ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

# Developments in kernel design

Lluís A. Belanche *

Computer Science School - Dept. of Software
Technical University of Catalonia
Jordi Girona, 1-3 08034, Barcelona, SPAIN

**Abstract**. The aim of this paper is to give a concise overview of kernels, with a special attention to non-standard or heterogeneous data sources (*e.g.* non-numerical or structured data). A second goal is to discuss the world of possibilities that kernel design opens for the principled analysis of special or new application domains. The reader is referred to some of the excellent survey publications –as [1, 2, 3]– for an in-depth coverage.

## 1 Introduction

The kernel function allows learning methods to represent and make use of similarities (rather than explicit vector representations) of objects. Kernel-based methods are a two-blade sword in the sense that the choice of a proper kernel for a given problem is both an open issue and an opportunity to develop better performing solutions by adapting the kernel to the problem. Kernel methods involve the use of positive definite matrices as suitable object descriptors, providing a solid framework in which to represent many types of data, as vectors in $\mathbb{R}^d$, strings, trees, graphs, and functional data, among others [4, 2]. The kernel function is a very flexible container under which to express knowledge about the problem as well as to capture the meaningful relations in input space. Let $k$ be a kernel defined on the space of objects and consider a dataset $D = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$. All the information contained in $D$ is represented as a symmetric positive semi-definite *kernel matrix* $K_{N\times N} = (k_{ij})$, where $k_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

## 2 Preliminaries

A kernel function implicitly defines a map $\phi : \mathcal{X} \to \mathcal{H}$ from an input space of objects $\mathcal{X}$ into some Hilbert space $\mathcal{H}$ (called the *feature space*). The "kernel trick" consists in performing the mapping and the inner product simultaneously by defining its associated kernel function:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle_{\mathcal{H}}\,, \;\; \boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}, \tag{1}$$

where $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ denotes inner product in $\mathcal{H}$. This way it is possible to perform computations (like distances or inner products) in $\mathcal{H}$ without using (or even knowing) the mapping function explicitly. Probably the simplest characterization for a symmetric function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ being a kernel is via the matrix it generates on finite subsets:

*Financial support from the Spanish CICYT project TIN2012-31377 is greatly appreciated.

ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

**Definition 1 (Positive semi-definite function)** *A symmetric function $k$ is positive semi-definite in $\mathcal{X}$ if for every $N \in \mathbb{N}$, and every choice $x_1, \cdots, x_N \in \mathcal{X}$, the matrix $K_{N\times N} = (k_{ij})$, where $k_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, is positive semi-definite (PSD).*

**Theorem 2 (Characterization of kernels)** *A symmetric function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ admits the existence of a map $\phi : \mathcal{X} \to \mathcal{H}$ such that $\mathcal{H}$ is a Hilbert space and $k(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle_{\mathcal{H}}$ if and only if $k$ is a PSD symmetric function.*

There are many equivalent characterizations of the PSD property for real symmetric matrices. Here are some:

**Theorem 3 (Positive semi-definite matrix)** *A real symmetric matrix $A_{N\times N}$ is PSD if and only if all of its eigenvalues are non-negative; equivalently, if and only if all of its leading principal minors are non-negative; equivalently, if and only if there is a PSD matrix $B$ such that $BB^T = A$ (this matrix $B$ is unique and called the square root of $A$); equivalently, if and only if for every $\boldsymbol{c} \in \mathbb{R}^N$, $\boldsymbol{c}^T A\, \boldsymbol{c} \geq 0$.*

The following is a basic example to understand the connection between PSD matrices, inner products and kernels.

**Theorem 4 (General linear kernel)** *If $A_{d\times d}$ is a PSD matrix, then the function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ given by $k(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^T A \boldsymbol{x}'$ is a kernel.*

*Proof.* Since A is PSD we can write it in the form $A = BB^T$. For every $N \in \mathbb{N}$, and every choice $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_N \in \mathbb{R}^d$, we form the matrix $K = (k_{ij})$, where $k_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^T A \boldsymbol{x}_j$. Then for every $\boldsymbol{c} \in \mathbb{R}^d$:

$$\sum_{i=1}^{d}\sum_{j=1}^{d} c_i c_j k_{ij} = \sum_{i=1}^{d}\sum_{j=1}^{d} c_i c_j \boldsymbol{x}_i^T A \boldsymbol{x}_j = \sum_{i=1}^{d}\sum_{j=1}^{d} c_i c_j (B^T \boldsymbol{x}_i)^T (B^T \boldsymbol{x}_j)$$

$$= \left\| \sum_{i=1}^{d} c_i (B^T \boldsymbol{x}_i) \right\|^2 \geq 0. \text{ Note that } \phi(\boldsymbol{z}) = B^T \boldsymbol{z}.$$

### 2.1 Support Vector Machines and other kernel methods

Consider a data set $D = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ with labels $y_i \in \{-1, +1\}$. The function $f_{\text{SVM}}$ computed by a classification Support Vector Machine (SVM) is:

$$f_{\text{SVM}}(\boldsymbol{x}) = \text{sgn}\left( \sum_{i=1}^{N} \alpha_i y_i k(\boldsymbol{x}, \boldsymbol{x}_i) + b \right),\ \boldsymbol{x} \in \mathcal{X} \tag{2}$$

with $\alpha_i \geq 0$. Those $\boldsymbol{x}_i \in D$ for which $\alpha_i > 0$ are called the *support vectors* of $f_{\text{SVM}}$. The role of the kernel $k$ is to make $D$ "sufficiently" linearly separable in the feature space, regardless of the separability in the original input space $\mathcal{X}$.

ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

SVMs for classification and regression are not the only possible methods making use of the kernel trick. Over the years, many other kernel-based methods have gained prominence, both for supervised tasks (such as classification and regression) and unsupervised tasks (such as novelty detection, clustering, and feature extraction). These include the Relevance Vector Machine [5], Gaussian processes [6], Spectral clustering [7], Kernel Linear Discriminant Analysis [8], Kernel Principal Components Analysis [9], Kernel Canonical Correlation Analysis [10], Kernel Independent Component Analysis [11], and many others.

## 3 Specific kernels

### 3.1 Kernels on real vectors

**Definition 5 (Polynomial kernel)** *For a pair of vectors* $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^d$*:*

$$k^{\text{poly}}(\boldsymbol{x}, \boldsymbol{x}') = (a \langle \boldsymbol{x}, \boldsymbol{x}' \rangle + 1)^m, \ m \in \mathbb{N}, a > 0 \tag{3}$$

**Definition 6 (RBF kernel)** *For a pair of vectors* $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^d$*:*

$$k^{\text{RBF}}(\boldsymbol{x}, \boldsymbol{x}') = \exp\left( -\sum_{i=1}^{d} \gamma_i (x_i - x_i')^\beta \right), \gamma_i > 0, \beta \in (0, 2] \tag{4}$$

**Definition 7 (Hyperbolic tangent kernel)** *For a pair of vectors* $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^d$*:*

$$k^{\tanh}(\boldsymbol{x}, \boldsymbol{x}') = \tanh(a \langle \boldsymbol{x}, \boldsymbol{x}' \rangle + b), a > 0, b < 0 \tag{5}$$

**Definition 8 (Triangular kernel)** *For a pair of vectors* $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^d$*:*

$$k^{\text{trian}}(\boldsymbol{x}, \boldsymbol{x}') = \begin{cases} 1 - \frac{||\boldsymbol{x} - \boldsymbol{x}'||}{a} & \text{if } ||\boldsymbol{x} - \boldsymbol{x}'|| \leq a \\ 0 & \text{otherwise} \end{cases}, \ a > 0 \tag{6}$$

**Definition 9 (ANOVA radial basis kernel)** *For a pair of vectors* $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^d$*:*

$$k^{\text{ANOVA}}(\boldsymbol{x}, \boldsymbol{x}') = \left( \sum_{i=1}^{d} \exp\left\{ -\gamma_i (x_i - x_i')^2 \right\} \right)^m, \gamma_i > 0, m \in \mathbb{N} \tag{7}$$

**Definition 10 (Rational quadratic kernel)** *For a pair of vectors* $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^d$*:*

$$k^{\text{quad}}(\boldsymbol{x}, \boldsymbol{x}') = 1 - \frac{||\boldsymbol{x} - \boldsymbol{x}'||^2}{||\boldsymbol{x} - \boldsymbol{x}'||^2 + a}, \ a > 0 \tag{8}$$

**Definition 11 (Canberra kernel)** *For a pair of positive vectors* $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^d$*:*

$$k^{\text{Can}}(\boldsymbol{x}, \boldsymbol{x}') = 1 - \frac{1}{d} \sum_{i=1}^{d} \gamma_i \frac{|x_i - x_i'|}{x_i + x_i'}, \ \gamma_i \in (0, 1] \tag{9}$$

ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

**Definition 12 (Truncated Euclidean kernel)** *For a pair of vectors* $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^d$*:*

$$k^{\text{trun}}(\boldsymbol{x}, \boldsymbol{x}') = \frac{1}{d} \sum_{i=1}^{d} \max \left\{ 0, 1 - \frac{|x_i - x_i'|}{\gamma_i} \right\}, \ \gamma_i > 0 \tag{10}$$

The kernels (6), (9) and (10) are useful in that they could fit some applications better than the normal Euclidean distance and derived kernels (like the RBF kernel). Computational considerations should not be overlooked: the use of the exponential function considerably increases the cost of evaluating the kernel. Hence, kernels not involving this function are specially welcome. In particular, (6) and (10) can be useful when differences greater than a specified threshold have to be ignored. In similarity terms, they model situations where data examples can become more and more similar until they are suddenly indistinguishable. Being compactly supported, they lead to more sparse matrices than those obtainable with other metrics. The kernel in (9) is self-normalised and is multiplicative rather than additive, being specially sensitive to small changes near zero [12]. In the last decade, new promising families of new kernels have been proposed –*e.g.* those based on Wavelets [13] or Chebyshev polynomials [14].

We say that a kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is *translation invariant* (also called stationary) if it has the form $k(\boldsymbol{x}, \boldsymbol{x}') = T(\boldsymbol{x} - \boldsymbol{x}')$, where $T : \mathbb{R}^d \to \mathbb{R}$ is a differentiable function. A kernel is *radial* (also called isotropic) if it has the form $k(\boldsymbol{x}, \boldsymbol{x}') = t(\|\boldsymbol{x} - \boldsymbol{x}'\|)$, where $t : [0, \infty) \to [0, \infty)$ is a differentiable function. Radial kernels fulfill $k(\boldsymbol{x}, \boldsymbol{x}) = t(0)$. As an example, the choices $t(z) = exp(-\gamma z^{\beta}), \gamma > 0$ result in the radial kernels (4), known as the Gaussian RBF kernel ($\beta = 2$) and the Exponential RBF kernel ($\beta = 1$)[1]. Many of these classes of kernels are discussed at length in [15]. An important difference between the Gaussian RBF and triangular kernels is that latter has a compact support, which has the effect of introducing sparsity in the kernel matrix, allowing the kernel computation to benefit from sparse matrix algorithms. Note that all these are actually kernel families that depend on one or more parameters. These parameters should be optimized as part of the training of the learner.

### 3.2 Set kernels

Consider a feature space with one feature for every subset $A \subseteq \{1, \cdots, n\}$ of the input features. For $\boldsymbol{x} \in \mathbb{R}^d$, the feature $A$ is given by $\phi_A(\boldsymbol{x}) = \prod_{i \in A} x_i$. The *all-subsets* kernel is defined by the mapping $\phi : \boldsymbol{x} \to (\phi_A(\boldsymbol{x}))_{A \subseteq \{1, \cdots, n\}}$ and then

$$k(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle = \sum_{A \subseteq \{1, \cdots, n\}} \phi_A(\boldsymbol{x}) \phi_A(\boldsymbol{x}') = \sum_{A \subseteq \{1, \cdots, n\}} \prod_{i \in A} x_i x_i' = \prod_{i=1}^{n} (1 + x_i x_i')$$

[1] The Gaussian is so popular that people simply call it the RBF kernel *par excellence*!

ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

We have the freedom to downplay some features (and thus emphasize others) by introducing weighting factors $w_i \geq 0$ for each feature $i$:

$$\phi_A(\boldsymbol{x}) = \prod_{i \in A} \sqrt{w_i} x_i \qquad \text{and} \qquad k_{\boldsymbol{w}}(\boldsymbol{x}, \boldsymbol{x}') = \prod_{i \in A} \sqrt{w_i} x_i \sqrt{w_i} x'_i = \prod_{i=1}^{n} (1 + w_i x_i x'_i).$$

### 3.3 Graph kernels

These are kernels based on the representation of objects as labeled graphs $G = (\mathcal{V}, \mathcal{E})$, given by a set of vertices $\mathcal{V}$ and a set of edges $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ connecting pairs of vertices. The graph has a "start" vertex $s$ and a "finish" vertex $f$. The idea is to see every edge $e \in E$ as indexing a kernel $k_e(\boldsymbol{x}, \boldsymbol{x}')$.

Let $P_{sf}$ be the set of (directed) paths from $s$ to $f$ and denote by $p = (e_1, \cdots, e_{l_p})$ one of such paths, of length $l_p$. Then,

$$k_G(\boldsymbol{x}, \boldsymbol{x}') = \sum_{p \in P_{sf}} \prod_{i=1}^{l_p - 1} k_{(e_i, e_{i+1})}(\boldsymbol{x}, \boldsymbol{x}')$$

It is important to assume that no vertex has an edge to itself. If the graph has no cycles, all sums are finite. If it has, some sums are infinite: we may need an additional term to ensure convergence of the series. The kernels can be extended according to labeling functions $\Lambda_1 : \mathcal{V} \to \Sigma$ and $\Lambda_2 : \mathcal{E} \to \Sigma$, that assign labels to edges and/or vertices from a prescribed alphabet $\Sigma$ [4].

### 3.4 Generative Kernels

These kernels are adequate when a statistical model for the data is available. A simple example is the kernel based on the Jensen-Shannon divergence:

$$k_H(\mathcal{P}, \mathcal{P}') = \exp\left(-\gamma \left\{ H\left(\frac{\mathcal{P} + \mathcal{P}'}{2}\right) + \frac{1}{2}\left(H(\mathcal{P}) + H(\mathcal{P}')\right)\right\}\right), \ \gamma > 0$$

where $\mathcal{P}, \mathcal{P}'$ are two probability distributions with support in $\mathcal{X}$ and $H$ is Shannon's entropy [16]. A wide family is formed by kernels generated from latent variable models, like mixtures of Gaussians or more generally mixtures of distributions of the exponential family. The generalization to graphical models, such as Hidden Markov Models (HMMs) has also been developed [17]. In these models there is a distinction between the variables, such that some are *observed* while others are *hidden*. A joint kernel is first created for the complete data $(\mathcal{X}, \mathcal{Z})$, comprising both observed $(\mathcal{X})$ and hidden $(\mathcal{Z})$ variables; then a *marginalized kernel* for the visible variables is obtained by taking the expectation with respect to the hidden ones (*i.e.,* by marginalizing them away) [18]:

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sum_{h \in \mathcal{Z}} \sum_{h' \in \mathcal{Z}} p(h|\boldsymbol{x}) \tilde{k}(\boldsymbol{x}, \boldsymbol{x}') p(h'|\boldsymbol{x}') \tag{11}$$

ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

where $\tilde{k}$ is a kernel on the product space $\mathcal{X} \times \mathcal{Z}$. When the prior distributions $p(\boldsymbol{x})$ are represented by HMMs, the posteriors $p(h|\boldsymbol{x})$ can be computed by the forward-backward algorithm [19]. *Fisher kernels* derived from latent variable models [20] can be cast as special cases of marginalized kernels of the form (11).

### 3.5 Convolution Kernels

Convolution kernels [21] define a kernel between composite objects by building on kernels defined on their respective parts. Given two such objects $\boldsymbol{x}, \boldsymbol{x}'$ and a way $R(\boldsymbol{x})$ to obtain all possible decompositions into $P$ parts $\hat{x}_1, \ldots, \hat{x}_P$ (this set has to be finite), their R-convolution is a kernel:

$$k_R(\boldsymbol{x}, \boldsymbol{x}') = \sum_{\hat{\boldsymbol{x}} \in R(\boldsymbol{x})} \sum_{\hat{\boldsymbol{x}}' \in R(\boldsymbol{x}')} \prod_{p=1}^{P} k_p(\hat{x}_p, \hat{x}'_p)$$

### 3.6 Kernels as similarity measures

In addition to the formal perspective, a kernel can be conceptually regarded as a similarity measure [3] between two data objects, although many kernels do not fulfill the classical properties for a similarity (*e.g.* boundedness)[2]. However, it seems natural to base kernel design on similarity functions, for example on distance-based similarities. Moreover, in many important domains, objects are described by a mixture of variable types. The work of Gower in general similarity measures [22] shows some partial coefficients of similarity for variables of the binary, categorical or real types, that are shown to produce PSD (similarity) matrices; these functions can therefore be seen as valid kernels.

For any two vector objects $\boldsymbol{x}_i, \boldsymbol{x}_j$ to be compared on the basis of feature $k$, a score $s_{ijk}$ is defined, described below. First set $\delta_{ijk} = 0$ when the comparison of $\boldsymbol{x}_i, \boldsymbol{x}_j$ cannot be performed on the basis of feature $k$ for some reason; for example, by the presence of missing values, by the feature semantics, etc; $\delta_{ijk} = 1$ when such comparison is meaningful. If $\delta_{ijk} = 0$ for all the features, then $s(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is undefined. The partial scores $s_{ijk}$ are defined as follows:

*Binary* (dichotomous) variables indicate the presence/absence of a trait, marked by the symbols + and −. Their similarities are given in Table 1 (left).

| | Values of feature $k$ | | | |
|---|---|---|---|---|
| Object $\boldsymbol{x}_i$ | + | + | − | − |
| Object $\boldsymbol{x}_j$ | + | − | + | − |
| $s_{ijk}$ | 1 | 0 | 0 | 0 |
| $\delta_{ijk}$ | 1 | 1 | 1 | 0 |

| Feature no. | #1 | #2 | #3 | #4 |
|---|---|---|---|---|
| Object $\boldsymbol{x}_i$ | 1.0 | 2.0 | 3.0 | 1.0 |
| Object $\boldsymbol{x}_j$ | 1.0 | 3.0 | 3.0 | ? |
| Object $\boldsymbol{x}_l$ | 1.0 | 3.0 | 3.0 | 5.0 |

Table 1: Left: Similarities for dichotomous variables. Right: Example data. The symbol ? denotes a missing value.

[2] In a very general sense, a similarity function $s$ is some function that at least satisfies the *Similarity Principle*: $s(x, x) > s(x, y) \geq 0$ for all $x \neq y$.

ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

*Categorical* variables can take a number of discrete values, which are commonly known as *modalities*. For these variables no order relation can be assumed. Their *overlap* is $s_{ijk} = 1$ if $x_{ik} = x_{jk}$ and $s_{ijk} = 0$ if $x_{ik} \neq x_{jk}$.

*Real-valued* variables are compared with the standard metric in $\mathbb{R}$: $s_{ijk} = 1 - |x_{ik} - x_{jk}|/R_k$, where $R_k$ is the *range* of feature $k$ (the difference between the maximum and minimum values). The overall coefficient of similarity is defined as the average score over all partial comparisons:

$$s_{ij} = s(\boldsymbol{x}_i, \boldsymbol{x}_j) = \frac{\sum_{k=1}^{n} s_{ijk}\delta_{ijk}}{\sum_{k=1}^{n} \delta_{ijk}}$$

Gower proves that, if there are no missing values, the similarity matrix $S = (s_{ij})$ is PSD. This property may be lost when there are missing values: consider three objects in $[1, 5] \subset \mathbb{R}^4$, that is, $R_k = 4$ as in Table 1 (right). Then we have

$$S = \begin{pmatrix} 1 & \frac{11}{12} & \frac{11}{16} \\ \frac{11}{12} & 1 & 1 \\ \frac{11}{16} & 1 & 1 \end{pmatrix}, \qquad \det(S) = -\frac{121}{2304} < 0$$

and therefore $S$ is not PSD. However, if we replace ? by *any* value in $[1, 5]$, then the matrix $S$ is certainly PSD.

### 3.7 Kernels as dissimilarity measures

Application of eq. (1) allows to (implicitly) compute distances in feature space:

$$||\phi(\boldsymbol{x}) - \phi(\boldsymbol{x}')||_{\mathcal{H}} = \sqrt{\langle\phi(\boldsymbol{x}), \phi(\boldsymbol{x})\rangle_{\mathcal{H}} + \langle\phi(\boldsymbol{x}'), \phi(\boldsymbol{x}')\rangle_{\mathcal{H}} - 2\langle\phi(\boldsymbol{x}), \phi(\boldsymbol{x}')\rangle_{\mathcal{H}}}$$

and then $d_{\mathcal{H}}(\boldsymbol{x}, \boldsymbol{x}') = \sqrt{k(\boldsymbol{x}, \boldsymbol{x}) + k(\boldsymbol{x}', \boldsymbol{x}') - 2k(\boldsymbol{x}, \boldsymbol{x}')}$ is Euclidean. However, given the importance of metric distances (and more generally dissimilarity measures) in data analysis, the definition of distance-based kernels has been an active area of research. Among them, Euclidean metrics have an added appeal linked to our perception of the Euclidean space $\mathbb{R}^d$.

**Definition 13 (Euclidean metric)** *Call $D = (d_{ij})$ a dissimilarity matrix if $d_{ij} = 0$ and $d_{ij} = d_{ji}$. Let $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a metric (distance function); then $d$ is Euclidean if for any positive $N \in \mathbb{N}$ and every choice of objects $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ forming its associated dissimilarity matrix $D_{N\times N} = (d(\boldsymbol{x}_i, \boldsymbol{x}_j))$, there exists a configuration of points $\{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_N\}$ in $\mathbb{R}^M$, $M \leq N$, such that $d_{ij} = ||\boldsymbol{z}_i - \boldsymbol{z}_j||_2$.*

**Definition 14 (Conditionally PSD matrix)** *In the real case, the symmetric matrix $A_{N\times N}$ is called conditionally positive semi-definite (CPSD) if, for all vectors $\boldsymbol{c} \in \mathbb{R}^N$ such that $\sum_{i=1}^{N} c_i = 0$, the inequality $\boldsymbol{c}^T A \boldsymbol{c} \geq 0$ holds.*

There is a close connection between Euclidean metrics and CPSD kernels.

**Theorem 15** *A dissimilarity is Euclidean if and only if the matrix $K = (k_{ij})$ with $k_{ij} = -d_{ij}^2$ is CPSD [23].*

ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

Many common distance measures are non-Euclidean (*e.g.* almost all probabilistic distance measures [24]). When a dissimilarity measure of interest is metric (Euclidean or not), the *kernel trick for distances* [25] can be applied:

If an algorithm is distance-based (rather than inner-product based), the idea is to substitute $||\boldsymbol{x}_i - \boldsymbol{x}_j||$ by the feature space counterpart $||\phi(\boldsymbol{x}_i) - \phi(\boldsymbol{x}_j)||$, and then replace it by $\sqrt{-k(\boldsymbol{x}_i, \boldsymbol{x}_j)}$, where $k(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{x}') \rangle_{\mathcal{H}}$ is any CPSD kernel. Note that necessarily $k(\boldsymbol{x}, \boldsymbol{x}) = 0$ and $k(\boldsymbol{x}, \boldsymbol{x}') \neq 0$ for $\boldsymbol{x} \neq \boldsymbol{x}'$. This process therefore amounts to move from a given metric in input space to a Euclidean metric in feature space and is conformant to Theorem (15). In contrast, the kernel trick for inner products involves substituting $\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$ by the feature space counterpart $\langle \phi(\boldsymbol{x}_i), \phi(\boldsymbol{x}_j) \rangle_{\mathcal{H}}$, and then replace it by $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, where $k$ is any PSD kernel. Since the class of CPSD kernels is larger than that of PSD kernels, a larger set of learning algorithms are prone to kernelization.

## 4 Construction of kernels

Let $\kappa, \kappa'$ be kernels defined on $\mathcal{X}$. Given $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$ and $a_1, a_2 \geq 0$, the function given by $\kappa_+(\boldsymbol{x}, \boldsymbol{x}') = a_1\kappa(\boldsymbol{x}, \boldsymbol{x}') + a_2\kappa'(\boldsymbol{x}, \boldsymbol{x}')$ is a kernel. More generally, when $\kappa, \kappa'$ are kernels defined on $\mathcal{X}, \mathcal{Z}$, respectively, the function given by $\kappa_\oplus((\boldsymbol{x}, \boldsymbol{z}), (\boldsymbol{x}', \boldsymbol{z}')) = \kappa(\boldsymbol{x}, \boldsymbol{x}') + \kappa'(\boldsymbol{z}, \boldsymbol{z}')$ is called a *direct sum* kernel on $\mathcal{X} \times \mathcal{Z}$. In both cases, the operation corresponds to the sum of the corresponding kernel matrices. In addition, the kernel given by $\kappa_\times(\boldsymbol{x}, \boldsymbol{x}') = \kappa(\boldsymbol{x}, \boldsymbol{x}')\kappa'(\boldsymbol{x}, \boldsymbol{x}')$, is called a *pointwise product* (or product) kernel. Analogously, when $\kappa, \kappa'$ are kernels defined on $\mathcal{X}, \mathcal{Z}$, the function given by $\kappa_\otimes((\boldsymbol{x}, \boldsymbol{z}), (\boldsymbol{x}', \boldsymbol{z}')) = \kappa(\boldsymbol{x}, \boldsymbol{x}')\kappa'(\boldsymbol{z}, \boldsymbol{z}')$ is a *tensor product* kernel on $\mathcal{X} \times \mathcal{Z}$. In these cases, the operation corresponds to the *element-wise* product of the corresponding kernel matrices.

The fact that the closure properties of kernels include both sums and products allows to design more powerful kernels, for example through polynomial combinations with positive coefficients. Indeed, if $k$ is a kernel and $p$ is a polynomial of degree $m$ with positive coefficients, then the function $k_p(\boldsymbol{x}, \boldsymbol{x}') = p(k(\boldsymbol{x}, \boldsymbol{x}'))$ is also a kernel. The special case where $k$ is linear and $p(z) = (az + 1)^m$ leads to the kernel in (3). As a more developed example, consider the kernel family:

$$k_i(\boldsymbol{x}, \boldsymbol{x}') = \{\alpha_i(\langle \boldsymbol{x}, \boldsymbol{x}' \rangle + a_i)^{\beta_i},\ \beta_i \in \mathbb{N}, \alpha_i > 0, a_i \geq 0\}$$

For any positive $q \in \mathbb{N}$, $\sum_{i=0}^{q} k_i(\boldsymbol{x}, \boldsymbol{x}')$ is a kernel. Consider the particular case $a_i = 0, \beta_i = i$ and $\alpha_i = \frac{\alpha^i}{i!}$, for some real $\alpha > 0$, and take the limit $q \to \infty$. The obtained series is convergent for all $\alpha$ and the resulting kernel is:

$$\sum_{i=0}^{\infty} \frac{\alpha^i}{i!}(\langle \boldsymbol{x}, \boldsymbol{x}' \rangle)^i = e^{\alpha \langle \boldsymbol{x}, \boldsymbol{x}' \rangle}$$

What is the feature map here? For simplicity, assume that $x, x' \in \mathbb{R}$; then $\exp(\alpha x x') = \langle \phi(x), \phi(x') \rangle$ with $\phi(z) = \left( \sqrt{\frac{\alpha^i}{i!}} z^i \right)_{i=0}^{\infty}$, and therefore we have designed a feature space of infinite dimension.

ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

We might wonder how general is this result: if $k$ is a kernel, under what condition is $f(k(\boldsymbol{x}, \boldsymbol{x}'))$ a kernel? The following result gives a sufficient condition:

**Theorem 16** *Let $A = (a_{ij})$ be a PSD matrix. If $f$ is an analytic function with positive radius of convergence $R > |a_{ij}|$ and all the coefficients in its power series expansion are non-negative, then the matrix $f(A) := (f(a_{ij}))$ is PSD [26].*

A last important operation is normalization. If $k$ is a kernel, then so is:

$$k_n(\boldsymbol{x}, \boldsymbol{x}') = \frac{k(\boldsymbol{x}, \boldsymbol{x}')}{\sqrt{k(\boldsymbol{x}, \boldsymbol{x})}\sqrt{k(\boldsymbol{x}', \boldsymbol{x}')}}. \text{ Moreover, } |k_n(\boldsymbol{x}, \boldsymbol{x}')| \leq 1 \text{ and } k_n(\boldsymbol{x}, \boldsymbol{x}) = 1.$$

## 5 Discussion and conclusions

Using kernel-based methods requires users to make two choices. First, choosing how to represent the objects (this is inherent to most statistical pattern recognition methods) and second, selecting adequate kernels acting on these representations. These kernels should capture meaningful similarities between objects, in that similar objects should tend to belong to the same class (or have a similar target value, in the regression case). Note that the opposite is not true: sharing the same class or having a similar numerical target value should not necessarily imply a high similarity.

We would like to close this brief excursion into kernels with two general considerations and two suggestions for future research. The first consideration deals with the importance of designing kernels that do not constitute explicit inner products between objects, and therefore fully exploit the kernel trick. The second is the possibility of learning the kernel function (or the kernel matrix) from the training data. The first suggestion is the need for more research in the design of kernels for special situations –like missing values [27], imprecise values ("lower than", "approximately") or not-applicable (NA) values. The second suggestion deals with theoretical analyses on the implications of the kernel choice for the success of kernel-based methods.

## References

[1] T. Hofmann, B. Schölkopf and A. J. Smola. Kernel methods in machine learning. *The Annals of Statistics*, 36(3):1171-1220, 2008.

[2] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[3] B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, 2002.

[4] T. Gärtner. A survey of kernels for structured data. *ACM SIGKDD Explorations Newsletter*, 5(1): 49-58, 2003.

[5] M.E. Tipping. Bayesian Learning and the Relevance Vector Machine. *Journal of Machine Learning Research*, 1:211-244, 2001.

[6] C. Williams and C. Rasmussen. Gaussian Processes for Regression. *Advances in Neural Information Processing*, 8, 1995.

ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2013, i6doc.com publ., ISBN 978-2-87419-081-0. Available from http://www.i6doc.com/en/livre/?GCOI=28001100131010.

[7] A. Ng, M. Jordan and Y. Weiss. On Spectral Clustering: Analysis and an Algorithm. *Neural Information Processing Symposium*, 2001.

[8] G. Baudat and F. Anouar. Generalized discriminant analysis using a kernel approach. *Neural computation*, 12(10): 2385-2404, 2000.

[9] B. Schölkopf, A. Smola and K. Müller. Nonlinear Component Analysis as a Kernel Eigenvalue Problem. Neural Computation, 10:1299-1319, 1998.

[10] P. Lai and C. Fyfe. Kernel and nonlinear canonical correlation analysis. *Intl. Journal of Neural Systems*, 10(5): 365-377, 2000.

[11] F. Bach and M. Jordan. Kernel independent component analysis. *Journal of Machine Learning Research*, 3: 1-48, 2002.

[12] Ll. Belanche, J.L. Vázquez and M. Vázquez. Distance-Based Kernels for Real-Valued Data. *Studies in Classification, Data Analysis & Knowledge Organization*, Springer, 2007.

[13] L. Zhang, W. Zhou and L. Jiao. Wavelet support vector machine. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(1): 34-39, 2004.

[14] S. Ozer, C. Chen and H. Cirpan. A Set of New Chebyshev Kernel Functions for Support Vector Machine Pattern Classification. *Pattern Recognition*, 44 (7), 1435-1447, 2011.

[15] M. G. Genton. Classes of kernels for machine learning: a statistics perspective. *Journal of Machine Learning Research*, 2: 299-312, 2002.

[16] M. Cuturi, K. Fukumizu, and J.P. Vert. Semigroup kernels on measures. *Journal of Machine Learning Research*, 6: 1169-1198, 2005.

[17] T. Jebara, R. Kondor, and A. Howard. Probability Product Kernels. *Journal of Machine Learning Research*, 5: 819-844, 2004.

[18] K. Tsuda, T. Kin and K. Asai. Marginalized kernels for biological sequences. *Bioinformatics*, 18 (Suppl. 1): S268-275, 2002.

[19] R. Durbin, S. Eddy, A. Krogh and G. Mitchison. *Biological sequence analysis: Probabilistic models of proteins and nucleic acids*. Cambridge University Press, 1998.

[20] T.S. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. *Advances in Neural Information Processing Systems*, 487-493, 1999.

[21] D. Haussler. Convolution kernels on discrete structures. TechRep UCSC-CRL-99-10, 1999.

[22] J.C. Gower. A general coefficient of similarity and some of its properties. *Biometrics* 27: 857-874, 1971.

[23] J.C. Gower and P. Legendre. Metric and Euclidean Properties of Dissimilarity Coefficients. *Journal of Classification*, 3:5-48, 1986.

[24] R. Duin and E. Pękalska. Non-Euclidean Dissimilarities: Causes and Informativeness. E.R. Hancock et al. (Eds.): SSPR & SPR 2010, LNCS 6218, pp. 324-333, 2010.

[25] B. Schölkopf. The kernel trick for distances. *Neural Information Processing Systems* (NIPS), pp. 301-07, 2000.

[26] R. Horn and C. Johnson. *Topics in Matrix Analysis*, Cambridge University Press, 1991.

[27] G. Nebot and Ll. Belanche. A kernel extension to handle missing data. In Bramer, Ellis, Petridis (Eds.) *Research and Development in Intelligent Systems XXVI*, 2010.