# Chapter 2 :: Kernels

Karthikesh Raju
Lab. of Comp. & Info. Sc.
`karthik@james.hut.fi`

2003.02.03

# Reference:


★ Bernhard Schölkopf and Alex Smola, **Learning with Kernels - Support Vector Machines, Regularization, Optimization and Beyond**, MIT Press, Cambridge, MA, 2002, pp 25-60

★ Steve R. Gunn, **Support Vector Machines for Classification and Regression**, Technical Report, Faculty of Engg. and App. Sc., Dept. of ECE. `http://www.isis.ecs.soton.ac.uk/isystems/kernel/`

# Outline

★ Introduction
★ Polynomial Kernels
★ Kernels to Feature Spaces
★ Reproducing Kernel Hilbert Spaces & Mercer Kernels
★ Empirical Kernel Map
★ Examples and Properties of Kernels
★ Conclusions
★ Problems

# Introduction

$$X(\omega_k) = \sum_{n=0}^{N-1} x(n) e^{-j2\pi nk/N}, \quad k = 0, 1, \ldots, N-1$$

- ★ Is a DFT of $x(n)$
- ★ The function $e^{-j2\pi nk/N}$ gives raise to the Fourier operator
- ★ This function can be regarded as **Kernel** of the Fourier Transform.
- ★ **So, what are kernels?**

**Terminology:** *A function* $k$ *which gives rise to an operator* $T_k$ *via*

$$\boxed{(T_k f)(x) = \int_{\mathcal{X}} k(x, x') f(x') dx'}$$

*is called the kernel of* $T_k$

**History:** The term *kernel* was first used in the field of integral operators as studied by Hilbert and others.

**Specific Names:** [1] Reproducing Kernel, admissible kernel, Mercer Kernel, Support Vector Kernel, nonnegative definite kernel, covariance kernel.

---

[1] Only applicable to PD kernels

# Kernels of Interest

★ Here, we are interested in kernels $k$ of the type

$$\begin{aligned} \Phi \quad : \quad & \mathcal{X} \rightarrow \mathcal{H} \\ & x \longrightarrow \mathbf{x} := \Phi(x) \end{aligned}$$

★ i.e Kernels that correspond to dot products in feature spaces $\mathcal{H}$ via a map $\Phi$

$$k(x, x') = \langle \Phi(x), \phi(x') \rangle$$

★ What kind of functions $k(x, x')$ admit such representations?

# Polynomial Kernels

★ Given 2D patterns $\mathcal{X} = \mathbb{R}^2$, consider the nonlinear map

$$\begin{array}{ccc} \Phi : \mathbb{R}^2 & \longrightarrow & \mathcal{H} = \mathbb{R}^3 \\ (x_1, x_2) & \longrightarrow & (x_1^1, x_2^2, x_1 x_2) \end{array}$$

★ This is a collection of product features of degree 2

★ Such *polynomial classification* works for small examples, fails when $N$ is large

★ **Example:** $16 \times 16$ images with a monomial degree $d = 5$ yields a dimension of $10^{10}$ **Impractical !!!**

- Kernels provide methods to compute dot products in higher dimensional spaces without explicitly mapping into these spaces
- Consider the map:

$$\Phi : (x_1, x_2) \quad \rightarrow \quad (x_1^1, x_2^2, x_1 x_2, x_2 x_1)$$

- Dot products in the feature space $\mathcal{H}$ are the form

$$\langle \Phi(x), \Phi(y) \rangle = x_1^2 y_1^2 + x_2^2 y_2^2 + 2 x_1 x_2 y_1 y_2 = \langle x, y \rangle^2$$

- The kernel is the square of the dot product in the input space
- So, in general kernels for polynomials the kernel is computed as

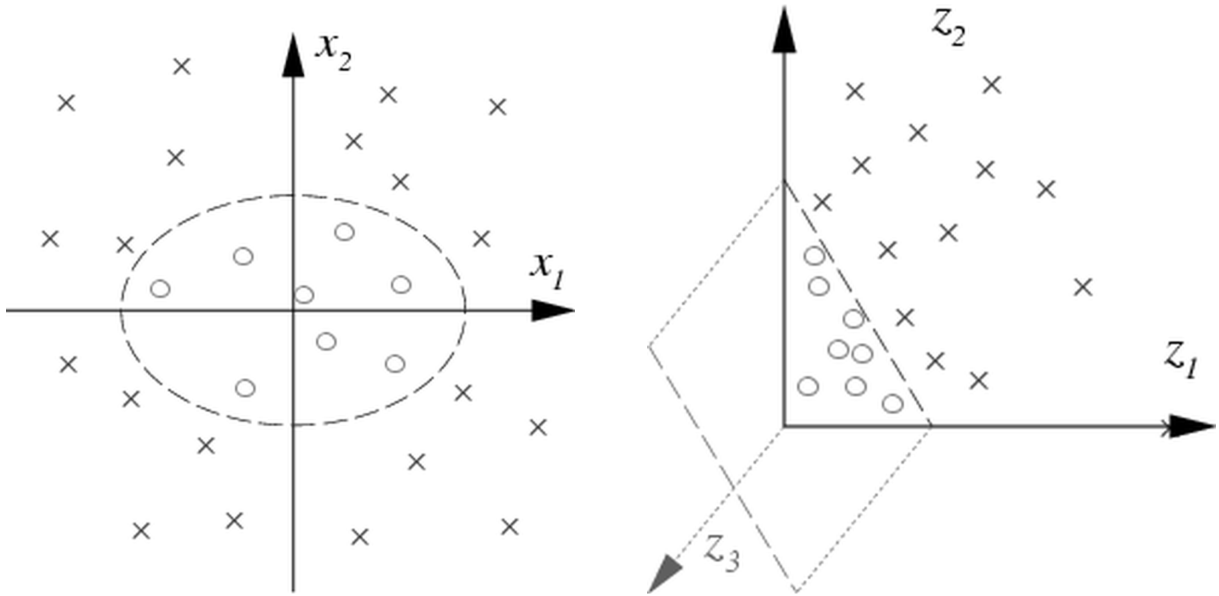
$$k(x, y) = \langle \Phi_d(x), \Phi_d(y) \rangle = \langle x, y \rangle^d$$

- ★ Ordered and unordered polynomial products lead to different maps.
- ★ Multiple occurrences of unordered polynomials are compensated by scaling them with $\sqrt{(d-n+1)!}$, $n$ the number of such occurrences as

$$\Phi_2(x) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

- ★ **Although ordered ($C_d$) and unordered ($\Phi_d$) map into different feature spaces, they are valid instantiations of feature maps for**

$$k(x, y) = \langle x, y \rangle^d$$




True boundary:: Ellipse in the input space

Boundary:: Hyperplane in the feature space

Figure 1: Binary Classification mapped into feature space

## Definitions of Kernelogy

**Gram Matrix:** A function $k : \mathcal{X}^2 \to \mathbb{K}$ and patterns $x_1, \ldots, x_m \in \mathcal{X}$, the $m \times m$ matrix

$$\boxed{K_{ij} = k(x_i, x_j)}$$

is the *Gram matrix* or *Kernel Matrix* of $k$

**PD Matrix:** A complex $m \times m$ matrix $K$ satisfying

$$\boxed{\sum_{i,j} c_i \bar{c}_j K_{ij} \geq 0}$$

for all $c_i \in \mathbb{C}$ is positive definite.

**PD Kernel:** A function $k$ on $\mathcal{X} \times \mathcal{X}$ that gives rise to a positive definite Gram matrix is a pd kernel.

## Additional Points

★ Kernels can be considered as generalized dot products.

★ Linearity of dot products does not carry over to kernels

★ Cauchy-Schwarz inequality can be extended to kernels as

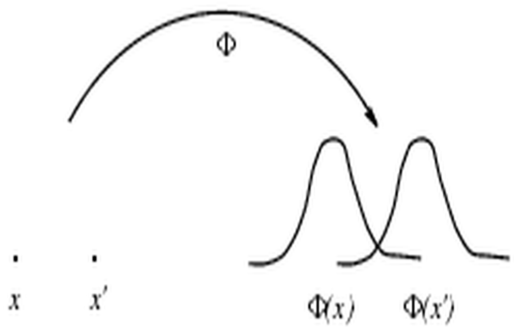
$$\boxed{|k(x, y)|^2 \leq k(x, x)k(y, y)}$$

# Reproducing Kernel Map

- ★ $k$ a real valued, pd kernel, $\mathcal{X}$ a nonempty set.
- ★ Define a map from $\mathcal{X}$ into a space of functions mapping $\mathcal{X}$ to $\mathbb{R}$, denoted as $\mathbb{R}^{\mathcal{X}} := \{f : \mathcal{X} \to \mathbb{R}\}$ as

$$\boxed{\begin{aligned} \Phi \quad : \quad & \mathcal{X} \to \mathbb{R}^{\mathcal{X}} \\ & x \to k(., x) \end{aligned}}$$

$\Phi(x)$ denotes the function that assign the value $k(x', x)$ to $x' \in \mathcal{X}$ i.e., $\Phi(x)(.) = k(., x)$




★ Each pattern has been turned into a function on domain $\mathcal{X}$

★ Now the pattern is represented by the similarity to all other points in the input domain.

★ To construct a feature space associated with $\Phi$:

- Create a vector space out of the image $\Phi$
- Define a dot product in this space has a strictly pd bilinear form
- See to that it satisfies $k(x, x') = \langle \Phi(x), \Phi(x') \rangle$

- ★ Then this kernel is called **Reproducing Kernel** and the map is **Reproducing Kernel Map**
- ★ It is also possible to define a mapping $\Phi$ from $\mathcal{X}$ into a dot product space and obtain a pd kernel.
- ★ Defines the equivalence of kernels.

## Kernel Trick

**Given an algorithm, formulated in terms of a pd kernel $k$, an alternative algorithm can be constructed by replacing $k$ by another pd kernel $\tilde{k}$**

★ After replacement the dot product operates on $\tilde{\Phi}(x_1), \ldots, \tilde{\Phi}(x_1)$ instead of $\Phi(x_1), \ldots, \Phi(x_1)$

- Example: $k$ is a dot product in the input domain
- However, $k$ and $\tilde{k}$ can be nonlinear algorithms
- **Caution:** Certain algorithm work only subject to additional input conditions on the data
- **Hence, not every conceivable pd kernel will make sense.**

# Reproducing Kernel Hilbert Spaces

$$\Phi : \mathbb{R}^N \longrightarrow \mathcal{H}, \quad \mathbf{x} \longrightarrow k(\mathbf{x}, .)$$

★ These functions were defined in dot product spaces

★ Endowing a norm $||x|| := \sqrt{\langle x, x \rangle}$ , then $\mathcal{H}$ is a RKHS if

- $k$ has the reproducing property

$$\begin{aligned} \langle \Phi, k(x, .) \rangle &= \Phi(x), \quad \forall \Phi \in \mathcal{H} \\ \langle k(x, .), k(y, .) \rangle &= k(x, y) \end{aligned}$$

- $k$ spans $\mathcal{H}$

$$f(x) = \sum_i a_i k(x, x_i)$$

# Mercer Kernel

- ★ Let $k$ be a symmetric real valued kernel such that

$$\boxed{k(x,y) = \sum_{j}^{N_{\mathcal{H}}} \lambda_j \psi_j(x) \psi_j(y)}$$

holds for almost all $(x, y)$

- ★ where $\lambda_j > 0$ the eigen values, $\psi_j$ normalized orthogonal eigen functions i.e $\psi_i \psi_j = \delta_{ij}$
- ★ **$k$ is a Mercer Kernel Map**

# Empirical Kernel Map

★ For a given set $\{z_1, \ldots, z_n\} \subset \mathcal{X}, n \in \mathbb{N}$,

$$\begin{array}{rcl} \Phi_n : \mathbb{R}^N & \longrightarrow & \mathbb{R}^n \\ x \longrightarrow k(., x)|_{\{z_1, \ldots z_n\}} & = & (k(z_1, x), \ldots k(z_n, x))^T \end{array}$$

is the empirical kernel map wrt $\{z_1, \ldots z_n\}$.

★ Evaluation of the kernel map on the training patterns

★ Direct extension of this concept is **Kernel PCA map**

# Examples of kernels

★ Polynomial Kernel

$$k(x, y) = \langle x, y \rangle^d$$

★ Gaussian RBF kernels

$$k(x, y) = \exp(-\frac{||x - y||^2}{2\sigma^2})$$

★ Sigmoid

$$k(x, y) = \tanh(\kappa \langle x, y \rangle + \vartheta)$$

★ Inhomogeneous polynomials

$$k(x, y) = (\langle x, y \rangle + c)^d$$

## Properties

- ★ The above kernels are unitary invariant

$$k(x, y) = k(\mathcal{U}x, \mathcal{U}y), \text{if } \mathcal{U}^T = \mathcal{U}^{-1}$$

where $\mathcal{U}$ is for instance a rotation

- ★ RBF kernels are translation invariant

$$k(x, y) = k(x + x_o, y + y_o) \forall x_o \in \mathcal{X}$$

- ★ Polynomial kernels are invariant under orthogonal transformations of $\mathbb{R}^N$ up to a scaling factor

★ Gram Matrix of a Gaussian RBF kernel is full rank

- Implies $\Phi(x_1), \ldots, \Phi(x_m)$ are linearly independent
- They span the $m$ dimensional subspace of $\mathcal{H}$
- RBKs defined on domains of infinite cardinality, with no a priori restriction of training examples, produces an infinite dimension feature space.
- The data is mapped in a way that smooth and simple estimates are possible.

# Kernel Selection

★ With so many different mappings to choose from, which is the best for a particular application?

★ SVMs can be seen as one framework for comparison of these mappings

★ The upper-bound is provided by SLT, which provides an avenue to compare these kernels

★ The question has remained for a long time and **cross-validation remains the preferred method for kernel selection**

# Conclusions

- ★ Kernels - from the cornerstone of SVM and other Kernel methods
- ★ Permit the computation of dot products in high-dimensional spaces, using functions defined on pairs of input patterns.
- ★ **Kernel trick** allows formation of nonlinear variants of any algorithm cast in terms of dot products.
- ★ Though, any dot product based algorithm can be kernelized care must be taken to choose the kernel, which until now is only through cross validation.

## Problems

- ★ (**2.1 Monomial Features in $\mathbb{R}^2$ •**) Verify (2.9) on page 27
- ★ (**2.33 Translation of a Dot Product •**) Prove (2.79) on page 48
- ★ (**2.35 Polarization Identity ••**) For any symmetric bilinear form $\langle ., . \rangle : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, we have, $\forall x, y \in \mathcal{X}$

$$\boxed{\langle x, y \rangle = \frac{1}{4}(\langle x + y, x + y \rangle - \langle x - y, x - y \rangle)}$$

Now consider the spl. case where $\langle ., . \rangle$ is an

Euclidean dot product and $\langle x - y, x - y \rangle$ is the squared Euclidean distance between $x$ and $y$. Discuss why the polarization identity does not imply that the value of the dot product can be recovered from the distances alone. What else does one need?