Kernel Machines

From linear to Kernel SVM

Gilles Gasso
Benoit Gaüzère - Stéphane Canu

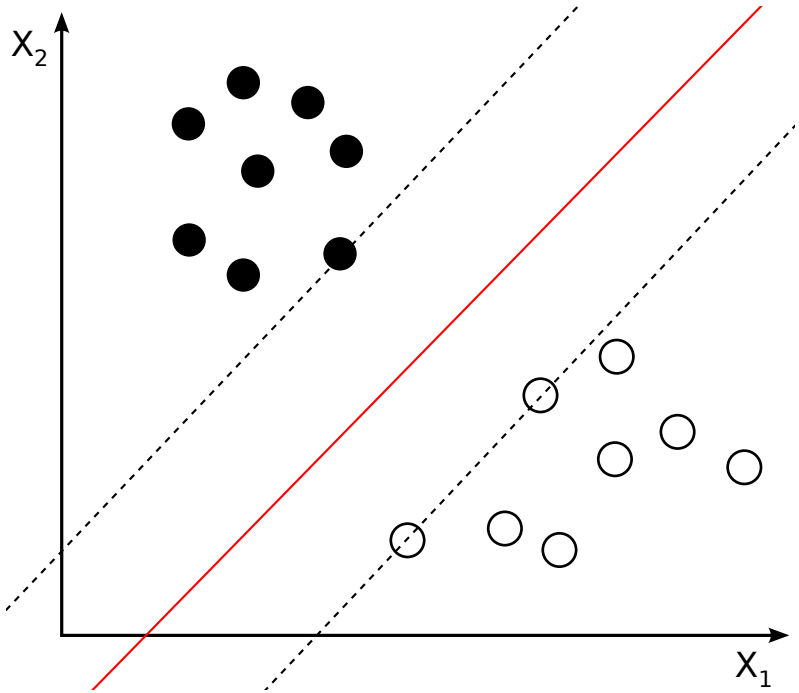
INSA Rouen - ISIA Department
Laboratory LITIS

November 1, 2025

Road map

1. Intuition and Motivation
2. Kernel
   - Theoretical framework
3. Kernel functions
   - Kernel on vectors
   - Kernels on generic data
4. Non-Linear Kernel Machine
   - SVM for classification

Linear SVM



- Inputs are vectors of $\mathbb{R}^d$
- Linear SVM seeks linear classification function

Limitations

- Data are not always vectors: (string, time series, graphs, images . . .)
- The decision function can not always be linear (text categorization; email filtering; gene detection; protein classification; prediction of loan defaulting)

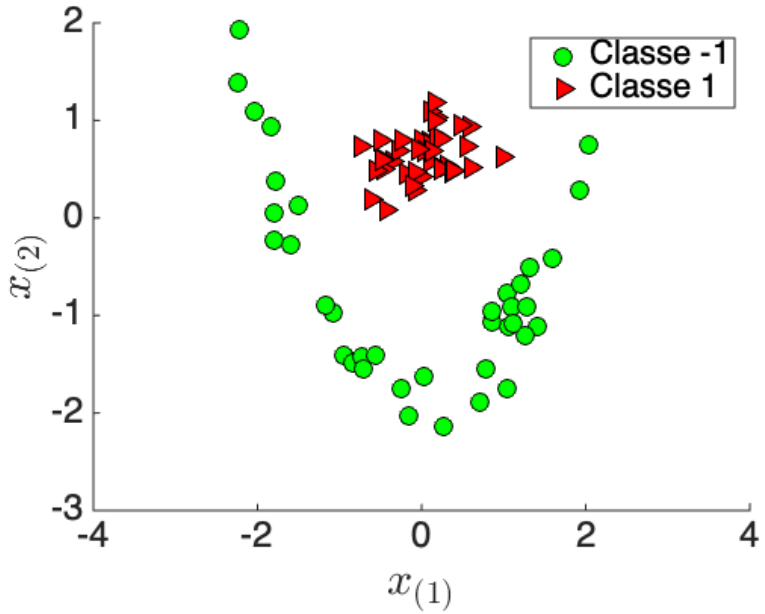
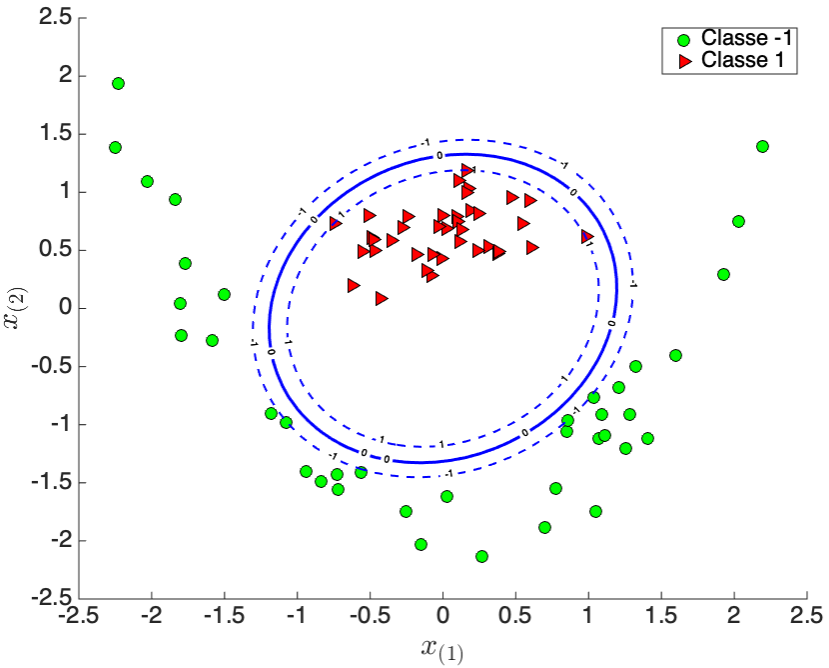


Figure: How to classify these data?

From linear to non-linear decision function

Data might be separable with a non-linear function



- Non-linear embedding of $\mathbf{x} = \begin{pmatrix} x_{(1)} \\ x_{(2)} \end{pmatrix}$

$$\begin{aligned} \mathbb{R}^2 &\rightarrow \mathcal{H} \\ \mathbf{x} &\mapsto \Phi(\mathbf{x}) = \begin{pmatrix} x_{(1)}^2 \\ x_{(2)}^2 \\ \sqrt{2}x_{(1)}x_{(2)} \end{pmatrix} \end{aligned}$$

- Train a linear SVM with samples $\{(\Phi(\mathbf{x}_i), y_i)\}$

Resulting SVM model

$$f(\mathbf{x}) = b + \sum_{i \in SV} \alpha_i y_i \Phi(\mathbf{x}_i)^\top \Phi(\mathbf{x})$$

Non-linear decision function

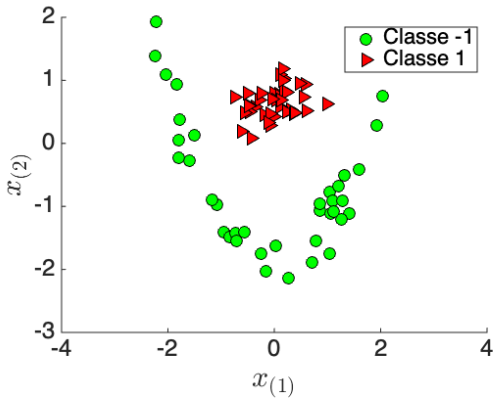
Decision function

$$f(\mathbf{x}) = b + \sum_{i \in SV} \alpha_i y_i \underbrace{\Phi(\mathbf{x}_i)^\top \Phi(\mathbf{x})}_{\text{Kernel } k(\mathbf{x}_i, \mathbf{x})}$$

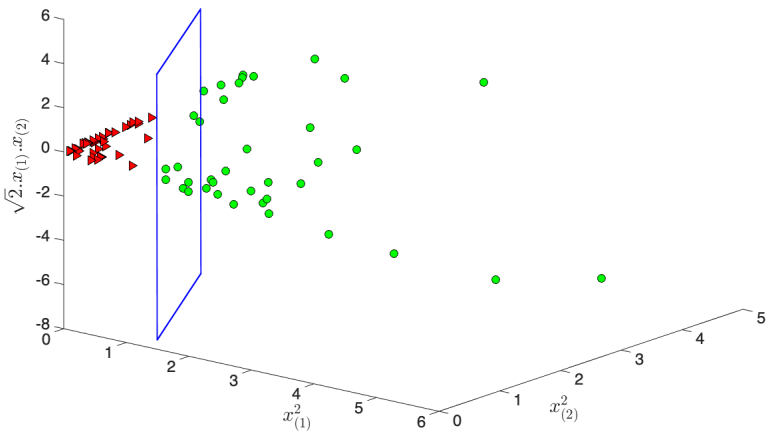
Kernel function: the trick

- No explicit knowledge of $\Phi(\mathbf{x})$
- We only need to define a function $k(\cdot, \cdot) : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$

Problem linearly non-separable in the original space $\mathcal{X}$

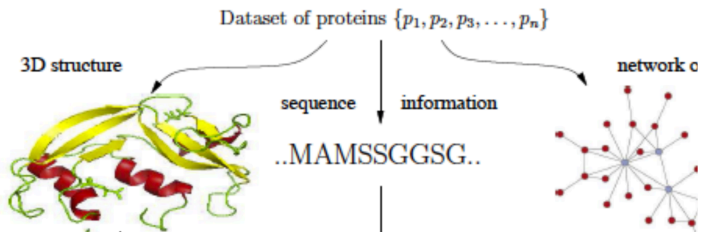


but linear separable in the space $\mathcal{H}$ induced by the kernel k

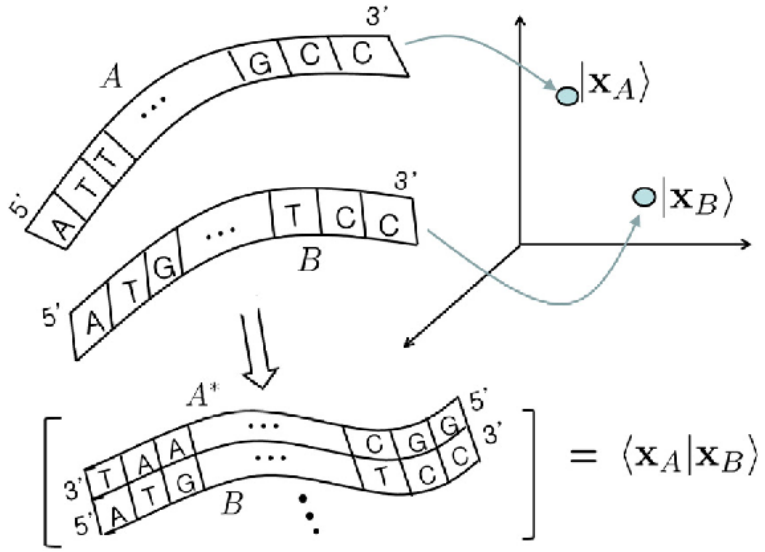


Also...

How do we classify dataset composed of proteins?



Use the kernel trick: $f(\text{protein}) = b + \sum_{i \in SV} \alpha_i y_i k(\text{protein}_i, \text{protein})$



Remark

Intuition

By simply modifying the dot product, SVM works in another space

Kernel Trick

1. Linear SVM relies on inner product between the Support Vectors and the sample to predict
2. $\rightarrow$ Replaces the inner product between the sample in the ambient space by a kernel $k(\cdot, \cdot)$
3. $\rightarrow$ Leads to a non-linear version of the SVM

Motivation of Kernel Machine

- From
  - linear techniques
  - operating on vector spaces
- to
  - non linear prediction models
  - operating on various, structured, high-dimensional data
- Using a:
  well known mathematical framework
- leading to:
  efficient and powerful algorithm and tools

$\longrightarrow$ Kernel method

What is a kernel ?

What can be k ?

Prerequisites

Definitions and notations

- $\mathcal{X}$: non empty input space ($\mathbb{R}^N$, graphs, objects, ...)
- $\mathbf{x} \in \mathcal{X}$,
- $\mathcal{H}$: feature space endowed with a dot product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$
- $\Phi : \mathcal{X} \to \mathcal{H}$: embedding function from $\mathcal{X}$ to $\mathcal{H}$

Kernel

Kernel

A kernel k is a function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$:

$$k(\mathbf{x}, \mathbf{z}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{z}) \rangle_{\mathcal{H}}$$

Which kernel for SVM ? $\longrightarrow$ positive definite kernels
Why ? $\longrightarrow$ To ensure a well-defined SVM dual problem

Positive Definite Kernels (1)

Positive definite kernel

A kernel $k(\mathbf{x}, \mathbf{z})$ on $\mathcal{X} \times \mathcal{X}$ is said to be positive definite

- if it is symmetric: $k(\mathbf{x}, \mathbf{z}) = k(\mathbf{z}, \mathbf{x})$
- and if for any finite positive integer n:

$$\forall \{\alpha_i\}_{i=1,n} \in \mathbb{R}, \forall \{\mathbf{x}_i\}_{i=1,n} \in \Omega, \quad \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \geq 0$$

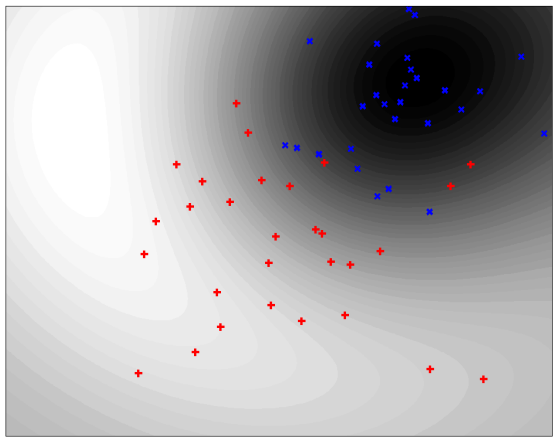
it is strictly positive definite if for $\alpha_i \neq 0$

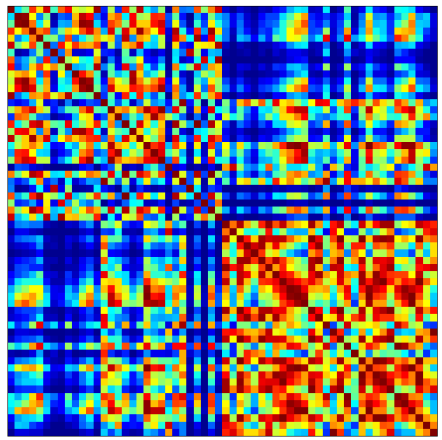
$$\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) > 0$$

Positive Definite Kernels (2)

Gram Matrix

Given a kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, and samples $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, the Gram Matrix $\mathbf{K}$ is a $n \times n$ matrix with entries $\mathbf{K}_{i,j} := k(\mathbf{x}_i, \mathbf{x}_j)$





Another way to characterize positive definite kernel

If for any set of $n \in \mathbb{N}$ samples $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ the associated Gram Matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$ is positive definite, then k is a positive definite kernel on $\mathcal{X}$.

Positive definite Kernels

Linear Kernel

$k(\mathbf{x}, \mathbf{z}) = \mathbf{x}^\top \mathbf{z}$

- $\mathbf{x}, \mathbf{z} \in \mathbb{R}^d$
- symmetric: $\mathbf{x}^\top \mathbf{z} = \mathbf{x}^\top \mathbf{z}$
- positive:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) = \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j \mathbf{x}_i^\top \mathbf{x}_j$$

$$= \left(\sum_{i=1}^{n} \alpha_i \mathbf{x}_i\right)^\top \left(\sum_{j=1}^{n} \alpha_j \mathbf{x}_j\right)$$

$$= \left\| \sum_{i=1}^{n} \alpha_i \mathbf{x}_i \right\|^2$$

Finite kernel

let $\phi_j, j = 1, p$ be a finite dictionary of functions from $\mathcal{X}$ to $\mathbb{R}$ (polynomials, wavelets...)

the feature map and linear kernel

feature map:
$$\begin{array}{rccc} \Phi: & \mathcal{X} & \to & \mathbb{R}^p \\ & \mathbf{x} & \mapsto & \Phi = \big(\phi_1(\mathbf{x}), ..., \phi_p(\mathbf{x})\big) \end{array}$$

Linear kernel in the feature space:
$$k(\mathbf{x}, \mathbf{z}) = \big(\phi_1(\mathbf{x}), ..., \phi_p(\mathbf{x})\big)^\top \big(\phi_1(\mathbf{z}), ..., \phi_p(\mathbf{z})\big)$$

e.g. the quadratic kernel: $\mathbf{x}, \mathbf{z} \in \mathbb{R}^d, \quad k(\mathbf{x}, \mathbf{z}) = \big(\mathbf{x}^\top \mathbf{z} + b\big)^2$

Closed form kernel: the quadratic kernel

Quadratic kernel $k(\mathbf{x},\mathbf{z}) = \left(\mathbf{x}^\top \mathbf{z} + 1\right)^2 = 1 + 2\mathbf{x}^\top \mathbf{z} + \left(\mathbf{x}^\top \mathbf{z}\right)^2$

- $\mathbf{x}, \mathbf{z} \in \mathbb{R}^d$,
- It computes the dot product of the dictionary

$$\begin{array}{rccl} \Phi: & \mathbb{R}^d & \to & \mathbb{R}^{p=1+d+\frac{d(d+1)}{2}} \\ & \mathbf{x} & \mapsto & \Phi = \left(1, \sqrt{2}x_1, \sqrt{2}x_2, ..., \sqrt{2}x_d, x_1^2, x_2^2, ..., x_d^2, ..., \sqrt{2}x_i x_j, ...\right) \end{array}$$

$p = 1 + d + \frac{d(d+1)}{2}$ multiplications *vs.* $d+1$

use kernel to save computation

Gaussian kernel

$$k(\mathbf{x}, \mathbf{z}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{z}\|^2}{2\sigma^2}\right)$$

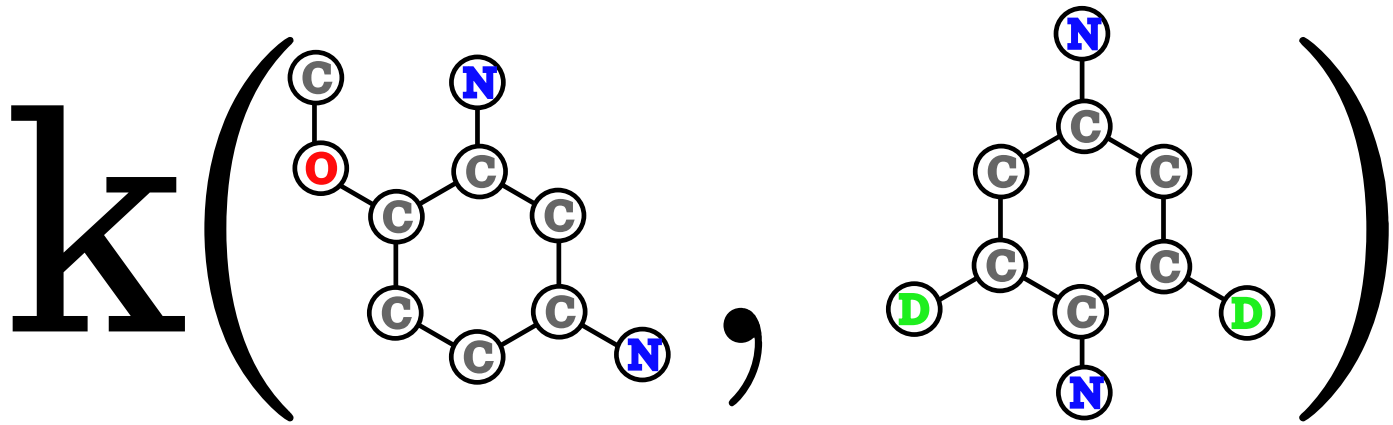
- for $\sigma = 1$:

$$\Phi(\mathbf{x}) = \left(\frac{\exp \frac{\|\mathbf{x}\|^2}{2j}}{\sqrt{j!}^{1/j}} \binom{j}{n_1, \ldots, n_k}^{1/2} \mathbf{x}_1^{n_1} \ldots \mathbf{x}_k^{n_k} \right)_{j=0,\ldots,\infty, \sum_{i=1}^k n_i = j}$$

- Feature space has an infinite dimension
- Overlearning 
- σ controls the influence area of the kernel

Kernels on structures

- $\mathcal{X}$ may not be a vector space.
- we can define kernels on any kind of data :
  - Strings
  - Time series
  - Graphs
  - Images
  - ...

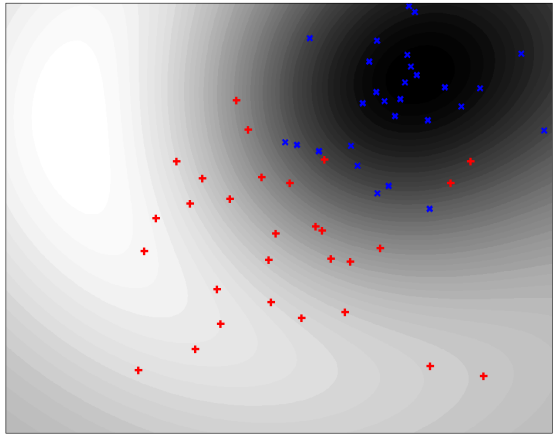


Positive definite kernels: some common examples

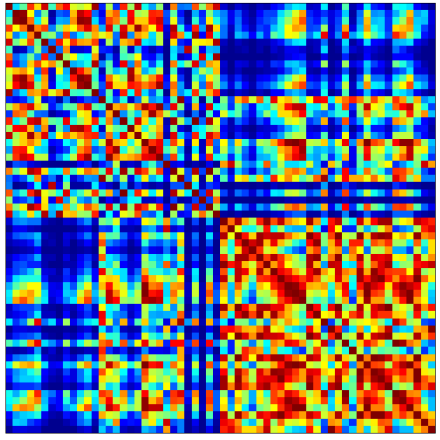
Type	Name	$k(\mathbf{x}, \mathbf{z})$
radial	Gaussian	$\exp\left(-\frac{\|\mathbf{x}-\mathbf{z}\|^2}{2\sigma^2}\right)$
radial	Laplacian	$\exp(-\|\mathbf{x}-\mathbf{z}\|/\sigma)$
non stat.	χ^2	$\exp(-r/\sigma),\ r = \sum_k \frac{(\mathbf{x}_k-\mathbf{z}_k)^2}{\mathbf{x}_k+\mathbf{z}_k}$
projectif	polynomial	$(\mathbf{x}^\top\mathbf{z}+\sigma)^p$
projectif	cosinus	$\mathbf{x}^\top\mathbf{z}/\|\mathbf{x}\|\|\mathbf{z}\|$
projectif	correlation	$\exp\left(\frac{\mathbf{x}^\top\mathbf{z}}{\|\mathbf{x}\|\|\mathbf{z}\|}-\sigma\right)$

- The kernel may involve hyper-parameter(s) to tune (polynom order p, bandwidth σ)
- Their value has to be set by cross-validation

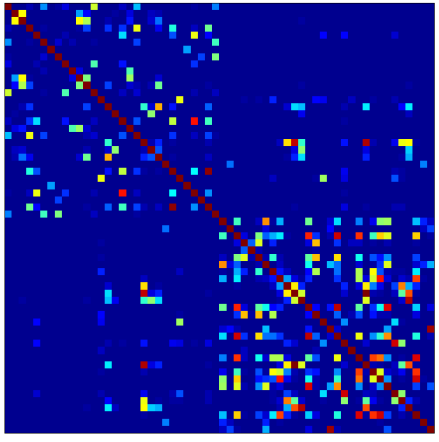
Gram matrices with different bandwidths



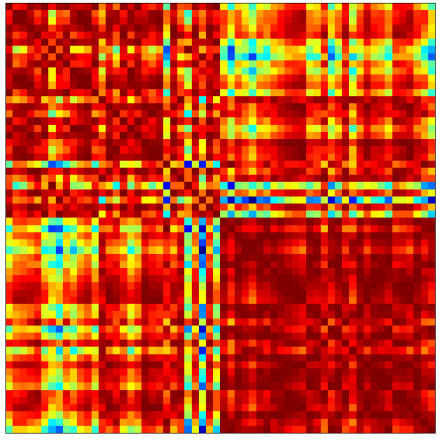
raw data



Gram matrix for $\sigma = 2$



$\sigma = .5$



$\sigma = 10$

Kernel Machine

How to exploit the kernel trick to implement non-linear methods ?

Non-Linear SVM: formulation

- Let $k(\cdot,\cdot)$ be a positive definite kernel inducing the space $\mathcal{H}$
- There exits the mapping function $\Phi : \mathcal{X} \to \mathcal{H}$ defined such that $\forall \mathbf{x}, \mathbf{z} \in \mathcal{X}$ we get $\langle \Phi(\mathbf{x}), \Phi(\mathbf{z}) \rangle_{\mathcal{H}} = k(\mathbf{x}, \mathbf{z})$

Non-Linear SVM: general case

- Dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i) \in \mathbb{R}^d \times \{-1, 1\}\}_{i=1}^n$
- Problem formulation

$$\begin{array}{ll} \min_{\mathbf{w}\in\mathcal{H},b,\{\xi_i\}_{i=1}^n} & \frac{1}{2}\|\mathbf{w}\|_{\mathcal{H}}^2 + C\sum_{i=1}^N \xi_i \\ \text{s.t.} & y_i(\langle \mathbf{w}, \phi(\mathbf{x}_i)\rangle_{\mathcal{H}} + b) \geq 1 - \xi_i \quad \forall i = 1, \cdots, n \\ & \xi_i \geq 0 \quad \forall i = 1, \ldots, n \end{array}$$

Non-Linear SVM: the solution

Dual problem

$$\max_{\{\alpha_i\}} \quad \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \underbrace{\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle_{\mathcal{H}}}_{k(\mathbf{x}_i, \mathbf{x}_j)}$$

$$\text{s.t.} \quad 0 \leq \alpha_i \leq C, \quad \forall\, i = 1, \cdots, N$$

$$\sum_{i=1}^{N} \alpha_i y_i = 0$$

Matrix form

$$\max_{\boldsymbol{\alpha} \in \mathbb{R}^N} \quad -\frac{1}{2} \boldsymbol{\alpha}^\top \mathbf{G} \boldsymbol{\alpha} + \mathbb{1}^\top \boldsymbol{\alpha} \quad \text{s.t.} \quad \mathbf{0} \leq \boldsymbol{\alpha} \leq C\, \mathbb{1}, \boldsymbol{\alpha}^\top \mathbf{y} = 0$$

with $\mathbf{G} \in \mathbb{R}^{n \times n}$ a matrix such that $\mathbf{G}_{ij} = y_i y_j k(\mathbf{x}_i, \mathbf{x}_j)$

$\mathbf{G}$ is positive definite for a positive definite kernel k

Classification function

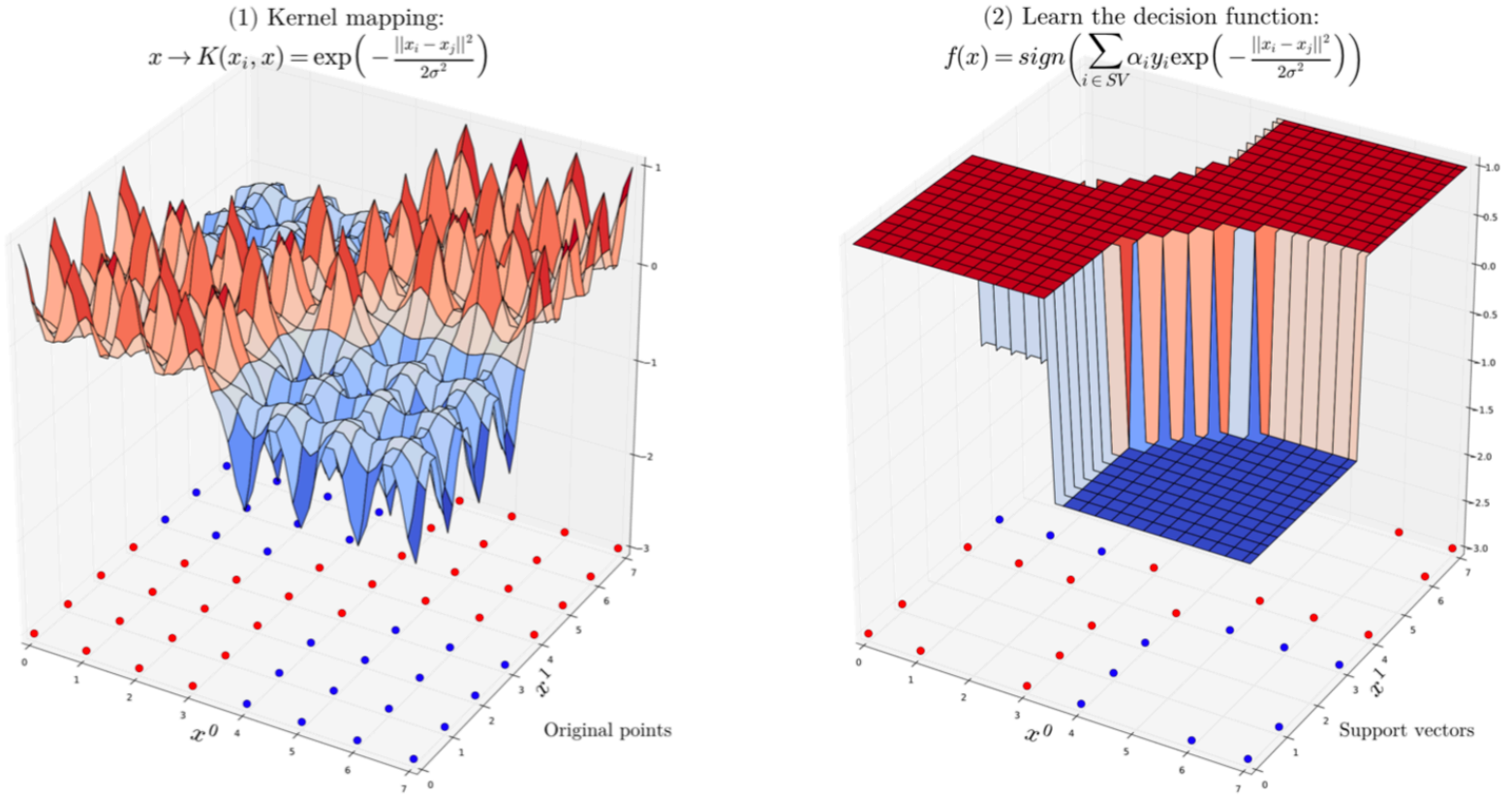
$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b$$

- Linear SVM = SVM with a linear kernel $k(\mathbf{x}, \mathbf{z}) = \mathbf{x}^\top \mathbf{z}$

Illustration: from kernel mapping to decision function

Classification function:

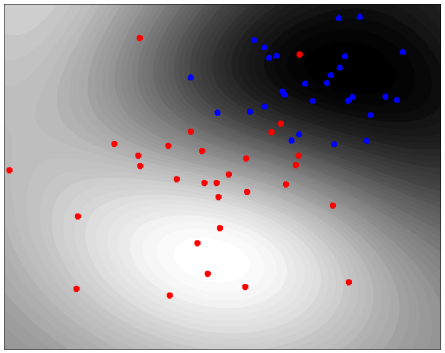
$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b$$

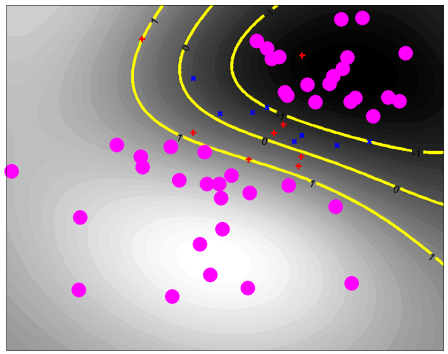


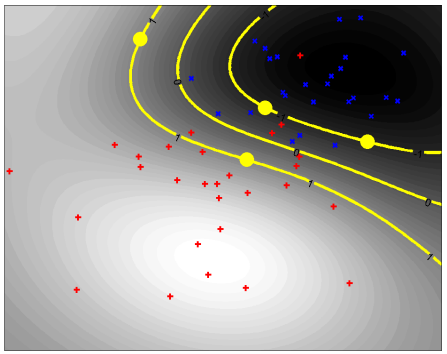
ftp://ftp.cea.fr/pub/unati/people/educhesnay/pystatml/StatisticsMachineLearningPythonDraft.pdf

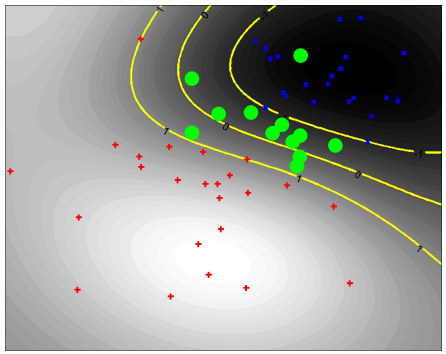
Sparsity of the SVM

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}, \mathbf{x}_i)$$
$$D(x) = \text{sign}\big(f(\mathbf{x}) + b\big)$$





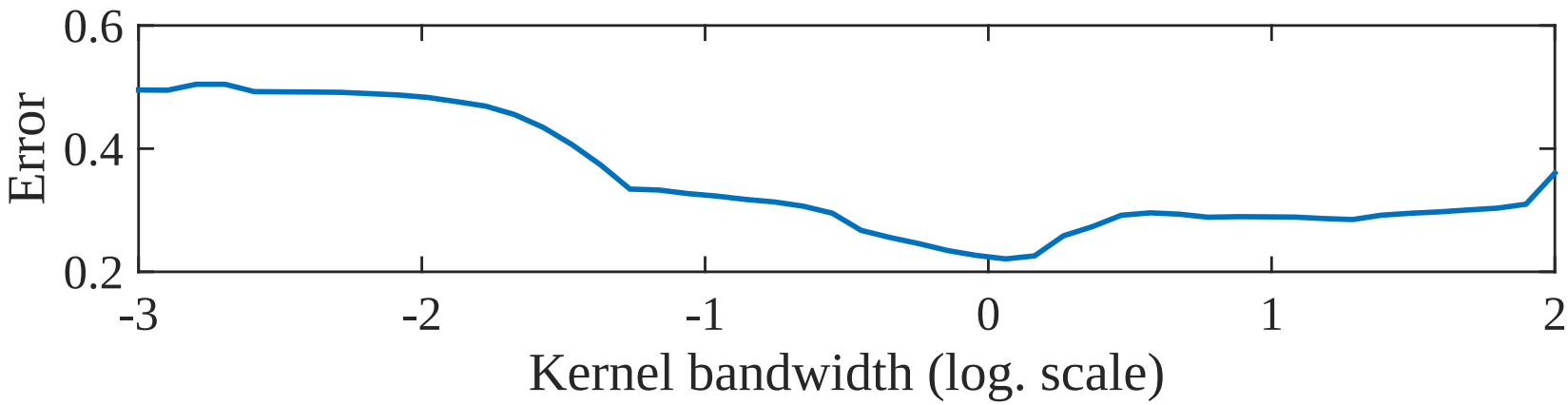


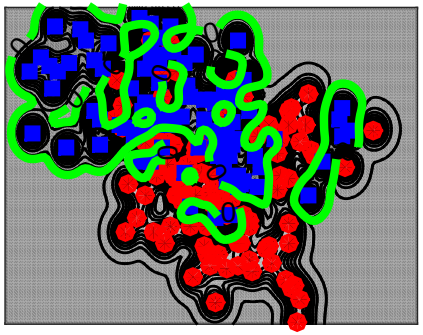


useless data	important data	suspicious data
well classified	support	
$\alpha = 0$	$0 < \alpha < C$	$\alpha = C$

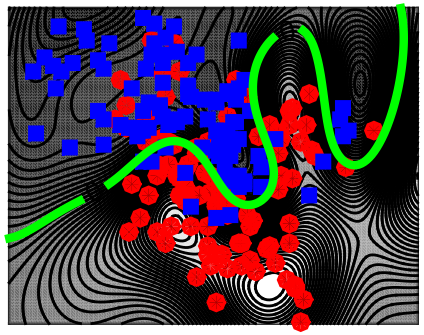
Influence of kernel parameter

Gaussian kernel : $\exp\left(-\frac{\|\mathbf{x}-\mathbf{z}\|^2}{2\sigma^2}\right)$ with bandwidth σ

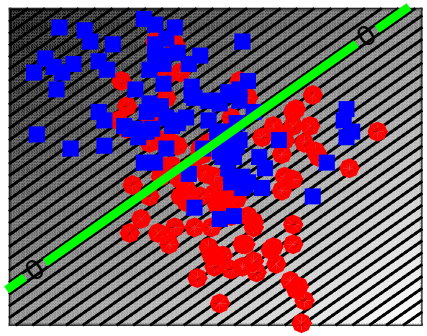




σ too small



nice σ



σ too large

$\longrightarrow$ select σ using cross-validation

Conclusion

What we've seen

- Kernels corresponds to scalar product in some Hilbert space:
  - value corresponds to high dimensional scalar product,
  - on non linear embedding
  - without explicit representations of Φ
- Can be defined on any kind of data provided we are able to define a measure of similarity

Applications

- Algorithms operating on these functions
- Non linear prediction models for classification and regression