Machine Learning — Statistical Methods for Machine Learning

Kernel functions

Instructor: *Nicolò Cesa-Bianchi* version of March 8, 2026

Linear predictors may potentially suffer from a large approximation error because they are always described by a number of coefficients which can not be larger than the number of features. A popular technique to reduce this bias is feature expansion, which adds new parameters by constructing new features through nonlinear combinations of the base features. Formally, this can be viewed in terms of a function $\boldsymbol{\phi} : \mathbb{R}^d \to \mathcal{H}$ mapping data points $\boldsymbol{x} \in \mathbb{R}^d$ to a higher-dimensional space $\mathcal{H}$. By training a linear predictor on a feature-expanded training set, one actually learns a more complex nonlinear predictor in the original space.

For example, consider the quadratic feature-expansion map $\boldsymbol{\phi} : \mathbb{R}^2 \to \mathbb{R}^6$ defined by $\boldsymbol{\phi}(x_1, x_2) = \big(1, x_1^2, x_2^2, x_1, x_2, x_1x_2\big)$. Recall that a homogeneous hyperplane in $\mathbb{R}^6$ with coefficients given by $\boldsymbol{w} = (w_1, \dots, w_6)$ is the set of points $\big\{\boldsymbol{z} \in \mathbb{R}^6 \,:\, \boldsymbol{w}^\top\boldsymbol{z} = 0\big\}$. Now note that $\boldsymbol{w}^\top\boldsymbol{\phi}(\boldsymbol{x}) = w_1 + w_2x_1^2 + w_3x_2^2 + w_4x_1 + w_5x_2 + w_6x_1x_2$. Sets of the form $\big\{\boldsymbol{x} \in \mathbb{R}^2 \,:\, w_1 + w_2x_1^2 + w_3x_2^2 + w_4x_1 + w_5x_2 + w_6x_1x_2 = 0\big\}$ describe second-degree curves on the plane $\mathbb{R}^2$. These surfaces include ellipses, parabolas, and hyperbolas.

In general, we may consider polynomial feature expansion maps $\boldsymbol{\phi} : \mathbb{R}^d \to \mathcal{H}$, where $\mathcal{H} \equiv \mathbb{R}^N$, that use monomial features of the form $\phi(\boldsymbol{x})_{\boldsymbol{k}} = \prod_{s=1}^d x_s^{k_s}$ where

$$\boldsymbol{k} \in \mathcal{M}(d,n) = \Big\{(k_1, \dots, k_d) \in \mathbb{N}^d \,:\, k_1 + \cdots + k_d \le n\Big\}$$

is the set of monomial feature indices (the previous example is a special case for $d = 2$ and $n = 2$). Now consider the classifier $h : \mathbb{R}^d \to \{-1, 1\}$ defined by

$$h(\boldsymbol{x}) = \operatorname{sgn}\big(\boldsymbol{w}^\top\boldsymbol{\phi}(\boldsymbol{x})\big) \qquad \text{where} \quad \boldsymbol{w}^\top\boldsymbol{\phi}(\boldsymbol{x}) = \sum_{\boldsymbol{k} \in \mathcal{M}(d,n)} w_{\boldsymbol{k}}\, \phi(\boldsymbol{x})_{\boldsymbol{k}}$$

This nonlinear classifier in $\mathbb{R}^d$ corresponds to a linear classifier in the space $\mathbb{R}^N$ where

$$N = \big|\mathcal{M}(d,n)\big| = \sum_{k=0}^n \binom{d+k-1}{k} = \binom{d+n}{n}$$

This implies that N is exponential in the degree n, and computing $\boldsymbol{\phi}$ becomes infeasible even for moderately large n.

This computational barrier can be fully sidestepped using the so-called **kernel trick**, which can be applied to many algorithms for learning linear predictors. For example, recall the Perceptron update rule $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + y_t\boldsymbol{x}_t$ where $\boldsymbol{w}_1 = \boldsymbol{0}$. Then, linear classifiers learned through the Perceptron are of the form

$$h(\boldsymbol{x}) = \operatorname{sgn}\left(\sum_{s \in S} y_s \boldsymbol{x}_s^\top \boldsymbol{x}\right)$$

where S is the set of indices s of training examples $(\boldsymbol{x}_s, y_s)$ on which the Perceptron made an update. If we run the Perceptron in the space $\mathbb{R}^N$, the linear classifier h becomes

$$h_{\boldsymbol{\phi}}(\boldsymbol{x}) = \operatorname{sgn}\left(\sum_{s\in S} y_s\, \boldsymbol{\phi}(\boldsymbol{x}_s)^\top \boldsymbol{\phi}(\boldsymbol{x})\right) .$$

Hence, in order to compute $h_{\boldsymbol{\phi}}(\boldsymbol{x})$ quickly we need a way of computing quickly each term $\boldsymbol{\phi}(\boldsymbol{x}_s)^\top \boldsymbol{\phi}(\boldsymbol{x})$. The kernel trick helps us find an efficiently computable kernel function $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ such that

$$K(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{\phi}(\boldsymbol{x}') \qquad \text{for all } \boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^d. \tag{1}$$

For example, the quadratic kernel, corresponding to the quadratic feature-expansion map

$$\boldsymbol{\phi}(x_1, x_2) = \bigl(1, x_1^2, x_2^2, \sqrt{2}\, x_1, \sqrt{2}\, x_2, \sqrt{2}\, x_1 x_2\bigr)$$

is $K(\boldsymbol{x}, \boldsymbol{x}') = \bigl(1 + \boldsymbol{x}^\top \boldsymbol{x}'\bigr)^2$, as one can easily verify (the presence of the $\sqrt{2}$ coefficients, which is needed for the math, does not change the class of linear predictors that are representable using the mapping).

Given a kernel K, we can then write the linear classifier generated by the Perceptron as

$$h_K(\boldsymbol{x}) = \operatorname{sgn}\left(\sum_{s\in S} y_s K(\boldsymbol{x}_s, \boldsymbol{x})\right) . \tag{2}$$

Below here, we give the pseudo-code of the Kernel Perceptron algorithm (run on a stream of data points).

```
Algorithm: **Kernel Perceptron**
Let $S$ be the empty list.
For all $t = 1, 2, \dots$
    1. Get next example $(\boldsymbol{x}_t, y_t)$
    2. Compute $\widehat{y}_t = \operatorname{sgn}\left(\sum_{s\in S} y_s\, K(\boldsymbol{x}_s, \boldsymbol{x}_t)\right)$
    3. If $\widehat{y}_t \neq y_t$ add $t$ to the list $S$
```

The polynomial kernel $K_n(\boldsymbol{x}, \boldsymbol{x}') = \bigl(1 + \boldsymbol{x}^\top \boldsymbol{x}'\bigr)^n$ for all $n \in \mathbb{N}$ generalizes the quadratic kernel defined earlier. Using Newton's binomial theorem, we can explicitely compute the map $\boldsymbol{\phi}$ such that $K_n(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{\phi}(\boldsymbol{x}')$,

$$\bigl(1 + \boldsymbol{x}^\top \boldsymbol{x}'\bigr)^n = \sum_{i=0}^{n} \binom{n}{i} \bigl(\boldsymbol{x}^\top \boldsymbol{x}'\bigr)^i . \tag{3}$$

Now observe that, using the multinomial theorem,

$$\bigl(\boldsymbol{x}^\top \boldsymbol{x}'\bigr)^i = \left(\sum_{j=1}^{d} x_j x_j'\right)^i = \sum_{\substack{\boldsymbol{k}\in\mathbb{N}^d \\ k_1+\cdots+k_d=i}} c_{\boldsymbol{k}} \left(\prod_{s=1}^{d} x_s^{k_s} (x_s^{k_s})'\right)$$

where $c_{\boldsymbol{k}}$ are multinomial coefficients. Therefore,

$$\boldsymbol{\phi}(\boldsymbol{x}) = \left(\sqrt{c'_{\boldsymbol{k}}} \prod_{s=1}^{d} x_s^{k_s} \right)_{\boldsymbol{k} \in \mathcal{M}(d,n)} \tag{4}$$

where $c'_{\boldsymbol{k}}$ is a numerical coefficient resulting from the combination of the binomial theorem with the multinomial theorem. In other words, the feature map $\boldsymbol{\phi} : \mathbb{R}^d \to \mathbb{R}^N$ associated with the polynomial kernel K_n sends each $\boldsymbol{x} \in \mathbb{R}^d$ to a vector whose components are indexed by all monomials of degree at most n and weighted by roots of binomial and multinomial coefficients.

Another type of kernel is the Gaussian kernel,

$$K_\gamma(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{1}{2\gamma}\left\|\boldsymbol{x} - \boldsymbol{x}'\right\|^2\right) \qquad \gamma > 0 \ .$$

In order to derive the map $\boldsymbol{\phi}_\gamma$ associated with K_γ we proceeed as follows,

$$\begin{aligned}
\exp\left(-\frac{1}{2\gamma}\left\|\boldsymbol{x} - \boldsymbol{x}'\right\|^2\right) &= \exp\left(-\frac{1}{2\gamma}\left(\|\boldsymbol{x}\|^2 + \left\|\boldsymbol{x}'\right\|^2\right)\right) \exp\left(\frac{1}{\gamma}\left(\boldsymbol{x}^\top \boldsymbol{x}'\right)\right) \\
&= \exp\left(-\frac{\|\boldsymbol{x}\|^2}{2\gamma}\right) \exp\left(-\frac{\left\|\boldsymbol{x}'\right\|^2}{2\gamma}\right) \sum_{n=0}^{\infty} \frac{1}{n!} \frac{\left(\boldsymbol{x}^\top \boldsymbol{x}'\right)^n}{\gamma^n} \qquad (5)
\end{aligned}$$

where we used the Taylor series expansion $e^x = 1 + x + \frac{x^2}{2!} + \cdots$. A closer look at (5) reveals that the Gaussian kernel is a linear combination of infinitely many polynomial kernels (3) of increasing degree, each weighted by the reciprocal of the factorial of its degree. The parameter γ is a scaling factor for the products $\boldsymbol{x}^\top \boldsymbol{x}'$. Finally, note that $K_\gamma(\boldsymbol{x}, \boldsymbol{x}) = 1$ for all $\boldsymbol{x} \in \mathbb{R}^d$ because $\|\boldsymbol{x} - \boldsymbol{x}\|^2 = 0$.

Note that predictors of the form (2) that utilize Gaussian kernels predict any point $\boldsymbol{x}$ using a linear combination (with coefficients y_t) of Gaussians $e^{-z^2/(2\gamma)}$ centered on $\boldsymbol{x}_s$ for $s \in S$ and evaluated at $\boldsymbol{x}$.

The Gaussian kernel is *universal* in the following sense: for each $\gamma > 0$, for each continuous function $f : \mathbb{R}^d \to \mathbb{R}$, and for each $\varepsilon > 0$, there exists a function g of the form

$$g = \sum_{i=1}^{N} \alpha_i \, K_\gamma(\boldsymbol{x}_i, \cdot)$$

for some $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in \mathcal{X}$, $\alpha_1, \ldots, \alpha_N \in \mathbb{R}$, and $N \in \mathbb{N}$ such that[1]

$$\sup_{\boldsymbol{x} \in \mathcal{X}} \left| f(\boldsymbol{x}) - g(\boldsymbol{x}) \right| \le \varepsilon$$

An important consequence of this fact is that learning algorithms that use Gaussian kernels can be consistent (that is, the expected risk of their predictors converges to the Bayes risk as the sample size grows to infinity).

Given a data space $\mathcal{X}$ (not necessarily $\mathbb{R}^d$) and a symmetric function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ how can we check whether K is a kernel? In other words, we want to know whether there exists a feature map

[1]Technically, we need $\mathcal{X}$ to be a compact subset of $\mathbb{R}^d$.

$\phi_K : \mathcal{X} \to \mathcal{H}_K$, where $\mathcal{H}_K$ is a linear space with inner product $\langle \cdot, \cdot \rangle_K$, such that $\big\langle \phi_K(\boldsymbol{x}), \phi_K(\boldsymbol{x}') \big\rangle_K = K(\boldsymbol{x}, \boldsymbol{x}')$ for all $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$. The space $\mathcal{H}_K$ must be linear because our algorithms build linear predictors using additive updates. Luckily, there is a very simple characterization: K is a kernel if and only if for all $m \in \mathbb{N}$ and for all $\boldsymbol{x}_1, \dots, \boldsymbol{x}_m \in \mathcal{X}$, the $m \times m$ matrix $\boldsymbol{K}$ such that $\boldsymbol{K}_{i,j} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is positive semidefinite. That is, $\boldsymbol{z}^\top \boldsymbol{K} \boldsymbol{z} \ge 0$ for all $\boldsymbol{z} \in \mathbb{R}^m$.

The above result tells us the conditions under which ϕ_K exists for a given K. On the other hand, it does not tell us how ϕ_K looks like. In fact, there is no unique representation of the pair $\big(\phi_K, \mathcal{H}_K\big)$ for a given kernel K. However, it can be shown that if K is a kernel, then we can always represent $\phi_K : \mathcal{X} \to \mathcal{H}_K$ as $\phi_K(\boldsymbol{x}) = K(\boldsymbol{x}, \cdot)$. Note that, in this representation, $\phi_K(\boldsymbol{x})$ is a function from $\mathcal{X}$ to $\mathbb{R}$. Now, since $\mathcal{H}_K$ is a linear space, it contains all linear combinations of $K(\boldsymbol{x}, \cdot)$ for arbitrary choices of the coefficients and of the points $\boldsymbol{x} \in \mathcal{X}$. One can indeed show that any element of $\mathcal{H}_K$ can be written in this way,

$$\mathcal{H}_K \equiv \left\{ \sum_{i=1}^{N} \alpha_i \, K(\boldsymbol{x}_i, \cdot) \; : \; \boldsymbol{x}_1, \dots, \boldsymbol{x}_N \in \mathcal{X}, \; \alpha_1, \dots, \alpha_N \in \mathbb{R}, \; N \in \mathbb{N} \right\}$$

An element $f \in \mathcal{H}_K$ is then any function $f : \mathcal{X} \to \mathbb{R}$ such that

$$f(\boldsymbol{x}) = \sum_{i=1}^{N} \alpha_i \, K(\boldsymbol{x}_i, \boldsymbol{x})$$

for some $\boldsymbol{x}_1, \dots, \boldsymbol{x}_N \in \mathcal{X}$, $\alpha_1, \dots, \alpha_N \in \mathbb{R}$, and $N \in \mathbb{N}$.

A learning algorithm producing linear predictors $g \in \mathcal{H}_K$ becomes nonparametric when K is the Gaussian kernel and N (the number of terms in the sum defining any $g \in \mathcal{H}_K$) is free to grow unbounded as the sample size increases. If, instead, K is a kernel such that ϕ_K maps $\mathcal{X}$ to a finite dimensional space, then any $g \in \mathcal{H}_K$ can be always represented with a fixed number of parameters, and therefore any algorithm choosing predictors from $\mathcal{H}_K$ is parametric.

For linear predictors $\boldsymbol{w} \in \mathbb{R}^d$, we compute $\boldsymbol{w}^\top \boldsymbol{x}$ using the standard notion of Euclidean inner product. For linear predictors $f \in \mathcal{H}_K$, instead, we compute $\big\langle f, \phi_K(\boldsymbol{x}) \big\rangle_K$ using the inner product for $\mathcal{H}_K$. Now, since ϕ_K is the feature map for K, we must have $\big\langle \phi_K(\boldsymbol{x}), \phi_K(\boldsymbol{x}') \big\rangle_K = K(\boldsymbol{x}, \boldsymbol{x}')$. Thus, recalling that the inner product is a bilinear operator, we can write

$$\big\langle f, \phi_K(\boldsymbol{x}) \big\rangle_K = \sum_{i=1}^{N} \alpha_i \big\langle K(\boldsymbol{x}_i, \cdot), \phi_K(\boldsymbol{x}) \big\rangle_K = \sum_{i=1}^{N} \alpha_i \big\langle \phi_K(\boldsymbol{x}_i), \phi_K(\boldsymbol{x}) \big\rangle_K = \sum_{i=1}^{N} \alpha_i \, K(\boldsymbol{x}_i, \boldsymbol{x}) = f(\boldsymbol{x})$$

The equality $\big\langle f, \phi_K(\boldsymbol{x}) \big\rangle_K = f(\boldsymbol{x})$ is known as *reproducing property*. For this reason, $\mathcal{H}_K$ (or, more precisely, an appropriate completion of $\mathcal{H}_K$) is also known as *reproducing kernel Hilbert space* (RKHS).

Next, we see how the inner product $\big\langle f, g \big\rangle_K$ between two arbitrary $f, g \in \mathcal{H}_K$ is computed, where

$$f = \sum_{i=1}^{N} \alpha_i \, K(\boldsymbol{x}_i, \cdot) \qquad \text{and} \qquad g = \sum_{j=1}^{M} \beta_j \, K(\boldsymbol{x}'_j, \cdot)$$

Using once more the bilinearity of inner products,

$$\begin{aligned}
\langle f, g \rangle_K &= \left\langle \sum_{i=1}^{N} \alpha_i \, K(\boldsymbol{x}_i, \cdot), \sum_{j=1}^{M} \beta_j \, K(\boldsymbol{x}'_j, \cdot) \right\rangle_K \\
&= \sum_{i=1}^{N} \alpha_i \left\langle K(\boldsymbol{x}_i, \cdot), \sum_{j=1}^{M} \beta_j \, K(\boldsymbol{x}'_j, \cdot) \right\rangle_K \\
&= \sum_{i=1}^{N} \sum_{j=1}^{M} \alpha_i \beta_j \left\langle K(\boldsymbol{x}_i, \cdot), K(\boldsymbol{x}'_j, \cdot) \right\rangle_K \\
&= \sum_{i=1}^{N} \sum_{j=1}^{M} \alpha_i \beta_j K(\boldsymbol{x}_i, \boldsymbol{x}'_j)
\end{aligned}$$

We can lift to any RKHS the bounds we derived for learning algorithms that can be run with kernels. For instance recall the bound on the number of mistakes provided by the Perceptron convergence theorem,

$$\|\boldsymbol{u}\|^2 \left(\max_t \|\boldsymbol{x}_t\|^2 \right)$$

which holds for any $\boldsymbol{u} \in \mathbb{R}^d$ such that $y_t \boldsymbol{u}^\top \boldsymbol{x}_t \geq 1$ for $t = 1, \ldots, m$.

In a generic RKHS $\mathcal{H}_K$, the linear separator $\boldsymbol{u}$ is some $g \in \mathcal{H}_K$ such that $y_t \, g(\boldsymbol{x}_t) \geq 1$ for $t = 1, \ldots, m$. The squared norm $\|\boldsymbol{x}_t\|^2 = \boldsymbol{x}_t^\top \boldsymbol{x}_t$ becomes $\|\phi_K(\boldsymbol{x})\|_K^2 = \langle K(\boldsymbol{x}, \cdot), K(\boldsymbol{x}, \cdot) \rangle_K = K(\boldsymbol{x}, \boldsymbol{x})$. Finally $\|\boldsymbol{u}\|^2$ is replaced by

$$\|f\|_K^2 = \left\| \sum_{i=1}^{N} \alpha_i \, K(\boldsymbol{x}_i, \cdot) \right\|_K^2 = \left\langle \sum_{i=1}^{N} \alpha_i \, K(\boldsymbol{x}_i, \cdot), \sum_{j=1}^{N} \alpha_j \, K(\boldsymbol{x}_j, \cdot) \right\rangle_K = \sum_{i,j=1}^{N} \alpha_i \alpha_j \, K(\boldsymbol{x}_i, \boldsymbol{x}_j) \ .$$

A more complex linear predictor that can be kernelized is ridge regression, $\boldsymbol{w} = \left(\alpha I_d + X^\top X\right)^{-1} X^\top \boldsymbol{y}$, where X is the $m \times d$ matrix whose rows are the training points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m \in \mathbb{R}^d$ and $\boldsymbol{y} = (y_1, \ldots, y_m)$ is the vector of training labels $y_t \in \mathbb{R}$. Using the identity

$$\left(\alpha I_d + X^\top X\right)^{-1} X^\top = X^\top \left(\alpha I_m + X X^\top\right)^{-1} \tag{6}$$

we can represent the ridge regression predictor in a generic RHKS by $\boldsymbol{y}^\top \left(\alpha I + \boldsymbol{K}\right)^{-1} \boldsymbol{k}(\cdot)$, where $\boldsymbol{K}$ is the $m \times m$ matrix with entries $\boldsymbol{K}_{i,j} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and $\boldsymbol{k}(\cdot)$ is the vector $\left(K(\boldsymbol{x}_1, \cdot), \ldots, K(\boldsymbol{x}_m, \cdot)\right)$ of functions $K(\boldsymbol{x}_t, \cdot) = \langle \phi_K(\boldsymbol{x}_t), \phi_K(\cdot) \rangle_K$. Indeed, using (6),

$$\boldsymbol{w}^\top \boldsymbol{x} = \boldsymbol{y}^\top X \left(\alpha I + X^\top X\right)^{-1} \boldsymbol{x} = \boldsymbol{y}^\top \left(\alpha I + X X^\top\right)^{-1} X \boldsymbol{x}$$

Now, the elements of $XX^\top$ are $\boldsymbol{x}_i^\top \boldsymbol{x}_j$, that in kernel space become $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$. Similarly, the components of $X\boldsymbol{x}$ are $\boldsymbol{x}_i^\top \boldsymbol{x}$, that correspond to $K(\boldsymbol{x}_i, \boldsymbol{x})$ in kernel space. Hence, the ridge regression prediction $\boldsymbol{w}^\top \boldsymbol{x} = \boldsymbol{y}^\top X \left(\alpha I_d + X^\top X\right)^{-1} \boldsymbol{x}$ in kernel space becomes $\langle g, \phi_K(\boldsymbol{x}) \rangle_K = \boldsymbol{y}^\top \left(\alpha I + \boldsymbol{K}\right)^{-1} \boldsymbol{k}(\boldsymbol{x})$.

Linear predictors in RKHS can incur overfitting. For example, by increasing the degree n of a polynomial kernel K_n we reduce the training error because higher-degree curves can be used to separate the training points. If the degree is too high, the predictor will overfit. A similar reasoning applies to Gaussian kernels K_γ. The γ parameter corresponds to the width of the Gaussians centered on training points $\boldsymbol{x}_s$. If γ is small relatively to the typical squared distances $\|\boldsymbol{x}_s - \boldsymbol{x}\|^2$ between training and test points, then the classification of a test point $\boldsymbol{x}$ is essentially determined by the training point closest to it. This implies a training error equal or close to zero, because —similarly to 1-NN classifiers— the training points are never misclassified. Once again, the resulting predictor is likely to overfit. On the other hand, for values of γ that are large with respect to the squared distances $\|\boldsymbol{x}_s - \boldsymbol{x}\|^2$, the Gaussians centered on training points are very wide, and the resulting predictors are similar to k-NN classifiers when k is chosen close to the training set size, which is likely to cause underfitting.

We established that any symmetric function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a kernel if and only if the kernel matrix $\boldsymbol{K}$ is positive semidefinite. This is an important result, because it holds irrespective to the choice of the data space $\mathcal{X}$. We can therefore define kernels on any set $\mathcal{X}$: be it a set of matrices, sequences, trees, graphs, and so on. Kernels can be viewed as a way of encapsulating the data space, offering a uniform interface to a learning algorithm that can be efficiently run in the corresponding RKHS. In order to guide the intuition when designing a kernel function on a given data space $\mathcal{X}$, one should recall that $K(\boldsymbol{x}, \boldsymbol{x}')$ implements an inner product between $K(\boldsymbol{x}, \cdot)$ and $K(\boldsymbol{x}', \cdot)$ in some RKHS. We can then first define a notion of similarity on $\mathcal{X}$, and then adjust it so that arbitrary kernel matrices are always positive semidefinite.