kernel functions for Support Vector Machines: A Survey

Boutkhil SIDAOUI[1,2] and Halima ABDELMOUMENE[3]

***Abstract*— Support Vector Machines (SVMs) are powerful supervised learning algorithms that are extensively used for both classification and regression tasks. An important component of SVMs is the kernel function, which enables the algorithm to perform correctly in nonlinear problems by transforming the input data into higher-dimensional linear spaces. This survey examines various kernel functions employed in Support Vector Machines (SVMs), concentrating on their mathematical formulations, properties, and applications. We examine the most commonly used kernels, including the linear, polynomial, radial basis function (RBF), and sigmoid kernels, and discuss their strengths, weaknesses, and suitable applications. Additionally, we review specialized kernels, such as string kernels and graph kernels, for specific domains, including speech recognition, text mining, and bioinformatics. The paper also addresses kernel selection methods, the impact of kernel parameters on model performance, and challenges in kernel-based learning. This comprehensive overview aims to provide insights into how different kernels can influence SVM performance and guide practitioners in selecting the most suitable kernel for their specific application.**

I. INTRODUCTION

Many fundamental problems involve non-homogeneous data, requiring methods that can address the heterogeneous nature of real-world data, such as vectors, sequences, trees, and other complex data structures. These challenges are commonly faced in practical applications, such as the classification and prediction of biological sequences in bioinformatics, where data are represented by strings [1], [2]. Another example is natural language processing, where data are often in the form of parse trees [3]. In these cases, the classification approach must account for the non-uniformity of the data. Several approaches have been proposed to support this type of data and enhance the performance of recognition systems. The first class focuses on new feature extraction and normalization techniques during preprocessing to produce rows of data with the same dimension. Methods in the second category preserve feature and derivative information and employ powerful new discriminative techniques, as seen with kernel-based methods [4], [5]. Kernel-based methods, especially support vector machines (SVM), are universal algorithms that use kernels to adapt to various data structures. These kernels can be defined for various data types, including heterogeneous data, allowing the SVM algorithm to adapt to both scenarios. In the first case, SVMs can be applied with standard kernels, including linear, polynomial, and Gaussian kernels. In the second case, SVM enables the development of new kernels that support heterogeneous data. However, creating an effective kernel for SVM is a challenging task that requires a deep understanding of the problem and an appropriate data representation. This paper provides an overview of various kernels employed by the SVM method. This paper also recapitulates the works related to SVM using different kernels. Many other studies have been conducted to address kernel-related challenges, among which we can highlight the works cited from [6], [7], [8], [9]. The remainder of this paper is organized as follows. Section 2 discusses kernel-based methods. Section 3 introduces standard kernel functions. Section 4 examines various suitable kernels for different problems. Finally, Section 5 concludes the paper with a discussion.

II. KERNEL-BASED METHODS

Most classification algorithms utilize the concept of similarity between objects, typically in a linear form, to create a separation surface. The similarity between two vectors with n attributes can be calculated as the usual dot product in the n-dimensional Euclidean space $\Re^n$, where these objects can be represented. However, using this separation surface may be inadequate in real situations where the data are not linearly separable. In addition, some practical problems, such as those in bioinformatics, are non-homogeneous and do not admit a Euclidean representation. The idea is to project the input objects, utilizing a mapping function, into a non-Euclidean space where the data are easier to separate. Under certain conditions, called the Mercer condition [10], on the structure of a symmetric function $k(.,.)$ that associates to two objects x and y a real value $k(x, y)$, which represents their similarity, there exists a mapping such that the similarity between the images is exactly equal to $k(x, y)$. The function $k(.,.)$ is called the kernel, which will be formally defined in the next section.

A. *Definition of kernel*

A kernel is a mathematical function used to structure and analyze data. The primary objective of the SVM is to identify a hyperplane that optimally separates different classes of data points. However, data is not linearly separable in its original feature space in many real-world cases. Kernels address this challenge by implicitly transforming the data into a higher-dimensional space, where it may become more easily separable. Formally, we call kernel on any set X, any symmetric function $k : X \times X \mapsto \Re$ verifying:

$$\forall (x, y) \in X \times X \; k(x, y) = k(y, x) \tag{1}$$

*This work was not supported by any organization

[1]Laboratory of Mathematics, Statistics and Computer Science for Scientific Research (W1550900), University Center SALHI Ahmed, Algeria.

[2]Department of Computer Science, University Center SALHI Ahmed, BP-66 45000, Naâma, Algeria. b.sidaoui@cuniv-naama.dz

[3]University of Science and Technology Mohamed Boudiaf USTOMB El Mnaouar, BP 1505, Bir El Djir 31000, Oran, Algeria.

The kernel is said to be positive definite if:

$$\forall x_1, .., x_n \in X, \forall c_1, ..c_n \times X, \sum_{i=1}^{m}\sum_{j=1}^{m} c_i c_j k(x_i x_j) \quad (2)$$

In practice, if we consider $S = x_1, .., x_m$ as a training set, the matrix whose entries represent the similarities between all the objects, taken two by two, is called Gram matrix and denoted G.

$$G = (k(x_i x_j))_{i,j \in [1,..,m]} \quad (3)$$

The kernel $k(.,.)$ is considered positive semi-definite or a kernel that satisfies Mercer's conditions if the corresponding Gram matrix is positive semi-definite. Otherwise, the kernel is referred to as indefinite. Any kernel satisfying Mercer's conditions (1.2) can be viewed as an implicit scalar product in a Hilbert space, mapping the input space X. While Mercer's theorem ensures its existence, the mapping is typically implicit, which prevents the direct computation of point images. A Mercer kernel k possesses the following properties:

- $\forall k$ Positive kernel : $k : X \times X \mapsto \Re$
- $\exists (F, <, >)$ Hilbert space the function : $\phi : X \mapsto F$ such as: $\forall (x, y) \in X \times X \; k(x, y) =< phi(x), \phi(y) >$

Where F is the feature space associated with X, the figure (fig. 1) shows the projection process from the original space to the new linear space using kernels.

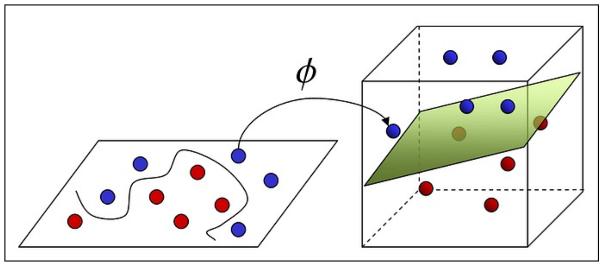

Fig. 1. Projection from the initial space to a new high-dimensional space.

III. Standard kernel functions

This section introduces a set of popular kernel functions used in the SVM method, including linear and nonlinear problems.

A. Linear kernel

The simplest kernel that does not transform the input data. It is used when the data are linearly separable. The constant c is usually set to 0 or 1. The following formula gives this kernel:

$$k(x, y) = x^T y + c \quad (4)$$

1) Polynomial kernel: A kernel representing the data in a higher-dimensional space based on polynomial features. The parameter c is a constant, and d is the degree of the polynomial; the following formula represents the polynomial kernel:

$$k(x, y) = (x * y + c)^d \quad (5)$$

B. Gaussian or RBF kernel

The most widely used kernel, especially for non-linear problems. It measures similarity based on the Euclidean distance between data points. The parameter σ (sigma) controls the kernel's spread, thereby influencing the flexibility of the decision boundary. RBF kernel is given by:

$$k(x, y) = \exp(-\frac{\|x - y\|^2}{2\sigma^2}) \quad (6)$$

C. Laplacian kernel

Similar to the RBF kernel but based on the Manhattan distance ($L1$ norm) instead of the Euclidean distance ($L2$ norm). It is given by:

$$k(x, y) = \exp(-\frac{\|x - y\|_1}{\sigma}) \quad (7)$$

D. Sigmoid kernel

Sigmoid kernel, inspired by neural networks, uses the hyperbolic tangent function. It can model complex nonlinear relationships and is sometimes called the "neural network kernel. This kernel takes the following formula:

$$k(x, y) = \tanh(\gamma * x^T y + c) \quad (8)$$

The above kernels require data with identical dimensions, especially those using classical distance measures such as Euclidean and Manhattan. This constraint is challenging for users since real-world problems often involve variable-dimensional data. Many works propose adequate kernels based on the RBF function as [26], [27].

IV. Specific kernel functions

Defining suitable kernels for specific problems can be an effective solution. The following sections present popular kernels that handle challenges across different domains.

A. String kernel

String kernels are widely used in bioinformatics to assess similarity between biological sequences. It is designed for string-based data, such as text or DNA sequences, employing methods like substring matching, sequence alignment, or edit distance to measure similarity [11].

Let F be a finite set, called an alphabet, whose elements are characters or terminal symbols. The set F^l consists of all character strings of length l, and ϵ is the empty string. Let us introduce the following notations:

- For text data, the alphabet is defined by:$F = \{a, .., z, 0, .., 9\}$
- For biological DNA sequences, the alphabet is defined by:$F = \{A, C, G, T\}$
- For binary sequences, the alphabet is:$F = \{0, 1\}$
- F^* is the set of all non-empty strings defined from F.
- $|x|$ is the length of the string x.
- $uv \in F^*$ is the concatenation of two strings u and v, αu is the concatenation of a character α with u.
- $t^-1 = t_n t_{n-1}..t_1$ is the inverse of the string with $t = t_1 t_2..t_n \in F$.

- $x[i:j]$ is the sub-string of the x string containing the elements whose indices between i and j (both indices i, j are inclusive) where $1 < i, j < |x|$.
- $x = vuw$ so v is called the prefix of the string x, u is called the substring of x ($v \subset x$) and w is called the suffix of the string x.

For two strings x and y, $nbr_y(x)$ represents the count of y as a substring in x. String kernels rely on substring occurrence counts and are defined as follows:

$$k(x, y) = \sum_{s \subset x, s \subset y} nbr_s(x) nbr_s(y) w_s \tag{9}$$

Here, s represents a substring. In other words, we count the occurrences of each substring s in x and y, assigning a weight w, which may be predefined or data-dependent [12], [13]. This can encompass several cases:

- Bag-of-characters: The weight $w = 0$ for all $|s| > 1$, meaning only single-character occurrences are counted, the result is a simple character kernel.
- A duration parameter is introduced to weight the substring s of a string x based on its length: $w_s = \lambda^{|s|}$. If $\lambda < 1$, shorter substrings are favored; otherwise, longer substrings are preferred.
- The weight $w_s = 0$ for any $|s| > T$, where T is a predefined threshold on substring length. Only substrings of length $\leq T$ are considered, which is useful in information retrieval applications.
- k-Spectral Kernel: The weight w is nonzero only for substrings of length k, where k is the substring size. This kernel is widely used in protein classification.

String kernels, which can be efficiently computed via suffix trees, are extensively used in natural language processing and information retrieval to measure text similarity [3].

B. Tree kernel

Specific genetic data can be represented as trees, requiring specialized kernels. Trees, as acyclic-directed graphs, are compared structurally to measure similarity. These kernels are widely used in bioinformatics, natural language processing, and social network analysis [14] and [15]. The following are key properties of trees:

- A leaf is any node that doesn't have children.
- A subtree T_n is any graph of a node n and all its descendants.
- A subset tree is any subgraph of the tree that is a tree.
- A labeled tree is if each node n contains a label label(n).
- A leaf-labeled tree is if only the leaves contain labels.

We denote $t \subset T$ to indicate that t is a subtree of T. Given two trees T and T', the tree kernel $k(T, T')$ is defined based on their subgraphs as follows:

$$k(T, T') = \sum_{t \subset T, t \subset T'} nbr_t(T) nbr_t(T) w_t \tag{10}$$

Where $nbr_t(T)$ represents the number of times subtree t appears in T. The tree kernel counts the occurrences of each subtree t in both trees T and T'.

C. Graph kernel

The graph kernel provides an alternative approach for representing biological data as labeled graphs, rather than simple strings, and has proven helpful in drug research [16], [17], [18].

1) Labeled graph: Let V be a finite non-empty set of nodes, E a set of directed edges, L_V and L_E the sets of node and edge labels, respectively. Given mappings $v : V \to L_V$ and $e : E \to L_E$, a directed labeled graph is $G = (V, E, v, e)$. The traversal of a graph starts at x_1 with an initial probability $p_s(x_1)$. At the step i, it either proceeds to x_i with a transition probability $p_i(x_i/x_{i-1})$ or stops with a final probability $p_q(x_l)$. If $x = x_1, x_2, .., x_l$ then:

$$p(x/G) = p_s(x_1) \prod_{i=2}^{l} p_i(x_i/x_{i-1}) p_q(x_l) \tag{11}$$

2) Distance between two sequences: The sequence corresponding to the path $x = x_1, x_2, .., x_l$ is $h_x = v_{x1}, e_{x1x2}, v_{x2}, .., v_{xl}$. If h is a sequence of labels, then we have:

$$p(h/G) = \sum_{x} \delta(h) h_x) p_s(x_1) \prod_{i=2}^{l} p_i(x_i/x_{i-1}) p_q(x_l) \tag{12}$$

If k_v and k_e define two kernels on V and E. The distance between two sequences h and h' can be calculated using the following kernel:

$$k(h, h') = k_v(h_1, h'_1) \prod_{i=2}^{l} k_e(h_{2i-2}, h'_{2i-2}) k_v(h_{2i-2}, h'_{2i-2}) \tag{13}$$

If h and h' and are of different lengths, then $k(h, h') = 0$. Thus, the kernel between two graphs is computed as the sum of the kernels over all possible label sequences extracted from these graphs as follows:

$$k(G, G') = \sum_{h} \sum_{h'} k(h, h) p(h/G) p(h'/G') \tag{14}$$

D. Local alignment kernel

Aligning biological sequences is a popular method for measuring similarity. Authors in [19] adapt this concept for supervised sequence classification using SVMs, introducing the local alignment kernel based on the Smith-Waterman score. Formally, the alignment between sequences is defined as follows:

$$\pi = (\pi_1(1), .., \pi_1(p), \pi_2(1), .., \pi_2(p)) \in N^{2p} \tag{15}$$

That satisfies the constraints:

$$1 \leq \pi_1(1) \leq \pi_1(2) \leq, .., \leq \pi_1(p) \leq |x| \tag{16}$$

$$1 \leq \pi_2(1) \leq \pi_2(2) \leq, .., \leq \pi_2(p) \leq |y| \tag{17}$$

Where p $(p > 0)$ is the number of positions between the two sequences x and y. A common way to represent sequence alignment is by writing them in parallel, aligning matching characters, and using $'\#'$ symbols to denote gaps.

Consider $x = GAATCCG$ and $y = GATTGC$, the alignment of the 4 letters is $\pi = ((1, 2, 4, 6), (1, 3, 4, 5))$, where: $G\#AATCCG\#$ and $GAT\#T\#G\#C$.

Let $\Pi(x, y)$ be the set of possible alignments between two sequences x and y. $\pi \in \Pi(x, y)$ represents the number of aligned letters in a given alignment. Various scoring functions have been developed to determine the optimal alignment, such as the local alignment and the Smith-Waterman scores.

1) Smith-Waterman score: Let s be a substitution matrix and $g : N \mapsto \Re$ is a gap penalty function such that $g(0) = 0$. We define the local alignment score of $\pi \in \Pi(x, y)$ by:

$$S'(\pi) = \sum_{i=1}^{|\pi|} S(x_{\pi(i)}, y_{\pi(i)}) \tag{18}$$

$$S''(\pi) = \sum_{j=1}^{|\pi|-1} g(\pi_1(j+1)\pi_1(1)-1) + g(\pi_2(j+1)\pi_2(1)-1) \tag{19}$$

The local alignment score π is the sum of the substitution scores between aligned letters $S'(\pi)$, minus the total gap penalties when gaps are present $S''(\pi)$. The optimal alignment score obtained among all possible alignments between two sequences is the Smith-Waterman (SW) score:

$$SW(x, y) = max_{\pi \in \Pi(x,y)} S(\pi) \tag{20}$$

To use this alignment, kernels defined from alignment scores are based on the convolution operation between two kernels k_1 and k_2 is defined as follows, $\forall x, y \in X^2$:

$$k_1 * k_2 = \sum_{x=x1x2, y=y1y2} k_1(x1, y1)k_2(x2, y2) \tag{21}$$

Let $\beta \geq 0$, S be the letter similarity matrix used in the alignment score, and g be the gap penalty function. Elementary kernels k_0, k_a and k_g are defined by :

$$k_0(x, y) = 1 \tag{22}$$

$$k_a(x, y) = \begin{cases} 0 & if|x| = 0or|y| = 0 \\ \exp(\beta S(x, y)) & otherwise \end{cases} \tag{23}$$

$$k_g(x, y) = \exp(\beta(g(|x|) + g(|y|))) \tag{24}$$

The kernel $k_n(x, y) = k_0 * (k_a * k_g)^{n-1} * k_a * k_0$ measures the similarity between sequences x and y based on local alignment of n letters. It sums over all decompositions of x and y into initial parts (similarity measured by k_0), followed by n aligned letters (k_a), possibly separated by $n - 1$ gaps (k_g), and ending with terminal parts (k_0). The local alignment kernel of the Smith-Waterman score is:

$$k_{LocalAlign}(x, y) = \sum_{\pi \in \Pi(x,y)} \exp(\beta S_{s,g}(x, y, \pi)) \tag{25}$$

The local alignment kernel measures similarity via sequence alignments, accounting for biological phenomena such as mutations and insertions. [19], [20].

E. Fisher kernel

Initially designed for biological sequence classification, the Fisher kernel transforms a generative model into a discriminative one to enhance classification performance, enabling its use in SVMs and other discriminative methods. Let $p(X|H_{1,\theta})$ be the probabilistic model corresponding to the hidden Markov chain $H1$ estimated for a particular family. We call the Fisher score the vector:

$$U_X = \nabla_\theta \log p(X|H_{1,\theta}) \tag{26}$$

∇_θ represents the gradient concerning θ, and U_X is a vector whose dimension corresponds to the cardinality of θ. In this context, U_X represents the sequence X based on a generative model with parameters θ. Each component of U_X reflects the impact of a parameter θ on the generating sequence X, where the distance between X and X' is measured via the Fisher kernel.

$$k(X, X') = \frac{1}{2}(U_X - U_{X'})^T F^{-1}(U_X - U_{X'}) \tag{27}$$

Where, F is the autocorrelation function given by:

$$F = \int_x U_x U_x^T p(X|H_{1,\theta}) dx \tag{28}$$

Fisher kernel implementations for biological sequences utilize dynamic programming and hidden Markov models [21].

F. Image kernel

Images convey rich information through their two-dimensional structures. Thus, designing an effective image kernel requires accounting for their properties [22]. A generic convolution kernel is defined as follows:

$$k(X, Y) = \sum_{s \in R(X), s' \in R(Y)} k(s, s') \tag{29}$$

Given two images, X and Y, with $R(.)$ representing their connected regions, direct kernel computation is complex due to the vast number of regions. However, treating images as matrices allows for the representation of connected regions as submatrices. Using $s \subset X$ to denote a submatrix of X, the kernel is redefined accordingly.

$$k(X, Y) = \sum_{s \subset X, s' \subset Y} w_s nbr_s(X) nbr_s(Y) \tag{30}$$

Here, $nbr_s(X)$ represents the occurrence count of s in image X. The image kernel is computed by counting occurrences of each submatrix t in images X and Y, using a method similar to suffix trees for string kernels.

G. Kernel for acoustic data

The speech signal conveys linguistic content, articulation, and paralinguistic traits, such as fatigue, health, age, gender, and speaker-specific characteristics. Comparing words or speech signals involves measuring distance or similarity between speech vectors. However, this is challenging, even for a single speaker, as word duration and rhythm vary. Even trained speakers cannot consistently replicate the same sequence with identical timing and rhythm.

To overcome this challenge, various methods utilize kernel functions, such as the Gaussian and polynomial functions, combined with the dynamic time-warping algorithm to quantify the similarity between speech signals.

1) Dynamic time warping: The DTW algorithm identifies the optimal alignment path between two sequences by minimizing accumulated local distances. These distances are stored in matrix A, where each element (m, n) represents the distance d between the m^{th} element of the first sequence and the n^{th} element of the second. The algorithm iteratively updates A to find the path that minimizes the total distance between $(1, 1)$ and (M, N), where M and N are the sequence lengths [23]. DTW is well described in the following algorithm1.

Algorithm 1 Dynamic time warping algorithm

```
Require: Matrix $A$ of $NxM$, $w_h$, $w_d$, $w_v$ are weights.
Ensure: Distance $D = A(M, N)/(M + N)$.
  $A(1, 1) = d(1, 1) * w_d$;
  for n =2 to N do
    $A(1, n) = A(1, n - 1) + (w_h * d(1, n))$;
  end for
  for m =2 to M do
    $A(m, 1) = A(m - 1, 1) + (w_d * d(m, 1))$;
    for n =2 to N do
      A(m,n)=min $\begin{cases} A(m-1, n-1) + w_d * d(m, n) \\ A(m-1, n) + w_v * d(m, n) \\ A(m, n-1) + w_h * d(m, n) \end{cases}$
    end for
  end for
```

2) Gaussian DTW kernel: Bahlmann et al. [24] introduced a kernel for measuring similarity between acoustic vectors, replacing the Euclidean distance in the RBF kernel with the global distance of the DTW algorithm, as follows:

$$k(x, y) = \exp(-\frac{DTW(x, y)}{2\sigma^2}) \tag{31}$$

3) Polynomial DTW kernel: The polynomial DTW kernel introduced in [25] transforms the global DTW (Dynamic Time Warping) distance into a scalar product by applying a technique known as spherical normalization. Let H be the space where the vectors x and y reside, and let S be a hypersphere embedded in a higher-dimensional space than H, as illustrated in Figure (fig. 2).

The vectors x and y are mapped on the hypersphere using the following equation:

$$\hat{x} = \frac{1}{\sqrt{x^2 + \alpha^2}} \begin{bmatrix} x \\ \alpha \end{bmatrix} \tag{32}$$

where α represents the perpendicular distance from the center of the hypersphere S to the space H, and the value of α is determined by the DTW algorithm. Thus, the distance between sequences is based on the equation:

$$d(\hat{X}, \hat{X}) = \arccos(\hat{X}.\hat{X}) \tag{33}$$

The kernel is defined as the cosine similarity of that distance.

$$k_{poly}(\hat{X}, \hat{Y}) = \cos^n d_s(\hat{X}, \hat{Y}) \tag{34}$$

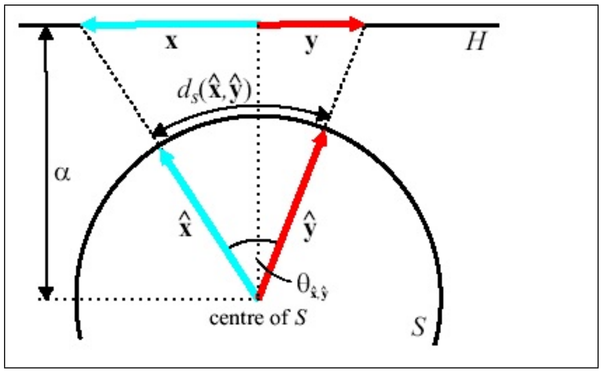

Fig. 2. Projection of the inputs x and y into S

H. Kernel for dynamic systems

Several kernels have been designed to model time-dependent dynamical systems. This section briefly introduces some of them [22]. Let D be a dynamical system with an initial state $x_0 \in X$ at time $t = 0$ (e.g. position, velocity, etc), where $x(t, D, x_0, \epsilon(t)) \in X$ and ϵ is a random variable. The kernel is defined as follows $k(D, D') = k((D, x_0), (D', x'_0))$, with:

$$k(D, D') = \langle (D, x_0), (D', x'_0) \rangle \tag{35}$$

$$k(D, D') = \int \langle x(t, D, x_0, \epsilon(t), x(t, D, x'_0, \epsilon(t)\rangle du(t) \tag{36}$$

The kernel is defined as the correlation time between (D, x_0) and (D', x'_0) to compare trajectories from different dynamical systems with varying initial conditions, as:

$$k_{initial}(x, x') = \int k((D, x), (D', x'))du(D) \tag{37}$$

$$k_{dynamic}(D, D') = \int ((D, x), (D', x'))du(x) \tag{38}$$

In linear time-invariant systems, we have the case of discrete time (formula 39) and the case of continuous time (formula 40), for $x(t) \in \Re^n$ and $A \in \Re^{nxn}$, as follows:

$$x(t + 1) = Ax(t) \tag{39}$$

$$\frac{d}{dt}x(t) = Ax(t) \tag{40}$$

We distinguish two cases: linear systems with discrete time-invariant $x(t) = A^t x(0)$ and continuous time-invariant linear systems $x(t) = x(0)\exp(At)$.

1) Kernels with Different Initial Conditions: Kernels for dynamical systems are categorized by their evolution. For discrete-time systems, where state transitions occur at fixed intervals, a kernel can be defined using equations (37) and (39).

$$k(x, x') = \sum_{0}^{\infty} c_t (A^t x)^T W (A^t x') \tag{41}$$

c_t is a matrix of weights and $W \in \Re^{nxn}$ is the covariance matrix. This kernel can generally be calculated for some values of c_t. For continuous-time systems, where state evolution occurs continuously with minor changes, kernels are derived using equations (38) and (40) as follows:

$$k(x, x') = x^T \left[\int_0^\infty c_t \exp((At)^T) W \exp(At) dt\right] x' \quad (42)$$

c_t is a continuous function of the weights and $W \in \Re^{nxn}$ is the covariance matrix. This kernel can be calculatedly for particular values of $c_t = e^{\lambda t}$.

2) Kernels with identical initial conditions: We extend the kernel concept to dynamical systems by comparing the trajectories $\chi(D, x(0))$ and $\chi(D', x(0))$ of two systems, D and D', with identical initial conditions. If their evolution remains similar for a limited set of inputs, a corresponding kernel is defined as follows:

$$k(D, D') = E_\epsilon E_{x(0)} k_\chi(\chi(D, x(0)), \chi(D', x(0))) \quad (42)$$

k_χ is a kernel and E is the mathematical expectation. This formulation encodes knowledge, and like initial conditions, we can distinguish between discrete-time systems that evolve in steps.

For discrete-time systems, let A and B be two discrete-time invariant linear systems. Using equations (38) and (39), we define the kernel; $k(\chi) = k(\chi(A, x(0)), \chi(B, x(0)))$ as:

$$k(\chi) = x(0)^T \left[\sum_0^\infty c_t (At)^T W B^t\right] x(0) \quad (43)$$

For continuous-time systems, let A and B be two continuous-time invariant linear systems without noise. We define the kernel using equations (38) and (40).

$$k(\chi) = x(0)^T \left[\int_0^\infty c_t \exp(At)^T W \exp(Bt) dt\right] x(0) \quad (44)$$

Where c_t is a continuous function of the weights. These kernels apply to dynamical systems and are computed using specific weights, such as matrices for discrete systems or functions for continuous systems.

V. Conclusion

This paper presents a general survey of kernel functions. While these kernels are widely applicable to various data types, their limitations lead researchers to develop specialized kernels for discrete data, such as biological sequences, linguistic data, images, and dynamical systems. In general, each kernel is tailored to a specific data structure, underscoring the need for specialized kernels that effectively capture the unique characteristics of each data type. Selecting the appropriate kernel for a Support Vector Machine (SVM) is crucial for achieving optimal performance. A linear kernel is ideal when the data is linearly separable, offering computational efficiency and suitability for high-dimensional data. For more complex patterns that require a nonlinear decision boundary, kernels such as the RBF or polynomial kernels are more suitable. Each kernel includes specific hyperparameters that can be fine-tuned to enhance model accuracy. Typically, cross-validation is used to test different parameter combinations and determine the best values.

In real-world problems, standard kernels have limitations due to the complexity and heterogeneity of data. Therefore, a specialized or appropriately designed kernel is more suitable for tasks in biology, text retrieval, biosignals, and related domains.

References

[1] A. Patle and D. S. Chouhan, "SVM kernel functions for classification," 2013 International Conference on Advances in Technology and Engineering (ICATE), Mumbai, India, pp. 1-9, 2013.

[2] T. Tran and Q. Nguyen and M. Lam, "Impact of Kernel Functions on Support Vector Machine Models in Classification and Regression Problems," Proceedings of the International Conference on Sustainable Energy Technologies, Springer, pp. 813–820, 2024.

[3] M. Collins and N. Duffy, "Convolution kernels for natural language," In advances in Neural Information Processing Systems 14, Cambridge, MA, MIT Press, 2001.

[4] X. Ding and J. Liu and F. Yang and J. Cao, "Random radial basis function kernel-based support vector machine," Journal of the Franklin Institute, vol. 358, no. 18, pp. 10121-10140, 2021.

[5] B. Sidaoui, K. Sadouni, "Efficient Normalization of Features Extracted from Vocal Signal for Support Vector Machine," Inter. Journal of Computer Science and Telecom., vol. 10, no 6, pp. 1-5, 2019.

[6] B. Jeong B et al, "Multiple-Kernel Support Vector Machine for Predicting Internet Gaming Disorder Using Multimodal Fusion of PET, EEG, and Clinical Features," Front. Neurosci, vol. 16, 2022.

[7] S. Jiang S, R. Hartley and B. Fernando, "Kernel Support Vector Machines and Convolutional Neural Networks," The Digital Image Computing: Techniques and Applications, Australia, pp. 1-7, 2018.

[8] L. Rongrong et al, "Kernel Support Vector Machine Classifiers with the ℓ 0-Norm Hinge Loss," arXiv-Computer Science, 2023.

[9] G. Yuzhou et all, "Faster Algorithms for Structured Linear and Kernel Support Vector Machines," arXiv-Mathematics, 2025.

[10] J. Mercer, "Functions of positive and negative type and their connection with the theory of integral equations," Proc. R. Soc. Lond, vol. 83, no. 559, pp. 69–70, 1909.

[11] C. Watkins, "Dynamic Alignment Kernels," In Advances in Large Margin Classifiers, 2002.

[12] E. Leopold and J. Kindermann, "Text categorization with support vector machines: How to represent text in input space?,' Machine Learning, vol. 46, no. 3, pp. 423–444, 2002.

[13] C. Leslie et al, "The spectrum kernel: a string kernel for SVM protein classification," Biocomputing, pp. 564-575, 2002.

[14] S. Jun and Z. Min and L. Chew, "Tree Sequence Kernel for Natural Language," Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, 2011.

[15] B. Schölkopf, K. Tsuda and J. Vert, Kernel methods in computational biology. MIT Press, 2004.

[16] H. Lodhi, "Computational biology perspective: kernel methods and deep learning," WIREs Computational Statistics, vol. 4, no. 4, pp. 455-465, 2012.

[17] A. Azencott, Implementation and validation of kernels for biological sequences, Rennes I, 2005.

[18] G. Nikolentzos and V. Michalis, "Graph Alignment Kernels using Weisfeiler and Leman Hierarchies," Proc. of The 26th Inter. Conference on Artificial Intelligence and Statistics, vol. 206, pp. 2019-2034, 2023.

[19] J. V. Philippe et al, "A local alignment kernel for biological sequence classification," Book Chapter in Kernel Methods in Computational Biology, 2004.

[20] P. Pavlidis et al, "Gene functional classification from heterogeneous data," Department of Computer Science, Columbia University, 450, New York, 2001.

[21] T. Jaakkola et al, "Using the Fisher kernel method to detect remote protein homologies," In Proceedings of the International Conference on Intelligence Systems for Molecular Biology, pp. 149-158, 1999.

[22] S. V. Vishwanathan, "Kernel Methods Fast Algorithms and Real Life Applications," Ph.D, Department of Computer Science and Automation, Indian Institute of Science, Bangalore, 2003.

[23] R. Julien, N. Schnell and D. Schwarz, "Phoneme recognition and classification," IRCAM, Paris, 2005.

[24] C. Bahlmann and B. Haasdonk, "On-line handwriting recognition with support vector machines a kernel approach," In Proc. 8th International Workshop on Frontiers in Handwriting Recognition, 2002.

[25] V.Wan and J. Carmichael, "Polynomial Dynamic Time Warping Kernel Support Vector Machines for Dysarthric Speech Recognition with Sparse Training Data," AMI, FP6-506811, InterSpeech, 2005.

[26] J. A. Danial et al, "Effect of SVM Kernel Functions on Bearing Capacity Assessment of Deep Foundations," Journal of Soft Computing in Civil Engineering, col. 7, no. 3, pp. 111-128, 2023.

[27] A. Elen et al, "An Adaptive Gaussian Kernel for Support Vector Machine," Arab J Sci Eng, vol. 47, no. 8, pp. 2191-4281, 2022.