Chapter 8: Unsupervised Learning: Dimensionality Reduction

Reza Rezazadegan

Sharif University of Technology

Fall 2022

High-dimensional data

Data can be very high dimensional. Examples include:

High-dimensional data

Data can be very high dimensional. Examples include:

- Digital images have as many as $height \times width \times c$ features where c is the number of color channels (usually 3 for color images and 1 for monochrome).

High-dimensional data

Data can be very high dimensional. Examples include:

- Digital images have as many as *height* $\times$ *width* $\times$ c features where c is the number of color channels (usually 3 for color images and 1 for monochrome).
- Time series $\{z_t\}_{t=1}^{N}$.

High-dimensional data

Data can be very high dimensional. Examples include:

- Digital images have as many as *height* $\times$ *width* $\times$ c features where c is the number of color channels (usually 3 for color images and 1 for monochrome).
- Time series $\{z_t\}_{t=1}^N$.
- Vector representations of words and documents (typically have hundreds of dimensions).

High-dimensional data

Data can be very high dimensional. Examples include:

- Digital images have as many as *height* $\times$ *width* $\times$ c features where c is the number of color channels (usually 3 for color images and 1 for monochrome).
- Time series $\{z_t\}_{t=1}^{N}$.
- Vector representations of words and documents (typically have hundreds of dimensions).

Problems with high dimensional data:

- Difficult (if not impossible) to visualize.

High-dimensional data

Data can be very high dimensional. Examples include:

- Digital images have as many as $height \times width \times c$ features where c is the number of color channels (usually 3 for color images and 1 for monochrome).
- Time series $\{z_t\}_{t=1}^{N}$.
- Vector representations of words and documents (typically have hundreds of dimensions).

Problems with high dimensional data:

- Difficult (if not impossible) to visualize.
- Increased number of features can dramatically increase the computational cost of ML algorithms.

High-dimensional data

Data can be very high dimensional. Examples include:

- Digital images have as many as *height* $\times$ *width* $\times$ c features where c is the number of color channels (usually 3 for color images and 1 for monochrome).
- Time series $\{z_t\}_{t=1}^{N}$.
- Vector representations of words and documents (typically have hundreds of dimensions).

Problems with high dimensional data:

- Difficult (if not impossible) to visualize.
- Increased number of features can dramatically increase the computational cost of ML algorithms.
- Notions of Euclidean distance and orthogonolity differ significantly in higher dimensions.

High-dimensional data cont.

More precisely we have:

High-dimensional data cont.

More precisely we have:

Theorem: In d-dimensional Euclidean space, if we randomly pick n points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ from the unit ball $||\mathbf{x}|| \leq 1$ then with probability $1 - \mathcal{O}(1/n)$ we have:

- $||\mathbf{x}_i|| \geq 1 - \frac{2 \log n}{d}$,

High-dimensional data cont.

More precisely we have:

Theorem: In d-dimensional Euclidean space, if we randomly pick n points $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$ from the unit ball $||\mathbf{x}|| \leq 1$ then with probability $1 - \mathcal{O}(1/n)$ we have:

- $||\mathbf{x}_i|| \geq 1 - \frac{2 \log n}{d}$,
- $|\langle \mathbf{x}_i, \mathbf{x}_j \rangle| \leq \sqrt{\frac{6 \log n}{d-1}}$ or $i \neq j$.

High-dimensional data cont.

More precisely we have:

Theorem: In d-dimensional Euclidean space, if we randomly pick n points $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ from the unit ball $||\mathbf{x}|| \leq 1$ then with probability $1 - \mathcal{O}(1/n)$ we have:

- $||\mathbf{x}_i|| \geq 1 - \frac{2 \log n}{d}$,
- $|\langle \mathbf{x}_i, \mathbf{x}_j \rangle| \leq \sqrt{\frac{6 \log n}{d-1}}$ or $i \neq j$.

In words this means that in high dimensions, random points drawn from the unit ball lie close to its boundary (have length near 1), and they are nearly orthogonal to each other.

Dimensionality reduction

- **Dimensionality reduction** means mapping data from a high dimensional space $\mathbb{R}^D$ to a low dimensional space $\mathbb{R}^d$ in such a way the minimizes the information loss (e.g. variance loss).

Dimensionality reduction

- **Dimensionality reduction** means mapping data from a high dimensional space $\mathbb{R}^D$ to a low dimensional space $\mathbb{R}^d$ in such a way the minimizes the information loss (e.g. variance loss).
- Note: High dimensional features are often highly correlated.

Dimensionality reduction

- **Dimensionality reduction** means mapping data from a high dimensional space $\mathbb{R}^D$ to a low dimensional space $\mathbb{R}^d$ in such a way the minimizes the information loss (e.g. variance loss).
- Note: High dimensional features are often highly correlated. For example pixel values are correlated with their neighbor pixels.

Dimensionality reduction

- **Dimensionality reduction** means mapping data from a high dimensional space $\mathbb{R}^D$ to a low dimensional space $\mathbb{R}^d$ in such a way the minimizes the information loss (e.g. variance loss).
- Note: High dimensional features are often highly correlated. For example pixel values are correlated with their neighbor pixels. Or future time series values correlated with their current values.

Dimensionality reduction

- **Dimensionality reduction** means mapping data from a high dimensional space $\mathbb{R}^D$ to a low dimensional space $\mathbb{R}^d$ in such a way the minimizes the information loss (e.g. variance loss).
- Note: High dimensional features are often highly correlated. For example pixel values are correlated with their neighbor pixels. Or future time series values correlated with their current values.
- Dimensionality reduction removes these extraneous features and can be either linear or nonlinear.

Dimensionality reduction

- **Dimensionality reduction** means mapping data from a high dimensional space $\mathbb{R}^D$ to a low dimensional space $\mathbb{R}^d$ in such a way the minimizes the information loss (e.g. variance loss).
- Note: High dimensional features are often highly correlated. For example pixel values are correlated with their neighbor pixels. Or future time series values correlated with their current values.
- Dimensionality reduction removes these extraneous features and can be either linear or nonlinear.
- Nonlinear dimensionality reduction is also called **Manifold Learning**.

Dimensionality reduction

- **Dimensionality reduction** means mapping data from a high dimensional space $\mathbb{R}^D$ to a low dimensional space $\mathbb{R}^d$ in such a way the minimizes the information loss (e.g. variance loss).
- Note: High dimensional features are often highly correlated. For example pixel values are correlated with their neighbor pixels. Or future time series values correlated with their current values.
- Dimensionality reduction removes these extraneous features and can be either linear or nonlinear.
- Nonlinear dimensionality reduction is also called **Manifold Learning**.
- A *manifold* is a generalization of smooth shapes, such as sphere, torus, etc. to higher dimensions.

Dimensionality reduction

- **Dimensionality reduction** means mapping data from a high dimensional space $\mathbb{R}^D$ to a low dimensional space $\mathbb{R}^d$ in such a way the minimizes the information loss (e.g. variance loss).
- Note: High dimensional features are often highly correlated. For example pixel values are correlated with their neighbor pixels. Or future time series values correlated with their current values.
- Dimensionality reduction removes these extraneous features and can be either linear or nonlinear.
- Nonlinear dimensionality reduction is also called **Manifold Learning**.
- A *manifold* is a generalization of smooth shapes, such as sphere, torus, etc. to higher dimensions.
- The assumption in manifold learning is that our data lies on a submanifold of $\mathbb{R}^d$ with a high codimension.

Dimensionality reduction

- **Dimensionality reduction** means mapping data from a high dimensional space $\mathbb{R}^D$ to a low dimensional space $\mathbb{R}^d$ in such a way the minimizes the information loss (e.g. variance loss).
- Note: High dimensional features are often highly correlated. For example pixel values are correlated with their neighbor pixels. Or future time series values correlated with their current values.
- Dimensionality reduction removes these extraneous features and can be either linear or nonlinear.
- Nonlinear dimensionality reduction is also called **Manifold Learning**.
- A *manifold* is a generalization of smooth shapes, such as sphere, torus, etc. to higher dimensions.
- The assumption in manifold learning is that our data lies on a submanifold of $\mathbb{R}^d$ with a high codimension. For example natural images or images of handwritten digits form a very small subset of all images.

Dimensionality reduction

- **Dimensionality reduction** means mapping data from a high dimensional space $\mathbb{R}^D$ to a low dimensional space $\mathbb{R}^d$ in such a way the minimizes the information loss (e.g. variance loss).
- Note: High dimensional features are often highly correlated. For example pixel values are correlated with their neighbor pixels. Or future time series values correlated with their current values.
- Dimensionality reduction removes these extraneous features and can be either linear or nonlinear.
- Nonlinear dimensionality reduction is also called **Manifold Learning**.
- A *manifold* is a generalization of smooth shapes, such as sphere, torus, etc. to higher dimensions.
- The assumption in manifold learning is that our data lies on a submanifold of $\mathbb{R}^d$ with a high codimension. For example natural images or images of handwritten digits form a very small subset of all images.
- Nonlinear methods can detects nonlinear transformations of features as well, e.g. nonlinear mappings of pictures.

Principal Component Analysis (PCA)

- PCA is a linear dimensionality reduction method.

Principal Component Analysis (PCA)

- PCA is a linear dimensionality reduction method. For a dataset $\mathcal{D} \subset \mathbb{R}^D$, it finds an orthonormal basis $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_D$ of $\mathbb{R}^D$ such that the variance of $\mathcal{D}$ along the coordinates $\mathbf{v}_1, \ldots, \mathbf{v}_i, \ldots, \mathbf{v}_D$ is monotonically decreasing in i.

Principal Component Analysis (PCA)

- PCA is a linear dimensionality reduction method. For a dataset $\mathcal{D} \subset \mathbb{R}^D$, it finds an orthonormal basis $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_D$ of $\mathbb{R}^D$ such that the variance of $\mathcal{D}$ along the coordinates $\mathbf{v}_1, \ldots, \mathbf{v}_i, \ldots, \mathbf{v}_D$ is monotonically decreasing in i.
- By the variance of a dataset $\mathcal{D}$ along a basis vector v_i, we mean the variance of the $\mathbf{v}_i$ coordinate of the dataset i.e. $\{\langle \mathbf{x}, \mathbf{v}_i \rangle | \mathbf{x} \in \mathcal{D}\}$.

Principal Component Analysis (PCA)

- PCA is a linear dimensionality reduction method. For a dataset $\mathcal{D} \subset \mathbb{R}^D$, it finds an orthonormal basis $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_D$ of $\mathbb{R}^D$ such that the variance of $\mathcal{D}$ along the coordinates $\mathbf{v}_1, \ldots, \mathbf{v}_i, \ldots, \mathbf{v}_D$ is monotonically decreasing in i.
- By the variance of a dataset $\mathcal{D}$ along a basis vector v_i, we mean the variance of the $\mathbf{v}_i$ coordinate of the dataset i.e. $\{\langle \mathbf{x}, \mathbf{v}_i \rangle | \mathbf{x} \in \mathcal{D}\}$.
- After applying the PCA transformation T, we can keep only the first 2 or 3 coordinates of the data, for visualization.

Principal Component Analysis (PCA)

- PCA is a linear dimensionality reduction method. For a dataset $\mathcal{D} \subset \mathbb{R}^D$, it finds an orthonormal basis $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_D$ of $\mathbb{R}^D$ such that the variance of $\mathcal{D}$ along the coordinates $\mathbf{v}_1, \ldots, \mathbf{v}_i, \ldots, \mathbf{v}_D$ is monotonically decreasing in i.
- By the variance of a dataset $\mathcal{D}$ along a basis vector v_i, we mean the variance of the $\mathbf{v}_i$ coordinate of the dataset i.e. $\{\langle \mathbf{x}, \mathbf{v}_i \rangle | \mathbf{x} \in \mathcal{D}\}$.
- After applying the PCA transformation T, we can keep only the first 2 or 3 coordinates of the data, for visualization. We know that the variance of our data is highest along these coordinates.

Principal Component Analysis (PCA)

- PCA is a linear dimensionality reduction method. For a dataset $\mathcal{D} \subset \mathbb{R}^D$, it finds an orthonormal basis $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_D$ of $\mathbb{R}^D$ such that the variance of $\mathcal{D}$ along the coordinates $\mathbf{v}_1, \ldots, \mathbf{v}_i, \ldots, \mathbf{v}_D$ is monotonically decreasing in i.
- By the variance of a dataset $\mathcal{D}$ along a basis vector v_i, we mean the variance of the $\mathbf{v}_i$ coordinate of the dataset i.e. $\{\langle \mathbf{x}, \mathbf{v}_i \rangle | \mathbf{x} \in \mathcal{D}\}$.
- After applying the PCA transformation T, we can keep only the first 2 or 3 coordinates of the data, for visualization. We know that the variance of our data is highest along these coordinates.
- Alternatively we can keep as many coordinates $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d$, (with $d < D$) that contain most (e.g. 95%) of the variance of the data and discard the rest.

Principal Component Analysis (PCA)

- PCA is a linear dimensionality reduction method. For a dataset $\mathcal{D} \subset \mathbb{R}^D$, it finds an orthonormal basis $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_D$ of $\mathbb{R}^D$ such that the variance of $\mathcal{D}$ along the coordinates $\mathbf{v}_1, \ldots, \mathbf{v}_i, \ldots, \mathbf{v}_D$ is monotonically decreasing in i.
- By the variance of a dataset $\mathcal{D}$ along a basis vector v_i, we mean the variance of the $\mathbf{v}_i$ coordinate of the dataset i.e. $\{\langle \mathbf{x}, \mathbf{v}_i \rangle | \mathbf{x} \in \mathcal{D}\}$.
- After applying the PCA transformation T, we can keep only the first 2 or 3 coordinates of the data, for visualization. We know that the variance of our data is highest along these coordinates.
- Alternatively we can keep as many coordinates $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_d$, (with $d < D$) that contain most (e.g. 95%) of the variance of the data and discard the rest.
- We can think of PCA as a linear transformation $T : \mathbb{R}^D \to \mathbb{R}^D$ that sends the standard basis vectors $\{\mathbf{e}_i\}_{i=1}^D$ to $\{\mathbf{v}_i\}_{i=1}^D$.

The math of PCA

- Remember that the *covariance* of two random variables X, Y is defined as

$$cov(X, Y) = E[(X - E(X))(Y - E(Y))] \tag{1}$$

The math of PCA

- Remember that the *covariance* of two random variables X, Y is defined as

$$cov(X, Y) = E[(X - E(X))(Y - E(Y))] \tag{1}$$

- In PCA we assume that data is centered around the origin and thus the expectation values of its coordinates are zero.

The math of PCA

- Remember that the *covariance* of two random variables X, Y is defined as

$$cov(X, Y) = E[(X - E(X))(Y - E(Y))] \tag{1}$$

- In PCA we assume that data is centered around the origin and thus the expectation values of its coordinates are zero.
- If $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^n \subset \mathbb{R}^D$ is a dataset and $\mathcal{F}_i$ is the n-dimensional vector whose components are the i'th component (feature) of datapoints, then the *covariance matrix* of $\mathcal{D}$ is the matrix C such that $C_{i,j} = \langle \mathcal{F}_i, \mathcal{F}_j \rangle / (n-1)$.

The math of PCA

- Remember that the *covariance* of two random variables X, Y is defined as

$$cov(X, Y) = E[(X - E(X))(Y - E(Y))] \tag{1}$$

- In PCA we assume that data is centered around the origin and thus the expectation values of its coordinates are zero.
- If $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^{n} \subset \mathbb{R}^D$ is a dataset and $\mathcal{F}_i$ is the n-dimensional vector whose components are the i'th component (feature) of datapoints, then the *covariance matrix* of $\mathcal{D}$ is the matrix C such that $C_{i,j} = \langle \mathcal{F}_i, \mathcal{F}_j \rangle / (n-1)$.
- In other words, if X is the $n \times D$ matrix whose rows are the $\mathbf{x}_i$ then $C = \frac{X^t X}{n-1}$.

The math of PCA

- Remember that the *covariance* of two random variables X, Y is defined as

$$cov(X, Y) = E[(X - E(X))(Y - E(Y))] \tag{1}$$

- In PCA we assume that data is centered around the origin and thus the expectation values of its coordinates are zero.
- If $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^n \subset \mathbb{R}^D$ is a dataset and $\mathcal{F}_i$ is the n-dimensional vector whose components are the i'th component (feature) of datapoints, then the *covariance matrix* of $\mathcal{D}$ is the matrix C such that $C_{i,j} = \langle \mathcal{F}_i, \mathcal{F}_j \rangle / (n-1)$.
- In other words, if X is the $n \times D$ matrix whose rows are the $\mathbf{x}_i$ then $C = \frac{X^t X}{n-1}$.
- C is symmetric and therefore it is diagonalizable i.e. $C = VLV^t$ where $L = \mathrm{diag}(\lambda_1, \lambda_2, \dots, \lambda_D)$.

The math of PCA

- Remember that the *covariance* of two random variables X, Y is defined as

$$cov(X, Y) = E[(X - E(X))(Y - E(Y))] \tag{1}$$

- In PCA we assume that data is centered around the origin and thus the expectation values of its coordinates are zero.
- If $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^n \subset \mathbb{R}^D$ is a dataset and $\mathcal{F}_i$ is the n-dimensional vector whose components are the i'th component (feature) of datapoints, then the *covariance matrix* of $\mathcal{D}$ is the matrix C such that $C_{i,j} = \langle \mathcal{F}_i, \mathcal{F}_j \rangle / (n-1)$.
- In other words, if X is the $n \times D$ matrix whose rows are the $\mathbf{x}_i$ then $C = \frac{X^t X}{n-1}$.
- C is symmetric and therefore it is diagonalizable i.e. $C = VLV^t$ where $L = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_D)$.
- We can sort the eigenvectors so that $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_D$.

The math of PCA cont.

- The columns $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_D$ of V are the eigenvectors of C and are called *principal directions* of data.

The math of PCA cont.

- The columns $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_D$ of V are the eigenvectors of C and are called *principal directions* of data.
- The projections of the data into these directions are called the *principal components* of the data

The math of PCA cont.

- The columns $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_D$ of V are the eigenvectors of C and are called *principal directions* of data.
- The projections of the data into these directions are called the *principal components* of the data i.e.

$$\mathbf{x} = \sum_{i=1}^{D} x_i \mathbf{e}_i = \sum_{i=1}^{D} x_i' \mathbf{v}_i. \tag{2}$$

The math of PCA cont.

- The columns $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_D$ of V are the eigenvectors of C and are called *principal directions* of data.
- The projections of the data into these directions are called the *principal components* of the data i.e.

$$\mathbf{x} = \sum_{i=1}^{D} x_i \mathbf{e}_i = \sum_{i=1}^{D} x_i' \mathbf{v}_i. \tag{2}$$

- The PCA trandformation is given by $T(x_1, \ldots, x_D) = (x_1', \ldots, x_D')$.

The math of PCA cont.

- The columns $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_D$ of V are the eigenvectors of C and are called *principal directions* of data.
- The projections of the data into these directions are called the *principal components* of the data i.e.

$$\mathbf{x} = \sum_{i=1}^{D} x_i \mathbf{e}_i = \sum_{i=1}^{D} x_i' \mathbf{v}_i. \tag{2}$$

- The PCA trandformation is given by $T(x_1, \ldots, x_D) = (x_1', \ldots, x_D')$.
- The principal directions can be obtained from the Singular Value Decomposition of X as well: $X = USV^t$.

The math of PCA cont.

- The columns $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_D$ of V are the eigenvectors of C and are called *principal directions* of data.
- The projections of the data into these directions are called the *principal components* of the data i.e.

$$\mathbf{x} = \sum_{i=1}^{D} x_i \mathbf{e}_i = \sum_{i=1}^{D} x_i' \mathbf{v}_i. \tag{2}$$

- The PCA trandformation is given by $T(x_1, \ldots, x_D) = (x_1', \ldots, x_D')$.
- The principal directions can be obtained from the Singular Value Decomposition of X as well: $X = USV^t$.
- We can see that $C = V \frac{S^2}{n-1} V^t$.

Inverse map for PCA

- Once we apply PCA and choose a number $d < D$ of the coordinates to keep, we can map the dimensionaly reduced data

$$\bar{\mathbf{x}} = \sum_{i=1}^{d} x_i' \mathbf{v}_i \tag{3}$$

back using the inverse T^{-1}.

Inverse map for PCA

- Once we apply PCA and choose a number $d < D$ of the coordinates to keep, we can map the dimensionaly reduced data

$$\bar{\mathbf{x}} = \sum_{i=1}^{d} x_i' \mathbf{v}_i \tag{3}$$

back using the inverse T^{-1}.

- We can use the error $\sum_i ||\mathbf{x}_i - T^{-1}\bar{\mathbf{x}}_i||^2$ to choose the value of d.

Inverse map for PCA

- Once we apply PCA and choose a number $d < D$ of the coordinates to keep, we can map the dimensionaly reduced data

$$\bar{\mathbf{x}} = \sum_{i=1}^{d} x'_i \mathbf{v}_i \tag{3}$$

back using the inverse T^{-1}.

- We can use the error $\sum_i ||\mathbf{x}_i - T^{-1}\bar{\mathbf{x}}_i||^2$ to choose the value of d.
- Note: not all dimensionality reduction methods have an inverse map!

Kernel PCA

- Kernel PCA is a nonlinear dimensionality reduction method.

Kernel PCA

- Kernel PCA is a nonlinear dimensionality reduction method.
- Remember that in the kernel method for SVM, we had a mapping $\Phi : \mathbb{R}^d \to \mathbb{R}^D$ and a kernel function $K(\mathbf{x}, \mathbf{y})$ such that $K(\mathbf{x}, \mathbf{y}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle$.

Kernel PCA

- Kernel PCA is a nonlinear dimensionality reduction method.
- Remember that in the kernel method for SVM, we had a mapping $\Phi : \mathbb{R}^d \to \mathbb{R}^D$ and a kernel function $K(\mathbf{x}, \mathbf{y})$ such that $K(\mathbf{x}, \mathbf{y}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle$.
- In Kernel PCA, we compute the variance matrix in the target space of Φ;

Kernel PCA

- Kernel PCA is a nonlinear dimensionality reduction method.
- Remember that in the kernel method for SVM, we had a mapping $\Phi : \mathbb{R}^d \to \mathbb{R}^D$ and a kernel function $K(\mathbf{x}, \mathbf{y})$ such that $K(\mathbf{x}, \mathbf{y}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle$.
- In Kernel PCA, we compute the variance matrix in the target space of Φ; however the mapping Φ is not used explicitly.

Kernel PCA

- Kernel PCA is a nonlinear dimensionality reduction method.
- Remember that in the kernel method for SVM, we had a mapping $\Phi : \mathbb{R}^d \to \mathbb{R}^D$ and a kernel function $K(\mathbf{x}, \mathbf{y})$ such that $K(\mathbf{x}, \mathbf{y}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle$.
- In Kernel PCA, we compute the variance matrix in the target space of Φ; however the mapping Φ is not used explicitly.
- In other words we have $C_{i,j} = K(\mathcal{F}_i, \mathcal{F}_j)$ (where are the feature vectors of the dataset)

Kernel PCA

- Kernel PCA is a nonlinear dimensionality reduction method.
- Remember that in the kernel method for SVM, we had a mapping $\Phi : \mathbb{R}^d \to \mathbb{R}^D$ and a kernel function $K(\mathbf{x}, \mathbf{y})$ such that $K(\mathbf{x}, \mathbf{y}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{y}) \rangle$.
- In Kernel PCA, we compute the variance matrix in the target space of Φ; however the mapping Φ is not used explicitly.
- In other words we have $C_{i,j} = K(\mathcal{F}_i, \mathcal{F}_j)$ (where are the feature vectors of the dataset) and we diagonalize this matrix.