Journal of Applied Mathematics and Physics, 2022, 10, 1732-1747
https://www.scirp.org/journal/jamp
ISSN Online: 2327-4379
ISSN Print: 2327-4352

Legendre Polynomial Kernel: Application in SVM

Habib Rebei[1], Nouf S. H. Alharbi[2]

[1]Department of Mathematics, College of Science, Qassim University, Buraydah, Kingdom of Saudi Arabia
[2]Department of Mathematics, College of Science and Arts in Nabhaniya, Qassim University, Buraydah, Kingdom of Saudi Arabia
Email: h.rebl@qu.edu.sa, nos.alharbi@qu.edu.sa

How to cite this paper: Rebei, H. and Alharbi, N.S.H. (2022) Legendre Polynomial Kernel: Application in SVM. *Journal of Applied Mathematics and Physics*, **10**, 1732-1747.
https://doi.org/10.4236/jamp.2022.105121

Received: March 25, 2022
Accepted: May 27, 2022
Published: May 30, 2022

Abstract

In machines learning problems, Support Vector Machine is a method of classification. For non-linearly separable data, kernel functions are a basic ingredient in the SVM technic. In this paper, we briefly recall some useful results on decomposition of RKHS. Based on orthogonal polynomial theory and Mercer theorem, we construct the high power Legendre polynomial kernel on the cube $[-1,1]^d$. Following presentation of the theoretical background of SVM, we evaluate the performance of this kernel on some illustrative examples in comparison with Rbf, linear and polynomial kernels.

Keywords

SVM, Polynomial Legendre Kernel, Classification Problem, Mercer Theorem

1. Introduction

The approach of reproducing kernel function has many applications in probability theory, in statistics and recently in machine learning (see [1] [2] [3]). They have been applied in many fields and domains of life sciences such as pattern recognition, biology, medical diagnosis, chemistry and bio-informatics.

It is well-known from many years that data sets can be modeled by a family of points $\mathcal{X} \subset \mathbb{R}^d$ and similarity between them is given by an inner product on $\mathbb{R}^d$. The classification problem consists of separating these points into classes with respect to given properties. In the simplest situation, the points are linearly separable in the sense that there exists an hyperplane H_s separating the two classes.

Support vector machines (SVMs) have become a very powerful tool in machine learning in particular in classification problems and regression ([1] [4] [5]).

In classification problem, we can consider only two-classes of data, so we speak about a binary classification. Also we can rencounter more than two classes of data. This is called a multi-task classification problem. In our cases, we focus on binary classification and the application of kernel functions in SVM classification.

In simplest situations, the linear SVM technic allows to find the optimal hyperplane separating points in two classes. Unfortunately, in concrete examples, these points are not linearly separable. Then, one can traduce similarity of points in term of positive definite kernel function k on $\mathcal{X}$. This leads to invent a new technic named non-linear SVM which combines linear SVM and kernel tools.

Mathematically, this new approach of non-linear SVM is related to the Kolmogorov representation $(\mathcal{H},\Phi)$, where $\mathcal{H}$ represents the feature space and Φ is the feature map (see [1]). Since separation expresses a degree of similarity between points in the same class, Vapnick used the kernel approach to translate the problem from initial space $\mathbb{R}^d$ to the feature space. The transfer between initial space and the feature space is made under the feature map and similarities are expressed by the inner product in the feature space which is given by a kernel k. The crucial idea of the kernel function methods is that non-linearly separable points can be transformed to linearly separable points in the feature space guarding similarities which is expressed by

$$\left\langle \Phi\left(x_i\right), \Phi\left(x_j\right) \right\rangle = k\left(x_i, x_j\right).$$

Furthermore, the solution of the classification problem using non-linear SVM is given by the decision function which depends only on the kernel and support vectors ($x_i \in S$). Precisely, it takes the following form

$$F\left(x\right) = sgn\left(b + \sum_{x_i \in S} \alpha_i k\left(x_i, x\right) \right).$$

Since the choice of the kernel function is not canonic, the first problem of this approach is how to choose a suitable kernel function for such given data points. The second problem is that the feature space has an infinite dimensional and this causes a technical problem in modeling computational algorithm for such a solution. This problem is also related to the feature map Φ which is in general unknown. In our case, we use the Mercer decomposition theorem in order to obtain a type polynomial kernel having a good separation property. This means that infinite dimensional feature space can be approximated by finite dimensional feature space. This reduces the dimensionality of new data in feature space.

The paper is organized as follows: In Section 2, we recall some known results on Reproducing Kernel Hilbert Space (RKHS in what follow), in particular, the Mercer decomposition theorem 2.2 and the high power kernel theorem 2.3. Section 3 is devoted to the main result in which we introduce the one-dimensional Legendre polynomial kernel K_n and we give its canonical decomposition in term of Legendre orthogonal polynomials (see Theorem 3.1). Next, we deduce

the high power Legendre polynomial kernel $\mathbf{K}_n = K_n^d$ defined on the cube $[-1,1]^d$ for which the feature space $\mathbf{H}_n = \mathcal{H}_n^{\otimes d}$ is the tensor product Hilbert space of the RKHS associated with K_n. In Section 4, we recall the theoretical foundation of linear and non-linear SVMs. In Section 4, we give some illustrative examples in order to evaluate the performance of the Legendre polynomial kernel in comparison with some predefined kernels in Python.

2. Preliminaries

In this section, we begin by describing the RHKS and its associated kernel. Then we give the decomposition of a given positive definite kernel on a measure space $\mathcal{X}$ (see Theorem 2.2).

Definition 1. (Positive definite kernel).

Let $|XX$ be a nonempty set. A symmetric function $k:\mathcal{X}\times\mathcal{X}\to\mathbb{R}$ is called positive definite kernel if,

$$\sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j k\left(x_i, x_j\right) \geq 0 \quad \forall n \geq 1,\ a_1,\cdots,a_n \in \mathbb{R},\ x_1,\cdots,x_n \in \mathcal{X} \tag{2.1}$$

and for mutually distinct x_i, the equality holds only when all the a_i are zero.

Clearly that every inner product is a positive definite function. Moreover if $\mathcal{H}$ is any Hilbert space and $\mathcal{X}$ a nonempty set and $\phi:\mathcal{X}\to\mathcal{H}$, then the function $h(x,y) := \langle \phi(x), \phi(y) \rangle$ is a positive definite.

Now let $\mathcal{H}$ be a Hilbert space of functions mapping from some non-empty set $\mathcal{X}$ to $\mathbb{R}$, *i.e.*, $\mathcal{H}$ is considered as a subset of $\mathbb{R}^{\mathcal{X}}$. We write the inner product on $\mathcal{H}$ as

$$\langle f, g \rangle, \quad f, g \in \mathcal{H}$$

and the associated norm will be denoted by

$$\|f\|_2 = \langle f, f \rangle^{\frac{1}{2}}, \quad f \in \mathcal{H}.$$

We may alternatively write the function f as $f(\cdot)$, to indicate it takes an argument in $\mathcal{X}$.

Definition 2. (Reproducing kernel Hilbert space).

Let $\left(\mathcal{H}, \langle\cdot,\cdot\rangle\right)$ be a Hilbert space of $\mathbb{R}$-valued functions defined on a non-empty set $\mathcal{X}$. We say that $\mathcal{H}$ is a Reproducing Kernel Hilbert Space if there exists a kernel function $k:\mathcal{X}\times\mathcal{X}\to\mathbb{R}$ such that the following properties are satisfied:

1) $\forall x \in \mathcal{X}$, $k_x := k(.,x) \in \mathcal{H}$,
2) $\forall x \in \mathcal{X}$, $\forall f \in \mathcal{H}$, $f(x) = \langle k_x, f \rangle$ (the reproducing property).

Remark 1. From the reproducing property, we note that

$$k(x,y) = k_y(x) = \langle k_x, k_y \rangle = \langle k(.,x), k(.,y) \rangle,\ x, y \in \mathcal{X}. \tag{2.2}$$

Thus necessarily, k is positive definite. Note also that reproducing kernel k associated with $\mathcal{H}$, it is unique.

Theorem 2.1. (Moore-Aronszajn [6])

Let $k:\mathcal{X}\times\mathcal{X}\to\mathbb{R}$ be a positive definite kernel. There is a unique RKHS $\mathcal{H}\subset\mathbb{R}^{\mathcal{X}}$ with reproducing kernel *k*. Moreover, if space

$$\mathcal{H}_0 = span\left[\left\{k_x\right\}_{x\in\mathcal{X}}\right] \tag{2.3}$$

is endowed with the inner product

$$\left\langle f,g\right\rangle_{\mathcal{H}_0} = \sum_{i=1}^{n}\sum_{j=1}^{m}\alpha_i\beta_j k\left(x_i,y_j\right), \tag{2.4}$$

where $f=\sum_{i=1}^{n}\alpha_i k_{x_i}$ and $g=\sum_{j=1}^{m}\beta_j k_{y_j}$, then $\mathcal{H}$ is the completion of $\mathcal{H}_0$ w.r.t the inner product (4).

Now let us consider a compact metric space $\mathcal{X}$ and a finite Borel measure ν on it. We consider a positive definite continuous kernel on $\mathcal{X}$. Let S_k be the linear map

$$S_k : L_2\left(\mathcal{X};\nu\right)\to\mathcal{C}\left(\mathcal{X}\right),$$

$$\left(S_k f\right)(x) = \int_{\mathcal{X}} k\left(x,y\right) f\left(y\right)\mathrm{d}\nu\left(y\right)$$

and $T_k = I_k\circ S_k$, where $I_k:\mathcal{C}\left(\mathcal{X}\right)\hookrightarrow L_2\left(\mathcal{X};\nu\right)$. Then we have the following results:

Theorem 2.2. (Mercer's theorem [6])

1) T_k is a compact self-adjoint positive definite from $L_2\left(\mathcal{X};\nu\right)$ to itself. It is called the integral operator corresponding to the kernel *k*.

2) There exists an ortho-normal system $\left\{\tilde{e}_j\right\}_{j\in J}$ of eigenvectors of T_k with eigenvalues $\left\{\lambda_j\right\}_{j\in J}$, where *J* is at most countable set of indices, corresponding to the strictly positive eigenvalues of T_k such that

$$T_k\left(f\right) = \sum_{j\in J}\lambda_j\left\langle\tilde{e}_j,f\right\rangle\tilde{e}_j,\quad f\in L_2\left(\mathcal{X},\nu\right). \tag{2.5}$$

3) For all $x,y\in\mathcal{X}$

$$k\left(x,y\right) = \sum_{j\in J}\lambda_j e_j\left(x\right)e_j\left(y\right), \tag{2.6}$$

where $e_j=\lambda_j^{-1}S_k\tilde{e}_j\in\mathcal{C}\left(\mathcal{X}\right)$ and where the convergence is uniform on $\mathcal{X}\times\mathcal{X}$, and absolute for each pair $\left(x,y\right)\in\mathcal{X}\times\mathcal{X}$.

4) The Hilbert space

$$\mathcal{H} = \left\{f=\sum_{j\in J}a_j e_j : \left(\frac{a_j}{\sqrt{\lambda_j}}\right)\in\ell^2\left(J\right)\right\}$$

endowed with the inner product

$$\left\langle\sum_{j\in J}a_j e_j,\sum_{j\in J}b_j e_j\right\rangle_{\mathcal{H}} = \sum_{j\in J}\frac{a_j b_j}{\lambda_j} \tag{2.7}$$

is the RKHS associated with *k*.

Theorem 2.3. (Power of kernel [4] [7])

Let *k* be a kernel on $\mathcal{X}$. Then

$$K_d(x,y) := \prod_{i=1}^{d} k(x_i, y_i),\ x = (x_1, \cdots, x_d), y = (y_1, \cdots, y_d) \in \mathcal{X}^d$$

is a kernel on $\mathcal{X}^d$. In addition, there is an isometric isomorphism between $\mathcal{H}_K$ and the Hilbert space tensor product $\mathcal{H}_k^{\otimes d}$.

Example 1. Here we give some most known kernels used in SVM.

1) The linear kernel: $k_{lin}(x,y) = \langle x, y\rangle + c$, $c \geq 0$.

2) The polynomial kernel: $k_{poly}(x,y) = \left(a\langle x, y\rangle + b\right)^m$, $a > 0$, $b \geq 0$, $m \geq 1$.

3) The exponential kernel: $k_{exp}(x,y) = \mathrm{e}^{\sigma\langle x,y\rangle}$, $\sigma > 0$.

4) The radial basis function (Rbf): $k_{rbf}(x,y) = \mathrm{e}^{-\gamma\|x-y\|^2}$, $\gamma > 0$. It is also called the Guassian kernel.

5) The sigmoid kernel: $k_{sigm}(x,y) = \tanh\left(\langle x, y\rangle + c\right)$, $c \geq 0$.

3. Legendre Polynomial Kernel and High Power Legendre Polynomial Kernel

Let us consider the family of Legendre polynomials $\{L_n\}_{n\geq 0}$ defined by the following recurrence relation (see [8])

$$\begin{cases} L_0(x) = 1,\ L_1(x) = x, \\ L_{n+1}(x) = xL_n(x) - w_n L_{n-1}(x),\ n \geq 1, \end{cases} \tag{3.1}$$

where

$$w_n = \frac{n^2}{(2n-1)(2n+1)},\ n \geq 1. \tag{3.2}$$

It well-known [8] that they are orthogonal w.r.t the inner product

$$\langle f, g\rangle := \int_{-1}^{1} f(x)g(x)\mathrm{d}x.$$

Moreover we have

$$\|L_n\|^2 = w_1 \cdots w_n. \tag{3.3}$$

In order to apply the Mercer theorem in SVM, we consider the *Legendre polynomial kernel* defined on $[-1,1]^2$ by

$$K_n(x,y) = \sum_{k=0}^{n} L_k(x)L_k(y). \tag{3.4}$$

Theorem 3.1. Let

$$\mathcal{H}_n = \left\{ f = \sum_{k=0}^{n} \alpha_k L_k, \alpha_k \in \mathbb{R} \right\} \tag{3.5}$$

be the linear space generated by this family $\{L_k\}_{0\leq k\leq n}$. Then

1) $\mathcal{H}_n$ is a Hilbert space endowed with the inner product

$$\left\langle \sum_{i=0}^{n} \alpha_i L_i, \sum_{j=0}^{n} \beta_j L_j \right\rangle = \sum_{j=0}^{n} \alpha_j \beta_j \tag{3.6}$$

2) The sequence $\{L_k\}_{0\leq k\leq n}$ is an orthonormal basis of $\mathcal{H}_n$.

3) $\mathcal{H}_n$ is a RKHS with reproducing kernel K_n.

4) The feature map associated with K_n is

$$\phi_n : [-1,1] \to \mathcal{H}_n, t \mapsto \phi(t) = \sum_{j=0}^{n} L_j(t) L_j.$$

Proof. **First:** Clearly that (3.6) defines a inner product on $\mathcal{H}_n$, for which $\mathcal{H}_n$ becomes an Hilbert space since it has a finite dimension.

Second: Form the definition of the inner product (3.6), clearly that $\{L_k\}_{0\le k\le n}$ is an orthonormal system of $\mathcal{H}_n$ whose cardinality coincides with the dimension of $\mathcal{H}_n$. Thus it is an orthonormal basis of $\mathcal{H}_n$.

Third: Let us consider the operator S_n defined on $\mathcal{H}_L := L^2([-1,1])$ by

$$S_n : f \mapsto \int_{-1}^{1} K_n(x,y) f(y) \mathrm{d}y \in \mathcal{C}([-1,1]).$$

Setting $T_n = I_n \circ S_n$, where I_n is the canonic injection of $\mathcal{C}([-1,1])$ into $\mathcal{H}_L$.

1) K_n is positive definite. In fact

$$\begin{aligned}\sum_{1\le i,j\le m} a_i a_j K_n(x_i,x_j) &= \sum_{1\le i,j\le m} a_i a_j \sum_{k=0}^{n} L_k(x_i) L_k(x_j) \\ &= \sum_{k=0}^{n} \sum_{1\le i,j\le m} a_i a_j L_k(x_i) L_k(x_j) \\ &= \sum_{k=0}^{n} \left(\sum_{i=1}^{m} a_i L_k(x_i) \right)^2 \ge 0.\end{aligned}$$

2) T_n is symmetric since K_n is symmetric, so by Mercer theorem 2.2, T_n is self-adjoint compact and positive definite.

3) For all $j = 0,1,\cdots,n$, we have

$$\begin{aligned}T_n(L_j)(x) &= \int_{-1}^{1} K_n(x,y) L_j(y) \mathrm{d}y \\ &= \int_{-1}^{1} \left(\sum_{k=0}^{n} L_k(x) L_k(y) \right) L_j(y) \mathrm{d}y \\ &= \sum_{k=0}^{n} \int_{-1}^{1} L_k(x) L_k(y) L_j(y) \mathrm{d}y \\ &= \sum_{k=0}^{n} L_k(x) \int_{-1}^{1} L_k(y) L_j(y) \mathrm{d}y \\ &= \sum_{k=0}^{n} L_k(x) \|L_j\|^2 \delta_{k,j} \\ &= \|L_j\|^2 L_j(x).\end{aligned}$$

So the eigenvectors of T_n are exactly the polynomials L_j and the corresponding eigenvalues are $\lambda_j = \|L_j\|^2$. Thus $\{L_j, 0 \le j \le n\}$ is an orthonormal basis of $\mathcal{H}_n$ formed by eigenvectors of T_n.

From Mercer theorem 2.2, The RKHS associated with K_n is given by

$$\mathcal{H}_{K_n} = \left\{ f = \sum_{j=0}^{n} \alpha_j L_j, \frac{\alpha_j}{\sqrt{\lambda_j}} \in l^2 \right\}$$

Since $\frac{\alpha_j}{\sqrt{\lambda_j}} \in l^2$ for all sequence α_j, the space $\mathcal{H}_{K_n}$ coincides with $\mathcal{H}_n$ given

by (3.5). Thus $\mathcal{H}_n$ is the RKHS associated with the reproducing kernel K_n.

The reproducing property is immediate from Mercer theorem but also it can be checked by hand using (3.6)

$$\left\langle K_n(x,\cdot), f(\cdot)\right\rangle = \left\langle \sum_{k=0}^{n} L_k(x)L_k(\cdot), \sum_{j=0}^{n}\alpha_j L_j(\cdot)\right\rangle = \sum_{k=0}^{n}\alpha_k L_k(x) = f(x).$$

Forth: The feature map is given by

Let ϕ_n given by

$$\phi_n : [-1,1] \to \mathcal{H}_n, t \mapsto \phi_n(t) = \sum_{i=0}^{n} L_j(t)L_j(\cdot).$$

So from (3.6), we get

$$\left\langle \phi_n(s), \phi_n(t)\right\rangle = \left\langle \sum_{i=0}^{n} L_i(s)L_i, \sum_{j=0}^{n} L_j(t)L_j\right\rangle = \sum_{j=0}^{n} L_j(s)L_j(t) = K_n(s,t). \quad \square$$

Now let us consider the set $\mathcal{X} = [-1,1]^d$. Define the *high power Legendre polynomial kernel* function $\mathbf{K}_n : \mathcal{X}\times\mathcal{X} \to \mathbb{R}$ as follows:

$$\mathbf{K}_n(x,y) = \prod_{i=1}^{d} K_n(x_i,y_i) = \prod_{i=1}^{d}\sum_{k=0}^{n} L_k(x_i)L_k(y_i),\ x = (x_1,\cdots,x_d),\ y = (y_1,\cdots,y_d). \quad (3.7)$$

Let us introduce $\mathbf{H}_n = \mathcal{H}_n^{\otimes d} = \underbrace{\mathcal{H}_n \otimes \cdots \otimes \mathcal{H}_n}_{d\text{-times}}$ called the d--times tensor product of $\mathcal{H}_n$.

It's known that $\mathbf{H}_n$ is a real Hilbert space endowed with the inner product

$$\langle f | g\rangle_{\mathbf{H}_n} = \prod_{i=1}^{d}\langle f_i, g_i\rangle,\ f = f_1\otimes\cdots\otimes f_d,\ g = g_1\otimes\cdots\otimes g_d \in \mathbf{H}_n,\ f_i, g_i \in \mathcal{H}_n. \quad (3.8)$$

For $(x,y) \in \mathcal{X}\times\mathcal{X}$, we have

$$\mathbf{K}_n(x,y) = \prod_{i=1}^{d} K_n(x_i,y_i) = \prod_{i=1}^{d}\langle \phi_n(x_i), \phi_n(y_i)\rangle = \left\langle \bigotimes_{i=1}^{d}\phi_n(x_i) \,|\, \bigotimes_{i=1}^{d}\phi_n(y_i)\right\rangle_{\mathbf{H}_n}. \quad (3.9)$$

Define now the function $\Phi_n : \mathcal{X} \to \mathbf{H}_n$ given by

$$\Phi_n(x) = \bigotimes_{i=1}^{d}\phi_n(x_i), \quad x = (x_1,\cdots,x_d). \quad (3.10)$$

by substituting (3.10) in (3.9), we get

$$\mathbf{K}_n(x,y) = \left\langle \Phi_n(x) \middle| \Phi_n(y)\right\rangle_{\mathbf{H}_n}. \quad (3.11)$$

Hence $\mathbf{K}_n$ is a kernel, with the feature map Φ_n and feature space $\mathbf{H}_n$.

Now, applying Theorem 2.3 to the kernel $\mathbf{K}_n$, we deduce the following result.

Corollary 3.1.1. *The space* $\mathbf{H}_n$ *is a RKHS with corresponding kernel* $\mathbf{K}_n$.

4. Application of Kernels in Classification Problem

The binary classifications means that given a date points $\mathcal{X} = (x_i)_{1\le i\le L} \subset \mathbb{R}^d$ which are belonging to two classes. One is denoted **Class(1)**, the other is denoted **Class(−1)**. To each point x_i we associate $y_i = \pm 1$ indicating to which class x_i belongs. The points $(x_i)_{1\le i\le L}$ are called training points. The idea of SVM is to find an hyperplane $\mathcal{H}_S$ separating the two classes and construct a

decision function f from which a new point can be classified (x belongs to **Class(1)** or x belongs to **Class(−1)**).

4.1. Support Vector Machine: Linearly Separable Classification

When the training samples $\mathcal{X} = (x_i)_{1\le i\le L} \subset \mathbb{R}^d$ are linearly separable, we speak about a linear classification (*i.e.*, there exists an hyperplane separating the two classes), the idea of SVM is the following:

Let us consider the set of training points, $\{(x_i, y_i)\}_{1\le i\le L}$, where $x_i = \left(x_i^{(1)}, \cdots, x_i^{(d)}\right) \in \mathbb{R}^d$ and the corresponding labels $y_i \in \{-1,1\}$. The first step is to find an hyperplane $\mathcal{H}_S$ in the space $\mathbb{R}^d$ separating the two classes. The second step is to construct the decision function f.

Support Vectors are the examples closest to the separating hyperplane $\mathcal{H}_S$ and the aim of SVM is to orientate this hyperplane in such a way as to be as far as possible from the closest members of both classes. For example in 2-dimensional space this means that we can draw a line on a graph of x_i v.s x_j separating the two classes. In high dimensional space ($d > 2$), the line will be replaced by an affine hyperplane $\mathcal{H}_S$, *i.e.*, $\dim \mathcal{H}_S = d - 1$ (see Figure 1).

Step.1.

Recall that any affine hyperplane is described by the equation

$$w \cdot x + b = 0,$$

where "$\cdot$" is the dot product in $\mathbb{R}^d$ and w is normal to the hyperplane. So we have to determinate the appropriate values of w and b for the hyperplane $\mathcal{H}_S$. If we now just consider the points that lie closest to the separating hyperplane, *i.e.*, the Support Vectors (shown in circles in the diagram), then the two planes $\mathcal{H}_1$ and $\mathcal{H}_{-1}$ that these points lie on can be described by:

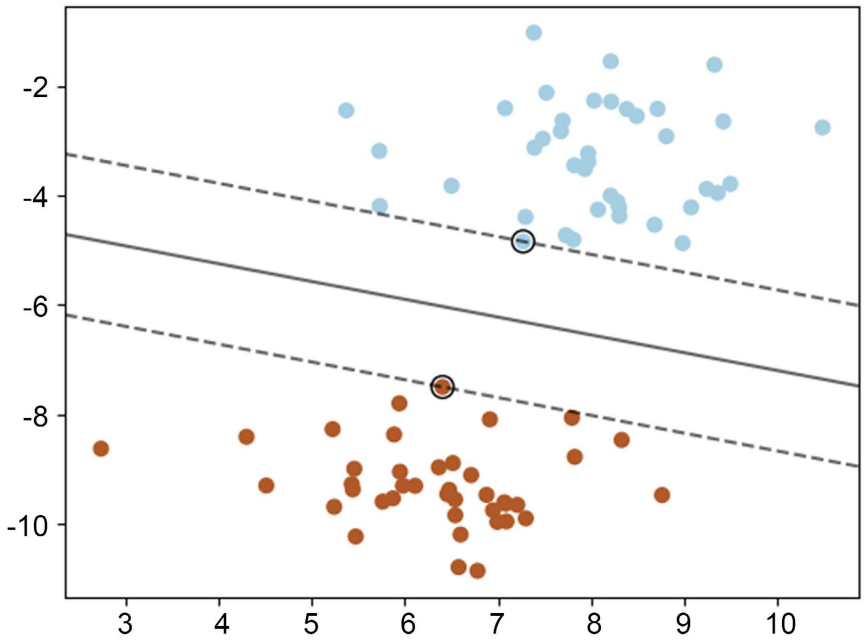

Figure 1. Separating hyperplane and marginal distance.

$$x_i \cdot w + b = 1 \text{ for } \mathcal{H}_1 \tag{4.1}$$

$$x_i \cdot w + b = -1 \text{ for } \mathcal{H}_{-1} \tag{4.2}$$

Referring to **Figure 1**, we define δ_1 as being the distance from $\mathcal{H}_1$ to the hyperplane $\mathcal{H}_S$ and δ_2 from $\mathcal{H}_{-1}$ to it. The hyperplane's equidistance from $\mathcal{H}_1$ and $\mathcal{H}_{-1}$ means that $\delta = \delta_1 = \delta_2$. The quantity δ is known as the SVM's margin. In order to orientate the hyperplane to be as far from the Support Vectors as possible, we need to maximize this margin. It is known from [1] that this margin is equal to

$$\delta = \frac{1}{\|w\|}.$$

Now the problem is to find w and b such that the marginal distance δ is maximal and for all $i = 1, \cdots, L$,

$$w \cdot x_i + b \geq 1 \text{ for } x \in \mathcal{H}_1 \text{ and } w \cdot x_i + b \leq -1 \text{ for } x \in \mathcal{H}_{-1}. \tag{4.3}$$

This is equivalent to the optimization problem under constraint

$$\min\left(\frac{1}{2}\|w\|^2\right) \quad \text{S.T} \quad y_i\left(w \cdot x_i + b - 1\right) \geq 0 \quad \forall i = 1, \cdots, L, \tag{4.4}$$

which is in fact a quadratic programming optimization (Q.P-optimization). Using the Lagrange multipliers $\lambda = (\lambda_1, \cdots, \lambda_L)$, where $\lambda_i \geq 0 \quad \forall i = 1, \cdots, L$, the problem (4.4) will be equivalent to

$$\max_{\lambda}\left[\sum_{i=1}^{L}\lambda_i - \frac{1}{2}\lambda H \lambda^{\mathrm{T}}\right] \text{ S.T } \lambda_i \geq 0, \sum \lambda_i y_i = 0, \tag{4.5}$$

where

$$H = \left(H_{i,j}\right)_{1 \leq i,j \leq L} = \left(y_i y_j x_i \cdot x_j\right)_{1 \leq i,j \leq L}.$$

This is a convex quadratic optimization problem, and we run a QP--solver which will return λ. Thus we can deduce w and b which are given by [1]

$$w = \sum_{i=1}^{L}\lambda_i y_i x_i; \quad b = \frac{1}{N_S}\sum_{s \in S}\left(y_s - \sum_{k \in S}\lambda_k y_k x_k \cdot x_s\right), \tag{4.6}$$

where S the set of the support vectors x_s (*i.e.*, the vectors of indices i corresponding to $\lambda_i > 0$) and N_S is the number of support vectors.

Step.2.

The second step is to create the decision function f which determines to which class a new point belongs. From [1], the decision function is given by

$$f(x) = sgn\left(y_i\left(w \cdot x_i + b\right) - 1\right) \in \{\pm 1\}. \tag{4.7}$$

Thus for a new data x, $f(x) = 1$ means x belongs to the first class and $f(x) = -1$ means that x belongs to the second class.

In practise, in order to use an SVM to solve a linearly separable, binary classification problem we need to:

1) Create the matrix $H = \left(H_{i,j}\right) \in M_L(\mathbb{R})$, where $H_{i,j} = y_i y_j x_i \cdot x_j$.

2) Find λ so that

$$\sum_{i=1}^{L}\lambda_i - \frac{1}{2}\lambda H \lambda^{\mathrm{T}}$$

is maximized, subject to the constraints ($\lambda_i \geq 0$ and $\forall i$, $\sum_{i=1}^{L}\lambda_i y_i = 0$).

This is can be done using a QP solver.

3) Calculate $w = \sum_{i=1}^{L}\lambda_i y_i x_i$.

4) Determine the set of Support Vectors S by finding the indices such that $\lambda_i > 0$.

5) Calculate $b = \frac{1}{N_S}\sum_{s\in S}\left(y_s - \sum_{k\in S}\lambda_k y_k x_k \cdot x_s\right)$.

6) Each new point x' is classified by evaluating $f(x') = sgn(w\cdot x' + b)$.

4.2. SVM for Data That Is Not Fully Linearly Separable

In order to extend the SVM methodology to handle data that is not fully linearly separable, we relax the constraints (4.3) slightly to allow for misclassified points. This is done by introducing a positive slack variable $\xi_i \geq 0$, $i = 1,\cdots,L$

$$w\cdot x_i + b \geq 1 - \xi_i \text{ for } y_i = +1,$$

$$w\cdot x_i + b \leq -1 + \xi_i \text{ for } y_i = -1.$$

which can be combined into:

$$y_i(w\cdot x_i + b) - 1 + \xi_i \geq 0 \quad \forall i = 1,\cdots,L. \tag{4.8}$$

In this soft margin SVM, data points on the incorrect side of the margin boundary have a penalty that increases with the distance from it. As we are trying to reduce the number of misclassifications, a sensible way to adapt our objective function (4.8) from previously, is to find

$$\min\left(\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{L}\xi_i\right) \quad \text{S.T} \quad y_i(w\cdot x_i + b) - 1 + \xi_i \geq 0 \quad \forall i = 1,\cdots L, \tag{4.9}$$

where the parameter C controls the trade-off between the slack variable penalty and the size of the margin.

Similarly to what have been done in the previous case, the Lagrange method leads to the following convex quadratic optimization problem

$$\max_{\lambda}\left[\sum_{i=1}^{L}\lambda_i - \frac{1}{2}\lambda H\lambda^{\mathrm{T}}\right] \quad \text{S.T} \quad 0 \leq \lambda_i \leq C, \sum\lambda_i y_i = 0. \tag{4.10}$$

We run a QP-solver which will return λ. The values of w and b are calculated in the same way as (4.6), though in this instance the set of Support Vectors used to calculate b is determined by finding the indices i for which $0 < \lambda_i < C$.

In practise, we need to:

1) Create the matrix H, where $H_{i,j} = y_i y_j x_i \cdot x_j$.

2) Select a suitable value for the parameter C which determines how significantly misclassifications should be treated.

3) Find λ satisfying

$$\sum_{i=1}^{L}\lambda_i - \frac{1}{2}\lambda H\lambda^{\mathrm{T}} \text{ is maximized } \text{S.T} \quad 0 \leq \lambda_i \leq C \ \ \forall i \text{ and } \sum_{i=1}^{L}\lambda_i y_i = 0.$$

This is done using a QP-solver.

4) Calculate $w = \sum_{i=1}^{L} \lambda_i y_i x_i$.

5) Determine the set of Support Vectors S by finding the indices such that $0 < \lambda_i < C$.

6) Calculate $b = \frac{1}{N_S} \sum_{s \in S} \left(y_s - \sum_{k \in S} \lambda_k y_k x_k \cdot x_s \right)$.

7) Each new point x' is classified by evaluating

$$f(x') = sgn(w \cdot x' + b) = sgn\left[\left(\sum_{i=1}^{L} \lambda_i y_i x_i \cdot x' \right) + b \right].$$

4.3. Non-Linear SVM

In the case when the data points are not linearly separable, *i.e.*, there is no hyperplane separating data in two classes, we have to insert some modification on the data in order to obtain a linearly separable points. This is based on the kernel functions. It is worth noting that in the case of linearly separable data, the decision function requires only the dot product of the data points x_i and the input vector x with each x_i. In fact, when applying the SVM technic to linearly separable data we have started by creating a matrix H and the scalar b from the dot product of our input variables

$$H_{i,j} = y_i y_j x_i \cdot x_j; \quad b = \frac{1}{N_S} \sum_{s \in S} \left(y_s - \sum_{k \in S} \lambda_k y_k x_k \cdot x_s \right).$$

This is an important constatation for the Kernel Trick. In fact the dot product will be replaced by such a kernel which also a positive definite function. The idea is based on the choice of such kernel function k and the trick is to maps the data into a high-dimensional feature space $\mathcal{H}$ via a transformation ϕ related to k, in such way that the transformed data are linearly separable. ϕ is called feature map $\phi : \mathcal{X} \to \mathcal{H}$. When this hyperplane is back into the original space it describes a surface.

Similarly to the previous section, we adopt the same procedure of separation at the level of the feature space $\mathcal{H}$. This leads to the following steps.

1) Choose a kernel function k.

2) Create the matrix H, where $H_{i,j} = y_i y_j k(x_i, x_j)$.

3) Choose how significantly misclassifications should be treated, by selecting a suitable value for the parameter C.

4) Determine λ so that

$$\sum_{i=1}^{L} \lambda_i - \frac{1}{2} \lambda H \lambda^{\mathrm{T}}$$

is maximal under the constraint $0 \le \lambda_i \le C \quad \forall i$ and $\sum_{i=1}^{L} \lambda_i y_i = 0$.

This is can be done by using a QP-solver.

5) Determine the set of Support Vectors S by finding the indices such that $0 < \lambda_i < C$.

6) Calculate $b = \frac{1}{N_S} \sum_{s \in S} \left(y_s - \sum_{i \in S} \lambda_i y_i k(x_i, x_s) \right)$.

7) Each new point x' is classified by evaluating

$$f(x') = sgn(w \cdot x' + b) = sgn\left[\left(\sum_{i=1}^{L} \lambda_i y_i k(x_i, x')\right) + b\right].$$

Note that in general, the feature map ϕ is unknown so w is also unknown. But we don't need it! We need only the values of the kernel at the training points (*i.e.*, $k(x_i, x_j)$ for all i, j) to know b, and $k(x_i, x')$ to determine the values of the decision function f at the input vector x'.

5. Numerical Simulations

Dataset of female patients with minimum twenty one year age of Pima Indian population has been taken from UCI machine learning repository. This dataset is originally owned by the National institute of diabetes and digestive and kidney diseases. In this dataset, there are total 768 instances classified into two classes: diabetic and non diabetic with eight different risk factors: Pregnancies, Glucose, Blood Pressure, Skin Thickness, Insulin, BMI, Diabetes Pedigree Function and Age. To diagnose diabetes for Pima Indian population, performance of all kernels is evaluated upon parameters like Accuracy, precision, recall score, F_1 score and execution time.

Before giving experiment results, we would like to recall some characteristics of the confusion matrix (Table 1) and related metric evaluation.

5.1. Terminology and Derivations from a Confusion Matrix

Table 1. Confusion Matrix.

Total population (Tp)	PP	PN
RP	TP	FN
RN	FP	TN

where:

- (TP) is the number of True Positive cases (real positive cases and detected positive),
- (TN) is the number of True Negative cases (real negative cases and detected negative),
- (FP) is the number of False Positive cases (real negative cases and detected positive),
- (FN) is the number of False Negative cases (real positive cases and detected negative),
- (PP) is the number of Predicted Positive cases $PP = TP + FP$,
- (PN) is the number of Predicted Negative cases $PN = FN + TN$,
- (RP) is the number of Real Positive cases $RP = TP + FN$,
- (RN) is the number of Real Negative $RN = FP + TN$,
- The Total Population is given by

$$\mathbf{Tp} = PP + PN = RP + RN.$$

These are some metrics used in order to compare performance of Legendre polynomial kernel with linear, rbf and polynomial kernels.

– Precision (or positive predictive value PPV) has been used to determine classifier's ability that provides correct positive predictions of diabetes.

$$\mathrm{PPV} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}.$$

– Recall score (or true positive rate or sensitivity) is used in our work to find the proportion of actual positive cases of diabetes correctly identified by the classifier used.

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{RP}} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}.$$

– Precision and recall score provide F_1 score, so this score takes into account of both. It is the harmonic mean of precision and sensitivity:

$$\mathrm{F}_1 = \frac{2\mathrm{PPV} \times \mathrm{TPR}}{\mathrm{PPV} + \mathrm{TPR}}.$$

– Accuracy (ACC) is the ratio of true positives and true negatives to all positive and negative observations. In other words, accuracy tells us how often we can expect our machine learning model will correctly predict an outcome out of the total number of times it made predictions.

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{RP} + \mathrm{RN}} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}.$$

5.2. Numerical Results

Since Legendre polynomial kernel is defined on $[-1,1]^d$, a scaling procedure is needed in order to make data in $[-1,1]$. For this, we suggest the following transformation $z_i^{(j)} = \dfrac{2x_i^{(j)} - \left(M_j + m_j\right)}{M_j - m_j}$, $j = 1, \cdots, d$, $i = 1, \cdots, L$, where $M_j = \max_{1 \le i \le L} x_i^{(j)}$; $m_j = \min_{1 \le i \le L} x_i^{(j)}$.

The confusion matrices corresponding to each kernels mentioned above are given in Figures 2-5, where the test size is 0.05 and the penalty coefficient $C = 0.01$.

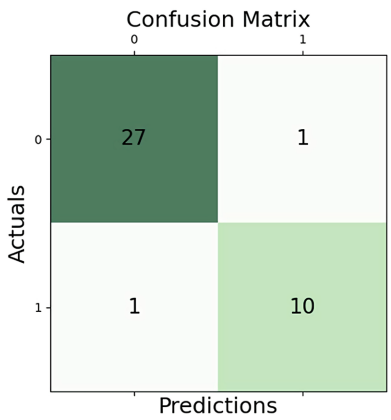

Figure 2. Polynomial Legendre kernel.

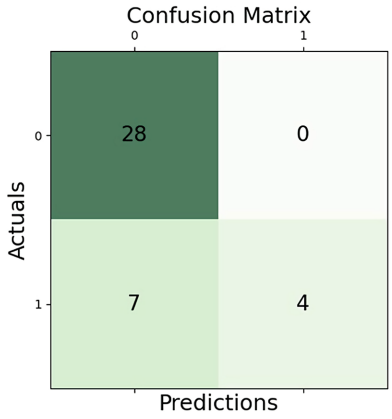

Figure 3. Liner kernel.

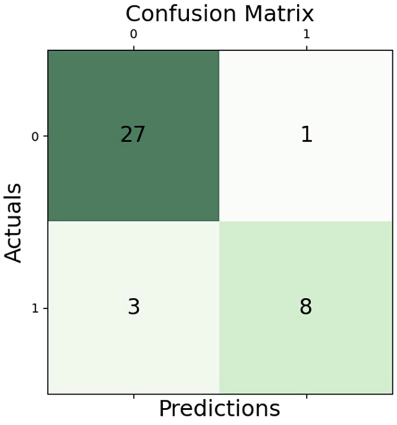

Figure 4. Rbf kernel.

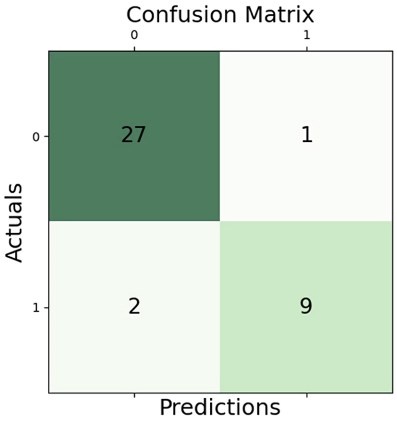

Figure 5. Polynomial kernel.

In order to show the powerful separation properties of the kernel defined by (3.1), our numerical simulations were carried out on the diabetes detection model mentioned above. In this example we apply SVM with different kernels in Pima Indian diabetes dataset in order to compare the performance of the Legendre polynomial kernel with linear, rbf, polynomial kernels with respect to their Accuracy, Precision, Recall score, F_1 score and Execution time (see **Table 2**).

Table 2. Numerical results.

Kernel	Execution time	Accuracy	Precision	Recall	F_1 score	Parameters
Linear	12.86	0.8831	0.9107	0.9273	0.9189	-
Rbf	5.92	0.9221	0.9636	0.9298	0.9464	$\gamma = 0.01$
Polynomial	4.68	0.9221	0.9259	0.9615	0.9434	$N = 20$
Legendre polynomial	241.05	0.9740	0.9818	0.9818	0.9818	$N = 20$

The programs were implemented in Python.3.7 a Windows 7 computer with $3G$ memory. The test size is equal to 0.2 and the penalty is taken to be $C = 0.001$.

5.3. Conclusion

Form the comparative table above, it is clear that the polynomial Legendre kernel have a good separation property with a good precision and accuracy w.r.t the classical predefined kernel in Python. Essentially, we have shown that it is not necessary to have an RKHS with an infinite dimension in order to separate by an hyperplane all dataset in the feature space. In fact we can obtain a good classification with non big degree $N = 20$. The more N increases, the more we obtain a better separation. The only disadvantage is the time required to fit the model with the polynomial Legendre kernel. The idea used here can be generalized with arbitrary sequence of orthogonal polynomials in order to get new kernel functions. The performance of separation property depends on the polynomials sequence and it should be tested on some examples of dataset.

Acknowledgements

The authors gratefully acknowledge Qassim University, represented by the Deanship of Scientific Research, for the support for this research.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Cortes, C. and Vapnik, V. (1995) Support-Vector Networks. *Machine Learning*, **20**, 273-297. https://doi.org/10.1007/BF00994018

[2] Chatterjee, R. and Yu, T. (2017) Generalized Coherent States, Reproducing Kernels and Quantum Support Vector Machines. arXiv:1612.03713v2.

[3] Abbassa, N., Abdessamad, A. and Bahri, S.M. (2019) Regularized Jacobi Wavelets Kernel for Support Vector Machines. *Statistics, Optimization & Information Computing*, **7**, 669-685. https://doi.org/10.19139/soic-2310-5070-634

[4] Jakkula, V. (2010) Tutorial on Support Vector Machine (SVM). School of EECS, Washington State University, Pullman.

[5] Hastie, T., Tibshirani, R. and Friedman, J. (2009) The Elements of Statistical Learn-

ing: Data Mining, Inference, and Prediction. Springer-Verlag, New York.
https://doi.org/10.1007/978-0-387-84858-7

[6] Aronszajn, N. (1950) Theory of Reproducing Kernels. *Transactions of the American Mathematical Society*, **686**, 337-404.
https://doi.org/10.1090/S0002-9947-1950-0051437-7

[7] De Vito, E., Umanità, V. and Villa, S. (2011) An Extension of Mercer Theorem to Vector-Valued Measurable Kernels. arXiv:1110.4017v1.

[8] Rebei, H., Riahi, A. and Chammam, W. (2022) Hankel Determinant of Linear Combination of the Shifted Catalan Numbers. Multilinear Algebra.