n} = \begin{bmatrix} \sigma_1 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & \dots & 0 \\ 0 & 0 & \sigma_3 & \dots & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \dots & \sigma_n \end{bmatrix}, \tag{1}$$

where $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$ are known as singular values. If $\text{rank}(\boldsymbol{R}) = r$ $(r \leq n)$, we have $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$ and $\sigma_{r+1} = \sigma_{r+2} = \cdots = \sigma_n = 0$.

Relation to Eigen-Decomposition

The columns of $\boldsymbol{U}$ (left-singular vectors) are orthonormal eigenvectors of $\boldsymbol{R}\boldsymbol{R}^T$, and the columns of $\boldsymbol{V}$ (right-singular vectors) are orthonormal eigenvectors of $\boldsymbol{R}^T\boldsymbol{R}$. In other words, we have

$$\boldsymbol{R}\boldsymbol{R}^T = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^{-1},$$

$$\boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^{-1}.$$

It is easy to verify them as we have

$$\boldsymbol{R}^T\boldsymbol{R} = (\boldsymbol{U}\begin{bmatrix}\boldsymbol{\Sigma}\\ 0\end{bmatrix}\boldsymbol{V}^T)^T\boldsymbol{U}\begin{bmatrix}\boldsymbol{\Sigma}\\ 0\end{bmatrix}\boldsymbol{V}^T = \boldsymbol{V}(\begin{bmatrix}\boldsymbol{\Sigma} & 0\end{bmatrix}\begin{bmatrix}\boldsymbol{\Sigma}\\ 0\end{bmatrix})\boldsymbol{V}^T = \boldsymbol{V}\boldsymbol{\Sigma}^2\boldsymbol{V}^T,$$

$$\boldsymbol{R}\boldsymbol{R}^T = \boldsymbol{U}\begin{bmatrix}\boldsymbol{\Sigma}\\ 0\end{bmatrix}\boldsymbol{V}^T(\boldsymbol{U}\begin{bmatrix}\boldsymbol{\Sigma}\\ 0\end{bmatrix}\boldsymbol{V}^T)^T = \boldsymbol{U}(\begin{bmatrix}\boldsymbol{\Sigma}\\ 0\end{bmatrix}\begin{bmatrix}\boldsymbol{\Sigma} & 0\end{bmatrix})\boldsymbol{U}^T = \boldsymbol{U}\begin{bmatrix}\boldsymbol{\Sigma}^2 & 0\\ 0 & 0\end{bmatrix}\boldsymbol{U}$$

and $\boldsymbol{V}^T = \boldsymbol{V}^{-1}$, $\boldsymbol{U}^T = \boldsymbol{U}^{-1}$.

SVD and eigen-decomposition

1. Under what conditions are SVD and eigen-decomposition the same? First, $\boldsymbol{R}$ is a symmetric matrix, *i.e.*, $\boldsymbol{R} = \boldsymbol{R}^T$. Second, $\boldsymbol{R}$ is a positive semi-definite matrix, *i.e.*, $\forall \boldsymbol{x} \in \mathbb{R}^n, \boldsymbol{x}^T \boldsymbol{R} \boldsymbol{x} \geq 0$.
2. The difference between $\boldsymbol{\Lambda}$ in eigen-decomposition and $\boldsymbol{\Sigma}$ in SVD is that, the diagonal entries of $\boldsymbol{\Lambda}$ can be negative, while the diagonal entries of $\boldsymbol{\Sigma}$ are non-negative. What are the fundamental reasons underlying this difference? Why the requirements on the singular values in SVD (non-negative and in sorted order) do not prevent the generality of SVD?

Compact SVD

If $\text{rank}(\boldsymbol{R}) = r$ $(r \leq n)$, we have

$$\begin{aligned}
\boldsymbol{R} &= \boldsymbol{U}\tilde{\boldsymbol{\Sigma}}\boldsymbol{V}^T \\
&= [\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_r, \ldots, \boldsymbol{u}_m] \begin{bmatrix} \sigma_1 & \ldots & 0 & & & \\ \vdots & \ddots & \vdots & & \ddots & \\ 0 & \ldots & \sigma_r & & & \\ & & & 0 & \ldots & 0 \\ & \ddots & & \vdots & \ddots & \vdots \\ & & & 0 & \ldots & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_1^T \\ \boldsymbol{v}_2^T \\ \vdots \\ \boldsymbol{v}_r^T \\ \vdots \\ \boldsymbol{v}_n^T \end{bmatrix}.
\end{aligned}$$

By removing zero components, we obtain

$$\begin{aligned}
\boldsymbol{R} &= \boldsymbol{U}_r\boldsymbol{\Sigma}_r\boldsymbol{V}_r^T \\
&= [\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_r] \begin{bmatrix} \sigma_1 & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & \sigma_r \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_1^T \\ \boldsymbol{v}_2^T \\ \vdots \\ \boldsymbol{v}_r^T \end{bmatrix} \\
&= [\sigma_1\boldsymbol{u}_1, \sigma_2\boldsymbol{u}_2, \ldots, \sigma_r\boldsymbol{u}_r] \begin{bmatrix} \boldsymbol{v}_1^T \\ \boldsymbol{v}_2^T \\ \vdots \\ \boldsymbol{v}_r^T \end{bmatrix} = \sum_{i=1}^{r} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T,
\end{aligned}$$

where $\text{rank}(\sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T) = 1$, $i = 1, 2, \ldots, r$.

Truncated SVD and Best Low-Rank Approximation

We can also approximate the matrix $\boldsymbol{R}$ with the k largest singular values as

$$\boldsymbol{R}_k = \boldsymbol{U}_k \boldsymbol{\Sigma}_k \boldsymbol{V}_k^T = \sum_{i=1}^{k} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T .$$

Apparently, $\boldsymbol{R} \neq \boldsymbol{R}_k$ unless $\text{rank}(\boldsymbol{R}) = k$. This approximation is the best in following sense:

$$\begin{aligned} \min_{\boldsymbol{B}:\text{rank}(\boldsymbol{B})\leq k} ||\boldsymbol{R} - \boldsymbol{B}||_F &= ||\boldsymbol{R} - \boldsymbol{R}_k||_F = \sqrt{\sum_{i=k+1}^{n} \sigma_i^2}, \\ \min_{\boldsymbol{B}:\text{rank}(\boldsymbol{B})\leq k} ||\boldsymbol{R} - \boldsymbol{B}||_2 &= ||\boldsymbol{R} - \boldsymbol{R}_k||_2 = \sigma_{k+1}, \end{aligned}$$

where $||\cdot||_F$ denotes the Frobenius norm and $||\cdot||_2$ denotes the spectral norm, defined as the largest singular value of the matrix. That is, $\boldsymbol{R}_k$ is the best rank-k approximation to $\boldsymbol{R}$ in terms of both the Frobenius norm and spectral norm. Note the difference in terms of approximation errors when different matrix norms are used.

What is PCA?

1. Principal Component Analysis (PCA) is a statistical procedure that can be used to achieve feature (dimensionality) reduction.
2. Note, feature reduction is different from feature selection. After feature reduction, we still use all the features, while feature selection selects a subset of features to use.
3. The goal of PCA is to project the high-dimensional features to a lower-dimensional space with maximal variance and minimum reconstruction error simultaneously.
4. We derive PCA based on maximizing variance, and then we show the solution also minimizes reconstruction error.
5. In machine learning, PCA is an unsupervised learning technique, and therefore does not need labels.

PCA to 1D

1. To introduce PCA, we start from the simple case where PCA projects the features to a 1-dimensional space.
2. Formally, suppose we have n p-dimensional ($p > 1$) features $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n \in \mathbb{R}^p$.
3. Let $\boldsymbol{a} \in \mathbb{R}^p$ represent a projection that $\boldsymbol{a}^T \boldsymbol{x}_i = z_i$, $i = 1, 2, \ldots, n$ where $z_1, z_2, \ldots, z_n \in \mathbb{R}^1$.
4. PCA aims to solve

$$\boldsymbol{a}^* = \arg\max_{||\boldsymbol{a}||=1} \frac{1}{n} \sum_{i=1}^{n} (z_i - \bar{z})^2.$$

5. Note that the variance of the reduced data is

$$\frac{1}{n} \sum_{i=1}^{n} (z_i - \bar{z})^2,$$

which means that PCA tries to find the projection with the maximum variance in reduced data.

PCA to 1D

Since

$$\bar{z} = \frac{1}{n}\sum_{i=1}^{n} z_i = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{a}^T \boldsymbol{x}_i = \boldsymbol{a}^T\left(\frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i\right) = \boldsymbol{a}^T \bar{\boldsymbol{x}},$$

the problem can be written as

$$\begin{aligned}
\boldsymbol{a}^* &= \arg\max_{||\boldsymbol{a}||=1} \frac{1}{n}\sum_{i=1}^{n}(z_i - \bar{z})^2 \\
&= \arg\max_{||\boldsymbol{a}||=1} \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{a}^T\boldsymbol{x}_i - \bar{z})^2 \\
&= \arg\max_{||\boldsymbol{a}||=1} \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{a}^T\boldsymbol{x}_i - \boldsymbol{a}^T\bar{\boldsymbol{x}})^2 \\
&= \arg\max_{||\boldsymbol{a}||=1} \frac{1}{n}\sum_{i=1}^{n}\boldsymbol{a}^T(\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T\boldsymbol{a} \\
&= \arg\max_{||\boldsymbol{a}||=1} \boldsymbol{a}^T \underbrace{\left(\frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T\right)}_{p\times p \text{ covariance matrix}} \boldsymbol{a} \\
&= \arg\max_{||\boldsymbol{a}||=1} \boldsymbol{a}^T\boldsymbol{C}\boldsymbol{a},
\end{aligned}$$

where $\boldsymbol{C} = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T$, denotes the covariance matrix.

PCA to k-dimensional space

1. What if we want to project the features to a k-dimensional space? Then the PCA problem becomes

$$\boldsymbol{A}^* = \underset{\boldsymbol{A} \in \mathbb{R}^{p \times k} : \boldsymbol{A}^T \boldsymbol{A} = \boldsymbol{I}_k}{\arg\max} \operatorname{trace}\left(\boldsymbol{A}^T \boldsymbol{C} \boldsymbol{A}\right), \tag{2}$$

where $\boldsymbol{A} = [\boldsymbol{a}_1, \boldsymbol{a}_2, \cdots, \boldsymbol{a}_k] \in \mathbb{R}^{p \times k}$. Note that when projecting onto k-dimensional space, PCA requires different projection vectors to be orthogonal. Also, the trace above is the sum of the variances after projecting the data to each of the k directions as

$$\operatorname{trace}\left(\boldsymbol{A}^T \boldsymbol{C} \boldsymbol{A}\right) = \sum_{i=1}^{k} \left(\boldsymbol{a}_i^T \boldsymbol{C} \boldsymbol{a}_i\right).$$

Ky Fan Theorem

1. Solving the problem in Eqn. (2) requires the follow theorem.
2. **Theorem.** (*Ky Fan*) Let $\boldsymbol{H} \in \mathbb{R}^{n \times n}$ be a symmetric matrix with eigenvalues

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n,$$

and the corresponding eigenvectors $\boldsymbol{U} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n]$. Then

$$\lambda_1 + \cdots \lambda_k = \max_{\boldsymbol{A} \in \mathbb{R}^{n \times k} : \boldsymbol{A}^T \boldsymbol{A} = \boldsymbol{I}_k} \text{trace} \left(\boldsymbol{A}^T \boldsymbol{H} \boldsymbol{A} \right).$$

And the optimal $\boldsymbol{A}^*$ is given by $\boldsymbol{A}^* = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k]\boldsymbol{Q}$ with $\boldsymbol{Q}$ an arbitrary orthogonal matrix. ■

Solutions to PCA

1. Note that in Eqn. (2), the covariance matrix $\boldsymbol{C}$ is a symmetric matrix. Given the above theorem, we directly obtain

$$\begin{aligned} \lambda_1 + \cdots \lambda_k &= \underset{\boldsymbol{A} \in \mathbb{R}^{n \times k} : \boldsymbol{A}^T \boldsymbol{A} = \boldsymbol{I}_k}{\arg\max} \operatorname{trace}\left(\boldsymbol{A}^T \boldsymbol{C} \boldsymbol{A}\right), \\ \boldsymbol{A}^* &= [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k]\boldsymbol{Q}, \end{aligned}$$

   where $\lambda_1, \ldots, \lambda_k$ are the k largest eigenvalues of the covariance matrix $\boldsymbol{C}$, and the solution $\boldsymbol{A}^*$ is the matrix whose columns are corresponding eigenvectors.

2. It also follows from the above theorem that solutions to PCA are not unique, and they differ by an orthogonal matrix. We used the special case where $\boldsymbol{Q} = \boldsymbol{I}$, *i.e.*, $\boldsymbol{A}^* = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_k]$.

How to compute PCA efficiently?

1. We define

$$\begin{aligned} \boldsymbol{X} &= [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{p \times n}, \\ \bar{\boldsymbol{X}} &= \frac{1}{n} \boldsymbol{X} 1_n \in \mathbb{R}^{p \times 1}, \\ \tilde{\boldsymbol{X}} &= (\boldsymbol{X} - \bar{\boldsymbol{X}} 1_n^T) \in \mathbb{R}^{p \times n}, \end{aligned}$$

   where 1_n is the n-dimensional all-one vector. Here, $\tilde{\boldsymbol{X}}$ is the centered $\boldsymbol{X}$, which is obtained by subtracting the mean from each column.

2. Then we have

$$\boldsymbol{C} = \frac{1}{n} \sum_{i=1}^{n} (\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T = \frac{1}{n} \tilde{\boldsymbol{X}} \tilde{\boldsymbol{X}}^T \in \mathbb{R}^{p \times p}.$$

   If p is large, it is very costly to compute eigenvectors of $\boldsymbol{C}$ directly. Moreover, if $p >> n$, there are known computational problems.

How to compute PCA efficiently?

1. However, we show that the SVD of $\tilde{\boldsymbol{X}}$ provides what we need. If the SVD of $\tilde{\boldsymbol{X}}$ is $\tilde{\boldsymbol{X}} = \boldsymbol{U}\tilde{\boldsymbol{\Sigma}}\boldsymbol{V}^T$, the columns of $\boldsymbol{U}$ (left-singular vectors) are orthonormal eigenvectors of $\tilde{\boldsymbol{X}}\tilde{\boldsymbol{X}}^T$. Note that we have $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$ in Eqn. (1) and thus, the first k columns of $\boldsymbol{U}$ correspond to the largest k eigenvalues of $\tilde{\boldsymbol{X}}\tilde{\boldsymbol{X}}^T$.
2. With the SVD of $\tilde{\boldsymbol{X}}$, if we want to project features to a k-dimensional ($k < p$) space, simply take the first k columns of $\boldsymbol{U}$ as the projection matrix. Generally, the process of computing PCA can be described as

$$\underbrace{\boldsymbol{X}}_{p\times n} \rightarrow \underbrace{\tilde{\boldsymbol{X}}}_{p\times n} \rightarrow \tilde{\boldsymbol{X}} = \boldsymbol{U}\tilde{\boldsymbol{\Sigma}}\boldsymbol{V}^T \rightarrow \boldsymbol{G} = \boldsymbol{U}_k \in \mathbb{R}^{p\times k}.$$

Then the projected features are

$$\boldsymbol{G}^T\tilde{\boldsymbol{X}} = \boldsymbol{Z} \in \mathbb{R}^{k\times n}. \tag{3}$$

It is worth noting that the k-th row in $\boldsymbol{Z}$ is called the k-th principal components (PC). Therefore, PCA achieves feature reduction by keeping only the first k PCs.

Centered or not?

1. Note that we project the centered data matrix $\tilde{\boldsymbol{X}}$ instead of the original data matrix $\boldsymbol{X}$ in Eqn (3). Maximal variance can be achieved in both cases, as the the covariance matrix $\boldsymbol{C}$ will not change.
2. But the minimal reconstruction error can only be achieved when $\tilde{\boldsymbol{X}}$ is used, as shown in the next section.
3. A common practice to avoid any confusion is to center the data before applying PCA, and use the centered data matrix in all computations.

Have We Achieved the Minimal Reconstruction Error?

1. First, we perform the verification in the case where we project and reconstruct the centered data matrix $\tilde{\boldsymbol{X}}$.
2. Given the projection
$$\boldsymbol{G}^T\tilde{\boldsymbol{X}} = \boldsymbol{Z},$$
the reconstruction process is
$$\check{\boldsymbol{X}} = \boldsymbol{G}\boldsymbol{Z} = \boldsymbol{G}\boldsymbol{G}^T\tilde{\boldsymbol{X}},$$
and the reconstruction error is
$$||\tilde{\boldsymbol{X}} - \check{\boldsymbol{X}}||_F = ||\tilde{\boldsymbol{X}} - \boldsymbol{G}\boldsymbol{G}^T\tilde{\boldsymbol{X}}||_F.$$
3. We will show that
$$\boldsymbol{G}\boldsymbol{G}^T\tilde{\boldsymbol{X}} = \boldsymbol{B}^* = \underset{\boldsymbol{B}:\text{rank}(\boldsymbol{B})\leq k}{\arg\min} ||\tilde{\boldsymbol{X}} - \boldsymbol{B}||_F, \tag{4}$$
which means $\check{\boldsymbol{X}}$ gives the minimum reconstruction error.
4. We've already learned that $\boldsymbol{B}^*$ is the truncated SVD of $\tilde{\boldsymbol{X}}$. So, we only need to show that $\boldsymbol{G}\boldsymbol{G}^T\tilde{\boldsymbol{X}}$ is indeed the truncated SVD of $\tilde{\boldsymbol{X}}$ as follows. Note that similar arguments can be made for the spectral norm.

More details

Given $\tilde{\boldsymbol{X}} = \boldsymbol{U}\tilde{\boldsymbol{\Sigma}}\boldsymbol{V}^T$ and $\boldsymbol{G} = \boldsymbol{U}_k$, we have

$$\begin{aligned}
\boldsymbol{G}\boldsymbol{G}^T\tilde{\boldsymbol{X}} &= \boldsymbol{U}_k\boldsymbol{U}_k^T\boldsymbol{U}\tilde{\boldsymbol{\Sigma}}\boldsymbol{V}^T \\
&= \boldsymbol{U}_k\boldsymbol{U}_k^T\begin{bmatrix}\boldsymbol{U}_k & \boldsymbol{U}_{-k}\end{bmatrix}\tilde{\boldsymbol{\Sigma}}\boldsymbol{V}^T \\
&= \boldsymbol{U}_k\begin{bmatrix}\boldsymbol{I} & 0\end{bmatrix}\tilde{\boldsymbol{\Sigma}}\boldsymbol{V}^T \\
&= \begin{bmatrix}\boldsymbol{U}_k & 0\end{bmatrix}\tilde{\boldsymbol{\Sigma}}\boldsymbol{V}^T \\
&= \boldsymbol{U}_k\boldsymbol{\Sigma}_k\boldsymbol{V}_k^T \\
&= \sum_{i=1}^{k}\sigma_i\boldsymbol{u}_i\boldsymbol{v}_i^T.
\end{aligned}$$

As a result, $\check{\boldsymbol{X}}$ gives the minimum reconstruction error, which is

$$\min_{\boldsymbol{B}:\text{rank}(\boldsymbol{B})\leq k}||\tilde{\boldsymbol{X}} - \boldsymbol{B}||_F = ||\tilde{\boldsymbol{X}} - \boldsymbol{G}\boldsymbol{G}^T\tilde{\boldsymbol{X}}||_F = \sqrt{\sum_{i=k+1}^{n}\sigma_i^2}.$$

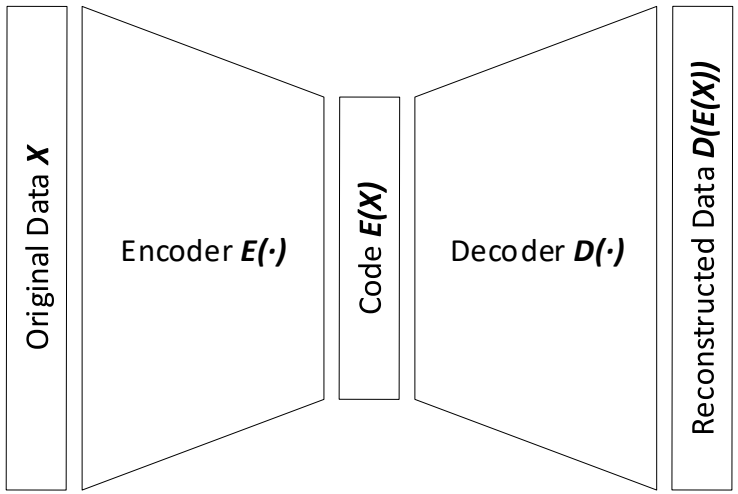
The case of spectral norm

1. Similarly, it can be shown that

$$\min_{\boldsymbol{B}:\text{rank}(\boldsymbol{B})\leq k} ||\tilde{\boldsymbol{X}} - \boldsymbol{B}||_2 = ||\tilde{\boldsymbol{X}} - \boldsymbol{G}\boldsymbol{G}^T\tilde{\boldsymbol{X}}||_2 = \sigma_{k+1}.$$

2. Therefore, PCA projects the high-dimensional features to a lower-dimensional space with minimum reconstruction error in terms of both Frobenius and spectral norms.

Connections with Autoencoders



1. We assume that $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{p \times n}$ is already centered, for the simplicity of notations.
2. The outputs of this framework have the same dimension as inputs, and are supposed to reconstruct the inputs exactly. Without any constraint, the reconstruction task can be easily solved by directly copying. However, autoencoders come with a crucial restriction that the dimension of intermediate outputs should be smaller than inputs.

Autoencoders

1. Concretely, the framework of autoencoders can be divided into two parts: the *encoder* $E(\cdot)$ and the *decoder* $D(\cdot)$. Given $\boldsymbol{X} \in \mathbb{R}^{p \times n}$ as inputs, the encoder is supposed to output a reduced representation of the inputs, *i.e.*, $E(\boldsymbol{X}) \in \mathbb{R}^{k \times n}$, where $k < p$. And the decoder tries to reconstruct the inputs from this reduced representation.
2. It is straightforward to see that the objective function of autoencoders is to minimize the reconstruction error:

$$L(\boldsymbol{X}, D(E(\boldsymbol{X}))),$$

where $L(\cdot, \cdot)$ is a loss function that measures the reconstruction error.

PCA as Autoencoders

1. We've shown that PCA is able to achieve the minimum reconstruction error. Both PCA and autoencoders are unsupervised learning models with the goal of minimizing the reconstruction error. Are they connected?
2. Consider a special autoencoder with the encoder and decoder defined as

$$\begin{aligned} E(\boldsymbol{X}) &= \boldsymbol{W}^T\boldsymbol{X} \in \mathbb{R}^{k\times n}, \\ D(E(\boldsymbol{X})) &= \boldsymbol{W}E(\boldsymbol{X}) = \boldsymbol{W}\boldsymbol{W}^T\boldsymbol{X} \in \mathbb{R}^{p\times n}, \end{aligned}$$

   where $\boldsymbol{W} \in \mathbb{R}^{p\times k}$. Note that

$$\text{rank}(\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{X}) \leq \min(\text{rank}(\boldsymbol{W}), \text{rank}(\boldsymbol{W}^T\boldsymbol{X})) \leq k$$

3. If the loss function $L(\cdot,\cdot)$ is the mean square error, the objective of this autoencoder is equivalent to

$$\min_{\boldsymbol{W}:\text{rank}(\boldsymbol{W}\boldsymbol{W}^T\boldsymbol{X})\leq k} ||\boldsymbol{X} - \boldsymbol{W}\boldsymbol{W}^T\boldsymbol{X}||_F. \tag{5}$$

PCA as Autoencoders

1. In an autoencoder, $\boldsymbol{W}$ can be randomly initialized and optimized through gradient decent. However, comparing Eqn. (4) and Eqn. (5), it is easy to see that the optima can be achieved when $\boldsymbol{W} = \boldsymbol{G}$, where $\boldsymbol{G}$ is given by PCA.
2. Therefore, a natural question is: Will gradient decent give the same results as PCA? The answer is two-fold: *yes* in terms of the minimal reconstruction error and *not necessary* in terms of optimal $\boldsymbol{W}$. Gradient decent converges to the global optima, resulting in the same minimal reconstruction error as PCA. However, the solutions to PCA are not unique and can differ by an orthogonal matrix. Thus, the optimal $\boldsymbol{W}$ can be different from a PCA solution $\boldsymbol{G}$. But one can be computed from another through an orthogonal matrix and the sets of possible solutions are the same.
3. To summarize, PCA is a special case of autoencoders, where the encoder and decoder are both one-layer linear transformations, which share the weights and do not have bias terms.

THANKS!