Principal Component Analysis

Statistical Machine Learning (ENGG*6600*02)

School of Engineering,
University of Guelph, ON, Canada

Course Instructor: Benyamin Ghojogh

Summer 2023

Projection and Reconstruction in PCA

Projection and Reconstruction in PCA

- If there exist n training data points, i.e., $\{\boldsymbol{x}_i\}_{i=1}^n$, the projection of a training data point $\boldsymbol{x}$ is:

$$\mathbb{R}^p \ni \widetilde{\boldsymbol{x}} := \boldsymbol{U}^\top \breve{\boldsymbol{x}}, \tag{1}$$

  where:

$$\mathbb{R}^d \ni \breve{\boldsymbol{x}} := \boldsymbol{x} - \boldsymbol{\mu}_x, \tag{2}$$

  is the centered data point and:

$$\mathbb{R}^d \ni \boldsymbol{\mu}_x := \frac{1}{n} \sum_{i=1}^n \boldsymbol{x}_i, \tag{3}$$

  is the mean of training data points.

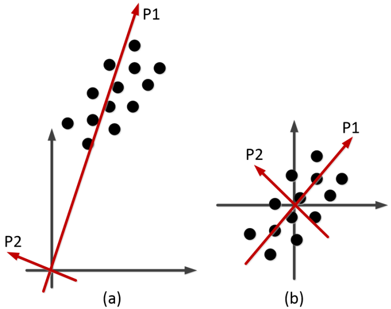
- The reconstruction of a training data point $\boldsymbol{x}$ after projection onto the PCA subspace is:

$$\mathbb{R}^d \ni \widehat{\boldsymbol{x}} := \boldsymbol{U}\boldsymbol{U}^\top \breve{\boldsymbol{x}} + \boldsymbol{\mu}_x = \boldsymbol{U}\widetilde{\boldsymbol{x}} + \boldsymbol{\mu}_x, \tag{4}$$

  where the mean is added back because it was removed before projection.

Projection and Reconstruction in PCA

- In PCA, all the data points should be centered, i.e., the mean should be removed first. The reason is shown in this figure.



- In some applications, centering the data does not make sense. For example, in natural language processing, the data are text and centering the data makes some negative measures which is non-sense for text. Therefore, data is not sometimes centered and PCA is applied on the non-centered data. This method is called Latent Semantic Indexing (LSI) or **Latent Semantic Analysis (LSA)** [1].

Projection and Reconstruction in PCA

- If we stack the n data points column-wise in a matrix $\boldsymbol{X} = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_n] \in \mathbb{R}^{d \times n}$, we first center them:

$$\mathbb{R}^{d \times n} \ni \breve{\boldsymbol{X}} := \boldsymbol{X}\boldsymbol{H} = \boldsymbol{X} - \boldsymbol{\mu}_x, \tag{5}$$

  where $\breve{\boldsymbol{X}} = [\breve{\boldsymbol{x}}_1, \dots, \breve{\boldsymbol{x}}_n] = [\boldsymbol{x}_1 - \boldsymbol{\mu}_x, \dots, \boldsymbol{x}_n - \boldsymbol{\mu}_x]$ is the centered data and:

$$\mathbb{R}^{n \times n} \ni \boldsymbol{H} := \boldsymbol{I} - (1/n)\mathbf{1}\mathbf{1}^\top, \tag{6}$$

  is the centering matrix.

- The projection and reconstruction, Eqs. (1) and (4), for the whole training data are:

$$\mathbb{R}^{p \times n} \ni \widetilde{\boldsymbol{X}} := \boldsymbol{U}^\top \breve{\boldsymbol{X}}, \tag{7}$$

$$\mathbb{R}^{d \times n} \ni \widehat{\boldsymbol{X}} := \boldsymbol{U}\boldsymbol{U}^\top \breve{\boldsymbol{X}} + \boldsymbol{\mu}_x = \boldsymbol{U}\widetilde{\boldsymbol{X}} + \boldsymbol{\mu}_x, \tag{8}$$

  where $\widetilde{\boldsymbol{X}} = [\widetilde{\boldsymbol{x}}_1, \dots, \widetilde{\boldsymbol{x}}_n]$ and $\widehat{\boldsymbol{X}} = [\widehat{\boldsymbol{x}}_1, \dots, \widehat{\boldsymbol{x}}_n]$ are the projected data onto PCA subspace and the reconstructed data, respectively.

Projection and Reconstruction in PCA

- We can also project a new data point onto the PCA subspace for $\boldsymbol{X}$ where the new data point is not a column of $\boldsymbol{X}$. In other words, the new data point has not had impact in constructing the PCA subspace. This new data point is also referred to as "test data point" or "out-of-sample data" in the literature. The Eq. (7) was for projection of $\boldsymbol{X}$ onto its PCA subspace. If $\boldsymbol{x}_t$ denotes an out-of-sample data point, its projection onto the PCA subspace ($\widetilde{\boldsymbol{x}}_t$) and its reconstruction ($\widehat{\boldsymbol{x}}_t$) are:

$$\mathbb{R}^p \ni \widetilde{\boldsymbol{x}}_t = \boldsymbol{U}^\top \breve{\boldsymbol{x}}_t, \tag{9}$$

$$\mathbb{R}^d \ni \widehat{\boldsymbol{x}}_t = \boldsymbol{U}\boldsymbol{U}^\top \breve{\boldsymbol{x}}_t + \boldsymbol{\mu}_x = \boldsymbol{U}\widetilde{\boldsymbol{x}}_t + \boldsymbol{\mu}_x, \tag{10}$$

  where:

$$\mathbb{R}^d \ni \breve{\boldsymbol{x}}_t := \boldsymbol{x}_t - \boldsymbol{\mu}_x, \tag{11}$$

  is the centered out-of-sample data point which is centered using the mean of training data. Note that for centering the out-of-sample data point(s), we should use the mean of the training data and not the out-of-sample data.
- If we consider the n_t out-of-sample data points, $\mathbb{R}^{d\times n_t} \ni \boldsymbol{X}_t = [\boldsymbol{x}_{t,1}, \dots, \boldsymbol{x}_{t,n_t}]$, all together, the projection and reconstruction of them are:

$$\mathbb{R}^{p\times n_t} \ni \widetilde{\boldsymbol{X}}_t = \boldsymbol{U}^\top \breve{\boldsymbol{X}}_t, \tag{12}$$

$$\mathbb{R}^{d\times n_t} \ni \widehat{\boldsymbol{X}}_t = \boldsymbol{U}\boldsymbol{U}^\top \breve{\boldsymbol{X}}_t + \boldsymbol{\mu}_x = \boldsymbol{U}\widetilde{\boldsymbol{X}}_t + \boldsymbol{\mu}_x, \tag{13}$$

  respectively, where:

$$\mathbb{R}^{d\times n_t} \ni \breve{\boldsymbol{X}}_t := \boldsymbol{X}_t - \boldsymbol{\mu}_x. \tag{14}$$

PCA Using Eigen-Decomposition

Projection Onto One Direction

- In Eq. (4), if $p = 1$, we are projecting $\boldsymbol{x}$ onto only one vector $\boldsymbol{u}$ and reconstruct it. If we ignore adding the mean back, we have:

$$\widehat{\boldsymbol{x}} = \boldsymbol{u}\boldsymbol{u}^\top \breve{\boldsymbol{x}}.$$

- The squared length (squared ℓ_2-norm) of this reconstructed vector is:

$$\begin{aligned} ||\widehat{\boldsymbol{x}}||_2^2 &= ||\boldsymbol{u}\boldsymbol{u}^\top \breve{\boldsymbol{x}}||_2^2 = (\boldsymbol{u}\boldsymbol{u}^\top \breve{\boldsymbol{x}})^\top (\boldsymbol{u}\boldsymbol{u}^\top \breve{\boldsymbol{x}}) \\ &= \breve{\boldsymbol{x}}^\top \boldsymbol{u} \underbrace{\boldsymbol{u}^\top \boldsymbol{u}}_{1} \boldsymbol{u}^\top \breve{\boldsymbol{x}} \overset{(a)}{=} \breve{\boldsymbol{x}}^\top \boldsymbol{u}\, \boldsymbol{u}^\top \breve{\boldsymbol{x}} \overset{(b)}{=} \boldsymbol{u}^\top \breve{\boldsymbol{x}}\, \breve{\boldsymbol{x}}^\top \boldsymbol{u}, \end{aligned} \tag{15}$$

where (a) is because $\boldsymbol{u}$ is a unit (normal) vector, i.e., $\boldsymbol{u}^\top \boldsymbol{u} = ||\boldsymbol{u}||_2^2 = 1$, and (b) is because $\breve{\boldsymbol{x}}^\top \boldsymbol{u} = \boldsymbol{u}^\top \breve{\boldsymbol{x}} \in \mathbb{R}$.

- Suppose we have n data points $\{\boldsymbol{x}_i\}_{i=1}^n$ where $\{\breve{\boldsymbol{x}}_i\}_{i=1}^n$ are the centered data. The summation of the squared lengths of their projections $\{\widehat{\boldsymbol{x}}_i\}_{i=1}^n$ is:

$$\sum_{i=1}^n ||\widehat{\boldsymbol{x}}_i||_2^2 \overset{(15)}{=} \sum_{i=1}^n \boldsymbol{u}^\top \breve{\boldsymbol{x}}_i\, \breve{\boldsymbol{x}}_i^\top \boldsymbol{u} = \boldsymbol{u}^\top \Big(\sum_{i=1}^n \breve{\boldsymbol{x}}_i\, \breve{\boldsymbol{x}}_i^\top \Big) \boldsymbol{u}. \tag{16}$$

Projection Onto One Direction

- Considering $\breve{\boldsymbol{X}} = [\breve{\boldsymbol{x}}_1, \dots, \breve{\boldsymbol{x}}_n] \in \mathbb{R}^{d \times n}$, we have:

$$\mathbb{R}^{d \times d} \ni \boldsymbol{S} := \sum_{i=1}^{n} \breve{\boldsymbol{x}}_i \, \breve{\boldsymbol{x}}_i^\top = \breve{\boldsymbol{X}} \breve{\boldsymbol{X}}^\top \overset{(5)}{=} \boldsymbol{X}\boldsymbol{H}\boldsymbol{H}^\top \boldsymbol{X}^\top = \boldsymbol{X}\boldsymbol{H}\boldsymbol{H}\boldsymbol{X}^\top = \boldsymbol{X}\boldsymbol{H}\boldsymbol{X}^\top, \qquad (17)$$

  where $\boldsymbol{S}$ is called the "covariance matrix". If the data were already centered, we would have $\boldsymbol{S} = \boldsymbol{X}\boldsymbol{X}^\top$.
- Plugging Eq. (17) in Eq. (16) gives us:

$$\sum_{i=1}^{n} ||\widehat{\boldsymbol{x}}_i||_2^2 = \boldsymbol{u}^\top \boldsymbol{S} \boldsymbol{u}. \qquad (18)$$

- Note that we can also say that $\boldsymbol{u}^\top \boldsymbol{S} \boldsymbol{u}$ is the variance of the projected data onto PCA subspace. In other words, $\boldsymbol{u}^\top \boldsymbol{S} \boldsymbol{u} = \mathbb{V}\text{ar}(\boldsymbol{u}^\top \breve{\boldsymbol{X}})$. This makes sense because when some non-random thing (here $\boldsymbol{u}$) is multiplied to the random data (here $\breve{\boldsymbol{X}}$), it will have squared (quadratic) effect on variance, and $\boldsymbol{u}^\top \boldsymbol{S} \boldsymbol{u}$ is quadratic in $\boldsymbol{u}$.
- Therefore, $\boldsymbol{u}^\top \boldsymbol{S} \boldsymbol{u}$ can be interpreted in two ways: (I) the squared length of reconstruction and (II) the variance of projection.

Projection Onto One Direction

- We want to find a projection direction $\boldsymbol{u}$ which maximizes the squared length of reconstruction (or variance of projection):

$$\begin{aligned} \underset{\boldsymbol{u}}{\text{maximize}} \quad & \boldsymbol{u}^\top \boldsymbol{S} \boldsymbol{u}, \\ \text{subject to} \quad & \boldsymbol{u}^\top \boldsymbol{u} = 1, \end{aligned} \tag{19}$$

  where the constraint ensures that the $\boldsymbol{u}$ is a unit (normal) vector as we assumed beforehand.
- Using Lagrange multiplier [2], we have:

$$\mathcal{L} = \boldsymbol{u}^\top \boldsymbol{S} \boldsymbol{u} - \lambda(\boldsymbol{u}^\top \boldsymbol{u} - 1),$$

  Taking derivative of the Lagrangian and setting it to zero gives:

$$\mathbb{R}^p \ni \frac{\partial \mathcal{L}}{\partial \boldsymbol{u}} = 2\boldsymbol{S}\boldsymbol{u} - 2\lambda \boldsymbol{u} \overset{\text{set}}{=} 0 \implies \boldsymbol{S}\boldsymbol{u} = \lambda \boldsymbol{u}. \tag{20}$$

- The Eq. (20) is the eigen-decomposition of $\boldsymbol{S}$ where $\boldsymbol{u}$ and λ are the leading eigenvector and eigenvalue of $\boldsymbol{S}$, respectively [3].
- Note that the leading eigenvalue is the largest one. The reason of being leading is that we are maximizing in the optimization problem.
- As a conclusion, if projecting onto one PCA direction, the PCA direction $\boldsymbol{u}$ is the leading eigenvector of the covariance matrix.
- Note that the "PCA direction" is also called "principal direction" or "principal axis" in the literature. The dimensions (features) of the projected data onto PCA subspace are called "principal components".

Projection Onto Span of Several Directions

- In Eq. (4) or (8), if $p > 1$, we are projecting $\breve{\boldsymbol{x}}$ or $\breve{\boldsymbol{X}}$ onto PCA subspace with dimensionality more than one and then reconstruct back. If we ignore adding the mean back, we have:

$$\widehat{\boldsymbol{X}} = \boldsymbol{U}\boldsymbol{U}^\top \breve{\boldsymbol{X}}.$$

- It means that we project every column of $\breve{\boldsymbol{X}}$, i.e., $\breve{\boldsymbol{x}}$, onto a space spanned by the p vectors $\{\boldsymbol{u}_1, \dots, \boldsymbol{u}_p\}$ each of which is d-dimensional. Therefore, the projected data are p-dimensional and the reconstructed data are d-dimensional.
- The squared length (squared Frobenius Norm) of this reconstructed matrix is:

$$\begin{aligned} ||\widehat{\boldsymbol{X}}||_F^2 &= ||\boldsymbol{U}\boldsymbol{U}^\top \breve{\boldsymbol{X}}||_F^2 = \mathbf{tr}\big((\boldsymbol{U}\boldsymbol{U}^\top \breve{\boldsymbol{X}})^\top (\boldsymbol{U}\boldsymbol{U}^\top \breve{\boldsymbol{X}})\big) \\ &= \mathbf{tr}(\breve{\boldsymbol{X}}^\top \boldsymbol{U}\, \underbrace{\boldsymbol{U}^\top \boldsymbol{U}}_{\boldsymbol{I}}\, \boldsymbol{U}^\top \breve{\boldsymbol{X}}) \overset{(a)}{=} \mathbf{tr}(\breve{\boldsymbol{X}}^\top \boldsymbol{U}\boldsymbol{U}^\top \breve{\boldsymbol{X}}) \overset{(b)}{=} \mathbf{tr}(\boldsymbol{U}^\top \breve{\boldsymbol{X}}\breve{\boldsymbol{X}}^\top \boldsymbol{U}), \end{aligned}$$

where $\mathbf{tr}(.)$ denotes the trace of matrix, (a) is because $\boldsymbol{U}$ is an orthogonal matrix (its columns are orthonormal), and (b) is because $\mathbf{tr}(\breve{\boldsymbol{X}}^\top \boldsymbol{U}\boldsymbol{U}^\top \breve{\boldsymbol{X}}) = \mathbf{tr}(\breve{\boldsymbol{X}}\breve{\boldsymbol{X}}^\top \boldsymbol{U}\boldsymbol{U}^\top) = \mathbf{tr}(\boldsymbol{U}^\top \breve{\boldsymbol{X}}\breve{\boldsymbol{X}}^\top \boldsymbol{U})$. According to Eq. (17), the $\boldsymbol{S} = \breve{\boldsymbol{X}}\breve{\boldsymbol{X}}^\top$ is the covariance matrix; therefore:

$$||\widehat{\boldsymbol{X}}||_F^2 = \mathbf{tr}(\boldsymbol{U}^\top \boldsymbol{S}\, \boldsymbol{U}). \tag{21}$$

Projection Onto Span of Several Directions

- We want to find several projection directions $\{\boldsymbol{u}_1, \dots, \boldsymbol{u}_p\}$, as columns of $\boldsymbol{U} \in \mathbb{R}^{d \times p}$, which maximize the squared length of reconstruction (or variance of projection):

$$\begin{aligned} \underset{\boldsymbol{U}}{\text{maximize}} \quad & \mathbf{tr}(\boldsymbol{U}^\top \boldsymbol{S}\, \boldsymbol{U}), \\ \text{subject to} \quad & \boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}, \end{aligned} \tag{22}$$

  where the constraint ensures that the $\boldsymbol{U}$ is an orthogonal matrix as we assumed beforehand.

- Using Lagrange multiplier [2], we have:

$$\mathcal{L} = \mathbf{tr}(\boldsymbol{U}^\top \boldsymbol{S}\, \boldsymbol{U}) - \mathbf{tr}\big(\boldsymbol{\Lambda}^\top (\boldsymbol{U}^\top \boldsymbol{U} - \boldsymbol{I})\big),$$

  where $\boldsymbol{\Lambda} \in \mathbb{R}^{p \times p}$ is a diagonal matrix $\mathbf{diag}([\lambda_1, \dots, \lambda_p]^\top)$ including the Lagrange multipliers.

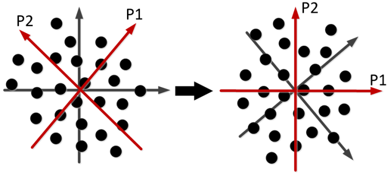
$$\mathbb{R}^{d \times p} \ni \frac{\partial \mathcal{L}}{\partial \boldsymbol{U}} = 2\boldsymbol{S}\boldsymbol{U} - 2\boldsymbol{U}\boldsymbol{\Lambda} \overset{\text{set}}{=} 0 \implies \boldsymbol{S}\boldsymbol{U} = \boldsymbol{U}\boldsymbol{\Lambda}. \tag{23}$$

- The Eq. (23) is the eigen-decomposition of $\boldsymbol{S}$ where the columns of $\boldsymbol{U}$ and the diagonal of $\boldsymbol{\Lambda}$ are the eigenvectors and eigenvalues of $\boldsymbol{S}$, respectively [3]. The eigenvectors and eigenvalues are sorted from the leading (largest eigenvalue) to the trailing (smallest eigenvalue) because we are maximizing in the optimization problem. As a conclusion, if projecting onto the PCA subspace or $\mathbf{span}\{\boldsymbol{u}_1, \dots, \boldsymbol{u}_p\}$, the PCA directions $\{\boldsymbol{u}_1, \dots, \boldsymbol{u}_p\}$ are the sorted eigenvectors of the covariance matrix of data $\boldsymbol{X}$.

Truncating U

Truncating $\boldsymbol{U}$

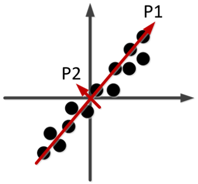
- If $\mathbf{rank}(\boldsymbol{S}) = d$: we have $p = d$ (we have d non-zero eigenvalues of $\boldsymbol{S}$), so that $\boldsymbol{U} \in \mathbb{R}^{d\times d}$. It means that the dimensionality of the PCA subspace is d, equal to the dimensionality of the original space. Why does this happen? That is because $\mathbf{rank}(\boldsymbol{S}) = d$ means that the data are spread wide enough in all dimensions of the original space up to a possible rotation. Therefore, the dimensionality of PCA subspace is equal to the original dimensionality; however, PCA might merely rotate the coordinate axes. In this case, $\boldsymbol{U} \in \mathbb{R}^{d\times d}$ is a square orthogonal matrix so that $\mathbb{R}^{d\times d} \ni \boldsymbol{U}\boldsymbol{U}^\top = \boldsymbol{U}\boldsymbol{U}^{-1} = \boldsymbol{I}$ and $\mathbb{R}^{d\times d} \ni \boldsymbol{U}^\top\boldsymbol{U} = \boldsymbol{U}^{-1}\boldsymbol{U} = \boldsymbol{I}$ because $\mathbf{rank}(\boldsymbol{U}) = d$, $\mathbf{rank}(\boldsymbol{U}\boldsymbol{U}^\top) = d$, and $\mathbf{rank}(\boldsymbol{U}^\top\boldsymbol{U}) = d$. That is why in the literature, PCA is also referred to as coordinate rotation.



- If $\mathbf{rank}(\boldsymbol{S}) < d$ and $n > d$: it means that we have enough data points but the data points exist on a subspace and do not fill the original space wide enough in every direction. In this case, $\boldsymbol{U} \in \mathbb{R}^{d\times p}$ is not square and $\mathbf{rank}(\boldsymbol{U}) = p < d$ (we have p non-zero eigenvalues of $\boldsymbol{S}$). Therefore, $\mathbb{R}^{d\times d} \ni \boldsymbol{U}\boldsymbol{U}^\top \neq \boldsymbol{I}$ and $\mathbb{R}^{p\times p} \ni \boldsymbol{U}^\top\boldsymbol{U} = \boldsymbol{I}$ because $\mathbf{rank}(\boldsymbol{U}) = p$, $\mathbf{rank}(\boldsymbol{U}\boldsymbol{U}^\top) = p < d$, and $\mathbf{rank}(\boldsymbol{U}^\top\boldsymbol{U}) = p$.

Truncating $\boldsymbol{U}$

- If $\textbf{rank}(\boldsymbol{S}) \leq n-1 < d$: it means that we do not have enough data points to properly represent the original space and the points have an "intrinsic dimensionality". For example, we have two three-dimensional points which are one a two-dimensional line (subspace). So, similar to previous case, the data points exist on a subspace and do not fill the original space wide enough in every direction. The discussions about $\boldsymbol{U}$, $\boldsymbol{U}\boldsymbol{U}^\top$, and $\boldsymbol{U}^\top\boldsymbol{U}$ are similar to previous case.
- Note that we might have $\textbf{rank}(\boldsymbol{S}) = d$ and thus $\boldsymbol{U} \in \mathbb{R}^{d\times d}$ but want to "truncate" the matrix $\boldsymbol{U}$ to have $\boldsymbol{U} \in \mathbb{R}^{d\times p}$. Truncating $\boldsymbol{U}$ means that we take a subset of best (leading) eigenvectors rather than the whole d eigenvectors with non-zero eigenvalues. In this case, again we have $\boldsymbol{U}\boldsymbol{U}^\top \neq \boldsymbol{I}$ and $\boldsymbol{U}^\top\boldsymbol{U} = \boldsymbol{I}$. The intuition of truncating is this: the variance of data might be noticeably smaller than another direction; in this case, we can only keep the $p < d$ top eigenvectors (PCA directions) and "ignore" the PCA directions with smaller eigenvalues to have $\boldsymbol{U} \in \mathbb{R}^{d\times p}$.



- From all the above analyses, we conclude that as long as the columns of the matrix $\boldsymbol{U} \in \mathbb{R}^{d\times p}$ are orthonormal, we always have $\boldsymbol{U}^\top\boldsymbol{U} = \boldsymbol{I}$ regardless of the value p. If the orthogonal matrix $\boldsymbol{U}$ is not truncated and thus is a square matrix, we also have $\boldsymbol{U}\boldsymbol{U}^\top = \boldsymbol{I}$.

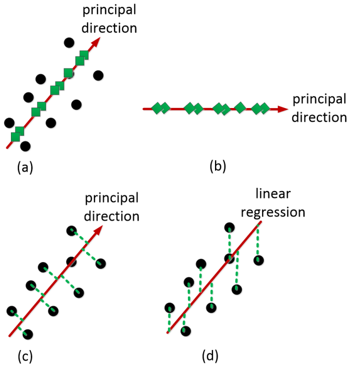
Reconstruction Error in PCA

Reconstruction Error in PCA

- If we center the data, the residual (reconstruction error) becomes $\boldsymbol{r} = \breve{\boldsymbol{x}} - \widehat{\boldsymbol{x}}$ because the reconstructed data will also be centered according to Eq. (4). According to Eqs. (2), and (4), we have:

$$\boldsymbol{r} = \boldsymbol{x} - \widehat{\boldsymbol{x}} = \breve{\boldsymbol{x}} + \boldsymbol{\mu}_x - \boldsymbol{U}\boldsymbol{U}^\top \breve{\boldsymbol{x}} - \boldsymbol{\mu}_x = \breve{\boldsymbol{x}} - \boldsymbol{U}\boldsymbol{U}^\top \breve{\boldsymbol{x}}. \tag{24}$$

- This figure shows the projection of a two-dimensional point (after the data being centered) onto the first principal direction, its reconstruction, and its reconstruction error. As can be seen in this figure, the reconstruction error is different from least square error in linear regression.



Reconstruction Error in PCA

- For n data points, we have:

$$\boldsymbol{R} := \boldsymbol{X} - \widehat{\boldsymbol{X}} = \breve{\boldsymbol{X}} + \boldsymbol{\mu}_x - \boldsymbol{U}\boldsymbol{U}^\top \breve{\boldsymbol{X}} - \boldsymbol{\mu}_x = \breve{\boldsymbol{X}} - \boldsymbol{U}\boldsymbol{U}^\top \breve{\boldsymbol{X}}, \tag{25}$$

where $\mathbb{R}^{d \times n} \ni \boldsymbol{R} = [\boldsymbol{r}_1, \dots, \boldsymbol{r}_n]$ is the matrix of residuals.

- If we want to minimize the reconstruction error subject to the orthogonality of the projection matrix $\boldsymbol{U}$, we have:

$$\begin{aligned} \underset{\boldsymbol{U}}{\text{minimize}} \quad & ||\breve{\boldsymbol{X}} - \boldsymbol{U}\boldsymbol{U}^\top \breve{\boldsymbol{X}}||_F^2, \\ \text{subject to} \quad & \boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}. \end{aligned} \tag{26}$$

- The objective function can be simplified:

$$\begin{aligned} & ||\breve{\boldsymbol{X}} - \boldsymbol{U}\boldsymbol{U}^\top \breve{\boldsymbol{X}}||_F^2 \\ & = \mathbf{tr}\big((\breve{\boldsymbol{X}} - \boldsymbol{U}\boldsymbol{U}^\top \breve{\boldsymbol{X}})^\top (\breve{\boldsymbol{X}} - \boldsymbol{U}\boldsymbol{U}^\top \breve{\boldsymbol{X}})\big) = \mathbf{tr}\big((\breve{\boldsymbol{X}}^\top - \breve{\boldsymbol{X}}^\top \boldsymbol{U}\boldsymbol{U}^\top)(\breve{\boldsymbol{X}} - \boldsymbol{U}\boldsymbol{U}^\top \breve{\boldsymbol{X}})\big) \\ & = \mathbf{tr}(\breve{\boldsymbol{X}}^\top \breve{\boldsymbol{X}} - 2\breve{\boldsymbol{X}}^\top \boldsymbol{U}\boldsymbol{U}^\top \breve{\boldsymbol{X}} + \breve{\boldsymbol{X}}^\top \boldsymbol{U} \underbrace{\boldsymbol{U}^\top \boldsymbol{U}}_{\boldsymbol{I}} \boldsymbol{U}^\top \breve{\boldsymbol{X}}) \\ & = \mathbf{tr}(\breve{\boldsymbol{X}}^\top \breve{\boldsymbol{X}} - \breve{\boldsymbol{X}}^\top \boldsymbol{U}\boldsymbol{U}^\top \breve{\boldsymbol{X}}) = \mathbf{tr}(\breve{\boldsymbol{X}}^\top \breve{\boldsymbol{X}}) - \mathbf{tr}(\breve{\boldsymbol{X}}^\top \boldsymbol{U}\boldsymbol{U}^\top \breve{\boldsymbol{X}}) = \mathbf{tr}(\breve{\boldsymbol{X}}^\top \breve{\boldsymbol{X}}) - \mathbf{tr}(\breve{\boldsymbol{X}}\breve{\boldsymbol{X}}^\top \boldsymbol{U}\boldsymbol{U}^\top). \end{aligned}$$

- Using Lagrange multiplier [2], we have:

$$\mathcal{L} = \mathbf{tr}(\breve{\boldsymbol{X}}^\top \breve{\boldsymbol{X}}) - \mathbf{tr}(\breve{\boldsymbol{X}}\breve{\boldsymbol{X}}^\top \boldsymbol{U}\boldsymbol{U}^\top) + \mathbf{tr}\big(\boldsymbol{\Lambda}^\top (\boldsymbol{U}^\top \boldsymbol{U} - \boldsymbol{I})\big),$$

where $\boldsymbol{\Lambda} \in \mathbb{R}^{p \times p}$ is a diagonal matrix $\mathbf{diag}([\lambda_1, \dots, \lambda_p]^\top)$ containing the Lagrange multipliers.

Reconstruction Error in PCA

- Equating the derivative of Lagrangian to zero gives:

$$\begin{aligned}\mathbb{R}^{d\times p} \ni \frac{\partial \mathcal{L}}{\partial \boldsymbol{U}} &= -2\breve{\boldsymbol{X}}\breve{\boldsymbol{X}}^\top \boldsymbol{U} + 2\boldsymbol{U}\boldsymbol{\Lambda} \overset{\text{set}}{=} 0 \\ \implies \breve{\boldsymbol{X}}\breve{\boldsymbol{X}}^\top \boldsymbol{U} &= \boldsymbol{U}\boldsymbol{\Lambda}, \\ \overset{(17)}{\implies} \boldsymbol{S}\boldsymbol{U} &= \boldsymbol{U}\boldsymbol{\Lambda}, \qquad (27)\end{aligned}$$

  which is again the eigenvalue problem [3] for the covariance matrix $\boldsymbol{S}$.
- We had the same eigenvalue problem in PCA. Therefore, **PCA subspace is the best linear projection in terms of reconstruction error. In other words, PCA has the least squared error in reconstruction.**

Reconstruction Error in PCA

- We saw that PCA is the best in reconstruction error for *linear* projection. If we have $m > 1$ successive linear projections, the reconstruction is:

$$\widehat{\boldsymbol{X}} = \underbrace{\boldsymbol{U}_1 \cdots \boldsymbol{U}_m}_{\text{reconstruct}} \underbrace{\boldsymbol{U}_m^\top \cdots \boldsymbol{U}_1^\top}_{\text{project}} \breve{\boldsymbol{X}} + \boldsymbol{\mu}_x, \tag{28}$$

  which can be seen as an undercomplete *autoencoder* [4] with $2m$ layers without activation function (or with identity activation functions $f(\boldsymbol{x}) = \boldsymbol{x}$). The $\boldsymbol{\mu}_x$ is modeled by the intercepts included as input to the neurons of autoencoder layers.

- As we do not have any non-linearity between the projections, we can define:

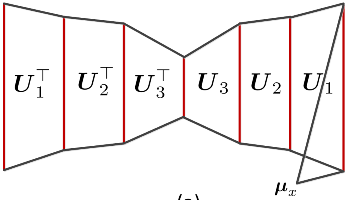
$$\ddot{\boldsymbol{U}} := \boldsymbol{U}_1 \cdots \boldsymbol{U}_m \implies \ddot{\boldsymbol{U}}^\top = \boldsymbol{U}_m^\top \cdots \boldsymbol{U}_1^\top, \tag{29}$$

$$\therefore \quad \widehat{\boldsymbol{X}} = \ddot{\boldsymbol{U}}\ddot{\boldsymbol{U}}^\top \boldsymbol{X} + \boldsymbol{\mu}_x. \tag{30}$$

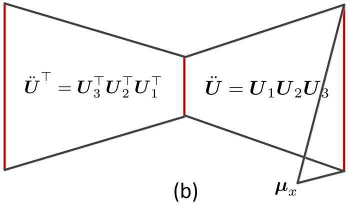
  The Eq. (30) shows that the whole autoencoder can be reduced to an undercomplete autoencoder with one hidden layer where the weight matrix is $\ddot{\boldsymbol{U}}$. In other words, in autoencoder neural network, every layer excluding the activation function behaves as a linear projection.

- Comparing the Eqs. (8) and (30) shows that the whole autoencoder is reduced to PCA. Therefore, **PCA is equivalent to an undercomplete autoencoder with one hidden layer without activation function**. Therefore, if we trained weights of such an autoencoder by back-propagation [5], they will be roughly equal to the PCA directions.

Reconstruction Error in PCA



(a)



(b)

PCA Using Singular Value Decomposition

PCA Using Singular Value Decomposition

- The PCA can be done using Singular Value Decomposition (SVD) of $\breve{\boldsymbol{X}}$, rather than eigen-decomposition of $\boldsymbol{S}$. Consider the complete SVD of $\breve{\boldsymbol{X}}$:

$$\mathbb{R}^{d \times n} \ni \breve{\boldsymbol{X}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top, \tag{31}$$

  where the columns of $\boldsymbol{U} \in \mathbb{R}^{d \times d}$ (called left singular vectors) are the eigenvectors of $\breve{\boldsymbol{X}}\breve{\boldsymbol{X}}^\top$, the columns of $\boldsymbol{V} \in \mathbb{R}^{n \times n}$ (called right singular vectors) are the eigenvectors of $\breve{\boldsymbol{X}}^\top\breve{\boldsymbol{X}}$, and the $\boldsymbol{\Sigma} \in \mathbb{R}^{d \times n}$ is a rectangular diagonal matrix whose diagonal entries (called singular values) are the square root of eigenvalues of $\breve{\boldsymbol{X}}\breve{\boldsymbol{X}}^\top$ and/or $\breve{\boldsymbol{X}}^\top\breve{\boldsymbol{X}}$. See Preliminaries slides for proof of this claim.
- According to Eq. (17), the $\breve{\boldsymbol{X}}\breve{\boldsymbol{X}}^\top$ is the covariance matrix $\boldsymbol{S}$. In Eq. (23), we saw that the eigenvectors of $\boldsymbol{S}$ are the principal directions. On the other hand, here, we saw that the columns of $\boldsymbol{U}$ are the eigenvectors of $\breve{\boldsymbol{X}}\breve{\boldsymbol{X}}^\top$. Hence, we can apply SVD on $\breve{\boldsymbol{X}}$ and take the left singular vectors (columns of $\boldsymbol{U}$) as the principal directions.
- An interesting thing is that in SVD of $\breve{\boldsymbol{X}}$, the columns of $\boldsymbol{U}$ are automatically sorted from largest to smallest singular values (eigenvalues) and we do not need to sort as we did in using eigenvalue decomposition for the covariance matrix.

Determining the Number of Principal Directions

Determining the Number of Principal Directions

- Usually in PCA, the components with smallest eigenvalues are cut off to reduce the data. There are different methods for estimating the best number of components to keep (denoted by p), such as using Bayesian model selection [6], scree plot [7], and comparing the ratio $\lambda_j / \sum_{k=1}^{d} \lambda_k$ with a threshold [8] where λ_i denotes the eigenvalue related to the j-th principal component.
- Here, we explain the two methods of **scree plot** and the **ratio**.
- The **scree plot** [7] is a plot of the eigenvalues versus sorted components from the leading (having largest eigenvalue) to trailing (having smallest eigenvalue). A threshold for the vertical (eigenvalue) axis chooses the components with the large enough eigenvalues and removes the rest of the components. A good threshold is where the eigenvalue drops significantly. In most of the datasets, a significant drop of eigenvalue occurs.
- Another way to choose the best components is the **ratio** [8]:

$$\frac{\lambda_j}{\sum_{k=1}^{d} \lambda_k}, \tag{32}$$

  for the j-th component. Then, we sort the features from the largest to smallest ratio and select the p best components or up to the component where a significant drop of the ratio happens.

Dual Principal Component Analysis

Dual Principal Component Analysis

- Assume the case where the dimensionality of data is high and much greater than the sample size, i.e., $d \gg n$. In this case, consider the incomplete SVD of $\breve{\boldsymbol{X}}$:

$$\breve{\boldsymbol{X}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top, \tag{33}$$

  where here, $\boldsymbol{U} \in \mathbb{R}^{d \times p}$ and $\boldsymbol{V} \in \mathbb{R}^{n \times p}$ contain the p leading left and right singular vectors of $\breve{\boldsymbol{X}}$, respectively, where p is the number of "non-zero" singular values of $\breve{\boldsymbol{X}}$ and usually $p \ll d$.
- Here, the $\boldsymbol{\Sigma} \in \mathbb{R}^{p \times p}$ is a square matrix having the p largest non-zero singular values of $\breve{\boldsymbol{X}}$.
- As the $\boldsymbol{\Sigma}$ is a square diagonal matrix and its diagonal includes non-zero entries (is full-rank), it is invertible [9]. Therefore, $\boldsymbol{\Sigma}^{-1} = \textbf{diag}([\frac{1}{\sigma_1}, \dots, \frac{1}{\sigma_p}]^\top)$ if we have $\boldsymbol{\Sigma} = \textbf{diag}([\sigma_1, \dots, \sigma_p]^\top)$.

Dual Principal Component Analysis: Projection

- We had:

$$\breve{\boldsymbol{X}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top,$$

- Recall Eq. (7) for projection onto PCA subspace: $\widetilde{\boldsymbol{X}} = \boldsymbol{U}^\top \breve{\boldsymbol{X}}$. On the other hand, according to Eq. (33), we have:

$$\breve{\boldsymbol{X}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top \implies \boldsymbol{U}^\top \breve{\boldsymbol{X}} = \underbrace{\boldsymbol{U}^\top \boldsymbol{U}}_{\boldsymbol{I}} \boldsymbol{\Sigma}\boldsymbol{V}^\top = \boldsymbol{\Sigma}\boldsymbol{V}^\top. \tag{34}$$

- According to Eqs. (7) and (34), we have:

$$\therefore \qquad \widetilde{\boldsymbol{X}} = \boldsymbol{\Sigma}\boldsymbol{V}^\top \tag{35}$$

The Eq. (35) can be used for projecting data onto PCA subspace instead of Eq. (7). This is projection of training data in dual PCA.

Dual Principal Component Analysis: Reconstruction

- We had:

$$\breve{\boldsymbol{X}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top,$$

- According to Eq. (33), we have:

$$\breve{\boldsymbol{X}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top \implies \breve{\boldsymbol{X}}\boldsymbol{V} = \boldsymbol{U}\boldsymbol{\Sigma}\underbrace{\boldsymbol{V}^\top\boldsymbol{V}}_{\boldsymbol{I}} = \boldsymbol{U}\boldsymbol{\Sigma}$$
$$\implies \boldsymbol{U} = \breve{\boldsymbol{X}}\boldsymbol{V}\boldsymbol{\Sigma}^{-1}. \quad (36)$$

- Plugging Eq. (36) in Eq. (8) gives us:

$$\widehat{\boldsymbol{X}} = \boldsymbol{U}\widetilde{\boldsymbol{X}} + \boldsymbol{\mu}_x \overset{(36)}{=} \breve{\boldsymbol{X}}\boldsymbol{V}\boldsymbol{\Sigma}^{-1}\widetilde{\boldsymbol{X}} + \boldsymbol{\mu}_x$$
$$\overset{(35)}{=} \breve{\boldsymbol{X}}\boldsymbol{V}\underbrace{\boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma}}_{\boldsymbol{I}}\boldsymbol{V}^\top + \boldsymbol{\mu}_x$$
$$\implies \widehat{\boldsymbol{X}} = \breve{\boldsymbol{X}}\boldsymbol{V}\boldsymbol{V}^\top + \boldsymbol{\mu}_x. \quad (37)$$

The Eq. (37) can be used for reconstruction of data instead of Eq. (8). This is reconstruction of training data in dual PCA.

Dual Principal Component Analysis: Out-of-sample Projection

- Recall Eq. (9) for projection of an out-of-sample point $\boldsymbol{x}_t$ onto PCA subspace:

$$\widetilde{\boldsymbol{x}}_t = \boldsymbol{U}^\top \breve{\boldsymbol{x}}_t.$$

- According to Eq. (36):

$$\boldsymbol{U} = \breve{\boldsymbol{X}} \boldsymbol{V} \boldsymbol{\Sigma}^{-1},$$

  we have:

$$\boldsymbol{U}^\top \overset{(36)}{=} \boldsymbol{\Sigma}^{-\top} \boldsymbol{V}^\top \breve{\boldsymbol{X}}^\top \overset{(a)}{=} \boldsymbol{\Sigma}^{-1} \boldsymbol{V}^\top \breve{\boldsymbol{X}}^\top \tag{38}$$

$$\overset{(9)}{\Longrightarrow} \widetilde{\boldsymbol{x}}_t = \boldsymbol{\Sigma}^{-1} \boldsymbol{V}^\top \breve{\boldsymbol{X}}^\top \breve{\boldsymbol{x}}_t, \tag{39}$$

  where (a) is because $\boldsymbol{\Sigma}^{-1}$ is diagonal and thus symmetric. The Eq. (39) can be used for projecting out-of-sample data point onto PCA subspace instead of Eq. (9). This is out-of-sample projection in dual PCA.

- Considering all the n_t out-of-sample data points, the projection is:

$$\widetilde{\boldsymbol{X}}_t = \boldsymbol{\Sigma}^{-1} \boldsymbol{V}^\top \breve{\boldsymbol{X}}^\top \breve{\boldsymbol{X}}_t. \tag{40}$$

Dual Principal Component Analysis: Out-of-sample Reconstruction

- Recall Eq. (10) for reconstruction of an out-of-sample point $\boldsymbol{x}_t$:

$$\widehat{\boldsymbol{x}}_t = \boldsymbol{U}\boldsymbol{U}^\top \breve{\boldsymbol{x}}_t + \boldsymbol{\mu}_x = \boldsymbol{U}\widetilde{\boldsymbol{x}}_t + \boldsymbol{\mu}_x.$$

- According to Eqs. (36) and (38), we have:

$$\begin{aligned} \boldsymbol{U}\boldsymbol{U}^\top &= \breve{\boldsymbol{X}}\boldsymbol{V}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma}^{-1}\boldsymbol{V}^\top\breve{\boldsymbol{X}}^\top \\ \overset{(10)}{\Longrightarrow}\ \widehat{\boldsymbol{x}}_t &= \breve{\boldsymbol{X}}\boldsymbol{V}\boldsymbol{\Sigma}^{-2}\boldsymbol{V}^\top\breve{\boldsymbol{X}}^\top\breve{\boldsymbol{x}}_t + \boldsymbol{\mu}_x. \end{aligned} \tag{41}$$

  The Eq. (41) can be used for reconstruction of an out-of-sample data point instead of Eq. (10). This is out-of-sample reconstruction in dual PCA.

- Considering all the n_t out-of-sample data points, the reconstruction is:

$$\widehat{\boldsymbol{X}}_t = \breve{\boldsymbol{X}}\boldsymbol{V}\boldsymbol{\Sigma}^{-2}\boldsymbol{V}^\top\breve{\boldsymbol{X}}^\top\breve{\boldsymbol{X}}_t + \boldsymbol{\mu}_x. \tag{42}$$

Why is Dual PCA Useful?

- The dual PCA can be useful for two reasons:
  1. As can be seen in Eqs. (35), (37), (39), and (41):

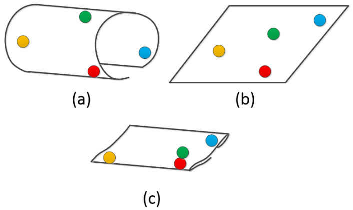
$$\widetilde{\boldsymbol{X}} = \boldsymbol{\Sigma}\boldsymbol{V}^\top,$$
$$\widehat{\boldsymbol{X}} = \breve{\boldsymbol{X}}\boldsymbol{V}\boldsymbol{V}^\top + \boldsymbol{\mu}_x,$$
$$\widetilde{\boldsymbol{x}}_t = \boldsymbol{\Sigma}^{-1}\boldsymbol{V}^\top\breve{\boldsymbol{X}}^\top\breve{\boldsymbol{x}}_t,$$
$$\widehat{\boldsymbol{x}}_t = \breve{\boldsymbol{X}}\boldsymbol{V}\boldsymbol{\Sigma}^{-2}\boldsymbol{V}^\top\breve{\boldsymbol{X}}^\top\breve{\boldsymbol{x}}_t + \boldsymbol{\mu}_x,$$

     the formulae for dual PCA only include $\boldsymbol{V}$ and not $\boldsymbol{U}$. The columns of $\boldsymbol{V}$ are the eigenvectors of $\breve{\boldsymbol{X}}^\top\breve{\boldsymbol{X}} \in \mathbb{R}^{n\times n}$ and the columns of $\boldsymbol{U}$ are the eigenvectors of $\breve{\boldsymbol{X}}\breve{\boldsymbol{X}}^\top \in \mathbb{R}^{d\times d}$. In case the dimensionality of data is much high and greater than the sample size, i.e., $n \ll d$, computation of eigenvectors of $\breve{\boldsymbol{X}}^\top\breve{\boldsymbol{X}}$ is easier and faster than $\breve{\boldsymbol{X}}\breve{\boldsymbol{X}}^\top$ and also requires less storage. Therefore, dual PCA is more efficient than direct PCA in this case in terms of both speed and storage. Note that the results of PCA and dual PCA are exactly the same.
  2. Some inner product forms, such as $\breve{\boldsymbol{X}}^\top\breve{\boldsymbol{x}}_t$, have appeared in the formulae of dual PCA. This provides opportunity for kernelizing the PCA to have kernel PCA using the so-called kernel trick. As will be seen in the next section, we use dual PCA in formulation of kernel PCA.

Kernel Principal Component Analysis

Kernel Principal Component Analysis

- The PCA is a linear method because the projection is linear. In case the data points exist on a non-linear sub-manifold, the linear subspace learning might not be completely effective.



- In order to handle this problem of PCA, we have two options. We should either change PCA to become a nonlinear method or we can leave the PCA to be linear but change the data hoping to fall on a linear or close to linear manifold.
- Here, we do the latter so we change the data. We increase the dimensionality of data by mapping the data to feature space with higher dimensionality hoping that in the feature space, it falls on a linear manifold. This is referred to as "blessing of dimensionality" in the literature [10] which is pursued using kernels [11]. This PCA method which uses the kernel of data is named "kernel PCA" [12].

Kernel and Centered Kernel

- Kernel:

$$k(\boldsymbol{x}_1, \boldsymbol{x}_2) := \phi(\boldsymbol{x}_1)^\top \phi(\boldsymbol{x}_2), \tag{43}$$
$$\boldsymbol{K}(\boldsymbol{X}_1, \boldsymbol{X}_2) := \boldsymbol{\Phi}(\boldsymbol{X}_1)^\top \boldsymbol{\Phi}(\boldsymbol{X}_2). \tag{44}$$

- If we want the pulled data $\boldsymbol{\Phi}(\boldsymbol{X})$ to be centered, i.e.:

$$\breve{\boldsymbol{\Phi}}(\boldsymbol{X}) := \boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{H}, \tag{45}$$

  we should double center the kernel matrix because if we use centered pulled data, we have:

$$\breve{\boldsymbol{\Phi}}(\boldsymbol{X})^\top \breve{\boldsymbol{\Phi}}(\boldsymbol{X}) = \big(\boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{H}\big)^\top \big(\boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{H}\big) = \boldsymbol{H}\boldsymbol{\Phi}(\boldsymbol{X})^\top \boldsymbol{\Phi}(\boldsymbol{X})\boldsymbol{H} = \boldsymbol{H}\boldsymbol{K}_x\boldsymbol{H},$$

  which is the double-centered kernel matrix. Thus:

$$\breve{\boldsymbol{K}}_x := \boldsymbol{H}\boldsymbol{K}_x\boldsymbol{H} = \breve{\boldsymbol{\Phi}}(\boldsymbol{X})^\top \breve{\boldsymbol{\Phi}}(\boldsymbol{X}), \tag{46}$$

  where $\breve{\boldsymbol{K}}_x$ denotes the double-centered kernel matrix.

Kernel Principal Component Analysis: Projection

- We apply incomplete SVD on the centered pulled (mapped) data $\breve{\boldsymbol{\Phi}}(\boldsymbol{X})$:

$$\mathbb{R}^{t\times n} \ni \breve{\boldsymbol{\Phi}}(\boldsymbol{X}) = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top, \tag{47}$$

  where $\boldsymbol{U} \in \mathbb{R}^{t\times p}$ and $\boldsymbol{V} \in \mathbb{R}^{n\times p}$ contain the p leading left and right singular vectors of $\breve{\boldsymbol{\Phi}}(\boldsymbol{X})$, respectively, where p is the number of "non-zero" singular values of $\breve{\boldsymbol{\Phi}}(\boldsymbol{X})$ and usually $p \ll t$. Here, the $\boldsymbol{\Sigma} \in \mathbb{R}^{p\times p}$ is a square matrix having the p largest non-zero singular values of $\breve{\boldsymbol{\Phi}}(\boldsymbol{X})$.

- However, as mentioned before, the pulled data are not necessarily available so Eq. (47) cannot be done. The kernel, however, is available. Therefore, we apply eigen-decomposition [3] to the double-centered kernel:

$$\breve{\boldsymbol{K}}_x \boldsymbol{V} = \boldsymbol{V}\boldsymbol{\Lambda}, \tag{48}$$

  where the columns of $\boldsymbol{V}$ and the diagonal of $\boldsymbol{\Lambda}$ are the eigenvectors and eigenvalues of $\breve{\boldsymbol{K}}_x$, respectively.

- The columns of $\boldsymbol{V}$ in Eq. (47) are the right singular vectors of $\breve{\boldsymbol{\Phi}}(\boldsymbol{X})$ which are equivalent to the eigenvectors of $\breve{\boldsymbol{\Phi}}(\boldsymbol{X})^\top \breve{\boldsymbol{\Phi}}(\boldsymbol{X}) = \breve{\boldsymbol{K}}_x$, according to the Preliminaries slides. Also, according to those slides, the diagonal of $\boldsymbol{\Sigma}$ in Eq. (47) is equivalent to the square root of eigenvalues of $\breve{\boldsymbol{K}}_x$.

- Therefore, in practice where the pulling function is not necessarily available, we use Eq. (48) in order to find the $\boldsymbol{V}$ and $\boldsymbol{\Sigma}$ in Eq. (47).

Kernel Principal Component Analysis: Projection

- We had:

$$\breve{\boldsymbol{K}}_x \boldsymbol{V} = \boldsymbol{V}\boldsymbol{\Lambda}.$$

- The Eq. (48) can be restated as:

$$\breve{\boldsymbol{K}}_x \boldsymbol{V} = \boldsymbol{V}\boldsymbol{\Sigma}^2, \tag{49}$$

  to be compatible to Eq. (47):

$$\mathbb{R}^{t \times n} \ni \breve{\boldsymbol{\Phi}}(\boldsymbol{X}) = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top.$$

- It is noteworthy that because of using Eq. (49) instead of Eq. (47), **the projection directions $\boldsymbol{U}$ are not available in kernel PCA to be observed or plotted.**
- Similar to what we did for Eq. (35):

$$\begin{aligned} & \breve{\boldsymbol{\Phi}}(\boldsymbol{X}) = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top \\ & \implies \boldsymbol{U}^\top \breve{\boldsymbol{\Phi}}(\boldsymbol{X}) = \underbrace{\boldsymbol{U}^\top \boldsymbol{U}}_{\boldsymbol{I}} \boldsymbol{\Sigma}\boldsymbol{V}^\top = \boldsymbol{\Sigma}\boldsymbol{V}^\top \\ \therefore \quad & \boldsymbol{\Phi}(\widetilde{\boldsymbol{X}}) = \boldsymbol{U}^\top \breve{\boldsymbol{\Phi}}(\boldsymbol{X}) = \boldsymbol{\Sigma}\boldsymbol{V}^\top, \end{aligned} \tag{50}$$

  where $\boldsymbol{\Sigma}$ and $\boldsymbol{V}$ are obtained from Eq. (49). The Eq. (50) is projection of the training data in kernel PCA.

Kernel Principal Component Analysis: Reconstruction

- Similar to what we did for Eq. (37):

$$\breve{\boldsymbol{\Phi}}(\boldsymbol{X}) = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^\top \implies \breve{\boldsymbol{\Phi}}(\boldsymbol{X})\boldsymbol{V} = \boldsymbol{U}\boldsymbol{\Sigma}\underbrace{\boldsymbol{V}^\top\boldsymbol{V}}_{\boldsymbol{I}} = \boldsymbol{U}\boldsymbol{\Sigma}$$
$$\implies \boldsymbol{U} = \breve{\boldsymbol{\Phi}}(\boldsymbol{X})\boldsymbol{V}\boldsymbol{\Sigma}^{-1}. \tag{51}$$

- Therefore, the reconstruction is:

$$\boldsymbol{\Phi}(\widehat{\boldsymbol{X}}) = \boldsymbol{U}\boldsymbol{\Phi}(\widetilde{\boldsymbol{X}}) + \boldsymbol{\mu}_x \overset{(51)}{=} \breve{\boldsymbol{\Phi}}(\boldsymbol{X})\boldsymbol{V}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Phi}(\widetilde{\boldsymbol{X}}) + \boldsymbol{\mu}_x$$
$$\overset{(50)}{=} \breve{\boldsymbol{\Phi}}(\boldsymbol{X})\boldsymbol{V}\underbrace{\boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma}}_{\boldsymbol{I}}\boldsymbol{V}^\top + \boldsymbol{\mu}_x$$
$$\implies \boldsymbol{\Phi}(\widehat{\boldsymbol{X}}) = \breve{\boldsymbol{\Phi}}(\boldsymbol{X})\boldsymbol{V}\boldsymbol{V}^\top + \boldsymbol{\mu}_x. \tag{52}$$

However, the $\breve{\boldsymbol{\Phi}}(\boldsymbol{X})$ is not available necessarily; therefore, we **cannot reconstruct the training data in kernel PCA**.

Kernel Principal Component Analysis: Out-of-sample Projection

- Similar to what we did for Eq. (39):

$$\begin{aligned} &\boldsymbol{U}^\top \overset{(51)}{=} \boldsymbol{\Sigma}^{-\top} \boldsymbol{V}^\top \breve{\boldsymbol{\Phi}}(\boldsymbol{X})^\top \overset{(a)}{=} \boldsymbol{\Sigma}^{-1} \boldsymbol{V}^\top \breve{\boldsymbol{\Phi}}(\boldsymbol{X})^\top \\ &\implies \phi(\widetilde{\boldsymbol{x}}_t) = \boldsymbol{U}^\top \breve{\phi}(\boldsymbol{x}_t) = \boldsymbol{\Sigma}^{-1} \boldsymbol{V}^\top \breve{\boldsymbol{\Phi}}(\boldsymbol{X})^\top \breve{\phi}(\boldsymbol{x}_t), \\ &\implies \phi(\widetilde{\boldsymbol{x}}_t) = \boldsymbol{\Sigma}^{-1} \boldsymbol{V}^\top \breve{\boldsymbol{k}}_t, \end{aligned} \tag{53}$$

  where (a) is because $\boldsymbol{\Sigma}^{-1}$ is diagonal and thus symmetric and the $\breve{\boldsymbol{k}}_t \in \mathbb{R}^n$ is calculated by (see Appendix C in our tutorial [13] for proof):

$$\mathbb{R}^n \ni \breve{\boldsymbol{k}}_t = \boldsymbol{k}_t - \frac{1}{n}\boldsymbol{1}_{n\times n}\boldsymbol{k}_t - \frac{1}{n}\boldsymbol{K}\boldsymbol{1}_n + \frac{1}{n^2}\boldsymbol{1}_{n\times n}\boldsymbol{K}\boldsymbol{1}_n. \tag{54}$$

  The Eq. (53) is the projection of out-of-sample data in kernel PCA.
- Considering all the n_t out-of-sample data points, $\boldsymbol{X}_t$, the projection is:

$$\phi(\widetilde{\boldsymbol{X}}_t) = \boldsymbol{\Sigma}^{-1} \boldsymbol{V}^\top \breve{\boldsymbol{K}}_t, \tag{55}$$

  where $\breve{\boldsymbol{K}}_t$ is calculated by (see Appendix C in our tutorial [13] for proof):

$$\mathbb{R}^{n\times n_t} \ni \breve{\boldsymbol{K}}_t = \boldsymbol{K}_t - \frac{1}{n}\boldsymbol{1}_{n\times n}\boldsymbol{K}_t - \frac{1}{n}\boldsymbol{K}\boldsymbol{1}_{n\times n_t} + \frac{1}{n^2}\boldsymbol{1}_{n\times n}\boldsymbol{K}\boldsymbol{1}_{n\times n_t}, \tag{56}$$

  where $\mathbb{R}^{n\times n} \ni \boldsymbol{1}_{n\times n} := \boldsymbol{1}_n\boldsymbol{1}_n^\top$, $\mathbb{R}^{n\times n_t} \ni \boldsymbol{1}_{n\times n_t} := \boldsymbol{1}_n\boldsymbol{1}_{n_t}^\top$, $\mathbb{R}^n \ni \boldsymbol{1}_n := [1,\dots,1]^\top$, and $\mathbb{R}^{n_t} \ni \boldsymbol{1}_{n_t} := [1,\dots,1]^\top$.

Kernel Principal Component Analysis: Out-of-sample Reconstruction

- Similar to what we did for Eq. (41):

$$\begin{aligned}
&\implies \boldsymbol{U}\boldsymbol{U}^\top \overset{(51)}{=} \breve{\boldsymbol{\Phi}}(\boldsymbol{X})\boldsymbol{V}\boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma}^{-1}\boldsymbol{V}^\top\breve{\boldsymbol{\Phi}}(\boldsymbol{X})^\top \\
&\implies \phi(\widehat{\boldsymbol{x}}_t) = \breve{\boldsymbol{\Phi}}(\boldsymbol{X})\boldsymbol{V}\boldsymbol{\Sigma}^{-2}\boldsymbol{V}^\top\breve{\boldsymbol{\Phi}}(\boldsymbol{X})^\top\breve{\phi}(\boldsymbol{x}_t) + \boldsymbol{\mu}_x \\
&\implies \phi(\widehat{\boldsymbol{x}}_t) = \breve{\boldsymbol{\Phi}}(\boldsymbol{X})\boldsymbol{V}\boldsymbol{\Sigma}^{-2}\boldsymbol{V}^\top\breve{\boldsymbol{k}}_t + \boldsymbol{\mu}_x, \qquad (57)
\end{aligned}$$

  where the $\breve{\boldsymbol{k}}_t \in \mathbb{R}^n$ is calculated by Eq. (54).
- Considering all the n_t out-of-sample data points, $\boldsymbol{X}_t$, the reconstruction is:

$$\boldsymbol{\Phi}(\widehat{\boldsymbol{X}}_t) = \breve{\boldsymbol{\Phi}}(\boldsymbol{X})\boldsymbol{V}\boldsymbol{\Sigma}^{-2}\boldsymbol{V}^\top\breve{\boldsymbol{K}}_t + \boldsymbol{\mu}_x, \qquad (58)$$

  where $\breve{\boldsymbol{K}}_t$ is calculated by Eq. (56).
- In Eq. (57), the $\breve{\boldsymbol{\Phi}}(\boldsymbol{X})$ appeared at the left of expression, is not available necessarily; therefore, we **cannot reconstruct an out-of-sample point in kernel PCA**.
- According to Eqs. (52) and (57), we conclude that **kernel PCA is not able to reconstruct any data, whether training or out-of-sample**.

Why is Kernel PCA Useful?

- Finally, it is noteworthy that as the choice of the best kernel might be hard, the kernel PCA is not "always" effective in practice [9].
- However, it provides us some useful theoretical insights for explaining the PCA, Multi-Dimensional Scaling (MDS) [14], Isomap [15], Locally Linear Embedding (LLE) [16], and Laplacian Eigenmap (LE) [17] as **special cases of kernel PCA with their own kernels** (see [18] and chapter 2 in [19]).

Supervised Principal Component Analysis Using HSIC

Hilbert-Schmidt Independence Criterion

- Suppose we want to measure the **dependence** of two random variables.
- Measuring the **correlation** between them is easier because correlation is just "linear" dependence.
- According to [20], two random variables are independent if and only if any bounded continuous functions of them are uncorrelated. Therefore, if we map the two random variables $\boldsymbol{x}$ and $\boldsymbol{y}$ to two different ("separable") Reproducing Kernel Hilbert Spaces (RKHSs) and have $\phi(\boldsymbol{x})$ and $\phi(\boldsymbol{y})$, we can measure the correlation of $\phi(\boldsymbol{x})$ and $\phi(\boldsymbol{y})$ in Hilbert space to have an estimation of dependence of $\boldsymbol{x}$ and $\boldsymbol{y}$ in the original space.
- An empirical estimation of the Hilbert-Schmidt Independence Criterion (HSIC) is [21]:

$$\text{HSIC} := \frac{1}{(n-1)^2}\,\textbf{tr}(\ddot{\boldsymbol{K}}_x \boldsymbol{H} \boldsymbol{K}_y \boldsymbol{H}), \tag{59}$$

  where $\ddot{\boldsymbol{K}}_x$ and $\boldsymbol{K}_y$ are the kernels over $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively. In other words, $\ddot{\boldsymbol{K}}_x = \phi(\boldsymbol{x})^\top \phi(\boldsymbol{x})$ and $\boldsymbol{K}_y = \phi(\boldsymbol{y})^\top \phi(\boldsymbol{y})$.
- The term $1/(n-1)^2$ is used for normalization. The $\boldsymbol{H}$ is the centering matrix:

$$\mathbb{R}^{n\times n} \ni \boldsymbol{H} = \boldsymbol{I} - (1/n)\boldsymbol{1}\boldsymbol{1}^\top. \tag{60}$$

- The $\boldsymbol{H}\boldsymbol{K}_y\boldsymbol{H}$ double centers the $\boldsymbol{K}_y$ in HSIC.
- The HSIC (Eq. (59)) measures the dependence of two random variable vectors $\boldsymbol{x}$ and $\boldsymbol{y}$. Note that HSIC $= 0$ and HSIC > 0 mean that $\boldsymbol{x}$ and $\boldsymbol{y}$ are independent and dependent, respectively. The greater the HSIC, the greater dependence they have.

Supervised PCA

- Supervised PCA (SPCA) [22] uses the HSIC. We have the data $\boldsymbol{X} = [\boldsymbol{x}_1, \dots, \boldsymbol{x}_n] \in \mathbb{R}^{d \times n}$ and the labels $\boldsymbol{Y} = [\boldsymbol{y}_1, \dots, \boldsymbol{y}_n] \in \mathbb{R}^{\ell \times n}$, where ℓ is the dimensionality of the labels and we usually have $\ell = 1$. However, in case the labels are encoded (e.g., one-hot-encoded) or SPCA is used for regression (e.g., see [23]), we have $\ell > 1$.
- SPCA tries to maximize the dependence of the projected data points $\boldsymbol{U}^\top \boldsymbol{X}$ and the labels $\boldsymbol{Y}$. It uses a linear kernel for the projected data points:

$$\ddot{\boldsymbol{K}}_x = (\boldsymbol{U}^\top \boldsymbol{X})^\top (\boldsymbol{U}^\top \boldsymbol{X}) = \boldsymbol{X}^\top \boldsymbol{U} \boldsymbol{U}^\top \boldsymbol{X}, \tag{61}$$

  and an arbitrary kernel $\boldsymbol{K}_y$ over $\boldsymbol{Y}$.
- For classification task, one of the best choices for the $\boldsymbol{K}_y$ is delta kernel [22] where the (i, j)-th element of kernel is:

$$\boldsymbol{K}_y = \delta_{\boldsymbol{y}_i, \boldsymbol{y}_j} := \begin{cases} 1 & \text{if } \boldsymbol{y}_i = \boldsymbol{y}_j, \\ 0 & \text{if } \boldsymbol{y}_i \neq \boldsymbol{y}_j, \end{cases} \tag{62}$$

  where $\delta_{\boldsymbol{y}_i, \boldsymbol{y}_j}$ is the Kronecker delta which is one if the $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ belong to the same class.
- Another good choice for kernel in classification task in SPCA is an arbitrary kernel (e.g., linear kernel $\boldsymbol{K}_y = \boldsymbol{Y}^\top \boldsymbol{Y}$) over $\boldsymbol{Y}$ where the columns of $\boldsymbol{Y}$ are one-hot encoded. This is a good choice because the distances of classes will be equal; otherwise, some classes will fall closer than the others for no reason and fairness between classes goes away.
- The SPCA can also be used for regression (e.g., see [23]) and that is one of the advantages of SPCA. In that case, a good choice for $\boldsymbol{K}_y$ is an arbitrary kernel (e.g., linear kernel $\boldsymbol{K}_y = \boldsymbol{Y}^\top \boldsymbol{Y}$) over $\boldsymbol{Y}$ where the columns of the $\boldsymbol{Y}$, i.e., labels, are the observations in regression. Here, the distances of observations have meaning and should not be manipulated.

Supervised PCA

- HSIC was:

$$\text{HSIC} := \frac{1}{(n-1)^2}\,\mathbf{tr}(\ddot{\boldsymbol{K}}_x \boldsymbol{H} \boldsymbol{K}_y \boldsymbol{H}).$$

- The HSIC in SPCA case becomes:

$$\text{HSIC} = \frac{1}{(n-1)^2}\,\mathbf{tr}(\boldsymbol{X}^\top \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{X}\boldsymbol{H}\boldsymbol{K}_y\boldsymbol{H}). \tag{63}$$

  where $\boldsymbol{U} \in \mathbb{R}^{d \times p}$ is the unknown projection matrix for projection onto the SPCA subspace and should be found. The desired dimensionality of the subspace is p and usually $p \ll d$.
- We should maximize the HSIC in order to maxzimize the dependence of $\boldsymbol{U}^\top \boldsymbol{X}$ and $\boldsymbol{Y}$. Hence:

$$\begin{aligned} \underset{\boldsymbol{U}}{\text{maximize}} \quad & \mathbf{tr}(\boldsymbol{X}^\top \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{X}\boldsymbol{H}\boldsymbol{K}_y\boldsymbol{H}), \\ \text{subject to} \quad & \boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}, \end{aligned} \tag{64}$$

  where the constraint ensures that the $\boldsymbol{U}$ is an orthogonal matrix, i.e., the SPCA directions are orthonormal.

Supervised PCA

- We had:

$$\begin{aligned} \underset{\boldsymbol{U}}{\text{maximize}} \quad & \textbf{tr}(\boldsymbol{X}^\top \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{X}\boldsymbol{H}\boldsymbol{K}_y\boldsymbol{H}), \\ \text{subject to} \quad & \boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}. \end{aligned}$$

- Using Lagrangian [2], we have:

$$\begin{aligned} \mathcal{L} &= \textbf{tr}(\boldsymbol{X}^\top \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{X}\boldsymbol{H}\boldsymbol{K}_y\boldsymbol{H}) - \textbf{tr}\big(\boldsymbol{\Lambda}^\top (\boldsymbol{U}^\top \boldsymbol{U} - \boldsymbol{I})\big) \\ &\overset{(a)}{=} \textbf{tr}(\boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{X}\boldsymbol{H}\boldsymbol{K}_y\boldsymbol{H}\boldsymbol{X}^\top) - \textbf{tr}\big(\boldsymbol{\Lambda}^\top (\boldsymbol{U}^\top \boldsymbol{U} - \boldsymbol{I})\big), \end{aligned}$$

  where (a) is because of the cyclic property of trace and $\boldsymbol{\Lambda} \in \mathbb{R}^{p\times p}$ is a diagonal matrix $\textbf{diag}([\lambda_1, \dots, \lambda_p]^\top)$ including the Lagrange multipliers.
- Setting the derivative of Lagrangian to zero gives:

$$\begin{aligned} \mathbb{R}^{d\times p} \ni \frac{\partial \mathcal{L}}{\partial \boldsymbol{U}} &= 2\boldsymbol{X}\boldsymbol{H}\boldsymbol{K}_y\boldsymbol{H}\boldsymbol{X}^\top \boldsymbol{U} - 2\boldsymbol{U}\boldsymbol{\Lambda} \overset{\text{set}}{=} 0 \\ \implies \boldsymbol{X}\boldsymbol{H}\boldsymbol{K}_y\boldsymbol{H}\boldsymbol{X}^\top \boldsymbol{U} &= \boldsymbol{U}\boldsymbol{\Lambda}, \end{aligned} \tag{65}$$

  which is the eigen-decomposition of $\boldsymbol{X}\boldsymbol{H}\boldsymbol{K}_y\boldsymbol{H}\boldsymbol{X}^\top$ where the columns of $\boldsymbol{U}$ and the diagonal of $\boldsymbol{\Lambda}$ are the eigenvectors and eigenvalues of $\boldsymbol{X}\boldsymbol{H}\boldsymbol{K}_y\boldsymbol{H}\boldsymbol{X}^\top$, respectively [3].
- The eigenvectors and eigenvalues are sorted from the leading (largest eigenvalue) to the trailing (smallest eigenvalue) because we are maximizing in the optimization problem.
- As a conclusion, if projecting onto the SPCA subspace or $\textbf{span}\{\boldsymbol{u}_1, \dots, \boldsymbol{u}_p\}$, the SPCA directions $\{\boldsymbol{u}_1, \dots, \boldsymbol{u}_p\}$ are the sorted eigenvectors of $\boldsymbol{X}\boldsymbol{H}\boldsymbol{K}_y\boldsymbol{H}\boldsymbol{X}^\top$. In other words, the columns of the projection matrix $\boldsymbol{U}$ in SPCA are the p leading eigenvectors of $\boldsymbol{X}\boldsymbol{H}\boldsymbol{K}_y\boldsymbol{H}\boldsymbol{X}^\top$.

Supervised PCA

- Similar to what we had in PCA, the projection, projection of out-of-sample, reconstruction, and reconstruction of out-of-sample in SPCA are:

$$\widetilde{\boldsymbol{X}} = \boldsymbol{U}^\top \boldsymbol{X}, \tag{66}$$
$$\widetilde{\boldsymbol{x}}_t = \boldsymbol{U}^\top \boldsymbol{x}_t, \tag{67}$$
$$\widehat{\boldsymbol{X}} = \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{X} = \boldsymbol{U}\widetilde{\boldsymbol{X}}, \tag{68}$$
$$\widehat{\boldsymbol{x}}_t = \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{x}_t = \boldsymbol{U}\widetilde{\boldsymbol{x}}_t, \tag{69}$$

  respectively.
- In SPCA, there is no need to center the data as the centering is already handled by $\boldsymbol{H}$ in HSIC.
- Considering all the n_t out-of-sample data points, the projection and reconstruction are:

$$\widetilde{\boldsymbol{X}}_t = \boldsymbol{U}^\top \boldsymbol{X}_t, \tag{70}$$
$$\widehat{\boldsymbol{X}}_t = \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{X}_t = \boldsymbol{U}\widetilde{\boldsymbol{X}}_t, \tag{71}$$

  respectively.

PCA is a special case of SPCA!

- Not considering the similarities of the labels means that we do not care about the class labels so we are unsupervised.
- If we do not consider the similarities of labels, the kernel over the labels becomes the identity matrix, $\boldsymbol{K}_y = \boldsymbol{I}$.
- According to Eq. (65), SPCA is the eigen-decomposition of $\boldsymbol{X}\boldsymbol{H}\boldsymbol{K}_y\boldsymbol{H}\boldsymbol{X}^\top$. In this case, this matrix becomes:

$$\begin{aligned}\boldsymbol{X}\boldsymbol{H}\boldsymbol{K}_y\boldsymbol{H}\boldsymbol{X}^\top &= \boldsymbol{X}\boldsymbol{H}\boldsymbol{I}\boldsymbol{H}\boldsymbol{X}^\top = \boldsymbol{X}\boldsymbol{H}\boldsymbol{I}\boldsymbol{H}^\top\boldsymbol{X}^\top \\ &= \boldsymbol{X}\boldsymbol{H}\boldsymbol{H}^\top\boldsymbol{X}^\top = (\boldsymbol{X}\boldsymbol{H})(\boldsymbol{X}\boldsymbol{H})^\top \\ &\overset{(5)}{=} \breve{\boldsymbol{X}}\breve{\boldsymbol{X}}^\top \overset{(17)}{=} \boldsymbol{S},\end{aligned}$$

  which is the covariance matrix whose eigenvectors are the PCA directions.
- Thus, if we do not consider the similarities of labels, i.e., we are unsupervised, SPCA reduces to PCA as expected.

More Information

- For reading about dual SPCA, kernel SPCA, and eigenfaces, see our tutorial: "Unsupervised and supervised principal component analysis: Tutorial" [13]

Acknowledgment

- Some slides are based on our tutorial paper: "Unsupervised and supervised principal component analysis: Tutorial" [13]
- Some slides of this slide deck are inspired by teachings of Prof. Ali Ghodsi at University of Waterloo, Department of Statistics.
- The code of PCA in my GitHub page (in Python language): `https://github.com/bghojogh/Principal-Component-Analysis`
- PCA in sklearn: `https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html`

References

[1] S. T. Dumais, "Latent semantic analysis," *Annual review of information science and technology*, vol. 38, no. 1, pp. 188–230, 2004.

[2] S. Boyd and L. Vandenberghe, *Convex optimization*.
Cambridge university press, 2004.

[3] B. Ghojogh, F. Karray, and M. Crowley, "Eigenvalue and generalized eigenvalue problems: Tutorial," *arXiv preprint arXiv:1903.11240*, 2019.

[4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*.
MIT press, 2016.

[5] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[6] T. P. Minka, "Automatic choice of dimensionality for pca," in *Advances in neural information processing systems*, pp. 598–604, 2001.

[7] R. B. Cattell, "The scree test for the number of factors," *Multivariate behavioral research*, vol. 1, no. 2, pp. 245–276, 1966.

[8] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley interdisciplinary reviews: computational statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[9] A. Ghodsi, "Dimensionality reduction: a short tutorial," *Department of Statistics and Actuarial Science, Univ. of Waterloo, Ontario, Canada*, vol. 37, 2006.

References (cont.)

[10] D. L. Donoho, “High-dimensional data analysis: The curses and blessings of dimensionality,” *AMS math challenges lecture*, 2000.

[11] T. Hofmann, B. Schölkopf, and A. J. Smola, “Kernel methods in machine learning,” *The annals of statistics*, pp. 1171–1220, 2008.

[12] B. Schölkopf, A. Smola, and K.-R. Müller, “Kernel principal component analysis,” in *International conference on artificial neural networks*, pp. 583–588, Springer, 1997.

[13] B. Ghojogh and M. Crowley, “Unsupervised and supervised principal component analysis: Tutorial,” *arXiv preprint arXiv:1906.03148*, 2019.

[14] M. A. Cox and T. F. Cox, “Multidimensional scaling,” in *Handbook of data visualization*, pp. 315–347, Springer, 2008.

[15] J. B. Tenenbaum, V. De Silva, and J. C. Langford, “A global geometric framework for nonlinear dimensionality reduction,” *science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[16] S. T. Roweis and L. K. Saul, “Nonlinear dimensionality reduction by locally linear embedding,” *science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[17] M. Belkin and P. Niyogi, “Laplacian eigenmaps for dimensionality reduction and data representation,” *Neural computation*, vol. 15, no. 6, pp. 1373–1396, 2003.

[18] J. H. Ham, D. D. Lee, S. Mika, and B. Schölkopf, “A kernel view of the dimensionality reduction of manifolds,” in *International Conference on Machine Learning*, 2004.

References (cont.)

[19] H. Strange and R. Zwiggelaar, *Open Problems in Spectral Dimensionality Reduction*. Springer, 2014.

[20] M. Hein and O. Bousquet, "Kernels, associated structures and generalizations," *Max-Planck-Institut fuer biologische Kybernetik, Technical Report*, 2004.

[21] A. Gretton, O. Bousquet, A. Smola, and B. Schölkopf, "Measuring statistical dependence with Hilbert-Schmidt norms," in *International conference on algorithmic learning theory*, pp. 63–77, Springer, 2005.

[22] E. Barshan, A. Ghodsi, Z. Azimifar, and M. Z. Jahromi, "Supervised principal component analysis: Visualization, classification and regression on subspaces and submanifolds," *Pattern Recognition*, vol. 44, no. 7, pp. 1357–1371, 2011.

[23] B. Ghojogh and M. Crowley, "Instance ranking and numerosity reduction using matrix decomposition and subspace learning," in *Advances in Artificial Intelligence: 32nd Canadian Conference on Artificial Intelligence, Canadian AI 2019*, Springer, 2019.