

Lecture 6:

Heuristic methods of nonlinear data analysis

- 1. Nonlinear dimensionality reduction: heuristic approaches**
- 2. Kernel Principal Component Analysis (KPCA) - an approach**
- 3. KPCA - kernel trick**
- 4. KPCA - algorithm**
- 5. Auto-associative Neural Networks - an approach**

General points

- we know a priori or have identified an Intrinsic dimension of Data space/Training dataset
- since the beginning of the 20th century, certain method of Dimensionality reduction – **Principal Component Analysis (PCA)** - was known
- the PCA can be considered from two different points of view:
 - ✓ as ‘the best’ method for **Linear data model** – data lie on or near unknown affine linear space of smaller dimension
 - ✓ as certain projection method which finds low-dimensional features the number of which may be substantially higher than the actual Intrinsic dimension
- before the start of the 21st century, does not exist yet a model describing the nonlinear data
- but the nonlinear data had to be processed and to find their low-dimensional features, so in 80-90 years there appeared various methods of reducing the dimension for nonlinear data
- these methods are not based on any data model, so their quality could be verified on a specific set of data, but it can not be predicted in advance

Most known methods are based on a few approaches:

- to convert original nonlinear data into transformed 'linear' data using some constructed transformation
- and, then, to apply the standard PCA

Kernel PCA

- since the desired dimension reduction procedure results in several constructed mappings:
 - ✓ to look for the desired mappings in the selected class determined by a finite number of parameters
 - ✓ the values of these parameters are chosen by minimizing the chosen cost function

Neural Networks

- to choose certain cost function which reflects the desired properties of the original high-dimensional features that should be preserved in low-dimensional reduced features

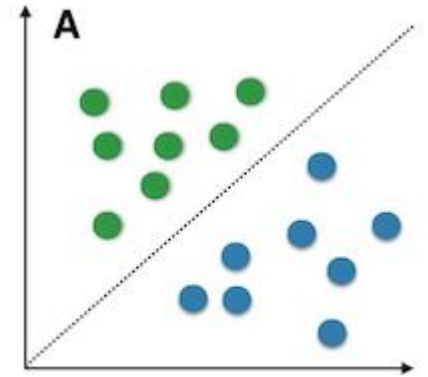
MDS

- and, then, to construct them by minimizing the chosen cost function
- to construct specific low-dimensional structures (**principal curves**, **principal surfaces**), etc.

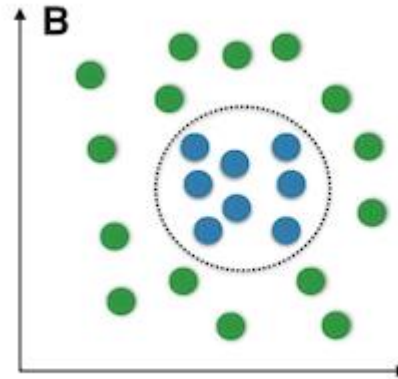
Kernel Principal Component Analysis (KPCA)

is based on applying the standard PCA to the transformed data – the problem is **how to transform the data**

Linear Discriminant Analysis techniques allows sometimes to separate the data



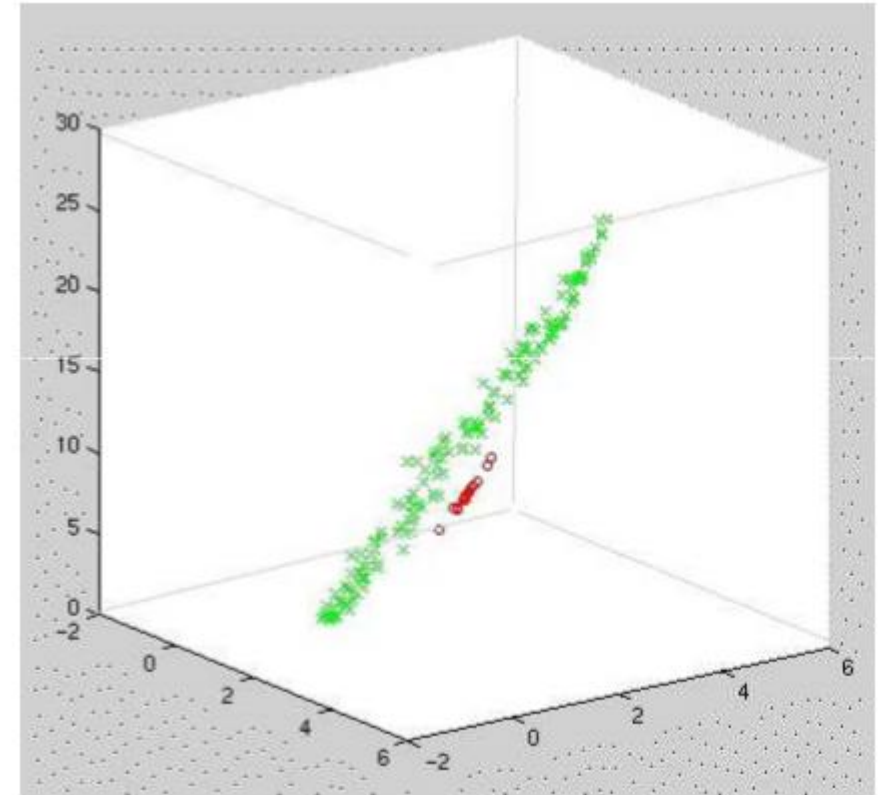
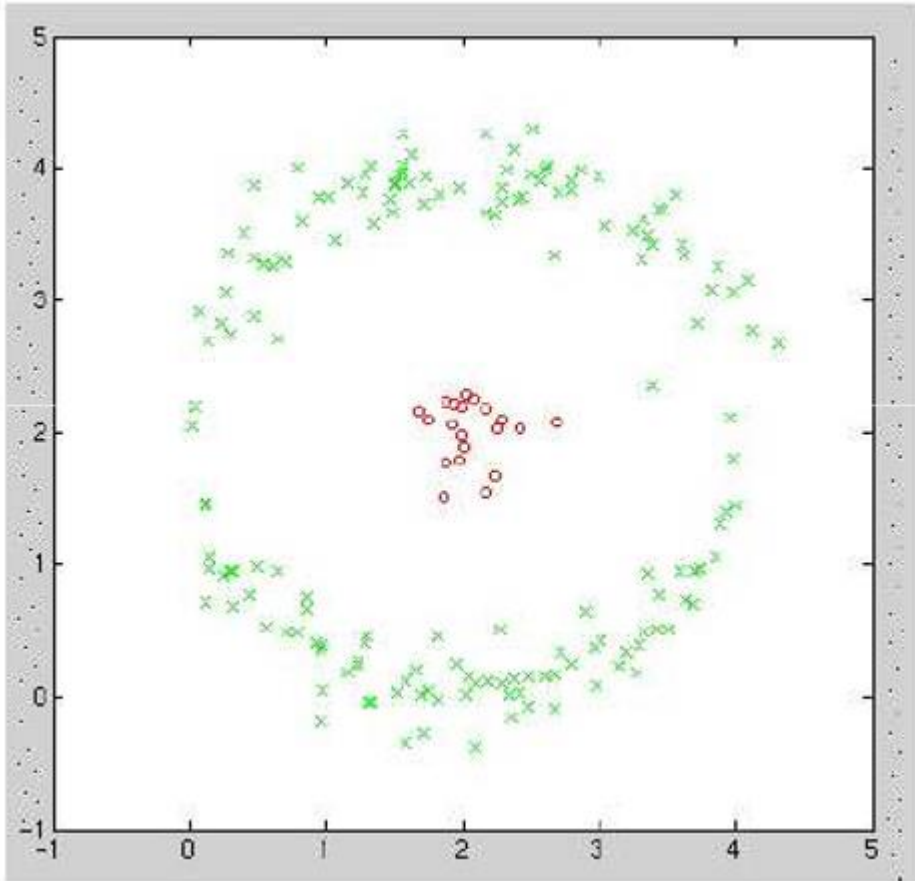
but not always



No linear method here will help

$$(x_1, x_2) \rightarrow (x_1, x_2, x_1^2 + x_2^2)$$

- transformation of the original data to
three-dimensional data



Transformed data can be separated

High dimensionality can be good

The basic idea: transforming original 'linearly inseparable' data to higher dimensional data where they become 'linearly separable'

The idea that originally arose for solving the data separation problem (classification) was useful for other problems, among which the dimension reduction

Let $\mathbf{x} \in \mathbb{R}^p \rightarrow \Phi(\mathbf{x})$ be desired transform from \mathbb{R}^p to the image space $\Phi = \Phi(\mathbb{R}^p)$, $d = \text{Dim}(\Phi)$

Original dataset $\mathbf{X}_{(n)} = \{\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_n\} \rightarrow$ transformed dataset $\Phi_{(n)} = \{\Phi_i = \Phi(\mathbf{X}_i), i = 1, 2, \dots, n\}$

Applying the PCA to the transformed dataset $\Phi_{(n)}$:

- Mean vector: $\bar{\Phi} = \frac{1}{n} \sum_{i=1}^n \Phi_i$
- Centered transformed dataset: $\{\bar{\Phi}_i = \Phi_i - \bar{\Phi}, i = 1, 2, \dots, n\}$
- Sample covariance $d \times d$ matrix: $\Sigma_{\Phi} = \frac{1}{n} \sum_{i=1}^n \bar{\Phi}_i \times \bar{\Phi}_i^T$

PCA solution:

- calculating a number (say, q) of orthonormal eigenvectors $e_1, e_2, \dots, e_q \in \Phi \subset R^d$ of the matrix Σ_Φ :

$$\Sigma_\Phi e_k = \lambda_k \times e_k, \quad k = 1, 2, \dots, q$$

corresponding to q largest eigenvalues $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_q$ of this matrix

E - $d \times q$ matrix with columns $e_1, e_2, \dots, e_q \in \Phi$

- q -dimensional PCA-representation of arbitrary vector $\Phi \in \Phi$: $y = E^T \times (\Phi - \bar{\Phi}) \in R^q$
- q -dimensional PCA-representation of original vector $X \in R^p$: $y = E^T \times \overline{\Phi(X)} \in R^q$
 $\overline{\Phi(X)} = \Phi(X) - \bar{\Phi}$

\mathbf{v} - arbitrary eigenvector of the matrix Σ_{Φ} : $\Sigma_{\Phi} \mathbf{v} = \lambda \times \mathbf{v}$ corresponding to eigenvalue $\lambda \neq 0$

$$\left(\frac{1}{n} \sum_{j=1}^n \bar{\Phi}_j \times \bar{\Phi}_j^T \right) \times \mathbf{v} = \lambda \times \mathbf{v}$$

$$\frac{1}{n} \sum_{j=1}^n \bar{\Phi}_j \times (\bar{\Phi}_j^T \times \mathbf{v}) = \lambda \times \mathbf{v} \quad \bar{\Phi}_j^T \times \mathbf{v} - \text{a scalar}$$

$$\frac{1}{n} \sum_{j=1}^n (\bar{\Phi}_j^T \times \mathbf{v}) \times \bar{\Phi}_j = \lambda \times \mathbf{v}$$

$$\mathbf{v} = \sum_{j=1}^n \alpha_j \times \bar{\Phi}_j \quad \alpha_j = \frac{1}{n\lambda} (\bar{\Phi}_j^T \times \mathbf{v})$$

This means that arbitrary eigenvector with eigenvalue $\lambda \neq 0$ lies in the linear space spanned by $\{\bar{\Phi}_j\}$

Thus, finding the eigenvectors $\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_q$ is equivalent to finding the coefficients $\{\alpha_j\}$

$$\begin{aligned} \Sigma_{\Phi} v &= \lambda \times v \\ v &= \sum_{j=1}^n \alpha_j \times \bar{\Phi}_j \end{aligned} \quad \Rightarrow \quad \left(\sum_{i=1}^n \bar{\Phi}_i \times \bar{\Phi}_i^T \right) \times \left(\sum_{j=1}^n \alpha_j \times \bar{\Phi}_j \right) = n\lambda \times \sum_{j=1}^n \alpha_j \times \bar{\Phi}_j$$

$$\sum_{i=1}^n \sum_{j=1}^n [\bar{\Phi}_i \times \bar{\Phi}_i^T \times \alpha_j \times \bar{\Phi}_j] = n\lambda \times \sum_{j=1}^n \alpha_j \times \bar{\Phi}_j$$

$$\sum_{i=1}^n \sum_{j=1}^n [\alpha_j \times \bar{\Phi}_i \times (\bar{\Phi}_i^T \times \bar{\Phi}_j)] = n\lambda \times \sum_{j=1}^n \alpha_j \times \bar{\Phi}_j$$

$$\bar{\Phi}_i^T \times \bar{\Phi}_j = (\bar{\Phi}_i, \bar{\Phi}_j) = (\bar{\Phi}(X_i), \bar{\Phi}(X_j)) - \text{inner-product in the image space } \Phi = \Phi(\mathbb{R}^p), d = \text{Dim}(\Phi)$$

$$K(X_i, X_j) = (\bar{\Phi}(X_i), \bar{\Phi}(X_j)) - \text{kernel functions}$$

$$\sum_{i=1}^n \sum_{j=1}^n [\alpha_j \bar{\Phi}(X_i) \times K(X_i, X_j)] = n\lambda \times \sum_{j=1}^n \alpha_j \times \bar{\Phi}(X_j)$$

$$v = \sum_{j=1}^n \alpha_j \times \bar{\Phi}_j$$

$$\longrightarrow e_k = \sum_{j=1}^n \alpha_{kj} \times \bar{\Phi}_j$$

$$\Sigma_{\Phi} e_k = \lambda_k \times e_k$$

$$\sum_{i=1}^n \sum_{j=1}^n [\alpha_j \bar{\Phi}(X_i) \times K(X_i, X_j)] = n\lambda \times \sum_{j=1}^n \alpha_j \times \bar{\Phi}(X_j) \longrightarrow$$

$$\frac{1}{n} \sum_{i=1}^n \sum_{j=1}^n [\alpha_{kj} \bar{\Phi}(X_i) \times K(X_i, X_j)] = \lambda_k \times \sum_{j=1}^n \alpha_{kj} \times \bar{\Phi}(X_j)$$

$$\frac{1}{n} \sum_{i=1}^n \bar{\Phi}(X_i) \times \left(\sum_{j=1}^n [\alpha_{kj} K(X_i, X_j)] \right) = \lambda_k \times \sum_{j=1}^n \alpha_{kj} \times \bar{\Phi}(X_j)$$

For some t , multiply this by $\bar{\Phi}^T(X_t)$ from the left:

$$\frac{1}{n} \sum_{i=1}^n \bar{\Phi}^T(X_t) \times \bar{\Phi}(X_i) \times \left(\sum_{j=1}^n [\alpha_{kj} K(X_i, X_j)] \right) = \lambda_k \times \sum_{j=1}^n \alpha_{kj} \times \bar{\Phi}^T(X_t) \times \bar{\Phi}(X_j)$$

$$\frac{1}{n} \sum_{i=1}^n \left(\bar{\Phi}^T(X_t) \times \bar{\Phi}(X_i) \right) \times \left(\sum_{j=1}^n [\alpha_{kj} K(X_i, X_j)] \right) = \lambda_k \times \sum_{j=1}^n \alpha_{kj} \times \left(\bar{\Phi}^T(X_t) \times \bar{\Phi}(X_j) \right)$$

$$\frac{1}{n} \sum_{i=1}^n K(X_t, X_i) \times \left(\sum_{j=1}^n [\alpha_{kj} K(X_i, X_j)] \right) = \lambda_k \times \sum_{j=1}^n \alpha_{kj} \times K(X_t, X_j) \quad t = 1, 2, \dots, n$$

Denote: $\alpha_k = \begin{pmatrix} \alpha_{k1} \\ \dots \\ \alpha_{kn} \end{pmatrix} \in \mathbb{R}^n$, $K = \|K(X_i, X_j)\|$ - $n \times n$ **kernel matrix**

$$K^2 \times \alpha_k = n \lambda_k \times K \times \alpha_k \quad \alpha_k - \text{eigenvector of the matrix } K^2$$

Let u – arbitrary eigenvector of the matrix K : $Ku = \beta \times u$ \longrightarrow u - eigenvector of the matrix K^2

$$K^2 u = K \times (Ku) = K \times (\beta \times u) = \beta \times Ku = \beta \times (\beta \times u) = \beta^2 \times u$$

u - eigenvector of the matrix K \longleftarrow u - eigenvector of the matrix K^2 (for nonzero eigenvalues)

α_k - eigenvector of the matrix \mathbf{K} \longleftrightarrow α_k - eigenvector of the matrix \mathbf{K} (for nonzero eigenvalues)

$$\mathbf{K} \times \alpha_k = \gamma_k \times \alpha_k$$

\mathbf{e}_k - arbitrary eigenvector of the matrix Σ_Φ

\mathbf{e}_k - eigenvector of the matrix \mathbf{K}

$\{\mathbf{e}_k\}$ – eigenvectors of the matrix \mathbf{K}

\mathbf{e}_k – unit vector: $(\mathbf{e}_k, \mathbf{e}_k) = 1$

$$\mathbf{e}_k = \sum_{j=1}^n \alpha_{kj} \times \bar{\Phi}_j$$

$$1 = (\mathbf{e}_k, \mathbf{e}_k) = \left(\sum_{j=1}^n \alpha_{kj} \times \bar{\Phi}_j, \sum_{j=1}^n \alpha_{kj} \times \bar{\Phi}_j \right) = \sum_{i,j=1}^n [\alpha_{ki} \times \bar{\Phi}_i^T \times \bar{\Phi}_j \times \alpha_{kj}]$$

$$= \sum_{i,j=1}^n [\alpha_{ki} \times K(X_i, X_j) \times \alpha_{kj}] = \alpha_k^T \times \mathbf{K} \times \alpha_k = \gamma_k \times (\alpha_k, \alpha_k)$$

$$\alpha_k = \begin{pmatrix} \alpha_{k1} \\ \dots \\ \alpha_{kn} \end{pmatrix}$$

Put: $\mathbf{e}_k = \gamma_k^{1/2} \times \alpha_k, k = 1, 2, \dots, q$

$$(\mathbf{e}_i, \mathbf{e}_j) = \delta_{ij}, \quad i, j = 1, 2, \dots, q$$

y - q -dimensional PCA-representation of original vector X : $y = E^T \times \overline{\Phi(X)}$, $\overline{\Phi(X)} = \Phi(X) - \bar{\Phi}$

E - $d \times q$ matrix with columns $\{e_k = \sum_{j=1}^n \alpha_{kj} \times \bar{\Phi}_j\}$

$$y = \begin{pmatrix} y_1 \\ \dots \\ y_q \end{pmatrix}: \quad y_k = e_k^T \times \overline{\Phi(X)} = \gamma_k^{1/2} \times \sum_{j=1}^n \alpha_{kj} \times \bar{\Phi}_j^T \times \overline{\Phi(X)}$$

$$= \gamma_k^{1/2} \times \sum_{j=1}^n \alpha_{kj} \times (\bar{\Phi}(X_j), \overline{\Phi(X)}) = \gamma_k^{1/2} \times \sum_{j=1}^n \alpha_{kj} \times K(X_j, X)$$

$$\mathbf{K} \times \alpha_k = \gamma_k \times \alpha_k, \quad k = 1, 2, \dots, q$$

$\mathbf{K} = \|\mathbf{K}(X_i, X_j)\|$ - $n \times n$ kernel matrix

$$y = \begin{pmatrix} y_1 \\ \dots \\ y_q \end{pmatrix}: \quad y_k = \gamma_k^{1/2} \times \sum_{j=1}^n \alpha_{kj} \times K(X_j, X)$$

These formulas don't require a computing of the transformed vectors $\Phi(X)$ in explicit form: we are only needed to have their inner products in Image space Φ called **kernel functions**

Computing these inner products without actually performing the transformation $\Phi(X)$ - **kernel trick**

We computed the inner products for centered the transformed vectors $\Phi(X)$, but we cannot explicitly compute the mean of the transformed vectors $\{\Phi(X_i)\}$

but centering is possible also without actually computing the transformed vectors $\{\Phi(X_i)\}$ by certain modification in Kernel PCA – a centering of the kernel matrix $\mathbf{K} = \|\mathbf{K}(X_i, X_j)\|$

Centering the kernel matrix **K**

H = $I_n - \frac{1}{n} \mathbf{1} \times \mathbf{1}^T$ - $n \times n$ centering matrix, **1** – n -dimensional vector of all 1's:

$$\mathbf{H} \times \begin{pmatrix} a_1 \\ a_2 \\ \dots \\ a_n \end{pmatrix} = \begin{pmatrix} a_1 - \bar{a} \\ a_2 - \bar{a} \\ \dots \\ a_n - \bar{a} \end{pmatrix}, \quad \bar{a} = \frac{1}{n} \sum_{i=1}^n a_i$$

Centered kernel matrix $\bar{\mathbf{K}} = \mathbf{H} \times \mathbf{K} \times \mathbf{H}$

$$\bar{\mathbf{K}} = \|\bar{\mathbf{K}}(X_i, X_j)\|: \bar{\mathbf{K}}(X_i, X_j) = K(X_i, X_j) - \frac{1}{n} \sum_{t=1}^n K(X_i, X_t) - \frac{1}{n} \sum_{s=1}^n K(X_j, X_s) + \frac{1}{n^2} \sum_{t,s} K(X_t, X_s)$$

Kernel PCA: main steps

Given: dataset $\mathbf{X}_{(n)} = \{\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_n\}$, kernel function $K(\mathbf{X}, \mathbf{X}')$, dimension $q < p$

Step 1: computing $n \times n$ kernel matrix $\mathbf{K} = \|K(\mathbf{X}_i, \mathbf{X}_j)\|$

Step 2: centering: $\mathbf{K} = \|K(\mathbf{X}_i, \mathbf{X}_j)\| \rightarrow \bar{\mathbf{K}} = \|\bar{K}(\mathbf{X}_i, \mathbf{X}_j)\|$

Step 3: solving an eigenvector problem: computing the eigenvectors: $\bar{\mathbf{K}} \times \alpha_k = \gamma_k \times \alpha_k$, $k = 1, 2, \dots, q$,
corresponding to q largest eigenvalues $\gamma_1 \geq \gamma_2 \geq \dots \geq \gamma_q$

Step 4: for arbitrary p -dimensional vector \mathbf{X} (new or from the dataset), its q -dimensional representation \mathbf{y} is computed:

$$\mathbf{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_q \end{pmatrix}: \quad y_k = \gamma_k^{1/2} \times \sum_{j=1}^n \alpha_{kj} \times K(\mathbf{X}_j, \mathbf{X})$$

Where we take the kernel function $K(X, X')$?

We have called function K the **kernel function** – why?

The function $K(X, X')$ has been defined as inner products $(\bar{\Phi}(X), \bar{\Phi}(X'))$ in the Image space Φ

By definition, the function $K(X, X')$ has the properties:

- it is **symmetrical** one: $K(X, X') = K(X', X)$
- it is **positive semi-defined**: for any integer number N , any points X_1, X_2, \dots, X_N , and any real numbers c_1, c_2, \dots, c_N , we have:

$$\begin{aligned}\sum_{i,j=1}^N K(X_i, X_j) c_i c_j &= \sum_{i,j=1}^N (\bar{\Phi}(X_i), \bar{\Phi}(X_j)) c_i c_j = (\sum_{i=1}^N \bar{\Phi}(X_i) c_i) \times (\sum_{j=1}^N \bar{\Phi}(X_j) c_j) \\ &= (\sum_{i=1}^N \bar{\Phi}(X_i) c_i)^2 \geq 0\end{aligned}$$

A function $K(X, X')$ that is **symmetrical** and **positive semi-defined** is called **kernel function**

Math: Hilbert space $H = \{f\}$ is linear vector space in which an inner product (f, f') is defined

Example 1: H consists of D -dimensional vectors $\{f\}$ with standard inner product (f, f')

Example 2: H consists of square integrable functions $\{f(x)\}$ defined on $X \subset R^D$: $\int f^2(x)dx < \infty$.

Inner product $(f, f') = \int f(x)f'(x)dx$

Math: Let $X \subset R^p$ and $K(X, X')$ is arbitrary symmetrical and positive semi-defined function of $X, X' \in X$.

There exist a unique Hilbert space H of functions $f(X)$ defined on X for which K is a **reproducing kernel**

there exist a mapping $X \rightarrow \Phi(X) \in H$ such that $K(X, X') = (\Phi(X), \Phi(X'))$

Mapping $x \in R^p \rightarrow \Phi(x)$ is straightening transformation

Kernel PCA meaning: what transformation does the selected kernel function $K(X, X')$ correspond to?

Formal definition

$\bar{K} = \|\bar{K}(X_i, X_j)\| = P \times \Lambda \times P^T$ - SVD, P - $n \times n$ orthogonal matrix, $\Lambda = \text{Diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$

$U = \Lambda^{1/2} \times P^T$ - $n \times n$ matrix

$U = (\mathbf{U}_1 \quad \dots \quad \mathbf{U}_n)$ consists of n n -dimensional columns \mathbf{U}_k , $k = 1, 2, \dots, n$

$\bar{\Phi}(X_k) = \mathbf{U}_k$, $k = 1, 2, \dots, n$

$\bar{K} = U^T \times U = \|\bar{K}(X_i, X_j)\| = \|(\bar{\Phi}(X_i), \bar{\Phi}(X_j))\|$

Informal definition

$K(X_i, X_j)$ - chosen kernel, $\mathbf{V}_k = \begin{pmatrix} K(X_k, X_1) \\ \cdots \\ K(X_k, X_n) \end{pmatrix}$ - n -dimensional vectors, $k = 1, 2, \dots, n$

$$(V_i, V_j) = \sum_{s=1}^n K(X_s, X_i) \times K(X_s, X_j), \quad i, j = 1, 2, \dots, n$$

$$\|(V_i, V_j)\| = K^2$$

The mapping $X \rightarrow V(X) = \begin{pmatrix} K(X, X_1) \\ \cdots \\ K(X, X_n) \end{pmatrix}$ corresponds to the kernel K^2 that has similar spectral properties as the kernel K

Where we take the kernel function $K(X, X')$?

We can take **arbitrary** function $K(X, X')$ that is **symmetrical** and **positive semi-defined** one and use this function as **kernel function** in the **Kernel PCA**

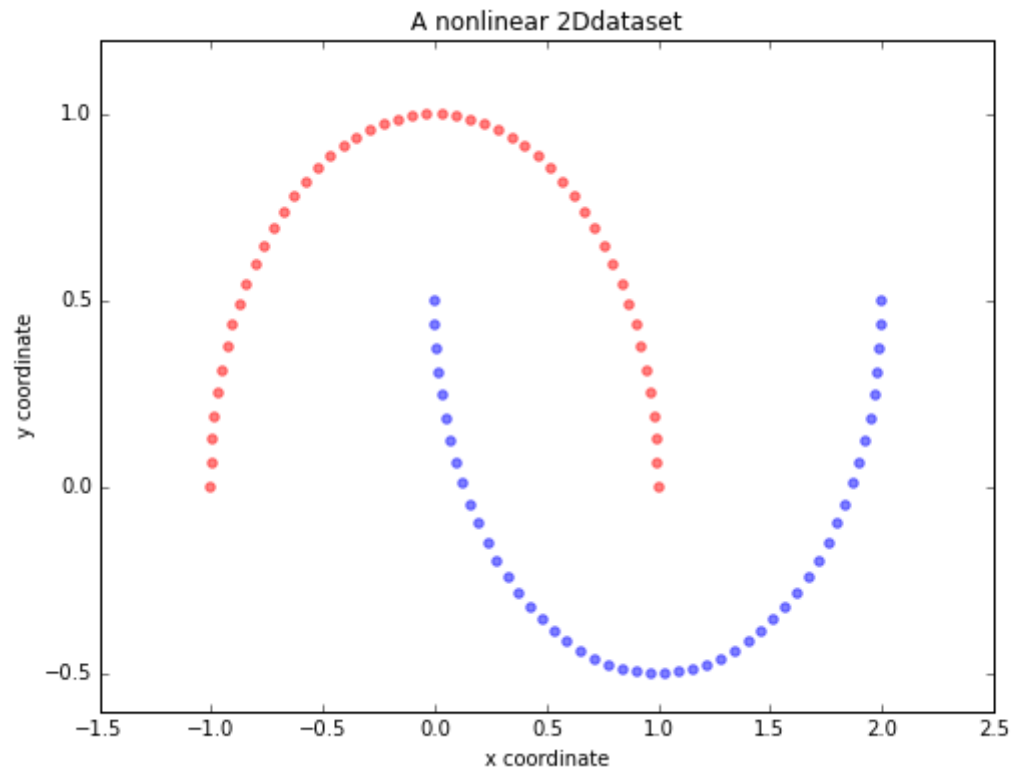
Popular kernel functions

Gaussian kernel function: $K(X, X') = \exp(-a\|X - X'\|^2)$, $a > 0$

Polynomial kernel function: $K(X, X') = (1 + (X, X'))^a$ a - integer

Hyperbolic kernel function: $K(X, X') = \tanh((X, X') + a)$ a - arbitrary

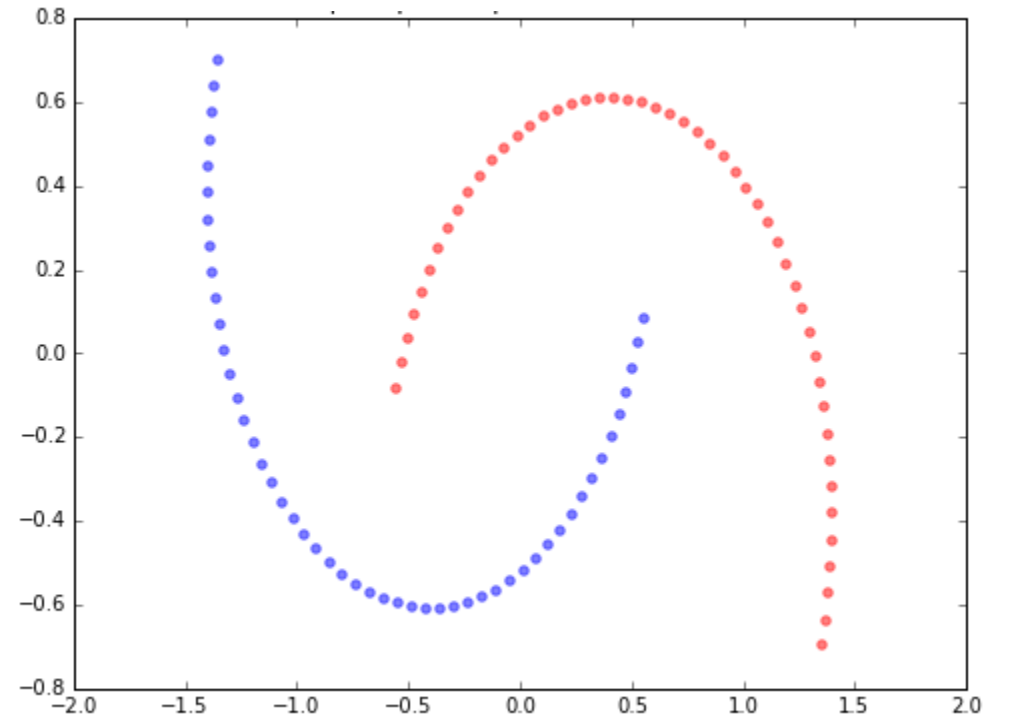
Example 1



PCA2



Linear PCA



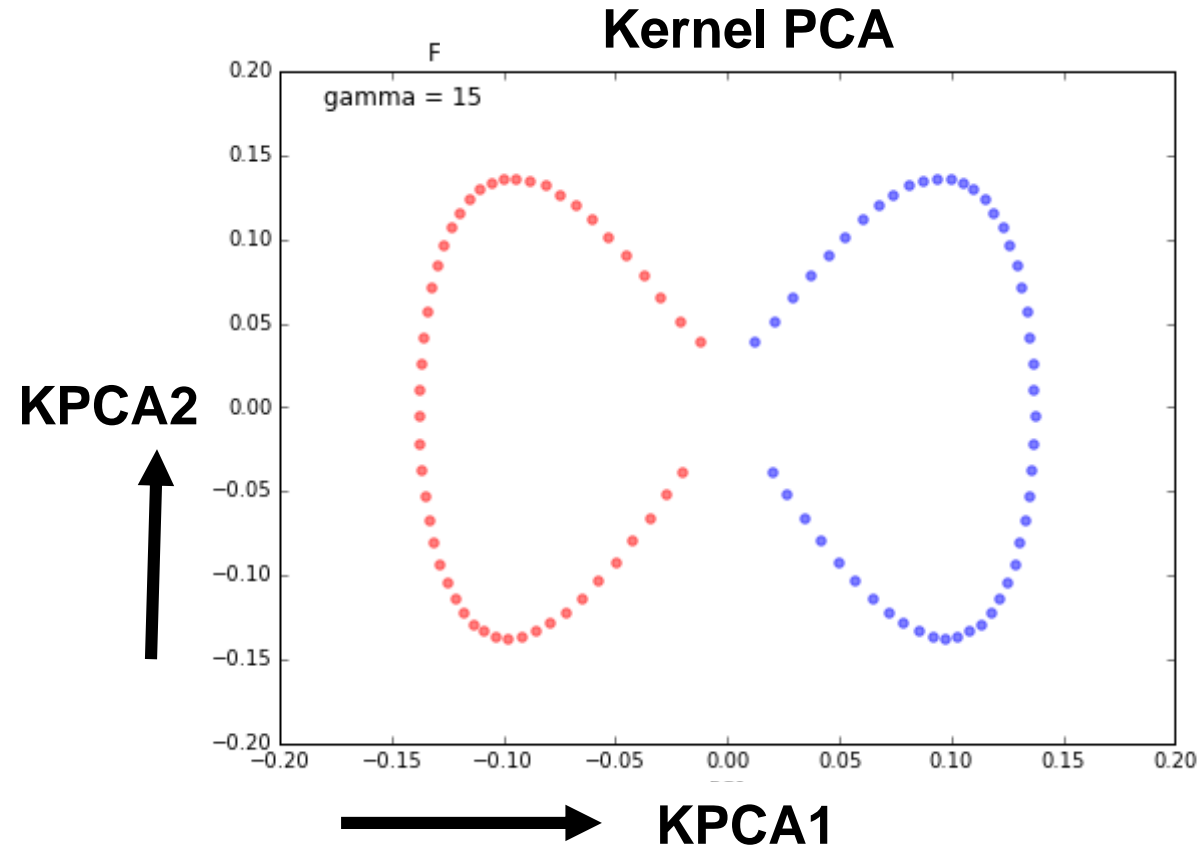
PCA1



Projection on the first PCA-eigenvector:



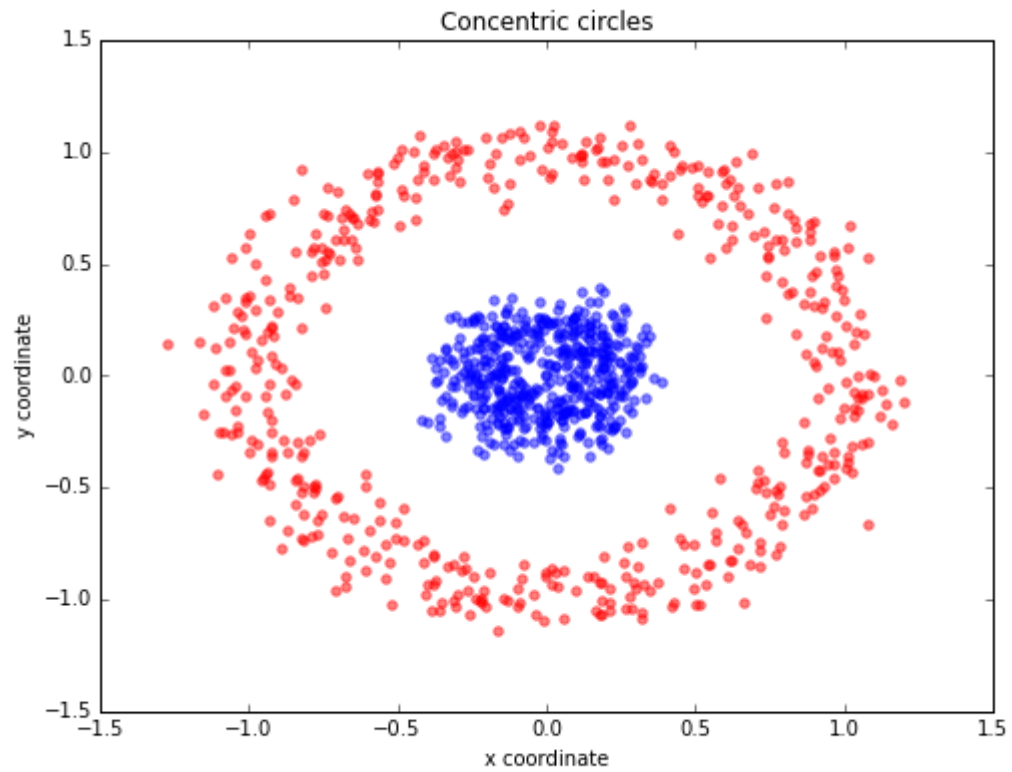
Kernel PCA with Gaussian kernel function: $K(X, X') = \exp(-a\|X - X'\|^2)$, $a > 0$



Projection on the first KPCA-eigenvector:



Example 2



Projection on the first PCA-eigenvector:



Projection on the first KPCA-eigenvector:



Neural Networks approach to dimensionality reduction

called also auto-associative networks, autoencoders, bottlenecks, or p-q-p networks

- $\mathbf{X}_{(n)} = \{X_1, X_2, \dots, X_n\}$ – p-dimensional original features
- $\{y_1, y_2, \dots, y_n\}$ – desired reduced q-dimensional reduced features

Features $\{y_1, y_2, \dots, y_n\}$ are the results of applying of **Embedding mapping**

$$h: X \in \mathbb{R}^p \rightarrow y = h(X) \in \mathbb{R}^q$$

to original p-dimensional features $\{X_1, X_2, \dots, X_n\}$

Low-dimensional reduced features must **preserve as much as possible available information** contained in high-dimensional original features: the possibility for **accurate recovery** of the original vectors from their low-dimensional features

An information preserving - the possibility for **recovery** of original vectors from reduced low-dimensional features **with small recovering error**

- an existence of a recovering mapping $g: y = h(X) \in \mathbb{R}^q \rightarrow g(y) \in \mathbb{R}^p$ from q -dimensional Reduced feature space to p -dimensional Original feature space

- a recovered value \hat{X} :

$$X \rightarrow y = h(X) \rightarrow \hat{X} = g(y) = g(h(X))$$

- a recovering error $\Delta(X) = \|\hat{X} - X\|$

Recovering error should be small

Dimensionality reduction: to find an **Embedding mapping** $h: X \in \mathbb{R}^p \rightarrow y = h(X) \in \mathbb{R}^q$ and recovering mapping $g: y = h(X) \in \mathbb{R}^q \rightarrow g(y) \in \mathbb{R}^p$ that together provide proximity $g(h(X)) \approx X$

We will look for the unknown mappings h and g in selected parametrical classes of function:

$$h(X) \in \mathbf{H} = \{u(X, \mu), \mu \in \mathbf{M}\}$$

u - known (chosen) function depending on unknown parameter $\mu \in \mathbf{M}$

$$g(y) \in \mathbf{G} = \{v(y, \tau), \tau \in \mathbf{T}\}$$

v - known (chosen) function depending on unknown parameter $\tau \in \mathbf{T}$

Recovering error: $\Delta(X) = \|g(h(X)) - X\| = \|v(u(X, \square), \square) - X\| = \Delta(X, \theta)$

$$\theta = (\mu, \tau) \in \Theta = \mathbf{M} \times \mathbf{T}$$

Averaged recovering error: $\Delta(\mathbf{X}_{(n)}, \theta) = \left(\frac{1}{n} \sum_{i=1}^n |\hat{X}_i - X_i|^2 \right)^{1/2} = \left(\frac{1}{n} \sum_{i=1}^n \Delta^2(X, \theta) \right)^{1/2}$

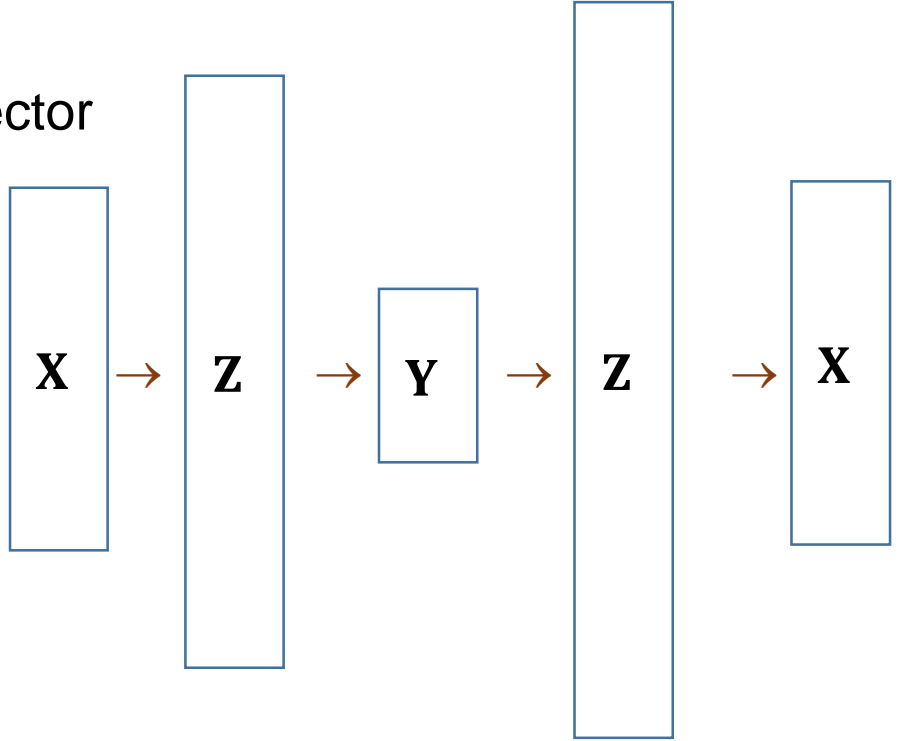
Dimensionality reduction: to find parameter $\theta \in \Theta$ that minimizes $\Delta(\mathbf{X}_{(n)}, \theta)$

- how to choose the classes **H** and **G** **Neural Networks approach:** concrete classes **H** and **G**
- how to optimize the cost function $\Delta(\mathbf{X}_{(n)}, \theta)$

Neural Networks: used classes

Neural network architecture: $X = \begin{pmatrix} x_1 \\ \dots \\ x_p \end{pmatrix} \in \mathbb{R}^p$ - input vector

Linear mapping: $X \in \mathbb{R}^p \rightarrow Z = a + W \times X = \begin{pmatrix} z_1 \\ \dots \\ z_r \end{pmatrix} \in \mathbb{R}^r$

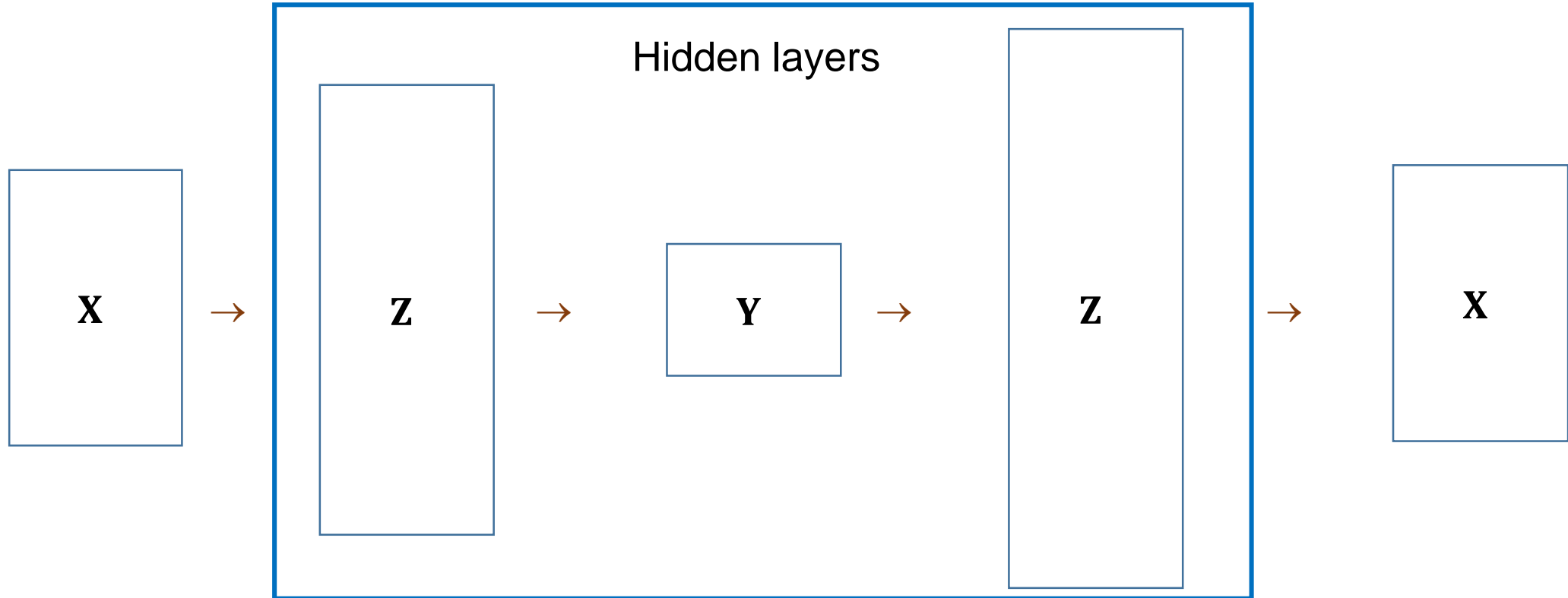


```
graph LR; X1[X] --> Z1[Z]; Z1 --> Y[Y]; Y --> Z2[Z]; Z2 --> X2[X]
```

Nonlinear mapping: $Z \in \mathbb{R}^r \rightarrow Y = f(Z) = \begin{pmatrix} y_1 \\ \dots \\ y_q \end{pmatrix} \in \mathbb{R}^q$

Nonlinear mapping: $Y \in \mathbb{R}^q \rightarrow S = Q(Y) = \begin{pmatrix} s_1 \\ \dots \\ s_t \end{pmatrix} \in \mathbb{R}^t$

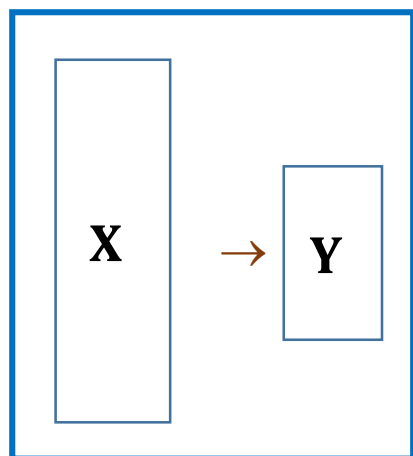
Linear mapping: $S \in \mathbb{R}^t \rightarrow X' = c + V \times S \in \mathbb{R}^p$



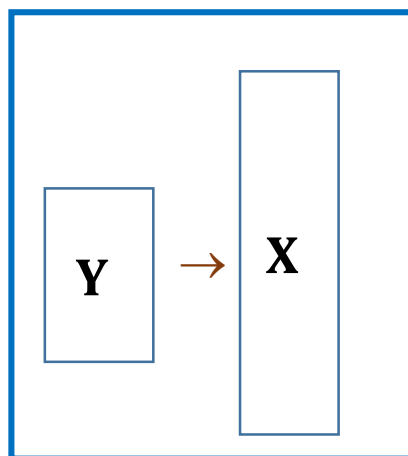
Input layer

bottleneck layer

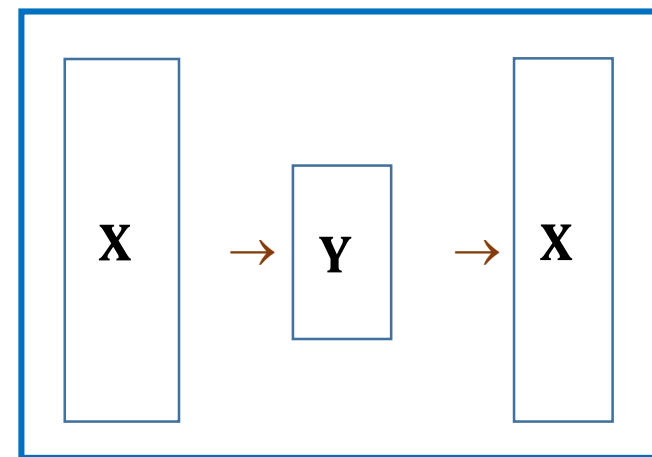
output layer



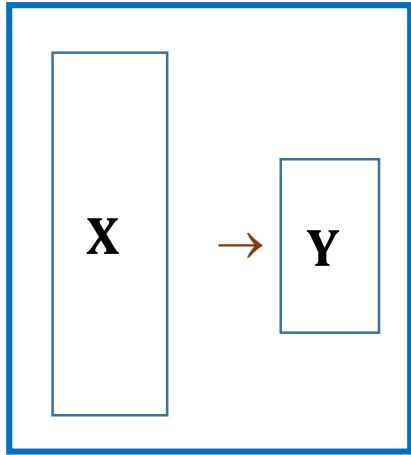
Encoder



Decoder



Autoencoder



Linear mapping: $X \in \mathbb{R}^p \rightarrow Z = a + W \times X \in \mathbb{R}^r$

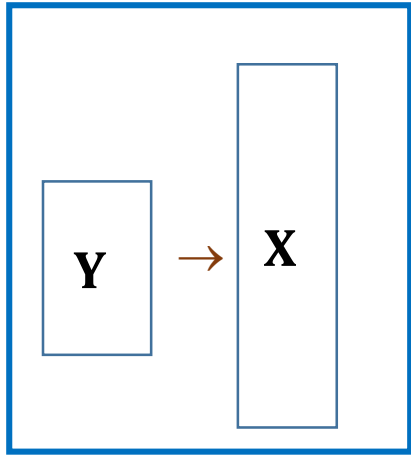
$$z_k = a_k + \sum_{i=1}^p w_{ki} \times X_i \quad k = 1, 2, \dots, r$$

Nonlinear mapping: $Z \in \mathbb{R}^r \rightarrow Y = f(Z) \in \mathbb{R}^q$

$$y_m = f_m(Z) = \sum_{j=1}^{N_1} \beta_{mj} \times \sigma(\gamma_j^T \times Z + b_j) \quad m = 1, 2, \dots, q$$

$\sigma(t)$ – sigmoid function, for example $\sigma(t) = (1 + \exp(-t))^{-1}$

$Y = h(X) = u(X, \mu)$ - depends on parameter $\mu = (a \in \mathbb{R}^r, W, N_1, \{\beta_{mj}, \gamma_j \in \mathbb{R}^r, b_j\})$



Nonlinear mapping: $Y \in R^q \rightarrow S = Q(Y) \in R^t$

$$S_l = T_l(Z) = \sum_{j=1}^{N_2} \varphi_{lj} \times \sigma(\omega_j^T \times Z + d_j) \quad l = 1, 2, \dots, t$$

$\sigma(t)$ – sigmoid function

Linear mapping: $S \in R^s \rightarrow X' = c + V \times S \in R^p$

$X = g(y) = v(y, \tau)$ - depends on parameter $\tau = (c \in R^p, V, N_2, \{\varphi_{lj}, \omega_j \in R^t, d_j\})$

How to optimize the cost function $\Delta(\mathbf{X}_{(n)}, \theta)$

- Back-propagation technique is used usually
- parameter θ has high dimension: certain optimization strategy is used

Other heuristic approaches:

- to choose certain cost function and, then, minimize the chosen cost function
- to construct specific low-dimensional structures (**principal curves**, **principal surfaces**), etc.

