Kernel Methods

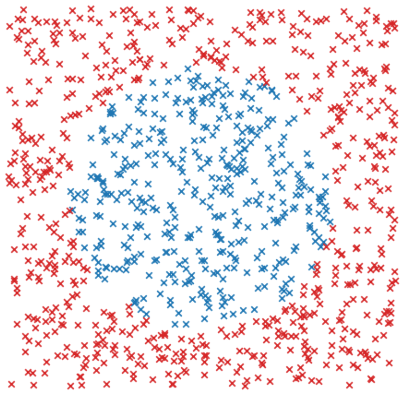
Vincent Divol

1 Linear regression with feature maps

Consider the dataset $\mathbf{x_1}, \dots, \mathbf{x_n}$ presented in Figure . Points inside the ball $B = \{x \in \mathbb{R}^2,\ \|x\| \leq 1\}$ are assigned to -1, whereas points outside B are assigned to $+1$. No linear classifiers will then be able to obtain a good accuracy. A first possibility is to look for more complicated classifiers (composed of intersections of hyperplanes for instance). Another possibility is to "lift" the dataset to a higher dimensional space by considering the map $\Phi : \mathbb{R}^2 \to \mathbb{R}^3$ defined by $\Phi(x) = (x, \|x\|^2)$. The transformed dataset $(\Phi(\mathbf{x_1}), \dots, \Phi(\mathbf{x_n}))$ is displayed in Figure . Remark that this transformed dataset, although in higher dimension than the original dataset, is considerably simpler to classify: there now exists a linear classifier in $\mathbb{R}^3$ that will make exactly zero classification errors on the training set. This toy example showcases an important concept: if a dataset has some complex structure, it may be a good idea to look at a transformation of the dataset into a larger space. In the larger space, the transformed dataset has hopefully a simpler structure, and a basic method (for instance a linear regression) will then have a very good performance. We call such a transformation Φ a **feature map**.

Let us consider another, less artificial example of this phenomenon. In polynomial regression, the goal is to fit a polynomial function of degree d to observations $(\mathbf{x_1}, \mathbf{y_1}), \dots, (\mathbf{x_n}, \mathbf{y_n})$, with $\mathbf{x_i} \in \mathbb{R}$ and $\mathbf{y_i} \in \mathbb{R}$. Let $\mathcal{F}_d$ be the set of polynomial functions of the form $f_a : x \mapsto \sum_{j=0}^d a_j x^j$ where $a = (a_0, \dots, a_d) \in \mathbb{R}^{d+1}$. We are looking for the minimizer of the risk

$$f_a \in \mathcal{F}_d \mapsto \mathcal{R}_n(f_a) := \frac{1}{n} \sum_{i=1}^n |\mathbf{y_i} - f_a(\mathbf{x_i})|^2. \tag{1}$$



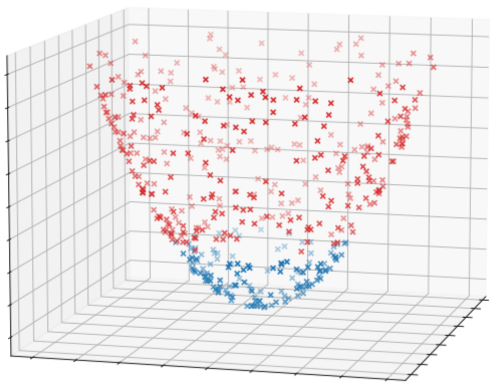


Figure 1: Top: We set $\mathbf{y} = -1$ for points $\mathbf{x}$ inside B, and $\mathbf{y} = +1$ otherwise. No linear classifier can attain good accuracy on this dataset. Bottom: In this new representation, there exists a linear classifier with zero risk (given by the equation $z = 1$).

Let $\Phi_d : \mathbb{R} \to \mathbb{R}^{d+1}$ be defined by $\Phi_d(x) = (1, x, x^2, \dots, x^d)$. Then, $f_a(\mathbf{x_i}) = \langle a, \Phi_d(\mathbf{x_i})\rangle$. It holds that

$$\mathcal{R}_n(f_a) = \frac{1}{n}\sum_{i=1}^n |\mathbf{y_i} - \langle a, \Phi_d(\mathbf{x_i})\rangle|^2 = \frac{1}{n}\|\mathbf{Y} - \tilde{\mathbf{X}}a\|^2,$$

where $\mathbf{Y} = (\mathbf{y_1}, \dots, \mathbf{y_n})^\top \in \mathbb{R}^n$ and $\tilde{\mathbf{X}}$ is the matrix of size $n \times (d+1)$ with ith row given by $\Phi_d(\mathbf{x_i})$. Therefore, minimizing the empirical risk $\mathcal{R}_n$ over $\mathcal{F}_d$ is equivalent to performing a linear regression over the transformed dataset $((\Phi_d(\mathbf{x_1}), \mathbf{y_1}), \dots, (\Phi_d(\mathbf{x_n}), \mathbf{y_n}))$. This is an instance of the same phenomenon: when a linear technique (linear regression) does not perform well, map the dataset in a larger space (here using Φ_d) and apply a linear technique in higher dimension.

Consider now a general feature map $\Phi : \mathcal{X} \to \mathbb{R}^D$, and the associated linear regression problem

$$\frac{1}{n}\sum_{i=1}^n |\mathbf{y_i} - \langle a, \Phi(\mathbf{x_i})\rangle|^2 = \frac{1}{n}\|\mathbf{Y} - \tilde{\mathbf{X}}a\|^2, \tag{2}$$

with $\mathbf{Y}$ defined as before and where $\tilde{\mathbf{X}}$ is the matrix of size $n \times D$ with rows given by $\Phi(\mathbf{x_i})$. Assuming that the matrix $\tilde{\mathbf{X}}^\top\tilde{\mathbf{X}}$ is invertible, the linear regression predictor is given by

$$\hat{a} = (\tilde{\mathbf{X}}^\top\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^\top\mathbf{Y}. \tag{3}$$

Computing this solution requires to compute the matrix $(\tilde{\mathbf{X}}^\top\tilde{\mathbf{X}})^{-1}$. In particular, we need to inverse a matrix of size $D \times D$.[1] This is computationally intractable if D is large (naively, it requires $O(D^3)$ operations). Therefore, it might look like we have found a limitation of the "lifting" approach: if we lift our observations in a space of dimension too large, then minimizing becomes a problem too hard to solve. However, there is a trick that allows us to bypass this limitation. Indeed, remark using (3) that $\hat{a}$ is of the form $\sum_{l=1}^n \hat{b}_l\Phi(\mathbf{x_l})$ for some coefficients $\hat{b}_l \in \mathbb{R}$. Therefore, it suffices to look for vectors a of the form $a = \sum_{l=1}^n b_l\Phi(\mathbf{x_l})$ when looking for the minimum of (2). This is equivalent to minimizing the functional

$$b \in \mathbb{R}^n \mapsto \frac{1}{n}\sum_{i=1}^n |\mathbf{y_i} - \sum_{l=1}^n b_l\langle\Phi(\mathbf{x_i}), \Phi(\mathbf{x_l})\rangle|^2 = \frac{1}{n}\|\mathbf{Y} - \mathbf{G}b\|^2, \tag{4}$$

[1] If the matrix $\tilde{\mathbf{X}}^\top\tilde{\mathbf{X}}$ is not invertible, it can be replaced by its so-called pseudo-inverse, so this discussion applies in both regimes where $D \geq n$ and $n \geq D$.

where $\mathbf{G}$ is the Gram matrix of size $n \times n$, defined by $G_{ij} = \langle \Phi(\mathbf{x_i}), \Phi(\mathbf{x_j}) \rangle$. Computing $\mathbf{G}$ requires a number of computations that is only linear in D, and once it is computed, the complexity of the problem will only depend on the number of observations n. Should $\mathbf{G}$ be invertible, the optimal vector $\hat{b}$ is given by

$$\hat{b} = \mathbf{G}^{-1}\mathbf{Y}. \tag{5}$$

Let us summarize what we have done so far.

- Instead of looking for a complex predictor on the dataset $(\mathbf{x_1}, \ldots, \mathbf{x_n})$, we lift the observations to a higher dimensional space using a feature map $\Phi : \mathcal{X} \to \mathbb{R}^D$.
- In this higher dimensional space, we then look for a simple prediction (e.g. a linear regression, a ridge regression, etc.).
- If the feature space $\mathbb{R}^D$ is very large, then "naive" computations become intractable. However, it turns out that, once the Gram matrix $\mathbf{G}$ of dot products $\langle \Phi(\mathbf{x_i}), \Phi(\mathbf{x_j}) \rangle$ is computed, the complexity of the problem only depends on the number of observations n. Such a remark is referred to as the **kernel trick**.

This last remark is particularly important. It shows that we never need to actually compute the vectors $\Phi(\mathbf{x_i})$, but only the dot products $\mathbf{G}_{ij} = \langle \Phi(\mathbf{x_i}), \Phi(\mathbf{x_j}) \rangle$. For some feature maps Φ, the dimension D is very large (or even infinite!) whereas a very simple expression exists for the dot product, making such an observation particularly appealing.

2 Reproducing Kernel Hilbert Spaces

The Gram matrix $\mathbf{G}$ that we introduced in the previous section is a matrix whose entries $\mathbf{G}_{ij} = \langle \Phi(\mathbf{x_i}), \Phi(\mathbf{x_j}) \rangle$ measure the proximity between the observations $\mathbf{x_i}$ and $\mathbf{x_j}$. If $\mathbf{G}_{ij}$ is large, then $\Phi(\mathbf{x_i})$ and $\Phi(\mathbf{x_j})$ are aligned, meaning that $\mathbf{x_i}$ and $\mathbf{x_j}$ are similar in some sense. On the contrary, if $\mathbf{G}_{ij} = 0$, then $\Phi(\mathbf{x_i})$ is orthogonal to $\Phi(\mathbf{x_j})$ and $\mathbf{x_i}$ and $\mathbf{x_j}$ are very different (at least if we believe that Φ captures relevant information on the observations). This suggests the following bold idea: what if we replace the dot product $\langle \Phi(\mathbf{x_i}), \Phi(\mathbf{x_j}) \rangle$ by a "measure of similarity" $k(\mathbf{x_i}, \mathbf{x_j})$ between $\mathbf{x_i}$ and $\mathbf{x_j}$? For instance, a possible notion of similarity is given by the so-called Gaussian (or RBF) kernel

$$k_\sigma(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma^2}\right) \tag{6}$$

for $x, x' \in \mathbb{R}^d$. Indeed, if x and x' are close, then $k_\sigma(x, x') = 1$ (high similarity), whereas if x and x' are far away, then $k_\sigma(x, x')$ is very small (low similarity).

It turns out that under mild conditions on the function k, we can show that there exists some feature map Φ with $k(x, x') = \langle \Phi(x), \Phi(x') \rangle$. In particular, the discussion of the previous section holds with this feature map Φ. To implement a linear regression, we only need to compute the Gram matrix $\mathbf{G}$, whose entries are given by

$$\mathbf{G}_{ij} = \langle \Phi(\mathbf{x_i}), \Phi(\mathbf{x_j}) \rangle = k(\mathbf{x_i}, \mathbf{x_j}).$$

Therefore, the feature map Φ never needs to be actually computed! Only the values $k(\mathbf{x_i}, \mathbf{x_j})$ have to be computed.

Proving the existence of this feature map Φ requires the introduction of some mathematical machinery. In particular, Φ will in general have outputs living in a space of infinite dimension, that we call a Hilbert space. A Hilbert space $\mathcal{H}$ is a vector space where we define a notion of dot product that behaves like the classical dot product on $\mathbb{R}^D$. This means the following:

- Symmetry: for all $x, y \in \mathcal{H}$, $\langle x, y \rangle_{\mathcal{H}} = \langle y, x \rangle_{\mathcal{H}}$.
- Linearity: for every vectors $x, y, z \in \mathcal{H}$ and $\lambda, \mu \in \mathbb{R}$, we have $\langle x, \lambda y + \mu z \rangle_{\mathcal{H}} = \lambda \langle x, y \rangle_{\mathcal{H}} + \mu \langle x, z \rangle_{\mathcal{H}}$.
- Positive definiteness: for all $x \in \mathcal{H}$, $\langle x, x \rangle_{\mathcal{H}} \geq 0$, with equality if and only if $x = 0$. We write $\|x\|_{\mathcal{H}}$ for $\sqrt{\langle x, x \rangle_{\mathcal{H}}}$. This defines a **norm** on $\mathcal{H}$.
- Completeness: for every continuous linear map $L : \mathcal{H} \to \mathbb{R}$, there exists a vector $h \in \mathcal{H}$ such that $L(x) = \langle h, x \rangle_{\mathcal{H}}$ for every $x \in \mathcal{H}$.

Thise last property might appear mysterious. One can check that it is always satisfied if $\mathcal{H}$ is a Hilbert space of finite dimension (that is $\mathcal{H} = \mathbb{R}^d$). In infinite dimension, completeness ensures that some common mathematical operations (e.g. taking infinite sums or projecting vectors on subspaces) are well-defined.

Example 2.1 (For those interested in math theory). Let us show that in a Hilbert space, infinite sums are well-defined. Let $(h_n)_{n\geq 0}$ be vectors in a Hilbert space $\mathcal{H}$, and assume that the infinite sum of real numbers $\sum_{n\geq 0} \|h_n\|_{\mathcal{H}}$ is finite. Let $L : \mathcal{H} \to \mathbb{R}$ be defined by $L(x) = \sum_{n\geq 0}\langle h_n, x\rangle$. One can check by Cauchy-Schwartz inequality that we have

$$|L(x)| \leq \sum_{n\geq 0} \|h_n\|_{\mathcal{H}}\|x\|_{\mathcal{H}} < \infty.$$

Also, by properties of the sum, L is linear. Therefore, L is a continuous linear map. By completeness, there exists a vector $g \in \mathcal{H}$ such that $L(x) = \langle g, x\rangle$ for every x. By definition, we denote this vector by $\sum_{n\geq 0} h_n$, and this is our definition of the infinite sum! One can then check that with this definition, infinite sums in the Hilbert space $\mathcal{H}$ satisfy the usual properties (e.g. linearity).

An example of vector space $\mathcal{H}$ that is not complete is given by the space of continuous bounded functions on $[0, 1]$, endowed with the dot product $\langle f, g\rangle = \int_0^1 f(t)g(t)\mathrm{d}t$. One can check that the linear map defined by $L(f) = \int_0^{1/2} f(t)\mathrm{d}t$ is continuous. However, there does not exist any continuous function h with

$$L(f) = \int_0^{1/2} f(t)h(t)\mathrm{d}t$$

for every continuous bounded function f. Such a function h should be 1 on $[0, 1/2)$ and 0 on $[1/2, 1]$, and therefore would not be a continuous bounded function. However, the space of L_2 functions on $[0, 1]$ is complete.

Let us fix some set $\mathcal{X}$ and some function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Assume that $k(x, x') = \langle \Phi(x), \Phi(x')\rangle_{\mathcal{H}}$ for some Hilbert space $\mathcal{H}$ and some feature map $\Phi : \mathcal{X} \to \mathcal{H}$. Consider some points $x_1, \dots, x_n \in \mathcal{X}$ and numbers $\lambda_1, \dots, \lambda_n \in \mathbb{R}$. Then,

$$\begin{aligned}0 &\leq \|\sum_{i=1}^{n} \lambda_i\Phi(x_i)\|_{\mathcal{H}}^2\\ &= \sum_{1\leq i,j\leq n} \lambda_i\lambda_j\langle\Phi(x_i), \Phi(x_j)\rangle_{\mathcal{H}}\\ &= \sum_{1\leq i,j\leq n} \lambda_i\lambda_j k(x_i, x_j).\end{aligned}$$

Therefore, to be represented by a feature map, the function k should at least satisfy that

$$\sum_{1\leq i,j\leq n} \lambda_i \lambda_j k(x_i, x_j) \geq 0. \tag{7}$$

It turns out that this condition is also sufficient.

Definition 2.2 (Kernel)**.** *Let $\mathcal{X}$ be a set and $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a function. We say that k is a (positive semi-definite) kernel if for every $n \in \mathbb{N}$, every $x_1, \ldots, x_n \in \mathcal{X}$ and every $\lambda_1, \ldots, \lambda_n \in \mathbb{R}$, condition (7) is satisfied.*

Alternatively, if G is the $n \times n$ matrix with entries $k(x_i, x_j)$, then (7) asserts that the matrix G is positive semi-definite, that is $G \succcurlyeq 0$. We are now in position to state our main theorem.

Theorem 2.3. *Let k be a kernel on a set $\mathcal{X}$. Then, there exists a Hilbert space $\mathcal{H}$ and a feature map $\Phi : \mathcal{X} \to \mathcal{H}$ such that for every $x, x' \in \mathcal{X}$,*

$$k(x, x') = \langle \Phi(x), \Phi(x') \rangle_{\mathcal{H}}. \tag{8}$$

Proof. Let $\mathbb{R}^{\mathcal{X}}$ be the set of functions from $\mathcal{X}$ to $\mathbb{R}$ (which is a vector space). Define a function $\Phi : \mathcal{X} \to \mathbb{R}^{\mathcal{X}}$ by $\Phi(x) = k(x, \cdot)$. Consider the set $\mathcal{H}_0$ obtained as the set of finite sums of the form $\sum_{i=1}^n \lambda_i \Phi(x_i)$ for elements $x_i \in \mathcal{X}$ and $\lambda_i \in \mathbb{R}$. The set $\mathcal{H}_0$ is a vector space, and we can define a dot product on it. Let $f = \sum_{i=1}^n \lambda_i \Phi(x_i)$ and $g = \sum_{j=1}^{n'} \lambda'_j \Phi(x'_j)$. The dot product is defined as

$$\langle f, g \rangle_{\mathcal{H}_0} := \sum_{i=1}^{n} \sum_{j=1}^{n'} \lambda_i \lambda'_j k(x_i, x'_j). \tag{9}$$

One can check that this expression indeed defines a dot product and also does not depend on the choice of expansions of f and g that we have chosen. Furthermore, we have indeed

$$\langle \Phi(x), \Phi(x') \rangle_{\mathcal{H}_0} = k(x, x') \tag{10}$$

for any $x, x' \in \mathcal{H}$. However, the vector space $\mathcal{H}_0$ is not necessarily complete. By using a process called completion, we can actually expand it to a larger space $\mathcal{H}$ that is complete, and that will still satisfy (8). □

Let us give some techniques to construct kernels.

Proposition 2.4. *Let k_1 and k_2 be kernels on $\mathcal{X}$.*

1. *The sum $k_1 + k_2$ is a kernel.*
2. *The product $k_1 \cdot k_2$ is a kernel.*
3. *If $\mathcal{X} \subset \mathbb{R}^d$ and $k(x, x')$ is of the form $k(x - x')$, then k is a kernel if its Fourier transform*

$$\mathcal{F}[k](\xi) = \int \exp(-2\pi i \langle \xi, x \rangle) k(x) \mathrm{d}x \tag{11}$$

is nonnegative for every $\xi \in \mathbb{R}^d$.

Proof. Let $x_1, \dots, x_n \in \mathcal{X}$ and $\lambda_1, \dots, \lambda_n \in \mathbb{R}$.

1. We have

$$\begin{aligned}&\sum_{1 \le i,j \le n} \lambda_i \lambda_j (k_1(x_i, x_j) + k_2(x_i, x_j)) \\ &\qquad = \sum_{1 \le i,j \le n} \lambda_i \lambda_j k_1(x_i, x_j) + \sum_{1 \le i,j \le n} \lambda_i \lambda_j k_2(x_i, x_j) \ge 0.\end{aligned} \tag{12}$$

   Therefore, $k_1 + k_2$ is a kernel.

2. Note that the covariance matrix G of a random variable $\mathbf{y} \in \mathbb{R}^n$ is always positive semi-definite (that is we have $G_{ij} = \mathbb{E}[\mathbf{y^{(i)}} \mathbf{y^{(j)}}]$ where $\mathbf{y} = (\mathbf{y^{(1)}}, \dots, \mathbf{y^{(n)}})$). Reciprocally, if G is a positive semi-definite matrix, then there exists a random variable $\mathbf{y} \in \mathbb{R}^n$ with covariance matrix G (take for instance a Gaussian random variable). Consider a random vector $\mathbf{y_1}$ with covariance matrix $G_1 = (k_1(x_i, x_j))_{ij}$ and a random vector $\mathbf{y_2}$ with covariance matrix $G_2 = (k_2(x_i, x_j))_{ij}$, independent from $\mathbf{y_1}$. Then, one can check that the vector $\mathbf{y} = (\mathbf{y_1^{(1)}} \mathbf{y_2^{(1)}}, \dots, \mathbf{y_1^{(n)}} \mathbf{y_2^{(n)}})$ has covariance matrix G with entries $(k_1(x_i, x_j) \cdot k_2(x_i, x_j))_{ij}$. Therefore, G is positive semi-definite and $k_1 \cdot k_2$ is a kernel.

3. We use the inverse Fourier transform formula

$$k(x) = \int \exp(2\pi i \langle \xi, x \rangle) \mathcal{F}[k](\xi) \mathrm{d}\xi.$$

Therefore,

$$\begin{aligned}\sum_{1\le i,j\le n} \lambda_i\lambda_j k(x_i - x_j) &= \sum_{1\le i,j\le n} \lambda_i\lambda_j \int \exp(2\pi i\langle \xi, x_i - x_j\rangle)\mathcal{F}[k](\xi)\mathrm{d}\xi \\ &= \int \sum_{1\le i,j\le n} \lambda_i\lambda_j \exp(2\pi i\langle \xi, x_i - x_j\rangle)\mathcal{F}[k](\xi)\mathrm{d}\xi \\ &= \int \|\sum_{i=1}^{n} \lambda_i \exp(2\pi i\langle \xi, x_i\rangle)\|^2\mathcal{F}[k](\xi)\mathrm{d}\xi \ge 0.\end{aligned} \tag{13}$$

Therefore, k is a kernel.

□

Let us give some examples.

Example 2.5. 1. If Φ is a map from the set $\mathcal{X}$ to a Hilbert space $\mathcal{H}$, then $k(x, x') = \langle \Phi(x), \Phi(x')\rangle_{\mathcal{H}}$ defines a kernel.

2. Polynomial kernels: the function $k(x, x') = \langle x, x'\rangle^\alpha$ for $\alpha \in \mathbb{N}$ is a kernel. This follows by induction from Proposition 2.4.2.

3. The radial basis function (RBF) kernel, or gaussian kernel: the function $k_\sigma(x, x') = \exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right)$ defines a kernel. Indeed, the Fourier transform of $x \in \mathbb{R}^d \mapsto \exp\left(-\frac{\|x\|^2}{2\sigma^2}\right)$ is given by

$$\xi \mapsto \sqrt{2\pi\sigma^2}\exp(-2\pi\sigma^2\|\xi\|^2), \tag{14}$$

which is nonnegative.

3 Kernel Ridge Regression

The general kernel approach can be applied to different algorithms, some that we have already encountered (e.g. PCA or logistic regression) and some that we have not mentionned (kernel SVM, kernel k-means, etc.). We will here present two algorithms that can be "kernelized": kernel ridge regression and kernel PCA. Recall the setting of ridge regression. Let $(\mathbf{x_1}, \mathbf{y_1}), \ldots, (\mathbf{x_n}, \mathbf{y_n})$

be a training sample, with $\mathbf{x_i} \in \mathbb{R}^d$ and $\mathbf{y_n} \in \mathbb{R}$. Ridge regression consists in minimizing the quantity

$$\beta \in \mathbb{R}^d \mapsto \frac{1}{n}\|\mathbf{X}\beta - \mathbf{Y}\|^2 + \lambda\|\beta\|^2, \tag{15}$$

where $\mathbf{X}$ is the $n \times d$ matrix with rows given by the $\mathbf{x_i}$s, $\mathbf{Y} \in \mathbb{R}^n$ is the vector with entries $\mathbf{y_i}$, $\|\beta\|$ is the 2-norm of the vector β and $\lambda > 0$ is a penalization parameter. This function has a unique minimizer which is given by

$$\hat{\beta} = \left(\mathbf{X}^\top\mathbf{X} + n\lambda\mathrm{Id}_d\right)^{-1}\mathbf{X}^\top\mathbf{Y}. \tag{16}$$

Let us kernelize this algorithm. We now assume that the inputs $\mathbf{x_1}, \ldots, \mathbf{x_n}$ are elements of an arbitrary set $\mathcal{X}$, and that we are given a kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Let $\Phi : \mathcal{X} \to \mathcal{H}$ be an associated feature map, such that $\langle\Phi(x), \Phi(x')\rangle_{\mathcal{H}} = k(x, x')$ for every $x, x' \in \mathcal{H}$. We consider predictions of the form $x \mapsto \langle\Phi(x), \beta\rangle_{\mathcal{H}}$ and are looking for a minimizer of the functional

$$\beta \in \mathcal{H} \mapsto \frac{1}{n}\|\tilde{\mathbf{X}}\beta - \mathbf{Y}\|^2 + \lambda\|\beta\|_{\mathcal{H}}^2, \tag{17}$$

where $\tilde{\mathbf{X}}$ is a linear operator from $\mathcal{H}$ to $\mathbb{R}^n$, defined by

$$\tilde{\mathbf{X}}\beta = (\langle\Phi(\mathbf{x_1}), \beta\rangle_{\mathcal{H}}, \ldots, \langle\Phi(\mathbf{x_n}), \beta\rangle_{\mathcal{H}}).$$

Note that in (17) there are two different norms appearing: the euclidean norm, and the Hilbert norm in the penalization term.

Theorem 3.1 (Representer theorem)**.** *The minimum of* (17) *is attained at a vector β of the form $\beta = \sum_{i=1}^n a_i\Phi(\mathbf{x_i})$ for some real numbers $a_1, \ldots, a_n$.*

Proof. Let E be the subspace of $\mathcal{H}$ that is spanned by the vectors $\Phi(\mathbf{x_i})$ and let $E^\perp$ be the orthogonal subspace of E. We can decompose every $\beta \in \mathcal{H}$ into $\beta_1 + \beta_2$, where $\beta_1 \in E$ and $\beta_2 \in E^\perp$. Note that $\langle\Phi(\mathbf{x_i}), \beta_2\rangle_{\mathcal{H}} = 0$ for every $\mathbf{x_i}$ by definition of orthogonality. This implies that $\langle\Phi(\mathbf{x_i}), \beta\rangle_{\mathcal{H}} = \langle\Phi(\mathbf{x_i}), \beta_1\rangle_{\mathcal{H}}$. Also, by orthogonality, it holds that $\|\beta\|_{\mathcal{H}}^2 = \|\beta_1\|_{\mathcal{H}}^2 + \|\beta_2\|_{\mathcal{H}}^2$. This yields

$$\begin{aligned}
&\frac{1}{n}\|\tilde{\mathbf{X}}\beta - \mathbf{Y}\|^2 + \lambda\|\beta\|_{\mathcal{H}}^2 \\
&\quad = \frac{1}{n}\|\tilde{\mathbf{X}}\beta_1 - \mathbf{Y}\|^2 + \lambda\|\beta_1\|_{\mathcal{H}}^2 + \lambda\|\beta_2\|_{\mathcal{H}}^2 \\
&\quad \leq \frac{1}{n}\|\tilde{\mathbf{X}}\beta_1 - \mathbf{Y}\|^2 + \lambda\|\beta_1\|_{\mathcal{H}}^2.
\end{aligned}$$

Therefore, the minimum of (17) is attained at a point β with $\beta_2 = 0$, that is at a point $\beta \in E$. □

Thus, to find the minimum of (17), it suffices to consider vectors β of the form $\sum_{i=1}^n a_i \Phi(\mathbf{x_i})$. It is equivalent to minimize over all $a = (a_1, \ldots, a_n) \in \mathbb{R}^n$ the quantity

$$\begin{aligned}
&\frac{1}{n}\|\sum_{i=1}^n a_i \tilde{\mathbf{X}}\Phi(\mathbf{x_i}) - \mathbf{Y}\|^2 + \lambda \|\sum_{i=1}^n a_i \Phi(\mathbf{x_i})\|_{\mathcal{H}}^2 \\
&= \frac{1}{n}\left\| \sum_{i=1}^n a_i \begin{pmatrix} \langle \Phi(\mathbf{x_1}), \Phi(\mathbf{x_i})\rangle \\ \vdots \\ \langle \Phi(\mathbf{x_n}), \Phi(\mathbf{x_i})\rangle \end{pmatrix} - \mathbf{Y} \right\|^2 + \lambda \sum_{1\le i,j\le n} a_i a_j \langle \Phi(\mathbf{x_i}), \Phi(\mathbf{x_j})\rangle \\
&= \frac{1}{n}\left\| \sum_{i=1}^n a_i \begin{pmatrix} k(\mathbf{x_1}, \mathbf{x_i}) \\ \vdots \\ k(\mathbf{x_n}, \mathbf{x_i}) \end{pmatrix} - \mathbf{Y} \right\|^2 + \lambda \sum_{1\le i,j\le n} a_i a_j k(\mathbf{x_i}, \mathbf{x_j}) \\
&= \frac{1}{n}\|\mathbf{G}a - \mathbf{Y}\|^2 + \lambda a^\top \mathbf{G} a,
\end{aligned} \tag{18}$$

where $\mathbf{G}$ is the Gram matrix with entries $\mathbf{G}_{ij} = k(\mathbf{x_i}, \mathbf{x_j})$. This last expression is convex in the parameter $a \in \mathbb{R}^n$ and is minimized at

$$\hat{a} = (\mathbf{G} + n\lambda \mathrm{Id}_n)^{-1}\mathbf{Y}. \tag{19}$$

Note that computing $\hat{a}$ requires the inversion of a $n \times n$ matrix, which is an expensive operation if done naively (it requires $O(n^3)$ operations). The Gram matrix of size $n \times n$ also has to be stored. Although some numerical tricks exist to speed up the computations, **kernel methods still are mostly useful when the number n of observations is relatively small (say $n \lesssim 10^4$)**. You should always try to implement a simpler linear method before trying a more complex kernel approach.

Example 3.2. We apply kernel ridge regression to a weather forecasting problem (data from *Weather Underground API*). For four years, we are given the weather day $\mathbf{y_i}$ at the ith day of the year $\mathbf{x_i}$. The goal is to predict the weather on a future day. We implement on Python a kernel ridge regression on the data points $(\mathbf{x_i}, \mathbf{y_i})$ using the `sklearn` function `KernelRidge` with $\lambda = 1$, while using the RBF kernel k_σ. In Figure 2, we plot the predictions for different values of σ. The importance of the scale parameter σ is showcased. Two points x, x' are considered similar if $\|x - x'\| \lesssim \sigma$ (that is $k_\sigma(x, x')$ is close to 1), whereas if $\|x - x'\| \gtrsim \sigma$, then the points x and x' are considered different (the kernel $k_\sigma(x, x')$ is small). In our example, the value σ reflects

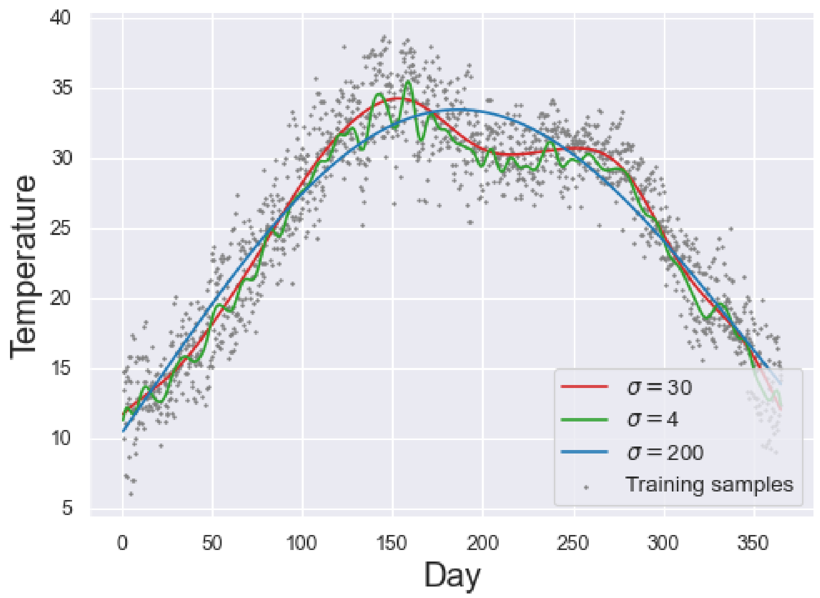


Figure 2: Kernel ridge regression on the weather forecast dataset with the RBF kernel k_σ for different values of σ.

the time scale on which we expect to have a significant temperature change. If σ is chosen too small (say $\sigma = 4$ days), then the prediction will vary quickly, as the model thinks that days that are one week apart could have very different temperatures. On the opposite, if σ is too large (say $\sigma = 200$), then we force the model to consider days that are 200 days apart as similar, leading to serious underfitting. In practice, cross validation should be used to tune the parameter σ.

4 Kernel Principal Component Analysis

Principal Component Analysis aims at finding the best k-dimensional subspace approximating a dataset of points $\mathbf{x_1}, \dots, \mathbf{x_n}$ in $\mathbb{R}^d$. Let $\mathbf{X}$ be the $n \times d$ matrix whose rows are given by the $\mathbf{x_i}$s. The principal components $v_1, \dots, v_k$ of the dataset are found by computing the eigenvectors corresponding to the k largest eigenvalues of the covariance matrix $\mathbf{X}\mathbf{X}^\top$ (of size $n \times n$). One can

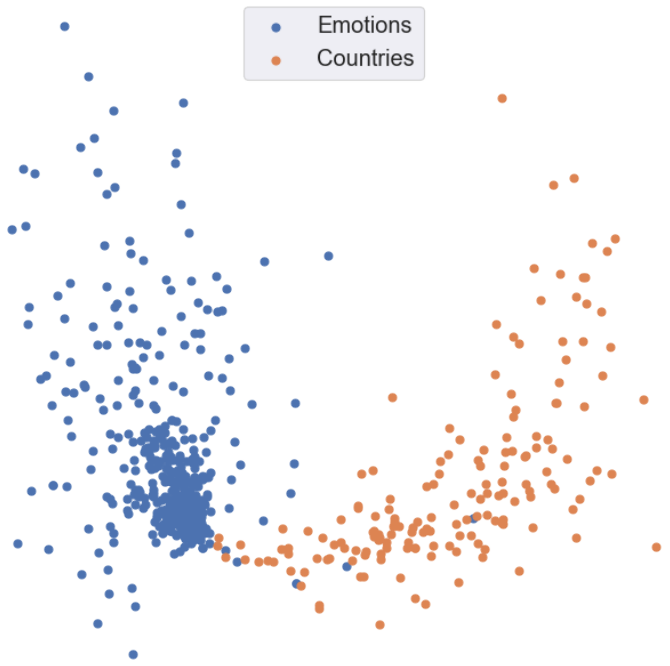


Figure 3: PCA (with two principal components displayed) on the dataset $(\Phi(\mathbf{x_1}), \ldots, \Phi(\mathbf{x_n}))$. Points corresponding to emotions are in blue, while points coresponding to countries are displayed in orange.

then project the data points to the vector space spanned by the k principal components to obtain a representation of the dataset in smaller dimension.

Principal Component Analysis relies on the idea that there exists a linear subspace of low dimension that can explain well the dataset. If this is not the case, one can still hope that a linear subspace of low dimension can be a good approximation of a transformation $\Phi(\mathbf{x_1}), \ldots, \Phi(\mathbf{x_n})$ of the dataset. Kernel PCA consists in applying PCA to this transformed dataset. As before, we can consider a very general framework, where the observations $\mathbf{x_1}, \ldots, \mathbf{x_n}$ lie in a general set $\mathcal{X}$, endowed with a kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. Let $\Phi : \mathcal{X} \to \mathcal{H}$ be an associated feature map.

Computing the principal components of $\Phi(\mathbf{x_1}), \ldots, \Phi(\mathbf{x_n})$ requires to compute the covariance matrix of the corresponding points, which is by definition the matrix with entries $\langle \Phi(\mathbf{x_i}), \Phi(\mathbf{x_j}) \rangle_{\mathcal{H}} = k(\mathbf{x_i}, \mathbf{x_j})$. Therefore, kernel PCA simply consists in computing the eigenvalues and eigenvectors of the Gram matrix $\mathbf{G}$ with entries $\mathbf{G}_{ij} = k(\mathbf{x_i}, \mathbf{x_j})$.

Example 4.1. One of the best features of kernel methods is their versatility: they can be applied not only to vectors, but to inputs living in any set $\mathcal{X}$. As an example, we consider the set of words $\mathcal{X}$. The Word2Vec algorithm provides a framework to learn feature maps from $\mathcal{X}$ to a vector space $\mathbb{R}^d$ [Mikolov et al., 2013]. We use a pretrained model in the `gensim` Python library. This pretrained model gives a ready-to-use feature map $\Phi : \mathcal{X} \to \mathbb{R}^d$ with $d = 25$ that reflects meaningful similarities between different words. To showcase the performance of this feature map, we use a dataset of $n = 785$ words either corresponding to a country (e.g. *bengladesh*, *kenya*, *luxembourg*) or to an emotion (e.g. *awful*, *cruelty*, *displeasure*), see [Sharma, 2017]. Each word $\mathbf{x_i}$ is mapped to the vector $\Phi(\mathbf{x_i})$, and we represent in Figure 3 the projection of the vectors $\Phi(\mathbf{x_i})$ on the plane spanned by the two first principal components of the transformed dataset $(\Phi(\mathbf{x_1}), \ldots, \Phi(\mathbf{x_n}))$. The points corresponding to emotions and countries are clearly separated, showing that the feature map Φ indeed captures the meaning of the words.

References

[Mikolov et al., 2013] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. In Bengio, Y. and LeCun, Y., editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

[Sharma, 2017] Sharma, A. (2017). Sentiment package for r. https://github.com/abhy/sentiment.