Kernel trick

(& Kernel PCA)

Remember linear regression:

⇝ linear functions

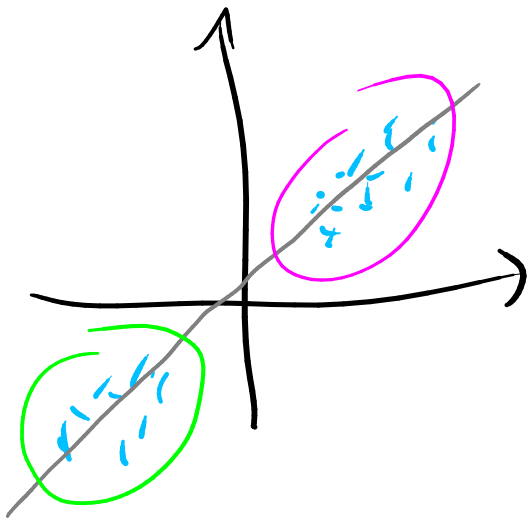
With „basis expansion map" / „feature map" could do polynomial regression:

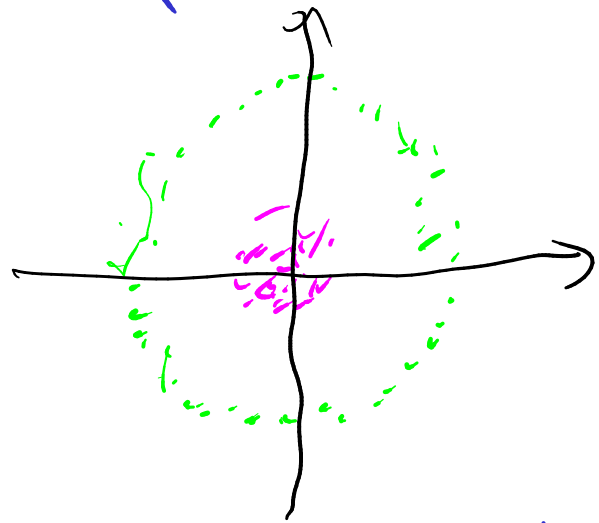
$(x_1, y_1), \ldots, (x_n, y_n)$

$\downarrow$

$(1, x_1, x_1^2, x_1^3, \ldots, x_1^d, y)$ ⇝ $(1, x_n, \ldots, x_n^d, y)$

How to use such an idea for PCA:

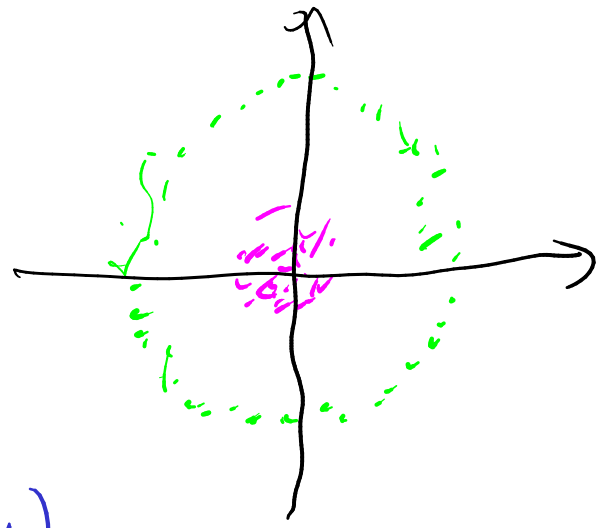




Data can't be separated in any lower dim. linear subspace.

Such examples do occur in natural data:

e.g. Lotka-Volterra equations



$\subset \mathbb{R}^2$

(x,y)

$\mathbb{R}^3$

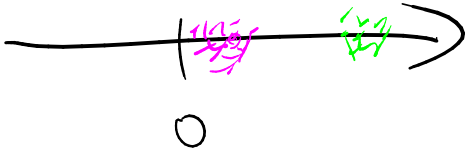
$(x, y, a\cdot(x^2+y^2))$

last coordinate

PCA

$\mathbb{R}$

(x^2+y^2)



0

The kernel trick

- see also Schölkopf, Mika, Burgers, Knirsch, Müller, Rätsch, Smola, “Input Space vs. Feature Space in Kernel-Based Methods”, 1999

Classification problem

- given patterns $x_1, \ldots, x_n \in \mathcal{X}$ (e.g. $\mathcal{X} = \mathbb{R}^M$)
- and class labels $y_1, \ldots, y_n \in \mathcal{Y} = \{+1, -1\}$
- find a decision function $f : \mathcal{X} \to \mathcal{Y}$ that predicts the label y_* of a new pattern x_*.

Notions:

- Linear case: the true decision function is linear.
- Nonlinear case: the true decision function is nonlinear.

From input space to feature space

Nonlinear example:

- e.g. class 1 close at the origin, class -1 further away.

Idea

- map the data to new features,
- e.g.

input space $\qquad\qquad\qquad\qquad$ feature space

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \longmapsto \phi(x) = \begin{bmatrix} x_1 \\ x_2 \\ x_1^2 + x_2^2 \end{bmatrix}$$

- or

input space $\qquad\qquad\qquad\qquad$ feature space

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \longmapsto \phi(x) = \begin{bmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}x_1 x_2 \end{bmatrix}$$

- both work (and there are many more)

Notational warning:

- on this and the next slides we use x_i to denote coordinates of x
- in most of the remaining slides x_i is again the ith data point

Feature maps and inner products

Feature map (another example):

$$\phi: \underbrace{\mathbb{R}^2}_{\text{input space}} \longrightarrow \underbrace{\mathbb{R}^3}_{\text{feature space}}$$

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \longmapsto \begin{bmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}\, x_1 x_2 \end{bmatrix}.$$

Linear SVM looks at the data through dot products:

$$\max_\alpha \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle$$

$$\text{s.t.} \quad C \geq \alpha \geq 0 \text{ and } \sum_{i=1}^{n} \alpha_i y_i = 0$$

Dot product is essential to compare data points:

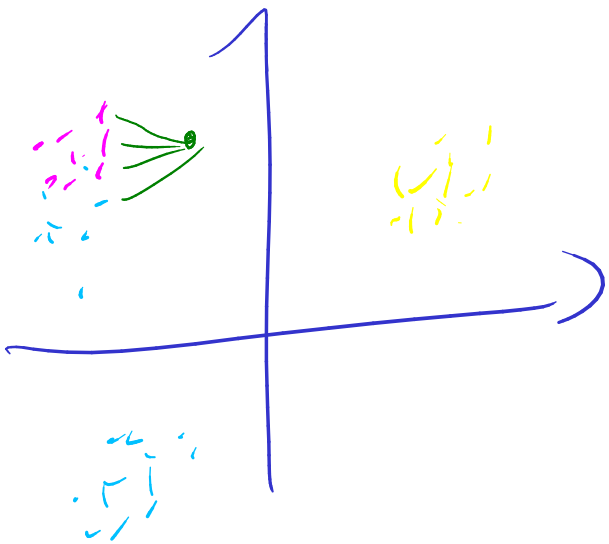
e.g. distance can be written with dot products

$$\|x - x'\|^2 = \langle x, x \rangle + \langle x', x' \rangle - 2\langle x, x' \rangle$$

Another example: K-nearest neighbors. (KNN)

a supervised clustering algorithm.

$$P(X=c) = \frac{\text{nr. of neighbors in class } c}{K}$$



Only need to measure distances $\|x - x'\|^2$ for $x' \in$ labelled data

$\|x-x'\|^2$

$\Rightarrow$ Enough to compute dot products: $\langle x-x', x-x' \rangle$

Feature maps induce kernel functions

Dot product in feature space:

$$\langle \phi(x), \phi(x') \rangle = \phi(x)^T \phi(x')$$

$$= \begin{bmatrix} x_1^2 \\ x_2^2 \\ \sqrt{2}\, x_1 x_2 \end{bmatrix}^T \begin{bmatrix} x'^2_1 \\ x'^2_2 \\ \sqrt{2}\, x'_1 x'_2 \end{bmatrix}$$

$$= x_1^2 x'^2_1 + x_2^2 x'^2_2 + 2 x_1 x_2 x'_1 x'_2$$

$$= (x^T x')^2 =: k(x, x').$$

- dot product in feature space is a non-linear function k in input space, called *kernel function*
- can be calculated without explicitly mapping to the feature space via ϕ
- ϕ induces a kernel function

Question:

Can we avoid defining ϕ and directly specify k?

Kernel functions

Answer:

Yes! E.g. $k(x, x') = (x^\mathsf{T} x')^2$ can be generalized:

$$k(x, x') = (x^\mathsf{T} x')^p$$

$x_1^p + x_1^{p-2} x_1^1 + x_1^{p-2} x_1^2 + \dots$

- called *homogeneous polynomial kernel*
- one can show: $k(x, x') = \phi(x)^\mathsf{T} \phi(x')$ with

$$\phi : \underset{\mathbb{R}^d}{\text{input space}} \longrightarrow \underset{\mathbb{R}^D}{\text{feature space}}$$

$$\begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix} \longmapsto \begin{bmatrix} x_1^p \\ x_1^{p-1} x_2 \\ \vdots \end{bmatrix}$$

- i.e. x is mapped to all the monomials of degree p
- note that $D \gg d$
- kernel function calculates inner product in $\mathbb{R}^D$ without calculating all monomials which might be computationally prohibitive

Common kernel functions (1)

from Murphy Chapter 14

$(1, x, x^2, x^3, \ldots)$

$(1, x, y, xy, x^2, y^2, x^2y, x^3, \ldots)$

- Polynomial kernel

$$k(x, x') = (x^T x' + b)^p$$

called *homogenous* for $b = 0$ (all monomials of degree p, *inhomogenous* otherwise (all monomials up to degree p)

- (experts only:) polynomial kernel compares the higher-order moments of the class distributions
- Linear kernel

$$k(x, x') = x^T x'$$

special case of the polynomial kernel

- (experts only:) linear kernel compares the first-order moments (means) of the class distributions and thus leads to a linear decision function

Common kernel functions (2)

from Murphy Chapter 14

- Gaussian kernel, aka RBF kernel, aka squared exponenetial kernel

$$k(x, x') = \exp\left(-\frac{(x - x')^T (x - x')}{2\sigma^2}\right)$$

  where σ^2 is called the bandwidth
- Why Gaussians?
- The Gaussian kernel does measure similarity, e.g. $1 = k(x, x) \geq k(x, x')$, for other x'.
- The corresponding feature space is ∞-dimensional...
- (experts only:) the Gaussian kernel compares the kernel-density estimates of the two classes

Common kernel functions (3)

from Murphy Chapter 14

Measuring similarity between documents

- Summarize documents by bag of word representation, i.e. for document i let x_{ij} be the number of times word j appears in document i.
- Cosine similarity

$$k(x, x') = \frac{x^T x'}{\|x\|_2 \|x'\|_2} = \cos(\angle(x, x'))$$

- Note:
  - normalized inner product, i.e. the length doesn't matter
  - measures the cosine of the angle between two documents
  - $x_{ij} \geq 0$ so $0 \leq k(x, x') \leq 1$
  - in practice this should be more sophisticated: use tf-idf (term frequency inverse document frequency) representation

Common kernel functions (4)

from Murphy Chapter 14.2.6

Comparing two strings of variable lengths:

- how similar are

$$x = \texttt{LKSDJFLJWELELVISKJNEROIUSGOGFKJ}$$
$$x' = \texttt{OREIUTKELVISGNKJNRGKJNREKJ}$$

- s is a substring of x, if $x = usv$ for possibly zero-length strings u and v
- let $\phi_s(x)$ be the number of times that s appears in x
- count the number of substrings have in common, i.e.

$$k(x, x') = \sum_{s \in \mathcal{A}^*} w_s \phi_s(x) \phi_s(x')$$

  where $w_s \geq 0$ and $\mathcal{A}^*$ is the set of all strings
- surprisingly this can be calculated in $O(|x| + |x'|)$ using suffix trees
- $\longrightarrow$ Bioinformatics ($\mathcal{A} = \{ACTG\}$)

The famous kernel trick

Goal:

- create a nonlinear version of an existing linear algorithm

Requirement:

- the linear algorithm must only calculate dot products of the data points

Kernelization:

- replace all dot products by a kernel function

Examples:

- kernel PCA, kernel LDA, kernel CCA, kernel FDA, kernel ICA, ...

What is a kernel function?
What is a p.d. kernel function

Note

- So far, we know that kernel function measure similarity.

Feature maps vs kernel functions

Kernel function

- A symmetric (continuous) function $k : \mathcal{X} \times \mathcal{X} \to \mathfrak{R}$, i.e. $k(x, x') = k(x', x)$, is called a *kernel (function)*.

Every feature map ϕ induces a kernel function:

$$k(x, x') = \phi(x)^T \phi(x') = \phi(x')^T \phi(x) = k(x', x)$$

Does every kernel function k induce a feature map?

$$\phi(x) = ?$$

Answer:

- No, only if it fulfills additional requirements (spoiler: positive definiteness)!
- BTW: this topic is part of Functional Analysis.

What is a p.d. kernel function?

Definition 11.3

An $n \times n$ matrix $K \in \mathbb{R}^{n \times n}$ is positive definite, iff for any vector $v \in \mathbb{R}^n$ the inner product of v with its image under K is positive, i.e. $v^T K v > 0$. If we have $v^T K v \geq 0$ we call it positive semi-definite.

Definition 11.4

Given data points $x_1, \ldots, x_n \in \mathcal{X}$ and a kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, the Gram matrix K is an $n \times n$ matrix with entries defined by the kernel function, i.e. $K_{ij} = k(x_i, x_j)$.

Definition 11.5

A kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is positive (semi-)definite, iff for any set of data points $x_1, \ldots, x_n \in \mathcal{X}$ the corresponding Gram K is positive (semi-)definite.

Why positive definite?

Notation, notation, changes...

So far:

$$X = \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_n^T \end{bmatrix}$$

- i.e. each row of X is a data point

In the following example:

$$X = [x_1, x_2 \cdots x_n]$$

- i.e. each column of X is a data point

Why is pos-def a useful property?

Given data $X = [x_1, \ldots, x_n] \in \mathbb{R}^{d \times n}$.

Linear kernel:

- linear kernel function $k(x, x') = x^T x'$
- calculate kernel matrix $K = X^T X$
- kernel matrix contains all inner products $x_i^T x_j$
- thus $\phi(x) = x$

Nonlinear kernel:

- nonlinear kernel function $k(x, x')$
- calculate kernel matrix with entries $K_{ij} = k(x_i, x_j)$
- for a p.d. kernel function the kernel matrix factorizes $K = Z^T Z$ (see next slide why this is true)
- thus we can read off the feature map e.g. with $Z \in \mathbb{R}^{D \times n}$ and $D >> d$ as $\phi(x_i) = z_i$
- thus, the pd-ness of a kernel function guarantees that the kernel matrix factorizes like this $Z^T Z$

$$\Lambda^{\frac{1}{2}} = \begin{pmatrix} \sqrt{\lambda_1} & & \\ & \ddots & \\ & & \sqrt{\lambda_n} \end{pmatrix}$$

$$G = U \Lambda^{\frac{1}{2}} \cdot \Lambda^{\frac{1}{2}} U^T = \underbrace{U \Lambda^{\frac{1}{2}} U^T}_{Z^T} \underbrace{U \Lambda^{\frac{1}{2}} U^T}_{Z}$$

$$\left(\underbrace{U \Lambda^{\frac{1}{2}} U^T}_{A \quad B} \right)^T = U^{TT} \cdot (U \Lambda^{\frac{1}{2}})^T$$

$$= U \cdot \Lambda^{\frac{1}{2}T} U^T$$

$$= U \cdot \Lambda^{\frac{1}{2}} U^T$$

$$(AB)^T = B^T A^T$$

What a minute? Why does K factorize?

- since K is symmetric, it has a Eigenvector-decomposition

$$K = V \Lambda V^T$$

- because k is pd, the kernel matrix K is pd and thus all Eigenvalues are positive, i.e. they can be written as squares,

$$\Lambda_{ii} = \tau_i^2$$

- so we can factorize K as

$$K = (\sqrt{\Lambda} V^T)^T (\sqrt{\Lambda} V^T) = Z^T Z$$

- and read off $\phi(x_i) = z_i$ for the data points (with z_i being the the columns of Z)

Sketchy summary

Question: What functions are p.d. kernel function?

Short answer (in words)

- kernel function $k(a,b)$ must be symmetric, i.e. $k(a,b) = k(b,a)$
- kernel function must be positive definite, i.e. ...
- kernel functions are for functions, what pos. def. matrices are for matrices
- in Functional Analysis we study functions instead of vectors, linear operators instead of matrices
- kernel functions are pos. def. linear operators
- (analogous to: certain matrices are pos. def. linear mappings)

Can we “understand” the feature space? What do we know about it?

Representations of the feature space for given data (1)

Theorem 11.8 (Eigenvector decomposition of PD matrix)

If an $n \times n$ (symmetric) matrix K is positive definite, then there exist eigenvectors $v_1, \ldots, v_n \in \mathbb{R}^n$ and positive eigenvalues $\lambda_1, \ldots, \lambda_n \geq 0$ such that K can be rewritten as

$$K = \sum_{i=1}^{n} \lambda_i v_i v_i^\mathsf{T} = V \Lambda V^\mathsf{T},$$

where $v_i v_i^\mathsf{T} \in \mathbb{R}^{n \times n}$ is the outer product and $V = [v_1, \ldots, v_n]$ is the matrix with the eigenvectors as columns and Λ the diagonal matrix with the eigenvalues along its diagonal.

- given (training) data $x_1, \ldots, x_n \in \mathcal{X}$
- k PD implies the gram matrix K is PD
- thus $K = V \Lambda V^T = Z^\mathsf{T} Z$ with $Z = \Lambda^{1/2} V^\mathsf{T}$
- define feature map $\phi(x_i) := Z_{:i}$ as the ith column of Z
- however, this construction can not map arbitrary (test) points x

Note on positive definiteness

Q: are pos def matrices symmetric?

- Typically, pos def is only defined for symmetric (Hermitian) matrices, i.e. $A = A^H = \overline{A^T}$.
- Several possible reasons:
  1. if a matrix is not Hermitian, its eigenvalues can be complex. then what is positivity?
  2. for a non-symmetric matrix A, consider quadratic form $f_A(x) = x^T Ax$. Note that $f_A = f_{(A+A^T)/2}$. Thus pos def of f_A is same as pos def of symmetric $(A + A^T)/2$.
  3. for complex pos def matrix A, A Hermitian is equiv. to $x^T Ax$ being real....

Sources:

- `https://math.stackexchange.com/questions/1107230/why-is-positive-semi-definite-only-defined-for-symmetric-matrices`
- `https://math.stackexchange.com/questions/516533/symmetric-vs-positive-semidefinite?rq=1`

Representations of the feature space for given data (2)

Given data $x_1, \ldots, x_n \in \mathcal{X}$.

Definition 11.9 (Empirical kernel map)

$$\begin{array}{rccc} \phi_n : & \mathcal{X} & \longrightarrow & \mathbb{R}^n \\ & x & \longmapsto & \begin{bmatrix} k(x_1, x) \\ \vdots \\ k(x_n, x) \end{bmatrix} \end{array}$$

Unfortunately:

$$k(x, x') \neq \phi_n(x)^\mathsf{T} \phi_n(x').$$

But we have:

$$k(x, x') = \phi_n(x)^\mathsf{T} K^{-1} \phi_n(x').$$

Representations of the feature space for given data (3)

Given data $x_1, \ldots, x_n \in \mathcal{X}$.

Definition 11.10 (Kernel PCA map)

$$\begin{array}{rccc} \phi: & \mathcal{X} & \longrightarrow & \mathbb{R}^n \\ & x & \longmapsto & K^{-1/2} \begin{bmatrix} k(x_1, x) \\ \vdots \\ k(x_n, x) \end{bmatrix} \end{array}$$

- ϕ is a feature map for k

$$k(x, x') = \phi_n(x)^\mathsf{T} K^{-1} \phi_n(x').$$

Representations of the feature space without data (1)

Notation:

$$3 := \{0, 1, 2\}$$
$$d := \{0, 1, \ldots, d-1\}$$

Vectors as a mappings:

$$\begin{array}{rccc} v: & d & \longrightarrow & \mathbb{R} \\ & i & \longmapsto & v_i \end{array}$$

Sets of functions:

$$\mathbb{R}^d = \{f : d \to \mathbb{R}\}$$
$$\mathbb{R}^{\mathcal{X}} = \{f : \mathcal{X} \to \mathbb{R}\}$$

Kernel functions as continuous generalizations of matrices:

$$\mathbb{R}^{n \times n} = \{f : n \times n \to \mathbb{R}\}$$
$$\mathbb{R}^{\mathcal{X} \times \mathcal{X}} = \{f : \mathcal{X} \times \mathcal{X} \to \mathbb{R}\}$$

Representations of the feature space without data (2)

Generalizing eigenvector decomposition to functions:

Theorem 11.11 (Mercer)

If a kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is positive definite, then there exist eigenfunctions $\phi_1, \phi_2, \ldots : \mathcal{X} \to \mathbb{R}$ and positive eigenvalues $\lambda_1, \lambda_2, \ldots \geq 0$ such that k can be rewritten as:

$$k(x, x') = \sum_{i=1}^{\infty} \lambda_i \phi_i(x) \phi_i(x').$$

Definition 11.12 (Mercer kernel map)

$$\begin{array}{rccc} \phi: & \mathcal{X} & \longrightarrow & \mathbb{R}^\infty \\ & x & \longmapsto & \begin{bmatrix} \sqrt{\lambda_1}\,\phi_1(x) \\ \sqrt{\lambda_2}\,\phi_2(x) \\ \vdots \end{bmatrix}. \end{array}$$

Note that $\mathbb{R}^\infty = \mathbb{R}^{\mathbb{N}}$ and $\langle \phi(x), \phi(x') \rangle = k(x, x')$ by Mercer's theorem.

Note

- Mercer's theorem doesn't give us a recipe to calculate ϕ
- AFAIK (as far as I know) it is only a result about the existence of ϕ
- if you want to be sure about this, check the proof, whether it is constructive, i.e. whether it provides the general recipe...

Representations of the feature space without data (3)

- Mercer kernel map maps to countably infinite dimensional space
- next we define a feature space of functions

Definition 11.13 (Reproducing kernel map)

$$\begin{array}{rccc} \phi: & \mathcal{X} & \longrightarrow & \mathbb{R}^{\mathcal{X}} \\ & x' & \longmapsto & k(\cdot, x') \end{array}$$

where $k(\cdot, x')$ is k curried, i.e.

$$\begin{array}{rccc} k(\cdot, x'): & \mathcal{X} & \longrightarrow & \mathbb{R} \\ & x & \longmapsto & k(x, x'). \end{array}$$

- $\mathbb{R}^{\mathcal{X}}$ is the space of functions from $\mathcal{X}$ to $\mathbb{R}$
- $\phi(\mathcal{X})$ is the feature space image of the input space

Representations of the feature space without data (4)

Question:

- How can we turn $\phi(\mathcal{X}) \subset \mathbb{R}^{\mathcal{X}}$ into a Hilbert space?

Three steps:

1. Turn $\phi(\mathcal{X})$ into a vector space $\mathcal{H}$.
2. Define dot-product in $\mathcal{H}$.
3. Complete $\mathcal{H}$ to a Hilbert space.

Ad 1.

Given data points $x_1, \ldots, x_m \in \mathcal{X}$ and coefficients $\alpha_1, \ldots, \alpha_m \in \mathbb{R}$ let

$$f(\cdot) = \sum_{i=1}^{m} \alpha_i k(\cdot, x_i) \in \mathcal{H}.$$

Thus for $x' \in \mathcal{X}$ we have $\phi(x') = k(\cdot, x') \in \mathcal{H}$.

Representations of the feature space without data (5)

- consider two function $f, g \in \mathcal{H}$:

$$f(\cdot) = \sum_{i=1}^{m} \alpha_i k(\cdot, x_i) \qquad g(\cdot) = \sum_{j=1}^{m'} \beta_i k(\cdot, x'_i)$$

- linear combinations: $\gamma f(\cdot) + g(\cdot) \in \mathcal{H}$
- define dot product in $\mathcal{H}$:

$$\langle f, g \rangle := \sum_{i=1}^{m} \sum_{j=1}^{m'} \alpha_i \beta_j k(x_i, x'_j) \in \mathbb{R}$$

Theorem 11.14

$\langle f, g \rangle$ is a dot product, i.e.

(i) $\langle f, g \rangle = \langle g, f \rangle$ — *symmetry*

(ii) $\langle af_1 + bf_2, g \rangle = a\langle f_1, g \rangle + b\langle f_2, g \rangle$ *and* $\langle f, ag_1 + bg_2 \rangle = a\langle f, g_1 \rangle + b\langle f, g_2 \rangle$ — *bilinearity*

(iii) $\langle f, f \rangle = 0$ *implies* $f(x) = 0$ *for any* $x \in \mathcal{X}$ — *strictly positive definite*

Representations of the feature space without data (6)

To show (iii) we need a lemma:

Lemma 11.15

(iv) $\langle k(\cdot, x), g\rangle = g(x)$ — *k is representer of evaluation*

(v) $\langle k(\cdot, x), k(\cdot, x')\rangle = k(x, x')$ — *k is reproducing*

(vi) $\langle f, f\rangle \geq 0$

(vii) $\langle \cdot, \cdot \rangle$ *is positive definite kernel function for two elements of the function space* $\mathbb{R}^{\mathcal{X}}$.

(viii) $|k(x, x')|^2 \leq k(x, x)k(x', x')$ — *Cauchy-Schwartz for kernels*

(ix) $|\langle f, g\rangle|^2 \leq \langle f, f\rangle\langle g, g\rangle$

Proof: To show (iv) note that the dot product can be rewritten with the definitions of f and g,

$$\sum_{i=1}^{m} \alpha_i g(x_i) = \langle f, g \rangle = \sum_{j=1}^{m'} \beta_j f(x'_j)$$

Replacing in the left equation f by $k(\cdot, x)$ we showed (iv). Replacing g in (iv) by $k(\cdot, x')$ we showed (v).
(vi) follows from the PDness of k, since it implies the following inequality:

$$\langle f, f \rangle = \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j k(x_i, x_j) = \alpha^T K \alpha \geq 0$$

where $\alpha = [\alpha_1, \ldots, \alpha_m]^T$ is the vector of coefficients of f and the data points $x_1, \ldots, x_m \in \mathcal{X}$ induce the Gram matrix K.

Proof continued:

(vii) appears at first glance a bit weird. It is only of technical importance to show that the Cauchy-Schwartz inequality holds for the dot-product. To show (vii) we follow the definition of PDness of kernel functions. We consider functions $f_1, \ldots, f_n \in \mathrm{H}$ and define the $n \times n$ Gram matrix G with entries $G_{ij} = \langle f_i, f_j \rangle$. For any vector $v \in \mathbb{R}^n$ we have to show $v^\mathsf{T} G v \geq 0$,

$$v^\mathsf{T} G v = \sum_{i=1}^{n} \sum_{j=1}^{n} v_i v_j G_{ij} = \sum_{i=1}^{n} \sum_{j=1}^{n} v_i v_j \langle f_i, f_j \rangle = \langle \sum_{i=1}^{n} v_i f_i, \sum_{j=1}^{n} v_j f_j \rangle \geq 0$$

where we used the bilinearity of the dot product, and (vi) for the inequality. We have shown that (vii) for the only reason to prove (ix) which follows from (viii) and (vii). For brevity we do not show a proof of (viii).

Proof (continued):
Having fully proven the lemma, we can now prove (iii). For arbitrary $x \in \mathcal{X}$ and $f \in \mathbb{R}^{\mathcal{X}}$,

(...) exercise

Thus $f(x) = 0$, which completes the proof.

Back to $\mathcal{H}$:
Up to now $\mathcal{H}$ is a pre-Hilbert space (a vector space with dot-product). To complete $\mathcal{H}$ to a Hilbert space, limits must be added to it and the dot-product appropriately extended. ...

End of appendix

Machine Learning

Section 12: Dimensionality reduction: PCA and kPCA

Stefan Harmeling

24./29. November 2021

What we have seen so far?

Sections:

1. Introduction
2. Plausible reasoning and Bayes Rule
3. From Logic to Probabilities
4. Bayesian networks
5. Continuous Probabilities
6. The Gaussian distribution
7. More on distributions, models, MAP, ML
8. Linear Regression
9. Matrix Differential Calculus
10. Model selection
11. Support Vector Machines

Dimensionality reduction

Dimensionality reduction

X $n \times d$ - matrix

n data points in $\mathbb{R}^d$

High-level idea

- try to map your high-dimensional data into low-dimensional data without loosing too much information
- possibly makes a classification/regression problem easier

Simple approaches

- ignore most of the dimensions
- ignore the dimensions with low variance

PCA

- an instance of dimensionality reduction
- consider a few linear combinations of all dimensions
- based on linear algebra, in particular SVD and Eigenvalue decomposition

Definition 12.1 (Dimensionality reduction)

Given n data points $X = [x_1, \ldots, x_n] \in \mathbb{R}^{D \times n}$. Find a low dimensional representation $Z = [z_1, \ldots, z_n] \in \mathbb{R}^{d \times n}$ with $d \ll D$ that keeps most of the properties of the higher dimensional data.

What properties should be kept?

- variance (PCA)
- neighborhood relationships (ISOMAP, LLE)

Dimensionality reduction

What is a representation?

- model the representation as a map $f : \mathbb{R}^D \to \mathbb{R}^d$
  - linear: $Z = W^T X$ with $D \times d$ matrix W
    - W rectangular matrix with few non-zeros along the diagonal: “ignore some of the components”
    - W arbitrary rectangular matrix: “define low dimensional data by linear projections”
    - W parameterizes the linear map
  - nonlinear: $Z = f(X)$
    - f could be neural network or any other nonlinear function
    - the weights of the neural network parameterize the map
- model the representation non-parametrically (e.g. ISOMAP, LLE)
  - how to do this without parameters? (non-parametrically?)
  - given a dataset $x_1, \ldots, x_n \in \mathbb{R}^D$ store the lower dimensional vectors $z_1, \ldots, z_n \in \mathbb{R}^d$ to represent the mapping
  - how to map other points? e.g. nearest neighbor among the data, or use linear combination for new data points...

PCA: step by step

Algorithm

1. given $D \times n$ data matrix $X = [x_1, \ldots, x_n] \in \mathbb{R}^{D \times n}$, assume mean is already removed
2. calculate sample covariance matrix $\Sigma = \frac{1}{n} \sum_{i=1}^{n} x_i x_i^T = \frac{1}{n} X X^T$
3. calculate eigenvector decomposition $\Sigma = V \Lambda V^T$
4. with $V = [v_1, \ldots, v_D]$ and $\lambda_1 \geq \ldots \geq \lambda_D$
5. pick the d largest eigenvalues that carry most of the variance (there might be a drop in the eigenvalue spectrum)
6. project the data onto $V_d = [v_1, \ldots, v_d]$

$$Z = V_d^T X$$

7. alternatively rescale the axis to have make the covariance matrix identity (aka whitening)

$$Z = \Lambda_d^{-0.5} V_d^T X$$

where Λ_d contains λ_1 through λ_d.

Covariance matrix: $X = \begin{pmatrix} \text{---} & x_1 & \text{---} \\ & \vdots & \\ \text{---} & x_n & \text{---} \end{pmatrix}$

Entry (i,j): $\frac{1}{n-1}\left(x_i - \underbrace{\frac{1}{n}\sum_{i=1}^{n} x_i}_{=0}\right)\left(x_j - \underbrace{\frac{1}{n}\sum_{i=1}^{n} x_i}_{=0}\right)$

For centered data:

$$\text{Cov. matrix} = X \cdot X^T$$

$$a^T \cdot b = (a_1, \dots, a_n)\begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}, \qquad \begin{pmatrix} a_1 \\ \vdots \\ a_n \end{pmatrix} \cdot (b_1, \dots, b_n) = \begin{pmatrix} a_1b_1 & a_1b_2 & \\ a_2b_1 & \vdots & \\ \vdots & & \\ a_nb_1 & & \end{pmatrix}$$

Summary / overview

Idea of PCA

- maximize Variance, i.e. $\operatorname{argmax}_W \|W^T X\|^2$
- minimize MSE, i.e. $\operatorname{argmin}_{Z,W} \|X - WZ\|^2$

Results

- minMSE and maxVar are equivalent: for MSE show that $Z = W^T X$ using constrained optimization, and plug it in
- direction of largest variance is Eigenvector of largest Eigenvalue of cov matrix, similar, second largest variance

Algorithm

- PCA: step by step (slide)

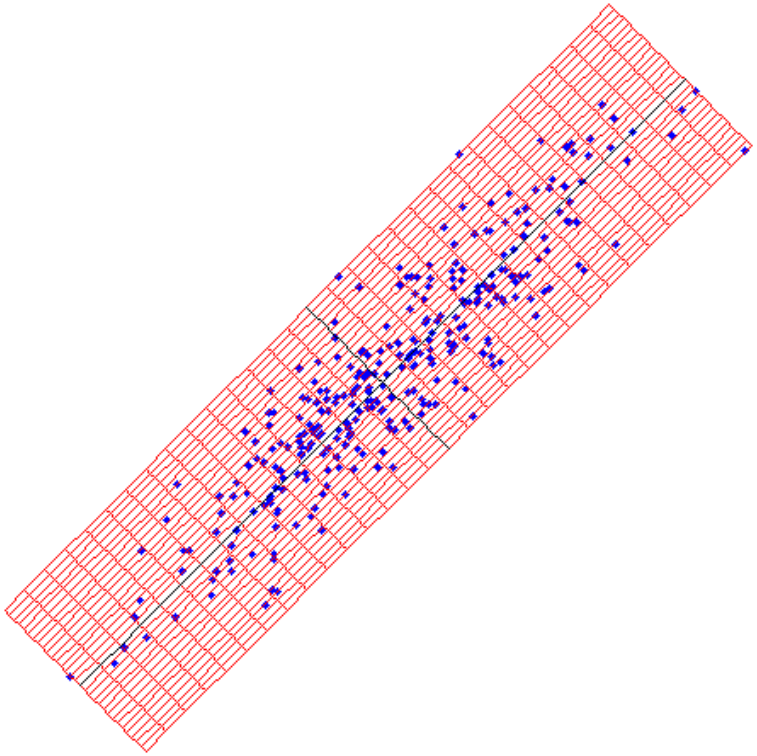
Part 2:
Kernel PCA

- first introduced: Schölkopf, Smola, Müller, “Nonlinear component analysis as a kernel eigenvalue problem”, 1998.
- many details in: Schölkopf, Smola, “Learning with kernels”, 2002.

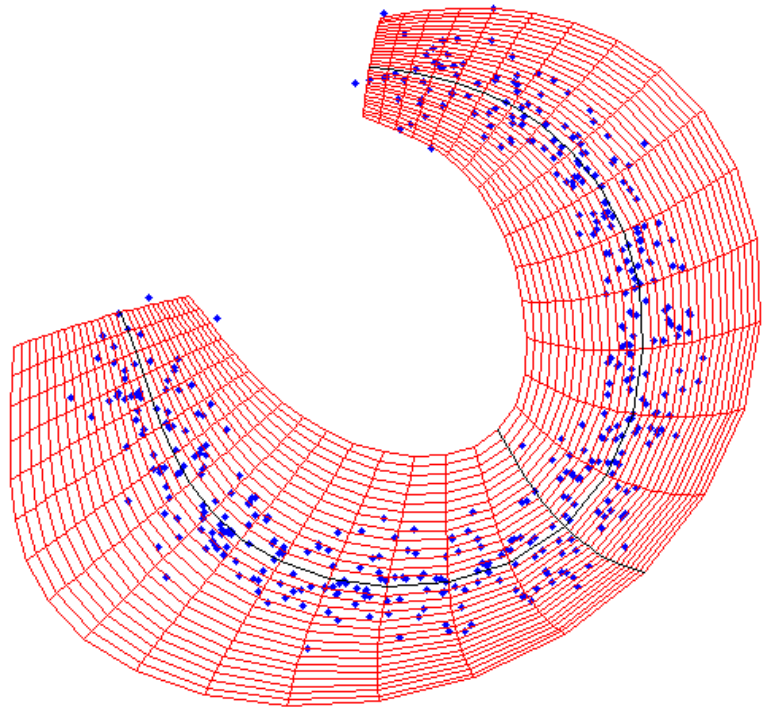
What is nonlinear PCA?

see blackboard

Linear PCA

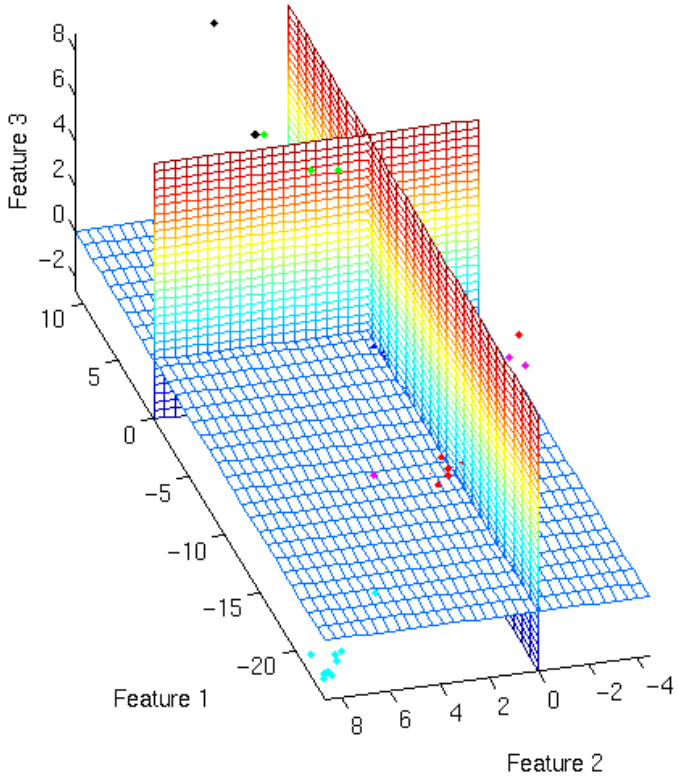


Nonlinear PCA

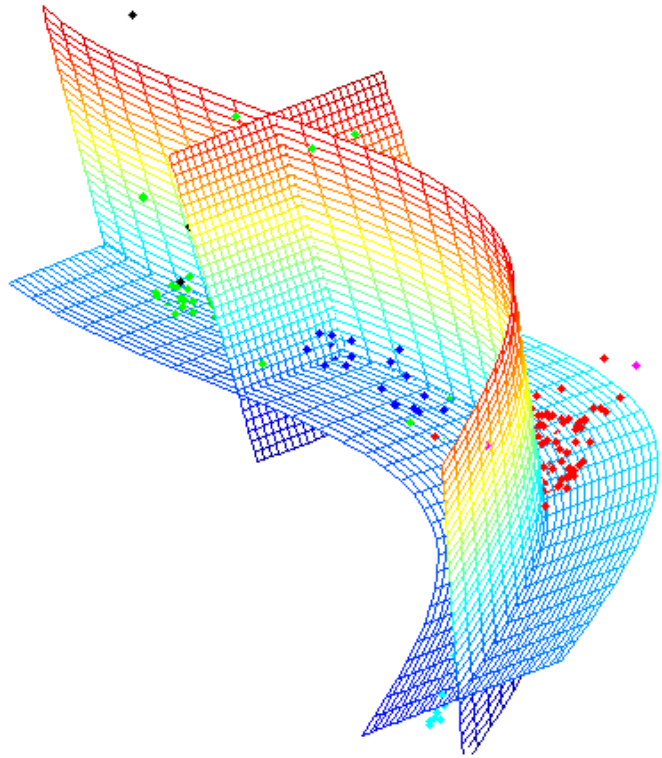


Images copied from Matthias Scholz, `http://nlpca.org`

Linear PCA



Nonlinear PCA



Images copied from Matthias Scholz, `http://nlpca.org`

Nonlinear PCA

Challenge

- find a curved coordinate system through the data
- fit by maximizing variance, while at the same time keeping the mapping simple (find a compromise)
- for an overview see: `https://en.wikipedia.org/wiki/Nonlinear_dimensionality_reduction`

Next

- let's try to "nonlinearize" PCA using the kernel trick!

Algorithm 12.2 (Linear PCA: based on covariance matrix)

Input: $D \times n$ *data matrix* $X = [x_1, \ldots, x_n] \in \mathbb{R}^{D \times n}$, *assume mean is already removed*
Output: a low dimensional embedding $Z = [z_1, \ldots, z_n] \in \mathbb{R}^{d \times n}$ *with* $d < D$ *that keeps variance of* X

1. *calculate sample covariance matrix* $\Sigma = \frac{1}{n} \sum_{i=1}^{n} x_i x_i^T = \frac{1}{n} X X^T$
2. *calculate eigenvector decomposition* $\Sigma = V \Lambda V^T$
3. *with* $V = [v_1, \ldots, v_D]$ *and* $\lambda_1 \geq \ldots \geq \lambda_D$
4. *pick the* d *largest eigenvalues that carry most of the variance (there might be a drop in the eigenvalue spectrum)*
5. *project the data onto* $V_d = [v_1, \ldots, v_d]$

$$Z = V_d^T X$$

6. *alternatively rescale the axis to have make the covariance matrix identity (aka whitening)*

$$Z = \Lambda^{-0.5} V_d^T X$$

The famous kernel trick

Goal:

- create a nonlinear version of an existing linear algorithm

Requirement:

- the linear algorithm must only calculate dot products of the data points

Kernelization:

- replace all dot products by a kernel function

Examples:

- kernel PCA, kernel LDA, kernel CCA, kernel FDA, kernel ICA, ...

PCA with inner products?

From Algorithm 12.2:

Where do we access the data points X?

3. calculate sample covariance matrix $\Sigma = \frac{1}{n}\sum_{i=1}^{n} x_i x_i^T = \frac{1}{n} XX^T$
6. project the data onto $V_d = [v_1, \ldots, v_d]$, i.e. $Z = V_d^T X$

Problem

- cov matrix requires outer products $x_i x_i^T$ but not inner products!

Challenge

- How can we formulate PCA solely using inner products?

Some background from linear algebra

Linear transformations

- A matrix A describes a linear transformation!
- E.g. a matrix $A = [a_1\ a_2]$ with two columns a_1 and a_2 describe what happens to the unit vectors:

$$A \begin{bmatrix} 1 \\ 0 \end{bmatrix} = a_1 \qquad A \begin{bmatrix} 0 \\ 1 \end{bmatrix} = a_2$$

- Any other vector gets mapped to a linear combination of those two, e.g.

$$A \begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \alpha a_1 + \beta a_2$$

- See 3Blue1Brown at
  `https://www.youtube.com/watch?v=kYB8IZa5AuE&ab_channel=3Blue1Brown`.

Puzzle number 1

- Given some matrix $A = [a_1\ a_2]$ with two columns.
- Find two orthonormal vectors v_1 and v_2 (i.e. $v_i^T v_j = [i = j]$), such that their images are orthogonal.
- Let's first write the images as scaled unit-length vectors:

$$\tau_1 w_1 = A v_1 \qquad\qquad \tau_2 w_2 = A v_2$$

where τ_1 and τ_2 are scalars, and w_1 and w_2 are unit-length vectors, written as matrices:

$$A \begin{bmatrix} v_1 & v_2 \end{bmatrix} = \begin{bmatrix} \tau_1 w_1 & \tau_2 w_2 \end{bmatrix} = \begin{bmatrix} w_1 & w_2 \end{bmatrix} \begin{bmatrix} \tau_1 & 0 \\ 0 & \tau_2 \end{bmatrix}$$

or even shorter: $AV = W\Lambda$

- Reformulation of the puzzle: given a matrix A find unitary matrix V and W and diagonal matrix Λ, s.t. $AV = W\Lambda$, or that (after right-multiplying V^T

$$A = W\Lambda V^T$$

ideas from https://www.youtube.com/watch?v=CpD9XlTu3ys&ab_channel=Stochastic

Solution to the puzzle:

Theorem 12.3 (Singular value decomposition/SVD)

Any matrix X can be factorized into three matrices, i.e. can be written as a product of three matrices

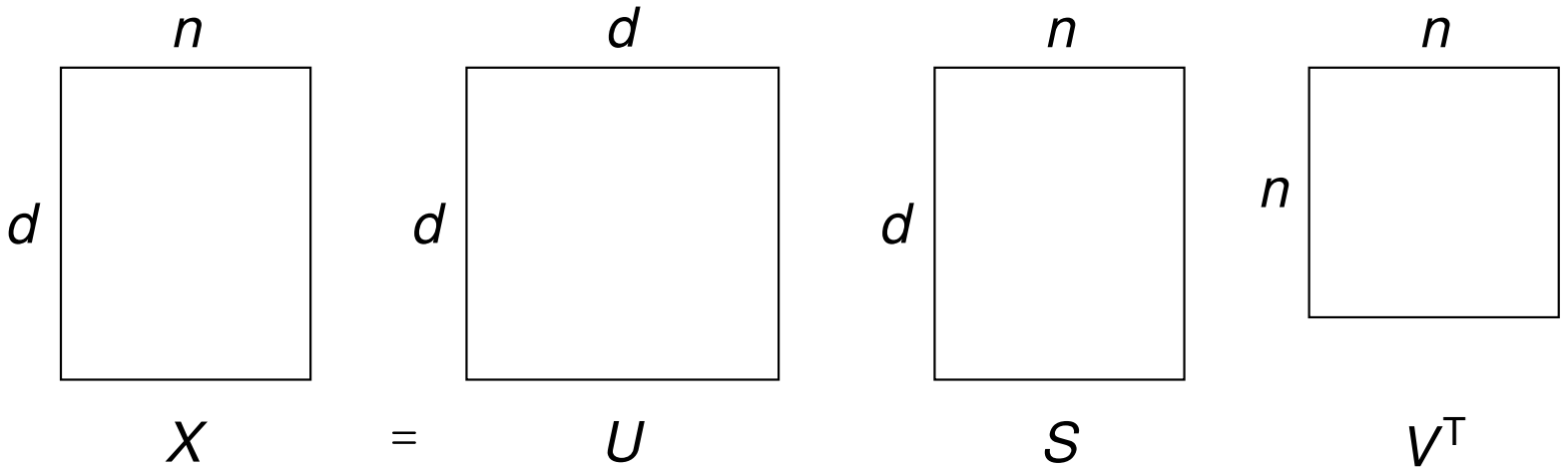
$$X = USV^T$$

such that

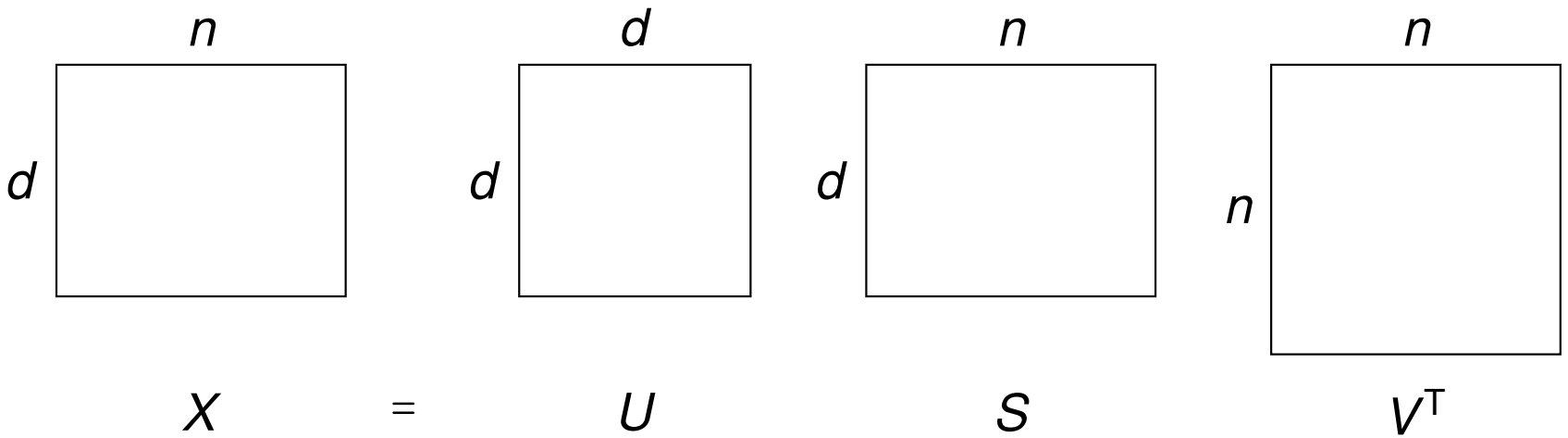
- *U is $d \times d$, and V is $n \times n$, so S is rectangular of size $d \times n$.*
- *U and V are unitary, i.e. $UU^T = I_d$ and $VV^T = I_n$*
- *I_d and I_n being $n \times n$ and $d \times d$ dimensional identity matrices*
- *S is diagonal matrix, entries are called singular values (SVs)*

SVD — graphically

Case (i): $d \geq n$

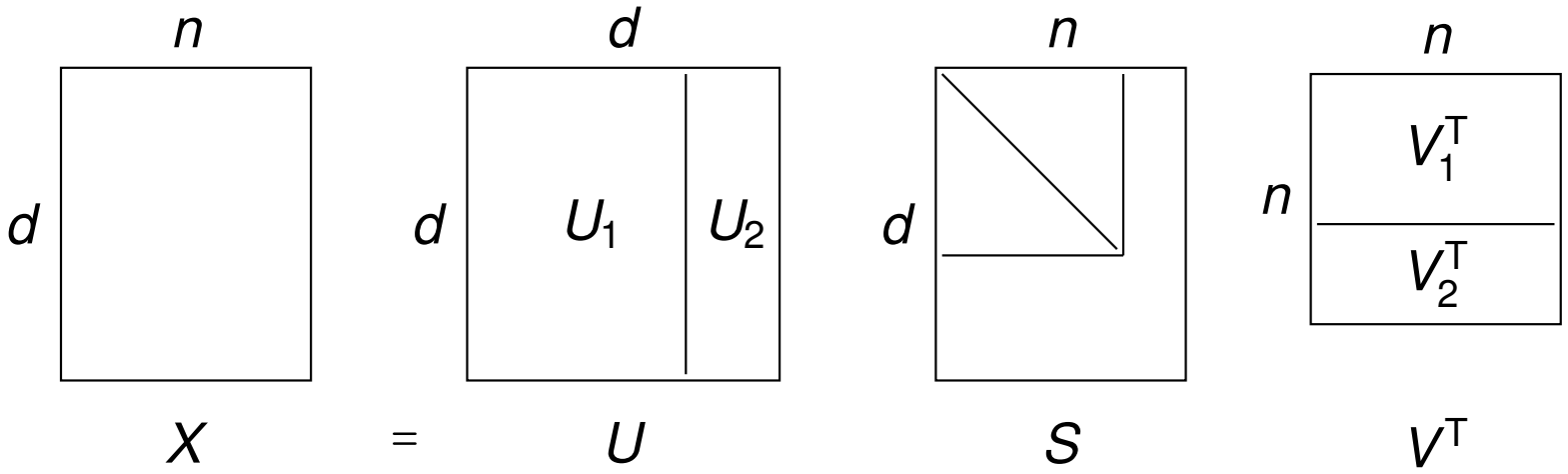


Case (ii): $d < n$

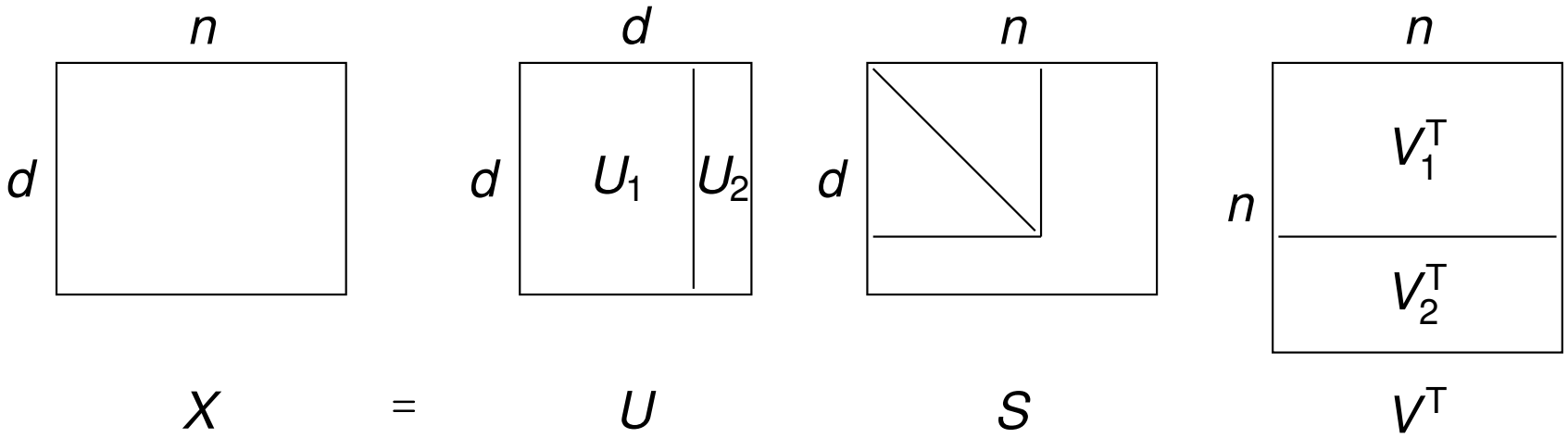


SVD — range and null space

Case (i): $d \geq n$



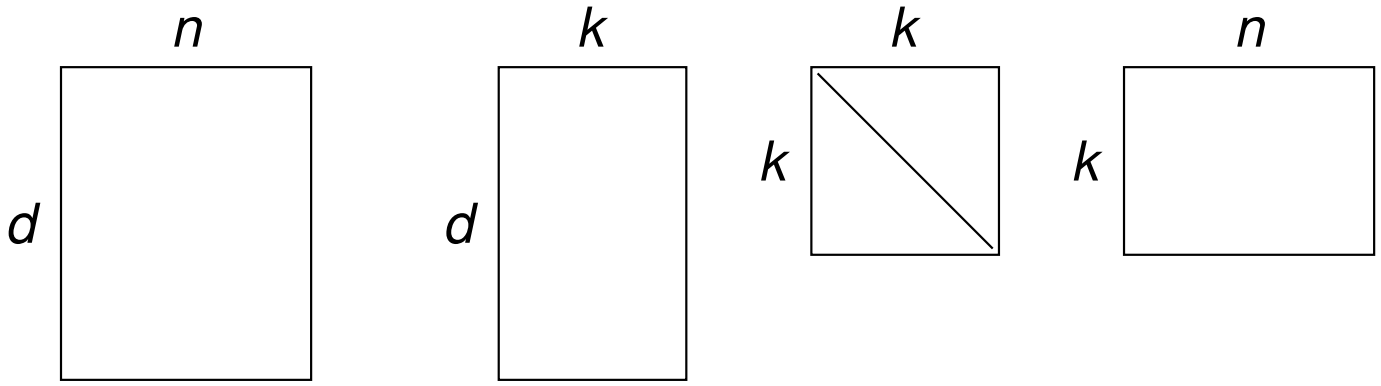
Case (ii): $d < n$



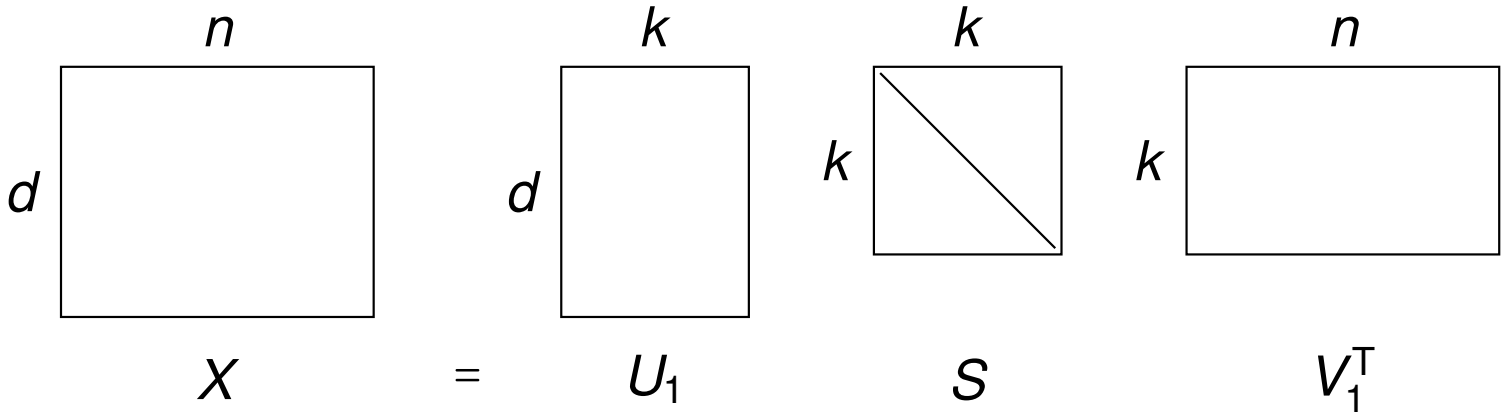
Null space of X is V_2. *Range* of X is U_1.

SVD — economy size and rank

Case (i): $d \geq n$



Case (ii): $d < n$



rank of X is k, the number of non-zero singular values.

Puzzle number 2

- Given some squared matrix $A = [a_1\ a_2]$ with two columns.
- Find two orthonormal vectors v_1 and v_2 (i.e. $v_i^T v_j = [i = j]$), such that their images are just scaled versions of v_1 and v_2.
- Let's first write the images as scaled versions:

$$\tau_1 v_1 = A v_1 \qquad\qquad \tau_2 v_2 = A v_2$$

  where τ_1 and τ_2 are scalars, written as matrices:

$$A \begin{bmatrix} v_1 & v_2 \end{bmatrix} = \begin{bmatrix} \tau_1 v_1 & \tau_2 v_2 \end{bmatrix} = \begin{bmatrix} v_1 & v_2 \end{bmatrix} \begin{bmatrix} \tau_1 & 0 \\ 0 & \tau_2 \end{bmatrix}$$

  or even shorter: $AV = V\Lambda$
- Reformulation of the puzzle: given a matrix A find unitary matrix V and diagonal matrix Λ, s.t. $AV = V\Lambda$, or that (after right-multiplying V^T

$$A = V \Lambda V^T$$

Solution for symmetric matrices:

Theorem 12.4 (Eigenvalue decomposition/EVD)

Any symmetric matrix A can be decomposed into

$$A = V \Lambda V^T$$

- *with Λ being diagonal matrix*
- *with V being unitary matrix, i.e. $VV^T = V^T V = I$*
- *values along the diagonal of Λ are called eigenvalues*
- *columns of V are called eigenvectors*

Note

- for eigenvalue λ and eigenvector v, the factorization $A = V \Lambda V^T$ implies $Av = \lambda v$
- EVs stands for “eigenvectors” or for “eigenvalues” depending on the contexts...

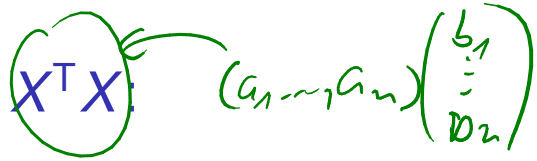
Are EVD and SVD related?

YES!

(Economy-sized) SVD of a (rectangular) data matrix X:

$$X = USV^T.$$

Calculate XX^T and X^TX:



$$XX^T = USV^T VSU^T = US^2U^T = U\Lambda U^T$$

$$X^TX = VSU^T USV^T = VS^2V^T = V\Lambda V^T$$

- left singular vectors U of X are the eigenvectors of XX^T
- right singular vectors V of X are the eigenvectors of X^TX
- the squared SVs of X are the eigenvalues of XX^T *and* of X^TX

XX^T $d \times d$ (d = nr. of features)

$X\bar{X}^T$ $n \times n$ (n = nr. of samples)

If $n < d$, then this is better to compute

(e.g. time series)

Lemma 12.5

The left SVs can be computed from the right SVs and vice versa:

$$U = XV\Lambda^{-1/2} \qquad \Leftarrow \qquad X = U\Lambda^{\frac{1}{2}}V^{\top}$$

$$V = X^{\mathsf{T}}U\Lambda^{-1/2}$$

Note: if the signs of $\Lambda^{1/2}$ do not match S, some of the vectors in U change their orientation. However, that is no problem. A similar result holds for the submatrices:

$$U_1 = XV_1\Lambda_1^{-1/2}$$

$$V_1 = X^{\mathsf{T}}U_1\Lambda_1^{-1/2}$$

where U_1 and V_1 contain the columns of U and V that correspond to large eigenvalues in Λ.

Proof:

Plug SVD of X into the formulas.

Computational trick (1)

Consider *very* high dimensional data points

$$X = [x_1, \ldots, x_n] \in \mathbb{R}^{D \times n} \qquad \text{with } D \gg n$$

EVD of the covariance matrix?

$$\Sigma = \frac{1}{n} X X^T \in \mathbb{R}^{D \times D}$$

- EVD of XX^T costs $O(D^3)$

Idea: calculate EVD of the inner product matrix

$$\frac{1}{n} X^T X = \frac{1}{n} \begin{bmatrix} x_1^T x_1 & \cdots & x_1^T x_n \\ \vdots & & \vdots \\ x_n^T x_1 & \cdots & x_n^T x_n \end{bmatrix} = \in \mathbb{R}^{n \times n}$$

- EVD of $X^T X$ costs $O(n^3)$

Computational trick (2)

Consider *very* high dimensional data points

$$X = [x_1, \ldots, x_n] \in \mathbb{R}^{D \times n} \qquad \text{with } D \gg n$$

Algorithm 12.6 (EVD of the covariance matrix)

1. *calculate the EVD of the inner product matrix*

$$X^T X = V \Lambda V^T \in \mathbb{R}^{n \times n}$$

2. *transform the EVs of $X^T X$ into EVs of XX^T*

$$U_1 = X V \Lambda^{-1/2} \in \mathbb{R}^{D \times n}$$

This gives us (economy-sized) EVD of (rank n) covariance matrix

$$XX^T = U_1 \Lambda U_1^T \in \mathbb{R}^{D \times D}$$

We never calculate XX^T, instead we are happy with U_1 and Λ

using the previous computational trick:

Algorithm 12.7 (PCA based on the Gram matrix)

1. *calculate EVD of Gram matrix*

$$X^T X = V \Lambda V^T$$

   Note that Λ are also the EVs of covariance matrix XX^T

2. *calculate U such that $XX^T = U \Lambda U^T$ via*

$$U = XV\Lambda^{-1/2}$$

   Possibly only for U_1 corresponding to the large EVs.

3. *project the data points onto the space spanned by U_1*

$$Y = U_1^T X = \Lambda_1^{-1/2} V_1^T X^T X$$

Note:

- We never have to calculate XX^T, not even U.
- Thus this works even for ∞-dimensional $\phi(X)$
- We only calculate $X^T X$, V_1, Λ_1 and Y.

How deal with non-zero mean the matrix-way (1)

Mean of data:

$$\begin{pmatrix} \mu_1 \\ \vdots \\ \mu_m \end{pmatrix} = \mu = \frac{1}{n} X 1_n$$

- where 1_n is the n dimensional one-vector, e.g. `ones(n)`.

Removing the mean:

$$X - \mu 1_n^T = X - \frac{1}{n} X 1_n 1_n^T = X(I - \frac{1}{n} 1_n 1_n^T) = XH$$

- note that $\mu 1_n^T = [\mu \cdots \mu] \in \mathbb{R}^{D \times n}$
- we defined the *centering matrix* $H = I - \frac{1}{n} 1_n 1_n^T$ which has properties $H = H^T$ and $H = HH$ (idempotent).
- multiplying X from the right with H, removes the mean

How deal with non-zero mean the matrix-way

Gram matrix for non-zero mean:

$$G = (XH)^\mathsf{T}(XH) = HX^\mathsf{T}XH$$

Removing the mean in feature space

$$HKH = H\phi(X)^\mathsf{T}\phi(X)H = (\phi(X)H)^\mathsf{T}(\phi(X)H)$$

in words: left- and right-multiplying the kernel matrix with the centering matrix removes the mean in feature space

- practically, we don't remove the mean of the data in feature space
- instead we know how the Gram matrix (kernel matrix) changes if the mean in feature space is removed

Algorithm 12.8 (Kernel PCA, kPCA)

Given data points $X = [x_1, \ldots, x_n] \in \mathbb{R}^{D \times n}$.

1. *calculate the kernel matrix (aka Gram matrix) for some kernel* k*:*

$$K = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_n) \\ \vdots & & \vdots \\ k(x_n, x_1) & \cdots & k(x_n, x_n) \end{bmatrix}$$

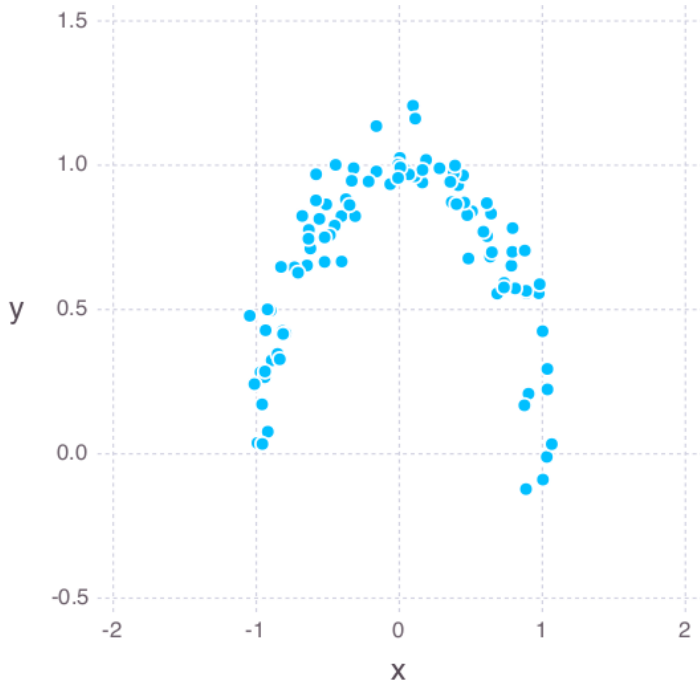
2. *find the eigenvalue decomposition of the centered Gram matrix:*

$$HKH = V \Lambda V^T$$

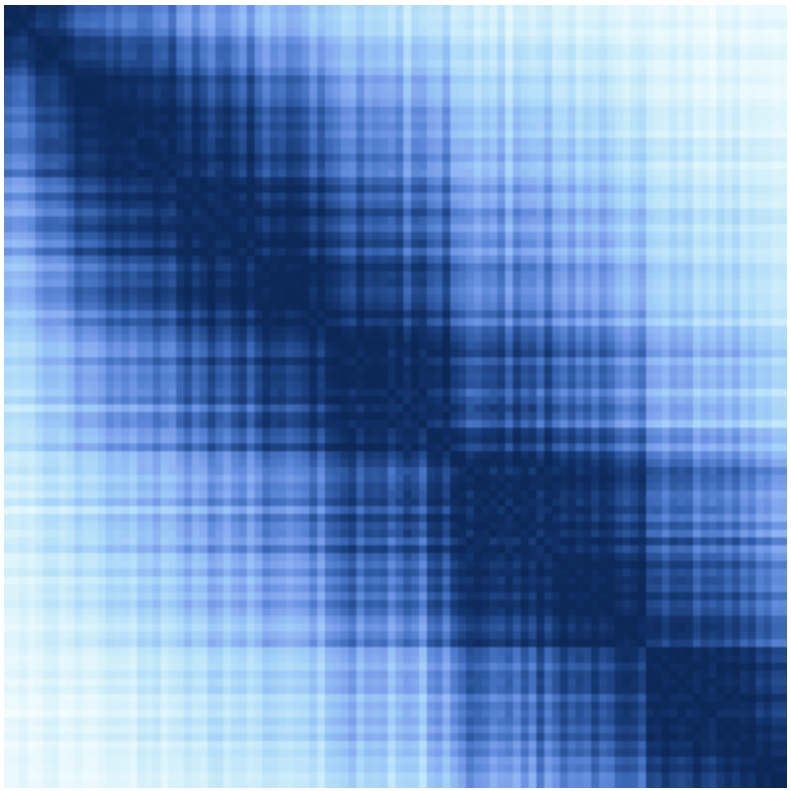
3. *project centered data onto the eigenvectors (as above):*

$$Y = \Lambda_1^{-1/2} V_1^T HKH$$

KPCA on toy data: arc (1)

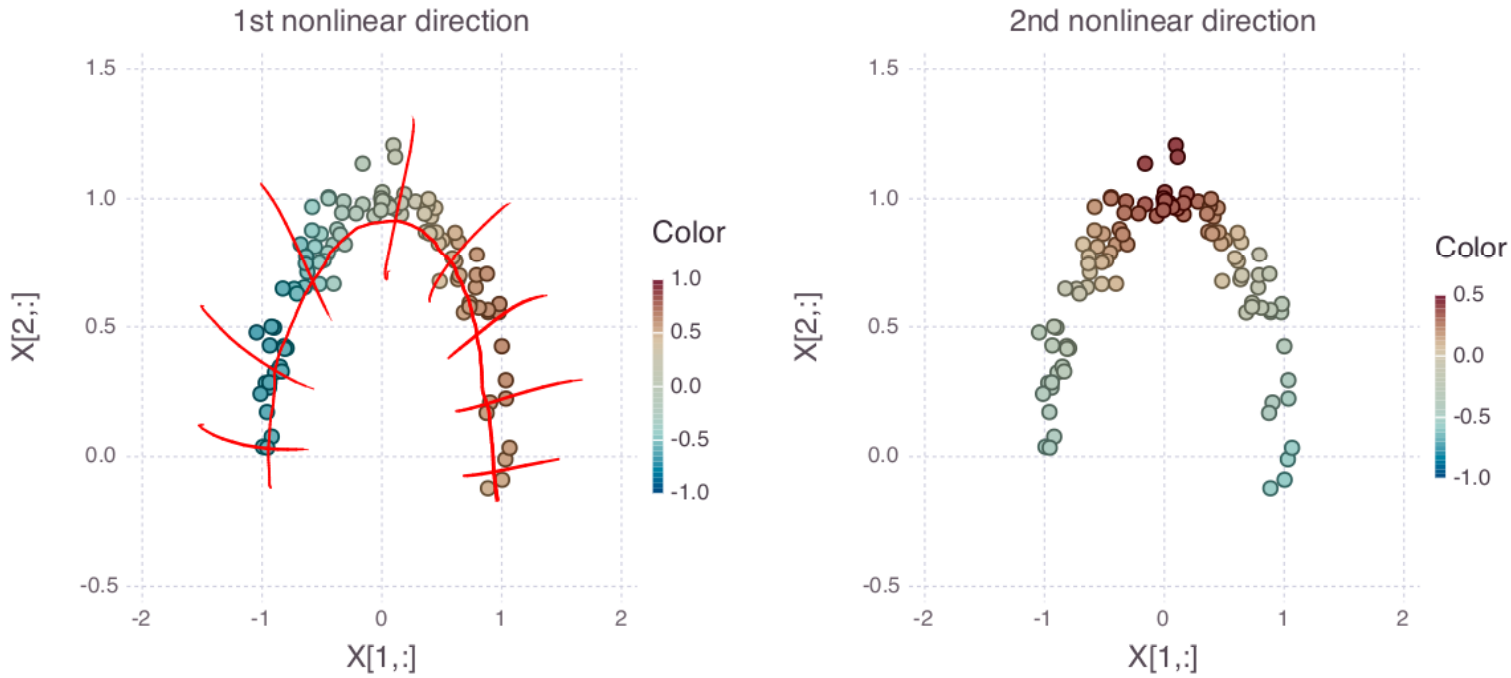


data



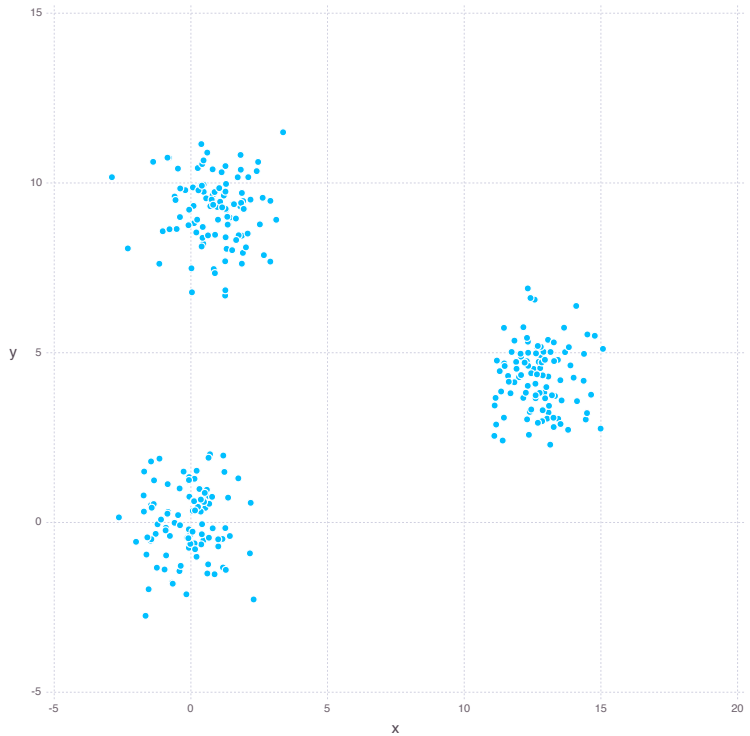
kernel matrix

KPCA on toy data: arc (2)

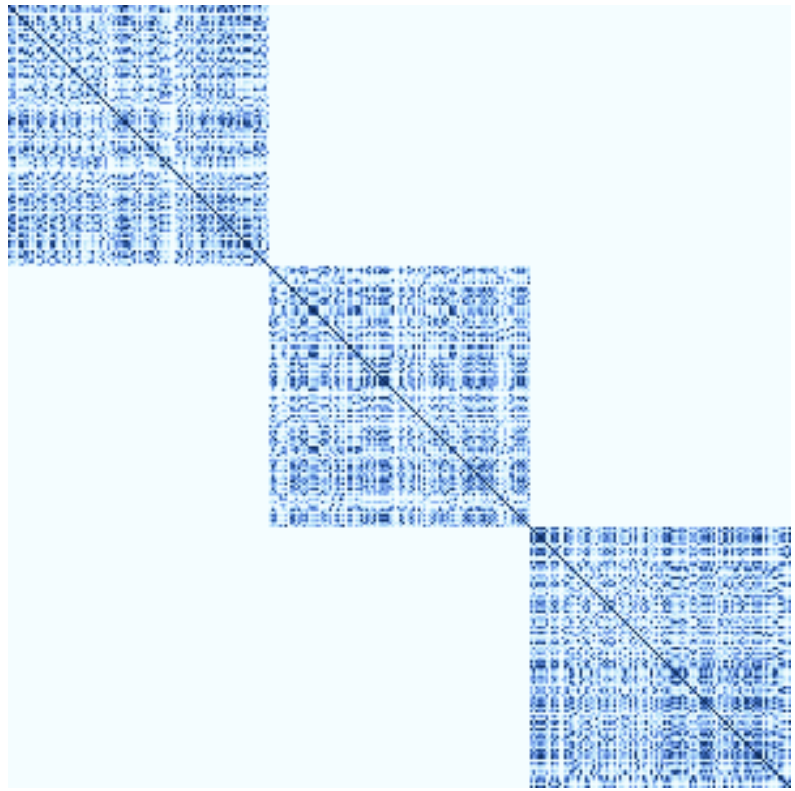


- 1st direction goes nicely along the arc
- 2nd direction looks like first squared, but it is orthogonal in feature space!
- kpca does not provide a 2nd direction locally orthogonal in input space!

KPCA on toy data: cluster (1)

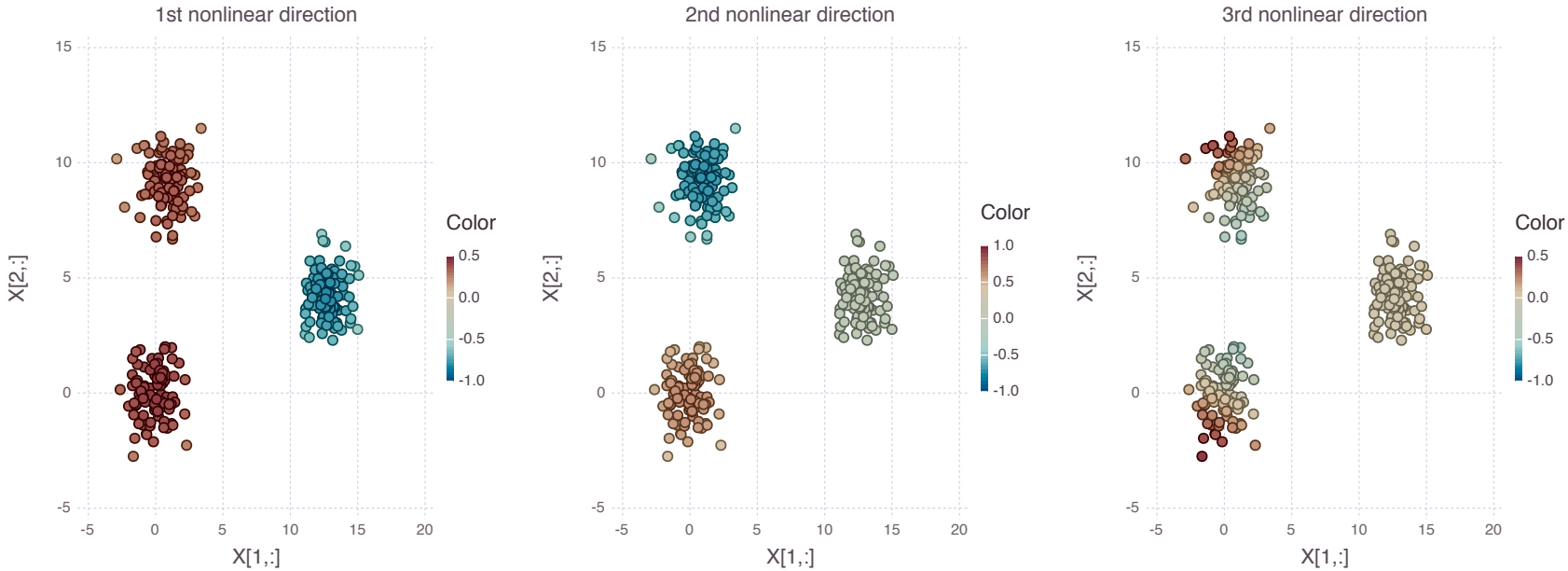


data



kernel matrix

KPCA on toy data: cluster (2)



- 1st direction splits off one cluster
- 2nd direction splits all, also orthogonal in feature space
- kpca is closely related to *spectral clustering*

Is there a third way possibility to do PCA?

So far two approaches:

1. PCs are the Eigenvectors of the covariance matrix XX^T
2. PCs can be calculated from the Eigenvectors of the Gram mtarix X^TX (this allows kPCA)

Approach number three

- do not compute the covariance matrix
- do not compute the Gram matrix
- directly calculate the SVD of the data matrix $X \in \mathbb{R}^{d \times n}$:

$$X = V \Lambda U^T$$

and project the data onto the left-singular vectors V
(which are the eigenvectors of XX^T)

PCA and KPCA Summary

Definition 12.9 (Dimensionality reduction)

Given n data points $X = [x_1, \ldots, x_n] \in \mathbb{R}^{D \times n}$. Find a low dimensional representation $Z = [z_1, \ldots, z_n] \in \mathbb{R}^{d \times n}$ with $d \ll D$ that keeps most of the properties of the higher dimensional data.

What properties should be kept?

- keeping the variance/minimizing MSE in input space (PCA)
- keeping the variance/minimizing MSE in feature space (kPCA)

Three possible solutions:

1. PCs are the Eigenvectors of the covariance matrix XX^T
2. PCs can be calculated from the Eigenvectors of the Gram mtarix $X^T X$ (this allows kPCA)
3. PCs are the left-singular vectors of the data matrix X

End of the PCA Section