Machine Learning and Neural Networks (MATH3431) Epiphany term, 2023

Handout 7: Kernel methods

Lecturer & author: Georgios P. Karagiannis georgios.karagiannis@durham.ac.uk

Aim. To introduce the ideas of learning machines by introducing data into high-dimensional feature spaces for accuracy gains; introduce the kernel trick, and kernel functions.

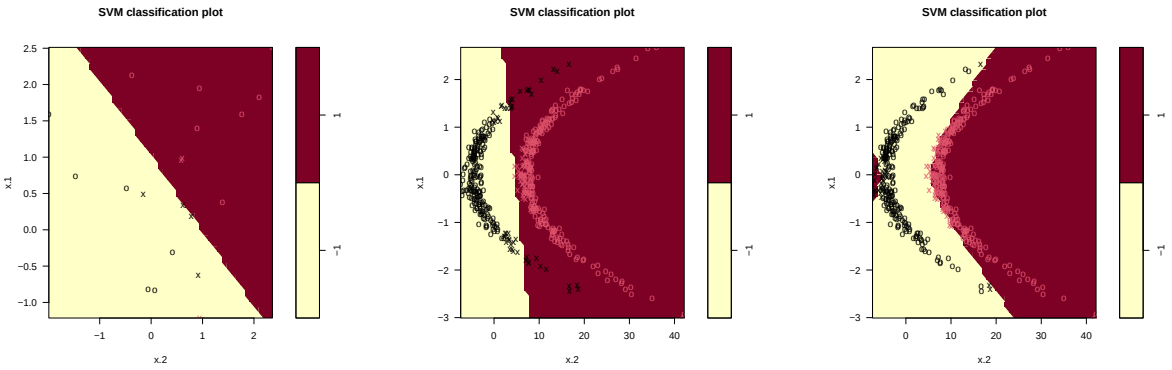
Reading list & references:

(1) Shalev-Shwartz, S., & Ben-David, S. (2014). Understanding machine learning: From theory to algorithms. Cambridge university press.
- Ch. 16.2 Support Vector Machine

(2) Bishop, C. M. (2006). Pattern recognition and machine learning (Vol. 4, No. 4, p. 738). New York: Springer.
- Ch. 6.1, 6.2 Kernel methods

1. Intro and motivation

Note 1. Consider the Soft SVM with predictive rule $h(x) = \text{sign}(\eta(x))$ with separator $\eta(x) = w_1 x_1 + w_2 x_2 + b = \langle w, x \rangle + b$ and $x \in \mathbb{R}^2$. It can address learning problems where the data can (up to some degree of violation) be separated by a line (Figure 1.1a). In more challenging cases where the geometry is strongly non-linear this can totally fail (Figure 1.1b).



(A) Linearly separable data set, with linear separator

(B) Non-linearly separable data set, with linear separator

(C) Non-linearly separable data set, with quadratic separator

Figure 1.1. Soft SVM from Computer practical 4

Note 2. The accuracy of predictive rule can be improved if I take into account the curvature by adding a quadratic term in the 2nd dimension as $\eta'(x) = w'_1 x_1 + w'_2 x_2 + w'_2 x_2^2 + b = \langle w', \psi(x) \rangle + b$ where $\psi(x) = \left(x_1, x_2, x_2^2\right)$ and learning the w' . It works, (Figure 1.1c).

Note 3. Consequently, in order to improve the expressiveness of a hypothesis class $\mathcal{H} = \{x \mapsto f(\langle w, x \rangle)\}$ (for some function f) with purpose to learn a more accurate predictive rule, it is reasonable to consider an embedding $\psi(x)$ and work on the learning problem with $\mathcal{H} = \{x \mapsto f(\langle w, \psi(x) \rangle)\}$. Such an embedding $\psi(x)$ can possibly be a vector of basis functions such as polynomials, splines, etc...

Note 4. The above may drastically increase the dimensionality of the problem hence the computational cost and required number of example to train. This challenge is addressed by the Kernel trick.

Note 5. In general, kernel trick allows the design of powerful and cheap extensions of many well known algorithms.

2. Projections in feature spaces (Improving expressive power)

Note 6. To make the class of hypotheses more expressive with purpose to improve accuracy, we can first map the original instance space $x \in \mathcal{X}$ into another space $\mathcal{F}$ (possibly of a higher dimension) via an embedding ψ and then learn a hypothesis in that space.

Summary 7. Consider a hypothesis class $\mathcal{H} = \{x \mapsto \langle w, x \rangle : w \in \mathbb{R}^n\}$ where the predictive rule $h \in \mathcal{H}$ defined over $\mathcal{X}$ is to be trained against data set $\mathcal{S} = \{z_i = (x_i, y_i)\}_{i=1}^m$. The basic paradigm involves:

(1) Choose a mapping $\psi : \mathcal{X} \to \mathcal{F}$ with $\psi(x) := (\psi_1(x), ..., \psi_d(x))$ for some feature space $\mathcal{F}$.
(2) Create the image sequence $\tilde{\mathcal{S}} = \left\{ z_i^\psi = (\psi(x_i), y_i) \right\}_{i=1}^m$ from the original training set $\mathcal{S}$.
(3) Train a linear predictor h against $\tilde{\mathcal{S}}$.
(4) Predict the label or the output of a new point x^{new} by $h^\psi(x^{\text{new}}) := h \circ \psi(x^{\text{new}}) = h(\psi(x^{\text{new}}))$

Note 8. The introduction of mapping $\psi : \mathcal{X} \to \mathcal{F}$ induces

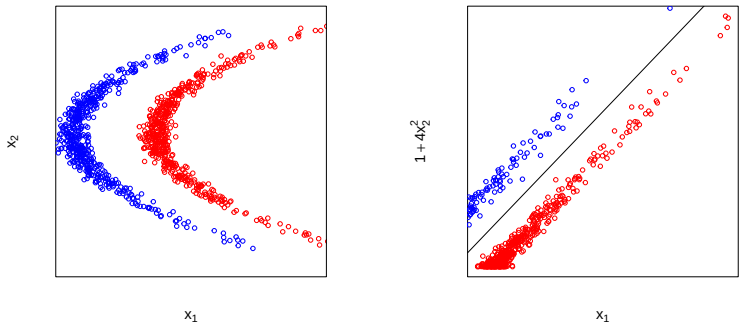
(1) probability distribution G^ψ over domain $\mathcal{X} \times \mathcal{F}$ with $G^\psi(A) = G\left(\psi^{-1}(A)\right)$ for every set $A \subseteq \mathcal{X} \times \mathcal{F}$.
(2) predictive rule $h^\psi(\cdot) := h \circ \psi(\cdot)$, where $h \circ \psi(\cdot) = h(\psi(\cdot))$
(3) risk function $R_{G^\psi}(h) := R_G(h \circ \psi)$, as

$$R_G(h \circ \psi) = \int \ell(h \circ \psi, z = (x, y)) \, \mathrm{d}G(z) = \int \ell\left(h, z^\psi\right) \mathrm{d}G^\psi(x, y) = R_{G^\psi}(h)$$

Definition 9. A Hilbert space is a vector space, with an inner product, which is also complete.

Lemma 10. *If $\mathcal{X}$ is a linear subspace of a Hilbert space, then every $x \in \mathcal{X}$ can be written as $x = u + v$ where $u \in \mathcal{X}$ and $\langle u, v \rangle = 0$ for all $v \in \mathcal{X}$.*

Note 11. Feature space $\mathcal{F}$ is a Hilbert space preferably due to Lemma 10 that enables the Kernel trick via the representation Theorem 20. Eg, a Euclidean space such as $\mathbb{R}^d$ for some d. That includes infinite dimensional spaces.



(A) Inputs in the original space x_1 vs x_2

(B) Projection of inputs tot he feature space $\psi_1(x) = x_1$ vs $\psi_2(x) = 4x_2^2+1$

FIGURE 2.1. Projection of the inputs living in the original space to the feature space

Note 12. As feature mapping ψ any function that maps the original instances $\mathcal{X}$ into some Hilbert space $\mathcal{F}$ can be used.

Note 13. The success of the learning paradigm in Summary 7 depends on choosing an embedding ψ that imposes a suitable deformation of the original feature space for the particular learning task to operate/perform effectively. Eg, in SVM, ψ will make the image of the data distribution (close to being) linearly separable in the feature space $\mathcal{F}$, thus making the resulting learning algorithm a good learner for a given task (Figure 2.1). This requires prior knowledge of the problem (In Section 4, we see popular recipes for that).

Note 14. Using a $\psi(x) := (\psi_1(x), ..., \psi_d(x))$ that is high dimensional (d is too large) may improve accuracy (expressiveness) of the learner (e.g. recall in polynomial regression increasing the polynomial degree). However this increases the computational effort/cost required to perform calculations to minimize the associated risk function in the high dimensional space, as well as we need more data. This is addressed via the Kernel trick.

Note 15. Recall the feed forward neural network (let's say with 1 hidden layer) formula mapping x to $h(x)$:

$$h(x) = \sigma_2\left(\sum_{j=}^{c} w_{2,1,j}\sigma_1\left(\sum_{i=1}^{n} w_{1,j,i}x_i\right)\right).$$

It can be considered as $h(x) = \sigma_2(\langle w_2, \psi(x)\rangle)$ with where the output of the hidden layer is the embeding $\psi(x) = (\sigma_1(\sum_{i=1}^n w_{1,1,i}x_i), ..., \sigma_1(\sum_{i=1}^n w_{1,c,i}x_i))$. Interestingly, FFNN, can be considered as providing an adaptive (semi-parametric) ψ as the weights $\{w_{1,j,i}\}$ are adapted during the training. Hence, FFNN performs a deformation of the input space in the same spirit to the projections in feature spaces method. Adding more hidden layers (multi-perceptron) potentially can improve this and provide adaptively a tailored deformation.

3. The Kernel trick

Definition 16. Kernel function K is defined as $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ with $K(x, x') = \langle \psi(x), \psi(x') \rangle$ given an embedding $\psi(x)$ of some domain space $\mathcal{X}$ into some Hilbert space $\mathcal{F}$. Kernel functions describe inner products in the feature space $\mathcal{F}$.

Problem 17. (Learning problem) Consider a prediction rule $h : \mathcal{X} \to \mathcal{Y}$ with $h(x) = \langle w, \psi(x) \rangle$ which is trained against a training sample $\{z_i = (x_i, y_i)\}_{i=1}^m$ with the following general optimization problem

$$\underset{w}{\text{minimize}} \left(f\left(\langle w, \psi(x_1) \rangle, ..., \langle w, \psi(x_m) \rangle \right) + R(\|w\|) \right), \tag{3.1}$$

where $f : \mathbb{R}^m \to \mathbb{R}$ is an arbitrary function and $R : \mathbb{R}_+ \to \mathbb{R}$ is a monotonically non-decreasing function.

Example 18. In Soft SVM (Proposition 24, Handout 6), it is $f(a_1, ..., a_m) = \frac{1}{m} \sum_{i=1}^m \max\{0, 1 - y_i a_i\}$ and $R(a) = \lambda a^2$.

Note 19. The following result states a duality in the learning problem 17 that facilitates the implementation of the extension to a possibly high dimensional feature space (hence improving the expressiveness/accuracy) by using kernel functions (hence reducing dimensionlality, computational cost, and required data size).

Theorem 20. *(Representation theorem) Assume mapping $\psi : \mathcal{X} \to \mathcal{F}$ where $\mathcal{F}$ is a Hilbert space. There exists a vector $\alpha \in \mathbb{R}^m$ such that $w = \sum_{i=1}^m \alpha_i \psi(x_i)$ is the optimal solution of (3.1) in Problem 17.*

Proof. Let w^* be the optimal solution of (3.1). Because w^* is element of Hilbert space, it can be written as $w^* = \sum_{i=1}^m \alpha_i \psi(x_i) + u$ where $\langle u, \psi(x_i) \rangle = 0$ for all $i = 1, ..., m$. Set $w := w^* - u$.

Because $\|w^*\|^2 = \|w\|^2 + \|u\|^2$ it is $\|w\| \leq \|w^*\|$ implying that

$$R(\|w\|) \leq R(\|w^*\|).$$

Because $\langle w, \psi(x_i) \rangle = \langle w^* - u, \psi(x_i) \rangle = \langle w^*, \psi(x_i) \rangle$ for all $i = 1, ..., m$, it is

$$f\left(\langle w, \psi(x_1) \rangle, ..., \langle w, \psi(x_m) \rangle \right) = f\left(\langle w^*, \psi(x_1) \rangle, ..., \langle w^*, \psi(x_m) \rangle \right)$$

Then the objective function of (3.1) at w is less than or equal to that of the minimizer w^* which implies that $w = \sum_{i=1}^m \alpha_i \psi(x_i)$ is an optimal solution. □

Note 21. Let $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a kernel function with $K(x, x') = \langle \psi(x), \psi(x') \rangle$. According to the representation Theorem 20, the Learning problem 17, can be equivalently addressed by re-writing the learning predictive rule as

$$h_\alpha(x) = \sum_{i=1}^m \alpha_i K(x_i, x)$$

and learning $\{\alpha_i\}$ as the solutions of

$$\underset{\alpha}{\text{minimize}} \left(f\left(\sum_{i=1}^{m}\alpha_i K\left(x_i,x\right),...,\sum_{i=1}^{m}\alpha_i K\left(x_m,x\right)\right) + R\left(\sqrt{\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j K\left(x_i,x_j\right)}\right)\right), \tag{3.2}$$

This is because

$$\langle w,\psi\left(x_j\right)\rangle = \langle\sum_{i=1}^{m}\alpha_i\psi\left(x_i\right),\psi\left(x_j\right)\rangle = \sum_{i=1}^{m}\alpha_i\langle\psi\left(x_i\right),\psi\left(x_j\right)\rangle = \sum_{i=1}^{m}\alpha_i K\left(x_i,x_j\right)$$

and

$$\|w\|^2 = \langle\sum_{i=1}^{m}\alpha_i\psi\left(x_i\right),\sum_{j=1}^{m}\alpha_j\psi\left(x_j\right)\rangle = \sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j\langle\psi\left(x_i\right),\psi\left(x_j\right)\rangle = \sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j K\left(x_i,x_j\right)$$

Example 22. In Soft SVM (Proposition 24, Handout 6), we can have

$$\underset{\alpha}{\text{minimize}} \left(\lambda\sum_i\sum_j\alpha_i\alpha_j K\left(x_i,x_j\right) + \frac{1}{m}\sum_i \max\left(0,1-y_i\sum_j\alpha_j K\left(x_i,x_j\right)\right)\right)$$

for $h\left(x\right) = \text{sign}\left(\sum_j \alpha_j K\left(x_i,x_j\right)\right)$. It can be minimised via SGD. We can call this form of SVM as Kernel SVM since we can just directly specify the kernel function without the need to even think about feature mapping $\psi\left(\cdot\right)$ (which is eliminated and replaced by the kernel).

Note 23. In Learning problem 17, direct access to elements $\psi\left(\cdot\right)$ in the feature space is not necessary, as equivalently one can calculate or just specify the associated kernel function (that is inner products in the feature space).

Example 24. (Polynomial Kernels) Let $x \in \mathcal{X} \subseteq \mathbb{R}^n$. Assume we want to extend the linear mapping $x \mapsto \langle w,x\rangle$ to the k degree polynomial mapping $x \mapsto h\left(x\right)$. The multivariate polynomial can be written as $h\left(x\right) = \langle w,\psi\left(x\right)\rangle$, where $\psi : \mathbb{R}^n \to \mathbb{R}^d$ with $\psi\left(x\right)$ is a vector of elements $\psi_J\left(x\right) = \prod_{i=1}^{r} x_{J_i}$ for $J \in \{1,...,n\}^r$ and $r \leq k$. This learning problem can be equivalently be addressed with the k degree polynomial kernel

$$K\left(x,x'\right) = \left(1+\langle x,x'\rangle\right)^k$$

Solution. It is

$$\begin{aligned} K\left(x,x'\right) = \left(1+\langle x,x'\rangle\right)^k &= \left(\sum_{j=0}^{n} x_j x'_j\right)^k \quad \text{(by setting } x_0 = x'_0 = 1\text{)} \\ &= \sum_{J\in\{1,...,n\}^k}\prod_{i=0}^{n} x_{J_i}x'_{J_i} = \sum_{J\in\{1,...,n\}^k}\left(\prod_{i=0}^{n} x_{J_i}\right)\left(\prod_{i=0}^{n} x'_{J_i}\right) \\ &= \langle\psi\left(x\right),\psi\left(x'\right)\rangle \end{aligned}$$

where $\psi\left(x\right)$ is as defined.

Example 25. (Radial basis kernel) Let the original input space be $x \in \mathcal{X} \subseteq \mathbb{R}$. Consider the Radial Basis Functions Kernel (or Gaussian kernel)

$$K\left(x, x'\right) = \exp\left(-\frac{1}{2\sigma^2}\left\|x - x'\right\|_2^2\right).$$

Show that it is a kernel indeed, by presenting it as an inner product in a feature space of infinite dimension, and state the bases of the mapping $\psi\left(\cdot\right)$.

Solution. It is

$$\begin{aligned} K\left(x, x'\right) &= \exp\left(-\frac{1}{2\sigma^2}\left\|x - x'\right\|_2^2\right) = \exp\left(\frac{1}{\sigma^2}xx' - \frac{1}{2}x^2 - \frac{1}{2}\left(x'\right)^2\right) \\ &= \exp\left(\frac{1}{\sigma^2}xx'\right)\exp\left(-\frac{1}{2\sigma^2}x^2\right)\exp\left(-\frac{1}{2\sigma^2}\left(x'\right)^2\right) \\ &= \sum_{k=0}^{\infty}\frac{\left(xx'/\sigma^2\right)^k}{k!}\exp\left(-\frac{1}{2\sigma^2}x^2\right)\exp\left(-\frac{1}{2\sigma^2}\left(x'\right)^2\right) \\ &= \sum_{k=0}^{\infty}\left[\frac{x^k}{\sqrt{k!}\sigma^k}\exp\left(-\frac{1}{2\sigma^2}x^2\right)\right]\left[\frac{\left(x'\right)^k}{\sqrt{k!}\sigma^k}\exp\left(-\frac{1}{2\sigma^2}\left(x'\right)^2\right)\right] \end{aligned}$$

hence it is $K\left(x, x'\right) = \left\langle\psi\left(x\right), \psi\left(x'\right)\right\rangle$ with $\psi_k\left(x\right) = \frac{x^k}{\sqrt{k!}\sigma^k}\exp\left(-\frac{1}{2\sigma^2}x^2\right)$.

4. Construction of kernels

Note 26. The kernel formulated as an inner product in a feature space allows us to build interesting extensions of many well-known algorithms by making use of the kernel trick and without the need to have direct access to the feature space (E.g. Example 22).

Note 27. Specifying a kernel function is a way to express prior knowledge without the need to have direct access to the feature space. This is consequence of the Representation theorem 20 that kernel is the inner product of feature mappings ψ which sufficiently replaces them in the learning problem, and the fact that ψ is a way to express and utilize prior knowledge about the problem at hand.

Note 28. The question is whether the specified function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ (by the practitioner) is indeed a kernel function; i.e. if K can be written as inner product $K\left(x, x'\right) = \left\langle\psi\left(x\right), \psi\left(x'\right)\right\rangle$ of feature functions $\psi\left(x\right)$. Theorem 31 provides sufficient and necessary conditions to check that.

Definition 29. Gram matrix is called the $m \times m$ matrix G s.t. $\left[G\right]_{i,j} = K\left(x_i, x_j\right)$.

Definition 30. A symmetric function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is positive semi-definite if its Gram matrix G, $\left[G\right]_{i,j} = K\left(x_i, x_j\right)$, is a positive semi-definite matrix.

Theorem 31. *A symmetric function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ ~~implements an inner product in some Hilbert space~~ is a valid kernel function in terms of if and only if it is positive semi-definite i.e. its Gram matrix G , $\left[G\right]_{i,j} = K\left(x_i, x_j\right)$, is a positive semi-definite matrix.*

Proof. Assume K is a valid kernel function (i.e. it implements an inner product in some Hilbert space) $K\left(x, x'\right) = \left\langle\psi\left(x\right), \psi\left(x'\right)\right\rangle$; let's consider $\psi : \mathcal{X} \to \mathbb{R}^d$ for simplicity. Let G be its Gram

matrix with $G = \Psi^\top \Psi$ and $\psi(x_i)$ is the i-th column of Ψ. For any $\xi \in \mathbb{R}^d - \{0\}$

$$\xi^\top G \xi = \sum_i \sum_j \xi_i K(x_i, x_j) \xi_j = \sum_i \sum_j \xi_i \langle \psi(x_i), \psi(x_j) \rangle \xi_j = \sum_i \sum_j \langle \xi_i \psi(x_i), \psi(x_j) \xi_j \rangle$$

$$= \langle \sum_i \xi_i \psi(x_i), \sum_j \psi(x_j) \xi_j \rangle = \left\| \sum_i \xi_i \psi(x_i) \right\|_2^2 \geq 0$$

Assume the symmetric function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is positive semi-definite. Let $\mathbb{R}^f = \{f : \mathcal{X} \to \mathbb{R}\}$. For $x \in \mathcal{X}$ let function ψ over $\mathbb{R}^f$ with $\psi(x) = K(\cdot, x)$. This allows to define a vector space consisting of all the linear combinations of elements of the form $K(\cdot, x)$, having an inner product

$$\langle \sum_i \alpha_i K(\cdot, x_i), \sum_j \beta_j K(\cdot, x_j) \rangle = \sum_i \sum_i \alpha_i \beta_j \underbrace{\langle K(\cdot, x_i), K(\cdot, x_j) \rangle}_{= K(x_i, x_j)}.$$

This is satisfies all the properties of inner product, s.t. it is symmetric, linearity, positive definite as $K(x, x') \geq 0$. Then there is some feature vector ψ such that $K(x, x') = \langle \psi(x), \psi(x') \rangle$. □

Claim 32. In Figure 1.1c, we could see that examples can be distinguished by some ellipse, so it was reasonable to we can define ψ as a vector with elements all the monomials up to order ; alternatively we could use a degree 2 polynomial kernel.

Proposition 33. *A powerful technique for constructing new kernels is to build them out of simpler kernels as building blocks. Below are some properties. Assume $K_1 : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ and $K_2 : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ are valid kernels, then the following are kernels too*

(1) $K(x, x') = K_1(x, x') + K_2(x, x')$
(2) $K(x, x') = K_1(x, x') K_2(x, x')$
(3) $K(x, x') = K_1(x_1, x_1') + K_2(x_2, x_2')$, *where* $x = (x_1, x_2)^\top$, $x' = (x_1', x_2')^\top$
(4) $K(x, x') = K_1(x_1, x_1') K_2(x_2, x_2')$, *where* $x = (x_1, x_2)^\top$, $x' = (x_1', x_2')^\top$
(5) $K(x, x') = f(x) K_1(x, x') f(x')$ *for any function* f
(6) $K(x, x') = K_1(f(x), f(x'))$ *for any function* f

Solution. We present the first two and the rest are proved similarly.

For (1). Let Gram matrix, G_j induced by kernel function K_j. For any $\xi \in \mathbb{R}^d - \{0\}$

$$\xi^\top G_3 \xi = \xi^\top (G_1 + G_2) \xi = \xi^\top G_1 \xi + \xi^\top G_2 \xi \geq 0$$

For (2). Assume that $K_j(x, x') = (\psi_j(x))^\top \psi_j(x)$. Then

$$K(x, x') = K_1(x, x') K_2(x, x') = (\psi_1(x))^\top \psi_1(x') (\psi_2(x))^\top \psi_2(x')$$

$$= (\psi_1(x))^\top \psi_2(x) (\psi_1(x'))^\top \psi_2(x') = \left((\psi_1(x))^\top \psi_2(x) \right)^\top (\psi_1(x'))^\top \psi_2(x')$$

which can be represented as an inner product of feature vectors.

See the proo
solution to a
ample

Note 34. The concept of a kernel formulated as an inner product in a feature space allows us to build interesting extensions of many well-known algorithms by making use of the kernel trick. One example was the Kernel SVM. Some other popular cases are Gaussian process regression, and Kernel PCA.